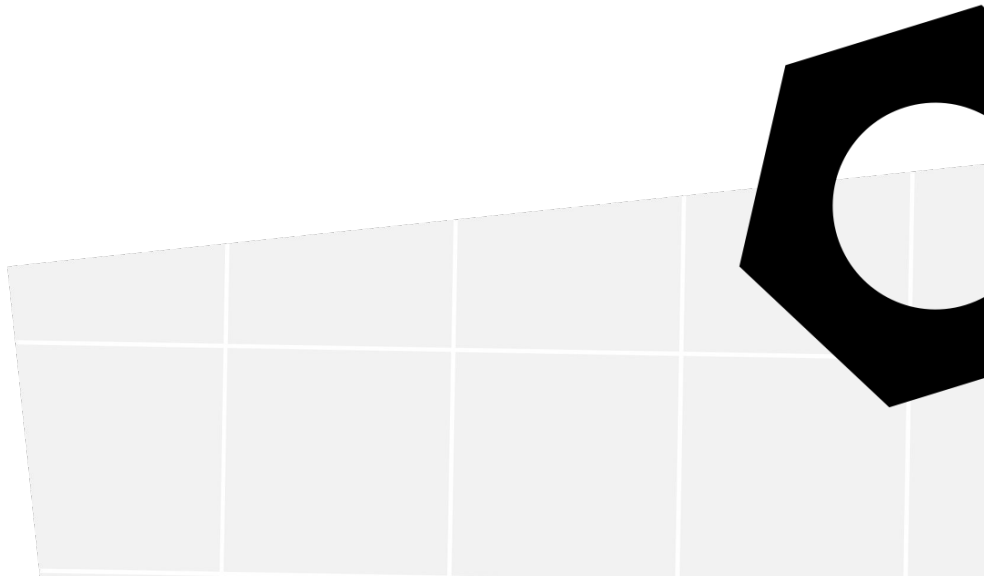
Statystyka i prawdopodobieństwo

Kurs Data Science



software development academy

Zadania praktyczne podczas zajęć

W prezentacji do bloku **Statystyka i prawdopodobieństwo** zamieszczone zostały dodatkowe zadania praktyczne.

Rozwiązania do nich możesz pobrać z sekcji “Dodatkowe materiały do bloku” tutaj.

Elementy algebry liniowej z użyciem numpy

Elementy algebry liniowej z użyciem numpy

software
development
academy

Elementy algebry liniowej

- Skalary, wektory, macierze, tensory
- Operacje na macierzach i wektorach
- Wyznacznik macierzy
- Rozkład na wartości własne

Wektory

TROCHĘ TEORII

Wektor to tablica liczb, których identyfikacja jest możliwa za pomocą indeksu określającego położenie.

Wektory identyfikują punkty w przestrzeni. Załóżmy, że mamy wektor x. Pierwszym elementem tego wektora jest x1, drugim x2, a ostatnim xn, co przedstawiono na poniższym rysunku:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$$

PRAKTYCZNE ZASTOSOWANIE Z UŻYCIEM BIBLIOTEKI NUMPY

```
import numpy as np
```

Tworzenie macierzy

Wektor jako wiersz

```
A = np.array([3,6,8,1])
A
```

```
array([3, 6, 8, 1])
```

Wektor jako kolumna

```
A1 = np.array([[3], [6], [8], [1]])
A1
```

```
array([[3],
       [6],
       [8],
       [1]])
```

Macierze

TROCHĘ TEORII

Macierz to, podobnie jak wektor, tablica liczb, tyle że dwuwymiarowa. Identyfikacja konkretnej liczby jest możliwa za pomocą dwóch indeksów określających położenie.

Macierze są zapisywane w postaci prostokątnej tablicy i są oznaczane zazwyczaj dużą literą alfabetu, co pokazano na przykładzie poniższej macierzy:

Elementy aji nazywamy elementami macierzy. Przykładowo element a12 znajduje się w pierwszym wierszu i drugiej kolumnie macierzy, bowiem wiersze numerujemy "od góry", a kolumny - "od lewej strony". Prezentowana macierz ma wymiary m x n.

Macierz 2×3 (2 wiersze 3 kolumny)

```
B = np.array([[3,7,4], [9,3,5]])
B
```

```
array([[3, 7, 4],
       [9, 3, 5]])
```

Tworzenie macierzy za pomocą np.arange()

```
C = np.arange(2,8)
C
```

```
array([2, 3, 4, 5, 6, 7])
```

Przy uzyciu np.linspace(start, end, count) – (0,1,5) – 5 liczb od 0 do 1 w równych odległościach

```
D = np.linspace(0,1,5)
D
```

```
array([ 0.  ,  0.25,  0.5 ,  0.75,  1.  ])
```

Macierze

Opis macierzy

Tworzymy macierz 3×3

```
m = np.array([[6,8,3], [4,2,6], [8,4,3]])
m
```

```
array([[6, 8, 3],
       [4, 2, 6],
       [8, 4, 3]])
```

Ilość wierszy i kolumn:

```
m.shape
```

```
(3, 3)
```

Ilość elementów w macierzy:

```
m.size
```

```
9
```

Ilość wymiarów:

```
m.ndim
```

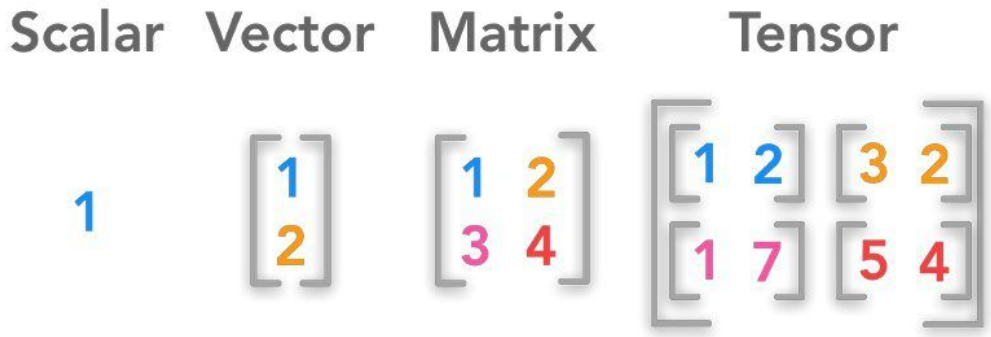
```
2
```

Tensor

Czasami potrzebujemy tablic o większej liczbie wymiarów.

Ogólnie rzecz biorąc, liczby w tablicy tworzą regularną siatkę ze zmienną liczbą osi, którą nazywamy tensorem. Tensor to „pojemnik” na dane. Możemy mieć tensor 0D, który zawiera jeden skalar.

Skalar jest tensorem 0-wymiarowym (0D). Ma zatem 0 osi i ma rangę 0 (mowa tensora dla „liczby osi”). I tu pojawia się niuans: pojedyncza liczba może być wyrażona jako tensor, nie oznacza to, że tak powinno być, ani ogólnie tak jest. Istnieje dobry powód, aby móc je traktować jako takie (operacje tensorowe), ale jako mechanizm przechowywania, ta zdolność może być myląca. Tensor 1D nazywamy wektorem. Tensor 2D to macierz.



Definiowanie tensorów w numpy można wykonać za pomocą np.array. Na przykład tensor A można zaimplementować w następujący sposób:

A = np. tablica ([[3,7,4], [9,3,5], [2,0,1]])

Tworzy dwuwymiarową macierz (tensor 2D)

Dostęp do elementów macierzy

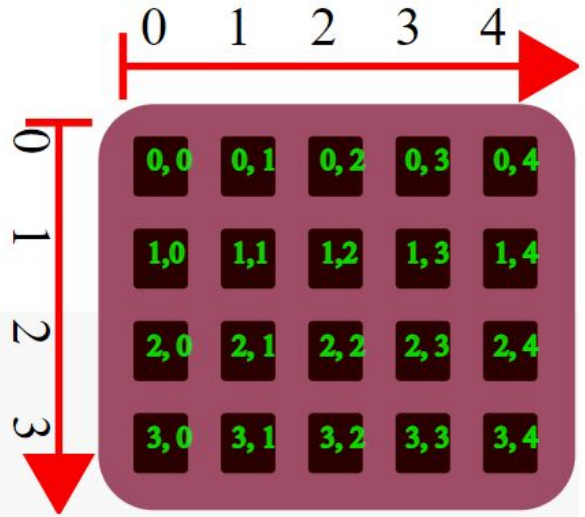
Dostęp do elementów (i pod-macierzy) jest możliwy poprzez wykorzystanie notacji indeksowej (`macierz[i]`) jak i wycinkowej (`macierz[i:j]`).

Dostęp do pojedynczego elementu:

```
>>> A = array([[1, 2, 3],[4,5,6]])
>>> A
array([[1, 2, 3],
       [4, 5, 6]])
>>> A[0][2]    # podobnie jak w Pythonie,numeracja od 0
3
>>> A[0, 2]
3
```

Pod-macierze

Macierz `A` jest tablicą dwuwymiarową, i sposób numerowania zawartych w niej obiektów jest następujący: pierwszy indeks przebiega pierwszy wymiar (wybiera wiersz), drugi indeks przebiega drugi wymiar (wybiera kolumnę).



Pod-macierze

Dostęp do pod-macierzy:

```
>>> A[1]            # wiersz 1
array([4, 5, 6])
>>> A[1, :]         # wiersz 1, wszystkie kolumny
array([4, 5, 6])
>>> A[:, 1]         # wszystkie wiersze, kolumna 1
array([2, 5])
```

Jak widać, ograniczenie się do pojedynczego punktu w danym wymiarze, powoduje degenerację tego wymiaru. Uzyskuje się macierz, w której liczba wymiarów jest mniejsza o jeden.

```
>>> A[:, 1:]
array([[2, 3],
       [5, 6]])
```

W pierwszym wymiarze (wiersze) bierzemy wszystko, natomiast w drugim od 1 do końca. Efektywnie wycinamy kolumnę 0.

Generowanie liczb losowych

Losowo wybrana liczba(y) z rozkładu normalnego (Gaussa)

```
np.random.normal()
```

```
0.2915993384486327
```

```
np.random.normal(size=6)
```

```
array([ 0.91551674,  1.16095325, -0.20223517, -0.64148055, -2.22242932,
        0.84629985])
```

Losowo wybrane liczby z układu jednolitego

```
np.random.uniform(size=6)
```

```
array([ 0.09001786,  0.24629594,  0.83394455,  0.70530724,  0.62448293,
        0.33314339])
```

Losowo wybrane liczby z określonego przedziału

```
np.random.randint(3,30, size=6)
```

```
array([ 7, 18, 13, 14, 24,  4])
```

Losowo wybrana liczba zmiennoprzecinkowa (float)

```
np.random.random()
```

```
0.37724086595580364
```

Indeksowanie

Tworzenie macierzy 3×3

```
m3x3 = np.array([[6,8,3], [4,2,6], [8,4,3]])
m3x3
```

```
array([[6, 8, 3],
       [4, 2, 6],
       [8, 4, 3]])
```

Wybierz pierwszy wiersz, drugą kolumnę (pamietaj indeksowanie w Pythonie zaczyna się od 0!).

```
m3x3[0,1]
```

```
8
```

Wybierz cały trzeci wiersz

```
m3x3[2,:]
```

```
array([8, 4, 3])
```

Wybierz całą drugą kolumnę

```
m3x3[:,1]
```

```
array([8, 2, 4])
```

Wybór warunkowy – wybierz indeks piłkarzy wyższych niż 1.8m

```
wzrost_pilkarzy = np.array([1.72, 1.65, 1.88, 1.91, 1.82, 1.76, 1.89])
wysocy = np.where(wzrost_pilkarzy>1.8)
wysocy
```

```
(array([2, 3, 4, 6], dtype=int64),)
```

Aby uzyskać faktyczne wartości wzrostu piłkarzy, indeksujemy powyższy filtr

```
wzrost_pilkarzy[wysocy]
```

```
array([ 1.88,  1.91,  1.82,  1.89])
```

Zadanie 1

Stwórz numpy array z wartościami od 1 do do 9, wyświetl, a następnie odwróć tę tablicę (pierwszy element zostaje ostatnim, itd.).

Zadanie 2

Stwórz następujący numpy array: [1, 23, 4, 31, 1, 1, 4, 23, 4, 1]; a następnie wypisz wszystkie unikalne (nie powtarzające się) elementy.

Podstawowe działania na macierzach

Dodawanie macierzy

Odejmowanie macierzy

```
import numpy as np
```

Tworzymy 2 macierze do dalszych obliczeń

```
A = np.array([[3,4,5], [1,2,3], [4,5,6]])
B = np.array([[-2, 5,1], [7, 0, 2], [-1, 0,5]])
print ('macierz A', A)
print('macierz B', B)
```

```
macierz A [[3 4 5]
 [1 2 3]
 [4 5 6]]
macierz B [[-2  5  1]
 [ 7  0  2]
 [-1  0  5]]
```

- możemy dodawać i odejmować macierze ale tylko tych samych rozmiarów
- możemy mnożyć macierze jeśli liczba kolumn pierwszej jest równa liczbie wierszy drugiej

```
# dodawanie A+B
np.add(A,B)
```

```
array([[ 1,  9,  6],
       [ 8,  2,  5],
       [ 3,  5, 11]])
```

```
# odejmowanie A-B
np.subtract(A,B)
```

```
array([[ 5, -1,  4],
       [-6,  2,  1],
       [ 5,  5,  1]])
```

Podstawowe działania na macierzach

Mnożenie macierzy

Transpozycja macierzy

Macierz odwrotna

```
# mnożenie
A*B
```

```
array([[-6, 20,  5],
       [ 7,  0,  6],
       [-4,  0, 30]])
```

```
# mnożenie macierzy przez liczbę
5*B
```

```
array([[-10,  25,   5],
       [ 35,   0,  10],
       [ -5,   0,  25]])
```

Macierz transponowana do macierzy , to macierz , która powstaje przez zamianę wierszy na kolumny.

```
# macierz A
A
```

```
array([[3, 4, 5],
       [1, 2, 3],
       [4, 5, 6]])
```

```
# transpozycja macierzy A
A.T
```

```
array([[3, 1, 4],
       [4, 2, 5],
       [5, 3, 6]])
```

Macierz odwrotna jest określona tylko dla macierzy kwadratowych, których wyznacznik jest niezerowy.

```
C = np.array([[1, 4],[2, 5]])
np.linalg.inv(C)
```

```
array([[-1.66666667,  1.33333333],
       [ 0.66666667, -0.33333333]])
```

Podstawowe operacje na macierzach

Zmiana kształtu macierzy

Wyznaczanie głównej przekątnej

np.reshape – Daje nowy kształt macierzy bez zmiany jej danych

```
# macierz wyjściowa D o wymiarach 4x3
D = np.array([[3,4,5], [1,2,3], [4,5,6], [7,3,1]])
D
```

```
array([[3, 4, 5],
       [1, 2, 3],
       [4, 5, 6],
       [7, 3, 1]])
```

```
# zmieniamy kształt macierzy na 6x2
D.reshape(6,2)
```

```
array([[3, 4],
       [5, 1],
       [2, 3],
       [4, 5],
       [6, 7],
       [3, 1]])
```

Aby otrzymać przekątną macierzy A:

```
print (A)
A.diagonal()
```

```
[[3 4 5]
 [1 2 3]
 [4 5 6]]

array([3, 2, 6])
```

```
# suma przekątnej
A.diagonal().sum()
```

```
11
```

Podstawowe operacje na macierzach

Spłaszczanie macierzy

Funkcje statystyczne

Spłaszczenie macierzy B do wektora, domyślnie wierszowo, jeśli chcemy kolumnowo – parametr: order='F'

```
# macierz wyjściowa B
B
```

```
array([[-2,  5,  1],
       [ 7,  0,  2],
       [-1,  0,  5]])
```

```
# i spłaszczona wierszowo
B.flatten()
```

```
array([-2,  5,  1,  7,  0,  2, -1,  0,  5])
```

inne podstawowe funkcje, które możemy użyć na macierzach

```
print (np.max(A))
print (np.min(A))
print (np.mean(A))
print (np.var(A))
print (np.std(A))
```

```
6
1
3.66666666667
2.22222222222
1.490711985
```

Macierzą jednostkową nazywamy macierz kwadratową, która na przekątnej ma 1 , a pozostałe elementy poza przekątną to 0.

```
np.identity(3)
```

```
array([[ 1.,  0.,  0.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  1.]])
```

Zadanie 3

Stwórz macierz 3x3 (używając reshape) z wartościami od 2 do 10.

Zadanie 4

Stwórz tablicę z sześcioma losowymi wartościami z przedziału od 10 do 30.

Wyznacznik macierzy

Obliczanie wyznacznika macierzy:

Do obliczenia wyznacznika macierzy – np.linalg.det()

np.linalg.det(B)
-184.99999999999991

Aby obliczyć wyznacznik macierzy 1 na 1 należy skorzystać z wzoru:

$$det(A) = |\, a_{11} \,| = a_{11}$$

np.

$$det(A) = |\, 5 \,| = 5$$

Aby obliczyć wyznacznik macierzy 2 na 2 wystarczy skorzystać z wzoru:

$$det(A) = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11} \cdot a_{22} - a_{21} \cdot a_{12}$$

np.

$$det(A) = \begin{vmatrix} -3 & 5 \\ -4 & 2 \end{vmatrix} = (-3) \cdot 2 - (-4) \cdot 5 = -6 \cdot -(-20) = -6 + 20 = 14$$

Obliczając wyznacznik macierzy 3 na 3 korzystamy z wzory:

$$det(A) = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11} \cdot a_{22} \cdot a_{33} + a_{12} \cdot a_{23} \cdot a_{31} +$$

$$+ a_{13} \cdot a_{21} \cdot a_{32} - a_{13} \cdot a_{22} \cdot a_{31} - a_{11} \cdot a_{23} \cdot a_{32} - a_{12} \cdot a_{21} \cdot a_{33}$$

Bardzo dobrym sposobem obliczania wyznacznika macierzy 3 na 3 jest metoda Sarrusa.

Wyznacznik macierzy 4 na 4 posiada dość długą formę dlatego wyznaczniki 4 na 4, 5 na 5 i większe najlatwiej obliczać metodą Laplace'a.

Obliczanie wartości własnych i wektorów własnych

Za pomocą numpy.linalg.eig:

```
matrix = numpy.ones((M,M), dtype=float);
values, vectors = numpy.linalg.eig(matrix);
```

Obliczanie wartości i wektorów własnych krok po kroku

1. Na starcie daną masz macierz **kwadratową** (wyłącznie), powiedzmy A. Tylko.
2. Obliczasz macierz $A_\lambda = A - \lambda I$ gdzie λ to jakaś **liczba**, która jest **niewiadomą**, a I to macierz jednostkowa (czyli kwadratowa, która ma jedynki na przekątnej, a poza nimi same zera).
3. Liczysz wyznacznik macierzy A_λ.
4. Ten wyznacznik to tzw. **równanie charakterystyczne macierzy**. Przyrównujesz go do zera i liczysz jego pierwiastki. Te pierwiastki to właśnie **wartości własne macierzy**. Oznaczasz je $\lambda_1, \lambda_2, \lambda_3, \ldots$.
5. Pierwiastki wstawiasz kolejno do równania: $A_\lambda X = 0$, gdzie X jest niewiadomym **wektorem** (czyli macierzą jednokolumnową). Rozwiązujesz te równanie. Rozwiązaniem będzie pewien zbiór wektorów X, z których każdy można nazwać **wektorem własnym**.

Zadanie 5

Spośród tablicy wartości: [23, 45, 112, 150, 43, 254, 95, 8] wyfiltruj te, które są większe od 100.

Zadanie 6

Stwórz poniższą macierz 4x4, a następnie:

- wyświetl element z drugiego wiersza i trzeciej kolumny,
- oblicz jej wyznacznik,
- oblicz jej ślad (sumę elementów na głównej przekątnej),
- znajdź element największy i najmniejszy.

$$\begin{bmatrix} 1 & 15 & 4 & 13 \\ 8 & 21 & 3 & 12 \\ 11 & 13 & 11 & 5 \\ 32 & 13 & 0 & 2 \end{bmatrix}$$

Statystyka – podstawowe informacje

Głównym celem statystyki jest pozyskiwanie, prezentacja i analiza danych. Statystyka pozwala na wnioskowanie z surowych danych. Np. mając próbkę danych pewnej populacji możemy podjąć próbę wyznaczenia wartości oczekiwanej dla średniej. Następnie, można zweryfikować hipotezę, czy obliczona wartość jest zgodna z założeniami. Istnieje wiele dziedzin statystyki, które poddają analizie dane w bardzo różny sposób. Możemy wyróżnić:

Statystyka opisowa – głównym celem jest opis danych statystycznych uzyskanych podczas badania statystycznego za pomocą wybranych parametrów, dzięki czemu można dokonać pewnych uogólnień na temat danych, jak również wyciągnąć podstawowe wnioski.

Testowanie hipotez – głównym celem testowania hipotez jest weryfikacja ogólnie postawionych założeń na temat rozkładu, bądź określonych parametrów (takich jak wariancja, czy średnia).

Analiza korelacji i regresji – głównym celem jest określenie współzależności zjawisk za pomocą korelacji oraz wyznaczenie funkcji zależności pomiędzy zmiennymi za pomocą regresji.

Analiza szeregów czasowych – głównym celem jest badanie zależności obserwowanych danych w stosunku do określonego stałego przedziału czasowego (np. godzina, dzień, miesiąc).

Głównymi pakietami w Pythonie do zastosowania w statystyce są:

- NumPy
- SciPy

Statystyki opisowe

Wśród podstawowych elementów opisujących zbiorowość, zaliczanych do statystyki opisowej możemy wyróżnić:

- wartość minimalną
- wartość maksymalną
- średnią
- odchylenie standardowe
- medianę

Podstawowe statystyki można znaleźć w bibliotece numpy.
Przykład:
import numpy as np from scipy.stats import scoreatpercentile

data = np.loadtxt(„przykładowyplik.csv", delimiter=',', skiprows=0, unpack=True) print(data)

Statystyki opisowe

Kolejne statystyki opisowe można znaleźć w module ***statistics***. Można tu wyróżnić np.

- ***median_high()*** – zwraca górną granicę mediany

- ***median_low()*** – zwraca dolną granicę mediany

- ***mode()*** – zwraca wartość najczęściej występującą

- **pvariance(), variance()** – zwraca wariancję

- **pstdev(), stdev()** – zwraca odchylenie standardowe

Kolejną biblioteką gromadzącą najwięcej funkcji statystycznych i najbardziej odpowiednią do zastosowań w statystyce jest biblioteka ***scipy.stats***. Wśród funkcji należących do podstawowych statystyk opisowych można zaliczyć:

- ***kurtosis()*** – zwraca miarę koncentracji – Kurtozę. Informuje ona o poziomie spłaszczenia rozkładu. Wartość 0 informuje o spłaszczeniu zbliżonym do normalnego, natomiast liczba dodatnia mówi o większej koncentracji wokół średniej, a liczba ujemna o większym spłaszczeniu rozkładu.

- ***skewness()*** – zwraca skośność rozkładu. Gdy wartość jest bliska zero, oznacza, że rozkład jest symetryczny. Gdy wartość jest mniejsza od zera – rozkład lewostronnie skośny, gdy większa od zera – rozkład prawostronnie skośny.

- ***describe()*** – zwraca podstawowe statystyki opisowe rozkładu

Zadanie 7

Stwórz dowolną tablicę wartości, a następnie oblicz średnią i medianę.

Zadanie 8

Stwórz dowolną tablicę wartości, a następnie je znormalizuj - tj. od każdego elementu tablicy odejmij średnią i podziel przez odchylenie standardowe.

Zadanie 9

Stwórz funkcję, która przyjmuje dwa wektory (oba o tym samym wymiarze) w postaci numpy tablicy, a zwróci odległość euklidesową między nimi.

$$d(p, q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

Co to jest prawdopodobieństwo?

Rachunek prawdopodobieństwa pomaga obliczyć **szansę** zaistnienia pewnego określonego zdarzenia.

Przykład:

Jaka jest szansa, że dziś jest niedziela?

Mamy 7 możliwości (7 dni tygodnia), zatem szansa na to, że dziś jest niedziela, wynosi 1/7.

Rachunek prawdopodobieństwa opiera się na kombinatoryce

Żeby obliczyć szansę dowolnego zdarzenia (nazwijmy go literką A), musimy określić liczbę zdarzeń sprzyjających oraz liczbę wszystkich możliwych zdarzeń (do tego celu stosujemy kombinatorykę). Następnie do obliczenia prawdopodobieństwa korzystamy z jednego wzoru:

$$P(A) = \frac{|A|}{|\Omega|}$$

gdzie:
|A| - to liczba zdarzeń sprzyjających (moc zbioru |A|)
|Ω| - to liczba wszystkich możliwych zdarzeń (moc zbioru |Ω|)

Pojęcia stosowane w rachunku prawdopodobieństwa:
- **Doświadczenie losowe** - czynność, którą wykonujemy, np.: rzut kostką, wybór dnia tygodnia.
- **Zdarzenie elementarne** - zdarzenie (tylko jedno!) jakie może wydarzyć się w doświadczeniu losowym, np.: wypadło 5 oczek, wybrano środę.
- **Zdarzenie losowe** - zbiór jednego lub kilku zdarzeń elementarnych, np.: wypadła parzysta liczba oczek (2, 4 lub 6), wybrano dzień powszedni.
- **Moc zbioru** - liczba elementów danego zbioru, np.: |{2,4,6}|=3, |{dni powszednie}|=5.

Własności prawdopodobieństwa

• Prawdopodobieństwo dowolnego zdarzenia losowego A jest zawsze liczbą z przedziału $\langle 0;1 \rangle$.
$0 \leq P(A) \leq 1$

• Prawdopodobieństwo zdarzenia pewnego jest równe 1.
$P(\Omega)=1$

•Prawdopodobieństwo zdarzenia niemożliwego jest równe 0.
$P(\varnothing)=0$

Prawdopodobieństwo warunkowe

- **Prawdopodobieństwo zajścia jakiegoś zdarzenia pod warunkiem, że zaistniało jakieś inne zdarzenie np.:**

Wykonano rzut sześcienną kostką do gry. Wyznacz prawdopodobieństwo wyrzucenia więcej niż trzech oczek, jeśli wiadomo, że wypadła parzysta liczba oczek.
Rozwiązanie:
Wprowadźmy oznaczenia:
Ω - jeden rzut kostką do gry,
A - wyrzucono więcej niż trzy oczka,
B - wyrzucono parzystą liczbę oczek,
A∩B - wyrzucono parzystą liczbę oczek większą od 3.

Chcemy obliczyć prawdopodobieństwo zajścia zdarzenia A pod warunkiem, że zaszło zdarzenie B. Skorzystamy ze wzoru:
$P(A|B)=P(A\cap B)/P(B)$

Prawdopodobieństwo całkowite (w teorii)

Jeżeli zdarzenia B1, B2, ..., Bn są parami rozłączne oraz mają prawdopodobieństwa dodatnie, które sumują się do jedynki, to dla dowolnego zdarzenia A zachodzi wzór:

$$P(A)=P(A|B1)\cdot P(B1)+P(A|B2)\cdot P(B2)+...+P(A|Bn)\cdot P(Bn)$$

Prawdopodobieństwo całkowite (w praktyce)

Przykład:

W urnie mamy 10 kul białych i 7 kul czarnych. Wyciągamy jedną losową kulę i wyrzucamy ją, nie sprawdzając koloru. Jaka jest szansa wyciągnięcia za drugim razem kuli białej?

Rozwiązanie:
Wprowadźmy oznaczenia:
A - za drugim razem wyciągnęliśmy kulę białą,
B - za pierwszym razem wyciągnęliśmy kulę białą,
C - za pierwszym razem wyciągnęliśmy kulę czarną.
Chcemy obliczyć prawdopodobieństwo zajścia zdarzenia A. Stosujemy wzór na prawdopodobieństwo całkowite:

$$P(A) = P(A|B) \cdot P(B) + P(A|C) \cdot P(C) = \frac{9}{16} \cdot \frac{10}{17} + \frac{10}{16} \cdot \frac{7}{17} = \frac{90+70}{16 \cdot 17} = \frac{10}{17}$$

Twierdzenie i wzór Bayesa – wstęp teoretyczny

Jeżeli zdarzenia $B_1, B_2, ..., B_n$ wykluczają się parami i mają prawdopodobieństwa dodatnie, to dla każdego zdarzenia A zawartego w sumie zdarzeń $B_1 \cup B_2 \cup ... \cup B_n$:

$$P\left(B_k \middle| A\right) = \frac{P(A|B_k)P(B_k)}{P(A|B_1)P(B_1)+P(A|B_2)P(B_2)+...+P(A|B_n)P(B_n)}$$

Powyższy wzór nazywamy wzorem Bayesa. Twierdzenie Bayesa stosujemy głównie wtedy, gdy znamy wynik doświadczenia i pytamy o jego przebieg.

NB w praktyce

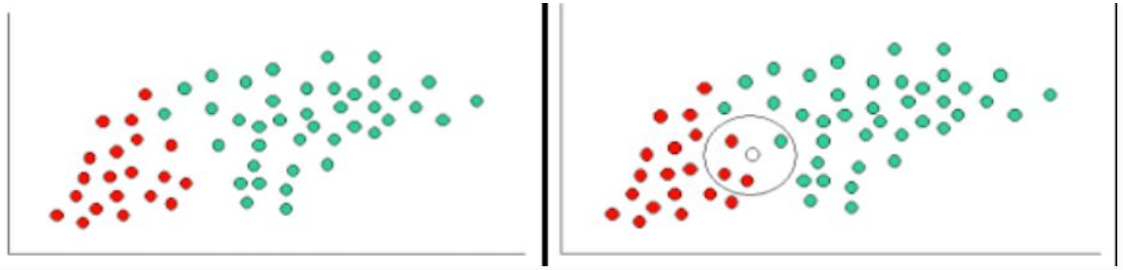


Figure 7.10. Naive Bayes

Dla ilustracji koncepcji Naiwnej metody Bayesa, rozpatrzmy przykład z powyższego rysunku. Jak widać, mamy tu obiekty zielone i czerwone. Naszym zadaniem będzie zaklasyfikowanie nowego obiektu, który może się tu pojawić.

Ponieważ zielonych kółek jest dwa razy więcej niż czerwonych, rozsądnie będzie przyjąć, że nowy obiekt (którego jeszcze nie mamy) będzie miał dwa razy większe prawdopodobieństwo bycia zielonym niż czerwonym.

W analizie Bayesowskiej, takie prawdopodobieństwa nazywane są prawdopodobieństwami a priori. Prawdopodobieństwa a priori wynikają z posiadanych, wcześniejszych (a priori) obserwacji. W tym wypadku, chodzi o procent zielonych względem czerwonych. Prawdopodobieństwa a priori często służą do przewidywania klasy nieznanych przypadków, zanim one się pojawią.

Mając obliczone prawdopodobieństwa a priori, jesteśmy gotowi do zaklasyfikowania nowego obiektu (kółko białe). Ponieważ obiekty są dobrze pogrupowane sensownie będzie założyć, że im więcej jest zielonych (albo czerwonych) obiektów w pobliżu nowego obiektu, tym bardziej prawdopodobne jest, że obiekt ten ma kolor zielony (czerwony). Narysujmy więc okrąg wokół nowego obiektu, taki by obejmował, wstępnie zadaną liczbę obiektów (niezależnie od ich klasy). Teraz będziemy mogli policzyć, ile wewnątrz okręgu jest zielonych, a ile czerwonych kółek. Skąd obliczymy wielkość, którą można nazwać szansą.

Jasne jest, że w powyższym przykładzie szansa, że X będzie czerwone jest większa niż szansa, że X będzie zielone.

Mimo, że prawdopodobieństwo a priori wskazuje, że X raczej będzie zielone (bo zielonych jest dwa razy więcej niż czerwonych), to szanse są odwrotne, ze względu na bliskość czerwonych. Końcowa klasyfikacja w analizie Bayesowskiej bazuje na obu informacjach, wg reguły Bayesa (Thomas Bayes 1702-1761).

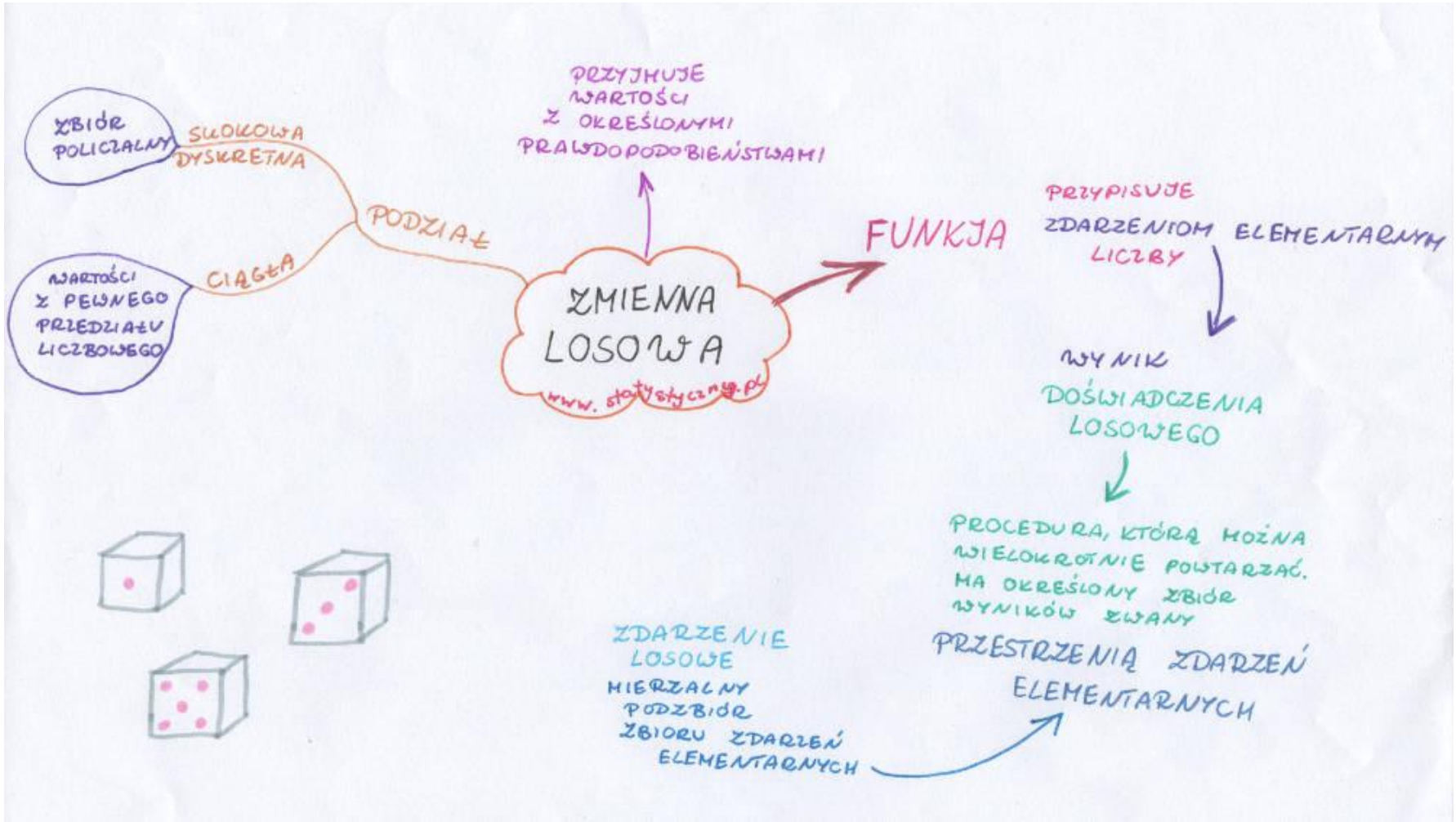
Przykład praktyczny – detekcja spamu

$$P(spam|words) = \frac{P(words|spam)P(spam)}{P(words)}$$

P(spam) - prawdopodobieństwo, że wiadomość jest spamem

P(spam|words) - prawdopodobieństwo, że wiadomość jest spamem, gdy słowo należy do czarnej listy

Zmienna losowa



Rozkład prawdopodobieństwa zmiennej losowej

W teorii, **rozkład prawdopodobieństwa** jest matematyczną funkcją, która zapewnia prawdopodobieństwo wystąpienia różnych możliwych wyników w eksperymencie.

Na przykład:

jeśli zmienna losowa *X* stosuje się do określenia wyników z monetą („eksperyment”), a następnie rozkład prawdopodobieństwa *X* będzie mieć wartość 0,5 dla *X* = głowy i 0,5 dla *X* = ogona (zakładając, że moneta jest fair). Przykłady zjawisk losowych mogą zawierać wyniki z eksperymentu lub badania.

Rozkład prawdopodobieństwa - rodzaje

- Rozkład ciągły (kiedy dane, które analizujemy są ciągłe, np. zmiennoprzecinkowe: informacje o BMI (ang. Body mass index), zarobkach).

- Rozkład dyskretny lub skokowy (gdy dane zmieniają się skokowo np. kolor oczu, kolor włosów, marka samochodu).

Rozkłady dyskretne użyteczne w data science

- Rozkład Bernouliiego
- Rozkład wielomianowy
- Rozkład Poissona

Rozkład Bernoulliego

Jest to rozkład prawdopodobieństwa, który przyjmuje wartość 1 (sukces) z zadanym prawdopodobieństwem p, oraz wartość 0 z prawdopodobieństwem q = 1 – p.

Do wygenerowania zbioru danych oraz policzenia podstawowych statystyk może posłużyć następujący kod:

```
import scipy.stats as scs

p = 0.3
data = scs.bernoulli.rvs(p, size=1000)
mean, var, skew, kurt = scs.bernoulli.stats(p, moments = 'mvsk')
```

Metoda rvs generuje zbiór danych wielkości size z podanym prawdopodobieństwem, natomiast metoda stats oblicza podstawowe statystyki

Rozkład dwumianowy jako szczególny przypadek wielomianowego

Rozkład Dwumianowy

Jest to rozkład prawdopodobieństwa opisujący liczbę sukcesów (na ogół oznaczanych jako k) przy wykonaniu N niezależnych eksperymentów. Każdy z eksperymentów ma założone to samo prawdopodobieństwo sukcesu.

Funkcja prawdopodobieństwa określona jest jako:

`scs.binom.pmf(k,n,p)` i zwraca prawdopodobieństwo k sukcesów w n próbach z prawdopodobieństwem sukcesu równym p.

Rozkład Poissona

Jest to rozkład, który wyraża prawdopodobieństwo zdarzeń następujących po sobie z daną częstotliwością. Zdarzenia te zachodzą niezależnie. Np. ilość nieobecności w ciągu miesiąca w pewniej grupie można opisać rozkładem Poissona z intensywnością $\lambda = 2$ (może się zdarzyć że w miesiącu były 3 nieobecności, a w innym 1. Średnia wyjdzie 2).

Do narysowania rozkładu Poissona można posłużyć się kodem:

```
import scipy.stats as scs
import numpy as np
import matplotlib.pyplot as plt
mu = 2

x = np.arange(scs.poisson.ppf(0.01, mu), scs.poisson.ppf(0.99, mu))
rv = scs.poisson(mu)
fig,ax = plt.subplots(1 ,1)
ax.vlines(x,0,rv.pmf(x), colors='b', linestyles='-', lw=1, label='frozen
pmf')
ax.legend(loc='best', frameon=False)
plt.show()
```

gdzie pmf – funkcja prawdopodobieństwa, ppf – funkcja procentu w punkcie, funkcja kwantylu, arrange – przygotowuje tablicę danych od punktu do punktu,

Rozkłady ciągłe użyteczne w data science

- Rozkład normalny (Gaussa)
- Rozkład t-Studenta

Rozkład normalny (Gaussa)

Najbardziej popularny rozkład ciągły i najszerzej stosowany w statystyce. Gęstość prawdopodobieństwa standardowego rozkładu normalnego jest dana wzorem:

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

Do wygenerowania wykresu gęstości możemy posłużyć się następującym kodem:

```
import scipy.stats as sps
import matplotlib.pyplot as plt
import numpy as np
fig, ax = plt.subplots(1, 1)

x = np.linspace(sps.norm.ppf(0.01), sps.norm.ppf(0.99), 100)
print(x)
ax.plot(x, sps.norm.pdf(x), 'r-', lw=6, alpha=0.3, label='Normalny -
teoretyczny')
rv = sps.norm(loc = 0, scale = 1)
ax.plot(x, rv.pdf(x), 'k-', lw=3, label='Normalny - z próby')
ax.legend(loc='best')
plt.show()
```

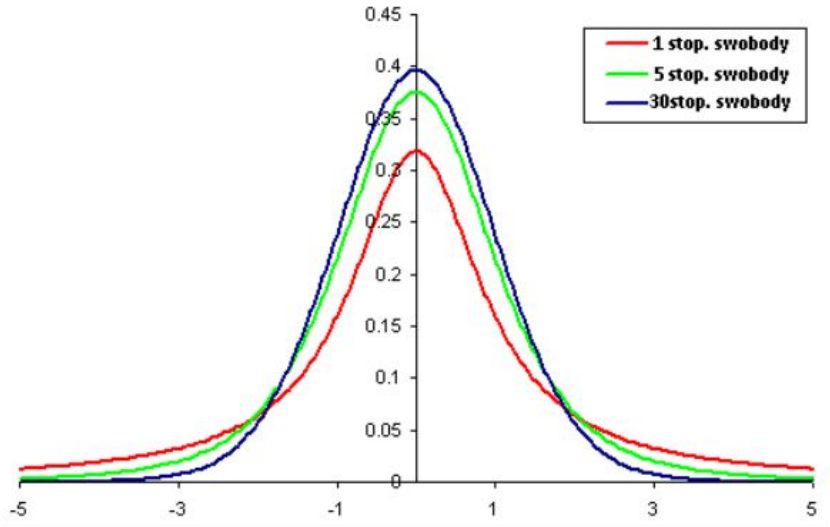
Mamy tu dwa sposoby rysowania funkcji gęstości:

- z rozkładu teoretycznego

- z rozkładu o zadanych parametrach wejściowych (średnia i wariancja)

Rozkład t-Studenta

Rozkład t studenta **stosujemy tylko** w sytuacji gdy odchylenie standardowe populacji jest nieznane, a rozmiar próby(ilość obserwacji) jest mniejsza niż 30. W przypadku gdy rozmiar próby jest większy lub równy 30 wtedy zamiast brać rozkład t bierzemy rozkład normalny. Wynika to z faktu, że rozkład t studenta dla $n \geq 30$ jest bardzo podobny do rozkładu normalnego. Dla n < 30 rozkład studenta jest „szerszy", tzn. bardziej prawdopodobne są wartości mocno odbiegające od średniej niż w przypadku rozkładu normalnego.



Przedstawienie rozkładu t studenta dla 1, 5 i 30 stopni swobody. Czerwona linia jest dużo bardziej rozległa niż niebieska.

Testowanie rozkładu t-Studenta

Test t-studenta porównuje średnią z populacji z jakąś założoną przez nas średnią.

Jest to test parametryczny.

Założenia:

- Dobrze jakby rozkład badanej zmiennej był normalny.
- Dobrze jakby grupy były równoliczne. Niektórzy przyjmują, że nawet jak jedna grupa jest do dwóch razy większa niż druga, to można stosować test t-studenta.
- Dobrze jakby wariancja była taka sama.
- Często okazuje się, że test t-studenta jest odporny na złamanie powyższych założeń.

Test t-studenta dla 1 średniej

Najprostszym testem do weryfikacji hipotezy mówiącej o równości średniej i pewnej stałej C jest test T studenta. Do wykonania testu weryfikującego hipotezę służy funkcja ttest_1samp.

Funkcja zwraca dwie wartości: wartość statystyki t oraz wartość prawdopodobieństwa p określającego poziom, od jakiego nie ma podstaw do odrzucenia hipotezy zerowej. Przy wysokiej wartości prawdopodobieństwa p, hipoteza zerowa nie jest odrzucana. Na ogół przyjmuje się poziom graniczny na p = 0.05 lub p = 0.01.

Test t-Studenta dla 2 średnich – próby niezależne

Bardzo podobny test do poprzedniego (również T studenta) weryfikuje hipotezę o równości średnich dwóch populacji.
Test ten przyjmuje dwa zbiory danych jako zbiory wejściowe i weryfikuje hipotezę o równości średnich.
W scipy.stats służy do tego funkcja ttest_ind. Jako przykład wykorzystania może posłużyć poniższy kod:

```
import scipy.stats as sps

data = np.loadtxt("Wzrost.csv", delimiter=',', skiprows=0, unpack=True) data1 =
np.loadtxt(„example.csv", delimiter=',', skiprows=0, unpack=True)
results = sps.ttest_ind(data, data1)
```

Podobnie jak w poprzednim przypadku, funkcja zwraca dwie wartości: wartość statystyki t oraz wartość prawdopodobieństwa p określającego poziom od jakiego nie ma podstaw do odrzucenia hipotezy zerowej.
Przy wysokiej wartości prawdopodobieństwa p hipoteza zerowa nie jest odrzucana.
Na ogół przyjmuje się poziom graniczny na p = 0.05 lub p = 0.01.

Test t-Studenta dla 2 średnich – próby zależne

Kolejnym przypadkiem, jaki należy rozpatrzeć przy testowaniu hipotez statystycznych jest sytuacja, w której porównujemy dwa parametry struktury dla prób, które powstały w sposób zależny. Dobrym przykładem jest np. badanie temperatury pacjentów przed i po zażyciu leku przeciwgorączkowego. Najczęściej zmienne zależne dotyczą tej samej próbki przed i po zaistnieniu jakiegoś zjawiska. Do przeprowadzenia testu dla zmiennych zależnych służy funkcja ttest_rel i przyjmuje jako parametry dwa zbiory danych (zbiory powinny być równoliczne). Test ten można zobrazować za pomocą następującego kodu:

```
import scipy.stats as sps
import pandas

brain_data = pandas.read_csv('brain_size.csv', sep=';', na_values='.')
results = sps.ttest_rel(brain_data['FSIQ'],brain_data['PIQ'])
```

lub ostatnia linia równoważnie:

```
results = sps.ttest_1samp(brain_data['FSIQ'] - brain_data['PIQ'],0)
```

Statystyka w data science

- Użyteczne pojęcia w data science z zakresu statystyki matematycznej to:
- Średnia
- Mediana
- Odchylenie standardowe i wariancja
- Kowariancja
- Korelacja
- Wartość oczekiwana

Podstawowe pojęcia ze statystyki

- Średnia arytmetyczna – suma liczb podzielona przez ich liczbę. Dla liczb jest to więc wyrażenie. W języku potocznym średnią arytmetyczną określa się po prostu jako średnią.

- Mediana:

Aby obliczyć medianę ze zbioru n obserwacji, sortujemy je w kolejności od najmniejszej do największej i numerujemy od 1 do n. Następnie, jeśli n jest nieparzyste, medianą jest wartość obserwacji w środku (czyli obserwacji numer $\frac{n+1}{2}$). Jeśli natomiast n jest parzyste, wynikiem jest średnia arytmetyczna między dwiema środkowymi obserwacjami, czyli obserwacją numer $\frac{n}{2}$ i obserwacją numer $\frac{n}{2}+1$.

Podstawowe pojęcia z zakresu statystyki

Wariancja – miara rozproszenia wyników wokół średniej, możliwa do obliczenia tylko dla zmiennych o ilościowym poziomie pomiaru.

Wariancja jest wyliczana poprzez iloraz zsumowanych kwadratów odchyleń wyników od średniej, przez liczbę wyników pomniejszoną o 1.

Wariancja przybiera wartość od 0 do + nieskończoności. Przy zerowej wartości w danym zbiorze wyników nie ma żadnego zróżnicowania (wszystkie wyniki badanych są takie same). Z kolei wraz ze wzrostem wartości wariancji, zróżnicowanie wyników rośnie.

Wariancja jest kluczowym pojęciem dla testów porównujących wartości średnich (np. test t studenta dla prób niezależnych, analiza wariancji), w których jednym z założeń jest założenie o jednorodności wariancji. Ponadto, na podstawie wariancji i średniej, szacowane są wyniki na poziomie populacji.

Odchylenie standardowe to pierwiastek z wariancji, również mierzy zróżnicowanie zbioru.

Podstawowe pojęcia z zakresu statystyki

Wartość oczekiwana - zwana również wartością średnią, wartością przeciętną, nadzieją matematyczną - nazywamy spodziewany wynik doświadczenia losowego przy założonym prawdopodobieństwie jego wystąpienia.

Dla lepszego opisu czym jest, i jak rozumieć pojęcie wartości oczekiwanej, należy podzielić temat na dwa oddzielne zagadnienia: wartość oczekiwana w terminologii rachunku prawdopodobieństwa oraz wartość oczekiwana rozkładu danej cechy w populacji. Takie rozdzielenie ułatwi nam zrozumienie pojęcia, pomimo iż to dalej jest to samo pojęcie.

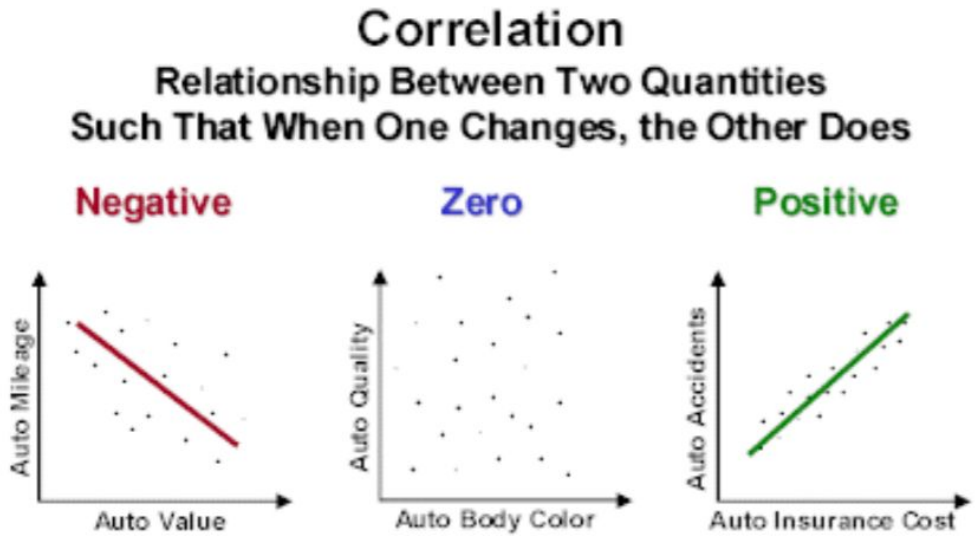
Podstawowe pojęcia z zakresu statystyki

- Wartość oczekiwana w prawdopodobieństwie oznacza spodziewany wynik danego doświadczenia losowego.

Przykład: Rzucamy 100 razy symetryczną monetą. Prawdopodobieństwo wyrzucenia orła w jednym rzucie wynosi 0,5; tak samo reszki. W każdym ze 100 rzutów jest takie samo prawdopodobieństwo wyrzucenia orła bądź reszki. Zatem w 100 rzutach oczekujemy, że wylosujemy 50 reszek i 50 orłów. Jeżeli zadamy pytanie ile wynosi wartość oczekiwana dla liczby reszek w 100-krotnym rzucie monetą, odpowiedź brzmi: Wartość oczekiwana wynosi 50.

Podstawowe pojęcia z zakresu statystyki

Zależność korelacyjna pomiędzy cechami X i Y charakteryzuje sie tym, że wartościom jednej cechy są przyporządkowane ściśle określone wartości średnie drugiej cechy.



Podstawowe pojęcia z zakresu statystyki

Kowariancja jest to wielkość charakteryzująca wspólne zmiany dwóch zmiennych X i Y. Jest oczekiwana wartością iloczynu odchyleń wartości zmiennych X i Y od ich wartości oczekiwanych.
Zakładając, że X i Y to para zmiennych losowych o rozkładach normalnych i średnich $\mu_x|$ i $\mu_y|$ oraz standardowych odchyleniach $\sigma_x|$ i $\sigma_y|$. Kowariancję dwóch zmiennych X i Y liczymy ze wzoru

$$cov(X,Y) = E[X - E(X)][Y - E(Y)]|$$

co można też przedstawić w postaci

$$cov(X,Y) = E(XY) - E(X)E(Y)|$$

Badanie normalności i równości wariancji

Aby można było zastosować testy parametryczne przy weryfikacji hipotez, dane wejściowe muszą spełniać założenia o normalności rozkładu danych.
W Pythonie dostępne są następujące testy oceniające normalność rozkładu:

a) normaltest – wykorzystuje kurtozę i skośność do sprawdzenia normalności rozkładu (jeśli rozkład jest mało skośny i podobnie spłaszczony do normalnego, to można założyć jego normalność)

Przykład zastosowania w Pythonie:

```
import scipy.stats as sps import pandas
brain_data = pandas.read_csv('brain_size.csv', sep=';', na_values='.')
test_results = sps.normaltest(brain_data['PIQ'])
print(test_results)
```

Test Kołomogorowa-Smirnowa

Testuje, czy rozkład zmiennej jest zbliżony do zadanego rozkładu teoretycznego (w przypadku testowania normalności – do rozkładu normalnego).

Przykład w Pythonie:

```
import scipy.stats as sps

normal_data = sps.norm.rvs(size=1000) ks_results =
sps.kstest(normal_data,'norm’)
print(ks_results)
```

Test Shapiro-Wilka

Test ten uważany jest za mocniejszy niż test Kołomogorowa – Smirnowa.

Przykład w Pythonie:

```
import scipy.stats as sps

normal_data = sps.norm.rvs(size=1000)
shapiro_test_results = sps.shapiro(normal_data)
 print(shapiro_test_results)
```

Zadanie 10

Korzystając z testu Shapiro-Wilka sprawdź, czy podana próba ma rozkład normalny: [0.873, 2.817, 0.121, -0.945, -0.055, -1.436, 0.360, -1.478, -1.637, -1.869].

Testy równości wariancji

Drugim kluczowym warunkiem do możliwości zastosowania testów parametrycznych jest założenie o równości wariancji w próbach, które testujemy. Do zbadania równości wariancji w Pythonie możemy wykorzystać następujące testy:

a) Test Levene – przykład w Pythonie:

```
import scipy.stats as sps

normal_data = sps.norm.rvs(size=1000)
normal_data1 = sps.norm.rvs(size=1000)
normal_data2 = sps.norm.rvs(loc=1, scale=20, size=1000)
levene_test_results = sps.levene(normal_data, normal_data1,
normal_data2)
print(levene_test_results)
```

b) Test Bartlett - przykład w Pythonie:

```
import scipy.stats as sps

normal_data = sps.norm.rvs(size=1000)
normal_data1 = sps.norm.rvs(size=1000)
normal_data2 = sps.norm.rvs(loc=1, scale=20, size=1000)
bartlett_test_results = sps.bartlett(normal_data, normal_data1,
normal_data2)
print(bartlett_test_results)
```

Dodatkowe materiały dla chętnych

- Blog o analizie i wizualizacji danych
http://szychtawdanych.pl/
- About Data - blog o uczeniu maszynowym
https://ksopyla.com
- Tutorial 10 minutes to pandas
http://pandas.pydata.org/pandas-docs/stable/10min.html
- Pandas cookbook
http://pandas.pydata.org/pandas-docs/stable/cookbook.html
- Biblioteka numpy
https://numpy.org/doc/stable/user/basics.creation.html