

Machine Learning

ML Python

ML tutorial python

Python File Handling
Python Read Files
Python Write/Create Files
Python Delete Files

Python Modules
NumPy Tutorial
Pandas Tutorial
SciPy Tutorial
Django Tutorial

Python Matplotlib
Matplotlib Intro
Matplotlib Get Started
Matplotlib Pyplot
Matplotlib Plotting
Matplotlib Markers
Matplotlib Line
Matplotlib Labels
Matplotlib Grid
Matplotlib Subplot
Matplotlib Scatter
Matplotlib Bars
Matplotlib Histograms
Matplotlib Pie Charts

Machine Learning
Getting Started
Mean Median Mode
Standard Deviation
Percentile

Machine Learning

[← Previous](#)

Machine Learning is making the computer learn from studying data and statistics.

Machine Learning is a step into the direction of artificial intelligence (AI).

Machine Learning is a program that analyses data and learns to predict the outcome.

Where To Start?

In this tutorial we will go back to mathematics and study statistics, and how to calculate important numbers based on data sets.

We will also learn how to use various Python modules to get the answers we need.

And we will learn how to make functions that are able to predict the outcome based on what we have learned.

Data Set

In the mind of a computer, a data set is any collection of data. It can be anything from an array to a complete database.

Example of an array:

```
[99, 86, 87, 88, 111, 86, 103, 87, 94, 78, 77, 85, 86]
```

Visual Studio Code

wygodny edytor kodu

The image shows a screenshot of the Visual Studio Code website and a code editor interface. The website header includes the Visual Studio Code logo, navigation links (Docs, Updates, Blog, API, Extensions, FAQ, Learn), a search bar, and a 'Download' button. A banner for 'Version 1.76' is visible. The main content area features the text 'Code editing. Redefined.' and 'Free. Built on open source. Runs everywhere.' A red circle highlights the 'Download for Windows' button, which is labeled 'Stable Build'. Below this, there are links for 'Web, Insiders edition, or other platforms' and a note about the license and privacy statement. The code editor interface shows a file explorer on the left with a list of extensions (Python, GitLens, C/C++, ESLint, Debugger for Chrome, Language Support for Java, vscode-icons, Vetur, C#). The main editor area displays JavaScript code for a service worker, and the terminal at the bottom shows the command '1: node' and the output 'You can now view create-react-app in the browser.' with local and network URLs.

Visual Studio Code Docs Updates Blog API Extensions FAQ Learn

Search Docs Download

Version 1.76 is now available! Read about the new features and fixes from February.

Code editing.
Redefined.

Free. Built on open source. Runs everywhere.

Download for Windows
Stable Build

Web, Insiders edition, or other platforms

By using VS Code, you agree to its
license and privacy statement.

File Edit Selection View Go Debug Terminal Help serviceWorker.js - create-react-app - Visual Studio Code - In...

EXTENSIONS: MARKETPLACE @sortinstalls

- Python 2019.6.24221 54.9M ★ 4.5
Linting, Debugging (multi-threaded, ...
Microsoft Install
- GitLens — Git sup... 9.8.5 23.1M ★ 5
Supercharge the Git capabilities buil...
Eric Amodio Install
- C/C++ 0.24.0 23M ★ 3.5
C/C++ IntelliSense, debugging, and ...
Microsoft Install
- ESLint 1.9.0 21.9M ★ 4.5
Integrates ESLint JavaScript into VS ...
Dirk Baeumer Install
- Debugger for Ch... 4.11.6 20.6M ★ 4
Debug your JavaScript code in the C...
Microsoft Install
- Language Supp... 0.47.0 18.7M ★ 4.5
Java Linting, IntelliSense, formatting, ...
Red Hat Install
- vscode-icons 8.8.0 17.2M ★ 5
Icons for Visual Studio Code
VSCode Icons Team Install
- Vetur 0.21.1 17M ★ 4.5
Vue tooling for VS Code
Pine Wu Install
- C# 1.21.0 15.6M ★ 4
C# for Visual Studio Code (powered ...
Microsoft Install

```
src > JS serviceWorker.js > register > window.addEventListener("load") callback
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
function registerValidSW(swUrl, config) {
  navigator.serviceWorker
    .register(swUrl)
    .then(registration => {
      checkValidServiceWorker(swUrl, config);
      // Add some additional logging to localhost, p...
      // service worker/PWA documentation.
      navigator.serviceWorker.ready.then(() => {
        product
        productSub
        removeSiteSpecificTrackingException
        removeWebWideTrackingException
        requestMediaKeySystemAccess
        sendBeacon
        serviceWorker (property) Navigator.serviceWorke...
        storage
        storeSiteSpecificTrackingException
        storeWebWideTrackingException
        userAgent
        vendor
      })
    })
}
```

TERMINAL 1: node

You can now view create-react-app in the browser.

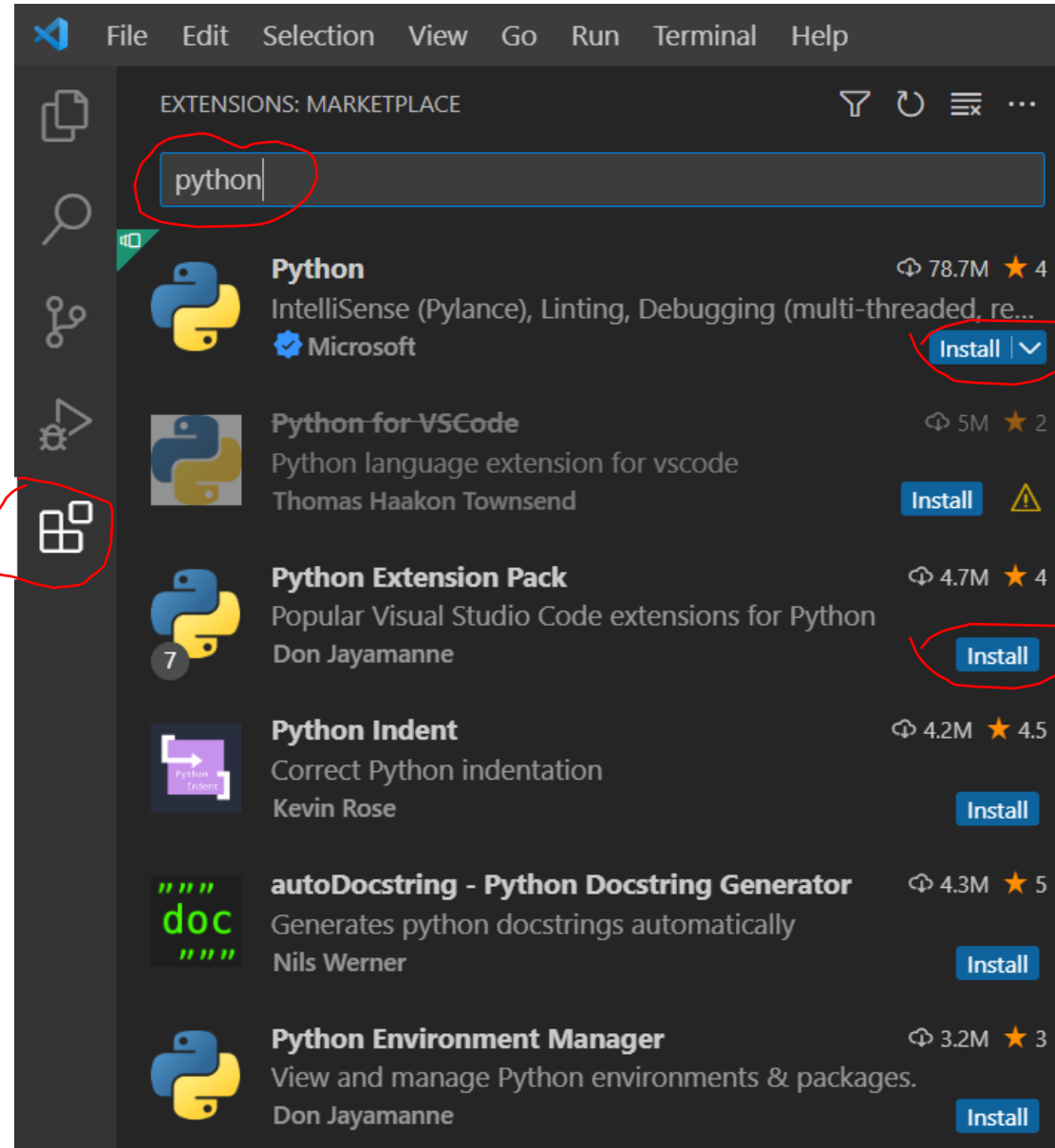
Local: http://localhost:3000/
On Your Network: http://10.211.55.3:3000/

Note that the development build is not optimized.

Ln 43, Col 19 Spaces: 2 UTF-8 LF JavaScript

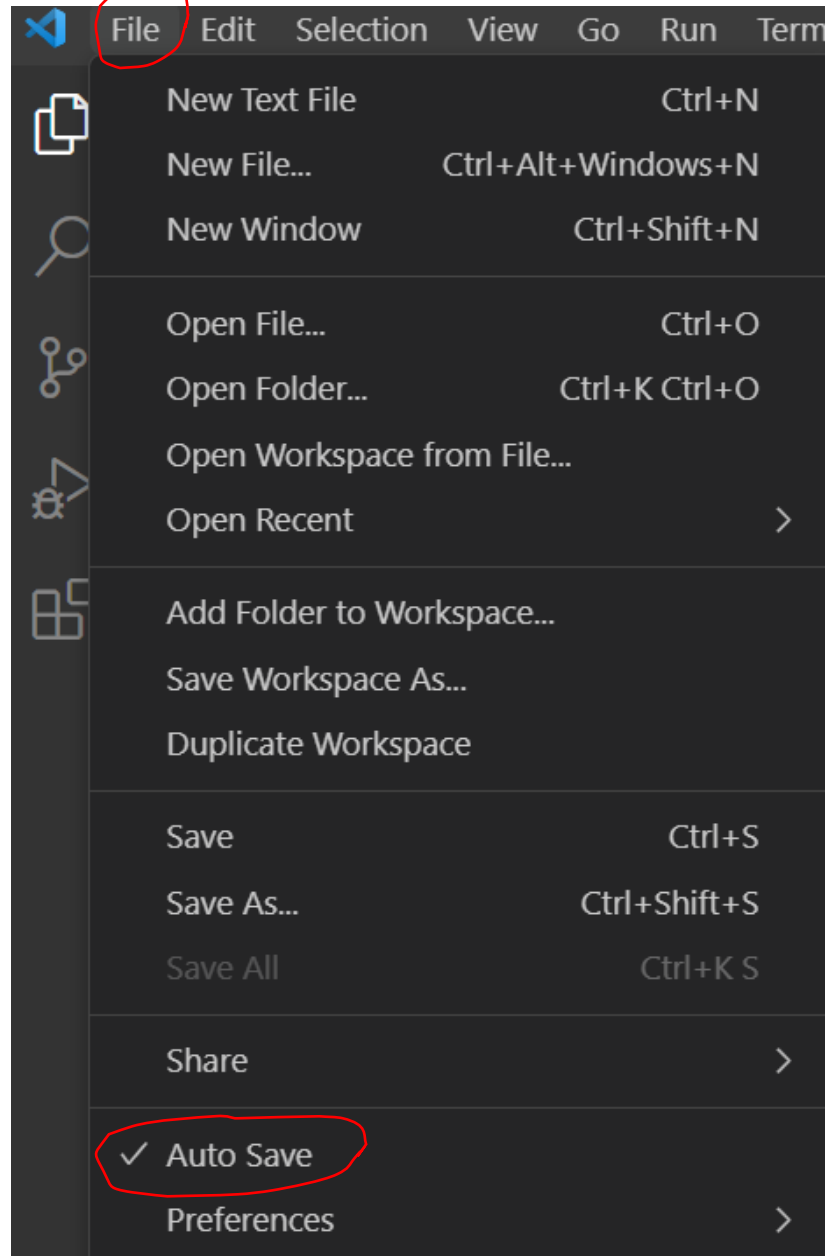
Visual Studio Code

instalacja Pythona



VSC Auto Save

włączenie auto zapisu
w Visual Studio Code



Typy danych



znając typ danych można zastosować odpowiednią technikę do ich analizy

biblioteka NumPy

The fundamental package for scientific computing with Python

biblioteka Pythona do wykonywania szybkich operacji na tablicach (Python nie ma zaimplementowanych tablic)

<https://numpy.org>

[Install](#) [Documentation](#) [Learn](#) [Community](#) [About Us](#) [News](#) [Contribute](#)



The fundamental package for scientific computing with Python

LATEST RELEASE: NUMPY
1.24.2. [VIEW ALL RELEASES.](#)

Meet the new NumPy docs team leads

POWERFUL N-DIMENSIONAL ARRAYS

Fast and versatile, the NumPy vectorization, indexing, and broadcasting concepts are the de-facto standards of array computing today.

NUMERICAL COMPUTING TOOLS

NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.

OPEN SOURCE

Distributed under a liberal [BSD license](#), NumPy is developed and maintained [publicly on GitHub](#) by a vibrant, responsive, and diverse [community](#).

INTEROPERABLE

NumPy supports a wide range of hardware and computing platforms, and plays well with distributed, GPU, and sparse array libraries.

PERFORMANT

The core of NumPy is well-optimized C code. Enjoy the flexibility of Python with the speed of compiled code.

EASY TO USE

NumPy's high level syntax makes it accessible and productive for programmers from any background or experience level.

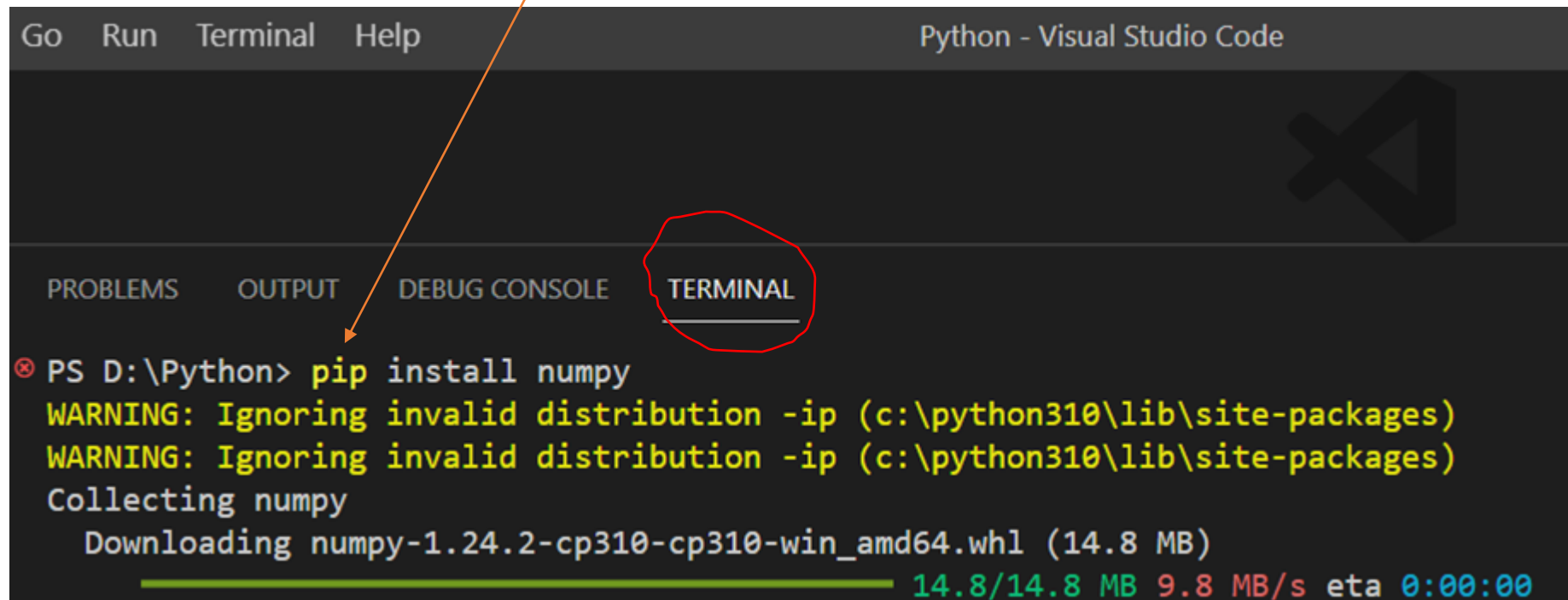
<https://numpy.org/>

<https://analytk.edu.pl/python-numpy/>

biblioteka NumPy

instalacja biblioteki NumPy w VSC

```
pip install numpy
```



```
Go Run Terminal Help Python - Visual Studio Code
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
⊗ PS D:\Python> pip install numpy
WARNING: Ignoring invalid distribution -ip (c:\python310\lib\site-packages)
WARNING: Ignoring invalid distribution -ip (c:\python310\lib\site-packages)
Collecting numpy
  Downloading numpy-1.24.2-cp310-cp310-win_amd64.whl (14.8 MB)
  14.8/14.8 MB 9.8 MB/s eta 0:00:00
```

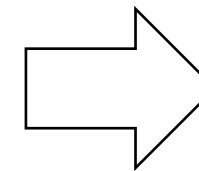

Mean (średnia arytmetyczna)

suma liczb podzielona przez ich liczbę

tablica z ocenami

```
mean.py ×
mean.py > ...
1 import numpy
2
3 oceny = [1,5,3,3,4,6,2,5]
4
5 # wyliczenie średniej arytmetycznej 1 sposób
6 sumaOcen = 1 + 5 + 3 + 3 + 4 + 6 + 2 + 5
7 iloscOcen = 8
8 srednia = sumaOcen/iloscOcen
9
10 # wyliczenie średniej arytmetycznej 2 sposób
11 mean = numpy.mean(oceny)
12
13 print("srednia: ")
14 print(srednia)
15 print("mean ")
16 print(mean)
```

średnia ocen



```
• srednia:
  3.625
  mean
  3.625
```

wyliczenie średniej z tablicy ocen za pomocą biblioteki NumPy

Mean (średnia arytmetyczna)

Zadanie

Napisz program `mean.py` wyliczający średnią arytmetyczną wymyślonego przez Ciebie zbioru liczb

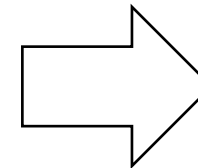
Median (mediana)

wartość środkowa posortowanego zbioru liczb

wyliczenie mediany z tablicy ocen
za pomocą biblioteki NumPy

```
median.py ×  
median.py > ...  
1 import numpy  
2  
3 oceny = [1,5,4,3,6,2,5]  
4  
5 # oceny posortowane:  
6 # 1 2 3 4 5 5 6  
7  
8 median = numpy.median(oceny)  
9  
10 print("median: ")  
11 print(median)
```

liczba ocen jest nieparzysta –
można znaleźć wartość środkową
posortowanego zbioru



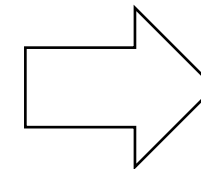
```
median:  
4.0
```

Median (mediana)

```
median1.py X
median1.py > ...
1  import numpy
2
3  oceny = [1,5,4,3,1,6,2,5]
4
5  # oceny posortowane:
6  # 1 1 2 3 4 5 5 6
7
8  median = numpy.median(oceny)
9
10 print("median: ")
11 print(median)
```

liczba ocen jest parzysta – nie ma jednej wartości środkowej posortowanego zbioru

w tym przypadku medianą jest średnia arytmetyczna dwóch środkowych liczb:
 $(2+4)/2=3.5$



```
median:
3.5
```

Median (mediana)

Zadanie

Napisz program `median.py` wyliczający medianę (2 przypadki) wymyślonego przez Ciebie zbioru liczb

biblioteka SciPy

Fundamental algorithms for scientific computing in Python

SciPy to biblioteka służąca do obliczeń, która opiera się na bibliotece NumPy

https://scipy.org

[Install](#) [Documentation](#) [Community](#) [About Us](#) [Contribute](#)



Fundamental algorithms for scientific computing in Python

GET STARTED

SciPy 1.10.1 released! 2023-02-19

FUNDAMENTAL ALGORITHMS

SciPy provides algorithms for optimization, integration, interpolation, eigenvalue problems, algebraic equations, differential equations, statistics and many other classes of problems.

BROADLY APPLICABLE

The algorithms and data structures provided by SciPy are broadly applicable across domains.

FOUNDATIONAL

Extends NumPy providing additional tools for array computing and provides specialized data structures, such as sparse matrices and k-dimensional trees.

PERFORMANT

SciPy wraps highly-optimized implementations written in low-level languages like Fortran, C, and C++. Enjoy the flexibility of Python with the speed of compiled code.

EASY TO USE

SciPy's high level syntax makes it accessible and productive for programmers from any background or experience level.

OPEN SOURCE

Distributed under a liberal [BSD license](#), SciPy is developed and maintained [publicly on GitHub](#) by a vibrant, responsive, and diverse [community](#).

<https://scipy.org/>

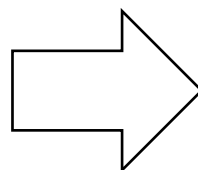
https://www.w3schools.com/python/scipy/scipy_intro.php

Mode (moda)

najczęściej występująca wartość liczbowa

```
mode.py ×  
mode.py > ...  
1 from scipy import stats  
2  
3 oceny = [1,5,4,3,1,6,2,5,5,5,6,1,3,2,2]  
4  
5 mode = stats.mode(oceny)  
6  
7 print("mode: ")  
8 print(mode)
```

wyliczenie najczęściej występującej wartości liczbowej w zbiorze za pomocą biblioteki SciPy



```
mode:  
ModeResult(mode=array([5]), count=array([4]))
```

najczęściej występująca ocena to 5

5 występuje 4 razy

Mode (moda)

Zadanie

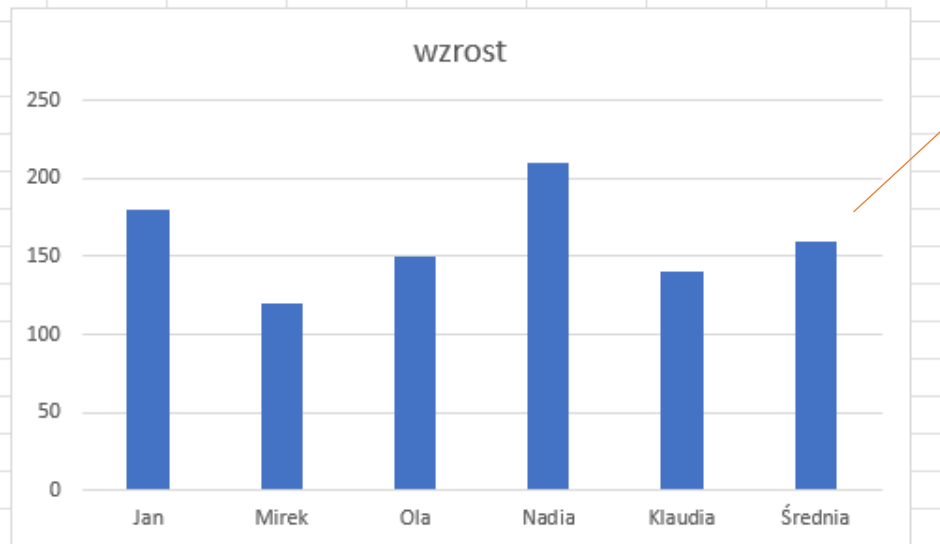
Napisz program `mode.py` wyliczający modę wymyślonego przez Ciebie zbioru liczb

Standard Deviation (Odchylenie standardowe)

Odchylenie standardowe opisuje rozłożenie danej wielkości (np. wzrostu uczniów) wokół średniej

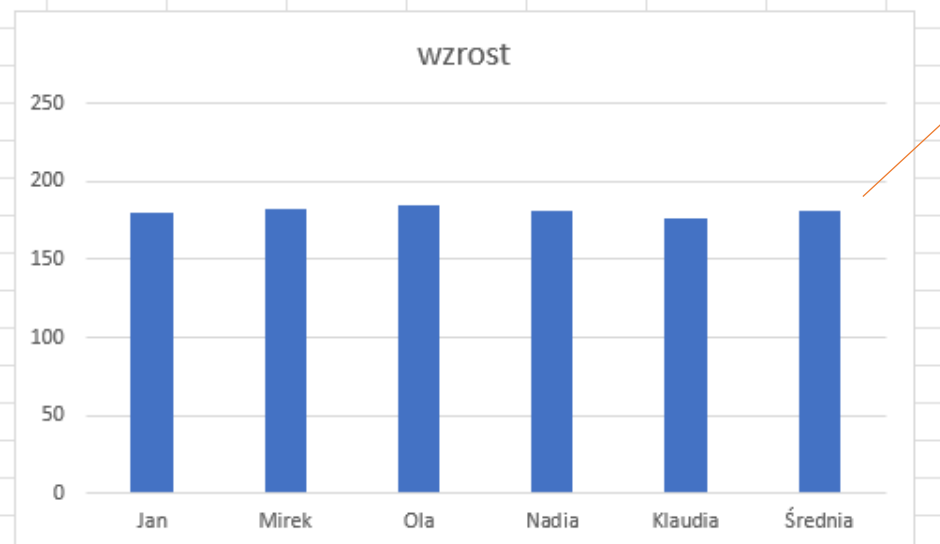
duże odchylenie standardowe

	Jan	Mirek	Ola	Nadia	Klaudia	Średnia
wzrost	180	120	150	210	140	160



małe odchylenie standardowe

	Jan	Mirek	Ola	Nadia	Klaudia	Średnia
wzrost	180	182	185	181	177	181



Standard Deviation (Odchylenie standardowe)

Założmy że mamy dane liczby x_1, x_2, \dots, x_n oraz że ich średnia arytmetyczna wynosi \bar{X}

Wówczas **odchylenie standardowe** tych liczb od ich średniej arytmetycznej, to pierwiastek kwadratowy z wariancji, czyli:

$$\sigma = \sqrt{\frac{(x_1 - \bar{X})^2 + (x_2 - \bar{X})^2 + \dots + (x_n - \bar{X})^2}{n}}$$

The diagram includes several annotations: an orange line points from the Greek letter sigma (σ) to the word "sigma" in orange text; another orange line points from the entire square root expression to the text "odchylenie standardowe jest pierwiastkiem z wariancji" in orange text; and a blue line points from the fraction inside the square root to the word "wariancja" in blue text.

odchylenie standardowe jest pierwiastkiem z wariancji

<https://www.matemaks.pl/odchylenie-standardowe.html>

Standard Deviation (Odchylenie standardowe)

liczymy odchylenie standardowe „ręcznie” ze wzoru (pierwiastek z wariancji)

	Jan	Mirek	Ola	Nadia	Klaudia	Średnia
wzrost	180	120	150	210	140	160

większość wartości mieści się w przedziale ($\text{średnia} - \sigma$, $\text{średnia} + \sigma$),
czyli około (128.4 , 191.6)

wariancja:

$$(180-160)^2 + (120-160)^2 + (150-160)^2 + (210-160)^2 + (140-160)^2 = 5000$$

$$5000/5 = 1000$$

odchylenie standardowe wzrostu uczniów od średniej arytmetycznej ich wzrostu:

$$\sigma = \sqrt{1000} = 31,62277660168379$$

Standard Deviation (Odchylenie standardowe)

liczymy odchylenie standardowe „ręcznie” ze wzoru (pierwiastek z wariancji)

	Jan	Mirek	Ola	Nadia	Klaudia	Średnia
wzrost	180	182	185	181	177	181

większość wartości mieści się w przedziale (średnia - σ , średnia + σ), czyli około (178.4 , 183.6)

wariancja:

$$(180-181)^2 + (182-181)^2 + (185-181)^2 + (181-181)^2 + (177-181)^2 = 34$$

$$34/5 = 6,8$$

odchylenie standardowe wzrostu uczniów od średniej arytmetycznej ich wzrostu:

$$\sigma = \sqrt{6,8} = 2,607680962081059$$

Standard Deviation (Odchylenie standardowe)

wyniki działania programu
zgadzą się z obliczeniami

```
standardDeviation.py > ...
1  # Odchylenie standardowe
2
3  import numpy
4
5  wzrost1 = [180, 120, 150, 210, 140]
6
7  sigma1 = numpy.std(wzrost1)
8
9  print(sigma1) # 31.622776601683793
10
11
12
13  wzrost2 = [180, 182, 185, 181, 177]
14
15  sigma2 = numpy.std(wzrost2)
16
17  print(sigma2) # 2.6076809620810595
```

duże odchylenie standardowe

małe odchylenie standardowe

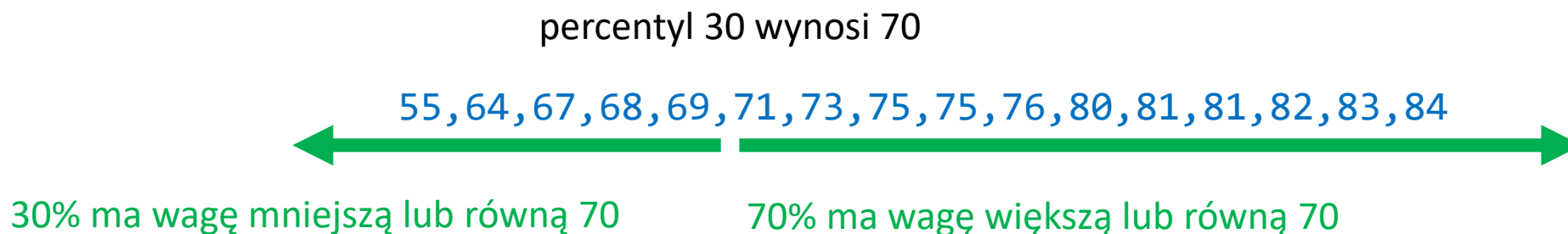
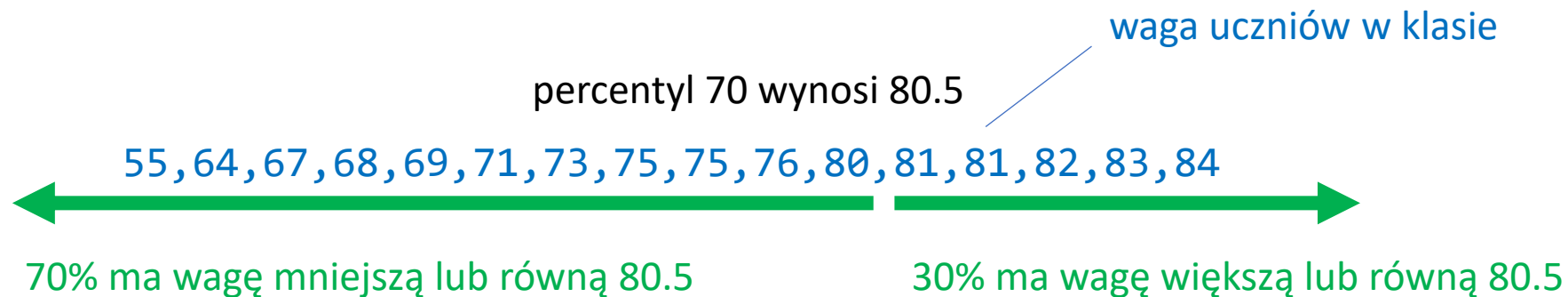
Standard Deviation (Odchylenie standardowe)

Zadanie

Napisz program `standardDeviation.py` wyliczający odchylenie standardowe wymyślonego przez Ciebie zbioru liczb za pomocą biblioteki `numpy` oraz „ręcznie” ze wzoru (pierwiastek z wariacji) i porównaj oba wyniki.

Percentile (percentyl, centyl)

jednostka statystyczną opisującą położenie danego wyniku
względem całej grupy wyników



Percentile (percentyl, centyl)

```
percentile.py > ...
1  # percentyl
2
3  import numpy
4
5  # waga uczniów w klasie
6  wagaUcznow = [55,64,67,68,69,71,73,75,75,76,80,81,81,82,83,84]
7
8
9  x = numpy.percentile(wagaUcznow, 70)
10
11 print(x) # 80,5
12
13 # percentyl 70 wynosi 80.5, czyli 70% uczniów w klasie waży 80.5 kg lub mniej,
14 # a 30% waży 80.5 lub więcej
15
16 x1 = numpy.percentile(wagaUcznow, 30)
17
18 print(x1) # 70
19
20 # percentyl 30 wynosi 70, czyli 30% uczniów w klasie waży 70 kg lub mniej,
21 # a 70% waży 70 kg lub więcej
```

Percentile (percentyl, centyl)

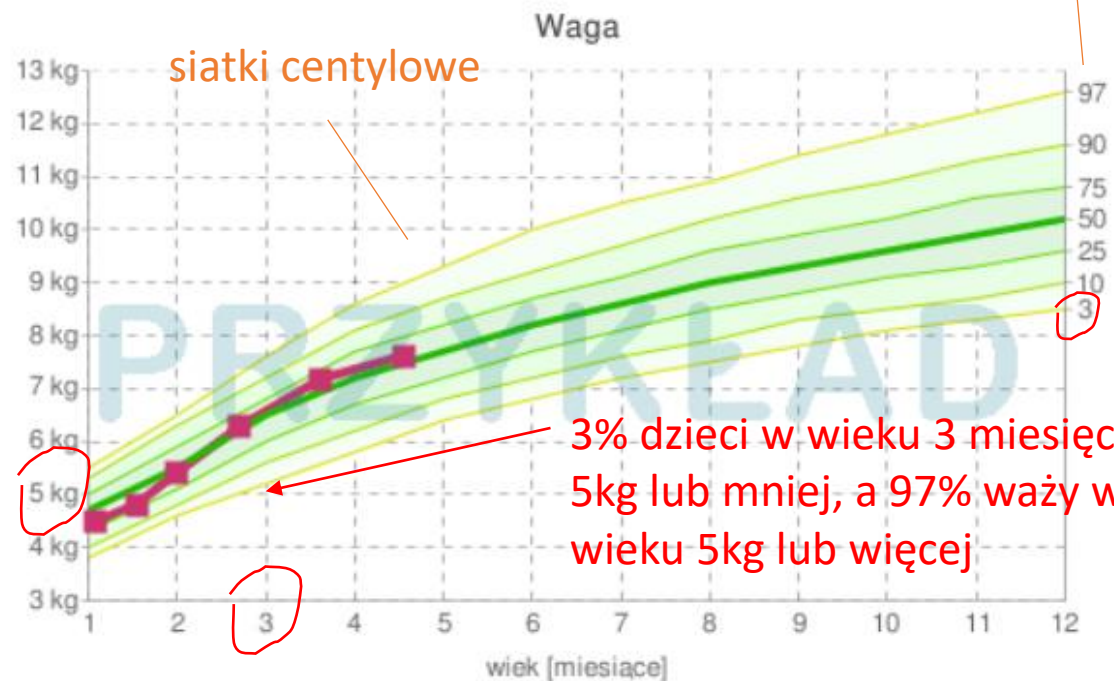


SiatkiCentylowe.pl
Sprawdź czy Twoje dziecko rośnie prawidłowo!



GAZETA.PL eDZIECKO

Stwórz wykres rozwoju swojego dziecka



Imię dziecka

Chłopiec

Dziewczynka

Data urodzenia

Poród

Aktualna waga

kg lub g

Aktualny wzrost

m lub cm

Sprawdź

Percentile (percentyl, centyl)

Zadanie

Napisz program `percentile.py`
wyliczający percentyle (np. 30, 70, 90)
wymyślonego przez Ciebie zbioru liczb.

Data Distribution

generowanie
większych
zestawów
danych o
losowych
wartościach z
zadanego
przedziału

dataDistribution.py > ...

```
1 import numpy
2
3 # generujemy tablicę z 100 liczbami zmiennoprzecinkowymi
4 # od 0.0 do 10.0
5 x = numpy.random.uniform(0.0, 10.0, 100)
6
7 print(x)
8
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

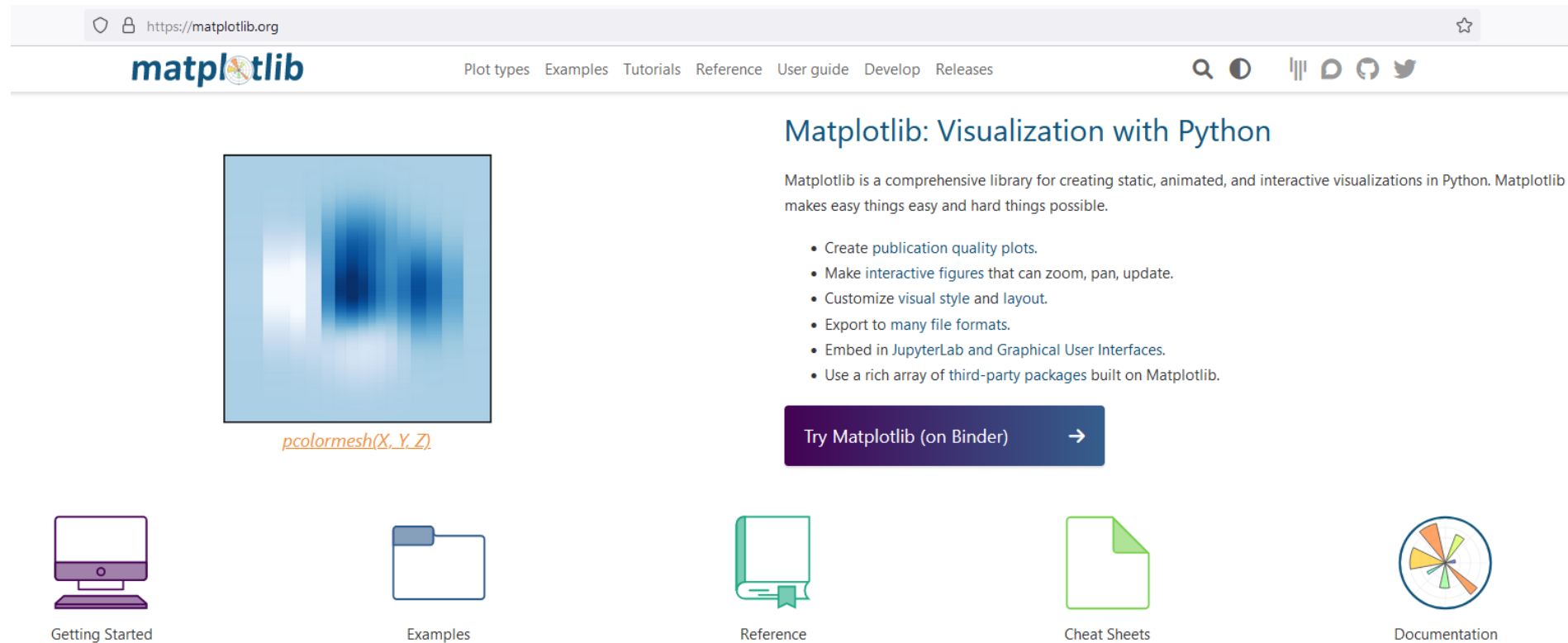
```
[7.674779  7.69883332 0.92209627 9.83660979 3.08315446 0.60584779
6.09662585 8.36481537 5.56000548 2.62851999 3.37267935 4.76898585
1.49754479 1.90010583 4.73398192 4.82357879 6.42636956 2.57747062
5.5997542  9.75040714 0.63285461 5.69627121 7.82804193 4.2911846
8.64308559 1.91042692 3.74896261 1.58336151 6.73568663 6.73575566
2.73016232 0.57961486 6.27834905 1.11411084 3.17106164 2.57413344
5.97488182 1.49737393 3.50607135 3.89399608 7.408972  5.14622145
9.43683799 3.87058616 9.86684954 7.81324592 8.30541873 6.63463676
8.46751301 3.07323543 0.06238049 8.76128193 2.21374294 1.71140466
5.39412918 3.2636824  7.66549917 3.27410594 8.94793225 8.09374154
9.39442246 9.24498564 2.94694031 2.764871  0.82347865 9.85358219
9.67886126 3.69106715 8.28428369 6.72515776 3.57139894 2.13184733
9.57992567 2.07685891 5.44043542 4.32569312 6.49344519 2.29765696
7.82494696 7.68550252 2.81913035 8.93955411 0.63003125 7.06999215
3.3745049  8.83460066 5.74506609 5.06203303 1.29873823 0.96009959
4.42899987 4.2416471  3.15878843 6.84989385 0.30176548 4.00414833
9.78681706 3.60114878 0.83930493 6.51612493]
```

Matplotlib

biblioteka do
wizualizacji danych

instalacja biblioteki do
wizualizacji danych
(polecenie konsoli VSC)

```
pip install matplotlib
```



The screenshot shows the Matplotlib website homepage. At the top, there is a navigation bar with the Matplotlib logo and links for Plot types, Examples, Tutorials, Reference, User guide, Develop, and Releases. Below the navigation bar, the main heading reads "Matplotlib: Visualization with Python". To the left of the heading is a large image of a heatmap visualization, with the code `pcolormesh(X, Y, Z)` written below it. To the right of the heading is a list of bullet points describing the library's capabilities. Below the list is a dark blue button with the text "Try Matplotlib (on Binder)" and a right-pointing arrow. At the bottom of the page, there are five icons representing different sections: "Getting Started" (a computer monitor), "Examples" (a folder), "Reference" (an open book), "Cheat Sheets" (a document with a green tab), and "Documentation" (a circular chart).

https://matplotlib.org

matplotlib

Plot types Examples Tutorials Reference User guide Develop Releases

Matplotlib: Visualization with Python

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

- Create publication quality plots.
- Make interactive figures that can zoom, pan, update.
- Customize visual style and layout.
- Export to many file formats.
- Embed in JupyterLab and Graphical User Interfaces.
- Use a rich array of third-party packages built on Matplotlib.

Try Matplotlib (on Binder) →

Getting Started Examples Reference Cheat Sheets Documentation

<https://matplotlib.org/>
[Matplotlib Tutorial](#)

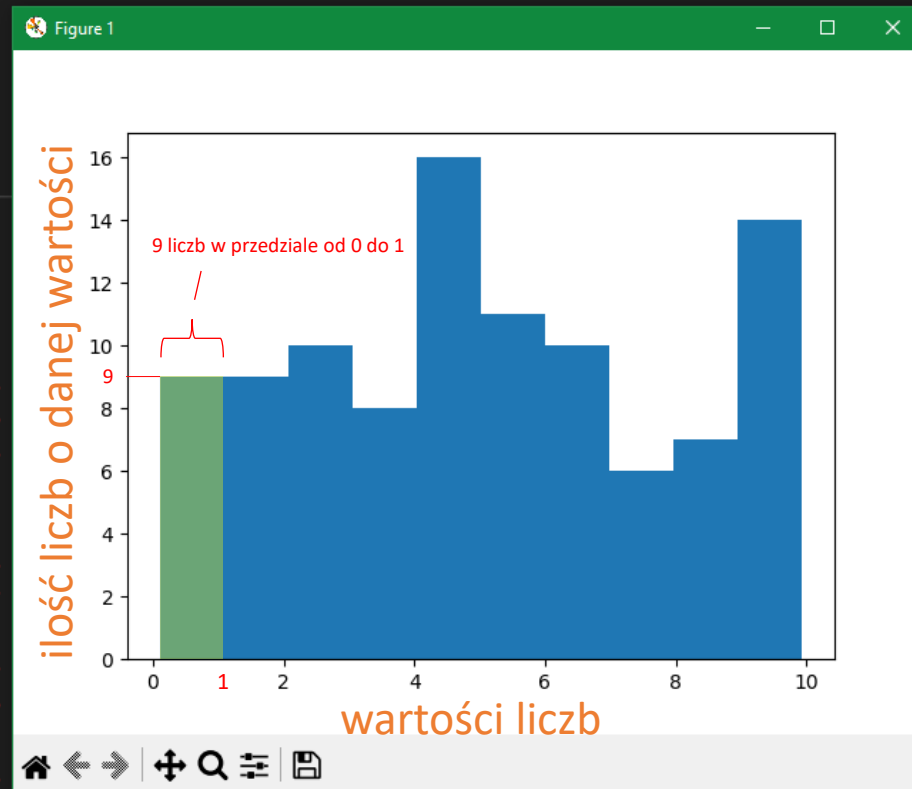
Histogram

wizualizacja danych

```
histogram.py > ...
1 import numpy
2 import matplotlib.pyplot as plt
3 # generujemy tablicę 100 liczb od 0.0 do 10.0
4 x = numpy.random.uniform(0.0, 10.0, 100)
5 # printujemy wygenerowaną tablicę w konsoli
6 print(x)
7 # rysujemy wykres, dzielimy wartości liczb w tablicy na 10 części
8 # (0-1, 1-2, 2-3, 3-4, 4-5, 5-6, 6-7, 7-8, 8-9, 9-10)
9 plt.hist(x, 10)
10 plt.show()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
[1.92179086 5.98150425 5.94373096 6.14511435 9.23256774 2.8359206
4.12616451 6.69787685 3.19985642 8.48468981 4.61581537 2.95289493
2.07073596 4.96036226 5.21528547 3.52346587 0.10252719 8.00412672
9.09351464 6.84431797 8.78220438 3.39492939 0.72205618 1.73004308
5.97051677 6.17851747 4.665998 0.58243455 6.65103211 1.92881416
3.52439853 9.47233201 4.65297173 1.13391145 7.96241203 4.86049233
2.76540945 3.86651517 5.41103277 1.4162206 9.08593993 0.66104275
6.71714142 4.69211213 2.49520636 8.9098651 2.82854516 3.61967312
2.4218803 7.69995261 9.76259368 5.46836413 9.46304091 7.60461799
4.83357485 9.02502164 6.22683394 4.43410032 0.91141602 4.7919888
4.91433794 8.02934503 5.51602024 4.86131828 8.76303273 9.85742086
0.79407558 2.08266313 1.41496389 0.75169256 5.70705268 7.51512718
9.93874135 8.6803627 1.0375709 1.33474207 3.00148903 9.81997885
3.9907255 4.63053324 1.94705201 5.93010293 9.91223935 4.46037871
9.17946869 5.79040179 6.8266409 3.63845849 7.38982254 9.85350806
6.44424543 4.34043448 0.42971314 6.30665949 5.23082335 9.70468113
1.43708385 4.72039859 2.27684719 7.0696808 ]
```



interpretacja wykresu:

Wartość liczb	Ilość liczb o danej wartości
0-1	9
1-2	9
2-3	10
3-4	8
4-5	16
5-6	11
6-7	10
7-8	6
8-9	7
9-10	14

w sumie 100 liczb

Histogram

Zadanie

Wygeneruj większy zestaw danych i przedstaw go na wykresie.

Zinterpretuj dane na wykresie (opisz wykres).

Normal Data Distribution (rozkład normalny, krzywa Gaussa)

normalDataDistribution.py > ...

```
1 import numpy
2 import matplotlib.pyplot as plt
3
4 # generujemy 100000 losowych liczb
5 # o średniej 10.0 oraz odchyleniu standardowym 1.0,
6 # które tworzą rozkład normalny
7 x = numpy.random.normal(10.0, 1.0, 100000)
8
9 # generujemy wykres podzielony na 100 słupków
10 plt.hist(x, 100)
11 plt.show()
```

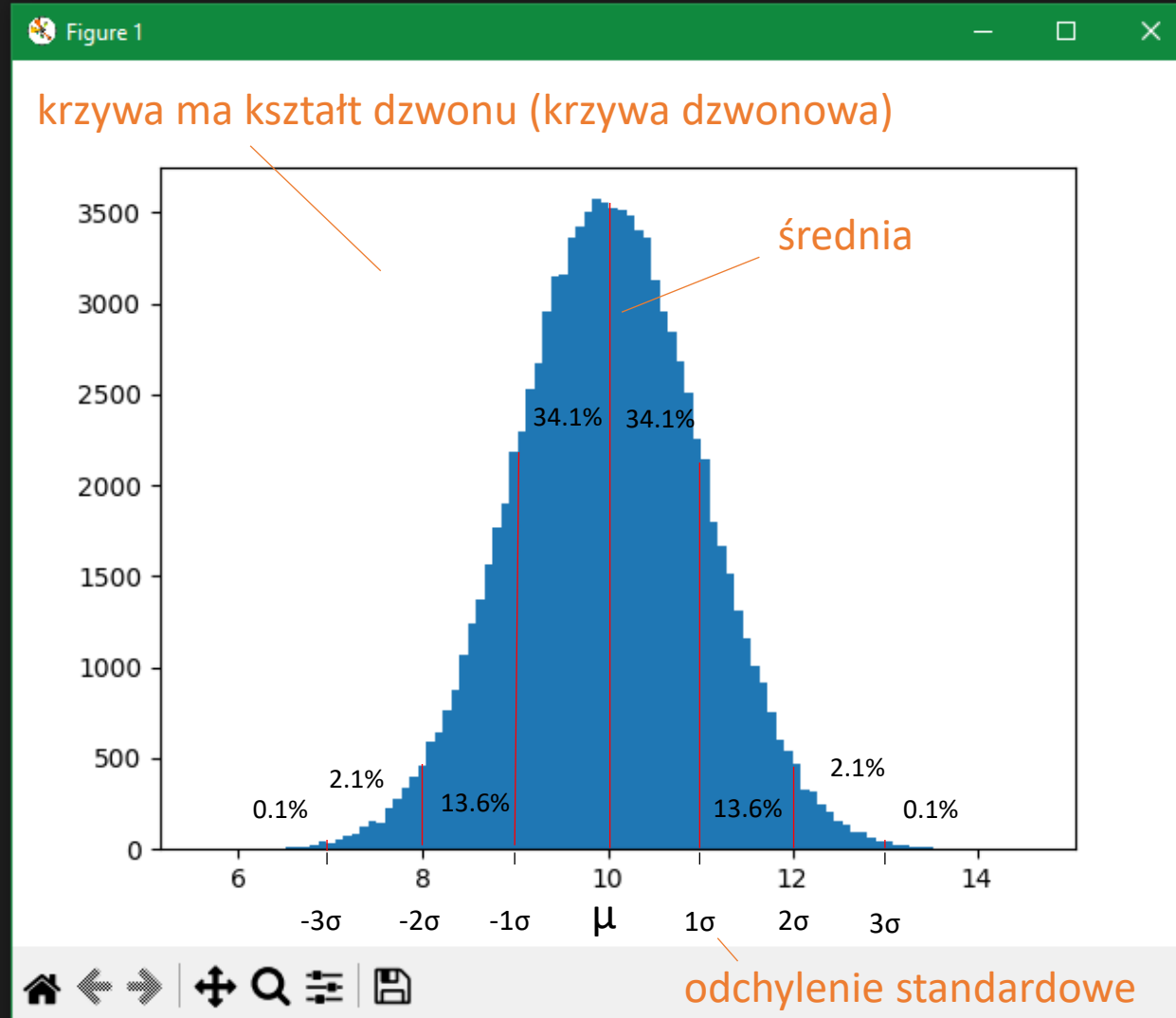
reguła 3 sigm

68.2% liczb znajduje się w zakresie pomiędzy ($\mu - \sigma$, $\mu + \sigma$)
95.4% liczb znajduje się w zakresie pomiędzy ($\mu - 2\sigma$, $\mu + 2\sigma$)
99.7% liczb znajduje się w zakresie pomiędzy ($\mu - 3\sigma$, $\mu + 3\sigma$)

σ – odchylenie standardowe (sigma)

μ – średnia (mi)

20



Normal Data Distribution

(rozkład normalny, krzywa Gaussa)

Rozkład normalny spotykamy w życiu codziennym np.:

- poziom inteligencji człowieka - najczęściej osób posiada średni poziom inteligencji, $IQ = 100$.
Dużo mniej osób posiada poziom inteligencji równy $IQ = 80$, czy $IQ = 120$, a jeszcze mniej, $IQ = 65$, czy $IQ = 135$.
Inteligencja mierzona testami inteligencji uważana jest za zmienną o rozkładzie normalnym.
- wzrost człowieka może być uznany w przybliżeniu za zmienną o rozkładzie normalnym

Normal Data Distribution

(rozkład normalny, krzywa Gaussa)

Zadanie

Wygeneruj losowy zestaw danych,
którego wartości tworzą rozkład normalny.
Wygeneruj krzywą dzwonową rozkładu.

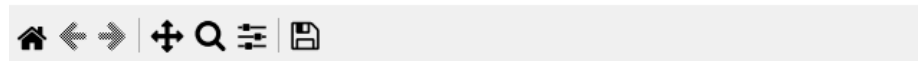
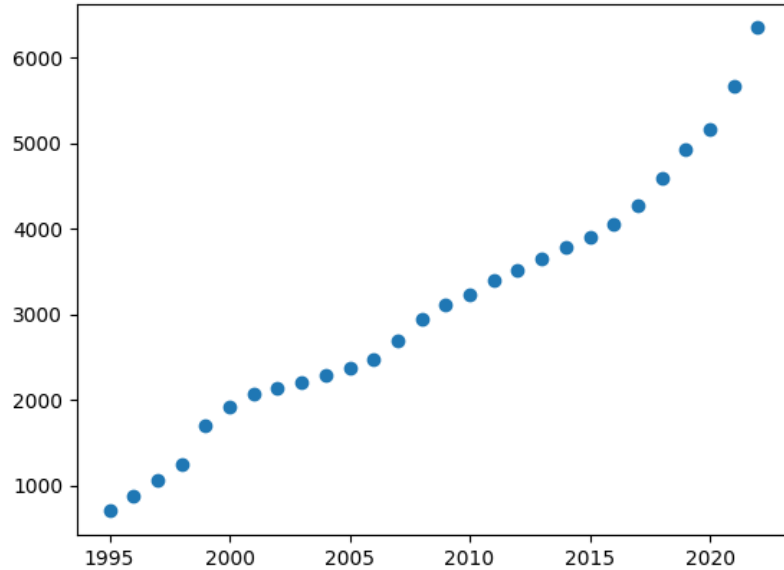
Scatter Plot (wykres punktowy)

```
scatterPlot.py > ...
1  # dane pobrane ze strony ZUS:
2  # https://www.zus.pl/baza-wiedzy/skladki-wskazniki-odsetki/wskazniki/przecietne-wynagrodzenie-w-latach
3
4  import numpy
5  import matplotlib.pyplot as plt
6
7  # lata
8  x = [1995,1996,1997,1998,1999,2000,2001,2002,2003,2004,2005,2006,2007,2008,2009,2010,
9      | 2011,2012,2013,2014,2015,2016,2017,2018,2019,2020,2021,2022]
10
11 # przeciętna pensja w Polsce
12 y = [702.62,873.00,1061.93,1239.49,1706.74,1923.81,2061.85,2133.21,2201.47,2289.57,
13      | 2380.29,2477.23,2691.03,2943.88,3102.96,3224.98,3399.52,3521.67,3650.06,3783.46,
14      | 3899.78,4047.21,4271.51,4585.03,4918.17,5167.47,5662.53,6346.15]
15
16
17 # przecietne roczne zarobki w Polsce
18 y1 = [8431.44,10476.00,12743.16,14873.88,20480.88,23085.72,24742.20,25598.52,26417.64,27474.84,
19      | 28563.48,29726.76,32292.36,35326.56,37235.52,38699.76,40794.24,42260.04,43800.72,45401.52,
20      | 46797.36,48566.52,51258.12,55020.36,59018.04,62009.64,67950.36,76153.80]
21
22
23 plt.scatter(x, y)
24 plt.show()
```

dane pobrane ze strony [ZUS](https://www.zus.pl/baza-wiedzy/skladki-wskazniki-odsetki/wskazniki/przecietne-wynagrodzenie-w-latach)

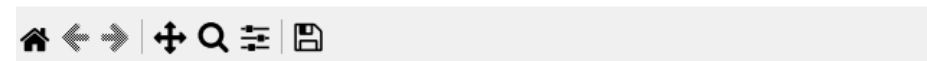
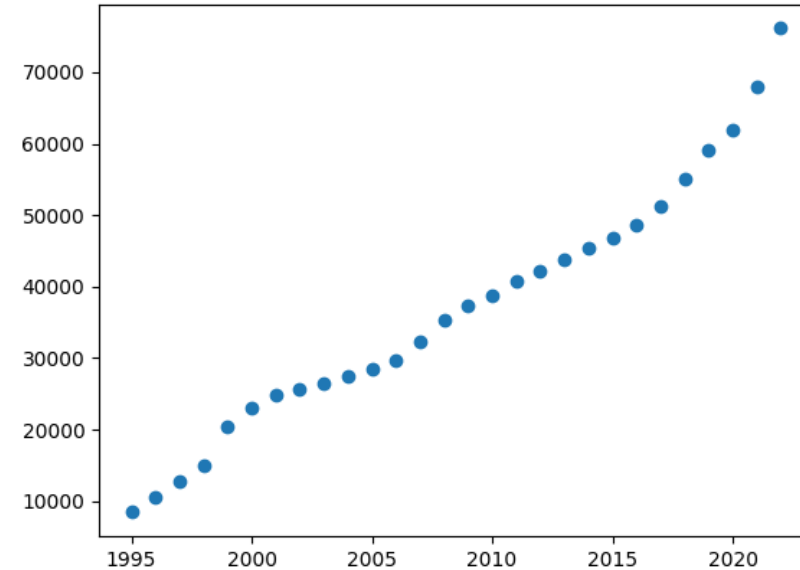
Scatter Plot (wykres punktowy)

Figure 1



przeciętne zarobki miesięczne w Polsce od 1995 roku wg. ZUS

Figure 1



przeciętne zarobki roczne w Polsce od 1995 roku wg. ZUS

Scatter Plot (wykres punktowy)

```
scatterPlotRandom.py > ...  
1 import numpy  
2 import matplotlib.pyplot as plt  
3  
4 x = numpy.random.normal(20.0, 1.0, 1000)  
5 y = numpy.random.normal(10.0, 2.0, 1000)  
6  
7 plt.scatter(x, y)  
8 plt.show()
```

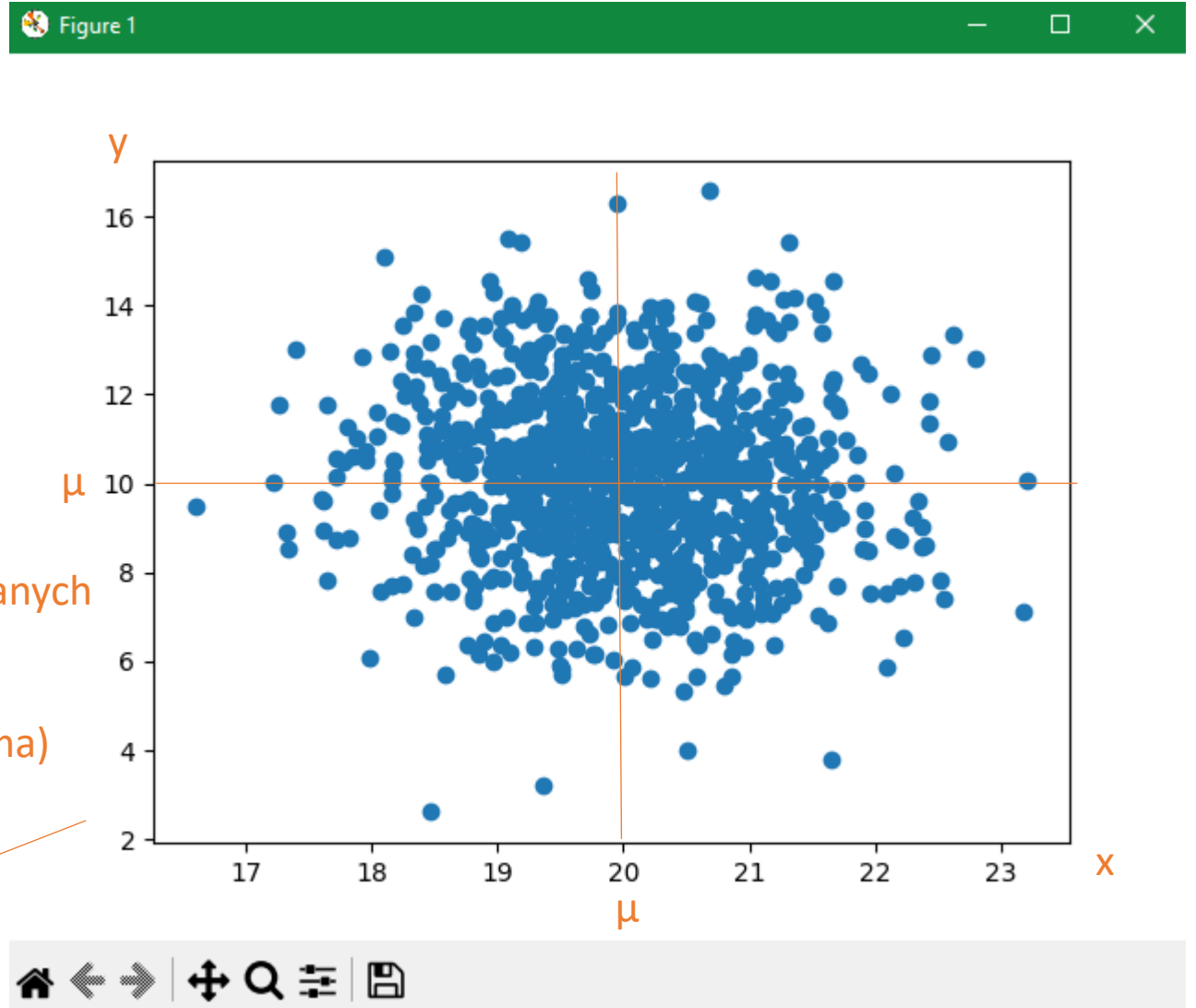
x oraz y są
rozkładami
normalnymi

μ (średnia – mi)

ilość liczb
wygenerowanych
losowo

σ (odchylenie standardowe - sigma)

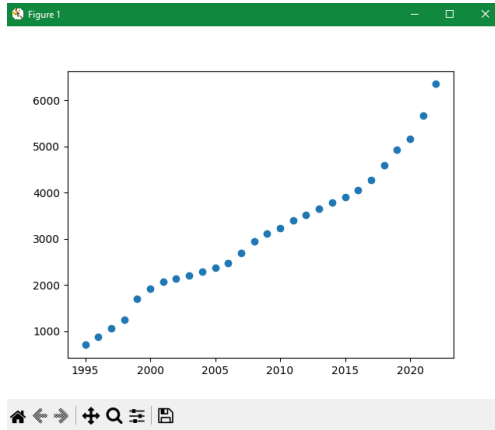
kropki koncentrują się po osi x wokół wartości 20 (średnia) oraz po osi y wokół wartości 10 (średnia). Wartości liczb są bardziej rozstrzelone po osi y (większe odchylenie standardowe - bardziej odbiegają od średniej)



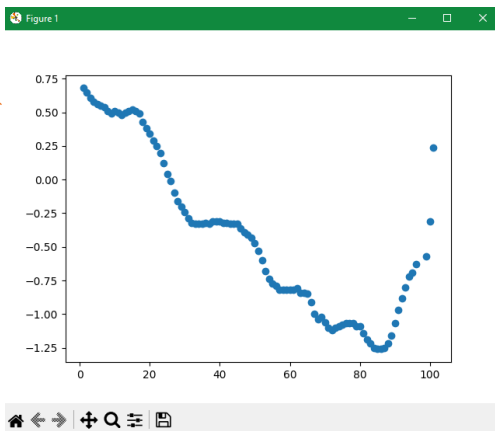
Model

aproxymacja danych za pomocą funkcji

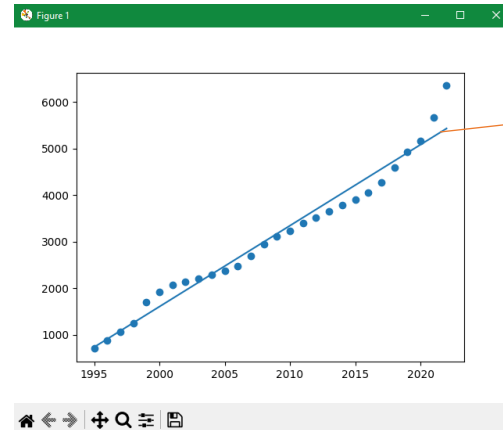
zbiór
danych



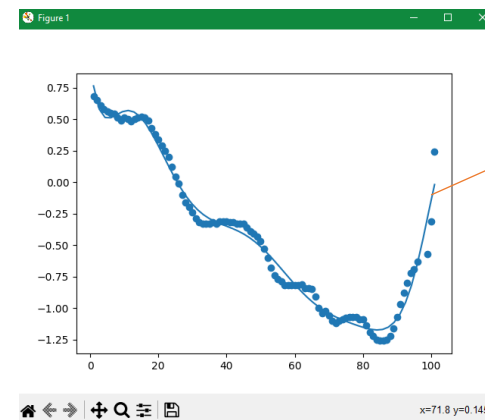
zbiór
danych



tworzenie
modelu



model liniowy
(*Linear Regression*)



model wielomianowy
(*Polynomial Regression*)

Linear Regression (Regresja liniowa)

liniowa zależność pomiędzy zmiennymi

zależność pomiędzy zmiennymi wykorzystywana do przewidywania wyniku przyszłych zdarzeń.

```
linearRegression.py > ...
1  import matplotlib.pyplot as plt
2  from scipy import stats
3
4  # lata
5  x = [1995,1996,1997,1998,1999,2000,2001,2002,2003,2004,2005,2006,2007,2008,2009,2010,
6      | 2011,2012,2013,2014,2015,2016,2017,2018,2019,2020,2021,2022]
7
8  # przeciętna pensja w Polsce
9  y = [702.62,873.00,1061.93,1239.49,1706.74,1923.81,2061.85,2133.21,2201.47,2289.57,
10     | 2380.29,2477.23,2691.03,2943.88,3102.96,3224.98,3399.52,3521.67,3650.06,3783.46,
11     | 3899.78,4047.21,4271.51,4585.03,4918.17,5167.47,5662.53,6346.15]
12
13
14  slope, intercept, r, p, std_err = stats.linregress(x, y)
15
16  def myfunc(x):
17  |   return slope * x + intercept
18
19  mymodel = list(map(myfunc, x))
20
21  plt.scatter(x, y)
22  plt.plot(x, mymodel)
23  plt.show()
```

wyliczenie
kluczowych
wartości regresji
liniowej

definicja funkcji
dopasowującej
wartości y w
zależności od x

wartości na osi x

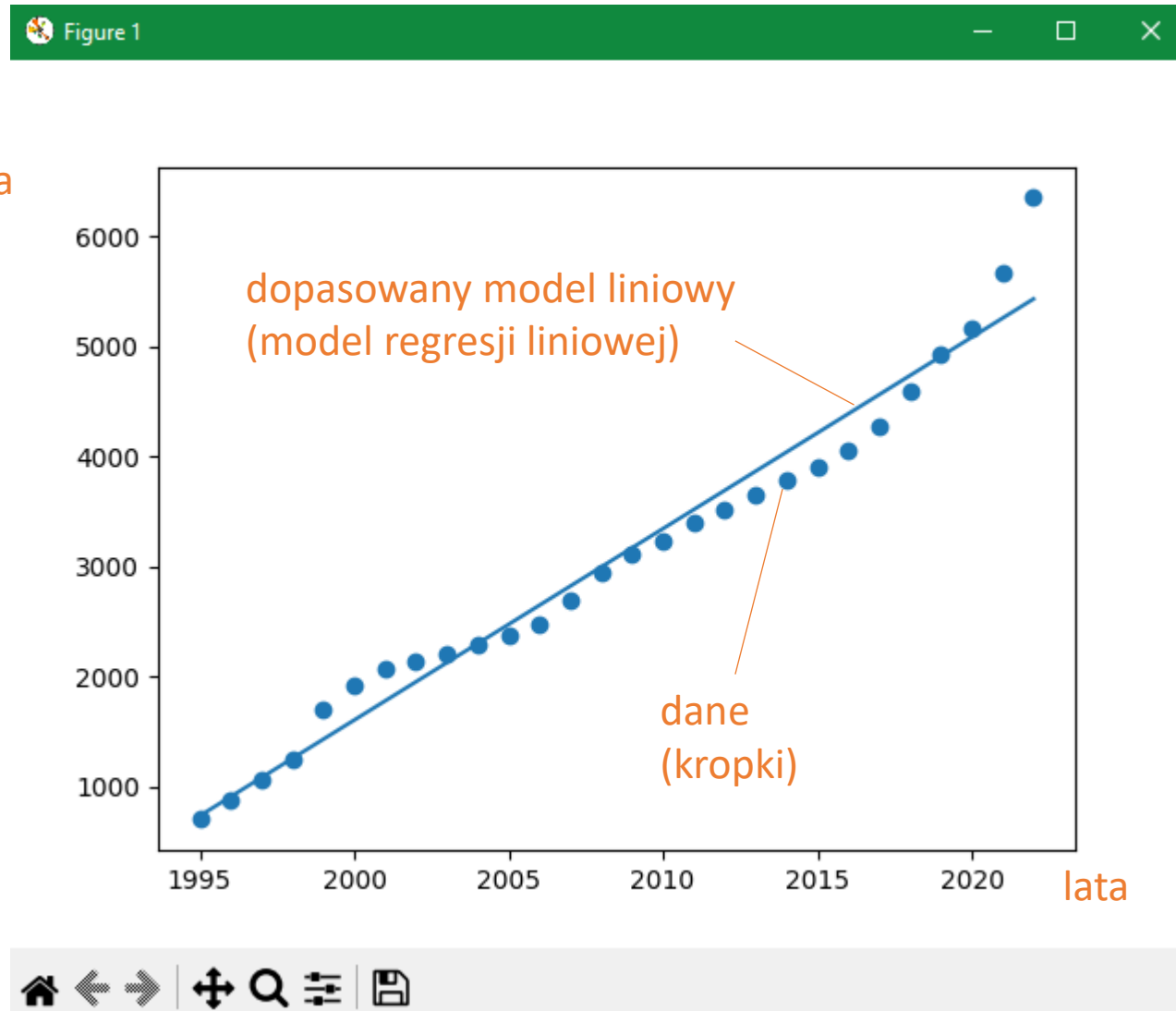
wartości na osi y

tworzenie modelu – tablicy
z dopasowanymi wartościami y

rysowanie wykresów

Linear Regression (Regresja liniowa)

przeciętna pensja w Polsce



coefficient of correlation r (współczynnik korelacji r Pearsona)

liczba określająca stopień zależności liniowej pomiędzy zmiennymi (czy taka zależność jest i w jakim stopniu)

r jak relationship (relacja)

r został opracowany przez angielskiego matematyka [Karla Pearsona](#).

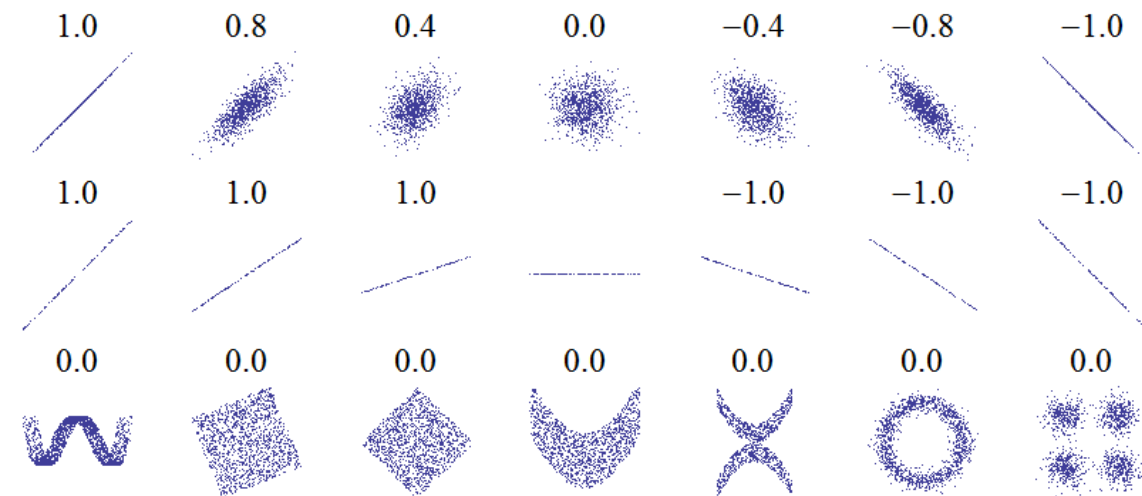
Wartość r waha się od -1 do 1:

- 0 oznacza brak zależności liniowej
- 1 oraz -1 oznacza 100% zależność liniową

za pomocą r można badać zależność danych, (czy można utworzyć regresję liniową)

silna zależność

brak zależności



coefficient of correlation r (współczynnik korelacji r Pearsona)

```
linearRegressionR.py > ...
1  # coefficient of correlation r
2  # współczynnik korelacji r Pearsona
3  # Czy regresja liniowa dobrze pasuje do danych?
4
5
6  import matplotlib.pyplot as plt
7  from scipy import stats
8
9  # lata
10 x = [1995,1996,1997,1998,1999,2000,2001,2002,2003,2004,2005,2006,2007,2008,2009,2010,
11      | 2011,2012,2013,2014,2015,2016,2017,2018,2019,2020,2021,2022]
12
13 # przeciętna pensja w Polsce
14 y = [702.62,873.00,1061.93,1239.49,1706.74,1923.81,2061.85,2133.21,2201.47,2289.57,
15      | 2380.29,2477.23,2691.03,2943.88,3102.96,3224.98,3399.52,3521.67,3650.06,3783.46,
16      | 3899.78,4047.21,4271.51,4585.03,4918.17,5167.47,5662.53,6346.15]
17
18 slope, intercept, r, p, std_err = stats.linregress(x, y)
19
20 print(r)
```

↓
0.983556773600779

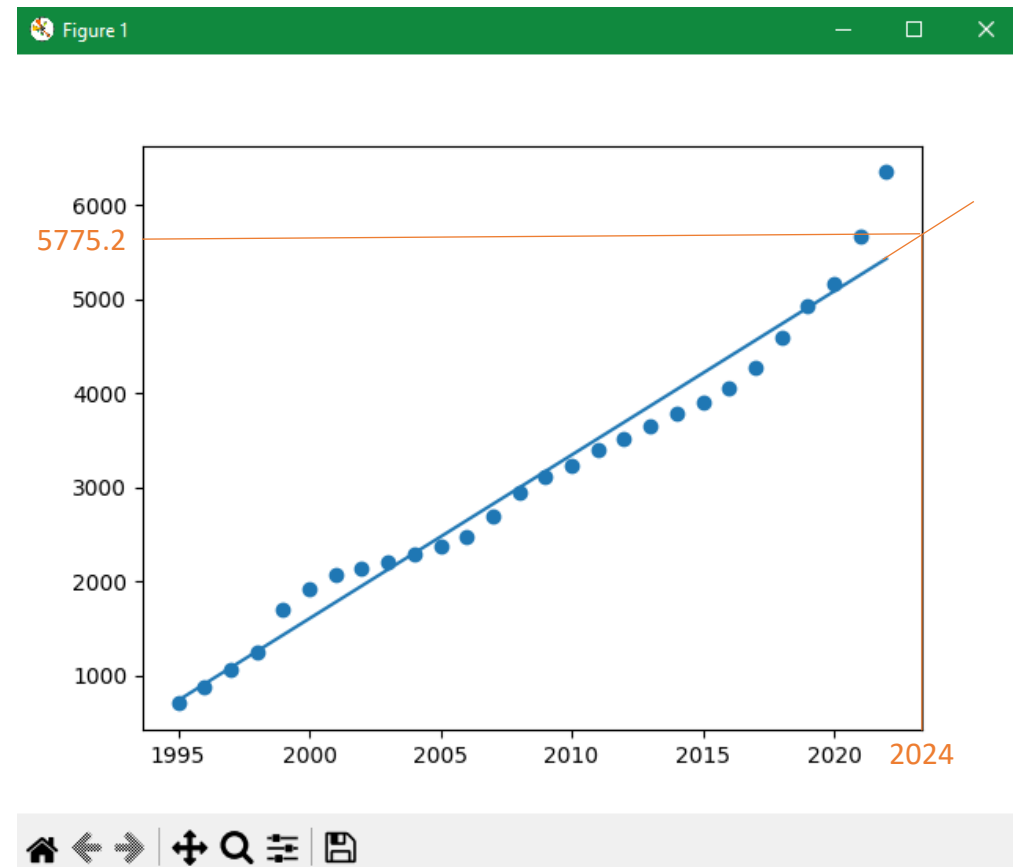
duża zależność liniowa y od x (model liniowy przeciętnej pensji w Polsce wydaje się być dobrze dopasowanym)

Linear Regression (Regresja liniowa)

jeśli r jest bliskie jedynki, możemy przewidzieć przyszłość

```
linearRegressionPredictFuture.py > ...
1  from scipy import stats
2
3  # lata
4  x = [1995,1996,1997,1998,1999,2000,2001,2002,2003,2004,2005,2006,2007,2008,2009,2010,
5      | 2011,2012,2013,2014,2015,2016,2017,2018,2019,2020,2021,2022]
6
7  # przeciętna pensja w Polsce
8  y = [702.62,873.00,1061.93,1239.49,1706.74,1923.81,2061.85,2133.21,2201.47,2289.57,
9      | 2380.29,2477.23,2691.03,2943.88,3102.96,3224.98,3399.52,3521.67,3650.06,3783.46,
10     | 3899.78,4047.21,4271.51,4585.03,4918.17,5167.47,5662.53,6346.15]
11
12 slope, intercept, r, p, std_err = stats.linregress(x, y)
13
14 def myfunc(x):
15     return slope * x + intercept
16
17 pensja2024 = myfunc(2024)
18
19 print(pensja2024)
```

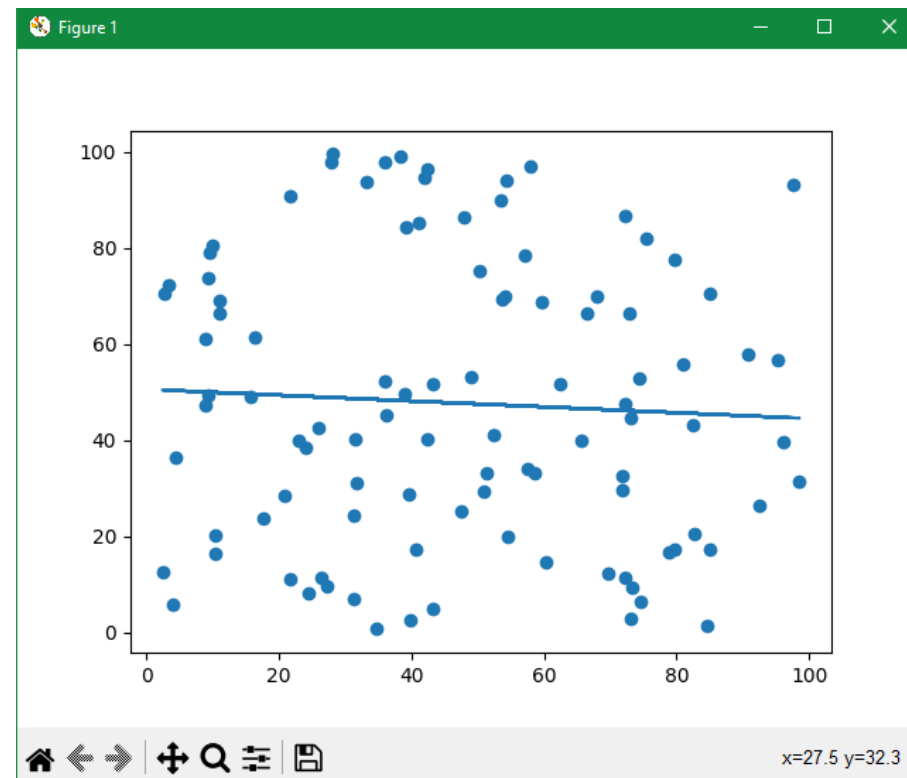
5775.192698412691



liczymy jaka będzie przeciętna pensja w Polsce w 2024 roku

Linear Regression (Regresja liniowa)

```
linearRegressionBadFit.py > ...
1 # złe dopasowanie regresji liniowej
2
3 import numpy
4 import matplotlib.pyplot as plt
5 from scipy import stats
6
7 # generujemy tablice z 100 losowymi liczbami zmiennoprzecinkowymi
8 # od 0.0 do 100.0
9 x = numpy.random.uniform(0.0, 100.0, 100)
10 y = numpy.random.uniform(0.0, 100.0, 100)
11
12 slope, intercept, r, p, std_err = stats.linregress(x, y)
13
14 def myfunc(x):
15     return slope * x + intercept
16
17 mymodel = list(map(myfunc, x))
18
19 plt.scatter(x, y)
20 plt.plot(x, mymodel)
21 plt.show()
22
23 print(r)
```



ten zestaw danych nie nadaje się do regresji liniowej.

-0.023567404078000325

współczynnik korelacji r Pearsona jest bliski zeru – słaba zależność liniowa między zmiennymi x oraz y

Scatter Plot, Linear Regression, r

Zadanie

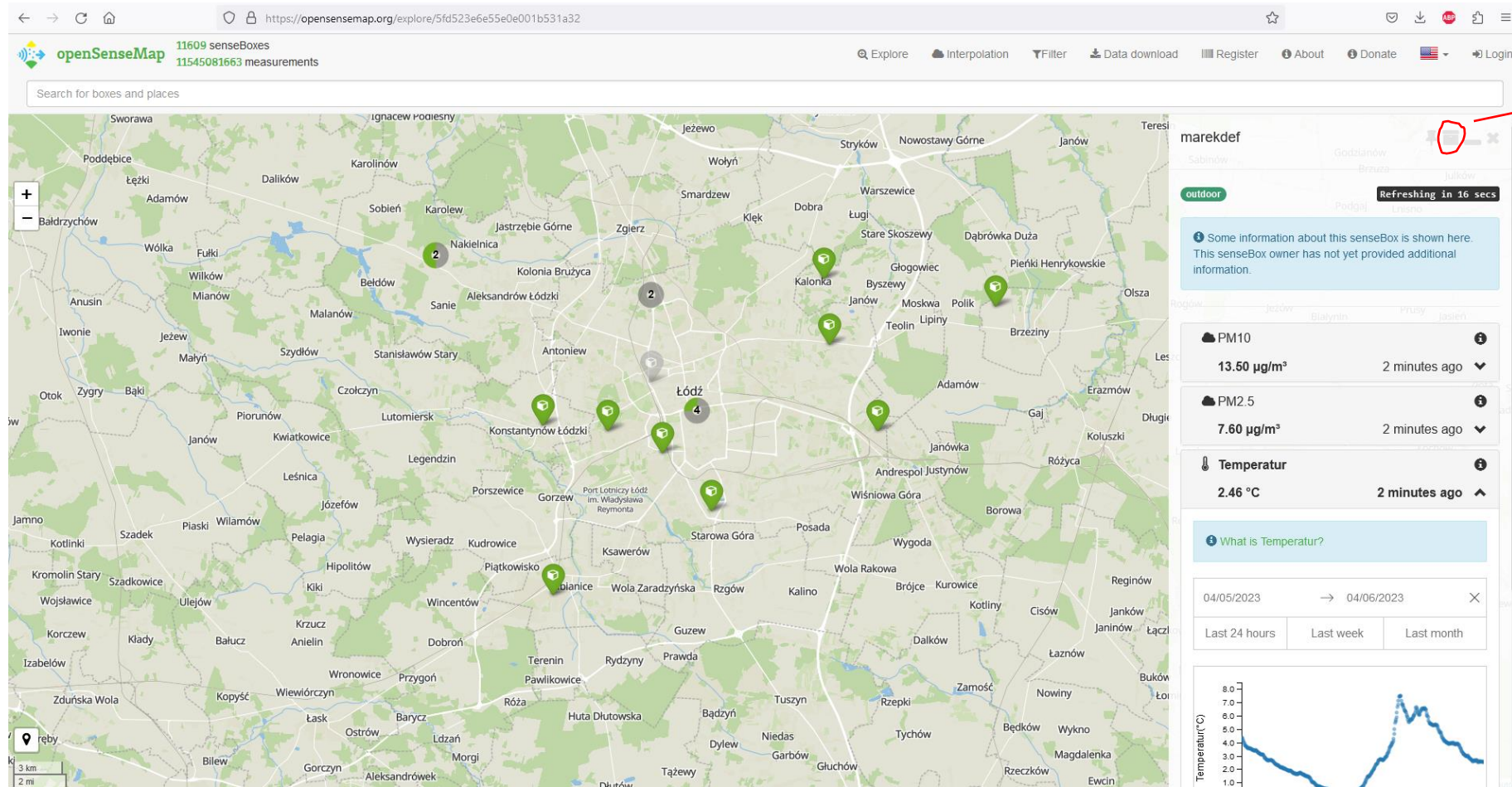
Znajdź dane, które nadają się do przedstawienia w formie wykresu punktowego i zastosuj do nich regresję liniową (możesz skorzystać z danych [ZUS](#))
Wylicz współczynnik korelacji r Pearsona, oceń stopień zależności liniowej pomiędzy zmiennymi, znajdź nowe dane na podstawie regresji liniowej.

Dane pogodowe

mapa czujników wilgotności, temperatury, ciśnienia, zanieczyszczeń powietrza itp..

data, godzina

wartość temperatury



	A	B	C
1	createdAt	value	
2	2023-04-05T00:00:12.195Z	0.68	1
3	2023-04-05T00:03:18.825Z	0.65	2
4	2023-04-05T00:06:13.310Z	0.61	3
5	2023-04-05T00:09:04.477Z	0.58	4
6	2023-04-05T00:11:57.128Z	0.56	5
7	2023-04-05T00:14:47.522Z	0.55	6
8	2023-04-05T00:17:38.312Z	0.54	7
9	2023-04-05T00:20:29.546Z	0.51	8
10	2023-04-05T00:23:21.507Z	0.49	9
11	2023-04-05T00:26:12.169Z	0.51	10
12	2023-04-05T00:29:03.119Z	0.50	11
13	2023-04-05T00:31:53.710Z	0.48	12
14	2023-04-05T00:34:44.591Z	0.50	13
15	2023-04-05T00:37:35.309Z	0.51	14
16	2023-04-05T00:40:26.204Z	0.52	15
17	2023-04-05T00:43:16.870Z	0.51	16
18	2023-04-05T00:46:07.856Z	0.49	17
19	2023-04-05T00:48:58.693Z	0.43	18
20	2023-04-05T00:51:49.149Z	0.38	19
21	2023-04-05T00:54:40.249Z	0.34	20
22	2023-04-05T00:57:31.269Z	0.29	21
23	2023-04-05T01:00:23.195Z	0.25	22
24	2023-04-05T01:03:14.317Z	0.20	23
25	2023-04-05T01:06:11.030Z	0.12	24
26	2023-04-05T01:09:05.628Z	0.04	25

dane pobrane z jednego czujnika w Łodzi (temperatura w nocy 5.04.23)

Polynomial Regression (Aproksymacja wielomianowa)

czas od 12.00 w
nocy do 4.45 nad
ranem

(przeliczony na
system
dziesiętny)

wartości
temperatur

określenie
rozciągłości liczb po
osi x – od 1 do 101

ponieważ
dopasowanie modelu
do danych jest dobre
(R-Squared = 0.999)
można na tej
podstawie określić
temperaturę w czasie
późniejszym, niż
pobrane dane
(godzina ok. 4.48)

```
polynomialRegression.py > ...
1 import numpy
2 import matplotlib.pyplot as plt
3 from sklearn.metrics import r2_score
4
5 time = [
6 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,
7 38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,
8 72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,99,100,101]
9
10 temperature = [
11 0.68,0.65,0.61,0.58,0.56,0.55,0.54,0.51,0.49,0.51,0.50,0.48,0.50,0.51,0.52,0.51,0.49,0.43,0.38,0.34,0.29,0.25,
12 0.20,0.12,0.04,-0.01,-0.10,-0.16,-0.20,-0.24,-0.29,-0.32,-0.33,-0.33,-0.33,-0.32,-0.33,-0.31,-0.31,-0.31,-0.32,
13 -0.32,-0.33,-0.33,-0.33,-0.36,-0.39,-0.41,-0.43,-0.47,-0.53,-0.60,-0.68,-0.74,-0.77,-0.79,-0.82,-0.82,-0.82,
14 -0.82,-0.82,-0.81,-0.84,-0.84,-0.85,-0.91,-1.00,-1.04,-1.02,-1.06,-1.10,-1.12,-1.10,-1.09,-1.08,-1.07,-1.07,
15 -1.07,-1.09,-1.09,-1.14,-1.19,-1.22,-1.25,-1.26,-1.26,-1.25,-1.22,-1.16,-1.07,-0.97,-0.88,-0.80,-0.72,-0.69,
16 -0.63,-0.57,-0.31,0.24]
17
18 mymodel = numpy.poly1d(numpy.polyfit(time, temperature, 20))
19
20 myline = numpy.linspace(1, 101, 100)
21
22 plt.scatter(time, temperature)
23 plt.plot(myline, mymodel(myline))
24 plt.show()
25
26 print(r2_score(temperature, mymodel(time))) → 0.9990408357796161
27
28 temp = mymodel(102)
29 print(temp) → 1.6768274722494845
```

`pip install scikit-learn`

wyliczenie wielomianowego modelu

parametry określające dopasowanie modelu, czym większe, tym lepsze dopasowanie

time rozciąga się od 1 do 101

wrysowanie diagramów

R-Squared (R-Kwadrat) - określa stopień zależności wielomianowej pomiędzy zmiennymi:

- 0 oznacza brak zależności
- 1 oznacza 100% zależność

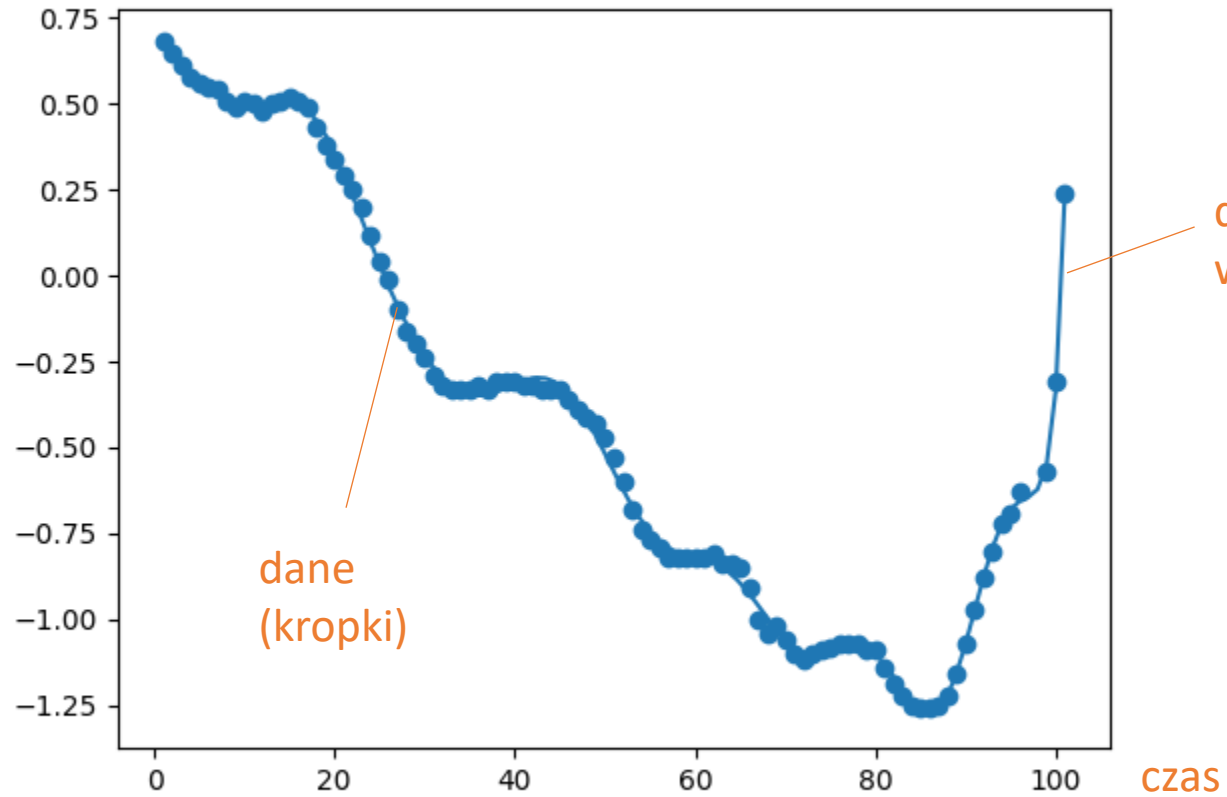
→ 0.9990408357796161

→ 1.6768274722494845 — temperatura o godz. ok. 4.48

Polynomial Regression (Aproksymacja wielomianowa)

Figure 1

temperatura



Polynomial Regression

Zadanie

Znajdź dane, które nadają się do przedstawienia w formie wykresu punktowego i zastosuj do nich regresję wielomianową.

Wylicz współczynnik korelacji R-Squared, oceń stopień zależności wielomianowej pomiędzy zmiennymi, znajdź nowe dane na podstawie wyliczonego modelu.

Multiple Regression (Regresja wielokrotna)

Regresja wielokrotna jest podobna do regresji liniowej, ale z więcej niż jedną niezależną wartością, co oznacza, że próbujemy przewidzieć wartość na podstawie dwóch lub więcej zmiennych

import danych
w postaci pliku csv



Użytkowanie lasu - pozyskanie drewna

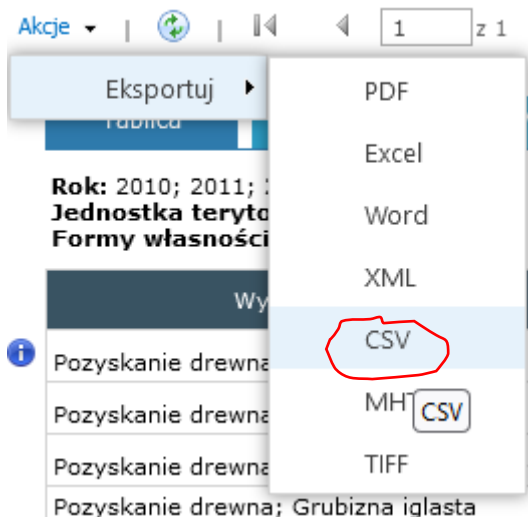
Akcje | 1 z 1 | Znajdź Dalej

Tablica Wykres Mapa Metadane

Rok: 2010; 2011; 2012; 2013; 2014; 2015; 2016; 2017; 2018; 2019; 2020; 2021
Jednostka terytorialna: POLSKA
Formy własności leśnej: Ogółem

Wyszczególnienie	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021
Pozyskanie drewna; Ogółem [m³]	35 467 471	37 179 983	37 044 584	37 944 495	39 740 543	40 247 249	40 900 708	44 275 096	45 589 688	42 365 686	39 668 819	42 244 165
Pozyskanie drewna; Grubizna [m³]	33 568 291	34 877 069	34 977 961	35 796 037	37 661 503	38 327 082	39 129 329	42 698 966	43 931 540	40 637 873	38 064 321	40 691 613
Pozyskanie drewna; Grubizna iglasta [m³]	25 579 421	26 277 617	26 042 317	26 791 974	28 533 037	29 077 586	30 077 828	33 470 400	34 352 104	31 688 773	29 611 049	32 073 489
Pozyskanie drewna; Grubizna iglasta wielkowymiarowa [m³]	11 691 348	11 877 257	11 747 699	12 111 278	13 108 536	13 189 439	13 589 579	14 904 843	14 840 917	14 141 442	14 021 756	15 550 129
Pozyskanie drewna; Grubizna iglasta wielkowymiarowa ogólnego przeznaczenia [m³]	11 625 095	11 807 330	11 672 384	12 031 565	13 016 583	13 091 438	13 502 234	14 815 930	14 753 209	14 039 990	13 899 174	15 418 346
Pozyskanie drewna; Grubizna iglasta wielkowymiarowa specjalna [m³]	66 253	69 927	75 315	79 713	91 953	98 001	87 345	88 913	87 708	101 452	122 582	131 783
Pozyskanie drewna; Grubizna iglasta średniowymiarowa [m³]	12 523 720	12 832 631	12 661 021	13 026 932	13 729 067	14 154 665	14 822 316	16 738 496	17 687 342	15 978 810	14 215 003	15 231 518
Pozyskanie drewna; Grubizna iglasta średniowymiarowa dtuzycowa [m³]	541 811	502 403	449 049	412 695	388 278	311 889	269 234	245 152	188 274	170 443	162 209	148 820
Pozyskanie drewna; Grubizna iglasta średniowymiarowa do przerobu przemysłowego [m³]	11 981 909	12 330 228	12 211 972	12 614 237	13 340 789	13 842 776	14 553 082	16 493 344	17 499 068	15 808 367	14 052 794	15 082 698
Pozyskanie drewna; Grubizna iglasta opałowa [m³]	1 364 383	1 567 727	1 633 598	1 653 764	1 695 433	1 733 481	1 665 931	1 827 061	1 823 848	1 568 520	1 374 291	1 291 837
Pozyskanie drewna; Grubizna liściasta [m³]	7 988 866	8 599 452	8 935 644	9 004 062	9 128 466	9 249 496	9 051 501	9 228 567	9 579 437	8 949 100	8 453 272	8 618 124
Pozyskanie drewna; Grubizna liściasta wielkowymiarowa [m³]	2 628 571	2 844 392	2 827 372	2 779 589	2 831 368	2 773 126	2 717 635	2 668 031	2 613 894	2 539 171	2 542 791	2 619 691
Pozyskanie drewna; Grubizna liściasta wielkowymiarowa ogólnego przeznaczenia [m³]	2 436 812	2 654 949	2 646 411	2 588 806	2 636 288	2 561 888	2 511 680	2 477 718	2 420 763	2 349 130	2 336 480	2 403 090
Pozyskanie drewna; Grubizna liściasta												

dana na stronie Głównego Urzędu
Statystycznego:
Użytkowanie lasu - pozyskanie drewna



te dane zaimportujemy do słownika

http://swaid.stat.gov.pl/Lesnictwo_dashboards/Raporty_predefiniowane/RAP_DBD_LES_6.aspx

https://www.w3schools.com/python/python_ml_multiple_regression.asp

Multiple Regression

Dane ze strony Urzędu Statystycznego „Użytkowanie lasu - pozyskanie drewna” zostały zaimportowane do słownika w module multiRegressionModule

słownik ma budowę:
wood = {
[klucz: wartość],
„years”:[2010,2011,...],
„ogółem”:[35467471,...],
„grubizna”:[33568291,...]
}

[42801232.27077913]

```
multipleRegression.py X
programy > multipleRegression.py > ...
1 import multipleRegressionModule as data # moduł do importu danych z pliku multipleRegression.txt
2
3 import pandas as pd
4 from sklearn import linear_model
5
6 # pobieramy dane o drewnie z pliku do słownika
7 wood = data.getData()
8
9 #print(wood)
10
11 # printujemy klucze ze słownika
12 woodKeys = wood.keys()
13 #print(woodKeys)
14 '''
15 dict_keys(['years', 'ogółem', 'grubizna', 'grubizna_iglasta', 'grubizna_iglasta_wielkowymiarowa', 'grubizna_iglasta_wielkowymiarowa_ogólnego_przeznaczenia', 'grubizna_iglasta_wielkowymiarowa_specjalna', 'grubizna_iglasta_średniowymiarowa', 'grubizna_iglasta_średniowymiarowa_dłużycowa', 'grubizna_iglasta_średniowymiarowa_do_przerobu_przemysłowego', 'grubizna_iglasta_opałowa', 'grubizna_liściasta', 'grubizna_liściasta_wielkowymiarowa', 'grubizna_liściasta_wielkowymiarowa_ogólnego_przeznaczenia', 'grubizna_liściasta_wielkowymiarowa_specjalna', 'grubizna_liściasta_średniowymiarowa', 'grubizna_liściasta_średniowymiarowa_dłużycowa', 'grubizna_liściasta_średniowymiarowa_do_przerobu_przemysłowego', 'grubizna_liściasta_opałowa', 'drewno_małowymiarowe', 'drewno_małowymiarowe_do_przerobu_przemysłowego', 'drewno_małowymiarowe_opałowe', 'karpina'])
16
17
18
19
20
21 '''
22
23 # tworzymy obiekt DataFrame na podstawie danych w słowniku
24 df = pd.DataFrame(wood)
25 print(df)
26
27 # lista niezależnych zmiennych na podstawie których jest wyliczana zależna zmienna
28 X = df[['years', 'drewno_małowymiarowe_opałowe']]
29
30 # zmienna zależna, którą wyliczamy
31 y = df['ogółem']
32
33 # tworzymy obiekt liniowej regresji
34 regr = linear_model.LinearRegression()
35
36 # przekazujemy listy zależnych zmiennych i zmienną niezależną do obiektu regresji liniowej
37 regr.fit(X.values, y)
38
39 # sprawdzamy ile drewna będzie ogółem w roku 2024, jeśli Drewno małowymiarowe opałowe ma podaną wartość
40 predictedOgolem = regr.predict([[2024, 1785030]])
41
42 print(predictedOgolem)
```

pip install pandas

moduł Pandas jest używany do analizowania danych.
[pandas tutorial](#)

Multiple Regression

zawartość obiektu
DataFrame
(dane statystyczne
dotyczące
pozyskiwania
drewna)

```
   years  ogółem  ...  drewno_małowymiarowe_opałowe  karpina
0   2010  35467471  ...  1407841  70
1   2011  37179983  ...  1785030  39
2   2012  37044584  ...  1618597  133
3   2013  37944495  ...  1692744  1556
4   2014  39740543  ...  1654558  2224
5   2015  40247249  ...  1512322  268
6   2016  40900708  ...  1389381  0
7   2017  44275096  ...  1239355  428
8   2018  45589688  ...  1350067  7158
9   2019  42365686  ...  1461952  11499
10  2020  39668819  ...  1437412  4378
11  2021  42244165  ...  1407423  1649

[12 rows x 23 columns]
[42801232.27077913]
```

wyliczona wartość
(ile drewna będzie ogółem w roku 2024, jeśli Drewna małowymiarowego opałowego pozyskano 1785030 m³)

Multiple Regression

Zadanie

Znajdź dane, które mają wiele zmiennych.
Z pomocą Regresji Wielokrotnej spróbuj
zbudować model oraz na jego podstawie
przewidzieć nową wartość korzystając z dwóch
lub więcej zmiennych.

Scale

skalowanie danych

Jeśli porównywane dane występują w różnych jednostkach np. metry i sekundy, albo mają dużą różnicę w wartościach np. 10000 oraz 1, można je przeskalować do wartości, które są łatwiejsze do porównania.

wzór na skalowanie danych metodą *standardization* (normalizacja)

$$z = (x - \mu) / \sigma$$

nowa, przeskalowana wartość

oryginalna wartość

średnia arytmetyczna danych

odchylenie standardowe danych

Scale

skalujemy dane

```
scale.py x
programy > scale.py > ...
1 import multipleRegressionModule as data # moduł do importu danych z pliku multipleRegression.txt
2
3 import pandas as pd
4 from sklearn import linear_model
5 from sklearn.preprocessing import StandardScaler
6 scale = StandardScaler()
7
8 # pobieramy dane o drewnie z pliku do słownika
9 wood = data.getData()
10
11 # tworzymy nowy słownik z 3 kluczami, których wartości bardzo się różnią
12 newWood = {'years':wood['years'],
13           'ogółem':wood['ogółem'],
14           'grubizna': wood['karpina']}
15
16 #print(newWood)
17
18 # skalujemy dane
19
20 # tworzymy obiekt DataFrame na podstawie danych w słowniku
21 df = pd.DataFrame(newWood)
22 print("dane przed skalowaniem: ")
23 print(df)
24
25 # lista zmiennych do przeskalowania
26 X = df[['years', 'ogółem','grubizna']]
27
28 # skalujemy wartości w kolumnach
29 print("dane po przeskalowaniu: ")
30 scaledX = scale.fit_transform(X)
31
32 print(scaledX)
```

biblioteki pomocne przy skalowaniu danych

```
• dane przed skalowaniem:
  years  ogółem  grubizna
0   2010  35467471    70
1   2011  37179983    39
2   2012  37044584   133
3   2013  37944495  1556
4   2014  39740543  2224
5   2015  40247249   268
6   2016  40900708    0
7   2017  44275096   428
8   2018  45589688  7158
9   2019  42365686 11499
10  2020  39668819  4378
11  2021  42244165  1649

dane po przeskalowaniu:
[[-1.59325501 -1.63039318 -0.69447267]
 [-1.30357228 -1.04319552 -0.70351769]
 [-1.01388955 -1.08962204 -0.67609086]
 [-0.72420682 -0.78105449 -0.26089531]
 [-0.43452409 -0.16521343 -0.06598974]
 [-0.14484136  0.00852934 -0.63670126]
 [ 0.14484136  0.23259177 -0.71489691]
 [ 0.43452409  1.38962468 -0.59001729]
 [ 0.72420682  1.84038086  1.37362717]
 [ 1.01388955  0.7349133  2.64022161]
 [ 1.30357228 -0.18980664  0.5624932 ]
 [ 1.59325501  0.69324536 -0.23376025]]
```


Scale

wyliczamy za pomocą skalowania ile drewna będzie ogółem w roku 2024, jeśli Drewna małowymiarowego opałowego pozyskano 1785030 m³

```
scaleMultipleRegression.py X
programy > scaleMultipleRegression.py > ...
1 import multipleRegressionModule as data # moduł do importu danych z pliku multipleRegression.txt
2
3 import pandas as pd
4 from sklearn import linear_model
5 from sklearn.preprocessing import StandardScaler
6 scale = StandardScaler()
7
8
9 # pobieramy dane o drewnie z pliku do słownika
10 wood = data.getData()
11
12 # tworzymy obiekt DataFrame na podstawie danych w słowniku
13 df = pd.DataFrame(wood)
14 print("dane przed skalowaniem: ")
15 print(df)
16
17 # lista niezależnych zmiennych na podstawie których jest wyliczana zależna zmienna
18 X = df[['years', 'drewno_małowymiarowe_opałowe']]
19
20 # zmienna zależna, którą wyliczamy
21 y = df['ogółem']
22
23 # bierzemy same wartości bez nagłówków kolumn
24 X = X.values
25
26 # skalujemy wartości w kolumnach
27 print("dane po przeskalowaniu: ")
28 scaledX = scale.fit_transform(X)
29 print(scaledX)
30
31 # tworzymy obiekt liniowej regresji
32 regr = linear_model.LinearRegression()
33
34 # przekazujemy listy zależnych zmiennych i zmienną niezależną do obiektu regresji liniowej
35 regr.fit(scaledX, y)
36
37 # skalujemy parametry
38 scaled = scale.transform([[2024, 1785030]])
39
40 # sprawdzamy ile drewna będzie ogółem w roku 2024, jeśli Drewno małowymiarowe opałowe ma podaną wartość
41 predictedOgolem = regr.predict([scaled[0]])
42
43 print(predictedOgolem) # [42801232.27077918]
```

```
dane przed skalowaniem:
   years  ogółem  ...  drewno_małowymiarowe_opałowe  karpina
0   2010  35467471  ...                1407841             70
1   2011  37179983  ...                1785030             39
2   2012  37044584  ...                1618597            133
3   2013  37944495  ...                1692744            1556
4   2014  39740543  ...                1654558            2224
5   2015  40247249  ...                1512322             268
6   2016  40900708  ...                1389381              0
7   2017  44275096  ...                1239355             428
8   2018  45589688  ...                1350067            7158
9   2019  42365686  ...                1461952           11499
10  2020  39668819  ...                1437412            4378
11  2021  42244165  ...                1407423            1649
```

```
[12 rows x 23 columns]
dane po przeskalowaniu:
[[-1.59325501 -0.57822374]
 [-1.30357228  1.88481055]
 [-1.01388955  0.79800742]
 [-0.72420682  1.28218539]
 [-0.43452409  1.03283181]
 [-0.14484136  0.10403445]
 [ 0.14484136 -0.69876706]
 [ 0.43452409 -1.67843291]
 [ 0.72420682 -0.95548645]
 [ 1.01388955 -0.22488033]
 [ 1.30357228 -0.38512588]
 [ 1.59325501 -0.58095327]]
[42801232.27077918]
```

wyliczona wartość przy użyciu skalowania danych jest prawie identyczna z tą wyliczoną bez skalowania

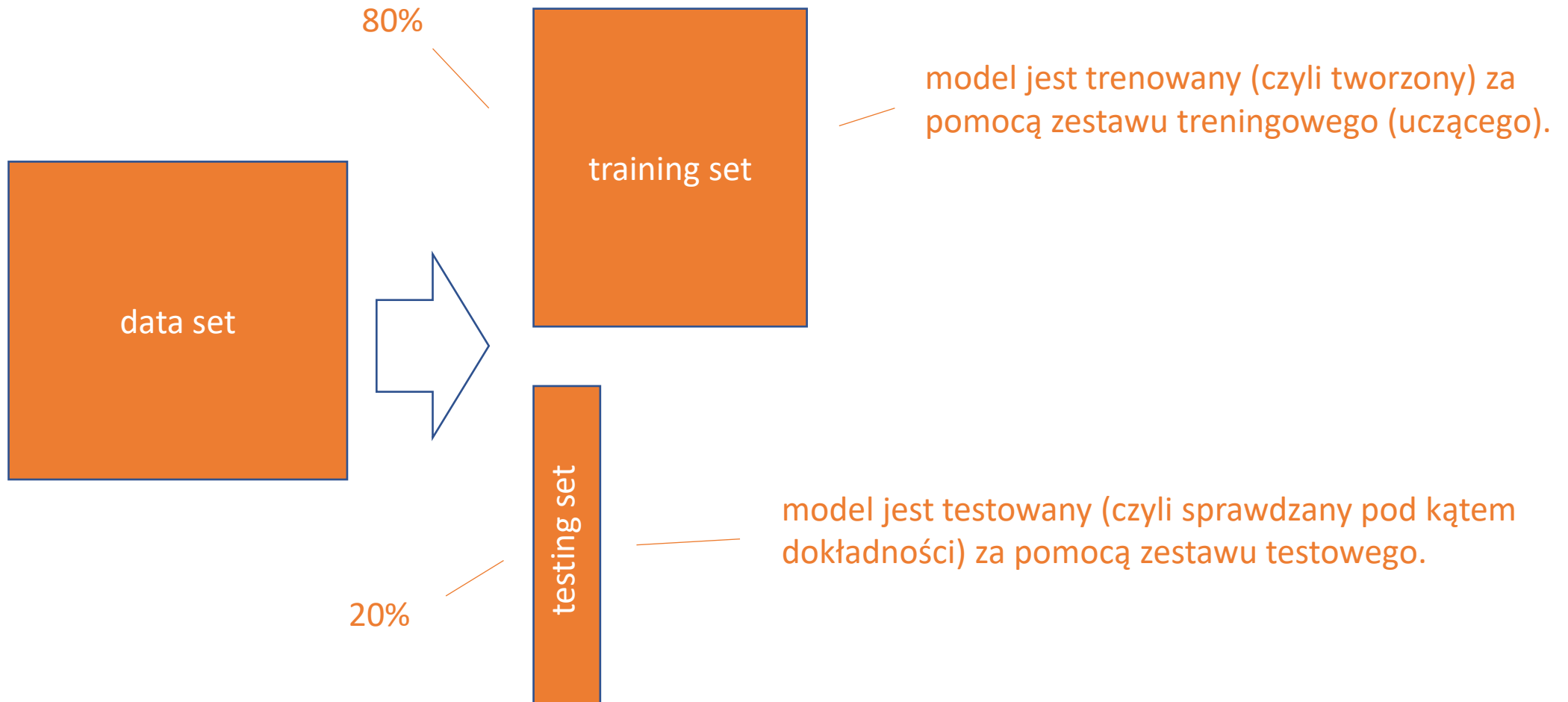
Scale

Zadanie

Znajdź dane, które mają wiele zmiennych.
Z pomocą Regresji Wielokrotnej i skalowania danych spróbuj zbudować model oraz na jego podstawie przewidzieć nową wartość korzystając z dwóch lub więcej zmiennych.

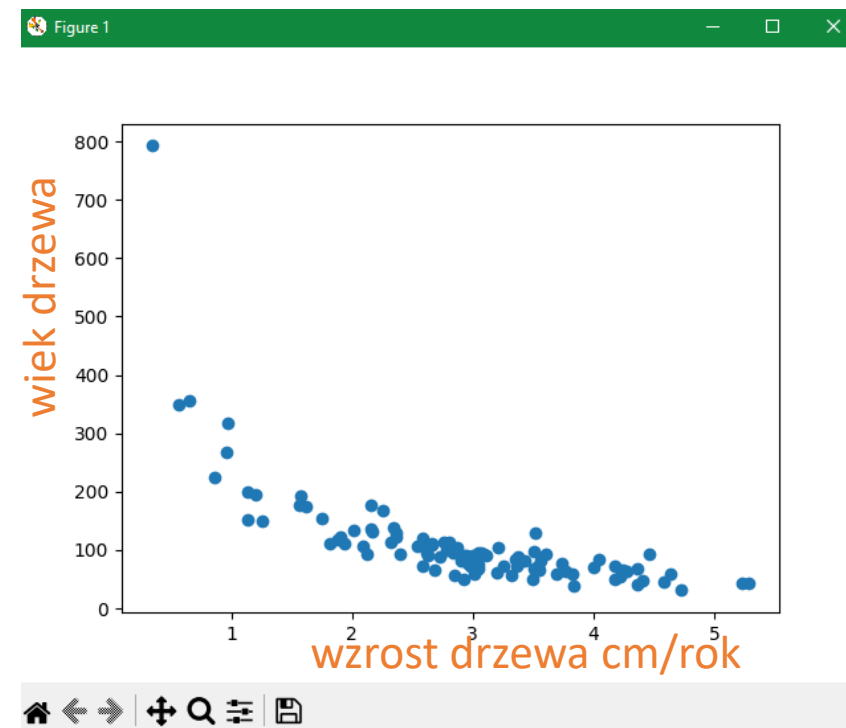
Train/Test

metoda ewaluacji (oceny) modelu polegająca na podziale zbioru danych na dwie części: dane treningowe (*training set*) oraz testowe (*testing set*)



Train/Test

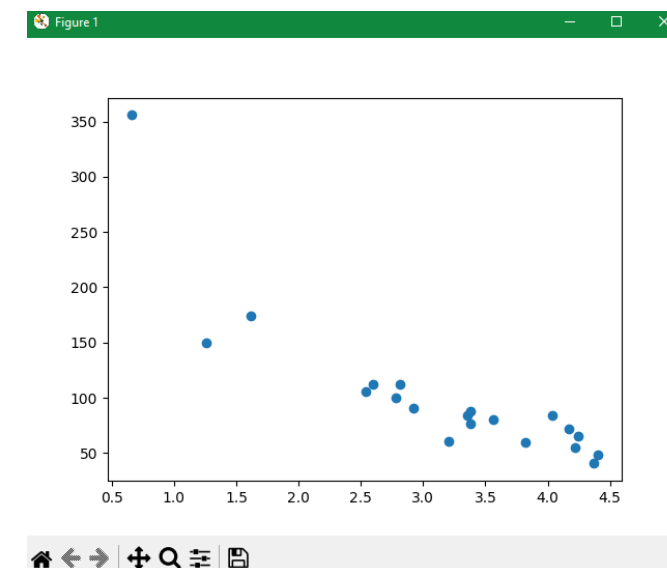
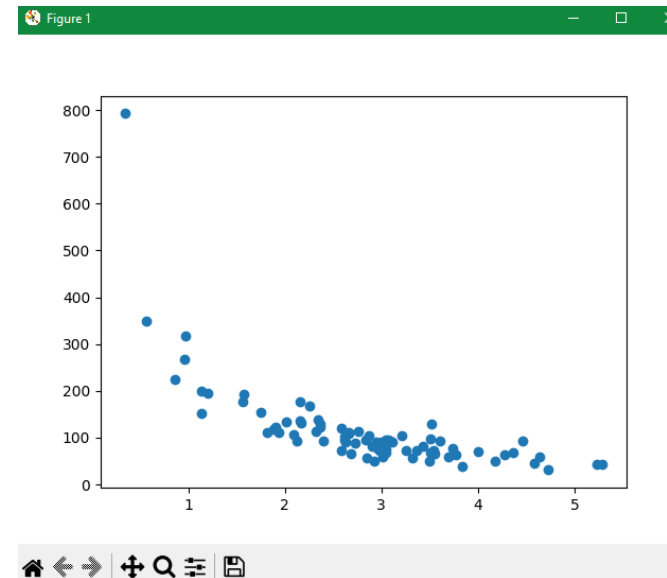
```
trainTest.py > ...  
1 import numpy  
2 import matplotlib.pyplot as plt  
3 from sklearn.metrics import r2_score  
4 numpy.random.seed(2)  
5  
6 # ile cm rośnie drzewo/rok  
7 x = numpy.random.normal(3, 1, 100)  
8 # wiek drzewa w latach  
9 y = numpy.random.normal(250, 50, 100) / x  
10  
11 plt.scatter(x, y)  
12 plt.show()  
13
```



Uwaga! Dane są losowe i nie mają nic wspólnego z rzeczywistością!

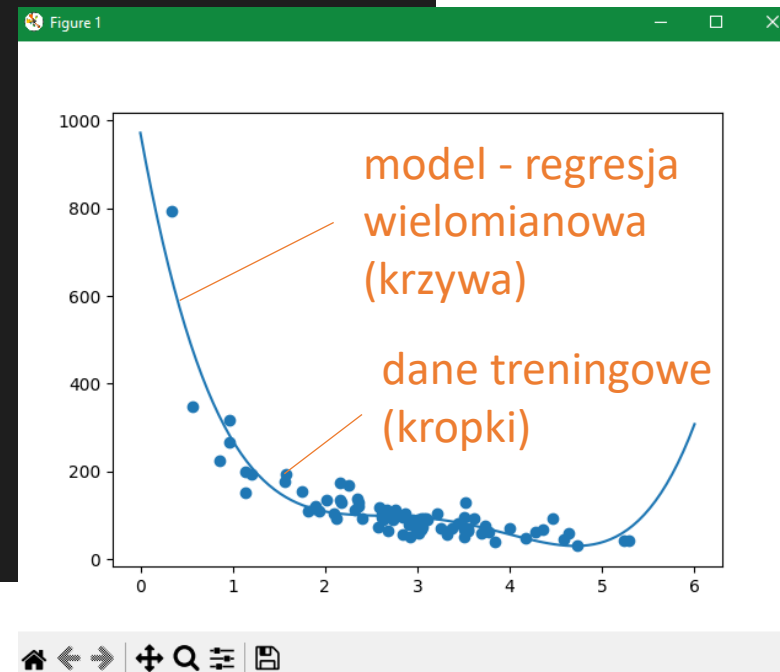
Train/Test

```
14 # tworzymy zbiór treningowy (uczący),
15 # który stanowi losowy wybór 80% oryginalnych danych
16 train_x = x[:80]
17 train_y = y[:80]
18
19 # wykres punktowy danych treningowych
20 plt.scatter(train_x, train_y)
21 plt.show()
22
23 # tworzymy zbiór testowy z pozostałych 20% danych
24 test_x = x[80:]
25 test_y = y[80:]
26
27 # wykres punktowy danych testowych
28 plt.scatter(test_x, test_y)
29 plt.show()
30
```



Train/Test

```
31 # tworzymy model - do aproksymacji danych używamy regresji wielomianowej
32 mymodel = numpy.poly1d(numpy.polyfit(train_x, train_y, 4))
33
34 # zakładamy, że x zmienia się od 0 do 5
35 myline = numpy.linspace(0, 6, 100)
36
37 # wykres punktowy oraz modelu (krzywej aproksymującej dane)
38 plt.scatter(train_x, train_y)
39 plt.plot(myline, mymodel(myline))
40 plt.show()
41 # sprawdzamy poprawność modelu za pomocą
42 # współczynnika R-Kwadrat
43 # dane uczące
44 r2Train = r2_score(train_y, mymodel(train_x))
45 print(r2Train)
46
47 # dane testowe
48 r2Test = r2_score(test_y, mymodel(test_x))
49 print(r2Test)
```



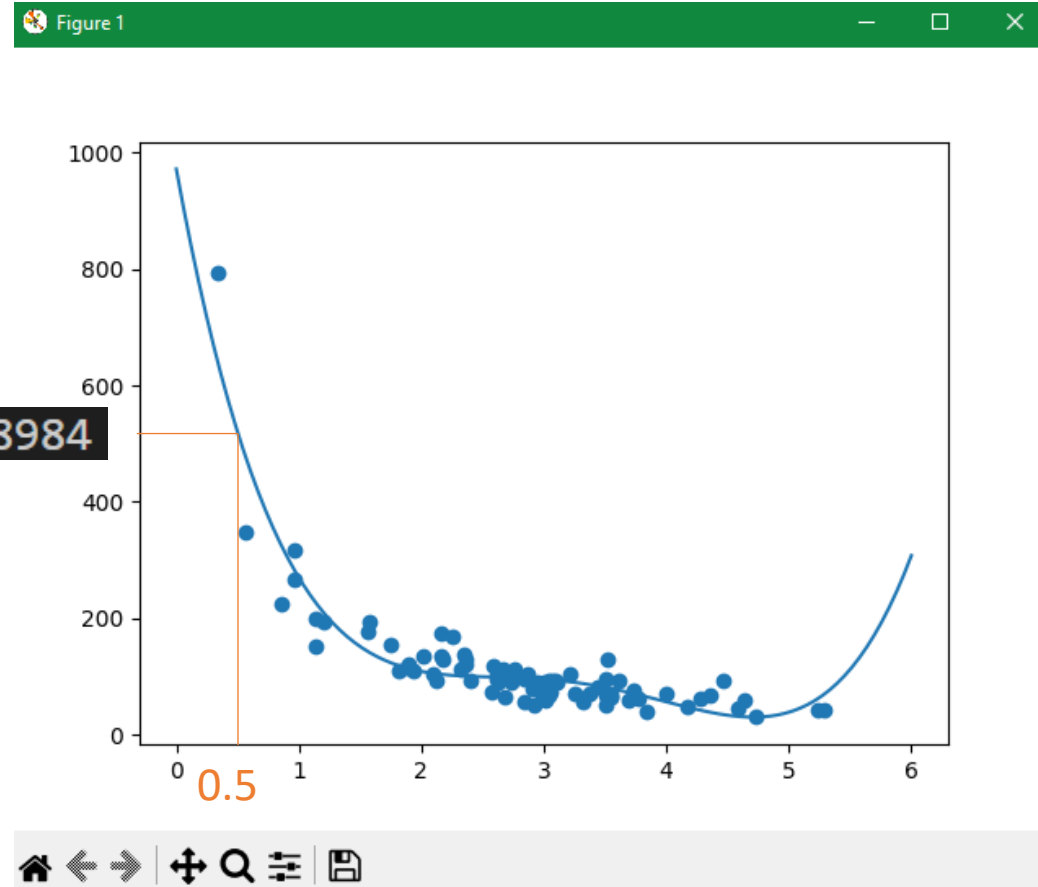
R2 (R-squared) zarówno danych testowych jak i treningowych jest bliski jedności, czyli regresja wielomianowa jest dobrze dopasowana – model jest OK

Train/Test

```
50  
51 print(mymodel(0.5))
```

520.1426166298984

ponieważ model dobrze odwzorowuje dane,
można przewidzieć, że dla drzewa mającego ok.
520 lat jego wzrost wyniesie 0.5 cm/rok



Train/Test

Zadanie

Podziel dane na część treningowa oraz testową, zbuduj model korzystając z aproksymacji liniowej lub wielomianowej, określ stopień odwzorowania modelu za pomocą współczynnika korelacji r lub R -Kwadrat oraz (jeśli model dobrze odwzorowuje dane) wylicz nową, nieznaną wcześniej wartość.

Titanic

The screenshot shows the Kaggle website interface for the 'Titanic - Machine Learning from Disaster' competition. The browser address bar displays 'https://www.kaggle.com/competitions/titanic/data'. The left sidebar contains navigation options like 'Create', 'Home', 'Competitions', 'Datasets', 'Models', 'Code', 'Discussions', 'Learn', and 'More'. The main content area features a search bar, a competition banner with the title 'Titanic - Machine Learning from Disaster', and a navigation menu with tabs for 'Overview', 'Data', 'Code', 'Discussion', 'Leaderboard', and 'Rules'. A 'Submit Predictions' button is visible in the top right of the main area. Below the navigation, the 'Dataset Description' section is displayed, including an 'Overview' with a list of files (train.csv and test.csv), a detailed explanation of the training and test sets, and a note about the 'gender_submission.csv' file. On the right side of the description, a summary table lists the dataset's characteristics.

Files
3 files
Size
93.08 kB
Type
CSV

dane dotyczące pasażerów Titanica można pobrać ze strony [Kaggle](https://www.kaggle.com/competitions/titanic/data) (po zalogowaniu)

Image Source: [Kaggle](https://www.kaggle.com/competitions/titanic/data)

Titanic

opis danych dotyczących pasażerów Titanica

Data Dictionary

Variable	Definition	Key
survival	Survival	0 = No, 1 = Yes
pclass	Ticket class	1 = 1st, 2 = 2nd, 3 = 3rd
sex	Sex	
Age	Age in years	
sibsp	# of siblings / spouses aboard the Titanic	
parch	# of parents / children aboard the Titanic	
ticket	Ticket number	
fare	Passenger fare	
cabin	Cabin number	
embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton

Variable Notes

pclass: A proxy for socio-economic status (SES)

1st = Upper

2nd = Middle

3rd = Lower

age: Age is fractional if less than 1. If the age is estimated, is it in the form of xx.5

sibsp: The dataset defines family relations in this way...

Sibling = brother, sister, stepbrother, stepsister

Spouse = husband, wife (mistresses and fiancés were ignored)

parch: The dataset defines family relations in this way...

Parent = mother, father

Child = daughter, son, stepdaughter, stepson

Some children travelled only with a nanny, therefore parch=0 for them.


Titanic

```
titanic > titanic.py > ...
50
51 # import danych
52 df = pd.read_csv("titanicData/train.csv")
53 #print(df)
54
55 # printujemy klucze
56 keys = df.keys()
57 #print(keys)
58 #['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
59 #   'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
60
61 # zmieniamy wartości typu string w kolumnie płeć 'Sex' na wartości typu integer
62 # tworzymy lista zmiennych niezależnych
63 X = transformSexFromCategoricalToNumerical()
64
65 # usuwamy puste wartości w kolumnie 'Age'
66 X = removeNanFromAge(X)
67
68 # zmienna zależna, którą wyliczamy
69 # (czy przeżył)
70 y = df['Survived']
71
72 # tworzymy obiekt liniowej regresji
73 regr = linear_model.LinearRegression()
74
75 # przekazujemy listy zależnych zmiennych i zmienną niezależną do obiektu regresji liniowej
76 regr.fit(X.values, y)
77
78 # przewidujemy, czy dla podanych parametrów osoba przeżyłaby na Titanicu:
79 # klasa biletu - 1,2,3
80 # wiek - np.20
81 # kobieta - 0 lub 1
82 # mężczyzna - 0 lub 1
83 predicted = regr.predict([[1, 20, 1, 0]])
84
85 print(predicted) # [0.9875129]
```

program
analizujący czy
pasażer przeżył
w zależności od
płci, wieku oraz
klasy biletu.
Program
korzysta z
Multiple
Regression.

przy założeniu, że
pasażerem byłaby
kobieta w wieku 20
lat w klasie 1,
miałaby duże szanse
przeżycia

Titanic

```
titanic >  titanic.py > ...
1 import pandas as pd
2 from sklearn import linear_model
3 import numpy as np
4
5 # wypełnia puste wartości (NaN) w kolumnie 'Age'
6 # wartościami randomowymi w zakresie
7 # <średnia - odchylenie standardowe, średnia + odchylenie standardowe>
8 def removeNanFromAge(X):
9     # w kolumnie 'Age' są puste wartości NaN w ilości:
10    ageNanCount = X['Age'].isnull().sum() # 177
11
12    # średnia
13    ageMean = df['Age'].mean()
14    # odchylenie standardowe
15    ageStd = df['Age'].std()
16
17    # tworzymy tablicę o wielkości ageNanCount z wartościami randomowymi w zakresie
18    # <średnia - odchylenie standardowe, średnia + odchylenie standardowe>
19    randomTab = np.random.randint(ageMean - ageStd, ageMean + ageStd, size=ageNanCount)
20
21    # wymieniamy wartości NaN w kolumnie 'Age' na te z tablicy randomTab
22    X.loc[X['Age'].isnull(), 'Age'] = randomTab
23
24    # sprawdzamy, czy zostały wartości NaN w kolumnie 'Age'
25    # print("Pozostałe wartości NaN: {}".format(X['Age'].isnull().sum()))
26
27    # zwracamy X
28    return X
```

dane
pasażerów
Titanica nie są
kompletne, np.
kolumna Age
ma puste
wartości, które
wypełniamy
wartościami
randomowymi

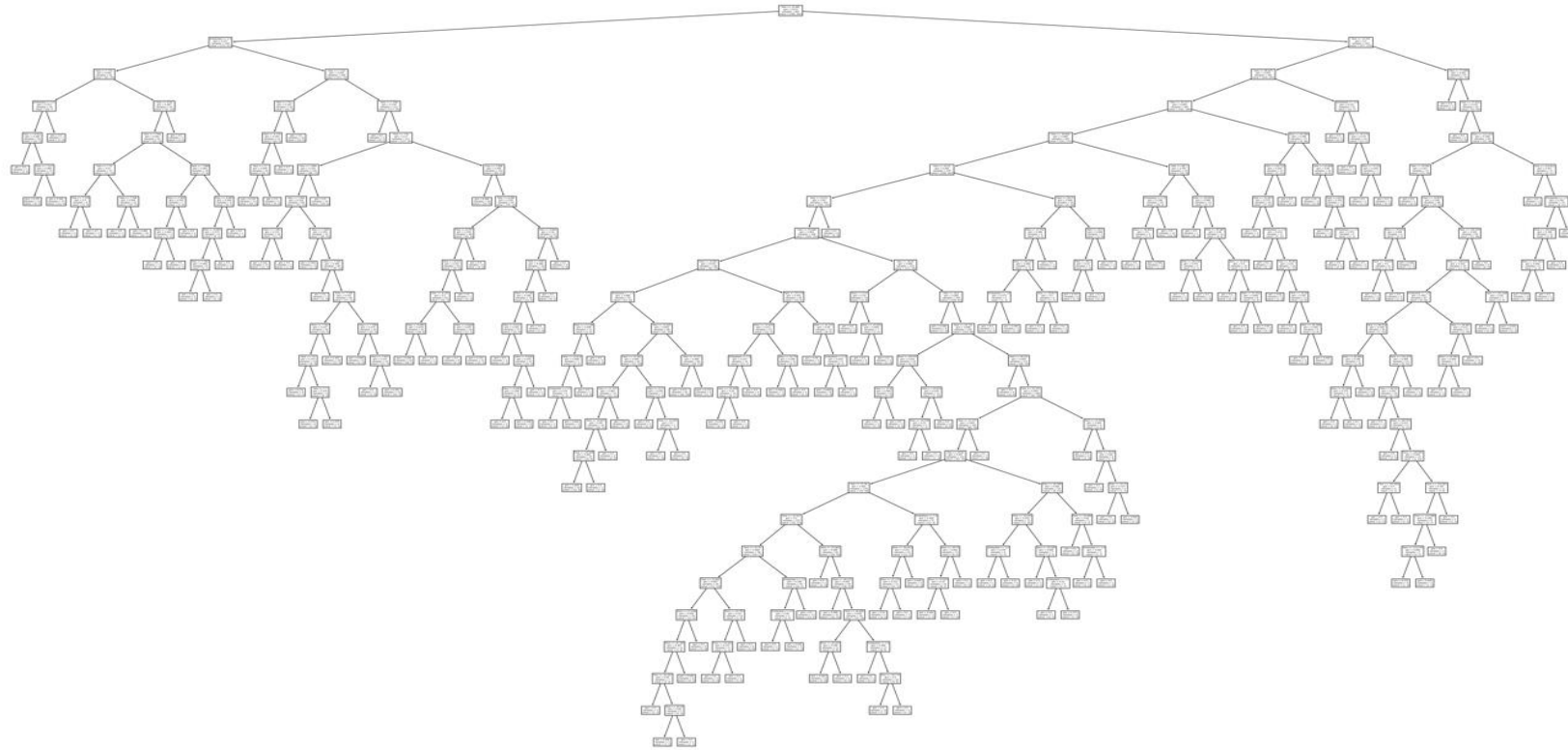
Titanic

```
29
30 # zmiana wartości typu string w kolumnie płeć 'Sex' na wartości typu integer
31 # za pomocą metody One Hot Encoding
32 def transformSexFromCategoricalToNumerical():
33     # kolumna 'Sex' ma wartości typu string: 'female', 'male'
34     # zamieniamy ją na dwie kolumny Sex_female, Sex_male z wartościami 1 oraz 0
35     # Sex      => Sex_female      Sex_male
36     # 'female'      1              0
37     # 'male'        0              1
38     sex_female_male = pd.get_dummies(df[['Sex']], dtype=int)
39
40     # print(sex_female_male.to_string())
41
42     # tworzymy listę niezależnych zmiennych 'Pclass', 'Age'
43     # oraz Sex_female, Sex_male
44     X = pd.concat([df[['Pclass', 'Age']], sex_female_male], axis=1)
45
46     #print(X)
47
48     return X
49
50
```

zamiana danych
typu string dane
typu numerycznego
w kolumnie płeć

[Categorical Data](#)

Decision Tree



Drzewo decyzyjne to schemat blokowy, który może pomóc w podejmowaniu decyzji na podstawie wcześniejszych doświadczeń. Stosowane jest w uczeniu maszynowym do pozyskiwania wiedzy na podstawie danych.

https://pl.wikipedia.org/wiki/Drzewo_decyzyjne

https://www.w3schools.com/python/python_ml_decision_tree.asp

Decision Tree

tworzymy drzewo decyzyjne
na podstawie danych
pasażerów z Titanica

zakładamy, że zmiennymi
niezależnymi są: 'Fare'
(opłata), 'Embarked'
(miejsce zaokrętowania),
a zmienną zależną
'Survived' (czy pasażer
przeżył)

```
titanic > titanicDecisionTree.py > ...
1 import pandas as pd
2 from sklearn import tree
3 from sklearn.tree import DecisionTreeClassifier
4 import matplotlib.pyplot as plt
5 import numpy as np
6
7
8 # import danych
9 df = pd.read_csv("titanicData/train.csv")
10 #print(df)
11
12 # zamiana stringów w kolumnie 'Embarked' - port zaokrętowania - na wartości liczbowe
13 # C = Cherbourg, Q = Queenstown, S = Southampton
14 d = {'C': 0, 'Q': 1, 'S': 2}
15 df['Embarked'] = df['Embarked'].map(d)
16 # print(df['Embarked'])
17
18 # tworzymy listę niezależnych zmiennych 'Fare', 'Embarked'
19 X = df[['Fare', 'Embarked']]
20 # print(X)
21
22 # zmienna zależna, którą wyliczamy
23 # (czy przeżył)
24 y = df['Survived']
25
26 embarkedNanCount = X['Embarked'].isnull().sum() # 2
27
28 # print(embarkedNanCount)
29
30 # wymieniamy wartości NaN w kolumnie 'Embarked' na te z tablicy [0,1] - zahardkodowane wartości
31 X.loc[X['Embarked'].isnull(), 'Embarked'] = [0,1]
32
33 # sprawdzamy czy zostały NaN
34 # print(X['Embarked'].isnull().sum())
35
36 # printujemy Decision Tree do pliku tree.png
37 features = ['Fare', 'Embarked']
38
39 dtree = DecisionTreeClassifier()
40 dtree = dtree.fit(X, y)
41
42 plt.figure(figsize=(50,30))
43
44 tree.plot_tree(dtree, feature_names=features)
45
46 plt.savefig('tree.png', dpi=300)
47
48 plt.clf()
```

mapowanie
zaokrętowania:
0 = Cherbourg,
1 = Queenstown,
2 = Southampton

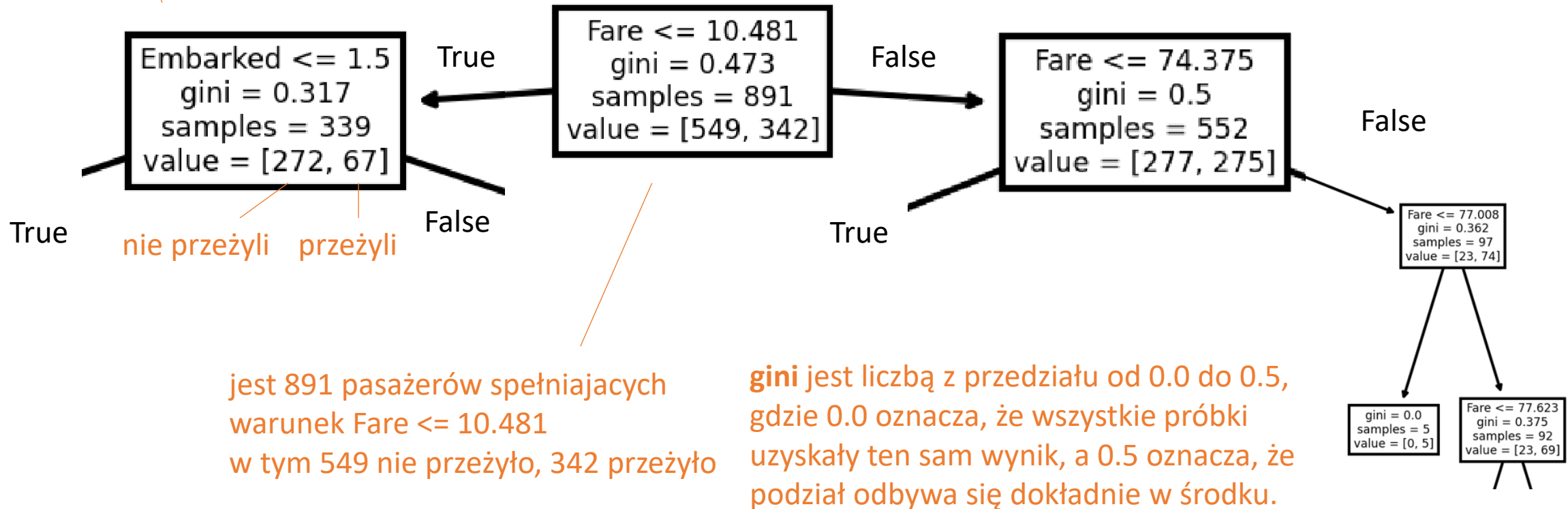
Decision Tree

pasażerowie, którzy zaokrętownali się w 0 (Cherbourg) lub 1 (Queenstown) są rozpatrywani dalej po lewej (True)

pasażerowie, którzy płacili poniżej 10.481 są rozpatrywani po lewej (True), reszta po prawej (False)

obliczenie współczynnika gini

$$\begin{aligned} \text{gini} &= 1 - (x/n)^2 - (y/n)^2 = \\ &= 1 - (549/891)^2 - (342/891)^2 = \\ &= 1 - 0.379 - 0.147 = 0.473 \end{aligned}$$



Decision Tree

po stworzeniu drzewa decyzyjnego
można na jego podstawie przewidzieć
przeżycie pasażera o założonych
parametrach

```
51  
52 # przewidujemy, czy ktoś przeżyje w zależności od zapłaconej  
53 # sumy za bilet oraz portu zaokrętowania  
54 print(dtree.predict([[80000, 1]]))  
55 print("[1] - przeżył")  
56 print("[0] - nie przeżył")
```

```
[1]  
[1] - przeżył  
[0] - nie przeżył
```

osoba płacąca 80000 oraz zaokrętowana
w Queenstown najprawdopodobniej by
przeżyła.

Titanic

Zadanie

Przeanalizuj dane pasażerów Titanica według własnego pomysłu.

Analiza danych pasażerów Titanica w Internecie:

https://ttsteiger.github.io/projects/titanic_report.html#Feature-description

<https://betterprogramming.pub/titanic-survival-prediction-using-machine-learning-4c5ff1e3fa16>

https://medium.com/@melodyyip_/titanic-survival-prediction-using-machine-learning-89a779656113

<https://www.analyticsvidhya.com/blog/2021/05/titanic-survivors-a-guide-for-your-first-data-science-project/>

Confusion Matrix

tabela z wizualną oceną dotyczącą błędów w modelu.

rzeczywisty stan pacjentów

numer pacjenta

stan zdrowia:

1: chory

0: zdrowy

1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	0	0	0

przewidywany stan pacjentów
(przykład działania
hipotetycznego modelu))

numer pacjenta

stan zdrowia:

1: chory

0: zdrowy

1	2	3	4	5	6	7	8	9	10
1	1	0	0	1	1	1	1	0	0
TP	TP	FN	FN	TP	TP	TP	FP	TN	TN

TP: True Positive

TN: True Negative

FP: False Positive

FN: False Negative

Confusion Matrix

		Predicted condition	
		Positive (PP)	Negative (PN)
Actual condition	Total population = P + N		
	Positive (P)	True positive (TP)	False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

tabela z wizualną oceną dotyczącą błędów w modelu
(Confusion Matrix wg. [wikipedii](https://en.wikipedia.org/wiki/Confusion_matrix))

		Przewidywany stan pacjenta	
		Chory	Zdrowy
Rzeczywisty stan pacjenta	Wszyscy 7+3=10	6	4
	Chory 7	5	2
	Zdrowy 3	1	2

wartości prawdziwe

wartości błędne

https://en.wikipedia.org/wiki/Confusion_matrix

Confusion Matrix

Wskaźniki wydajności

dokładność modelu

Accuracy = $(\text{True Positive} + \text{True Negative}) / \text{Total Predictions}$

jakość pozytywnych prognoz

Precision = $\text{True Positive} / (\text{True Positive} + \text{False Positive})$

na ile model jest dokładny w przewidywaniu pozytywnych prognoz TP

Sensitivity (Recall) = $\text{True Positive} / (\text{True Positive} + \text{False Negative})$

na ile model jest dokładny w przewidywaniu negatywnych prognoz

Specificity = $\text{True Negative} / (\text{True Negative} + \text{False Positive})$

średnia harmoniczna Precision oraz Sensitivity

F-score = $2 * ((\text{Precision} * \text{Sensitivity}) / (\text{Precision} + \text{Sensitivity}))$

Confusion Matrix

```
confusionMatrix.py > ...
1 import matplotlib.pyplot as plt
2 from sklearn import metrics
3
4 actual = [1,1,1,1,1,1,1,0,0,0]
5 predicted = [1,1,0,0,1,1,1,1,0,0]
6
7 confusion_matrix = metrics.confusion_matrix(actual, predicted)
8
9 cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])
10
11 cm_display.plot()
12 plt.show()
13
14 # Wskaźniki wydajności
15 Accuracy = metrics.accuracy_score(actual, predicted)
16 Precision = metrics.precision_score(actual, predicted)
17 Sensitivity_recall = metrics.recall_score(actual, predicted)
18 Specificity = metrics.recall_score(actual, predicted, pos_label=0)
19 F1_score = metrics.f1_score(actual, predicted)
20
21 print("\nAccuracy: " + str(Accuracy) +
22       "\nPrecision: " + str(Precision) +
23       "\nSensitivity_recall: " + str(Sensitivity_recall) +
24       "\nSpecificity: " + str(Specificity) +
25       "\nF1_score: " + str(F1_score))
```

lista z rzeczywistym stanem zdrowia pacjentów

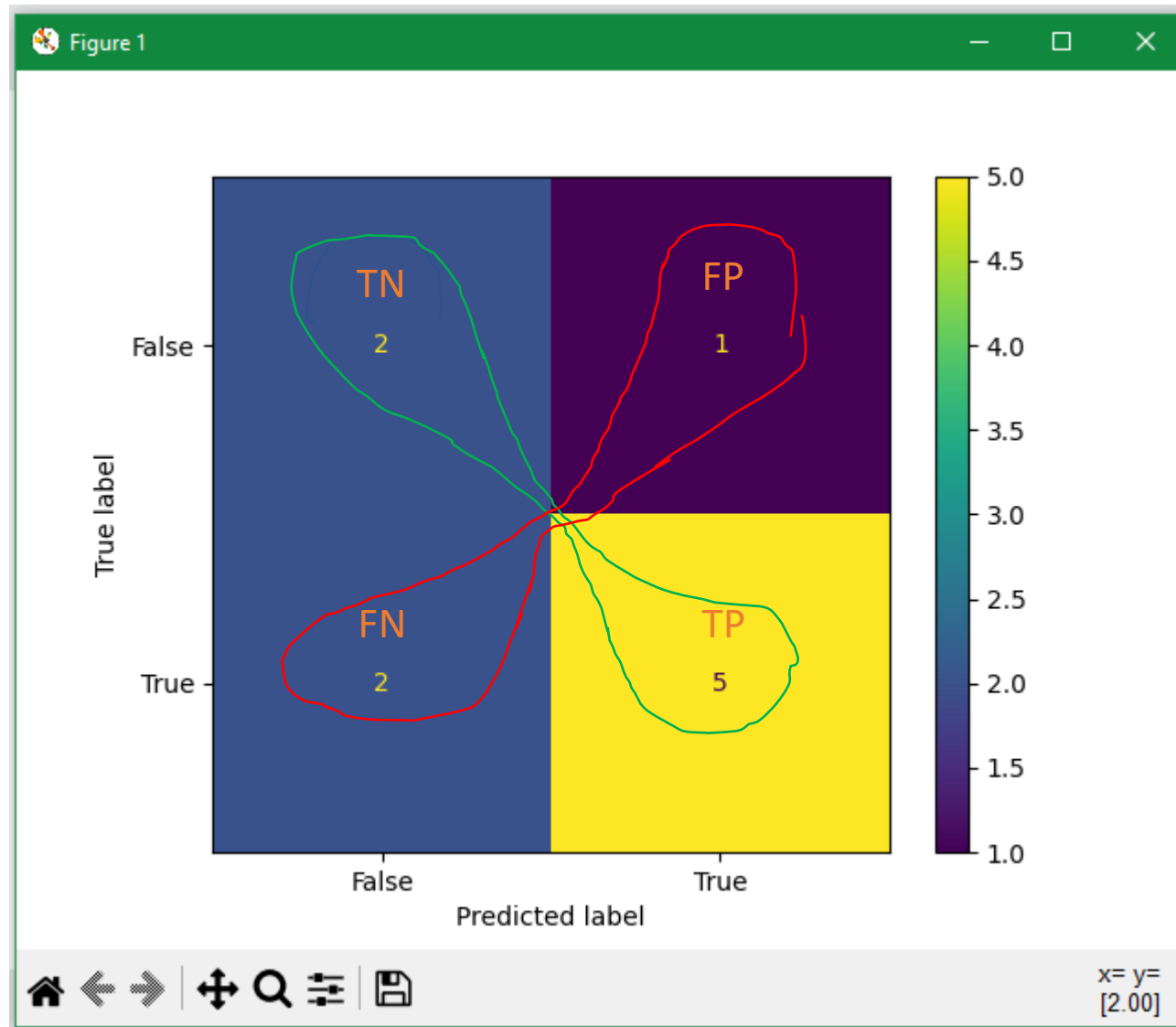
lista z przewidywanym stanem zdrowia pacjentów

wskaźniki wydajności

```
Accuracy: 0.7
Precision: 0.8333333333333334
Sensitivity_recall: 0.7142857142857143
Specificity: 0.6666666666666666
F1_score: 0.7692307692307692
```

Confusion Matrix

tabela z wizualną oceną dotyczącą błędów w modelu (Confusion Matrix wg. [w3schools](https://www.w3schools.com))



Confusion Matrix

Zadanie

Oblicz Confusion Matrix oraz wskaźniki wydajności dla wybranych danych oraz zastosowanego modelu.

ML js

ML tutorial js







The screenshot shows a web browser window with the URL <https://www.w3schools.com/ai/default.asp>. The page title is "Machine Learning". The navigation bar includes links for HTML, CSS, JAVASCRIPT, SQL, PYTHON, JAVA, PHP, BOOTSTRAP, HOW TO, W3.CSS, C, C++, C#, REACT, R, JQUERY, DJANGO, and TYPESCRIPT. The sidebar menu is titled "Machine Learning" and includes sub-items like "ML Intro", "ML and AI", "ML Examples", "ML Languages", "ML in JavaScript", "ML in the Browser", "Mathematics", "Statistics", "Graphics", and "Data Science". The main content area features a large green header with the title "Machine Learning and Artificial Intelligence". Below the header, it states: "This **Tutorial** covers the basics of: **Artificial Intelligence** and **Machine Learning**. Using **JavaScript** examples, solutions and formulas." The "Image Classification Examples" section displays a photograph of a robin perched on a wooden post. Below the image are two buttons: a red button labeled "Click to Classify Picture" and a green button labeled "Click to Change Picture".

ML js

różne zastosowania ML

Build your first on-device ML app

The learning pathways below provide a step-by-step guide to help you write your first on-device machine learning app.

 Audio Classification Write an app that can classify sounds in the environment around you. In this example, identify birds based on their song. Get started	 Image Classification Build an app that takes a picture and gives you a list of labels that describe the image. Train a model to recognize newer labels and integrate it in your app. Get started	 Object Detection Detect specific objects within an image and draw bounding boxes around them. Train a model to identify new objects and integrate the model in your app. Get started
 Text Classification for Mobile Create an app that determines if your users are spamming your chatroom. Get started	 Text Classification for Web Use Machine Learning in your web site to help filter comment spam. Get started	 Visual Product Search Take a picture with your camera and search for matching products. Get started

ML js

ON-DEVICE MACHINE LEARNING

Text Classification for Web

Learn how to create and use a machine learning model that detects comment spam in the browser using TensorFlow.js



Get started with comment spam detection

Learn the fundamentals of making custom text classification models for your web apps using TensorFlow.js.

✓ 4 activities

Explore



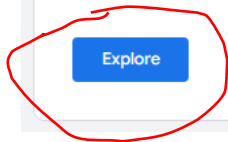
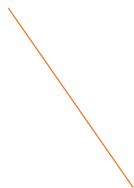
Go further with comment spam detection - model retraining

Learn how to retrain your comment spam model to account for edge cases it may miss when using the default pre-made model.

✓ 3 activities

Explore

detektor spamu w js



Czym jest uczenie maszynowe (machine learning)?

źródło: www.w3schools.com/