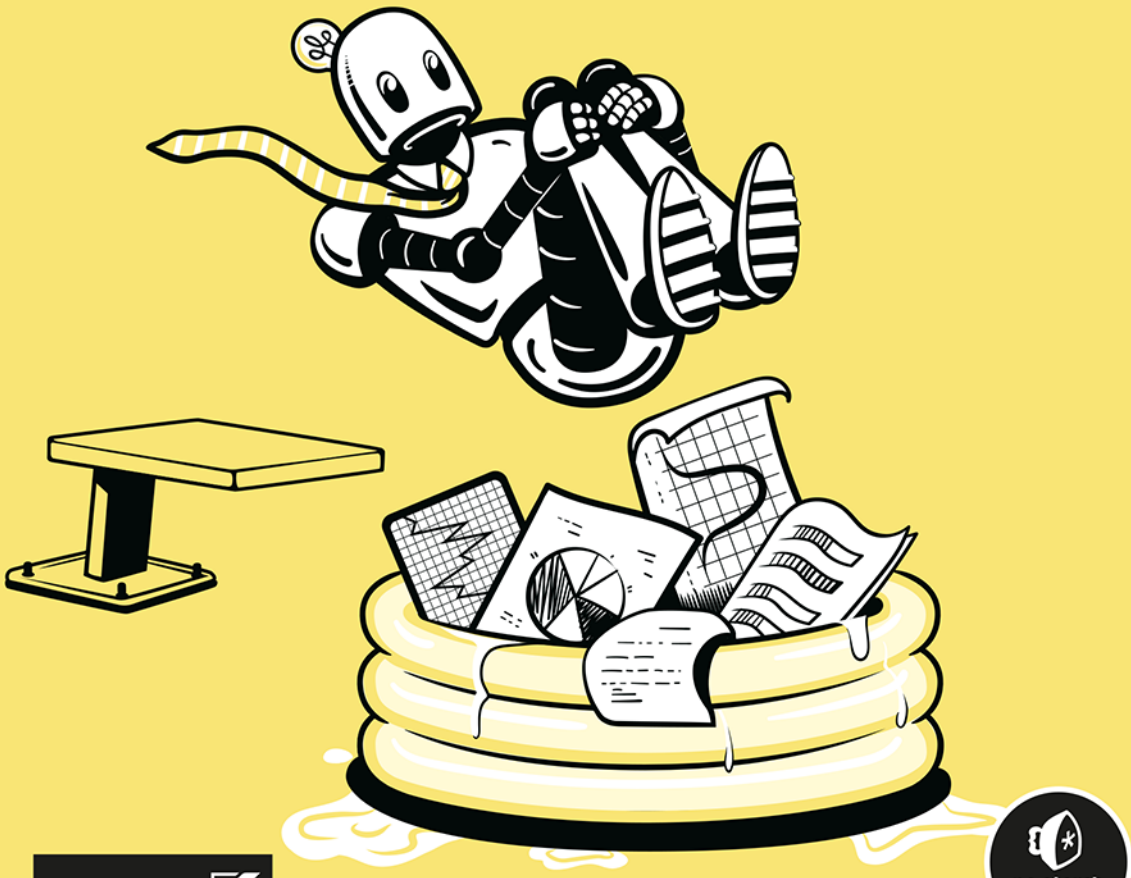
DATA SCIENCE I PYTHON

STAWIANIE CZOŁA NAJTRUDNIEJSZYM WYZWANIOM BIZNESOWYM

BRADFORD TUCKFIELD



Helion

no starch press

Tytuł oryginału: Dive Into Data Science: Use Python To Tackle Your Toughest Business Challenges

Tłumaczenie: Grzegorz Werner

ISBN: 978-83-289-1014-0

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
https://helion.pl/user/opinie/dascpy
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *https://helion.pl* (księgarnia internetowa, katalog książek)

Printed in Poland.

Spis treści

1

Eksploracyjna analiza danych

Jest to książka o data science, zacznijmy więc od zbadania jakichś danych. To coś, do czego powinieneś przywyknąć: pierwszym etapem każdego problemu data science jest eksplorowanie danych. Bliskie przyjrzenie się najdrobniejszym szczegółom pomoże Ci lepiej zrozumieć dane i zdecydować, co robić dalej i jak przeprowadzić bardziej wyrafinowane analizy. Pomoże też wcześnie wyłapać ewentualne błędy lub problemy z danymi. Te pierwsze kroki w procesie data science nazywa się **eksploracyjną analizą danych**.

Zaczniemy ten rozdział od przedstawienia scenariusza biznesowego i wyjaśnienia, jak można wykorzystać dane do lepszego zarządzania firmą. Opiszemy wczytywanie danych do Pythona i sprawdzanie podstawowych statystyk zbiorczych. Następnie wprowadzimy narzędzia Pythona przeznaczone do tworzenia wykresów. Omówimy proste analizy eksploracyjne i pytania, na które mogą odpowiedzieć. Na koniec przedyskutujemy, jak analizy mogą pomóc w poprawianiu praktyk biznesowych. Proste analizy, które przeprowadzimy w tym rozdziale, są pierwszym krokiem w rozwiązywaniu każdego problemu data science, jaki napotkasz w przyszłości. Zaczynajmy!

Twój pierwszy dzień na stanowisku prezesa

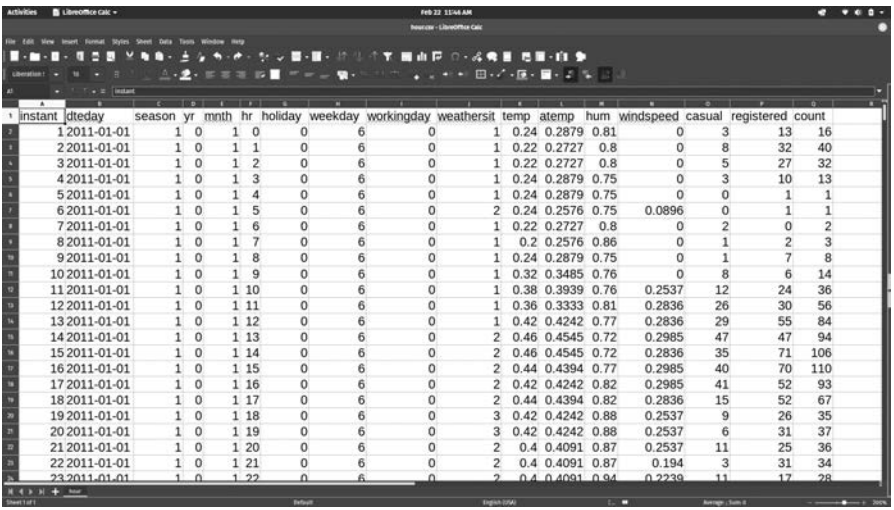
Wyobraź sobie, że zaoferowano Ci posadę prezesa w firmie udostępniającej rowery, które można wypożyczać na krótkie okresy w celu jeżdżenia po mieście. Choć nie masz żadnego doświadczenia w prowadzeniu firm zajmujących się wypożyczaniem rowerów, akceptujesz ofertę.

Zjawiasz się w pracy pierwszego dnia i myślisz o swoich celach biznesowych. Niektóre z nich mogą być związane z takimi kwestiami jak satysfakcja klienta, morale pracowników, rozpoznawalność marki, maksymalizacja udziału w rynku, redukcja kosztów lub wzrost przychodów. Jak zdecydować, które z tych celów są priorytetowe, i w jaki sposób do nich dążyć? Pomyśl na przykład o zwiększaniu satysfakcji klienta. Zanim skupisz się na tym celu, musisz się dowiedzieć, czy klienci są zadowoleni, a jeśli nie są, musisz odkryć, co obniża ich satysfakcję i jak można to poprawić. A może jesteś bardziej zainteresowany pracą nad zwiększaniem przychodów. Musiałbyś wiedzieć, jakie są obecne przychody, zanim spróbowałbyś je zwiększyć. Innymi słowy, nie będziesz wiedział, na czym powinieneś się początkowo skupić, dopóki nie zrozumiesz lepiej swojej firmy.

Aby zrozumieć swoją firmę, potrzebujesz danych. Mógłbyś przejrzeć wykresy i raporty, które podsumowują działalność firmy, ale żaden gotowy raport nie powie Ci tyle co samodzielne zbadanie danych.

Znajdowanie wzorców w zbiorach danych

Przyjrzyjmy się danym z prawdziwej wypożyczalni rowerów i wyobraźmy sobie, że są to dane z Twojej firmy. Możesz pobrać te dane pod adresem *https://bradfordtuckfield.com/hour.csv*. (Plik ten jest w specjalnym formacie *.csv*, o którym wkrótce powiemy więcej). Możesz otworzyć ten plik w edytorze arkuszy kalkulacyjnych, takim jak Microsoft Excel lub LibreOffice Calc; powinieneś zobaczyć okno podobne do pokazanego na rysunku 1.1.



instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	count
1	2011-01-01	1	0	1	0	0	6	0	1	0.24	0.2879	0.81	0	3	13	16
2	2011-01-01	1	0	1	1	0	6	0	1	0.22	0.2727	0.8	0	8	32	40
3	2011-01-01	1	0	1	2	0	6	0	1	0.22	0.2727	0.8	0	5	27	32
4	2011-01-01	1	0	1	3	0	6	0	1	0.24	0.2879	0.75	0	3	10	13
5	2011-01-01	1	0	1	4	0	6	0	1	0.24	0.2879	0.75	0	0	1	1
6	2011-01-01	1	0	1	5	0	6	0	2	0.24	0.2576	0.75	0.0896	0	1	1
7	2011-01-01	1	0	1	6	0	6	0	1	0.22	0.2727	0.8	0	2	0	2
8	2011-01-01	1	0	1	7	0	6	0	1	0.2	0.2576	0.86	0	1	2	3
9	2011-01-01	1	0	1	8	0	6	0	1	0.24	0.2879	0.75	0	1	7	8
10	2011-01-01	1	0	1	9	0	6	0	1	0.32	0.3485	0.76	0	8	6	14
11	2011-01-01	1	0	1	10	0	6	0	1	0.38	0.3939	0.76	0.2537	12	24	36
12	2011-01-01	1	0	1	11	0	6	0	1	0.36	0.3333	0.81	0.2836	26	30	56
13	2011-01-01	1	0	1	12	0	6	0	1	0.42	0.4242	0.77	0.2836	29	55	84
14	2011-01-01	1	0	1	13	0	6	0	2	0.46	0.4545	0.72	0.2985	47	47	94
15	2011-01-01	1	0	1	14	0	6	0	2	0.46	0.4545	0.72	0.2836	35	71	106
16	2011-01-01	1	0	1	15	0	6	0	2	0.44	0.4394	0.77	0.2985	40	70	110
17	2011-01-01	1	0	1	16	0	6	0	2	0.42	0.4242	0.82	0.2985	41	52	93
18	2011-01-01	1	0	1	17	0	6	0	2	0.44	0.4394	0.82	0.2836	15	52	67
19	2011-01-01	1	0	1	18	0	6	0	3	0.42	0.4242	0.88	0.2537	9	26	35
20	2011-01-01	1	0	1	19	0	6	0	3	0.42	0.4242	0.88	0.2537	6	31	37
21	2011-01-01	1	0	1	20	0	6	0	2	0.4	0.4091	0.87	0.2537	11	25	36
22	2011-01-01	1	0	1	21	0	6	0	2	0.4	0.4091	0.87	0.194	3	31	34
23	2011-01-01	1	0	1	22	0	6	0	2	0.4	0.4091	0.94	0.2239	11	17	28

Rysunek 1.1. Dane dotyczące wypożyczania rowerów wyświetlone w arkuszu kalkulacyjnym

UWAGA

*Pierwotnym źródłem tych danych jest witryna firmy Capital Bikeshare (*https://ride.capitalbike➥share.com/system-data*). Dane zostały skompilowane i wzbogacone przez Hadi Fanaee-T oraz Joao Gamę i opublikowane w internecie przez Marka Kaghazgariana.*

Ten zbiór danych nie różni się od wielu innych zbiorów, które prawdopodobnie widywałeś wcześniej: jest to prostokątna tablica wierszy i kolumn. W tym zbiorze każdy wiersz reprezentuje informacje o konkretnej godzinie od północy 1 stycznia 2011 r. do 23.59 31 grudnia 2012 r. — łącznie ponad 17 000 godzin. Wiersze są ułożone w kolejności chronologicznej, więc kilka pierwszych wierszy daje nam informacje o kilku pierwszych godzinach 2011 r., a kilka ostatnich — o kilku ostatnich godzinach 2012 r.

Każda kolumna zawiera konkretny wskaźnik zmierzony dla każdej z tych godzin. Na przykład kolumna `windspeed` zawiera pomiary prędkości wiatru w pewnej stacji meteorologicznej w Waszyngtonie. Zauważ, że pomiary te nie są wyrażone w znajomych jednostkach, takich jak mile czy kilometry na godzinę. Zamiast tego zostały przekształcone do zakresu od 0 do 1; wszystko, co musimy wiedzieć, to że 1 reprezentuje szybki wiatr, a 0 brak wiatru.

Jeśli przyjrzysz się kilku pierwszym wierszom, zobaczysz, że prędkość wiatru wynosi w nich 0, co oznacza, że w kilku pierwszych godzinach istnienia wypożyczalni rowerów nie zmierzono żadnego wiatru. W siódmym wierszu (licząc nagłówek jako pierwszy wiersz) nareszcie pojawia się jakiś wiatr, a jego zmierzona prędkość wynosi 0,0896. Jeśli spojrzysz na kolumnę `hr`, dowiesz się, że wiatr ten zmierzono, kiedy `hr = 5`, czyli o piątej nad ranem. Wiemy, że wiersz ten zawiera informacje o 1 stycznia, ponieważ kolumna `dteday` w siódmym wierszu ma wartość `2011-01-01`.

Po prostu przyglądając się kilku pierwszym wartościom danych, możemy zacząć opowiadać pewną historię, choć nieszczególnie ekscytującą: cicha noworoczna noc zmieniła się w nieco mniej cichy noworoczny poranek. Jeśli chcemy poznać inne historie o wypożyczalni rowerów i jej działalności, a nie tylko o pogodzie, musimy przyjrzeć się innym, bardziej istotnym kolumnom.

Najważniejsze informacje znajdują się w ostatnich trzech kolumnach: `casual`, `registered` i `count`. Kolumny te wskazują liczbę osób, które używały rowerów Twojej firmy w każdej godzinie. Osoby, które zarejestrowały się w Twojej usłudze, aby otrzymywać zniżki i inne korzyści, to użytkownicy zarejestrowani, a ich liczba jest zapisana w kolumnie `registered`. Z rowerów można jednak korzystać również bez rejestrowania się, a takie przejazdy są zapisane w kolumnie `casual`. Suma kolumn `casual` i `registered` to łączna liczba użytkowników w każdej godzinie, która jest zapisana w kolumnie `count`.

Teraz, kiedy zapoznałeś się z najistotniejszymi kolumnami w tym zbiorze danych, możesz dowiedzieć się sporo, po prostu przyglądając się zawartym w nich liczbom. Na przykład patrząc na 20 godzin pokazanych na rysunku 1.1, możesz stwierdzić, że w większości godzin masz więcej użytkowników zarejestrowanych niż okazjonalnych (wyższe wartości w kolumnie `registered` niż w kolumnie `casual`). Jest to prosty fakt liczbowy, ale jako prezes powinieneś się zastanowić, jakie ma to implikacje dla Twojej firmy. Więcej użytkowników zarejestrowanych niż okazjonalnych może oznaczać, że firma dobrze radzi sobie z przekonywaniem ludzi, żeby się zarejestrowali, ale może

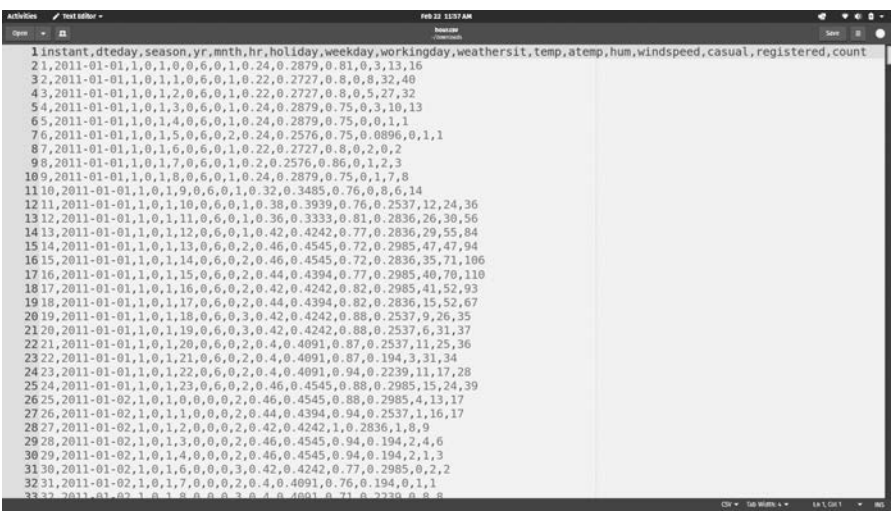
również oznaczać, że okazjonalne używanie usługi bez rejestracji nie jest tak łatwe, jak powinno. Będziesz musiał pomyśleć, który segment klientów jest dla Ciebie ważniejszy: regularni, zarejestrowani użytkownicy, na przykład codziennie dojeżdżający do pracy, czy użytkownicy okazjonalni, tacy jak zwiedzający miasto turyści.

Przyjrzyjmy się bliżej codziennym wzorcom okazjonalnych i zarejestrowanych użytkowników i zobaczmy, czego możemy się o nich dowiedzieć. Spójrz ponownie na godziny pokazane na rysunku 1.1. Jak widać, użytkownicy okazjonalni są nieliczni do popołudnia pierwszego dnia, a ich szczyt przypada mniej więcej na godzinę 13.00. Użytkownicy zarejestrowani są względnie liczni nawet o 1.00 w nocy pierwszego dnia, a ich szczyt przypada na godzinę 14.00. Różnice w zachowaniach użytkowników zarejestrowanych i okazjonalnych są niewielkie, ale mogą mieć znaczenie. Mogą na przykład wskazywać na różnice demograficzne między tymi grupami. To z kolei może wymagać różnych strategii marketingowych kierowanych do każdej z grup.

Pomyśl tylko, co już zrobiliśmy: po prostu przyglądając się kilku kolumnom pierwszych 24 wierszy danych, dowiedzieliśmy się kilku ważnych rzeczy o firmie i zaczęliśmy formułować pierwsze pomysły biznesowe. Nauka o danych jest znana z tego, że wymaga tajemnej wiedzy matematycznej i informatycznej, ale czasem wystarczy spojrzeć na zbiór danych, chwilę pomyśleć i posłużyć się zdrowym rozsądkiem, aby poprawić wyniki biznesowe.

Używanie plików .csv do przeglądania i przechowywania danych

Przyjrzymy się jeszcze bliżej naszym danym. Jeśli otworzysz plik danych (*hour.csv*) w edytorze arkuszy kalkulacyjnych, będzie on wyglądał tak jak na rysunku 1.1. Możesz jednak również otworzyć ten plik w edytorze tekstu, takimi jak Notatnik (jeśli używasz Windowsa), TextEdit (jeśli korzystasz z macOS-a) albo GNU Emacs lub gedit (jeśli używasz Linuksa). Plik otwarty w edytorze tekstu będzie wyglądał tak jak na rysunku 1.2.



```
instant,dteday,season,yr,mnth,hr,holiday,weekday,workingday,weathersit,temp,atemp,hum,windspeed,casual,registered,count
1,2011-01-01,1,0,1,0,0,6,0,1,0.24,0.2879,0.81,0,3,13,16
2,2011-01-01,1,0,1,1,0,6,0,1,0.22,0.2727,0.8,0,8,32,40
3,2011-01-01,1,0,1,2,0,6,0,1,0.22,0.2727,0.8,0,5,27,32
4,2011-01-01,1,0,1,3,0,6,0,1,0.24,0.2879,0.75,0,3,10,13
5,2011-01-01,1,0,1,4,0,6,0,1,0.24,0.2879,0.75,0,0,1,1
6,2011-01-01,1,0,1,5,0,6,0,2,0.24,0.2576,0.75,0.0896,0,1,1
7,2011-01-01,1,0,1,6,0,6,0,1,0.22,0.2727,0.8,0,2,0,2
8,2011-01-01,1,0,1,7,0,6,0,1,0.2,0.2576,0.86,0,1,2,3
9,2011-01-01,1,0,1,8,0,6,0,1,0.24,0.2879,0.75,0,1,7,8
10,2011-01-01,1,0,1,9,0,6,0,1,0.32,0.3485,0.76,0,8,6,14
11,2011-01-01,1,0,1,10,0,6,0,1,0.38,0.3939,0.76,0.2537,12,24,36
12,2011-01-01,1,0,1,11,0,6,0,1,0.36,0.3333,0.81,0.2836,26,30,56
13,2011-01-01,1,0,1,12,0,6,0,1,0.42,0.4242,0.77,0.2836,29,55,84
14,2011-01-01,1,0,1,13,0,6,0,2,0.46,0.4545,0.72,0.2985,47,47,94
15,2011-01-01,1,0,1,14,0,6,0,2,0.46,0.4545,0.72,0.2836,35,71,106
16,2011-01-01,1,0,1,15,0,6,0,2,0.44,0.4394,0.77,0.2985,40,70,110
17,2011-01-01,1,0,1,16,0,6,0,2,0.42,0.4242,0.82,0.2985,41,52,93
18,2011-01-01,1,0,1,17,0,6,0,2,0.44,0.4394,0.82,0.2836,15,52,67
19,2011-01-01,1,0,1,18,0,6,0,3,0.42,0.4242,0.88,0.2537,9,26,35
20,2011-01-01,1,0,1,19,0,6,0,3,0.42,0.4242,0.88,0.2537,6,31,37
21,2011-01-01,1,0,1,20,0,6,0,2,0.4,0.4091,0.87,0.2537,11,25,36
22,2011-01-01,1,0,1,21,0,6,0,2,0.4,0.4091,0.87,0.194,3,31,34
23,2011-01-01,1,0,1,22,0,6,0,2,0.4,0.4091,0.94,0.2239,11,17,28
24,2011-01-01,1,0,1,23,0,6,0,2,0.46,0.4545,0.88,0.2985,15,24,39
25,2011-02-01,1,0,1,0,0,0,0,2,0.46,0.4545,0.88,0.2985,4,13,17
26,2011-02-01,1,0,1,1,0,0,0,2,0.44,0.4394,0.94,0.2537,1,16,17
27,2011-02-01,1,0,1,2,0,0,0,2,0.42,0.4242,1,0.2836,1,8,9
28,2011-02-01,1,0,1,3,0,0,0,2,0.46,0.4545,0.94,0.194,2,4,6
29,2011-02-01,1,0,1,4,0,0,0,2,0.46,0.4545,0.94,0.194,2,1,3
30,2011-02-01,1,0,1,6,0,0,0,3,0.42,0.4242,0.77,0.2985,0,2,2
31,2011-02-01,1,0,1,7,0,0,0,2,0.4,0.4091,0.76,0.194,0,1,1
```

Rysunek 1.2. Dane dotyczące wypożyczeń rowerów wyświetlone jako zwykły tekst

Te surowe dane (tzn. każdy pojedynczy znak tekstowy) składają się na nasz plik *hour.csv*, bez wyrównanych kolumn, jak w arkuszu kalkulacyjnym. Zwróć uwagę na liczne przecinki. Rozszerzenie nazwy tego pliku, *.csv*, to skrót od angielskiego terminu *comma-separated values* (wartości oddzielone przecinkami).

Kiedy otwierasz plik *.csv* w edytorze arkuszy kalkulacyjnych, edytor próbuje zinterpretować każdy przecinek jako granicę między komórkami arkusza, żeby wyświetlić dane w prostych, wyrównanych wierszach i kolumnach. Same dane nie są jednak przechowywane w taki sposób; jest to po prostu surowy tekst z wierszami wartości, w których każda wartość jest oddzielona od innych przecinkami.

Prostota plików *.csv* oznacza, że można je łatwo tworzyć, otwierać w wielu typach programów i w prosty sposób zmieniać. Właśnie dlatego analitycy danych często przechowują swoje dane w formacie *.csv*.

Wyświetlanie danych w Pythonie

Użycie Pythona pozwoli nam dokonywać bardziej zaawansowanych analiz niż te, które można przeprowadzać w edytorach tekstu i programach arkusza kalkulacyjnego. Umożliwi nam też zautomatyzowanie procesów i przyspieszenie analiz. Możemy łatwo otwierać pliki *.csv* w Pythonie. Poniższe trzy wiersze kodu Pythona wczytują plik *hour.csv* do Twojej sesji Pythona i wyświetlają pięć pierwszych wierszy:

```
mport pandas as pd
hour=pd.read_csv('hour.csv')
print(hour.head())
```

Później przyjrzymy się bliżej wynikom tego fragmentu kodu. Na razie spójrzmy na sam kod. Służy on do wczytania i wyświetlenia naszych danych. Drugi wiersz wczytuje dane za pomocą metody `read_csv()`. **Metoda** to jednostka kodu, która realizuje jedną, dobrze zdefiniowaną funkcję. Jak sugeruje nazwa, metoda `read_csv()` jest przeznaczona specjalnie do wczytywania danych, które są przechowywane w plikach *.csv*. Kiedy wykonasz ten wiersz, zmienna `hour` będzie zawierać wszystkie dane z pliku *hour.csv*; następnie będziesz mógł uzyskać dostęp do tych danych w Pythonie.

W trzecim wierszu używamy funkcji `print()` do wyświetlenia (wypisania) naszych danych na ekranie. Moglibyśmy zmienić ten wiersz na `print(hour)`, aby wypisać całą zawartość zbioru danych. Zbiory danych bywają jednak bardzo duże, więc wypisywanie ich w całości mijałoby się z celem. Dlatego dodaliśmy metodę `head()`, która zwraca tylko pięć pierwszych wierszy zbioru danych.

Zarówno metoda `read_csv()`, jak i `head()` mogą być bardzo użyteczne. Nie są jednak częścią **standardowej biblioteki Pythona** — jego standardowych, domyślnie instalowanych funkcji. Stanowią część pakietu, czyli zewnętrznego kodu, który można opcjonalnie zainstalować i używać w skryptach Pythona.

Te dwie metody są częścią popularnego pakietu o nazwie *pandas*, który zawiera kod do pracy z danymi. Właśnie dlatego pierwszy wiersz powyższego przykładu to `import pandas as pd`; *importuje* on, czyli wczytuje do pamięci, pakiet pandas, żebyśmy mogli korzystać z niego w naszej sesji Pythona. Notacja `as pd` przypisuje pakietowi *alias*, dzięki czemu za każdym razem, kiedy chcemy skorzystać z funkcji pakietu pandas, możemy napisać `pd` zamiast pełnej nazwy „pandas". Pisząc `pd.read_csv()`, uzyskujemy zatem dostęp do metody `read_csv()`, która jest częścią pakietu pandas.

Jeśli otrzymasz komunikat o błędzie, kiedy spróbujesz wykonać polecenie `import pandas as pd`, możliwe, że pakiet pandas nie jest zainstalowany w Twoim komputerze (pakiety muszą zostać zainstalowane, zanim będziesz mógł je zaimportować). Aby zainstalować pandas albo dowolny inny pakiet Pythona, powinieneś użyć standardowego instalatora Pythona, czyli programu o nazwie `pip`. Instrukcję instalacji programu `pip` oraz używania go do instalowania pakietów Pythona, takich jak pandas, znajdziesz we wstępie do książki. W całej tej książce za każdym razem, kiedy będziesz importować pakiet, najpierw upewnij się, że zainstalowałeś go w swoim komputerze za pomocą programu `pip`.

Podczas wykonywania tego fragmentu kodu możesz też otrzymać inny komunikat o błędzie. Jeden z najbardziej typowych błędów występuje wtedy, gdy Python nie może znaleźć pliku *hour.csv*. Jeśli tak się stanie, Python wypisze raport o błędzie. Ostatni wiersz raportu będzie wyglądał mniej więcej tak:

```
FileNotFoundError: [Errno 2] No such file or directory: 'hour.csv'
```

Nawet jeśli nie jesteś znawcą Pythona, zapewne domyślasz się, co to znaczy: Python spróbował odczytać plik *hour.csv*, ale go nie znalazł. Błąd ten może być frustrujący, ale łatwo mu zaradzić. Najpierw upewnij się, że pobrałeś plik *hour.csv* i że ma on nazwę *hour.csv* w Twoim komputerze. Nazwa pliku w komputerze musi dokładnie odpowiadać nazwie pliku w kodzie Pythona.

Jeśli nazwa pliku jest zapisana poprawnie w Twoim kodzie Pythona (wyłącznie małymi literami), problemem jest zapewne lokalizacja pliku. Jak wiesz, każdy plik w Twoim komputerze ma unikatową ścieżkę, która dokładnie określa, gdzie można go znaleźć. Ścieżka do pliku może wyglądać tak:

```
C:\Users\DonQuixote\Documents\hour.csv
```

Ta ścieżka jest w formacie używanym w systemie operacyjnym Windows. Jeśli używasz Windowsa, upewnij się, że nazwy Twoich folderów i plików nie zawierają żadnych znaków specjalnych (na przykład znaków z alfabetów innych niż angielski), ponieważ ścieżki ze znakami specjalnymi mogą powodować błędy. Poniżej pokazano inny przykład ścieżki do pliku, tym razem w formacie używanym przez uniksowe systemy operacyjne (w tym macOS i Linux):

```
/home/DonQuixote/Documents/hour.csv
```

Jak widać, ścieżki do plików w Windowsie wyglądają inaczej niż ścieżki do plików w macOS-ie i Linuksie. W systemach uniksowych używa się wyłącznie ukośników prawych i zaczyna od ukośnika (/), a nie od nazwy dysku, takiej jak `C:\`. Kiedy wczytujesz plik do Pythona, najprostszym sposobem uniknięcia błędu jest określenie pełnej ścieżki do pliku, jak niżej:

```
import pandas as pd
hour=pd.read_csv('/home/DonQuixote/Documents/hour.csv')
print(hour.head())
```

Kiedy będziesz wykonywał ten kod, zastąp ścieżkę do pliku w metodzie `read_csv()` ścieżką do pliku w Twoim własnym komputerze. Po uruchomieniu powyższego przykładu, ze ścieżką do pliku poprawnie ustawioną na lokalizację pliku *hour.csv* w Twoim komputerze, powinieneś otrzymać następujące wyniki:

```
   instant      dteday  season  yr  ...  windspeed  casual  registered  count
0        1  2011-01-01       1   0  ...        0.0       3          13     16
1        2  2011-01-01       1   0  ...        0.0       8          32     40
2        3  2011-01-01       1   0  ...        0.0       5          27     32
3        4  2011-01-01       1   0  ...        0.0       3          10     13
4        5  2011-01-01       1   0  ...        0.0       0           1      1

[5 rows x 17 columns]
```

Wyniki te pokazują pięć pierwszych wierszy danych. Jak widzisz, dane są ułożone kolumnami w sposób przypominający arkusz kalkulacyjny. Tak jak na rysunku 1.1, każdy wiersz zawiera wartości liczbowe odpowiadające konkretnej godzinie w historii wypożyczalni rowerów.

W tym przykładzie zamiast niektórych kolumn wyświetlone są wielokropki, żeby dane były bardziej czytelne i żeby łatwiej było wycinać je i wklejać do dokumentów tekstowych (zamiast wielokropków mogą być wyświetlone wszystkie kolumny — zależy to od konfiguracji Pythona i pakietu pandas w Twoim komputerze). Podobnie jak wtedy, kiedy otworzyliśmy plik w edytorze arkuszy kalkulacyjnych, możemy zacząć przyglądać się niektórym liczbom, aby dowiedzieć się czegoś o historii firmy i poszukać pomysłów na prowadzenie biznesu.

Obliczanie statystyk zbiorczych

Oprócz samego przyglądania się danym warto ustalić ich najważniejsze atrybuty. Możemy zacząć od obliczenia średniej wartości jednej z kolumn w następujący sposób:

```
print(hour['count'].mean())
```

Uzyskujemy tu dostęp do kolumny `count` w naszym zbiorze danych `hour` przez użycie nawiasów kwadratowych (`[]`) i nazwy kolumny (`count`). Gdybyś wydał polecenie `print(hour['count'])`, zobaczyłbyś całą kolumnę wypisaną na ekranie. Interesuje nas jednak tylko *średnia* wartość kolumny, a nie sama kolumna, więc dodaliśmy metodę `mean()` — kolejną funkcję oferowaną przez pandas. Widzimy, że średnia wynosi około 189,46. Jest to interesująca informacja z perspektywy biznesowej; zapewnia przybliżoną miarę wielkości działalności na przestrzeni dwóch lat opisanych przez dane.

Oprócz obliczenia średniej moglibyśmy obliczyć inne ważne miary w następujący sposób:

```
print(hour['count'].median())
print(hour['count'].std())
print(hour['registered'].min())
print(hour['registered'].max())
```

Obliczamy tu medianę kolumny `count` za pomocą metody `median()`. Używamy też metody `std()` do obliczenia odchylenia standardowego zmiennej `count`. (Może wiesz już, że **odchylenie standardowe** mierzy, jak bardzo rozrzucony jest zbiór liczb. Pomaga zrozumieć godzinową zmienność w liczbie osób jeżdżących rowerami). Obliczamy również minimum i maksimum zmiennej `registered` odpowiednio z wykorzystaniem metod `min()` i `max()`. Liczba zarejestrowanych użytkowników zmienia się w zakresie od 0 do 886, co informuje nas o godzinowym rekordzie, który trzeba będzie pobić, aby poprawić wyniki firmy.

Te proste obliczenia to **statystyki zbiorcze**, które pomogą Ci zbadać dowolny zbiór danych, z jakim przyjdzie Ci pracować. Sprawdzenie statystyk zbiorczych zbioru danych pozwala lepiej zrozumieć dane, a w tym przypadku — lepiej zrozumieć działalność firmy.

Choć statystyki te wydają się proste, wielu dyrektorów, gdyby nagle ich o to zapytać, nie potrafiłoby podać nawet dokładnej liczby klientów firmy. Znajomość prostych miar, takich jak średnia liczba klientów w dowolnej godzinie dowolnego dnia, pomaga zrozumieć, jak duża jest firma i ile ma miejsca na wzrost.

Statystyki zbiorcze można połączyć z innymi informacjami, aby dowiedzieć się jeszcze więcej. Jeśli na przykład sprawdzisz, ile Twoja firma liczy sobie za godzinę użytkowania roweru, będziesz mógł pomnożyć tę stawkę przez średnią kolumny `count`, aby poznać łączne przychody w ciągu dwóch lat opisanych danymi.

Możesz ręcznie sprawdzać statystyki zbiorcze za pomocą takich metod pandas jak `mean()` i `median()`, jak zrobiliśmy to wcześniej. Istnieje jednak inna metoda, która ułatwia to zadanie:

```
print(hour.describe())
```

Używamy tu metody `describe()`, aby sprawdzić statystyki zbiorcze wszystkich zmiennych w zbiorze danych. Jej wyniki wyglądają następująco:

```
          instant        season  ...    registered         count
count  17379.0000  17379.000000  ...  17379.000000  17379.000000
mean    8690.0000      2.501640  ...    153.786869    189.463088
std     5017.0295      1.106918  ...    151.357286    181.387599
min        1.0000      1.000000  ...      0.000000      1.000000
25%     4345.5000      2.000000  ...     34.000000     40.000000
50%     8690.0000      3.000000  ...    115.000000    142.000000
75%    13034.5000      3.000000  ...    220.000000    281.000000
max    17379.0000      4.000000  ...    886.000000    977.000000

[8 rows x 16 columns]
```

Jak widzisz, metoda `describe()` wyświetla całą tabelę, a tabela ta zawiera kilka użytecznych miar, w tym średnią, minimum i maksimum każdej z naszych zmiennych. Wyniki `describe()` zawierają też percentyle. Na przykład wiersz `25%` zawiera 25. percentyl każdej zmiennej w zbiorze danych `hour`. Widzimy, że 25. percentyl zmiennej `count` wynosi 40, co oznacza, że 25 proc. godzin w naszym zbiorze danych miało 40 lub mniej użytkowników, a 75 proc. miało więcej niż 40 użytkowników.

Tabela wyświetlana przez metodę `describe()` przydaje się też do odkrywania problemów z danymi. Zbiory danych często zawierają poważne błędy, które można dostrzec w wynikach `describe()`. Jeśli na przykład wykonasz `describe()` na zbiorze osób i zobaczysz, że ich średni wiek wynosi 200, będziesz wiedział, że Twoje dane zawierają błędy. Może się to wydawać oczywiste, ale dokładnie ten błąd (średnie wieku wyższe niż 200) znaleziono niedawno w dobrze znanej pracy badawczej opublikowanej w prestiżowym czasopiśmie akademickim — gdyby tylko ci badacze użyli metody `describe()`! Powinieneś się przyjrzeć wynikom `describe()` dla każdego zbioru danych, z którym będziesz pracować, aby upewnić się, że wszystkie wartości mają chociaż pozory prawdopodobieństwa. Jeśli znajdziesz średnie wieku wyższe niż 200 albo inne dane, które nie wyglądają wiarygodnie, będziesz musiał zlokalizować problemy w danych i jakoś je naprawić.

Już w tym momencie to, czego dowiedzieliśmy się z danych, może nasunąć nam pomysły na rozwijanie działalności firmy. Widzieliśmy na przykład, że w pierwszych 24 godzinach danych liczba rowerzystów w nocy jest znacznie niższa niż za dnia. Widzieliśmy też dużą godzinową zmienność liczby użytkowników: 25 proc. godzin ma mniej niż 40 rowerzystów, a w jednej godzinie było ich 886. Jako prezes chciałbyś widzieć więcej godzin, w których liczba rowerzystów jest bliska 886, a mniej godzin, w których jest niższa od 40.

Mógłbyś zrealizować ten cel na wiele sposobów, na przykład obniżyć ceny w nocy, aby zyskać więcej klientów o tej porze, a przez to ograniczyć liczbę godzin z małą liczbą rowerzystów. Dalsza eksploracja danych może nasunąć Ci kolejne pomysły na rozwój biznesu.

Analizowanie podzbiorów danych

Sprawdziliśmy statystyki zbiorcze pełnego zbioru danych, a następnie rozważyliśmy zaoferowanie niższych cen w porze nocnej, aby zwiększyć liczbę przejazdów w nocy. Jeśli naprawdę chcielibyśmy zrealizować ten pomysł, powinniśmy sprawdzić statystyki zbiorcze dotyczące tylko pory nocnej.

Dane nocne

Na początek możemy użyć metody `loc()`:

```
print(hour.loc[3,'count'])
```

Metoda `loc()` pozwala nam wybrać podzbiór pełnych danych. Podzbiór, który chcemy wybrać, określa się w nawiasie kwadratowym w następującej postaci: [*<wiersz>*,*<kolumna>*]. W tym przykładzie podaliśmy `[3,'count']`, co oznacza, że chcemy wybrać wiersz 3. i kolumnę `count`. Polecenie to wypisuje `13`, a jeśli przyjrzysz się danym na rysunku 1.1 lub 1.2, przekonasz się, że jest to poprawny wynik.

Trzeba tu podkreślić, że w Pythonie, a także w pakiecie pandas, standardowo używa się **indeksowania od zera**. Liczymy od zera, więc jeśli nasz zbiór danych ma cztery wiersze, oznaczamy je jako wiersz nr 0, wiersz nr 1, wiersz nr 2 i wiersz nr 3. Czwarty wiersz danych nazywamy zatem wierszem nr 3 albo mówimy, że jego indeks to 3. Podobnie trzeci wiersz danych ma indeks 2, drugi ma indeks 1, a pierwszy ma indeks 0. Dlatego kiedy wydajemy polecenie `print(hour .loc[3,'count'])`, uzyskujemy liczbę 13, czyli czwartą wartość przechowywaną w danych (wartość z wiersza o indeksie 3), a nie 32, czyli trzecią wartość przechowywaną w danych (wartość z wiersza o indeksie 2). Wielu osobom indeksowanie od zera wydaje się nienaturalne, ale z czasem przyzwyczaisz się do niego i nie będziesz mieć z tym żadnych problemów.

W poprzednim przykładzie przyjrzeliśmy się podzbiorowi złożonemu z jednej liczby (wartości z jednego wiersza i jednej kolumny). Może jednak interesować Cię podzbiór złożony z wielu wierszy lub wielu kolumn. Za pomocą dwukropka (`:`) możemy określić zakres wierszy, które chcemy zobaczyć:

```
print(hour.loc[2:4,'registered'])
```

W tym fragmencie kodu określamy, że chcemy otrzymać wartości zmiennej `registered`. Podając `2:4` w nawiasie kwadratowym, wskazujemy, że potrzebujemy wierszy od 2. do 4., więc w wynikach otrzymujemy trzy liczby: 27, 10 i 1. Jeśli przyjrzysz się tym wierszom, zobaczysz, że obserwacje te dotyczą godzin 2.00, 3.00 i 4.00 w nocy. Zamiast wypisywać wszystkie dane, wypisaliśmy trzy wiersze. Ponieważ wypisujemy tylko podzbiór danych, możemy nazwać ten proces **wybieraniem podzbioru**. Może to być przydatne podczas eksploracji i analizy danych.

Zamiast przyglądać się kilku sąsiednim wierszom, spójrzmy na wszystkie nocne obserwacje w naszych danych. W metodzie `loc()` możemy użyć warunków logicznych:

```
print(hour.loc[hour['hr']<5,'registered'].mean())
```

W tym przykładzie używamy metody `loc()`, aby uzyskać dostęp do podzbioru danych, jak robiliśmy to wcześniej. Zamiast jednak określać konkretne numery wierszy, określamy warunek logiczny: `hour['hr']<5`; oznacza to, że wybieramy każdy wiersz danych, w którym wartość zmiennej `hr` jest mniejsza niż 5. W ten sposób uzyskujemy podzbiór danych odpowiadający najwcześniejszym godzinom dnia (od północy do 4.00 nad ranem). Możemy określić wiele warunków, aby użyć bardziej złożonej logiki. Aby na przykład sprawdzić liczbę użytkowników podczas zimniejszych nocy w porównaniu z cieplejszymi nocami, możemy napisać:

```
print(hour.loc[(hour['hr']<5) & (hour['temp']<.50),'count'].mean())
print(hour.loc[(hour['hr']<5) & (hour['temp']>.50),'count'].mean())
```

W przykładach tych określamy wiele warunków logicznych oddzielonych znakiem `&`, który znaczy `AND`, czyli wskazuje, że oba warunki muszą być spełnione jednocześnie. Pierwsze polecenie wybiera wiersze, które mają wartość `hr` mniejszą niż 5 i wartość `temp` mniejszą niż 0,50. W tym zbiorze danych zmienna `temp` oznacza temperaturę, ale nie w znajomych skalach Fahrenheita lub Celsjusza. Zamiast tego zbiór używa specjalnej skali, w której wszystkie temperatury należą do zakresu od 0 do 1, gdzie 0 oznacza bardzo niską, a 1 — bardzo wysoką temperaturę. Kiedykolwiek pracujesz z danymi, musisz się upewnić, że znasz jednostki każdej zmiennej. Określamy `hour['temp']<.50`, aby wybrać godziny z niższymi temperaturami, i `hour['temp']>.50`, aby wybrać godziny z cieplejszymi temperaturami. Te dwa polecenia pozwalają nam porównać średnią liczbę użytkowników podczas zimnych nocy z liczbą użytkowników podczas ciepłych nocy.

Możemy też użyć symbolu `|`, który oznacza `OR`. Może to być przydatne w przykładzie takim jak ten:

```
print(hour.loc[(hour['temp']>0.5) | (hour['hum']>0.5),'count'].mean())
```

Wiersz ten wypisuje średnią liczbę użytkowników dla wierszy, które mają albo wysokie temperatury, albo wysoką wilgotność — spełniony musi być tylko jeden z warunków. Możliwość określania złożonych warunków może pomóc Ci w znalezieniu sposobów na zwiększenie liczby przejazdów przy niekomfortowej pogodzie.

Dane sezonowe

Nocne zniżki to nie jedyna możliwa strategia zwiększania liczby przejazdów i przychodów. Możesz również rozważyć wprowadzenie specjalnych ofert w pewnych sezonach albo porach roku. W naszych danych zmienna `season` ma wartość 1 dla zimy, 2 dla wiosny,

3 dla lata i 4 dla jesieni. Możemy użyć metody `groupby()`, aby ustalić średnią liczbę użytkowników podczas każdej z tych pór roku:

```
print(hour.groupby(['season'])['count'].mean())
```

Znaczna część tego kodu powinna wyglądać znajomo. Używamy metody `print()`, aby przyjrzeć się miarom związanym z danymi `hour`. Używamy metody `mean()`, co oznacza, że interesują nas średnie. Korzystamy też z `['count']`, aby uzyskać dostęp do kolumny `count`. Wiadomo już zatem, że będziemy przyglądać się średnim liczbom użytkowników w naszych danych.

Jedynym nowym elementem jest `groupby(['season'])`. Jest to metoda, która dzieli dane na grupy — w tym przypadku tworzy jedną grupę na każdą unikatową wartość, która występuje w kolumnie `season`. Wyniki pokazują nam średnią liczbę użytkowników w poszczególnych porach roku:

```
season
1    111.114569
2    208.344069
3    236.016237
4    198.868856
Name: count, dtype: float64
```

Interpretacja tych wyników jest łatwa: w pierwszej porze roku (zimą) średnia godzinowa liczba użytkowników wynosi około 111,115; w drugiej porze roku (wiosną) wynosi 208,344 itd. Widać wyraźny wzorzec sezonowy: wyższa liczba użytkowników wiosną i latem oraz niższa jesienią i zimą. Metoda `groupby()` może również grupować dane według wielu kolumn, jak niżej:

```
print(hour.groupby(['season','holiday'])['count'].mean())
```

Wyniki są następujące:

```
season  holiday
1       0          112.685875
        1           72.042683
2       0          208.428472
        1          204.552083
3       0          235.976818
        1          237.822917
4       0          199.965998
        1          167.722222
Name: count, dtype: float64
```

W tym przykładzie określamy dwie kolumny, według których mają być grupowane dane: `season` i `holiday`. Dzieli to dane godzinowe na cztery pory roku, a następnie dzieli każdą porę roku na dni świąteczne (oznaczone jedynkami) i nieświąteczne (oznaczone zerami). Polecenie pokazuje nam średnią liczbę użytkowników w dni świąteczne i nieświąteczne dla każdej pory roku. Dzięki temu widzimy różnice między dniami świątecznymi i nieświątecznymi w ujęciu sezonowym. Wydaje się, że w zimniejszych porach roku liczba użytkowników podczas dni świątecznych jest niższa niż podczas dni nieświątecznych, a w cieplejszych porach roku liczby te są mniej więcej takie same. Znajomość takich różnic może pomóc Ci w podejmowaniu decyzji dotyczących prowadzenia firmy i zasugerować różne strategie, które możesz realizować podczas różnych pór roku lub świąt.

Nasz zbiór danych jest duży i można badać go na nieskończoną liczbę sposobów. Dotychczas przyjrzeliśmy się kilku podzbiorom i wpadliśmy na kilka pomysłów. Mógłbyś zrobić znacznie więcej: zbadać podzbiory związane z wszystkimi kolumnami i przyjrzeć się danym z wielu różnych perspektyw. Nawet bez zaawansowanej statystyki i uczenia maszynowego możesz się dowiedzieć bardzo dużo i poczynić wiele użytecznych spostrzeżeń.

Wizualizacja danych za pomocą Matplotlib

Statystyki zbiorcze są bardzo cenne i przydatne w eksploracji danych. Istnieje jednak niezwykle ważna część eksploracyjnej analizy danych, o której jeszcze nie mówiliśmy: wizualizowanie danych na wykresach.

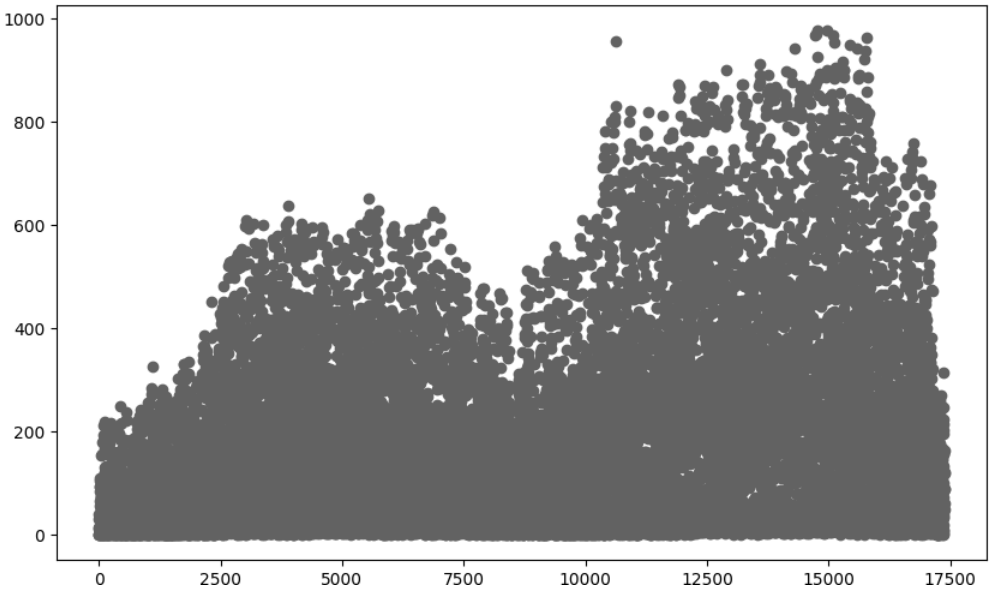
Rysowanie i wyświetlanie prostego wykresu

Kiedy analizujesz dane, powinieneś od samego początku regularnie tworzyć wykresy. Użyjemy w tym celu popularnego pakietu o nazwie Matplotlib. Prosty wykres naszych danych możemy narysować w następujący sposób:

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(10, 6))
ax.scatter(x = hour['instant'], y = hour['count'])
plt.show()
```

Importujemy tu pakiet Matplotlib, nadając mu alias `plt`. Następnie tworzymy rysunek o nazwie `fig` oraz oś o nazwie `ax`. Rysunek `fig` będzie zawierać wszystkie informacje o wykresie lub grupie wykresów, które narysujemy. Oś `ax` zapewni nam dostęp do użytecznych metod służących do rzeczywistego rysowania wykresów. Metoda `subplots()` tworzy dla nas oba te obiekty i pozwala określić rozmiar rysunku (`figsize`). W tym przypadku określamy rozmiar `(10,6)`, co oznacza, że rysunek będzie mieć szerokość 10 cali i wysokość 6 cali.

Następnie rysujemy wykres za pomocą metody `scatter()`. Przekazujemy do niej parametr `x=hour['instant']`, żeby oś *x* pokazywała zmienną `instant` w naszych danych `hour`. Przekazujemy też parametr `y=hour['count']`, żeby oś *y* pokazywała zmienną `count`. Wreszcie, używamy metody `plt.show()` w celu wyświetlenia wykresu na ekranie. Przykład ten tworzy wykres, który powinien wyglądać tak jak na rysunku 1.3.



Rysunek 1.3. Godzinowe liczby użytkowników na przestrzeni dwóch lat

Na tym wykresie każdy punkt reprezentuje godzinę, której charakterystyka jest zarejestrowana w zbiorze danych. Pierwsza godzina (początek 2011 r.) pojawia się po lewej skrajnej stronie wykresu. Ostatnia godzina (koniec 2012 r.) pojawia się po prawej skrajnej stronie wykresu, a wszystkie pozostałe godziny są pokazane kolejno pośrodku.

Ten wykres, zwany **wykresem punktowym**, jest dobrą pierwszą wizualizacją, ponieważ pokazuje wszystkie obserwacje w danych; ułatwia też wizualną identyfikację zależności. W tym przypadku widzimy pełną reprezentację sezonowej zmienności, którą zasugerowała wcześniejsza instrukcja `groupby()`. Widzimy też ogólny wzrost liczby użytkowników z biegiem czasu.

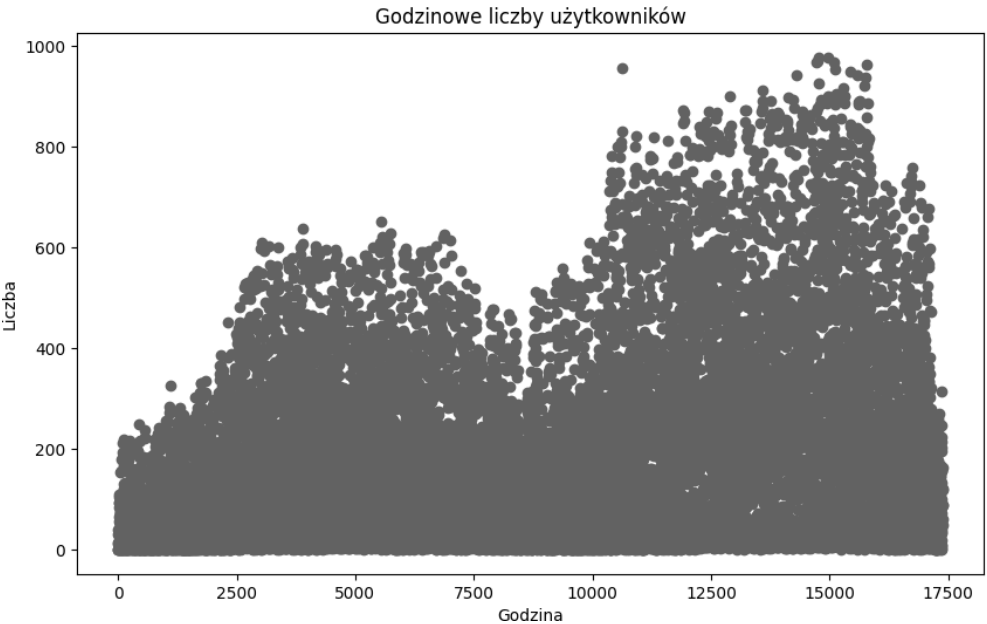
Tytuły i etykiety na wykresach

Wykres z rysunku 1.3 pokazuje dane, ale nie jest tak klarowny, jak być powinien. Możemy dodać do wykresu tytuł i etykiety w następujący sposób:

```
fig, ax = plt.subplots(figsize=(10, 6))
ax.scatter(x = hour['instant'], y = hour['count'])
plt.xlabel("Godzina")
```

```
plt.ylabel("Liczba")
plt.title("Godzinowe liczby użytkowników")
plt.show()
```

Ten fragment kodu używa metody `xlabel()` w celu dodania etykiety do osi *x*, `ylabel()` do dodania etykiety do osi *y* oraz `title()` w celu dodania tytułu wykresu. W metodach tych możesz podać dowolny tekst, aby wyświetlić żądane etykiety. Wynik powinien wyglądać tak, jak pokazano na rysunku 1.4.



Rysunek 1.4. Liczba użytkowników w ujęciu godzinowym, z tytułem i etykietami osi

Nasz zbiór danych jest bardzo duży, więc przyglądanie się wszystkim danym jednocześnie jest trudne. Zobaczmy więc, jak kreślić mniejsze podzbiory naszych danych.

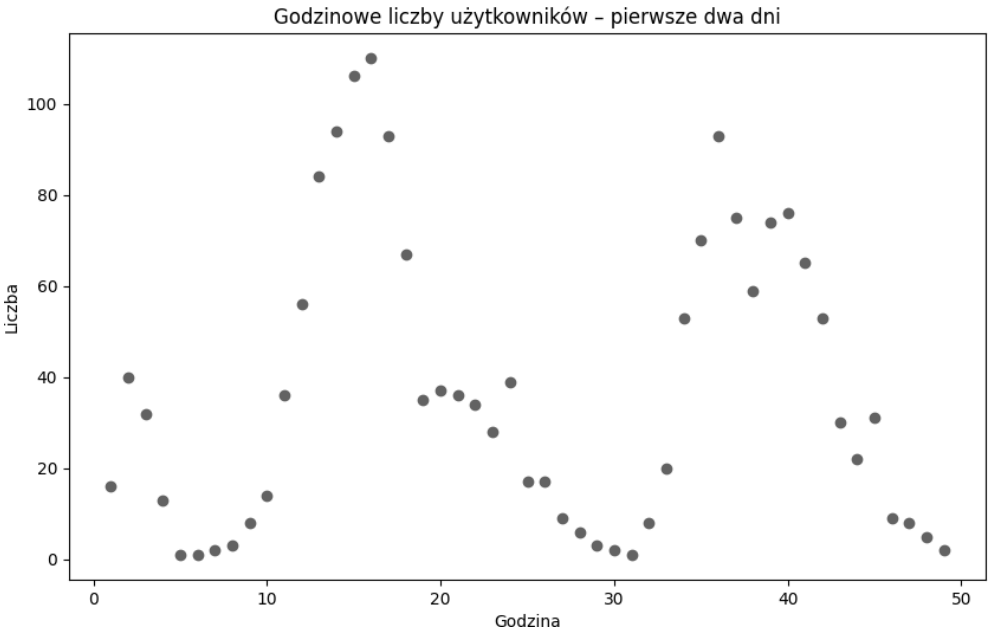
Kreślenie podzbiorów danych

Możemy skorzystać z wybierania podzbiorów, jak zrobiliśmy to wcześniej, aby wykreślić tylko pewien podzbiór danych:

```
hour_first48 = hour.loc[0:48,:]
fig, ax = plt.subplots(figsize=(10, 6))
ax.scatter(x = hour_first48['instant'], y = hour_first48['count'])
plt.xlabel("Godzina")
plt.ylabel("Liczba")
plt.title("Godzinowe liczby użytkowników – pierwsze dwa dni")
plt.show()
```

Definiujemy tu nową zmienną o nazwie `hour_first48`. Zmienna ta zawiera dane dotyczące wierszy od 0 do 48 w pierwotnych danych, odpowiadających mniej więcej dwóm pełnym pierwszym dniom.

Zauważ, że wybieramy ten podzbiór, pisząc `hour.loc[0:48,:]`. Jest to ta sama metoda `loc()`, której używaliśmy wcześniej. Skorzystaliśmy z notacji `0:48`, aby określić, że interesują nas wiersze o indeksach do 48, ale nie określiliśmy żadnych kolumn — wpisaliśmy po prostu dwukropek (`:`) tam, gdzie zwykle podalibyśmy nazwy kolumn. Jest to przydatny skrót: sam dwukropek informuje pakiet pandas, że chcemy wybrać wszystkie kolumny zbioru danych, więc nie musimy osobno wpisywać każdej nazwy. Wykres tego podzbioru wygląda tak jak na rysunku 1.5.



Rysunek 1.5. Liczba użytkowników w ujęciu godzinowym, dane z pierwszych dwóch dni

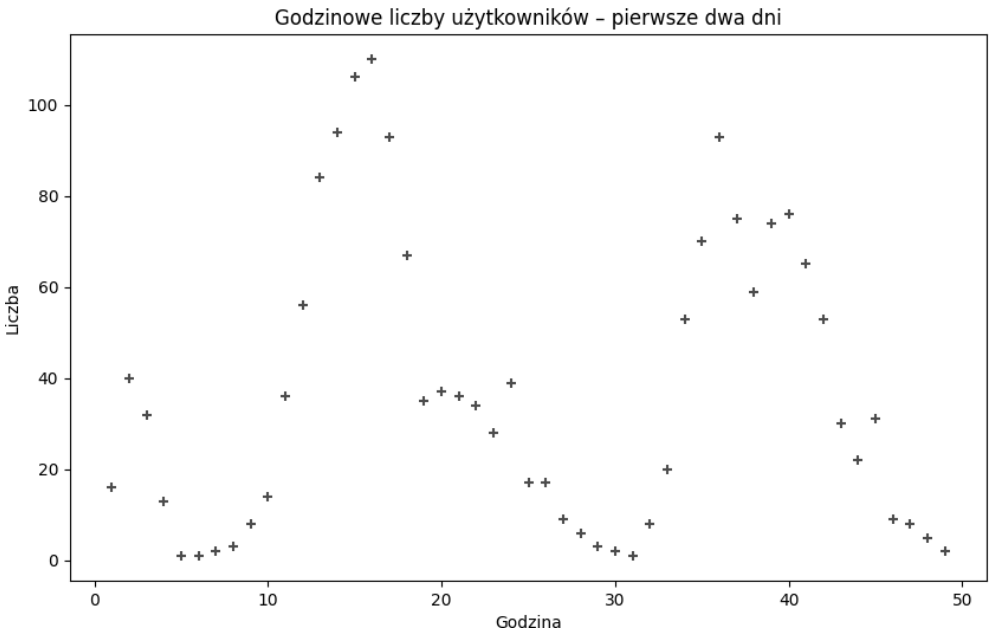
Kreśląc tylko dwa dni zamiast dwóch lat danych, unikamy problemu nakładania się punktów. Każdą obserwację widać znacznie wyraźniej. Kiedy masz duży zbiór danych, warto zrobić jedno i drugie: wykreślić cały zbiór danych (żeby zobaczyć ogólne wzorce) oraz mniejsze podzbiory danych (żeby zrozumieć pojedyncze obserwacje i wzorce występujące w mniejszej skali). W tym przypadku widzimy wzorce dzienne obok długoterminowych, sezonowych wzorców rocznych.

Testowanie różnych typów wykresów

Wygląd wykresu można zmieniać na wiele różnych sposobów. Funkcja `scatter()` przyjmuje parametry, które możemy zmodyfikować, aby uzyskać inny wygląd:

```
fig, ax = plt.subplots(figsize=(10, 6))
ax.scatter(x = hour_first48['instant'], y = hour_first48['count'],c='red',marker='+')
plt.xlabel("Godzina")
plt.ylabel("Liczba")
plt.title("Godzinowe liczby użytkowników – pierwsze dwa dni")
plt.show()
```

Używamy tu argumentu `c`, aby zmienić kolor punktów na wykresie (na czerwony). Korzystamy też z argumentu `marker`, aby zmienić **styl znaczników**, czyli kształt rysowanych punktów. Podając argument `marker='+'`, uzyskujemy punkty, które wyglądają jak małe plusy zamiast małych kropek. Wynik pokazano na rysunku 1.6.



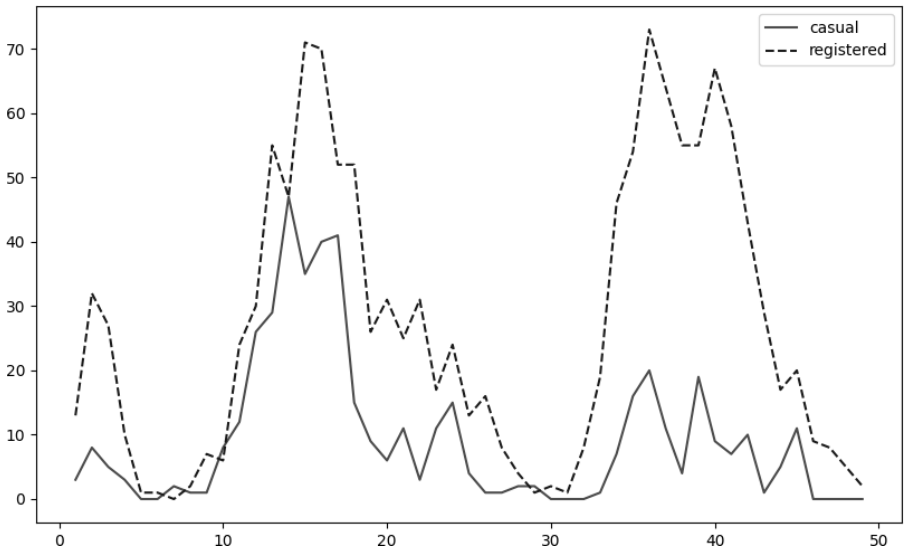
Rysunek 1.6. Liczba użytkowników w ujęciu godzinowym, z różnymi opcjami stylu

Książka nie jest wydrukowana w kolorze, więc czerwonego koloru nie widać na rysunku. Powinieneś jednak zobaczyć czerwone plusy, jeśli wykonasz ten kod na swoim komputerze.

Wykres punktowy nie jest jedynym dostępnym typem wykresu. Wypróbujmy **wykres liniowy**:

```
fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(hour_first48['instant'], hour_first48['casual'], c='red', label='casual', linestyle='-')
ax.plot(hour_first48['instant'],hour_first48['registered'], c='blue', label='registered',
↪linestyle='--')
ax.legend()
plt.show()
```

W tym przypadku używamy metody `ax.plot()` zamiast `ax.scatter()` do narysowania wykresu. Metoda `ax.plot()` umożliwia narysowanie wykresu liniowego. Metodę `ax.plot()` wywołujemy dwukrotnie, aby narysować dwie linie na jednym wykresie. Pozwala nam to porównać użytkowników okazjonalnych i zarejestrowanych (patrz rysunek 1.7).



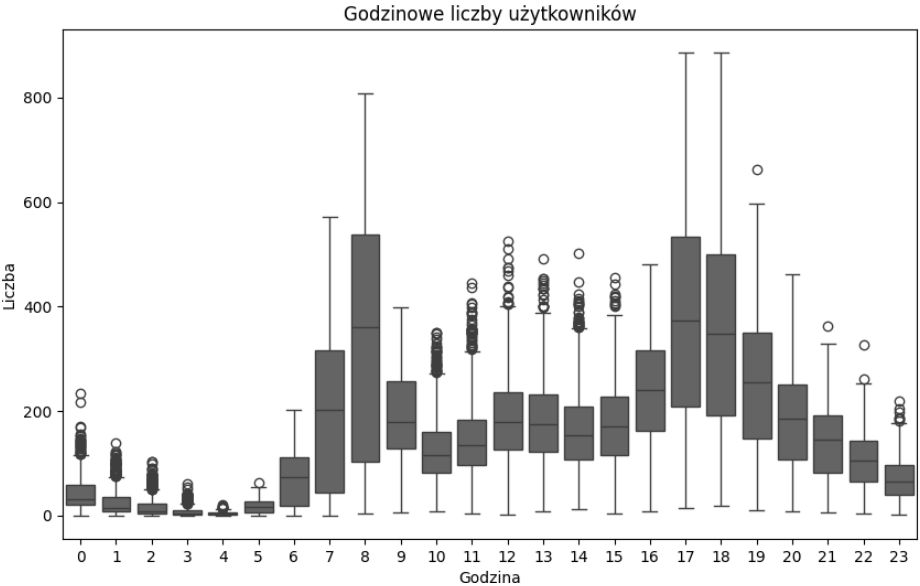
Rysunek 1.7. Wykres liniowy pokazujący użytkowników okazjonalnych i zarejestrowanych w ciągu pierwszych dwóch dni

Wykres ten pokazuje, że liczba rowerzystów okazjonalnych jest niemal zawsze niższa niż liczba rowerzystów zarejestrowanych. Legenda wskazuje zarówno kolory, jak i style linii dla różnych typów użytkowników (linia pełna dla okazjonalnych, przerywana dla zarejestrowanych). Wykonaj ten kod w swoim komputerze, aby wyraźniej zobaczyć kolory i ich kontrast.

Możemy też wypróbować inny rodzaj wykresu:

```
import seaborn as sns
fig, ax = plt.subplots(figsize=(10, 6))
sns.boxplot(x='hr', y='registered', data=hour)
plt.xlabel("Godzina")
plt.ylabel("Liczba")
plt.title("Godzinowe liczby użytkowników")
plt.show()
```

Tym razem importujemy pakiet o nazwie **seaborn**. Pakiet ten jest oparty na Matplotlib, więc oferuje wszystkie jego możliwości, a także dodatkowe funkcje, które pomagają szybko tworzyć estetyczne, pouczające wykresy. Używamy metody `boxplot()` z pakietu seaborn do narysowania nowego rodzaju wykresu: **wykresu skrzynkowego**. Wynik powyższego fragmentu kodu pokazano na rysunku 1.8.



Rysunek 1.8. Wykres skrzynkowy pokazujący liczby rowerzystów pogrupowane według godziny dnia

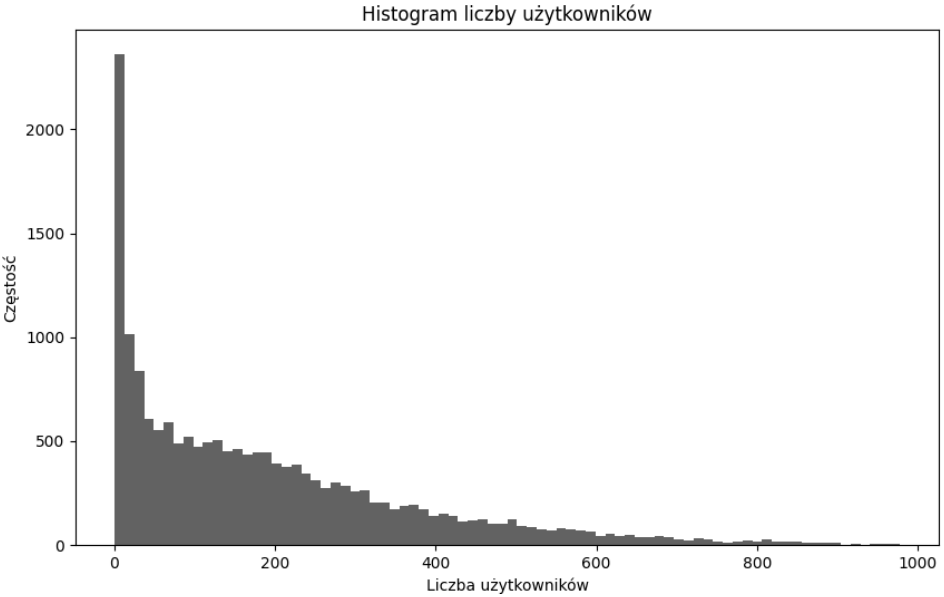
Wykres pokazuje 24 prostokąty umieszczone jeden obok drugiego — każdy reprezentuje informacje o konkretnej godzinie dnia. Wykres skrzynkowy jest prosty, ale dostarcza mnóstwa informacji. Górne i dolne boki każdego prostokąta („skrzynki") reprezentują odpowiednio 75. i 25. percentyl wykreślonych danych. Pozioma linia wewnątrz prostokąta reprezentuje medianę (czyli 50. percentyl). Pionowe linie wychodzące z góry i dołu każdego prostokąta reprezentują pełen zakres obserwacji, które nie są uważane za wartości odstające. Osobne punkty poza zakresem pionowych linii są uważane za wartości odstające.

Skrzynki widoczne na rysunku 1.8 pozwalają porównać liczby rowerzystów o różnych porach dnia. Na przykład mediana liczby kierowców podczas godziny 5.00 (około 5.00 rano) jest dość niska, ale podczas godziny 6.00 (około 6.00 rano) jest znacznie wyższa. Rośnie jeszcze bardziej podczas godziny 7.00 (około 7.00 rano). Duża liczba kierowców pojawia się ponownie w okolicach godziny 5.00 i 6.00 po południu; może te szczyty oznaczają, że wielu Twoich klientów dojeżdża rowerami do pracy, a potem wraca nimi do domu.

Jak się zapewne domyślasz, możemy rysować znacznie więcej typów wykresów. Innym przydatnym rodzajem wykresu jest **histogram**, który można utworzyć w następujący sposób:

```
fig, ax = plt.subplots(figsize=(10, 6))
ax.hist(hour['count'],bins=80)
plt.xlabel("Liczba użytkowników")
plt.ylabel("Częstość")
plt.title("Histogram liczby użytkowników")
plt.show()
```

Powyższy fragment kodu używa metody `hist()` do narysowania histogramu. Wynik pokazano na rysunku 1.9.



Rysunek 1.9. Histogram pokazujący częstość występowania każdej liczby użytkowników

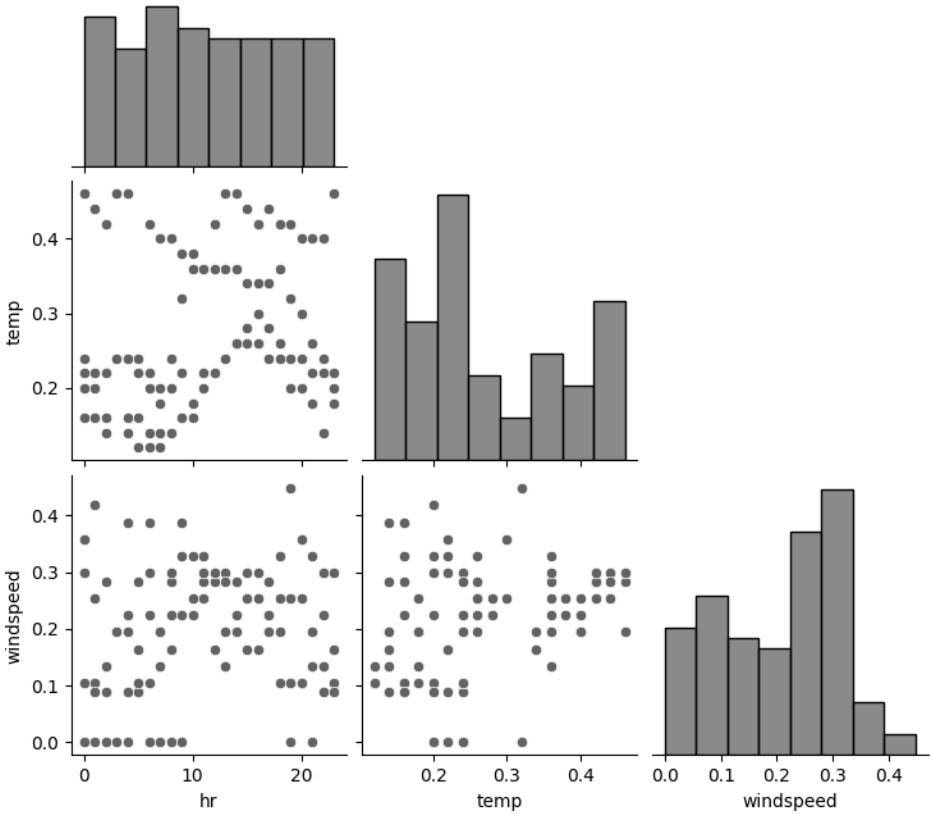
Na histogramie wysokość każdego słupka reprezentuje częstość. W tym przypadku histogram pokazuje częstości występowania każdej liczby rowerzystów. Jeśli na przykład spojrzysz na oś *x* w okolicach wartości 800, zobaczysz słupki o wysokości bliskiej zeru. Oznacza to, że bardzo niewiele godzin w naszym zbiorze danych miało około 800 rowerzystów. Natomiast przy wartości 200 na osi *x* widać wyższe słupki, o wysokości zbliżonej do 500. Wskazuje to, że dla około 500 godzin w naszych danych liczba rowerzystów była zbliżona do 200. Wzorzec widoczny w tym histogramie jest typowy dla wielu firm: wiele godzin z niewieloma klientami i niewiele godzin z wieloma klientami.

Mógłbyś użyć tego histogramu, aby rozważyć „moce przerobowe" swojej firmy. Przypuśćmy, że obecnie firma ma do wypożyczenia 1000 rowerów. Mógłbyś zaoszczędzić pieniądze, sprzedając 200 rowerów — w ten sposób zarobiłbyś trochę dodatkowej gotówki i nie musiałbyś się martwić o przechowywanie i konserwację niepotrzebnych rowerów. Zostałoby Ci wówczas 800 rowerów do wypożyczenia. Przyglądając się histogramowi, widzisz dokładnie, jak taka zmiana wpłynęłaby na Twoją firmę: ponieważ popyt przekraczający 800 rowerów występuje w małym odsetku godzin, wpływ zmiany powinien być względnie niewielki. Histogram pomoże Ci zdecydować, ile dokładnie rowerów możesz bezpiecznie sprzedać.

Inny rodzaj wykresu, **wykres par**, rysuje jeden wykres punktowy dla każdej możliwej pary zmiennych w Twoich danych:

```
thevariables=['hr', 'temp', 'windspeed']
hour_first100=hour.loc[0:100, thevariables]
sns.pairplot(hour_first100, corner=True)
plt.show()
```

Utworzyliśmy tu zmienną `thevariables` — listę trzech zmiennych, które chcemy wykreślić (wykreślamy tylko trzy zmienne ze względu na ograniczoną ilość miejsca w książce). Utworzyliśmy też zmienną `hour_first100`, która jest podzbiorem pełnych danych zawierającym tylko wiersze o indeksie 100 lub niższym. Ponownie korzystamy z pakietu seaborn i wywołujemy zawartą w nim metodę `pairplot()`, aby utworzyć nasz wykres. Wynik, pokazany na rysunku 1.10, to zbiór wykresów zawierający zarówno wykresy skrzynkowe, jak i histogramy.



Rysunek 1.10. Wykresy par pokazujące zależności między wybranymi zmiennymi

Wykres par pokazuje wykres skrzynkowy dla każdej możliwej kombinacji zmiennych w wybranym przez nas zbiorze danych, a także histogram każdej zmiennej. Na rysunku widać mnóstwo danych, ale wykresy skrzynkowe nie pokazują żadnych oczywistych zależności między zmiennymi; związki te wydają się zasadniczo losowe.

Czasem, kiedy rysujemy wykres par, zamiast losowości widzimy jasne relacje między zmiennymi. Gdybyśmy na przykład mieli w naszych danych pomiar opadów śniegu, zobaczylibyśmy, że w miarę wzrostu temperatury maleją opady śniegu (i odwrotnie). Taki typ zależności między zmiennymi nazywa się **korelacją**; zbadamy go w następnym podrozdziale.

Eksplorowanie korelacji

Dwie zmienne są *skorelowane*, jeśli zmiana jednej zmiennej występuje jednocześnie ze zmianą drugiej. Mówimy, że zmienne są skorelowane dodatnio, jeśli zmieniają się *tak samo*: kiedy rośnie jedna, rośnie druga, a kiedy jedna maleje, druga maleje razem z nią. W rzeczywistym świecie znajdziemy niezliczone przykłady korelacji dodatniej. Liczba kotów domowych w mieście jest skorelowana dodatnio z ilością karmy dla kotów kupowanej w tym mieście. Jeśli jedna z tych zmiennych jest wysoka, druga również zwykle jest wysoka, a jeśli jedna z nich jest niska, druga także zwykle jest niska.

Możemy też mówić o korelacjach ujemnych: dwie zmienne są skorelowane ujemnie, jeśli jedna rośnie, kiedy druga maleje, albo jedna maleje, kiedy druga rośnie. Korelacje negatywne również są częste. Na przykład średnia temperatura w mieście jest skorelowana ujemnie ze średnią ilością pieniędzy, które typowy mieszkaniec wydaje co roku na grube zimowe płaszcze. W miastach, w których jedna z tych liczb jest wysoka, druga zwykle jest niska, a w miastach, w których jedna z tych liczb jest niska, druga zwykle jest wysoka.

W świecie data science znajdowanie i rozumienie korelacji — zarówno dodatnich, jak i ujemnych — jest niezwykle istotne. Jeśli nauczysz się odkrywać te korelacje, będziesz odnosić większe sukcesy jako prezes. Możesz na przykład odkryć, że liczba rowerzystów jest skorelowana dodatnio z temperaturą. Oznacza to, że liczba rowerzystów jest niska, kiedy temperatura jest niska. Mógłbyś nawet rozważyć sprzedaż części rowerów podczas pór roku o niskiej liczbie rowerzystów, żeby wygenerować jakieś przepływy pieniężne, zamiast pozwolić, żeby rowery stały bezczynnie. Ostateczna decyzja będzie zależeć od wielu innych szczegółów Twojej sytuacji, ale głębokie zrozumienie danych pomoże Ci podejmować optymalne decyzje biznesowe.

Obliczanie korelacji

Korelacje w Pythonie możemy obliczać w następujący sposób:

```
print(hour['casual'].corr(hour['registered']))
print(hour['temp'].corr(hour['hum']))
```

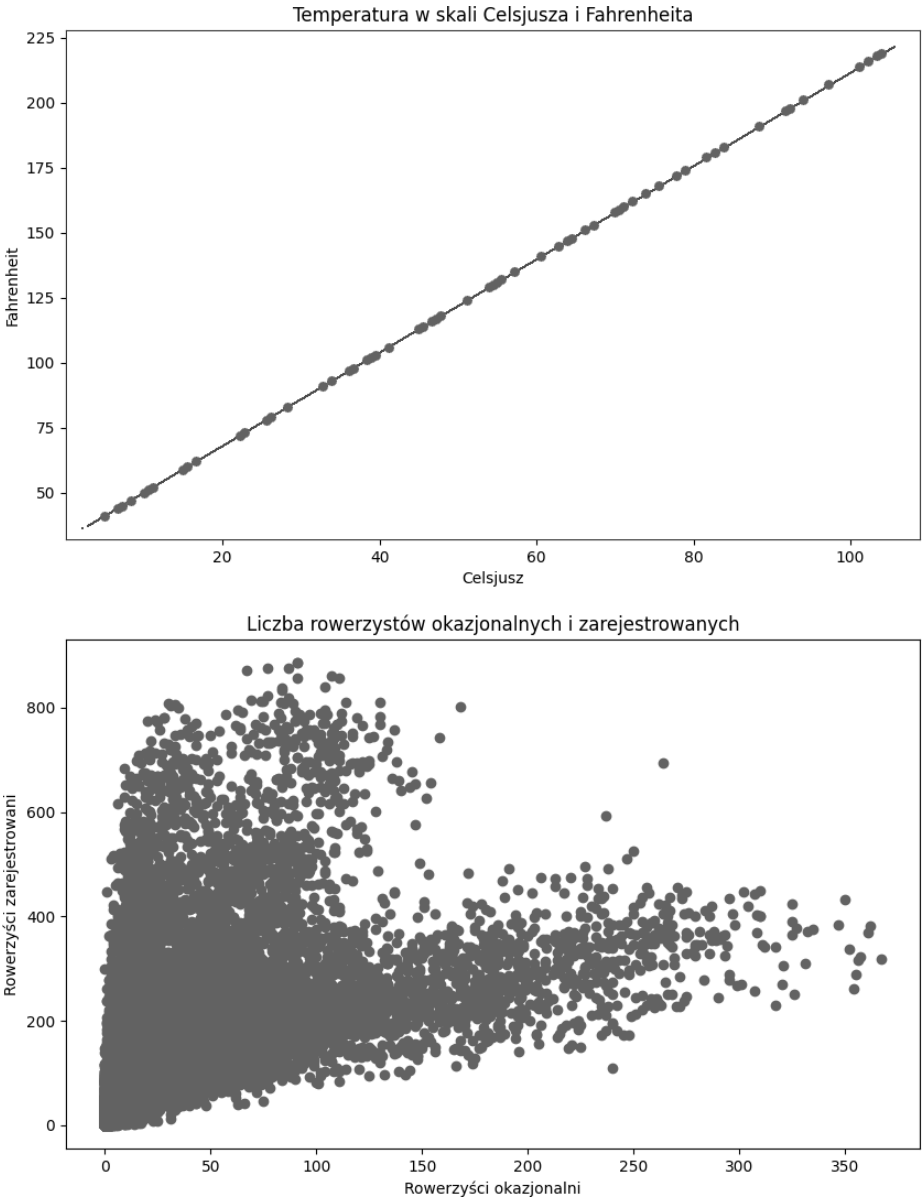
Używamy tu metody `corr()`, kolejnej funkcji z pakietu pandas. Metoda `corr()` oblicza tak zwany **współczynnik korelacji**. Istnieje wiele typów współczynników korelacji, ale domyślnie `corr()` oblicza współczynnik korelacji Pearsona. Jest to najczęściej używany współczynnik korelacji, więc kiedykolwiek w tej książce wspominamy o współczynniku korelacji, mamy na myśli współczynnik korelacji Pearsona.

Współczynnik korelacji Pearsona jest liczbą z zakresu od –1 do 1 i często oznacza się go literą *r*. Opisuje on zależność między dwiema zmiennymi: znak wskazuje typ korelacji, a wielkość określa jej siłę. Jeśli współczynnik korelacji *r* jest liczbą dodatnią, dwie zmienne są skorelowane dodatnio, a jeśli jest liczbą ujemną, zmienne są skorelowane ujemnie. Jeśli współczynnik korelacji wynosi 0 albo jest bardzo bliski zeru, mówimy, że zmienne są *nieskorelowane*.

W powyższym przykładzie pierwsze polecenie oblicza współczynnik korelacji, który opisuje zależność między zmiennymi `casual` i `registered` w naszych danych. Dla tych zmiennych *r* wynosi około 0,51; jest to liczba dodatnia, która wskazuje korelację dodatnią.

Silna i słaba korelacja

Oprócz sprawdzania, czy współczynniki korelacji są dodatnie, ujemne, czy równe zeru, zwracamy też uwagę na ich dokładną *wielkość*. Jeśli współczynnik korelacji jest duży (daleki od zera, a bliski 1 lub –1), często mówimy, że korelacja jest *silna*. Przykłady korelacji pokazano na rysunku 1.11.



Rysunek 1.11. Zmienne skorelowane dodatnio

Na rysunku są dwa wykresy. Pierwszy pokazuje zależność między temperaturami w skali Fahrenheita i Celsjusza. Jak widać, są one skorelowane: kiedy rośnie jedna, rośnie druga i odwrotnie. Drugi wykres pokazuje zależność między okazjonalnymi i zarejestrowanymi klientami Twojej firmy. Ponownie widać korelację dodatnią: kiedy rośnie liczba klientów okazjonalnych, rośnie też liczba klientów zarejestrowanych i odwrotnie.

Obie korelacje z rysunku 1.11 są dodatnie, ale widać jakościową różnicę między nimi. Zależność między temperaturami w skali Fahrenheita i Celsjusza jest deterministyczna: znajomość temperatury w skali Fahrenheita pozwala nam dokładnie ustalić temperaturę w skali Celsjusza, bez niepewności i domysłów. Tego rodzaju deterministyczna korelacja dodatnia, która tworzy linię prostą na wykresie, jest również nazywana **korelacją doskonałą**, a kiedy obliczamy współczynnik doskonałej korelacji dodatniej, uzyskujemy $r = 1$.

Natomiast zależność między użytkownikami okazjonalnymi i zarejestrowanymi *nie* jest deterministyczna. Wyższa liczba użytkowników okazjonalnych często odpowiada wyższej liczbie użytkowników zarejestrowanych, ale nie zawsze: nie możemy idealnie przewidywać jednej zmiennej na podstawie drugiej. Kiedy dwie zmienne są skorelowane, ale ich zależność nie jest deterministyczna, mówimy, że występuje w niej „szum" albo losowość.

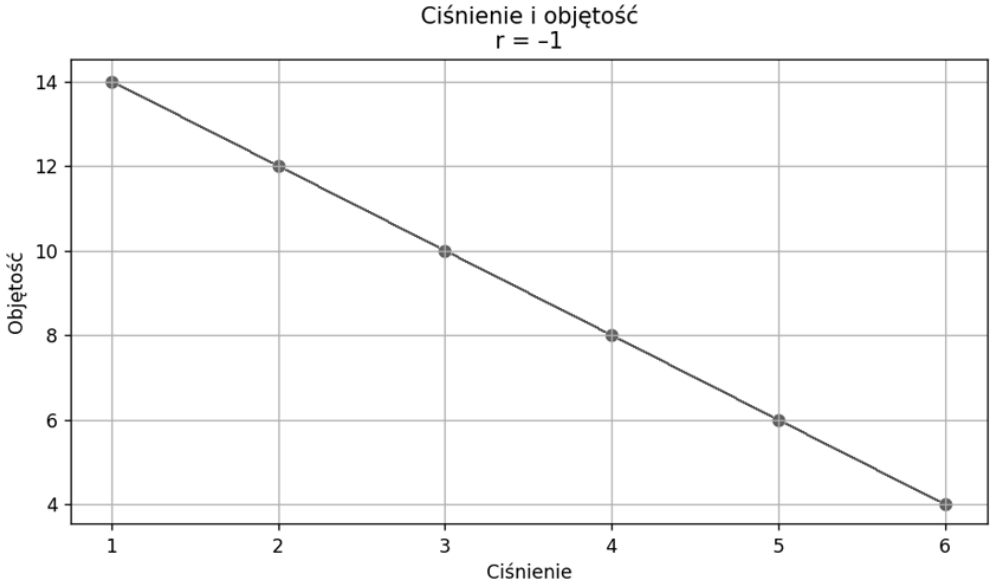
Losowość to koncepcja trudna do precyzyjnego zdefiniowania, ale możesz o niej myśleć jak o nieprzewidywalności. Kiedy znasz temperaturę w skali Fahrenheita, możesz określić temperaturę w skali Celsjusza z idealną dokładnością. Natomiast kiedy znasz liczbę użytkowników okazjonalnych, możesz przewidzieć liczbę użytkowników zarejestrowanych, ale Twoja prognoza może nie być idealnie dokładna. Kiedy istnieje taka nieprzewidywalność, dwie zmienne mają współczynnik korelacji mniejszy od jedności. W tym przypadku, jeśli obliczymy korelację między liczbą użytkowników okazjonalnych i zarejestrowanych, uzyskamy $r = 0{,}51$.

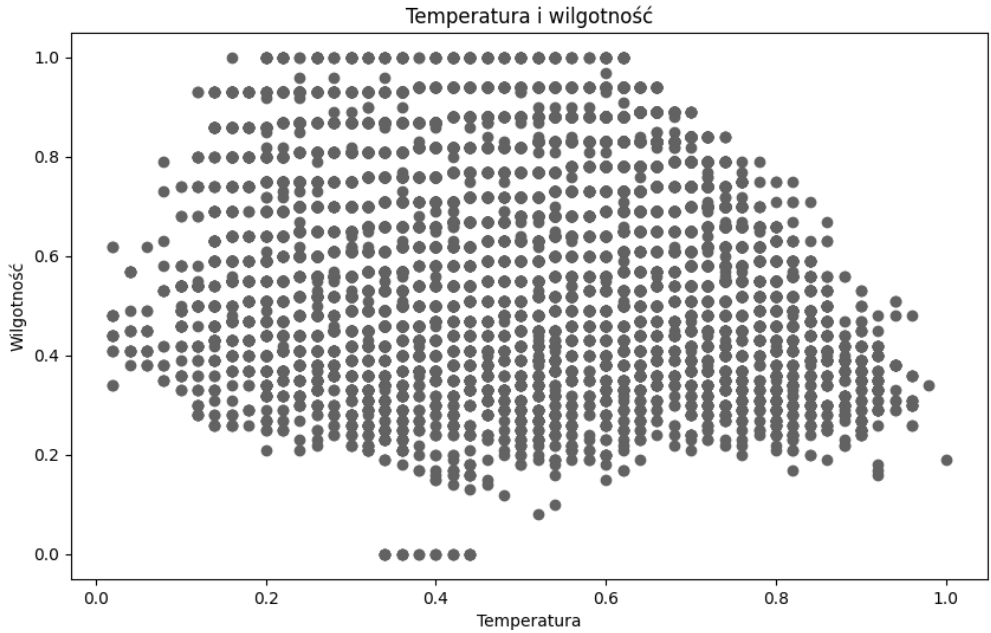
Wielkość współczynnika korelacji możesz traktować jak miarę losowości w zależności między dwiema zmiennymi. Większy współczynnik korelacji odpowiada mniejszej losowości (zależność jest bliższa deterministycznej, jak w przykładzie z temperaturami w skali Fahrenheita i Celsjusza). Mniejszy współczynnik korelacji odpowiada większej losowości i mniejszej przewidywalności. Zerowy współczynnik korelacji, który oznacza całkowity brak zależności między zmiennymi, możesz traktować jak wskaźnik czystej losowości albo czystego szumu.

Przykłady korelacji ujemnych o różnych wielkościach przedstawiono na rysunku 1.12.

Widzimy tu te same koncepcje co na rysunku 1.11. Pierwszy wykres pokazuje *doskonałą* korelację ujemną: tym razem jest to deterministyczna zależność między ciśnieniem a objętością. Współczynnik korelacji wynosi tu dokładnie $r = -1$, co wskazuje, że związek między zmiennymi jest wolny od losowości; każdą zmienną można idealnie przewidzieć na podstawie drugiej.

Drugi wykres pokazuje zależność między temperaturą a wilgotnością w naszych danych. Te dwie zmienne również mają korelację ujemną, ale ze znacznie niższym współczynnikiem; r wynosi około $-0{,}07$. Podobnie jak w przypadku korelacji dodatnich z rysunku 1.11, możemy interpretować te współczynniki korelacji jako miary losowości: współczynnik o większej wielkości (bliższy 1 lub -1) wskazuje korelację wysoce przewidywalną i mało losową, a współczynnik o mniejszej wielkości (bliższy 0) wskazuje korelację bardziej losową. W tym przykładzie współczynnik $r = -0{,}07$ oznacza, że temperatura i wilgotność są skorelowane ujemnie, ale ich korelacja jest bardzo słaba — niedaleka od czystej losowości.





Rysunek 1.12. Zmienne skorelowane ujemnie

Ważną rzeczą, o której należy pamiętać podczas badania korelacji, jest słynne powiedzenie: „Korelacja nie oznacza przyczynowości”. Kiedy obserwujemy silne korelacje, jedyne, czego możemy być pewni, to że dwie zmienne zmieniają się razem; nie możemy być pewni, że jedna „powoduje” drugą.

Przypuśćmy na przykład, że badamy startupy z Doliny Krzemowej i odkrywamy, że ich miesięczne przychody są skorelowane z liczbą kupowanych przez nie stołów pingpongowych. Z tej korelacji moglibyśmy wyciągnąć przedwczesny wniosek, że stoły pingpongowe powodują wzrost przychodów; może zrelaksowani pracownicy stają się bardziej produktywni, a może atmosfera dobrej zabawy pomaga zatrzymać istniejących pracowników i skuteczniej zatrudniać nowych.

Z drugiej strony te przypuszczenia mogą być zupełnie błędne, a przyczynowość może działać w drugą stronę; firmy, które odnoszą sukces (zupełnie niezależnie od ich stołów pingpongowych), mają wyższe przychody, a nagły wzrost budżetu pozwala im wykorzystać dodatkowe pieniądze na zakup stołów pingpongowych. W takim przypadku to przychody powodują zakup stołów pingpongowych, a nie odwrotnie.

Wreszcie, korelacja może być zupełnie przypadkowa. Może stoły pingpongowe nie prowadzą do wyższych przychodów, a przychody nie prowadzą do stołów pingpongowych, a my zaobserwowaliśmy **pozorną korelację** — taką, która występuje przez przypadek i nie wskazuje żadnej przyczynowości ani szczególnej zależności. Korelacja może być również spowodowana **pominiętą zmienną**, czymś, czego nie zaobserwowaliśmy, a co powoduje jednoczesny wzrost przychodów i zakupów stołów pingpongowych.

Tak czy owak, kiedy znajdujesz i interpretujesz korelacje, musisz zawsze zachowywać ostrożność. Korelacja oznacza, że dwie zmienne zmieniają się razem, co może nam pomóc w dokonywaniu prognoz, ale niekoniecznie implikuje, że jedna zmienna powoduje zmiany drugiej albo że w ogóle istnieje między nimi jakakolwiek rzeczywista zależność.

Zrozumienie współczynników korelacji może Ci pomóc w pełnieniu obowiązków prezesa, zwłaszcza jeśli odkryjesz zaskakujące korelacje. Możesz na przykład znaleźć silną korelację dodatnią między rozmiarem grup, które wspólnie wypożyczają rowery, a ich poziomem satysfakcji po wypożyczeniu. Może to nasunąć Ci pomysły na zachęcanie użytkowników do wypożyczania rowerów wraz z przyjaciółmi i zyskania w ten sposób bardziej zadowolonych klientów. Znajdowanie korelacji i rozumienie wielkości korelacji oraz tego, co mówi ona o przewidywalności, mogą być cenne w biznesie.

Znajdowanie korelacji między zmiennymi

Nie jesteśmy ograniczeni do obliczania indywidualnych korelacji między parami zmiennych. Możemy pójść o krok dalej i utworzyć **macierz korelacji**, czyli prostokątną tablicę liczb, w której każdy element jest współczynnikiem korelacji mierzącym zależność między dwiema konkretnymi zmiennymi. Macierz korelacji pokazuje związki między wszystkimi zmiennymi:

```
thenames=['hr','temp','windspeed']
cor_matrix = hour[thenames].corr()
print(cor_matrix)
```

Korzystamy tu z tej samej metody `corr()`, której używaliśmy wcześniej. Metoda `corr()` wywołana bez żadnych argumentów w nawiasie tworzy macierz korelacji dla wszystkich zmiennych w zbiorze danych. Macierz korelacji obliczona przez powyższy kod wygląda tak:

```
                hr      temp  windspeed
hr          1.000000  0.137603   0.137252
temp        0.137603  1.000000  -0.023125
windspeed   0.137252 -0.023125   1.000000
```

Mamy tu macierz o wymiarach 3×3. Każdy wpis w tej macierzy jest współczynnikiem korelacji. Na przykład w drugim wierszu i trzeciej kolumnie widać, że korelacja między prędkością wiatru a temperaturą to mniej więcej –0,023. Formalnie rzecz biorąc, jest to korelacja ujemna, choć jest tak bliska zeru, że zwykle opisalibyśmy te dwie zmienne jako nieskorelowane.

Trzy korelacje w macierzy są równe 1,0. Jest to zgodne z oczekiwaniami: te doskonałe korelacje mierzą zależność każdej zmiennej od samej siebie (korelację `hr` z `hr`, `temp` z `temp` i `windspeed` z `windspeed`). Każda zmienna zawsze ma doskonałą korelację sama ze sobą. Tworzenie macierzy korelacji to szybki, prosty sposób na zbadanie zależności między wszystkimi zmiennymi w danych i odkrycie zaskakujących korelacji dodatnich lub ujemnych.

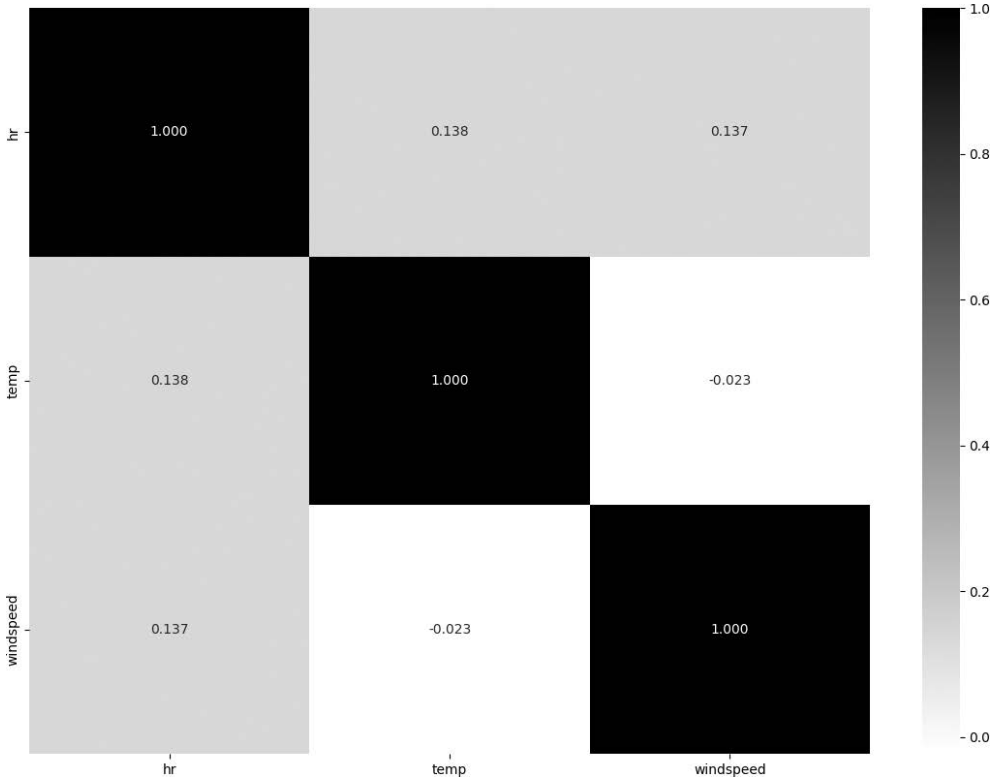
Tworzenie map cieplnych

Po utworzeniu macierzy korelacji możemy przedstawić wszystkie korelacje na wykresie, aby ułatwić sobie interpretację macierzy:

```
plt.figure(figsize=(14,10))
corr = hour[thenames].corr()
sns.heatmap(corr, annot=True,cmap='binary',
        fmt=".3f",
        xticklabels=thenames,
        yticklabels=thenames)
plt.show()
```

W tym przykładzie tworzymy **mapę cieplną**. Na wykresie kolor lub stopień zaciemnienia komórki wskazuje wartość liczby w tej komórce. Mapa cieplna z rysunku 1.13 pokazuje pomiary korelacji między zmiennymi.

Ta mapa cieplna składa się z dziewięciu prostokątów. Jak wskazuje legenda po prawej stronie, ciemniejsze wypełnienie prostokąta informuje, że dana korelacja jest wyższa, a jaśniejsze — że jest niższa. Mapa cieplna macierzy korelacji pozwala jeszcze szybciej wyszukiwać wzorce i zależności w danych, ponieważ silnie korelacje przyciągają wzrok.



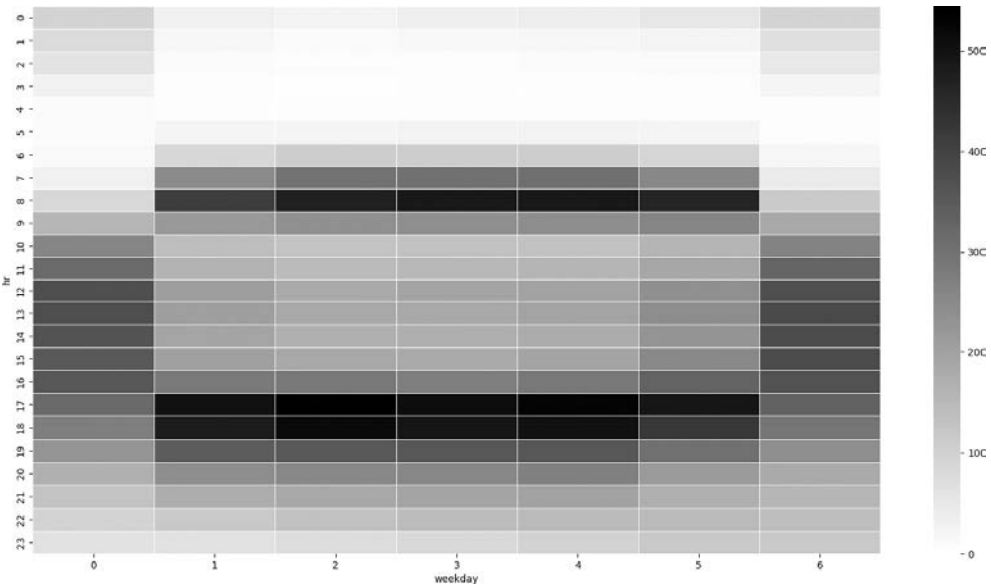
Rysunek 1.13. Korelacje pokazane na mapie cieplnej

Jeśli wolałbyś wykres w kolorach, a nie w skali szarości, zmień parametr `cmap='binary'` w metodzie `sns.heatmap()`. Parametr `cmap` określa *odwzorowanie kolorów* na mapie cieplnej, a poprzez wybranie innej wartości `cmap` możesz uzyskać różne układy kolorów. Jeśli na przykład użyjesz wartości `cmap='coolwarm'`, zobaczysz mapę cieplną, na której wyższe wartości są reprezentowane przez odcienie czerwieni, a niższe — przez odcienie koloru niebieskiego.

Mapy cieplne można rysować nie tylko dla macierzy korelacji. Możemy na przykład narysować mapę cieplną pokazującą liczbę rowerzystów podczas każdej godziny w ciągu tygodnia:

```
# Tworzymy tabelę przestawną
df_hm = hour.pivot_table(index = 'hr',columns = 'weekday',values = 'count')
# Rysujemy mapę cieplną
plt.figure(figsize = (20,10)) # Zmieniamy rozmiar wykresu
sns.heatmap(df_hm, fmt="d", cmap='binary',linewidths=.5, vmin = 0)
plt.show()
```

Aby narysować ten wykres, musimy najpierw utworzyć **tabelę przestawną**, czyli tabelę pogrupowanych wartości. Jeśli często pracujesz z Excelem lub innymi programami arkusza kalkulacyjnego, prawdopodobnie spotkałeś się już z tabelami przestawnymi. W tym przypadku nasza tabela przestawna grupuje wartości z pełnego zbioru danych według dnia tygodnia i godziny dnia. Mamy średnią liczbę rowerzystów w każdej godzinie (od 0.00 do 23.00) każdego dnia (od niedzieli do soboty). Po utworzeniu tabeli przestawnej z danymi zgrupowanymi w taki sposób możemy użyć tej samej metody `heatmap()` do utworzenia mapy cieplnej pokazanej na rysunku 1.14.



Rysunek 1.14. Liczba rowerzystów w każdej godzinie każdego dnia

Mapa ta zawiera ciemniejsze prostokąty dla godzin z większą liczbą rowerzystów i jaśniejsze dla godzin z mniejszą liczbą rowerzystów. Widzimy tu dojazdy do pracy ze wzrostami aktywności w okolicach godziny 8.00 rano i 5.00 po południu. Widzimy też weekendowe wypady w sobotnie i niedzielne popołudnia.

Ta mapa cieplna mogłaby podsunąć nam kilka pomysłów biznesowych. Na przykład duża liczba rowerzystów około godziny 8.00 rano w dni robocze sugeruje sposób na zwiększenie przychodów. Podobnie jak wyobraziliśmy sobie oferowanie zniżek w porach małej aktywności, moglibyśmy rozważyć lustrzaną strategię: tymczasowe podnoszenie stawek w porach szczególnie dużej aktywności. Strategii tej używają duże firmy transportowe, takie jak Uber, Lyft i Grab, nie tylko w celu zwiększenia przychodów, ale również w celu zapewnienia większej dostępności usług.

Dalsza eksploracja

Dotychczas przyjrzeliśmy się tylko jednemu zbiorowi danych i przeprowadziliśmy zaledwie kilka spośród niezliczonej liczby badań. W miarę jak Twój pierwszy poranek na stanowisku prezesa zmieni się w pierwsze popołudnie, potem drugi dzień itd., będziesz musiał podejmować wiele decyzji dotyczących firmy i jej działalności. Eksploracyjną analizę danych, którą przeprowadziliśmy w tym rozdziale, możesz stosować do innych problemów biznesowych, które napotkasz na swojej drodze. Mógłbyś na przykład rozważyć oferowanie odświeżających napojów w pakiecie z wypożyczeniem roweru, żeby zarobić w ten sposób dodatkowe pieniądze (a jednocześnie zadbać o zdrowie i bezpieczeństwo rowerzystów). Analiza danych związanych z Twoimi klientami, ich wzorcami użytkowania rowerów oraz stopniem pragnienia podczas jazdy pomoże Ci ustalić, czy ta strategia byłaby dobrym pomysłem.

Inne analizy mogą być związane z naprawami rowerów. Jak często trzeba naprawiać rowery i ile kosztują naprawy? Mógłbyś sprawdzić pory napraw i upewnić się, że nie przeprowadza się ich w godzinach szczytu. Mógłbyś sprawdzić koszty napraw różnych typów rowerów. Mógłbyś sprawdzić histogram cen napraw i dowiedzieć się, czy jakieś odstające przypadki nie zwiększają nadmiernie Twoich kosztów. Eksploracje te pomogą Ci lepiej zrozumieć firmę i podsuną pomysły, jak lepiej nią zarządzać.

Do tej pory nasze analizy nie były szczególnie wyrafinowane; obliczaliśmy tylko statystyki zbiorcze i rysowaliśmy wykresy. Ale te proste obliczenia i wykresy, w połączeniu ze zdrowym rozsądkiem, mogą być cennym pierwszym krokiem w kierunku podejmowania decyzji biznesowych. Niektórzy prezesi lekceważą dane, a inni chcą się im przyglądać, ale polegają na raportach przygotowywanych przez pracowników, które mogą być spóźnione albo niedokładne. Prezes, który potrafi sprawdzać dane dotyczące firmy, to prezes, który może działać efektywniej. Prezesi mogą się nauczyć pracować z danymi i połączyć to z wiedzą biznesową, aby lepiej wykonywać swoje obowiązki. Podobnie specjaliści data science mogą nauczyć się biznesu, a kiedy połączą swoje umiejętności analizy danych ze zmysłem biznesowym, mogą się stać siłą, z którą trzeba się liczyć.

Podsumowanie

W tym rozdziale zaczęliśmy od prostego scenariusza biznesowego: wyobraziliśmy sobie, że obejmujemy stanowisko prezesa i musimy podjąć decyzje dotyczące lepszego prowadzenia firmy. Omówiliśmy kilka pomysłów na to, co powinien zrobić prezes i jak może w tym pomóc eksploracyjna analiza danych. Dowiedziałeś się, jak wczytywać dane do Pythona, obliczać statystyki zbiorcze, rysować wykresy i interpretować wyniki w kontekście biznesowym. W następnym rozdziale poznasz regresję liniową, bardziej zaawansowaną metodę, którą można wykorzystać nie tylko do eksploracji, ale również do prognozowania. Kontynuujmy!

Skorowidz

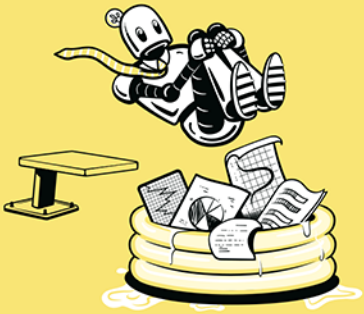
R

S

Ś

T


PROGRAM
PARTNERSKI
GRUPY HELION
1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ
Zmień swoją stronę WWW w działający bankomat!
Dowiedz się więcej i dołącz już dzisiaj!
http://program-partnerski.helion.pl
GRUPA
Helion



DATA SCIENCE DLA BIZNESU: CZARNA MAGIA? NIE Z PYTHONEM!

Data science to nieocenione wsparcie w rozwoju biznesu i działaniach mających na celu poprawę wyników finansowych firmy. Pomaga naukowcom lepiej obserwować i rozumieć otaczający ich świat. Bywa też źródłem świetnej zabawy. Jako analityk danych staniesz się częścią branży, która ciągle rośnie i się rozwija, a to znaczy, że wyzwania, jakie napotkasz, będą coraz ciekawsze i bardziej ekscytujące. Musisz się tylko nauczyć pracować z danymi.

Dzięki tej książce dowiesz się, jak pozyskiwać, analizować i wizualizować dane, a potem używać ich do rozwiązywania problemów biznesowych. Wystarczy, że znasz podstawy Pythona i matematyki na poziomie liceum, aby zacząć stosować naukę o danych w codziennej pracy. Znajdziesz tu szereg praktycznych i zrozumiałych przykładów: od usprawniania działalności wypożyczalni rowerów, poprzez wyodrębnianie danych z witryn internetowych, po budowę systemów rekomendacyjnych. Poznasz rozwiązania oparte na danych, przydatne w podejmowaniu decyzji biznesowych. Nauczysz się korzystać z eksploracyjnej analizy danych, przeprowadzać testy A/B i klasyfikację binarną, a także używać algorytmów uczenia maszynowego.

Sprawdź, jak w prosty sposób:

- prognozować popyt
- optymalizować kampanie marketingowe
- ograniczać odpływ klientów
- przewidywać ruch w witrynie internetowej
- budować systemy rekomendacyjne

Dr Bradford Tuckfield jest analitykiem danych, konsultantem i autorem książek (w tym *Zanurz się w algorytmach*, Helion, 2022). Pracował jako specjalista data science i menedżer do spraw technologii w czołowych firmach finansowych i startupach. Publikował swoje badania w czasopismach akademickich z zakresu matematyki, medycyny i zarządzania biznesem.

Helion
helion.pl
HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej! ▶

ISBN 978-83-289-1014-0
9 788328 910140

Cena: 69,00 zł

no starch press