Bez względu na branżę, w której pracujesz: IT, moda, żywność czy finanse, nie ma wątpliwości, że dane wpływają na Twoje życie i pracę. W pewnym momencie w tym tygodniu odbędziesz lub usłyszysz rozmowę na temat danych. Serwisy informacyjne relacjonują coraz więcej historii o wyciekach danych, cyberprzestępczości oraz o tym, jak dane mogą dać nam wgląd w nasze życie. Ale dlaczego teraz? Co sprawia, że ta era jest tak siedliskiem branż związanych z danymi? W XIX wieku świat był w szponach epoki przemysłowej. Ludzkość badała swoje miejsce w przemyśle obok gigantycznych wynalazków mechanicznych. Kapitanowie przemysłu, tacy jak Henry Ford, dostrzegli duże możliwości rynkowe tkwiące w tych maszynach i byli w stanie osiągnąć niewyobrażalne wcześniej zyski. Oczywiście epoka przemysłowa miała swoje plusy i minusy. Podczas gdy masowa produkcja umieszczała towary w rękach większej liczby konsumentów, nasza walka z zanieczyszczeniem również zaczęła się w tym czasie. W XX wieku byliśmy już dość biegli w tworzeniu ogromnych maszyn; teraz celem było uczynienie ich mniejszymi i szybszymi. Epoka przemysłowa się skończyła i została zastąpiona przez to, co nazywamy epoką informacji. Zaczęliśmy używać maszyn do gromadzenia i przechowywania informacji (danych) o nas samych i naszym środowisku w celu zrozumienia naszego wszechświata. Począwszy od lat 40. maszyny takie jak ENIAC (uważane za jeden z, jeśli nie pierwszy, komputer) obliczały równania matematyczne oraz uruchamiały modele i symulacje jak nigdy dotąd. W końcu mieliśmy przyzwoitego asystenta laboratoryjnego, który potrafił liczyć lepiej niż my! Podobnie jak w epoce przemysłowej, epoka informacji przyniosła nam zarówno dobre, jak i złe. Dobra była niezwykła technologia, w tym telefony komórkowe i telewizory. Zło w tym przypadku nie było tak złe, jak zanieczyszczenie na całym świecie, ale nadal pozostawiło nam problem w XXI wieku, tak wiele danych. Zgadza się, era informacji, w swoim dążeniu do pozyskiwania danych, eksplodowała produkcją danych elektronicznych. Szacunki pokazują, że w 2011 roku utworzyliśmy około 1,8 biliona gigabajtów danych (poświęć chwilę, aby pomyśleć, ile to jest). Zaledwie rok później, w 2012 roku, stworzyliśmy ponad 2,8 biliona gigabajtów danych! Ta liczba będzie rosła jeszcze dalej, aby osiągnąć około 40 bilionów gigabajtów danych w ciągu zaledwie jednego roku do 2020 roku. Ludzie przyczyniają się do tego za każdym razem, gdy tweetują, publikują na Facebooku, zapisują nowe CV w Microsoft Word lub po prostu wysyłają swojej mamie zdjęcie za pośrednictwem wiadomości tekstowej. Nie tylko tworzymy dane w niespotykanym dotąd tempie, ale także zużywamy je w przyspieszonym tempie. Zaledwie w 2013 roku, przeciętny użytkownik telefonu komórkowego zużył mniej niż 1 GB danych miesięcznie. Obecnie szacuje się, że liczba ta wynosi grubo ponad 2 GB miesięcznie. Nie szukamy tylko następnego quizu osobowości, szukamy wglądu. Wszystkie te dane tam są, niektóre z nich muszą być dla mnie przydatne! I może być! Więc my, w XXI wieku, mamy problem. Mamy tak dużo danych i wciąż robimy więcej. Zbudowaliśmy szalenie malutkie maszyny, które zbierają dane 24 godziny na dobę, 7 dni w tygodniu, a naszym zadaniem jest nadanie im sensu. Wprowadź wiek danych. To jest wiek, kiedy bierzemy maszyny wymyślone przez naszych XIX-wiecznych przodków i dane stworzone przez naszych odpowiedników z XX wieku i tworzymy spostrzeżenia i źródła wiedzy, z których każdy człowiek na Ziemi może skorzystać. Stany Zjednoczone stworzyły dla głównego naukowca danych zupełnie nową rolę w rządzie. Firmy technologiczne, takie jak Reddit, które do tej pory nie miały w swoim zespole analityka danych, teraz zatrudniają ich na prawo i lewo. Korzyść jest dość oczywista – wykorzystanie danych do tworzenia dokładnych prognoz i symulacji daje nam wgląd w nasz świat jak nigdy dotąd. Brzmi świetnie, ale w czym tkwi haczyk? W tej części poznamy terminologię i słownictwo współczesnego naukowca danych. Zobaczymy kluczowe słowa i wyrażenia, które są kluczowe w naszej dyskusji na temat nauki o danych. Zanim zaczniemy przyglądać się kodowi w Pythonie, podstawowym języku używanym przez nas, przyjrzymy się również, dlaczego korzystamy z nauki o danych i trzech kluczowych domenach, z których wywodzi się nauka o danych:

- Podstawowa terminologia nauki o danych
- Trzy dziedziny nauki o danych

• Podstawowa składnia Pythona

Co to jest nauka o danych?

Zanim przejdziemy dalej, spójrzmy na kilka podstawowych definicji, których będziemy używać. Wielką/okropną rzeczą w tej dziedzinie jest to, że jest ona tak młoda, że definicje te mogą się różnić w zależności od podręcznika, gazety i białej księgi.

Podstawowa terminologia

Poniższe definicje są na tyle ogólne, że można je stosować w codziennych rozmowach i pracy, przy wprowadzeniu do zasad data science. Zacznijmy od zdefiniowania, czym są dane. Może się to wydawać głupią pierwszą definicją, ale jest to bardzo ważne. Ilekroć używamy słowa „dane", odnosimy się do zbierania informacji w zorganizowanym lub niezorganizowanym formacie:

• Dane uporządkowane: Odnosi się to do danych posortowanych w strukturę wiersz/kolumna, gdzie każdy wiersz reprezentuje pojedynczą obserwację, a kolumny reprezentują cechy tej obserwacji.

• Niezorganizowane dane: Jest to rodzaj danych w dowolnej formie, zwykle tekstu lub nieprzetworzonego dźwięku/sygnałów, które muszą być dalej analizowane w celu uporządkowania. Za każdym razem, gdy otwierasz program Excel (lub dowolny inny program do obsługi arkuszy kalkulacyjnych), patrzysz na pustą strukturę wiersza/kolumny czekającą na uporządkowane dane. Te programy nie radzą sobie dobrze z niezorganizowanymi danymi. W większości będziemy mieli do czynienia z uporządkowanymi danymi, ponieważ najłatwiej jest z nich uzyskać wgląd, ale nie będziemy stronić od surowego tekstu i metod przetwarzania niezorganizowane formy danych.

Nauka o danych to sztuka i nauka zdobywania wiedzy poprzez dane.

Cóż za mała definicja tak dużego tematu i słusznie! Nauka o danych obejmuje tak wiele rzeczy, że zajęłoby strony, aby to wszystko wymienić.

Nauka o danych polega na tym, jak zbieramy dane, wykorzystujemy je do zdobywania wiedzy, a następnie wykorzystujemy tę wiedzę do wykonywania następujących czynności:

• Podejmować decyzje

• Przewidzieć przyszłość

• Zrozumieć przeszłość/teraźniejszość

• Twórz nowe branże/produkty

Ten tekst dotyczy metod nauki o danych, w tym sposobu przetwarzania danych, gromadzenia spostrzeżeń i wykorzystywania tych spostrzeżeń do podejmowania świadomych decyzji i prognoz. Nauka o danych polega na wykorzystywaniu danych w celu uzyskania nowych spostrzeżeń, których w przeciwnym razie byś nie zauważył. Jako przykład wyobraź sobie, że siedzisz przy stole z trzema innymi osobami. Wasza czwórka musi podjąć decyzję w oparciu o pewne dane. Należy wziąć pod uwagę cztery opinie. Użyłbyś nauki o danych, aby przedstawić piątą, szóstą, a nawet siódmą opinię. Dlatego nauka o danych nie zastąpi ludzkiego mózgu, ale go uzupełni, będzie z nim współpracować. Nauka o danych nie powinna być traktowana jako ostateczne rozwiązanie naszych problemów z danymi; jest to tylko opinia, bardzo poinformowana opinia, niemniej jednak opinia. Zasługuje na miejsce przy stole.

Dlaczego nauka o danych?

W erze danych jasne jest, że mamy nadwyżkę danych. Ale dlaczego miałoby to wymagać całego nowego zestawu słownictwa? Co było nie tak z naszymi poprzednimi formami analizy? Po pierwsze, sama ilość danych sprawia, że dosłownie niemożliwe jest przeanalizowanie ich przez człowieka w rozsądnym czasie. Dane zbierane są w różnych formach i z różnych źródeł, często bardzo niezorganizowane. Może brakować danych, mogą być niekompletne lub po prostu niepoprawne. Często dysponujemy danymi w bardzo różnych skalach, co utrudnia ich porównywanie. Weź pod uwagę, że patrzymy na dane w odniesieniu do wyceny używanych samochodów. Jedną cechą samochodu jest rok produkcji, a inną może być liczba mil na tym samochodzie. Kiedy oczyścimy nasze dane (któremu poświęciliśmy dużo czasu na przeglądanie w tej książce), relacje między danymi stają się bardziej oczywiste, a wiedza, która kiedyś była zakopana głęboko w milionach wierszy danych, po prostu wyskakuje. Jednym z głównych celów nauki o danych jest stworzenie wyraźnych praktyk i procedur w celu odkrycia i zastosowania tych relacji w danych. Wcześniej przyjrzeliśmy się nauce o danych w bardziej historycznej perspektywie, ale poświęćmy chwilę na omówienie jej roli w dzisiejszym biznesie na bardzo prostym przykładzie.

Przykład - Sigma Technologies

Ben Runkle, dyrektor generalny Sigma Technologies, próbuje rozwiązać ogromny problem. Firma konsekwentnie traci wieloletnich klientów. Nie wie, dlaczego odchodzą, ale musi coś szybko zrobić. Jest przekonany, że aby zmniejszyć ewakuację, musi tworzyć nowe produkty i funkcje oraz konsolidować istniejące technologie. Aby być bezpiecznym, wzywa swojego głównego analityka danych, dr Jessie Hughan. Nie jest jednak przekonany, że same nowe produkty i funkcje uratują firmę. Zamiast tego sięga do transkrypcji ostatnich biletów obsługi klienta. Pokazuje Runkle'owi najnowsze transkrypcje i znajduje coś zaskakującego:

• … Nie wiesz, jak to wyeksportować; prawda?"

• „Gdzie jest przycisk, który tworzy nową listę?"

• „Czekaj, czy wiesz, gdzie jest suwak?"

• „Jeśli nie mogę tego dzisiaj rozgryźć, to jest prawdziwy problem …"

Oczywiste jest, że klienci mieli problemy z istniejącym interfejsem użytkownika/UX i nie byli zmartwieni brakiem funkcji. Runkle i Hughan zorganizowali masową przebudowę UI/UX, a ich sprzedaż nigdy nie była lepsza. Oczywiście nauka zastosowana w ostatnim przykładzie była minimalna, ale ma sens. Zwykle nazywamy takich ludzi jak Runkle, kierowca. Dzisiejszy, typowy dyrektor generalny, który trzyma się przeczucia, chce szybko podejmować wszystkie decyzje i powtarzać rozwiązania, aż coś zadziała. Dr Haghun jest znacznie bardziej analityczny. Chce rozwiązać problem tak samo jak Runkle, ale szuka odpowiedzi na dane generowane przez użytkowników, a nie na instynktowne wyczucie. Nauka o danych polega na stosowaniu umiejętności analitycznego umysłu i używaniu ich tak, jak zrobiłby to kierowca. Obie te mentalności mają swoje miejsce we współczesnych przedsiębiorstwach; jednak to sposób myślenia Hagun dominuje w koncepcjach data science – wykorzystanie danych generowanych przez firmę jako źródła informacji, a nie tylko wybieranie rozwiązania i podążanie z nim.

Diagram Venna

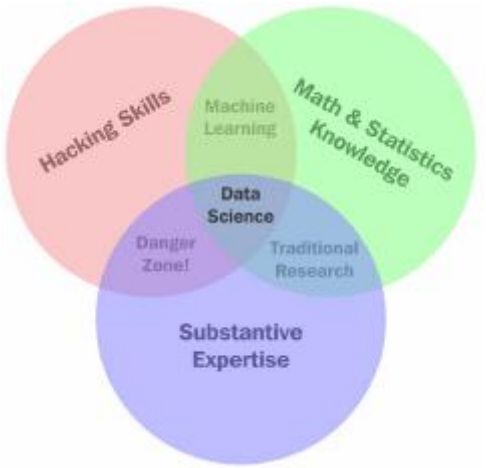
Powszechnym błędem jest przekonanie, że tylko osoby z doktoratem lub geniusze mogą zrozumieć matematykę/programowanie stojące za nauką o danych. To jest całkowicie fałszywe. Zrozumienie nauki o danych zaczyna się od trzech podstawowych obszarów:

• Matematyka/statystyka: jest to użycie równań i wzorów do przeprowadzenia analizy

• Programowanie komputerowe: jest to umiejętność używania kodu do tworzenia wyników na komputerze

• Wiedza dziedzinowa: odnosi się do zrozumienia problematycznej domeny (medycyna, finanse, nauki społeczne itd.)

Poniższy diagram Venna przedstawia wizualną reprezentację tego, jak przecinają się trzy obszary nauki o danych:



Osoby z umiejętnościami hakerskimi mogą konceptualizować i programować skomplikowane algorytmy za pomocą języków komputerowych. Posiadanie bazy wiedzy matematycznej i statystycznej pozwala teoretyzować i oceniać algorytmy oraz dostosowywać istniejące procedury do konkretnych sytuacji. Posiadanie wiedzy merytorycznej (ekspertyzy dziedzinowej) pozwala stosować koncepcje i wyniki w znaczący i skuteczny sposób.

Chociaż posiadanie tylko dwóch z tych trzech cech może uczynić cię inteligentnym, pozostawi również lukę. Weź pod uwagę, że jesteś bardzo biegły w kodowaniu i masz formalne szkolenie w zakresie daytradingu. Możesz stworzyć zautomatyzowany system do handlu na swoim miejscu, ale nie masz umiejętności matematycznych, aby ocenić swoje algorytmy, a zatem stracisz pieniądze w końcu. Tylko wtedy, gdy możesz pochwalić się umiejętnościami w zakresie kodowania, matematyki i znajomości domeny, możesz naprawdę wykonywać analizę danych.

To, co prawdopodobnie była dla Ciebie niespodzianką, to Domain Knowledge. To naprawdę tylko wiedza o obszarze, w którym pracujesz. Jeśli analityk finansowy zaczął analizować dane dotyczące zawałów serca, może potrzebować pomocy kardiologa, aby zrozumieć wiele liczb. Data Science to skrzyżowanie trzech kluczowych obszarów wspomnianych wcześniej. Aby uzyskać wiedzę z danych, musimy być w stanie wykorzystać programowanie komputerowe w celu uzyskania dostępu do danych, zrozumieć matematykę kryjącą się za modelami, które wyprowadzamy, oraz przede wszystkim zrozumieć miejsce naszych analiz w domenie, w której się znajdujemy. Obejmuje to prezentację danych. Jeśli tworzymy model do przewidywania zawałów serca u pacjentów, czy lepiej stworzyć plik PDF z informacjami lub aplikację, w której można wpisywać liczby i uzyskać szybką prognozę? Wszystkie te decyzje muszą podjąć badacz danych.

Pamiętaj też, że przecięciem matematyki i kodowania jest uczenie maszynowe. Szczegółowo przyjrzymy się uczeniu maszynowemu później, ale ważne jest, aby pamiętać, że bez wyraźnej możliwości uogólnienia jakichkolwiek modeli lub wyników na domenę, algorytmy uczenia maszynowego pozostają tylko algorytmami znajdującymi się na twoim komputerze. Być może masz najlepszy algorytm do przewidywania raka. Możesz być w stanie przewidzieć raka z ponad 99%

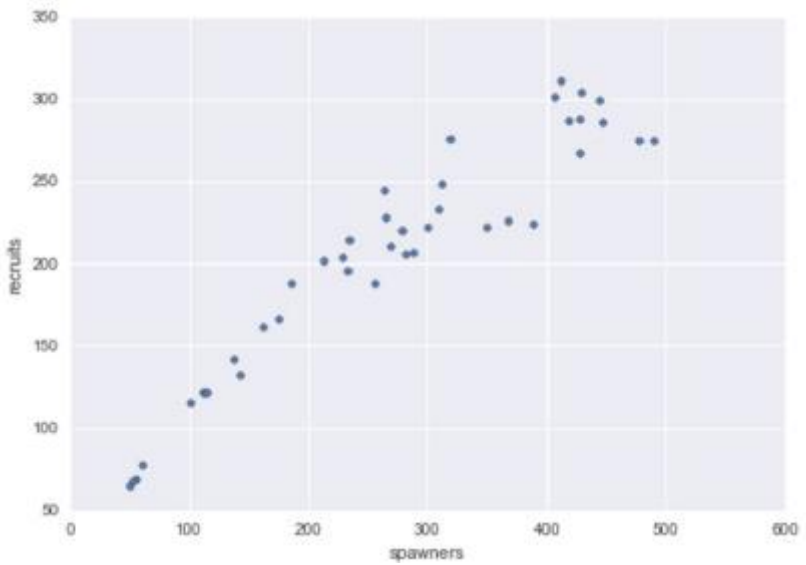
dokładnością na podstawie wcześniejszych danych pacjentów z rakiem, ale jeśli nie rozumiesz, jak zastosować ten model w praktycznym sensie, tak aby lekarze i pielęgniarki mogli z niego łatwo korzystać, Twój model może być bezużyteczny.

Matematyka

Większość ludzi przestaje słuchać, gdy ktoś wypowie słowo „matematyka". Kiwają głową, próbując ukryć całkowitą pogardę dla tematu. Te wpisy poprowadzą Cię przez matematykę potrzebną do nauki o danych, w szczególności statystyki i prawdopodobieństwa. Wykorzystamy te poddziedziny matematyki do stworzenia tak zwanych modeli. Model danych odnosi się do zorganizowanej i formalnej relacji między elementami danych, zwykle przeznaczonej do symulowania zjawiska w świecie rzeczywistym. Zasadniczo użyjemy matematyki w celu sformalizowania relacji między zmiennymi. Wiem, jak trudne może to być. Zrobię co w mojej mocy, aby wyjaśnić wszystko tak jasno, jak tylko potrafię. Pomiędzy trzema obszarami nauki o danych, matematyka jest tym, co pozwala nam przechodzić od domeny do domeny. Zrozumienie tej teorii pozwala nam zastosować model, który zbudowaliśmy dla branży modowej, do modelu finansowego. Matematyka omawiana tu obejmuje zakres od podstawowej algebry do zaawansowanego modelowania probabilistycznego i statystycznego. Nie pomijaj tych części, nawet jeśli już je znasz lub się ich boisz. Każde pojęcie matematyczne, które wprowadzam, robię z rozwagą, przykładami i celem. Matematyka jest niezbędna dla naukowców zajmujących się danymi.

Przykład - modele rekrutacji ikry

W biologii używamy, między innymi, modelu znanego jako model rekrutacji ikry, aby ocenić zdrowie biologiczne gatunku. Jest to podstawowa zależność między liczbą zdrowych jednostek rodzicielskich gatunku a liczbą nowych jednostek w grupie zwierząt. W publicznym zbiorze danych dotyczących liczby tarlaków i ikry łososia utworzono poniższy wykres, aby zobrazować związek między nimi.



Widzimy, że na pewno istnieje jakaś pozytywna relacja (w miarę, jak jeden idzie w górę, drugi też). Ale jak możemy sformalizować ten związek? Na przykład, gdybyśmy znali liczbę ikry w populacji, czy moglibyśmy przewidzieć liczbę rekrutów, które pozyska ta grupa i na odwrót? Zasadniczo modele pozwalają nam podłączyć jedną zmienną, aby uzyskać drugą. Rozważmy następujący przykład:

Rekruci = 0,5 * Ikra + 60

W tym przykładzie załóżmy, że wiedzieliśmy, że grupa łososi miała 1,15 (w tysiącach) tarła. Wtedy mielibyśmy:

Rekruci = 0,5 * 1,15 + 60

Rekruci = 60,575 (w tysiącach)

Ten wynik może być bardzo korzystny przy oszacowaniu, jak zmienia się stan zdrowia populacji. Jeśli uda nam się stworzyć te modele, możemy wizualnie zaobserwować, jak mogą zmieniać się relacje między dwiema zmiennymi. Istnieje wiele typów modeli danych, w tym modele probabilistyczne i statystyczne. Oba są podzbiorami większego paradygmatu, zwanego uczeniem maszynowym. Zasadniczą ideą stojącą za tymi trzema tematami jest to, że wykorzystujemy dane w celu opracowania najlepszego możliwego modelu. Nie polegamy już na ludzkich instynktach, raczej polegamy na danych. Celem tego przykładu jest pokazanie, w jaki sposób możemy zdefiniować relacje między elementami danych za pomocą równań matematycznych. Fakt, że wykorzystałem dane dotyczące zdrowia łososia, był nieistotny! Przyjrzymy się relacjom obejmującym dolary marketingowe, dane dotyczące nastrojów, recenzje restauracji i wiele innych. Głównym tego powodem jest to, że chciałbym, abyście (czytelnicy) mieli dostęp do jak największej liczby domen. Matematyka i kodowanie to narzędzia, które pozwalają analitykom danych cofnąć się i zastosować swoje umiejętności praktycznie w dowolnym miejscu

Programowanie komputerowe

Bądźmy szczerzy. Prawdopodobnie myślisz, że informatyka jest o wiele fajniejsza niż matematyka. W porządku, nie obwiniam cię. Wiadomości nie są wypełnione wiadomościami matematycznymi, tak jak wiadomościami na froncie technologicznym. Nie włączasz telewizora, aby zobaczyć nową teorię na temat liczb pierwszych, raczej zobaczysz raporty śledcze na temat tego, jak najnowszy smartfon może lepiej robić zdjęcia kotom czy coś takiego. Języki komputerowe to sposób, w jaki komunikujemy się z maszyną i każemy jej wykonywać nasze polecenia. Komputer mówi wieloma językami i, podobnie jak książka, może być napisany w wielu językach; podobnie, nauka o danych może być również wykonywana w wielu językach. Python, Julia i R to tylko niektóre z wielu dostępnych dla nas języków. My skupimy się wyłącznie na używaniu Pythona.

Dlaczego Python?

Pythona będziemy używać z różnych powodów:

• Python jest niezwykle prostym językiem do czytania i pisania, nawet jeśli nigdy wcześniej nie programowałeś, co sprawi, że przyszłe przykłady będą łatwe do przyswojenia i przeczytania później.

• Jest to jeden z najpopularniejszych języków, zarówno w środowisku produkcyjnym, jak i akademickim (w rzeczywistości jeden z najszybciej rozwijających się)

• Społeczność internetowa języka jest rozległa i przyjazna. Oznacza to, że szybkie wyszukiwanie w Google powinno dać wiele wyników osób, które napotkały i rozwiązały podobne (jeśli nie dokładnie te same) sytuacje

• Python ma gotowe moduły analizy danych, z których mogą korzystać zarówno nowicjusze, jak i doświadczeni naukowcy zajmujący się danymi

Ten ostatni jest prawdopodobnie największym powodem, dla którego skupimy się na Pythonie. Te gotowe moduły są nie tylko potężne, ale także łatwe do pobrania. Pod koniec pierwszych kilku rozdziałów te moduły będą bardzo wygodne. Niektóre z tych modułów to:

• pandas

• sci-kit learn

• seaborn

• numpy/scipy

• requests (to mine data from the Web)

• BeautifulSoup (do analizowania stron WWW-HTML)

Praktyczny Python

Zanim przejdziemy dalej, ważne jest sformalizowanie wielu wymaganych umiejętności kodowania w Pythonie. W Pythonie mamy zmienne, które są symbolami zastępczymi dla obiektów. Na początku skupimy się tylko na kilku typach podstawowych obiektów:

• int (liczba całkowita)

- Przykłady: 3, 6, 99, -34, 34, 11111111

• liczba zmiennoprzecinkowa (dziesiętna):

- Przykłady: 3,14159, 2,71, -0,34567

• wartość logiczna (prawda lub fałsz)

- Stwierdzenie, że niedziela jest weekendem, jest prawdziwe

- Stwierdzenie, że piątek jest weekendem, jest fałszywe

- Stwierdzenie, pi jest dokładnie stosunkiem obwodu koła do jego średnica, to prawda (szalony, prawda?)

• ciąg (tekst lub słowa złożone ze znaków)

- „Kocham hamburgery” (przy okazji, kto nie?)

- "Mat jest niesamowity"

- Tweet to ciąg

• lista (zbiór obiektów)

- Przykład: [1, 5.4, Prawda, "jabłko"]

Będziemy musieli również zrozumieć kilka podstawowych operatorów logistycznych. W przypadku tych operatorów należy pamiętać o typie danych logicznych. Każdy operator oceni True lub False. Rzućmy okiem na następujące ilustracje:

• == zwraca wartość Prawda, jeśli obie strony są równe; w przeciwnym razie zwraca się do Fałsz

- 3 + 4 == 7 (oznacza wartość Prawda)

- 3 – 2 == 7 (oznacza wartość Fałsz)

• < (mniej niż)

- 3 < 5 (prawda)

- 5 < 3 (fałsz)

• <= (mniejszy lub równy)

- 3 <= 3 (prawda)

- 5 <= 3 (fałsz)

• > (większe niż)

- 3 > 5 (fałsz)

- 5 > 3 (prawda)

• >= (większe lub równe)

- 3 >= 3 (prawda)

- 5 >= 3 (fałsz)

Podczas kodowania w Pythonie użyję znaku krzyżyka (#), aby utworzyć komentarz", który nie będzie przetwarzany jako kod, ale służy jedynie do komunikacji z czytelnikiem. Wszystko na prawo od znaku # jest komentarzem do wykonywany kod.

Przykład podstawowego Pythona

W Pythonie używamy spacji/tabulatorów do oznaczania operacji należących do innych linii kodu. Zwróć uwagę na użycie instrukcji if. Oznacza dokładnie to, co myślisz, że oznacza. Jeśli instrukcja po instrukcji if ma wartość True, część z kartami pod nią zostanie wykonana, jak pokazano w poniższym kodzie:

```
X = 5,8

Y = 9,5

X + Y == 15,3 # To prawda!

X - Y == 15,3 # To jest nieprawda!

5if x + y == 15.3: # Jeśli stwierdzenie jest prawdziwe

print "Prawda!" # wyświetl coś!
```

Instrukcja print „Prawda!" należy do linii if x + y == 15.3: linii poprzedzającej ją, ponieważ znajduje się tuż pod nią. Oznacza to, że instrukcja print zostanie wykonana wtedy i tylko wtedy, gdy x + y równa się 15.3.

Zauważ, że następująca zmienna list, my_list, może przechowywać wiele typów obiektów. Ma dane wejściowe typu int, float, boolean i string (w tej kolejności):

```
my_list = [1, 5.7, Prawda, "jabłka"]

len(my_list) == 4 # 4 obiekty na liście

my_list[0] == 1 # pierwszy obiekt

my_list[1] == 5.7 # drugi obiekt
```

W poprzednim kodzie:

• Użyłem polecenia len, aby uzyskać długość listy (która wynosiła cztery).

• Zwróć uwagę na indeksowanie zerowe w Pythonie. Większość języków komputerowych zaczyna liczyć od zera zamiast od jednego. Więc jeśli chcę pierwszy element, nazywam indeks zero, a jeśli chcę 95 element, nazywam indeks 94.

Przykład - parsowanie pojedynczego tweeta

Oto trochę kodu w Pythonie. W tym przykładzie przeanalizuję kilka tweetów na temat cen akcji (jednym z ważnych studiów przypadku będzie próba przewidzenia ruchów rynkowych w oparciu o popularne nastroje dotyczące akcji w mediach społecznościowych):

```
tweet = "RT @j_o_n_dnger: $TWTR now top holding for Andor, unseating $AAPL"

words_in_tweet = first_tweet.split(' ') # list of words in tweet

for word in words_in_tweet: # for each word in list

 if "$" in word: # if word has a "cashtag"

print "THIS TWEET IS ABOUT", word # alert the user
```

Wskażę kilka rzeczy na temat tego fragmentu kodu, linia po linii, w następujący sposób:

• Ustawiamy zmienną do przechowywania tekstu (znanego jako ciąg znaków w Pythonie). W tym przykładzie tweet, o którym mowa, to „RT @robdv: $TWTR teraz trzyma górę dla Andora, usuwa $AAPL”

• Zmienna words_in_tweet tokenizuje tweet (oddziela go słowem). Gdybyś miał wydrukować tę zmienną, zobaczyłbyś co następuje:

```
['RT',

'@robdv:',

'$TWTR',

'now',

'top',

'holding',

'for',

'Andor,',

'unseating',

'$AAPL']
```

• Powtarzamy tę listę słów. Nazywa się to pętlą for. Oznacza to po prostu, że przeglądamy listę jeden po drugim.

• Tutaj mamy kolejną instrukcję if. Dla każdego słowa w tym tweecie, jeśli słowo zawiera znak $ (w ten sposób ludzie odwołują się do notowań giełdowych na Twitterze).

• Jeśli poprzednie stwierdzenie if jest prawdziwe (tzn. jeśli tweet zawiera tag cashtag), wydrukuj je i pokaż użytkownikowi. Wynik tego kodu będzie następujący:

```
THIS TWEET IS ABOUT $TWTR
```

THIS TWEET IS ABOUT $AAPL

Otrzymujemy to wyjście, ponieważ są to jedyne słowa w tweecie, które używają cashtagu. Za każdym razem, gdy używam Pythona, upewnię się, że w każdym wierszu kodu mówię tak jasno, jak to tylko możliwe, o tym, co robię.

Wiedza domenowa

Jak wspomniałem wcześniej, ta kategoria skupia się głównie na posiadaniu wiedzy na konkretny temat, nad którym pracujesz. Na przykład, jeśli jesteś analitykiem finansowym pracującym na danych giełdowych, masz dużą wiedzę domenową. Jeśli jesteś dziennikarzem i przyglądasz się światowym wskaźnikom adopcji, możesz skorzystać z konsultacji z ekspertem w tej dziedzinie. Spróbujemy pokazać przykłady z kilku problematycznych dziedzin, w tym medycyny, marketingu, finansów, a nawet obserwacji UFO! Czy to oznacza, że jeśli nie jesteś lekarzem, nie możesz pracować z danymi medycznymi? Oczywiście nie! Świetni analitycy danych mogą zastosować swoje umiejętności w dowolnym obszarze, nawet jeśli nie są w tym biegli. Analitycy danych mogą dostosować się do dziedziny i wnieść znaczący wkład po zakończeniu analizy. Dużą częścią wiedzy domenowej jest prezentacja. W zależności od odbiorców, sposób prezentacji wyników może mieć duże znaczenie. Twoje wyniki są tak dobre, jak Twój środek komunikacji. Możesz przewidzieć ruch na rynku z 99,99% dokładnością, ale jeśli Twój program jest niemożliwy do wykonania, Twoje wyniki pozostaną niewykorzystane. Podobnie, jeśli twój pojazd jest nieodpowiedni do pola, twoje wyniki będą równie niewykorzystane.

Trochę więcej terminologii

To dobry moment, aby zdefiniować więcej słownictwa. W tym momencie prawdopodobnie podekscytowany przeglądasz wiele materiałów z zakresu analizy danych i widzisz słowa i wyrażenia, których jeszcze nie używałem. Oto kilka typowych terminologii, z którymi możesz się spotkać:

• Uczenie maszynowe: odnosi się do zapewnienia komputerom możliwości uczenia się na podstawie danych bez wyraźnych „reguł" narzucanych przez programistę. Widzieliśmy wcześniej koncepcję uczenia maszynowego jako połączenie kogoś, kto ma zarówno umiejętności kodowania, jak i matematyki. Tutaj próbujemy sformalizować tę definicję. Uczenie maszynowe łączy moc komputerów z inteligentnymi algorytmami uczenia w celu zautomatyzowania odkrywania relacji w danych i tworzenia potężnych modeli danych. Mówiąc o modelach danych, zajmiemy się następującymi dwoma podstawowymi typami modeli danych:

• Model probabilistyczny: Odnosi się do wykorzystania prawdopodobieństwa do znalezienia związku między elementami, który zawiera pewien stopień losowości.

• Model statystyczny: Odnosi się do wykorzystania twierdzeń statystycznych do sformalizowania relacji między elementami danych w (zwykle) prostym wzorze matematycznym.

Chociaż zarówno modele statystyczne, jak i probabilistyczne mogą być uruchamiane na komputerach i mogą być pod tym względem uważane za uczenie maszynowe, oddzielimy te definicje, ponieważ algorytmy uczenia maszynowego generalnie próbują uczyć się relacji na różne sposoby.

• Eksploracyjna analiza danych (EDA) odnosi się do przygotowania danych w celu standaryzowania wyników i szybko uzyskiwać wgląd. EDA zajmuje się wizualizacją i przygotowaniem danych. W tym miejscu zamieniamy niezorganizowane dane w uporządkowane dane, a także usuwamy brakujące/niepoprawne punkty danych. Podczas EDA stworzymy wiele rodzajów wykresów i użyjemy

tych wykresów do zidentyfikowania kluczowych cech i relacji do wykorzystania w naszych modelach danych.

• Eksploracja danych to proces znajdowania relacji między elementami danych. Eksploracja danych to część nauki o danych, w której staramy się znaleźć relacje między zmiennymi (pomyśl model spawn-recruit).

• Do tej pory bardzo starałem się nie używać terminu big data. To dlatego, że uważam, że ten termin jest często nadużywany. Chociaż definicja tego słowa różni się w zależności od osoby, duże zbiory danych. Big Data to dane, które są zbyt duże, aby mogły zostać przetworzone przez pojedynczą maszynę (jeśli Twój laptop uległ awarii, może to oznaczać przypadek big data).

Studia przypadków związane z nauką o danych

Połączenie matematyki, programowania komputerowego i wiedzy dziedzinowej sprawia, że nauka o danych jest tak potężna. Często jednej osobie trudno jest opanować wszystkie trzy z tych obszarów. Dlatego bardzo często firmy zatrudniają zespoły analityków danych zamiast jednej osoby. Przyjrzyjmy się kilku potężnym przykładom analizy danych w działaniu i ich wynikom.

Studium przypadku - automatyzacja rządowej pracy kancelaryjnej

Wiadomo, że roszczenia z tytułu ubezpieczenia społecznego są poważnym problemem zarówno dla agenta, który je czyta, jak i dla osoby, która je napisała. Niektóre roszczenia zajmują ponad 2 lata, aby zostać rozwiązane w całości, a to absurd! Co składa się na roszczenie? To głównie tekst. Wypełnij to, potem tamto, potem to i tak dalej. Widać, jak trudno byłoby agentowi czytać je przez cały dzień, formularz po formularzu. Musi być lepszy sposób! Cóż, jest. Elder Research Inc. przeanalizował te niezorganizowane dane i był w stanie zautomatyzować 20% wszystkich formularzy ubezpieczenia społecznego osób niepełnosprawnych. Oznacza to, że komputer mógłby przejrzeć 20% tych pisemnych formularzy i wydać opinię na temat zatwierdzenia. Co więcej, firma zewnętrzna, która jest wynajęta do oceny aprobat formularzy, faktycznie nadała formom maszynowym wyższą ocenę niż formy ludzkie. Tak więc komputer nie tylko poradził sobie z 20% obciążeniem, ale średnio radził sobie lepiej niż człowiek.

Wystrzelaj wszystkich ludzi, prawda?

Zanim dostanę mnóstwo wściekłych e-maili, w których twierdzą, że data science oznacza koniec ludzkiej pracy, pamiętaj, że komputer był w stanie obsłużyć tylko 20% obciążenia. Oznacza to, że prawdopodobnie spisał się fatalnie na 80% form! To dlatego, że komputer był chyba świetny w prostych formach. Obliczenie twierdzeń, które zajęłoby człowiekowi minuty, zajęło komputerowi sekundy. Ale te minuty sumują się i zanim się zorientujesz, każdy człowiek jest ratowany przez ponad godzinę dziennie! Formularze, które mogą być łatwe do odczytania przez człowieka, są również łatwe do odczytania przez komputer. Dopiero gdy forma staje się bardzo zwięzła lub gdy pisarz zaczyna odchodzić od zwykłej gramatyki, komputer zaczyna zawodzić. Ten model jest świetny, ponieważ pozwala ludziom poświęcić więcej czasu na te trudne roszczenia i poświęca im więcej uwagi bez rozpraszania się natłokiem papierów.

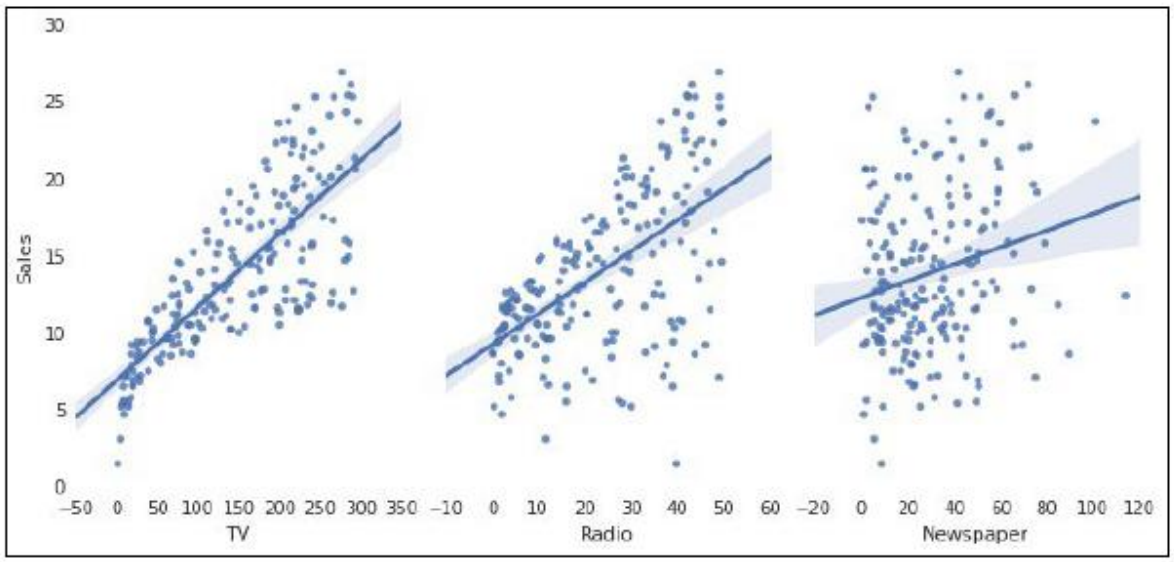
Zauważ, że użyłem słowa model. Pamiętaj, że model to relacja między elementami. W tym przypadku związek zachodzi między słowami pisemnymi a statusem zatwierdzenia roszczenia.

Studium przypadku - dolary marketingowe

Zestaw danych pokazuje związek między pieniędzmi wydanymi w kategoriach telewizji, radia i gazety. Celem jest analiza relacji między trzema różnymi mediami marketingowymi i ich wpływu na sprzedaż

produktu. Nasze dane mają postać struktury wierszowej i kolumnowej. Każdy wiersz reprezentuje region sprzedaży, a kolumny informują nas, ile pieniędzy wydano na każde medium i jaki zysk uzyskano w tym regionie.

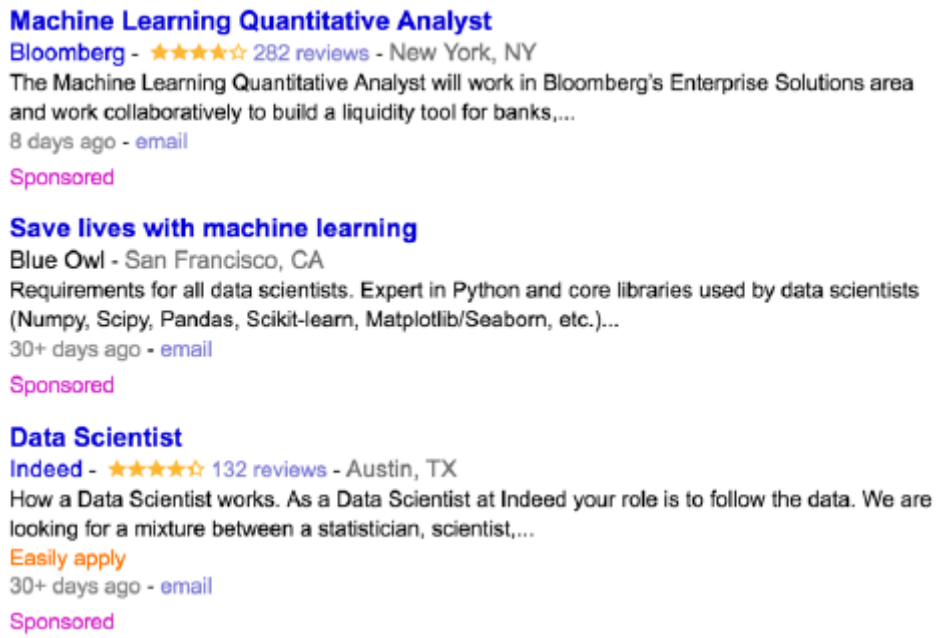
Zwykle badacz danych musi poprosić o jednostki i skalę. W tym przypadku powiem wam, że telewizja, radio i gazeta są mierzone w „tysiącach dolarów", a sprzedaż w „tysiącach sprzedanych gadżetów". Oznacza to, że w pierwszym regionie wydano 230 100 USD na reklamę telewizyjną, 37 800 USD na reklamę radiową i 69 200 USD na reklamę w prasie. W tym samym regionie sprzedano 22 100 sztuk. Na przykład w trzecim regionie wydaliśmy 17 200 USD na reklamę telewizyjną i sprzedaliśmy 9 300 widżetów. Jeśli wykreślimy każdą zmienną w funkcji sprzedaży, otrzymamy następujące wykresy:



Zwróć uwagę, że żadna z tych zmiennych nie tworzy bardzo silnej linii i dlatego może nie działać dobrze przy przewidywaniu sprzedaży (samodzielnie). Telewizja jest najbliższa w tworzeniu oczywistej relacji, ale nawet to nie jest wspaniałe. W takim przypadku będziemy musieli uformować bardziej złożony model niż ten, który zastosowaliśmy w modelu spawner-recruiter i połączyć wszystkie trzy zmienne w celu modelowania sprzedaży. Ten typ problemu jest bardzo powszechny w nauce o danych. W tym przykładzie próbujemy zidentyfikować kluczowe cechy związane ze sprzedażą produktu. Jeśli uda nam się wyizolować te kluczowe cechy, możemy wykorzystać te relacje i zmienić wysokość wydatków na reklamę w różnych miejscach z nadzieją na zwiększenie sprzedaży.

Studium przypadku - co zawiera opis stanowiska?

Szukasz pracy w data science? Świetnie, pozwól mi pomóc. W tym studium przypadku „wydobyłem" (zaczerpnięte z Internetu) 1000 opisów stanowisk dla firm aktywnie zatrudniających analityków danych (stan na styczeń 2016 r.). Celem jest przyjrzenie się niektórym z najczęstszych słów kluczowych, których ludzie używają w opisach stanowisk.




```
import requests

# used to grab data from the web

from BeautifulSoup import BeautifulSoup

# used to parse HTML

from sklearn.feature_extraction.text import CountVectorizer

# used to count number of words and phrases (we will be using this module a lot)
```

Pierwsze dwa importy służą do pobierania danych internetowych z witryny Indeed.com, a trzeci import ma po prostu zliczać, ile razy pojawia się słowo lub fraza.

```
texts = []

# hold our job descriptions in this list

for index in range(0,1000,10): # go through 100 pages of indeed

page = 'indeed.com/jobs?q=data+scientist&start='+str(index)

# identify the url of the job listings

web_result = requests.get(page).text

# use requests to actually visit the url

soup BeautifulSoup(web_result)

# parse the html of the resulting page

for listing in soup.findAll('span', {'class':'summary'}:

# for each listing on the page

texts.append(listing.text)
```

append the text of the listing to our list

Okej, zanim cię stracę, wszystko, co robi ta pętla, to przeglądanie 100 stron opisów stanowisk i dla każdej strony chwytanie każdego opisu stanowiska. Ważną zmienną są tutaj teksty, czyli lista ponad 1000 opisów stanowisk:

type(texts) # == list

vect = CountVectorizer(ngram_range=(1,2), stop_words='english')

Get basic counts of one and two word phrases

matrix = vect.fit_transform(texts)

fit and learn to the vocabulary in the corpus

print len(vect.get_feature_names()) # how many features are there

There are 11,293 total one and two words phrases in my case!!

Pominąłem tutaj trochę kodu. Wyniki są następujące (reprezentowane jako fraza, a następnie liczba wystąpień):

experience 320

machine 306

learning 305

machine learning 294

techniques 266

statistical 215

team 197

analytics 173

business 167

statistics 159

algorithms 152

datamining 149

software 144

applied 141

programming 132

understanding 127

world 127

research 125

datascience 123

methods 122

join 122

quantitative 122

group 121

real 120

large 120

Godne uwagi rzeczy:

• Uczenie maszynowe i doświadczenie znajdują się na szczycie listy. Doświadczenie przychodzi z praktyką.

• Po tych słowach znajdują się słowa statystyczne, które sugerują znajomość matematyki i teorii.

• Słowo zespół jest bardzo wysoko, co oznacza, że będziesz musiał pracować z zespołem analityków danych; nie będziesz samotnym wilkiem.

• Słowa informatyki, takie jak algorytmy i programowanie, są rozpowszechnione.

• Słowa techniki, rozumienie i metody oznaczają więcej podejścia teoretycznego, ambiwalentne dla dowolnej dziedziny.

• Słowo biznes implikuje określoną dziedzinę problemu.

Jest wiele interesujących rzeczy, o których warto wspomnieć w tym studium przypadku, ale największym wnioskiem jest to, że istnieje wiele kluczowych słów i fraz, które składają się na rolę nauki o danych. To nie tylko matematyka, kodowanie czy wiedza domenowa; to naprawdę połączenie tych trzech pomysłów (zilustrowanych na przykładzie jednej osoby lub w zespole wieloosobowym) sprawia, że nauka o danych jest możliwa i skuteczna.

PODSUMOWANIE

Na początku zadałem proste pytanie, jaki jest haczyk nauki o danych? Cóż, jest jeden. To nie tylko zabawa, gry i modelowanie. Nasze poszukiwania coraz inteligentniejszych maszyn i algorytmów muszą mieć swoją cenę. Gdy szukamy nowych i innowacyjnych sposobów odkrywania trendów w danych, bestia czai się w cieniu. Nie mówię o krzywej uczenia się matematyki czy programowania, ani o nadmiarze danych. Epoka przemysłowa pozostawiła nas w ciągłej walce z zanieczyszczeniem. Kolejna era informacji pozostawiła po sobie ślad big data. Jakie więc niebezpieczeństwa może nam przynieść wiek danych? Epoka danych może prowadzić do czegoś znacznie bardziej złowrogiego - odczłowieczenia jednostki przez masowe dane. Coraz więcej osób wskakuje w dziedzinę nauki o danych, większość bez wcześniejszego doświadczenia w matematyce lub CS, co na pierwszy rzut oka jest świetne. Przeciętni analitycy danych mają dostęp do milionów danych profili randkowych, tweetów, recenzji online i wielu innych, aby przyspieszyć swoją edukację. Jeśli jednak wskoczysz do nauki o danych bez odpowiedniego kontaktu z teorią lub praktykami kodowania i bez poszanowania domeny, w której pracujesz, ryzykujesz nadmierne uproszczenie samego zjawiska, które próbujesz modelować. Załóżmy na przykład, że chcesz zautomatyzować lejek sprzedaży, tworząc uproszczony program, który szuka w LinkedIn bardzo konkretnych słów kluczowych w profilu LinkedIn danej osoby.

keywords = ["Saas", "Sprzedaż", "Przedsiębiorstwo"]

Świetnie, teraz możesz szybko przeskanować LinkedIn, aby znaleźć osoby spełniające Twoje kryteria. Ale co z osobą, która pisze „Software as a Service” zamiast „Saas” lub błędnie pisze „enterprise”. W jaki sposób Twój model zorientuje się, że ci ludzie również dobrze do siebie pasują? Nie należy ich zostawiać tylko dlatego, że data scientist na skróty w tak łatwy sposób nadmiernie uogólnił ludzi. Programista zdecydował się uprościć wyszukiwanie drugiego człowieka, szukając trzech podstawowych słów kluczowych i skończył z wieloma niewykorzystanymi szansami na stole. W następnym rozdziale przyjrzymy się różnym typom danych, które istnieją na świecie, od dowolnego tekstu po wysoce ustrukturyzowane pliki wierszy/kolumn. Przyjrzymy się również operacjom matematycznym, które są dozwolone dla różnych typów danych, a także wywnioskujemy spostrzeżenia na podstawie formy danych, które przychodzą.

Typy Danych

Teraz, gdy mamy podstawowe wprowadzenie do świata nauki o danych i rozumiemy, dlaczego ta dziedzina jest tak ważna, przyjrzyjmy się różnym sposobom tworzenia danych. W szczególności w tej części przyjrzymy się następującym tematom:

• Dane ustrukturyzowane a nieustrukturyzowane

• Dane ilościowe a dane jakościowe

• Cztery poziomy danych

Zagłębimy się w każdy z tych tematów, pokazując przykłady tego, jak naukowcy zajmujący się danymi patrzą na dane i jak z nimi pracują. Ten rozdział ma na celu zapoznanie się z podstawowymi ideami nauki o danych.

Smaki danych

W terenie ważne jest, aby zrozumieć różne rodzaje danych z kilku powodów. Nie tylko rodzaj danych będzie dyktować metody stosowane do analizy i wyodrębniania wyników, wiedza o tym, czy dane są nieustrukturyzowane, czy może ilościowe, może również wiele powiedzieć o mierzonym zjawisku w świecie rzeczywistym. Przyjrzymy się trzem podstawowym klasyfikacjom danych:

• Strukturalny a nieustrukturyzowany (czasami nazywany zorganizowanym lub niezorganizowanym)

• Ilościowe a jakościowe

• Cztery poziomy danych

Pierwszą rzeczą, na którą należy zwrócić uwagę, jest użycie przeze mnie słowa data. W ostatniej części określiłem dane jako jedynie zbiór informacji. Ta niejasna definicja istnieje, ponieważ możemy podzielić dane na różne kategorie i potrzebujemy, aby nasza definicja była luźna. Następną rzeczą do zapamiętania podczas przechodzenia przez tą część jest to, że w większości przypadków, kiedy mówię o rodzaju danych, będę odnosić się do określonej cechy zestawu danych lub do całego zestawu danych jako całości. Będę bardzo jasno określał, do którego w danym momencie się odwołuję.

Po co patrzeć na te rozróżnienia?

Zatrzymanie się i zastanowienie nad rodzajem danych może wydawać się bezwartościowe. Mamy przed sobą zabawne rzeczy, takie jak statystyki i uczenie maszynowe, ale jest to prawdopodobnie jeden z najważniejszych kroków, które musisz podjąć, aby przeprowadzić analizę danych. Rozważmy przykład, w którym patrzymy na wyniki wyborów dla hrabstwa. W zbiorze danych osób znajduje się kolumna „rasa", która jest oznaczona numerem identyfikacyjnym, aby zaoszczędzić miejsce. Na przykład być może kaukaski jest oznaczany przez 7, podczas gdy azjatycki amerykański to 2. Bez zrozumienia, że te liczby nie są w rzeczywistości liczbami uporządkowanymi, jak o nich myślimy (gdzie 7 jest większe niż 2, a zatem kaukaski jest „większy niż" azjatycki Amerykanin) zrobimy straszne błędy w naszej analizie.

Ta sama zasada dotyczy nauki o danych. Po otrzymaniu zestawu danych kuszące jest, aby przejść od razu do eksploracji, stosowania modeli statystycznych i badania zastosowań uczenia maszynowego w celu szybszego uzyskania wyników. Jeśli jednak nie rozumiesz typu danych, z którymi pracujesz, możesz stracić dużo czasu na stosowanie modeli, o których wiadomo, że są nieefektywne w przypadku tego konkretnego typu danych. Kiedy otrzymujesz nowy zestaw danych, zawsze zalecam poświęcenie około godziny (zwykle mniej) na dokonanie rozróżnień wymienionych w kolejnych sekcjach.

Dane ustrukturyowane i nieustrukturyzowane

Rozróżnienie między danymi ustrukturyzowanymi i nieustrukturyzowanymi jest zwykle pierwszym pytaniem, jakie chcesz zadać sobie na temat całego zbioru danych. Odpowiedź na to pytanie może oznaczać różnicę między potrzebą trzech dni lub trzech tygodni na wykonanie prawidłowej analizy. Podstawowy podział jest następujący (jest to przerobiona definicja uporządkowanych i niezorganizowanych danych w pierwszej Części):

• Dane ustrukturyzowane (zorganizowane): Są to dane, które można traktować jako obserwacje i cechy. Zwykle jest zorganizowany przy użyciu metody tabelarycznej (wiersze i kolumny).

• Dane nieustrukturyzowane (niezorganizowane): Dane te istnieją jako wolna jednostka i nie podlegają żadnej standardowej hierarchii organizacyjnej. Oto kilka przykładów, które mogą pomóc w rozróżnieniu między nimi:

• Większość danych, które istnieją w formie tekstowej, w tym dzienniki serwera i posty na Facebooku, jest nieustrukturyzowana

• Obserwacje naukowe, zarejestrowane przez uważnych naukowców, są przechowywane w bardzo schludnym i zorganizowanym (ustrukturyzowanym) formacie

• Sekwencja genetyczna nukleotydów chemicznych (na przykład ACGTATTGCA) jest pozbawiona struktury, nawet jeśli kolejność nukleotydów ma znaczenie, ponieważ nie możemy utworzyć deskryptorów sekwencji przy użyciu formatu wiersza/kolumny bez dalszego przyjrzenia się

Ogólnie uważa się, że ustrukturyzowane dane są znacznie łatwiejsze w pracy i analizie. Większość modeli statystycznych i uczenia maszynowego została zbudowana z myślą o danych strukturalnych i nie może działać na luźnej interpretacji danych niestrukturalnych. Naturalna struktura rzędów i kolumn jest łatwa do przyswojenia dla oczu człowieka i maszyny. Po co więc w ogóle mówić o danych nieustrukturyzowanych? Bo to takie powszechne! Większość szacunków umieszcza dane nieustrukturyzowane jako 80-90% danych światowych. Dane te istnieją w wielu formach iw większości pozostają niezauważone przez ludzi jako potencjalne źródło danych. Tweety, e-maile, literatura i logi serwera to ogólnie nieustrukturyzowane formy danych. Chociaż naukowcy zajmujący się danymi prawdopodobnie preferują dane ustrukturyzowane, muszą być w stanie poradzić sobie z ogromnymi ilościami nieustrukturyzowanych danych na świecie. Jeśli 90% światowych danych jest nieustrukturyzowanych, oznacza to, że około 90% światowych informacji jest uwięzionych w trudnym formacie. Tak więc, ponieważ większość naszych danych istnieje w tym dowolnym formacie, musimy zwrócić się do technik analizy wstępnej, zwanej przetwarzaniem wstępnym, aby nadać strukturę przynajmniej części danych do dalszej analizy. Następny rozdział zajmie się bardzo szczegółowo przetwarzaniem wstępnym; na razie rozważymy część przetwarzania wstępnego, w której próbujemy zastosować przekształcenia, aby przekonwertować nieustrukturyzowane dane na ustrukturyzowany odpowiednik.

Przykład wstępnego przetwarzania danych

Patrząc na dane tekstowe (które prawie zawsze uważane są za nieustrukturyzowane), mamy wiele możliwości przekształcenia zbioru w format ustrukturyzowany. Możemy to zrobić, stosując nowe cechy opisujące dane. Oto kilka takich cech:

• Liczba słów/fraz

• Istnienie pewnych znaków specjalnych

• Względna długość tekstu

• Wybieranie tematów

Użyję poniższego tweeta jako szybkiego przykładu nieustrukturyzowanych danych, ale możesz użyć dowolnego nieustrukturyzowanego tekstu w dowolnej formie, który Ci się podoba, w tym tweetów i postów na Facebooku. Czy w ten środowy poranek wstajesz wcześnie? Następnie spójrz na wschód. Półksiężyc dołącza do Wenus i Saturna. Unosząc się na niebie o świcie. Ważne jest, aby powtórzyć, że dla tego tweeta konieczne jest wstępne przetwarzanie, ponieważ zdecydowana większość algorytmów uczenia wymaga danych liczbowych (do których omówimy po tym przykładzie). Wstępne przetwarzanie nie tylko wymaga określonego rodzaju danych, ale pozwala nam zbadać funkcje, które zostały utworzone na podstawie istniejących funkcji. Na przykład ze wspomnianego tweeta możemy wyodrębnić takie funkcje, jak liczba słów i znaki specjalne. Przyjrzyjmy się teraz kilku funkcjom, które możemy wyodrębnić z tekstu.

Liczba słów/fraz

Możemy podzielić tweeta na liczbę słów/fraz. Słowo to pojawia się w tweecie raz, podobnie jak każde inne słowo. Możemy przedstawić ten tweet w ustrukturyzowanym formacie w następujący sposób, konwertując w ten sposób nieustrukturyzowany zestaw słów na format wiersza/kolumny:

	this	wednesday	morn	are	this wednesday
Word Count	1	1	1	1	1

Zauważ, że aby uzyskać ten format, możemy wykorzystać scikit-learn's CountVectorizer, który widzieliśmy w poprzedniej części.

Obecność niektórych znaków specjalnych

Możemy również przyjrzeć się obecności znaków specjalnych, takich jak znak zapytania i wykrzyknik. Pojawienie się tych znaków może sugerować pewne koncepcje dotyczące danych, które w inny sposób są trudne do poznania. Na przykład fakt, że ten tweet zawiera znak zapytania, może silnie sugerować, że zawiera on pytanie do czytelnika. Możemy dołączyć do poprzedniej tabeli nową kolumnę, jak pokazano:

	this	wednesday	morn	are	this wednesday	?
Word Count	1	1	1	1	1	1

Względna długość tekstu

Ten tweet ma 121 znaków.

len("This Wednesday morn, are you early to rise? Then look East. The Crescent Moon joins Venus & Saturn. Afloat in the dawn skies.")

get the length of this text (number of characters for a string)

121

Przeciętny tweet, jak odkryli analitycy, ma około 30 znaków. Możemy więc narzucić nową cechę, zwaną długością względną (która jest długością tweeta podzieloną przez średnią długość), informującą nas o

długości tego tweeta w porównaniu z przeciętnym tweetem. Ten tweet jest w rzeczywistości 4,03 razy dłuższy niż przeciętny tweet, jak pokazano:

121/30 = 4,03

Możemy dodać kolejną kolumnę do naszej tabeli za pomocą tej metody:

	this	wednesday	morn	are	this wednesday	?	Relative length
Word Count	1	1	1	1	1	1	4.03

Wybieranie tematów

Możemy wybrać niektóre tematy tweeta, aby dodać je jako kolumny. Ten tweet dotyczy astronomii, więc możemy dodać kolejną kolumnę, jak pokazano na ilustracji:

	this	wednesday	morn	are	this wednesday	?	Relative length	Topic
Word Count	1	1	1	1	1	1	4.03	astronomy

I tak po prostu, możemy przekonwertować fragment tekstu na ustrukturyzowane/zorganizowane dane gotowe do użycia w naszych modelach i analizie eksploracyjnej. Temat jest jedyną wyodrębnioną funkcją, którą przyjrzeliśmy się, która nie jest automatycznie wyprowadzana z tweeta. Sprawdzanie liczby słów i długości tweetów w Pythonie jest łatwe; jednak bardziej zaawansowane modele (zwane modelami tematycznymi) są w stanie wyprowadzać i przewidywać tematy jako tekst naturalny. Możliwość szybkiego rozpoznania, czy Twoje dane są ustrukturyzowane czy nieustrukturyzowane, może zaoszczędzić godziny, a nawet dni pracy w przyszłości. Gdy już jesteś w stanie rozróżnić organizację prezentowanych danych, następne pytanie dotyczy indywidualnych cech zestawu danych.

Dane ilościowe a dane jakościowe

Kiedy pytasz analityka danych „co to za dane?", zazwyczaj zakładają, że pytasz go, czy są to dane głównie ilościowe, czy jakościowe. Jest to prawdopodobnie najczęstszy sposób opisywania specyficznych cech zestawu danych. W większości przypadków, gdy mówimy o danych ilościowych, zwykle (nie zawsze) mówimy o ustrukturyzowanym zbiorze danych o ścisłej strukturze wierszy/kolumn (ponieważ nie zakładamy, że nieustrukturyzowane dane mają nawet jakiekolwiek cechy). Tym bardziej, że etap przetwarzania wstępnego jest tak ważny. Te dwa typy danych można zdefiniować w następujący sposób:

• Dane ilościowe: Dane te można opisać za pomocą liczb, a podstawowe procedury matematyczne, w tym dodawanie, są możliwe na zestawie.

• Dane jakościowe: Tych danych nie można opisać za pomocą liczb i podstawowej matematyki. Te dane są ogólnie uważane za opisane przy użyciu kategorii i języka „naturalnego".

Przykład - dane kawiarni

Powiedzmy, że przetwarzaliśmy obserwacje kawiarni w dużym mieście przy użyciu następujących pięciu deskryptorów (cech):

Dane: kawiarnia

• Nazwa kawiarni

• Przychody (w tysiącach złotych)

• Kod pocztowy

• Przeciętni klienci miesięcznie

• Kraj pochodzenia kawy

Każdą z tych cech można sklasyfikować jako ilościową lub jakościową, a to proste rozróżnienie może wszystko zmienić. Przyjrzyjmy się każdemu z nich:

• Nazwa kawiarni - Jakościowa

Nazwa kawiarni nie jest wyrażona jako liczba i nie możemy wykonać obliczeń matematycznych na nazwie sklepu.

• Przychody - ilościowe

Ile pieniędzy przynosi kawiarnia, z pewnością można opisać za pomocą liczby. Możemy również wykonywać podstawowe operacje, takie jak sumowanie przychodów przez 12 miesięcy, aby uzyskać roczny przychód.

• Kod pocztowy - jakościowy

Ten jest trudny. Kod pocztowy jest zawsze przedstawiany za pomocą liczb, ale to, co czyni go jakościowym, to fakt, że nie pasuje do drugiej części definicji ilościowej - nie możemy wykonać podstawowych operacji matematycznych na kodzie pocztowym. Jeśli dodamy do siebie dwa kody pocztowe, jest to bezsensowny pomiar. Niekoniecznie otrzymujemy nowy kod pocztowy i zdecydowanie nie otrzymujemy „podwójnego kodu pocztowego".

• Przeciętni miesięczni klienci - ilościowo

Ponownie, opisanie tego czynnika za pomocą liczb i dodawania ma sens. Dodaj wszystkich swoich miesięcznych klientów, a otrzymasz rocznych klientów.

• Kraj pochodzenia kawy – jakościowy

Załóżmy, że jest to bardzo mała kawiarnia z kawą jednego pochodzenia. Ten kraj jest opisany nazwą (etiopską, kolumbijską), a nie cyframi.

Kilka ważnych rzeczy do zapamiętania:

• Mimo że kod pocztowy jest opisywany za pomocą liczb, nie jest on ilościowy. Dzieje się tak, ponieważ nie można mówić o sumie wszystkich kodów pocztowych lub średnim kodzie pocztowym. To są bezsensowne opisy.

• Prawie zawsze, gdy słowo jest używane do opisania cechy, jest to czynnik jakościowy.

Jeśli masz problem z określeniem, który z nich jest zasadniczo, próbując zdecydować, czy dane są jakościowe, czy ilościowe, zadaj sobie kilka podstawowych pytań dotyczących cech danych:

• Czy możesz to opisać za pomocą liczb?

- Nie? To jest jakościowe.

- Tak? Przejdź do następnego pytania.

• Czy po dodaniu ich do siebie nadal ma to sens?

- Nie? Są jakościowe.

- Tak? Prawdopodobnie masz dane ilościowe.

Ta metoda pomoże Ci zaklasyfikować większość, jeśli nie wszystkie, dane do jednej z tych dwóch kategorii. Różnica między tymi dwiema kategoriami określa rodzaje pytań, które możesz zadać w każdej kolumnie. W przypadku kolumny ilościowej możesz zadawać pytania takie jak:

• Jaka jest średnia wartość?

• Czy ta ilość rośnie czy maleje z upływem czasu (jeśli czas jest czynnikiem)?

• Czy istnieje próg, dla którego wzrost liczby powyżej lub zbyt niski będzie sygnalizował kłopoty dla firmy?

W przypadku kolumny jakościowej nie można odpowiedzieć na żadne z poprzednich pytań; jednak poniższe pytania dotyczą tylko wartości jakościowych:

• Która wartość występuje najczęściej, a która najmniej?

• Ile jest unikalnych wartości?

• Jakie są te wyjątkowe wartości?

Przykład - dane dotyczące spożycia alkoholu na świecie

Światowa Organizacja Zdrowia opublikowała zestaw danych opisujący przeciętne nawyki związane z piciem ludzi w krajach na całym świecie. Użyjemy Pythona i narzędzia do eksploracji danych Pandas, aby uzyskać lepszy wygląd:

```
import pandas as pd

# read in the CSV file from a URL

drinks = pd.read_csv('https://raw.githubusercontent.com/sinanuozdemir/

principles_of_data_science/master/data/chapter_2/drinks.csv')

# examine the data's first five rows

drinks.head() # print the first 5 rows
```

Te trzy wiersze wykonały następujące czynności:

• Importowane pandy, które w przyszłości będą określane jako pd

• Wczytaj plik CSV (wartości oddzielone przecinkami) jako zmienną o nazwie napoje

• Wywołano metodę, head, która ujawnia pierwsze pięć wierszy zbioru danych

	country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol	continent
0	Afghanistan	0	0	0	0.0	AS
1	Albania	89	132	54	4.9	EU
2	Algeria	25	0	14	0.7	AF
3	Andorra	245	138	312	12.4	EU
4	Angola	217	57	45	5.9	AF

W tym przykładzie pracujemy z sześcioma różnymi kolumnami:

• kraj: Jakościowy

• beer_servings: ilościowe

• spirit_servings: ilościowo

• wine_servings: Ilościowe

• total_litres_of_pure_alcohol: ilościowe

• kontynent: jakościowy

Spójrzmy na jakościowy kontynent kolumnowy. Możemy użyć Pand, aby uzyskać podstawowe statystyki podsumowujące dotyczące tej nienumerycznej cechy. Stosowana jest tutaj metoda description(), która najpierw określa, czy kolumna jest prawdopodobnie ilościowa, czy jakościowa, a następnie podaje podstawowe informacje o kolumnie jako całości. Jest to pokazane w następujący sposób:

drinks['continent'].describe()

>> count 193

>> unique 5

>> top AF

>> freq 53

Wynika z niego, że WHO zgromadziła dane dotyczące pięciu unikalnych kontynentów, z których najczęstszym jest AF (Afryka), który wystąpił 53 razy w 193 obserwacjach. Jeśli spojrzymy na jedną z kolumn ilościowych i wywołamy tę samą metodę, zobaczymy różnicę w wynikach, jak pokazano:

drinks['beer_servings'].describe()

>> mean 106.160622

>> min 0.000000

>> max 376.000000

Teraz możemy przyjrzeć się średniej (średniej) porcji piwa na osobę w danym kraju (106,2 porcji), jak również najniższej porcji piwa, zero, i najwyższej zarejestrowanej porcji piwa, 376 (to więcej niż jedno piwo dziennie).

Kopać głębiej

Dane ilościowe można podzielić krok dalej na wielkości dyskretne i ciągłe. Można je zdefiniować w następujący sposób:

• Dane dyskretne: Opisuje zliczane dane. Może przyjmować tylko określone wartości.

Przykładami dyskretnych danych ilościowych są rzut kostką, ponieważ może przyjąć tylko sześć wartości, oraz liczbę klientów w kawiarni, ponieważ nie można mieć prawdziwego zakresu osób.

• Dane ciągłe: Opisuje dane, które są mierzone. Istnieje na nieskończonym zakresie wartości.

Dobrym przykładem danych ciągłych może być waga osoby, ponieważ może ona wynosić 150 funtów lub 197,66 funtów (zwróć uwagę na liczby dziesiętne). Wysokość osoby lub budynku jest liczbą ciągłą, ponieważ możliwa jest nieskończona liczba miejsc dziesiętnych. Innymi przykładami danych ciągłych są czas i temperatura.

Droga do tej pory…

Do tej pory przyjrzeliśmy się różnicom między danymi ustrukturyzowanymi i nieustrukturyzowanymi, a także między cechami jakościowymi i ilościowymi. Te dwa proste rozróżnienia mogą mieć drastyczny wpływ na przeprowadzaną analizę. Pozwólcie, że podsumuję, zanim przejdę do drugiej połowy. Dane jako całość mogą być ustrukturyzowane lub nieustrukturyzowane, co oznacza, że dane mogą mieć zorganizowaną strukturę wierszy/kolumn z odrębnymi cechami opisującymi każdy wiersz zestawu danych lub istnieć w stanie swobodnym, który zwykle musi być wstępnie przetworzony forma, która jest łatwo przyswajalna. Jeśli dane są ustrukturyzowane, możemy spojrzeć na każdą kolumnę (cechę) zbioru danych jako ilościową lub jakościową. Zasadniczo, czy kolumnę można opisać za pomocą matematyki i liczb, czy nie? W kolejnej części dane zostaną podzielone na cztery bardzo szczegółowe i szczegółowe poziomy. Przy każdym zamówieniu będziemy stosować bardziej skomplikowane reguły matematyki, a dzięki temu możemy uzyskać bardziej intuicyjne i wymierne zrozumienie danych.

Cztery poziomy danych

Ogólnie przyjmuje się, że konkretną cechę (cecha/kolumnę) danych strukturalnych można podzielić na jeden z czterech poziomów danych. Poziomy to:

• Poziom nominalny

• Poziom porządkowy

• Poziom interwału

• Poziom proporcji

W miarę przesuwania się w dół listy zyskujemy większą strukturę, a tym samym więcej zwrotów z naszej analizy. Każdy poziom ma własną przyjętą praktykę pomiaru środka danych. Zwykle myślimy o średniej/średniej jako o akceptowalnej formie centrum, jednak dotyczy to tylko określonego typu danych.

Poziom nominalny

Pierwszy poziom danych, poziom nominalny (który również brzmi jak słowo nazwa) składa się z danych, które są opisane wyłącznie nazwą lub kategorią. Podstawowe przykłady obejmują płeć, narodowość, gatunek lub szczep drożdży w piwie. Nie są one opisane liczbami i dlatego są jakościowe. Oto kilka przykładów:

• Rodzaj zwierzęcia znajduje się na nominalnym poziomie danych. Możemy również powiedzieć, że jeśli jesteś szympansem, to należysz również do klasy ssaków.

• Część mowy jest również uwzględniana na nominalnym poziomie danych. Słowo ona jest zaimkiem, a także rzeczownikiem.

Oczywiście, będąc jakościowymi, nie możemy wykonywać żadnych ilościowych operacji matematycznych, takich jak dodawanie czy dzielenie. To nie miałoby sensu.

Dozwolone operacje matematyczne

Nie możemy wykonywać matematyki na nominalnym poziomie danych, z wyjątkiem podstawowych funkcji równości i przynależności zbioru, jak pokazano w następujących dwóch przykładach:

• Bycie przedsiębiorcą technologicznym jest tym samym, co bycie w branży technologicznej, ale nie odwrotnie

• Figura opisana jako kwadrat podpada pod opis bycia prostokątem, ale nie odwrotnie

Miary centrum

Miarą środka jest liczba opisująca tendencję do danych. Czasami określa się go jako punkt równowagi danych. Typowe przykłady to średnia, mediana i tryb.

Aby znaleźć centrum danych nominalnych, zazwyczaj zwracamy się do trybu (najczęstszy element) zbioru danych. Spójrz na przykład wstecz na dane WHO dotyczące spożycia alkoholu. Najczęściej badanym kontynentem była Afryka, co czyni go możliwym wyborem na środek kolumny kontynentu. Miary środka, takie jak średnia i mediana, nie mają sensu na tym poziomie, ponieważ nie możemy uporządkować obserwacji ani nawet dodać ich do siebie.

Jak wyglądają dane na poziomie nominalnym

Dane na poziomie nominalnym mają głównie charakter kategoryczny. Ponieważ generalnie możemy używać tylko słów do opisania danych, mogą one zostać utracone w tłumaczeniu między krajami, a nawet mogą zostać błędnie napisane. Chociaż dane na tym poziomie z pewnością mogą być przydatne, musimy uważać na to, jakie wnioski możemy z nich wyciągnąć. Mając tylko tryb jako podstawową miarę środka, nie jesteśmy w stanie wyciągnąć wniosków na temat średniej obserwacji. Ta koncepcja nie istnieje na tym poziomie. Dopiero na następnym poziomie możemy zacząć wykonywać prawdziwą matematykę na naszych obserwacjach.

Poziom porządkowy

Poziom nominalny nie zapewniał nam dużej elastyczności w zakresie działań matematycznych ze względu na jeden pozornie nieistotny fakt - nie mogliśmy w naturalny sposób uporządkować obserwacji. Dane na poziomie porządkowym zapewniają nam porządek rangowy lub sposób na umieszczenie jednej obserwacji przed drugą; jednak nie zapewnia nam względnych różnic między obserwacjami, co oznacza, że chociaż możemy uporządkować obserwacje od pierwszej do ostatniej, nie możemy ich dodawać ani odejmować, aby uzyskać jakiekolwiek rzeczywiste znaczenie.

Przykłady

Likert jest jedną z najpopularniejszych skal porządkowych. Za każdym razem, gdy otrzymujesz ankietę z prośbą o ocenę zadowolenia w skali od 1 do 10, podajesz dane na poziomie porządkowym. Twoja odpowiedź, która musi zawierać się w przedziale od 1 do 10, może być uporządkowana: osiem jest

lepsze niż siedem, a trzy gorsze niż dziewięć. Jednak różnice między liczbami nie mają większego sensu. Różnica między siódemką a szóstką może być inna niż różnica między dwójką a jedynką.

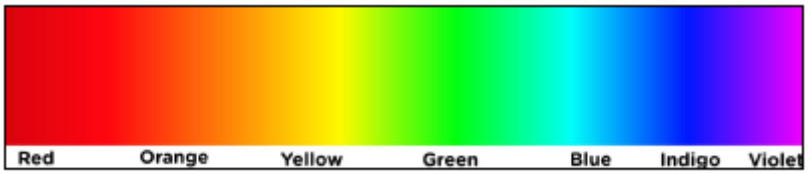
Dozwolone operacje matematyczne

Na tym poziomie mamy dużo więcej swobody w operacjach matematycznych. Dziedziczymy całą matematykę z poziomu porządkowego (równość i przynależność do zbioru), a do listy operacji dozwolonych na poziomie nominalnym możemy również dodać:

• Porządkowanie

• Porównanie

Zamówienie odnosi się do naturalnego porządku, jaki zapewniają nam dane; jednak czasami może to być trudne do rozszyfrowania. Mówiąc o widmie światła widzialnego, możemy odnieść się do nazw kolorów - czerwony, pomarańczowy, żółty, zielony, niebieski, indygo i fioletowy. Naturalnie, gdy poruszamy się od lewej do prawej, światło nabiera energii i inne właściwości. Możemy to nazwać porządkiem naturalnym.



Jednak w razie potrzeby artysta może narzucić innym porządek danych, np. sortować kolory w oparciu o koszt materiału, z którego dany kolor będzie wykonany. Może to zmienić kolejność danych, ale dopóki jesteśmy konsekwentni w tym, co definiuje kolejność, nie ma znaczenia, co ją definiuje. Porównania to kolejna nowa operacja dozwolona na tym poziomie. Na poziomie porządkowym nie miałoby sensu mówienie, że jeden kraj był z natury lepszy od drugiego lub że jedna część mowy jest gorsza od drugiej. Na poziomie porządkowym możemy dokonać tych porównań. Na przykład możemy porozmawiać o tym, że umieszczenie „7” w ankiecie jest gorsze niż umieszczenie „10”.

Miary centrum

Na poziomie porządkowym mediana jest zwykle odpowiednim sposobem zdefiniowania centrum danych. Środek byłby jednak niemożliwy, ponieważ podział nie jest dozwolony na tym poziomie. Możemy też korzystać z trybu, tak jak moglibyśmy na poziomie nominalnym. Przyjrzymy się teraz przykładowi użycia mediany: Wyobraź sobie, że przeprowadziłeś ankietę wśród swoich pracowników, pytając „jak bardzo jesteś zadowolony, że tu pracujesz w skali od 1 do 5”, a Twoje wyniki są następujące:

5, 4, 3, 4, 5, 3, 2, 5, 3, 2, 1, 4, 5, 3, 4, 4, 5, 4, 2, 1, 4, 5, 4,

3, 2, 4, 4, 5, 4, 3, 2, 1

Użyjmy Pythona, aby znaleźć medianę tych danych. Warto zauważyć, że większość ludzi twierdzi, że średnia z tych wyników działałaby dobrze. Powodem, dla którego średnia nie byłaby tak matematycznie realna, jest to, że jeśli odejmiemy/dodamy dwie oceny, powiedzmy, że cztery minus dwa, różnica dwóch tak naprawdę nic nie znaczy. Jeśli dodawanie/odejmowanie wyników nie ma sensu, średnia też nie ma sensu.

import numpy

```
results = [5, 4, 3, 4, 5, 3, 2, 5, 3, 2, 1, 4, 5, 3, 4, 4, 5, 4, 2, 1,

4, 5, 4, 3, 2, 4, 4, 5, 4, 3, 2, 1]

sorted_results = sorted(results)

print sorted_results

'''

[1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4,

4, 4, 5, 5, 5, 5, 5, 5, 5]

'''

print numpy.mean(results) # == 3.4375

print numpy.median(results) # == 4.0
```

Znak „” (potrójny apostrof) oznacza dłuższy (ponad dwie linie) komentarz. Działa w sposób podobny do #.

Okazuje się, że mediana jest nie tylko bardziej dźwięczna, ale sprawia, że wyniki ankiety wyglądają znacznie lepiej.

Szybkie podsumowanie i sprawdzenie

Do tej pory widzieliśmy połowę poziomów danych:

• Poziom nominalny

• Poziom porządkowy

Na poziomie nominalnym mamy do czynienia z danymi zwykle opisywanymi za pomocą słownictwa (ale czasami za pomocą liczb), bez uporządkowania i z niewielkim wykorzystaniem matematyki. Na poziomie porządkowym mamy dane, które można opisać za pomocą liczb, a także mają „naturalny” porządek, co pozwala nam umieścić jedno przed drugim. Spróbujmy sklasyfikować następujący przykład jako porządkowy lub nominalny :

• Pochodzenie ziaren w filiżance kawy

• Miejsce, które ktoś otrzymuje po ukończeniu wyścigu pieszego

• Metal użyty do wykonania medalu, który otrzymują po zajęciu miejsca we wspomnianym wyścigu

• Numer telefonu klienta

• Ile filiżanek kawy wypijasz dziennie

Poziom interwału

Teraz docieramy do czegoś ciekawego. Na poziomie przedziału zaczynamy patrzeć na dane, które można wyrazić za pomocą bardzo wymiernych środków i gdzie dozwolone są znacznie bardziej skomplikowane formuły matematyczne. Podstawowa różnica między poziomem porządkowym a poziomem interwału to po prostu różnica. Dane na poziomie interwału umożliwiają znaczące odejmowanie między punktami danych.

Przykład

Temperatura jest doskonałym przykładem danych na poziomie interwału. Jeśli jest 100 stopni Fahrenheita w Teksasie i 80 stopni Fahrenheita w Stambule w Turcji, to w Teksasie jest o 20 stopni cieplej niż w Stambule. Ten prosty przykład pozwala na o wiele więcej manipulacji na tym poziomie niż poprzednie przykłady.

(Nie) Przykład

Wydaje się, że przykład na poziomie porządkowym (przy użyciu ankiety od jednego do pięciu) pasuje do rachunku poziomu przedziału. Pamiętaj jednak, że różnica między wynikami (gdy je odejmiesz) nie ma sensu, dlatego danych tych nie można wywołać na poziomie interwału.

Dozwolone operacje matematyczne

Możemy użyć wszystkich operacji dozwolonych na niższych poziomach (porządkowanie, porównania itp.) wraz z dwoma innymi ważnymi operacjami:

• Dodawanie

• Odejmowanie

Uwzględnienie tych dwóch operacji pozwala nam mówić o danych na tym poziomie w zupełnie nowy sposób.

Miary centrum

Na tym poziomie możemy użyć mediany i trybu do opisania tych danych; jednak zwykle najdokładniejszym opisem centrum danych byłaby średnia arytmetyczna, częściej nazywana po prostu „średnią". Przypomnijmy, że definicja średniej wymaga zsumowania wszystkich pomiarów. Na poprzednich poziomach dodawanie było bez znaczenia; dlatego średnia straciłaby na wartości ekstremalnej. Dopiero na poziomie interwału i powyżej średnia arytmetyczna ma sens. Przyjrzymy się teraz przykładowi użycia średniej. Załóżmy, że przyjrzymy się temperaturze lodówki zawierającej nową szczepionkę firmy farmaceutycznej. Mierzymy temperaturę umiarkowaną co godzinę za pomocą następujących punktów danych (w stopniach Fahrenheita):

31, 32, 32, 31, 28, 29, 31, 38, 32, 31, 30, 29, 30, 31, 26

Używając ponownie Pythona, znajdźmy średnią i medianę danych:

```
import numpy

temps = [31, 32, 32, 31, 28, 29, 31, 38, 32, 31, 30, 29, 30, 31, 26]

print numpy.mean(temps) # == 30.73

print numpy.median(temps) # == 31.0
```

Zauważ, że średnia i mediana są dość blisko siebie i obie wynoszą około 31 stopni. Pytanie jak średnio zimna jest lodówka?, około 31, jednak do szczepionki dołączone jest ostrzeżenie: Nie trzymaj tej szczepionki w temperaturze poniżej 29 stopni. Zauważ, że co najmniej dwa razy temperatura spadła poniżej 29 stopni, ale ostatecznie założyłeś, że to nie wystarczy, aby była szkodliwa. W tym miejscu miara zmienności może pomóc nam zrozumieć, jak zła może być sytuacja w lodówce.

Miary zmienności

To jest coś nowego, o czym jeszcze nie rozmawialiśmy. Jedną rzeczą jest mówienie o centrum danych, ale w nauce o danych bardzo ważne jest również, aby wspomnieć, jak „rozprzestrzeniają się" dane.

Miary opisujące to zjawisko nazywane są miarami zmienności. Prawdopodobnie słyszałeś już o „odchyleniu standardowym” i teraz doświadczasz łagodnego PTSD na zajęciach statystycznych. Ten pomysł jest niezwykle ważny i chciałbym się nim pokrótce omówić. Miarą zmienności (jak odchylenie standardowe) jest liczba, która próbuje opisać, jak rozłożone są dane. Wraz z miarą środka, miara zmienności może prawie całkowicie opisywać zbiór danych zawierający tylko dwie liczby.

Odchylenie standardowe

Prawdopodobnie odchylenie standardowe jest najczęstszą miarą zmienności danych na poziomie interwału i poza nim. Odchylenie standardowe można traktować jako „średnią odległość punktu danych od średniej”. Chociaż ten opis jest technicznie i matematycznie niepoprawny, jest to dobry sposób na myślenie o tym. Wzór na odchylenie standardowe można podzielić na następujące kroki:

1. Znajdź średnią danych.

2. Dla każdej liczby w zbiorze danych odejmij ją od średniej, a następnie podnieś ją do kwadratu.

3. Znajdź średnią każdego kwadratu różnicy.

4. Wyciągnij pierwiastek kwadratowy z liczby otrzymanej w kroku trzecim. To jest odchylenie standardowe.

Zwróć uwagę, jak w tych krokach faktycznie bierzemy średnią arytmetyczną jako jeden z kroków. Na przykład spójrz wstecz na zestaw danych temperatury. Znajdźmy odchylenie standardowe zbioru danych za pomocą Pythona:

```
import numpy

temps = [31, 32, 32, 31, 28, 29, 31, 38, 32, 31, 30, 29, 30, 31, 26]

mean = numpy.mean(temps) # == 30.73

squared_differences = []

# empty list o squared differences

for temperature in temps:

difference = temperature - mean

# how far is the point from the mean

squared_difference = difference**2

# square the difference

squared_differences.append(squared_difference)

# add it to our list

average_squared_difference = numpy.mean(squared_differences)

# This number is also called the "Variance"

standard_deviation = numpy.sqrt(average_squared_difference)

# We did it!

print standard_deviation # == 2.5157
```

Cały ten kod doprowadził nas do ustalenia, że odchylenie standardowe zestawu danych wynosi około 2,5, co oznacza, że „średnio” punkt danych jest oddalony o 2,5 stopnia od średniej temperatury wynoszącej około 31 stopni, co oznacza, że temperatura może prawdopodobnie spaść poniżej 29 stopni ponownie w najbliższej przyszłości. Powodem, dla którego chcemy „kwadratowej różnicy” między każdym punktem a średnią, a nie „rzeczywistą różnicą”, jest to, że podniesienie do kwadratu wartości faktycznie kładzie nacisk na wartości odstające - punkty danych, które są nienormalnie odległe.

Miary zmienności dają nam bardzo jasny obraz tego, jak rozproszone lub rozproszone są nasze dane. Jest to szczególnie ważne, gdy zajmujemy się zakresami danych i ich wahaniami (pomyśl o procentowym zwrocie z akcji). Duża różnica między danymi na tym a kolejnym poziomie polega na czymś, co nie jest oczywiste. Dane na poziomie interwału nie mają „naturalnego punktu startowego ani naturalnego zera”. Jednak przebywanie w temperaturze zerowej nie oznacza, że nie masz „temperatury”.

Poziom wskaźnika

Na koniec przyjrzymy się poziomowi wskaźnika. Po przejściu przez trzy różne poziomy z różnymi poziomami dozwolonych operacji matematycznych, poziom współczynnika okazuje się najsilniejszy z czterech. Nie tylko możemy zdefiniować kolejność i różnicę, ale także poziom proporcji pozwala nam mnożyć i dzielić. Może się wydawać, że nie jest to zbyt wielkie zamieszanie, ale zmienia prawie wszystko w sposobie, w jaki patrzymy na dane na tym poziomie.

Przykłady

Podczas gdy Fahrenheit i Celsjusz utknęli na poziomie interwału, skala temperatury Kelvina może pochwalić się naturalnym zerem. Pomiar zero Kelvin dosłownie oznacza brak ciepła. Jest to niearbitralne początkowe zero. W rzeczywistości możemy naukowo powiedzieć, że 200 kelwinów to dwa razy więcej ciepła niż 100 kelwinów. Pieniądze w banku są na poziomie ratio. Możesz „nie mieć pieniędzy w banku” i ma sens, że 200 000 dolarów to „dwa razy więcej niż” 100 000 dolarów.

Wiele osób może argumentować, że stopnie Celsjusza i Fahrenheita również mają punkt wyjścia (głównie dlatego, że możemy nawrócić się z Kelvina na jeden z nich). Prawdziwa różnica może wydawać się głupia, ale ponieważ przeliczenie na stopnie Celsjusza i Fahrenheita powoduje, że obliczenia przechodzą w ujemną wartość, nie definiuje ona wyraźnego i „naturalnego” zera.

Miary centrum

Średnia arytmetyczna nadal ma znaczenie na tym poziomie, podobnie jak nowy typ średniej, zwany średnią geometryczną. Miara ta na ogół nie jest tak często stosowana nawet na poziomie wskaźnika, ale warto o tym wspomnieć. Jest to pierwiastek kwadratowy iloczynu wszystkich wartości.

Na przykład w naszych danych dotyczących temperatury lodówki możemy obliczyć średnią geometryczną, jak pokazano tutaj:

```
import numpy

temps = [31, 32, 32, 31, 28, 29, 31, 38, 32, 31, 30, 29, 30, 31, 26]

num_items = len(temps)

product = 1.

for temperature in temps:
```

```
product *= temperature

geometric_mean = product**(1./num_items)

print geometric_mean # == 30.634
```

Zwróć uwagę, jak bardzo zbliża się do średniej arytmetycznej i mediany, jak obliczono wcześniej. Nie zawsze tak jest i będzie szczegółowo omówione w części poświęconej statystykom

Problemy z poziomem wskaźnika

Nawet przy całej tej dodanej funkcjonalności na tym poziomie, musimy ogólnie przyjąć bardzo duże założenie, które w rzeczywistości sprawia, że poziom współczynnika jest nieco restrykcyjny. Dane na poziomie wskaźnika są zwykle nieujemne. Już z tego powodu wielu analityków danych woli poziom interwału od poziomu współczynnika. Powodem tej restrykcyjnej właściwości jest to, że gdybyśmy dopuścili wartości ujemne, stosunek może nie zawsze mieć sens. Weźmy pod uwagę, że na przykładzie banku pozwoliliśmy, aby w naszych pieniądzach pojawił się dług. Gdybyśmy mieli saldo w wysokości 50 000 USD, następujący stosunek nie miałby żadnego sensu:

50 000$ / -50 000$ = -1

Dane są w oku patrzącego

Istnieje możliwość narzucenia struktury na dane. Na przykład, chociaż powiedziałem, że technicznie nie można użyć średniej dla danych od jednego do pięciu w skali porządkowej, wielu statystyków nie miałoby problemu z użyciem tej liczby jako deskryptora zbioru danych. Poziom interpretacji danych jest ogromnym założeniem, które należy przyjąć na początku każdej analizy. Jeśli patrzysz na dane, o których ogólnie myśli się na poziomie porządkowym, i stosujesz narzędzia, takie jak średnia arytmetyczna i odchylenie standardowe, to jest to coś, czego naukowcy zajmujący się danymi muszą być świadomi. Dzieje się tak głównie dlatego, że jeśli nadal będziesz uważać te założenia za ważne w swojej analizie, możesz napotkać problemy. Na przykład, jeśli przez pomyłkę zakładasz podzielność na poziomie porządkowym, narzucasz strukturę, w której struktura może nie istnieć.

Podsumowanie

Typ danych, z którymi pracujesz, to bardzo duża część nauki o danych. Musi poprzedzać większość twojej analizy, ponieważ rodzaj danych, które posiadasz, wpływa na typ analizy, który jest nawet możliwy! Za każdym razem, gdy masz do czynienia z nowym zbiorem danych, pierwsze trzy pytania, które powinieneś zadać, są następujące:

• Czy dane są zorganizowane lub niezorganizowane?

Na przykład, czy nasze dane istnieją w ładnej, czystej strukturze wierszy/kolumn?

• Czy każda kolumna jest ilościowa czy jakościowa?

Na przykład, czy wartości są liczbami, ciągami, czy też reprezentują ilości?

• Na jakim poziomie danych znajduje się każda kolumna?

Na przykład, czy wartości są na poziomie nominalnym, porządkowym, przedziałowym czy ilorazowym?

Odpowiedzi na te pytania nie tylko wpłyną na twoją znajomość danych na końcu, ale także będą dyktować kolejne etapy analizy. Będą dyktować rodzaje wykresów, których możesz użyć, i sposób ich interpretacji w przyszłych modelach danych. Czasami będziemy musieli przejść z jednego poziomu na drugi, aby zyskać większą perspektywę. W kolejnych częściach przyjrzymy się znacznie głębiej, jak

radzić sobie z danymi i badać je na różnych poziomach. Pod koniec tej książki będziemy w stanie nie tylko rozpoznawać dane na różnych poziomach, ale także będziemy wiedzieć, jak sobie z nimi radzić na tych poziomach.

Pięć kroków nauki o danych

Spędziliśmy dużo czasu przyglądając się wstępnym analizom danych, w tym opisując typy danych i sposoby podejścia do zestawów danych w zależności od ich typu. Tu skupimy się głównie na trzecim etapie eksploracji. Użyjemy pakietów pandas i matplotlib Pythona do eksploracji różnych zestawów danych.

Wprowadzenie do nauki o danych

Wiele osób pyta o największą różnicę między nauką o danych a analizą danych. Chociaż można argumentować, że nie ma między nimi żadnej różnicy, wielu będzie twierdziło, że są setki! Uważam, że niezależnie od tego, ile różnic jest między tymi dwoma terminami, największą jest to, że nauka o danych postępuje zgodnie z ustrukturyzowanym, krok po kroku procesem, który, gdy jest przestrzegany, zachowuje integralność wyników. Jak każde inne przedsięwzięcie naukowe, proces ten musi być przestrzegany, w przeciwnym razie analiza i wyniki są zagrożone analizą. Mówiąc prościej, przestrzeganie ścisłego procesu może znacznie ułatwić amatorskim naukowcom danych uzyskanie wyników szybciej niż w przypadku eksploracji danych bez wyraźnej wizji. Chociaż te kroki są lekcją przewodnią dla analityków amatorów, stanowią również podstawę dla wszystkich naukowców zajmujących się danymi, nawet tych na najwyższych szczeblach biznesowych i akademickich. Każdy analityk danych dostrzega wartość tych kroków i postępuje zgodnie z nimi w taki czy inny sposób.

Przegląd pięciu kroków

Pięć podstawowych kroków do wykonania analizy danych to:

1. Zadawanie ciekawego pytania

2. Uzyskiwanie danych

3. Eksploracja danych

4. Modelowanie danych

5. Komunikowanie i wizualizacja wyników

Najpierw spójrzmy na pięć kroków w odniesieniu do całościowego obrazu.

Zadaj ciekawe pytanie

To chyba mój ulubiony krok. Jako przedsiębiorca codziennie zadaję sobie (i innym) ciekawe pytania. Traktowałbym ten krok tak, jakbyś traktował sesję burzy mózgów. Zacznij spisywać pytania niezależnie od tego, czy uważasz, że dane, które pozwolą odpowiedzieć na te pytania, w ogóle istnieją. Powód tego jest dwojaki. Po pierwsze, nie chcesz zaczynać stronniczości nawet przed wyszukiwaniem danych. Po drugie, pozyskiwanie danych może wiązać się z wyszukiwaniem zarówno w miejscach publicznych, jak i prywatnych, a zatem może nie być zbyt proste. Możesz zadać pytanie i od razu powiedzieć sobie „Och, ale założę się, że nie ma danych, które mogłyby mi pomóc" i skreślić je z listy. Nie rób tego! Zostaw to na swojej liście.

Uzyskaj dane

Po wybraniu pytania, na którym chcesz się skoncentrować, nadszedł czas, aby przeszukać świat w poszukiwaniu danych, które mogą być w stanie odpowiedzieć na to pytanie. Jak wspomniano wcześniej, dane mogą pochodzić z różnych źródeł; więc ten krok może być bardzo kreatywny!

Przeglądaj dane

Kiedy już mamy dane, korzystamy z lekcji wyciągniętych z Części 2, Typy danych, i zaczynamy rozbijać typy danych, z którymi mamy do czynienia. To kluczowy krok w całym procesie. Po zakończeniu tego kroku analityk zazwyczaj spędza kilka godzin, ucząc się o domenie, używając kodu lub innych narzędzi do manipulowania danymi i ich eksploracji, i ma bardzo dobre wyczucie tego, co dane mogą próbować im przekazać.

Modeluj dane

Ten krok obejmuje wykorzystanie modeli statystycznych i uczenia maszynowego. Na tym etapie nie tylko dopasowujemy i wybieramy modele, ale także wszczepiamy matematyczne metryki walidacji w celu ilościowego określenia modeli i ich skuteczności.

Komunikuj się i wizualizuj wyniki

To prawdopodobnie najważniejszy krok. Choć może się to wydawać oczywiste i proste, umiejętność podsumowania wyników w przystępnej formie jest znacznie trudniejsza, niż się wydaje. Przyjrzymy się różnym przykładom przypadków, w których wyniki były słabo komunikowane i były bardzo dobrze wyświetlane. Skoncentrujemy się głównie na krokach 3, 4 i 5. Dlaczego pomijamy kroki 1 i 2? Chociaż pierwsze dwa kroki są niewątpliwie niezbędne dla procesu, generalnie poprzedzają systemy statystyczne i programistyczne. W dalszej części tej książki omówimy różne sposoby pozyskiwania danych, jednak w celu skupienia się na bardziej naukowych aspektach procesu, od razu zaczniemy od eksploracji.

Przeglądaj dane

Proces eksploracji danych nie jest zdefiniowany w prosty sposób. Wiąże się to z umiejętnością rozpoznawania różnych typów danych, przekształcania typów danych i używania kodu do systematycznego poprawiania jakości całego zestawu danych w celu przygotowania go do etapu modelowania. Aby jak najlepiej reprezentować i uczyć sztuki eksploracji, przedstawię kilka różnych zestawów danych i użyję pakietu pandas z pakietu Pythona do eksploracji danych. Po drodze natkniemy się na różne wskazówki i triki dotyczące obsługi danych. Istnieją trzy podstawowe pytania, które powinniśmy sobie zadać, gdy mamy do czynienia z nowym zbiorem danych, którego być może wcześniej nie widzieliśmy. Pamiętaj, że te pytania nie są początkiem i końcem nauki o danych; są to pewne wskazówki, których należy przestrzegać podczas eksploracji nowo uzyskanego zestawu danych.

Podstawowe pytania do eksploracji danych

Patrząc na nowy zbiór danych, niezależnie od tego, czy jest Ci znany, czy nie, ważne jest, aby użyć następujących pytań jako wskazówek do wstępnej analizy:

• Czy dane są uporządkowane, czy nie?

Sprawdzamy, czy dane są prezentowane w strukturze wiersz/kolumna. W większości dane będą prezentowane w sposób zorganizowany. U nas ponad 90% naszych przykładów zaczyna się od uporządkowanych danych. Niemniej jednak jest to najbardziej podstawowe pytanie, na które możemy odpowiedzieć, zanim zagłębimy się w naszą analizę. Ogólna zasada jest taka, że jeśli mamy niezorganizowane dane, chcemy je przekształcić w strukturę wiersz/kolumna. Na przykład we wcześniejszej części przyjrzeliśmy się sposobom przekształcania tekstu w strukturę wierszową/kolumnową, licząc liczbę słów/fraz.

• Co reprezentuje każdy rząd?

Gdy mamy już odpowiedź na to, jak dane są zorganizowane i teraz patrzymy na ładny zestaw danych oparty na wierszach/kolumnach, powinniśmy określić, co tak naprawdę reprezentuje każdy wiersz. Ten krok jest zwykle bardzo szybki i może pomóc znacznie szybciej spojrzeć na sprawy z innej perspektywy.

• Co reprezentuje każda kolumna?

Powinniśmy identyfikować każdą kolumnę według poziomu danych oraz tego, czy jest ilościowa/jakościowa, i tak dalej. Ta kategoryzacja może się zmieniać w miarę postępu naszej analizy, ale ważne jest, aby rozpocząć ten krok jak najwcześniej.

• Czy brakuje jakichkolwiek punktów danych?

Dane nie są doskonałe. Czasami może brakować danych z powodu błędu ludzkiego lub mechanicznego. Kiedy tak się dzieje, my, jako badacze danych, musimy podejmować decyzje, jak radzić sobie z tymi rozbieżnościami.

• Czy musimy wykonać jakieś przekształcenia na kolumnach?

W zależności od poziomu/typu danych, na którym znajduje się każda kolumna, może być konieczne wykonanie pewnych typów przekształceń. Na przykład, ogólnie rzecz biorąc, ze względu na modelowanie statystyczne i uczenie maszynowe chcielibyśmy, aby każda kolumna była numeryczna. Oczywiście użyjemy Pythona do wykonania wszelkich przekształceń.

Cały czas zadajemy sobie ogólne pytanie, co możemy wywnioskować ze wstępnych statystyk inferencyjnych? Chcemy być w stanie zrozumieć nasze dane nieco bardziej niż wtedy, gdy je znaleźliśmy.

Zestaw danych - Yelp

Pierwszym zbiorem danych, któremu się przyjrzymy, jest publiczny zbiór danych udostępniony przez witrynę z recenzjami restauracji, Yelp. Wszystkie informacje umożliwiające identyfikację osoby zostały usunięte. Przeczytajmy najpierw dane, jak pokazano tutaj:

```
import pandas as pd

yelp_raw_data = pd.read_csv("yelp.csv")

yelp_raw_data.head()
```

Krótkie podsumowanie tego, co robi poprzedni kod:

• Zaimportuj pakiet pandy i nazwij go jako pd.

• Czytaj w .csv z sieci; wywołanie to yelp_raw_data.

• Spójrz na początek danych (tylko kilka pierwszych wierszy).

	business_id	date	review_id	stars	text	type	user_id	cool	useful	funny
0	9yKzy9PApeiPPOUJEtnvkg	2011-01-26	fWKvX83p0-ka4JS3dc6E5A	5	My wife took me here on my birthday for breakf...	review	rLtl8ZkDX5vH5nAx9C3q5Q	2	5	0
1	ZRUwVLyzEJq7VAhDhYow	2011-07-27	IjZ33sJrzXqU-0X6U9NwyA	5	I have no idea why some people give bad review...	review	Ga7KyELDd3Yb1V6aivbIuQ	0	0	0
2	6oRAC4uyJCsJl1X0WZpVSA	2012-06-14	IESLBzqUCLdSzSqm0eCSxQ	4	love the gyro plate. Rice is so good and I als...	review	0hT2KtfLiobPvh6cDC8JQg	0	1	0
3	_1QQZuf4zZOyFCvXc0o6Vg	2010-05-27	G-WvGaISbqqaMHlNnByodA	5	Rosie, Dakota, and I LOVE Chaparral Dog Park!t...	review	uZetl9T0NcROGOyFfughhg	1	2	0
4	6ozycU1RpktNG2-1BroVtw	2012-01-05	1uJFq2r5QfJG_6ExMRCaGw	5	General Manager Scott Petello is a good egg!!t...	review	vYmM4KTsC8ZfQBg-j5MWkw	0	0	0

Czy dane są uporządkowane, czy nie?

• Ponieważ mamy ładną strukturę wierszy/kolumn, możemy wywnioskować, że te dane wydają się dość uporządkowane.

Co reprezentuje każdy wiersz?

• Wydaje się dość oczywiste, że każdy wiersz reprezentuje użytkownika oceniającego firmę. Następną rzeczą, którą powinniśmy zrobić, jest zbadanie każdego wiersza i oznaczenie go typem danych, które zawiera. W tym momencie możemy również użyć Pythona, aby dowiedzieć się, jak duży jest nasz zbiór danych. Aby to sprawdzić, możemy użyć jakości kształtu Dataframe, jak pokazano:

yelp_raw_data.shape

(10000,10)

• Mówi nam, że ten zbiór danych ma 10000 wierszy i 10 kolumn. Innym sposobem na powiedzenie tego jest to, że ten zbiór danych zawiera 10 000 obserwacji i 10 cech.

Co reprezentuje każda kolumna?

Zauważ, że mamy 10 kolumn:

• business_id: jest to prawdopodobnie unikalny identyfikator firmy, której dotyczy opinia. Byłoby to na poziomie nominalnym, ponieważ nie ma naturalnego porządku tego identyfikatora.

• data: prawdopodobnie jest to data opublikowania recenzji. Zauważ, że wydaje się, że dotyczy tylko dnia, miesiąca i roku. Chociaż czas jest zwykle uważany za ciągły, ta kolumna prawdopodobnie zostałaby uznana za dyskretną i na poziomie porządkowym ze względu na naturalny porządek dat.

• review_id: Jest to prawdopodobnie unikalny identyfikator recenzji, którą reprezentuje każdy post. Byłoby to na poziomie nominalnym, ponieważ znowu nie ma naturalnego porządku tego identyfikatora.

• stars: Po szybkim spojrzeniu (nie martw się, wkrótce przeprowadzimy dalszą analizę), widzimy, że jest to uporządkowana kolumna, która przedstawia ocenę końcową restauracji przez recenzenta. To jest uporządkowane i jakościowe; więc to jest na poziomie porządkowym.

• text: jest to prawdopodobnie nieprzetworzony tekst, który napisał każdy recenzent. Jak w przypadku większości tekstów, umieszczamy to na poziomie nominalnym.

• type: W pierwszych pięciu kolumnach widzimy tylko słowo przegląd. Może to być kolumna, która wskazuje, że każdy wiersz jest recenzją, co oznacza, że może istnieć inny typ wiersza niż recenzja. Przyjrzymy się temu później. Umieszczamy to na poziomie nominalnym.

• user_id: jest to prawdopodobnie unikalny identyfikator użytkownika, który pisze recenzję. Podobnie jak w przypadku innych unikalnych identyfikatorów, umieszczamy te dane na poziomie nominalnym.

Zauważ, że po przejrzeniu wszystkich kolumn i stwierdzeniu, że wszystkie dane są albo na poziomie porządkowym, albo na poziomie nominalnym, musimy spojrzeć na następujące rzeczy. Nie jest to rzadkie, ale warto o tym wspomnieć.

Czy brakuje jakichś punktów danych?

• Wykonaj operację isnull. Na przykład, jeśli twoja ramka danych nazywa się awesome_dataframe, wypróbuj polecenie python awesome_dataframe. isnull().sum() co pokaże liczbę braków danych w każdej kolumnie.

Czy musimy wykonać jakieś przekształcenia na kolumnach?

• W tym momencie szukamy kilku rzeczy. Na przykład, czy będziemy musieli zmienić skalę niektórych danych ilościowych, czy też musimy utworzyć zmienne fikcyjne dla zmiennych jakościowych? Ponieważ ten zbiór danych zawiera tylko kolumny jakościowe, możemy skupić się tylko na przekształceniach w skali porządkowej i nominalnej.

Zanim zaczniemy, przejrzyjmy krótką terminologię dotyczącą pand, modułu eksploracji danych Pythona.

Ramki danych

Kiedy czytamy w zbiorze danych, Pandas tworzy niestandardowy obiekt o nazwie Dataframe. Pomyśl o tym jako o wersji arkusza kalkulacyjnego w Pythonie (ale o wiele lepiej). W tym przypadku zmienna yelp_raw_data jest ramką danych. Aby sprawdzić, czy tak jest w Pythonie, wpisz następujący kod:

typ(yelp_raw_data)

pandas.core.frame.DataFrame

Ramki danych mają charakter dwuwymiarowy, co oznacza, że są zorganizowane w strukturę wierszy/kolumn, tak jak arkusz kalkulacyjny. Główną zaletą korzystania z Dataframes w porównaniu z, powiedzmy, oprogramowaniem do arkuszy kalkulacyjnych, jest to, że Dataframe może obsługiwać znacznie większe dane niż większość popularnych programów do obsługi arkuszy kalkulacyjnych. Jeśli znasz język R, możesz rozpoznać słowo Dataframe. To dlatego, że nazwa została faktycznie zapożyczona z języka! Ponieważ większość danych, którymi będziemy się zajmować jest zorganizowana, Dataframes są prawdopodobnie najczęściej używanym obiektem w pandach, ustępując tylko obiektowi Series.

Series

Obiekt Series to po prostu Dataframe, ale tylko z jednym wymiarem. Zasadniczo jest to lista punktów danych. Każda kolumna Dataframe jest uważana za obiekt serii. Sprawdźmy to. Pierwszą rzeczą, którą musimy zrobić, to pobrać pojedynczą kolumnę z naszego Dataframe; zazwyczaj używamy tak zwanej notacji nawiasowej. Oto przykład:

yelp_raw_data['business_id'] # pobiera pojedynczą kolumnę Dataframe

Wymienimy kilka pierwszych i kilka ostatnich wierszy:

0 9yKzy9PApeiPPOUJetnvkg

1 ZRJwVLyzEJq1VAihDhYiow

2 6oRAC4uyJCsJl1X0WZpVSA

3 _1QQZuf4zZOyFCvXc0o6Vg

4 6ozycU1RpktNG2-1BroVtw

5 -yxfBYGB6SEqszmxJxd97A

6 zp713qNhx8d9KCJJnrw1xA

Użyjmy funkcji type, aby sprawdzić, czy ta kolumna jest serią:

type(yelp_raw_data['business_id'])

pandas.core.series.Series

Wskazówki dotyczące eksploracji danych jakościowych

Korzystając z tych dwóch obiektów Pandy, zacznijmy przeprowadzać wstępną eksplorację danych. W przypadku danych jakościowych przyjrzymy się konkretnie poziomom nominalnym i porządkowym.

Kolumny poziomu nominalnego

Ponieważ jesteśmy na poziomie nominalnym, przypomnijmy sobie, że na tym poziomie dane są jakościowe i są opisane czysto z nazwy. W tym zbiorze danych odnosi się to do business_id, review_ id, tekstu, typu i user_id. Użyjmy Pand, aby zanurkować nieco głębiej, jak pokazano tutaj:

yelp_raw_data['business_id'].describe()

count 10000

unique 4174

top JokKtdXU7zXHcr20Lrk29A

freq 37

Funkcja opisu da nam kilka szybkich statystyk na temat kolumny, której nazwę wpisujemy w cudzysłów. Zwróć uwagę, jak Pandas automatycznie rozpoznała, że business_id to kolumna jakościowa i podała nam statystyki, które mają sens. Gdy opis zostanie wywołany w kolumnie jakościowej, zawsze otrzymamy następujące cztery pozycje:

• count: Ile wartości jest wpisanych

• unikalny: Ile unikalnych wartości jest wypełnionych

• u góry: nazwa najczęściej występującego elementu w zbiorze danych

• freq: jak często w zbiorze danych pojawia się najczęstszy element

Na poziomie nominalnym zwykle szukamy kilku rzeczy, które sygnalizowałyby transformację:

• Czy mamy rozsądną liczbę (zwykle poniżej 20) unikalnych pozycji?

• Czy ta kolumna jest wolna od tekstu?

• Czy ta kolumna jest całkowicie unikalna we wszystkich wierszach?

Tak więc dla kolumny business_id mamy 10000. Nie daj się jednak zwieść! Nie oznacza to, że recenzujemy tu 10 000 firm. Oznacza to po prostu, że z 10 000 wierszy opinii kolumna business_id jest wypełniana 10 000 razy. Kolejny kwalifikator, unikalny, informuje nas, że w tym zbiorze danych sprawdzanych jest 4174 unikalnych firm. Najczęściej recenzowaną firmą jest biznes JokKtdXU7zXHcr20Lrk29A, który był recenzowany 37 razy.

yelp_raw_data['review_id'].describe()

count 10000

unique 10000

top eTa5KD-LTgQv6UT1Zmijmw

freq 1

Liczymy 10000, a unikatowe 10000. Zastanów się przez chwilę, czy to ma sens? Zastanów się, co reprezentuje każdy wiersz i co reprezentuje ta kolumna. (tu wstaw piosenkę o niebezpieczeństwie)

Oczywiście, że tak! Każdy wiersz tego zbioru danych ma reprezentować jedną, niepowtarzalną recenzję firmy, a ta kolumna ma służyć jako unikalny identyfikator recenzji; więc sensowne jest, aby kolumna review_id zawierała 10000 unikalnych elementów. Dlaczego więc eTa5KD-LTgQv6UT1Zmijmw jest najczęstszą recenzją? To tylko losowy wybór z 10 000 i nic nie znaczy.

yelp_raw_data['text'].describe()

count 10000

unique 9998

top This review is for the chain in general. The l &helip;

freq 2

Ta kolumna, która reprezentuje rzeczywisty tekst, który napisali ludzie, jest interesująca. Wyobrażamy sobie, że powinno to być również podobne do review_id, ponieważ cały tekst powinien być unikalny, ponieważ byłoby dziwne, gdyby dwie osoby napisały dokładnie to samo; ale mamy dwie recenzje z dokładnie tym samym tekstem! Poświęćmy chwilę, aby dowiedzieć się więcej o filtrowanie ramek danych, aby dokładniej to zbadać.

Filtrowanie w Pandas

Porozmawiajmy trochę o tym, jak działa filtrowanie. Filtrowanie wierszy na podstawie określonych kryteriów w Pandas jest dość łatwe. W Dataframe, jeśli chcemy odfiltrować wiersze na podstawie pewnych kryteriów wyszukiwania, musimy przejść wiersz po wierszu i sprawdzić, czy wiersz spełnia ten konkretny warunek. Pandas radzi sobie z tym, przekazując serię Prawd i Fałszów (Boolean). Dosłownie przekazujemy do Dataframe listę danych Prawda i Fałsz, które oznaczają:

• Prawda: ten wiersz spełnia warunek

• Fałsz: ten wiersz nie spełnia warunku

Więc najpierw ustalmy warunki. W kolejnych wierszach kodu chwycę tekst, który występuje dwukrotnie:

duplicate_text = yelp_raw_data['text'].describe()['top']

Oto fragment tekstu:

„Ta recenzja jest ogólnie dla sieci. Lokalizacja, do której się udaliśmy, jest nowa, więc nie ma jej jeszcze na Yelp. Gdy już będzie, umieszczę tam również tę recenzję …”.

Od razu możemy się domyślać, że może to być jedna osoba, która poszła zrecenzować dwie firmy należące do tej samej sieci i napisała dokładnie tę samą recenzję. Jednak to tylko przypuszczenie.

Zmienna duplikat_tekstu jest typu string

Teraz, gdy mamy ten tekst, użyjmy magii, aby stworzyć tę serię prawdy i fałszu:

text_is_the_duplicate = yelp_raw_data['text'] == duplicate_text

Od razu możesz być zdezorientowany. To, co tutaj zrobiliśmy, to pobranie kolumny tekstowej ramki danych i porównanie jej z ciągiem tekstowym duplikat_tekst. Jest to dziwne, ponieważ wydaje się, że porównujemy listę 10 000 elementów z pojedynczym ciągiem. Oczywiście odpowiedź powinna być fałszywa, prawda? Seria Pand ma bardzo interesującą cechę polegającą na tym, że jeśli porównasz Serie do obiektu, zwróci kolejną Serię Boole'ów o tej samej długości, gdzie każda prawda i fałsz jest odpowiedzią na pytanie, czy ten element jest taki sam jak element, do którego go porównujesz? Bardzo przydatne!

type(text_is_the_duplicate) # it is a Series of Trues and Falses

text_is_the_duplicate.head() # shows a few Falses out of the Series

W Pythonie możemy dodawać i odejmować prawdę i fałsz, tak jakby były odpowiednio 1 i 0. Na przykład True + False – True + False + True == 1. Możemy więc zweryfikować, czy ta seria jest poprawna, dodając wszystkie wartości. Ponieważ tylko dwa z tych wierszy powinny zawierać zduplikowany tekst, suma serii powinna wynosić tylko 2, co jest prawdą! Jest to pokazane w następujący sposób:

sum(text_is_the_duplicate) # == 2

Teraz, gdy mamy już naszą serię wartości logicznych, możemy przekazać ją bezpośrednio do naszej ramki danych, używając notacji nawiasów i uzyskać nasze przefiltrowane wiersze, jak pokazano na ilustracji:

filtered_dataframe = yelp_raw_data[text_is_the_duplicate]

the filtered Dataframe

filtered_dataframe

Wygląda na to, że nasze podejrzenia były słuszne i jedna osoba tego samego dnia przekazała dokładnie tę samą recenzję dwóm różnym identyfikatorom firmy, prawdopodobnie należącym do tej samej sieci. Przejdźmy dalej do reszty naszych kolumn:

yelp_raw_data['type'].describe()

count 10000

unique 1

top review

freq 10000

Pamiętasz tę kolumnę? Okazuje się, że wszystkie są dokładnie tego samego typu, a mianowicie przegląd.

yelp_raw_data['user_id'].describe()

count 10000

unique 6403

top fczQCSmaWF78toLEmb0Zsw

freq 38

Podobnie jak w kolumnie business_id, wszystkie 10000 wartości są wypełniane z 6403 unikalnymi użytkownikami i jednym użytkownikiem sprawdzającym 38 razy! W tym przykładzie nie będziemy musieli wykonywać żadnych przekształceń

Kolumny poziomu porządkowego

Jeśli chodzi o kolumny porządkowe, patrzymy na datę i gwiazdy. Dla każdej z tych kolumn przyjrzyjmy się, co zwraca metoda opisu:

yelp_raw_data['stars'].describe()

count 10000.000000

mean 3.777500

std 1.214636

min 1.000000

25% 3.000000

50% 4.000000

75% 5.000000

max 5.000000

Och! Mimo że ta kolumna jest porządkowa, metoda opisu zwracała statystyki, których moglibyśmy oczekiwać dla kolumny ilościowej. Dzieje się tak, ponieważ oprogramowanie widziało wiele liczb i po prostu założyło, że chcemy mieć statystyki, takie jak średnia lub minimalna i maksymalna. To nie jest problem. Użyjmy metody o nazwie value_counts, aby zobaczyć liczbę

yelp_raw_data['stars'].value_counts()

4 3526

5 3337

3 1461

2 927

1 749

Metoda value_counts zwróci rozkład wartości dla dowolnej kolumny. W tym przypadku widzimy, że ocena w postaci gwiazdek 4 jest najbardziej powszechna, z 3526 wartościami, tuż za nią znajduje się ocena 5. Możemy również wykreślić te dane, aby uzyskać ładny obraz. Najpierw posortujmy według ocen, a następnie użyjmy gotowej metody kreślenia, aby utworzyć wykres słupkowy.

```
dates = yelp_raw_data['stars'].value_counts()

dates.sort()

dates.plot(kind='bar')
```

<img src="images/bar.PNG">

Ludzie zdecydowanie częściej dają dobre oceny w gwiazdkach niż złe! Możemy postępować zgodnie z tą procedurą dla kolumny daty. Zostawię cię, abyś sam spróbował. Na razie spójrzmy na nowy zbiór danych.

Zestaw danych 2 - tytaniczny

Zbiór danych Titanica zawiera próbkę ludzi, którzy byli na Titanicu, kiedy uderzył w górę lodową w 1912 roku. Przejdźmy dalej i zaimportujmy go, jak pokazano tutaj:

```
titanic = pd.read_csv('short_titanic.csv')

titanic.head()
```

	Survived	Pclass	Name	Sex	Age
0	0	3	Braund, Mr. Owen Harris	male	22
1	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38
2	1	3	Heikkinen, Miss. Laina	female	26
3	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35
4	0	3	Allen, Mr. William Henry	male	35

Ta ramka danych ma zwykle więcej kolumn; jednak w naszym przykładzie skupimy się tylko na podanych kolumnach. Te dane są zdecydowanie zorganizowane w strukturę wierszy/kolumn, podobnie jak większość danych w arkuszach kalkulacyjnych. Rzućmy okiem na jego rozmiar, jak pokazano tutaj:

```
titanic.shape

# (891, 5)
```

Mamy więc 891 wierszy i 5 kolumn. Każdy wiersz wydaje się reprezentować jednego pasażera na statku, a jeśli chodzi o kolumny, poniższa lista mówi nam, co one oznaczają:

• Przeżył: Jest to zmienna binarna, która wskazuje, czy pasażer przeżył wypadek (1, jeśli przeżył, 0, jeśli zginął). Byłoby to prawdopodobnie na poziomie nominalnym, ponieważ są tylko dwie opcje.

• Pclass: jest to klasa, w której podróżował pasażer (3 dla trzeciej klasy itd.). To jest na poziomie porządkowym.

• Imię i nazwisko: to imię i nazwisko pasażera, i to zdecydowanie na poziomie nominalnym.

• Płeć: Wskazuje płeć pasażera. Jest na poziomie nominalnym.

• Wiek: Ten jest trochę trudny. Zapewne można umieścić wiek albo na poziomie jakościowym, albo ilościowym, jednak uważam, że wiek należy do stanu ilościowego, a więc do poziomu relacyjnego.

Jeśli chodzi o przekształcenia, zwykle chcemy, aby wszystkie kolumny były numeryczne, niezależnie od ich stanu jakościowego. Oznacza to, że Imię i Płeć będą musiały zostać jakoś przekonwertowane na kolumny liczbowe. W przypadku Płeć możemy zmienić kolumnę tak, aby zawierała 1, jeśli pasażer była kobietą i 0, jeśli był mężczyzną. Użyjmy Pand, aby dokonać zmiany. Będziemy musieli zaimportować inny moduł Pythona, zwany numpy lub Python numeryczny, jak pokazano na ilustracji:

```
import numpy as np

titanic['Sex'] = np.where(titanic['Sex']=='female', 1, 0)
```

Metoda np.where obejmuje trzy rzeczy:

• Lista wartości logicznych (prawda lub fałsz)

• Nowa wartość

• Wartość zapasowa

Metoda zastąpi wszystkie true pierwszą wartością (w tym przypadku 1) i false drugą wartością (w tym przypadku 0), pozostawiając nam nową kolumnę liczbową, która reprezentuje to samo, co oryginalna kolumna Sex.

```
titanic['Sex']

# 0 0

# 1 1

# 2 1

# 3 1

# 4 0

# 5 0

# 6 0

# 7 0
```

Użyjmy skrótu i opiszmy wszystkie kolumny jednocześnie, jak pokazano:

```
titanic.describe()
```

	Survived	Pclass	Sex	Age
count	891.000000	891.000000	891.000000	714.000000
mean	0.383838	2.308642	0.352413	29.699118
std	0.486592	0.836071	0.477990	14.526497
min	0.000000	1.000000	0.000000	0.420000
25%	0.000000	2.000000	0.000000	20.125000
50%	0.000000	3.000000	0.000000	28.000000
75%	1.000000	3.000000	1.000000	38.000000
max	1.000000	3.000000	1.000000	80.000000

Zwróć uwagę, jak nasze kolumny jakościowe są traktowane jako ilościowe; jednak szukam czegoś nieistotnego dla typu danych. Zwróć uwagę na wiersz liczenia: Przeżyli, Pklasa i Płeć mają 891 wartości (liczba wierszy), ale Wiek ma tylko 714 wartości. Niektórych brakuje! Aby dwukrotnie zweryfikować, użyjmy funkcji Pandy, zwanych isnull i sum, jak pokazano:

titanic.isnull().sum()

Survived 0

Pclass 0

Name 0

Sex 0

Age 177

To pokaże nam liczbę braków danych w każdej kolumnie. Tak więc Wiek jest jedyną kolumną z brakującymi wartościami, z którą należy się uporać. Kiedy masz do czynienia z brakującymi wartościami, zwykle masz dwie opcje:

• Upuść wiersz z brakującą wartością

• Spróbuj go wypełnić

Upuszczenie rzędu to łatwy wybór; jednak ryzykujesz utratę cennych danych! Na przykład w tym przypadku mamy 177 brakujących wartości wieku (891-714), co stanowi prawie 20% danych. Aby uzupełnić dane, możemy albo wrócić do podręczników historii, znaleźć każdą osobę po kolei i wpisać jej wiek, albo możemy wpisać wiek wartością zastępczą. Uzupełnijmy każdą brakującą wartość w kolumnie Wiek ogólnym średnim wiekiem osób w zbiorze danych. W tym celu zastosujemy dwie nowe metody, zwane mean i fillna. Używamy isnull, aby powiedzieć nam, które wartości są null, oraz funkcji średniej, aby uzyskać średnią wartość kolumny Age. fillna to metoda Pandy, która zastępuje wartości null podaną wartością.

print sum(titanic['Age'].isnull()) # == 177 missing values

average_age = titanic['Age'].mean() # get the average age

titanic['Age'].fillna(average_age, inplace = True) #use the fillna

method to remove null values

print sum(titanic['Age'].isnull()) # == 0 missing values

Skończyliśmy! Zamieniliśmy każdą wartość na 26,69, czyli średni wiek w zbiorze danych.

titanic.isnull().sum()

Survived 0

Pclass 0

Name 0

Sex 0

Age 0

Świetny! Niczego nie brakuje i nie musieliśmy usuwać żadnych rzędów.

titanic.head()

	Survived	Pclass	Name	Sex	Age
0	0	3	Braund, Mr. Owen Harris	0	22
1	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	38
2	1	3	Heikkinen, Miss. Laina	1	26
3	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1	35
4	0	3	Allen, Mr. William Henry	0	35

W tym momencie możemy zacząć nieco bardziej skomplikować nasze pytania. Na przykład, jaki jest średni wiek kobiety lub mężczyzny? Aby odpowiedzieć na to pytanie, możemy filtrować według płci i wziąć średni wiek. Pandas ma w tym celu wbudowaną funkcję o nazwie groupby, jak pokazano poniżej:

titanic.groupby('Sex')['Age'].mean()

Oznacza to pogrupowanie danych według kolumny Płeć, a następnie podanie średniej wieku dla każdej grupy. Daje nam to następujący wynik:

Sex

0 30.505824

1 28.216730

Zadamy więcej tych trudnych i złożonych pytań i będziemy mogli na nie odpowiedzieć za pomocą Pythona i statystyk.

Podsumowanie

Chociaż jest to tylko nasze pierwsze spojrzenie na eksplorację danych, nie martw się — to zdecydowanie nie ostatni raz, kiedy wykonamy te kroki w celu nauki i eksploracji danych. Od teraz, za każdym razem, gdy przyjrzymy się nowej porcji danych, wykorzystamy nasze etapy eksploracji, aby przekształcić, podzielić i ujednolicić nasze dane. Kroki opisane w tym rozdziale, choć są jedynie

wskazówkami, stanowią standardową praktykę, którą każdy analityk danych może stosować w swojej pracy. Kroki można również zastosować do dowolnego zestawu danych, który wymaga analizy. Szybko zbliżamy się do części książki, która zajmuje się modelami statystycznymi, probabilistycznymi i uczeniem maszynowym. Zanim naprawdę będziemy mogli wskoczyć do tych modeli, musimy przyjrzeć się podstawom matematyki. W następnej części przyjrzymy się niektórym matematyce niezbędnej do wykonania niektórych bardziej skomplikowanych operacji w modelowaniu, ale nie martw się - matematyka wymagana do tego proces jest minimalny i przejdziemy go krok po kroku.

Matematyka podstawowa

Czas zacząć przyglądać się podstawowym zasadom matematycznym, które są przydatne w pracy z nauką o danych. Słowo matematyka budzi strach w wielu sercach, ale staram się, aby było to tak przyjemne, jak to tylko możliwe. W tym rozdziale omówimy podstawy następujących tematów:

• Podstawowe symbole/terminologia

• Logarytmy/wykładniki

• Teoria mnogości

• Rachunek

• Algebra macierzowa (liniowa)

Zajmiemy się także innymi dziedzinami matematyki. Co więcej, zobaczymy, jak zastosować każdy z nich do różnych aspektów nauki o danych, a także innych przedsięwzięć naukowych. Przypomnijmy, że w poprzedniej części zidentyfikowaliśmy matematykę jako jeden z trzech kluczowych elementów nauki o danych. Tu przedstawimy koncepcje, które staną się ważne w dalszej części - patrząc na modele probabilistyczne i statystyczne - a także przyjrzymy się pojęciom, które będą przydatne w tej części. Niezależnie od tego wszystkie koncepcje zawarte w tym rozdziale należy traktować jako podstawy w dążeniu do zostania naukowcem danych.

Matematyka jako dyscyplina

Matematyka jako nauka jest jedną z najstarszych znanych form logicznego myślenia ludzkości. Od starożytnej Mezopotamii i prawdopodobnie wcześniej (3000 p.n.e.) ludzie polegali na arytmetyce i trudniejszych formach matematyki, aby odpowiedzieć na największe życiowe pytania.Dzisiaj polegamy na matematyce w większości aspektów naszego codziennego życia; tak, wiem, że to brzmi banalnie, ale mam to na myśli. Niezależnie od tego, czy podlewasz rośliny, czy karmisz psa, Twój wewnętrzny silnik matematyczny nieustannie się kręci – oblicza, ile wody roślina miała dziennie w ciągu ostatniego tygodnia i przewiduje, kiedy następnym razem Twój pies będzie głodny, biorąc pod uwagę, że jedzą teraz. Niezależnie od tego, czy świadomie używasz zasad matematyki, czy nie, koncepcje żyją głęboko w mózgach każdego. Moim zadaniem jako nauczyciela matematyki jest uświadomienie ci tego.

Podstawowe symbole i terminologia

Najpierw przyjrzyjmy się najbardziej podstawowym symbolom używanym w procesie matematycznym, a także bardziej subtelnym zapisom używanym przez naukowców zajmujących się danymi.

Wektory i macierze

Wektor jest zdefiniowany jako obiekt o wielkości i kierunku. Ta definicja jest jednak nieco skomplikowana dla naszego użytku. Dla naszego celu wektor jest po prostu jednowymiarową tablicą reprezentującą szereg liczb. Innymi słowy, wektor to lista liczb. Jest to zwykle reprezentowane za pomocą strzałki lub pogrubionej czcionki, jak pokazano:

$$\vec{x} \quad or \quad x$$

Wektory są podzielone na komponenty, które są indywidualnymi członkami wektora. Używamy notacji indeksowych do oznaczenia elementu, do którego się odnosimy, jak pokazano na ilustracji:

$$\text{If} \quad \vec{x} = \begin{pmatrix} 3 \\ 6 \\ 8 \end{pmatrix} \quad \text{then } x_1 = 3$$

W matematyce na ogół odnosimy się do pierwszego elementu jako do indeksu 1, w przeciwieństwie do informatyki, gdzie zazwyczaj odnosimy się do pierwszego elementu jako indeksu 0. Ważne jest, aby pamiętać, jakiego systemu indeksów używasz. W Pythonie możemy reprezentować tablice na wiele sposobów. Moglibyśmy po prostu użyć listy Pythona do reprezentowania poprzedniej tablicy:

```
x = [3,6,8]
```

Jednak lepiej jest użyć typu tablicy numpy do reprezentowania tablic, jak pokazano, ponieważ daje nam to znacznie większą użyteczność podczas wykonywania operacji wektorowych:

```
import numpy as np.

x = np.array([3, 6, 8])
```

Niezależnie od reprezentacji Pythona wektory dają nam prosty sposób przechowywania wielu wymiarów pojedynczego punktu danych/obserwacji. Weź pod uwagę, że mierzymy średnią ocenę satysfakcji (0-100) pracowników z trzech działów firmy jako 57 dla HR, 89 dla inżynierii i 94 dla kierownictwa. Możemy przedstawić to jako wektor za pomocą następującego wzoru:

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 57 \\ 89 \\ 94 \end{pmatrix}$$

Ten wektor zawiera trzy różne bity informacji o naszych danych. To idealne zastosowanie wektora w nauce o danych. Możesz również myśleć o wektorze jako teoretycznej generalizacji obiektu serii Pandy. Tak więc naturalnie potrzebujemy czegoś, co będzie reprezentować Dataframe. Możemy rozszerzyć nasze pojęcie tablicy, aby wyjść poza jeden wymiar i reprezentować dane w wielu wymiarach. Macierz to dwuwymiarowa reprezentacja tablic liczb. Macierze (liczba mnoga) mają dwie główne cechy, o których musimy pamiętać. Wymiar macierzy, oznaczony jako n x m (n na m), mówi nam, że macierz ma n wierszy i m kolumn. Macierze są zazwyczaj oznaczane wielką, pogrubioną literą, taką jak X. Rozważmy następujący przykład:

$$\begin{pmatrix} 3 & 4 \\ 8 & 55 \\ 5 & 9 \end{pmatrix}$$

Jest to macierz 3 x 2 (3 na 2), ponieważ ma trzy wiersze i dwie kolumny.

Jeśli macierz ma taką samą liczbę wierszy i kolumn, nazywa się ją macierzą kwadratową. Matryca jest naszym uogólnieniem Pandas Dataframe. Jest to prawdopodobnie jeden z najważniejszych obiektów matematycznych w naszym zestawie narzędzi. Służy do przechowywania uporządkowanych informacji, w naszym przypadku danych. Wracając do naszego poprzedniego przykładu, załóżmy, że mamy trzy biura w różnych lokalizacjach, każde z tymi samymi trzema działami: HR, inżynierii i zarządzania.

Moglibyśmy stworzyć trzy różne wektory, z których każdy zawiera wyniki zadowolenia z innego biura, jak pokazano:

$$x = \begin{pmatrix} 57 \\ 89 \\ 94 \end{pmatrix}, y = \begin{pmatrix} 67 \\ 87 \\ 84 \end{pmatrix}, z = \begin{pmatrix} 65 \\ 98 \\ 60 \end{pmatrix}$$

Jest to jednak nie tylko kłopotliwe, ale także nieskalowalne. A jeśli masz 100 różnych biur? Wtedy musielibyśmy mieć 100 różnych jednowymiarowych tablic do przechowywania tych informacji. W tym miejscu matryca łagodzi ten problem. Stwórzmy macierz, w której każdy wiersz reprezentuje inny dział, a każda kolumna reprezentuje inne biuro, jak pokazano:

	Office 1	Office 2	Office 3
HR	57	67	65
Engineering	89	87	98
Management	94	84	60

To jest o wiele bardziej naturalne. Teraz zdejmijmy etykiety i zostajemy z matrycą!

$$X = \begin{pmatrix} 57 & 67 & 65 \\ 89 & 87 & 98 \\ 94 & 84 & 60 \end{pmatrix}$$

Szybkie ćwiczenia

1. Jeśli dodamy czwarte biuro, czy będziemy potrzebować nowego wiersza lub kolumny?

2. Jaki byłby wymiar macierzy po dodaniu czwartego biura?

3. Jeśli wyeliminujemy dział zarządzania z pierwotnej macierzy X, jaki byłby wymiar nowej macierzy?

4. Jaki jest ogólny wzór na poznanie liczby elementów w macierzy?

Odpowiedzi

1. Kolumna.

2. 3 x 4.

3. 2 x 3.

4. n x m (n to liczba wierszy, a m to liczba kolumn).

Symbole arytmetyczne

W tej sekcji omówimy niektóre symbole związane z podstawową arytmetyką, które pojawiają się w większości, jeśli nie we wszystkich, samouczkach i książkach dotyczących nauki o danych.

Sumowanie

Wielki symbol sigma Σ jest uniwersalnym symbolem dodawania. To, co znajduje się na prawo od symbolu sigma, jest zwykle czymś iterowalnym, co oznacza, że możemy przechodzić przez to jeden po drugim (na przykład wektor). Na przykład stwórzmy reprezentację wektora:

X = [1, 2, 3, 4, 5]

Aby obliczyć sumę zawartości, możemy skorzystać z następującego wzoru:

$\Sigma X_i = 15$

W Pythonie możemy użyć następującej formuły:

sum(x) # == 15

Na przykład wzór na obliczanie średniej szeregu liczb jest dość powszechny. Jeśli mamy wektor (x) o długości n, to średnią tego wektora można obliczyć w następujący sposób:

mean = $1/n \Sigma X_i$

Oznacza to, że dodamy każdy element x, oznaczony przez x_i , a następnie pomnożymy sumę przez 1/n, inaczej zwana dzieleniem przez n, długość wektora.

Proporcjonalny

Mały symbol alfa α reprezentuje wartości, które są do siebie proporcjonalne. Oznacza to, że wraz ze zmianą jednej wartości zmienia się druga. Kierunek, w którym przesuwają się wartości, zależy od tego, jak są one proporcjonalne. Wartości mogą się zmieniać bezpośrednio lub pośrednio. Jeśli wartości zmieniają się bezpośrednio, obie poruszają się w tym samym kierunku (w miarę wzrostu jednego, drugie też). Jeśli różnią się pośrednio, poruszają się w przeciwnych kierunkach (jeśli jeden idzie w dół, drugi idzie w górę). Rozważ następujące przykłady:

• Sprzedaż firmy zależy bezpośrednio od liczby klientów. Można to zapisać jako Sales α Customer.

• Ceny gazu zmieniają się (zwykle) pośrednio w zależności od dostępności ropy, co oznacza, że w miarę zmniejszania się dostępności ropy (jest jej coraz mniej), ceny gazu będą rosły. Można to oznaczyć jako Gas α Oil Availability.

Później zobaczymy bardzo ważną formułę zwaną formułą Bayesa, która zawiera symbol wariacji.

Iloczyn skalarny

Iloczyn skalarny jest operatorem takim jak dodawanie i mnożenie. Służy do łączenia dwóch wektorów, jak pokazano:

$$\begin{pmatrix} 3 \\ 7 \end{pmatrix} \cdot \begin{pmatrix} 9 \\ 5 \end{pmatrix} = 3*9 + 7*5 = 62$$

Więc co to oznacza? Załóżmy, że mamy wektor, który reprezentuje sentyment klienta do trzech gatunków filmów - komediowego, romantycznego i akcji.

Używając iloczynu skalarnego, zauważ, że odpowiedzią jest pojedyncza liczba, zwana skalarem.

Weź pod uwagę, że w skali od 1 do 5 klient uwielbia komedie, nienawidzi filmów romantycznych i nie ma nic przeciwko filmom akcji. Możemy to przedstawić w następujący sposób:

$$\begin{pmatrix} 5 \\ 1 \\ 3 \end{pmatrix}$$

Tutaj:

• 5 oznacza miłość do komedii,

• 1 to nienawiść do romantyków

• 3 to obojętność działania

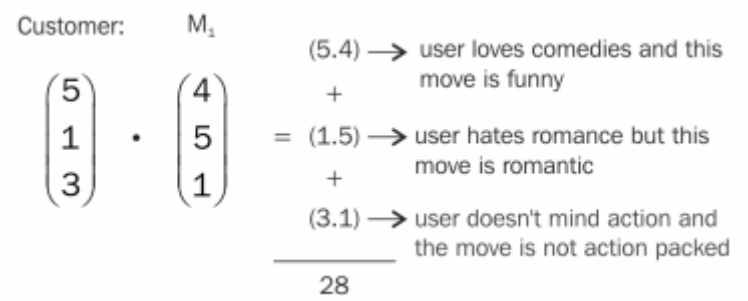
Załóżmy teraz, że mamy dwa nowe filmy, z których jeden to komedia romantyczna, a drugi to zabawny film akcji. Filmy miałyby swój własny wektor cech, jak pokazano:

$$m_1 = \begin{pmatrix} 4 \\ 5 \\ 1 \end{pmatrix} \text{ and } m_2 = \begin{pmatrix} 5 \\ 1 \\ 5 \end{pmatrix}$$

Tutaj m₁ to nasza komedia romantyczna, a m₂ to nasz zabawny film akcji. W celu dokonania rekomendacji zastosujemy iloczyn skalarny pomiędzy preferencjami klienta dla każdego filmu. Wyższa wartość wygra, a zatem zostanie polecona użytkownikowi. Obliczmy wynik rekomendacji dla każdego filmu. Dla filmu 1 chcemy obliczyć:

$$\begin{pmatrix} 5 \\ 1 \\ 3 \end{pmatrix} . \begin{pmatrix} 4 \\ 5 \\ 1 \end{pmatrix}$$

Możemy myśleć o tym problemie w następujący sposób:



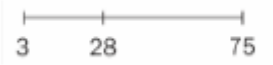
Uzyskujemy odpowiedź 28, ale co oznacza ta liczba? Na jaką skalę to jest? Cóż, najlepszy wynik, jaki można uzyskać, to gdy wszystkie wartości wynoszą 5, co daje następujący wynik:

$$\begin{pmatrix} 5 \\ 5 \\ 5 \end{pmatrix} . \begin{pmatrix} 5 \\ 5 \\ 5 \end{pmatrix} = 5^2 + 5^2 + 5^2 = 75$$

Najniższy możliwy wynik jest wtedy, gdy wszystkie wartości wynoszą 1, jak pokazano:

$$\begin{pmatrix}1\\1\\1\end{pmatrix}.\begin{pmatrix}1\\1\\1\end{pmatrix}=1^2+1^2+1^2=3$$

Musimy więc pomyśleć o 28 w skali od 3-75. Aby to zrobić, wyobraź sobie linię liczbową od 3 do 75 i gdzie będzie na niej 28. Jest to zilustrowane w następujący sposób:



Nie tak daleko. Spróbujmy do filmu 2:

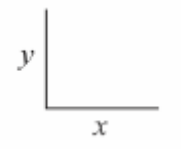
$$\begin{pmatrix}5\\1\\3\end{pmatrix}.\begin{pmatrix}5\\1\\5\end{pmatrix}=(5*5)+(1*1)+(3*5)=41$$

To więcej niż 28! Umieszczając tę liczbę na tej samej osi czasu, co poprzednio, możemy również wizualnie zaobserwować, że jest to znacznie lepszy wynik, jak pokazano:

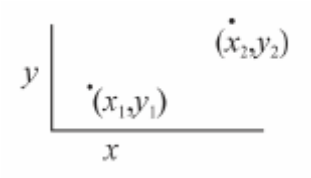
Tak więc, pomiędzy filmem 1 a filmem 2, zdecydowanie polecimy film 2 naszemu użytkownikowi. Tak właśnie działa większość silników przewidywania filmów. Tworzą profil klienta, który jest reprezentowany jako wektor. Następnie biorą wektorową reprezentację każdego filmu, który mają do zaoferowania, łączą je z profilem klienta (być może z iloczynem skalarnym) i stamtąd wydają rekomendacje. Oczywiście większość firm musi to robić na znacznie większą skalę, czyli tam, gdzie określona dziedzina matematyki, zwana algebrą liniową, może być bardzo przydatna.

Wykresy

Bez wątpienia w swoim życiu spotkałeś już dziesiątki, jeśli nie setki wykresów. Chciałbym porozmawiać głównie o konwencjach w odniesieniu do wykresów i notacji.



To jest podstawowy wykres kartezjański (współrzędne x i y). Notacja x i y jest bardzo standardowa, ale czasami nie wyjaśnia całego obrazu. Czasami nazywamy zmienną x jako zmienną niezależną, a y jako zmienną zależną. Dzieje się tak, ponieważ kiedy piszemy funkcje, mówimy o nich jako o tym, że y jest funkcją x, co oznacza, że wartość y zależy od wartości x. To właśnie próbuje pokazać wykres. Załóżmy, że na wykresie mamy dwa punkty, jak pokazano:



Odnosimy się do punktów jako (x_1,y_1) i (x_2,y_2) .

Nachylenie między tymi dwoma punktami definiuje się w następujący sposób:

$$slope = m = \frac{y_2 - y_1}{x_2 - x_1}$$

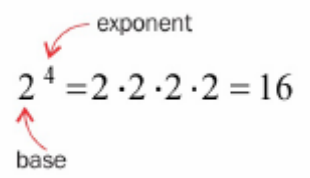
Zapewne widziałeś już tę formułę, ale warto o niej wspomnieć, gdyby nie jej znaczenie. Nachylenie określa szybkość zmian między dwoma punktami. Tempo zmian może być bardzo ważne w nauce o danych, szczególnie w obszarach związanych z równaniami różniczkowymi i rachunkiem różniczkowym. Tempo zmian to sposób na przedstawienie, jak zmienne poruszają się razem i w jakim stopniu. Weź pod uwagę, że modelujemy temperaturę Twojej kawy w zależności od czasu, w którym siedziała. Być może tempo zmian jest następujące:

$$-\frac{2\,degrees\,F}{1\,minute}$$

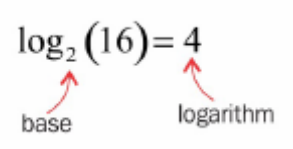
Ta szybkość zmian mówi nam, że z każdą minutą temperatura naszej kawy spada o dwa stopnie Fahrenheita. W dalszej części tej książki przyjrzymy się algorytmowi uczenia maszynowego, zwanemu regresją liniową. W regresji liniowej zajmujemy się szybkościami zmian między zmiennymi, ponieważ pozwalają one wykorzystać tę zależność do celów predykcyjnych. Pomyśl o płaszczyźnie kartezjańskiej jako o nieskończonej płaszczyźnie wektorów z dwoma elementami. Kiedy ludzie odnoszą się do wyższych wymiarów, takich jak 3D lub 4D, mają na myśli po prostu nieskończoną przestrzeń, w której znajdują się wektory z większą liczbą elementów. Przestrzeń 3D zawiera wektory o długości trzy, podczas gdy przestrzeń 7D zawiera wektory z siedmioma elementami.

Logarytmy/wykładniki

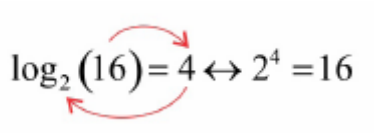
Wykładnik mówi Ci, ile razy musisz pomnożyć przez siebie liczbę, jak pokazano na ilustracji:



Logarytm to liczba odpowiadająca na pytanie: „jaki wykładnik przenosi mnie z podstawy do tej innej liczby?” Można to opisać w następujący sposób:



Jeśli te dwie koncepcje wydają się podobne, to masz rację! Wykładniki i logarytmy są ściśle powiązane. W rzeczywistości słowa wykładnik i logarytm oznaczają to samo! Logarytm jest wykładnikiem. Poprzednie dwa równania to w rzeczywistości dwie wersje tego samego. Podstawowym założeniem jest to, że 2 razy 2 razy 2 razy 2 daje 16. Poniżej znajduje się obraz, w jaki sposób możemy użyć obu wersji, aby powiedzieć to samo. Zwróć uwagę, jak używam strzałek, aby przejść z formuły logarytmicznej do formuły wykładniczej:



Rozważ następujące przykłady:

• $\log_3 81 = 4$ ponieważ $3^4 = 81$

• $\log_5 125 = 3$ ponieważ $5^3 = 125$

Zwróć uwagę na coś interesującego, jeśli przepiszemy pierwsze równanie tak:

$\log_3 81 = 4$

Następnie zastępujemy 81 równoważnym stwierdzeniem 34, w następujący sposób:

$\log_3 3^4 = 4$

Możemy zauważyć coś interesującego: trójki wydają się znosić. W rzeczywistości jest to bardzo ważne, gdy mamy do czynienia z liczbami trudniejszymi do pracy niż trójki i czwórki. W przypadku wzrostu najważniejsze są wykładniki i logarytmy. Najczęściej, jeśli jakaś wielkość rośnie (lub maleje), wykładnik/logarytm może pomóc w modelowaniu tego zachowania. Na przykład liczba e wynosi około 2,718 i ma wiele praktycznych zastosowań. Bardzo powszechnym zastosowaniem jest obliczanie wzrostu dla pieniędzy. Załóżmy, że masz 5000 USD zdeponowane w banku z ciągle naliczanymi odsetkami w wysokości 3%, wtedy możemy użyć następującego wzoru do modelowania wzrostu naszego depozytu:

$A = Pe^{rt}$

Gdzie:

• A oznacza ostateczną kwotę

• P oznacza inwestycję główną (5000)

• e oznacza stałą (2.718)

• r oznacza tempo wzrostu (0,03)

• t oznacza czas (w latach)

Jesteśmy ciekawi, kiedy nasza inwestycja się podwoi? Jak długo musiałbym mieć pieniądze w tej inwestycji, aby osiągnąć 100% wzrost? Zasadniczo:

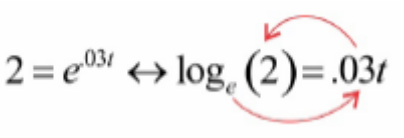
$10\,000 = 5000e^{0.3t}$

Czy formuła, którą chcemy rozwiązać:

$10\,000 = 5000e^{0.3t}$

$2 = e^{0.3t}$ (dzielone przez 5000 po obu stronach)

W tym momencie mamy zmienną w wykładniku, którą chcemy rozwiązać. Kiedy tak się stanie, możemy użyć notacji logarytmicznej, aby to rozgryźć!



To pozostawia nas z $\log_e(2) = .03t$

Logarytm liczby o podstawie e nazywamy logarytmem naturalnym. Przepisujemy logarytm w następujący sposób:

$\ln(2) = .03t$

Używając kalkulatora (lub Pythona), stwierdzamy, że ln(2) = 0,69.

$0.69 = 03.t$

$t = 2,31$

Oznacza to, że podwojenie naszych pieniędzy zajęłoby 2,31 roku.

Teoria mnogości

Teoria mnogości obejmuje operacje matematyczne na ustalonym poziomie. Czasami uważa się ją za podstawową, podstawową grupę twierdzeń, która rządzi resztą matematyki. Do naszych celów wykorzystujemy teorię mnogości, aby manipulować grupami elementów. Zestaw to zbiór odrębnych przedmiotów. Otóż to! Zestaw można traktować jako listę w Pythonie, ale bez powtarzających się obiektów. W rzeczywistości w Pythonie istnieje nawet zestaw obiektów:

```
s = set()

s = set([1, 2, 2, 3, 2, 1, 2, 2, 3, 2])

# usunie duplikaty z listy
```

s == {1, 2, 3}

Zauważ, że w Pythonie nawiasy klamrowe - {, } - mogą oznaczać zbiór lub słownik. Pamiętaj, że słownik w Pythonie to zestaw par klucz-wartość, na przykład:

dict = {"dog": "human's best friend", "cat": "destroyer of world"}

dict["dog"]# == "human's best friend"

len(dict["cat"]) # == 18

but if we try to create a pair with the same key as an existing key

dict["dog"] = "Arf"

dict

{"dog": "Arf", "cat": "destroyer of world"}

It will override the previous value

dictionaries cannot have two values for one key.

Dzielą tę notację, ponieważ mają tę samą cechę, że zestawy nie mogą zawierać zduplikowanych elementów, podobnie jak słowniki nie mogą mieć zduplikowanych kluczy. Wielkość zestawu to liczba elementów w zestawie i jest reprezentowana w następujący sposób:

|A| = wielkość A

s # == {1,2,3}

len(s) == 3 # magnitude of s

Pojęcie zbioru pustego istnieje i jest oznaczane przez znak . Mówi się, że ten pusty zestaw ma wartość 0. Jeśli chcemy zaznaczyć, że element znajduje się w zestawie, używamy notacji epsilon, jak pokazano:

$2 \in \{1,2,3\}$

Oznacza to, że element 2 istnieje w zbiorze 1, 2 i 3. Jeśli jeden zbiór jest całkowicie wewnątrz innego zbioru, mówimy, że jest podzbiorem jego większego odpowiednika.

A ={1,5,6} , B= {1,5,6,7,8}

$A \sqsubseteq B$

Tak więc A jest podzbiorem B, a B nazywa się nadzbiorem A. Jeśli A jest podzbiorem B, ale A nie jest równe B (co oznacza, że w B jest co najmniej jeden element, który nie znajduje się w A), to A nazywa się właściwym podzbiorem B.

Rozważ następujące przykłady:

• Zbiór liczb parzystych jest podzbiorem wszystkich liczb całkowitych

• Każdy zbiór jest sam w sobie podzbiorem, ale nie właściwym podzbiorem

• Zestaw wszystkich tweetów to nadzbiór angielskich tweetów

W nauce o danych używamy zestawów (i list) do reprezentowania listy obiektów i często do uogólniania zachowań konsumentów. Często sprowadza się klienta do zestawu cech. Weź pod uwagę, że jesteśmy firmą marketingową, która próbuje przewidzieć, gdzie dana osoba chce kupować ubrania. Dostajemy zestaw marek odzieżowych, które użytkownik odwiedził wcześniej, a naszym celem jest przewidzenie nowego sklepu, który również by mu się spodobał. Załóżmy, że określony użytkownik robił wcześniej zakupy w następujących sklepach:

user1 = {"Target","Banana Republic","Old Navy"}

note that we use {} notation to create a set

compare that to using [] to make a list

Tak więc użytkownik 1 wcześniej robił zakupy w Target, Banana Republic i Old Navy. Spójrzmy również na innego użytkownika, zwanego user2, jak pokazano:

user2 = {"Banana Republic","Gap","Kohl&priem;s"}

Załóżmy, że zastanawiamy się, jak podobni są ci użytkownicy. Mając ograniczone informacje, którymi dysponujemy, jednym ze sposobów na zdefiniowanie podobieństwa jest sprawdzenie, w ilu sklepach oboje robią zakupy. Nazywa się to skrzyżowaniem. Przecięcie dwóch zbiorów to zbiór, którego elementy występują w obu zbiorach. Jest oznaczony symbolem ∩, jak pokazano:

user1 ∩ user2 = { Banana Republic}

|user1 ∩ user2| = 1

Skrzyżowanie tych dwóch użytkowników to tylko jeden sklep. Więc od razu to nie wydaje się wspaniałe. Jednak każdy użytkownik ma w swoim zestawie tylko trzy elementy, więc posiadanie 1/3 nie wydaje się takie złe. Załóżmy, że jesteśmy ciekawi, ile sklepów jest reprezentowanych między nimi dwoma; nazywa się to związkiem. Połączenie dwóch zestawów to zestaw, którego elementy pojawiają się w każdym zestawie. Jest oznaczony symbolem ∪, jak pokazano:

user1 ∪; user2 = {Banana Republic, Target, Old Navy, Gap, Kohl′s}

|user1 ∪ user2| = 5

Patrząc na podobieństwo między user1 i user2 powinniśmy użyć kombinacji sumy i przecięcia ich zbiorów. user1 i user2 mają jeden wspólny element z pięciu różnych elementów między sobą. Możemy więc zdefiniować podobieństwo między dwoma użytkownikami w następujący sposób:

|user1 ∪user2| / |user1 ∪; user2| = 1/5 =.2

W rzeczywistości ma to swoją nazwę w teorii mnogości. Nazywa się to miarą jaccarda. Ogólnie rzecz biorąc, dla zestawów A i B miara jaccarda (podobieństwo jaccarda) między tymi dwoma zestawami jest zdefiniowana w następujący sposób:

$JS(A,B) = |A \cap B| / |A \cup B|$

Można go również zdefiniować jako wielkość przecięcia dwóch zbiorów podzieloną przez wielkość sumy tych dwóch zbiorów. Daje nam to sposób na ilościowe określenie podobieństw między elementami reprezentowanymi za pomocą zestawów. Intuicyjnie, miara jaccarda jest liczbą z zakresu od 0 do 1, tak że gdy liczba jest bliższa 0, ludzie są bardziej do siebie podobni, a gdy miara jest bliższa 1, ludzie są uważani za podobnych do siebie. Jeśli myślimy o definicji, to faktycznie ma to sens. Jeszcze raz spójrz na miarę:

JS = Liczba wspólnych sklepów / Unikalna liczba sklepów, które lubią łączyć

Tutaj licznik reprezentuje liczbę sklepów, które użytkownicy mają wspólnego (w tym sensie, że lubią tam robić zakupy), podczas gdy mianownik reprezentuje unikalną liczbę sklepów, które lubią razem. Możemy to przedstawić w Pythonie za pomocą prostego kodu, jak pokazano:

```
user1 = {"Target","Banana Republic","Old Navy"}

user2 = {"Banana Republic","Gap","Kohl's"}

def jaccard(user1, user2):

stores_in_common = len(user1 & user2)

stores_all_together = len(user1 | user2)

return stores / float(stores_all_together)

# I cast stores_all_together as a float to return a decimal answer instead of python's default integer division

# so

jaccard(user1, user2) == # 0.2 or 1/5
```

Teoria mnogości staje się bardzo rozpowszechniona, gdy wkraczamy w świat prawdopodobieństwa, a także gdy mamy do czynienia z danymi wielowymiarowymi. Użyjemy zbiorów do reprezentowania zachodzących wydarzeń w świecie rzeczywistym, a prawdopodobieństwo stanie się teorią mnogości z dodanym słownictwem.

Algebra liniowa

Pamiętasz silnik rekomendacji filmów, który widzieliśmy wcześniej? Co by było, gdybyśmy mieli 10 000 filmów do polecenia i musielibyśmy wybrać tylko 10 do przekazania użytkownikowi? Musielibyśmy umieścić iloczyn skalarny między profilem użytkownika a każdym z 10 000 filmów. Algebra liniowa dostarcza narzędzi, które znacznie usprawniają te obliczenia. Jest to dziedzina matematyki zajmująca się matematyką macierzy i wektorów. Ma na celu rozbicie tych obiektów i zrekonstruowanie ich w celu zapewnienia praktycznych zastosowań. Przyjrzyjmy się kilku zasadom algebry liniowej, zanim przejdziemy dalej.

Mnożenie macierzy

Podobnie jak liczby, możemy wielokrotne macierze razem. Mnożenie macierzy jest w istocie masową metodą pobierania kilku iloczynów skalarnych na raz. Spróbujmy na przykład pomnożyć następujące macierze:

$$\begin{pmatrix} 1 & 5 \\ 5 & 8 \\ 7 & 8 \end{pmatrix} \cdot \begin{pmatrix} 3 & 4 \\ 2 & 5 \end{pmatrix}$$

Kilka rzeczy:

• W przeciwieństwie do liczb, mnożenie nie jest przemienne, co oznacza, że kolejność mnożenia macierzy ma ogromne znaczenie.

• Aby pomnożyć macierze, ich wymiary muszą się zgadzać. Oznacza to, że pierwsza macierz musi mieć taką samą liczbę kolumn, jak druga macierz ma wiersze.

Aby to zapamiętać, wypisz wymiary matryc. W tym przypadku mamy macierz 3x2 razy 2x2. Możesz mieć wiele macierzy razem, jeśli druga liczba w pierwszej parze wymiarów jest taka sama jak pierwsza liczba w drugiej parze wymiarów.

$$3 \times 2 \cdot 2 \times 2$$

Otrzymana macierz zawsze będzie miała wymiary równe liczbom zewnętrznym w parach wymiarów (te, których nie zakreśliłeś w drugim punkcie). W takim przypadku otrzymana macierz będzie miała wymiar 3 x 2.

Jak pomnożyć macierze

Aby pomnożyć macierze, istnieje całkiem prosta procedura. Zasadniczo wykonujemy kilka produktów skalarnych. Przypomnij sobie nasz wcześniejszy przykładowy problem, który wyglądał następująco:

$$\begin{pmatrix} 1 & 5 \\ 5 & 8 \\ 7 & 8 \end{pmatrix} \cdot \begin{pmatrix} 3 & 4 \\ 2 & 5 \end{pmatrix}$$

Wiemy, że nasza wynikowa macierz będzie miała wymiar 3 x 2. Więc wiemy, że będzie wyglądać mniej więcej tak:

$$\begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \\ m_{31} & m_{32} \end{pmatrix}$$

Zauważ, że każdy element macierzy jest indeksowany przy użyciu podwójnego indeksu. Pierwsza liczba reprezentuje wiersz, a druga - kolumnę. Tak więc element jest elementem m_3 w trzecim rzędzie, drugiej kolumnie. Każdy element jest wynikiem iloczynu skalarnego między wierszami i kolumnami macierzy oryginalnych. Element m_{xy} jest wynikiem iloczynu skalarnego x-tego wiersza pierwszej macierzy i y-tej kolumny drugiej macierzy. Rozwiążmy kilka:

$$m_{11} = \begin{pmatrix} 1 \\ 5 \end{pmatrix} \cdot \begin{pmatrix} 3 \\ 2 \end{pmatrix} = 13$$

$$m_{12} = \begin{pmatrix} 1 \\ 5 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 5 \end{pmatrix} = 29$$

Idąc dalej, ostatecznie otrzymamy wynikową macierz w następujący sposób:

$$\begin{pmatrix} 13 & 29 \\ 31 & 60 \\ 37 & 68 \end{pmatrix}$$

Tak trzymać! Wróćmy do przykładu rekomendacji filmu. Przypomnij sobie preferencje użytkownika dotyczące gatunku filmu, takie jak komedia, romans i akcja, które są zilustrowane w następujący sposób:

$$U = user\ prefs = \begin{pmatrix} 5 \\ 1 \\ 3 \end{pmatrix}$$

Załóżmy teraz, że mamy 10 000 filmów, wszystkie z oceną w tych trzech kategoriach. Aby dokonać rekomendacji, musimy wziąć iloczyn skalarny wektora preferencji dla każdego z 10 000 filmów. Aby to przedstawić, możemy użyć mnożenia macierzy. Zamiast wypisywać je wszystkie, wyraźmy to za pomocą notacji macierzowej. Mamy już U, zdefiniowane tutaj jako wektor preferencji użytkownika (można go również traktować jako macierz 3x1) i potrzebujemy również macierzy filmowej:

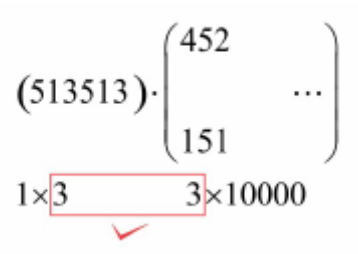
M = filmy = matryca 3 x 10 000 wymiarów

Więc teraz mamy dwie macierze, jedna to 3 x 1, a druga to 3 x 10 000. Nie możemy pomnożyć tych macierzy, ponieważ wymiary się nie sprawdzają. Będziemy musieli trochę zmienić U. Możemy wziąć

transpozycję macierzy (zamieniając wszystkie wiersze w kolumny, a kolumny w wiersze). To zmieni wymiary wokół:

U^T = transpozycja U = (513)

Więc teraz mamy dwie macierze, które można przez siebie pomnożyć. Aby zwizualizować, jak to wygląda:



Otrzymana macierz będzie macierzą 1 x 1000 (wektor) z 10 000 przewidywań dla każdego pojedynczego filmu. Wypróbujmy to w Pythonie!

```
# create user preferences

user_pref = np.array([5, 1, 3])

# create a random movie matrix of 10,000 movies

movies = np.random.randint(5,size=(3,1000))+1

# Note that the randint will make random integers from 0-4

# so I added a 1 at the end to increase the scale from 1-5
```

Używamy funkcji numpy array do tworzenia naszych macierzy. Będziemy mieli zarówno macierz user_pref, jak i macierz filmów do reprezentowania naszych danych. Aby sprawdzić nasze wymiary, możemy użyć zmiennej kształtu numpy, jak pokazano:

```
print user_pref.shape # (1, 3)

print movies.shape # (3, 1000)
```

To się sprawdza. Na koniec użyjmy metody mnożenia macierzy numpy (zwanej kropką), aby wykonać operację, jak zilustrowano:

```
# np.dot does both dot products and matrix multiplication

np.dot(user_pref, movies)
```

Wynikiem jest tablica liczb całkowitych, które reprezentują rekomendacje każdego filmu. Aby to szybko rozszerzyć, uruchommy kod, który przewiduje w ponad 10 000 filmów, jak pokazano:

```
import time
```

```
for num_movies in (10000, 100000, 1000000, 10000000, 100000000):

movies = np.random.randint(5,size=(3, movies))+1

now = time.time()

np.dot(user_pref, movies)

print (time.time() - now), "seconds to run", movies, "movies"
```

0.000160932540894 seconds to run 10000 movies

0.00121188163757 seconds to run 100000 movies

0.0105860233307 seconds to run 1000000 movies

0.096577167511 seconds to run 10000000 movies

4.16197991371 seconds to run 100000000 movies

Przejrzenie 100 000 000 filmów za pomocą mnożenia macierzy zajęło tylko nieco ponad 4 sekundy.

Podsumowanie

W tej części przyjrzeliśmy się kilku podstawowym zasadom matematycznym, które staną się bardzo ważne w miarę postępów. Pomiędzy logarytmami/wykładnikami, algebrą macierzową i proporcjonalnością matematyka wyraźnie odgrywa dużą rolę nie tylko w analizie danych, ale w wielu aspektach naszego życia. W kolejnych częściach zajmiemy się znacznie głębiej dwoma dużymi obszarami matematyki: prawdopodobieństwem i statystyką. Naszym celem będzie zdefiniowanie i zinterpretowanie najmniejszych i największych twierdzeń w tych dwóch gigantycznych dziedzinach matematyki. To w kilku następnych rozdziałach wszystko zacznie się układać. Do tej pory w tej książce przyjrzeliśmy się przykładom matematycznym, wskazówkom dotyczącym eksploracji danych oraz podstawowym wglądowi w typy danych. Czas zacząć łączyć wszystkie te koncepcje razem.

Niemożliwe lub nieprawdopodobne - delikatne wprowadzenie do prawdopodobieństwa

W następnych kilku częściach będziemy badać zarówno prawdopodobieństwo, jak i statystyki jako metody badania zarówno sytuacji opartych na danych, jak i rzeczywistych scenariuszy. Zasady prawdopodobieństwa rządzą podstawami przewidywania. Używamy prawdopodobieństwa do określenia prawdopodobieństwa wystąpienia zdarzenia. W tej części przyjrzymy się następującym tematom:

• Co to jest prawdopodobieństwo?

• Różnice między podejściem Frequentystycznym a Bayesowskim

• Jak wizualizować prawdopodobieństwo

• Jak korzystać z reguł prawdopodobieństwa

• Używanie macierzy pomyłek do przyjrzenia się podstawowym metrykom

Prawdopodobieństwo pomoże nam w modelowaniu rzeczywistych wydarzeń, które zawierają poczucie przypadkowości i przypadku. W kolejnych dwóch częściach przyjrzymy się terminologii stojącej za twierdzeniami prawdopodobieństwa i sposobom ich zastosowania w sytuacjach modelowych, które mogą pojawić się nieoczekiwanie.

Podstawowe definicje

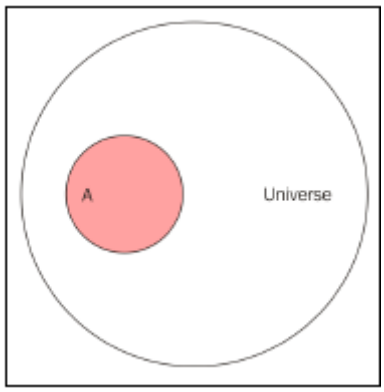
Jednym z podstawowych pojęć prawdopodobieństwa jest pojęcie procedury. Procedura to czynność, która prowadzi do rezultatu. Na przykład rzucanie kostką lub odwiedzanie strony internetowej. Zdarzenie to zbiór wyników procedury, takich jak trafienie orzełkiem w monetę lub opuszczenie strony internetowej po zaledwie 4 sekundach. Proste zdarzenie to wynik/zdarzenie procedury, którego nie można dalej rozbić. Na przykład rzut dwiema kośćmi można podzielić na dwa proste zdarzenia: rzut 1 kostką, i rzut 2. Przestrzeń próbki procedury to zbiór wszystkich możliwych prostych zdarzeń. Na przykład przeprowadzany jest eksperyment, w którym moneta jest rzucana trzy razy z rzędu. Jaka jest wielkość przestrzeni próbnej dla tego eksperymentu? Odpowiedzią jest osiem, ponieważ wynikiem może być dowolna z możliwości w następującej przestrzeni próbek - {OOO, OOR, ORR, ORO, RRR, RRO, ROO lub ROR}.

Prawdopodobieństwo

Prawdopodobieństwo zdarzenia reprezentuje częstotliwość lub szansę, że zdarzenie się wydarzy. Dla notacji, jeśli A jest zdarzeniem, P(A) jest prawdopodobieństwem wystąpienia zdarzenia. Możemy zdefiniować rzeczywiste prawdopodobieństwo zdarzenia A w następujący sposób:

P(A) = liczba razy występowania A / wielkość przestrzeni próbki

Tutaj A jest wydarzeniem, o którym mowa. Pomyśl o całym wszechświecie wydarzeń, w których wszystko jest możliwe, i przedstawmy to jako okrąg. Możemy myśleć o pojedynczym zdarzeniu A jako o mniejszym okręgu w tym większym wszechświecie, jak pokazano na poniższym diagramie:



Załóżmy teraz, że w naszym wszechświecie prowadzone są badania naukowe na ludziach, a wydarzeniem A są ludzie z tymi badaniami, którzy mają raka. Jeśli nasze badanie obejmuje 100 osób, a A ma 25 osób, prawdopodobieństwo wystąpienia A lub P(A) wynosi 25/100. Maksymalne prawdopodobieństwo wystąpienia dowolnego zdarzenia wynosi 1. Można to rozumieć, jako że czerwony okrąg powiększa się tak bardzo, że jest to rozmiar wszechświata (większy okrąg). Najbardziej podstawowe przykłady (obiecuję, że będą ciekawsze) to rzuty monetą. Powiedzmy, że mamy dwie monety i chcemy mieć prawdopodobieństwo, że wyrzucimy dwie głowy. Bardzo łatwo możemy policzyć, w jaki sposób dwie monety mogą być dwiema orłami. Jest tylko jeden! Obie monety muszą być głowami. Ale ile jest opcji? Mogą to być dwa orły, dwa reszki lub kombinacja orła/reszki. Najpierw zdefiniujmy A. Jest to zdarzenie, w którym występują dwie głowy. Liczba sposobów, w jakie A może wystąpić, wynosi 1. Przestrzeń próbki eksperymentu to {HH, HT, TH, TT}, gdzie każde dwuliterowe słowo wskazuje jednocześnie wynik pierwszej i drugiej monety. Wielkość przestrzeni próbki to cztery. Tak więc P (uzyskanie dwóch głów) = 1/4. Aby to udowodnić, odwołajmy się do szybkiej tabeli wizualnej. Poniższa tabela przedstawia opcje dla monety 1 jako kolumny i opcje dla monety 2 jako rzędy. W każdej komórce znajduje się prawda lub fałsz. Wartość True wskazuje, że spełnia warunek (oba orły), a False wskazuje inaczej.

	Moneta 1 to Orzeł	Moneta 1 to Reszka
Moneta 2 to Orzeł	True	False
Moneta 2 to Reszka	False	False

Mamy więc jeden z czterech możliwych wyników.

Bayesian kontra Frequentist

Poprzedni przykład był prawie zbyt prosty. W praktyce prawie nigdy nie jesteśmy w stanie naprawdę policzyć, na ile może się coś wydarzyć. Załóżmy na przykład, że chcemy poznać prawdopodobieństwo, że przypadkowa osoba pali papierosy przynajmniej raz dziennie. Gdybyśmy chcieli podejść do tego problemu w sposób klasyczny (poprzednia formuła), musielibyśmy dowiedzieć się, na ile różnych sposobów dana osoba jest palaczem - ktoś, kto pali przynajmniej raz dziennie - co nie jest możliwe! W obliczu takiego problemu, przy obliczaniu prawdopodobieństw w praktyce, rozważa się dwie główne szkoły myślenia: podejście Frequentystyczne i podejście bayesowskie. Tu skupimy się głównie na podejściu Frequentystycznym, podczas gdy kolejna część zagłębi się w analizę bayesowską.

Frequentystyczne podejście

W podejściu Frequentystycznym prawdopodobieństwo zdarzenia jest obliczane poprzez eksperymenty. Wykorzystuje przeszłość, aby przewidzieć przyszłą szansę na wydarzenie. Podstawowa formuła wygląda następująco:

P(A) = liczba razy występowania A / ile razy procedura została powtórzona

Zasadniczo obserwujemy kilka przypadków zdarzenia i liczymy, ile razy A było spełnione. Dzielenie tych liczb jest przybliżeniem prawdopodobieństwa. Podejście bayesowskie różni się tym, że dyktuje, że prawdopodobieństwa należy rozróżniać za pomocą środków teoretycznych. Stosując podejście Bayesa, musielibyśmy bardziej krytycznie myśleć o zdarzeniach i ich przyczynach. Żadna metodologia nie jest zawsze w 100% poprawną odpowiedzią. Zwykle sprowadza się to do problemu i trudności w stosowaniu obu podejść. Sednem podejścia Frequentist jest względna częstotliwość. Względna częstotliwość zdarzenia to częstotliwość występowania zdarzenia podzielona przez całkowitą liczbę

Przykład - statystyki marketingowe

Załóżmy, że interesuje Cię ustalenie, jak często osoba odwiedzająca Twoją witrynę prawdopodobnie wróci w późniejszym terminie. Jest to czasami nazywane wskaźnikiem powracających gości. W poprzedniej definicji nasze wydarzenie A zdefiniowalibyśmy jako odwiedzający powracający na stronę. Musielibyśmy wtedy obliczyć, na ile dana osoba może wrócić, co nie ma żadnego sensu! W tym przypadku wiele osób zwróciłoby się ku podejściu bayesowskiemu; jednak możemy obliczyć tak zwaną częstotliwość względną. Tak więc w tym przypadku możemy wziąć dzienniki odwiedzających i obliczyć względną częstotliwość zdarzenia A (powtarzający się odwiedzający). Załóżmy, że z 1458 unikalnych użytkowników w zeszłym tygodniu 452 było powracającymi użytkownikami. Możemy to obliczyć w następujący sposób:

P(A) RF(A) = 452/1458 = 0,31

Tak więc około 31% Twoich odwiedzających to powracający odwiedzający.

Prawo wielkich liczb

Powodem, dla którego nawet podejście Frequentystyczne może to zrobić, jest prawo wielkich liczb, które mówi, że jeśli będziemy powtarzać procedurę w kółko, prawdopodobieństwo względnej częstotliwości zbliży się do prawdopodobieństwa rzeczywistego. Spróbujmy to zademonstrować za pomocą Pythona. Gdybym zadał ci średnią z liczb 1 i 10, bardzo szybko odpowiedziałbyś na około 5. To pytanie jest identyczne z prośbą o wybranie średniej liczby z przedziału od 1 do 10. Zaprojektujmy eksperyment w następujący sposób: Python wybierze n losowych liczb od 1 do 10 i znajdzie ich średnią. Powtórzymy ten eksperyment kilka razy, używając za każdym razem większej liczby n, a następnie wykreślimy wynik. Kroki są następujące:

1. Wybierz losową liczbę od 1 do 10 i znajdź średnią.

2. Wybierz dwie losowe liczby od 1 do 10 i znajdź ich średnią.

3. Wybierz trzy losowe liczby od 1 do 10 i znajdź ich średnią.

4. Wybierz 10 000 losowych liczb od 1 do 10 i znajdź ich średnią.

5. Wykres wyników.

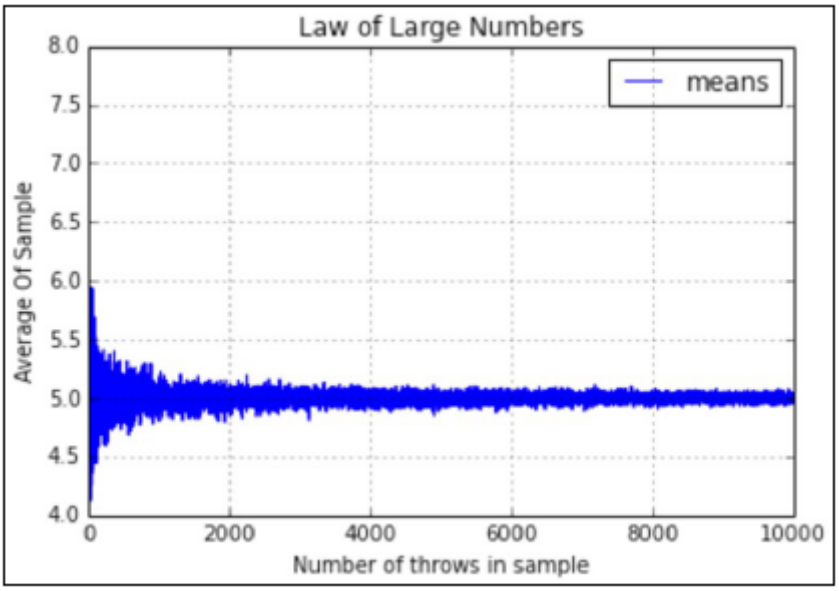
Rzućmy okiem na kod:

```
import numpy as np
```

```
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
results = []
for n in range(1,10000):
nums = np.random.randint(low=1,high=10, size=n) # choose n numbers
between 1 and 10
mean = nums.mean() # find the average
of these numbers
results.append(mean) # add the average
to a running list
# POP QUIZ: How large is the list results?
len(results) # 9999
# This was tricky because I took the range from 1 to 10000 and usually
we do from 0 to 10000
df = pd.DataFrame({ 'means' : results})
print df.head() # the averages in the beginning are all over the place!
# means
# 9.0
# 5.0
# 6.0
# 4.5
# 4.0
print df.tail() # as n, our size of the sample size, increases, the averages get closer to 5!
# means
# 4.998799
# 5.060924
# 4.990597
# 5.008802
# 4.979198
df.plot(title='Law of Large Numbers')
```

plt.xlabel("Number of throws in sample")

plt.ylabel("Average Of Sample")



Fajnie, prawda? Zasadniczo pokazuje nam to, że gdy zwiększamy wielkość próbki naszej względnej częstości, częstotliwość zbliża się do rzeczywistej średniej (prawdopodobieństwa) równej 5. W naszych rozdziałach dotyczących statystyk będziemy pracować nad zdefiniowaniem tego prawa znacznie bardziej rygorystycznie, ale na razie , po prostu wiedz, że jest używany do powiązania względnej częstotliwości zdarzenia z jego rzeczywistym prawdopodobieństwem.

str. 93

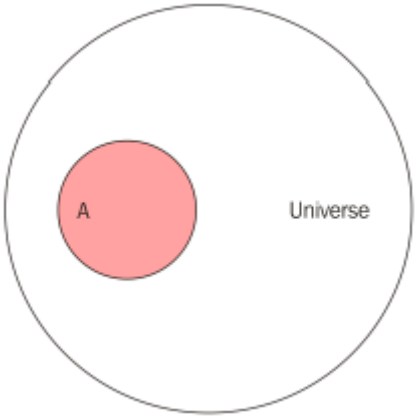
Wydarzenia złożone

Czasami musimy mieć do czynienia z dwoma lub więcej zdarzeniami. Są to tak zwane zdarzenia złożone. Zdarzenie złożone to dowolne zdarzenie, które łączy dwa lub więcej prostych zdarzeń. Kiedy tak się dzieje, potrzebujemy specjalnej notacji. Biorąc pod uwagę zdarzenia A i B:

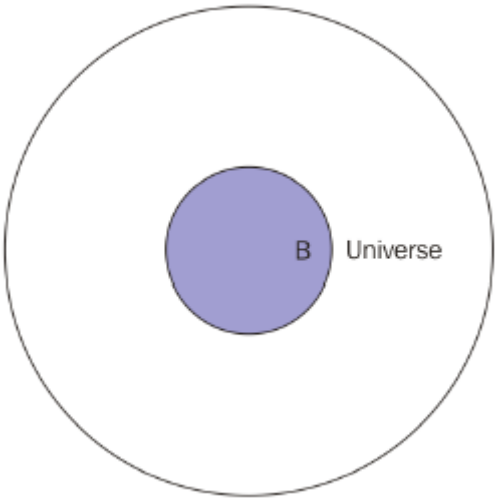
• Prawdopodobieństwo wystąpienia A i B wynosi P(A ∩ B) = P(A i B)

• Prawdopodobieństwo wystąpienia A lub B wynosi P(A $\&cu \cup p$; B) = P(A lub B)

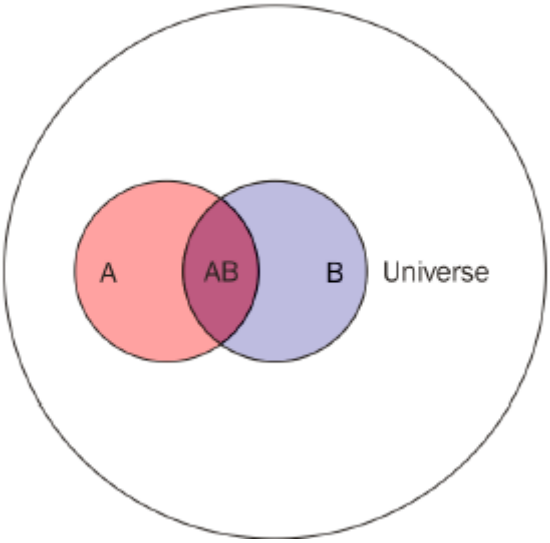
Zrozumienie, dlaczego używamy notacji zbiorowej dla tych złożonych zdarzeń, jest bardzo ważne. Pamiętasz, jak wcześniej przedstawialiśmy wydarzenia we wszechświecie za pomocą kół? Załóżmy, że nasz Wszechświat to 100 osób, które zgłosiły się na eksperyment, w którym opracowywany jest nowy test na raka:



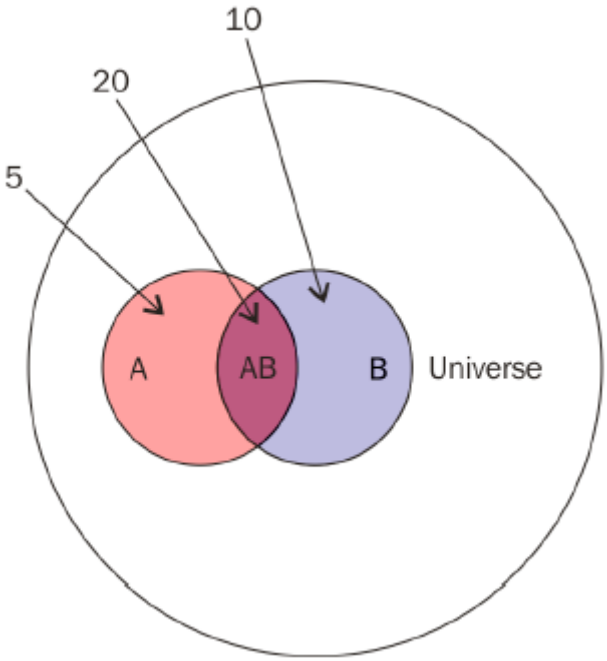
Na poprzednim schemacie czerwone kółko A przedstawia 25 osób, które rzeczywiście mają raka. Stosując podejście względnej częstotliwości, możemy powiedzieć, że P(A) = liczba osób z rakiem/liczba osób w badaniu, czyli 25/100 = ¼ = 0,25. Oznacza to, że istnieje 25% szans, że ktoś ma raka. Wprowadźmy drugie zdarzenie, nazwane B, jako show , które zawiera osoby, u których wynik testu był pozytywny (twierdził, że mają raka). Powiedzmy, że to jest dla 30 osób. Tak więc P(B) = 30/100 = 3/10 = 0,3. Oznacza to, że istnieje 30% szans na to, że test wykaże pozytywny wynik dla danej osoby:



Są to dwa oddzielne wydarzenia, ale wzajemnie na siebie oddziałują. Mianowicie mogą się przecinać lub mieć wspólnych ludzi, jak pokazano tutaj:



Każdy w przestrzeni, którą zajmują zarówno A, jak i B, inaczej znany jako A przecina B lub A ∩ B, to ludzie, dla których test stwierdził, że mają pozytywny wynik na raka (A) i faktycznie mają raka. Powiedzmy, że to 20 osób. Test wykazał pozytywny wynik dla 20 osób, to znaczy, że mają raka, jak pokazano tutaj:



Oznacza to, że P(A i B) = 20/100 = 1/5 = 0,2 = 20%. Jeśli chcemy powiedzieć, że ktoś ma raka lub wynik testu wypadł pozytywnie. Byłaby to łączna suma (lub suma) dwóch zdarzeń, a mianowicie suma 5, 20 i 10, czyli 35. Zatem 35/100 osób albo ma raka, albo miało pozytywny wynik testu. Oznacza to, że P(A lub B) = 35/100 = 0,35 = 35%. W sumie mamy ludzi w następujących czterech różnych klasach:

• Różowy: dotyczy osób z rakiem i z negatywnym wynikiem testu

• Fioletowy (A przecięcie B): Ci ludzie mają raka i mają pozytywny wynik testu

• Niebieski: dotyczy osób bez raka i z pozytywnym wynikiem testu

• Biały: dotyczy osób bez raka i z negatywnym wynikiem testu

Tak więc, w praktyce, test był dokładny tylko wtedy, gdy był w obszarach białych i fioletowych. W obszarach niebieskim i różowym test był nieprawidłowy.

Warunkowe prawdopodobieństwo

Wybierzmy dowolną osobę z badania 100 osób. Załóżmy również, że powiedziano ci, że ich wynik testu był pozytywny. Jakie jest prawdopodobieństwo, że rzeczywiście zachorują na raka? Tak więc powiedziano nam, że wydarzenie B już się odbyło i że ich test wypadł pozytywnie. Teraz pytanie brzmi: jakie jest prawdopodobieństwo, że mają raka, czyli P(A)? Nazywa się to prawdopodobieństwem warunkowym danego B lub P(A|B). W rzeczywistości jest to prośba o obliczenie prawdopodobieństwa zdarzenia, biorąc pod uwagę, że inne zdarzenie już się wydarzyło. Możesz myśleć o prawdopodobieństwie warunkowym jako o zmianie odpowiedniego wszechświata. P(A|B) (nazywane prawdopodobieństwem A przy danym B) to sposób na powiedzenie, biorąc pod uwagę, że cały mój wszechświat jest teraz B, jakie jest prawdopodobieństwo A? Jest to również znane jako przekształcanie przestrzeni próbki. Wzór może być podany w następujący sposób:

P(A|B) = P(A i B) / P(B) = (20/100) / (30/100) = 20/30 = 0,66 = 66%

Istnieje 66% szans, że jeśli wynik testu będzie pozytywny, ta osoba ma raka. W rzeczywistości jest to główne prawdopodobieństwo, którego chcą eksperymentatorzy. Chcą wiedzieć, jak dobry jest test w przewidywaniu raka.

Zasady prawdopodobieństwa

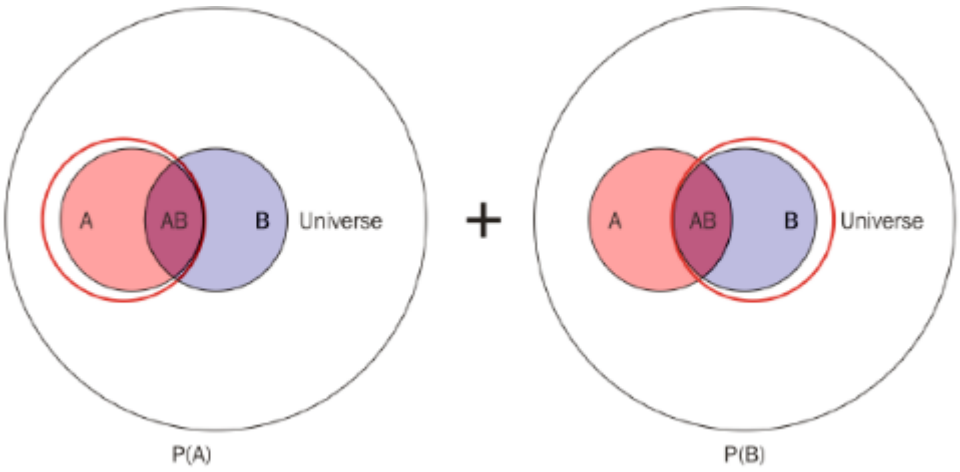
W prawdopodobieństwie mamy pewne zasady, które stają się bardzo przydatne, gdy wizualizacja staje się zbyt uciążliwa. Te reguły pomagają nam z łatwością obliczać prawdopodobieństwa złożone.

Zasada dodawania

Reguła dodawania służy do obliczania prawdopodobieństwa zdarzenia lub zdarzeń. Licząc P(A ∪ B) = P(A lub B), posługujemy się następującym wzorem:

P(A ∪ B) = P(A) + P(B) − P(A ∩ B)

Pierwsza część wzoru (P(A) + P(B)) ma sens. Aby uzyskać połączenie tych dwóch wydarzeń, musimy zsumować obszar okręgów we wszechświecie. Ale dlaczego odejmowanie P(A i B)? Dzieje się tak, ponieważ dodając dwa okręgi, dwukrotnie dodajemy obszar przecięcia, jak pokazano na poniższym diagramie:



Widzisz, jak oba czerwone kółka obejmują przecięcie A i B? Tak więc, kiedy je dodajemy, musimy odjąć tylko jedną z nich, aby to uwzględnić, pozostawiając nam naszą formułę. Przypomnij sobie, że chcieliśmy uzyskać liczbę osób, które albo miały raka, albo miały pozytywny wynik testu? Jeśli A to zdarzenie, że ktoś ma raka, a B to, że wynik testu był pozytywny, mamy:

P(A lub B) = P(A) + P(B) – P(A i B) = 0,25 + 0,30 - 0,2 = 0,35

Zostało to obliczone wcześniej wizualnie na schemacie.

Wzajemna wyłączność

Mówimy, że dwa zdarzenia wykluczają się wzajemnie, jeśli nie mogą wystąpić w tym samym czasie. Oznacza to, że A ∩ B= ∅ lub po prostu przecięcie zdarzeń jest zbiorem pustym. Kiedy tak się dzieje, P(A ∩ B) = P(A i B) = 0. Jeżeli dwa zdarzenia wzajemnie się wykluczają, to:

P(A ∪ B) = P(A lub B) = P(A) + P(B) − P(A ∩ B) = P(A) + P(B)

To znacznie ułatwia regułę dodawania. Oto kilka przykładów wzajemnie wykluczających się wydarzeń:

• Klient widzi Twoją witrynę po raz pierwszy zarówno na Twitterze, jak i na Facebooku

• Dzisiaj jest sobota, a dziś jest środa

• Oblałem Econ 101 i zdałem Econ 101

Żadne z tych zdarzeń nie może wystąpić jednocześnie.

Zasada mnożenia

Reguła mnożenia służy do obliczania prawdopodobieństwa zdarzeń i zdarzeń. Aby obliczyć P(A ∩ B) = P(A i B), używamy następującego wzoru:

P(A ∩ B) = P(A i B) = P(A) · P(B|A)

Dlaczego używamy B|A zamiast B? Dzieje się tak, ponieważ możliwe jest, że B zależy od A. Jeśli tak jest, to samo pomnożenie P(A) i P(B) nie daje pełnego obrazu. W naszym przykładzie z rakiem znajdźmy P(A i B). Aby to zrobić, przedefiniujmy A jako zdarzenie, w którym badanie jest pozytywne, a B jako osobę z rakiem (ponieważ nie ma znaczenia, jak nazywamy zdarzenia). Równanie będzie wyglądało następująco:

P(A ∩; B) = P(A i B) = P(A) · P(B|A) = 0,3 * 0,6666 = 0,2 = 20%

Zostało to obliczone wcześniej wizualnie. Trudno dostrzec prawdziwą konieczność korzystania z prawdopodobieństwa warunkowego, spróbujmy więc inny, trudniejszy problem. Na przykład z losowo wybranego zestawu 10 osób 6 ma iPhone'y, a 4 Androidy. Jakie jest prawdopodobieństwo, że jeśli losowo wybiorę dwie osoby, obie będą miały iPhone'y? Ten przykład można powtórzyć za pomocą przestrzeni zdarzeń w następujący sposób:

Mam następujące dwa wydarzenia:

• O: To zdarzenie pokazuje prawdopodobieństwo, że jako pierwszy wybiorę osobę z iPhonem

• B: To zdarzenie pokazuje prawdopodobieństwo, że wybiorę drugą osobę z iPhonem

Więc zasadniczo chcę, co następuje:

• P(A i B): P(wybieram osobę z iPhone'em i osobę z iPhone'em)

Więc mogę użyć mojego wzoru P(A i B) = P(A) · P(B|A). P(A) jest proste, prawda? Osób z iPhone'ami jest 6 na 10, więc mam 6/10 = 3/5 = 0,6 szansy na A. To oznacza, że P(A) = 0,6. Tak więc, jeśli mam 0,6 szansy na wybranie kogoś z iPhonem, prawdopodobieństwo wyboru dwóch powinno wynosić 0,6 * 0,6, prawda?

Ale poczekaj! Zostało nam tylko 9 osób do wyboru naszej drugiej osoby, bo jedna została zabrana. Tak więc w naszej nowej przekształconej przestrzeni próbnej mamy w sumie 9 osób, 5 z iPhone'ami i 4 z Androidami, co daje P(B) = 5/9 = 0,555. Tak więc prawdopodobieństwo wyboru dwóch osób z iPhone'ami wynosi 0,6 * 0,555 = 0,333 = 33%. Mam 1/3 szansy na wybranie dwóch osób z iPhone'ami spośród 10. Prawdopodobieństwo warunkowe jest bardzo ważne w regule mnożenia, ponieważ może drastycznie zmienić twoją odpowiedź.

Niezależność

Dwa zdarzenia są niezależne, jeśli jedno zdarzenie nie wpływa na wynik drugiego, czyli P(B|A) = P(B) i P(A|B) = P(A).

Jeżeli dwa zdarzenia są niezależne, to:

P(A ∩ B) = P(A) ·; P(B|A) = P(A) · P(B)

Oto kilka przykładów niezależnych wydarzeń:

• W San Francisco padało, a w Indiach urodził się szczeniak

• Rzuć monetą i zdobądź orły i rzucaj kolejną monetą i zdobądź reszki

Żadna z tych par wydarzeń nie wpływa na siebie nawzajem.

Wydarzenia uzupełniające

Uzupełnienie A jest przeciwieństwem lub negacją A. Jeśli A jest wydarzeniem, $\overline{A}$ reprezentuje uzupełnienie A. Na przykład, jeśli A jest wydarzeniem, w którym ktoś ma raka, $\overline{A}$ jest wydarzeniem, w którym ktoś jest wolny od raka. Aby obliczyć prawdopodobieństwo $\overline{A}$, użyj następującego wzoru:

$$P(\overline{A}) = 1 - P(A)$$

Na przykład, kiedy rzucasz dwiema kostkami, jakie jest prawdopodobieństwo, że wyrzuciłeś więcej niż 3? Niech A reprezentuje toczenie wyższe niż 3.

$\overline{A}$ oznacza wyrzucenie 3 lub mniej.

$$P(A) = 1 - P(\overline{A})$$

$$P(A) = 1 - (P(2)+P(3))$$

$$= 1 - (2/36 + 2/36)$$

$$= 1 - (4/36)$$

$$= 32/36 = 8/9$$

$$= .89$$

Na przykład zespół start-upów ma przed sobą trzy spotkania z inwestorami. Będziemy mieli następujące prawdopodobieństwa:

• 60% szans na zdobycie pieniędzy od pierwszego spotkania

• 15% szansy na otrzymanie pieniędzy od drugiego

• 45% szansy na zdobycie pieniędzy od trzeciego

Jakie jest prawdopodobieństwo, że zdobędą pieniądze z przynajmniej jednego spotkania? Niech A będzie zespołem otrzymującym pieniądze od co najmniej jednego inwestora i niech będzie zespołem, który nie otrzyma żadnych pieniędzy. P(A) można obliczyć w następujący sposób:

$$P(A) = 1 - P(\overline{A})$$

Aby obliczyć $P(\overline{A})$, musimy obliczyć:

$P(\overline{A}) = P$ (brak pieniędzy od inwestora 1 AND brak pieniędzy od inwestora 2 ORAZ brak pieniędzy od inwestora 3) Jeśli założymy, że te zdarzenia są niezależne (nie rozmawiają ze sobą), to:

$P(\overline{A}) = P$ (brak pieniędzy od inwestora 1) * P(brak pieniędzy od inwestora 2) * P(brak pieniędzy od inwestora 3) =

0,4 * 0,85 * 0,55 = 0,187

P(A) = 1 - 0,187 = 0,813 = 81 %

Tak więc startup ma 81% szans na zdobycie pieniędzy z przynajmniej jednego spotkania!

Trochę głębiej

Bez zbytniego zagłębiania się w terminologię uczenia maszynowego, ten test jest tak zwany klasyfikatorem binarnym, co oznacza, że próbuje przewidzieć tylko dwie opcje: mieć raka lub nie mieć raka. Kiedy mamy do czynienia z klasyfikatorami binarnymi, możemy narysować tak zwane macierze pomyłek, które są macierzami 2 x 2, które zawierają wszystkie cztery możliwe wyniki naszego eksperymentu. Spróbujmy różnych liczb. Powiedzmy, że na badanie przyszło 165 osób. Tak więc nasze n (wielkość próbki) to 165 osób. Wszystkim 165 osobom poddaje się test i pyta, czy ma raka (przeprowadzanego różnymi innymi środkami). Poniższa macierz pomyłek pokazuje nam wyniki tego eksperymentu:

n=165	Predicted: NO	Predicted: YES
Actual: NO	50	10
Actual: YES	5	100

Macierz pokazuje, że przewidziano, że 50 osób nie ma i nie ma raka, 100 osób miało raka i rzeczywiście go miało, i tak dalej. Ponownie mamy następujące cztery klasy, wszystkie o różnych nazwach:

• Prawdziwie pozytywne są testy prawidłowo przewidujące wynik pozytywny (rak) == 100

• Prawdziwie negatywy to testy prawidłowo przewidujące wynik negatywny (brak raka) == 50

• Fałszywie pozytywne to testy, które błędnie przewidują wynik pozytywny (rak) == 10

• Wyniki fałszywie ujemne to testy błędnie przewidujące wynik ujemny (brak raka) == 5

Pierwsze dwie klasy wskazują, gdzie test był poprawny lub prawdziwy. Dwie ostatnie klasy wskazują, gdzie test był błędny lub fałszywy. Fałszywe trafienia są czasami nazywane błędami typu I, podczas gdy fałszywie ujemne są nazywane błędami typu II.

Zajmiemy się tym w dalszych częściach. Na razie musimy tylko zrozumieć, dlaczego używamy notacji zbioru do oznaczania prawdopodobieństw zdarzeń złożonych. Dzieje się tak, ponieważ tym właśnie są. Kiedy zdarzenia A i B istnieją w tym samym wszechświecie, możemy użyć skrzyżowań i związków, aby przedstawić je w tym samym czasie lub przedstawić jedno i drugie. Zajmiemy się tym znacznie więcej w dalszych rozdziałach, ale dobrze jest przedstawić to teraz.

Podsumowanie

W tej części przyjrzeliśmy się podstawom prawdopodobieństwa i będziemy dalej zagłębiać się w tę dziedzinę w następnej części. Do większości naszego myślenia podeszliśmy jako częsty i przedstawiliśmy podstawy eksperymentowania i wykorzystywania prawdopodobieństwa do przewidywania wyników. W następnej części przyjrzymy się bayesowskiemu podejściu do prawdopodobieństwa, a także zbadamy wykorzystanie prawdopodobieństwa do rozwiązywania znacznie bardziej złożonych problemów. Te podstawowe zasady prawdopodobieństwa uwzględnimy w znacznie trudniejszych scenariuszach.

Zaawansowane prawdopodobieństwo

W poprzedniej Części omówiliśmy podstawy prawdopodobieństwa i sposoby zastosowania prostych twierdzeń do złożonych zadań. Krótko mówiąc, prawdopodobieństwo to matematyka modelowania zdarzeń, które mogą wystąpić lub nie. Używamy formuł, aby opisać te zdarzenia, a nawet przyjrzeć się, jak wiele zdarzeń może zachowywać się razem. Tu przyjrzymy się bardziej skomplikowanym twierdzeniom dotyczącym prawdopodobieństwa i sposobom ich wykorzystania jako predykcyjnego. Zaawansowane tematy, takie jak twierdzenie Bayesa i zmienne losowe, dają początek powszechnym algorytmom uczenia maszynowego, takim jak algorytm Naïve Bayes (również omówiony). Skupimy się na niektórych bardziej zaawansowanych tematach teorii prawdopodobieństwa, w tym na następujących tematach:

• Wyczerpujące wydarzenia

• Twierdzenie Bayesa

• Podstawowe zasady przewidywania

• Zmienne losowe

Mamy jeszcze jedną definicję, na którą musimy się przyjrzeć, zanim zaczniemy (ostatnią przed zabawnymi rzeczami, obiecuję). Musimy przyjrzeć się zbiorowo wyczerpującym wydarzeniom.

Zbiorowo wyczerpujące wydarzenia

Gdy dany zestaw dwóch lub więcej zdarzeń musi zajść przynajmniej jedno ze zdarzeń, wówczas mówi się, że taki zestaw zdarzeń jest zbiorowo wyczerpujący. Rozważ następujące przykłady:

• Biorąc pod uwagę zestaw zdarzeń {temperatura < 60, temperatura > 90}, zdarzenia te nie są łącznie wyczerpujące, ponieważ istnieje trzecia opcja, która nie jest podana w tym zestawie zdarzeń: Temperatura może wynosić od 60 do 90. Jednak są wzajemnie wyczerpujące, ponieważ oba nie mogą się zdarzyć w tym samym czasie.

• W rzucie kostką zbiór wydarzeń związanych z rzutem {1, 2, 3, 4, 5 lub 6} jest łącznie wyczerpujący, ponieważ są to jedyne możliwe zdarzenia i przynajmniej jedno z nich musi się wydarzyć.

Ponowne przemyślenia bayesowskie

W ostatniej części rozmawialiśmy bardzo krótko o bayesowskich sposobach myślenia. Krótko mówiąc, mówiąc o Bayesie, mówisz o następujących trzech rzeczach i o tym, jak wszystkie ze sobą współdziałają:

• Wcześniejsza dystrybucja

• Dystrybucja tylna

• Prawdopodobieństwo

Zasadniczo zajmujemy się znalezieniem tyłu. To właśnie chcemy wiedzieć. Innym sposobem wyrażenia bayesowskiego sposobu myślenia jest to, że dane kształtują i aktualizują nasze przekonania. Mamy prawdopodobieństwo a priori, czyli to, co naiwnie myślimy o hipotezie, a potem mamy prawdopodobieństwo a posteriori, czyli to, co myślimy o hipotezie, biorąc pod uwagę pewne dane.

Twierdzenie Bayesa

Twierdzenie Bayesa jest dużym wynikiem wnioskowania bayesowskiego. Zobaczmy, jak to się w ogóle dzieje. Przypomnijmy, że wcześniej zdefiniowaliśmy:

• P(A) = Prawdopodobieństwo wystąpienia zdarzenia A

• P(A|B) = Prawdopodobieństwo wystąpienia A, zakładając, że wystąpiło B

• P(A, B) = Prawdopodobieństwo wystąpienia A i B

• P(A, B) = P(A) * P(B|A)

Ten ostatni punkt można odczytać jako prawdopodobieństwo wystąpienia A i B to prawdopodobieństwo wystąpienia A razy prawdopodobieństwo wystąpienia B, biorąc pod uwagę, że A już wystąpiło.To z tego ostatniego punktu nabiera kształtu twierdzenie Bayesa. Wiemy to:

P(A, B) = P(A) * P(B|A)

P(B, A) = P(B) * P(A|B)

P(A, B) = P(B, A)

Więc:

P(B) * P(A|B) = P(A) * P(B|A)

Dzieląc obie strony przez P(B) otrzymujemy twierdzenie Bayesa, jak pokazano:

P(A,B) = P(A) * P(B|A) / P(B)

Możesz myśleć o twierdzeniu Bayesa w następujący sposób:

• Jest to sposób na przejście z P(A|B) do P(B|A) (jeśli masz tylko jeden)

• Jest to sposób na uzyskanie P(A|B), jeśli już znasz P(A) (bez znajomości B)

Spróbujmy pomyśleć o Bayesie, używając terminów hipoteza i dane. Załóżmy, że H = twoja hipoteza na temat podanych danych, a D = dane, które otrzymujesz. Bayesa można interpretować jako próbę obliczenia P(H|D) (prawdopodobieństwo, że nasza hipoteza jest poprawna, biorąc pod uwagę dostępne dane). Aby skorzystać z naszej terminologii sprzed:

P(H|D) = P(D|H)P(H)/P(D)

• P(H) to prawdopodobieństwo hipotezy przed obserwowaniem danych, nazywane prawdopodobieństwem a priori lub po prostu przed

• P(H|D) to to, co chcemy obliczyć, prawdopodobieństwo hipotezy po zaobserwowaniu danych, zwane a posteriori

• P(D|H) to prawdopodobieństwo danych w ramach danej hipotezy, zwane prawdopodobieństwem

• P(D) to prawdopodobieństwo danych przy dowolnej hipotezie, zwane stałą normalizującą

Ta koncepcja nie odbiega od idei uczenia maszynowego i analityki predykcyjnej. W wielu przypadkach, rozważając analitykę predykcyjną, wykorzystujemy podane dane do przewidywania wyniku. Używając obecnej terminologii, H (naszą hipotezę) można uznać za nasz wynik, a P(H|D) (prawdopodobieństwo, że nasza hipoteza jest prawdziwa, biorąc pod uwagę nasze dane) to inny sposób powiedzenia: jaka jest szansa, że moja hipoteza jest poprawna , biorąc pod uwagę dane przede mną?. Rzućmy okiem na przykład, jak możemy wykorzystać formułę Bayesa w miejscu pracy. Weź pod uwagę, że masz dwie osoby odpowiedzialne za pisanie postów na blogu dla Twojej firmy - Lucy i Avinash. Z poprzednich występów podobało ci się 80% prac Lucy i tylko 50% prac Avinasha. Nowy wpis na blogu przychodzi na

twoje biurko rano, ale autor nie jest wymieniony. Uwielbiasz ten artykuł. A+. Co to jest prawdopodobieństwo, że pochodzi od Avinasha? Każdy bloger bloguje w bardzo podobnym tempie. Zanim zwariujemy, zróbmy to, co zrobiłby każdy doświadczony matematyk (a teraz ty). Zapiszmy wszystkie nasze informacje, jak pokazano:

• H = hipoteza = blog pochodzi z Avinasha

• D = dane = podobał Ci się post na blogu

P(H|D) = szansa, że pochodzi od Avinasha, biorąc pod uwagę, że go pokochałeś

P(D|H) = szansa, że to pokochałeś, biorąc pod uwagę, że pochodzi od Avinasha

P(H) = szansa, że artykuł pochodził od Avinasha

P(D) = szansa, że pokochasz artykuł

Zauważ, że niektóre z tych zmiennych nie mają prawie żadnego sensu bez kontekstu. P(D), prawdopodobieństwo, że spodobałbyś się każdemu artykułowi postawionemu na biurku, jest dziwną koncepcją, ale uwierz mi, w kontekście formuły Bayesa, wkrótce będzie to istotne. Zauważ też, że w dwóch ostatnich punktach nie zakładają niczego innego. P(D) nie zakłada pochodzenia wpisu na blogu; pomyśl o P(D) tak, jakby artykuł został umieszczony na twoim biurku z jakiegoś nieznanego źródła, jaka jest szansa, że ci się spodoba? (ponownie, wiem, że to brzmi dziwnie poza kontekstem). Więc chcemy znać P(H|D). Spróbujmy użyć twierdzenia Bayesa, jak pokazano tutaj:

P(H|D) = P(D|H)P(H)/P(D)

Ale czy znamy liczby po prawej stronie tego równania? Twierdzę, że tak! Zobaczmy tutaj:

• P(H) to prawdopodobieństwo, że dany post na blogu pochodzi z Avinasha. Ponieważ blogerzy piszą w bardzo podobnym tempie, możemy założyć, że jest to 0,5, ponieważ mamy 50/50 szans, że pochodzi od któregokolwiek z blogerów (zauważ, że nie założyłem D, danych, w tym celu).

• P(D|H) to prawdopodobieństwo, że pokochasz post z Avinasha, które wcześniej powiedzieliśmy, że wynosi 50%, czyli 0,5.

• P(D) jest interesujące. To jest szansa, że ogólnie pokochasz artykuł. Oznacza to, że musimy wziąć pod uwagę scenariusz, jeśli post pochodził od Lucy czy Avinasha. Teraz, jeśli hipoteza tworzy zestaw, możemy użyć naszych praw prawdopodobieństwa, jak wspomniano w poprzednim rozdziale. Zestaw powstaje, gdy zestaw hipotez jest zarówno zbiorowo wyczerpujący, jak i wzajemnie się wykluczający. Mówiąc potocznie, w zestawie wydarzeń może wystąpić dokładnie jedna i tylko jedna hipoteza. W naszym przypadku dwie hipotezy są takie, że artykuł pochodzi od Lucy lub że artykuł pochodzi od Avinasha. Jest to zdecydowanie pakiet z następujących powodów:

- Przynajmniej jeden z nich to napisał

- Co najwyżej jeden z nich to napisał

- Dlatego napisał to dokładnie jeden z nich

Mając pakiet, możemy skorzystać z naszych reguł mnożenia i dodawania w następujący sposób:

$$D = (From\ Avinash\ AND\ loved\ it) \quad OR \quad (From\ Lucy\ AND\ loved\ it)$$

$$P(D) = P(Loved\ AND\ from\ Avinash) \quad OR \quad P(Loved\ AND\ from\ Lucy)$$

$$P(D) = P(From\ Avinash)\,P(Loved \mid form\ Avinash) + P(from\ Lucy)\,P(Loved \mid from\ Lucy)$$

$$P(D) = .5(.5) + .5(.8) = \mathbf{.65}$$

Uff! Tak trzymać. Teraz możemy zakończyć nasze równanie, jak pokazano:

P(H|D) = P(D|H)P(H)/P(D)

P(H|D) = 5 * 0,5/0,65 = 0.38

Oznacza to, że istnieje 38% szans, że ten artykuł pochodzi od Avinasha. Co ciekawe, P(H) = 0,5 i P(H|D) = 0,38. Oznacza to, że bez jakichkolwiek danych szansa, że post na blogu pochodzi od Avinasha, była rzutem monetą, czyli 50/50. Biorąc pod uwagę pewne dane (Twoje przemyślenia na temat artykułu), zaktualizowaliśmy nasze przekonania dotyczące hipotezy i faktycznie obniżyliśmy szansę. O to właśnie chodzi w myśleniu bayesowskim — aktualizowanie naszych wcześniejszych przekonań na temat czegoś z wcześniejszego założenia, biorąc pod uwagę nowe dane na ten temat.

Więcej zastosowań twierdzenia Bayesa

Twierdzenie Bayesa pojawia się w wielu aplikacjach, zwykle wtedy, gdy musimy podejmować szybkie decyzje na podstawie danych i prawdopodobieństwa. Większość silników rekomendacji, takich jak Netflix, korzysta z niektórych elementów aktualizacji Bayesa. A jeśli zastanowisz się, dlaczego tak może być, ma to sens. Załóżmy, że w naszym uproszczonym świecie Netflix ma do wyboru tylko 10 kategorii. Załóżmy teraz, że przy braku danych szansa użytkownika na polubienie filmu komediowego z 10 kategorii wynosi 10% (tylko 1/10). Ok, teraz załóżmy, że użytkownik przyznał kilku filmom komediowym 5/5 gwiazdek. Teraz, gdy Netflix zastanawia się, jakie jest prawdopodobieństwo, że użytkownik polubi kolejną komedię, prawdopodobieństwo, że może polubić komedię, P(H|D), będzie większe niż losowe przypuszczenie wynoszące 10%! Wypróbujmy więcej przykładów zastosowania twierdzenia Bayesa przy użyciu większej ilości danych. Tym razem, zróbmy trochę bardziej twardzieli.

Przykład - Titanic

Bardzo znany zbiór danych obejmuje przyjrzenie się ocalałym z zatonięcia Titanica w 1912 roku. Użyjemy rachunku prawdopodobieństwa, aby dowiedzieć się, czy istnieją jakieś cechy demograficzne, które wykazywały związek z przeżyciem pasażerów. Przede wszystkim jesteśmy ciekawi, czy możemy wyizolować jakiekolwiek cechy naszego zbioru danych, które mogłyby nam powiedzieć więcej o typach ludzi, którzy prawdopodobnie przeżyli tę katastrofę. Najpierw przeczytajmy dane, jak pokazano tutaj:

```
titanic = pd.read_csv(data/titanic.csv')#read in a csv

titanic = titanic[['Sex', 'Survived']] #the Sex and Survived column

titanic.head()
```

	Sex	Survived
0	male	no
1	female	yes
2	female	yes
3	female	yes
4	male	no

W powyższej tabeli każdy wiersz reprezentuje jednego pasażera na statku i na razie przyglądamy się dwóm specyficznym cechom: płci osobnika i czy przeżył zatonięcie. Na przykład pierwszy wiersz reprezentuje mężczyznę, który nie przeżył, podczas gdy czwarty wiersz (z indeksem 3, pamiętaj, jak lista indeksów Pythona) reprezentuje kobietę, która przeżyła. Zacznijmy od podstaw. Zacznijmy od obliczenia prawdopodobieństwa, że jakakolwiek osoba na statku przeżyła, niezależnie od płci. Aby to zrobić, policzmy liczbę tak w kolumnie Przeżyli i podzielmy tę liczbę przez całkowitą liczbę wierszy, jak pokazano tutaj:

num_rows = float(titanic.shape[0]) # == 891 rows

p_survived = (titanic.Survived=="yes").sum() / num_rows # == .38

p_notsurvived = 1 - p_survived # == .61

Zauważ, że musiałem tylko obliczyć P (przeżył) i użyłem prawa sprzężonych prawdopodobieństw do obliczenia P (zmarł), ponieważ te dwa zdarzenia są komplementarne. Teraz obliczmy prawdopodobieństwo, że pojedynczy pasażer jest mężczyzną lub kobietą:

p_male = (titanic.Sex=="male").sum() / num_rows # == .65

p_female = 1 - p_male # == .35

Teraz zadajmy sobie pytanie, czy posiadanie określonej płci wpłynęło na przeżywalność? W tym celu możemy oszacować P (przeżył | kobieta) lub szansę, że ktoś przeżył, biorąc pod uwagę, że był kobietą. W tym celu musimy podzielić liczbę kobiet, które przeżyły, przez całkowitą liczbę kobiet, jak pokazano tutaj:

$$P(Survived \mid Female) = \frac{P(Female\ AND\ Survived)}{P(Female)}$$

number_of_women = titanic[titanic.Sex=='female'].shape[0] # == 314

women_who_lived = titanic[(titanic.Sex=='female') & (titanic.

Survived=='yes')].shape[0] # == 233

p_survived_given_woman = women_who_lived / float(number_of_women)

p_survived_given_woman # == .74

To całkiem duża różnica. Wydaje się, że płeć odgrywa dużą rolę w tym zbiorze danych.

Przykład - studia medyczne

Klasycznym zastosowaniem twierdzenia Bayesa jest interpretacja badań medycznych. Rutynowe testy na nielegalne zażywanie narkotyków są coraz powszechniejsze w miejscach pracy i szkołach. Firmy, które przeprowadzają te testy, utrzymują, że testy mają wysoką czułość, co oznacza, że prawdopodobnie dadzą pozytywny wynik, jeśli w ich systemie znajdują się leki. Twierdzą, że te testy są również bardzo specyficzne, co oznacza, że mogą dać wynik negatywny, jeśli nie ma leków. Średnio załóżmy, że czułość powszechnych testów narkotykowych wynosi około 60%, a swoistość około 99%. Oznacza to, że jeśli pracownik zażywa narkotyki, test ma 60% szans na wynik pozytywny, natomiast jeśli pracownik nie zażywa narkotyków, test ma 99% szans na wynik negatywny. Załóżmy teraz, że te testy są stosowane do siły roboczej, w której faktyczny wskaźnik używania narkotyków wynosi 5%. Prawdziwe pytanie dotyczy osób, które uzyskały pozytywny wynik testu, ilu faktycznie używa narkotyków? W kategoriach bayesowskich chcemy obliczyć prawdopodobieństwo zażywania narkotyków po pozytywnym teście.

Niech D = zdarzenie, w którym narkotyki są w użyciu

Niech E = zdarzenie, w którym test jest pozytywny

Niech N = przypadek, w którym narkotyki NIE są używane

Poszukujemy P(D|E).

Korzystając z twierdzenia Bayesa, możemy go ekstrapolować w następujący sposób:

P(D|E) = P(E}D)(PD)/P(E)

Poprzednie, P(D) to prawdopodobieństwo zażycia narkotyku, zanim zobaczymy wynik testu, który wynosi 5%. Prawdopodobieństwo, P(E|D), to prawdopodobieństwo pozytywnego wyniku testu przy założeniu zażywania narkotyków, które jest tym samym, co czułość testu. Stała normalizująca P(E) jest nieco trudniejsza. Musimy wziąć pod uwagę dwie rzeczy: P(E i D) oraz P(E i N). Zasadniczo musimy założyć, że test może być niepoprawny, gdy użytkownik nie używa narkotyków. Sprawdź następujące równania:

P(E) = P(E i D) lub P(E i N)

P(E) = P(D)P(E|D) + P(N)P(E|N)

P(E) = 0,5 * 0,6 + 0,95 * 0,1

P(E) = 0,0395

Tak więc nasze pierwotne równanie wygląda następująco:

P(D|E) = 0,6 * 0,5 / 0,0395

P(D}E) = 0,76

Oznacza to, że spośród osób, które uzyskały pozytywny wynik testu na zażywanie narkotyków, około jedna czwarta jest niewinna!

Zmienne losowe

Zmienna losowa wykorzystuje rzeczywiste wartości liczbowe do opisania zdarzenia probabilistycznego. W naszej poprzedniej pracy ze zmiennymi (zarówno matematycznymi, jak i programistycznymi) byliśmy przyzwyczajeni do tego, że zmienna przyjmuje określoną wartość. Na przykład, możemy mieć trójkąt, w którym otrzymamy zmienną h dla przeciwprostokątnej i musimy obliczyć długość przeciwprostokątnej. Możemy też mieć w Pythonie:

x = 5

Obie te zmienne są równe jednej wartości na raz. W zmiennej losowej podlegamy losowości, co oznacza, że wartości naszych zmiennych są, no właśnie, zmiennymi! W zależności od środowiska mogą przybierać różne wartości. Zmienna losowa nadal, jak pokazano wcześniej, przechowuje wartość. Główną różnicą między zmiennymi, jakie widzieliśmy, a zmienną losową jest fakt, że wartość zmiennej losowej może się zmieniać w zależności od sytuacji. Jeśli jednak zmienna losowa może mieć wiele wartości, w jaki sposób możemy je wszystkie śledzić? Każda wartość, jaką może przyjąć zmienna losowa, jest powiązana z wartością procentową. Dla każdej wartości, jaką może przyjąć zmienna losowa, istnieje jedno prawdopodobieństwo, że zmienna będzie tą wartością. Za pomocą zmiennej losowej możemy również uzyskać nasz rozkład prawdopodobieństwa losowej zmiennej, która podaje możliwe wartości zmiennej i ich prawdopodobieństwa. Napisane, zwykle używamy pojedynczych wielkich liter (głównie konkretnej litery X) do oznaczenia zmiennych losowych. Na przykład możemy mieć:

• X = wynik rzutu kostką

• Y = przychód osiągnięty przez firmę w tym roku

• Z = wynik kandydata w quizie z kodowania rozmowy kwalifikacyjnej (0-100%)

W efekcie zmienna losowa to funkcja, która odwzorowuje wartości z przestrzeni próbki zdarzenia (zbiór wszystkich możliwych wyników) na wartość prawdopodobieństwa (między 0 a 1). Pomyśl o wydarzeniu jako wyrażonym w następujący sposób:

f(zdarzenie) = prawdopodobieństwo

Przypisze prawdopodobieństwo każdej indywidualnej opcji. Istnieją dwa główne typy zmiennych losowych: dyskretne i ciągłe.

Dyskretne zmienne losowe

Dyskretna zmienna losowa przyjmuje tylko policzalną liczbę możliwych wartości. Na przykład wynik rzutu kostką, jak pokazano tutaj:

X = wynik pojedynczego rzutu kostką

Value	$X=1$	$X=2$	$X=3$	$X=4$	$X=5$	$X=6$
Probability	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$

Zwróć uwagę, jak używam dużej litery X, aby zdefiniować zmienną losową. To powszechna praktyka. Zwróć także uwagę, w jaki sposób zmienna losowa odwzorowuje prawdopodobieństwo dla każdego indywidualnego wyniku. Zmienne losowe mają wiele właściwości, z których dwie to ich wartość oczekiwana i wariancja. Użyjemy funkcji masy prawdopodobieństwa (PMF) do opisania dyskretnej zmiennej losowej. Przybierają wygląd:

P(X = x) = PMF

Tak więc, dla rzutu kostką, P(X = 1) = 1/6 i P(X = 5) = 1/6.

Rozważ następujące przykłady zmiennych dyskretnych:

• Prawdopodobny wynik pytania ankietowego (na przykład w skali 1-10)

• Czy dyrektor generalny zrezygnuje w ciągu roku (prawda lub fałsz)

Oczekiwana wartość zmiennej losowej określa średnią wartość długiego przebiegu powtarzanych próbek zmiennej losowej. Jest to czasami nazywane średnią zmiennej. Na przykład zapoznaj się z następującym kodem Pythona, który definiuje zmienną losową rzutu kostką:

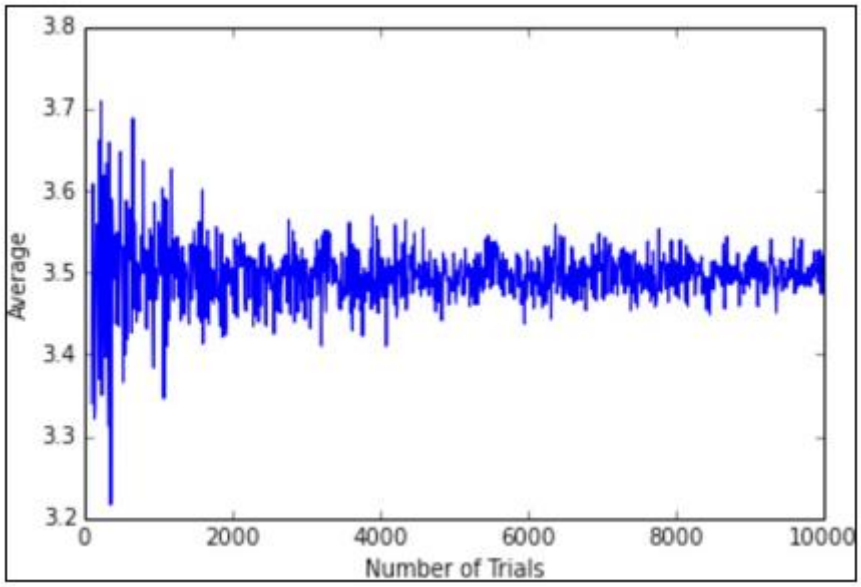
```
import random

def random_variable_of_dice_roll():

return random.randint(1, 7) # a range of (1,7) # includes 1, 2, 3,

4, 5, 6, but NOT 7
```

Ta funkcja wywoła zmienną losową i wyjdzie z odpowiedzią. Rzućmy 100 kostkami i uśrednimy wynik w następujący sposób:

```
trials = []

num_trials = 100

for trial in range(num_trials):

trials.append( random_variable_of_dice_roll() )

print sum(trials)/float(num_trials) # == 3.77
```

Tak więc wzięcie 100 rzutów kostką i uśrednienie ich daje nam wartość 3,77! Wypróbujmy to z szeroką gamą numerów próbnych, jak pokazano tutaj:

```
num_trials = range(100,10000, 10)

avgs = []

for num_trial in num_trials:

trials = []

for trial in range(1,num_trial):

trials.append( random_variable_of_dice_roll() )

avgs.append(sum(trials)/float(num_trial))

plt.plot(num_trials, avgs)

plt.xlabel('Number of Trials')

plt.ylabel("Average")
```



Powyższy wykres przedstawia średni rzut kostką, gdy przyglądamy się coraz większej liczbie rzutów kośćmi. Widzimy, że średni rzut kostką szybko zbliża się do 3,5. Jeśli spojrzymy w lewo na wykresie, zobaczymy, że jeśli rzucimy kostką tylko około 100 razy, nie gwarantujemy, że przeciętny rzut kostką wyniesie 3,5. Jednakże, jeśli rzucamy 10,000 kostkami jedna po drugiej, widzimy, że bardzo prawdopodobne jest, że średni rzut kostką wyniesie około 3,5. Dla dyskretnej zmiennej losowej możemy również użyć prostego wzoru, pokazanego poniżej, aby obliczyć wartość oczekiwaną:

Wartość oczekiwana = $E[X] = \mu x = \Sigma x_i p_i$

Gdzie x_i jest i-tym wynikiem, a p_i jest i-tym prawdopodobieństwem. Tak więc dla naszego rzutu kostką możemy znaleźć dokładną oczekiwaną wartość w następujący sposób:

$$\frac{1}{6}(1)+\frac{1}{6}(2)+\frac{1}{6}(3)+\frac{1}{6}(4)+\frac{1}{6}(5)+\frac{1}{6}(6)=3.5$$

Powyższy wynik pokazuje nam, że dla każdego rzutu kostką możemy „oczekiwać” rzutu kostką o wartości 3,5. Oczywiście nie ma to sensu, ponieważ nie możemy uzyskać 3,5 na rzucie kostką, ale ma to sens w kontekście wielu rzutów kośćmi. Jeśli rzucisz 10 000 kostkami, średni rzut powinien zbliżyć się do 3,5, jak pokazano na powyższym wykresie i kodzie. Średnia wartości oczekiwanej zmiennej losowej na ogół nie wystarcza, aby zrozumieć pełną ideę zmiennej. Z tego powodu wprowadzamy nową koncepcję, zwaną wariancją. Wariancja zmiennej losowej reprezentuje rozrzut zmiennej. Określa ilościowo zmienność wartości oczekiwanej. Wzór na wariancję dyskretnej zmiennej losowej wyraża się następująco:

$$\text{Variance} = V[X] = \sigma_X^2 = \sum (x_i - \mu_X)^2 p_i$$

Gdzie x_i i p_i reprezentują te same wartości co poprzednio i reprezentuje oczekiwaną wartość zmiennej. W tym wzorze wspomniałem również o sigma X. Sigma, w tym przypadku, to odchylenie standardowe, które definiuje się po prostu jako pierwiastek kwadratowy z wariancji. Spójrzmy na bardziej skomplikowany przykład dyskretnej zmiennej losowej. Wariancję można traktować jako miernik

dawania lub brania. Jeśli powiem, że możesz spodziewać się wygrania 100 $ z rozdania w pokera, możesz być bardzo szczęśliwy. Jeśli dołączę do tego stwierdzenia dodatkowy szczegół, że możesz wygrać 100 USD, dać lub wziąć 80 USD, masz teraz do czynienia z wieloma oczekiwaniami, co może być frustrujące i może sprawić, że gracz niechętny ryzyku będzie bardziej ostrożny przed dołączeniem do gry . Zwykle możemy powiedzieć, że mamy wartość oczekiwaną, podając lub przyjmując odchylenie standardowe. Weź pod uwagę, że Twój zespół mierzy sukces nowego produktu w skali Likerta, czyli w jednej z pięciu kategorii, gdzie wartość 0 oznacza całkowitą porażkę, a 4 oznacza wielki sukces. Szacują, że nowy projekt ma następujące szanse powodzenia na podstawie testów użytkowników i wstępnych wyników działania produktu. Najpierw musimy zdefiniować naszą zmienną losową. Niech zmienna losowa X reprezentuje sukces naszego produktu. X jest rzeczywiście dyskretną zmienną losową, ponieważ zmienna X może przyjąć tylko jedną z pięciu opcji: 0, 1, 2, 3 lub 4. Poniżej znajduje się rozkład prawdopodobieństwa naszej zmiennej losowej X. Zauważ, że mamy kolumnę dla każdego potencjalnego wyniku X, a poniżej każdego wyniku mamy prawdopodobieństwo, że ten konkretny wynik zostanie osiągnięty:

Value	$X = 0$	$X = 1$	$X = 2$	$X = 3$	$X = 4$
Probability	0.02	0.07	0.25	0.4	0.26

Na przykład projekt ma 2% szansy na całkowitą porażkę i 26% szansy na wielki sukces! Naszą oczekiwaną wartość możemy obliczyć w następujący sposób:

E[X] = 0(0,02) + 1(0,07) + 2(0,25) + 3(0,4) + 4(0,26) = 2,81

Ta liczba oznacza, że menedżer może liczyć na sukces około 2,81 z tego projektu. Sama liczba ta nie jest zbyt użyteczna. Być może, mając do wyboru kilka produktów, oczekiwana wartość może być sposobem na porównanie potencjalnych sukcesów kilku produktów. Jednak w tym przypadku, gdy mamy do oceny tylko jeden produkt, będziemy potrzebować więcej. Sprawdźmy teraz wariancję, jak pokazano tutaj:

Wariancja=V[X]=σX2 = (x_i −μX)2pi = (0 − 2,81)2(0,02) + (1 − 2,81)2(0,07)+ (2 − 2,81)2(0,25) + (3 − 2,81)2(0,4) + (4 −2,81)2(0,26) = 0,93

Teraz, gdy mamy już zarówno odchylenie standardowe, jak i oczekiwaną wartość punktacji projektu, spróbujmy podsumować nasze wyniki. Można powiedzieć, że nasz projekt będzie miał oczekiwany wynik 2,81 plus minus 0,96, co oznacza, że można spodziewać się czegoś między 1,85 a 3,77. Tak więc jednym ze sposobów, w jaki możemy zająć się tym projektem, jest to, że prawdopodobnie będzie miał on wskaźnik sukcesu wynoszący 2,81, co daje lub bierze około punktu. Możesz pomyśleć, wow, Sinan, więc w najlepszym przypadku projekt będzie miał 3,8, a w najgorszym 1,8?. Nie do końca. Może być lepszy niż 4 i może być też gorszy niż 1,8. Aby zrobić ten jeden krok dalej obliczmy, co następuje:

P(X >= 3)

Najpierw poświęć chwilę i przekonaj się, że możesz przeczytać sobie tę formułę. O co pytam, gdy proszę o P(X >= 3)? Szczerze, poświęć chwilę i zastanów się. P(X >= 3) to prawdopodobieństwo, że nasza zmienna losowa przyjmie wartość co najmniej tak dużą jak 3. Innymi słowy, jaka jest szansa, że nasz produkt będzie miał ocenę sukcesu 3 lub wyższą? Aby to obliczyć, możemy obliczyć:

P(X >= 3) = P(X = 3) + P(X = 4) = 0,66 = 66%

Oznacza to, że mamy 66% szans, że nasz produkt oceni się na 3 lub 4. Innym sposobem obliczenia tego będzie sposób sprzężony, jak pokazano tutaj:

P(X >= 3) = 1 – P(X < 3)

Ponownie poświęć chwilę, aby przekonać się, że ta formuła się sprawdza. Twierdzę, że znalezienie prawdopodobieństwa, że produkt otrzyma ocenę co najmniej 3, jest takie samo jak 1 minus prawdopodobieństwo, że produkt otrzyma ocenę poniżej 3. Jeśli to prawda, to dwa zdarzenia (X >=3 a X < 3) muszą się wzajemnie uzupełniać. To oczywiście prawda! Produkt może mieć jedną z dwóch następujących opcji:

• mieć ocenę 3 lub wyższą

• Mieć ocenę poniżej 3

Sprawdźmy naszą matematykę:

P(X < 3) = P(X = 0) + P(X = 1) + P(X = 2) = 0,02 + 0,07 + 0,25 = 0,034

1 – P(X < 3) = 1 - 0,34 = 0,66 = P(x >= 3)

To się sprawdza!

Rodzaje dyskretnych zmiennych losowych

Możemy lepiej zrozumieć, jak działają zmienne losowe w praktyce, przyglądając się określonym typom zmiennych losowych. Te specyficzne typy zmiennych losowych modelują różne rodzaje sytuacji i kończą się ujawnieniem znacznie prostszych obliczeń dla bardzo złożonego modelowania zdarzeń.

Zmienne losowe dwumianowe

Pierwszy typ dyskretnej zmiennej losowej, któremu się przyjrzymy, nazywa się zmienną losową dwumianową. W przypadku dwumianowej zmiennej losowej patrzymy na ustawienie, w którym pojedyncze zdarzenie zdarza się w kółko i próbujemy policzyć, ile razy wynik jest dodatni. Zanim zrozumiemy samą zmienną losową, musimy przyjrzeć się warunkom, w jakich jest ona nawet odpowiednia.

Ustawienie dwumianowe ma następujące cztery warunki:

• Możliwe wyniki to sukces lub porażka

• Wyniki badań nie mogą wpływać na wyniki innego badania

• Ustalono liczbę prób (stała wielkość próby)

• Szansa powodzenia każdej próby musi zawsze wynosić p

Zmienna losowa dwumianowa to dyskretna zmienna losowa X, która zlicza liczbę sukcesów w układzie dwumianowym. Parametry to n = liczba prób i p = szansa powodzenia każdej próby.

Przykład - spotkania fundraisingowe:

Start-up bierze 20 spotkań VC, aby sfinansować i zliczyć otrzymywane oferty. Funkcja masy prawdopodobieństwa (PMF) dla dwumianowej zmiennej losowej jest następująca:

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

Tu

$$\binom{n}{k} = \text{the binomial coefficient} = \frac{n!}{(n-k)!k!}.$$

Przykład - otwarcia restauracji

Nowa restauracja w mieście ma 20% szans na przetrwanie pierwszego roku. Jeśli w tym roku zostanie otwartych 14 restauracji, sprawdź prawdopodobieństwo, że dokładnie cztery restauracje przetrwają pierwszy rok otwarcia dla publiczności. Najpierw powinniśmy udowodnić, że jest to ustawienie dwumianowe:

• Możliwe wyniki to sukces lub porażka (restauracje albo przetrwają, albo nie)

• Wyniki prób nie mogą wpływać na wyniki innej próby (załóżmy, że otwarcie jednej restauracji nie wpływa na otwarcie i przetrwanie innej restauracji)

• Ustalono liczbę prób (otworzono 14 restauracji)

• Szansa powodzenia każdej próby musi zawsze wynosić p (zakładamy, że zawsze wynosi 20%)

Tutaj mamy nasze dwa parametry n = 14 i p = .2. Możemy więc teraz wstawić te liczby do naszego wzoru dwumianowego, jak pokazano tutaj:

$$P(X=4) = \binom{14}{4}.2^4.8^{10} = .17$$

Mamy więc 17% szans, że dokładnie 4 z tych restauracji będą otwarte za rok

Przykład - grupy krwi

Para ma 25% szans na urodzenie dziecka z grupą krwi O. Jaka jest szansa, że 3 z ich 5 dzieci ma krew typu O?

Niech X = liczba dzieci z krwią typu O z n = 5 i p = 0,25, jak pokazano tutaj:

P(X = 3) = 5 0,253 (0,75)5−3 = 10(0,25)3(0,75)2 = 0,087

Możemy obliczyć to prawdopodobieństwo dla wartości 0, 1, 2, 3, 4 i 5, aby uzyskać sens rozkładu prawdopodobieństwa:

value x_i	0	1	2	3	4	5
Probability	0.23730	0.39551	0.26367	0.08789	0.01465	0.00098

Stąd możemy obliczyć wartość oczekiwaną i wariancję tej zmiennej:

$$\text{Expected value} = E[X] = \mu_X = \sum x_i p_i = 1.25$$

$$\text{Variance} = V[X] = \sigma_X^2 = \sum (x_i - \mu_X)^2 p_i = 0.9375$$

Tak więc ta rodzina może spodziewać się prawdopodobnie 1 lub 2 dzieci z grupą krwi O! A jeśli chcemy poznać prawdopodobieństwo, że przynajmniej troje ich dzieci ma grupę krwi 0? Aby poznać prawdopodobieństwo, że co najmniej troje ich dzieci ma krew typu 0, możemy użyć następującego wzoru na dyskretne zmienne losowe:

$$P(X \geq 3) = P(X = 5) + P(X = 4) + P(X = 3)$$

$$= .00098 + .01465 + .08789 = 0.103$$

Tak więc istnieje około 10% szans, że troje ich dzieci ma grupę 0 krwi.

Skróty do dwumianowej wartości oczekiwanej i wariancji

Zmienne losowe dwumianowe mają specjalne obliczenia dla dokładnych wartości oczekiwanych wartości i wariancji. Jeśli X jest dwumianową zmienną losową, to:

E(X) = np

V(X) = np(1 − p)

W naszym poprzednim przykładzie możemy użyć następujących formuł, aby obliczyć dokładną wartość oczekiwaną i wariancję:

• E(X) = 0,25(5) = 1,25

• V(X) = 1,25(,75) = 0,9375

Zmienna losowa dwumianowa to dyskretna zmienna losowa, która zlicza liczbę sukcesów w układzie dwumianowym. Jest używany w wielu różnych eksperymentach opartych na danych, takich jak liczenie osób, które zarejestrują się w witrynie, mając szansę na konwersję, lub nawet, na prostym poziomie, przewidywanie ruchów cen akcji przy szansie na spadek (nie martw się; będziemy później stosować znacznie bardziej wyrafinowane modele do przewidywania rynku akcji).

Geometryczne zmienne losowe

Druga dyskretna zmienna losowa, której się przyjrzymy, nazywa się geometryczną zmienną losową. W rzeczywistości jest dość podobna do dwumianowej zmiennej losowej, ponieważ zajmujemy się ustawieniem, w którym pojedyncze zdarzenie ma miejsce w kółko. Jednak w przypadku ustawienia geometrycznego główną różnicą jest to, że nie ustalamy wielkości próbki. Nie idziemy na dokładnie 20 spotkań VC jako start-up, ani nie mamy dokładnie 5 dzieci. Zamiast tego w ustawieniu geometrycznym modelujemy liczbę prób, które będziemy musieli zobaczyć, zanim osiągniemy choćby jeden sukces. W szczególności ustawienie geometryczne ma następujące cztery warunki:

• Możliwe wyniki to sukces lub porażka

• Wyniki badań nie mogą wpływać na wyniki innego badania

• Nie ustalono liczby prób

• Szansa powodzenia każdej próby musi zawsze wynosić p

Zauważ, że są to dokładnie te same warunki, co zmienna dwumianowa, z wyjątkiem trzeciego warunku. Geometryczna zmienna losowa to dyskretna zmienna losowa X, która zlicza liczbę prób potrzebną do osiągnięcia jednego sukcesu. Parametry to p = szansa powodzenia każdej próby i (1 − p) = szansa niepowodzenia każdej próby. Aby przekształcić poprzednie przykłady dwumianowe w przykłady geometryczne, możemy wykonać następujące czynności:

• Policz liczbę spotkań VC, które musi odbyć start-up, aby uzyskać pierwsze tak

• Policz liczbę rzutów monetą potrzebną do uzyskania orła (tak, wiem, że to nudne, ale to dobry przykład!)

Wzór na PMF jest następujący:

P(X = x) = (1−p)[x−1]p

Zarówno ustawienia dwumianowe, jak i geometryczne wiążą się z wynikami, które są albo sukcesami, albo porażkami. Duża różnica polega na tym, że dwumianowe zmienne losowe mają ustaloną liczbę prób, oznaczoną jako n. Geometryczne zmienne losowe nie mają stałej liczby prób. Zamiast tego geometryczne zmienne losowe modelują liczbę próbek potrzebnych do uzyskania pierwszej udanej próby, niezależnie od tego, jaki sukces może oznaczać w tych warunkach eksperymentalnych.

Przykład – pogoda

Jest 34% szans na deszcz w dowolnym dniu kwietnia. Znajdź prawdopodobieństwo, że pierwszy dzień deszczu w kwietniu nastąpi czwartego kwietnia.

Niech X = liczba dni do deszczu (sukces) przy p = 0,34 i (1 − p) = 0,66

Tak więc P(X = 8) = (0,66)8−1 (0,34)

= (0,66)7 (0,34)

= 0,01855

Prawdopodobieństwo, że do czwartego kwietnia będzie padać, przedstawia się następująco:

$$P(X \le 4) = P(1) + P(2) + P(3) + P(4) =$$

$$= .34 + .22 + .14 + .1 = .8$$

Tak więc istnieje 80% szans na to, że pierwszy deszcz miesiąca wystąpi w ciągu pierwszych czterech dni.

Skróty do geometrycznej wartości oczekiwanej i wariancji

Geometryczne zmienne losowe mają również specjalne obliczenia dla dokładnych wartości oczekiwanych wartości i wariancji. Jeśli X jest geometryczną zmienną losową, to

E(X) = 1/p

V(X) = (1-p)/p2

str 124

Zmienna losowa Poissona

Trzecim i ostatnim konkretnym przykładem dyskretnej zmiennej losowej jest zmienna losowa Poissona. Aby zrozumieć, dlaczego potrzebujemy tej zmiennej losowej, wyobraź sobie, że zdarzenie, które chcemy modelować, ma małe prawdopodobieństwo wystąpienia i że chcemy policzyć, ile razy zdarzenie wystąpiło w określonym przedziale czasowym. Jeśli mamy wyobrażenie o średniej liczbie wystąpień &mul, w określonym okresie czasu, podanej z poprzednich wystąpień, to zmienna losowa Poissona, oznaczona przez X = Poi(μ), zlicza całkowitą liczbę wystąpień wydarzenie w danym okresie. Innymi słowy, rozkład Poissona jest dyskretnym rozkładem prawdopodobieństwa, który zlicza

liczbę zdarzeń występujących w danym przedziale czasu. Rozważ następujące przykłady zmiennych losowych Poissona:

• Znalezienie prawdopodobieństwa posiadania określonej liczby odwiedzających witrynę w ciągu godziny, znając wcześniejsze wyniki witryny

• Oszacowanie liczby wypadków samochodowych na skrzyżowaniu na podstawie wcześniejszych raportów policyjnych

Jeśli X = liczba zdarzeń w danym przedziale, a średnia liczba zdarzeń na przedział to λ liczby, to prawdopodobieństwo zaobserwowania x zdarzeń w danym przedziale określa wzór:

$$P(X = x) = \frac{e^{-\lambda}\lambda^{x}}{x!}$$

Tutaj e = stała Eulera (2,718…).

Przykład - call center

Liczba połączeń przychodzących do call center jest zgodna z rozkładem Poissona w tempie 5 połączeń na godzinę. Jakie jest prawdopodobieństwo, że dokładnie sześć telefonów pojawi się między 22:00 a 23:00?

Aby skonfigurować ten przykład, wypiszmy podane przez nas informacje. Niech X będzie liczbą połączeń przychodzących między 22:00 a 23:00. To jest nasza zmienna losowa Poissona ze średnią λ = 5. Średnia wynosi 5, ponieważ używamy 5 jako naszej poprzedniej oczekiwanej wartości liczby połączeń przychodzących w tym czasie. Ta liczba mogła pochodzić z cennej pracy nad oszacowaniem liczby połączeń przychodzących co godzinę, a konkretnie po godzinie 22:00. Główną ideą jest to, że mamy pewne pojęcie o tym, ile połączeń powinno nadejść, a następnie wykorzystujemy te informacje do stworzenia naszej zmiennej losowej Poissona i używamy jej do przewidywania. Kontynuując nasz przykład, mamy:

P(X = 6) = = 0,146

Oznacza to, że istnieje około 14,6% szans na to, że dokładnie sześć połączeń nadejdzie między 22:00 a 23:00.

Skróty do oczekiwanej wartości Poissona i wariancji

Zmienne losowe Poissona mają również specjalne obliczenia dla dokładnych wartości oczekiwanych wartości i wariancji. Jeśli X jest zmienną losową Poissona ze średnią, to:

E(X) = λ

V(X) = λ

Jest to interesujące, ponieważ zarówno wartość oczekiwana, jak i wariancja to ta sama liczba, a ta liczba jest po prostu podanym parametrem! Teraz, gdy widzieliśmy już trzy przykłady dyskretnych zmiennych losowych, musimy przyjrzeć się drugiemu typowi zmiennej losowej, zwanej ciągłą zmienną losową.

Ciągłe zmienne losowe

Całkowite przełączanie biegów, w przeciwieństwie do dyskretnej zmiennej losowej, ciągła zmienna losowa może przyjmować nieskończoną liczbę możliwych wartości, a nie tylko kilka policzalnych. Nazywamy funkcje, które opisują krzywe gęstości rozkładu zamiast funkcji mas prawdopodobieństwa. Rozważ następujące przykłady zmiennych ciągłych:

• Długość rozmowy telefonicznej z przedstawicielem handlowym (nie liczba rozmów)

• Rzeczywista ilość oleju w beczce oznaczona 20 galonami (nie liczba beczek z olejem)

Jeśli X jest ciągłą zmienną losową, to istnieje funkcja f(x), taka, że dla dowolnych stałych a i b:

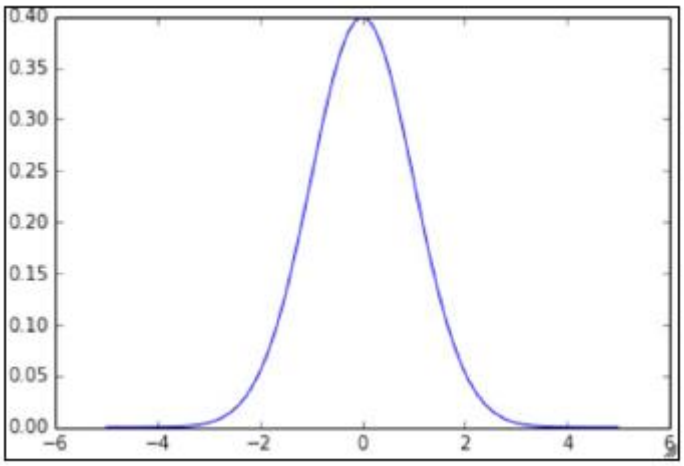
$$P(a \leq X \leq b) = \int_a^b f(x)dx$$

Poprzednia funkcja f(x) jest znana jako funkcja gęstości prawdopodobieństwa (PDF). PDF to wersja PMF z ciągłą zmienną losową dla dyskretnych zmiennych losowych. Najważniejszym rozkładem ciągłym jest standardowy rozkład normalny. Bez wątpienia albo słyszałeś o normalnym rozkładzie, albo miałeś z nim do czynienia. Idea stojąca za tym jest dość prosta. Plik PDF tej dystrybucji wygląda następująco:

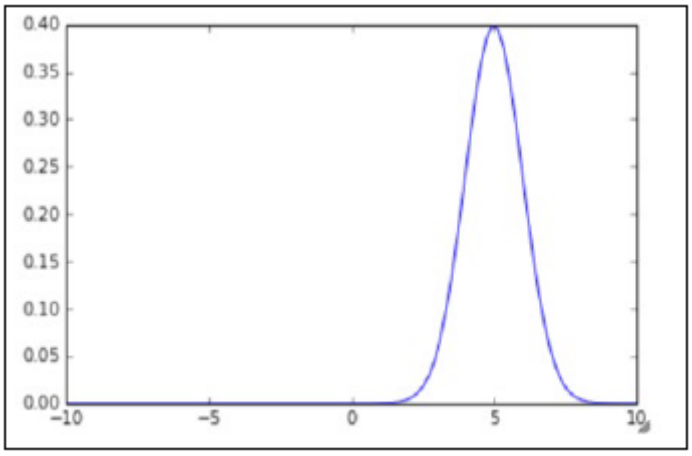
$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(x-\mu)^2}{2\sigma^2}}$$

Tu μ oznacza średnią zmiennej a σ odchylenie standardowe. Może to wyglądać na mylące, ale narysujmy to w Pythonie ze średnią równą 0 i odchyleniem standardowym równym 1, jak pokazano tutaj:

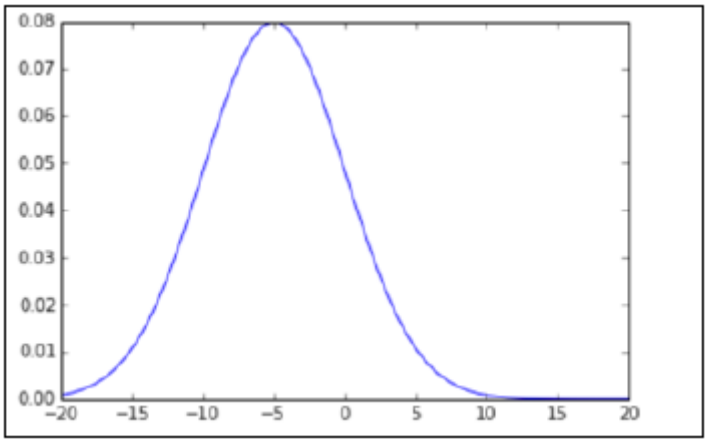
```
def normal_pdf(x, mu = 0, sigma = 1):

return (1./np.sqrt(2*3.14 * sigma**2)) * 2.718**(-(x-mu)**2 / (2.

* sigma**2))

x_values = np.linspace(-5,5,100)

y_values = [normal_pdf(x) for x in x_values]

plt.plot(x_values, y_values)
```



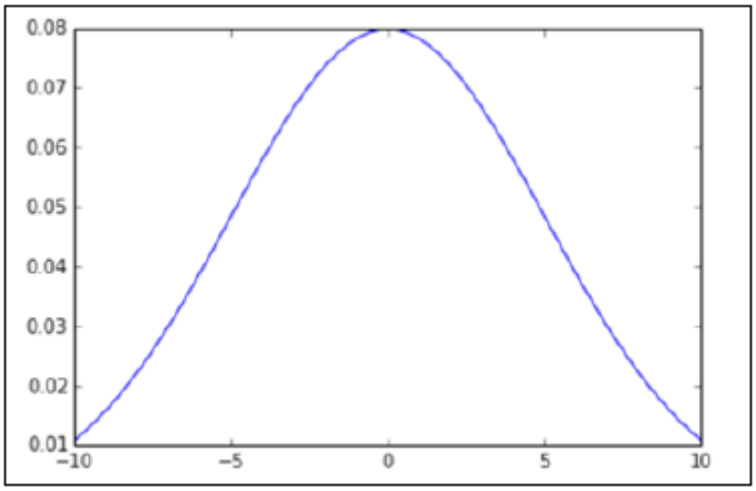
Co daje początek aż nazbyt znanej krzywej dzwonowej. Zauważ, że wykres jest symetryczny wokół linii x = 0. Spróbujmy zmienić niektóre parametry. Najpierw spróbujmy z μ = 5 :



Następnie spróbujmy z wartością σ = 5 :



Na koniec spróbujemy z wartościami μ = 5 i σ = 5 :



Na wszystkich wykresach mamy standardowy kształt dzwonka, który wszyscy znamy, ale gdy zmieniamy nasze parametry, widzimy, że dzwonek może stać się cieńszy, grubszy lub przesunąć się od lewej do prawej. W następnych rozdziałach, które skupiają się na statystyce, w dużo większym stopniu wykorzystamy rozkład normalny, ponieważ odnosi się on do myślenia statystycznego.

Podsumowanie

Prawdopodobieństwo jako pole wyjaśnia nasz losowy i chaotyczny świat. Korzystając z podstawowych praw prawdopodobieństwa, możemy modelować rzeczywiste zdarzenia z udziałem losowości. Możemy użyć zmiennych losowych do reprezentowania wartości, które mogą przyjmować kilka wartości, a funkcji masy lub gęstości prawdopodobieństwa możemy użyć do porównania linii produktów lub przyjrzenia się wynikom testów. Widzieliśmy niektóre z bardziej skomplikowanych zastosowań prawdopodobieństwa w przewidywaniu. Korzystanie ze zmiennych losowych i twierdzenia Bayesa to doskonałe sposoby przypisywania prawdopodobieństw do sytuacji z życia codziennego. W kolejnych rozdziałach ponownie przyjrzymy się twierdzeniu Bayesa i wykorzystamy je do stworzenia bardzo potężnego i szybkiego algorytmu uczenia maszynowego, zwanego algorytmem Naïve Bayes. Algorytm ten wychwytuje moc myślenia bayesowskiego i stosuje go bezpośrednio do problemu uczenia się predykcyjnego. Kolejne dwa rozdziały koncentrują się na myśleniu statystycznym. Podobnie jak prawdopodobieństwo, rozdziały te będą wykorzystywać formuły matematyczne do modelowania rzeczywistych wydarzeń. Główną różnicą będzie jednak terminologia, której używamy do opisywania świata oraz sposób, w jaki modelujemy różnego rodzaju zdarzenia. W kolejnych częściach postaramy się zamodelować całe populacje punktów danych wyłącznie na podstawie próbki. Z prawdopodobieństwem ponownie przyjrzymy się wielu pojęciom, aby nadać sens twierdzeniom statystycznym, ponieważ są one ściśle powiązane i oba są ważnymi pojęciami matematycznymi w dziedzinie nauki o danych

Statystyki podstawowe

W tej części skupimy się na statystykach wymaganych przez każdego początkującego naukowca danych. Zbadamy sposoby próbkowania i uzyskiwania danych bez wpływu na uprzedzenia, a następnie użyjemy miar statystycznych do kwantyfikacji i wizualizacji naszych danych. Wykorzystując z-score i regułę empiryczną, zobaczymy, jak możemy standaryzować dane na potrzeby tworzenia wykresów i interpretacji. Przyjrzymy się następującym tematom:

• Jak pozyskiwać i próbkować dane?

• Miary środka, wariancji i względnej pozycji

• Normalizacja danych za pomocą z-score

• Reguła empiryczna

Czym są statystyki?

To może wydawać się dziwne pytanie, ale często dziwi mnie liczba osób, które nie potrafią odpowiedzieć na to proste, a jednocześnie mocne pytanie: czym są statystyki? Statystyki to liczby, które zawsze widzisz w wiadomościach i w gazecie. Statystyki są przydatne, gdy próbujesz udowodnić swoją rację lub próbujesz cię przestraszyć, ale czym one są? Aby odpowiedzieć na to pytanie, musimy zrobić kopię zapasową na minutę i porozmawiać o tym, dlaczego w ogóle je mierzymy. Celem tej dziedziny jest próba wyjaśnienia i modelowania otaczającego nas świata. Aby to zrobić, musimy przyjrzeć się populacji. Możemy zdefiniować populację jako całą pulę podmiotów eksperymentu lub modelu. Zasadniczo zależy ci na twojej populacji. O kim próbujesz porozmawiać? Jeśli próbujesz sprawdzić, czy palenie prowadzi do chorób serca, twoja populacja byłaby palaczami na całym świecie. Jeśli próbujesz badać problemy nastolatków związanych z piciem, twoją populacją będą wszyscy nastolatkowie. Teraz pomyśl, że chcesz zadać pytanie dotyczące twojej populacji, na przykład, jeśli twoja populacja to wszyscy twoi pracownicy (załóżmy, że masz ponad 1000 pracowników), być może chcesz wiedzieć, jaki procent z nich używa nielegalnych narkotyków. Pytanie nazywa się parametrem. Parametr możemy zdefiniować jako pomiar liczbowy opisujący charakterystykę populacji. Na przykład, jeśli zapytasz wszystkich 1000 pracowników i 100 z nich zażywa narkotyki, wskaźnik zażywania narkotyków wynosi 10%. Parametr tutaj wynosi 10%. Jednak bądźmy szczerzy, prawdopodobnie nie można zapytać każdego pracownika, czy zażywa narkotyki. A jeśli masz ponad 10 000 pracowników? Byłoby bardzo trudno wyśledzić wszystkich, aby uzyskać odpowiedź. Kiedy tak się dzieje, niemożliwe jest ustalenie tego parametru. W takim przypadku możemy oszacować parametr. Najpierw weźmiemy próbkę populacji. Możemy zdefiniować próbkę populacji jako podzbiór (losowy niewymagany) populacji. Więc być może zapytamy 200 z 1000 pracowników, których masz. Załóżmy, że z tych 200 osób 26 używa narkotyków, co oznacza, że wskaźnik zażywania narkotyków wynosi 13%. Tutaj 13% nie jest parametrem, ponieważ nie mieliśmy okazji zapytać wszystkich. To 13% to oszacowanie parametru. Czy wiesz, jak to się nazywa? Zgadza się, statystyka! Możemy zdefiniować statystykę jako pomiar liczbowy opisujący charakterystykę próbki populacji. Statystyka to tylko oszacowanie parametru. Jest to liczba, która próbuje opisać całą populację, opisując jej podzbiór. Jest to konieczne, ponieważ nigdy nie możesz mieć nadziei na przeprowadzenie ankiety każdemu nastolatkowi lub każdemu palaczowi na świecie. O to właśnie chodzi w dziedzinie statystyki - pobieranie próbek populacji i przeprowadzanie testów na tych próbkach. Zatem następnym razem, gdy otrzymasz statystykę, pamiętaj tylko, że ta liczba reprezentuje tylko próbkę tej populacji, a nie całą pulę badanych.

Jak pozyskujemy i przykładamy dane?

Jeśli statystyki dotyczą pobierania próbek populacji, bardzo ważne jest, aby wiedzieć, w jaki sposób uzyskujemy te próbki, i masz rację. Skoncentrujmy się tylko na kilku z wielu sposobów pozyskiwania i próbkowania danych.

Uzyskiwanie danych

Istnieją dwa główne sposoby zbierania danych do naszej analizy: obserwacyjne i eksperymentalne. Oba te sposoby mają oczywiście swoje plusy i minusy. Każdy z nich wytwarza różne rodzaje zachowań, a zatem uzasadnia różne rodzaje analizy.

Obserwacje

Możemy pozyskiwać dane za pomocą środków obserwacyjnych, które polegają na mierzeniu określonych cech, ale nie próbowaniu modyfikowania badanych osób. Na przykład masz na swojej stronie oprogramowanie śledzące, które obserwuje zachowanie użytkowników w witrynie, takie jak czas spędzony na określonych stronach i szybkość klikania reklam, a jednocześnie nie wpływa na wrażenia użytkownika, wtedy byłoby to badanie obserwacyjne. Jest to jeden z najczęstszych sposobów pozyskiwania danych, ponieważ jest to po prostu łatwe. Wszystko, co musisz zrobić, to obserwować i zbierać dane. Badania obserwacyjne są również ograniczone pod względem rodzajów danych, które możesz gromadzić. Dzieje się tak, ponieważ obserwator (ty) nie kontroluje środowiska. Możesz jedynie obserwować i zbierać naturalne zachowania. Jeśli chcesz wywołać określony rodzaj zachowania, badanie obserwacyjne nie będzie przydatne.

Eksperymentalnie

Eksperyment polega na leczeniu i obserwacji jego wpływu na badanych. Osoby biorące udział w eksperymencie nazywane są jednostkami eksperymentalnymi. W ten sposób zwykle zbiera dane większość laboratoriów naukowych. Umieszczą ludzi w dwóch lub więcej grupach (zwykle tylko dwóch) i nazywają ich grupą kontrolną i eksperymentalną. Grupa kontrolna jest wystawiona na określone środowisko, a następnie obserwowana. Grupa eksperymentalna jest następnie wystawiana na działanie innego środowiska, a następnie obserwowana. Eksperymentator następnie agreguje dane z obu grup i podejmuje decyzję o tym, które środowisko było korzystniejsze (korzystne to jakość, o której decyduje eksperymentator).

Weźmy pod uwagę przykład marketingowy, że pokazujemy połowę naszych użytkowników na określonej stronie docelowej z określonymi obrazami i określonym stylem (witryna A) i mierzymy, czy rejestrują się w usłudze. Następnie drugą połowę udostępniamy innej stronie docelowej, innym obrazom i różnym stylom (witryna B) i ponownie mierzymy, czy się zarejestrowali. Możemy wtedy zdecydować, która z dwóch stron wypadła lepiej i powinna być wykorzystywana dalej. Nazywa się to w szczególności testem A/B. Zobaczmy przykład w Pythonie! Załóżmy, że uruchamiamy poprzedni test i otrzymujemy następujące wyniki jako listę list:

results = [['A', 1], ['B', 1], ['A', 0], ['A', 0] …]

Tutaj każdy obiekt w wyniku listy reprezentuje podmiot (osobę). Każda osoba ma wtedy następujące dwa atrybuty:

• Na jakiej stronie internetowej byli obecni, reprezentowanej przez pojedynczy znak

• Czy dokonali konwersji (0 za nie i 1 za tak)

Następnie możemy dokonać agregacji i uzyskać następującą tabelę wyników:

users_exposed_to_A = []

users_exposed_to_B = []

utwórz dwie listy do przechowywania wyników każdej indywidualnej witryny

Po utworzeniu tych dwóch list, które ostatecznie będą zawierały wartość logiczną każdej indywidualnej konwersji (0 lub 1), powtórzymy wszystkie nasze wyniki testu i dodamy je do odpowiedniej listy, jak pokazano:

for website, converted in results: # iterate through the results

will look something like website == 'A' and converted == 0

if website == 'A':

users_exposed_to_A.append(converted)

elif website == 'B':

users_exposed_to_B.append(converted)

Teraz każda lista zawiera serię jedynek i zer. Pamiętaj, że 1 oznacza użytkownika, który faktycznie przechodzi na witrynę po obejrzeniu tej strony internetowej, a 0 oznacza, że użytkownik widzi tę stronę i opuszcza ją przed zarejestrowaniem się/konwersją. Aby uzyskać całkowitą liczbę osób narażonych na witrynę A, możemy użyć funkcji len() w Pythonie, jak pokazano:

len(users_exposed_to_A) == 188 #number of people exposed to website A

len(users_exposed_to_B) == 158 #number of people exposed to website B

Aby policzyć liczbę osób, które dokonały konwersji, możemy użyć sum() z listy, jak pokazano:

sum(users_exposed_to_A) == 54 # people converted from website A

sum(users_exposed_to_B) == 48 # people converted from website B

Jeśli odejmiemy długość list i sumę listy, otrzymamy liczbę osób, które nie dokonały konwersji dla każdej witryny, jak pokazano

len(users_exposed_to_A) - sum(users_exposed_to_A) == 134 # did not convert from website A

len(users_exposed_to_B) - sum(users_exposed_to_B) == 110 # did not convert from website B

Możemy zebrać i podsumować nasze wyniki w poniższej tabeli, która przedstawia nasz eksperyment testowania konwersji w witrynie:

	Did not sign up	Signed up
Website A	134	54
Website B	110	48

Możemy szybko zebrać statystyki opisowe. Można powiedzieć, że współczynniki konwersji witryny dla obu witryn są następujące:

• Konwersja dla witryny A: 54/134 + 54 = 0,288

• Konwersja dla witryny B: 48 / 110 + 48 = 0,3

Niewielka różnica, ale jednak inna. Mimo że B ma wyższy współczynnik konwersji, czy naprawdę możemy powiedzieć, że wersja B znacznie lepiej konwertuje? Jeszcze nie. Aby sprawdzić istotność statystyczną takiego wyniku, należy zastosować test hipotezy. Testy te zostaną szczegółowo omówione w następnym rozdziale, w którym ponownie przyjrzymy się dokładnie temu samemu przykładowi i zakończymy go odpowiednim testem statystycznym.

Próbkowanie danych

Pamiętaj, że statystyki są wynikiem pomiaru próbki populacji. Cóż, powinniśmy porozmawiać o dwóch bardzo powszechnych sposobach decydowania, kto dostąpi zaszczytu bycia w mierzonej przez nas próbce. Omówimy główny rodzaj doboru próby, zwany doborem losowym, który jest najczęstszym sposobem decydowania o wielkości naszej próby i naszych członków próby.

Próbkowanie prawdopodobieństwa

Próbkowanie prawdopodobieństwa to sposób próbkowania z populacji, w którym każda osoba ma znane prawdopodobieństwo wybrania, ale ta liczba może mieć inne prawdopodobieństwo niż inny użytkownik. Najprostszą (i prawdopodobnie najczęstszą) metodą próbkowania prawdopodobieństwa jest próbkowanie losowe.

Próbkowanie losowe

Załóżmy, że przeprowadzamy test A/B i musimy dowiedzieć się, kto będzie w grupie A, a kto w grupie B. Zespół ds. danych przedstawia następujące trzy sugestie:

• Oddzielni użytkownicy na podstawie lokalizacji: użytkownicy na zachodnim wybrzeżu są umieszczani w grupie A, podczas gdy użytkownicy na wschodnim wybrzeżu są umieszczani w grupie B

• Oddziel użytkowników na podstawie pory dnia, kiedy odwiedzają witrynę: Użytkownicy, którzy odwiedzają witrynę między 19:00. i o 4 rano dostaniesz miejsce A, podczas gdy reszta zostanie umieszczona w grupie B

• Zrób to całkowicie losowo: każdy nowy użytkownik ma 50/50 szans na umieszczenie w jednej z grup

Pierwsze dwie są prawidłowymi opcjami wyboru próbek i są dość proste do wdrożenia, ale obie mają jedną podstawową wadę: oba są narażone na wprowadzenie błędu próbkowania.

Błąd w doborze próby występuje, gdy sposób, w jaki pozyskiwana jest próbka, faworyzuje pewien wynik w porównaniu z wynikiem docelowym.

Nietrudno zrozumieć, dlaczego wybór opcji 1 lub opcji 2 może wprowadzać stronniczość. Jeśli wybieramy nasze grupy na podstawie tego, gdzie mieszkają lub o której godzinie się logują, to niepoprawnie przygotowujemy nasz eksperyment i teraz mamy znacznie mniejszą kontrolę nad wynikami. W szczególności istnieje ryzyko wprowadzenia do naszej analizy czynnika mylącego, co jest złą wiadomością.

Czynnikiem mylącym jest zmienna, której nie mierzymy bezpośrednio, ale łączymy zmienne, które są mierzone.

Zasadniczo czynnik mylący jest jak brakujący element w naszej analizie, który jest niewidoczny, ale wpływa na nasze wyniki. W tym przypadku wariant 1 nie uwzględnia potencjalnego mylącego czynnika smaku geograficznego. Na przykład, jeśli witryna A jest ogólnie nieatrakcyjna dla użytkowników z zachodniego wybrzeża, drastycznie wpłynie to na wyniki.

Podobnie opcja 2 może wprowadzić czasowy (oparty na czasie) czynnik zakłócający. Co jeśli witryna B jest lepiej oglądana w nocy (co było zarezerwowane dla A), a użytkownicy są wyłączani z tego stylu wyłącznie z powodu godziny. Są to oba czynniki, których chcemy uniknąć, dlatego powinniśmy wybrać opcję 3, która jest losowa próbka.

Chociaż stronniczość próbkowania może wprowadzać w błąd, jest to inne pojęcie niż mylące. Warianty 1 i 2 były obiema stronniczymi próbkami, ponieważ wybraliśmy próbki nieprawidłowo, a także były przykładami czynników mylących, ponieważ w każdym przypadku istniała trzecia zmienna, która wpływała na naszą decyzję.

Próba losowa jest wybierana w taki sposób, aby każdy członek populacji miał taką samą szansę na wybór jak każdy inny członek. Jest to prawdopodobnie jeden z najłatwiejszych i najwygodniejszych sposobów decydowania, kto będzie częścią twojej próbki. Każdy ma taką samą szansę na bycie w określonej grupie. Próbkowanie losowe to skuteczny sposób na zmniejszenie wpływu czynników zakłócających.

Próbkowanie z nierównym prawdopodobieństwem

Przypomnijmy, że wcześniej powiedziałem, że próbkowanie prawdopodobieństwa może mieć różne prawdopodobieństwa dla różnych potencjalnych członków próby. Ale co, jeśli faktycznie spowodowało to problemy? Załóżmy, że jesteśmy zainteresowani pomiarem poziomu szczęścia naszych pracowników. Wiemy już, że nie możemy zapytać wszystkich osób z personelu, ponieważ byłoby to głupie i wyczerpujące. Więc musimy pobrać próbkę. Nasz zespół danych sugeruje losowe pobieranie próbek i na początku każdemu przybija piątkę, ponieważ czują się bardzo inteligentni i statystycznie. Ale wtedy ktoś zadaje pozornie nieszkodliwe pytanie - czy ktoś zna procent mężczyzn/kobiet, którzy tu pracują? Przybijanie piątki ustaje i w pokoju zapada cisza. To pytanie jest niezwykle ważne, ponieważ seks może być czynnikiem mylącym. Zespół przygląda się temu i odkrywa, że w firmie jest 75% mężczyzn i 25% kobiet. Oznacza to, że jeśli wprowadzimy próbę losową, nasza próba prawdopodobnie będzie miała podobny podział, a tym samym będzie faworyzować wyniki dla mężczyzn, a nie kobiet. Aby temu zaradzić, możemy faworyzować uwzględnienie większej liczby kobiet niż mężczyzn w naszym badaniu, aby podział naszej próby był mniej korzystny dla mężczyzn. Na pierwszy rzut oka wprowadzenie systemu faworyzowania do losowego doboru próby wydaje się złym pomysłem, jednak złagodzenie nierównego doboru próby, a zatem praca nad usunięciem systematycznych uprzedzeń dotyczących płci, rasy, niepełnosprawności itd. jest znacznie bardziej adekwatna. Prosta próba losowa, w której każdy ma takie same szanse jak wszyscy, z dużym prawdopodobieństwem zagłuszy głosy i opinie członków populacji mniejszości. Dlatego wprowadzenie takiego systemu faworyzowania do technik pobierania próbek może być w porządku.

Jak mierzymy statystyki?

Gdy już mamy naszą próbkę, nadszedł czas, aby oszacować nasze wyniki. Załóżmy, że chcemy uogólnić szczęście naszych pracowników lub chcemy dowiedzieć się, czy wynagrodzenia w firmie bardzo różnią się w zależności od osoby. Oto kilka typowych sposobów mierzenia naszych wyników.

Miary środka

Miary środka to sposób, w jaki definiujemy środek lub środek zbioru danych. Robimy to, ponieważ czasami chcemy dokonać uogólnień na temat wartości danych. Na przykład, być może jesteśmy ciekawi, jakie są średnie opady w Seattle lub jaka jest mediana wzrostu dla europejskich mężczyzn. Jest to sposób na uogólnienie dużego zestawu danych, aby łatwiej było komuś przekazać. Miara środka to wartość w „środku" zbioru danych. Może to jednak oznaczać różne rzeczy dla różnych osób. Kto może

powiedzieć, gdzie jest środek zbioru danych? Jest tak wiele różnych sposobów definiowania centrum danych. Przyjrzyjmy się kilku. Średnią arytmetyczną zbioru danych można znaleźć, sumując wszystkie wartości, a następnie dzieląc ją przez liczbę wartości danych. Jest to prawdopodobnie najczęstszy sposób definiowania centrum danych, ale może być wadliwy! Załóżmy, że chcemy znaleźć średnią z następujących liczb:

import numpy as np.

np.mean([11, 15, 17, 14]) == 14.25

Dość prosto, nasza średnia wynosi 14,25 i wszystkie nasze wartości są do niej dość zbliżone. A co, jeśli wprowadzimy nową wartość: 31?

np.mean([11, 15, 17, 14, 31]) == 17.6

Ma to duży wpływ na średnią, ponieważ średnia arytmetyczna jest wrażliwa na wartości odstające. Nowa wartość, 31, jest prawie dwa razy większa niż reszta liczb, a zatem odchyla średnią. Inną, czasem lepszą miarą centrum jest mediana. Mediana to liczba znaleziona w środku zbioru danych, gdy jest on posortowany w zamów, jak pokazano:

np.median([11, 15, 17, 14]) == 14.5

np.median([11, 15, 17, 14, 31]) == 15

Zauważ, że wprowadzenie 31 przy użyciu mediany nie wpłynęło znacząco na medianę zbioru danych. Dzieje się tak, ponieważ mediana jest mniej wrażliwa na wartości odstające. Podczas pracy z zestawami danych z wieloma wartościami odstającymi czasami bardziej przydatne jest użycie mediany zbioru danych, natomiast jeśli dane nie zawierają wielu wartości odstających, a punkty danych są w większości blisko siebie, prawdopodobnie lepszym rozwiązaniem jest średnia. Ale jak możemy stwierdzić, czy dane są rozproszone? Cóż, będziemy musieli wprowadzić nowy rodzaj statystyki.

Miary zmienności

Miary środka są używane do ilościowego określenia środka danych, ale teraz zbadamy sposoby mierzenia, jak „rozprzestrzeniają się" gromadzone przez nas dane. Jest to przydatny sposób sprawdzenia, czy w naszych danych czai się wiele odstających elementów. Zacznijmy od przykładu. Weź pod uwagę, że bierzemy losową próbkę 24 naszych znajomych na Facebooku i piszemy ilu znajomych mieli na Facebooku. Oto lista:

friends = [109, 1017, 1127, 418, 625, 957, 89, 950, 946, 797, 981,

125, 455, 731, 1640, 485, 1309, 472, 1132, 1773, 906, 531, 742, 621]

np.mean(friends) == 789.1

Średnia tej listy to nieco ponad 789. Można więc powiedzieć, że według tej próbki przeciętny znajomy z Facebooka ma 789 znajomych. Ale co z osobą, która ma tylko 89 przyjaciół lub osobą, która ma ponad 1600 przyjaciół? W rzeczywistości niewiele z tych liczb jest naprawdę zbliżonych do 789. A może użyjemy mediany, jak pokazano, ponieważ na medianę zasadniczo nie wpływają wartości odstające:

np.median(friends) == 769.5

Mediana wynosi 769,5, co jest dość zbliżone do średniej. Hmm, dobra myśl, ale tak naprawdę nie wyjaśnia to, jak drastycznie wiele z tych punktów danych różni się od siebie. To właśnie statystycy

nazywają mierzeniem zmienności danych. Zacznijmy od wprowadzenia najbardziej podstawowej miary zmienności: zakresu. Zakres to po prostu wartość maksymalna minus wartość minimalna, jak pokazano

np.max(friends) – np.min(friends) == 1684

Zakres mówi nam, jak daleko znajdują się dwie najbardziej skrajne wartości. Obecnie zakres ten nie jest powszechnie używany, ale ma swoje zastosowanie w zastosowaniu. Czasami chcemy po prostu wiedzieć, jak rozłożone są wartości odstające. Jest to najbardziej przydatne w pomiarach naukowych lub pomiarach bezpieczeństwa. Załóżmy, że firma samochodowa chce zmierzyć, ile czasu zajmuje rozwinięcie się poduszki powietrznej. Znajomość średniej z tego czasu jest fajna, ale naprawdę chcą również wiedzieć, jak rozkłada się najwolniejszy czas w porównaniu z najszybszym. To dosłownie może być różnica między życiem a śmiercią. Wracając do przykładu z Facebooka, 1684 to nasz zakres, ale nie jestem pewien, czy mówi za dużo o naszych danych. Przyjrzyjmy się teraz najczęściej używanej mierze zmienności, czyli odchyleniu standardowemu. Jestem pewien, że wielu z was często słyszało ten termin i może nawet wzbudzać strach, ale co to tak naprawdę oznacza? Zasadniczo odchylenie standardowe, oznaczane przez s, gdy pracujemy z próbką populacji, mierzy, jak bardzo wartości danych odbiegają od średniej arytmetycznej. Jest to w zasadzie sposób, aby zobaczyć, jak rozłożone są dane. Istnieje ogólna formuła obliczania odchylenia standardowego, która wygląda następująco:

$$s = \sqrt{\frac{\sum (x - \overline{x})^2}{n}}$$

Tutaj:

• s jest naszym odchyleniem standardowym próbki

• x to każdy indywidualny punkt danych.

• $\overline{x}$ jest średnią danych

• n jest liczba punktów danych

Zanim wpadniesz, rozbijmy to. Dla każdej wartości w próbie weźmiemy tę wartość, odejmiemy od niej średnią arytmetyczną, podniesiemy do kwadratu różnicę, a kiedy dodamy w ten sposób każdy punkt, podzielimy całość przez n, liczbę punktów w próbie. Na koniec ze wszystkiego wyciągamy pierwiastek kwadratowy. Nie wchodząc w dogłębną analizę wzoru, pomyśl o tym w ten sposób: w zasadzie wywodzi się z wzoru na odległość. Zasadniczo to, co oblicza odchylenie standardowe, jest rodzajem średniej odległości odle,głości wartości danych od średniej arytmetycznej. Jeśli przyjrzysz się bliżej tej formule, zobaczysz, że ma ona sens:

• Biorąc , $x - \overline{x}$ znajdujesz dosłowną różnicę między wartością a średnią próbki.

• Podnosząc wynik do kwadratu, $(x - \overline{x})^2$ nakładamy większą karę na wartości odstające, ponieważ podniesienie do kwadratu dużego błędu powoduje, że jest on znacznie większy.

• Dzieląc przez liczbę pozycji w próbie, bierzemy (dosłownie) średnią kwadratową odległość między każdym punktem a średnią.

• Wyciągając pierwiastek kwadratowy z odpowiedzi, podajemy liczbę w zrozumiały dla nas sposób. Na przykład, podnosząc do kwadratu liczbę znajomych minus średnią, zmieniliśmy nasze jednostki na

kwadrat znajomych, co nie ma sensu. Wyciągnięcie pierwiastka kwadratowego sprawia, że nasze jednostki stają się z powrotem tylko „przyjaciółmi”.

Wróćmy do naszego przykładu na Facebooku, aby uzyskać wizualizację i dalsze wyjaśnienie tego. Zacznijmy obliczyć odchylenie standardowe. Więc zaczniemy obliczać kilka z nich. Przypomnijmy, że średnia arytmetyczna danych wynosiła około 789, więc użyjemy 789 jako średniej. Zaczynamy od różnicy między każdą wartością danych a średnią, podnosząc ją do kwadratu, dodając je wszystkie, dzieląc przez jeden mniej niż liczba wartości, a następnie wyciągając pierwiastek kwadratowy. Wyglądałoby to następująco:

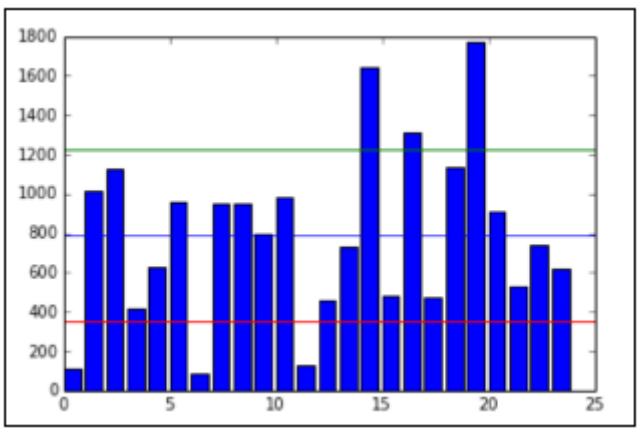
$$s = \sqrt{\frac{(109-789)^2 + (1017-789)^2 + \cdots + (621-789)^2}{24}}$$

Z drugiej strony możemy przyjąć podejście Pythona i zrobić to wszystko programowo (co jest zwykle preferowane).

np.std(friends) # == 425.2

Liczba 425 reprezentuje rozproszenie danych. Można powiedzieć, że 425 to rodzaj średniej odległości wartości danych od średniej. W prostych słowach oznacza to, że te dane są dość rozproszone. Tak więc nasze odchylenie standardowe wynosi około 425. Oznacza to, że liczba znajomych tych osób na Facebooku nie wydaje się być zbliżona do jednej liczby i jest to całkiem oczywiste, gdy wykreślimy dane na wykresie słupkowym, a także na wykresie średniej oraz wizualizacje odchylenia standardowego. Na poniższym wykresie każda osoba będzie reprezentowana przez pojedynczy słupek na wykresie słupkowym, a wysokość słupków reprezentuje liczbę znajomych, których mają osoby:

```
import matplotlib.pyplot as plt

%matplotlib inline

y_pos = range(len(friends))

plt.bar(y_pos, friends)

plt.plot((0, 25), (789, 789), 'b-')

plt.plot((0, 25), (789+425, 789+425), 'g-')

plt.plot((0, 25), (789-425, 789-425), 'r-')
```



Niebieska linia w środku jest narysowana na średniej (789), czerwona linia na dole jest narysowana na średniej minus odchylenie standardowe (789-425 = 364), a na koniec narysowana jest zielona linia w kierunku góry przy średniej plus odchylenie standardowe (789+425 = 1,214). Zwróć uwagę, że większość danych znajduje się między zieloną a czerwoną linią, podczas gdy osoby odstające żyją poza liniami. Mianowicie, są trzy osoby, które mają znajomego poniżej czerwonej linii i trzy osoby, które mają znajomego, powyżej zielonej linii. Należy wspomnieć, że jednostki odchylenia standardowego są w rzeczywistości tymi samymi jednostkami, co jednostki danych. W tym przykładzie powiedzielibyśmy, że odchylenie standardowe wynosi 425 znajomych na Facebooku. Inną miarą zmienności jest wariancja, jak opisano w poprzedniej części. Wariancja to po prostu odchylenie standardowe do kwadratu. Teraz wiemy, że odchylenie standardowe i wariancja są dobre do sprawdzenia, jak rozłożone są nasze dane i że możemy je wykorzystać wraz ze średnią do stworzenia pewnego rodzaju zakresu, w którym znajduje się wiele naszych danych. Ale co, jeśli chcemy porównać rozprzestrzenianie się dwóch różnych zbiorów danych, może nawet z zupełnie różnymi jednostkami? Tutaj w grę wchodzi współczynnik zmienności.

Definicja

Współczynnik zmienności definiuje się jako stosunek odchylenia standardowego danych do ich średniej. Ten współczynnik (który, nawiasem mówiąc, jest pomocny tylko wtedy, gdy pracujemy na poziomie pomiaru współczynnika, gdzie podział jest dozwolony i ma sens) jest sposobem na standaryzację odchylenia standardowego, co ułatwia porównywanie między zestawami danych. Często używamy tej miary, gdy próbujemy porównywać średnie, i rozprzestrzenia się ona na populacje, które istnieją w różnych skalach.

Przykład – pensje pracowników

Jeśli spojrzymy na średnią i odchylenie standardowe wynagrodzeń pracowników w tej samej firmie, ale w różnych działach, widzimy, że na pierwszy rzut oka porównanie różnic może być trudne.

Salaries of Company XYZ

Department	Mean Salary	SD	CoV
Mailroom	$25,000	$2,000	8.0%
Human Resources	$52,000	$7,000	13.5%
Executive	$124,000	$42,000	33.9%

Jest to szczególnie prawdziwe, gdy średnia pensja jednego działu wynosi 25 000 USD, podczas gdy w innym dziale średnia pensja jest sześciocyfrowa. Jeśli jednak spojrzymy na ostatnią kolumnę, która jest naszym współczynnikiem zmienności, staje się jasne, że ludzie w dziale wykonawczym mogą zarabiać więcej, ale pracownicy w dziale wykonawczym dostają szalenie różne pensje. Dzieje się tak prawdopodobnie dlatego, że dyrektor generalny zarabia znacznie więcej niż kierownik biura, który nadal jest w dziale wykonawczym, co sprawia, że dane są bardzo rozproszone. Z drugiej strony, wszyscy w kancelarii, chociaż nie zarabiają tyle pieniędzy, zarabiają mniej więcej tyle samo, co wszyscy inni w kancelarii, dlatego ich współczynnik zmienności wynosi tylko 8%. Dzięki miarom zmienności możemy zacząć odpowiadać na ważne pytania, takie jak rozłożenie tych danych lub ustalenie dobrego zakresu, w którym mieści się większość danych.

Miary względnej pozycji

Możemy łączyć zarówno miary centrów, jak i wariacji, aby tworzyć miary pozycji względnych. Miary zmienności miara, gdzie poszczególne wartości danych są pozycjonowane, względne do całego zbioru

danych. Zacznijmy od poznania bardzo ważnej wartości w statystyce, wskaźnika Z. Z-score to sposób na powiedzenie nam, jak daleko od średniej znajduje się pojedyncza wartość danych. Wynik z wartości danych x jest następujący:

$$z = \frac{x - \overline{x}}{s}$$

Gdzie:

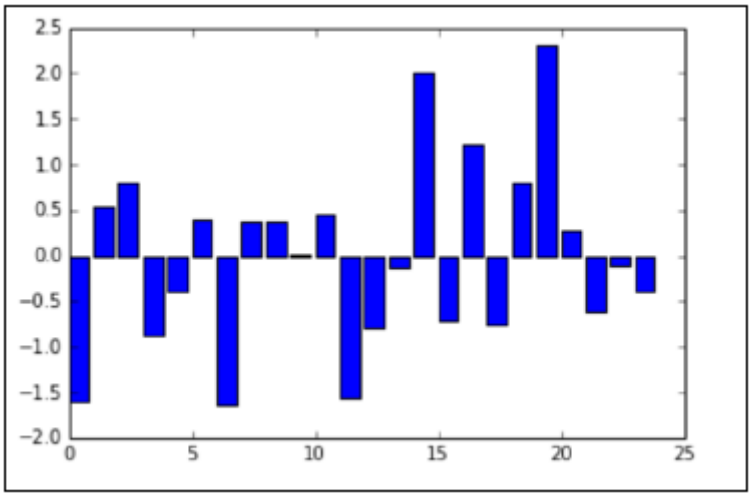
- x jest punktem danych
- $\overline{x}$ jest średnią
- s to odchylenie standardowe.

Pamiętaj, że odchylenie standardowe było (w pewnym sensie) średnią odległością danych od średniej, a teraz z-score jest indywidualną wartością dla każdego konkretnego punktu danych. Możemy znaleźć wynik Z wartości danych, odejmując ją od średniej i dzieląc przez odchylenie standardowe. Wynikiem będzie standaryzowana odległość, w jakiej wartość jest od średniej. Używamy wskaźnika Z w całej statystyce. Jest to bardzo skuteczny sposób normalizowania danych, które istnieją w bardzo różnych skalach, a także umieszczania danych w kontekście ich średniej. Weźmy nasze poprzednie dane o liczbie znajomych na Facebooku i ujednolicimy dane do z-score. Dla każdego punktu danych znajdziemy jego z-score, stosując poprzednią formułę. Weźmiemy każdą osobę, odejmiemy przeciętnych przyjaciół od wartości i podzielimy ją przez odchylenie standardowe, jak pokazano:

```
z_scores = []

m = np.mean(friends) # przeciętnych znajomych na Facebooku

s = np.std(friends) # znajomi z odchyleniem standardowym na Facebooku dla znajomego w znajomych:

z = (przyjaciel - m)/s # z-score

z_scores.append(z) # zrób listę wyników do wykreślenia
```

Teraz wykreślmy te wyniki Z na wykresie słupkowym. Poniższy wykres pokazuje te same osoby z naszego poprzedniego przykładu używającego znajomych na Facebooku, ale zamiast wysokości słupka pokazującej surową liczbę znajomych, teraz każdy słupek jest wynikiem z liczby znajomych, których mają na Facebooku. Jeśli wykreślimy z-scores, zauważymy parę rzeczy:

```
plt.bar(y_pos, z_scores)
```



• Mamy wartości ujemne (co oznacza, że punkt danych znajduje się poniżej średniej)

• Długości słupków nie reprezentują już surowej liczby znajomych, ale stopień, w jakim ta liczba znajomych różni się od średniej

Ten wykres bardzo ułatwia wytypowanie osób, które mają średnio dużo niższych i wyższych przyjaciół. Na przykład osoba z indeksem 0 ma średnio mniej znajomych . A co, jeśli chcemy narysować na wykresie odchylenia standardowe? Przypomnijmy, że wcześniej narysowaliśmy trzy poziome linie: jedną przy średniej, jedną przy średniej plus odchylenie standardowe (x + s) i jedną przy średniej minus odchylenie standardowe (x - s). Jeśli wstawimy te wartości do wzoru na z-score, otrzymamy:

$$\text{Z-score of } (\overline{x}\,\overline{x}) = \frac{x-\overline{x}}{s} = \frac{0}{s} = 0$$

$$\text{Z-score of } (x+s) = \frac{(\overline{x}+s)-\overline{x}}{s} = \frac{s}{s} = 1$$

$$\text{Z-score of } (x-s) \ \frac{(\overline{x}-s)-\overline{x}}{s} = \frac{-s}{s} = -1$$

To nie przypadek! Kiedy standaryzujemy dane za pomocą wskaźnika Z, nasze odchylenia standardowe stają się miarą z wyboru. Zobaczmy nasz nowy wykres z naniesionymi odchyleniami standardowymi:

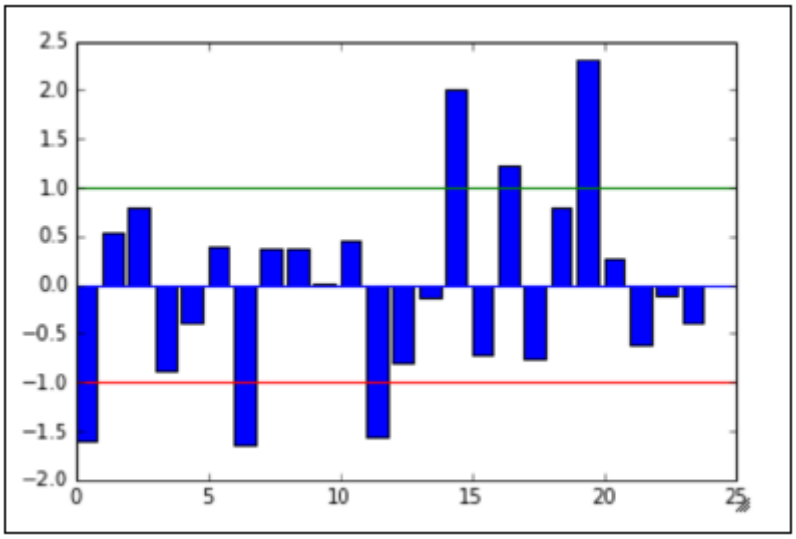
```
plt.bar(y_pos, z_scores)

plt.plot((0, 25), (1, 1), 'g-')

plt.plot((0, 25), (0, 0), 'b-')

plt.plot((0, 25), (-1, -1), 'r-')
```

Poprzedni kod dodaje w następujących trzech wierszach:

• Niebieska linia przy y = 0, która reprezentuje zero odchyleń standardowych od średniej (która znajduje się na osi x)

• Zielona linia, która reprezentuje jedno odchylenie standardowe powyżej średniej

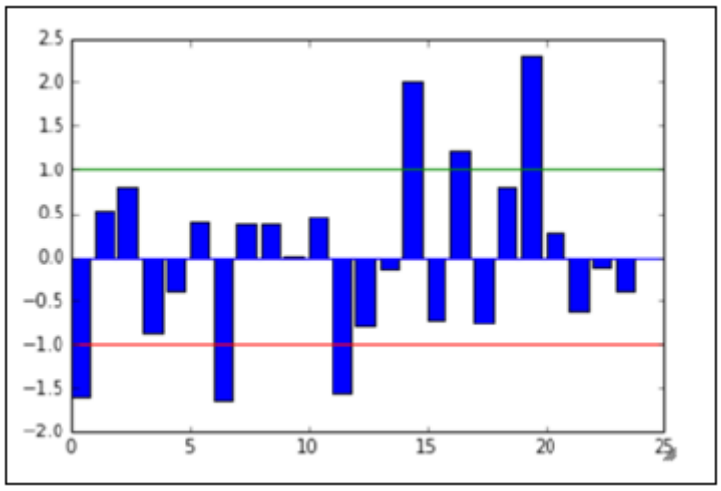
• Czerwona linia, która reprezentuje jedno odchylenie standardowe poniżej średniej



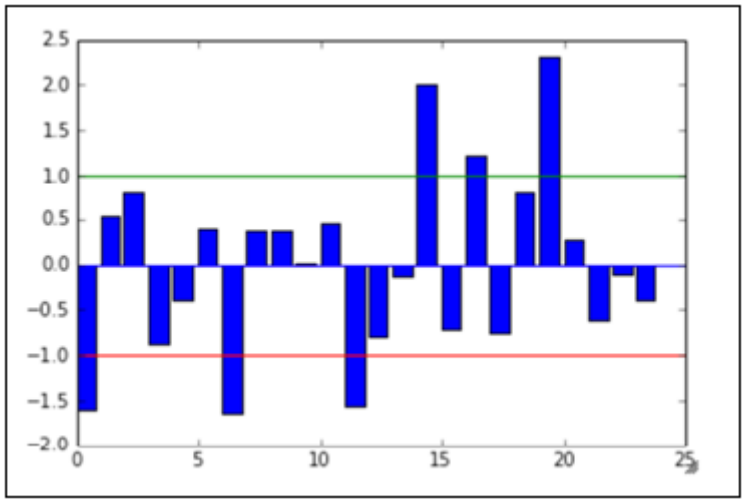
Kolory linii pasują do linii narysowanych na wcześniejszym wykresie surowej liczby znajomych. Jeśli przyjrzysz się uważnie, te same osoby nadal znajdują się poza zieloną i czerwoną linią. Mianowicie te same trzy osoby nadal spadają poniżej czerwonej (dolnej) linii, a te same trzy osoby spadają powyżej zielonej (górnej) linii.

W ramach tego skalowania możemy również użyć następujących stwierdzeń:

• Ten punkt danych jest o ponad jedno odchylenie standardowe od średniej:



• Ta osoba ma liczbę znajomych w granicach jednego odchylenia standardowego od średniej:



Wyniki Z to skuteczny sposób na standaryzację danych. Oznacza to, że cały zestaw możemy postawić na tej samej skali. Na przykład, jeśli zmierzymy również ogólną skalę szczęścia każdej osoby (która wynosi od 0 do 1), możemy mieć zestaw danych podobny do następującego zestawu danych:

```
friends = [109, 1017, 1127, 418, 625, 957, 89, 950, 946, 797, 981,
125, 455, 731, 1640, 485, 1309, 472, 1132, 1773, 906, 531, 742, 621]

happiness = [.8, .6, .3, .6, .6, .4, .8, .5, .4, .3, .3, .6, .2, .8,
1, .6, .2, .7, .5, .3, .1, 0, .3, 1]


import pandas as pd

df = pd.DataFrame({'friends':friends, 'happiness':happiness})
df.head()
```

	friends	happiness
0	109	0.8
1	1017	0.6
2	1127	0.3
3	418	0.6
4	625	0.6

Te punkty danych znajdują się w dwóch różnych wymiarach, z których każdy ma zupełnie inną skalę. Liczba znajomych może wynosić w tysiącach, podczas gdy nasz wynik zadowolenia utrzymuje się między 0 a 1. Aby temu zaradzić (a w przypadku niektórych modeli statystycznych/uczenia maszynowego ta koncepcja stanie się niezbędna), możemy po prostu ustandaryzować zbiór danych za pomocą gotowego pakietu standaryzacyjnego w scikit-learn, jak następuje:

from sklearn import preprocessing

df_scaled = pd.DataFrame(preprocessing.scale(df), columns = ['friends_

scaled', 'happiness_scaled'])

df_scaled.head()

Ten kod skaluje jednocześnie zarówno kolumny przyjaciół, jak i szczęście, ujawniając w ten sposób wynik Z dla każdej kolumny. Należy zauważyć, że w ten sposób moduł przetwarzania wstępnego w sklearn wykonuje następujące czynności osobno dla każdej kolumny:

• Znalezienie średniej kolumny

• Znalezienie odchylenia standardowego kolumny

• Zastosowanie funkcji z-score do każdego elementu w kolumnie

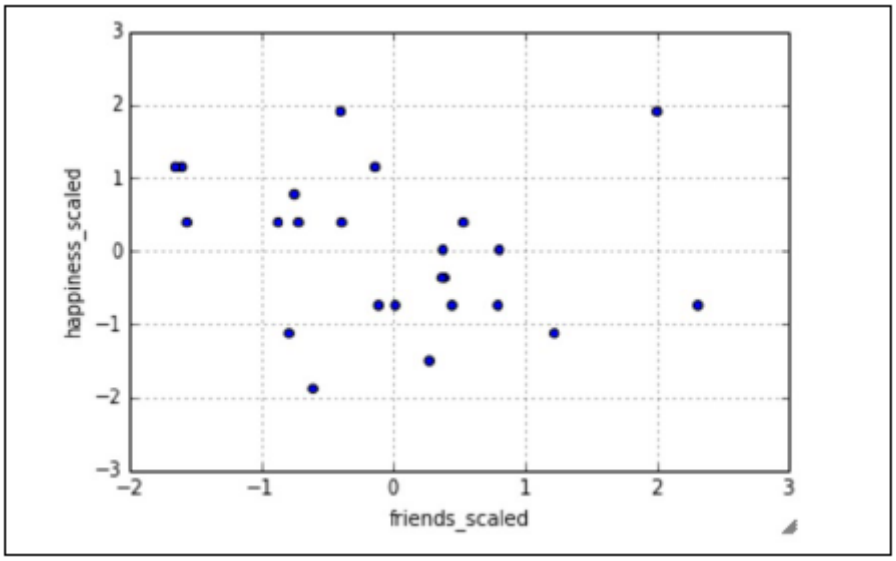
Wynikiem są dwie kolumny, jak pokazano, które istnieją w tej samej skali, nawet jeśli wcześniej nie były:

	friends_scaled	happiness_scaled
0	-1.599495	1.153223
1	0.536040	0.394939
2	0.794750	-0.742486
3	-0.872755	0.394939
4	-0.385909	0.394939

Teraz możemy wykreślić przyjaciół i szczęście w tej samej skali, a wykres będzie przynajmniej czytelny.

df_scaled.plot(kind='scatter', x = 'friends_scaled', y = 'happiness_

scaled')



Teraz nasze dane są standaryzowane do wskaźnika Z, a ten wykres punktowy jest dość łatwy do zinterpretowania! W dalszej części ta idea standaryzacji nie tylko sprawi, że nasze dane będą bardziej zrozumiałe, ale będzie również niezbędna w optymalizacji naszego modelu. Wiele algorytmów uczenia maszynowego będzie wymagało od nas standardowych kolumn, ponieważ są one zależne od pojęcia skali.

Część wnikliwa – korelacje w danych

W tym tekście omówimy różnicę między posiadaniem danych a posiadaniem praktycznych spostrzeżeń na temat danych. Posiadanie danych to tylko jeden krok do udanej operacji analizy danych. Możliwość uzyskania, oczyszczenia i wykreślenia danych pomaga opowiedzieć historię, którą dane mają do zaoferowania, ale nie może ujawnić morału. Aby posunąć cały ten przykład o krok dalej, przyjrzymy się relacji między posiadaniem znajomych na Facebooku a szczęściem. W kolejnych rozdziałach przyjrzymy się konkretnemu algorytmowi uczenia maszynowego, który próbuje znaleźć relacje między cechami ilościowymi, zwaną regresją liniową, ale nie musimy czekać do tego czasu, aby zacząć formułować hipotezy. Mamy próbkę ludzi, miarę ich obecności w sieci i zgłaszanego szczęścia. Pytanie dnia brzmi: czy możemy znaleźć związek między liczbą znajomych na Facebooku a ogólnym szczęściem? Oczywiście jest to duże pytanie i należy je traktować z szacunkiem. Eksperymenty mające odpowiedzieć na to pytanie powinny być prowadzone w warunkach laboratoryjnych,

ale możemy zacząć formułować hipotezę na temat tego pytania. Biorąc pod uwagę charakter naszych danych, tak naprawdę mamy tylko trzy możliwości postawienia hipotezy:

• Istnieje pozytywny związek między liczbą znajomych online a szczęściem (w miarę wzrostu jednego, rośnie drugi)

• Istnieje między nimi negatywny związek (w miarę wzrostu liczby przyjaciół twoje szczęście spada)

• Nie ma powiązania między zmiennymi (gdy jedna się zmienia, druga nie zmienia się tak bardzo)

Czy możemy użyć podstawowych statystyk do sformułowania hipotezy na to pytanie? Mówię, że możemy! Ale najpierw musimy wprowadzić pojęcie zwane korelacją. Współczynniki korelacji są miarą ilościową opisującą siłę powiązania/związek między dwiema zmiennymi. Korelacja między dwoma zestawami danych mówi nam o tym, jak poruszają się razem. Czy zmiana jednego pomoże nam przewidzieć drugie? Ta koncepcja jest nie tylko interesująca w tym przypadku, ale jest jednym z podstawowych założeń, które wiele modeli uczenia maszynowego przyjmuje na danych. Aby wiele algorytmów predykcyjnych działało, opierają się na fakcie, że istnieje jakiś związek między zmiennymi, na które patrzymy. Algorytmy uczące wykorzystują następnie tę zależność, aby dokonać dokładnych prognoz. Kilka rzeczy, o których należy pamiętać o współczynniku korelacji, jest następujące:

• Będzie leżeć między -1 a 1

• Im większa wartość bezwzględna (bliższa -1 lub 1), tym silniejszy związek między zmiennymi:

- Najsilniejsza korelacja to -1 lub 1

- Najsłabsza korelacja to 0

• Dodatnia korelacja oznacza, że gdy jedna zmienna wzrasta, druga również ma tendencję do wzrostu

• Ujemna korelacja oznacza, że gdy jedna zmienna wzrasta, druga ma tendencję do zmniejszania się

Możemy użyć Pand, aby szybko pokazać nam współczynniki korelacji między każdą funkcją a każdą inną funkcją w Dataframe, jak pokazano na rysunku:

```
# correlation between variables
df.corr()
```

	friends	happiness
friends	1.000000	-0.216199
happiness	-0.216199	1.000000

Ta tabela pokazuje korelację między przyjaciółmi a szczęściem. Zwróć uwagę na dwie pierwsze rzeczy, pokazane w następujący sposób:

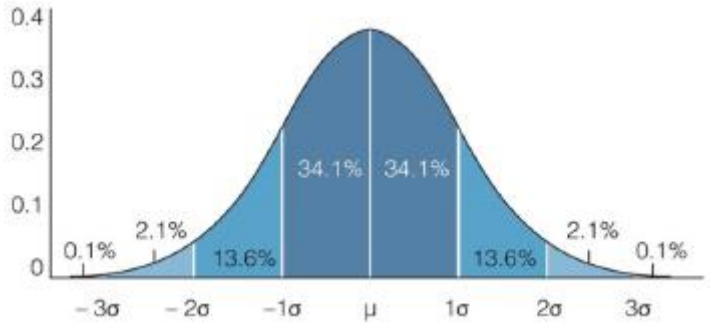
• Przekątna matrycy jest wypełniona pozytywem. Dzieje się tak, ponieważ reprezentują one korelację między zmienną a nią samą, która oczywiście tworzy idealną linię, dzięki czemu korelacja jest idealnie dodatnia!

• Matryca jest symetryczna na przekątnej. Dotyczy to każdej macierzy korelacji wykonanej w Pandas.

Istnieje kilka zastrzeżeń do zaufania do współczynnika korelacji. Po pierwsze, ogólnie rzecz biorąc, korelacja będzie próbowała zmierzyć liniową zależność między zmiennymi. Oznacza to, że jeśli nie ma widocznej korelacji ujawnionej przez tę miarę, nie oznacza to, że nie ma związku między zmiennymi, a jedynie, że nie ma linii najlepszego dopasowania, która łatwo przechodzi przez te linie. Może istnieć nieliniowa relacja, która definiuje te dwie zmienne. Ważne jest, aby zdać sobie sprawę, że związek przyczynowy nie jest implikowany przez korelację. Tylko dlatego, że istnieje słaba ujemna korelacja między tymi dwiema zmiennymi, niekoniecznie oznacza to, że ogólne szczęście spada wraz ze wzrostem liczby znajomych, których utrzymujesz na Facebooku. Ta przyczynowość musi być dalej testowana i w dalszej części postaramy się właśnie to zrobić. Podsumowując, możemy użyć korelacji, aby postawić hipotezy dotyczące związku między zmiennymi, ale będziemy musieli użyć bardziej wyrafinowanych metod statystycznych i algorytmów uczenia maszynowego, aby utrwalić te założenia i hipotezy.

Empiryczna reguła

Przypomnijmy, że rozkład normalny jest zdefiniowany jako mający określony rozkład prawdopodobieństwa, który przypomina krzywą dzwonową. W statystykach uwielbiamy, gdy nasze dane zachowują się normalnie. Na przykład, jeśli mamy dane przypominające rozkład normalny, tak:



Reguła empiryczna mówi, że możemy oczekiwać, że pewna ilość danych będzie istnieć między zestawami odchyleń standardowych. W szczególności reguła empiryczna określa dla danych, które są dystrybuowane normalnie:

• około 68% danych mieści się w 1 odchyleniu standardowym

• około 95% danych mieści się w 2 odchyleniach standardowych

• około 99,7% danych mieści się w 3 odchyleniach standardowych

Na przykład zobaczmy, czy dane naszych znajomych z Facebooka to wytrzymują. Użyjmy naszej ramki danych, aby znaleźć odsetek osób, które mieszczą się w zakresie 1, 2 i 3 odchyleń standardowych od średniej, jak pokazano:

```
# znalezienie odsetka osób w obrębie jednego odchylenia standardowego od
średnia
w obrębie_1_std = df_skalowane[(df_skalowane['znajomi_skalowane'] <= 1) & (df_
scaled['friends_scaled'] >= -1)].shape[0]
within_1_std / float(df_scaled.shape[0])
# 0,75
# znalezienie odsetka osób w obrębie dwóch odchyleń standardowych od
średnia
w ciągu_2_std = df_skalowane[(df_skalowane['znajomi_skalowane'] <= 2) & (df_
scaled['friends_scaled'] >= -2)].shape[0]
within_2_std / float(df_scaled.shape[0])
# 0,916
# znalezienie odsetka osób w obrębie trzech odchyleń standardowych od
średnia
w_3_std = df_skalowane[(df_skalowane['przyjaciele_skalowane'] <= 3) & (df_
scaled['friends_scaled'] >= -3)].shape[0]
within_3_std / float(df_scaled.shape[0])
# 1,0
```

Widzimy, że nasze dane wydają się być zgodne z regułą empiryczną. Około 75% ludzi mieści się w obrębie jednego odchylenia standardowego średniej. Około 92% ludzi mieści się w granicach dwóch odchyleń standardowych, a wszystkie mieszczą się w granicach trzech odchyleń standardowych.

Przykład - wyniki egzaminu

Załóżmy, że mierzymy wyniki egzaminu, a wyniki zazwyczaj mają rozkład normalny w kształcie dzwonu. Średnia z egzaminu wyniosła 84%, a odchylenie standardowe 6%. Z pewną dozą pewności możemy powiedzieć, że:

• Około 68% klasy uzyskało od 78% do 90%, ponieważ 78 to 6 jednostek poniżej 84, a 90 to 6 jednostek powyżej 84

• Gdybyśmy zostali zapytani, jaki procent klasy uzyskał wynik między 72 a 96%, zauważylibyśmy, że 72 to 2 odchylenia standardowe poniżej średniej, a 96 to 2 odchylenia standardowe powyżej średniej, więc reguła empiryczna mówi nam, że około 95 % klasy, która zdobyła punkty w tym zakresie.

Jednak nie wszystkie dane mają rozkład normalny, więc nie zawsze możemy zastosować regułę empiryczną. Mamy inne twierdzenie, które pomaga nam analizować każdy rodzaj dystrybucji. W następnym rozdziale omówimy szczegółowo, kiedy możemy przyjąć rozkład normalny. Dzieje się tak, ponieważ wiele testów statystycznych i hipotez wymaga, aby dane bazowe pochodziły z populacji o rozkładzie normalnym.

Wcześniej, gdy standaryzowaliśmy nasze dane do wyniku z, nie wymagaliśmy założenia rozkładu normalnego.

Podsumowanie

W tej części omówiliśmy wiele podstawowych statystyk wymaganych przez większość analityków danych. Omówiono wszystko, od tego, jak uzyskujemy/próbkujemy dane, po standaryzację danych zgodnie z z-score i zastosowaniami reguły empirycznej. W kolejnej części przyjrzymy się znacznie bardziej zaawansowanym aplikacjom statystyki. Jedną z rzeczy, które rozważymy, jest to, jak używać testów hipotez na danych, które możemy założyć, że są normalne. Korzystając z tych testów, będziemy również określać ilościowo nasze błędy i określać najlepsze praktyki w celu ich rozwiązania.

Zaawansowane statystyki

Zajmiemy się wnioskowaniem o całych populacjach na podstawie pewnych próbek danych. Będziemy używać testów hipotez wraz z różnymi testami estymacji, aby lepiej zrozumieć populacje na danych próbkach danych. Kluczowe tematy, które omówimy, to:

• Szacunki punktowe

• Przedziały ufności

• Centralne twierdzenie graniczne

• Testowanie hipotez

Szacunki punktowe

Przypomnijmy, że w poprzedniej części wspomnieliśmy, jak trudno było uzyskać parametr populacji; więc musieliśmy użyć przykładowych danych, aby obliczyć statystykę, która była oszacowaniem parametru. Kiedy dokonujemy tych szacunków, nazywamy je szacunkami punktowymi. Oszacowanie punktowe to oszacowanie parametru populacji na podstawie danych z próby. Szacunków punktowych używamy do oszacowania średnich populacji, wariancji i innych statystyk. Aby uzyskać te szacunki, po prostu stosujemy funkcję, którą chcemy zmierzyć dla naszej populacji, do próbki danych. Załóżmy na przykład, że istnieje firma która ma 9000 pracowników i interesuje nas ustalenie średniej długości przerw pracowników w ciągu jednego dnia. Ponieważ prawdopodobnie nie możemy zapytać każdej osoby, weźmiemy próbkę z 9000 osób i weźmiemy średnią z próby. Ta średnia z próby będzie naszym oszacowaniem punktowym. Poniższy kod jest podzielony na trzy części:

• Użyjemy rozkładu prawdopodobieństwa, zwanego rozkładem Poissona, aby losowo wygenerować 9000 odpowiedzi na pytanie: przez ile minut dziennie zwykle robisz sobie przerwy? Będzie to reprezentować naszą „populację". Pamiętaj, z Części 6, Prawdopodobieństwo zaawansowane, że zmienna losowa Poissona jest używana, gdy znamy średnią wartość zdarzenia i chcemy modelować rozkład wokół niej. Zauważ, że ta średnia wartość nie jest zwykle znana. Obliczam to, aby pokazać różnicę między naszym parametrem a naszą statystyką. Ustawiłem również losowy ziarno, aby zachęcić do powtarzalności (pozwala nam to za każdym razem uzyskać te same liczby losowe).

• Weźmiemy próbkę 100 pracowników (za pomocą metody losowej próby Pythona) i znajdziemy punktową ocenę średniej (nazywaną średnią z próby).Zauważ, że jest to nieco ponad 1% naszej populacji.

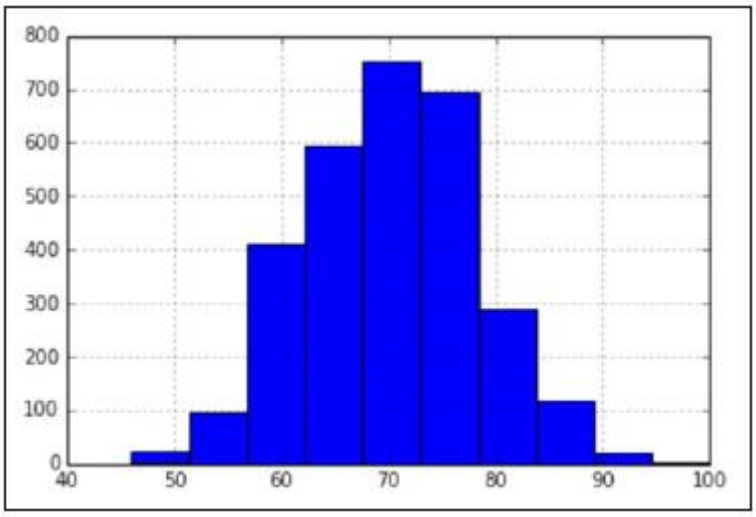
• Porównaj naszą średnią próby (średnią próby 100 pracowników) ze średnią naszej populacji.

Rzućmy okiem na następujący kod:

```
np.random.seed(1234)

long_breaks = stats.poisson.rvs(loc=10, mu=60, size=3000)

# represents 3000 people who take about a 60 minute break
```

Zmienna long_breaks reprezentuje 3000 odpowiedzi na pytanie: ile średnio minut robisz przerwy? i te odpowiedzi będą dłuższe. Zobaczmy wizualizację tego rozkładu, pokazaną w następujący sposób:

```
pd.Series(long_breaks).hist()
```

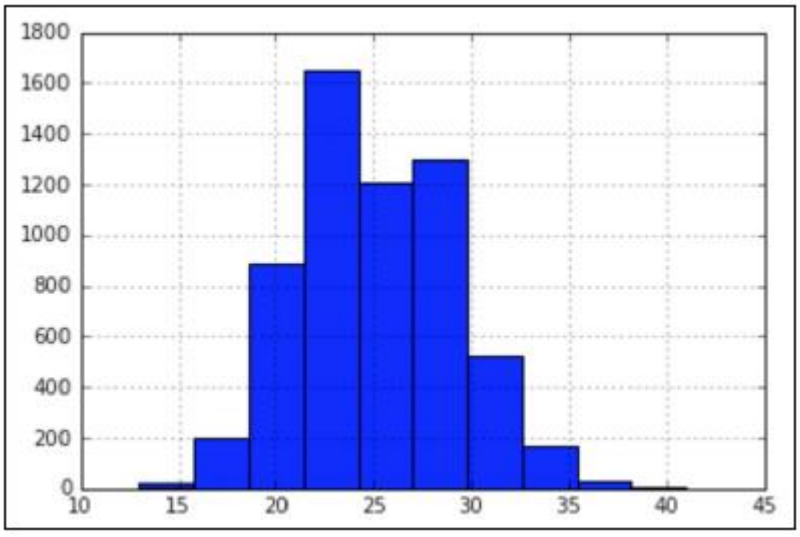


Widzimy, że nasza średnia 60 minut znajduje się na lewo od rozkładu. Ponadto, ponieważ zbadaliśmy tylko 3000 osób, nasze kosze są najwyższe, około 700-800 osób. Teraz weźmy model 6000 osób, które robią sobie średnio około 15 minut przerwy. Ponownie użyjmy rozkładu Poissona do symulacji 6000 osób, jak pokazano:

short_breaks = stats.poisson.rvs(loc=10, mu=15, size=6000)

represents 6000 people who take about a 15 minute break

pd.Series(short_breaks).hist()



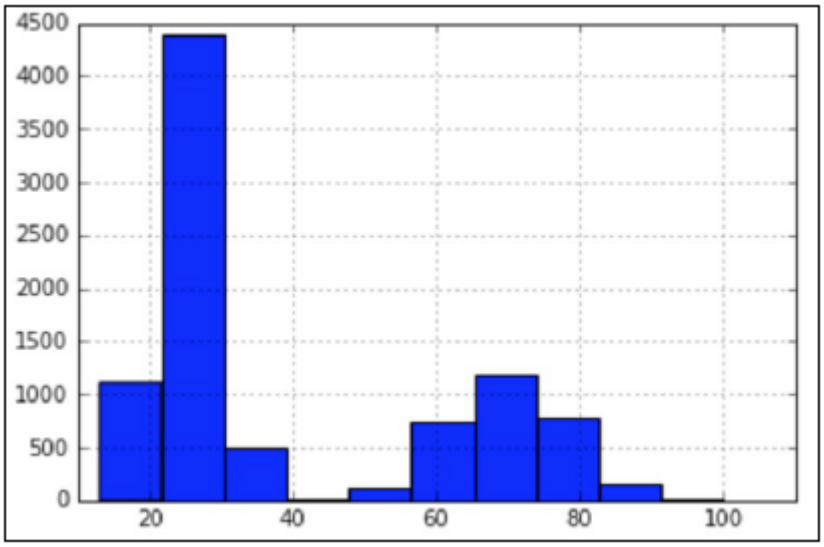
Ok, mamy więc rozkład dla osób, które robią dłuższe przerwy i rozkład dla osób, które robią sobie krótsze przerwy. Ponownie zauważ, jak nasza średnia długość 15-minutowej przerwy spada na lewą stronę rozkładu i zauważ, że najwyższy pasek ma około 1600 osób.

breaks = np.concatenate((long_breaks, short_breaks))

put the two arrays together to get our "population" of 9000 people

Zmienna breaks jest połączeniem wszystkich 9000 pracowników, zarówno biorących udział w długich, jak i krótkich przerwach. Zobaczmy całą dystrybucję osób na jednej wizualizacji:

pd.Series(breaks).hist()



Widzimy, że mamy dwa garby. Po lewej stronie mamy większy garb ludzi, którzy robią sobie około 15 minut przerwy, a po prawej mniejszy garb ludzi, którzy robią sobie dłuższe przerwy. Później przyjrzymy się dokładniej temu wykresowi. Całkowitą średnią długość przerwy możemy znaleźć, uruchamiając następujący kod:

breaks.mean()

39.99 minutes is our parameter.

Nasza średnia długość przerwy w firmie wynosi około 40 minut. Pamiętaj, że nasza populacja to zatrudniony w całej firmie 9000 osób, a nasz parametr to 40 minut. W prawdziwym świecie naszym celem byłoby oszacowanie parametru populacji, ponieważ z wielu powodów nie mielibyśmy zasobów, aby zapytać każdego pracownika w ankiecie o jego średnią długość przerwy. Zamiast tego użyjemy oszacowania punktowego. Tak więc, aby wyrazić naszą opinię, chcemy zasymulować świat, w którym pytamy 100 przypadkowych osób o długość ich przerw. Aby to zrobić, weźmy losową próbkę 100 pracowników spośród 9000 symulowanych pracowników, jak pokazano:

sample_breaks = np.random.choice(a = breaks, size=100)

taking a sample of 100 employees

Teraz weźmy średnią z próby i odejmijmy ją od średniej populacji i zobaczmy, jak daleko byliśmy:

breaks.mean() - sample_breaks.mean()

różnica między średnimi to 4,09 minuty, nieźle!

Jest to niezwykle interesujące, ponieważ mając tylko około 1% naszej populacji (100 z 9000), byliśmy w stanie uzyskać w ciągu 4 minut nasz parametr populacji i uzyskać bardzo dokładne oszacowanie naszej średniej populacji. Nie jest zły! Tutaj obliczyliśmy oszacowanie punktowe dla średniej, ale możemy to również zrobić dla parametrów proporcji. Proporcjonalnie mam na myśli stosunek dwóch wartości ilościowych. Załóżmy, że w firmie liczącej 10 000 osób nasi pracownicy są w 20% biali, 10% czarni, 10% Latynosi, 30% Azjaci, a 30% identyfikuje się jako inni. Weźmiemy próbkę 1000 pracowników i zobaczymy, czy ich proporcje rasowe są podobne.

```
employee_races = (["white"]*2000) + (["black"]*1000) +\
(["hispanic"]*1000) + (["asian"]*3000) +\
(["other"]*3000)
```

employee_races reprezentuje naszą populację pracowników. Na przykład w naszej firmie liczącej 10 000 osób 2000 osób to biali (20%), a 3000 to Azjaci (30%). Weźmy losową próbkę 1000 osób, jak pokazano:

```
demo_sample = random.sample(employee_races, 1000) # Sample 1000 values
for race in set(demo_sample):
print( race + " proportion estimate:" )
print( demo_sample.count(race)/1000. )
```

Otrzymany wynik byłby następujący:

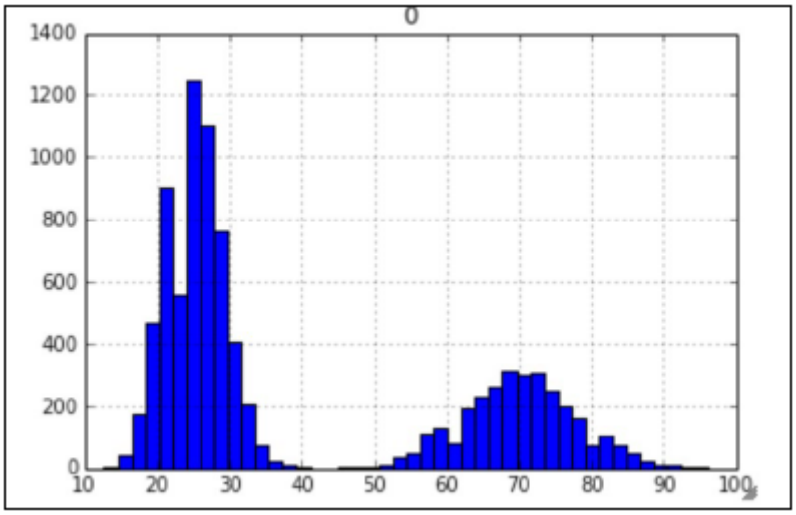
```
hispanic proportion estimate:
0.103
white proportion estimate:
0.192
other proportion estimate:
0.288
black proportion estimate:
0.1
asian proportion estimate:
0.317
```

Widzimy, że szacunki proporcji rasy są bardzo zbliżone do proporcji populacji bazowej. Na przykład w naszej próbce uzyskaliśmy 10,3% dla Latynosów, a odsetek populacji dla Latynosów wynosił 10%.

Rozkłady próbkowania

W Części 7 wspomnieliśmy, jak bardzo lubimy, gdy dane mają rozkład normalny. Jednym z powodów jest to, że wiele testów statystycznych (w tym te, których użyjemy w tym rozdziale) opiera się na danych, które mają normalny wzorzec, a większość danych ze świata rzeczywistego nie jest normalnych (zaskoczenie?). Weźmy na przykład dane o przerwach dla naszych pracowników, możesz pomyśleć, że po prostu chciałem tworzyć dane za pomocą rozkładu Poissona, ale miałem ku temu powód - szczególnie chciałem nienormalnych danych, jak pokazano:

```
pd.DataFrame(breaks).hist(bins=50,range=(5,100))
```



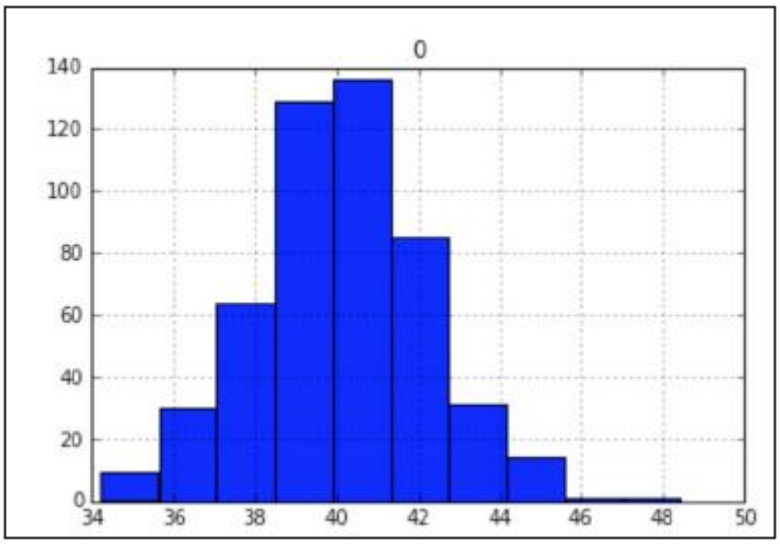
Jak widać, nasze dane zdecydowanie nie są zgodne z rozkładem normalnym, wydają się być bimodalne, co oznacza, że występują dwa szczyty czasów przerwy, około 25 i 70 minut. Ponieważ nasze dane nie są normalne, wiele najpopularniejszych testów statystycznych może nie mieć zastosowania, jednak jeśli będziemy postępować zgodnie z podaną procedurą, możemy utworzyć normalne dane! Myślisz, że jestem szalony? Cóż, przekonaj się sam. Po pierwsze, będziemy musieli wykorzystać tak zwany rozkład próbkowania, który jest rozkładem oszacowań punktowych kilku próbek o tej samej wielkości. Nasza procedura tworzenia rozkładu próbkowania będzie następująca:

1. Pobierz 500 różnych próbek czasów przerwy o rozmiarze 100 każda.

2. Zrób histogram tych 500 różnych oszacowań punktowych (ujawniając ich rozkład).

Liczba elementów w próbie (100) była dowolna, ale wystarczająco duża, aby stanowić reprezentatywną próbę populacji. Liczba pobranych przeze mnie próbek (500) była również dowolna, ale wystarczająco duża, aby zapewnić zbieżność naszych danych do rozkładu normalnego:

```
point_estimates = []

for x in range(500): # Generate 500 samples

sample = np.random.choice(a= breaks, size=100)

#take a sample of 100 points

point_estimates.append( sample.mean() )

# add the sample mean to our list of point estimates

pd.DataFrame(point_estimates).hist()

# look at the distribution of our sample means
```



Spójrz! Rozkład próbkowania średniej próbki wydaje się być normalny, mimo że pobraliśmy dane z bazowego rozkładu populacji bimodalnej. Należy zauważyć, że słupki na tym histogramie reprezentują średnią długość przerwy 500 próbek pracowników, z których każda zawiera 100 osób. Innymi słowy, rozkład próbkowania to rozkład kilku szacunków punktowych. Nasze dane zbliżyły się do rozkładu normalnego z powodu czegoś, co nazywa się centralnym twierdzeniem granicznym, które mówi, że rozkład próbkowania (rozkład oszacowań punktowych) zbliża się do rozkładu normalnego, gdy zwiększamy liczbę pobranych próbek.

Co więcej, w miarę jak pobieramy coraz więcej próbek, średnia rozkładu prób będzie zbliżać się do rzeczywistej średniej populacji, jak pokazano:

```
breaks.mean() - np.array(point_estimates).mean()

# .047 minutes difference
```

Jest to w rzeczywistości bardzo ekscytujący wynik, ponieważ oznacza, że możemy zbliżyć się nawet do oszacowania jednopunktowego, biorąc wiele oszacowań punktowych i wykorzystując centralne twierdzenie graniczne!

Ogólnie rzecz biorąc, w miarę zwiększania liczby pobieranych próbek, nasze oszacowanie zbliża się do parametru (wartości rzeczywistej).

Przedziały ufności

Chociaż oszacowania punktowe są prawidłowymi szacunkami parametru populacji, a rozkłady próbkowania są jeszcze lepsze, istnieją następujące dwa główne problemy związane z tymi podejściami:

• Szacunki jednopunktowe są bardzo podatne na błędy (między innymi ze względu na błąd próbkowania)

• Pobranie wielu próbek o określonej wielkości w celu uzyskania rozkładów próbkowania może być niewykonalne, a czasami może być nawet bardziej niewykonalne niż faktyczne znalezienie parametru populacji

Z tych i innych powodów, aby znaleźć statystyki, możemy sięgnąć do koncepcji zwanej przedziałem ufności.

Przedział ufności to zakres wartości oparty na oszacowaniu punktowym, który zawiera prawdziwy parametr populacji na pewnym poziomie ufności.

Zaufanie to ważna koncepcja w zaawansowanych statystykach. Jego znaczenie bywa błędnie rozumiane. Nieformalnie poziom ufności nie oznacza „prawdopodobieństwa poprawności"; zamiast tego reprezentuje częstotliwość, z jaką uzyskana odpowiedź będzie dokładna. Na przykład, jeśli chcesz mieć 95% szansy na uchwycenie prawdziwego parametru populacji przy użyciu tylko jednopunktowego oszacowania, musielibyśmy ustawić nasz poziom ufności na 95%.

Wyższe poziomy ufności skutkują szerszymi (większymi) przedziałami ufności w celu uzyskania większej pewności.

Obliczenie przedziału ufności obejmuje znalezienie oszacowania punktowego, a następnie włączenie marginesu błędu w celu utworzenia zakresu. Margines błędu to wartość, która reprezentuje naszą pewność, że nasze oszacowanie punktowe jest dokładne i oparte na pożądanym poziomie ufności, wariancji danych i wielkości próbki. Istnieje wiele sposobów obliczania przedziałów ufności; w celu zwięzłości i prostoty przyjrzymy się jednemu sposobowi przyjmowania przedziału ufności średniej populacji. Dla tego przedziału ufności potrzebujemy:

• Oszacowanie punktowe. W tym celu weźmiemy naszą przykładową średnią długości przerw z naszego poprzedniego przykładu.

• Oszacowanie odchylenia standardowego populacji, które reprezentuje wariancję danych.

- Oblicza się to, biorąc odchylenie standardowe próbki (odchylenie standardowe danych próbki) i dzieląc tę liczbę przez pierwiastek kwadratowy z wielkości populacji.

• Stopnie swobody (czyli wielkość próbki -1).

Uzyskanie tych liczb może wydawać się arbitralne, ale uwierz mi, wszystkie z nich mają powód. Jednak, dla uproszczenia, użyję gotowych modułów Pythona, jak pokazano, do obliczenia naszego przedziału ufności, a następnie zademonstrowania jego wartości:

```
sample_size = 100

# the size of the sample we wish to take

sample = np.random.choice(a= breaks, size = sample_size)

# a sample of sample_size taken from the 9,000 breaks population from

before

sample_mean = sample.mean()

# the sample mean of the break lengths sample

sample_stdev = sample.std()

# sample standard deviation

sigma = sample_stdev/math.sqrt(sample_size)

# population standard deviation estimate

stats.t.interval(alpha = 0.95, # Confidence level 95%
```

df= sample_size - 1, # Degrees of freedom

loc = sample_mean, # Sample mean

scale = sigma) # Standard deviation

(36.36, 45.44)

Powtórzmy, ten zakres wartości (od 36,36 do 45,44) reprezentuje przedział ufności dla średniego czasu przerwania z 95% ufnością. Wiemy już, że nasz parametr populacji wynosi 39,99 i zauważamy, że przedział obejmuje średnią populacji 39,99. Wspomniałem wcześniej, że poziom ufności nie był procentem dokładności naszego przedziału, ale procentową szansą, że przedział w ogóle będzie zawierał parametr populacji. Aby lepiej zrozumieć poziom ufności, weźmy 10 000 przedziałów ufności i zobaczmy, jak często średnia naszej populacji mieści się w przedziale. Najpierw utwórzmy funkcję, jak pokazano na ilustracji, która tworzy pojedynczy przedział ufności z naszych danych o rozbiciach:

function to make confidence interval

def makeConfidenceInterval():

sample_size = 100

sample = np.random.choice(a= breaks, size = sample_size)

sample_mean = sample.mean()

sample mean

sample_stdev = sample.std()

sample standard deviation

sigma = sample_stdev/math.sqrt(sample_size)

population Standard deviation estimate

return stats.t.interval(alpha = 0.95, df= sample_size - 1, loc =

sample_mean, scale = sigma)

Teraz, gdy mamy funkcję, która utworzy pojedynczy przedział ufności, stwórzmy procedurę, która przetestuje prawdopodobieństwo, że pojedynczy przedział ufności będzie zawierał prawdziwy parametr populacji 39,99:

1. Weź 10 000 przedziałów ufności średniej próbki.

2. Policz, ile razy parametr populacji mieści się w naszych przedziałach ufności.

3. Podaj stosunek liczby razy, gdy parametr wpadł do przedziału o 10 000:

times_in_interval = 0.

for i in range(10000):

interval = makeConfidenceInterval()

if 39.99 >= interval[0] and 39.99 <= interval[1]:

if 39.99 falls in the interval

```
times_in_interval += 1

print times_in_interval / 10000

# 0.9455
```

Powodzenie! Widzimy, że około 95% naszych przedziałów ufności zawierało naszą rzeczywistą średnią populacji. Szacowanie parametrów populacji za pomocą oszacowań punktowych i przedziałów ufności jest stosunkowo prostą i potężną formą wnioskowania statystycznego. Przyjrzyjmy się również, jak zmienia się wielkość przedziałów ufności, gdy zmieniamy nasz poziom ufności. Obliczmy przedziały ufności dla wielu poziomów ufności i przyjrzyjmy się, jak duże są przedziały, patrząc na różnicę między tymi dwiema liczbami. Nasza hipoteza będzie taka, że gdy zwiększymy nasz poziom ufności, prawdopodobnie zobaczymy większe przedziały ufności, aby mieć pewność, że złapiemy prawdziwy parametr populacji:

```
for confidence in (.5, .8, .85, .9, .95, .99):

confidence_interval = stats.t.interval(alpha = confidence, df=

sample_size – 1, loc = sample_mean, scale = sigma)

length_of_interval = round(confidence_interval[1] - confidence_

interval[0], 2)

# the length of the confidence interval

print "confidence {0} has a interval of size {1}".

format(confidence, length_of_interval)
```

```
confidence 0.5 has an interval of size 2.56

confidence 0.8 has an interval of size 4.88

confidence 0.85 has an interval of size 5.49

confidence 0.9 has an interval of size 6.29

confidence 0.95 has an interval of size 7.51

confidence 0.99 has an interval of size 9.94
```

Widzimy, że gdy chcemy być „bardziej pewni” w naszym interwale, nasz interwał wydłuża się, aby to zrekompensować. Następnie przyjrzymy się naszej koncepcji poziomów ufności i przyjrzymy się testowaniu hipotez statystycznych, aby zarówno rozwinąć te tematy, jak i stworzyć (zazwyczaj) jeszcze silniejsze wnioski statystyczne.

Testy hipotez

Testy hipotez są jednymi z najczęściej stosowanych testów w statystyce. Występują w wielu formach; jednak wszystkie mają ten sam podstawowy cel. Test hipotez jest testem statystycznym, który służy do ustalenia, czy wolno nam założyć, że pewien warunek jest prawdziwy dla całej populacji, biorąc pod uwagę próbkę danych. Zasadniczo test hipotez jest testem na pewną hipotezę, którą mamy na temat całej populacji. Wynik testu mówi nam, czy powinniśmy wierzyć w hipotezę, czy odrzucić ją na rzecz innej. Możesz pomyśleć o ramach testów hipotez, aby określić, czy obserwowane dane próbki odbiegają od tego, czego można było oczekiwać od samej populacji. Teraz brzmi to jak trudne zadanie,

ale na szczęście Python przychodzi na ratunek i zawiera wbudowane biblioteki ułatwiające przeprowadzanie tych testów. Test hipotez na ogół obejmuje dwie przeciwstawne hipotezy dotyczące populacji. Nazywamy je hipotezą zerową i hipotezą alternatywną. Hipoteza zerowa jest testowanym stwierdzeniem i jest domyślną poprawną odpowiedzią; to jest nasz punkt wyjścia i nasza pierwotna hipoteza. Hipotezą alternatywną jest stwierdzenie, które sprzeciwia się hipotezie zerowej. Nasz test powie nam, której hipotezie powinniśmy zaufać, a którą odrzucić. Na podstawie przykładowych danych z populacji test hipotezy określa, czy należy odrzucić hipotezę zerową. Zwykle używamy wartości p (która jest oparta na naszym poziomie istotności), aby wyciągnąć ten wniosek. Bardzo powszechnym błędnym przekonaniem jest to, że testy hipotez statystycznych są zaprojektowane tak, aby wybrać bardziej prawdopodobną z dwóch hipotez. To jest niepoprawne. Test hipotezy będzie domyślnie przyjmował hipotezę zerową, dopóki nie będzie wystarczającej ilości danych, aby poprzeć alternatywną hipotezę. Oto kilka przykładów pytań, na które możesz odpowiedzieć za pomocą testu hipotezy:

• Czy średni czas przerwy pracowników różni się od 40 minut?

• Czy istnieje różnica między osobami, które weszły w interakcję z witryną A, a osobami, które weszły w interakcję z witryną B (testy A/B)?

• Czy próbka ziaren kawy znacznie różni się smakiem od całej populacji ziaren?

Przeprowadzanie testu hipotez

Istnieje wiele rodzajów testów hipotez, a wśród nich są dziesiątki różnych procedur i metryk. Niemniej jednak istnieje pięć podstawowych kroków, które wykonuje większość testów hipotez, które są następujące:

1. Określ hipotezy:

- Tutaj formułujemy nasze dwie hipotezy: zerową i alternatywną

- Zwykle używamy zapisu H0 do przedstawienia hipotezy zerowej, a Ha do przedstawienia naszej alternatywnej hipotezy

2. Określ wielkość próbki dla próbki testowej:

- To obliczenie zależy od wybranego testu. Zwykle musimy określić odpowiednią wielkość próby, aby wykorzystać twierdzenia, takie jak centralne twierdzenie graniczne, i założyć normalność danych.

3. Wybierz poziom istotności (zwykle nazywany alfa lub α):

- Poziom istotności 0,05 jest powszechny

4. Zbierz dane:

- Zbierają próbkę danych do przeprowadzenia testu

5. Zdecyduj, czy odrzucić lub nie odrzucić hipotezy zerowej:

- Ten krok zmienia się nieznacznie w zależności od typu używanego testu.

Ostatecznym wynikiem będzie albo odrzucenie hipotezy zerowej na rzecz alternatywy, albo nieodrzucenie hipotezy zerowej. W tym rozdziale przyjrzymy się następującym trzem rodzajom testów hipotez:

• Testy t dla jednej próbki

• Dobroć dopasowania chi-kwadrat

• Test chi-kwadrat na asocjację/niezależność

Testów jest znacznie więcej. Jednak te trzy są świetną kombinacją odrębnych, prostych i potężnych testów. Jedną z najważniejszych rzeczy do rozważenia przy wyborze testu, który powinniśmy wdrożyć, jest rodzaj danych, z którymi pracujemy, a konkretnie, czy mamy do czynienia z danymi ciągłymi czy kategorycznymi. Aby naprawdę zobaczyć skutki hipotezy, proponuję zagłębić się w przykład. Najpierw przyjrzyjmy się wykorzystaniu testów t do radzenia sobie z danymi ciągłymi.

Jedna próbka t-testów

t-test dla jednej próbki jest testem statystycznym używanym do określenia, czy próbka danych ilościowych (numerycznych) różni się znacząco od innego zestawu danych (populacji lub innej próbki). Załóżmy, że w naszym poprzednim przykładzie przerw dla pracowników przyjrzymy się w szczególności przerwom w dziale inżynieryjnym, jak pokazano:

```
long_breaks_in_engineering = stats.poisson.rvs(loc=10, mu=55,

size=100)

short_breaks_in_engineering = stats.poisson.rvs(loc=10, mu=15,

size=300)

engineering_breaks = np.concatenate((long_breaks_in_engineering,

short_breaks_in_engineering))

print breaks.mean()

# 39.99

print engineering_breaks.mean()

# 34.825
```

Zauważ, że zastosowałem to samo podejście, co w przypadku oryginalnych czasów przerwy, ale z następującymi dwiema różnicami:

• Pobrałem mniejszą próbkę z rozkładu Poissona (aby zasymulować, że pobraliśmy próbkę 400 osób z działu inżynieryjnego)

• Zamiast używać mi 60, jak poprzednio, użyłem 55, aby zasymulować fakt, że zachowanie działu inżynieryjnego podczas przerw nie jest dokładnie takie samo, jak zachowanie firmy jako całości

Łatwo zauważyć, że istnieje różnica (ponad 5 minut) między działem inżynieryjnym a firmą jako całością. Zwykle nie mamy do dyspozycji całej populacji i parametrów populacji, ale mam je zasymulowane, aby zobaczyć przykład pracy. Tak więc, mimo że my (wszechwiedzący czytelnicy) widzą różnicę, załóżymy, że nic nie wiemy o tych parametrach populacji, a zamiast tego polegamy na teście statystycznym, aby ustalić te różnice.

Przykład jednego przykładowego testu t

Naszym celem jest tutaj ustalenie, czy istnieje różnica między czasami przerw w całej populacji (pracowników firmy) a czasami przerw pracowników działu inżynieryjnego. Przeprowadźmy teraz test

t przy 95% poziomie ufności, aby znaleźć różnicę (lub nie!). Technicznie rzecz biorąc, ten test powie nam, czy próbka pochodzi z tego samego rozkładu co populacja.

Założenia jednopróbkowych t-testów

Zanim przejdziemy do pięciu kroków, musimy najpierw przyznać, że t-testy muszą spełniać dwa następujące warunki, aby działały prawidłowo:

• Rozkład populacji powinien być normalny lub próba powinna być duża ($n \geq 30$).

• Aby założyć, że próba jest losowo losowana niezależnie, wystarczy wymusić, aby liczebność populacji była co najmniej 10 razy większa niż liczebność próby ($10n < N$).

Zwróć uwagę, że nasz test wymaga, aby dane bazowe były normalne (co wiemy, że nie są dla nas prawdziwe) lub aby wielkość próbki była co najmniej 30 punktów. W przypadku testu t warunek ten jest wystarczający do przyjęcia normalności. Ten test wymaga również niezależności, którą zapewnia pobranie odpowiednio małej próby. Brzmi dziwnie, prawda? Podstawową ideą jest to, że nasza próba musi być wystarczająco duża, aby przyjąć normalność (poprzez wnioski podobne do centralnego twierdzenia granicznego), ale na tyle mała, aby była niezależna od populacji. Przejdźmy teraz do naszych pięciu kroków:

1. Określ hipotezy. Pozwolimy, aby H₀ = dział inżynierski robił sobie przerwy tak samo jak firma jako całość. Jeśli pozwolimy, aby była to średnia firmy, możemy napisać:

H:

Zauważ, że jest to nasza hipoteza zerowa, czyli domyślna. Tak byśmy założyli, biorąc pod uwagę brak danych. Chcielibyśmy pokazać hipotezę alternatywną. Teraz, gdy mamy już kilka opcji dla naszej alternatywy, możemy albo powiedzieć, że średnia inżynieryjna (nazwijmy to tak) jest niższa niż średnia firmy, wyższa niż średnia firmy, albo po prostu inaczej (wyższa lub niższa) niż średnia firmy:

- Jeśli chcemy odpowiedzieć na pytanie, czy średnia z próby różni się od średniej firmy, to nazywamy to testem dwustronnym i nasza alternatywna hipoteza byłaby następująca:

H_a

- Jeśli chcemy odpowiedzieć, czy średnia z próby jest niższa od średniej firmy, czy też średnia z próby jest wyższa od średniej firmy, to mamy do czynienia z testem jednostronnym i nasza alternatywna hipoteza byłaby jedną lub drugą z poniższych hipotezy:

H_a:(praca inżynierska zajmuje dłuższe przerwy)

H_a :(praca inżynierska ma krótsze przerwy)

Różnica między jednym a dwoma ogonami polega na podzieleniu liczby później przez 2 lub nie. Proces pozostaje całkowicie niezmieniony dla obu. W tym przykładzie wybierzmy test dwustronny. Dlatego testujemy, czy ta próbka średnich czasów przerw w dziale inżynieryjnym różni się od średniej firmy. Nasz test zakończy się jednym z dwóch możliwych wniosków: albo odrzucimy hipotezę zerową, co oznacza, że czasy przerw w dziale inżynieryjnym różnią się od średniej firmy, albo nie odrzucimy hipotezy zerowej, co oznacza, że nie było nie ma wystarczających dowodów w próbce, aby poprzeć odrzucenie wartości null.

2. Określ wielkość próbki dla próbki testowej. Jak wspomniano wcześniej, większość testów (w tym ten) zakłada, że albo dane bazowe są normalne, albo nasza próbka jest prawidłowym zakresem.

- Próbka ma minimum 30 punktów (jest to 400)

- Próbka stanowi mniej niż 10% populacji (która wynosiłaby 900 osób)

3. Wybierz poziom istotności (zwykle nazywany alfa lub α). Wybierzemy 95% poziom istotności, co oznacza, że nasza alfa faktycznie może wynosić 1 - 0,95 = 0,05

4. Zbierz dane. Gotowe! Zostało to wygenerowane przez dwa rozkłady Poissona

5. Zdecyduj, czy odrzucić lub nie odrzucić hipotezy zerowej. Jak wspomniano wcześniej, ten krok różni się w zależności od użytego testu. Dla jednego przykładowego t-testu, musimy obliczyć dwie liczby: statystykę testową i naszą wartość p. Na szczęście możemy to zrobić w jednym wierszu w Pythonie. Statystyka testowa to wartość uzyskana z danych próbki podczas testu typu hipotezy. Służą do określenia, czy odrzucić hipotezę zerową. Statystyka testowa służy do porównania obserwowanych danych z oczekiwanymi w ramach hipotezy zerowej. Statystyka testowa jest używana w połączeniu z wartością p. Wartość p to prawdopodobieństwo, że zaobserwowane dane pojawiły się w ten sposób przypadkowo. Gdy dane wykazują bardzo mocne dowody przeciwko hipotezie zerowej, statystyka testu staje się duża (dodatnia lub ujemna), a wartość p zwykle staje się bardzo mała, co oznacza, że nasz test wykazuje bardzo dobre wyniki i prawdopodobnie , nie dzieje się to przypadkiem. W przypadku testu t wartość t jest naszą statystyką testową, jak pokazano:

t_statistic, p_value = stats.ttest_1samp(a= engineering_breaks,

popmean= breaks.mean())

Wprowadzamy zmienną engineering_breaks (która zawiera 400 czasów przerwy) oraz średnią populacji i otrzymujemy następujące liczby:

t_statistic == -5.742

p_value == .00000018

Wynik testu pokazuje, że wartość t wynosi -5,742. Jest to znormalizowana metryka, która ujawnia odchylenie średniej próbki od hipotezy zerowej. Wartość p daje nam ostateczną odpowiedź. Nasza wartość p mówi nam, jak często nasz wynik pojawiałby się przez przypadek. Na przykład, jeśli nasza wartość p wynosiła 0,06, oznaczałoby to, że powinniśmy spodziewać się przypadkowej obserwacji tych danych w około 6% przypadków. Oznacza to, że około 6% próbek dałoby takie wyniki. Interesuje nas, jak nasza wartość p ma się do naszego poziomu istotności:

• Jeśli wartość p jest mniejsza niż poziom istotności, możemy odrzucić hipotezę zerową

• Jeśli wartość p jest większa niż poziom istotności, to nie odrzuciliśmy hipotezy zerowej

Nasza wartość p jest znacznie niższa niż 0,05 (wybrany przez nas poziom istotności), co oznacza, że możemy odrzucić naszą hipotezę zerową na rzecz alternatywy. Oznacza to, że dział inżynierii wydaje się przyjmować inne długości przerw niż cała firma!

Stosowanie wartości p jest kontrowersyjne. Wiele czasopism faktycznie zakazało stosowania wartości p w testach istotności. Wynika to z natury wartości. Załóżmy, że nasza wartość p wyszła na 0,04. Oznacza to, że w 4% przypadków nasze dane przypadkowo pojawiały się w ten sposób i nie są w żaden sposób istotne. 4% to nie tak mały procent! Z tego powodu wiele osób przechodzi na różne testy statystyczne. Nie oznacza to jednak, że wartości p są bezużyteczne. Oznacza to po prostu, że musimy być ostrożni i świadomi tego, co mówi nam liczba. Istnieje wiele innych rodzajów testów t, w tym testy jednostronne (wspomniane wcześniej) i testy sparowane, a także dwa testy t dla próbek (o których

jeszcze nie wspomniano). Procedury te można łatwo znaleźć w literaturze statystycznej; jednak powinniśmy zwrócić uwagę na coś bardzo ważnego - co się dzieje, gdy robimy to źle.

Błędy typu I i typu II

O błędach typu I i typu II wspomnieliśmy w poprzedniej Części dotyczącym prawdopodobieństwa w przykładach klasyfikatora binarnego, ale dotyczą one również testów hipotez. Błąd typu I pojawia się, gdy odrzucamy hipotezę zerową, gdy jest ona rzeczywiście prawdziwa. Nazywa się to również fałszywym alarmem. Poziom błędu I typu jest równy poziomowi istotności α, co oznacza, że jeśli ustawimy wyższy poziom ufności, na przykład poziom istotności 99%, nasze α wynosi 0,01, a zatem nasz współczynnik fałszywych trafień wynosi 1%. Błąd typu II pojawia się, jeśli nie odrzucimy hipotezy zerowej, podczas gdy w rzeczywistości jest ona fałszywa. Jest to również znane jako fałszywie negatywny wynik. Im wyższy poziom ufności ustalimy, tym większe prawdopodobieństwo, że faktycznie zobaczymy błąd typu II.

Test hipotez dla zmiennych kategorialnych

t-testy (między innymi testami) to testy hipotez, które służą do porównywania i kontrastowania zmiennych ilościowych i leżących u ich podstaw rozkładów populacji. W tej sekcji przyjrzymy się dwóm nowym testom, z których oba służą do badania danych jakościowych. Oba są rodzajem testu zwanego testami chi-kwadrat. Te dwa testy wykonają dla nas następujące dwa zadania:

• Określ, czy próbka zmiennych kategorialnych pochodzi z określonej populacji (podobnie jak w t-teście)

• Określ, czy dwie zmienne wpływają na siebie nawzajem i są ze sobą powiązane.

Test zgodności chi-kwadrat

t-test dla jednej próby zastosowano do sprawdzenia, czy średnia próbki różniła się od średniej populacji. Test zgodności chi-kwadrat jest bardzo podobny do testu t dla jednej próbki, ponieważ sprawdza, czy rozkład danych próbki pasuje do oczekiwanego rozkładu, podczas gdy duża różnica polega na tym, że testuje zmienne kategorialne. Na przykład, test zgodności chi-kwadrat zostanie użyty do sprawdzenia, czy dane demograficzne rasy Twojej firmy są zgodne z danymi demograficznymi całego miasta w USA. Można go również użyć do sprawdzenia, czy użytkownicy Twojej witryny wykazują podobne cechy do przeciętnych internautów. Pracując z danymi kategorialnymi, musimy być ostrożni, ponieważ kategorie takie jak „mężczyzna", „kobieta" lub „inne" nie mają żadnego matematycznego znaczenia. Dlatego musimy wziąć pod uwagę liczebności zmiennych, a nie same rzeczywiste zmienne. Ogólnie rzecz biorąc, stosujemy test zgodności chi-kwadrat w następujących przypadkach:

• Chcemy przeanalizować jedną zmienną kategorialną z jednej populacji

• Chcemy określić, czy zmienna pasuje do określonego lub oczekiwanego rozkładu

W teście chi-kwadrat porównujemy to, co obserwujemy, z tym, czego oczekujemy.

Założenia testu zgodności chi-kwadrat

Istnieją dwa typowe założenia tego testu, które są następujące:

• Wszystkie oczekiwane liczby to co najmniej 5

• Poszczególne obserwacje są niezależne, a populacja powinna być co najmniej 10 razy większa niż próbka ($10n < N$)

Drugie założenie powinno wyglądać znajomo w teście t; jednak pierwsze założenie powinno wyglądać obco. Oczekiwane liczby to coś, o czym jeszcze nie rozmawialiśmy, ale wkrótce! Formułując nasze hipotezy zerowe i alternatywne dla tego testu, bierzemy pod uwagę domyślny rozkład zmiennych kategorialnych. Na przykład, jeśli mamy kostkę i testujemy, czy wyniki pochodzą z uczciwej kości, nasza hipoteza może wyglądać następująco:

H_0: Podany rozkład zmiennej kategorialnej jest poprawny.

p1 = 1/6, p2 = 1/6, p3 = 1/6, p4 = 1/6, p5 = 1/6, p6 = 1/6

Nasza alternatywna hipoteza jest dość prosta, jak pokazano:

H_a : Podany rozkład zmiennej kategorialnej jest nieprawidłowy. Przynajmniej jedna z wartości pi jest nieprawidłowa.

W teście t wykorzystaliśmy naszą statystykę testową (wartość t), aby znaleźć naszą wartość p. W teście chi-kwadrat nasza statystyka testu to, cóż, chi-kwadrat.

Statystyka testu X^2 = ponad k kategorii

Stopnie swobody = k − 1

Wartość krytyczna jest wtedy, gdy używamy χ2 oraz naszych stopni swobody i naszego poziomu istotności, a następnie odrzucamy hipotezę zerową, jeśli wartość p jest poniżej naszego poziomu istotności (takiego samego jak poprzednio).

Zobaczmy przykład, aby zrozumieć dalej.

Przykład testu chi-kwadrat na dobroć dopasowania

CDC dzieli BMI dla dorosłych na cztery klasy: poniżej/normalne, nadwaga, otyłość i ekstremalna otyłość. Badanie z 2009 roku wykazało, że rozkład dla dorosłych w USA wynosi odpowiednio 31,2%, 33,1%, 29,4% i 6,3%. W sumie losowo wybiera się 500 osób dorosłych i rejestruje ich kategorie BMI. Czy istnieją dowody sugerujące, że trendy BMI zmieniły się od 2009 roku? Testuj na poziomie istotności 0,05.

	Under/Normal	Over	Obesity	Extreme Obesity	Total
Observed	102	178	186	34	500

Najpierw obliczmy nasze oczekiwane wartości. W próbce 500 oczekujemy, że 156 będzie poniżej/normalnie (czyli 31,2% z 500), a pozostałe pola wypełniamy w ten sam sposób.

	Under/Normal	Over	Obesity	Extreme Obesity	Total
Observed	102	178	186	34	500
Expected	156	165.5	147	31.5	500

Najpierw sprawdź warunki:

• Wszystkie oczekiwane liczby są większe niż 5

• Każda obserwacja jest niezależna, a nasza populacja jest bardzo duża (znacznie ponad 10 razy po 500 osób)

Następnie przeprowadź test dopasowania. Postawimy nasze hipotezy zerowe i alternatywne:

• H₀ : Rozkład BMI z 2009 r. jest nadal poprawny.

• H_a : Rozkład BMI z 2009 r. nie jest już prawidłowy (przynajmniej jedna z proporcji jest teraz inna). Statystyki naszych testów możemy obliczyć ręcznie:

$$\text{Test Statistic: } \chi^2 = \sum \frac{(Observed - Expected)^2}{Expected} \text{ for } df = 3$$
$$= \frac{(102-156)^2}{156} + \frac{(178-165.5)^2}{165.5} + \frac{(186-147)^2}{147} + \frac{(34-31.5)^2}{31.5} = 30.18$$

Alternatywnie możemy użyć naszych przydatnych, dandysowych umiejętności Pythona, jak pokazano:

observed = [102, 178, 186, 34]

expected = [156, 165.5, 147, 31.5]

chi_squared, p_value = stats.chisquare(f_obs= observed, f_exp= expected)

chi_squared, p_value

#(30.1817679275599, 1.26374310311106e-06)

Nasza wartość p jest niższa niż 0,05; w związku z tym możemy odrzucić hipotezę zerową na rzecz faktu, że obecne trendy BMI różnią się od tych z 2009 roku.

Test chi-kwadrat na asocjację/niezależność

Niezależność jako pojęcie prawdopodobieństwa ma miejsce wtedy, gdy znajomość wartości jednej zmiennej nie mówi nic o wartości innej. Na przykład możemy oczekiwać, że kraj i miesiąc urodzenia są niezależne. Jednak wiedza, jakiego typu telefonu używasz, może wskazywać na Twój poziom kreatywności. Te zmienne mogą nie być niezależne. Test chi-kwadrat dla asocjacji/niezależności pomaga nam ustalić, czy dwie zmienne kategoryczne są od siebie niezależne. Test niezależności jest powszechnie używany do określenia, czy zmienne, takie jak poziom wykształcenia lub przedziały podatkowe, różnią się w zależności od czynników demograficznych, takich jak płeć, rasa i religia. Spójrzmy wstecz na przykład przedstawiony w poprzednim rozdziale, test podziału A/B. Przypomnijmy, że przeprowadziliśmy test i wystawiliśmy połowę naszych użytkowników na określoną stronę docelową (Witryna A), drugą połowę na inną stronę docelową (Witryna B), a następnie zmierzyliśmy współczynniki rejestracji w obu witrynach. Uzyskaliśmy następujące wyniki:

	Did not sign up	Signed up
Website A	134	54
Website B	110	48

Obliczyliśmy konwersje w witrynie, ale tak naprawdę chcemy wiedzieć, czy istnieje różnica między tymi dwiema zmiennymi: na którą witrynę miał kontakt użytkownik? i czy użytkownik się zarejestrował?. W tym celu użyjemy naszego testu chi-kwadrat.

Założenia testu niezależności chi-kwadrat

Istnieją dwa następujące założenia tego testu:

• Wszystkie oczekiwane liczby to co najmniej 5

• Poszczególne obserwacje są niezależne, a populacja powinna być co najmniej 10 razy większa niż próbka (10n < N)

Zauważ, że są one dokładnie takie same jak w ostatnim teście chi-kwadrat. Postawmy nasze hipotezy:

• H_0 : Nie ma związku między dwiema zmiennymi kategorialnymi w badanej populacji

• H_0 : Dwie zmienne kategorialne są niezależne w badanej populacji

• H_a : Istnieje związek między dwiema zmiennymi kategorialnymi w populacji będącej przedmiotem zainteresowania

• H_a : dwie zmienne kategoryczne nie są niezależne w badanej populacji

Możesz zauważyć, że brakuje nam tutaj czegoś ważnego. Gdzie są oczekiwane liczby? Wcześniej mieliśmy wcześniejszą dystrybucję, aby porównać nasze obserwowane wyniki, ale teraz tego nie robimy. Z tego powodu będziemy musieli je stworzyć. Do obliczenia oczekiwanych wartości dla każdej wartości możemy użyć następującego wzoru. W każdej komórce taeli możemy wykorzystać:

Oczekiwana liczba = aby obliczyć naszą statystykę testu chi-kwadrat i nasze stopnie swobody

$$\text{Test Statistic: } \chi^2 = \sum \frac{(Observed_{r,c} - Expected_{r,c})^2}{Expected_{r,c}}$$
$$\text{over } r \text{ rows and } c \text{ columns}$$
$$\text{Degrees of Freedom} = (r-1)\cdot(c-1)$$

Tutaj r to liczba wierszy, a c to liczba kolumn. Oczywiście, tak jak poprzednio, kiedy obliczamy naszą wartość p, odrzucimy wartość null, jeśli ta wartość p jest mniejsza niż poziom istotności. Użyjmy kilku wbudowanych metod Pythona, jak pokazano, aby szybko uzyskać nasze wyniki:

```
observed = np.array([[134, 54],[110, 48]])

# built a 2x2 matrix as seen in the table above

chi_squared, p_value, degrees_of_freedom, matrix = stats.chi2_

contingency(observed= observed)

chi_squared, p_value

# (0.04762692369491045, 0.82724528704422262)
```

Widzimy, że nasza wartość p jest dość duża; w związku z tym nie odrzucamy naszej hipotezy zerowej i nie możemy powiedzieć z całą pewnością, że wyświetlenie określonej witryny internetowej ma jakikolwiek wpływ na rejestrację użytkownika. Nie ma związku między tymi zmiennymi.

Podsumowanie

Przyjrzeliśmy się różnym testom statystycznym, w tym testom chi-kwadrat i t, a także oszacowaniom punktowym i przedziałom ufności, aby ustalić parametry populacji na podstawie danych z próby. Udało nam się odkryć, że nawet przy niewielkich próbkach danych możemy poczynić potężne założenia dotyczące podstawowej populacji jako całości. Statystyka to bardzo szeroki i ekspansywny temat, którego nie da się naprawdę omówić w jednym rozdziale, jednak nasze zrozumienie tematu pozwoli nam kontynuować i porozmawiać o tym, jak możemy wykorzystać statystyki i prawdopodobieństwo, aby przekazać nasze pomysły za pomocą nauce o danych nauka w następnej części.

Przekazywanie danych

W tej Części omówiono różne sposoby komunikowania wyników z naszej analizy. Tutaj przyjrzymy się różnym stylom prezentacji, a także technikom wizualizacji. Celem tego rozdziału jest zebranie naszych wyników i umiejętność ich wyjaśnienia w spójny, zrozumiały sposób, tak aby każdy, niezależnie od tego, czy znał dane, czy nie, może zrozumieć i wykorzystać nasze wyniki. Wiele z tego, co omówimy, będzie dotyczyło tworzenia skutecznych wykresów za pomocą etykiet, klawiszy, kolorów i nie tylko. Przyjrzymy się również bardziej zaawansowanym technikom wizualizacji, takim jak równoległe wykresy współrzędnych. W tej części przyjrzymy się następującym tematom:

• Identyfikacja skutecznych i nieefektywnych wizualizacji

• Rozpoznawanie, kiedy wykresy próbują „oszukać" publiczność

• Umiejętność określenia związku przyczynowego i korelacji

• Konstruowanie atrakcyjnych wizualizacji, które oferują cenny wgląd

Dlaczego komunikacja ma znaczenie?

Umiejętność przeprowadzania eksperymentów i manipulowania danymi w języku kodowania nie wystarczy do prowadzenia praktycznej i stosowanej nauki o danych. Dzieje się tak, ponieważ nauka o danych jest na ogół tak dobra, jak jest wykorzystywana w praktyce. Na przykład, naukowiec zajmujący się danymi medycznymi może być w stanie przewidzieć prawdopodobieństwo, że turysta zachoruje na malarię w krajach rozwijających się z dokładnością >98%, jednak jeśli wyniki te zostaną opublikowane w słabo sprzedawanym czasopiśmie, a wzmianki online o badaniu są minimalne, ich przełomowe wyniki, które mogłyby potencjalnie zapobiec śmierci, nigdy nie ujrzą prawdziwego światła dziennego. Z tego powodu przekazywanie wyników jest prawdopodobnie tak samo ważne jak same wyniki. Znanym przykładem złego zarządzania dystrybucją wyników jest przypadek Gregora Mendla. Mendel jest powszechnie uznawany za jednego z twórców współczesnej genetyki. Jednak jego wyniki (w tym dane i wykresy) zostały dobrze przyjęte dopiero po jego śmierci. Mendel wysłał je nawet do Karola Darwina, który w dużej mierze zignorował dokumenty Mendla, spisane w nieznanych morawskich dziennikach. Generalnie istnieją dwa sposoby prezentacji wyników: werbalny i wizualny. Oczywiście zarówno werbalne, jak i wizualne formy komunikacji można podzielić na dziesiątki podkategorii, w tym slajdy, wykresy, artykuły prasowe, a nawet wykłady uniwersyteckie. Jednak możemy znaleźć wspólne elementy prezentacji danych, które mogą sprawić, że każdy w terenie będzie bardziej świadomy i skuteczny w swoich umiejętnościach komunikacyjnych. Przejdźmy od razu do skutecznych (i nieefektywnych) form komunikacji, zaczynając od wizualizacji.

Identyfikacja skutecznych i nieefektywnych wizualizacji

Głównym celem wizualizacji danych jest umożliwienie czytelnikowi szybkiego przetrawienia danych, w tym możliwych trendów, relacji i nie tylko. Idealnie byłoby, gdyby czytelnik nie musiał spędzać więcej niż 5-6 sekund na trawieniu pojedynczej wizualizacji. Z tego powodu musimy bardzo poważnie tworzyć wizualizacje i upewnić się, że tworzymy wizualizację tak skuteczną, jak to tylko możliwe. Przyjrzyjmy się czterem podstawowym typom wykresów: wykresom punktowym, wykresom liniowym, wykresom słupkowym, histogramom i wykresom skrzynkowym.

Wykresy punktowe

Wykres punktowy jest prawdopodobnie jednym z najprostszych do stworzenia wykresów. Odbywa się to poprzez utworzenie dwóch osi ilościowych i wykorzystanie punktów danych do reprezentacji obserwacji. Głównym celem wykresu punktowego jest podkreślenie relacji między dwiema zmiennymi

oraz, jeśli to możliwe, ujawnić korelację. Na przykład możemy spojrzeć na dwie zmienne: średnią liczbę godzin oglądania telewizji w ciągu dnia oraz skalę wydajności pracy od 0 do 100 (0 oznacza bardzo słabą wydajność, a 100 oznacza doskonałą wydajność). Celem jest znalezienie związku (jeśli istnieje) między oglądaniem telewizji a średnią wydajnością pracy. Poniższy kod symuluje ankietę wśród kilku osób, w której ujawniły one, ile telewizji oglądały średnio w ciągu dnia w porównaniu ze standardowym miernikiem wydajności pracy w firmie:

```
import pandas as pd

hours_tv_watched = [0, 0, 0, 1, 1.3, 1.4, 2, 2.1, 2.6, 3.2, 4.1, 4.4, 4.4, 5]
```

Ta linia kodu tworzy 14 przykładowych wyników ankiety osób odpowiadających na pytanie, ile godzin dziennie oglądają telewizję.

```
work_performance = [87, 89, 92, 90, 82, 80, 77, 80, 76, 85, 80, 75, 73, 72]
```

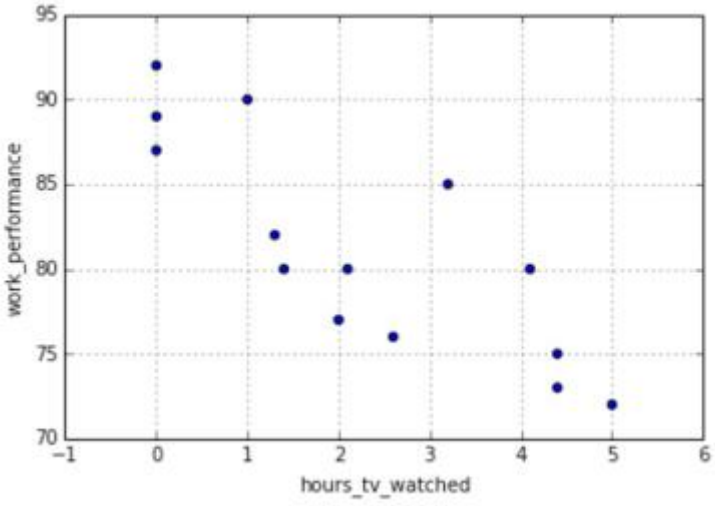
Ta linia kodu tworzy 14 nowych przykładowych wyników ankiety, w których te same osoby są oceniane pod kątem ich wydajności pracy w skali od 0 do 100. Na przykład pierwsza osoba oglądała 0 godzin telewizji dziennie i uzyskała ocenę 87/100 w swoim pracy, podczas gdy ostatnia osoba oglądała telewizję średnio 5 godzin dziennie i uzyskała ocenę 72/100:

```
df = pd.DataFrame({'hours_tv_watched':hours_tv_watched, 'work_

performance':work_performance})
```

Tutaj tworzymy Dataframe, aby ułatwić naszą eksploracyjną analizę danych i ułatwić tworzenie wykresu punktowego:

```
df.plot(x='hours_tv_watched', y='work_performance', kind='scatter')
```

Teraz tworzymy nasz wykres punktowy. Na poniższym wykresie widzimy, że nasze osie reprezentują liczbę godzin oglądania telewizji w ciągu dnia oraz wskaźnik wydajności pracy danej osoby:

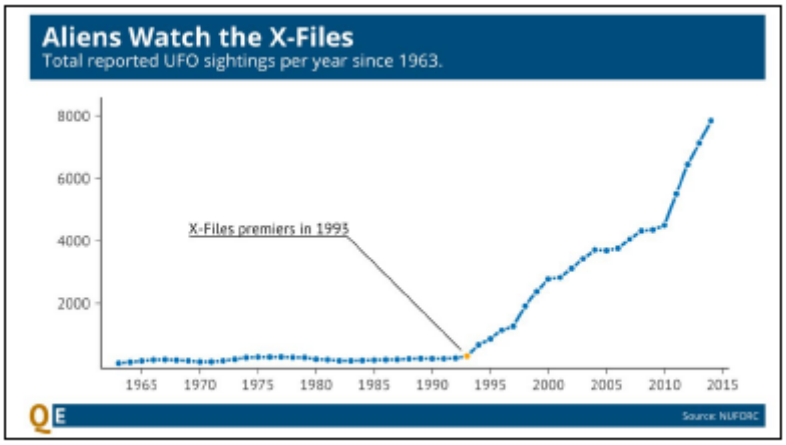


Każdy punkt na wykresie punktowym reprezentuje pojedynczą obserwację (w tym przypadku osobę), a jego lokalizacja jest wynikiem tego, gdzie obserwacja znajduje się na każdej zmiennej. Ten wykres punktowy wydaje się pokazywać związek, co oznacza, że oglądanie telewizji w ciągu dnia wydaje się wpływać na naszą wydajność w pracy. Oczywiście, ponieważ jesteśmy teraz ekspertami w statystykach z dwóch ostatnich rozdziałów, wiemy, że może to nie być przyczynowe. Wykres punktowy może działać tylko w celu ujawnienia korelacji lub związku między, ale nie związku przyczynowego. Zaawansowane

testy statystyczne, takie jak te, które widzieliśmy w części 8, Statystyki zaawansowane, mogą pomóc w ujawnieniu związku przyczynowego. W dalszej części tego rozdziału zobaczymy szkodliwe skutki, jakie może mieć korelacja zaufania.

Wykresy liniowe

Wykresy liniowe są prawdopodobnie jednym z najczęściej używanych wykresów w komunikacji danych. Wykres liniowy po prostu wykorzystuje linie do łączenia punktów danych i zwykle przedstawia czas na osi x. Wykresy liniowe to popularny sposób przedstawiania zmian zmiennych w czasie. Wykres liniowy, podobnie jak wykres punktowy, służy do wykreślania zmiennych ilościowych. Jako świetny przykład, wielu z nas zastanawia się nad możliwymi powiązaniami między tym, co oglądamy w telewizji, a naszym zachowaniem na świecie. Mój przyjaciel doprowadził kiedyś tę myśl do skrajności - zastanawiał się, czy mógłby znaleźć związek między programem telewizyjnym Z Archiwum X a liczbą obserwacji UFO w USA. rok i wykreślił je w czasie. Następnie dodał krótką grafikę, aby upewnić się, że czytelnicy będą w stanie zidentyfikować moment, w którym pliki X zostały wydane:



Source: http://www.questionable-economics.com/what-do-we-know-about-aliens/

Wydaje się jasne, że zaraz po 1993 roku, roku premiery Archiwum X, liczba obserwacji UFO zaczęła drastycznie rosnąć. Ta grafika, choć beztroska, jest doskonałym przykładem prostego wykresu liniowego. Mówi się nam, co mierzy każda oś, możemy szybko zobaczyć ogólny trend w danych i możemy utożsamić się z intencją autora, która polega na wykazaniu związku między liczbą obserwacji UFO a premierem pliku X. Z drugiej strony, poniższy wykres jest mniej imponującym wykresem liniowym:

Ten wykres liniowy próbuje uwydatnić zmianę ceny gazu poprzez wykreślenie trzech punktów w czasie. Na pierwszy rzut oka niewiele różni się od poprzedniego wykresu – mamy czas na dolnej osi x i wartość

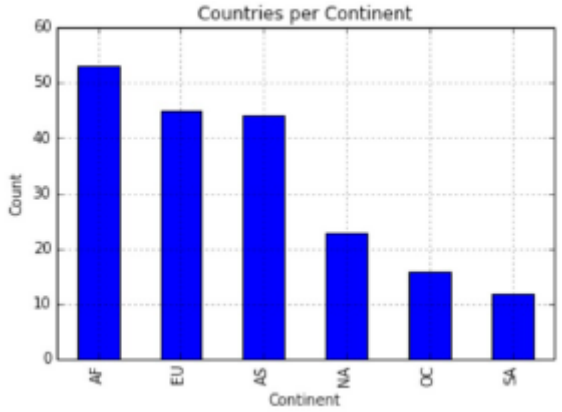
ilościową na pionowej osi y. Ta (nie tak) subtelna różnica polega na tym, że te trzy punkty są równomiernie rozmieszczone na osi x; jednak, jeśli czytamy ich rzeczywiste wskazania czasu, nie są one równomiernie rozłożone w czasie. Rok dzieli pierwsze dwa punkty, podczas gdy ostatnie dwa punkty dzieli zaledwie 7 dni.

Wykresy słupkowe

Na ogół korzystamy z wykresów słupkowych, gdy próbujemy porównać zmienne w różnych grupach. Na przykład możemy wykreślić liczbę krajów na kontynent za pomocą wykresu słupkowego. Zwróć uwagę, że oś x nie reprezentuje zmiennej ilościowej, w rzeczywistości podczas korzystania z wykresu słupkowego oś x jest ogólnie zmienną kategoryczną, podczas gdy oś y jest ilościowa. Zwróć uwagę, że w przypadku tego kodu korzystam z raportu Światowej Organizacji Zdrowia na temat spożycia alkoholu na świecie według krajów:

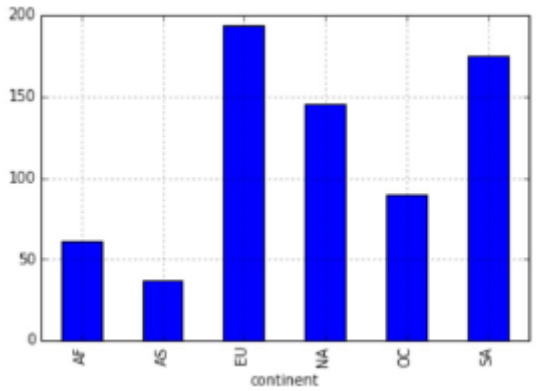
```
drinks = pd.read_csv('data/drinks.csv')

drinks.continent.value_counts().plot(kind='bar', title='Countries per Continent')

plt.xlabel('Continent')

plt.ylabel('Count')
```

Poniższy wykres pokazuje nam liczbę krajów na każdym kontynencie. Widzimy kod kontynentu na dole słupków, a wysokość słupka reprezentuje liczbę krajów, które mamy na każdym kontynencie. Na przykład widzimy, że Afryka ma najwięcej krajów reprezentowanych w naszym badaniu, podczas gdy Ameryka Południowa ma najmniej:



Oprócz liczby krajów możemy również wykreślić średnią porcji piwa na kontynent za pomocą wykresu słupkowego, jak pokazano:

```
drinks.groupby('continent').beer_servings.mean().plot(kind='bar')
```



Zwróć uwagę, że wykres punktowy lub wykres liniowy nie byłyby w stanie obsłużyć tych danych, ponieważ obsługują tylko zmienne ilościowe; wykresy słupkowe mają możliwość przedstawienia wartości kategorycznych. Możemy również używać wykresów słupkowych do tworzenia wykresów zmiennych w czasie, takich jak wykres liniowy.

Histogramy

Histogramy pokazują rozkład częstości pojedynczej zmiennej ilościowej, dzieląc dane według zakresu na równoodległe przedziały i wykreślając surową liczbę obserwacji w każdym przedziale. Histogram jest w rzeczywistości wykresem słupkowym, na którym oś x to przedział (podzakres) wartości, a oś y to liczba. Jako przykład zaimportuję dzienną liczbę unikalnych klientów sklepu, jak pokazano:

rossmann_sales = pd.read_csv('data/rossmann.csv')

rossmann_sales.head()

	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday
0	1	5	2015-07-31	5263	555	1	1	0	1
1	2	5	2015-07-31	6064	625	1	1	0	1
2	3	5	2015-07-31	8314	821	1	1	0	1
3	4	5	2015-07-31	13995	1498	1	1	0	1
4	5	5	2015-07-31	4822	559	1	1	0	1

Zwróć uwagę, że mamy wiele danych sklepu (w pierwszej kolumnie Sklep). Zróbmy podzbiór tych danych tylko dla pierwszego sklepu, jak pokazano:

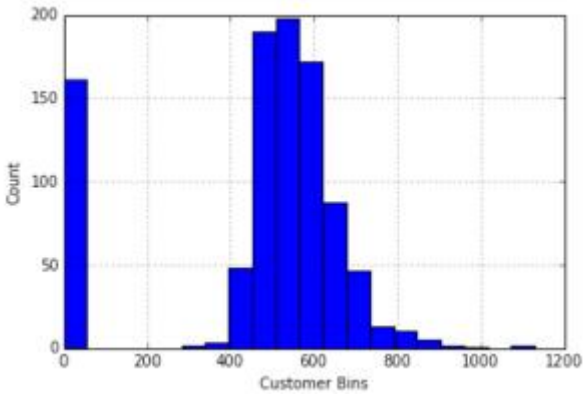
first_rossmann_sales = rossmann_sales[rossmann_sales['Store']==1]

Teraz narysujmy histogram liczby klientów pierwszego sklepu:

first_rossmann_sales['Customers'].hist(bins=20)

plt.xlabel('Customer Bins')

plt.ylabel('Count')



Oś x jest teraz kategoryczna, ponieważ każda kategoria jest wybranym zakresem wartości, na przykład 600-620 klientów byłoby potencjalnie kategorią. Oś y, podobnie jak wykres słupkowy, przedstawia liczbę obserwacji w każdej kategorii. Na tym wykresie, na przykład, można by odrzucić fakt, że w większości przypadków liczba klientów w danym dniu mieści się w zakresie od 500 do 700. W sumie histogramy służą do wizualizacji rozkładu wartości, które mogą przyjąć zmienną ilościową. W histogramie nie umieszczamy spacji między paskami.

Wykresy pudełkowe

Wykresy pudełkowe są również używane do pokazania rozkładu wartości. Są one tworzone przez wykreślenie podsumowania pięciu liczb w następujący sposób:

• Minimalna wartość

• Pierwszy kwartyl (liczba oddzielająca 25% najniższych wartości od pozostałych)

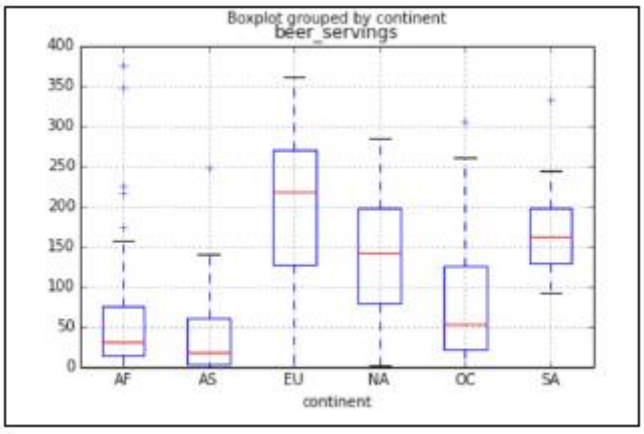
• Mediana

• Trzeci kwartyl (liczba oddzielająca 25% najwyższych wartości od pozostałych)

• Maksymalna wartość

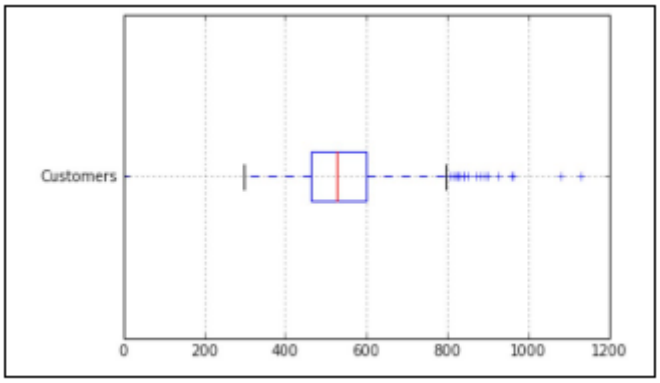
W Pandas, kiedy tworzymy wykresy pudełkowe, czerwona linia oznacza medianę, górna część pudełka (lub prawa, jeśli jest ustawiona poziomo) to trzeci kwartyl, a dolna (lewa) część pudełka to pierwszy kwartyl. Poniżej znajduje się seria wykresów pudełkowych pokazujących rozkład spożycia piwa według kontynentów:

drinks.boxplot(column='beer_servings', by='continent')

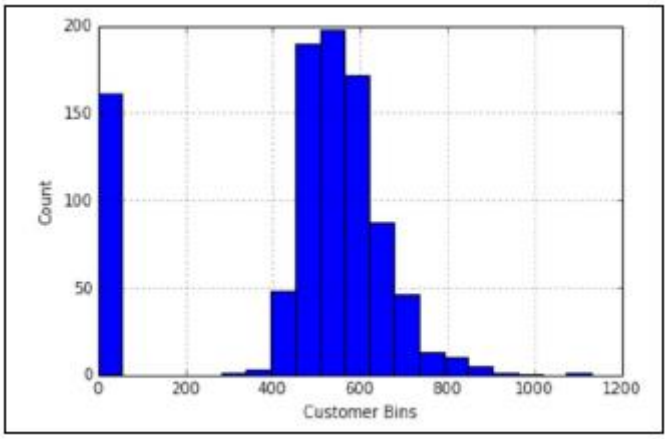


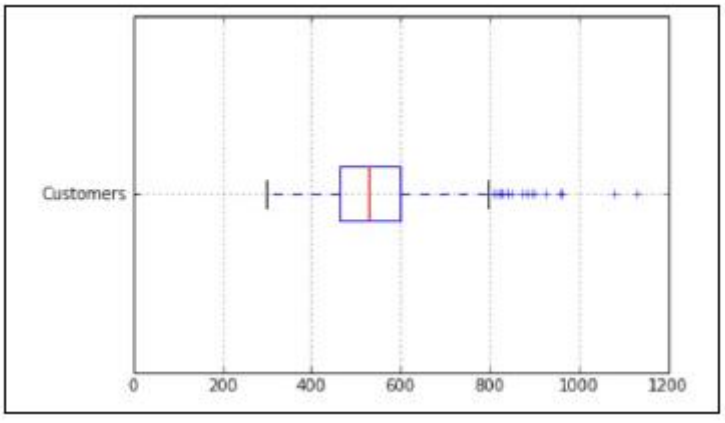
Teraz wyraźnie widzimy rozkład spożycia piwa na siedmiu kontynentach i różnice między nimi. Afryka i Azja mają znacznie niższą medianę spożycia piwa niż Europa czy Ameryka Północna. Wykresy pudełkowe mają również tę dodatkową zaletę, że są w stanie pokazać wartości odstające znacznie lepiej niż histogram. Dzieje się tak, ponieważ minimum i maksimum są częściami wykresu pudełkowego. Wracając do danych klientów, spójrzmy na te same numery klientów sklepu, ale za pomocą wykresu pudełkowego:

first_rossmann_sales.boxplot(column='Customers', vert=False)



Są to dokładnie te same dane, które przedstawiono wcześniej na histogramie; jednak teraz jest pokazany jako wykres pudełkowy. Dla porównania pokażę Wam oba wykresy jeden po drugim:





Zwróć uwagę, że oś x dla każdego wykresu jest taka sama, w zakresie od 0 do 1200. Wykres pudełkowy znacznie szybciej daje nam środek danych, czerwona linia jest medianą, podczas gdy histogram działa znacznie lepiej, pokazując nam, jak rozłożone są dane i gdzie znajdują się największe przedziały ludzi. Na przykład histogram pokazuje, że istnieje bardzo duży kosz, w którym jest zero osób. Oznacza to, że przez nieco ponad 150 dni danych nie było żadnych klientów. Zauważ, że możemy uzyskać dokładne liczby do skonstruowania wykresu pudełkowego za pomocą funkcji opisu w Pandas, jak pokazano:

first_rossmann_sales['Customers'].describe()

min 0.000000

25% 463.000000

50% 529.000000

75% 598.750000

max 1130.000000

Kiedy wykresy i statystyki kłamią

Powinnam jasno powiedzieć, statystyki nie kłamią, ludzie kłamią. Jednym z najłatwiejszych sposobów na oszukanie odbiorców jest pomylenie korelacji ze związkiem przyczynowym.

Korelacja a przyczynowość

W tym przykładzie będę nadal wykorzystywać moje dane dotyczące zużycia telewizora i wydajności pracy. Korelacja to metryka ilościowa od -1 do 1, która mierzy, jak dwie zmienne poruszają się względem siebie. Jeśli dwie zmienne mają korelację zbliżoną do -1 to znaczy, że gdy jedna zmienna rośnie, druga maleje, a jeśli dwie zmienne mają korelację bliską +1, to znaczy, że te zmienne poruszają się razem w tym samym kierunku - gdy jedna wzrasta , podobnie jak druga i na odwrót. Przyczynowość to idea, że jedna zmienna wpływa na inną. Na przykład możemy spojrzeć na dwie zmienne: średnią liczbę godzin oglądania telewizji w ciągu dnia oraz skalę wydajności pracy od 0 do 100 (0 oznacza bardzo słabą wydajność, a 100 oznacza doskonałą wydajność). Można by oczekiwać, że te dwa czynniki są ze sobą ujemnie skorelowane, co oznacza, że wraz ze wzrostem liczby godzin oglądania telewizji w ciągu doby spada ogólna wydajność pracy. Przypomnij sobie wcześniejszy kod, który wygląda następująco:

import pandas as pd

hours_tv_watched = [0, 0, 0, 1, 1.3, 1.4, 2, 2.1, 2.6, 3.2, 4.1, 4.4, 4.4, 5]

Tutaj patrzę na tę samą próbkę 14 osób co poprzednio i ich odpowiedzi na pytanie, ile godzin telewizji oglądasz średnio w ciągu nocy:

work_performance = [87, 89, 92, 90, 82, 80, 77, 80, 76, 85, 80, 75,73, 72]

To te same 14 osób, o których wspomnieliśmy wcześniej, w tej samej kolejności, ale teraz zamiast liczby godzin oglądania telewizji mamy ich wydajność pracy ocenianą przez firmę lub system firmy zewnętrznej:

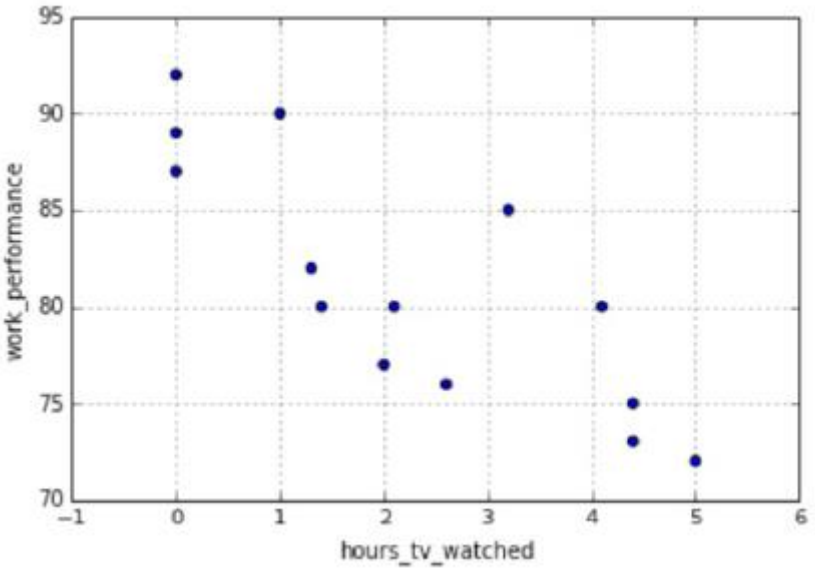
df = pd.DataFrame({'hours_tv_watched':hours_tv_watched, 'work_

performance':work_performance})

Wcześniej przyjrzeliśmy się wykresowi punktowemu tych dwóch zmiennych i wydawało się, że wyraźnie widać tendencję spadkową między tymi zmiennymi – w miarę wzrostu konsumpcji telewizji wydajność pracy wydawała się spadać. Jednak współczynnik korelacji, liczba od -1 do 1, to świetny sposób na identyfikację relacji między zmiennymi, a jednocześnie na ich ilościowe określenie i kategoryzację ich siły. Teraz możemy wprowadzić nowy wiersz kodu, który pokazuje nam korelację między tymi dwiema zmiennymi:

df.corr() # -0.824

Przypomnijmy, że korelacja bliska -1 implikuje silną korelację ujemną, podczas gdy korelacja bliska +1 implikuje silną korelację dodatnią. Liczba ta pomaga wesprzeć hipotezę, ponieważ współczynnik korelacji bliski -1 implikuje nie tylko korelację ujemną, ale i silną. Ponownie możemy to zobaczyć na wykresie punktowym między dwiema zmiennymi. Tak więc zarówno nasze wizualne, jak i nasze liczby są ze sobą dopasowane. Jest to ważna koncepcja, która powinna być prawdziwa podczas komunikowania wyników. Jeśli Twoje wizualizacje i liczby są niepoprawne, ludzie rzadziej traktują Twoją analizę poważnie:

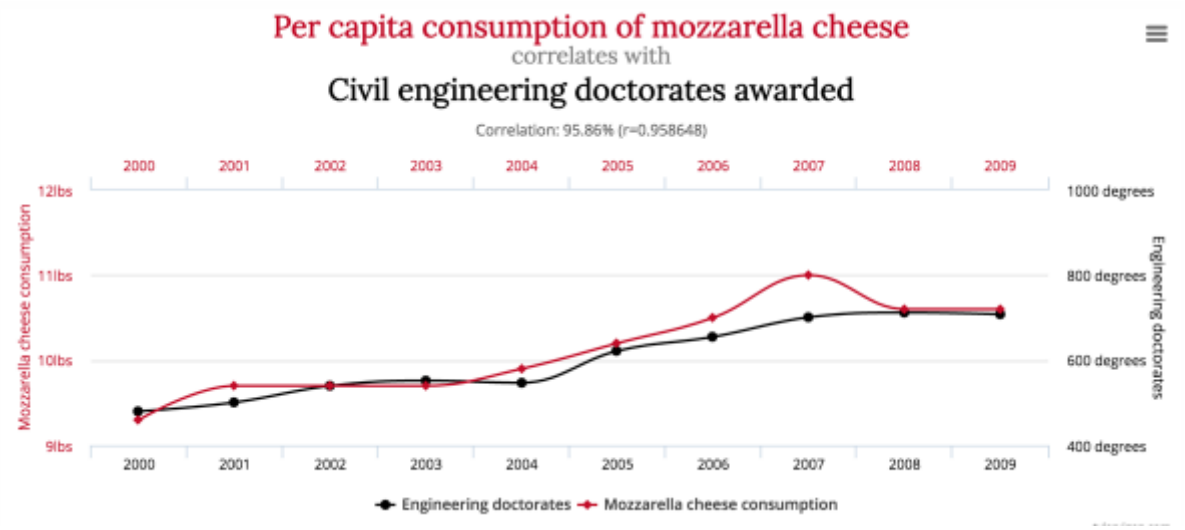


Nie mogę wystarczająco podkreślić, że korelacja i przyczynowość różnią się od siebie. Korelacja po prostu określa ilościowo stopień, w jakim zmienne zmieniają się razem, podczas gdy przyczynowość to idea, że jedna zmienna faktycznie określa wartość innej. Jeśli chcesz podzielić się wynikami swoich wyników pracy nad korelacjami, możesz spotkać się z prowokatorami na widowni, proszącymi o więcej pracy do wykonania. Co bardziej przerażające, nikt nie może wiedzieć, że analiza jest niekompletna, a Ty możesz podejmować praktyczne decyzje w oparciu o prostą pracę korelacyjną. Bardzo często zdarza się, że dwie zmienne mogą być ze sobą skorelowane, ale nie mają między sobą związku przyczynowego. Może to mieć różne przyczyny, niektóre są następujące:

• Może istnieć między nimi czynnik zakłócający. Oznacza to, że istnieje trzecia czająca się zmienna, która nie jest uwzględniana i działa jako pomost między tymi dwiema zmiennymi. Na przykład poprzednio pokazaliśmy, że może się okazać, że ilość oglądanej telewizji jest ujemnie skorelowana z wydajnością pracy, co oznacza, że wraz ze wzrostem liczby godzin oglądania telewizji ogólna wydajność pracy może się zmniejszyć. To jest korelacja. Sugerowanie, że oglądanie telewizji jest rzeczywistą przyczyną obniżenia jakości wykonywania pracy, wydaje się nie do końca słuszne. Bardziej prawdopodobna wydaje się sugestia, że istnieje trzeci czynnik, być może godziny snu każdej nocy, który

może odpowiedzieć na to pytanie. Być może oglądanie większej ilości telewizji zmniejsza ilość czasu na sen, co z kolei ogranicza wydajność pracy. Czynnikiem zakłócającym jest liczba godzin snu w ciągu nocy.

• Mogą nie mieć ze sobą nic wspólnego! To może być po prostu zbieg okoliczności. Istnieje wiele zmiennych, które są ze sobą skorelowane, ale po prostu się nie powodują. Rozważmy następujący przykład:



Jest o wiele bardziej prawdopodobne, że te dwie zmienne tylko korelują (mocniej niż w naszym poprzednim przykładzie, mogę dodać), że spożycie sera determinuje liczbę doktoratów z zakresu inżynierii lądowej na świecie. Prawdopodobnie słyszałeś, że korelacja stwierdzeń nie implikuje związku przyczynowego, a ostatni wykres jest dokładnie powodem, dla którego naukowcy zajmujący się danymi muszą w to wierzyć. Tylko dlatego, że istnieje matematyczna korelacja między zmiennymi, nie oznacza to, że mają między sobą związek przyczynowy. Mogą istnieć między nimi czynniki zakłócające lub po prostu mogą nie mieć ze sobą nic wspólnego! Zobaczmy, co się stanie, gdy zignorujemy zmienne zakłócające, a korelacje staną się niezwykle mylące.

Paradoks Simpsona

Paradoks Simpsona jest formalnym powodem, dla którego musimy poważnie traktować zmienne zakłócające. Paradoks mówi, że korelację między dwiema zmiennymi można całkowicie odwrócić, jeśli weźmiemy pod uwagę różne czynniki. Oznacza to, że nawet jeśli wykres może wykazywać pozytywną korelację, zmienne te mogą stać się antyskorelowane, gdy weźmie się pod uwagę inny czynnik (najprawdopodobniej mylący). Może to być bardzo kłopotliwe dla statystyków. Załóżmy, że chcemy zbadać relacje między dwiema różnymi stronami powitalnymi (przypomnijmy nasze poprzednie testy A/B w części 7, Statystyki podstawowe). Nazywamy te strony ponownie stroną A i stroną B. Mamy dwie strony powitalne, które chcemy porównać i skontrastować, a naszym głównym wskaźnikiem do wyboru będą współczynniki konwersji, tak jak wcześniej. Załóżmy, że przeprowadzamy wstępny test i znajdujemy następujące wyniki konwersji:

Page A	Page B
75% (263/350)	83% (248/300)

Oznacza to, że strona B ma prawie 10% wyższy współczynnik konwersji niż strona A. Tak więc od samego początku wydaje się, że strona B jest lepszym wyborem, ponieważ ma wyższy współczynnik konwersji. Gdybyśmy mieli przekazać te dane naszym kolegom, wydawałoby się, że wszystko jasne!

Zobaczmy jednak, co się stanie, gdy weźmiemy również pod uwagę wybrzeże, do którego użytkownik był bliżej, jak pokazano:

	Page A	Page B
West Coast	95% (76 / 80)	93% (231/250)
East Coast	72% (193/270)	34% (17/50)
Both	75% (263/350)	83% (248/300)

Zatem paradoks! Kiedy podzielimy próbkę według lokalizacji, wydaje się, że strona A była lepsza w obu kategoriach, ale ogólnie gorsza. Na tym polega piękno, a także przerażająca natura paradoksu. Dzieje się tak z powodu niezrównoważonych klas między czterema grupami. Grupa Strona A / Wschodnie Wybrzeże oraz grupa Strona B / Zachodnie Wybrzeże dostarczają większość osób w próbie, w związku z czym wyniki są nieoczekiwane. Mylącą zmienną może być tutaj fakt, że strony były wyświetlane o różnych porach dnia, a mieszkańcy zachodniego wybrzeża częściej widzieli stronę B, podczas gdy mieszkańcy wschodniego wybrzeża częściej odwiedzali stronę A. Paradoks Simpsona (a więc i odpowiedź), jednak dowód leży w złożonym systemie sieci bayesowskich i jest nieco poza zakresem tego tekstu. Głównym wnioskiem z paradoksu Simpsona jest to, że nie powinniśmy nadmiernie przypisywać mocy przyczynowej skorelowanym zmiennym. Mogą istnieć mylące zmienne, które należy zbadać. Dlatego, jeśli jesteś w stanie ujawnić korelację między dwiema zmiennymi (takimi jak kategoria strony internetowej i wskaźnik konwersacji lub konsumpcja telewizji i wydajność pracy), powinieneś bezwzględnie spróbować wyodrębnić jak najwięcej zmiennych, które mogą być przyczyną korelacji lub może przynajmniej pomóc w dalszym wyjaśnieniu twojej sprawy.

Jeśli korelacja nie implikuje związku przyczynowego, to co?

Jako naukowiec zajmujący się danymi, praca z korelacjami i niemożność wyciągnięcia rozstrzygającego związku przyczynowego jest często dość frustrująca. Najlepszym sposobem na pewne ustalenie związku przyczynowego jest zwykle przeprowadzanie losowych eksperymentów, takich jak te, które widzieliśmy w Części 8, Statystyki zaawansowane. Należałoby podzielić grupy populacji na losowo wybrane grupy i przeprowadzić testy hipotez, aby z pewną dozą pewności stwierdzić, że między zmiennymi istnieje prawdziwa przyczynowość.

Komunikacja werbalna

Oprócz wizualnej prezentacji danych, komunikacja werbalna jest równie ważna przy prezentowaniu wyników. Jeśli nie tylko przesyłasz wyniki lub publikujesz, zwykle prezentujesz dane w pokoju naukowców zajmujących się danymi, kierownictwie lub w sali konferencyjnej. W każdym razie istnieją kluczowe obszary, na których należy się skoncentrować podczas prezentacji werbalnej, zwłaszcza gdy prezentacja dotyczy ustaleń dotyczących danych. Generalnie istnieją dwa style prezentacji ustnych: jeden przeznaczony do bardziej profesjonalnych ustawień, w tym do biur korporacyjnych, gdzie dany problem jest zwykle związany bezpośrednio z wydajnością firmy lub innym KPI (kluczowym wskaźnikiem wydajności), a drugi przeznaczony jest bardziej do pokoju rówieśników, dla których kluczową ideą jest motywowanie odbiorców do dbania o Twoją pracę.

Chodzi o opowiadanie historii

Niezależnie od tego, czy jest to prezentacja formalna, czy zwykła, ludzie lubią słuchać historii. Kiedy przedstawiasz wyniki, nie tylko wypluwasz fakty i dane, ale próbujesz tak ukształtować umysły swoich odbiorców, aby wierzyli i dbali o to, co masz do powiedzenia. Podczas prezentacji zawsze bądź

świadomy swoich odbiorców i staraj się ocenić ich reakcje/zainteresowanie tym, co mówisz. Jeśli wydają się niezaangażowani, spróbuj odnieść im problem: „Pomyśl tylko, że kiedy powrócą popularne programy telewizyjne, takie jak Gra o tron, wszyscy Twoi pracownicy będą spędzać więcej czasu na oglądaniu telewizji, a zatem będą mieli niższą wydajność pracy". Teraz masz ich uwagę. Chodzi o odnoszenie się do odbiorców, bez względu na to, czy jest to twój szef, czy przyjaciel twojej mamy; musisz znaleźć sposób, aby było to istotne.

Po bardziej formalnej stronie rzeczy

Prezentując wyniki danych bardziej formalnym odbiorcom, lubię trzymać się następujących sześciu kroków:

1. Opisz stan problemu. Na tym etapie omawiamy aktualny stan problemu, w tym, na czym polega problem i w jaki sposób problem zwrócił uwagę zespołu analityków danych.

2. Określ charakter danych. Tutaj zagłębimy się w to, na kogo ten problem wpływa, w jaki sposób rozwiązanie zmieni sytuację oraz o wcześniejsze prace nad problemem, jeśli w ogóle.

3. Przedstaw początkową hipotezę. Tutaj określamy, co uważaliśmy za rozwiązanie przed wykonaniem jakiejkolwiek pracy. To może wydawać się bardziej początkującym podejściem do prezentacji; jednak może to być dobry moment, aby nakreślić nie tylko swoją początkową hipotezę, ale być może hipotezę całej firmy. Na przykład „przeprowadziliśmy ankietę i 61% firmy uważa, że nie ma korelacji między godzinami oglądania telewizji a wydajnością pracy".

4. Opisz rozwiązanie i ewentualnie narzędzia, które doprowadziły do rozwiązania. Zapoznaj się ze sposobem rozwiązania problemu, zastosowanymi testami statystycznymi i założeniami, które zostały przyjęte w trakcie rozwiązywania problemu.

5. Podziel się wpływem, jaki Twoje rozwiązanie będzie miało na problem. Porozmawiaj o tym, czy twoje rozwiązanie różniło się od początkowej hipotezy. Co to będzie oznaczać na przyszłość? Jak możemy podjąć działania na rzecz tego rozwiązania, aby poprawić siebie i naszą firmę?

6. Przyszłe kroki. Podziel się, jakie dalsze kroki można podjąć w związku z problemem, np. jak wdrożyć wspomniane rozwiązanie i jakie dalsze prace zaowocowały tym badaniem.

Postępując zgodnie z tymi krokami, możemy trafić we wszystkie główne obszary metody naukowej danych. Pierwszą rzeczą, na którą chcesz trafić podczas formalnej prezentacji, jest działanie. Chcesz, aby Twoje słowa i rozwiązania były wykonalne. Musi istnieć jasna ścieżka, którą należy obrać po zakończeniu projektu i należy określić przyszłe kroki.

Dlaczego/jak/jaka strategia prezentacji

Mówiąc na mniej formalnym poziomie, strategia dlaczego/jak/jak jest szybkim i łatwym sposobem na stworzenie godnej pochwały prezentacji. Jest to dość proste, jak pokazano:

1. Powiedz słuchaczom, dlaczego to pytanie jest ważne, nie zagłębiając się w to, co faktycznie robisz.

2. Następnie przejdź do sposobu, w jaki poradziłeś sobie z tym problemem, korzystając z eksploracji danych, czyszczenia danych, testowania hipotez i tak dalej.

3. Na koniec powiedz im, co Twoje wyniki oznaczają dla odbiorców.

Ten model jest zapożyczony ze znanych reklam. Taki, w którym nie powiedzieliby nawet, jaki jest produkt, dopóki nie pozostały 3 sekundy. Chcą przyciągnąć twoją uwagę, a potem wreszcie ujawnić, co było tak ekscytujące. Rozważmy następujący przykład:

„Witam wszystkich, jestem tutaj, aby opowiedzieć o tym, dlaczego wydaje nam się, że trudno jest nam skoncentrować się na naszej pracy, gdy nadawane są igrzyska olimpijskie. Po przeanalizowaniu wyników ankiet i połączeniu tych danych ze standardowymi danymi dotyczącymi wydajności pracy w firmie, udało mi się znaleźć korelację między liczbą godzin oglądania telewizji dziennie a średnią wydajnością pracy. Wiedząc o tym, wszyscy możemy być bardziej świadomi naszych nawyków oglądania telewizji i upewnić się, że nie wpływa to na naszą pracę. Dziękuję. "

Ta część została właściwie sformatowana w ten sposób! Zaczęliśmy od tego, dlaczego powinniśmy dbać o komunikację danych, następnie rozmawialiśmy o tym, jak to osiągnąć (poprzez korelację, wizualizacje itd.), a na koniec mówię Ci co, czyli dlaczego/jak/jaka strategia (wstaw tutaj niesamowity efekt dźwiękowy).

Podsumowanie

Komunikacja danych nie jest łatwym zadaniem. Jedną rzeczą jest zrozumienie matematyki działania nauki o danych, ale zupełnie inną rzeczą jest próba przekonania sali złożonej z naukowców zajmujących się danymi i nie zajmujących się danymi o swoich wynikach i ich wartości dla nich. W tym rozdziale omówiliśmy podstawowe tworzenie wykresów, a także sposoby identyfikacji wadliwych przyczyn i umiejętności doskonalenia umiejętności prezentacji ustnej. Nasze kilka następnych rozdziałów naprawdę zacznie trafiać w jeden z największych tematów do dyskusji w nauce o danych. W ostatnich dziewięciu Częściach mówiliśmy o wszystkim, między innymi o sposobach pozyskiwania danych, czyszczenia danych i wizualizacji danych w celu lepszego zrozumienia środowiska, które reprezentują dane. Następnie przyjrzeliśmy się podstawowym i zaawansowanym prawom prawdopodobieństwa/statystyki, aby użyć kwantyfikowalnych twierdzeń i testów na naszych danych w celu uzyskania praktycznych wyników i odpowiedzi. W kolejnych Częściach przyjrzymy się uczeniu maszynowemu i naturze, w której uczenie maszynowe działa dobrze, a w której nie działa dobrze. W miarę jak zagłębiamy się w ten materiał, zachęcam czytelnika, aby zachował otwarty umysł i naprawdę zrozumiał nie tylko, jak działa uczenie maszynowe, ale także dlaczego musimy z niego korzystać.

Jak stwierdzić, czy Twój toster się uczy - podstawy uczenia maszynowego

Uczenie maszynowe stało się frazą dekady. Wygląda na to, że za każdym razem, gdy słyszymy o kolejnym największym startupie lub włączamy wiadomości, słyszymy coś o rewolucyjnej technologii uczenia maszynowego i o tym, jak zmieni ona nasz styl życia. Ta część koncentruje się na uczeniu maszynowym jako praktycznej części nauki o danych. W tej części omówimy następujące tematy:

• Definiowanie różnych typów uczenia maszynowego wraz z przykładami każdego rodzaju

• Obszary regresji, klasyfikacji i nie tylko

• Co to jest uczenie maszynowe i jak jest wykorzystywane w nauce o danych

• Różnice między uczeniem maszynowym a modelowaniem statystycznym oraz to, że uczenie maszynowe jest szerszą kategorią tego ostatniego

Naszym celem będzie wykorzystanie statystyk, prawdopodobieństwa i myślenia algorytmicznego w celu zrozumienia i zastosowania podstawowych umiejętności uczenia maszynowego w praktycznych branżach, takich jak marketing. Przykładami będą przewidywanie gwiazdek recenzji restauracji, przewidywanie obecności choroby, wykrywanie spamu i wiele innych. Ta część skupia się bardziej na uczeniu maszynowym jako całości i pojedynczym modelu statystycznym. Kolejne części będą dotyczyć znacznie większej liczby modeli, z których niektóre są znacznie bardziej złożone. Skupimy się również na metrykach, które mówią nam, jak skuteczne są nasze modele. Wykorzystamy metryki w celu podsumowania wyników i prognozowania za pomocą uczenia maszynowego.

Co to jest uczenie maszynowe?

Kontynuowanie pracy bez konkretnej definicji tego, czym jest uczenie maszynowe, nie miałoby sensu. Cóż, cofnijmy się na chwilę. W Części 1, Jak brzmieć jak naukowiec danych zdefiniowaliśmy uczenie maszynowe jako dawanie komputerom możliwości uczenia się na podstawie danych bez narzucania im przez programistę wyraźnych reguł. Ta definicja nadal jest aktualna. Uczenie maszynowe dotyczy zdolności do ustalenia pewnych wzorców (sygnałów) z danych, nawet jeśli dane zawierają nieodłączne błędy (szum). Modele uczenia maszynowego są w stanie uczyć się na podstawie danych bez wyraźnej pomocy człowieka. To jest główna różnica między modelami uczenia maszynowego a klasycznymi algorytmami. Klasycznym algorytmom mówi się, jak znaleźć najlepszą odpowiedź w złożonym systemie, a następnie algorytm wyszukuje te najlepsze rozwiązania i często działa szybciej i wydajniej niż człowiek. Jednak wąskim gardłem jest to, że człowiek musi najpierw znaleźć najlepsze rozwiązanie. W uczeniu maszynowym modelowi nie podaje się najlepszego rozwiązania, a zamiast tego podaje kilka przykładów problemu i mówi mu, aby znaleźć najlepsze rozwiązanie. Uczenie maszynowe to tylko kolejne narzędzie w pasku narzędzi analityka danych. Jest na tym samym poziomie co testy statystyczne (testy chi-kwadrat lub t) lub wykorzystuje prawdopodobieństwo/statystykę bazową do oszacowania parametrów populacji. Uczenie maszynowe jest często uważane za jedyne, co wiedzą naukowcy zajmujący się danymi, a to po prostu nieprawda. Prawdziwy naukowiec danych jest w stanie rozpoznać, kiedy uczenie maszynowe ma zastosowanie, a co ważniejsze, kiedy nie. Uczenie maszynowe to gra korelacji i relacji. Większość istniejących algorytmów uczenia maszynowego dotyczy znajdowania i/lub wykorzystywania relacji między zestawami danych (często reprezentowanych jako kolumny w Pandas Dataframe). Gdy algorytmy uczenia maszynowego będą w stanie wskazać pewne korelacje, model może wykorzystać te relacje do przewidywania przyszłych obserwacji lub uogólnić dane w celu ujawnienia interesujących wzorców. Być może świetnym sposobem na wyjaśnienie uczenia maszynowego jest przedstawienie przykładu problemu połączonego z dwoma możliwymi

rozwiązaniami: jednym przy użyciu algorytmu uczenia maszynowego i inne wykorzystujące algorytm uczenia maszynowego.

Przykład - rozpoznawanie twarzy

Ten problem jest bardzo dobrze udokumentowany. Biorąc pod uwagę zdjęcie twarzy, do czyjej twarzy należy? Twierdzę jednak, że jest jeszcze ważniejsze pytanie, które należy zadać jeszcze przed tym. Załóżmy, że chcesz wdrożyć system bezpieczeństwa w domu, który rozpoznaje, kto wchodzi do Twojego domu. Najprawdopodobniej w ciągu dnia Twój dom będzie pusty przez większość czasu, a rozpoznawanie twarzy musi działać tylko wtedy, gdy w kadrze jest osoba. To jest właśnie pytanie, które proponuję, abyśmy spróbowali rozwiązać - czy biorąc pod uwagę zdjęcie, jest na nim twarz? Biorąc pod uwagę ten problem, proponuję następujące dwa rozwiązania:

• Algorytm uczenia maszynowego, który zdefiniuje twarz jako posiadającą okrągłą strukturę, dwoje oczu, włosy, nos i tak dalej. Algorytm następnie szuka tych zakodowanych na stałe funkcji na zdjęciu i zwraca, czy był w stanie znaleźć którąkolwiek z tych funkcji.

• Algorytm uczenia maszynowego będzie działał nieco inaczej. Modelka otrzyma tylko kilka zdjęć twarzy i osób niebędących twarzami, które są oznaczone jako takie. Z przykładów (zwanych zestawami szkoleniowymi) wymyśliłby własną definicję twarzy.

Wersja rozwiązania oparta na uczeniu maszynowym nigdy nie jest informowana, czym jest twarz, podaje tylko kilka przykładów, niektóre z twarzami, a niektóre bez. Następnie od modelu uczenia maszynowego zależy ustalenie różnicy między nimi. Gdy odkryje różnicę między nimi, wykorzystuje te informacje, aby zrobić zdjęcie i przewidzieć, czy na nowym zdjęciu jest twarz. Na przykład, aby trenować model, damy mu następujące trzy obrazy:

Model następnie określi różnicę między zdjęciami oznaczonymi jako Twarz a zdjęciami oznaczonymi jako Bez twarzy i będzie mógł wykorzystać tę różnicę do znalezienia twarzy na przyszłych zdjęciach.

Uczenie maszynowe nie jest idealne

Istnieje wiele zastrzeżeń dotyczących uczenia maszynowego. Wiele z nich jest specyficznych dla różnych wdrażanych modeli, ale istnieją pewne założenia, które są uniwersalne dla każdego modelu uczenia maszynowego, takie jak:

• Wykorzystywane dane są w większości wstępnie przetwarzane i czyszczone przy użyciu metod przedstawionych we wcześniejszych częściach. Prawie żaden model uczenia maszynowego nie będzie

tolerował brudnych danych z brakującymi wartościami lub wartościami kategorycznymi. Użyj fikcyjnych zmiennych i technik wypełniania/upuszczania, aby poradzić sobie z tymi rozbieżnościami.

• Każdy wiersz oczyszczonego zestawu danych reprezentuje pojedynczą obserwację środowiska, które próbujemy modelować.

• Jeśli naszym celem jest znalezienie relacji między zmiennymi, to zakłada się, że istnieje jakiś rodzaj związku między tymi zmiennymi. To założenie jest szczególnie ważne. Wiele modeli uczenia maszynowego traktuje to założenie bardzo poważnie. Te modele nie są w stanie komunikować, że może nie być związku.

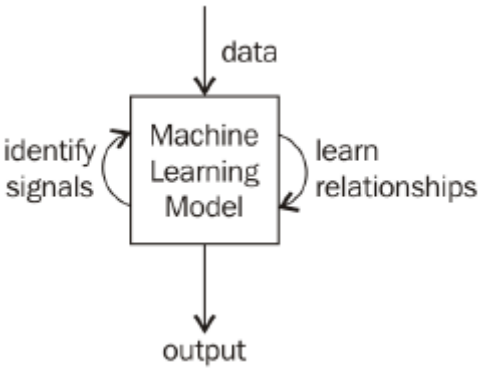
• Modele uczenia maszynowego są ogólnie uważane za półautomatyczne, co oznacza, że ludzie nadal potrzebują inteligentnych decyzji. Maszyna jest bardzo inteligentna, ale ma trudności z umieszczeniem rzeczy w kontekście. Wynikiem większości modeli jest seria liczb i metryk próbujących określić ilościowo, jak dobrze model się sprawdził. To do człowieka należy ujęcie tych wskaźników w odpowiedniej perspektywie i przekazanie wyników odbiorcom

• Większość modeli uczenia maszynowego jest wrażliwa na zaszumione dane. Oznacza to, że modele są zdezorientowane, gdy uwzględnisz dane, które nie mają sensu. Na przykład, jeśli próbujesz znaleźć relacje między danymi ekonomicznymi z całego świata, a jedną z Twoich kolumn jest wskaźnik adopcji szczeniąt w stolicy, informacje te prawdopodobnie nie będą istotne i wprowadzą w błąd model.

Te założenia będą pojawiać się raz po raz, gdy mamy do czynienia z uczeniem maszynowym. Wszystkie są zbyt ważne i często ignorowane przez początkujących analityków danych.

Jak działa uczenie maszynowe?

Każdy rodzaj uczenia maszynowego i każdy indywidualny model działa na bardzo różne sposoby, wykorzystując różne części matematyki i nauki o danych. Jednak ogólnie uczenie maszynowe działa poprzez pobieranie danych, znajdowanie relacji w danych i podawanie jako wynik tego, czego nauczył się model, jak pokazano na poniższym diagramie:



Badając różne typy modeli uczenia maszynowego, zobaczymy, jak w różny sposób manipulują danymi i uzyskują różne wyniki dla różnych aplikacji.

Rodzaje uczenia maszynowego

Istnieje wiele sposobów na segmentację uczenia maszynowego i głębsze zanurzenie się. W części 1 wspomniałem o modelach statystycznych i probabilistycznych. Modele te wykorzystują statystyki i prawdopodobieństwo, które widzieliśmy w poprzednich częściach, aby znaleźć relacje między danymi i dokonać prognoz. W tym rozdziale zaimplementujemy oba typy modeli. W następnym rozdziale zobaczymy uczenie maszynowe poza sztywnym matematycznym światem

statystyki/prawdopodobieństwa. Modele uczenia maszynowego można podzielić według różnych cech, w tym:

• Rodzaje danych/struktur organicznych, z których korzystają (drzewo/wykres/sieć neuronowa)

• Dziedzina matematyki, z którą są najbardziej związani (statystyczna/probabilistyczna)

• Poziom obliczeń wymagany do szkolenia (uczenie głębokie)

Na potrzeby edukacji zaproponuję własne zestawienie modeli uczenia maszynowego. Odchodząc od najwyższego poziomu uczenia maszynowego, istnieją następujące trzy podzbiory:

• Nadzorowana nauka

• Nauka nienadzorowana

• Uczenie się przez wzmacnianie

Nadzorowana nauka

Mówiąc najprościej, nadzorowane uczenie się znajduje powiązania między cechami zbioru danych a zmienną docelową. Na przykład nadzorowane modele uczenia się mogą próbować znaleźć związek między cechami zdrowotnymi danej osoby (tętno, poziom otyłości itd.) a ryzykiem wystąpienia zawału serca (zmienna docelowa). Te powiązania umożliwiają nadzorowanym modelom dokonywanie prognoz na podstawie przykładów z przeszłości. Często jest to pierwsza rzecz, która przychodzi ludziom do głowy, gdy słyszą frazę „uczenie maszynowe", ale w żaden sposób nie obejmuje ona sfery uczenia maszynowego. Nadzorowane modele uczenia maszynowego są często nazywane modelami analizy predykcyjnej, nazwanymi ze względu na ich zdolność do przewidywania przyszłości na podstawie przeszłości. Nadzorowane uczenie maszynowe wymaga określonego typu danych zwanych danymi oznaczonymi etykietami. Oznacza to, że musimy uczyć naszego modelu, podając mu historyczne przykłady, które są oznaczone poprawną odpowiedzią. Przypomnij sobie przykład rozpoznawania twarzy. Jest to nadzorowany model uczenia się, ponieważ trenujemy naszą modelkę za pomocą poprzednich zdjęć oznaczonych jako twarz lub nie, a następnie prosimy modelkę o przewidzenie, czy na nowym zdjęciu jest twarz. W szczególności nadzorowane uczenie się polega na wykorzystaniu części danych do przewidywania innej części. Najpierw musimy podzielić dane na dwie części, w następujący sposób:

• Predyktory, czyli kolumny, które zostaną użyte do wykonania naszego przewidywania. Są one czasami nazywane cechami, danymi wejściowymi, zmiennymi i zmiennymi niezależnymi.

• Odpowiedź, czyli kolumnę, którą chcemy przewidzieć. Jest to czasami nazywane zmienną wynikową, etykietą, docelową i zależną.

Uczenie nadzorowane próbuje znaleźć związek między predyktorami a odpowiedzią w celu dokonania prognozy. Chodzi o to, że w przyszłości pojawi się obserwacja danych i poznamy tylko predyktory. Następnie model będzie musiał użyć predyktorów, aby dokładnie przewidzieć wartość odpowiedzi.

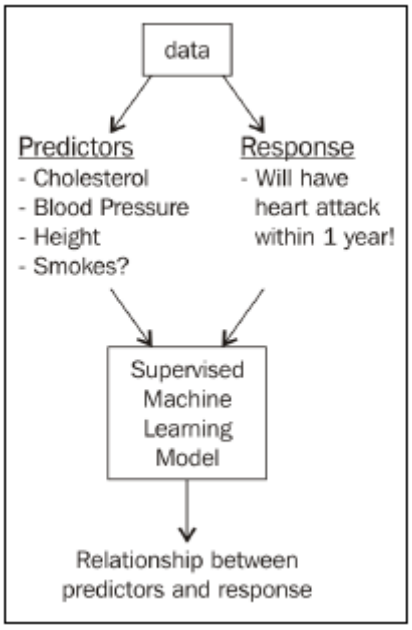
Przykład - przewidywanie zawału serca

Załóżmy, że chcemy przewidzieć, czy ktoś dostanie zawału serca w ciągu roku. Aby to przewidzieć, podajemy poziom cholesterolu, ciśnienie krwi, wzrost, nawyki palenia i być może więcej. Na podstawie tych danych musimy ustalić prawdopodobieństwo zawału serca. Załóżmy, że aby dokonać tej prognozy, przyjrzymy się poprzednim pacjentom i ich historii medycznej. Ponieważ są to poprzedni pacjenci, znamy nie tylko ich czynniki prognostyczne (cholesterol, ciśnienie krwi itd.), ale wiemy też,

czy rzeczywiście mieli zawał serca (bo to już się wydarzyło!). Jest to nadzorowany problem uczenia maszynowego, ponieważ:

• Przepowiada komuś

• Wykorzystuje historyczne dane treningowe do znajdowania związków między zmiennymi medycznymi a zawałami serca



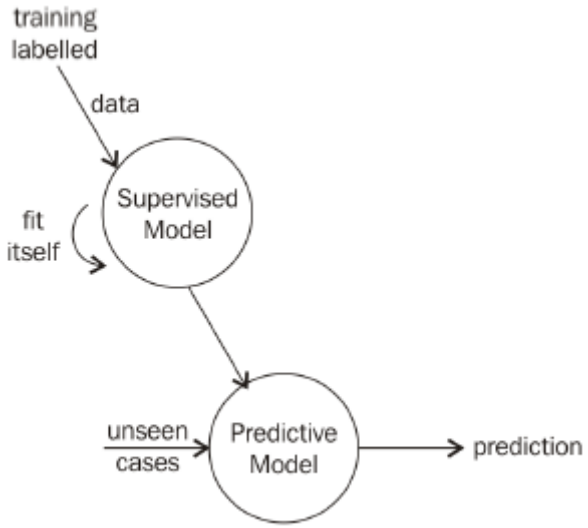
Mamy nadzieję, że jutro przyjdzie pacjent, a nasz model będzie w stanie określić, czy pacjent jest zagrożony zawałem serca na podstawie jej stanu (tak jak zrobiłby to lekarz!). W miarę jak model widzi coraz więcej oznaczonych danych, dostosowuje się, aby dopasować właściwe podane nam etykiety. Możemy użyć różnych wskaźników (wyjaśnionych w dalszej części tego rozdziału), aby dokładnie określić, jak dobrze radzi sobie nasz nadzorowany model uczenia maszynowego i jak może się lepiej dostosować. Jedną z największych wad nadzorowanego uczenia maszynowego jest to, że potrzebujemy tych oznaczonych danych, które mogą być bardzo trudne do zdobycia. Załóżmy, że chcemy przewidzieć zawał serca, możemy potrzebować tysięcy pacjentów wraz ze wszystkimi wypełnionymi informacjami medycznymi i wieloletnią historią obserwacji dla każdej osoby, co może być koszmarem do zdobycia. Krótko mówiąc, modele nadzorowane wykorzystują dane historyczne oznaczone etykietami w celu przewidywania przyszłości. Niektóre możliwe zastosowania uczenia nadzorowanego obejmują:

• Prognozy cen akcji

• Prognozy pogody

• Prognozy kryminalne

Zwróć uwagę, w jaki sposób każdy z powyższych przykładów używa słowa „przewidywanie”, co ma sens, biorąc pod uwagę, jak kładłem nacisk na zdolność nadzorowanego uczenia się do przewidywania przyszłości. Jednak w przewidywaniach historia się nie kończy. Oto wizualizacja, w jaki sposób nadzorowane modele wykorzystują dane oznaczone etykietami, aby dopasować się do siebie i przygotować się do prognozowania:



Zwróć uwagę, jak nadzorowany model uczy się na podstawie wielu danych szkoleniowych, a następnie, gdy jest gotowy, analizuje niewidoczne przypadki i generuje prognozę.

Nie chodzi tylko o przewidywania

Uczenie nadzorowane wykorzystuje związek między predyktorami i odpowiedzią, aby przewidywać, ale czasami wystarczy wiedzieć, że w ogóle istnieje związek. Załóżmy, że używamy nadzorowanego modelu uczenia maszynowego, aby przewidzieć, czy klient kupi dany przedmiot. Ewentualny zbiór danych może wyglądać następująco:

Person ID	Age	Gender	Employed?	Bought the product?
1	63	F	N	Y
2	24	M	Y	N

Zauważ, że w tym przypadku naszymi predyktorami są Wiek, Płeć i Zatrudnienie, podczas gdy naszą odpowiedzią jest Kupiłem produkt? Dzieje się tak, ponieważ chcemy sprawdzić, czy biorąc pod uwagę wiek, płeć i status zatrudnienia, ktoś kupi produkt. Załóżmy, że model jest wytrenowany na tych danych i może dokładnie przewidywać, czy ktoś coś kupi. To samo w sobie jest ekscytujące, ale jest jeszcze coś, co jest prawdopodobnie jeszcze bardziej ekscytujące. Fakt, że możemy dokonywać dokładnych prognoz, sugeruje, że istnieje związek między tymi zmiennymi, co oznacza, że aby wiedzieć, czy ktoś kupi Twój produkt, wystarczy znać jego wiek, płeć i status zatrudnienia! Może to być sprzeczne z wcześniejszymi badaniami rynkowymi wskazującymi, że o potencjalnym kliencie trzeba wiedzieć znacznie więcej, aby móc dokonać takiej prognozy. Odnosi się to do zdolności nadzorowanego uczenia maszynowego do zrozumienia, które predyktory wpływają na reakcję i w jaki sposób. Na przykład, czy kobiety są bardziej skłonne do zakupu produktu, które grupy wiekowe są skłonne do odrzucenia produktu, czy istnieje kombinacja wieku i płci, która jest lepszym predyktorem niż jakakolwiek pojedyncza kolumna? Czy w miarę wzrostu czyjegoś wieku jego szanse na zakup produktu rosną, maleją, czy pozostają takie same? Możliwe też, że nie wszystkie kolumny są potrzebne. Ewentualne wyniki uczenia maszynowego mogą sugerować, że tylko niektóre kolumny są potrzebne do wykonania prognozy, a inne kolumny są tylko szumem (nie korelują z odpowiedzią, a zatem mylą model).

Rodzaje nadzorowanego uczenia się

Ogólnie rzecz biorąc, istnieją dwa rodzaje nadzorowanych modeli uczenia się: regresja i klasyfikacja. Różnica między nimi jest dość prosta i polega na zmiennej odpowiedzi.

Regresja

Modele regresji próbują przewidzieć ciągłą odpowiedź. Oznacza to, że odpowiedź może przyjąć szereg nieskończonych wartości. Rozważ następujące przykłady:

• Kwoty w dolarach

- Wynagrodzenie

- Budżet

• Temperatura

• Czas

- Zwykle rejestrowany w sekundach lub minutach

Klasyfikacja

Klasyfikacja próbuje przewidzieć reakcję kategoryczną, co oznacza, że odpowiedź ma tylko skończoną liczbę wyborów. Przykłady obejmują te podane w następujący sposób:

• Stopień raka (1, 2, 3, 4, 5)

• Pytania prawda/fałsz, takie jak następujące przykłady:

- „Czy ta osoba dostanie zawału serca w ciągu roku?”

- "Dostaniesz tę pracę?"

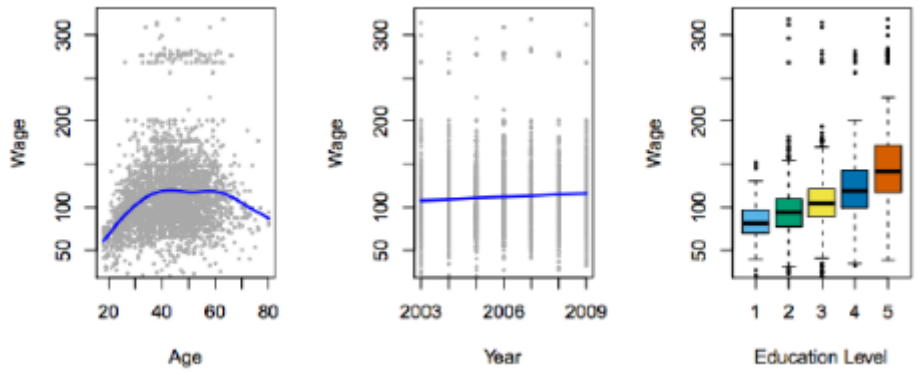
• Biorąc pod uwagę zdjęcie twarzy, do kogo ta twarz należy? (rozpoznawanie twarzy)

• Przewiduj rok, w którym ktoś się urodził:

°° Zauważ, że jest wiele możliwych odpowiedzi (ponad 100), ale wciąż znacznie więcej

Przykład – regresja

Poniższe wykresy pokazują zależność między trzema zmiennymi kategorialnymi (wiek, rok urodzenia i poziom wykształcenia) a wynagrodzeniem osoby:



Zauważ, że chociaż każdy predyktor jest jakościowy, ten przykład jest regresywny, ponieważ oś y, nasza zmienna zależna, nasza odpowiedź, jest ciągła. Naszym wcześniejszym przykładem zawału serca jest klasyfikacja, ponieważ odpowiedzią było, czy ta osoba będzie miała zawał serca w ciągu roku?, która ma tylko dwie możliwe odpowiedzi: Tak lub Nie.

Dane są w oczach patrzącego

Czasami podjęcie decyzji, czy należy użyć klasyfikacji czy regresji, może być trudne. Weź pod uwagę, że interesuje nas pogoda na zewnątrz. Moglibyśmy zadać pytanie, jak gorąco jest na zewnątrz?, w takim przypadku twoja odpowiedź jest w skali ciągłej, a niektóre możliwe odpowiedzi to 60,7 stopnia lub 98 stopni. Jednak w ramach ćwiczenia idź i zapytaj 10 osób, jaka jest temperatura na zewnątrz. Gwarantuję Ci, że ktoś (jeśli nie większość ludzi) nie odpowie w jakimś dokładnym stopniu, ale zbierze swoją odpowiedź i powie coś w stylu lat 60-tych. Możemy chcieć potraktować ten problem jako problem klasyfikacyjny, w którym zmienna odpowiedzi nie jest już w dokładnych stopniach, ale znajduje się w wiadrze. Teoretycznie liczba wiader byłaby skończona, dzięki czemu model być może nieco lepiej uczyłby się różnic między latami 60. i 70.

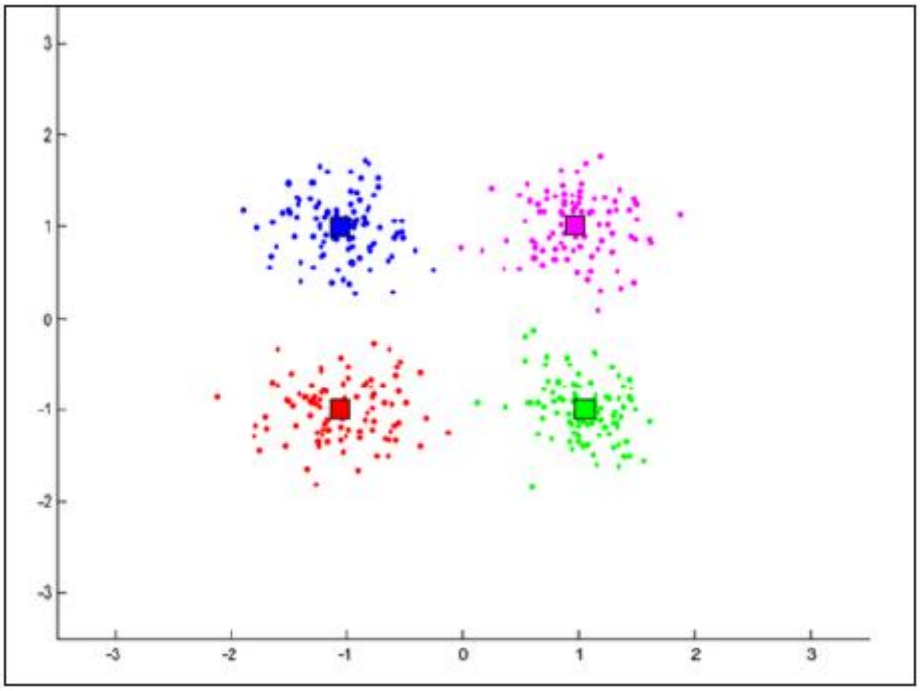
Nauka nienadzorowana

Drugi rodzaj uczenia maszynowego nie zajmuje się przewidywaniami, ale ma znacznie bardziej otwarty cel. Uczenie nienadzorowane wykorzystuje zestaw predyktorów i wykorzystuje relacje między predyktorami w celu wykonania zadań, takich jak:

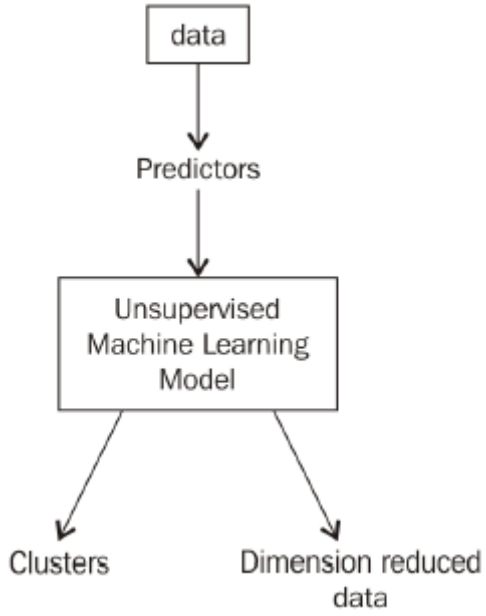
• Zmniejszenie rozmiaru danych poprzez kondensację zmiennych. Przykładem może być kompresja plików. Kompresja działa, wykorzystując wzorce w danych i przedstawiając dane w mniejszym formacie.

• Znajdowanie grup obserwacji, które zachowują się podobnie i grupowanie ich razem.

Pierwszy element na tej liście nazywa się redukcją wymiarów, a drugi nazywa się grupowaniem. Oba są przykładami uczenia się nienadzorowanego, ponieważ nie próbują znaleźć związku między predyktorami a konkretną odpowiedzią, a zatem nie są wykorzystywane do dokonywania jakichkolwiek przewidywań. Modele bez nadzoru, zamiast tego są wykorzystywane do wyszukiwania organizacji i reprezentacji danych, które były wcześniej nieznane.



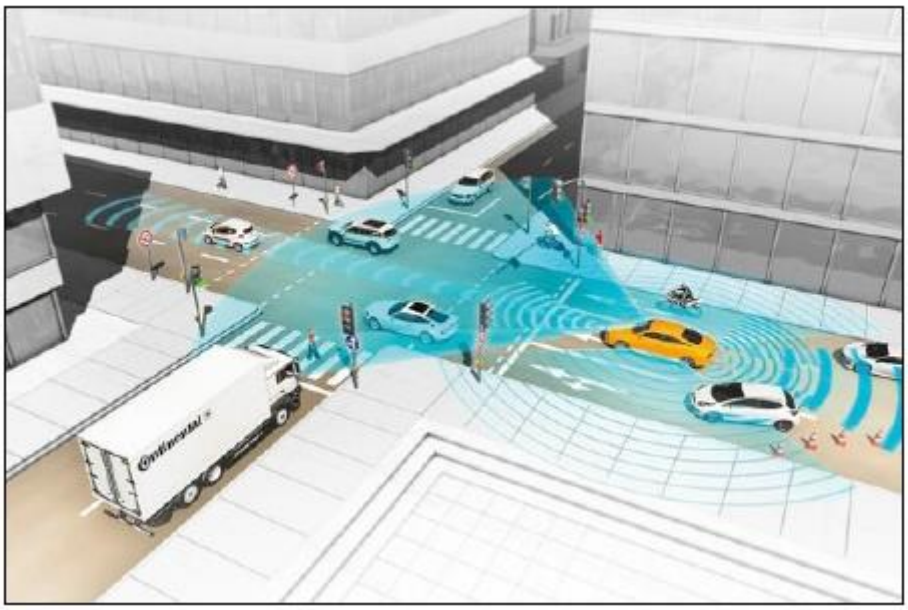
Powyższy zrzut ekranu jest reprezentacją analizy skupień. Model rozpozna, że każda unikatowo pokolorowana grupa obserwacji jest podobna do innej, ale różni się od pozostałych skupień. Dużą zaletą uczenia się nienadzorowanego jest to, że nie wymaga danych oznaczonych etykietami, co oznacza, że znacznie łatwiej jest uzyskać dane zgodne z modelami uczenia się nienadzorowanego. Oczywiście wadą jest to, że tracimy całą moc predykcyjną, ponieważ zmienna odpowiedzi zawiera informacje potrzebne do prognozowania, a bez niej nasz model będzie beznadziejny w dokonywaniu jakichkolwiek prognoz. Dużą wadą jest to, że trudno zobaczyć, jak dobrze sobie radzimy. W przypadku problemu regresji lub klasyfikacji możemy łatwo stwierdzić, jak dobrze nasze modele przewidują, porównując odpowiedzi naszych modeli z odpowiedziami rzeczywistymi. Na przykład, jeśli nasz nadzorowany model przewiduje deszcz, a na zewnątrz jest słonecznie, model był nieprawidłowy. Jeśli nasz nadzorowany model przewiduje, że cena wzrośnie o 1 dolara i wzrośnie o 99 centów, nasz model był bardzo bliski! W modelowaniu nadzorowanym ta koncepcja jest obca, ponieważ nie mamy odpowiedzi, z którą można by porównać nasze modele. Modele nienadzorowane jedynie sugerują różnice i podobieństwa, co następnie wymaga interpretacji przez człowieka.



Krótko mówiąc, głównym celem modeli nienadzorowanych jest znajdowanie podobieństw i różnic między obserwacjami danych.

Nauka przez wzmacnianie

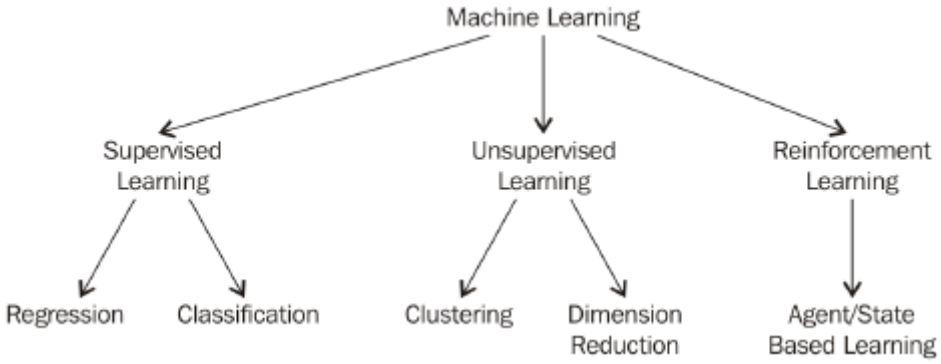
W uczeniu się przez wzmacnianie algorytmy wybierają działanie w środowisku, a następnie są nagradzane (pozytywnie lub negatywnie) za wybranie tego działania. Algorytm następnie dostosowuje się i modyfikuje swoją strategię, aby osiągnąć jakiś cel, którym zwykle jest uzyskanie większej liczby nagród. Ten rodzaj uczenia maszynowego jest bardzo popularny w grach wspomaganych przez sztuczną inteligencję, ponieważ agenci (AI) mogą eksplorować wirtualny świat i zbierać nagrody oraz uczyć się najlepszych technik nawigacji. Model ten jest również popularny w robotyce, zwłaszcza w zakresie maszyn samoautomatycznych, w tym samochodów:



Można sądzić, że wzmacnianie jest podobne do nadzorowanego uczenia się, ponieważ agent uczy się na podstawie swoich przeszłych działań, aby w przyszłości wykonywać lepsze ruchy; jednak główna różnica polega na nagrodzie. Nagroda nie musi być w żaden sposób powiązana z „poprawną" lub „nieprawidłową" decyzją. Nagroda po prostu zachęca (lub zniechęca) do różnych działań.

Przegląd rodzajów uczenia maszynowego

Spośród trzech typów uczenia maszynowego - nadzorowanego, nienadzorowanego i ze wzmocnieniem - możemy wyobrazić sobie świat uczenia maszynowego jako coś takiego:



Każdy z trzech typów uczenia maszynowego ma swoje zalety, a także wady, które wymieniono:

• Nadzorowane uczenie maszynowe: wykorzystuje relacje między predyktorami a zmiennymi odpowiedzi w celu przewidywania przyszłych obserwacji danych.

Plusy:

- Może dokonywać prognoz na przyszłość

- Potrafi określić ilościowo relacje między predyktorami a zmiennymi odpowiedzi

- Może nam pokazać, jak zmienne wpływają na siebie i w jakim stopniu

Minusy:

- Wymaga oznaczonych danych (które mogą być trudne do uzyskania)

• Nienadzorowane uczenie maszynowe: wykrywa podobieństwa i różnice między punktami danych.

Plusy:

- Może znaleźć grupy punktów danych, które zachowują się podobnie, czego człowiek nigdy by nie zauważył

- Może to być etap wstępnego przetwarzania dla nadzorowanego uczenia się. Pomyśl o grupowaniu kilku punktów danych, a następnie wykorzystaniu tych klastrów jako odpowiedzi!

- Może używać danych nieoznaczonych, co jest znacznie łatwiejsze do znalezienia

Minusy:

- Ma zerową moc predykcyjną

- Trudno określić, czy jesteśmy na dobrej drodze

- O wiele bardziej opiera się na ludzkiej interpretacji

• Uczenie się przez wzmacnianie: jest to uczenie się oparte na nagrodach, które zachęca agentów do podejmowania określonych działań w ich środowisku.

Plusy:

- Bardzo skomplikowane systemy nagród tworzą bardzo skomplikowane systemy sztucznej inteligencji

- Może się uczyć w prawie każdym środowisku, w tym na naszej własnej Ziemi

Minusy:

- Agent jest początkowo nieobliczalny i wcześniej dokonuje wielu okropnych wyborów zdając sobie sprawę, że te wybory mają negatywne nagrody. Na przykład samochód może uderzyć w ścianę i nie wiedzieć, że to nie jest w porządku, dopóki otoczenie go nie wynagrodzi

- Może minąć trochę czasu, zanim agent całkowicie uniknie decyzji

- Agent może zachować ostrożność i wybrać tylko jedną akcję i „zbyt przestraszony", aby spróbować czegokolwiek innego w obawie przed karą

Jak do tego wszystkiego pasuje modelowanie statystyczne?

Do tej pory używałem terminu uczenie maszynowe, ale możesz zapytać, jak modelowanie statystyczne odgrywa w tym wszystkim rolę. Jest to wciąż dyskutowany temat w dziedzinie nauki o danych. Uważam, że modelowanie statystyczne to kolejne określenie modeli uczenia maszynowego, które w dużej mierze opierają się na wykorzystaniu reguł matematycznych zapożyczonych z prawdopodobieństwa i statystyki do tworzenia relacji między zmiennymi danych (często w sensie predykcyjnym). Pozostała część tego rozdziału skupi się głównie na jednym modelu statystycznym/probabilistycznym – regresji liniowej.

Regresja liniowa

Wreszcie! Zbadamy nasz pierwszy prawdziwy model uczenia maszynowego. Regresje liniowe są formą regresji, co oznacza, że jest to model uczenia maszynowego, który próbuje znaleźć związek między predyktorami a zmienną odpowiedzi, a ta zmienna odpowiedzi jest, jak się domyślacie, ciągła! Pojęcie to jest równoznaczne z tworzeniem linii najlepszego dopasowania. W przypadku regresji liniowej spróbujemy znaleźć liniową zależność między naszymi predyktorami a zmienną odpowiedzi. Formalnie chcemy rozwiązać formułę o następującym formacie:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

• y jest naszą zmienną odpowiedzi

• x_i jest naszą i-tą zmienną (i-tą kolumną lub i-tym predyktorem)

• B_0 to punkt przecięcia

• B_i jest współczynnikiem dla członu x_i

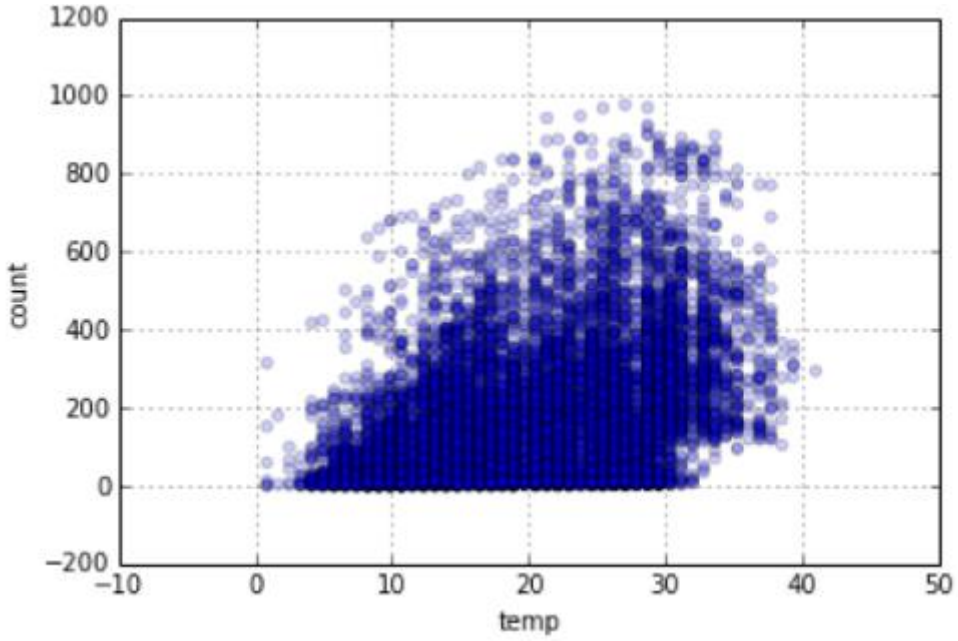
Rzućmy okiem na niektóre dane, zanim przejdziemy do szczegółów. Ten zbiór danych jest publicznie dostępny i próbuje przewidzieć liczbę rowerów potrzebnych w danym dniu do programu wypożyczania rowerów:

read the data and set the datetime as the index

```
# taken from Kaggle: https://www.kaggle.com/c/bike-sharing-demand/data

import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

url = 'https://raw.githubusercontent.com/justmarkham/DAT8/master/data/bikeshare.csv'

bikes = pd.read_csv(url)

bikes.head()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0	0	1	1

Widzimy, że każdy wiersz reprezentuje godzinę użytkowania roweru. W tym przypadku interesuje nas przewidzenie liczby, która reprezentuje całkowitą liczbę wypożyczonych rowerów w okresie tej godziny.

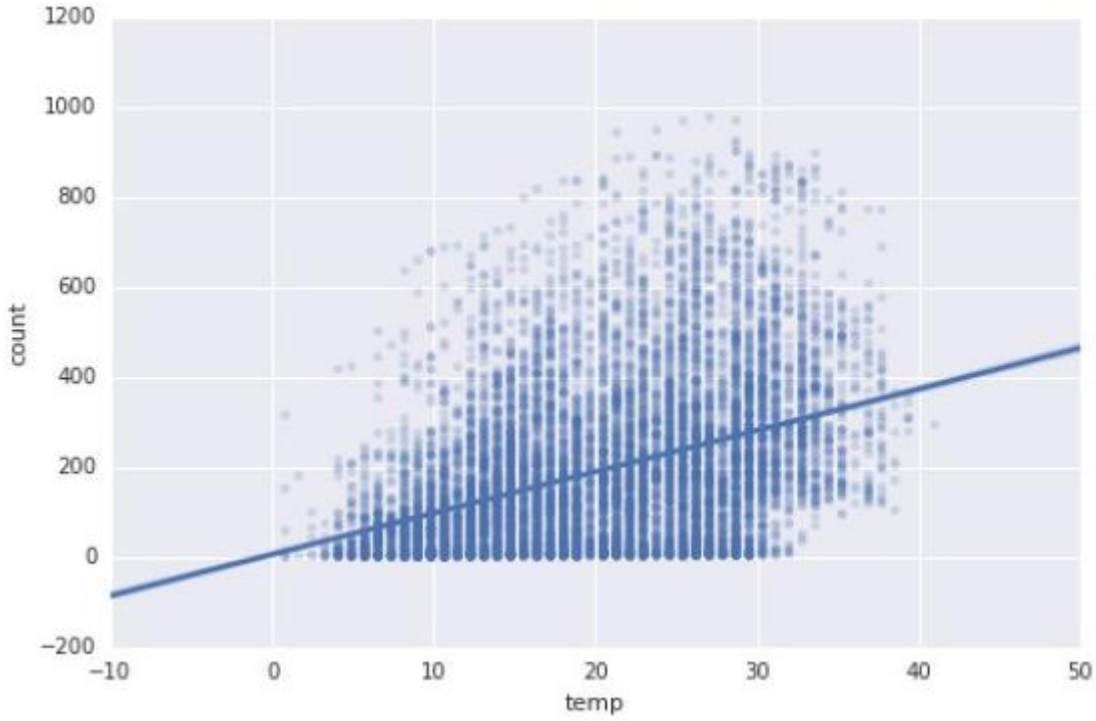


Spójrzmy na przykład na wykres punktowy między temperaturą (kolumna temp) a liczbą, jak pokazano:

```
bikes.plot(kind='scatter', x='temp', y='count', alpha=0.2
```

A teraz użyjmy modułu zwanego seaborn, aby narysować sobie linię najlepszego dopasowania w następujący sposób:

```
import seaborn as sns #using seaborn to get a line of best fit

sns.lmplot(x='temp', y='count', data=bikes, aspect=1.5, scatter_

kws={'alpha':0.2})
```



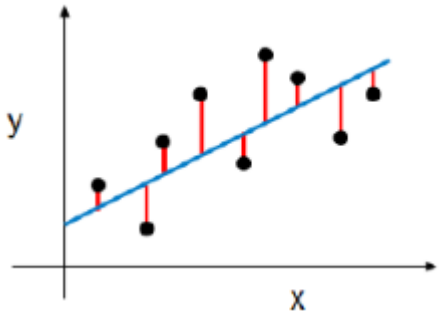
Ta linia na wykresie próbuje zwizualizować i określić ilościowo związek między temp i liczenie. Aby dokonać prognozy, po prostu znajdujemy daną temperaturę, a następnie widzimy, gdzie linia przewidziałaby liczbę. Na przykład, jeśli temperatura wynosi 20 stopni (uwaga Celsjusza), to nasza linia przewidziała, że wypożyczonych zostanie około 200 rowerów. Jeśli temperatura przekracza 40 stopni, potrzeba ponad 400 rowerów! Wygląda na to, że wraz ze wzrostem temperatury rośnie również nasze liczenie. Zobaczmy, czy nasza wartość korelacji, która określa ilościowo zależność liniową między zmiennymi, również pasuje do tego pojęcie:

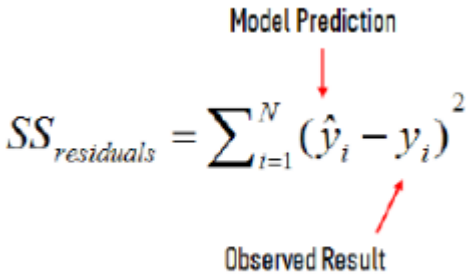
```
rowery[['liczba', 'temp']].corr()

# 0,3944
```

Między tymi dwiema zmiennymi istnieje (słaba) dodatnia korelacja! Wróćmy teraz do postaci regresji liniowej:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

Nasz model spróbuje narysować idealną linię między wszystkimi kropkami na poprzednim wykresie, ale oczywiście wyraźnie widać, że między tymi kropkami nie ma idealnej linii! Model znajdzie wtedy najlepszą możliwą linię. W jaki sposób? Możemy narysować nieskończone linie między punktami danych, ale co sprawia, że linia jest najlepsza? Rozważ następujący diagram:





W naszym modelu otrzymujemy x i y, a model uczy się współczynników Beta, znanych również jako współczynniki modelu:

• Czarne kropki to obserwowane wartości x i y.

• Niebieska linia to nasza linia najlepszego dopasowania.

• Czerwone linie między kropkami a linią nazywają się pozostałościami; są to odległości między obserwowanymi wartościami a linią. Oni są jak błędna linia. Każdy punkt danych ma resztę lub odległość do linii najlepszego dopasowania. Suma kwadratów reszt to suma kwadratów każdej reszty. Linia najlepszego dopasowania ma najmniejszą sumę kwadratu wartości rezydualnej. Zbudujmy tę linię w Pythonie, dobrze?

create X and y

feature_cols = ['temp'] # a lsit of the predictors

X = bikes[feature_cols] # subsetting our data to only the predictors

y = bikes['count'] # our response variable

Zwróć uwagę, jak stworzyliśmy zmienną X i Y. Reprezentują one nasze predyktory i zmienną odpowiedzi. Następnie zaimportujemy nasz moduł uczenia maszynowego, scikit learn, jak pokazano:

import scikit-learn, our machine learning module

from sklearn.linear_model import LinearRegression

Na koniec dopasujemy nasz model do predyktorów i zmiennej odpowiedzi w następujący sposób:

linreg = LinearRegression() #instantiate a new model

linreg.fit(X, y) #fit the model to our data

print the coefficients

print linreg.intercept_

print linreg.coef_

6.04621295962 # our Beta_0

[9.17054048] # our beta parameters

Interpretacja:

• B_0 (6.04) to wartość y, gdy X = 0.

Jest to szacunkowa liczba rowerów, które zostaną wypożyczone, gdy temperatura spadnie do 0 stopni Celsjusza. Przewiduje się, że w temperaturze 0 stopni w użyciu będzie sześć rowerów (jest zimno!). Czasami może nie mieć sensu w ogóle interpretować przechwycenia, ponieważ może nie istnieć pojęcie zera czegoś. Przypomnij sobie poziomy danych. Nie wszystkie poziomy mają to pojęcie. Temperatura istnieje na poziomie, który nieodłącznie wiąże się z brakiem rowerów; więc jesteśmy bezpieczni. Bądź jednak ostrożny w przyszłości i zawsze zadawaj sobie pytanie, czy ma sens nie mieć nic z tego:

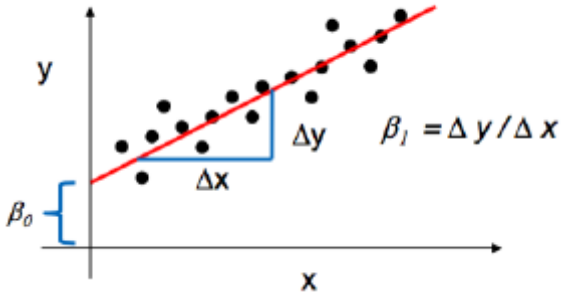
• B_1 (9.17) to nasz współczynnik temp.

- Jest to zmiana w y podzielona przez zmianę w x1.

- Przedstawia, jak x i y poruszają się razem.

- Zmiana o 1 stopień Celsjusza wiąże się ze wzrostem o około 9 wypożyczonych rowerów.

- Znak tego współczynnika jest ważny. Gdyby to było negatywne, to oznaczałoby, że wzrost temperatury wiąże się ze spadkiem wynajmów.



Rozważ poprzednią reprezentację współczynników Beta w regresji liniowej: Ważne jest, aby powtórzyć, że są to wszystkie stwierdzenia o korelacji, a nie o związku przyczynowym. Nie mamy możliwości stwierdzić, czy wzrost czynszu jest spowodowany zmianą temperatury, po prostu wydaje się, że istnieje wspólny ruch. Używanie scikit-learn do przewidywania jest łatwe!

linreg.predict(20)

189.4570

Oznacza to, że przy temperaturze 20 stopni prawdopodobnie wypożyczonych zostanie 190 rowerów.

Dodawanie kolejnych predyktorów

Dodanie większej liczby predyktorów do modelu jest tak proste, jak opisanie modelu regresji liniowej w scikit-naucz się o nich!

Zanim to zrobimy, powinniśmy spojrzeć na dostarczony nam słownik danych, aby lepiej zrozumieć te predyktory:

• sezon: 1 = wiosna, 2 = lato, 3 = jesień, 4 = zima

• święto: czy dzień jest uznawany za święto

• dzień roboczy: czy dzień jest weekendem czy świętem

• pogoda:

1. Pogodnie, Nieliczne zachmurzenie, Częściowe zachmurzenie

2. Mgła + Pochmurno, Mgła + Połamane chmury, Mgła + Kilka chmur, Mgła

3. Lekki śnieg, Lekki deszcz + Burza + Rozproszone chmury,Lekki deszcz + rozproszone chmury

4. Ulewny deszcz + lodowe palety + burza + mgła, śnieg + mgła

• temp: Temperatura w stopniach Celsjusza

• atemp: Czuje się jak temperatura w stopniach Celsjusza

• wilgotność: wilgotność względna

• prędkość wiatru: prędkość wiatru

• dorywczo: Liczba rozpoczętych wypożyczeń dla niezarejestrowanych użytkowników

• zarejestrowany: Liczba rozpoczętych wypożyczeń zarejestrowanych użytkowników

• count: Liczba wszystkich wypożyczeń

Teraz stwórzmy nasz model regresji liniowej. Tak jak poprzednio, najpierw utworzymy listę zawierającą funkcje, którym chcemy się przyjrzeć, stworzymy nasze funkcje i nasze zestawy danych odpowiedzi (X i y), a następnie dopasujemy naszą regresję liniową. Po dopasowaniu naszego modelu regresji przyjrzymy się współczynnikom modelu, aby zobaczyć, jak nasze cechy oddziałują na naszą odpowiedź:

```
# create a list of features

feature_cols = ['temp', 'season', 'weather', 'humidity']

# create X and y

X = bikes[feature_cols]

y = bikes['count']

# instantiate and fit

linreg = LinearRegression()

linreg.fit(X, y)

# pair the feature names with the coefficients

zip(feature_cols, linreg.coef_)
```

This gives us:

```
[('temp', 7.8648249924774403),

('season', 22.538757532466754),

('weather', 6.6703020359238048),

('humidity', -3.1188733823964974)]
```

Oznaczający:

• Przy utrzymaniu wszystkich innych predyktorów na stałym poziomie, wzrost temperatury o 1 jednostkę wiąże się ze wzrostem wypożyczenia o 7,86 rowerów

• Przy utrzymaniu wszystkich innych predyktorów na stałym poziomie, wzrost o 1 jednostkę w sezonie wiąże się ze wzrostem wynajmu o 22,5 rowerów.

• Przy utrzymaniu wszystkich innych predyktorów na stałym poziomie, wzrost pogody o 1 jednostkę wiąże się ze wzrostem wynajmu o 6,67 rowerów

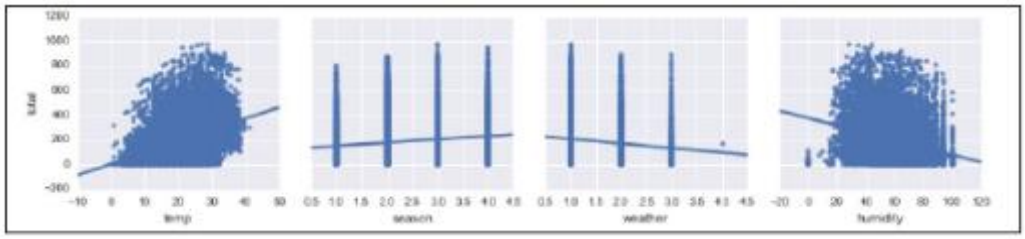
• Przy utrzymaniu wszystkich innych predyktorów na stałym poziomie, wzrost wilgotności o 1 jednostkę wiąże się ze spadkiem wynajmu o 3,12 rowerów

To jest interesujące. Zwróć uwagę, że wraz ze wzrostem pogody (co oznacza, że zbliża się zachmurzenie), popyt na rowery rośnie, podobnie jak w przypadku wzrostu zmiennych sezonowych (co oznacza, że zbliża się zima). Wcale nie tego się spodziewałem! Przyjrzyjmy się poszczególnym wykresom punktowym między każdym rediektorem a odpowiedzią, jak pokazano na ilustracji:

feature_cols = ['temp', 'season', 'weather', 'humidity']

multiple scatter plots

sns.pairplot(bikes, x_vars=feature_cols, y_vars='count', kind='reg')



Zwróć uwagę, jak linia pogody ma tendencję spadkową, co jest przeciwieństwem tego, co sugerował ostatni model liniowy. Teraz musimy się martwić, które z tych predyktorów faktycznie pomagają nam w przewidywaniu, a które są tylko szumem. Aby to zrobić, będziemy potrzebować bardziej zaawansowanych danych.

Metryki regresji

Podczas korzystania z modeli uczenia maszynowego regresji istnieją trzy główne metryki.

Są to:

• Średni błąd bezwzględny

• Średniokwadratowy błąd

• Pierwiastek błędu średniokwadratowego

Każda metryka próbuje opisać i określić ilościowo skuteczność modelu regresji, porównując listę przewidywań z listą poprawnych odpowiedzi. Każda z wymienionych metryk różni się nieco od pozostałych i opowiada inną historię.

Średni błąd bezwzględny (MAE) to średnia bezwzględnej wartości błędów

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Błąd średniokwadratowy (MSE) to średnia z kwadratów błędów

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Pierwiastek średniego błędu bezwzględnego (RMSE) to pierwiastek kwadratowy ze średniej kwadratów błędów

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Gdzie:

• n to liczba obserwacji

• y_i to rzeczywista wartość

• $\hat{y}_i$ jest przewidywaną wartością

Przyjrzyjmy się w Pythonie:

```
# example true and predicted response values

true = [9, 6, 7, 6]

pred = [8, 7, 7, 12]

# note that each value in the last represents a single prediction for a model

# So we are comparing four predictions to four actual answers

# calculate these metrics by hand!

from sklearn import metrics

import numpy as np

print 'MAE:', metrics.mean_absolute_error(true, pred)

print 'MSE:', metrics.mean_squared_error(true, pred)

print 'RMSE:', np.sqrt(metrics.mean_squared_error(true, pred))
```

MAE: 2.0

MSE: 9.5

RMSE: 3.08220700148

Podział tych liczb jest następujący:

• MAE jest prawdopodobnie najłatwiejszy do zrozumienia, ponieważ jest to tylko przeciętny błąd. Średnio wskazuje, jak błędny jest model.

• MSE jest bardziej efektywne niż MAE, ponieważ MSE karze większe błędy, co jest znacznie bardziej przydatne w rzeczywistym świecie.

• RMSE jest jeszcze bardziej popularne niż MSE, ponieważ jest znacznie bardziej interpretowalne.

RMSE jest zwykle preferowaną metryką regresji, ale bez względu na to, którą wybierzesz, wszystkie są funkcjami straty i dlatego należy je zminimalizować. Użyjmy RMSE, aby ustalić, które kolumny pomagają, a które bolą. Zacznijmy tylko od użycia temperatury. Zauważ, że nasza procedura będzie następująca:

1. Utwórz nasze zmienne x i y.

2. Dopasuj model regresji liniowej.

3. Za pomocą modelu sporządź listę prognoz opartych na X.

4. Oblicz RMSE między przewidywaniami a rzeczywistymi wartościami.

Rzućmy okiem na kod:

```
from sklearn import metrics

# import metrics from scikit learn

feature_cols = ['temp']

# create X and y

X = bikes[feature_cols]

linreg = LinearRegression()

linreg.fit(X, y)

y_pred = linreg.predict(X)

np.sqrt(metrics.mean_squared_error(y, y_pred)) # RMSE

# Can be interpreted loosely as an average error

#166.45
```

Teraz spróbujmy użyć temperatury i wilgotności, jak pokazano:

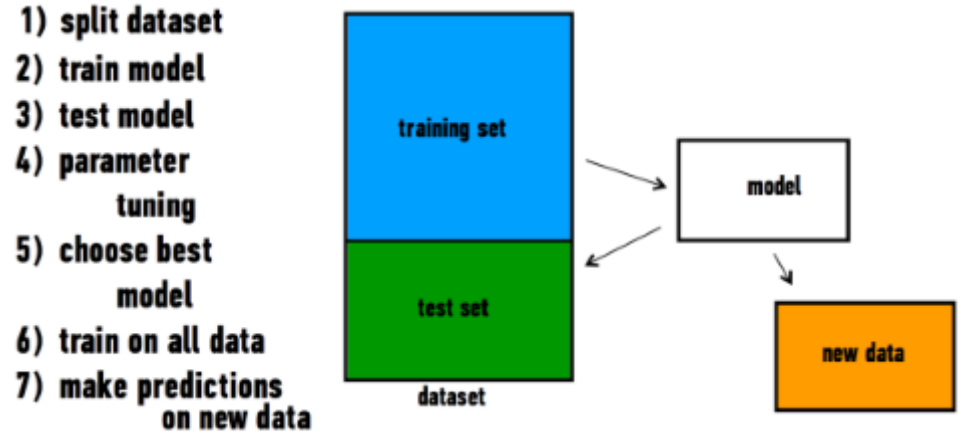
```
feature_cols = ['temp', 'humidity']

# create X and y

X = bikes[feature_cols]

linreg = LinearRegression()

linreg.fit(X, y)

y_pred = linreg.predict(X)

np.sqrt(metrics.mean_squared_error(y, y_pred)) # RMSE

# 157.79
```

Było lepiej!! Spróbujmy użyć jeszcze większej liczby predyktorów, jak pokazano na ilustracji:

```
feature_cols = ['temp', 'humidity', 'season', 'holiday', 'workingday', 'windspeed', 'atemp']

# create X and y
```

```
X = bikes[feature_cols]

linreg = LinearRegression()

linreg.fit(X, y)

y_pred = linreg.predict(X)

np.sqrt(metrics.mean_squared_error(y, y_pred)) # RMSE

# 155.75
```

Nawet lepiej! Na pierwszy rzut oka wydaje się to wielkim triumfem, ale w rzeczywistości kryje się tu ukryte niebezpieczeństwo. Zauważ, że trenujemy linię, aby dopasować się do X i y, a następnie prosimy ją, aby ponownie przewidziała X! W rzeczywistości jest to ogromny błąd w uczeniu maszynowym, ponieważ może prowadzić do nadmiernego dopasowania, co oznacza, że nasz model jedynie zapamiętuje dane i zwraca je z powrotem do nas. Wyobraź sobie, że jesteś uczniem i wchodzisz do pierwszego dnia zajęć, a nauczyciel mówi, że egzamin końcowy na tych zajęciach jest bardzo trudny. Aby cię przygotować, daje ci test praktyczny za testem praktycznym za testem praktycznym. Nadchodzi dzień egzaminu końcowego i jesteś w szoku, gdy dowiadujesz się, że każde pytanie na egzaminie jest dokładnie takie samo jak w teście praktycznym! Na szczęście robiłeś je tyle razy, że pamiętasz odpowiedź i dostajesz 100% na egzaminie. To samo dotyczy mniej więcej tego samego. Dopasowując i przewidując na tych samych danych, model zapamiętuje dane i staje się w tym lepszy. Świetnym sposobem na walkę z tym problemem jest zastosowanie podejścia trenowania/testowania w celu dopasowania modeli uczenia maszynowego, które działa tak, jak pokazano:



Zasadniczo podejmiemy następujące kroki:

1. Podziel zbiór danych na dwie części: zbiór uczący i zbiór testowy.

2. Dopasuj nasz model do zbioru uczącego, a następnie przetestuj go na zbiorze testowym. Tak jak w szkole, gdzie nauczyciel uczył z jednego zestawu notatek, a potem sprawdzał nas z różnych (ale podobnych) pytań.

3. Gdy nasz model jest wystarczająco dobry (na podstawie naszych metryk), zwracamy uwagę naszego modelu na cały zestaw danych.

4. Nasz model czeka na nowe dane, których nikt wcześniej nie widział.

Celem jest tutaj zminimalizowanie błędów naszego modelu poza próbą, które są błędami naszego modelu w danych, których nigdy wcześniej nie widział. Jest to ważne, ponieważ główną ideą

(zazwyczaj) nadzorowanego modelu jest przewidywanie niewidzianych przypadków testowych. Jeśli nasz model nie jest w stanie uogólniać danych treningowych i używać ich do przewidywania niezauważonych przypadków, nasz model nie jest zbyt dobry. Powyższy diagram przedstawia prosty sposób zapewnienia, że nasz model może skutecznie pozyskiwać dane szkoleniowe i używać ich do przewidywania punktów danych, których sam model nigdy nie widział. Oczywiście, jako analitycy danych wiemy, że zestaw testowy ma również dołączone do nich odpowiedzi, ale model tego nie wie. Wszystko to może wydawać się skomplikowane, ale na szczęście scikit-learn ma wbudowaną metodę, jak pokazano:

```
from sklearn.cross_validation import train_test_split

# function that splits data into training and testing sets

feature_cols = ['temp']

X = bikes[feature_cols]

y = bikes['count']

# setting our overall data X, and y

# Note that in this example, we are attempting to find an association

between the temperature of the day and the number of bike rentals.

X_train, X_test, y_train, y_test = train_test_split(X, y) # split the

data into training and testing sets

# X_train and y_train will be used to train the model

# X_test and y_test will be used to test the model

# Remember that all four of these variables are just subsets of the overall X and y.

linreg = LinearRegression()

# instantiate the model

linreg.fit(X_train, y_train)

# fit the model to our training set

y_pred = linreg.predict(X_test)

# predict our testing set

np.sqrt(metrics.mean_squared_error(y_test, y_pred)) # RMSE

# Calculate our metric: 166.91
```

Poświęcimy więcej czasu na uzasadnienie tego podziału treningu/testu w rozdziale 12, Poza podstawami, i przyjrzymy się jeszcze bardziej pomocnej metodzie, ale głównym powodem, dla którego musimy wykonać tę dodatkową pracę, jest to, że nie chcemy upaść w pułapkę, w której nasz model po prostu zwraca nasz zestaw danych z powrotem do nas i nie będzie w stanie obsłużyć niewidocznych punktów danych. Innymi słowy, nasz podział testów pociągów zapewnia, że wskaźniki, na które patrzymy, są bardziej uczciwymi szacunkami wydajności naszej próbki. Spróbujmy teraz ponownie z większą liczbą predyktorów, jak następuje:

```
feature_cols = ['temp', 'workingday']

X = bikes[feature_cols]

y = bikes['count']

X_train, X_test, y_train, y_test = train_test_split(X, y)

# Pick a new random training and test set

linreg = LinearRegression()

linreg.fit(X_train, y_train)

y_pred = linreg.predict(X_test)

# fit and predict

np.sqrt(metrics.mean_squared_error(y_test, y_pred))

# 166.95
```

Teraz nasz model pogorszył się z tym dodatkiem! Sugerując, że dzień roboczy może nie być zbyt przewidywalny dla naszej reakcji, liczy się wypożyczenie roweru. Wszystko to jest dobre i dobrze, ale jak nasz model naprawdę radzi sobie z przewidywaniem? Mamy nasz błąd średniokwadratowy wynoszący około 167 rowerów, ale czy to dobrze? Jednym ze sposobów na odkrycie tego jest ocena modelu zerowego. Model zerowy w nadzorowanym uczeniu maszynowym reprezentuje skuteczne odgadywanie w kółko oczekiwanego wyniku i sprawdzanie, jak Ci poszło. Na przykład w regresji, jeśli tylko odgadniemy średnią liczbę wypożyczeń roweru na godzinę, to jak dobrze poradzi sobie ten model? Najpierw weźmy średnią godzinową wypożyczenia roweru, jak pokazano:

```
average_bike_rental = bikes['count'].mean()

average_bike_rental

# 191.57
```

Oznacza to, że w sumie w tym zestawie danych, niezależnie od pogody, godziny, dnia tygodnia, wilgotności i wszystkiego innego, średnia liczba rowerów, które jeżdżą co godzinę, wynosi około 192. Zróbmy fałszywą listę przewidywań, w której co godzinę pojedynczy domysł to 191,57. Zgadnijmy dla każdej godziny w następujący sposób:

```
num_rows = bikes.shape[0]

num_rows

# 10886
```

All 10,886 of them.

```
null_model_predictions = [average_bike_rental]*num_rows

null_model_predictions

[191.57413191254824,

191.57413191254824,

191.57413191254824,
```

191.57413191254824,

…

191.57413191254824,

191.57413191254824,

191.57413191254824,

191.57413191254824]

Tak więc teraz mamy 10 886 wartości, wszystkie z nich to średnia godzinowa liczba wypożyczeń roweru. Zobaczmy teraz, jaki byłby pierwiastek błędu średniokwadratowego, gdyby nasz model tylko odgadł oczekiwaną wartość średniej godzinowej liczby wypożyczeń roweru:

np.sqrt(metrics.mean_squared_error(y, null_model_predictions))

181.13613

Po prostu zgadując, wygląda na to, że nasz błąd średniokwadratowy wyniósłby 181 rowerów. Tak więc, nawet z jedną lub dwiema funkcjami, możemy to pokonać! Pokonanie modelu zerowego to swego rodzaju punkt odniesienia w uczeniu maszynowym. Jeśli się nad tym zastanowisz, po co w ogóle podejmować wysiłek, jeśli uczenie maszynowe nie jest nawet lepsze niż zgadywanie! Spędziliśmy dużo czasu nad regresją liniową, ale chciałbym teraz poświęcić trochę czasu na przyjrzenie się naszemu następnemu modelowi uczenia maszynowego, który jest właściwie kuzynem regresji liniowej. Opierają się na bardzo podobnych pomysłach, ale mają jedną zasadniczą różnicę - podczas gdy regresja liniowa jest modelem regresji i może być używana tylko do przewidywania liczb ciągłych, naszym następnym modelem uczenia maszynowego będzie model klasyfikacyjny, co oznacza, że będzie próbował tworzyć skojarzenia między cechami a kategoryczną odpowiedzią.

Regresja logistyczna

Nasz pierwszy model klasyfikacji nazywa się regresją logistyczną. Już słyszę pytania, które masz w głowie: co sprawia, że jest logistyka, dlaczego nazywa się to regresją, jeśli twierdzisz, że jest to algorytm klasyfikacji? Wszystko we właściwym czasie, przyjacielu. Regresja logistyczna jest uogólnieniem modelu regresji liniowej dostosowanym do rozwiązywania problemów klasyfikacyjnych. W regresji liniowej używamy zestawu zmiennych cech ilościowych do przewidywania zmiennej odpowiedzi ciągłej. W regresji logistycznej używamy zestawu ilościowych zmiennych cech do przewidywania prawdopodobieństwa przynależności do klasy. Te prawdopodobieństwa można następnie zmapować do etykiet klas, przewidując w ten sposób klasę dla każdej obserwacji. Wykonując regresję liniową, używamy następującej funkcji, aby najlepiej dopasować naszą linię:

$y = \beta_1 + \beta_r x$

Tutaj y jest naszą zmienną odpowiedzi (rzecz, którą chcemy przewidzieć), nasza Beta reprezentuje parametry naszego modelu, a x reprezentuje naszą zmienną wejściową (w tym przypadku pojedynczą, ale może przyjąć więcej, jak widzieliśmy). Krótko mówiąc, załóżmy, że jedną z opcji odpowiedzi w naszym zadaniu klasyfikacyjnym jest klasa 1. Wykonując regresję logistyczną, posługujemy się następującą postacią:

$$\pi = \Pr(y = 1 \mid x) = \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}}$$

Probability of y = 1, given x

Tutaj π reprezentuje warunkowe prawdopodobieństwo, że nasza zmienna odpowiedzi należy do klasy 1, biorąc pod uwagę nasze dane x. Teraz możesz się zastanawiać, co to u licha ta potworność funkcji po prawej stronie i skąd wzięła się zmienna e? Cóż, ta potworność nazywana jest funkcją logistyczną i jest naprawdę cudowna. A ta zmienna, e, w ogóle nie jest zmienną. Zróbmy kopię zapasową kleszcza. Zmienna e jest liczbą specjalną, na przykład π. Jest to około 2,718 i nazywa się liczbą Eulera. Jest często używany w środowiskach modelowania z naturalnym wzrostem i zanikiem. Na przykład naukowcy używają e do modelowania wzrostu populacji bakterii i bawołów. Liczba Eulera jest używana do modelowania radioaktywnego rozpadu chemikaliów, a także do obliczania ciągłego procentu złożonego! Dzisiaj użyjemy e w bardzo szczególnym celu, do uczenia maszynowego. Dlaczego nie możemy po prostu wykonać regresji liniowej bezpośrednio do prawdopodobieństwa przynależności punktu danych do określonej klasy, takiej jak ta?

$Pr = (y = 1 | x) = y = \beta_1 + \beta_r x$

Nie możemy tego zrobić z kilku powodów, ale wskażę jeden duży. Regresja liniowa, ponieważ próbuje odnosić się do ciągłej zmiennej odpowiedzi, zakłada, że nasze y jest ciągłe. W naszym przypadku y reprezentuje prawdopodobieństwo wystąpienia zdarzenia. Nawet jeśli nasze prawdopodobieństwo jest w rzeczywistości zakresem ciągłym, jest to po prostu zakres od 0 do 1. Linia ekstrapolowałaby poza 0 do 1 i byłaby w stanie przewidzieć prawdopodobieństwo -4 lub 1542! Nie możemy tego mieć. Nasz wykres musi być ściśle związany między 0 a 1 na osi y, tak jak prawdziwe prawdopodobieństwo. Inny powód jest nieco bardziej filozoficzny. Stosując regresję liniową, przyjmujemy poważne założenie. Naszym wielkim założeniem jest tutaj liniowa zależność między prawdopodobieństwem a naszymi cechami. Ogólnie rzecz biorąc, jeśli myślimy o prawdopodobieństwie zdarzenia, zwykle myślimy o reprezentujących je gładkich krzywych, a nie o pojedynczej nudnej linii. Potrzebujemy więc czegoś bardziej odpowiedniego. W tym celu cofnijmy się przez chwilę i ponownie przyjrzyjmy się prawdopodobieństwu podstawowemu.

Prawdopodobieństwo, kursy i logarytmiczne kursy

Znamy podstawową koncepcję prawdopodobieństwa, ponieważ prawdopodobieństwo wystąpienia zdarzenia można po prostu modelować jako liczbę sposobów, w jakie zdarzenie może wystąpić, podzieloną przez wszystkie możliwe wyniki. Na przykład, jeśli na 3000 osób, które weszły do sklepu, 1000 faktycznie coś kupiło, to możemy powiedzieć, że prawdopodobieństwo, że jedna osoba kupi przedmiot, jest takie, jak pokazano:

$$\Pr(buy) = \frac{1,000}{3,000} = \frac{1}{3} = 33.3\%$$

Mamy jednak również pokrewną koncepcję, zwaną kursami. Szanse wystąpienia wyniku to stosunek liczby sposobów wystąpienia wyniku podzielonej przez każdy inny możliwy wynik zamiast wszystkich możliwych wyników. W tym samym przykładzie szanse na zakup czegoś przez osobę byłyby następujące:

$$Odds(buy) = \frac{1,000}{3,000} = \frac{1}{2} = .5$$

Oznacza to, że na każdego klienta, którego skonwertujesz, nie skonwertujesz dwóch klientów. Te pojęcia są tak powiązane, że istnieje nawet formuła, jaka można przejść od jednego do drugiego. Mamy to:

$$Odds = \frac{P}{1-P}$$

Sprawdźmy to na naszym przykładzie, jak pokazano na ilustracji:

$$Odds = \frac{\frac{1}{3}}{1-\frac{1}{3}} = \frac{\frac{1}{3}}{\frac{2}{3}} = \frac{1}{2}$$

To się sprawdza!

Użyjmy Pythona do stworzenia tabeli prawdopodobieństw i powiązanych z nimi szans, jak pokazano:

```
# create a table of probability versus odds

table = pd.DataFrame({'probability':[0.1, 0.2, 0.25, 0.5, 0.6, 0.8,

0.9]})

table['odds'] = table.probability/(1 - table.probability)

table
```

	probability	odds
0	0.10	0.111111
1	0.20	0.250000
2	0.25	0.333333
3	0.50	1.000000
4	0.60	1.500000
5	0.80	4.000000
6	0.90	9.000000

Widzimy więc, że wraz ze wzrostem prawdopodobieństwa wzrastają nasze szanse, ale w znacznie szybszym tempie! W rzeczywistości, ponieważ prawdopodobieństwo wystąpienia zdarzenia jest bliskie 1, nasze szanse będą spadać w nieskończoność. Wcześniej powiedzieliśmy, że nie możemy po prostu cofnąć się do prawdopodobieństwa, ponieważ nasza linia wystrzeliłaby w dodatnią i ujemną

nieskończoność, przewidując niewłaściwe prawdopodobieństwa, ale co, jeśli cofniemy się do szans? Cóż, szanse idą do dodatniej nieskończoności, ale niestety, po prostu zbliżą się do 0 na dole, ale nigdy nie spadną poniżej 0. Dlatego nie możemy po prostu cofnąć się do prawdopodobieństwa lub szans. Wygląda na to, że trafiliśmy na dno! Jednak poczekaj, liczby naturalne i logarytmy na ratunek! Pomyśl o logarytmach w następujący sposób:

$$if\ 2^4 = 16\ then\ \log_2 16 = 4$$

Zasadniczo logarytmy i wykładniki są jednym i tym samym. Jesteśmy tak przyzwyczajeni do pisania wykładników w pierwszy sposób, że zapominamy, że istnieje inny sposób ich zapisywania. A może inny przykład? Jeśli weźmiemy logarytm liczby, zadajemy pytanie, hej, jaki wykładnik musielibyśmy umieścić na tej liczbie, aby uzyskać podaną liczbę? Zauważ, że np.log automatycznie wykonuje wszystkie logarytmy w bazie e, a tego właśnie chcemy:

np.log(10) # == 2.3025

meaning that e ^ 2.302 == 10

to prove that

2.71828**2.3025850929940459 # == 9.9999

e ^ log(10) == 10

Przejdźmy dalej i dodajmy logarytm szans lub log-odds do naszej tabeli w następujący sposób:

add log-odds to the table

table['logodds'] = np.log(table.odds)

table

	probability	odds	logodds
0	0.10	0.111111	-2.197225
1	0.20	0.250000	-1.386294
2	0.25	0.333333	-1.098612
3	0.50	1.000000	0.000000
4	0.60	1.500000	0.405465
5	0.80	4.000000	1.386294
6	0.90	9.000000	2.197225

Tak więc teraz każdy wiersz ma prawdopodobieństwo wystąpienia pojedynczego zdarzenia, prawdopodobieństwo wystąpienia tego zdarzenia, a teraz logarytmiczne szanse wystąpienia tego zdarzenia. Idźmy dalej i upewnijmy się, że nasze liczby rosną. Wybierzmy prawdopodobieństwo 0,25, jak pokazano na rysunku:

prob = .25

```
odds = prob / (1 - prob)

odds

# 0.33333333

logodds = np.log(odds)

logodds

# -1.09861228
```

To się sprawdza! Czekaj, patrz! Nasza zmienna logodds wydaje się schodzić poniżej zera i tak naprawdę logodds nie jest ograniczone powyżej ani poniżej, co oznacza, że jest doskonałym kandydatem na zmienną odpowiedzi dla regresji liniowej. Właściwie to właśnie tutaj zaczyna się nasza historia logistycznej regresji.

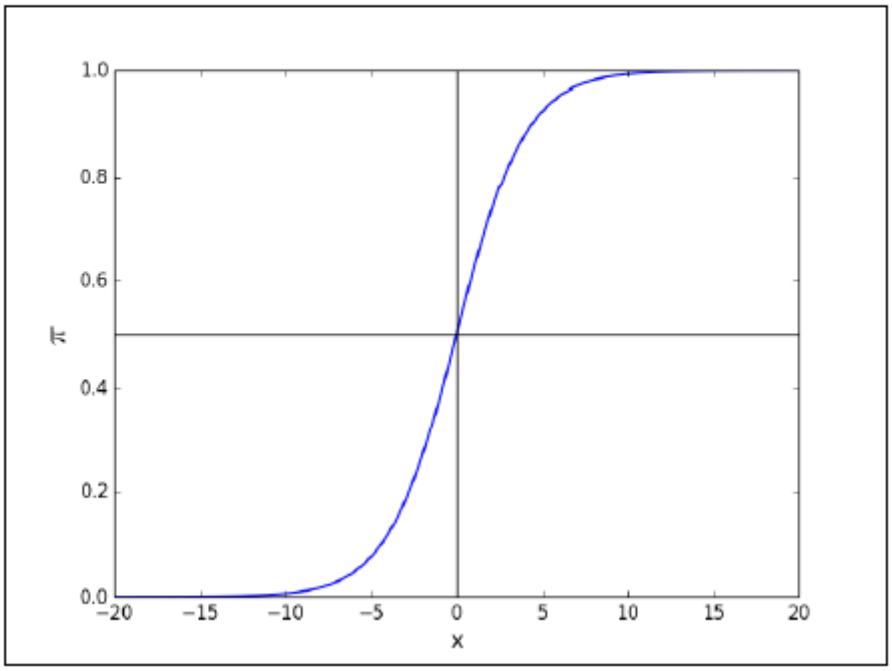
Matematyka regresji logistycznej

Krótko mówiąc, regresja logistyczna jest regresją liniową między naszą cechą X, a logarytmami naszych danych należących do pewnej klasy, którą nazwiemy prawdziwymi dla uogólnienia. Jeśli p reprezentuje prawdopodobieństwo przynależności punktu danych do określonej klasy, to regresję logistyczną można zapisać w następujący sposób:

$$\log_e\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x$$

Gdybyśmy przeorganizowali nasze zmienne i rozwiązali to dla p, otrzymalibyśmy funkcję logistyczną, która przyjmuje kształt S, gdzie y jest ograniczone przez [0,1]:

$$p = \frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}}$$



Powyższy wykres przedstawia zdolność funkcji logistycznej do odwzorowania naszych ciągłych danych wejściowych, x, na gładką krzywą prawdopodobieństwa, która zaczyna się po lewej stronie, bliskie prawdopodobieństwu 0, a gdy zwiększamy x, nasze prawdopodobieństwo przynależności do określonej klasy rośnie naturalnie i płynnie do prawdopodobieństwa 1. Innymi słowy:

• Regresja logistyczna daje wynik prawdopodobieństwa prawdziwości określonej klasy

• Te prawdopodobieństwa można przekształcić w przewidywania klas

Funkcja logistyczna ma kilka fajnych właściwości, takich jak:

Przybiera kształt litery S

• Wyjście jest ograniczone przez 0 i 1, tak jak powinno być prawdopodobieństwo Aby zinterpretować wyniki funkcji logistycznej, musimy zrozumieć różnicę między prawdopodobieństwem a szansami. Szanse zdarzenia są podawane przez iloraz prawdopodobieństwa zdarzenia przez jego uzupełnienie, jak pokazano:

$$odds = \frac{p}{1-p}$$

W regresji liniowej parametr β_1 reprezentuje zmianę zmiennej odpowiedzi dla zmiany jednostki w x. W regresji logistycznej β_1 reprezentuje zmianę logarytmu szans na zmianę jednostki w x. Oznacza to, że $e^{\beta 1}$daje nam zmianę prawdopodobieństwa na zmianę jednostki w x. Weź pod uwagę, że interesują nas zachowania zakupowe na urządzeniach mobilnych. Niech y będzie etykietą klasy oznaczającą zakup/brak zakupu, a x oznacza, czy telefon był iPhonem. Załóżmy również, że wykonujemy regresję

logistyczną i otrzymujemy β_1 = 0,693. W tym przypadku iloraz szans wynosi np.exp(0,693) = 2, co oznacza, że prawdopodobieństwo zakupu jest dwukrotnie większe, jeśli telefon jest iPhonem.

Nasze przykłady to głównie klasyfikacja binarna, co oznacza, że przewidujemy tylko jeden z dwóch wyników, ale regresja logistyczna może poradzić sobie z przewidywaniem wielu opcji w naszej odpowiedzi kategorycznej przy użyciu podejścia „jeden na wszystkie", co oznacza, że dopasuje krzywą prawdopodobieństwa dla każdej kategorii odpowiedź!

Wróćmy na chwilę do naszych motocykli, aby zobaczyć regres logistyczny scikit-learn w akcji. Zacznę od stworzenia nowej zmiennej odpowiedzi, która jest kategoryczna. Aby uprościć sprawę, stworzyłem kolumnę o nazwie above_average, która jest prawdziwa, jeśli godzinowa liczba wypożyczeń roweru jest powyżej średniej, a fałszywa w przeciwnym razie.

Make a cateogirical response

bikes['above_average'] = bikes['count'] >= average_bike_rental

Jak wspomniano wcześniej, powinniśmy przyjrzeć się naszemu modelowi zerowemu. W regresji nasz model zerowy zawsze przewiduje średnią odpowiedź, ale w klasyfikacji nasz model zerowy zawsze przewiduje najczęstszy wynik. W takim przypadku możemy użyć licznika wartości Pandy, aby to zobaczyć. Około 60% czasu wypożyczenia rowerów nie przekracza średniej:

bikes['above_average'].value_counts(normalize=True)

Teraz użyjmy regresji logistycznej, aby spróbować przewidzieć, czy godzinowa liczba wypożyczeń roweru będzie powyżej średniej, jak pokazano:

from sklearn.linear_model import LogisticRegression

feature_cols = ['temp']

using only temperature

X = bikes[feature_cols]

y = bikes['above_average']

make our overall X and y variables, this time our y is

out binary response variable, above_average

X_train, X_test, y_train, y_test = train_test_split(X, y)

make our train test split

logreg = LogisticRegression()

instantate our model

logreg.fit(X_train, y_train)

fit our model to our training set

logreg.score(X_test, y_test)

score it on our test set to get a better sense of out of sample

performance

0.65650257

Wygląda na to, że używając tylko temperatury, możemy pokonać zerowy model zgadywania fałszywego przez cały czas! To nasz pierwszy krok, aby uczynić nasz model najlepszym, jaki może być. Pomiędzy regresją liniową a logistyczną, powiedziałbym, że mamy już świetny pasek narzędzi do formowania uczenia maszynowego, ale mam pytanie - wydaje się, że oba te algorytmy są w stanie przyjąć tylko kolumny ilościowe jako cechy, ale co jeśli ja mam kategoryczną cechę, która moim zdaniem ma związek z moją odpowiedzią?

Zmienne fikcyjne

Zmienne pozorne są używane, gdy mamy nadzieję przekształcić cechę kategoryczną w cechę ilościową. Pamiętaj, że mamy dwa rodzaje cech kategorycznych: nominalne i porządkowe. Wśród nich cechy porządkowe mają naturalny porządek, podczas gdy dane nominalne nie. Kodowanie danych jakościowych (nominalnych) przy użyciu oddzielnych kolumn nazywa się tworzeniem zmiennych fikcyjnych i działa ono poprzez przekształcenie każdej unikalnej kategorii kolumny nominalnej we własną kolumnę, która jest albo prawdziwa, albo fałszywa. Na przykład, gdybyśmy mieli kolumnę dotyczącą czyjegoś kierunku studiów i chcielibyśmy umieścić te informacje w regresji liniowej lub logistycznej, nie moglibyśmy, ponieważ uwzględniają tylko liczby! Tak więc dla każdego wiersza mieliśmy nowe kolumny, które reprezentują pojedynczą kolumnę nominalną. W tym przypadku mamy cztery unikalne kierunki: informatykę, inżynierię, biznes i literaturę. Kończymy z trzema nowymi kolumnami (pomijamy informatykę, bo nie jest to konieczne).

Major (k=4)	Engineering	Business	Literature
Computer Science	0	0	0
Engineering	1	0	0
Business	0	1	0
Literature	0	0	1
Business	0	1	0
Engineering	1	0	0

Zwróć uwagę, że w pierwszym wierszu we wszystkich kolumnach jest 0, co oznacza, że ta osoba nie była specjalizacją inżynierską, biznesową ani literaturą. Druga osoba ma tylko 1 w kolumnie inżynierskiej, ponieważ jest to kierunek, który studiowali. W naszym przykładzie z rowerami zdefiniujmy nową kolumnę o nazwie when_is_it, która będzie jedną z czterech następujących opcji:

• Poranek

• Popołudnie

• Godziny szczytu

• poza godzinami

Aby to zrobić, naszym podejściem będzie utworzenie nowej kolumny, która jest po prostu godziną dnia, użyj tej kolumny, aby określić, kiedy jest dzień, i zbadaj, czy uważamy, że ta kolumna może nam pomóc przewidzieć kolumnę above_daily :

```
bikes['hour'] = bikes['datetime'].apply(lambda x:int(x[11]+x[12]))
```

make a column that is just the hour of the day

bikes['hour'].head()

0

1

2

3

Świetnie, teraz zdefiniujmy funkcję, która zamienia te godziny w ciągi. W tym przykładzie zdefiniujmy godziny między 5 a 11 jako rano, między 11 a 16 jako popołudnie, 4 i 6 jako godziny szczytu, a wszystko inne jako poza godzinami pracy:

this function takes in an integer hour

and outputs one of our four options

def when_is_it(hour):

if hour >= 5 and hour < 11:

return "morning"

elif hour >= 11 and hour < 16:

return "afternoon"

elif hour >= 16 and hour < 18:

return "rush_hour"

else:

return "off_hours"

Zastosujmy tę funkcję do naszej nowej kolumny godzin i utwórzmy nową kolumnę, when_is_it:

bikes['when_is_it'] = bikes['hour'].apply(when_is_it)

bikes[['when_is_it', 'above_average']].head()

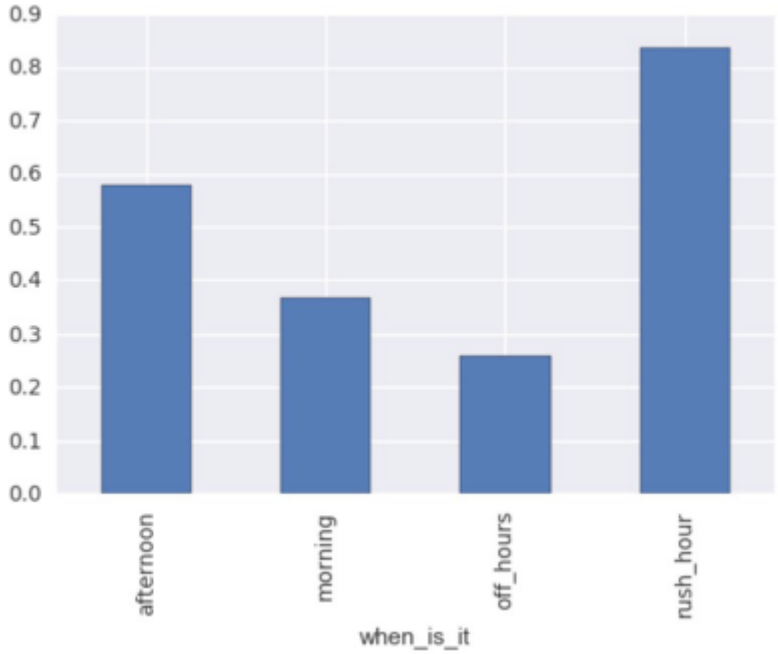
	when_is_it	above_average
0	off_hours	False
1	off_hours	False
2	off_hours	False
3	off_hours	False
4	off_hours	False

Spróbujmy użyć tylko tej nowej kolumny do określenia, czy godzinowa liczba wypożyczeń roweru będzie powyżej średniej. Zanim to zrobimy, zajmijmy się podstawami eksploracyjnej analizy danych i

stwórzmy wykres, aby zobaczyć, czy możemy zwizualizować różnicę między czterema porami dnia. Nasz wykres będzie wykresem słupkowym z jednym słupkiem na porę dnia.

Każdy słupek będzie przedstawiał procent przypadków, w których o tej porze dnia wypożyczono rowery większe niż normalne:

bikes.groupby('when_is_it').above_average.mean().plot(kind='bar')



Widzimy, że jest całkiem duża różnica! Na przykład, gdy jest poza godzinami pracy, szansa na ponadprzeciętne wypożyczenie roweru wynosi około 25%, podczas gdy w godzinach szczytu szansa na bycie powyżej średniej wynosi ponad 80%! OK, to jest ekscytujące, ale użyjmy wbudowanych narzędzi Pandy, aby wyodrębnić atrapy kolumn w następujący sposób:

when_dummies = pd.get_dummies(bikes['when_is_it'], prefix='when__')

when_dummies.head()

	when___afternoon	when___morning	when___off_hours	when___rush_hour
0	0.0	0.0	1.0	0.0
1	0.0	0.0	1.0	0.0
2	0.0	0.0	1.0	0.0
3	0.0	0.0	1.0	0.0
4	0.0	0.0	1.0	0.0

when_dummies = when_dummies.iloc[:, 1:]

remove the first column

when_dummies.head()

	when___morning	when___off_hours	when___rush_hour
0	0.0	1.0	0.0
1	0.0	1.0	0.0
2	0.0	1.0	0.0
3	0.0	1.0	0.0
4	0.0	1.0	0.0

Świetny! Teraz mamy Dataframe pełną liczb, które możemy podłączyć do naszej regresji logistycznej:

```
X = when_dummies

# our new X is our dummy variables

y = bikes.above_average

logreg = LogisticRegression()

# instantate our model

logreg.fit(X_train, y_train)

# fit our model to our training set

logreg.score(X_test, y_test)

# score it on our test set to get a better sense of out of sample

performance

# 0.685157
```

Co jest nawet lepsze niż samo używanie temperatury! A jeśli dołożymy do tego temperaturę i wilgotność? Więc teraz używamy temperatury, wilgotności i naszych fikcyjnych zmiennych pory dnia, aby przewidzieć, czy będziemy mieli więcej wypożyczonych rowerów:

```
new_bike = pd.concat([bikes[['temp', 'humidity']], when_dummies],

axis=1)

# combine temperature, humidity, and the dummy variables

X = new_bike

# our new X is our dummy variables

y = bikes.above_average

X_train, X_test, y_train, y_test = train_test_split(X, y)

logreg = LogisticRegression()

# instantate our model

logreg.fit(X_train, y_train)

# fit our model to our training set
```

logreg.score(X_test, y_test)

score it on our test set to get a better sense of out of sample

performance

0.7182218

Wow. Dobra, skończmy.

Podsumowanie

W tej Części przyjrzeliśmy się uczeniu maszynowemu i jego różnym podkategoriom. Zbadaliśmy nadzorowane, nienadzorowane i wzmacniające strategie uczenia się oraz przyjrzeliśmy się sytuacjom, w których każda z nich byłaby przydatna. Analizując regresję liniową, byliśmy w stanie znaleźć relacje między predyktorami a zmienną odpowiedzi ciągłej. Dzięki podziałowi na pociąg/test udało nam się uniknąć nadmiernego dopasowania naszych modeli uczenia maszynowego i uzyskać bardziej uogólnioną prognozę. Byliśmy w stanie wykorzystać metryki, takie jak pierwiastek błędu średniokwadratowego, również do oceny naszych modeli. Rozszerzając nasze pojęcie regresji liniowej na regresję logistyczną, byliśmy w stanie znaleźć związek między tymi samymi predyktorami, ale teraz z odpowiedziami kategorycznymi. Dzięki wprowadzeniu do miksu zmiennych fikcyjnych byliśmy w stanie dodać cechy kategoryczne do naszych modeli i jeszcze bardziej poprawić naszą wydajność. W następnych kilku Częściach przyjrzymy się znacznie większej liczbie modeli uczenia maszynowego, a po drodze nauczymy się nowych wskaźników, nowych technik walidacji i, co ważniejsze, nowych sposobów zastosowania naszej nauki o danych na świecie .

Prognozy nie rosną na drzewach — a może tak?

W tej części przyjrzymy się trzem rodzajom algorytmów uczenia maszynowego. Pierwsze dwa to przykłady uczenia nadzorowanego, podczas gdy ostateczny algorytm jest przykładem uczenia się nienadzorowanego. Naszym celem jest zobaczenie i zastosowanie pojęć poznanych w poprzednich częściach w celu skonstruowania i wykorzystania nowoczesnych algorytmów uczenia się w celu zebrania spostrzeżeń i przewidywania rzeczywistych zbiorów danych. Badając poniższe algorytmy, zawsze pamiętajmy, że stale pamiętamy o naszych metrykach. Weźmy się za to!

Naiwny klasyfikator bayesowski

Przejdźmy od razu! Zacznijmy od klasyfikacji Naïve Bayes. Ten model uczenia maszynowego w dużej mierze opiera się na wynikach z poprzednich części, w szczególności z twierdzeniem Bayesa:

$$P(H \mid D) = \frac{P(D \mid H)P(H)}{P(D)}$$

Przyjrzyjmy się bliżej szczegółowym cechom tej formuły:

• P(H) jest prawdopodobieństwem hipotezy przed zaobserwowaniem danych, zwanym prawdopodobieństwem a priori lub po prostu przed

• P(H|D) to to, co chcemy obliczyć, prawdopodobieństwo hipotezy po zaobserwowaniu danych, zwane a posteriori

• P(D|H) to prawdopodobieństwo danych w ramach danej hipotezy, zwane prawdopodobieństwem

• P(D) to prawdopodobieństwo danych przy dowolnej hipotezie, zwane normalizującą stałą

Klasyfikacja Naïve Bayes jest modelem klasyfikacyjnym, a więc modelem nadzorowanym. Biorąc to pod uwagę, jakiego rodzaju danych potrzebujemy?

• Oznaczone dane

• Dane nieoznakowane

(tu wstaw muzykę zagrożenia)

Jeśli odpowiedziałeś na dane oznaczone etykietą, jesteś na dobrej drodze, aby zostać naukowcem danych!

Załóżmy, że mamy zestaw danych z n cechami (x1, x2, …, xn) i etykietą klasy C. Weźmy na przykład dane dotyczące klasyfikacji tekstu spamu. Nasze dane składałyby się z rzędów pojedynczych próbek tekstu i kolumn zarówno naszych funkcji, jak i etykiet klas. Naszymi cechami byłyby słowa i wyrażenia zawarte w próbce tekstu i nasze etykiety klasowe są po prostu spamem lub nie są spamem. W tym scenariuszu zastąpię klasę nie spamuj łatwiejszym do powiedzenia słowem, ham:

```
import pandas as pd

import sklearn

df = pd.read_table('https://raw.githubusercontent.com/sinanuozdemir/

sfdat22/master/data/sms.tsv',
```

sep='\t', header=None, names=['label', 'msg'])

df

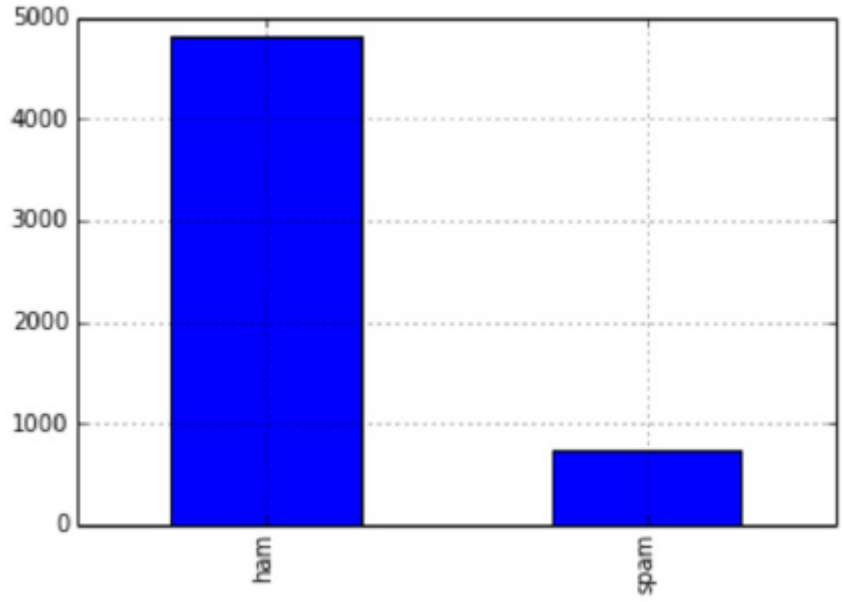
Oto próbka danych tekstowych w formacie kolumny wiersza:

	label	msg
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
5	spam	FreeMsg Hey there darling it's been 3 week's n...
6	ham	Even my brother is not like to speak with me. ...
7	ham	As per your request 'Melle Melle (Oru Minnamin...

Zróbmy wstępne statystyki, aby zobaczyć, z czym mamy do czynienia. Zobaczmy różnicę w ilości wiadomości ham i spam, którymi dysponujemy:

df.label.value_counts().plot(kind="bar")

Daje nam to wykres słupkowy w następujący sposób:



Więc mamy DUŻO więcej wiadomości krótkofalowych niż spamu. Ponieważ jest to problem z klasyfikacją, bardzo przydatne będzie poznanie naszego współczynnika zerowej dokładności, który jest procentową szansą na prawidłowe przewidzenie pojedynczego wiersza, jeśli nadal zgadujemy najczęstszą klasę, ham:

df.label.value_counts() / df.shape[0]

ham 0.865937

spam 0.134063

Więc gdybyśmy na ślepo odgadli ham, mielibyśmy rację w około 87% przypadków, ale możemy to zrobić lepiej. Jeśli mamy zbiór klas C i cechy xi, to możemy użyć twierdzenia Bayesa do przewidzenia prawdopodobieństwa przynależności pojedynczego wiersza do klasy C za pomocą następującego wzoru:

$$P(class\,C|\{x_i\}) = \frac{P(\{x_i\}|class\,C)\cdot P(class\,C)}{P(\{x_i\})}$$

Przyjrzyjmy się tej formule bardziej szczegółowo:

• P(klasa C | {xi}): Prawdopodobieństwo a posteriori to prawdopodobieństwo, że wiersz należy do klasy C przy danych cechach {xi}.

• P({xi} | klasa C): Jest to prawdopodobieństwo, że zaobserwowalibyśmy te cechy, biorąc pod uwagę, że wiersz był w klasie C.

• P(klasa C): Jest to prawdopodobieństwo a priori. Jest to prawdopodobieństwo, że punkt danych należy do klasy C, zanim zobaczymy jakiekolwiek dane.

• P({xi}): To jest nasza stała normalizacji.

Na przykład wyobraź sobie, że mamy wiadomość e-mail z trzema słowami: wyślij gotówkę teraz. Użyjemy Naïve Bayes do sklasyfikowania wiadomości e-mail jako spamu lub ham:

P(spam| wyślij gotówkę teraz) = P(wyślij gotówkę teraz| spam) P(spam) / P(wyślij gotówkę teraz)

P(ham| wyślij gotówkę teraz) = P(wyślij gotówkę teraz| ham) P(ham) / P(wyślij gotówkę teraz)

Interesuje nas różnica tych dwóch liczb. Aby sklasyfikować dowolny pojedynczy fragment tekstu, możemy zastosować następujące kryteria:

• Jeśli P(spam | wyślij gotówkę teraz) jest większe niż P(szynka | wyślij gotówkę teraz), zaklasyfikujemy tekst jako spam

• Jeśli P(ham | wyślij gotówkę teraz) jest większe niż P(spam | wyślij gotówkę teraz), wtedy oznaczymy tekst jako ham

Ponieważ oba równania mają P (wyślij pieniądze teraz) w mianowniku, możemy je zignorować. Więc teraz zajmujemy się następującymi kwestiami:

P(wyślij gotówkę teraz| spam) P(spam) VS P(wyślij gotówkę| ham) * P(ham)

Obliczmy liczby w tym równaniu:

• P(spam) = 0,134063

• P(ham) = 0,865937

• P(wyślij gotówkę teraz | spam)

• P (wyślij gotówkę teraz | ham)

Wydaje się, że ostatnie dwa prawdopodobieństwa nie są tak trudne do obliczenia. Wystarczy policzyć liczbę wiadomości spamowych zawierających frazę „wyślij pieniądze teraz” i podzielić ją przez całkowitą liczbę wiadomości spamowych:

df.msg = df.msg.apply(lambda x:x.lower())

zrób wszystkie ciągi małymi literami, abyśmy mogli łatwiej wyszukiwać

df[df.msg.str.contains('wyślij gotówkę teraz')] .shape

(0, 2)

O nie! Nie ma żadnych! Nie ma dosłownie 0 tekstów z dokładną frazą wyślij gotówkę teraz. Ukryty problem polega na tym, że ta fraza jest bardzo specyficzna i nie możemy zakładać, że będziemy mieć wystarczająco dużo danych na świecie, aby zobaczyć tę dokładną frazę wiele razy wcześniej. Zamiast tego możemy przyjąć naiwne założenie w naszym twierdzeniu Bayesa. Jeśli przyjmiemy, że cechy (słowa) są warunkowo niezależne (czyli żadne słowo nie wpływa na istnienie innego słowa), to możemy przepisać formułę:

P(send cash now| spam) = P(send | spam)P(cash | spam)P(now| spam)

spams = df[df.label == 'spam']

for word in ['send', 'cash', 'now']:

print word, spams[spams.msg.str.contains(word)].shape[0] /

float(spams.shape[0])revealing

• P(wyślij|spam) = 0,096

• P(gotówka |spam) = 0,091

• P(teraz|spam) = 0,280

Oznacza to, że możemy obliczyć:

P(wyślij gotówkę teraz| spam) * P(spam)= (0,096*0,091*,280)*0,134 = 0,00032

Powtórzenie tej samej procedury dla szynki daje nam następujące informacje:

• P(wyślij|ham) = 0,03

• P(gotówka|ham) = 0,003

• P(teraz|ham) = 0,109

P(wyślij gotówkę teraz| ham) * P(ham)= (.03*.003*.109)*.865 = 0.0000084

Fakt, że obie te liczby są bardzo niskie, nie jest tak ważny, jak fakt, że prawdopodobieństwo spamu jest znacznie większe niż w przypadku wyliczenia szynki. Jeśli obliczymy .00032 / .00000084 = 38,1 zobaczymy, że prawdopodobieństwo wysłania gotówki teraz dla spamu jest 38 razy wyższe niż dla spamu. Dzięki temu możemy teraz zaklasyfikować wysłanie gotówki jako spam! Proste, prawda? Użyjmy Pythona do zaimplementowania klasyfikatora Naïve Bayes bez konieczności samodzielnego wykonywania wszystkich tych obliczeń. Najpierw wróćmy do wektoryzatora licznika w scikit-learn, który zamienia dla nas tekst w dane liczbowe. Załóżmy, że będziemy szkolić się na trzech dokumentach (zdaniach):

```
# simple count vectorizer example

from sklearn.feature_extraction.text import CountVectorizer

# start with a simple example

train_simple = ['call you tonight',

'Call me a cab',

'please call me... PLEASE 44!']

# learn the 'vocabulary' of the training data

vect = CountVectorizer()

train_simple_dtm = vect.fit_transform(train_simple)

pd.DataFrame(train_simple_dtm.toarray(), columns=vect.get_feature_

names())
```

	44	cab	call	me	please	tonight	you
0	0	0	1	0	0	1	1
1	0	1	1	1	0	0	0
2	1	0	1	1	2	0	0

Zauważ, że każdy wiersz reprezentuje jeden z trzech dokumentów (zdań), każda kolumna reprezentuje jedno ze słów obecnych w dokumentach, a każda komórka zawiera liczbę wystąpień każdego słowa w każdym dokumencie. Możemy następnie użyć wektoryzatora liczby do przekształcenia nowych przychodzących dokumentów testowych, aby były zgodne z naszym zbiorem uczącym (trzy zdania):

```
# transform testing data into a document-term matrix (using existing

vocabulary, notice don't is missing)

test_simple = ["please don't call me"]

test_simple_dtm = vect.transform(test_simple)

test_simple_dtm.toarray()

pd.DataFrame(test_simple_dtm.toarray(), columns=vect.get_feature_

names())
```

	44	cab	call	me	please	tonight	you
0	0	0	1	1	1	0	0

Zwróć uwagę, że w naszym zdaniu testowym pojawiło się nowe słowo, a mianowicie nie. Kiedy dokonaliśmy wektoryzacji, ponieważ wcześniej nie widzieliśmy tego słowa w naszych danych treningowych, wektoryzator po prostu je zignorował. Jest to ważne i zachęca analityków danych do uzyskania jak największej ilości danych dla swoich zestawów szkoleniowych. Teraz zróbmy to dla naszych rzeczywistych danych:

```
# split into training and testing sets

from sklearn.cross_validation import train_test_split

X_train, X_test, y_train, y_test = train_test_split(df.msg, df.label,

random_state=1)

# instantiate the vectorizer

vect = CountVectorizer()

# learn vocabulary and create document-term matrix in a single step

train_dtm = vect.fit_transform(X_train)

train_dtm

<4179x7456 sparse matrix of type '<type 'numpy.int64'>'
```

Z 55209 przechowywanymi elementami w skompresowanym formacie surowym. Zauważ, że format jest w rzadkiej macierzy, co oznacza, że macierz jest tak duża i pełna zer, że istnieje specjalny format do obsługi obiektów takich jak ten. Spójrz na liczbę kolumn. 7456 słów!! Oznacza to, że w naszym zestawie szkoleniowym znajduje się 7456 unikalnych słów, na które warto zwrócić uwagę. Możemy teraz przekształcić nasze dane testowe, aby były zgodne z naszym słownictwem:

```
# transform testing data into a document-term matrix

test_dtm = vect.transform(X_test)

test_dtm

<1393x7456 sparse matrix of type '<type 'numpy.int64'>'
```

Z 17604 przechowywanymi elementami w skompresowanym formacie surowym. Zauważ, że mamy taką samą dokładną liczbę kolumn, ponieważ zgodnie z naszym zestawem testowym jest dokładnie to samo słownictwo co poprzednio. Nie więcej nie mniej. Teraz zbudujmy model Naïve Bayes (podobny do procesu regresji liniowej):

```
## MODEL BUILDING WITH NAIVE BAYES

# train a Naive Bayes model using train_dtm

from sklearn.naive_bayes import MultinomialNB

# import our model

nb = MultinomialNB()

# instantiate our model

nb.fit(train_dtm, y_train)
```

fit it to our training set

Teraz zmienna nb zawiera nasz dopasowany model. Faza uczenia modelu obejmuje obliczenie funkcji wiarygodności, która jest prawdopodobieństwem warunkowym każdej cechy danej klasy:

make predictions on test data using test_dtm

preds = nb.predict(test_dtm)

preds

array(['ham', 'ham', 'ham', ..., 'ham', 'spam', 'ham'],

dtype='|S4')

Faza predykcji modelu polega na obliczeniu prawdopodobieństwa a posteriori każdej klasy przy danych obserwowanych cechach i wybraniu klasy o najwyższym prawdopodobieństwie. Wykorzystamy wbudowaną macierz dokładności i pomyłek sklearna, aby sprawdzić, jak dobrze radzą sobie nasze modele Naïve Bayes:

compare predictions to true labels

from sklearn import metrics

print metrics.accuracy_score(y_test, preds)

print metrics.confusion_matrix(y_test, preds)

accuracy == 0.988513998564

confusion matrix ==

[[1203 5]

[11 174]]

Po pierwsze, nasza dokładność jest świetna! W porównaniu z naszą zerową dokładnością, która wynosiła 87%, 99% to fantastyczna poprawa. Przejdźmy teraz do naszej matrycy pomyłek. Już wcześniej wiemy, że każdy wiersz reprezentuje rzeczywiste wartości, podczas gdy kolumny reprezentują wartości przewidywane, więc lewa górna wartość, 1203, reprezentuje nasze prawdziwe negatywy. Ale co jest negatywne i pozytywne? Daliśmy modelowi ciągi spam i ham jako nasze klasy, a nie pozytywne i negatywne. Możemy skorzystać z:

nb.classes_

array(['ham', 'spam'])

Następnie możemy ułożyć indeksy w taki sposób, aby 1203 odnosiło się do prawdziwych podpowiedzi dotyczących szynek, a 174 do prawdziwych podpowiedzi dotyczących spamu. Istniało również pięć fałszywych klasyfikacji spamu, co oznacza, że pięć wiadomości zostało przewidzianych jako spam, ale w rzeczywistości były to szynki, a także 11 fałszywych klasyfikacji szynek. Podsumowując, klasyfikacja Naïve Bayes wykorzystuje twierdzenie Bayesa w celu dopasowania prawdopodobieństw a posteriori klas tak, aby punkty danych były poprawnie oznaczone jako należące do właściwej klasy.

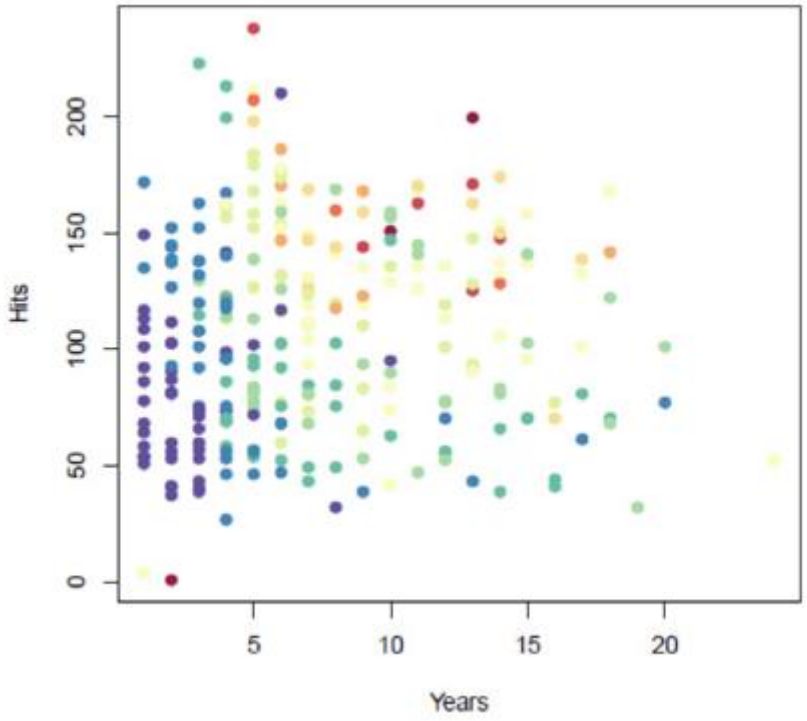
Drzewa decyzyjne

Drzewa decyzyjne to nadzorowane modele, które mogą przeprowadzać regresję lub klasyfikację. Rzućmy okiem na niektóre dane dotyczące graczy z głównych lig baseballowych z lat 1986-1987. Każda kropka reprezentuje jednego gracza w lidze:

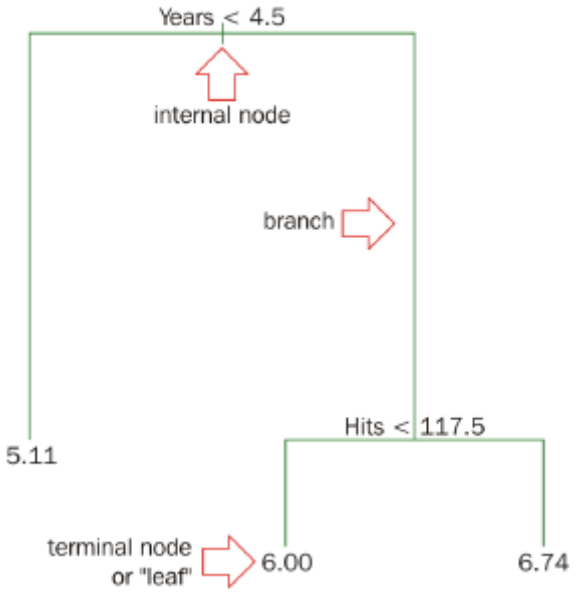
• Lata (oś x): liczba lat granych w głównych ligach

• Trafienia (oś y): liczba trafień gracza w poprzednim roku

• Wynagrodzenie (kolor): Niska pensja jest niebiesko/zielona, wysoka jest czerwono/żółta



Powyższe dane to nasze dane treningowe. Pomysł polega na zbudowaniu modelu, który przewiduje pensje przyszłych graczy w oparciu o lata i hity. Drzewo decyzyjne ma na celu dokonanie podziałów na naszych danych w celu segmentacji punktów danych, które działają podobnie do siebie, ale inaczej niż pozostałe. Drzewo wykonuje wielokrotność tych podziałów, aby stworzyć najdokładniejsze możliwe przewidywanie. Zobaczmy drzewo zbudowane dla poprzednich danych:



Czytanie od góry do dołu:

• Pierwszy podział to Lata < 4,5, gdy reguła podziału jest prawdziwa, podążaj za lewą gałęzią. Gdy reguła dzielenia jest fałszywa, podążasz za prawą gałęzią. Czyli dla nowego gracza, jeśli gra krócej niż 4,5 roku, zejdziemy w lewą gałąź.

• Dla graczy w lewej gałęzi średnia pensja wynosi 166 000 $, dlatego oznaczasz ją tą wartością (wynagrodzenie zostało podzielone przez 1000 i przeliczone logarytmicznie do 5,11 dla ułatwienia obliczeń).

• W przypadku graczy w prawej gałęzi istnieje dalszy podział na trafienia < 117,5, dzielący graczy na dwa dodatkowe regiony wynagrodzeń: 403 000 $ (przekształcone do 6,00) i 846 000 $ (przekształcone do 6,74).

To drzewo nie tylko daje nam prognozy; oznacza to również więcej informacji o naszych danych:

• Wydaje się, że liczba lat w lidze jest najważniejszym czynnikiem określającym wynagrodzenie, przy czym mniejsza liczba lat wiąże się z niższym wynagrodzeniem

• Jeśli gracz nie grał przez długi czas (< 4,5 roku), liczba trafień, które otrzymał, nie jest ważnym czynnikiem, jeśli chodzi o jego pensję

• W przypadku graczy, którzy mają za pasem co najmniej 5 lat, trafienia są ważnym czynnikiem wpływającym na ich pensję

• Nasze drzewo podjęło tylko dwie decyzje, zanim wypluło odpowiedź (dwie to nasza głębia drzewa)

Jak komputer buduje drzewo regresji?

Nowoczesne algorytmy drzew decyzyjnych mają tendencję do używania rekurencyjnego podejścia do dzielenia binarnego:

1. Proces rozpoczyna się na szczycie drzewa.

2. Dla każdej cechy zbada każdy możliwy podział i wybierze cechę i podział tak, aby wynikowe drzewo miało najniższy możliwy błąd średniokwadratowy (MSE). Algorytm dokonuje tego podziału.

3. Następnie zbada dwa powstałe regiony i ponownie dokona pojedynczego podziału (w jednym z regionów), aby zminimalizować MSE.

4. Powtarzaj krok 3 aż do spełnienia kryterium zatrzymania:

- Maksymalna głębokość drzewa (maksymalna liczba podziałów wymagana do uzyskania liścia)

- Minimalna liczba obserwacji w węźle liścia (końcowym)

W przypadku drzew klasyfikacyjnych algorytm jest bardzo podobny, a największą różnicą jest metryka, którą optymalizujemy. Ponieważ MSE istnieje tylko w przypadku problemów z regresją, nie możemy go używać. Jednak zamiast dokładności, drzewa klasyfikacyjne optymalizują albo indeks Giniego, albo entropię.

Jak komputer pasuje do drzewa klasyfikacyjnego?

Podobnie jak w przypadku drzewa regresji, drzewo klasyfikacyjne jest budowane poprzez optymalizację względem metryki (w tym przypadku indeksu giniego) i wybranie najlepszego podziału do przeprowadzenia tej optymalizacji. Bardziej formalnie, w każdym węźle drzewo wykona następujące kroki:

1. Oblicz czystość danych.

2. Wybierz podział kandydata.

3. Oblicz czystość danych po podziale.

4. Powtórz dla wszystkich zmiennych.

5. Wybierz zmienną o największym wzroście czystości.

6. Powtarzaj dla każdego podziału, aż spełnione zostaną pewne kryteria zatrzymania.

Załóżmy, że przewidujemy prawdopodobieństwo śmierci na pokładzie luksusowego statku wycieczkowego, biorąc pod uwagę cechy demograficzne. Załóżmy, że zaczynamy od 25 osób, z których 10 przeżyło, a 15 zmarło:

Before Split	All
Survived	10
Died	15

Najpierw obliczamy indeks gini, zanim cokolwiek zrobimy:

$$1-\sum\left(\frac{class_i}{total}\right)^2$$

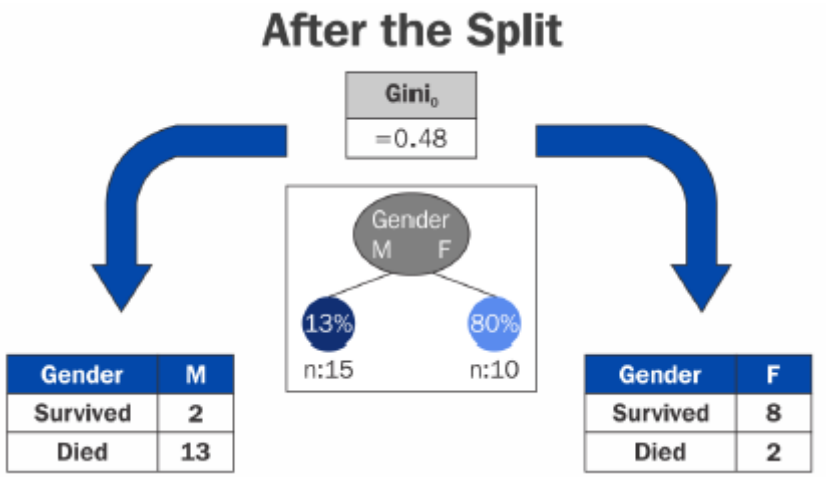
Klasy ogółem (w tym przypadku przeżyli i zmarli):

$$1-\left(\frac{survived}{total}\right)^2-\left(\frac{died}{total}\right)^2$$

$$1-\left(\frac{10}{25}\right)^2-\left(\frac{15}{25}\right)^2=0.48$$

Oznacza to, że czystość zbioru danych wynosi 0,48.

Rozważmy teraz potencjalny podział ze względu na płeć: Najpierw obliczamy indeks giniego dla każdej płci:



$$\text{gini(m)} = 1-\left(\frac{2}{15}\right)^2+\left(\frac{13}{15}\right)^2=.23$$

$$\text{gini(f)} = 1-\left(\frac{8}{10}\right)^2+\left(\frac{2}{10}\right)^2=.32$$

Po uzyskaniu indeksu giniego dla KAŻDEJ płci obliczamy ogólny indeks giniego dla podziału według płci w następujący sposób:

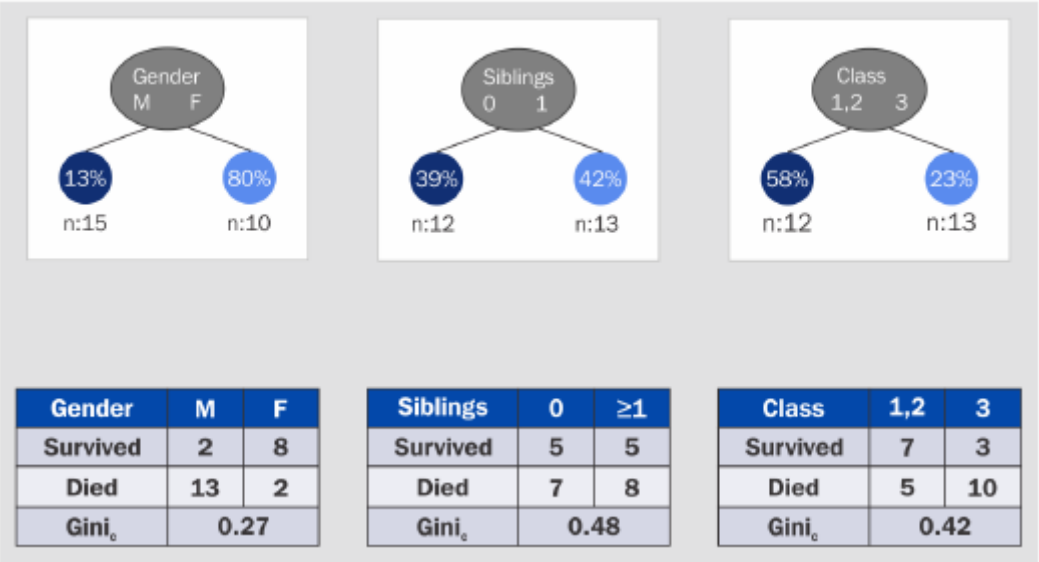
Gini (M)(M / M + F) + Gini (F)(F / M + F) = 0,23(15/10+15) + 0,32(10/10+15) = 0,27

Tak więc współczynnik giniego dla podziału ze względu na płeć wynosi 0,27. Następnie postępujemy zgodnie z tą procedurą dla trzech potencjalnych podziałów:

• Płeć (mężczyzna lub kobieta)

• Liczba rodzeństwa na pokładzie (0 lub 1+)

• Klasa (pierwsza i druga kontra trzecia)



Gender	M	F
Survived	2	8
Died	13	2
$Gini_g$	0.27	

Siblings	0	≥1
Survived	5	5
Died	7	8
$Gini_g$	0.48	

Class	1,2	3
Survived	7	3
Died	5	10
$Gini_g$	0.42	

W tym przykładzie wybralibyśmy płeć do podziału, ponieważ jest to najniższy wskaźnik gini! W poniższej tabeli krótko podsumowano różnice między drzewami decyzyjnymi klasyfikacji i regresji:

Drzewa regresji: Drzewa klasyfikacyjne

Przewiduj odpowiedź ilościową : Przewiduj odpowiedź jakościową

Przewidywanie to średnia wartość w każdym liściu : Przewidywanie to najczęstsza etykieta w każdym liściu

Podziały są wybierane w celu zminimalizowania MSE: Podziały są wybierane w celu zminimalizowania wskaźnika giniego (zwykle)

Użyjmy wbudowanego drzewa decyzyjnego scikit-learn, aby zbudować drzewo decyzyjne:

```
# read in the data

titanic = pd.read_csv('titanic.csv')

# encode female as 0 and male as 1

titanic['Sex'] = titanic.Sex.map({'female':0, 'male':1})

# fill in the missing values for age with the median age

titanic.Age.fillna(titanic.Age.median(), inplace=True)

# create a DataFrame of dummy variables for Embarked

embarked_dummies = pd.get_dummies(titanic.Embarked, prefix='Embarked')

embarked_dummies.drop(embarked_dummies.columns[0], axis=1,

inplace=True)

# concatenate the original DataFrame and the dummy DataFrame

titanic = pd.concat([titanic, embarked_dummies], axis=1)

# define X and y
```

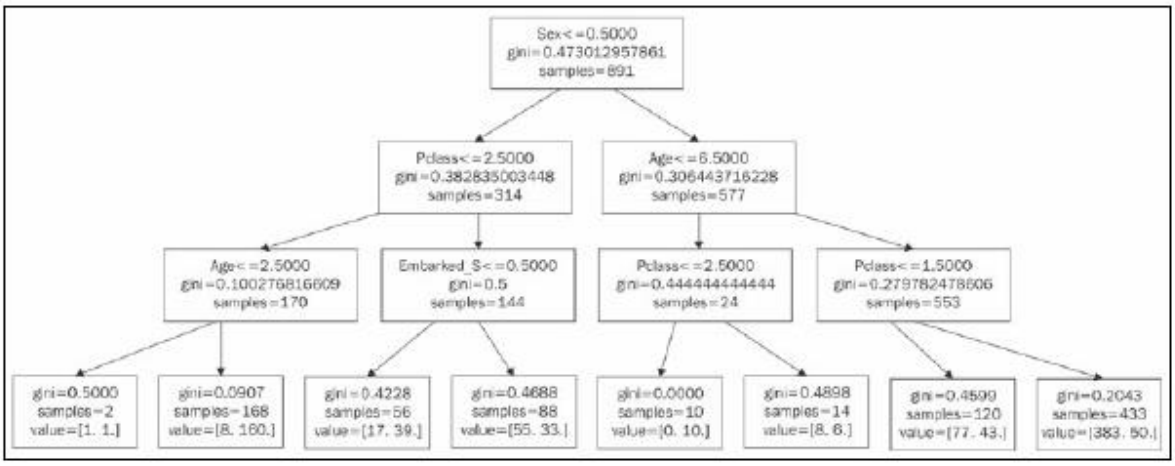
```
feature_cols = ['Pclass', 'Sex', 'Age', 'Embarked_Q', 'Embarked_S']

X = titanic[feature_cols]

y = titanic.Survived

X.head()
```

	Pclass	Sex	Age	Embarked_Q	Embarked_S
0	3	1	22.0	0.0	1.0
1	1	0	38.0	0.0	0.0
2	3	0	26.0	0.0	1.0
3	1	0	35.0	0.0	1.0
4	3	1	35.0	0.0	1.0

Zwróć uwagę, że będziemy używać zmiennych klasy, płci, wieku i fikcyjnych dla miasta, z którego wyruszamy, jako naszych funkcji:

```
# fit a classification tree with max_depth=3 on all data

from sklearn.tree import DecisionTreeClassifier

treeclf = DecisionTreeClassifier(max_depth=3, random_state=1)

treeclf.fit(X, y)
```

max_depth to ograniczenie głębokości naszego drzewa. Oznacza to, że dla dowolnego punktu danych nasze drzewo jest w stanie zadać maksymalnie trzy pytania i wykonać maksymalnie trzy podziały. Możemy wyprowadzić nasze drzewo w formacie wizualnym i otrzymamy:



Możemy zauważyć kilka rzeczy:

• Płeć to pierwszy podział, co oznacza, że płeć jest najważniejszym czynnikiem decydującym o tym, czy dana osoba przeżyła katastrofę

• Embarked_Q nigdy nie był używany w żadnym splicie

W przypadku drzew klasyfikacyjnych lub regresji możemy również zrobić coś bardzo interesującego z drzewami decyzyjnymi, co oznacza, że możemy wypisać liczbę reprezentującą znaczenie każdej cechy w przewidywaniu naszych punktów danych:

compute the feature importances

pd.DataFrame({'feature':feature_cols, 'importance':treeclf.feature_

importances_})

	feature	importance
0	Pclass	0.242664
1	Sex	0.655584
2	Age	0.064494
3	Embarked_Q	0.000000
4	Embarked_S	0.037258

Wyniki ważności są średnią różnicą wskaźnika Giniego dla każdej zmiennej, przy czym wyższe wartości odpowiadają wyższej ważności prognozy. Możemy wykorzystać te informacje do wybrania mniejszej liczby funkcji w przyszłości. Na przykład obydwie zmienne, na które się zaangażowaliśmy, są bardzo niskie w porównaniu z resztą cech, więc możemy powiedzieć, że nie są one ważne w przewidywaniu życia lub śmierci.

Nauka nienadzorowana

Czas zobaczyć kilka przykładów uczenia się bez nadzoru, biorąc pod uwagę, że większość tej książki poświęcamy modelom uczenia nadzorowanego.

Kiedy używać uczenia nienadzorowanego

Często zdarza się, że nauka bez nadzoru może być właściwa. Oto kilka bardzo popularnych przykładów:

• Gdy nie ma wyraźnej zmiennej odpowiedzi. Nie ma niczego, co wyraźnie próbujemy przewidzieć lub skorelować z innymi zmiennymi.

• Aby wyodrębnić strukturę z danych, w których nie ma widocznej struktury/wzorów (może to być nadzorowany problem uczenia się).

• Gdy używana jest nienadzorowana koncepcja nazywana wyodrębnianiem cech. Wyodrębnianie funkcji to proces tworzenia nowych funkcji z istniejących. Te nowe funkcje mogą być nawet silniejsze niż funkcje oryginalne.

Pierwszy jest najczęstszym powodem, dla którego analitycy danych decydują się na korzystanie z uczenia nienadzorowanego. Ten przypadek pojawia się często, gdy pracujemy z danymi i nie próbujemy wyraźnie przewidzieć żadnej z kolumn, a jedynie chcemy znaleźć wzorce podobnych (i niepodobnych) grup punktów. Pojawia się druga opcja w grze, nawet jeśli wyraźnie próbujemy użyć nadzorowanego modelu do przewidywania zmiennej odpowiedzi. Czasami proste EDA może nie tworzyć żadnych wyraźnych wzorców w danych w kilku wymiarach, które ludzie mogą sobie wyobrazić, gdzie jako maszyna może wychwycić punkty danych zachowujące się podobnie do siebie w większych

wymiarach. Trzecim powszechnym powodem korzystania z uczenia nienadzorowanego jest wyodrębnianie nowych funkcji z już istniejących. Ten proces (z miłością nazywany wyodrębnianiem cech) może wytworzyć cechy, które można wykorzystać w przyszłym nadzorowanym modelu lub wykorzystać do celów prezentacyjnych (marketingowych lub innych).

Grupowanie K-średnich

Klastrowanie K-średnich jest naszym pierwszym przykładem nienadzorowanego modelu uczenia maszynowego. Pamiętaj, że oznacza to, że nie robimy prognoz; zamiast tego próbujemy wyodrębnić strukturę z pozornie nieustrukturyzowanych danych. Klastrowanie to rodzina nienadzorowanych modeli uczenia maszynowego, które próbują grupować punkty danych w klastry z centroidami.

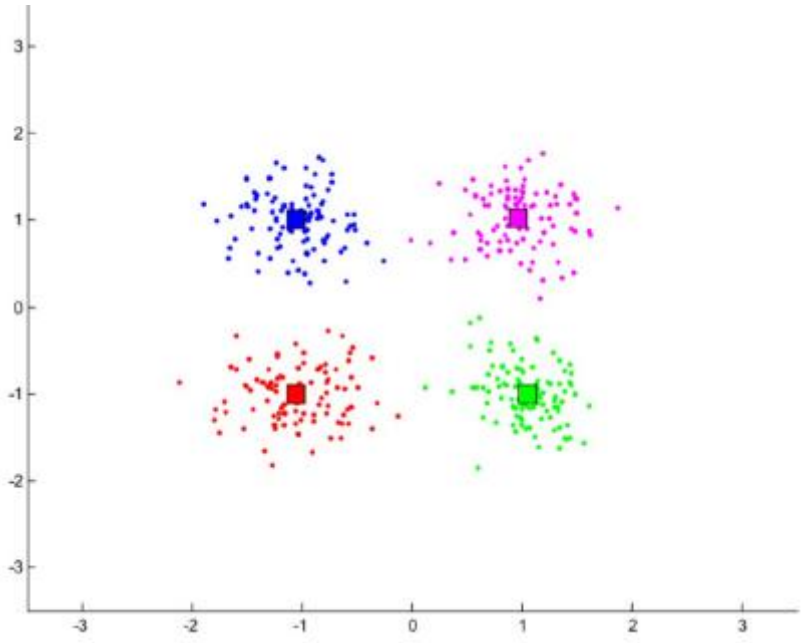
Definicja:

Klaster: grupa punktów danych, które zachowują się podobnie.

Definicja:

Centroid: Środek gromady. Może być traktowany jako średni punkt w klastrze.

Powyższa definicja może być dość niejasna, ale staje się konkretna po zawężeniu do określonych dziedzin. Na przykład kupujący online, którzy zachowują się podobnie, mogą kupować podobne rzeczy lub w podobnych sklepach, podczas gdy podobne firmy programistyczne mogą wytwarzać porównywalne oprogramowanie w porównywalnych cenach. Oto wizualizacja skupisk punktów:



Na powyższym rysunku nasze ludzkie mózgi bardzo łatwo dostrzegają różnicę między czterema skupiskami. Mianowicie, że czerwony klaster znajduje się w lewym dolnym rogu wykresu, podczas gdy zielony klaster znajduje się w prawej dolnej części wykresu. Oznacza to, że czerwone punkty danych są do siebie podobne, ale nie podobne do punktów danych w innych klastrach. Możemy również zobaczyć centroidy każdej gromady jako kwadrat w każdym kolorze. Należy zauważyć, że centroid nie jest rzeczywistym punktem danych, ale jest jedynie abstrakcją klastra i reprezentuje środek klastra. Pojęcie podobieństwa jest kluczowe dla definicji klastra, a tym samym do analizy klastrów. Ogólnie rzecz biorąc, większe podobieństwo między punktami prowadzi do lepszego grupowania. W większości przypadków zamieniamy dane w punkty w przestrzeni n-wymiarowej i wykorzystujemy odległość

między tymi punktami jako formę podobieństwa. Środek ciężkości skupienia jest wówczas zwykle średnią każdego wymiaru (kolumny) dla każdego punktu danych w każdym skupieniu. Na przykład środek ciężkości czerwonego skupienia jest wynikiem pobrania średniej wartości każdej kolumny z każdego czerwonego punktu danych. Celem analizy skupień jest lepsze zrozumienie zestawu danych poprzez podzielenie danych na grupy. Klastrowanie zapewnia warstwę abstrakcji z poszczególnych punktów danych. Celem jest wydobycie i wzmocnienie naturalnej struktury danych. Istnieje wiele rodzajów procedur klasyfikacyjnych. W naszej klasie skupimy się na klastrowaniu K-średnich, które jest jednym z najpopularniejszych algorytmów klastrowania. K-średnie to metoda iteracyjna, która dzieli zbiór danych na k klastrów. Działa w czterech krokach:

1. Wybierz k początkowych centroidów (zauważ, że k jest wejściem).

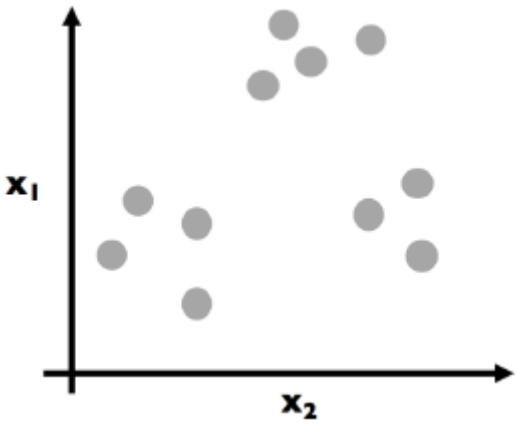
2. Dla każdego punktu przypisz punkt do najbliższego centroidu.

3. Przelicz pozycje środka ciężkości.

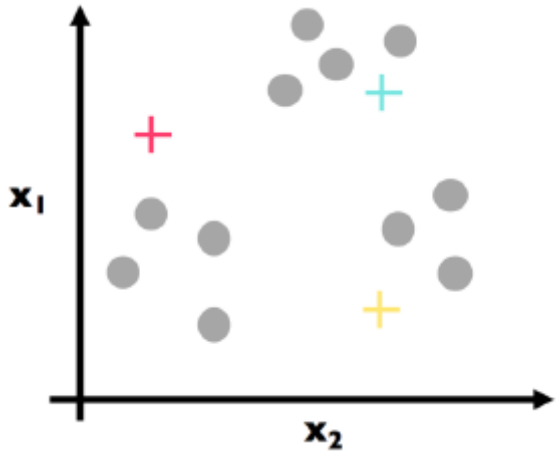
4. Powtarzaj kroki 2-3, aż zostaną spełnione kryteria zatrzymania.

Przykład ilustracyjny – punkty danych

Wyobraź sobie, że w dwuwymiarowej przestrzeni mamy następujące punkty danych:

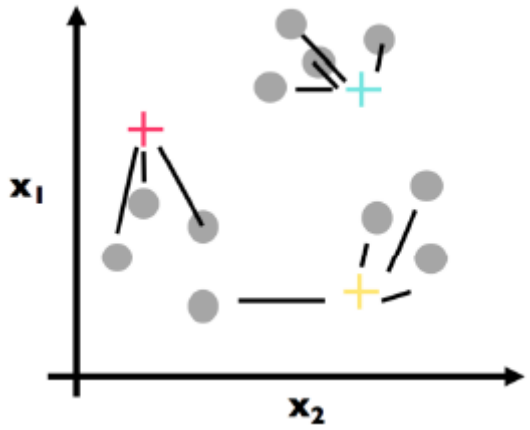


Każda kropka jest pomalowana na szaro, aby założyć brak wcześniejszego grupowania przed zastosowaniem algorytmu K-średnich. Celem jest tutaj ostateczne pokolorowanie każdej kropki i utworzenie grup (klastrów).

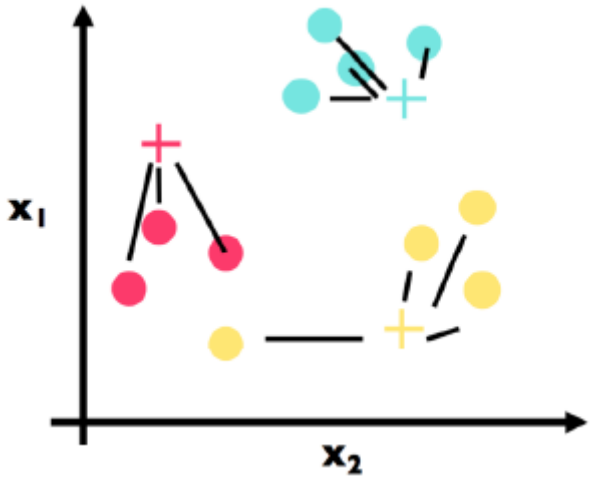


Tutaj zastosowano krok 1. Wybraliśmy (losowo) trzy centroidy (czerwony, niebieski i żółty).

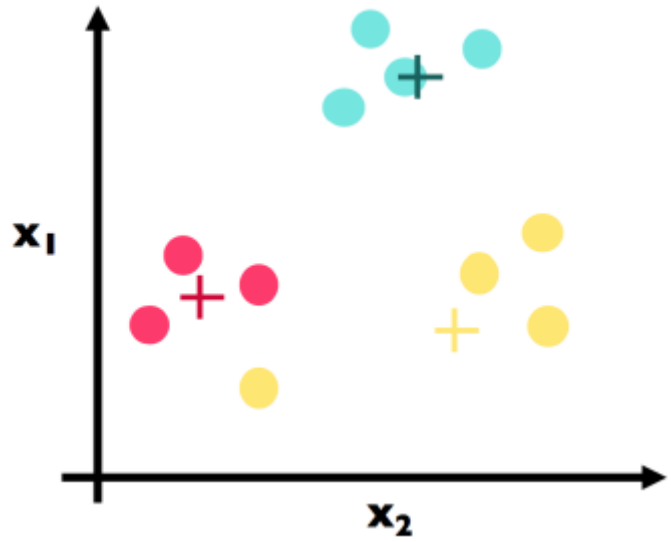
Większość algorytmów K-średnich umieszcza losowe początkowe centroidy, ale istnieją inne metody wstępnego obliczania umieszczania początkowych centroidów. Na razie losowość jest w porządku.



Zastosowano pierwszą część kroku 2. Dla każdego punktu danych znaleźliśmy najbardziej podobny centroid (najbliższy).



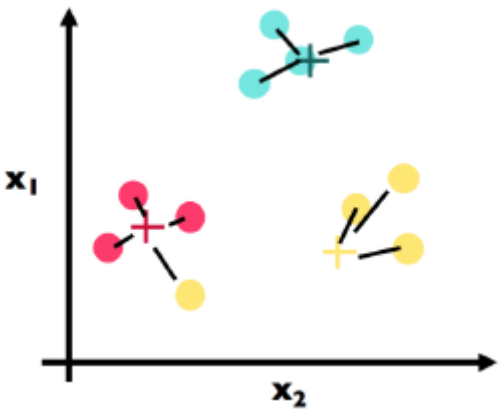
Tutaj zastosowano drugą część kroku 2. Pokolorowaliśmy każdy punkt danych zgodnie z jego najbardziej podobnym centroidem.



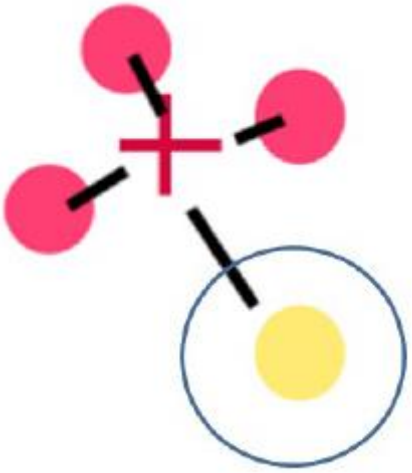
To jest krok 3 i sedno K-średnich. Zauważ, że fizycznie przesunęliśmy centroidy, aby były rzeczywistym środkiem każdej gromady. Dla każdego koloru obliczyliśmy średni punkt i uczyniliśmy go nowym środkiem ciężkości. Załóżmy na przykład, że trzy czerwone punkty danych mają współrzędne (1, 3), (2, 5) i (3, 4). Centrum (czerwony krzyż) zostanie obliczony w następujący sposób:

```
# centroid calculation

import numpy as np

red_point1 = np.array([1, 3])

red_point2 = np.array([2, 5])

red_point3 = np.array([3, 4])

red_center = (red_point1 + red_point2 + red_point3) / 3.

red_center

# array([ 2., 4.])
```

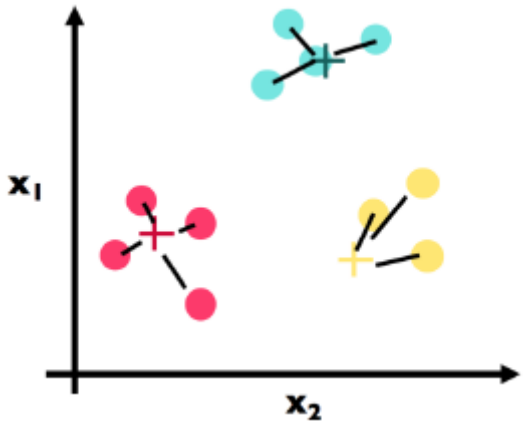
Oznacza to, że punkt (2, 4) byłby współrzędnymi poprzedniego czerwonego krzyża. Żaden z rzeczywistych punktów danych nigdy się nie poruszy. Oni nie mogą. Jedynymi jednostkami, które się poruszają, są centroidy, które NIE są rzeczywistymi punktami danych.



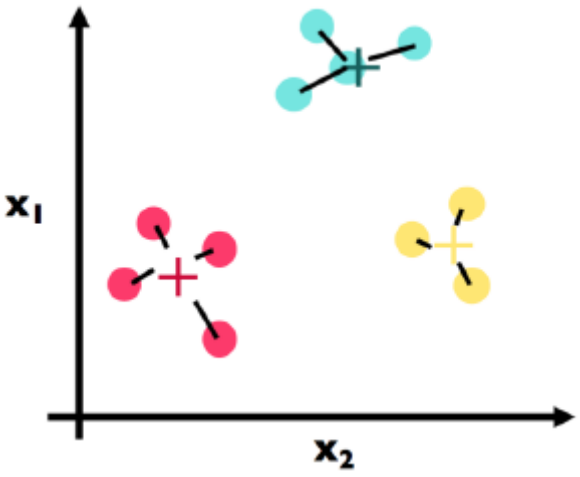
Kontynuujemy nasz algorytm, powtarzając krok 2. Oto pierwsza część, w której znajdujemy najbliższy środek każdego punktu. Zwróć uwagę na dużą zmianę: punkt zakreślony na poniższym rysunku był kiedyś żółtym punktem, ale zmienił się w czerwony punkt skupienia, ponieważ żółta gromada zbliżyła się do swoich żółtych składników.



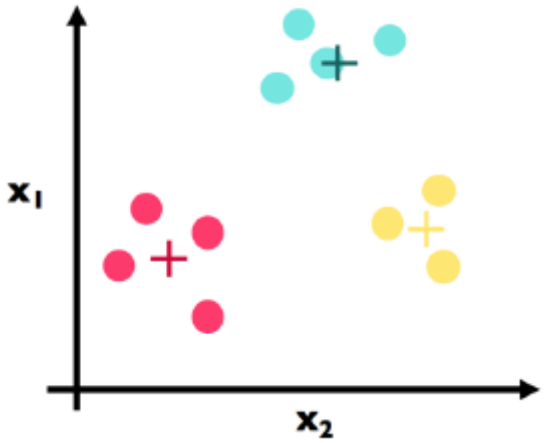
Pomocne może być myślenie o punktach jako o planetach w kosmosie z przyciąganiem grawitacyjnym. Każdy centroid jest przyciągany grawitacjami planet.



Oto druga część kroku 2 ponownie. Każdy punkt przypisaliśmy do koloru najbliższej gromady.



Tutaj ponownie obliczamy centroidy dla każdego klastra (krok 3). Zauważ, że niebieski środek w ogóle się nie poruszył, podczas gdy żółty i czerwony środek poruszył się. Ponieważ osiągnęliśmy kryterium zatrzymania (klastry nie poruszają się, jeśli powtórzymy krok 2 i 3), finalizujemy nasz algorytm i mamy nasze trzy klastry!



Przykład ilustracyjny – piwo!

Dosyć nauki o danych, piwo! Ok ok, uspokój się. Weźmy piwo. Czy wiesz, że istnieje wiele rodzajów piwa? Zastanawiam się, czy moglibyśmy podzielić piwa na różne kategorie w oparciu o różne cechy ilościowe... Spróbujmy!

import the beer dataset

url = '../data/beer.txt'

beer = pd.read_csv(url, sep=' ')

print beer.shape

(20, 5)

beer.head()

	name	calories	sodium	alcohol	cost
0	Budweiser	144	15	4.7	0.43
1	Schlitz	151	19	4.9	0.43
2	Lowenbrau	157	15	0.9	0.48
3	Kronenbourg	170	7	5.2	0.73
4	Heineken	152	11	5.0	0.77

Tutaj mamy 20 piw w pięciu kolumnach: nazwa, kalorie, sód, alkohol i koszt. W klasteringu (podobnie jak prawie we wszystkich modelach uczenia maszynowego) lubimy funkcje ilościowe, więc zignorujemy nazwę piwa w naszym klasteringu:

define X

X = beer.drop('name', axis=1)

Now we will perform K-Means using scikit-learn:

K-means with 3 clusters

from sklearn.cluster import KMeans

km = KMeans(n_clusters=3, random_state=1)

km.fit(X)

n_clusters to nasze k. Jest to wpisana przez nas liczba klastrów. random_state jak zawsze daje powtarzalne wyniki do celów edukacyjnych. Korzystanie z trzech klastrów na razie jest losowe. Nasz algorytm K-średnich uruchomił algorytm na naszych punktach danych i przedstawił trzy klastry:

save the cluster labels and sort by cluster

beer['cluster'] = km.labels_

Możemy spojrzeć na środek każdego skupienia, używając instrukcji groupby i mean:

calculate the mean of each feature for each cluster

beer.groupby('cluster').mean()

	calories	sodium	alcohol	cost
cluster				
0	150.00	17.0	4.521429	0.520714
1	102.75	10.0	4.075000	0.440000
2	70.00	10.5	2.600000	0.420000

Podczas inspekcji przez ludzi widzimy, że klaster 0 ma średnio wyższą zawartość kalorii, zawartość sodu i zawartość alkoholu, a także kosztuje więcej. Można je uznać za piwa cięższe. Klaster 2 ma średnio bardzo niską zawartość alkoholu i bardzo mało kalorii. Są to prawdopodobnie piwa jasne. Klaster 1 jest gdzieś pośrodku. Użyjmy Pythona do stworzenia wykresu, aby zobaczyć to bardziej szczegółowo:

import matplotlib.pyplot as plt

%matplotlib inline

save the DataFrame of cluster centers

centers = beer.groupby('cluster').mean()

create a "colors" array for plotting

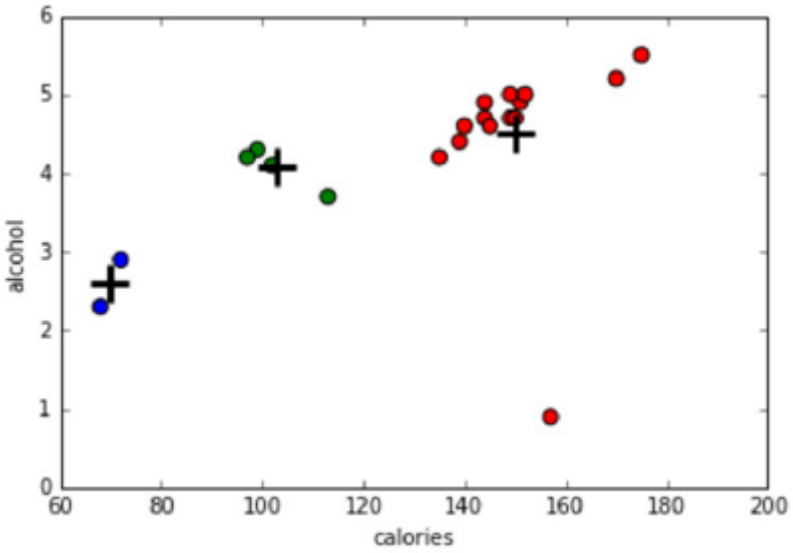
colors = np.array(['red', 'green', 'blue', 'yellow'])

scatter plot of calories versus alcohol, colored by cluster (0=red,

1=green, 2=blue)

```
plt.scatter(beer.calories, beer.alcohol, c=colors[list(beer.cluster)],
s=50)
# cluster centers, marked by "+"
plt.scatter(centers.calories, centers.alcohol, linewidths=3,
marker='+', s=300, c='black')
# add labels
plt.xlabel('calories')
plt.ylabel('alcohol')
```

Dużą częścią uczenia się bez nadzoru jest inspekcja człowieka. Klastrowanie nie ma kontekstu domeny problemu i może nam tylko powiedzieć, jakie klastry znalazło, ale nie może nam powiedzieć, co oznaczają klastry.



Wybór optymalnej liczby do walidacji K i klastra

Duża część grupowania K-średnich polega na znajomości optymalnej liczby skupień. Gdybyśmy znali tę liczbę z wyprzedzeniem, mogłoby to uniemożliwić nawet korzystanie z uczenia nienadzorowanego. Potrzebujemy więc sposobu na ocenę wyników naszej analizy skupień. Problem polega na tym, że ponieważ nie wykonujemy żadnych przewidywań, nie możemy ocenić, jak poprawny jest algorytm w przewidywaniu. Metryki, takie jak dokładność i RMSE, wychodzą prosto z okna.

Współczynnik sylwetki

Współczynnik Silhouette jest powszechnym miernikiem do oceny wydajności klastrowania w sytuacjach, gdy prawdziwe przypisania do klastrów nie są znane. Współczynnik sylwetki jest obliczany dla każdej obserwacji w następujący sposób:

$$SC = \frac{b - a}{\max(a, b)}$$

Przyjrzyjmy się bliżej szczegółowym cechom tej formuły:

• a: Średnia odległość do wszystkich innych punktów w jego klastrze

• b: Średnia odległość do wszystkich innych punktów najbliższego najbliższego klastra

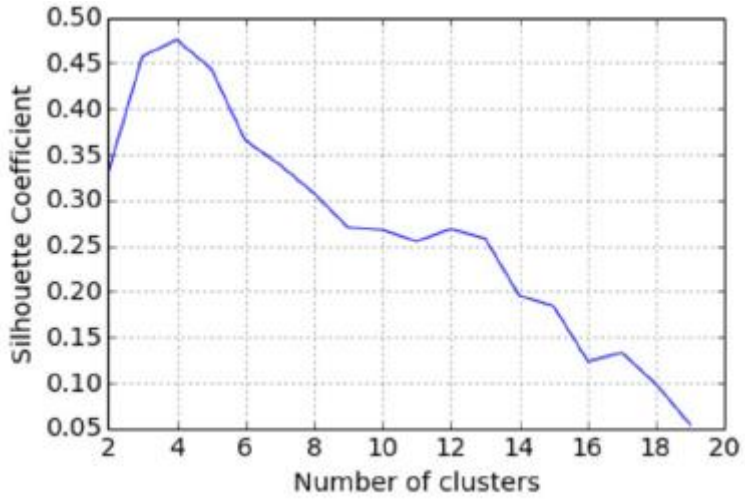
Waha się od -1 (najgorszy) do 1 (najlepszy). Ogólny wynik oblicza się, biorąc średni wynik dla wszystkich obserwacji. Ogólnie rzecz biorąc, preferowany jest współczynnik sylwetki równy 1, natomiast wynik -1 nie jest preferowany:

```
# calculate Silhouette Coefficient for K=3

from sklearn import metrics

metrics.silhouette_score(X, km.labels_)
```

0.4578

Spróbujmy obliczyć współczynnik dla wielu wartości K, aby znaleźć najlepszą wartość:

```
# calculate SC for K=2 through K=19

k_range = range(2, 20)

scores = []

for k in k_range:

km = KMeans(n_clusters=k, random_state=1)

km.fit(X_scaled)

scores.append(metrics.silhouette_score(X, km.labels_))

# plot the results

plt.plot(k_range, scores)

plt.xlabel('Number of clusters')

plt.ylabel('Silhouette Coefficient')

plt.grid(True)
```



Wygląda więc na to, że nasza optymalna liczba klastrów piwa to 4! Oznacza to, że nasz algorytm k-średnich ustalił, że istnieją cztery różne rodzaje piwa. K-średnie to popularny algorytm ze względu na swoją wydajność obliczeniową oraz prosty i intuicyjny charakter. K-średnie są jednak silnie zależne od skali i nie są odpowiednie dla danych o bardzo różnych kształtach i gęstościach. Istnieją sposoby na rozwiązanie tego problemu poprzez skalowanie danych przy użyciu standardowego skalaru scikit-learn:

```
# center and scale the data

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)

# K-means with 3 clusters on scaled data

km = KMeans(n_clusters=3, random_state=1)

km.fit(X_scaled)
```

Łatwo!

Przyjrzyjmy się teraz trzeciemu powodowi używania metod nienadzorowanych, który należy do trzeciej opcji w naszych powodach używania metod nienadzorowanych, czyli ekstrakcji cech.

Ekstrakcja cech i analiza głównych składowych

Czasami mamy przytłaczającą liczbę kolumn i prawdopodobnie niewystarczającą liczbę wierszy, aby obsłużyć dużą liczbę kolumn. Świetnym tego przykładem jest sytuacja, w której przyglądaliśmy się przykładowi wysyłania gotówki teraz w naszym przykładzie Naïve Bayes. Mieliśmy dosłownie 0 wystąpień tekstów z tym dokładnym wyrażeniem, więc zamiast tego zwróciliśmy się do naiwnego założenia, które pozwoliło nam ekstrapolować prawdopodobieństwo dla obu naszych kategorii. Powodem, dla którego mieliśmy ten problem, jest coś, co nazywa się przekleństwem wymiarowości. Klątwa wymiarowości zasadniczo mówi, że gdy wprowadzamy i rozważamy nowe kolumny funkcji, potrzebujemy niemal wykładniczo więcej wierszy (punktów danych), aby wypełnić puste przestrzenie, które tworzymy. Rozważmy przykład, w którym próbujemy użyć modelu uczenia się, który wykorzystuje odległość między punktami na korpusie tekstu, który ma 4086 fragmentów tekstu, i że całość została skrócona do wektorów. Załóżmy, że te teksty pomiędzy nimi mają 18 884 słów:

```
X.shape
```

```
(4086, 18884)
```

Teraz zróbmy eksperyment. Najpierw rozważę jedno słowo jako jedyny wymiar naszego tekstu. Następnie policzę, ile fragmentów tekstu znajduje się w odległości 1 jednostki od siebie. Na przykład, jeśli dwa zdania zawierają to słowo, będą oddalone o 0 jednostek i podobnie, jeśli żadne z nich nie zawierają tego słowa, będą oddalone od siebie o 0 jednostek:

```
d = 1

# Let's look for points within 1 unit of one another

X_first_word = X[:,:1]

# Only looking at the first column, but ALL of the rows
```

```
from sklearn.neighbors import NearestNeighbors

# this module will calculate for us distances between each point

neigh = NearestNeighbors(n_neighbors=4086)

neigh.fit(X_first_word)

# tell the module to calculate each distance between each point

A = neigh.kneighbors_graph(X_first_word, mode='distance').todense()

# This matrix holds all distances (over 16 million of them)

num_points_within_d = (A < d).sum()

# Count the number of pairs of points within 1 unit of distance

num_points_within_d

16258504
```

Tak więc 16,2 miliona par tekstów znajduje się w jednej jednostce odległości. Spróbujmy teraz ponownie z dwoma pierwszymi słowami:

```
X_first_two_words = X[:,:2]

neigh = NearestNeighbors(n_neighbors=4086)

neigh.fit(X_first_two_words)

A = neigh.kneighbors_graph(X_first_two_words, mode='distance').

todense()

num_points_within_d = (A < d).sum()

num_points_within_d

16161970
```

Świetnie! Dodając tę nową kolumnę, straciliśmy około 100 000 par punktów, które znajdowały się w obrębie jednej jednostki odległości. Dzieje się tak, ponieważ dodajemy przestrzeń między nimi dla każdego dodawanego wymiaru. Zróbmy ten test o krok dalej i obliczmy tę liczbę dla pierwszych 100 słów, a następnie wykreślmy wyniki:

```
d = 1

# Scan for points within one unit

num_columns = range(1, 100)

# Looking at the first 100 columns

points = []

# We will be collecting the number of points within 1 unit for a graph

neigh = NearestNeighbors(n_neighbors=X.shape[0])
```

for subset in num_columns:

X_subset = X[:,:subset]

look at the first column, then first two columns, then first three

columns, etc

neigh.fit(X_subset)

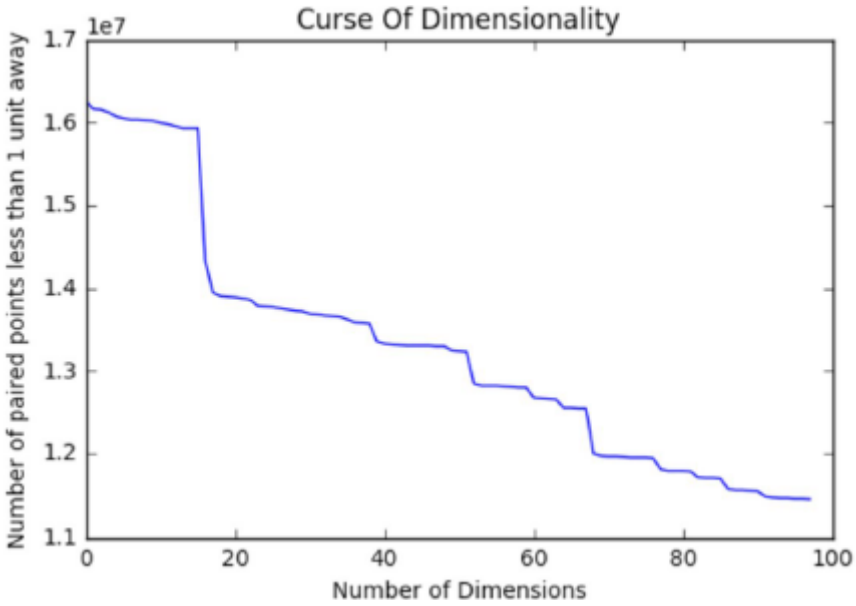
A = neigh.kneighbors_graph(X_subset, mode='distance').todense()

num_points_within_d = (A < d).sum()

calculate the number of points within 1 unit

points.append(num_points_within_d)

Teraz wykreślmy liczbę punktów w 1 jednostce w zależności od liczby wymiarów, które przyjrzeliśmy się:



Widzimy wyraźnie, że liczba punktów w jednej jednostce względem siebie drastycznie spada, gdy wprowadzamy coraz więcej kolumn. A to tylko pierwsze 100 kolumn! Zobaczmy, ile punktów znajduje się w jednej jednostce, zanim weźmiemy pod uwagę wszystkie 18 000+ słów:

neigh = NearestNeighbors(n_neighbors=4086)

neigh.fit(X)

A = neigh.kneighbors_graph(X, mode='distance').todense()

num_points_within_d = (A < d).sum()

num_points_within_d

4090

W końcu tylko 4000 zdań znajduje się w jednej jednostce. Cała ta przestrzeń, którą dodajemy, biorąc pod uwagę nowe kolumny, utrudnia skończoną liczbę punktów, które musimy szczęśliwie pozostawać

w swoim zasięgu. Musielibyśmy dodać więcej punktów, aby wypełnić tę lukę. I dlatego, moi przyjaciele, powinniśmy rozważyć zastosowanie redukcji wymiarów. Klątwa wymiarowości zostaje rozwiązana przez dodanie większej liczby punktów danych (co nie zawsze jest możliwe) lub wdrożenie redukcji wymiarów. Redukcja wymiarów to po prostu czynność polegająca na zmniejszeniu liczby kolumn w naszym zestawie danych, a nie liczby wierszy. Istnieją dwa sposoby realizacji redukcji wymiarów:

• Wybór funkcji: jest to czynność polegająca na podstawianiu funkcji naszych kolumn i używaniu tylko najlepszych funkcji

• Wyodrębnianie cech: jest to czynność matematycznego przekształcenia naszego zestawu cech w nowy wyodrębniony układ współrzędnych

Jesteśmy zaznajomieni z wyborem funkcji, ponieważ proces mówienia, że Emabrked_Q nie pomaga w moim drzewie decyzyjnym; pozbądźmy się go i zobaczmy, jak działa. Dosłownie, kiedy my (lub maszyna) podejmujemy decyzję o zignorowaniu pewnych kolumn.

Wyodrębnianie funkcji jest nieco trudniejsze…

W ekstrakcji fukcji używamy zwykle dość skomplikowanych wzorów matematycznych, aby uzyskać nowe superkolumny, które są zwykle lepsze niż jakakolwiek pojedyncza kolumna oryginalna. Nasz podstawowy model to analiza głównych składowych (PCA).

PCA wyodrębni określoną liczbę superkolumn, aby reprezentować nasze oryginalne dane za pomocą znacznie mniejszej liczby kolumn. Weźmy konkretny przykład. Wcześniej wspomniałem o tekście z 4086 wierszami i ponad 18 000 kolumn. Ten zbiór danych to w rzeczywistości zestaw recenzji online Yelp:

```
url = '../data/yelp.csv'

yelp = pd.read_csv(url, encoding='unicode-escape')

# create a new DataFrame that only contains the 5-star and 1-star

reviews

yelp_best_worst = yelp[(yelp.stars==5) | (yelp.stars==1)]

# define X and y

X = yelp_best_worst.text

y = yelp_best_worst.stars == 5
```

Naszym celem jest przewidzenie, czy dana osoba wystawiła ocenę 5, czy 1 gwiazdkę na podstawie słów, których użyła w recenzji. Ustalmy linię bazową za pomocą regresji logistycznej i zobaczmy, jak dobrze możemy przewidzieć tę kategorię binarną:

```
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression()

X_train, X_test, y_train, y_test = train_test_split(X, y, random_

state=100)

# Make our training and testing sets
```

```
vect = CountVectorizer(stop_words='english')

# Count the number of words but remove stop words like a, an, the,

you, etc

X_train_dtm = vect.fit_transform(X_train)

X_test_dtm = vect.transform(X_test)

# transform our text into document term matrices

lr.fit(X_train_dtm, y_train)

# fit to our training set

lr.score(X_test_dtm, y_test)

# score on our testing set

0.91193737
```

Zatem wykorzystując wszystkie słowa z naszego korpusu, nasz model wydaje się mieć ponad 91% dokładności. Nie jest zły! Spróbujmy użyć tylko 100 najczęściej używanych słów:

```
vect = CountVectorizer(stop_words='english', max_features=100)

# Only use the 100 most used words

X_train_dtm = vect.fit_transform(X_train)

X_test_dtm = vect.transform(X_test)

print X_test_dtm.shape # (1022, 100)

lr.fit(X_train_dtm, y_train)

lr.score(X_test_dtm, y_test)

0.8816
```

Zwróć uwagę, że nasze macierze treningowe i testowe mają 100 kolumn. To dlatego, że powiedziałem naszemu wektoryzatorowi, aby patrzył tylko na 100 pierwszych słów. Zobacz także, że nasza wydajność została uszkodzona i teraz spadła do 88% celności. Ma to sens, ponieważ ignorujemy ponad 4700 słów w naszym korpusie. Teraz przyjmijmy inne podejście. Zaimportujmy moduł PCA i powiedzmy mu, aby utworzył 100 NOWYCH superkolumn i zobaczmy, jak to działa:

```
from sklearn import decomposition

# We will be creating 100 super columns

vect = CountVectorizer(stop_words='english')

# Don't ignore any words

pca = decomposition.PCA(n_components=100)

# instantate a pca object

X_train_dtm = vect.fit_transform(X_train).todense()
```

```
# A dense matrix is required to pass into PCA, does not affect the

overall message

X_train_dtm = pca.fit_transform(X_train_dtm)

X_test_dtm = vect.transform(X_test).todense()

X_test_dtm = pca.transform(X_test_dtm)

print X_test_dtm.shape # (1022, 100)

lr.fit(X_train_dtm, y_train)

lr.score(X_test_dtm, y_test)

.89628
```

Nie tylko nasze macierze nadal mają 100 kolumn, ale te kolumny nie są już słowami w naszym korpusie. Są to złożone przekształcenia kolumn i 100 nowych kolumn. Pamiętaj też, że użycie 100 z tych nowych kolumn daje nam lepszą prognozę wydajności niż używanie 100 najlepszych słów! Wyodrębnianie cech to świetny sposób na wykorzystanie formuł matematycznych do wyodrębnienia zupełnie nowych kolumn, które generalnie mają lepsze wyniki niż tylko uprzednie wybranie najlepszych. Ale jak wizualizujemy te nowe super kolumny? Cóż, nie mogę wymyślić lepszego sposobu niż przyjrzenie się przykładowi za pomocą analizy obrazu. W szczególności zróbmy oprogramowanie do rozpoznawania twarzy. OK? OK. Zacznijmy od zaimportowania niektórych twarzy otrzymanych od scikit-learn:

```
from sklearn.datasets import fetch_lfw_people

lfw_people = fetch_lfw_people(min_faces_per_person=70, resize=0.4)

# introspect the images arrays to find the shapes (for plotting)

n_samples, h, w = lfw_people.images.shape

# for machine learning we use the 2 data directly (as relative pixel

# positions info is ignored by this model)

X = lfw_people.data

y = lfw_people.target

n_features = X.shape[1]

X.shape

(1288, 1850)
```

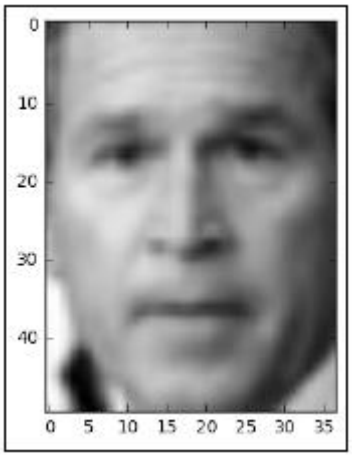
Zebraliśmy 1288 zdjęć twarzy ludzi, a każdy z nich ma 1850 cech (pikseli), które identyfikują tę osobę. Na przykład:

```
plt.imshow(X[0].reshape((h, w)), cmap=plt.cm.gray)

lfw_people.target_names[y[0]]

'Hugo Chavez'
```

plt.imshow(X[100].reshape((h, w)), cmap=plt.cm.gray)

lfw_people.target_names[y[100]]

'George W Bush'



Świetnie. Aby rzucić okiem na typ zestawu danych, na który patrzymy, przyjrzyjmy się kilku ogólnym wskaźnikom:

the label to predict is the id of the person

target_names = lfw_people.target_names

n_classes = target_names.shape[0]

print("Total dataset size:")

print("n_samples: %d" % n_samples)

print("n_features: %d" % n_features)

print("n_classes: %d" % n_classes)

Total dataset size:

n_samples: 1288

n_features: 1850

n_classes: 7

Mamy więc 1288 obrazów, 1850 funkcji i 7 klas (osób) do wyboru. Naszym celem jest stworzenie klasyfikatora, który przypisze twarzy danej osoby nazwę na podstawie przekazanych nam 1850 pikseli.

Przyjrzyjmy się linii bazowej i zobaczmy, jak regresja logistyczna działa na naszych danych bez robienia czegokolwiek:

```
from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score

from time import time # for timing our work

X_train, X_test, y_train, y_test = train_test_split(

X, y, test_size=0.25, random_state=1)

# get our training and test set

t0 = time() # get the time now

logreg = LogisticRegression()

logreg.fit(X_train, y_train)

# Predicting people's names on the test set

y_pred = logreg.predict(X_test)

print accuracy_score(y_pred, y_test), "Accuracy"

print (time() - t0), "seconds"
```

0.810559006211 Accuracy

6.31762504578 seconds

Tak więc w ciągu 6,3 sekundy byliśmy w stanie uzyskać 81% na naszym zestawie testowym. Nieźle…

Teraz spróbujmy tego z naszymi super twarzami:

```
# split into a training and testing set

from sklearn.cross_validation import train_test_split

# Compute a PCA (eigenfaces) on the face dataset (treated as unlabeled

# dataset): unsupervised feature extraction / dimensionality reduction

n_components = 75

# Extracting the top %d eigenfaces from %d faces

% (n_components, X_train.shape[0]))

pca = decomposition.PCA(n_components=n_components, whiten=True).fit(X_

train)
```

```
# This whiten parameter speeds up the computation of our extracted columns

# Projecting the input data on the eigenfaces orthonormal basis

X_train_pca = pca.transform(X_train)

X_test_pca = pca.transform(X_test)
```

The preceding code is collecting 75 extracted columns from our 1,850 unprocessed columns. These are our super faces. Now let's plug in our newly extracted columns into our logistic regression and compare:

```
t0 = time()

# Predicting people's names on the test set WITH PCA

logreg.fit(X_train_pca, y_train)

y_pred = logreg.predict(X_test_pca)

print accuracy_score(y_pred, y_test), "Accuracy"

print (time() - t0), "seconds"
```

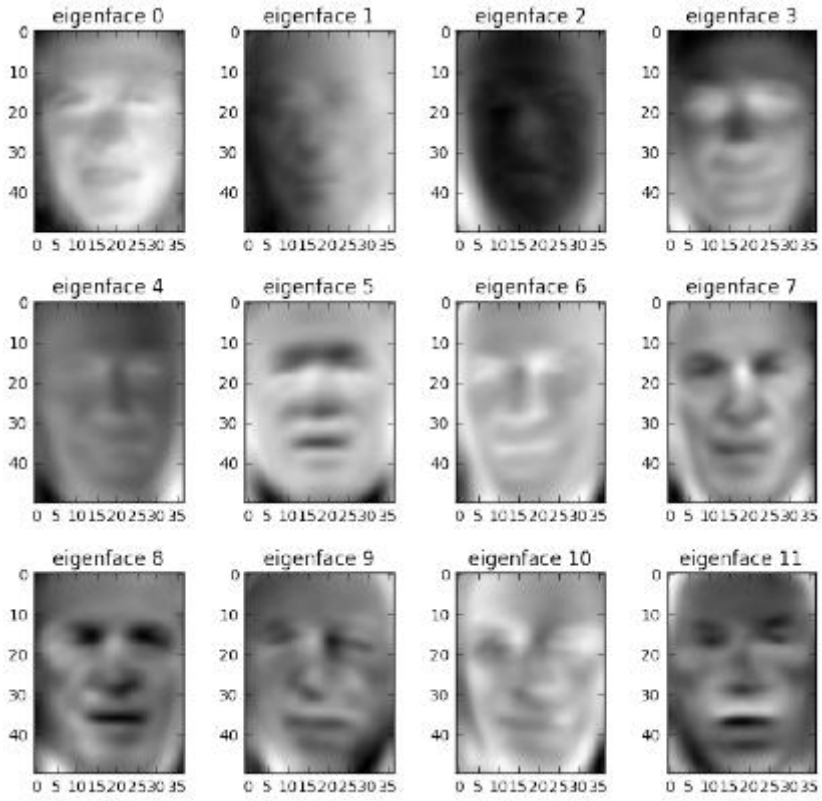
0.82298136646 Accuracy

0.194181919098 seconds

Wow! Całe to obliczenie było nie tylko około 30 razy szybsze niż nieprzetworzone obrazy, ale także lepsza wydajność predykcyjna! To pokazuje nam, że PCA i ekstrakcja funkcji w ogóle mogą nam pomóc podczas wykonywania uczenia maszynowego na złożonych zestawach danych z wieloma kolumnami. Wyszukując te wzorce w zestawie danych i wyodrębniając nowe kolumny funkcji, możemy przyspieszyć i ulepszyć nasze algorytmy uczenia się. Przyjrzyjmy się jeszcze jednej interesującej rzeczy… Wspomniałem wcześniej, że jednym z celów tego przykładu było zbadanie i zwizualizowanie naszych własnych twarzy, jak się je nazywa. Nasze super kolumny. Nie zawiodę. Napiszmy kod, który pokaże nam nasze superkolumny tak, jak wyglądałyby dla nas, ludzi:

```
def plot_gallery(images, titles, n_row=3, n_col=4):

"""Helper function to plot a gallery of portraits"""

plt.figure(figsize=(1.8 * n_col, 2.4 * n_row))

plt.subplots_adjust(bottom=0, left=.01, right=.99, top=.90,

hspace=.35)

for i in range(n_row * n_col):

plt.subplot(n_row, n_col, i + 1)

plt.imshow(images[i], cmap=plt.cm.gray)

plt.title(titles[i], size=12)

# plot the gallery of the most significative eigenfaces

eigenfaces = pca.components_.reshape((n_components, h, w))
```

```
eigenface_titles = ["eigenface %d" % i for i in range(eigenfaces.
shape[0])]
plot_gallery(eigenfaces, eigenface_titles)
plt.show()
```



Wow. Niesamowita, a jednocześnie piękna reprezentacja tego, co według danych jest najważniejszymi cechami twarzy. Gdy przechodzimy od lewego górnego rogu (pierwsza super kolumna) do dołu, dość łatwo jest zobaczyć, co obraz próbuje nam powiedzieć. Pierwsza super kolumna wygląda jak bardzo ogólna struktura twarzy z oczami, nosem i ustami. To prawie mówi „reprezentuję podstawowe cechy twarzy, które muszą mieć wszystkie twarze”. Wydaje się, że nasza druga super kolumna po prawej stronie mówi nam o cieniach na obrazie. Następna może nam powiedzieć, że odcień skóry odgrywa rolę w wykrywaniu, kto to jest, co może być powodem, dla którego trzecia twarz jest znacznie ciemniejsza niż dwie pierwsze. Korzystanie z wyodrębniania funkcji nienadzorowanych metod uczenia się, takich jak PCA, może dać nam bardzo dogłębny wgląd w nasze dane i ujawnić nam, jakie dane są uważane za najważniejsze funkcje, a nie tylko to, w co sądzimy. Wyodrębnianie funkcji to świetne narzędzie do przetwarzania wstępnego, które może przyspieszyć nasze przyszłe metody uczenia się, zwiększyć ich możliwości i dać nam lepszy wgląd w to, jak dane powinny być postrzegane. Podsumowując tę sekcję, wymienimy zalety i wady.

Zalety korzystania z ekstrakcji funkcji:

- Nasze modele stają się znacznie szybsze
- Nasza wydajność predykcyjna może stać się lepsza
- Może dać nam wgląd w wyodrębnione cechy (eigenfaces)

Wady korzystania z ekstrakcji funkcji:

• Tracimy interpretowalność naszych cech, ponieważ są to nowe kolumny wyprowadzone matematycznie, a nie nasze stare

• Możemy stracić wydajność predykcyjną, ponieważ tracimy informacje, ponieważ wyodrębniamy mniej kolumn

Podsumowanie

Pomiędzy drzewami decyzyjnymi, klasyfikacją Naïve Bayesa, wyodrębnianiem cech i grupowaniem K-średnich widzieliśmy, że uczenie maszynowe wykracza daleko poza prostotę regresji liniowej i logistycznej i może rozwiązywać wiele rodzajów skomplikowanych problemów. Widzieliśmy również przykłady zarówno nadzorowanego, jak i nienadzorowanego uczenia się, dzięki czemu zapoznaliśmy się z wieloma rodzajami problemów związanych z nauką o danych. W następnym rozdziale przyjrzymy się jeszcze bardziej skomplikowanym algorytmom uczenia, w tym sztucznym sieciom neuronowym i technikom asemblerowym. Zobaczymy również i zrozumiemy bardziej skomplikowane koncepcje w nauce o danych, w tym kompromis między stronniczością a wariancją, a także koncepcję overfittingu.

Poza podstawami

W tej Części omówimy niektóre z bardziej skomplikowanych części nauki o danych, które mogą niektórych zniechęcić. Powodem tego jest to, że data science to nie tylko zabawa i uczenie maszynowe. Czasami musimy przedyskutować i rozważyć paradygmaty teoretyczne i matematyczne oraz ocenić nasze procedury. W tym rozdziale omówimy wiele z tych procedur krok po kroku, tak abyśmy całkowicie i całkowicie zrozumieli tematy. Będziemy omawiać takie tematy jak:

• Krzyżowa walidacja

• Kompromis wariancji odchylenia

• Overfit i underfitting

• Techniki zespołowe

• Losowe lasy

• Sieci neuronowe

To tylko niektóre z poruszanych tematów. W żadnym momencie nie chcę, żebyś był zdezorientowany. Każdą procedurę/algorytm postaram się opisać z największą starannością oraz wieloma przykładami i wizualizacjami.

Kompromis wariancji odchylenia

W poprzednich Częściach pokrótce omówiliśmy koncepcję stronniczości i wariancji. Omawiając te dwie koncepcje, mówimy ogólnie o algorytmach uczenia nadzorowanego. Mówimy konkretnie o wyprowadzaniu błędów z naszych modeli predykcyjnych z powodu błędu systematycznego i wariancji.

Błąd spowodowany stronniczością

Mówiąc o błędach spowodowanych biasem, mówimy o różnicy między oczekiwanym przewidywaniem naszego modelu a rzeczywistą (poprawną) wartością, którą staramy się przewidzieć. W efekcie odchylenie mierzy, jak daleko, ogólnie rzecz biorąc, przewidywania naszego modelu odbiegają od prawidłowej wartości. Pomyśl o odchyleniu jako o różnicy między przewidywaną a rzeczywistą wartością. Załóżmy na przykład, że nasz model, przedstawiony jako F(x), przewiduje wartość 29 w następujący sposób:

$$F(29) = 88$$

Tutaj wartość 29 powinna była być przewidziana na 79, wtedy:

$$Bias(29) = 88 - 79 = 9$$

Jeśli model uczenia maszynowego jest bardzo dokładny w swoich przewidywaniach (regresja lub klasyfikacja), jest uważany za model o niskim odchyleniu, natomiast jeśli model jest najczęściej błędny, jest uważany za model o wysokim odchyleniu. Odchylenie jest miarą oceny modeli na podstawie dokładności lub tego, jak poprawny jest średnio poprawny model.

Błąd z powodu wariancji

Błąd spowodowany wariancją zależy od zmienności prognozy modelu dla danego punktu danych. Wyobraź sobie, że w kółko powtarzasz proces budowania modelu uczenia maszynowego. Wariancję

mierzy się, obserwując, jak bardzo prognozy dla ustalonego punktu różnią się między różnymi wynikami końcowymi. Aby wyobrazić sobie wariancję w swojej głowie, pomyśl o populacji punktów danych. Gdybyś miał pobierać losowe próbki w kółko, jak drastycznie model uczenia maszynowego zmienia się lub dopasowuje za każdym razem inaczej. Jeżeli model nie zmienia się znacząco między próbami, model zostanie uznany za model o niskiej wariancji. Jeśli Twój model zmienia się drastycznie między próbkami, wówczas model ten zostanie uznany za model o dużej wariancji. Wariancja jest świetną miarą oceny naszego modelu na podstawie uogólniania. Jeśli nasz model ma niską wariancję, możemy oczekiwać, że będzie zachowywał się w określony sposób, gdy zostanie wprowadzony na wolność i będzie przewidywał wartości bez nadzoru człowieka. Naszym celem jest optymalizacja zarówno stronniczości, jak i wariancji. Idealnie, szukamy możliwie najniższej wariancji i odchylenia. Uważam, że najlepiej to wyjaśnić na przykładzie.

Przykład - porównanie masy ciała i mózgu ssaków

Wyobraź sobie, że rozważamy związek między masą mózgu ssaków a odpowiadającą im masą ciała. Hipoteza może brzmieć, że istnieje między nimi pozytywna korelacja (w miarę wzrostu jednego, drugie rośnie). Ale jak silny jest ten związek? Czy to nawet liniowe? Być może wraz ze wzrostem masy mózgu następuje logarytmiczny lub kwadratowy wzrost masy ciała. Użyjmy Pythona do eksploracji, jak pokazano:

```
# # Exploring the Bias-Variance Tradeoff

import pandas as pd

import numpy as np

import seaborn as sns

%matplotlib inline
```

Do wizualizacji punktów danych jako wykresu punktowego, a także do tworzenia wykresów liniowych (i wyższych wielomianowych) modeli regresji użyję modułu zwanego seaborn:

```
# ## Brain and body weight

'''

This is a [dataset]) of the average

weight of the body and the brain for

62 mammal species. Let's read it into pandas and

take a quick look:

'''

df = pd.read_table('http://people.sc.fsu.edu/~jburkardt/

datasets/regression/x01.txt', sep='\s+', skiprows=33,

names=['id','brain','body'], index_col='id')

df.head()
```

	brain	body
id		
1	3.385	44.5
2	0.480	15.5
3	1.350	8.1
4	465.000	423.0
5	36.330	119.5

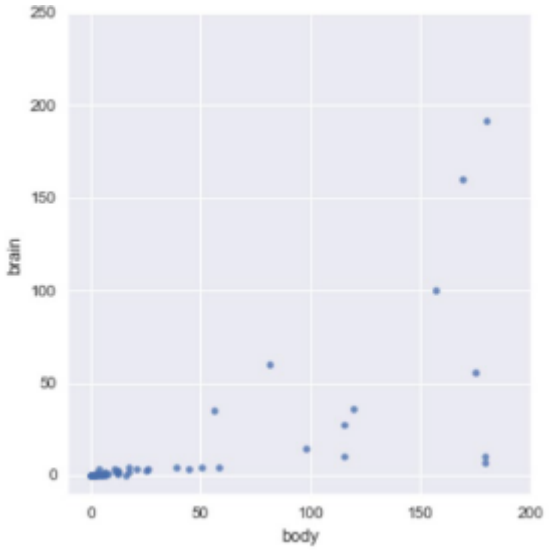
Weźmiemy mały podzbiór próbek, aby zaostrzyć wizualną reprezentację błędu systematycznego i wariancji w następujący sposób:

```
# We're going to focus on a smaller subset in which the body weight is less than 200:

df = df[df.body < 200]

df.shape
```

(51, 2)

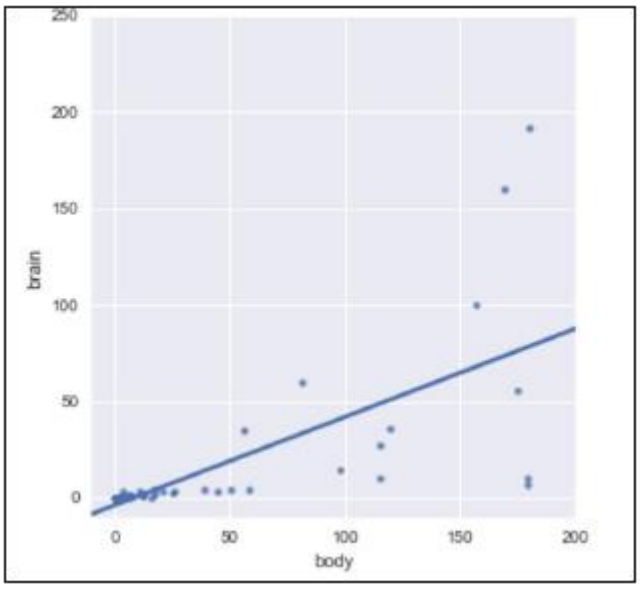
Zamierzamy udawać, że istnieje tylko 51 gatunków ssaków. Innymi słowy, udajemy, że jest to cały zestaw danych dotyczących masy mózgu i ciała każdego znanego gatunku ssaka.

```
# Let's create a scatterplot

sns.lmplot(x='body', y='brain', data=df, ci=None, fit_reg=False)

sns.plt.xlim(-10, 200)

sns.plt.ylim(-10, 250)
```



Wydaje się, że istnieje związek między mózgiem a masą ciała ssaków. Do tej pory możemy założyć, że jest to korelacja dodatnia. Teraz wrzućmy do miksu regresję liniową. Użyjmy seaborna do stworzenia i wykreślenia regresji wielomianowej (liniowej) pierwszego stopnia.

```
sns.lmplot(x='body', y='brain', data=df, ci=None)

sns.plt.xlim(-10, 200)

sns.plt.ylim(-10, 250)
```



Załóżmy teraz, że odkryto nowy gatunek ssaka. Mierzymy masę ciała każdego członka tego gatunku, którego możemy znaleźć i obliczamy średnią masę ciała wynoszącą 100. Chcemy przewidzieć średnią masę mózgu tego gatunku (zamiast mierzyć ją bezpośrednio). Używając tej linii, możemy przewidzieć wagę mózgu około 45. Coś, co można zauważyć, to to, że ta linia nie jest tak blisko punktów danych na wykresie, więc może nie jest to najlepszy model do użycia! Możesz argumentować, że błąd jest zbyt wysoki. I zgodziłbym się! Modele regresji liniowej mają zwykle wysoką stronniczość, ale regresja liniowa ma też coś w zanadrzu – ma bardzo niską wariancję. Ale co to tak naprawdę oznacza? Załóżmy, że bierzemy całą naszą populację ssaków i losowo dzielimy ją na dwie próbki w następujący sposób:

```
# set a random seed for reproducibility

np.random.seed(12345)

# randomly assign every row to either sample 1 or sample 2

df['sample'] = np.random.randint(1, 3, len(df))

df.head()
```

Dołącz nową kolumnę próbki:

	brain	body	sample
id			
1	3.385	44.5	1
2	0.480	15.5	2
3	1.350	8.1	2
5	36.330	119.5	2
6	27.660	115.0	1

Compare the two samples, they are fairly different!

df.groupby('sample')[['brain', 'body']].mean()

	brain	body
sample		
1	18.113778	52.068889
2	13.323364	34.669091

Możemy teraz powiedzieć seabornowi, aby utworzył dwa wykresy, w których lewy wykres wykorzystuje tylko dane z próbki 1, a prawy wykres wykorzystuje tylko dane z próbki 2:

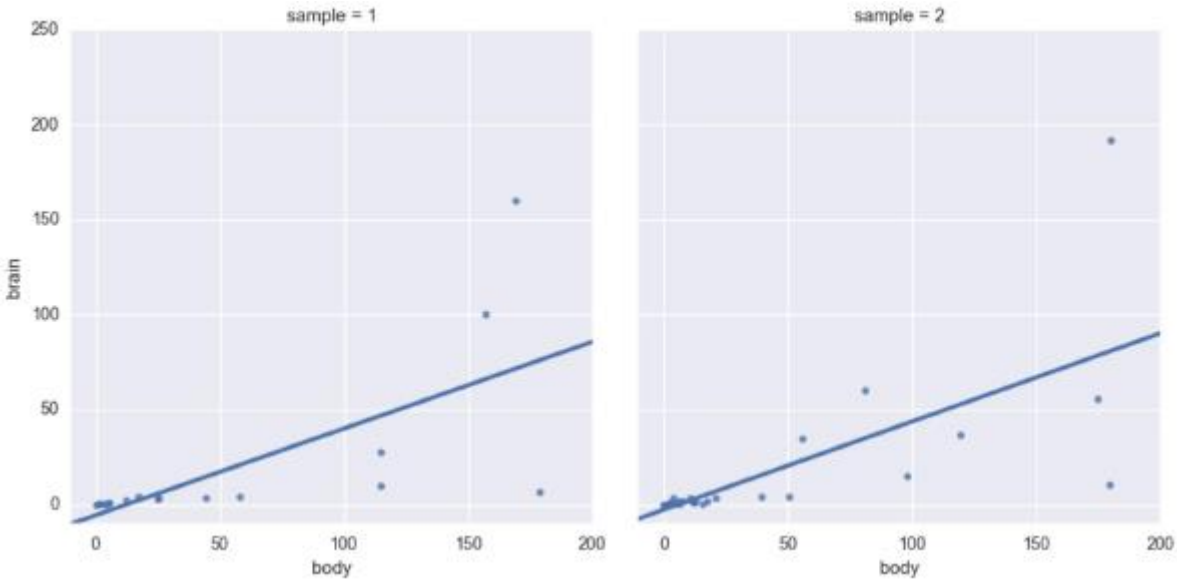
col='sample' subsets the data by sample and creates two

separate plots

sns.lmplot(x='body', y='brain', data=df, ci=None, col='sample')

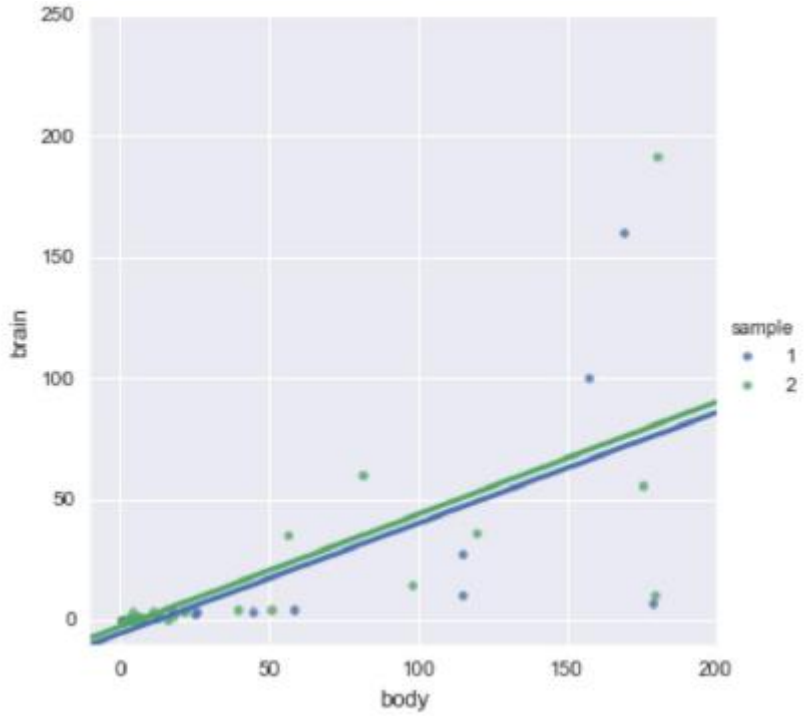
sns.plt.xlim(-10, 200)

sns.plt.ylim(-10, 250)



Ledwo wyglądają inaczej, prawda? Jeśli przyjrzysz się uważnie, zauważysz, że żaden punkt danych nie jest współdzielony między próbkami, a mimo to linia wygląda prawie identycznie. Aby dokładniej pokazać ten punkt, umieśćmy obie linie najlepszego dopasowania na tym samym wykresie i użyjmy kolorów do oddzielenia próbek, jak pokazano na ilustracji:

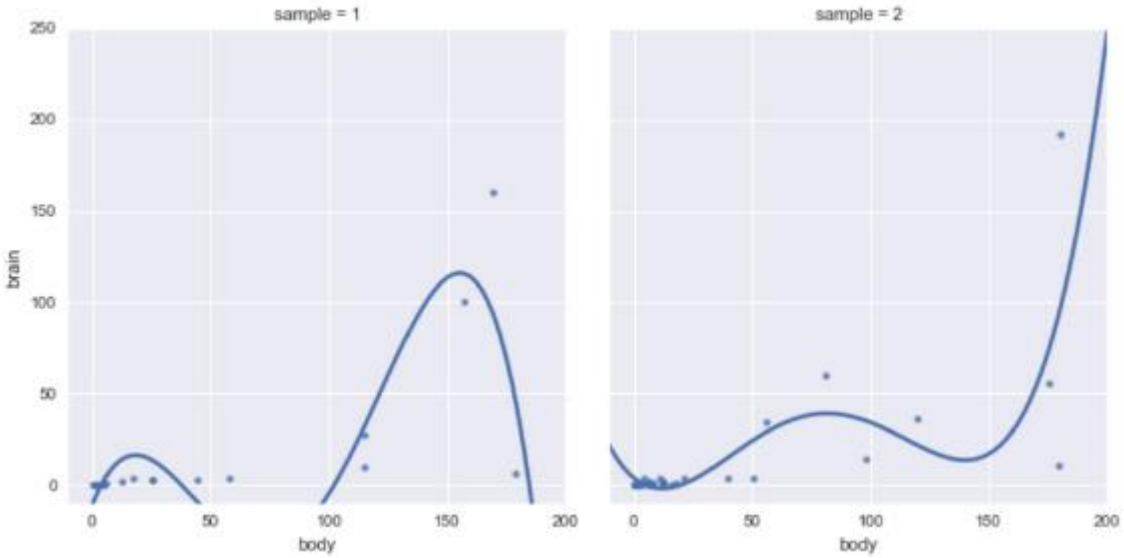
```
# hue='sample' subsets the data by sample and creates a

# single plot

sns.lmplot(x='body', y='brain', data=df, ci=None, hue='sample')

sns.plt.xlim(-10, 200)

sns.plt.ylim(-10, 250)
```



Linia wygląda dość podobnie między tymi dwoma wykresami, mimo że wykorzystano oddzielne próbki danych. W obu przypadkach przewidywalibyśmy wagę mózgu około 45. Fakt, że chociaż regresja liniowa została przypisana do całkowicie odrębnych zestawów danych pobranych z tej samej populacji, dała bardzo podobną linię, co sugeruje, że model ma niską wariancję . Co by było, gdybyśmy zwiększyli złożoność naszego modelu i pozwolili mu dowiedzieć się więcej? Zamiast dopasowywać linię, niech seaborn dopasuje wielomian czwartego stopnia (wielomian kwarcowy). Dodając do stopnia wielomianu, wykres będzie mógł wykonywać skręty i zakręty, aby lepiej dopasować nasze dane, jak pokazano:

What would a low bias, high variance model look like? Let's try polynomial regression, with an fourth order polynomial:

```
sns.lmplot(x='body', y='brain', data=df, ci=None, \

col='sample', order=4)

sns.plt.xlim(-10, 200)

sns.plt.ylim(-10, 250)
```

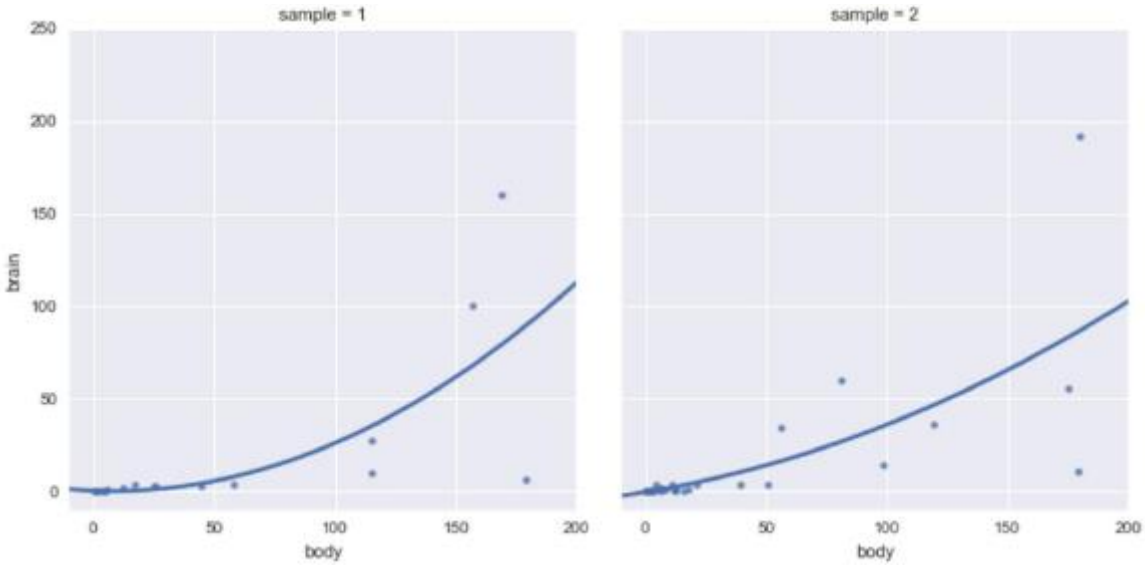


Zwróć uwagę, że dla dwóch różnych próbek z tej samej populacji wielomian kwarcowy wygląda zupełnie inaczej. To znak dużej wariancji. Ten model jest niski, ponieważ dobrze pasuje do naszych danych! Jest to jednak duża wariancja, ponieważ modele są bardzo różne w zależności od tego, które punkty znajdują się w próbce. (Dla masy ciała równej 100 przewidywana masa mózgu wynosiłaby 40 lub 0, w zależności od tego, jakie dane znalazły się w próbce).

Nasz wielomian nie jest również świadomy ogólnego związku danych. Wydaje się oczywiste, że istnieje pozytywna korelacja między mózgiem a masą ciała ssaków. Jednak w naszych wielomianach kwarcowych tej zależności nigdzie nie można znaleźć i jest ona zawodna. W naszym pierwszym przykładzie (wykres po lewej) wielomian kończy się strzelaniem w dół, podczas gdy w drugim wykresie wykres idzie w górę w kierunku końca. Nasz model jest nieprzewidywalny i może zachowywać się bardzo różnie w zależności od danego zestawu treningowego.

Naszym zadaniem, jako naukowców zajmujących się danymi, jest znalezienie złotego środka. Być może uda nam się stworzyć model, który ma mniejsze obciążenie niż model liniowy i mniejszą wariancję niż wielomian czwartego rzędu?

```
# Let's try a second order polynomial instead:

sns.lmplot(x='body', y='brain', data=df, ci=None, col='sample',

order=2)

sns.plt.xlim(-10, 200)

sns.plt.ylim(-10, 250)
```



Ten wykres wydaje się mieć dobrą równowagę między stronniczością i wariancją. Dwa skrajne przypadki kompromisu między stronniczością a wariancją. To, co właśnie zobaczyliśmy, to dwa ekstremalne przypadki dopasowania modelu: jeden był niedopasowany, a drugi nadmiernie dopasowany.

Niedopasowanie

Niedopasowanie ma miejsce, gdy nasze modele podejmują niewiele lub wcale nie próbują dopasować naszych danych. Modele charakteryzujące się wysokim odchyleniem i niską wariancją są podatne na niedopasowanie. W przypadku przykładu mózgu/masy ciała ssaka, regresja liniowa jest niedostateczna dla naszych danych. Chociaż mamy ogólny kształt relacji, pozostajemy z wysokim uprzedzeniem. Jeśli Twój algorytm uczenia się wykazuje wysokie błędy i/lub jest niedostateczny, pomocne mogą być następujące sugestie:

• Użyj większej liczby funkcji: spróbuj włączyć do modelu nowe funkcje, jeśli pomaga to w naszej zdolności predykcyjnej.

• Wypróbuj bardziej skomplikowany model: dodanie złożoności do modelu może pomóc poprawić stronniczość. Zbyt skomplikowany model też zaszkodzi!

Overfitting

Nadmierne dopasowanie jest wynikiem zbyt silnego starania modelu, aby dopasować się do zestawu treningowego, co skutkuje niższym odchyleniem, ale znacznie wyższą wariancją. Modele, które mają niską stronniczość i dużą wariancję, są podatne na nadmierne dopasowanie. W przypadku przykładu mózgu ssaka/masy ciała, regresja wielomianowa czwartego stopnia (kwartyczna) przewyższa nasze dane. Jeśli Twój algorytm uczenia się wykazuje dużą wariancję i/lub jest przesadnie dopasowany, pomocne mogą być następujące sugestie:

• Używaj mniejszej liczby funkcji: używanie mniejszej liczby funkcji może zmniejszyć naszą wariancję i zapobiec nadmiernemu dopasowaniu

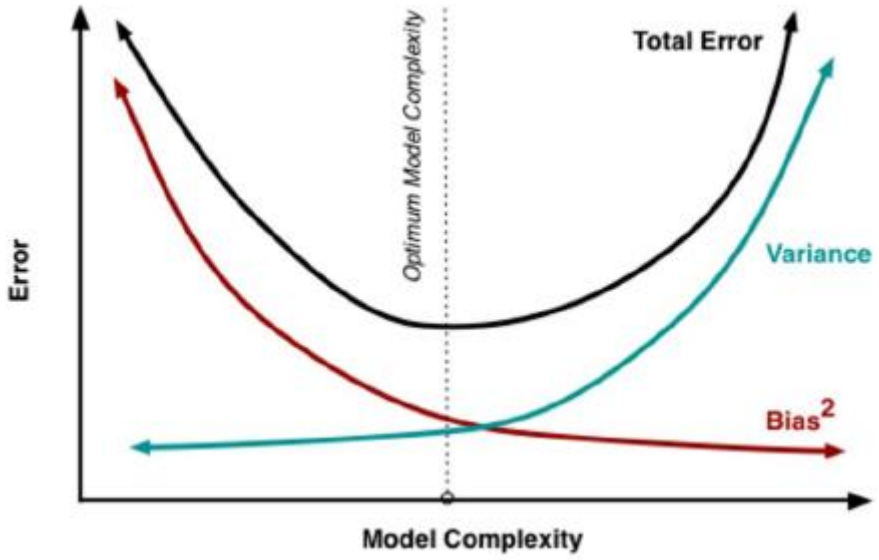
• Dopasuj do większej liczby próbek treningowych: użycie większej liczby treningowych punktów danych w naszej walidacji krzyżowej może zmniejszyć efekt nadmiernego dopasowania i poprawić nasz estymator wysokiej wariancji

Jak bias/wariancja wpływa na funkcje błędów

Funkcje błędu (które mierzą niepoprawność naszych modeli) można traktować jako funkcje błędu systematycznego, wariancji i nieredukowalnego błędu. Mówiąc matematycznie, błąd przewidywania zbioru danych przy użyciu naszego modelu uczenia nadzorowanego może wyglądać następująco:

$$Error(x) = Bias^2 + Variance + Irreducible\ Error$$

Tutaj $Bias^2$ jest naszym terminem odchylenia do kwadratu (pojawia się podczas upraszczania wspomnianego twierdzenia z bardziej skomplikowanych równań), wariancja jest miarą tego, jak bardzo różni się nasz model dopasowania między randomizowanymi próbkami. Mówiąc najprościej, zarówno stronniczość, jak i wariancja przyczyniają się do błędów. Gdy zwiększamy złożoność naszego modelu (na przykład przechodzimy z regresji liniowej do regresji wielomianowej ósmego stopnia lub pogłębiamy nasze drzewa decyzyjne), stwierdzamy, że $Bias^2$ maleje, wariancja wzrasta, a całkowity błąd modelu tworzy kształt paraboliczny. jak zilustrowano:



Naszym celem, jako naukowców zajmujących się danymi, jest znalezienie najlepszego punktu, który zoptymalizuje złożoność naszego modelu. Łatwo jest przepełnić nasze dane. Aby w praktyce przeciwdziałać nadmiernemu dopasowaniu, powinniśmy zawsze stosować walidację krzyżową (iteratywne dzielenie zbiorów danych oraz ponowne uczenie modeli i uśrednianie metryk), aby uzyskać najlepszy predyktor błędu. Aby zilustrować ten punkt, wprowadzę (szybko) nowy nadzorowany algorytm i zademonstruję wizualnie kompromis między stronniczością a wariancją. Będziemy używać algorytmu K-Nearest Neighbors (KNN), który jest nadzorowanym algorytmem uczenia się, który wykorzystuje paradygmat lookalike, co oznacza, że dokonuje przewidywań na podstawie podobnych punktów danych widzianych w przeszłości. KNN ma wejściową złożoność K, która reprezentuje liczbę podobnych punktów danych do porównania. Jeśli K = 3, to dla danego wejścia patrzymy na najbliższe

trzy punkty danych i wykorzystujemy je do naszego przewidywania. W tym przypadku K reprezentuje złożoność naszego modelu. from sklearn.neighbors import KNeighborsClassifier

read in the iris data

from sklearn.datasets import load_iris

iris = load_iris()

X, y = iris.data, iris.target

Mamy więc nasze X i nasze y. Świetnym sposobem na przeuczenie modelu jest trenowanie i przewidywanie na dokładnie tych samych danych.

knn = KNeighborsClassifier(n_neighbors=1)

knn.fit(X, y)

knn.score(X, y)

1.0

Wow, 100% celność?! To jest zbyt piękne by było prawdziwe. Trenując i przewidując na tych samych danych, zasadniczo mówimy naszym danym, aby całkowicie zapamiętały zbiór treningowy i wypluły go z powrotem do nas (nazywa się to naszym błędem treningowym).

Krzyżowa walidacja k-folds

Krzyżowa walidacja K-folds jest znacznie lepszym estymatorem wydajności naszego modelu, nawet bardziej niż nasz podział testowy. Oto jak to działa:

1. Weźmiemy skończoną liczbę równych wycinków naszych danych (zwykle 3, 5 lub 10). Załóżmy, że ta liczba nazywa się k.

2. Dla każdego „zagięcia" walidacji krzyżowej będziemy traktować k-1 sekcji jako zbiór uczący, a pozostałą sekcję jako nasz zbiór testowy.

3. Dla pozostałych fałd, inny układ sekcji k-1 jest rozważany dla naszego zbioru treningowego i inna sekcja to nasz zbiór treningowy.

4. Obliczamy zestaw metryk dla każdego fałdu walidacji krzyżowej.

5. Na koniec uśredniamy nasze wyniki.

Walidacja krzyżowa skutecznie wykorzystuje wiele podziałów testów pociągu wykonywanych na tym samym zbiorze danych. Dzieje się tak z kilku powodów, ale głównie dlatego, że walidacja krzyżowa jest najuczciwszym oszacowaniem błędu naszego modelu poza próbą. Aby wyjaśnić to wizualnie, przyjrzyjmy się przez chwilę przykładowi mózgu i masy ciała naszego ssaka. Poniższy kod ręcznie tworzy pięciokrotną walidację krzyżową, w której pięć różnych zestawów treningowych i testowych jest wykonanych z tej samej populacji:

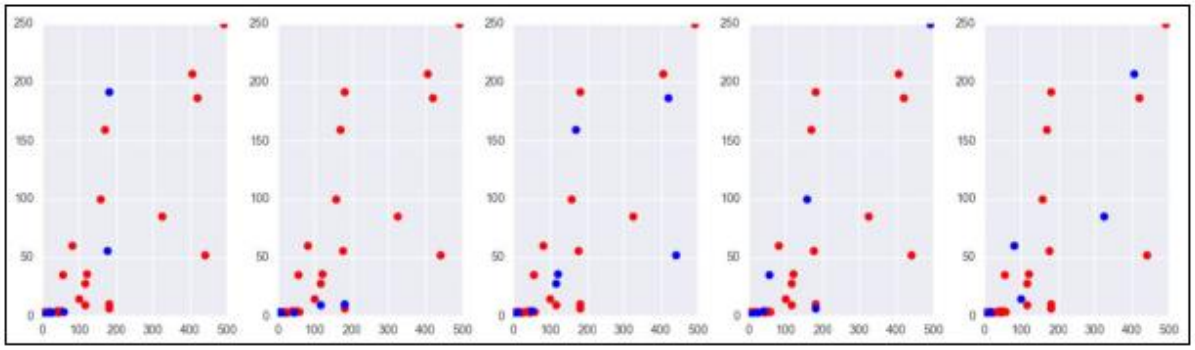
from sklearn.cross_validation import KFold

df = pd.read_table('http://people.sc.fsu.edu/~jburkardt/

datasets/regression/x01.txt', sep='\s+', skiprows=33,

names=['id','brain','body'])

```
df = df[df.brain < 300][df.body < 500]

# limit points for visibility

nfolds = 5

fig, axes = plt.subplots(1, nfolds, figsize=(14,4))

for i, fold in enumerate(KFold(len(df), n_folds=nfolds,

shuffle=True)):

training, validation = fold

x, y = df.iloc[training]['body'], df.iloc[training]['brain']

axes[i].plot(x, y, 'ro')

x, y = df.iloc[validation]['body'], df.iloc[validation]['brain']

axes[i].plot(x, y, 'bo')

plt.tight_layout()
```



Tutaj każdy wykres pokazuje dokładnie tę samą populację ssaków, ale kropki są pokolorowane na czerwono, jeśli należą do zestawu treningowego tej fałdy, i na niebiesko, jeśli należą do zestawu testowego. W ten sposób uzyskujemy pięć różnych wystąpień tego samego modelu uczenia maszynowego, aby sprawdzić, czy wydajność pozostaje spójna we wszystkich fałdach. Jeśli przyglądasz się kropkom wystarczająco długo, zauważysz, że każda kropka pojawia się w zestawie treningowym dokładnie cztery razy (k – 1), podczas gdy ta sama kropka pojawia się w zestawie testowym dokładnie raz i tylko raz. Niektóre funkcje walidacji krzyżowej K-fold obejmują:

• Jest to dokładniejsze oszacowanie błędu przewidywania OOS niż pojedynczy podział na test pociągu, ponieważ bierze kilka niezależnych podziałów na test pociągu i uśrednia wyniki razem.

• Jest to bardziej efektywne wykorzystanie danych niż pojedyncze podziały testów pociągu, ponieważ cały zestaw danych jest używany do wielu podziałów testów pociągu, a nie tylko jednego.

• Każdy rekord w naszym zbiorze danych jest używany zarówno do szkolenia, jak i testowania.

• Ta metoda przedstawia wyraźny kompromis między wydajnością a kosztami obliczeniowymi. Dziesięciokrotne CV jest 10 razy droższe obliczeniowo niż pojedynczy podział pociągu/testu.

• Ta metoda może być używana do strojenia parametrów i wyboru modelu.

Zasadniczo, gdy chcemy przetestować model na zbiorze danych, niezależnie od tego, czy właśnie zakończyliśmy dostrajanie niektórych parametrów, czy też inżynierię funkcji, k-krotna walidacja krzyżowa jest doskonałym sposobem na oszacowanie wydajności naszego modelu. Oczywiście sklearn jest wyposażony w łatwiejszy w użyciu moduł cross_validation o nazwie cross_val_score, który automatycznie dzieli dla nas nasz zbiór danych, uruchamia model na każdym foldzie i daje nam schludne i uporządkowane wyniki:

```
# Using a training set and test set is so important

# Just as important is cross validation. Remember cross validation

# is using several different train test splits and

# averaging your results!

## CROSS-VALIDATION

# check CV score for K=1

from sklearn.cross_validation import cross_val_score, train_test_split

tree = KNeighborsClassifier(n_neighbors=1)

scores = cross_val_score(tree, X, y, cv=5, scoring='accuracy')

scores.mean()

0.95999999999
```

Co jest znacznie bardziej rozsądną dokładnością niż nasz poprzedni wynik równy 1. Pamiętaj, że nie uzyskujemy już 100% dokładności, ponieważ mamy odrębny zestaw treningowy i testowy. Punkty danych, których KNN nigdy nie widziało punktów testowych i dlatego nie może ich dokładnie dopasować do siebie. Spróbujmy przeprowadzić walidację krzyżową KNN z K=5 (zwiększając złożoność naszego modelu), jak pokazano:

```
# check CV score for K=5

knn = KNeighborsClassifier(n_neighbors=5)

scores = cross_val_score(knn, X, y, cv=5, scoring='accuracy')

scores

np.mean(scores)

0.97333333
```

Nawet lepiej! Więc teraz musimy znaleźć najlepsze K? Najlepsze K to takie, które maksymalizuje naszą celność. Wypróbujmy kilka:

```
# search for an optimal value of K

k_range = range(1, 30, 2) # [1, 3, 5, 7, …, 27, 29]

errors = []

for k in k_range:

knn = KNeighborsClassifier(n_neighbors=k)
```

```
# instantiate a KNN with k neighbors

scores = cross_val_score(knn, X, y, cv=5, scoring='accuracy')

# get our five accuracy scores

accuracy = np.mean(scores)

# average them together

error = 1 – accuracy

# get our error, which is 1 minus the accuracy

errors.append(error)

# keep track of a list of errors
```

Mamy teraz wartość błędu (1 – dokładność) dla każdej wartości K (1, 3, 5, 7, 9.., .., 29):

```
# plot the K values (x-axis) versus the 5-fold CV score (y-axis)

plt.figure()

plt.plot(k_range, errors)

plt.xlabel('K')

plt.ylabel('Error')
```

Porównaj ten wykres z poprzednim wykresem złożoności modelu i obciążenia/wariancji. Po lewej stronie nasz wykres ma wyższe odchylenie i jest niedopasowany. Gdy zwiększaliśmy złożoność naszego modelu, składnik błędu zaczął się zmniejszać, ale po pewnym czasie nasz model stał się zbyt złożony i pojawiła się duża wariancja, co spowodowało, że nasz składnik błędu powrócił. Wydaje się, że optymalna wartość K wynosi od 6 do 10.

Wyszukiwanie siatki

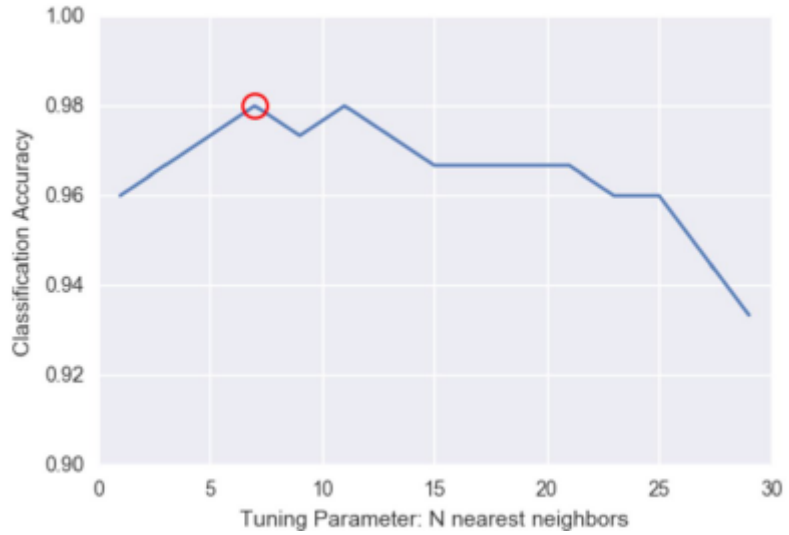
sklearn ma również w zanadrzu inne przydatne narzędzie zwane wyszukiwaniem siatki. Wyszukiwanie siatkowe będzie metodą brutalną wypróbować wiele różnych parametrów modelu i dać nam najlepszy

na podstawie wybranych przez nas metryk. Na przykład możemy wybrać optymalizację KNN pod kątem dokładności w następujący sposób:

```
from sklearn.grid_search import GridSearchCV

# import our grid search module

knn = KNeighborsClassifier()

# instantiate a blank slate KNN, no neighbors

k_range = range(1, 30, 2)

param_grid = dict(n_neighbors=k_range)

# param_grid = {"n_ neighbors": [1, 3, 5, …]}

grid = GridSearchCV(knn, param_grid, cv=5, scoring='accuracy')

grid.fit(X, y)
```

W wierszu kodu grid.fit() dzieje się tak, że dla każdej kombinacji funkcji, w tym przypadku mamy 15 różnych możliwości dla K, przeprowadzamy walidację krzyżową każdej z nich pięć razy. Oznacza to, że pod koniec tego kodu będziemy mieli 15 * 5 = 75 różnych modeli KNN! Możesz zobaczyć, jak stosując tę technikę do bardziej złożonych modeli, z czasem mogliśmy napotkać trudności:

```
# check the results of the grid search

grid.grid_scores_

grid_mean_scores = [result[1] for result in grid.grid_scores_]

# this is a list of the average accuracies for each parameter

# combination

plt.figure()

plt.ylim([0.9, 1])

plt.xlabel('Tuning Parameter: N nearest neighbors')

plt.ylabel('Classification Accuracy')

plt.plot(k_range, grid_mean_scores)

plt.plot(grid.best_params_['n_neighbors'], grid.best_score_, 'ro',

markersize=12, markeredgewidth=1.5,

markerfacecolor='None', markeredgecolor='r')
```



Zauważ, że poprzedni wykres jest w zasadzie taki sam jak ten, który osiągnęliśmy wcześniej za pomocą naszej pętli for, ale jest znacznie łatwiejszy! Widzimy, że siedmiu sąsiadów (zakreślonych na poprzednim wykresie) wydaje się mieć najlepszą dokładność. Jednak możemy również bardzo łatwo uzyskać nasze najlepsze parametry i nasz najlepszy model, jak pokazano:

grid.best_params_

{'n_neighbors': 7}

grid.best_score_

0.9799999999

grid.best_estimator_

actually returns the unfit model with the best parameters

KNeighborsClassifier(algorithm='auto', leaf_size=30,

metric='minkowski',

metric_params=None, n_jobs=1, n_neighbors=7, p=2,

weights='uniform')

Pójdę o krok dalej. Być może zauważyłeś, że KNN ma również inne parametry, takie jak algorytm, p i wagi. Szybkie spojrzenie na dokumentację scikit-learn pokazuje, że mamy kilka opcji dla każdego z nich, które są następujące:

• p jest liczbą całkowitą i reprezentuje rodzaj odległości, której chcemy użyć. Domyślnie używamy p=2, czyli naszego standardowego wzoru na odległość.

• Wagi są domyślnie jednolite, ale mogą być również odległością, która waży punkty na podstawie ich odległości, co oznacza, że bliżsi sąsiedzi mają większy wpływ na predykcję.

• Algorytm to sposób, w jaki model znajduje najbliższych sąsiadów. Możemy spróbować ball_tree, kd_tree lub brute. Wartość domyślna to auto, która automatycznie próbuje użyć najlepszego.

knn = KNeighborsClassifier()

k_range = range(1, 30)

```
algorithm_options = ['kd_tree', 'ball_tree', 'auto', 'brute']

p_range = range(1, 8)

weight_range = ['uniform', 'distance']

param_grid = dict(n_neighbors=k_range, weights=weight_range,

algorithm=algorithm_options, p=p_range)

# trying many more options

grid = GridSearchCV(knn, param_grid, cv=5, scoring='accuracy')

grid.fit(X, y)
```

Uruchomienie poprzedniego kodu na moim laptopie zajmuje około minuty, ponieważ testuje on wiele, 1648 różnych kombinacji parametrów i pięć razy sprawdza krzyżową walidację każdego z nich. Podsumowując, aby uzyskać najlepszą odpowiedź, pasuje do 8400 różnych modeli KNN!

```
grid.best_score_

0.98666666

grid.best_params_

{'algorithm': 'kd_tree', 'n_neighbors': 6, 'p': 3, 'weights':

'uniform'}
```

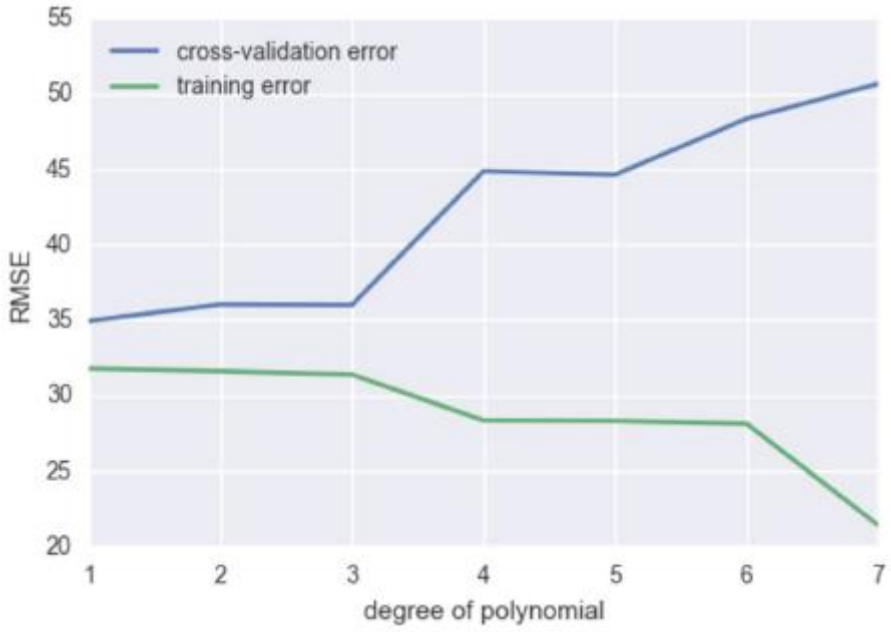
Wyszukiwanie w siatce jest prostym (ale nieefektywnym) sposobem dostrajania parametrów naszych modeli w celu uzyskania najlepszego możliwego wyniku. Należy zauważyć, że aby uzyskać najlepsze możliwe wyniki, naukowcy zajmujący się danymi powinni stosować manipulację cechami (zarówno redukcję, jak i inżynierię), aby uzyskać lepsze wyniki również w praktyce. Osiągnięcie najlepszej wydajności nie powinno zależeć tylko od modelu.

Wizualizacja błędu treningu a błąd walidacji krzyżowej

Myślę, że ważne jest, aby jeszcze raz przejrzeć i porównać błąd walidacji krzyżowej z błędem treningu. Tym razem umieśćmy je na tym samym wykresie, aby porównać, jak zmieniają się wraz ze zmianą złożoności modelu. Użyję zestawu danych ssaków jeszcze raz, aby pokazać błąd walidacji krzyżowej i błąd uczący (błąd przewidywania zestawu uczącego). Przypomnijmy, że próbujemy zredukować masę ciała ssaka do masy mózgu ssaka.

```
# This function uses a numpy polynomial fit function to

# calculate the RMSE of given X and y

def rmse(x, y, coefs):

yfit = np.polyval(coefs, x)

rmse = np.sqrt(np.mean((y - yfit) ** 2))

return rmse

xtrain, xtest, ytrain, ytest = train_test_split(df['body'],

df['brain'])
```

```
train_err = []
validation_err = []
degrees = range(1, 8)
for i, d in enumerate(degrees):
p = np.polyfit(xtrain, ytrain, d)
# built in numpy polynomial fit function
train_err.append(rmse(xtrain, ytrain, p))
validation_err.append(rmse(xtest, ytest, p))
fig, ax = plt.subplots()
# begin to make our graph
ax.plot(degrees, validation_err, lw=2, label = 'cross-validation
error')
ax.plot(degrees, train_err, lw=2, label = 'training error')
# Our two curves, one for training error, the other for cross
validation
ax.legend(loc=0)
ax.set_xlabel('degree of polynomial')
ax.set_ylabel('RMSE')
```



Widzimy więc, że gdy zwiększamy nasz stopień dopasowania, nasz błąd treningowy spada bez żadnych problemów, ale teraz jesteśmy wystarczająco sprytni, aby wiedzieć, że gdy zwiększamy złożoność modelu, nasz model nadmiernie pasuje do naszych danych i jedynie zwraca nasze dane z powrotem do

nas, podczas gdy nasza linia błędu walidacji krzyżowej jest znacznie bardziej uczciwa i zaczyna działać słabo po około stopniu 2 lub 3. Podsumowując:

• Niedopasowanie występuje, gdy zarówno błąd walidacji krzyżowej, jak i błąd uczący są wysokie

• Overrfitting występuje, gdy błąd walidacji krzyżowej jest wysoki, a błąd uczenia jest niski

• Mamy dobre dopasowanie, gdy błąd walidacji krzyżowej jest niski i tylko nieznacznie wyższy niż błąd uczący

Zarówno niedopasowanie (wysokie odchylenie), jak i nadmierne dopasowanie (wysoka wariancja) spowodują słabą generalizację danych. Oto kilka wskazówek, jeśli masz do czynienia z wysokimi uprzedzeniami lub wariancją. Jeśli Twój model ma tendencję do wysokiego odchylenia:

• Spróbuj dodać więcej funkcji do zestawów treningowych i testowych

• Zwiększ złożoność swojego modelu lub wypróbuj bardziej nowoczesny, wyrafinowany model. Jeśli Twój model ma tendencję do dużej wariancji:

• Postaraj się dołączyć więcej próbek treningowych, co zmniejsza efekt nadmiernego dopasowania

Ogólnie rzecz biorąc, kompromis między stronniczością a wariancją to walka o zminimalizowanie stronniczości i wariancji w naszych algorytmach uczenia się. Wiele nowszych algorytmów uczenia się, wynalezionych w ciągu ostatnich kilku dekad, zostało stworzonych z zamiarem posiadania tego, co najlepsze z obu światów.

Techniki zespołowe

Uczenie zespołowe lub tworzenie zestawów to proces łączenia wielu modeli predykcyjnych w celu stworzenia supermodelu, który jest dokładniejszy niż jakikolwiek pojedynczy model.

• Regresja: weźmiemy średnią prognoz dla każdego modelu

• Klasyfikacja: oddaj głos i użyj najczęstszego przewidywania lub weź średnią przewidywanych prawdopodobieństw Wyobraź sobie, że pracujemy nad problemem klasyfikacji binarnej (przewidywanie 0

lub 1).

ENSEMBLING

import numpy as np

set a seed for reproducibility

np.random.seed(12345)

generate 1000 random numbers (between 0 and 1) for each model,

representing 1000 observations

mod1 = np.random.rand(1000)

mod2 = np.random.rand(1000)

mod3 = np.random.rand(1000)

mod4 = np.random.rand(1000)

mod5 = np.random.rand(1000)

Teraz symulujemy pięć różnych modeli uczenia się, z których każdy ma dokładność około 70%, w następujący sposób:

each model independently predicts 1 (the "correct response") if

random number was at least 0.3

preds1 = np.where(mod1 > 0.3, 1, 0)

preds2 = np.where(mod2 > 0.3, 1, 0)

preds3 = np.where(mod3 > 0.3, 1, 0)

preds4 = np.where(mod4 > 0.3, 1, 0)

preds5 = np.where(mod5 > 0.3, 1, 0)

print preds1.mean()

0.699

print preds2.mean()

0.698

print preds3.mean()

0.71

print preds4.mean()

0.699

print preds5.mean()

0.685

Each model has an "accuracy of around 70% on its own

Now, let's apply my degrees in magic. Er sorry, math.

average the predictions and then round to 0 or 1

ensemble_preds = np.round((preds1 + preds2 + preds3 + preds4 +

preds5)/5.0).astype(int)

ensemble_preds.mean()

0.83

W miarę dodawania kolejnych modeli do procesu głosowania zmniejsza się prawdopodobieństwo błędów; jest to znane jako twierdzenie przysięgłych Condorceta. Szalony, prawda? Aby montaż działał dobrze w praktyce, modele muszą mieć następujące cechy:

• Dokładność: każdy model musi co najmniej przewyższać model zerowy

• Niezależność: na prognozę modelu nie ma wpływu proces prognozowania innego modelu

Jeśli masz kilka pojedynczych modeli OK, błędy dotyczące przypadków brzegowych popełnione przez jeden model prawdopodobnie nie zostaną popełnione przez inne modele, więc błędy zostaną zignorowane podczas łączenia modeli.

Istnieją dwie podstawowe metody składania:

• Ręcznie zestawiaj swoje indywidualne modele, pisząc dużo kodu

• Użyj modelu, który składa się dla Ciebie

Przyjrzymy się modelowi, który składa się dla nas. Aby to zrobić, spójrzmy ponownie na drzewa decyzyjne. Drzewa decyzyjne mają zwykle niską stronniczość i dużą wariancję. Biorąc pod uwagę dowolny zestaw danych, drzewo może nadal zadawać pytania (podejmować decyzje), dopóki nie będzie w stanie czepiać się i

rozróżnić każdy przykład w zbiorze danych. Może zadawać pytanie po pytaniu, dopóki w każdym węźle liścia (terminalu) nie będzie tylko jednego przykładu. Drzewo zbyt mocno się stara, rośnie zbyt głęboko i po prostu zapamiętuje każdy szczegół naszego zestawu treningowego. Jednak gdybyśmy zaczęli od nowa, drzewo mogłoby potencjalnie zadawać różne pytania i nadal rosnąć bardzo głęboko. Oznacza to, że istnieje wiele możliwych drzew, które mogłyby rozróżnić wszystkie elementy, co oznacza większą wariancję. Nie potrafi dobrze uogólniać. Aby zredukować wariancję pojedynczego drzewa, możemy nałożyć ograniczenie na liczbę zadawanych pytań w drzewie (parametr max_depth) lub możemy stworzyć zespołową wersję drzew decyzyjnych, czyli lasy losowe.

Losowe lasy

Główną słabością drzew decyzyjnych jest to, że różne podziały danych uczących mogą prowadzić do bardzo różnych drzew. Bagging to procedura ogólnego przeznaczenia mająca na celu zmniejszenie wariancji metody uczenia maszynowego, ale jest szczególnie przydatna w przypadku drzew decyzyjnych. Bagging jest skrótem od agregacji Bootstrap, co oznacza agregację próbek Bootstrap. Co to jest próbka Bootstrap? Jest to próbka losowa z wymianą:

```
# set a seed for reproducibility

np.random.seed(1)

# create an array of 1 through 20

nums = np.arange(1, 21)

print nums

[ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]

# sample that array 20 times with replacement

np.random.choice(a=nums, size=20, replace=True)

[ 6 12 13 9 10 12 6 16 1 17 2 13 8 14 7 19 6 19 12 11]

# This is our bootstrapped sample notice it has repeat variables!
```

Jak więc workowanie działa w przypadku drzew decyzyjnych?

1. Wyhoduj drzewa B, korzystając z próbek Bootstrap z danych treningowych.

2. Wytrenuj każde drzewo na jego próbce Bootstrap i dokonaj prognoz.

3. Połącz przewidywania:

- Średnie prognozy dla drzew regresji

- Oddaj głos na drzewa klasyfikacyjne

Oto kilka rzeczy, na które należy zwrócić uwagę:

• Każda próbka Bootstrap powinna mieć taki sam rozmiar jak oryginalny zestaw szkoleniowy

• B powinno być wystarczająco dużą wartością, aby błąd ustabilizował się

• Drzewa są hodowane celowo głęboko, aby miały niską stronniczość/wysoką wariancję

Powodem, dla którego celowo hodujemy drzewa głęboko, jest to, że workowanie z natury zwiększa dokładność predykcyjną poprzez zmniejszenie wariancji, podobnie jak walidacja krzyżowa zmniejsza wariancję związaną z szacowaniem naszego błędu poza próbą. Lasy losowe to odmiana drzew w workach. Jednak podczas budowania każdego drzewa, za każdym razem, gdy rozważamy podział między cechami, losowa próba m cech jest wybierana jako kandydaci do podziału z pełnego zestawu p cech. Podział może być tylko jedną z tych funkcji m:

• Nowa losowa próbka cech jest wybierana dla każdego drzewa przy każdym podziale

• Do klasyfikacji, m jest zwykle wybierane jako pierwiastek kwadratowy z p

• W przypadku regresji m jest zwykle wybierane jako gdzieś pomiędzy p/3 a p

Jaki jest sens?

Załóżmy, że w zestawie danych jest jedna bardzo silna cecha. Podczas korzystania z drzew decyzyjnych (lub w workach), większość drzew użyje tej funkcji jako podziału wierzchołkowego, w wyniku czego powstanie zespół podobnych drzew, które są ze sobą silnie skorelowane. Jeśli nasze drzewa są ze sobą silnie skorelowane, to uśrednienie tych wielkości nie zmniejszy znacząco wariancji (co jest całym celem zespolenia). Ponadto losowo pomijając cechy kandydujące z każdego podziału, lasy losowe zmniejszają wariancję modelu wynikowego. Lasy losowe mogą być używane zarówno w problemach z klasyfikacją, jak i regresją, a także można je łatwo wykorzystać w scikit-learn. Spróbujmy przewidzieć pensje MLB na podstawie statystyk dotyczących gracza, jak pokazano:

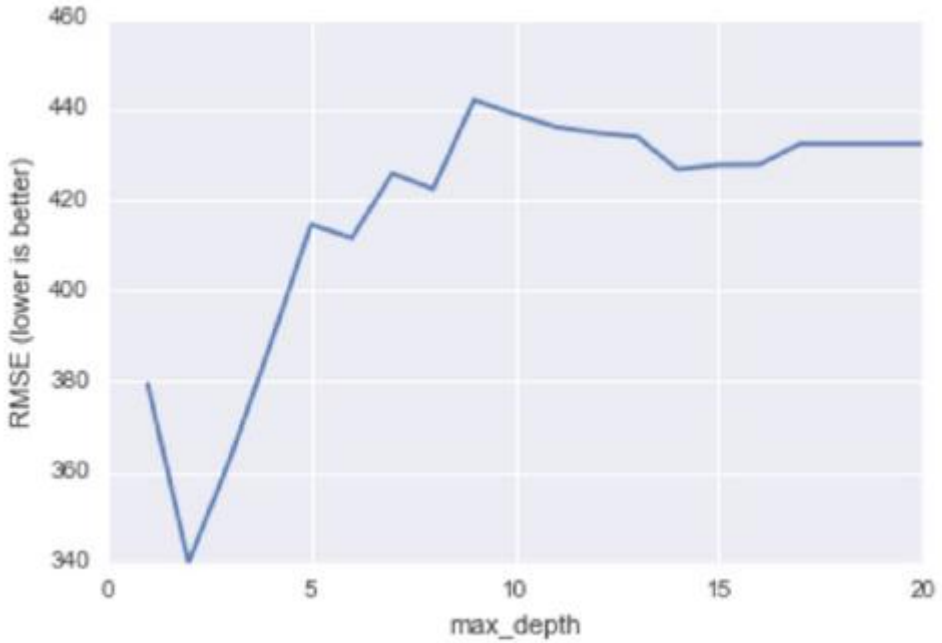
```
# read in the data

url = '../data/hitters.csv'

hitters = pd.read_csv(url)

# remove rows with missing values

hitters.dropna(inplace=True)

# encode categorical variables as integers

hitters['League'] = pd.factorize(hitters.League)[0]

hitters['Division'] = pd.factorize(hitters.Division)[0]

hitters['NewLeague'] = pd.factorize(hitters.NewLeague)[0]

# define features: exclude career statistics (which start with "C")
```

and the response (Salary)

```
feature_cols = [h for h in hitters.columns if h[0] != 'C' and h !=
'Salary']
# define X and y
X = hitters[feature_cols]
y = hitters.Salary
```

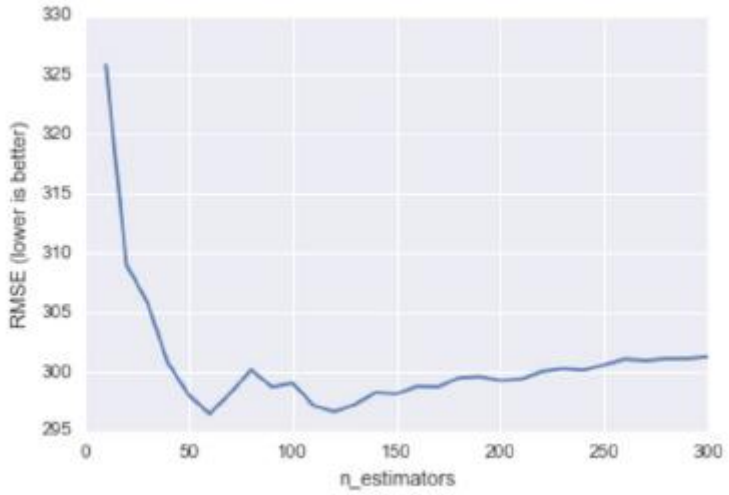
Spróbujmy najpierw przewidzieć wynagrodzenie za pomocą jednego drzewa decyzyjnego, jak pokazano na ilustracji:

```
from sklearn.tree import DecisionTreeRegressor
# list of values to try for max_depth
max_depth_range = range(1, 21)
# list to store the average RMSE for each value of max_depth
RMSE_scores = []
# use 10-fold cross-validation with each value of max_depth
from sklearn.cross_validation import cross_val_score
for depth in max_depth_range:
treereg = DecisionTreeRegressor(max_depth=depth, random_state=1)
MSE_scores = cross_val_score(treereg, X, y, cv=10, scoring='mean_
squared_error')
RMSE_scores.append(np.mean(np.sqrt(-MSE_scores)))
# plot max_depth (x-axis) versus RMSE (y-axis)
plt.plot(max_depth_range, RMSE_scores)
plt.xlabel('max_depth')
plt.ylabel('RMSE (lower is better)')
```



Zróbmy to samo, ale tym razem z losowym lasem:

```
from sklearn.ensemble import RandomForestRegressor
# list of values to try for n_estimators
estimator_range = range(10, 310, 10)
# list to store the average RMSE for each value of n_estimators
RMSE_scores = []
# use 5-fold cross-validation with each value of n_estimators
(WARNING: SLOW!)
for estimator in estimator_range:
rfreg = RandomForestRegressor(n_estimators=estimator, random_
state=1)
MSE_scores = cross_val_score(rfreg, X, y, cv=5, scoring='mean_
squared_error')
RMSE_scores.append(np.mean(np.sqrt(-MSE_scores)))
# plot n_estimators (x-axis) versus RMSE (y-axis)
plt.plot(estimator_range, RMSE_scores)
plt.xlabel('n_estimators')
plt.ylabel('RMSE (lower is better)')
```



Zwróć uwagę na oś y, nasz RMSE jest średnio znacznie niższy! Zobacz, jak możemy uzyskać znaczny wzrost mocy predykcyjnej, korzystając z lasów losowych. W lasach losowych nadal mamy pojęcie o ważnych cechach, takie jak w przypadku drzew decyzyjnych:

```
# n_estimators=150 is sufficiently good

rfreg = RandomForestRegressor(n_estimators=150, random_state=1)

rfreg.fit(X, y)

# compute feature importances

pd.DataFrame({'feature':feature_cols, 'importance':rfreg.feature_

importances_}).sort('importance', ascending = False)
```

	feature	importance
6	Years	0.263990
5	Walks	0.146786
1	Hits	0.139801
4	RBI	0.136265
0	AtBat	0.091551
9	PutOuts	0.060647
3	Runs	0.057460
2	HmRun	0.040183
11	Errors	0.024711
10	Assists	0.023367
8	Division	0.007628
12	NewLeague	0.004545
7	League	0.003067

Wygląda więc na to, że liczba lat, jakie zawodnik spędził w lidze, jest nadal najważniejszą cechą przy ustalaniu wynagrodzenia tego zawodnika.

Porównanie lasów losowych z drzewami decyzyjnymi

Ważne jest, aby zdać sobie sprawę, że samo korzystanie z losowych lasów nie jest rozwiązaniem problemów związanych z analizą danych. Chociaż losowe lasy mają wiele zalet, wiąże się z nimi również wiele wad, jak wymieniono. Zalety lasów losowych to:

• Jego wydajność jest konkurencyjna w stosunku do najlepszych nadzorowanych metod uczenia się

• Zapewnia bardziej wiarygodne oszacowanie ważności funkcji

• Pozwala oszacować błędy poza próbą bez użycia podziałów pociągu/testu lub walidacji krzyżowej

Wady lasów losowych są następujące:

• Jest mniej zrozumiały (nie można zwizualizować całego lasu drzew decyzyjnych)

• Wolniej trenuje się i przewiduje (nie jest to idealne rozwiązanie do celów produkcyjnych lub w czasie rzeczywistym).

Sieci neuronowe

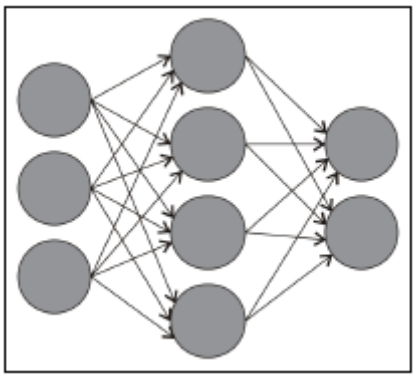
Prawdopodobnie jeden z najczęściej omawianych modeli uczenia maszynowego, sieci neuronowe to sieci obliczeniowe zbudowane do modelowania układu nerwowego zwierząt. Zanim zagłębimy się w strukturę, przyjrzyjmy się dużym zaletom sieci neuronowych. Kluczowym elementem sieci neuronowych jest to, że jest to nie tylko złożona struktura, to złożona i elastyczna struktura. Oznacza to dwie rzeczy:

• Sieci neuronowe są w stanie oszacować dowolny kształt funkcji (nazywa się to byciem nieparametrycznym)

• Sieci neuronowe mogą dostosowywać się i dosłownie zmieniać swoją wewnętrzną strukturę w oparciu o swoje środowisko

Podstawowa struktura

Sieci neuronowe składają się z połączonych ze sobą węzłów (perceptronów), z których każdy przyjmuje dane wejściowe (wartość ilościowa) i wyprowadza inne wartości ilościowe. Sygnały przechodzą przez sieć i ostatecznie trafiają do węzła predykcji.

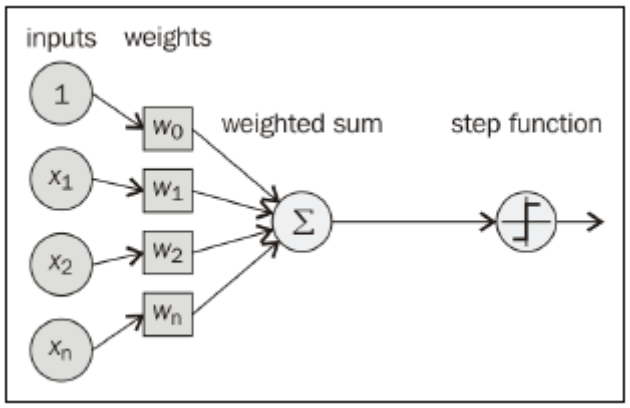


Kolejną ogromną zaletą sieci neuronowych jest to, że mogą być wykorzystywane do uczenia nadzorowanego, uczenia nienadzorowanego i uczenia się przez wzmacnianie. Zdolność do bycia tak elastycznym, przewidywania wielu funkcjonalnych kształtów i dostosowywania się do otoczenia sprawia, że sieci neuronowe są wysoce preferowane w wybranych dziedzinach, jak następuje:

• Rozpoznawanie wzorców: jest to prawdopodobnie najczęstsze zastosowanie sieci neuronowych. Niektóre przykłady to rozpoznawanie pisma ręcznego i przetwarzanie obrazu (rozpoznawanie twarzy).

• Ruch jednostek: Przykładami mogą być autonomiczne samochody, roboty-zwierzęta i ruch dronów.

• Wykrywanie anomalii: Ponieważ sieci neuronowe dobrze rozpoznają wzorce, można ich również używać do rozpoznawania, gdy punkt danych nie pasuje do wzorca. Pomyśl o sieci neuronowej monitorującej ruch cen akcji; po chwili poznania ogólnego wzoru ceny akcji, sieć może ostrzec Cię, gdy coś jest nietypowe w ruchu. Najprostszą formą sieci neuronowej jest pojedynczy perceptron. Perceptron, zwizualizowany w następujący sposób, pobiera pewne dane wejściowe i wysyła sygnał:



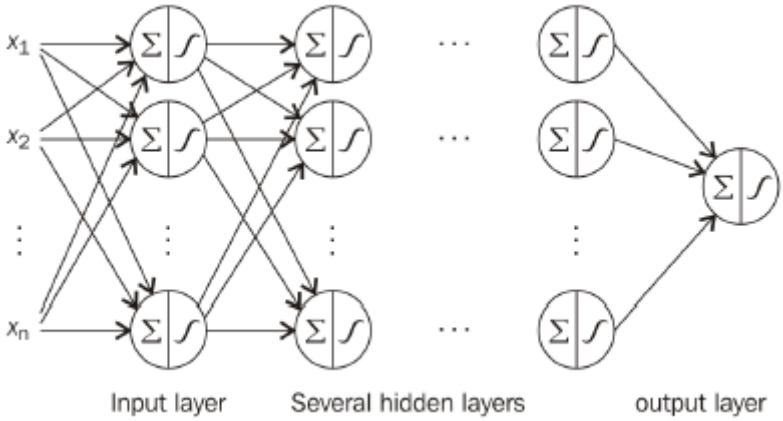
Sygnał ten jest uzyskiwany przez połączenie wejścia z kilkoma wagami, a następnie przechodzi przez jakąś funkcję aktywacji. W przypadku prostych wyjść binarnych zazwyczaj używamy funkcji logistycznej, jak pokazano:

$$f_{log}(z) = \frac{1}{1 + e^{-z}}$$

f_{log} jest azywa funkcją logistyczą

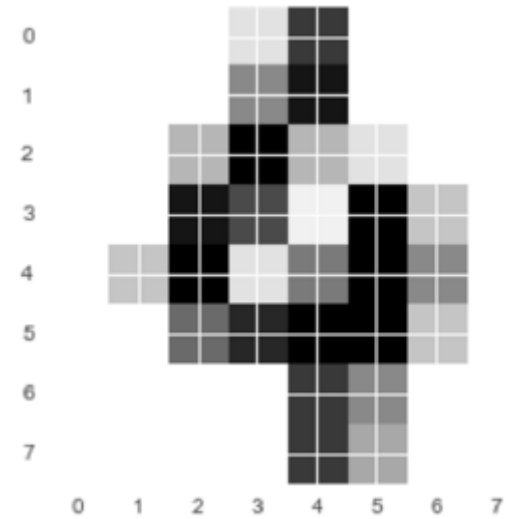
Aby stworzyć sieć neuronową, musimy połączyć ze sobą wiele perceptronów w sposób sieciowy, jak pokazano na poniższym wykresie.

Perceptrony wielowarstwowe (MLP) to skończony graf acykliczny. Węzły to neurony z aktywacją logistyczną.



Podczas uczenia modelu aktualizujemy wagi (na początku losowe) modelu, aby uzyskać najlepsze możliwe przewidywania. Jeśli obserwacja przechodzi przez model i jest wyprowadzana jako fałsz, kiedy powinna być prawdziwa, funkcje logistyczne w pojedynczych perceptronach ulegają nieznacznym zmianom. Nazywa się to propagacją wsteczną. Sieci neuronowe są zwykle uczone w partiach, co oznacza, że sieć otrzymuje kilka uczących punktów danych jednocześnie kilka razy, a za każdym razem algorytm propagacji wstecznej wyzwala wewnętrzną zmianę wagi w sieci. Nietrudno zauważyć, że możemy rozbudować sieć bardzo głęboko i mieć wiele ukrytych warstw, które są związane ze złożonością sieci neuronowej. Kiedy rozwijamy nasze sieci neuronowe bardzo głęboko, zanurzamy nasze stopy w idei głębokiego uczenia się. Główną zaletą głębokich sieci neuronowych (sieci z wieloma warstwami) jest to, że mogą aproksymować prawie każdą funkcję kształtu i mogą (teoretycznie) nauczyć się dla nas optymalnych kombinacji cech i wykorzystać te kombinacje, aby uzyskać najlepszą moc predykcyjną. Zobaczmy to w akcji. Do tworzenia sieci neuronowych będę używał modułu o nazwie PyBrain. Jednak najpierw przyjrzyjmy się nowemu zestawowi danych, który jest zestawem danych odręcznych cyfr. Najpierw spróbujemy rozpoznać cyfry za pomocą losowego lasu, jak pokazano:

```
from sklearn.cross_validation import cross_val_score

from sklearn import datasets

import matplotlib.pyplot as plt

from sklearn.ensemble import RandomForestClassifier

%matplotlib inline

digits = datasets.load_digits()

plt.imshow(digits.images[100], cmap=plt.cm.gray_r,

interpolation='nearest')

# a 4 digit
```




```
X, y = digits.data, digits.target

# 64 pixels per image

X[0].shape
```

```
# Try Random Forest
rfclf = RandomForestClassifier(n_estimators=100, random_state=1)
cross_val_score(rfclf, X, y, cv=5, scoring='accuracy').mean()
0.9382782
```

Całkiem dobre! Dokładność 94% to nic do śmiechu, ale czy możemy zrobić jeszcze lepiej?

Ostrzeżenie! Składnia PyBrain może być nieco trudna.

```
from pybrain.datasets                import ClassificationDataSet
from pybrain.utilities               import percentError
from pybrain.tools.shortcuts         import buildNetwork
from pybrain.supervised.trainers     import BackpropTrainer
from pybrain.structure.modules       import SoftmaxLayer
from numpy import ravel
 # pybrain has its own data sample class that we must add
# our training and test set to
ds = ClassificationDataSet(64, 1 , nb_classes=10)
for k in xrange(len(X)):
ds.addSample(ravel(X[k]),y[k])
# their equivalent of train test split
test_data, training_data = ds.splitWithProportion( 0.25 )
# pybrain's version of dummy variables
test_data._convertToOneOfMany( )
training_data._convertToOneOfMany( )
print test_data.indim # number of pixels going in
# 64
print test_data.outdim # number of possible options (10 digits)
# 10
# instantiate the model with 64 hidden layers (standard params)
fnn = buildNetwork( training_data.indim, 64, training_data.outdim,
outclass=SoftmaxLayer )
trainer = BackpropTrainer( fnn, dataset=training_data, momentum=0.1,
learningrate=0.01 , verbose=True, weightdecay=0.01)
```

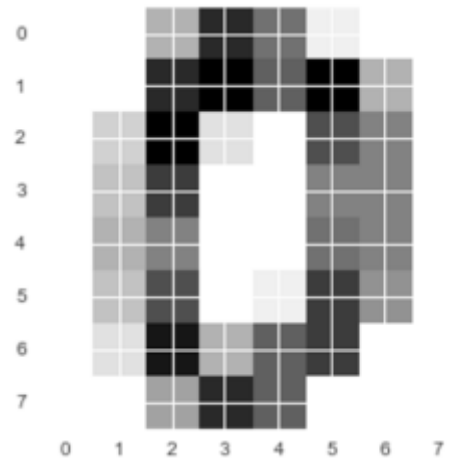
```
# change the number of epochs to try to get better results!

trainer.trainEpochs (10) # 10 batches

print 'Percent Error on Test dataset: ' , \

percentError( trainer.testOnClassData (

dataset=test_data )

, test_data['class'] )
```

Model wygeneruje końcowy błąd w zestawie testowym:

```
Percent Error on Test dataset: 4.67706013363

accuracy = 1 - .0467706013363

accuracy

0.95322
```

Już lepiej! Zarówno lasy losowe, jak i sieci neuronowe bardzo dobrze radzą sobie z tym problemem, ponieważ oba są nieparametryczne, co oznacza, że nie opierają się na podstawowym kształcie danych, aby dokonywać prognoz. Potrafią oszacować dowolny kształt funkcji. Aby przewidzieć kształt, możemy użyć następującego kodu:

```
plt.imshow(digits.images[0], cmap=plt.cm.gray_r,

interpolation='nearest')

fnn.activate(X[0])

array([ 0.92183643, 0.00126609, 0.00303146, 0.00387049,

0.01067609,

0.00718017, 0.00825521, 0.00917995, 0.00696929,

0.02773482])
```



Tablica reprezentuje prawdopodobieństwo dla każdej cyfry, co oznacza, że istnieje 92% szans, że cyfra na poprzednim zrzucie ekranu to 0 (co tak jest). Zwróć uwagę, że następne najwyższe

prawdopodobieństwo jest dla 9, co ma sens, ponieważ 9 i 0 mają podobne kształty (owalny). Sieci neuronowe mają poważną wadę. Pozostawione same sobie mają bardzo dużą wariancję. Aby to zobaczyć, uruchommy dokładnie ten sam kod co poprzedni i wytrenujmy dokładnie ten sam typ sieci neuronowej na dokładnie tych samych danych, jak pokazano na ilustracji:

```
# Do it again and see the difference in error

fnn = buildNetwork( training_data.indim, 64, training_data.outdim,

outclass=SoftmaxLayer )

trainer = BackpropTrainer( fnn, dataset=training_data, momentum=0.1,

learningrate=0.01 , verbose=True, weightdecay=0.01)

# change the number of eopchs to try to get better results!

trainer.trainEpochs (10)

print 'Percent Error on Test dataset: ' , \

percentError( trainer.testOnClassData (

dataset=test_data )

, test_data['class'] )

accuracy = 1 - .0645879732739

accuracy

0.93541
```

Zobacz, jak samo ponowne uruchomienie modelu i utworzenie instancji różnych wag sprawiło, że sieć okazała się inna niż wcześniej? Jest to symptom bycia modelem o dużej wariancji. Ponadto sieci neuronowe na ogół wymagają wielu próbek uczących w celu zwalczania dużej wariancji modelu, a także wymagają dużej mocy obliczeniowej do prawidłowego działania w środowiskach produkcyjnych.