Warsztaty Data Science

Gdańsk | 07.02.2019

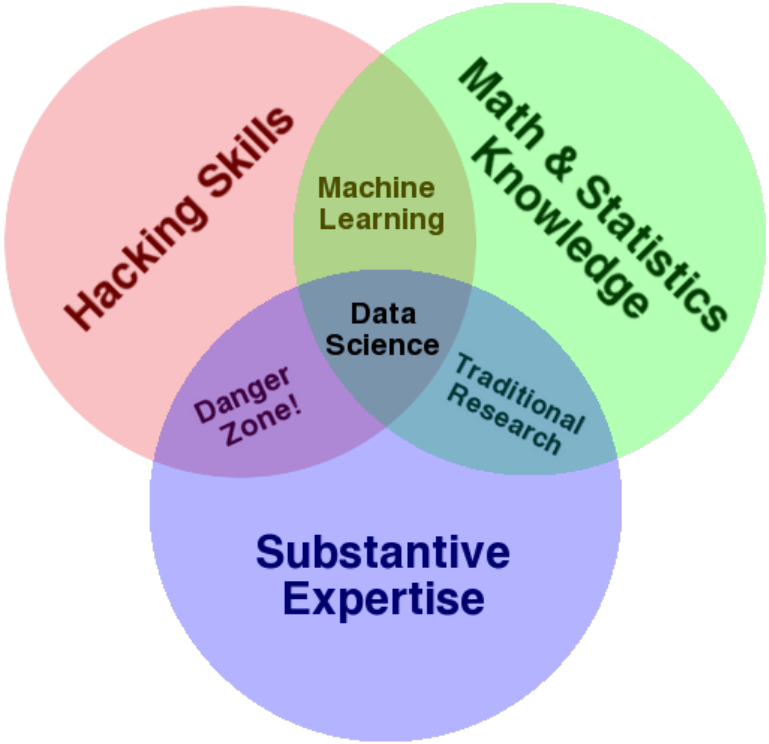
Plan warsztatu

1. Kim jest Data Scientist?
2. Co robi i jakich narzędzi używa?
3. Jak wygląda proces analizy danych?
4. Mały wstęp do Pythona.
5. Mały wstęp do statystyki.
6. Poznajmy dane, na których będziemy pracować.
7. Wspólne kodowanie.
8. Q & A.

Kim jest Data Scientist?

Data Science Venn Diagram

Autor: Drew Convey



Główne role Data Scientista

Analiza i doradzanie

Praca w firmach produkcyjnych lub usługowych, np. sieci drogerii. Analiza danych o zachowaniach klientów albo warunkach produkcji, przekazywanie rekomendacji dot. decyzji strategicznych, np. kiedy i jaką promocję wprowadzić, żeby była najskuteczniejsza. Wizualizacja danych w sposób zrozumiały dla osób biznesowych.

Budowanie produktu

Praca w firmach produktowych i badawczych, np. aplikacja automatycznie wykrywająca dni płodne. Data Scientist przygotowuje i rozwija algorytm Machine Learningowy, który jest sercem produktu. Dokonuje analiz, żeby sprawdzić jak dobrze działa algorytm, jak bardzo użytkowniczki są zadowolone, szuka możliwości polepszenia algorytmu.

Schemat pracy z problememem i danymi

Zrozumienie tematyki problemu

Postawienie hipotezy

Analiza danych

Czyszczenie danych

Eksperymenty

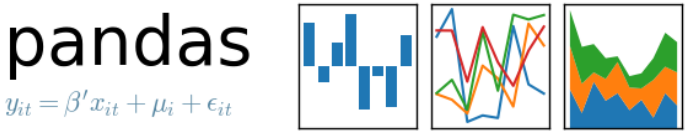
Raport końcowy

Projekt Data Science

https://en.wikipedia.org/wiki/Cross-industry_standard_process_for_data_mining

Popularne narzędzia dla Data Scientist

- Języki: **Python**, SQL
- Środowisko: **Jupyter Notebook**, Jupyter Lab
- Biblioteka do manipulacji zbiorami danych: **Pandas**
- Biblioteki obliczeniowe: Numpy, SciPy
- Biblioteki Machine Learningowe: **scikit-learn**, xgboost
- Biblioteki Deep Learningowe: Tensorflow, PyTorch
- Biblioteki do wizualizacji: **matplotlib**, **Seaborn**



Dystrybucja Anaconda Python

https://www.anaconda.com/distribution

Windows | macOS | Linux

Anaconda 2018.12 for macOS Installer

Python 3.7 version

Download

64-Bit Graphical Installer (652.7 MB)

64-Bit Command Line Installer (557 MB)

Python 2.7 version

Download

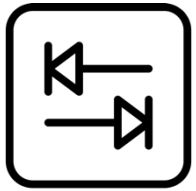
64-Bit Graphical Installer (640.7 MB)

64-Bit Command Line Installer (547 MB)

Mały wstęp do Pythona

Ogólne zasady kodowania w Pythonie

- Jedno polecenie to jedna linijka.
- Ważne są kropki, przecinki, spacje i wcięcia (taby).
- Wielkość liter ma znaczenie.
- Dobra praktyka - wszystko (zmienne, funkcje itp.) nazywajmy po angielsku!



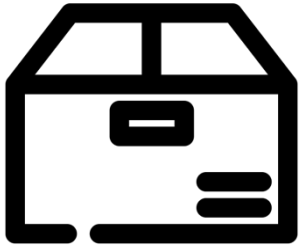
Zmienna i przypisanie do zmiennej

Ogólna zasada:

```
nazwa_zmiennej = “Coś do przypisania”
```

Przykład:

```
greeting = “Cześć wszystkim!”
```



Jak zobaczyć zawartość zmiennej

Korzystamy z funkcji `print`:

```
print(greeting)
>> Cześć wszystkim!
```

```
In [7]: greeting = "Cześć wszystkim!"
        print(greeting)

        Cześć wszystkim!
```

Jak zrobić notatki?

Komentarze, czyli polecenia, które nie będą interpretowane przez Pythona, oznaczamy przez wstawienie # na początku linijki. # działa tylko do końca linijki, w następnej musimy dać kolejny.

```
In [43]: # To jest komentarz, piszę go tylko dla siebie, Python się tym nie przejmie.
         # print("Hello")
         # I nic się nie pojawiło.
```

Przypisania czytamy od prawej strony

Przypiszmy sobie parę liczb:

```
working_days = 5
weekend = 2
```

To jest obiekt
typu Integer - liczba całkowita

I je dodajmy:

```
week = working_days + weekend
print(week)
>> 7
```

Czytanie od prawej daje ciekawe możliwości

```
print(week)
>> 7
week = week + 1
print(week)
>> 8
```

Obroty Ziemi zwolniły

Wywołanie funkcji

Nazwa funkcji

Argumenty, wejścia do funkcji

```
print(greeting)
```

Nazwa biblioteki

```
pd.read_csv("data/2017.csv")
```

```
data_2017.info()
```

Nazwa zmiennej z obiektem

Pandas DataFrame

Prawie jak plik Excela, tylko z większymi możliwościami :)

	Country	Happiness.Score	Economy..GDP.per.Capita.	Family	Health..Life.Expectancy.	Freedom	Generosity	Trust..Government.Corruption.
0	Norway	7.537	1.616463	1.533524	0.796667	0.635423	0.362012	0.315964
1	Denmark	7.522	1.482383	1.551122	0.792566	0.626007	0.355280	0.400770
2	Iceland	7.504	1.480633	1.610574	0.833552	0.627163	0.475540	0.153527
3	Switzerland	7.494	1.564980	1.516912	0.858131	0.620071	0.290549	0.367007
4	Finland	7.469	1.443572	1.540247	0.809158	0.617951	0.245483	0.382612
5	Netherlands	7.377	1.503945	1.428939	0.810696	0.585384	0.470490	0.282662
6	Canada	7.316	1.479204	1.481349	0.834558	0.611101	0.435540	0.287372
7	New Zealand	7.314	1.405706	1.548195	0.816760	0.614062	0.500005	0.382817
8	Sweden	7.284	1.494387	1.478162	0.830875	0.612924	0.385399	0.384399
9	Australia	7.284	1.484415	1.510042	0.843887	0.601607	0.477699	0.301184
10	Israel	7.213	1.375382	1.376290	0.838404	0.405989	0.330083	0.085242
11	Costa Rica	7.079	1.109706	1.416404	0.759509	0.580132	0.214613	0.100107
12	Austria	7.006	1.487097	1.459945	0.815328	0.567766	0.316472	0.221060
13	United States	6.993	1.546259	1.419921	0.774287	0.505741	0.392579	0.135639
14	Ireland	6.977	1.535707	1.558231	0.809783	0.573110	0.427858	0.298388

Mały wstęp do statystyki

Średnia

Wzór matematyczny: $$\bar{x} = \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right) = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

Wzór matematyczny mówi nam dokładnie **jak obliczyć** daną miarę. Nie mówi nam jednak o tym jak ta miara się zachowuje.

Średnia

Przykład:
Liczymy średnią płac miesięcznych 6 pracowników w firmie.
5 szeregowych pracowników i szef.

$$\bar{x} = \frac{2100 + 2700 + 2350 + 3010 + 2245 + 15140}{6} = 4590.83$$

Średnia jest miarą podatną na **wartości skrajne**.

Mediana

Wartość środkowa. Należy posortować wszystkie elementy i wybrać środkowy. Jeśli liczba elementów jest parzysta, to należy wziąć średnią z dwóch środkowych elementów.

$$\mathrm{median}(a) = a_{\lceil \#x \div 2 \rceil}$$

$$\mathrm{median}(a) = \frac{a_{\lceil \#x \div 2 \rceil} + a_{\lceil \#x \div 2 + 1 \rceil}}{2}$$

Mediana

Przykład:
Liczymy medianę płac miesięcznych 6 pracowników w firmie.
5 szeregowych pracowników i szef.

2100, 2245, <u>2350</u>, <u>2700</u>, 3010, 15140

$$\text{median}(a) = \frac{2350 + 2700}{2} = 2525$$

Mediana jest miarą odporną na **wartości skrajne**.

Średnia ważona

Wzór matematyczny:

$$\bar{x} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i} = \frac{w_1 x_1 + w_2 x_2 + \cdots + w_n x_n}{w_1 + w_2 + \cdots + w_n}$$

Średnią ważoną stosujemy, kiedy liczymy średnią różnych składników, które mają **różną ważność.**

Średnia ważona

Przykład:
Liczymy średnią ważoną ocen z algebry liniowej.
Wejściówki mają wagę - **1**, prace domowe - **2**, kolokwium - **5**.

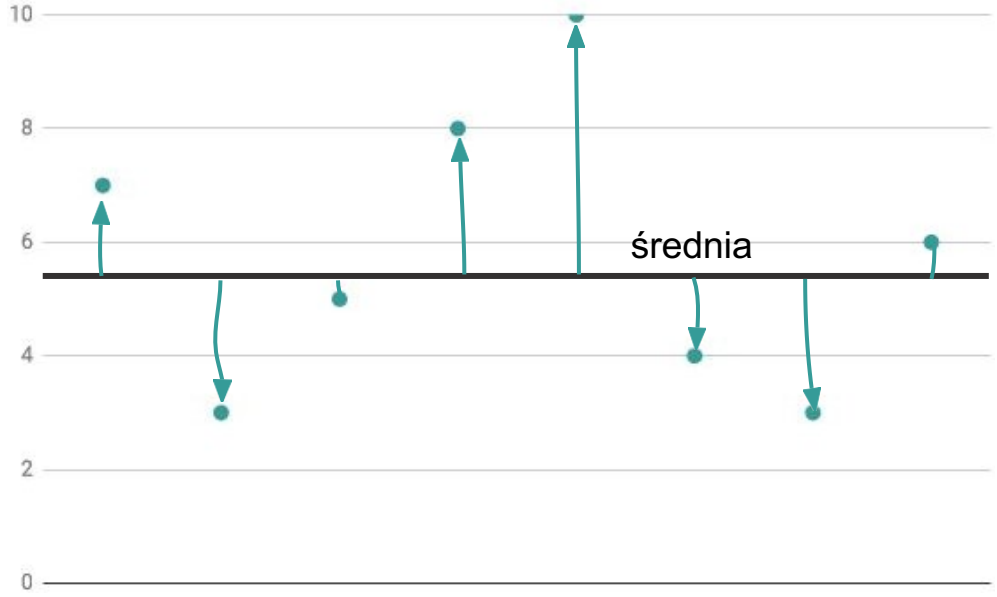
$$\bar{x} = \frac{1 \times 2 + 1 \times 3 + 1 \times 2.5 + 2 \times 5 + 2 \times 5 + 5 \times 4.5}{1 + 1 + 1 + 2 + 2 + 5} = 4.16$$

Uwzględniane wartości powinny być w **tej samej skali**. W tym przypadku wszystkie są w skali 2 - 5.

Wariancja

Miara zmienności, zróżnicowania w populacji, średnia arytmetyczna kwadratów odchyleń.

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2$$

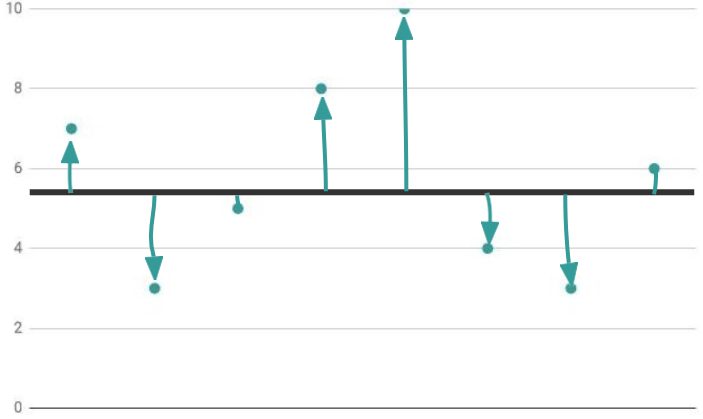


Wariancja

Przykład:
Zapytaliśmy 6 osób o ich oceny nowego sezonu serialu Black Mirror, w skali od 1 do 10.

Ich oceny: 7, 3, 5, 8, 10, 4, 3, 6

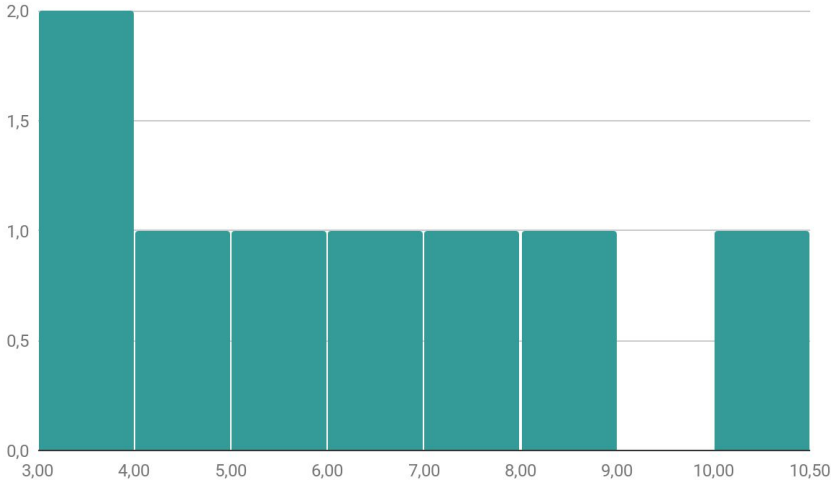
Średnia: 5.75
Odchylenie standardowe: 2.49284691

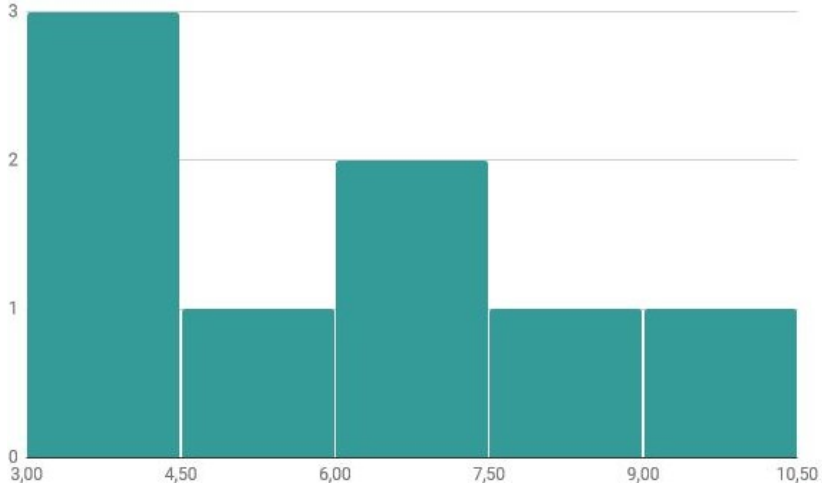


Histogram

Graficzna reprezentacja częstości występowania wartości.

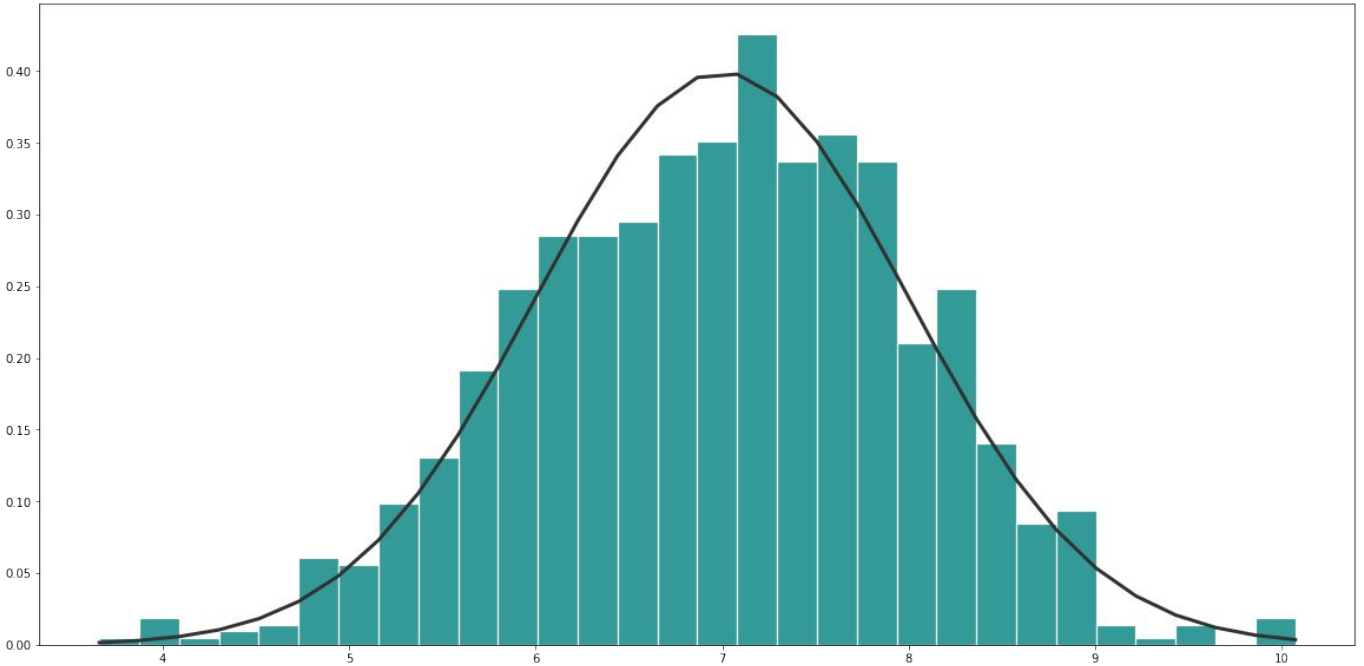
Oceny respondentów: 7, 3, 5, 8, 10, 4, 3, 6



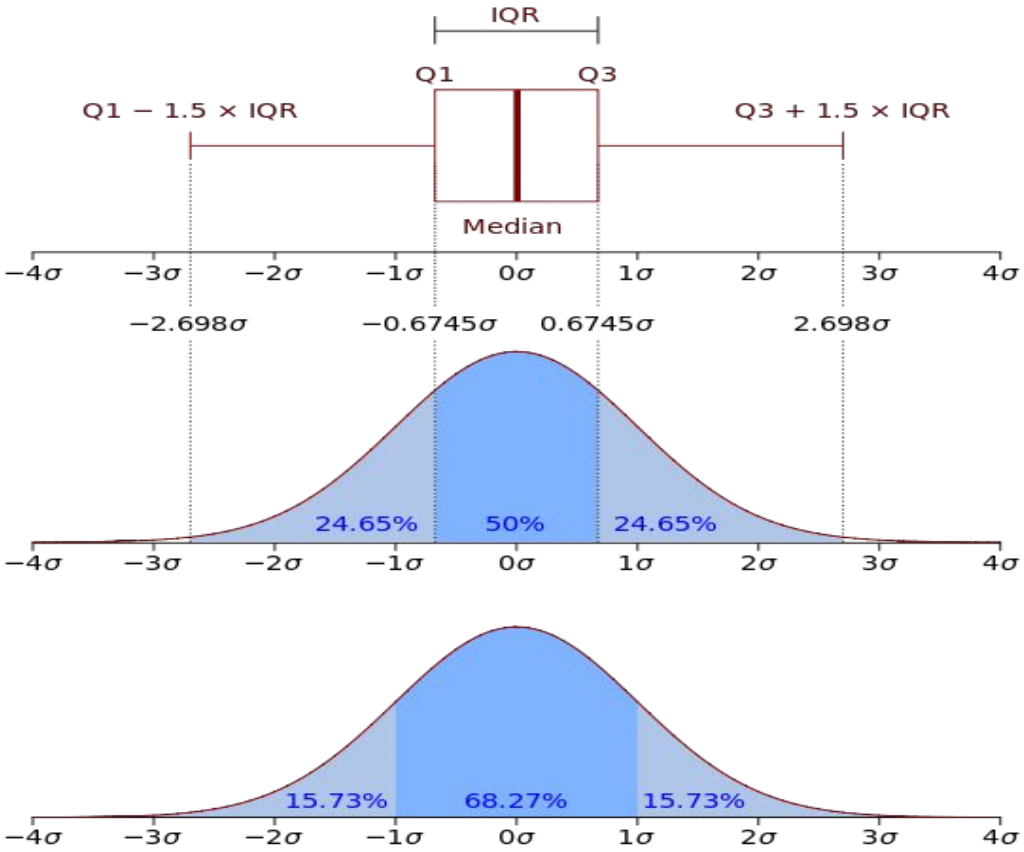


Histogram, a rozkład prawodpodobieństwa

Tym razem przepytaliśmy 1000 osób, pozwoliliśmy na oceny niecałkowite.



Rozkład prawodpodobieństwa i kwartyle

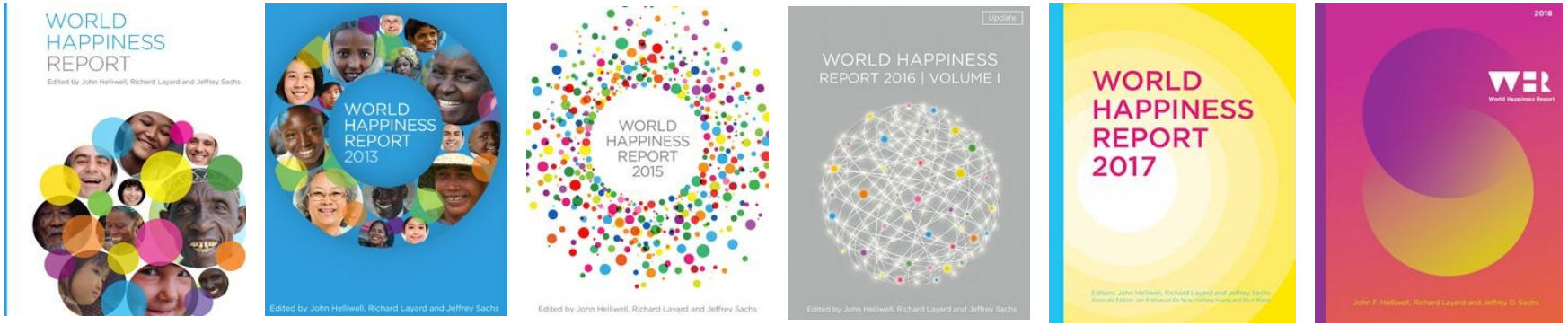


https://en.wikipedia.org/wiki/Quartile

World Happiness Report

World Happiness Report

Raport publikowany co roku przez Organizację Narodów Zjednoczonych. Pierwszy raport powstał w 2012 roku. Zawiera podsumowania odpowiedzi mieszkańców 155 państw świata na pytania związane z własnym dobrobytem, zadane w Gallup World Poll.



World Happiness Report

Czynniki uwzględnione w raporcie:

- Happiness Score,
- Economy (GDP per Capita),
- Family,
- Health (Life Expectancy),
- Freedom,
- Trust (Government Corruption),
- Generosity.

Please imagine a ladder with steps numbered from zero at the bottom to ten at the top. Suppose we say that the top of the ladder represents the best possible life for you and the bottom of the ladder represents the worst possible life for you.

If the top step is 10 and the bottom step is 0, on which step of the ladder do you feel you personally stand at the present time?

Python For Data Science *Cheat Sheet*

Python Basics

Learn More Python for Data Science Interactively at www.datacamp.com

Variables and Data Types

Variable Assignment

```
>>> x=5
>>> x
 5
```

Calculations With Variables

```
>>> x+2
 7
>>> x-2
 3
>>> x*2
 10
>>> x**2
 25
>>> x%2
 1
>>> x/float(2)
 2.5
```

Sum of two variables
Subtraction of two variables
Multiplication of two variables
Exponentiation of a variable
Remainder of a variable
Division of a variable

Types and Type Conversion

`str()`	`'5', '3.45', 'True'`	Variables to strings
`int()`	`5, 3, 1`	Variables to integers
`float()`	`5.0, 1.0`	Variables to floats
`bool()`	`True, True, True`	Variables to booleans

Asking For Help

```
>>> help(str)
```

Lists

Also see NumPy Arrays

```
>>> a = 'is'
>>> b = 'nice'
>>> my_list = ['my', 'list', a, b]
>>> my_list2 = [[4,5,6,7], [3,4,5,6]]
```

Selecting List Elements

Index starts at 0

Subset

```
>>> my_list[1]
>>> my_list[-3]
```

Select item at index 1
Select 3rd last item

Slice

```
>>> my_list[1:3]
>>> my_list[1:]
>>> my_list[:3]
>>> my_list[:]
```

Select items at index 1 and 2
Select items after index 0
Select items before index 3
Copy my_list

Subset Lists of Lists

```
>>> my_list2[1][0]
>>> my_list2[1][:2]
```

my_list[list][itemOfList]

List Operations

```
>>> my_list + my_list
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']
>>> my_list * 2
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']
>>> my_list2 > 4
True
```

List Methods

`>>> my_list.index(a)`	Get the index of an item
`>>> my_list.count(a)`	Count an item
`>>> my_list.append('!')`	Append an item at a time
`>>> my_list.remove('!')`	Remove an item
`>>> del(my_list[0:1])`	Remove an item
`>>> my_list.reverse()`	Reverse the list
`>>> my_list.extend('!')`	Append an item
`>>> my_list.pop(-1)`	Remove an item
`>>> my_list.insert(0,'!')`	Insert an item
`>>> my_list.sort()`	Sort the list

Strings

```
>>> my_string = 'thisStringIsAwesome'
>>> my_string
'thisStringIsAwesome'
```

String Operations

```
>>> my_string * 2
 'thisStringIsAwesomethisStringIsAwesome'
>>> my_string + 'Innit'
 'thisStringIsAwesomeInnit'
>>> 'm' in my_string
 True
```

String Operations

Index starts at 0

```
>>> my_string[3]
>>> my_string[4:9]
```

String Methods

`>>> my_string.upper()`	String to uppercase
`>>> my_string.lower()`	String to lowercase
`>>> my_string.count('w')`	Count String elements
`>>> my_string.replace('e', 'i')`	Replace String elements
`>>> my_string.strip()`	Strip whitespaces

Libraries

Import libraries

```
>>> import numpy
>>> import numpy as np
```

Selective import

```
>>> from math import pi
```

pandas — Data analysis
learn — Machine learning
NumPy — Scientific computing
matplotlib — 2D plotting

Install Python

ANACONDA — Leading open data science platform powered by Python

spyder — Free IDE that is included with Anaconda

jupyter — Create and share documents with live code, visualizations, text, ...

Numpy Arrays

Also see Lists

```
>>> my_list = [1, 2, 3, 4]
>>> my_array = np.array(my_list)
>>> my_2darray = np.array([[1,2,3],[4,5,6]])
```

Selecting Numpy Array Elements

Index starts at 0

Subset

```
>>> my_array[1]
  2
```

Select item at index 1

Slice

```
>>> my_array[0:2]
  array([1, 2])
```

Select items at index 0 and 1

Subset 2D Numpy arrays

```
>>> my_2darray[:,0]
  array([1, 4])
```

my_2darray[rows, columns]

Numpy Array Operations

```
>>> my_array > 3
  array([False, False, False,  True], dtype=bool)
>>> my_array * 2
  array([2, 4, 6, 8])
>>> my_array + np.array([5, 6, 7, 8])
  array([6, 8, 10, 12])
```

Numpy Array Functions

`>>> my_array.shape`	Get the dimensions of the array
`>>> np.append(other_array)`	Append items to an array
`>>> np.insert(my_array, 1, 5)`	Insert items in an array
`>>> np.delete(my_array,[1])`	Delete items in an array
`>>> np.mean(my_array)`	Mean of the array
`>>> np.median(my_array)`	Median of the array
`>>> my_array.corrcoef()`	Correlation coefficient
`>>> np.std(my_array)`	Standard deviation

Python For Data Science *Cheat Sheet*

Pandas Basics

Learn Python for Data Science Interactively at www.DataCamp.com

Pandas

The **Pandas** library is built on NumPy and provides easy-to-use **data structures** and **data analysis** tools for the Python programming language.

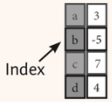
Use the following import convention:

```
>>> import pandas as pd
```

Pandas Data Structures

Series

A **one-dimensional** labeled array capable of holding any data type




```
>>> s = pd.Series([3, -5, 7, 4], index=['a', 'b', 'c', 'd'])
```

DataFrame

Columns

	Country	Capital	Population
0	Belgium	Brussels	11190846
1	India	New Delhi	1303171035
2	Brazil	Brasília	207847528

Index

A **two-dimensional** labeled data structure with columns of potentially different types

```
>>> data = {'Country': ['Belgium', 'India', 'Brazil'],
            'Capital': ['Brussels', 'New Delhi', 'Brasília'],
            'Population': [11190846, 1303171035, 207847528]}

>>> df = pd.DataFrame(data,
                      columns=['Country', 'Capital', 'Population'])
```

Asking For Help

```
>>> help(pd.Series.loc)
```

Selection

Also see NumPy Arrays

Getting

```
>>> s['b']
 -5

>>> df[1:]
   Country    Capital  Population
1   India  New Delhi  1303171035
2  Brazil   Brasilia   207847528
```

Get one element

Get subset of a DataFrame

Selecting, Boolean Indexing & Setting

By Position

```
>>> df.iloc[[0],[0]]
'Belgium'
>>> df.iat([0],[0])
'Belgium'
```

Select single value by row & column

By Label

```
>>> df.loc[[0], ['Country']]
'Belgium'
>>> df.at([0], ['Country'])
'Belgium'
```

Select single value by row & column labels

By Label/Position

```
>>> df.ix[2]
Country      Brazil
Capital    Brasilia
Population  207847528
```

Select single row of subset of rows

```
>>> df.ix[:,'Capital']
0     Brussels
1    New Delhi
2     Brasilia
```

Select a single column of subset of columns

```
>>> df.ix[1,'Capital']
'New Delhi'
```

Select rows and columns

Boolean Indexing

```
>>> s[~(s > 1)]
>>> s[(s < -1) | (s > 2)]
>>> df[df['Population']>1200000000]
```

Series `s` where value is not >1
`s` where value is <-1 or >2
Use filter to adjust DataFrame

Setting

```
>>> s['a'] = 6
```

Set index `a` of Series `s` to 6

Dropping

```
>>> s.drop(['a', 'c'])
>>> df.drop('Country', axis=1)
```

Drop values from rows (axis=0)
Drop values from columns(axis=1)

Sort & Rank

```
>>> df.sort_index()
>>> df.sort_values(by='Country')
>>> df.rank()
```

Sort by labels along an axis
Sort by the values along an axis
Assign ranks to entries

Retrieving Series/DataFrame Information

Basic Information

```
>>> df.shape
>>> df.index
>>> df.columns
>>> df.info()
>>> df.count()
```

(rows,columns)
Describe index
Describe DataFrame columns
Info on DataFrame
Number of non-NA values

Summary

```
>>> df.sum()
>>> df.cumsum()
>>> df.min()/df.max()
>>> df.idxmin()/df.idxmax()
>>> df.describe()
>>> df.mean()
>>> df.median()
```

Sum of values
Cummulative sum of values
Minimum/maximum values
Minimum/Maximum index value
Summary statistics
Mean of values
Median of values

Applying Functions

```
>>> f = lambda x: x*2
>>> df.apply(f)
>>> df.applymap(f)
```

Apply function
Apply function element-wise

Data Alignment

Internal Data Alignment

NA values are introduced in the indices that don't overlap:

```
>>> s3 = pd.Series([7, -2, 3], index=['a', 'c', 'd'])
>>> s + s3
  a    10.0
  b     NaN
  c     5.0
  d     7.0
```

Arithmetic Operations with Fill Methods

You can also do the internal data alignment yourself with the help of the fill methods:

```
>>> s.add(s3, fill_value=0)
  a    10.0
  b    -5.0
  c     5.0
  d     7.0
>>> s.sub(s3, fill_value=2)
>>> s.div(s3, fill_value=4)
>>> s.mul(s3, fill_value=3)
```

I/O

Read and Write to CSV

```
>>> pd.read_csv('file.csv', header=None, nrows=5)
>>> df.to_csv('myDataFrame.csv')
```

Read and Write to Excel

```
>>> pd.read_excel('file.xlsx')
>>> pd.to_excel('dir/myDataFrame.xlsx', sheet_name='Sheet1')
```

Read multiple sheets from the same file

```
>>> xlsx = pd.ExcelFile('file.xls')
>>> df = pd.read_excel(xlsx, 'Sheet1')
```

Read and Write to SQL Query or Database Table

```
>>> from sqlalchemy import create_engine
>>> engine = create_engine('sqlite:///:memory:')
>>> pd.read_sql("SELECT * FROM my_table;", engine)
>>> pd.read_sql_table('my_table', engine)
>>> pd.read_sql_query("SELECT * FROM my_table;", engine)
```

`read_sql()` is a convenience wrapper around `read_sql_table()` and `read_sql_query()`

```
>>> pd.to_sql('myDf', engine)
```

Data Wrangling with pandas Cheat Sheet

http://pandas.pydata.org

Tidy Data – A foundation for wrangling in pandas

In a tidy data set:

Each **variable** is saved in its own **column**

&

Each **observation** is saved in its own **row**

Tidy data complements pandas's vectorized operations. pandas will automatically preserve observations as you manipulate variables. No other format works as intuitively with pandas.

M * A

Syntax – Creating DataFrames

	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

```
df = pd.DataFrame(
        {"a" : [4 ,5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
       index = [1, 2, 3])
```

Specify values for each column.

```
df = pd.DataFrame(
    [[4, 7, 10],
     [5, 8, 11],
     [6, 9, 12]],
    index=[1, 2, 3],
    columns=['a', 'b', 'c'])
```

Specify values for each row.

		a	b	c
n	v			
d	1	4	7	10
	2	5	8	11
e	2	6	9	12

```
df = pd.DataFrame(
        {"a" : [4 ,5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
index = pd.MultiIndex.from_tuples(
        [('d',1),('d',2),('e',2)],
          names=['n','v']))
```

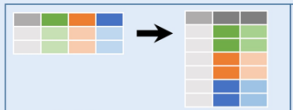
Create DataFrame with a MultiIndex

Method Chaining

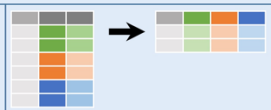
Most pandas methods return a DataFrame so that another pandas method can be applied to the result. This improves readability of code.

```
df = (pd.melt(df)
        .rename(columns={
                'variable' : 'var',
                'value' : 'val'})
        .query('val >= 200')
     )
```

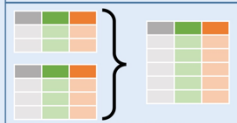
Reshaping Data – Change the layout of a data set



pd.melt(df)
Gather columns into rows.



df.pivot(columns='var', values='val')
Spread rows into columns.



pd.concat([df1,df2])
Append rows of DataFrames

pd.concat([df1,df2], axis=1)
Append columns of DataFrames

df.sort_values('mpg')
Order rows by values of a column (low to high).

df.sort_values('mpg',ascending=False)
Order rows by values of a column (high to low).

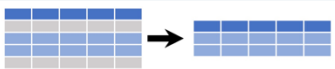
df.rename(columns = {'y':'year'})
Rename the columns of a DataFrame

df.sort_index()
Sort the index of a DataFrame

df.reset_index()
Reset index of DataFrame to row numbers, moving index to columns.

df.drop(columns=['Length','Height'])
Drop columns from DataFrame

Subset Observations (Rows)



df[df.Length > 7]
Extract rows that meet logical criteria.

df.drop_duplicates()
Remove duplicate rows (only considers columns).

df.head(n)
Select first n rows.

df.tail(n)
Select last n rows.

df.sample(frac=0.5)
Randomly select fraction of rows.

df.sample(n=10)
Randomly select n rows.

df.iloc[10:20]
Select rows by position.

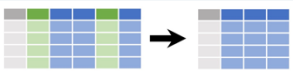
df.nlargest(n, 'value')
Select and order top n entries.

df.nsmallest(n, 'value')
Select and order bottom n entries.

Logic in Python (and pandas)

<	Less than	!=	Not equal to
>	Greater than	df.column.isin(*values*)	Group membership
==	Equals	pd.isnull(*obj*)	Is NaN
<=	Less than or equals	pd.notnull(*obj*)	Is not NaN
>=	Greater than or equals	&,|,~,^,df.any(),df.all()	Logical and, or, not, xor, any, all

Subset Variables (Columns)



df[['width','length','species']]
Select multiple columns with specific names.

df['width'] *or* **df.width**
Select single column with specific name.

df.filter(regex='*regex*')
Select columns whose name matches regular expression *regex*.

regex (Regular Expressions) Examples

'\.'	Matches strings containing a period '.'
'Length$'	Matches strings ending with word 'Length'
'^Sepal'	Matches strings beginning with the word 'Sepal'
'^x[1-5]$'	Matches strings beginning with 'x' and ending with 1,2,3,4,5
''^(?!Species$).*'	Matches strings except the string 'Species'

df.loc[:,'x2':'x4']
Select all columns between x2 and x4 (inclusive).

df.iloc[:,[1,2,5]]
Select columns in positions 1, 2 and 5 (first column is 0).

df.loc[df['a'] > 10, ['a','c']]
Select rows meeting logical condition, and only the specific columns .

Summarize Data

`df['w'].value_counts()`
Count number of rows with each unique value of variable

`len(df)`
# of rows in DataFrame.

`df['w'].nunique()`
# of distinct values in a column.

`df.describe()`
Basic descriptive statistics for each column (or GroupBy)

pandas provides a large set of **summary functions** that operate on different kinds of pandas objects (DataFrame columns, Series, GroupBy, Expanding and Rolling (see below)) and produce single values for each of the groups. When applied to a DataFrame, the result is returned as a pandas Series for each column. Examples:

`sum()`
Sum values of each object.

`count()`
Count non-NA/null values of each object.

`median()`
Median value of each object.

`quantile([0.25,0.75])`
Quantiles of each object.

`apply(function)`
Apply function to each object.

`min()`
Minimum value in each object.

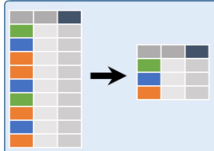
`max()`
Maximum value in each object.

`mean()`
Mean value of each object.

`var()`
Variance of each object.

`std()`
Standard deviation of each object.

Group Data



`df.groupby(by="col")`
Return a GroupBy object, grouped by values in column named "col".

`df.groupby(level="ind")`
Return a GroupBy object, grouped by values in index level named "ind".

All of the summary functions listed above can be applied to a group. Additional GroupBy functions:

`size()`
Size of each group.

`agg(function)`
Aggregate group using function.

The examples below can also be applied to groups. In this case, the function is applied on a per-group basis, and the returned vectors are of the length of the original DataFrame.

`shift(1)`
Copy with values shifted by 1.

`rank(method='dense')`
Ranks with no gaps.

`rank(method='min')`
Ranks. Ties get min rank.

`rank(pct=True)`
Ranks rescaled to interval [0, 1].

`rank(method='first')`
Ranks. Ties go to first value.

`shift(-1)`
Copy with values lagged by 1.

`cumsum()`
Cumulative sum.

`cummax()`
Cumulative max.

`cummin()`
Cumulative min.

`cumprod()`
Cumulative product.

Windows

`df.expanding()`
Return an Expanding object allowing summary functions to be applied cumulatively.

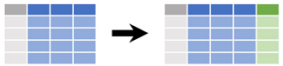
`df.rolling(n)`
Return a Rolling object allowing summary functions to be applied to windows of length n.

Handling Missing Data

`df.dropna()`
Drop rows with any column having NA/null data.

`df.fillna(value)`
Replace all NA/null data with value.

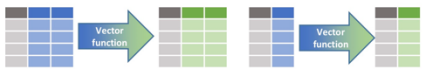
Make New Columns



`df.assign(Area=lambda df: df.Length*df.Height)`
Compute and append one or more new columns.

`df['Volume'] = df.Length*df.Height*df.Depth`
Add single column.

`pd.qcut(df.col, n, labels=False)`
Bin column into n buckets.



pandas provides a large set of **vector functions** that operate on all columns of a DataFrame or a single selected column (a pandas Series). These functions produce vectors of values for each of the columns, or a single Series for the individual Series. Examples:

`max(axis=1)`
Element-wise max.

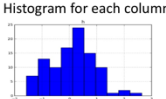
`clip(lower=-10,upper=10)`
Trim values at input thresholds

`min(axis=1)`
Element-wise min.

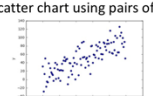
`abs()`
Absolute value.

Plotting

`df.plot.hist()`
Histogram for each column



`df.plot.scatter(x='w',y='h')`
Scatter chart using pairs of points



Combine Data Sets

adf

x1	x2
A	1
B	2
C	3

+

bdf

x1	x3
A	T
B	F
D	T

=

Standard Joins

x1	x2	x3
A	1	T
B	2	F
C	3	NaN

`pd.merge(adf, bdf, how='left', on='x1')`
Join matching rows from bdf to adf.

x1	x2	x3
A	1.0	T
B	2.0	F
D	NaN	T

`pd.merge(adf, bdf, how='right', on='x1')`
Join matching rows from adf to bdf.

x1	x2	x3
A	1	T
B	2	F

`pd.merge(adf, bdf, how='inner', on='x1')`
Join data. Retain only rows in both sets.

x1	x2	x3
A	1	T
B	2	F
C	3	NaN
D	NaN	T

`pd.merge(adf, bdf, how='outer', on='x1')`
Join data. Retain all values, all rows.

Filtering Joins

x1	x2
A	1
B	2

`adf[adf.x1.isin(bdf.x1)]`
All rows in adf that have a match in bdf.

x1	x2
C	3

`adf[~adf.x1.isin(bdf.x1)]`
All rows in adf that do not have a match in bdf.

ydf

x1	x2
A	1
B	2
C	3

+

zdf

x1	x2
B	2
C	3
D	4

=

Set-like Operations

x1	x2
B	2
C	3

`pd.merge(ydf, zdf)`
Rows that appear in both ydf and zdf (Intersection).

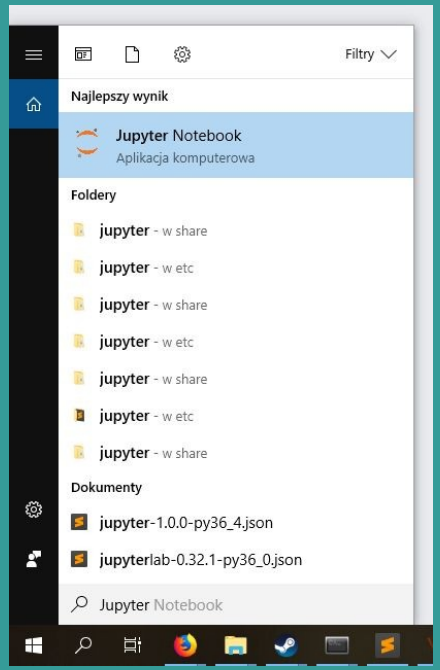
x1	x2
A	1
B	2
C	3
D	4

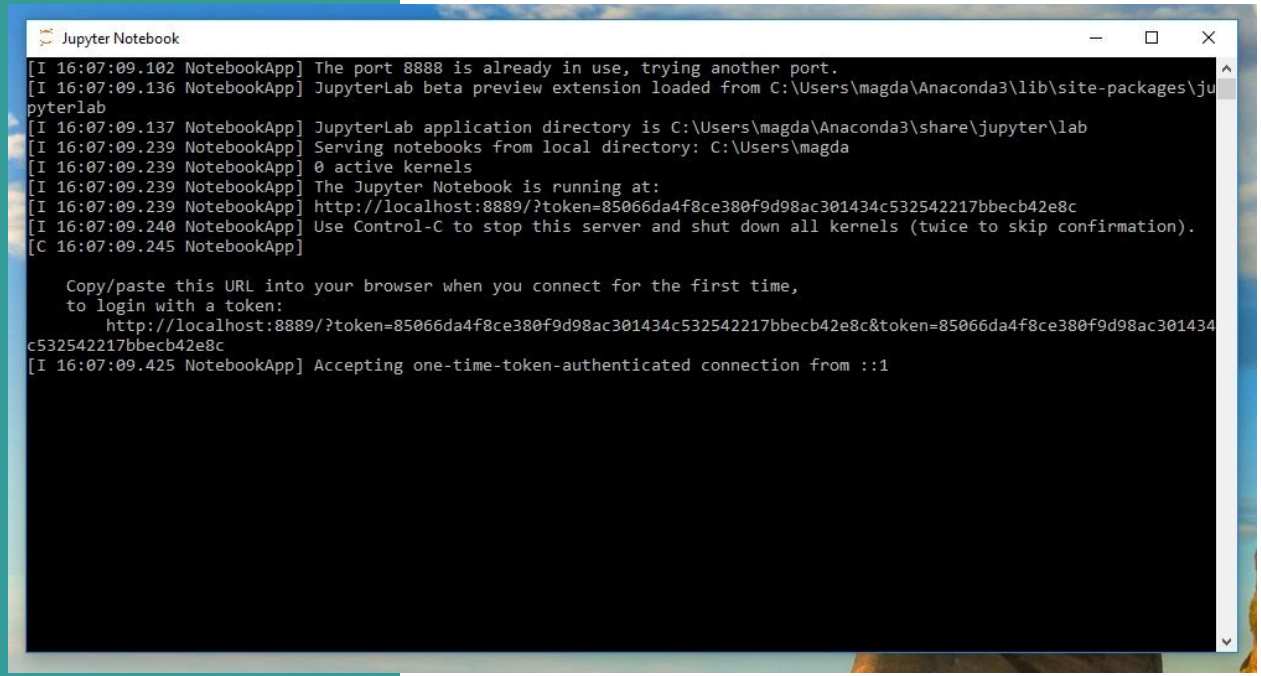
`pd.merge(ydf, zdf, how='outer')`
Rows that appear in either or both ydf and zdf (Union).

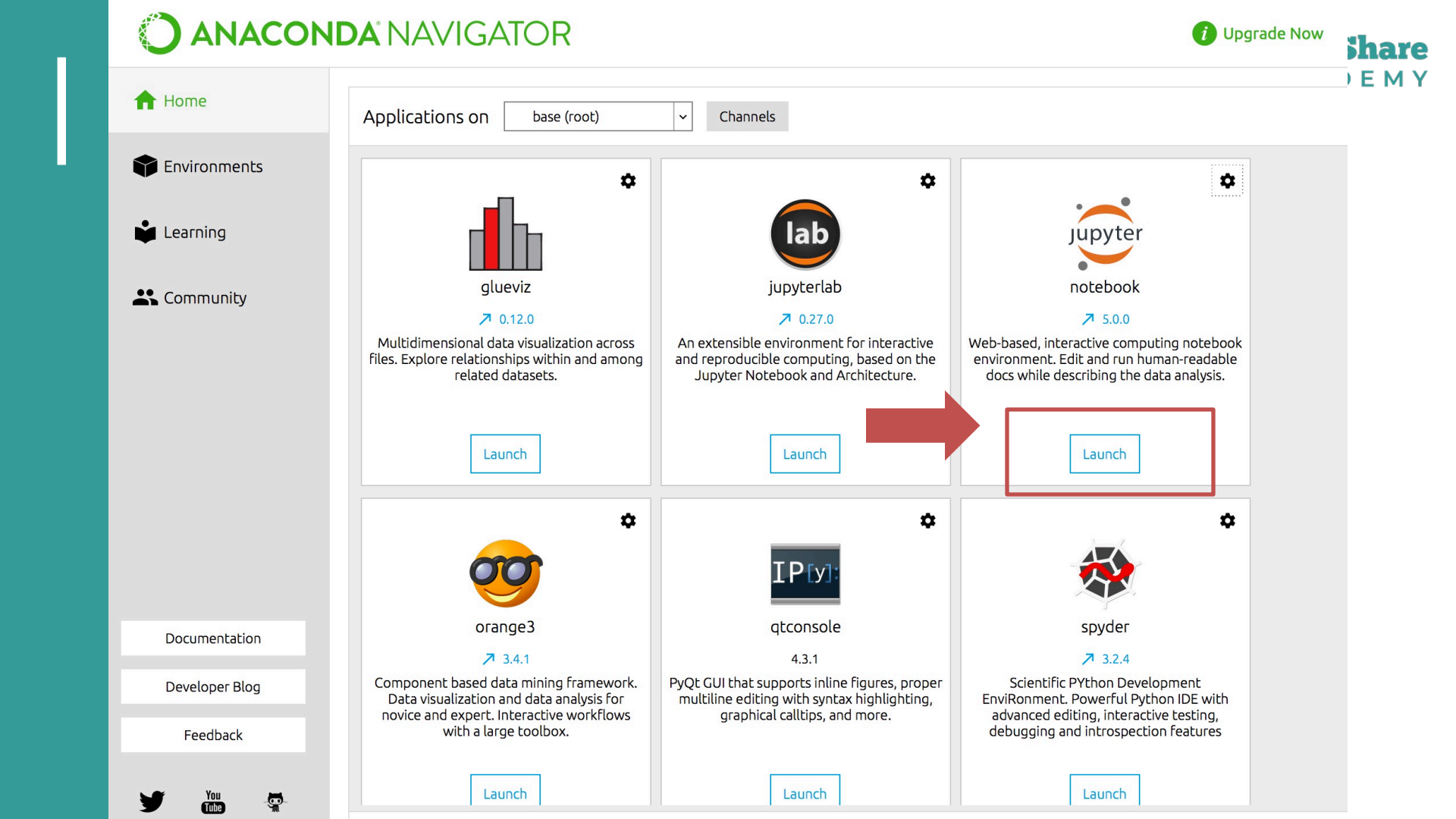
x1	x2
A	1

```
pd.merge(ydf, zdf, how='outer',
         indicator=True)
.query('_merge == "left_only"')
.drop(columns=['_merge'])
```
Rows that appear in ydf but not zdf (Setdiff).

Uruchamiany Jupyter Notebook






ANACONDA NAVIGATOR
Upgrade Now
Home
Environments
Learning
Community
Documentation
Developer Blog
Feedback
Applications on
base (root)
Channels
glueviz
0.12.0
Multidimensional data visualization across files. Explore relationships within and among related datasets.
Launch
jupyterlab
0.27.0
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch
notebook
5.0.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch
orange3
3.4.1
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.
Launch
qtconsole
4.3.1
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.
Launch
spyder
3.2.4
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
Launch

Co to jest Machine Learning?

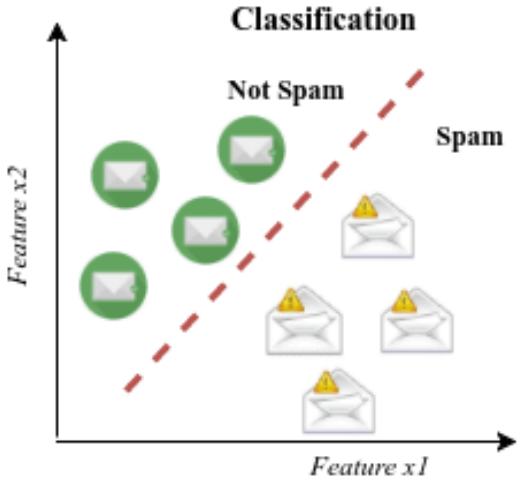
Grupa algorytmów umożliwiających automatyczne wykrywanie zależności w danych.
Dzieli się na uczenie nadzorowane i nienadzorowane.

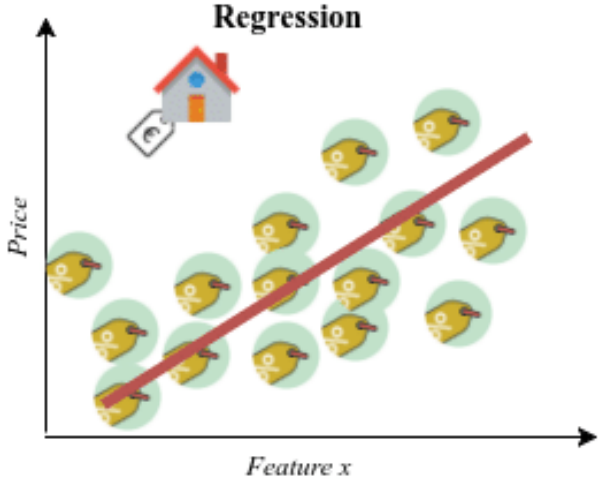
Uczenie nadzorowane to problemy:

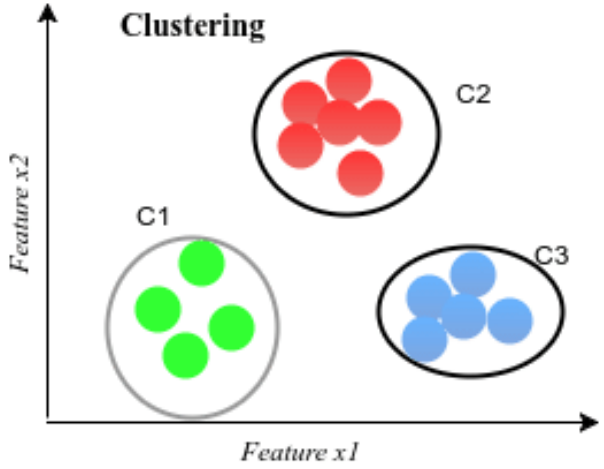
- **Klasyfikacji** - kiedy docelowa cecha jest jakąś klasą np. gatunek kwiatka
- **Regresji** - kiedy docelowa cecha jest wyrażona liczbowo np. cena mieszkania

Uczenie nienadzorowane typowo używa się w problemach **klastrowania** danych.

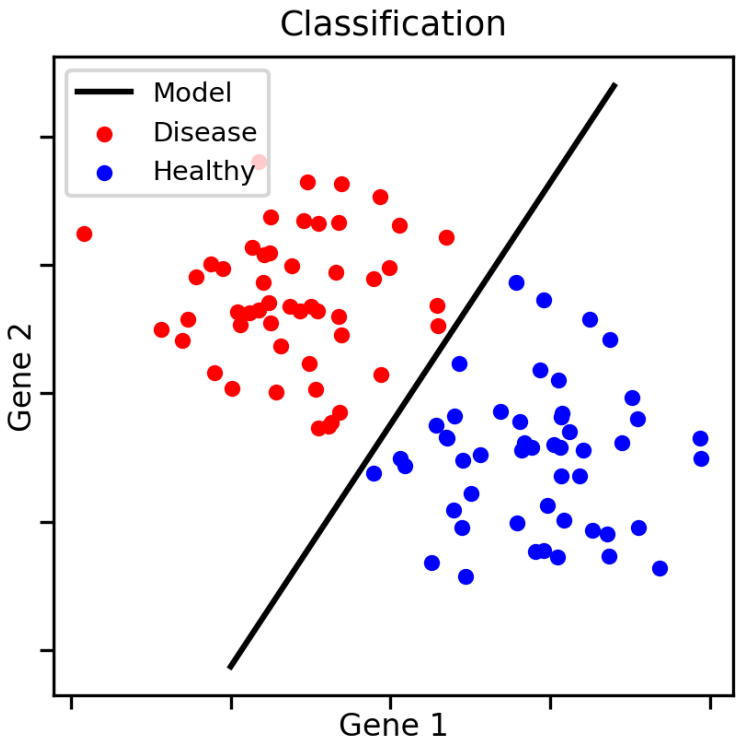
Co to jest Machine Learning?

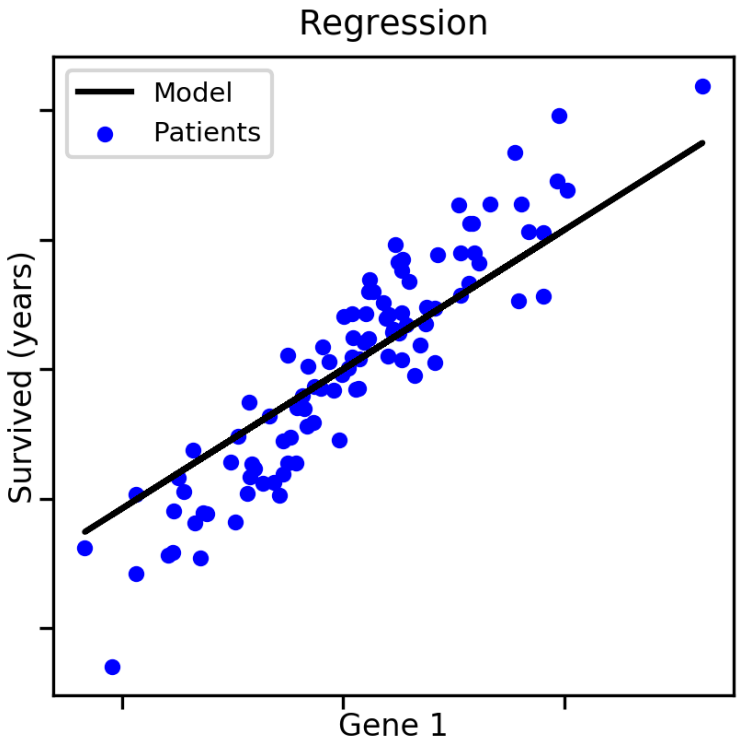


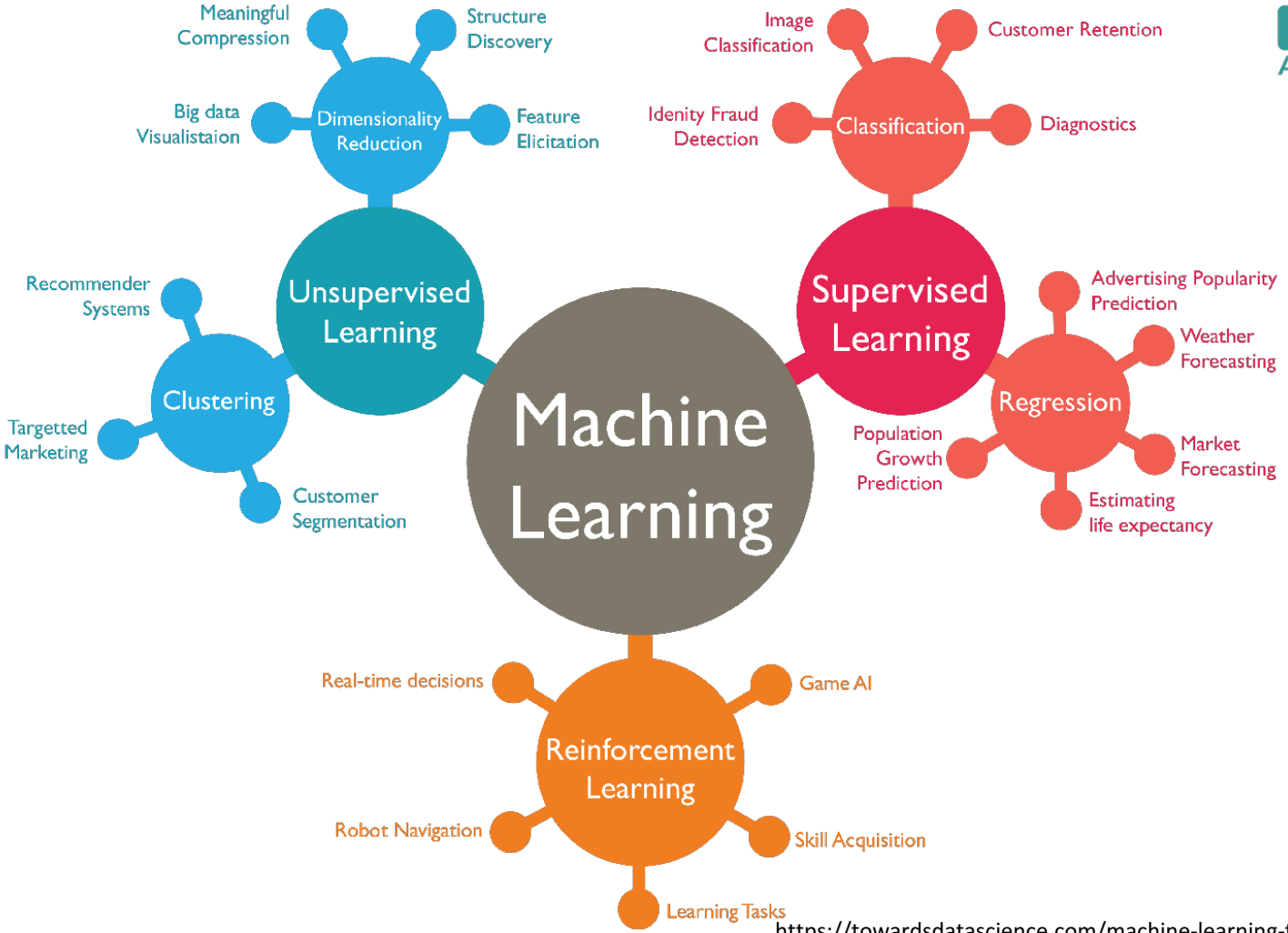




Co to jest Machine Learning?







https://towardsdatascience.com/machine-learning-types-2-c1291d4f04b1

Nazewnictwo w Machine Learning

Sample

Przypadki testowe

Obserwacje

Rzędy

	Country	Happiness.Score	Economy..GDP.per.Capita.	Family	Health..Life.Expectancy.	Freedom	Generosity	Trust..Government.Corruption.
0	Norway	7.537	1.616463	1.533524	0.796667	0.635423	0.362012	0.315964
1	Denmark	7.522	1.482383	1.551122	0.792566	0.626007	0.355280	0.400770
2	Iceland	7.504	1.480633	1.610574	0.833552	0.627163	0.475540	0.153527
3	Switzerland	7.494	1.564980	1.516912	0.858131	0.620071	0.290549	0.367007
4	Finland	7.469	1.443572	1.540247	0.809158	0.617951	0.245483	0.382612
5	Netherlands	7.377	1.503945	1.428939	0.810696	0.585384	0.470490	0.282662
6	Canada	7.316	1.479204	1.481349	0.834558	0.611101	0.435540	0.287372
7	New Zealand	7.314	1.405706	1.548195	0.816760	0.614062	0.500005	0.382817
8	Sweden	7.284	1.494387	1.478162	0.830875	0.612924	0.385399	0.384399
9	Australia	7.284	1.484415	1.510042	0.843887	0.601607	0.477699	0.301184
10	Israel	7.213	1.375382	1.376290	0.838404	0.405989	0.330083	0.085242
11	Costa Rica	7.079	1.109706	1.416404	0.759509	0.580132	0.214613	0.100107
12	Austria	7.006	1.487097	1.459945	0.815328	0.567766	0.316472	0.221060
13	United States	6.993	1.546259	1.419921	0.774287	0.505741	0.392579	0.135639
14	Ireland	6.977	1.535707	1.558231	0.809783	0.573110	0.427858	0.298388

Features Cechy Własności Kolumny

Nazewnictwo w Machine Learning

Zmienna objaśniana

Zmienna zależna

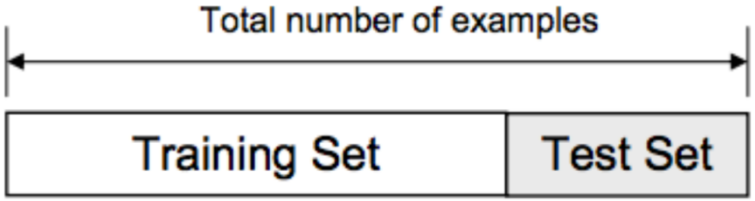
Y

y

Zmienne objaśniające Zmienne niezależne X

	Country	Happiness.Score	Economy..GDP.per.Capita.	Family	Health..Life.Expectancy.	Freedom	Generosity	Trust..Government.Corruption.
0	Norway	7.537	1.616463	1.533524	0.796667	0.635423	0.362012	0.315964
1	Denmark	7.522	1.482383	1.551122	0.792566	0.626007	0.355280	0.400770
2	Iceland	7.504	1.480633	1.610574	0.833552	0.627163	0.475540	0.153527
3	Switzerland	7.494	1.564980	1.516912	0.858131	0.620071	0.290549	0.367007
4	Finland	7.469	1.443572	1.540247	0.809158	0.617951	0.245483	0.382612
5	Netherlands	7.377	1.503945	1.428939	0.810696	0.585384	0.470490	0.282662
6	Canada	7.316	1.479204	1.481349	0.834558	0.611101	0.435540	0.287372
7	New Zealand	7.314	1.405706	1.548195	0.816760	0.614062	0.500005	0.382817
8	Sweden	7.284	1.494387	1.478162	0.830875	0.612924	0.385399	0.384399
9	Australia	7.284	1.484415	1.510042	0.843887	0.601607	0.477699	0.301184
10	Israel	7.213	1.375382	1.376290	0.838404	0.405989	0.330083	0.085242
11	Costa Rica	7.079	1.109706	1.416404	0.759509	0.580132	0.214613	0.100107
12	Austria	7.006	1.487097	1.459945	0.815328	0.567766	0.316472	0.221060
13	United States	6.993	1.546259	1.419921	0.774287	0.505741	0.392579	0.135639
14	Ireland	6.977	1.535707	1.558231	0.809783	0.573110	0.427858	0.298388

Przygotowanie danych



Regresja Liniowa

Wzór matematyczny:

$$y_i = w_0 1 + w_1 x_{i1} + \cdots + w_p x_{ip} + \varepsilon_i \qquad i = 1, \ldots, n$$

n - liczba przypadków uczących

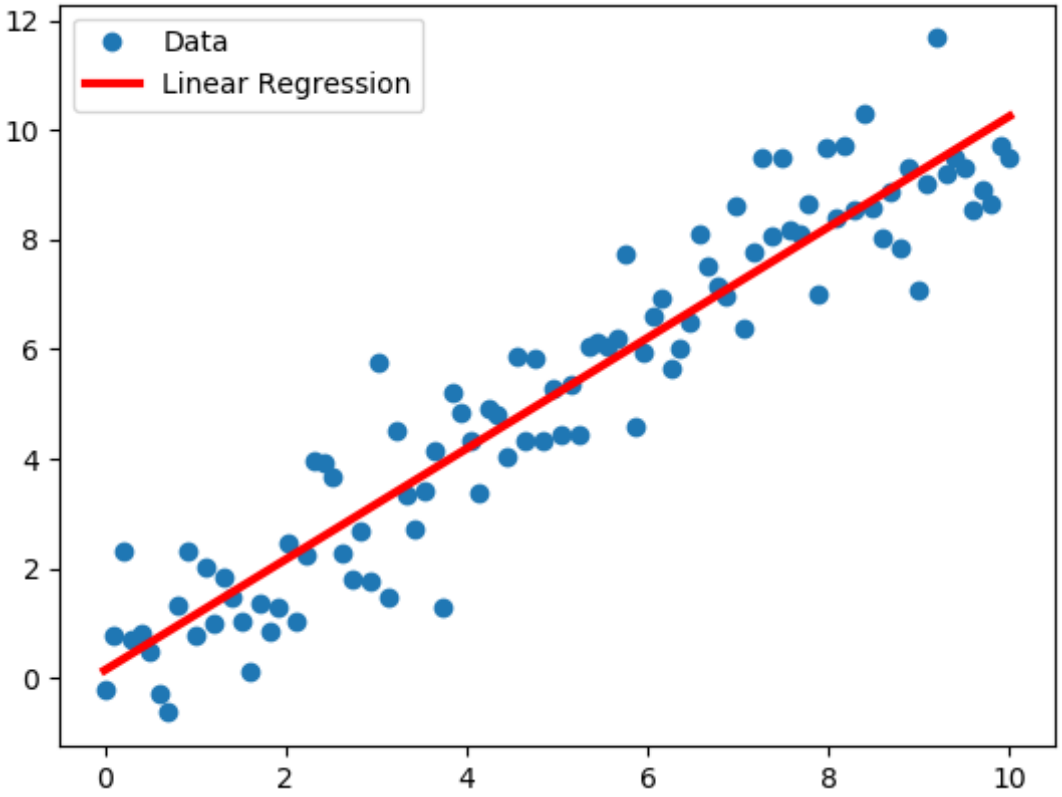
p - liczba cech

e - błąd

w - wagi, wartości które wylicza algorytm ML

Algorytm wylicza wagi tak, żeby błąd był **najmniejszy** dla wszystkich przypadków uczących jednocześnie.

Regresja Liniowa



https://medium.com/coinmonks/linear-regression-with-tensorflow-canned-estimators-6cc4ffddd14f

Błąd średniokwadratowy

Wzór matematyczny:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

n - liczba przypadków uczących

y - rzeczywista wartość docelowa

y w czapce - wartość przewidziana przez algorytm ML

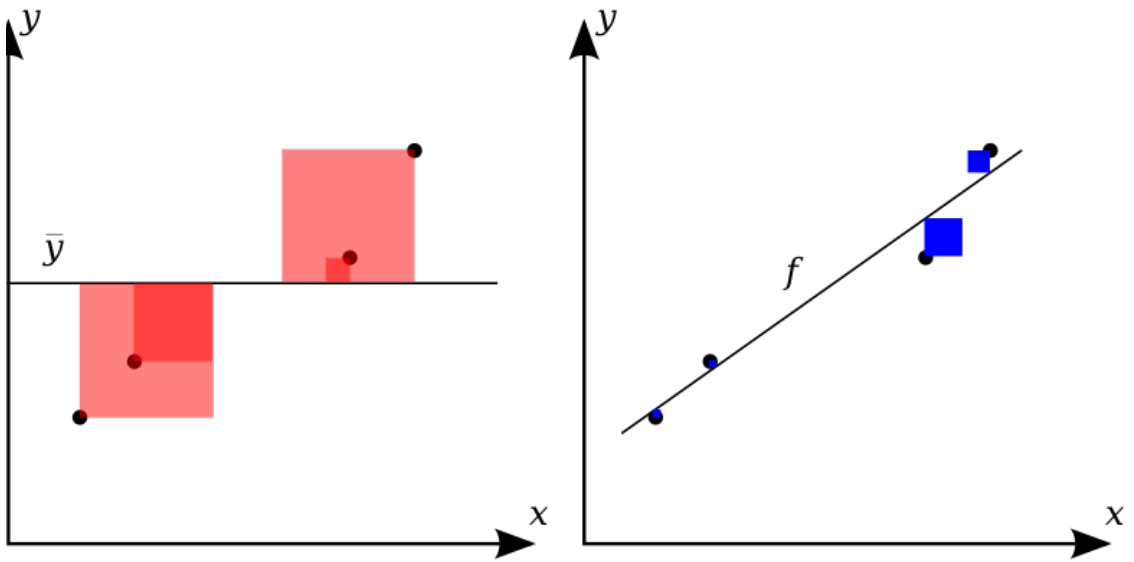
Różnicę między wartościami podnosimy do kwadratu, ponieważ może być zarówno ujemna jak i dodania.

Współczynnik determinacji R2

$$SS_{\text{res}} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2$$

$$SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2,$$

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$



Tworzenie modelu w Scikit-learn

FIT

uczenie modelu - znalezienie parametru modelu dla zadanych danych uczących

PREDICT

użycie modelu - znalezienie odpowiedzi modelu dla zadanych - nowych danych

Dzieki

Marcin.Zadroga@gmail.com

Źródła i ciekawe linki

https://www.datacamp.com/community/data-science-cheatsheets
https://en.wikipedia.org/wiki/Coefficient_of_determination
https://pandas.pydata.org
https://scikit-learn.org/stable/
https://towardsdatascience.com/machine-learning-types-2-c1291d4f04b1/
https://medium.freecodecamp.org/using-machine-learning-to-predict-the-quality-of-wines-9e2e13d7480d
https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6/
https://www.researchgate.net/figure/Examples-of-real-life-problems-in-the-context-of-supervised-and-unsupervised-learning_fig8_319093376
https://idatassist.com/avoid-inaccurate-conclusions-through-data-cleaning/

https://www.youtube.com/watch?v=L7R4HUQ-eQ0 - polecam!!!

scikit-learn algorithm cheat-sheet