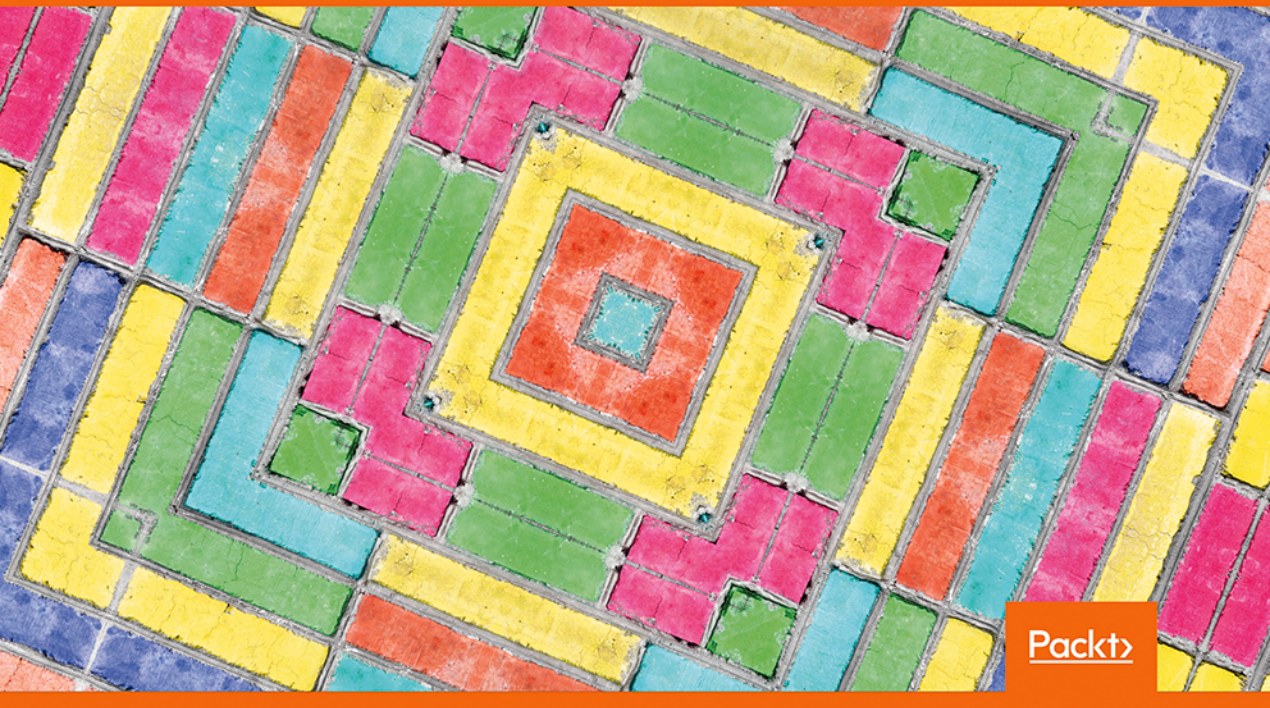
Wydanie II

Algorytmy Data Science

Siedmiodniowy przewodnik



Dávid Natingga

Tytuł oryginału: Data Science Algorithms in a Week: Top 7 algorithms for scientific computing, data analysis, and machine learning, 2nd Edition

Tłumaczenie: Andrzej Grażyński

ISBN: 978-83-283-5602-3

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/aldas2.zip

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/aldas2
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Printed in Poland.

Spis treści

O autorze 7

O recenzentach 8

Przedmowa 9

Rozdział 1. Klasyfikacja na podstawie najbliższego sąsiedztwa 13

Subiektywne odczuwanie temperatury 14
Implementacja algorytmu k najbliższych sąsiadów 16
Mapa Włoch — przykład doboru wartości k 20
Analiza 21
Skalowanie danych — prognozowanie statusu własności 25
Analiza 26
Nieeuklidesowe metryki odległości punktów — klasyfikowanie tekstów 27
Analiza 28
Klasyfikowania tekstów ciąg dalszy — wielowymiarowy algorytm k-NN 30
Analiza 31
Podsumowanie 32
Problemy 33
Subiektywne odczuwanie temperatury 33
Mapa Włoch — przykład doboru wartości k 33
Status własności 34
Analiza 34

Rozdział 2. Naiwny klasyfikator bayesowski 37

Testy medyczne — podstawowe zastosowanie twierdzenia Bayesa 38
Analiza 38
Podstawowe twierdzenie Bayesa i jego rozszerzenie 39
Twierdzenie Bayesa 39
Rozszerzone twierdzenie Bayesa 40

Zagramy w szachy? — niezależne zdarzenia warunkujące **41**
Analiza 42
Implementacja naiwnego klasyfikatora bayesowskiego **43**
Zagramy w szachy? — częściowo zależne zdarzenia warunkujące **45**
Analiza 45
Chłopak czy dziewczyna? — twierdzenie Bayesa dla ciągłych zmiennych losowych **48**
Analiza 48
Podsumowanie **50**
Problemy **51**
Analiza 53

Rozdział 3. Drzewa decyzyjne **59**

Pływamy? — reprezentowanie danych w postaci drzewa decyzyjnego **59**
Elementy teorii informacji **61**
Entropia informacyjna 61
Zysk informacyjny 63
Pływamy? — obliczanie zysku informacyjnego 63
Algorytm ID3 — konstruowanie drzewa decyzyjnego **65**
Pływamy? — budowanie drzewa decyzyjnego 65
Implementacja w języku Python 66
Klasyfikowanie danych za pomocą drzew decyzyjnych **71**
Przykład — pływamy czy nie? 72
Przykład — gra w szachy pod chmurką **72**
Analiza 72
Na zakupy — przykład niespójnych danych **77**
Analiza 77
Podsumowanie **78**
Problemy **78**
Analiza 80

Rozdział 4. Lasy losowe **83**

Ogólne zasady konstruowania lasów losowych **84**
Pływamy? — klasyfikacja za pomocą lasu losowego **84**
Analiza 84
Konstruowanie lasu losowego 85
Klasyfikowanie cechy na podstawie lasu losowego 89
Implementacja algorytmu konstruowania lasu losowego **90**
Przykład — zagramy w szachy? **93**
Analiza 93
Konstruowanie lasu losowego 94
Klasyfikacja w drodze głosowania 99
Idziemy na zakupy? — wnioskowanie z niespójnych danych i miara wiarygodności wyniku **100**
Analiza 100
Podsumowanie **101**
Problemy **102**
Analiza 103

Rozdział 5. Klasteryzacja **107**

Dochód gospodarstwa domowego — niski czy wysoki? **107**
Algorytm k-średnich **108**
Początkowy zbiór centroidów 109
Wyznaczanie centroidu klastera 109
Przykład — wykorzystanie algorytmu k-średnich do klasyfikacji dochodów **110**
Klasyfikowanie przez klasteryzację — prognozowanie płci nieznanej osoby **111**
Analiza 112
Implementacja algorytmu k-średnich **115**
Status własności — dobór optymalnej liczby klasterów **118**
Analiza 119
Klasyfikowanie dokumentów — semantyczne znaczenie klasteryzacji **125**
Analiza 126
Podsumowanie **132**
Problemy **132**
Analiza 133

Rozdział 6. Analiza regresji **143**

Konwersja temperatur — regresja liniowa dla danych doskonałych **144**
Rozwiązanie analityczne 144
Metoda najmniejszych kwadratów w regresji liniowej 145
Implementacja analizy regresji liniowej w Pythonie 146
Regresja dla danych pomiarowych — prognozowanie wagi na podstawie wzrostu **149**
Analiza 149
Metoda spadku gradientowego i jej implementacja **151**
Szczegóły algorytmu 151
Implementacja w Pythonie 152
Przewidywanie czasu przelotu na podstawie odległości **154**
Analiza 154
Obliczenia balistyczne — model nieliniowy **156**
Analiza 156
Podsumowanie **158**
Problemy **158**
Analiza 159

Rozdział 7. Analiza szeregów czasowych **163**

Zysk w biznesie — analiza trendu **164**
Analiza 164
Konkluzja 165
Sprzedaż w sklepie internetowym — analiza sezonowości **166**
Analiza 166
Analiza trendu 166
Analiza sezonowości 169
Podsumowanie **175**
Problemy **176**
Analiza 177

Dodatek A. Podstawy języka Python **181**

Przykład **181**
Komentarze **182**
Typy danych **182**
int 182
float 183
Napis 183
Krotka 183
Lista 184
Zbiór 184
Słownik 185
Przepływ sterowania **185**
Instrukcje warunkowe 186
Pętla for 187
Pętla while 188
Instrukcje break i continue 189
Funkcje **190**
Wejście-wyjście programu **192**
Argumenty wywołania programu 192
Operacje na plikach 192

Dodatek B. Statystyka **195**

Podstawowe koncepcje **195**
Notacja 195
Podstawowe pojęcia 195
Wnioskowanie bayesowskie **197**
Rozkład normalny Gaussa **197**
Walidacja krzyżowa **197**
Testowanie A/B **198**

Dodatek C. Słownik pojęć, algorytmów i metod Data Science **199**

Skorowidz **203**

ROZDZIAŁ 1

Klasyfikacja na podstawie najbliższego sąsiedztwa

Algorytm najbliższego sąsiedztwa dokonuje klasyfikacji elementów danych na podstawie ich sąsiadów. Klasa instancji danych określona przez algorytm k najbliższych sąsiadów (ang. *k-nearest neighbors*, w skrócie *k-NN*) jest klasą o najwyższej reprezentacji wśród k sąsiadów położonych najbliżej w sensie przyjętej metryki.

W tym rozdziale wyjaśniam szczegółowo:

- jak zaimplementować podstawową wersję algorytmu k-NN — na przykładzie subiektywnego odczuwania temperatury;
- jak wybierać właściwą wartość k, by algorytm działał prawidłowo i dawał wyniki o dużej dokładności— na podstawie mapy Włoch;
- jak przeskalowywać wartości w celu przygotowania ich jako dane wejściowe do algorytmu k-NN — na przykładzie domniemania własności nieruchomości;
- jak definiować właściwą metrykę wyznaczającą odległości między punktami danych;
- jak eliminować nieistotne wymiary z wysokowymiarowych przestrzeni w celu zapewnienia dokładności wyników algorytmu — na przykładzie klasyfikacji tekstu.

Subiektywne odczuwanie temperatury

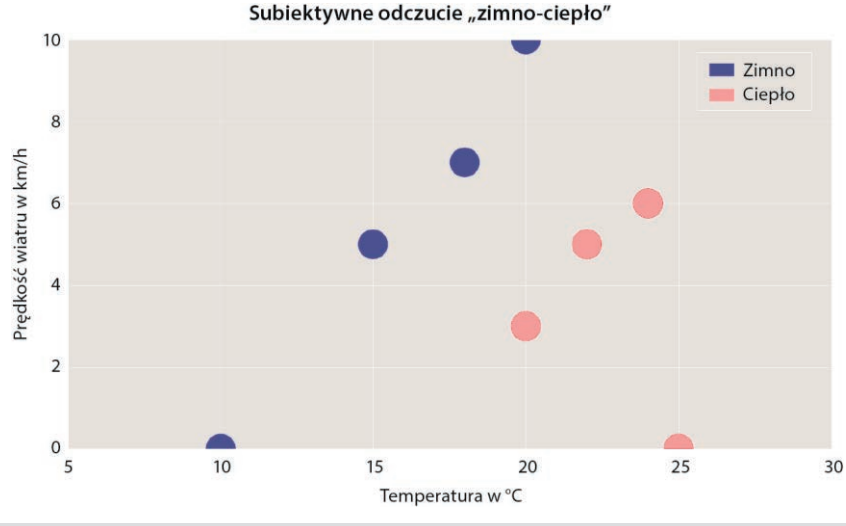
Załóżmy, że pewna sympatyczna Marysia subiektywnie odczuwa zimno przy temperaturze 10°C, lecz 25°C oznacza dla niej ciepło. Jakie będzie jej subiektywne odczucie w kategoriach „zimno-ciepło" w pokoju o temperaturze 22°C? Intuicyjnie, ponieważ temperatura 22°C jest wyraźnie bliższa 25°C niż 10°C — innymi słowy: bliżej jej do „prawego" niż „lewego" sąsiada — zgodnie z algorytmem 1-NN wspomniany pokój będzie wydawać się Marysi pokojem ciepłym.

Skomplikujmy nieco ów prosty przykład. Wiemy wszyscy, chociażby z prognoz pogody, że na subiektywne odczuwanie ciepła-zimna istotny wpływ ma, oprócz fizycznej temperatury powietrza, kilka innych czynników, między innymi prędkość wiatru. Po uwzględnieniu tejże Marysia skłonna jest klasyfikować swe odczucia zgodnie z tabelą 1.1.

Tabela 1.1. Subiektywne odczuwanie temperatury

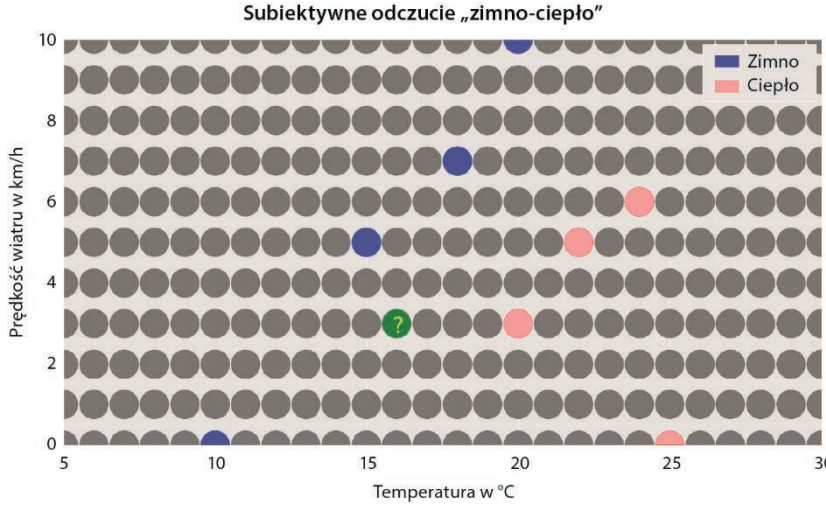
Temperatura w °C	Prędkość wiatru w km/h	Subiektywne odczucie
10	0	Zimno
25	0	Ciepło
15	5	Zimno
20	3	Ciepło
18	7	Zimno
20	10	Zimno
22	5	Ciepło
24	6	Ciepło

W układzie graficznym zależności te prezentują się tak, jak na rysunku 1.1:



Rysunek 1.1. Subiektywne odczuwanie zimna i ciepła w zależności od rzeczywistej temperatury i prędkości wiatru

Stosując ponownie algorytm 1-NN spróbujmy wydedukować subiektywne odczucie Marysi przy temperaturze 16°C i wietrze wiejącym z prędkością 3 km/h (rysunek 1.2):

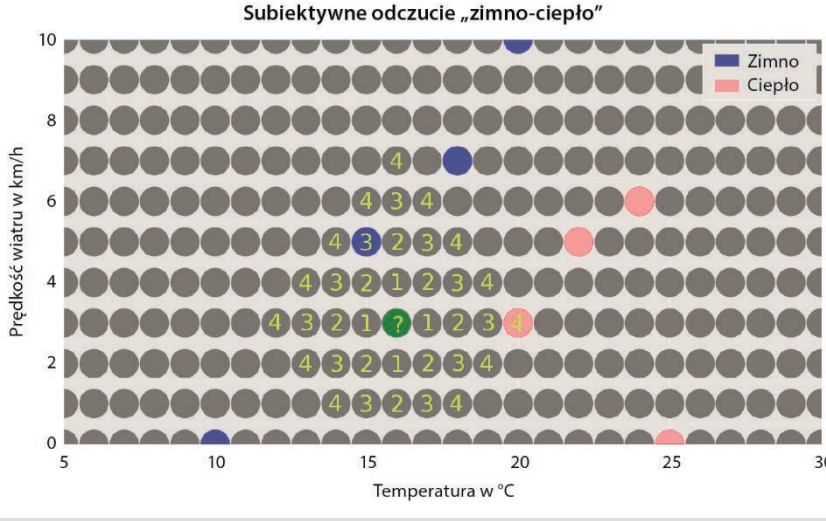


Rysunek 1.2. Niewiadomy punkt podlegający klasyfikacji

Do określenia odległości między dwoma punktami $N_1 = (x_1, y_1)$ i $N_2 = (x_2, y_2)$ użyjemy tzw. **metryki Manhattanu**, zdefiniowanej następująco:

$$d_{Man} = |x_1 - x_2| + |y_1 - y_2|$$

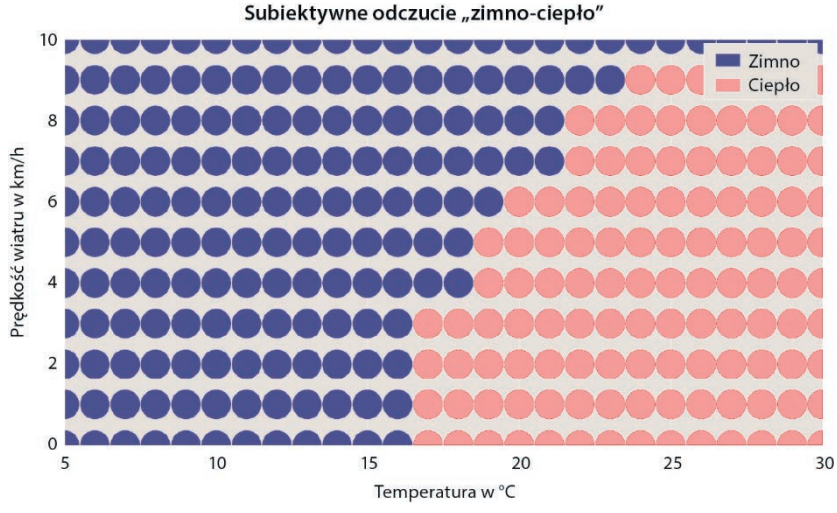
Na rysunku 1.3 badany punkt (16°C, 3 km/h) oznaczony jest znakiem zapytania, a jego sąsiadów oznaczono liczbami wyrażającymi odległość od niego według metryki d_{Man}.



Rysunek 1.3. Najbliższe sąsiedztwo badanego punktu

Spośród punktów jawnie wymienionych przez Marysię, najbliższym punktu badanego (w sensie odległości d_{Man}) jest punkt (15°C, 5 km/h) subiektywnie sklasyfikowany jako „zimny". Jest odległy od badanego punktu o 3 jednostki; najbliższy subiektywnie „zimny" punkt odległy jest od badanego o 4 jednostki. Zgodnie z algorytmem 1-NN decydujący jest sąsiad położony bliżej, czyli „zimny" punkt odległy od badanego o 3 jednostki. Ergo — przy temperaturze 16°C i wietrze 3 km/h Marysia będzie odczuwała zimno.

Opisaną procedurę klasyfikacyjną można zastosować do każdego punktu diagramu, otrzymując kompletny wykres, jak na rysunku 1.4.



Rysunek 1.4. Wynik klasyfikacji punktów

Co jednak począć w sytuacji, gdy najbliższy badanemu punkt „zimny" znajduje się w tej samej odległości, co najbliższy punkt „ciepły"? Na naszym wykresie takim punktem jest na przykład punkt (20°C, 6 km/h). Konieczne staje się wówczas określenie dodatkowych kryteriów, na przykład zasad pierwszeństwa jednej klasy przed drugą, czy nawet rezygnacja z klasyfikowania takich „granicznych" punktów — wszystko to jest specyfiką konkretnego problemu i konkretnej implementacji algorytmu.

Implementacja algorytmu k najbliższych sąsiadów

Opisana metoda zaimplementowana została w postaci programu w języku Python. Główny plik programu — *knn_to_data.py* — znajduje się w katalogu *temperature_preferences* wewnątrz pakietu przykładowych programów dostępnego do pobrania z witryny Wydawnictwa Helion; tam też znajdują się pliki z przykładowymi danymi. Na listingu 1.1 przedstawiam najważniejsze fragmenty wspomnianego programu, w dalszej części rozdziału zaprezentuję jego moduł odpowiedzialny za wizualizację dokonanej klasyfikacji.

Listing 1.1. Implementacja algorytmu k-NN w języku Python

```
# knn_to_data.py
# Implementacja algorytmu knn
# Plik wejściowy jest plikiem tekstowym, którego każda linia zawiera dwie pozycje:
#   temperaturę (w st. Celsjusza),
#   prędkość wiatru w km/h

import sys
sys.path.append('..')
sys.path.append('../../common')
import knn  # noqa
import common  # noqa


# Początek programu
# Np. "temperature_preferences.data"
input_file = sys.argv[1]
# Np. "temperature_preferences_completed.data"
output_file = sys.argv[2]
k = int(sys.argv[3])
x_from = int(sys.argv[4])
x_to = int(sys.argv[5])
y_from = int(sys.argv[6])
y_to = int(sys.argv[7])

data = common.load_3row_data_to_dic(input_file)
new_data = knn.knn_to_2d_data(data, x_from, x_to, y_from, y_to, k)
common.save_3row_data_from_dic(output_file, new_data)

# common\common.py
# *** Biblioteka wspólnych programów i funkcji ***
def dic_inc(dic, key):
    if key is None:
        pass
    if dic.get(key, None) is None:
        dic[key] = 1
    else:
        dic[key] = dic[key] + 1

# knn.py
# *** Biblioteka funkcji implementujących algorytm knn ***

# Resetuje liczniki sąsiadów i grup klas dla wybranego punktu
def info_reset(info):
    info['nbhd_count'] = 0
    info['class_count'] = {}

# Znajduje klasę sąsiada o współrzędnych (x, y)
# Jeśli klasa ta jest znana, sąsiad jest uwzględniany

def info_add(info, data, x, y):
    group = data.get((x, y), None)
    common.dic_inc(info['class_count'], group)
    info['nbhd_count'] += int(group is not None)

# Stosuje algorytm k-NN do danych dwuwymiarowych, uwzględniając k najbliższych
# sąsiadów według metryki Manhattan.
# Współrzędne pełnią w słowniku rolę kluczy, wartościami są klasy.
```

```
# Współrzędne (x, y) mieszczą się w zakresie [x_from, x_to] ... [y_from, y_to].
def knn_to_2d_data(data, x_from, x_to, y_from, y_to, k):
    new_data = {}
    info = {}
    # Przejdź przez wszystkie możliwe współrzędne
    for y in range(y_from, y_to + 1):
        for x in range(x_from, x_to + 1):
            info_reset(info)
            # Uwzględnij liczbę sąsiadów dla każdej grupy klas i dla każdej
            # odległości, począwszy od 0 aż do znalezienia co najmniej
            # k sąsiadów o znanej klasie
            for dist in range(0, x_to - x_from + y_to - y_from):
                # Uwzględnij sąsiadów oddalonych o "dist" od badanego
                # punktu (x, y)
                if dist == 0:
                    info_add(info, data, x, y)
                else:
                    for i in range(0, dist + 1):
                        info_add(info, data, x - i, y + dist - i)
                        info_add(info, data, x + dist - i, y - i)
                    for i in range(1, dist):
                        info_add(info, data, x + i, y + dist - i)
                        info_add(info, data, x - dist + i, y - i)
                # Ponieważ wiele punktów może być jednakowo odległych od badanego punktu (x, y)
                # możemy mieć więcej niż k najbliższych sąsiadów.
                # Gdy ich liczba osiągnie lub przekroczy k, przerywamy pętlę
                                if info['nbhd_count'] >= k:
                    break
            class_max_count = None
            # Wybierz klasę o największym liczniku sąsiadów
            # spośród k najbliższych sąsiadów
              for group, count in info['class_count'].items():
                if group is not None and (class_max_count is None or
                   count > info['class_count'][class_max_count]):
                    class_max_count = group
            new_data[x, y] = class_max_count
    return new_data
```

Dane wejściowe

Plik danych wejściowych do opisywanego programu — *temperature_preferences.data* — jest plikiem tekstowym, którego poszczególne wiersze odpowiadają punktom danych o znanych klasach. Każdy wiersz zawiera kolejno trzy elementy: temperaturę w stopniach Celsjusza, prędkość wiatru w km/h i nazwę klasy reprezentującej subiektywne odczucie („zimno” albo „ciepło”). Zgodnie z deklaracją Marysi, zawartość tego pliku jest następująca:

```
10 0 zimno
25 0 ciepło
15 5 zimno
20 3 ciepło
18 7 zimno
20 10 zimno
22 5 ciepło
24 6 ciepło
```

Wynik klasyfikacji

Program, uruchomiony za pomocą polecenia

```
$ python knn_to_data.py temperature_preferences.data
temperature_preferences_completed.data 1 5 30 0 10
```

wykonał, za pomocą algorytmu k-NN dla $k = 1$, klasyfikację dla punktów o temperaturze zmieniającej się od 5 do 30 z krokiem 1 i dla prędkości wiatru zmieniającej się od 0 do 10 z krokiem 1, czyli dla $(30–5+1) \times (10–0+1) = 286$ punktów. Wynik klasyfikacji zapisany został w pliku tekstowym *temperature_preferences_completed.data*. Pracując w Linuksie, za pomocą polecenia `wc` możemy się przekonać, że plik istotnie zawiera 286 wierszy, a stosując polecenie `head` możemy wyświetlić 10 początkowych:

```
$ wc -l temperature_preferences_completed.data
286 temperature_preferences_completed.data

$ head -10 temperature_preferences_completed.data
7 3 zimno
6 9 zimno
12 1 zimno
16 6 zimno
16 9 zimno
14 4 zimno
13 4 zimno
19 4 ciepło
18 4 zimno
15 1 zimno
```

(W Windows możemy użyć dowolnego programu zapewniającego numerowanie wierszy, na przykład edytora *PsPad*).

Wizualizacja

Wyniki klasyfikacji mogą zostać wyświetlone w postaci diagramu, utworzonego przy użyciu *biblioteki matplotlib*. Służy do tego program *temperature_preferences_draw_graph.py*, widoczny na listingu 1.2, wywoływany bez parametrów (nazwa pliku wejściowego *temperature_preferences_completed.data* jest ustalona w kodzie programu).

Listing 1.2. Prezentacja wyników klasyfikacji przy użyciu biblioteki matplotlib

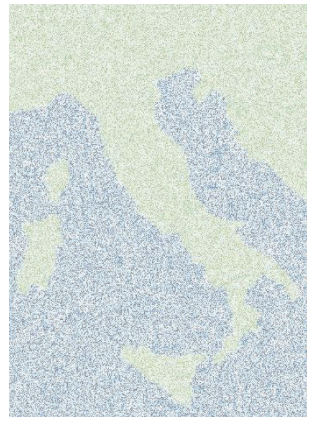
```
# common/common.py
# Zwraca słownik złożony z trzech list, zawierających:
#  1. Współrzędne x punktów
#  2. Współrzędne y punktów
#  3. Kolory wyświetlania punktów (w postaci numerycznej)
def get_x_y_colors(data):
    dic = {}
    dic['x'] = [0] * len(data)
    dic['y'] = [0] * len(data)
    dic['colors'] = [0] * len(data)
    for i in range(0, len(data)):
        dic['x'][i] = data[i][0]
        dic['y'][i] = data[i][1]
        dic['colors'][i] = data[i][2]
    return dic
```

```
# temperature_preferences_draw_graph.py
import sys
sys.path.append('../../common')  # noqa
import common
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import matplotlib
matplotlib.style.use('ggplot')

data_file_name = 'temperature_preferences_completed.data'
temp_from = 5
temp_to = 30
wind_from = 0
wind_to = 10

data = np.loadtxt(open(data_file_name, 'r'),
                  dtype={'names': ('temperature', 'wind', 'perception'),
                         'formats': ('i4', 'i4', 'S4')})

# Przyporządkowanie kolorów wyświetlania do poszczególnych klas
for i in range(0, len(data)):
    if data[i][2] == 'zimno':
        data[i][2] = 'blue'
    elif data[i][2] == 'ciepło':
        data[i][2] = 'red'
    else:
        data[i][2] = 'gray'

# Konwersja tablicy na format wyświetlania
data_processed = common.get_x_y_colors(data)

# Rysowanie wykresu.
plt.title('Subiektywne odczuwanie temperatury')
plt.xlabel('Temperatura w °C')
plt.ylabel('Prędkość wiatru w km/h')
plt.axis([temp_from, temp_to, wind_from, wind_to])
# Dodanie legendy do wykresu
blue_patch = mpatches.Patch(color='blue', label='zimno')
red_patch = mpatches.Patch(color='red', label='ciepło')
plt.legend(handles=[blue_patch, red_patch])
plt.scatter(data_processed['x'], data_processed['y'],
            c=data_processed['colors'], s=[1400] * len(data))
plt.show()
```

Mapa Włoch — przykład doboru wartości k

Załóżmy, że dysponujemy mapą Włoch i ich najbliższego otoczenia i że ok.1 procenta powierzchni tej mapy pokryte jest kolorowymi punktami. Punkt oznaczony kolorem zielonym znajduje się w obszarze lądu, analogicznie — punkt niebieski znajduje się na obszarze morza. Pozostała część mapy to biały obszar; naszym zadaniem jest sklasyfikowanie wybranych punktów tego białego obszaru w kategoriach „ląd-morze”.

Jak można się spodziewać, mapa z 1-procentowym pokryciem punktowym jest praktycznie niewidoczna, gdy jednak zwiększymy jej pokrycie do 33%, otrzymamy obraz widoczny na rysunku 1.5.



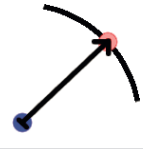
Rysunek 1.5. Mapa Włoch — wersja 1

Analiza

Do rozwiązania tak postawionego problemu, czyli zwiększenia stopnia pokrycia mapy, użyjemy algorytmu k-NN z dowolną wartością k. Będziemy mianowicie analizować k najbliższych sąsiadów punktu badanego i klasyfikować ten punkt na zasadzie większości: jeżeli mianowicie wśród k najbliższych sąsiadów będzie więcej punktów zielonych niż niebieskich, badany punkt sklasyfikujemy jako zielony (czyli znajdujący się na lądzie), w przeciwnym razie sklasyfikujemy go jako niebieski. Aby wykluczyć kłopotliwe sytuacje „remisowe", będziemy rozpatrywać wyłącznie nieparzyste wartości k. Odległość między punktami będziemy mierzyć w zwykłej metryce euklidesowej — dla punktów $X = (x_0, x_1)$ i $Y = (y_0, y_1)$ definiujemy ją jako

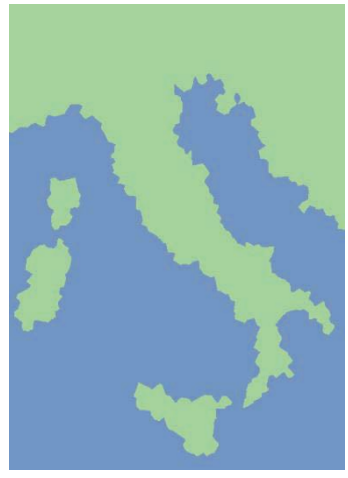
$$d_{euclid} = \sqrt{(x_0 - y_0)^2 + (x_1 - y_1)^2}$$

Metryka euklidesowa jest najbardziej intuicyjna, bowiem odpowiada odległości mierzonej zwykłą linijką (rysunek 1.6):

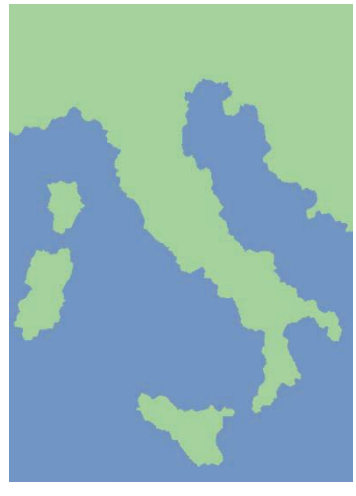


Rysunek 1.6. Odległość euklidesowa między punktami

Dążąc do wybrania najlepszej (z perspektywy jakości rozwiązania) wartości k, wykonaliśmy algorytm k-NN dla k równego (kolejno) 1, 3, 5, 7 i 9, otrzymując rezultaty przedstawione na rysunkach 1.7 – 1.11:



Rysunek 1.7. Mapa Włoch — pokrycie dla k=1



Rysunek 1.8. Mapa Włoch — pokrycie dla k=3

Jak można było oczekiwać, im większa wartość *k*, tym mniej „poszarpane" są granice między lądem a morzem. Oczywiście interesującym doświadczeniem będzie porównanie wyników klasyfikacji z autentyczną mapą Włoch, widoczną na rysunku 1.12.

Rysunek 1.9. Mapa Włoch — pokrycie dla k=5

Rysunek 1.10. Mapa Włoch — pokrycie dla k=7

Miarą dokładności algorytmu z pewnością będzie procentowy udział punktów błędnie sklasyfikowanych w ogólnej liczbie punktów — im mniejszy, tym lepiej. Wyniki porównania dla poszczególnych wartości *k* przedstawione są w tabeli 1.2.

Rysunek 1.11. Mapa Włoch — pokrycie dla k=9

Rysunek 1.12. Kompletna mapa Włoch

Tabela 1.2. Weryfikacja trafności klasyfikacji punktów na mapie Włoch

Liczba uwzględnianych sąsiadów (k)	Udział błędnie sklasyfikowanych punktów (w %)
1	2,97
3	3,24
5	3,29
7	3,40
9	3,57

Jak widać, w tym szczególnym przypadku najlepszą zgodność z oryginałem wykazuje klasyfikacja wykonana dla $k = 1$. Przypadek ten jest jednak o tyle nietypowy, że zwykle nie dysponujemy obiektywnym wzorcem dla porównania wykonywanych klasyfikacji, lecz jedynie przybliżeniem takowego. Powrócimy do tej kwestii, rozważając problem 4 na końcu tego rozdziału.

Skalowanie danych — prognozowanie statusu własności

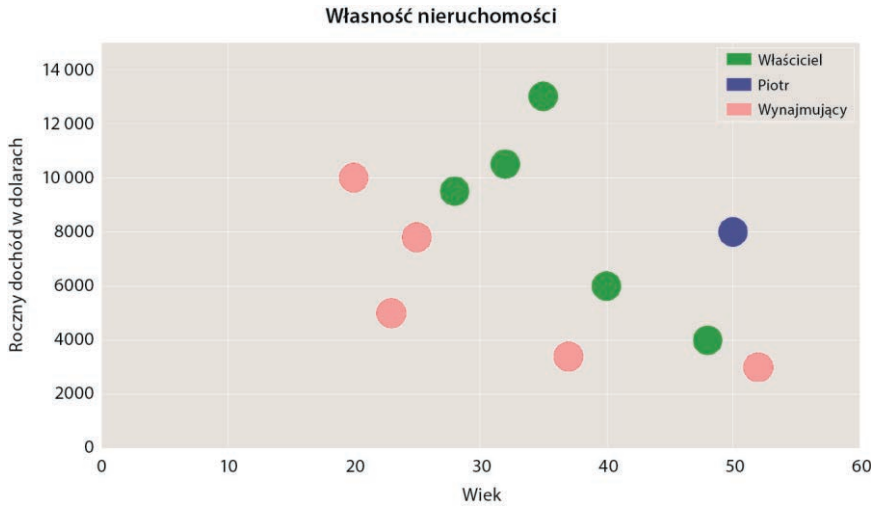
Pewna firma, specjalizująca się w ubezpieczeniach nieruchomości, zainteresowana jest kierowaniem swej oferty do właścicieli domów na pewnym obszarze; nie interesują jej z oczywistych względów lokatorzy zamieszkujący w domach na zasadzie wynajmu, a jedynie właściciele. W firmie tej zdają sobie sprawę z faktu, że bycie właścicielem nieruchomości pozostaje w pewnej korelacji z dwoma czynnikami: wiekiem lokatora i osiąganymi przez niego dochodami. Kierownictwo dysponuje także pewnymi wyrywkowymi danymi w tym względzie, przedstawionymi w tabeli 1.3.

Tabela 1.3. Status własności domu w zależności od wieku osoby zamieszkującej i jej dochodów

Wiek	Roczny dochód (w dolarach)	Czy właściciel?
23	50 000	Nie
37	34 000	Nie
48	40 000	Tak
52	30 000	Nie
28	95 000	Tak
25	78 000	Nie
35	130 000	Tak
32	105 000	Tak
20	100 000	Nie
40	60 000	Tak
50	80 000	?

Ostatni wiersz tej tabeli reprezentuje Piotra, do którego firma chciałaby skierować swą ofertę, nie wie jednak, czy jest on właścicielem domu, w którym mieszka (stąd znak zapytania w ostatniej kolumnie) i chciałaby tę informację wydedukować z poprzednich wierszy.

Zawartość tej tabeli można zilustrować na dwuwymiarowym diagramie, widocznym na rysunku 1.13.



Rysunek 1.13. Własność nieruchomości w zależności od wieku i dochodu — ujęcie graficzne

Analiza

Do rozwiązania tego problemu moglibyśmy wykorzystać algorytm 1-NN, powinniśmy jednak zwrócić uwagę na pewien istotny fakt: rozpiętość wartości rocznych dochodów jest znacznie większa niż rozpiętość wieku ankietowanych osób — odległość między (relatywnie) bliskimi dochodami 115 000 i 116 000 wynosi 1000 jednostek, podczas gdy odległość między skrajnymi wartościami wieku tylko 32 jednostki. Ponieważ obie te wielkości — dochód i wiek — reprezentowane będą na osiach tego samego wykresu, musimy dokonać ich przeskalowania, czyli odwzorowania zakresu zmian każdej z nich na przedział (domknięty) <0, 1>. Najbardziej oczywista formuła wyznaczająca to przeskalowanie ma postać:

$$wartość\ przeskalowana = \frac{wartość\ oryginalna - wartość\ minimalna}{wartość\ maksymalna - wartość\ minimalna}$$

w szczególności:

$$wiek\ przeskalowany = \frac{wiek\ oryginalny - wiek\ minimalny}{wiek\ maksymalny - wiek\ minimalny}$$

$$dochód\ przeskalowany = \frac{dochód\ oryginalny - dochód\ minimalny}{dochód\ maksymalny - dochód\ minimalny}$$

Wyniki takiego skalowania prezentują się jak w tabeli 1.4.

Jeśli do tak przeskalowanych danych zastosujemy algorytm 1-NN z metryką euklidesową, w przypadku Piotra bardziej prawdopodobne będzie to, że jest on właścicielem, a nie najemcą. Gdybyśmy jednak pominęli skalowanie i zastosowali algorytm 1-NN do oryginalnych danych, otrzymalibyśmy wynik odwrotny. Powrócimy do tej kwestii w problemie nr 5 na końcu tego rozdziału.

Tabela 1.4. Status własności domu — wynik przeskalowania danych

Wiek	Przeskalowany wiek	Dochód roczny	Przeskalowany dochód roczny	Czy właściciel?
23	0,09375	50 000	0,2	Nie
37	0,53125	34 000	0,04	Nie
48	0,875	40 000	0,1	Tak
52	1	30 000	0	Nie
28	0,25	95 000	0,65	Tak
25	0,15625	78 000	0,48	Nie
35	0,46875	130 000	1	Tak
32	0,375	105 000	0,75	Tak
20	0	100 000	0,7	Nie
40	0,625	60 000	0,3	Tak
50	0,9375	80 000	0,5	?

Nieeuklidesowe metryki odległości punktów — klasyfikowanie tekstów

Załóżmy, że dysponujemy zestawem dokumentów z dwóch dziedzin tematycznych — matematyki i informatyki. Dla każdego z tych dokumentów dysponujemy statystyką częstotliwości występowania słów „algorithm" i „computer"[1]. Dysponujemy również nieznanym dokumentem, dla którego chcielibyśmy wydedukować prawdopodobną tematykę (matematyka czy informatyka?) na podstawie wspomnianej statystyki. Dostępne dane przedstawione są w tabeli 1.5.

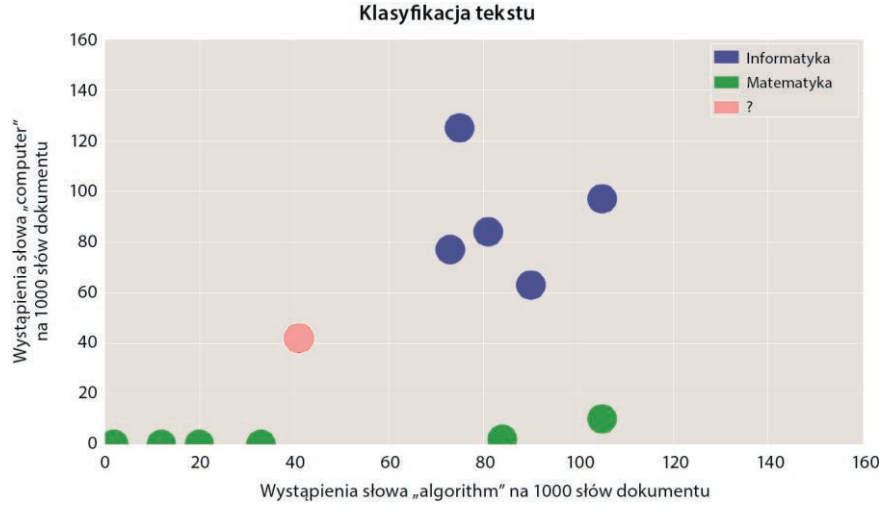
Już na pierwszy rzut oka dokumenty o dużej częstotliwości występowania obu słów kluczowych („algorithm" i „computer") można rozpoznać jako traktujące o informatyce. Teksty z dużą liczbą wystąpień słowa „algorithm", lecz z nikłą (lub zerową) obecnością słowa „computer" związane są z matematyką; wynikać to może poniekąd z faktu, że w historii ludzkości algorytmy pojawiały się już kilka tysięcy lat przed skonstruowaniem pierwszego komputera (czego przykładem jest chociażby algorytm obliczania największego wspólnego dzielnika autorstwa Euklidesa czy chińskie twierdzenie o resztach).

Zawartość przedstawionej tabeli prezentuje się na wykresie tak, jak pokazuje rysunek 1.14 — prognozowany dokument zawiera średnio 41 wystąpień słowa „algorithm" i 42 wyrazu „computer" na 1000 słów.

[1] Zachowaliśmy wersję oryginalną, w języku polskim wobec odmiany rzeczowników przez przypadki opisywana analiza nie miałaby racji bytu — *przyp. red.*

Tabela 1.5. Statystyka występowania słów „algorithm" i „computer" w dostępnych dokumentach o znanej tematyce

Wystąpienia słowa „algorithm" na 1000 słów dokumentu	Wystąpienia słowa „computer" na 1000 słów dokumentu	Tematyka dokumentu
153	150	Informatyka
105	97	Informatyka
75	125	Informatyka
81	84	Informatyka
73	77	Informatyka
90	63	Informatyka
20	0	Matematyka
33	0	Matematyka
105	10	Matematyka
2	0	Matematyka
84	2	Matematyka
12	0	Matematyka
41	42	?



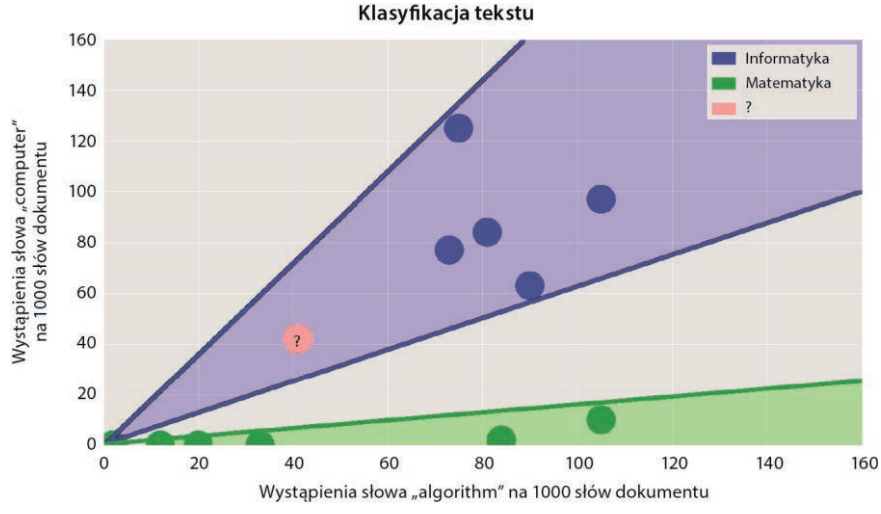
Rysunek 1.14. Badany dokument w kontekście dokumentów o znanej tematyce

Analiza

Gdybyśmy zastosowali do tego przypadku algorytm 1-NN z metryką euklidesową lub metryką Manhattanu, przedmiotowy dokument zostałby sklasyfikowany jako traktujący o matematyce. Intuicyjnie wydaje się to sprzeczne ze wstępnym spostrzeżeniem, według którego dokumenty

o „zrównoważonym” występowaniu obu słów kluczowych są raczej dokumentami informatycznymi, nie matematycznymi. Możemy podejrzewać, że jest to konsekwencja zastosowania nieodpowiedniej metryki.

Zastosujmy więc metrykę cokolwiek inną od wspomnianych: jako odległość między punktami przyjęlibyśmy ich *odległość kątową*, czyli rozwartość kąta, na którego ramionach leżą rzeczone punkty i którego wierzchołek znajduje się w początku układu współrzędnych. Koncepcję tę ilustruje rysunek 1.15.



Rysunek 1.15. Klasyfikacja punktów w metryce kątowej

Ze względu na efektywność obliczeń wygodniej jest jednak przyjąć inną miarę odległości między punktami: nie opisywany kąt bezpośrednio, lecz jego *kosinus* — ten bowiem łatwo daje się obliczyć na podstawie współrzędnych obu punktów, z definicji iloczynu skalarnego dwóch wektorów. Niech mianowicie przedmiotowe punkty mają współrzędne:

$$a = (a_x, a_y), b = (b_x, b_y)$$

Odległości (euklidesowe) obu punktów od początku układu współrzędnych wynoszą:

$$|a| = \sqrt{a_x^2 + a_y^2}, |b| = \sqrt{b_x^2 + b_y^2}$$

Ze wzoru na iloczyn skalarny wektorów

$$|a||b|cos(\theta) = a \cdot b = a_x \cdot b_x + a_y \cdot b_y$$

otrzymujemy[2]

$$cos(\theta) = \frac{a_x \cdot b_x + a_y \cdot b_y}{|a||b|}$$

Stosując do owej metryki algorytm 1-NN i pamiętając, że kosinus jest funkcją malejącą w przedziale $(0, \pi)$ — im mniejsza wartość, tym punkty bardziej odległe — otrzymamy werdykt, że badany dokument jest dokumentem informatycznym.

Klasyfikowania tekstów ciąg dalszy — wielowymiarowy algorytm k-NN

Opisane klasyfikowanie dokumentu na podstawie częstości występowania dwóch ustalonych słów można uogólnić na przypadek, gdy dysponujemy częstościami występowania większej liczby słów kluczowych. W tabeli 1.6 przedstawiona jest statystyka 120 słów występujących najczęściej w angielskiej wersji *Biblii Króla Jakuba* (ang. *King James Bible*), dostępnej w formie e-booka w ramach Projektu Gutenberg.

Tabela 1.6. Statystyka występowania słów w angielskiej wersji Biblii Króla Jakuba

1. the 8,07%	41. when 0,36%	81. go 0,19%
2. and 6,51%	42. this 0,36%	82. hand 0,18%
3. of 4,37%	43. out 0,35%	83. us 0,18%
4. to 1,72%	44. were 0,35%	84. saying 0,18%
5. that 1,63%	45. upon 0,35%	85. made 0,18%
6. in 1,60%	46. man 0,34	87. went 0,18%
7. he 1,31%	47. you 0,34%	88. even 0,18%
8. shall 1,24%	48. by 0,33%	89. do 0,17%
9. for 1,13%	49. Israel 0,32%	90. niw 0,17%
10. unto 1,13%	50. king 0,32%	91. behold 0,17%
11. i 1,11%	51. son 0,30%	92. saith 0,16%
12. his 1,07%	52. up 0,30%	93. therefore 0,16%
13. a 1,04%	53. there 0,29%	94. every 0,16%

[2] Równie dobrze można by przyjąć w tej roli *sinus* wspomnianego kąta, obliczany na podstawie iloczynu wektorowego:

$$\sin(\theta) = \frac{|a_x \cdot b_y - a_y \cdot b_x|}{|a||b|}$$

lub jego *tangens*

$$\mathrm{tg}(\theta) = \frac{|a_x \cdot b_y - a_y \cdot b_x|}{a_x \cdot b_x + a_y \cdot b_y}$$

Obie funkcje są bowiem funkcjami rosnącymi w przeciwieństwie do kosinusa — *przyp. tłum.*

Tabela 1.6. Statystyka występowania słów w angielskiej wersji Biblii Króla Jakuba — *ciąg dalszy*

14. lord 1.00%	54. hath 0,28%	95. these 0,15%
15. they 0,93%	55. then 0,27	96. becase 0,15%
16. be 0,88%	56. people 0,27%	97. or 0,15%
17. is 0,88%	57. came 0,26%	98. after 0,15%
18. him 0,84%	58. had 0,25%	99. our 0,15%
19. not 0,83%	59. house 0,25%	100. things 0,15%
20. them 0,81%	60. on 0,25%	101. father 0,14%
21. it 0,77%	61. into 0,25%	102. down 0,14%
22. with 0,76%	62. her 0,25%	103. sons 0,14%
23. all 0,71%	63. come 0,25%	104. hast 0,13%
24. thou 0,69%	64. one 0,25%	105. David 0,13%
25. thy 0,58%	65. we 0,23%	106. o 0,13%
26. was 0,57%	66. children 0,23%	107. make 0,13%
27. god 0,56	67. s 0,23%	108. say 0,13%
28. which 0,56%	68. before 0,23%	109. may 0,13%
29. my 0,55%	69. your 0,23%	110. over 0,13%
30. me 0,52%	70. also 0,22%	111. did 0,13%
31. said 0,50%	71. day 0,22%	112. earth 0,12%
32. but 0,50%	72. land 0,22%	113. what 0,12%
33. ye 0,50%	74. so 0,21%	114. Jesus 0,12%
34. their 0,50%	75. men 0,21%	115. she 0,12%
35. have 0,49%	76. against 0,21%	116. who 0,12%
36. will 0,48%	77. shalt 0,20%	117. great 0,12%
37. thee 0,48%	78. if 0,20%	118. name 0,12%
38. from 0,46%	79. at 0,20%	119. any 0,12%
39. as 0,44%	80. let 0,19%	120. thine 0,12%
40. are 0,37%		

Naszym zadaniem jest znalezienie metryki, która na podstawie częstości występowania słów kluczowych z ustalonego zbioru w każdym z dwóch dokumentów adekwatnie wyraża ich semantyczne podobieństwo. Wykorzystując tę metrykę w algorytmie k-NN, będziemy mogli klasyfikować semantycznie nowe teksty, na podstawie tych, których semantykę już znamy.

Analiza

Dysponując statystyką N najczęściej występujących słów kluczowych w tym dokumencie, możemy wprost zdefiniować reprezentujący go N-wymiarowy wektor; miarą „odległości" semantycznej między dwoma dokumentami może być wówczas odległość pomiędzy reprezentującymi je wektorami, mierzona w ustalonej metryce (na przykład euklidesowej).

Problem w tym, że tylko niektóre z najczęściej występujących słów kluczowych mają bezpośredni związek z semantyką dokumentu, obecność wielu wynika bowiem z reguł gramatycznych, obowiązujących we wszystkich tekstach w danym języku, bez względu na ich semantykę. Wśród wspomnianych 120 słów najbardziej reprezentatywne wydają się:

lord 1,0%	Israel 0,32%	David 0,13%
god 0,56%	king 0,32%	Jesus 0,12%

Wyrazy te bardziej pasują do treści o charakterze chrześcijańskim niż (na przykład) matematycznym. Inne z kolei są raczej semantycznie neutralne:

the 8,07%	of 4,37%	that 1,63%
and 6,51 %	to 1,72%	in 1,60%

Wykazują one tendencję do częstego występowania w tekstach o różnej tematyce — matematycznych, literackich itp. — a różna ich częstotliwość wynika raczej z przyjętego stylu pisarskiego niż z konkretnej treści. Stąd wniosek, że klasyfikując tekst na podstawie statystyki występowania w nim poszczególnych słów musimy ograniczyć się do wyrazów rzeczywiście istotnych, ignorując te mało znaczące. Nie tylko redukuje to wymiarowość badanej przestrzeni, lecz także zmniejsza prawdopodobieństwo błędnej klasyfikacji. Powrócimy do tej kwestii w problemie nr 6.

Podsumowanie

Ten rozdział poświęcony był algorytmowi *k* najbliższych sąsiadów (ang. *k-nearest neighbor algorithm*) jako algorytmowi klasyfikującemu badany punkt danych pod kątem przynależności do jednej z klas — tej, do której należy większość jego najbliższych sąsiadów. Odległość między punktami mierzona jest według przyjętej metryki, będącej funkcją współrzędnych tych punktów. Spośród rozmaitych możliwych metryk najczęściej stosowanymi są: metryka euklidesowa, metryka Manhattanu oraz metryki oparte na kącie, jaki tworzą promienie wodzące punktów, a dokładniej — na funkcjach trygonometrycznych tego kąta (sinusie, kosinusie lub tangensie). Pokazałem także przykłady eksperymentalnego doboru wartości *k* dla algorytmu k-NN oraz wpływ definiowania różnych metryk na adekwatność rozwiązań. W jednym z przykładów zilustrowałem także znaczenie, jakie dla dokładności wyniku ma redukowanie wymiarowości przestrzeni danych poprzez eliminowanie punktów mniej istotnych z perspektywy rozwiązywanego problemu. Podobne znaczenie ma również właściwe skalowanie danych w każdym wymiarze.

W następnym rozdziale zajmiemy się twierdzeniem Bayesa jako podstawą klasyfikacji elementów w oparciu o teorię prawdopodobieństwa.

Problemy

Zajmiemy się teraz przedyskutowaniem kilku problemów związanych z omawianymi wcześniej zagadnieniami, czyli

- prognozowaniem subiektywnego odczuwania temperatury,
- wyborem wartości k dla algorytmu k-NN,
- przewidywaniem statusu własności nieruchomości.

Zalecam Czytelnikom, w trosce o jak najlepsze zrozumienie tematyki tego rozdziału, próbę samodzielnego rozwiązania, a dopiero później skonfrontowanie własnych przemyśleń z treścią analizy.

Subiektywne odczuwanie temperatury

Problem 1. Wyobraźmy sobie, że Marysia deklaruje odczuwanie zimna przy temperaturze –50°C, lecz temperatura 20°C jest odczuwana przez nią jako ciepło. Jakie będą w tej sytuacji wyniki algorytmu 1-NN dla temperatur 22°C, 15°C i –10°C? Czy Twoim zdaniem algorytm trafnie przewidzi subiektywne odczucia Marysi? Jeśli uważasz, że nie, podaj tego przyczynę i zaproponuj usprawnienie algorytmu w kierunku dokonywania lepszych klasyfikacji.

Problem 2. Czy Twoim zdaniem algorytm 1-NN da lepsze wyniki niż algorytm k-NN dla $k > 1$?

Problem 3. Załóżmy, że wśród deklarowanych przez Marysię preferencji temperaturowych znajdują się takie, że temperatura 17°C odczuwana jest jako „ciepła", zaś temperatura 18°C — jako „zimna"; Marysia czuje się „cieplej" w niższej temperaturze. Jak mógłbyś wyjaśnić występowanie tego rodzaju sprzecznych danych? Jak można by je traktować w procesie analizy? Jak można by zapobiegać ich występowaniu? Czy może danym temperaturowym powinny towarzyszyć dane o innych charakterze? Zakładając, że dysponujemy wyłącznie danymi temperaturowymi — czy algorytm 1-NN wciąż okaże się lepszy niż k-NN, czy może istnieje lepsza wartość $k > 1$?

Mapa Włoch — przykład doboru wartości k

Problem 4. Załóżmy, że nie dysponujemy kompletną mapą Włoch, nie potrafimy więc obliczyć stopnia błędnej klasyfikacji „białych" punktów dla poszczególnych wartości k. Jaką w tej sytuacji zaproponowałbyś metodę wyboru wartości k optymalnej pod względem dokładności uzupełnienia mapy?

Status własności

Problem 5. Używając danych przedstawionych w podrozdziale poświęconym tej tematyce, wykonaj klasyfikację punktu reprezentującego Piotra, wykorzystując metrykę euklidesową w odniesieniu do:

a) danych oryginalnych (nieprzeskalowanych),

b) danych przeskalowanych.

Jak w każdym z tych przypadków przedstawia się sąsiad najbliższy Piotrowi? W którym przypadku sąsiad ten jest właścicielem nieruchomości?

Problem 6. Załóżmy, że wśród zasobów Projektu Gutenberg (*http://www.gutenberg.org*) poszukujemy publikacji podobnej semantycznie do wybranej pozycji (na przykład *Biblii Króla Jakuba*), wykorzystując algorytm 1-NN. Jaką w tym przypadku zaproponowałbyś metrykę mierzącą podobieństwo między publikacjami?

Analiza

Problem 1. Temperatura –8°C bliższa jest 20°C niż –50°C, powinna być więc odczuwana przez Marysię jako „ciepła", co jest raczej mało prawdopodobne w konfrontacji z doświadczeniami większości z nas. W sytuacjach bardziej skomplikowanych zwodnicze wyniki analizy mogą być punktem wyjścia do formułowania fałszywych wniosków, szczególnie gdy osobom wnioskującym brak jest doświadczenia w przedmiotowej dziedzinie. Nie zapominajmy bowiem, że *data science* zasadza się nie tylko na analizie danych, lecz także na rzeczowej wiedzy ekspertów. Warunkiem wyciągnięcia właściwych wniosków z analizy danych jest solidne zrozumienie zarówno problemu, jak i wspomnianych danych zebranych w drodze eksperymentu.

Zgodnie z rozstrzygnięciami algorytmu, temperaturę 22°C powinna Marysia odczuwać jako „ciepłą". „Ciepła" jest bowiem temperatura 20°C i — zgodnie z intuicją — wszystkie od niej wyższe. Także temperaturę 15°C algorytm powinien zaklasyfikować jako „ciepłą" — co jednak w kontekście naszych codziennych doświadczeń nie jest już takie oczywiste.

Nasz algorytm może dać lepsze wyniki, jeżeli dostarczymy mu większej ilości danych eksperymentalnych. Gdyby na przykład Marysia zadeklarowała, że odczuwa temperaturę 14°C jako „zimną", algorytm zaklasyfikuje jako „zimną" temperaturę 15°C, dla której 14°C jest najbliższym sąsiadem.

Problem 2. Nasze dane eksperymentalne są jednowymiarowe (tylko temperatura), klasyfikowane są w dwóch klasach („zimno" albo „ciepło") i mają własność monotoniczności — im wyższa temperatura, tym większe prawdopodobieństwo subiektywnego odczuwania jej jako „ciepłej". Ponadto, nawet gdyby Marysia określiła swe subiektywne odczucia dla temperatur –40, –39, … 39, 40°C w odstępie co jeden stopień, i tak nasz zestaw danych byłby bardzo ograniczony. Najlepiej więc w procesie klasyfikacji ograniczyć się do jednego najbliższego sąsiada.

Problem 3. Główną przyczyną sprzeczności (niespójności) danych jest niewłaściwy sposób pozyskiwania danych eksperymentalnych. Skutki tego łagodzić można głównie przez zwiększenie liczby wykonywanych eksperymentów.

Jest ponadto zrozumiałe, że subiektywne odczuwanie temperatury nie jest tylko funkcją samej temperatury, lecz uzależnione jest od kilku innych czynników: prędkości wiatru, wilgotności powietrza, nasłonecznienia itp. a także ubioru — przecież w określonych warunkach pogodowych Marysia ubrana w płaszcz będzie czuła się inaczej niż odziana w kostium kąpielowy. Wszystkie te czynniki są dobrymi kandydatami na dodatkowe wymiary przestrzeni danych, z pozytywnymi konsekwencjami dla (spodziewanych) wyników klasyfikacji.

Dysponując jednak tylko jednym wymiarem (temperaturą) moglibyśmy zwiększać liczebność danych wejściowych (na przykład deklarując odczucia temperatur w odstępie co 0,1°C) lub liczbę uwzględnianych najbliższych sąsiadów (k) z nadzieją na uzyskanie najlepszej klasyfikacji. Sukces tego podejścia byłby jednak uwarunkowany dostępnością niezbędnych danych. Alternatywnym pomysłem mogłoby być uwzględnianie *wszystkich* sąsiadów zlokalizowanych w odległości nie większej od badanego punktu niż ustalone d zamiast ustalonej ich liczby bez narzucania granicznej odległości.

Problem 4. W celu wyznaczenia wartości k skutkującej największą dokładnością, możemy wykorzystać tzw. **walidację krzyżową** (ang. *cross-validation*), której istota opisana jest w dodatku A. Dzielimy mianowicie zbiór dostępnych danych na dwa podzbiory: **podzbiór uczący** (ang. *learning subset*), wykorzystywany do przeprowadzania analiz (w tym przypadku za pomocą algorytmu k-NN) i **podzbiór testowy** (ang. *test subset*), służący do potwierdzania wiarygodności tychże analiz. W przypadku mapy Włoch możemy zaliczyć w poczet zbioru uczącego 80% znanych pikseli, pozostawiając 20% jako obiekt odniesienia do obliczania udziału prawidłowo sklasyfikowanych pikseli.

Problem 5. Zastosowanie algorytmu 1-NN z metryką euklidesową dla różnych typów danych wejściowych daje następujące rezultaty:

- **a)** dla danych oryginalnych — najbliższym sąsiadem Piotra jest 25-latek z rocznym dochodem 78 000 dolarów; nie jest właścicielem domu, w którym zamieszkuje;
- **b)** dla danych przeskalowanych — najbliższy sąsiad Piotra ma 40 lat, osiąga dochód roczny 60 000 dolarów i zamieszkuje we własnym domu.

Problem 6. Jeżeli miara podobieństwa między dokumentami ma opierać się na rankingach częstotliwości występowania poszczególnych słów, to w rankingu dla danego dokumentu preferowane muszą być słowa reprezentatywne dla tekstów o danej tematyce, czyli wyraźnie częściej występujące w pewnym podzbiorze dokumentów niż w pochodzących spoza tego podzbioru. Zwykły licznik wystąpień danego słowa w danym tekście należy w tym celu zastąpić *częstotliwością ważoną*, czyli odnoszoną do częstotliwości występowania danego słowa w całym zbiorze[3]:

[3] W rozumowaniu tym przyjęto milcząco, że wszystkie dokumenty mają zbliżony rozmiar (mierzony liczbą słów). W sytuacji, gdy rozmiary dokumentów różnią się znacznie między sobą, w ułamku po prawej stronie znaku równości powinny znaleźć się względne częstotliwości występowania danego słowa zamiast bezwzględnych liczb określających jego występowanie — *przyp. tłum.*

$$częstotliwość\_ważona(słowo, dokument) = \frac{liczba\_wystąpień(słowo, dokument)}{liczba\_wystąpień(słowo, zbiór\ dokumentów)}$$

Definicja taka wyraźnie deprecjonuje słowa, które często występują w każdym dokumencie, bez względu na jego tematykę.

Skorowidz

A

algorytm EM, 199
algorytm ID3, 65
algorytm k-NN, 13, 199
  dobór wartości k, 20
  implementacja, 16
  wielowymiarowy, 30
algorytm k-średnich, 108, 199
  implementacja, 115
  klasyfikacja dochodów, 110
algorytm najbliższego sąsiedztwa, 13
algorytmy genetyczne, 199
analiza głównych składowych, 199
analiza regresji, 143, 200
analiza sezonowości, 166, 169
analiza szeregów czasowych, 163, 200
analiza trendu, 164, 166
argumenty wywołania programu, 192

B

bagging, 200
budowanie drzewa decyzyjnego, 65

C

centroid klastera, 109
centrowanie klasterów, 109

D

dane doskonałe, 144
dane niespójne, 77
dobór liczby klasterów, 118
drzewa decyzyjne, 59
drzewo decyzyjne, 200

E

eksploracja tekstu, 200
entropia informacyjna, 61

F

funkcje, 190

G

gałęzie, 60
głębokie uczenie, 200
gra w szachy, 72
grupowanie, 118

I

iloczyn, 195
iloczyn skalarny wektorów, 29
iloczyn wektorowy, 30
implementacja algorytmu ID3, 66
implementacja algorytmu k-NN, 17
implementacja algorytmu konstruowania lasu losowego, 90

implementacja algorytmu k-średnich, 115
implementacja analizy regresji liniowej, 146
implementacja metody spadku gradientowego, 152
implementacja naiwnego klasyfikatora bayesowskiego, 43
instrukcje break i continue, 189
instrukcje warunkowe, 186

J

język Python, 181

K

klasteryzacja, 107, 111, 125
klasyfikacja dochodów, 110
klasyfikowanie dokumentów, 125
klasyfikowanie przez klasteryzację, 111
klasyfikowanie tekstów, 27, 30
komentarze, 182
konstruowanie drzewa decyzyjnego, 65
konstruowanie lasów losowych, 84
konstruowanie lasu losowego, 85, 94
konwersja temperatur, 144
korelacja, 196
krotka, 183

L

las losowy, 200
lasy losowe, 83
  implementacja algorytmu, 90
  klasyfikacja, 84
  klasyfikowanie cechy, 89
lista, 184
losowe drzewa decyzyjne, 97
losowe drzewa decyzyjne XE "losowe drzewa decyzyjne\", 97
losowe drzewo decyzyjne, 87, 200

Ł

łańcuch, 183

M

mapa Włoch, 24
maszyna wektorów nośnych, 200
mediana, 196
metoda najmniejszych kwadratów, 145
metoda spadku gradientowego, 151
metryka Manhattanu, 15
metryki nieeuklidesowe, 27
metryki odległości punktów, 27
miara wiarygodności wyniku, 100
model nieliniowy, 156

N

nachylenie, 196
naiwny klasyfikator bayesowski, 37, 200
najbliższe sąsiedztwo, 13
najbliższe sąsiedztwo punktu, 15
napis, 183

O

obliczanie zysku informacyjnego, 63
obliczenia balistyczne, 156
odchylenie standardowe, 196
odległość między punktami, 21
operacje na plikach, 192

P

PageRank, 200
pętla for, 187
pętla while, 188
populacja, 195
prawdopodobieństwo warunkowe, 37
prognozowanie statusu własności, 25
prognozowanie wagi, 149
próba, 195
przechwycenie, 196
przecięcie, 195
przepływ sterowania, 185
przetrenowanie, 198
przewidywanie czasu przelotu, 154
przyczynowość, 196
Python, 181

R

regresja, 143
regresja liniowa, 144, 145
reguły asocjacyjne a priori, 200
reprezentowanie danych, 59
rozkład normalny Gaussa, 197
rozkład według wartości osobliwych, 201

S

sezonowość, 163, 166, 169
sieć bayesowska, 201
sieć neuronowa, 201
skalowanie danych, 25
słownik, 185
spadek gradientowy, 151
statystyka, 195
subiektywne odczuwanie temperatury, 14, 33
szereg czasowy, 163

Ś

Średnia arytmetyczna, 196

T

temperatura, 14
teoria informacji, 61
testowanie A/B, 198
testy medyczne, 38
trend, 163
trend liniowy, 164, 178
twierdzenie Bayesa, 38, 39, 50
twierdzenie Bayesa dla ciągłych zmiennych losowych, 48
twierdzenie Bayesa rozszerzone, 40
typ danych, 182
  float, 183
  int, 182
  krotka, 183
  lista, 184
  napis, 183
  słownik, 185
  zbiór, 184

U

uczenie zespołowe, 201
ulepszanie, 201

W

walidacja krzyżowa, 140, 197
wariancja, 196
wartość oczekiwana, 196
wejście-wyjście, 192
wektor własny, 201
węzły, 60
wizualizacja, 19
wnioskowanie bayesowskie, 197
wnioskowanie indukcyjne, 201
wnioskowanie z niespójnych danych, 100
wynikowe drzewo decyzyjne, 76
wynikowy las losowy, 89
wyznaczanie centroidu klastera, 109

Z

zagadnienie własne macierzy, 201
zbiór, 184
zbiór centroidów, 109
zdarzenia niezależne warunkujące, 41
zdarzenia zależne warunkujące, 45
zmienna losowa, 196
zysk informacyjny, 63

Algorytmy Data Science: poznaj, zrozum, zastosuj!

Data Science jest interdyscyplinarną dziedziną naukową łączącą osiągnięcia uczenia maszynowego, statystyki i eksploracji danych. Umożliwia wydobywanie nowej wiedzy z istniejących danych poprzez stosowanie odpowiednich algorytmów i analizy statystycznej. Stworzono dotąd wiele algorytmów tej kategorii i wciąż powstają nowe. Stanowią one podstawę konstruowania modeli umożliwiających wyodrębnianie określonych informacji z danych odzwierciedlających zjawiska zachodzące w świecie rzeczywistym, pozwalają też na formułowanie prognoz ich przebiegu w przyszłości. Algorytmy Data Science są postrzegane jako ogromna szansa na zdobycie przewagi konkurencyjnej, a ich znaczenie stale rośnie.

Ta książka jest zwięzłym przewodnikiem po algorytmach uczenia maszynowego. Jej cel jest prosty: w ciągu siedmiu dni masz opanować solidne podstawy siedmiu najważniejszych dla uczenia maszynowego algorytmów. Opisom poszczególnych algorytmów towarzyszą przykłady ich implementacji w języku Python, a praktyczne ćwiczenia, które znajdziesz na końcu każdego rozdziału, ułatwią Ci lepsze zrozumienie omawianych zagadnień. Co więcej, dzięki książce nauczysz się właściwie identyfikować problemy z zakresu Data Science. W konsekwencji dobieranie odpowiednich metod i narzędzi do ich rozwiązywania okaże się dużo łatwiejsze.

W tej książce:

- efektywne implementacje algorytmów uczenia maszynowego w języku Python
- klasyfikacja danych przy użyciu twierdzenia Bayesa, drzew decyzyjnych i lasów losowych
- podział danych na klastery za pomocą algorytmu *k*-średnich
- stosowanie analizy regresji w parametryzacji modeli przewidywań
- analiza szeregów czasowych pod kątem trendów i sezonowości danych

Dávid Natingga — jest naukowcem specjalizującym się w dziedzinie sztucznej inteligencji. Zajmuje się teorią obliczeń i wykorzystaniem matematyki w algorytmach SI. Wcześniej optymalizował algorytmy na potrzeby uczenia maszynowego oraz *big data*. Jest autorem ciekawego algorytmu sugerowania produktów na podstawie preferencji klientów i cech gatunków kawy. W 2016 roku spędził osiem miesięcy jako *research visitor* w Japońskim Instytucie Naukowo-Technologicznym w Kanazawie.

ISBN 978-83-283-5602-3