FILIP WÓJCIK, SENIOR DATA SCIENTIST

FILIP.WOJCIK@OUTLOOK.COM, MADDATASCIENTIST.EU

DATA SCIENCE

Data science, ZIF, MNZ 2, sem 1

Filip Wójcik, filip.wojcik@outlook.com

AGENDA

1. Wstęp i narzędzia:

- ANACONDA
- JUPYTER NOTEBOOKS
- PYCHARM

2. Struktury danych w Pandas – DataFrame:

- Ładowanie danych z różnych źródeł
- Filtrowanie danych i obróbka wstępna
- Agregacja danych
- Kolumny wyliczeniowe i transformacje
- Obsługa wartości brakujących
- Tworzenie własnych funkcji
- Łączenie DataFrame'ów
- Wzorzec: split-apply-combine
- Tabele przestawne

5. Wizualizacja danych

- Silniki wizualizacyjne w Pythonie
- Wbudowane narzędzia wizualizacji
- Narzędzia pakietu Pandas
- Matplotlib
- Interaktywne wizualizacje z Bokeh

6. Podstawowa analiza danych

- StatModels vs Scipy.stats
- Statystyka matematyczna: testowanie hipotez
- T-testy, ANOVA
- Regresja logistyczna

7. Studium przypadku: Kaggle, Titanic

- Predykcja, kto przeżyje katastrofę
- Analiza eksploracyjna

CZĘŚĆ 1

NARZĘDZIA I PAKIETY

Data science, ZIF, MNZ 2, sem 1 Filip Wójcik, filip.wojcik@outlook.com

GŁÓWNE NARZĘDZIA DATA SCIENCE

Pandas

1. Ogólny pakiet do manipulacji danych
2. Czytanie i import danych

SciPy

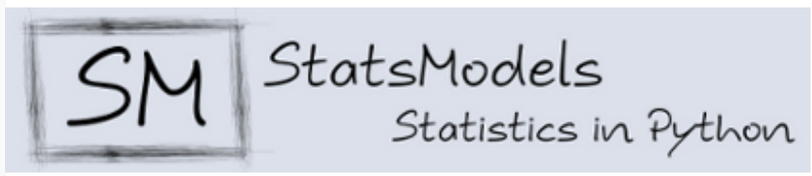
1. Pakiet naukowy
2. Specyficzne funkcje dla różnych działów matematyki
3. Szeroka gama narzędzi statystycznych

Scikit-learn

1. Główna biblioteka uczenia maszynowego
2. Implementuje większość najważniejszych algorytmów
3. Uwspólniony interfejs

Statsmodels

1. Pakiet statystyczny zawierający klasyczne metody
2. Statystyka matematyczna i opisowa
3. API zbliżone do języka R

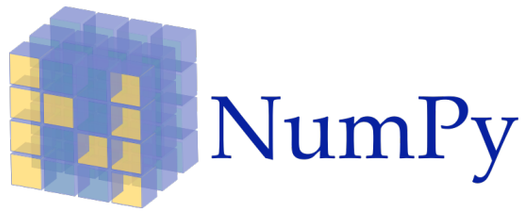


Matplotlib

1. Główny pakiet do wizualizaji
2. Najczęściej używany do tworzenie skomplikowanych grafów

Numpy

1. Część pakietu sci-py
2. Biblioteka do obliczzeń macierzowych
3. Silniki do optymalizacji numerycznej
4. Bardzo efektywny i wydajny (CPython)



Data science, ZIF, MNZ 2, sem 1 Filip Wójcik, filip.wojcik@outlook.com

ANACONDA – EKOSYSTEM ANALITYCZNY

- Stworzony przez Continuum Analytics
- Open source, zawiera Python i R
- Wieloplatformowy (Mac OS/Win/Linux)
- Zintegrowane narzędzia:
  - Jupyter
  - Sklearn/Scipy/Numpy/Pandas/etc
  - Wbudowane IDE: Spyder i Rstudio

ANACONDA®

Data science, ZIF, MNZ 2, sem 1 Filip Wójcik, filip.wojcik@outlook.com

JUPYTER NOTEBOOKS – ŚRODOWISKO DATA SCIENCE

- Rozwinięcie starszej technologii IPython
- Środowisko dla Data Science:
  - wizualizacje
  - połącznie kodu i dokumentacji
- Pracuje z Pythonem i R
- Proste podpowiadanie składni i intellisense
- Dostępny jako narzędzie online - e.g. kaggle.com, community.databricks.com

Filip Wójcik, filip.wojcik@outlook.com

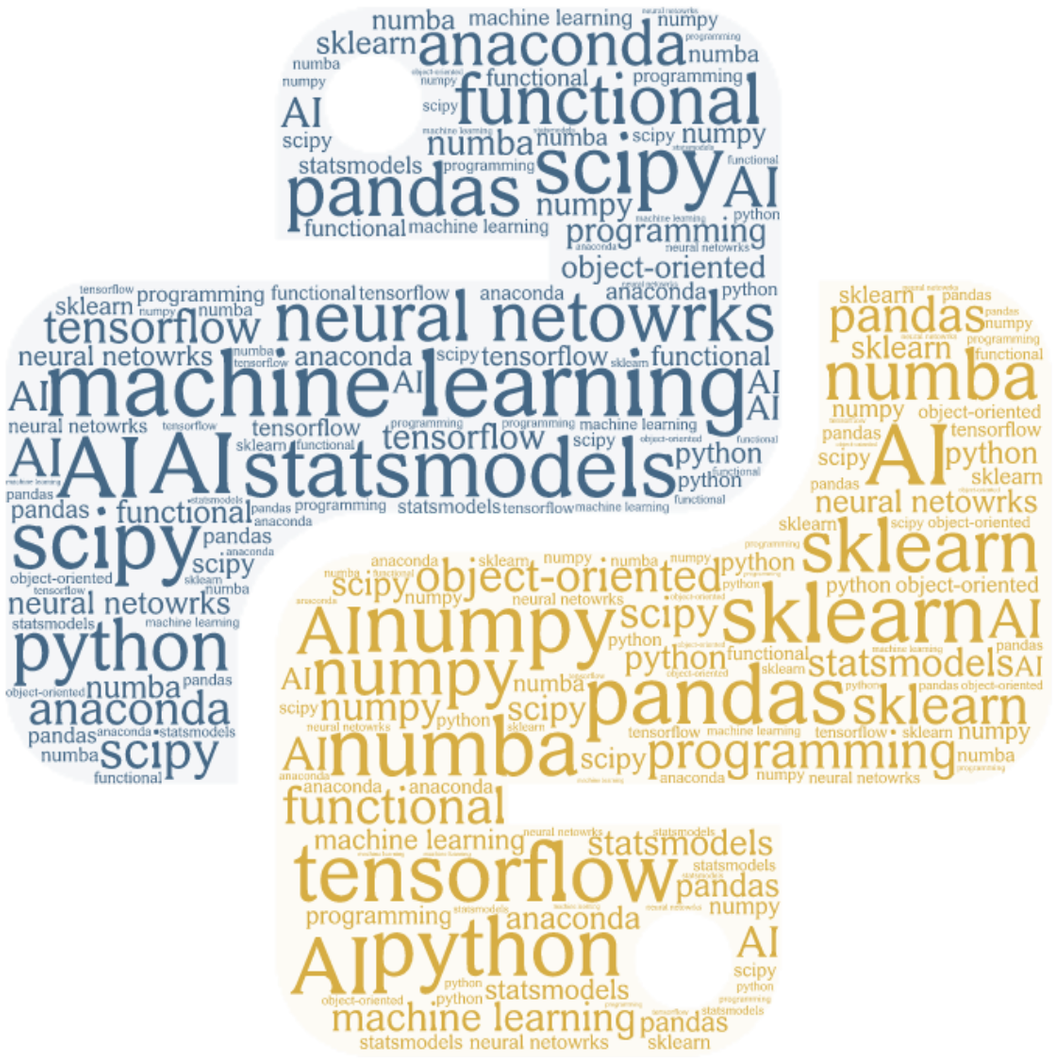
PYCHARM – IDE PROGRAMISTYCZNE

- Główne IDE do programowania w Pythonie
- Community (free)
- Łączy wiele funkcji:
  - dla webdeveloperów
  - dla developerów aplikacji
  - dla data scientistów

Data science, ZIF, MNZ 2, sem 1 Filip Wójcik, filip.wojcik@outlook.com

SETUP =)

<title>code ninja</title>



CZĘŚĆ 2

PODSTAWY PYTHONA

PODSTAWOWE TYPY DANYCH I DEKLARACJE

- Zmienne

```
1.  x = 'text variable'
2.
3.  i = 12 #numeric variable
4.
5.  j = [1, 2, 3] # array
6.
7.  k = ('a', 'b', 'c') # tuple
```

- Typy (3.6):

```
1.  x1: int = 3
2.  txt1: str = "aaa"
3.  lst1: list = [ 1, 2, 3 ]
```

TYP DANYCH	PRZYKŁAD

string	`str1 = "a"`
	`str2 = "aaa"`
	`str3 = "aaa aaa"`

numbers	`x1 = 1`
	`x2 = 10.4`
	`x3 = 12f`

lists	`list_of_numbers = [ 1, 2, 3 ]`
	`list_of_strings = [ "a", "b", "c" ]`
	`list_of_mixed_types = [ "A", 1, "B", 2 ]`

tuples	`tuple = ("a", 1, "b", 3)`

dictionaries	`dictionary = {`
	`"key1": 1,`
	`"key2": 2,`
	`"key3": "value3"`

booleans	`True` `False`

PODSTAWY – KOLEKCJE

- Operacje na listach

Deklaracja	`my_list = ['a', 'b', 'c']`	['a', 'b', 'c']

Indeksowanie	`my_list[0]`
	`my_list[1]`
	`my_list[2]`
		a'
		'b'
		'c'

Dodawanie	`my_list.append('d')`	['a', 'b', 'c', 'd']
Konkatenacja	`my_list + ['e']`	['a', 'b', 'c', 'e']
Sprawdzanie zawartości	`'a' in my_list` `'xxx' in my list`	True False

PODSTAWY – KOLEKCJE

- Operacje na słownikach

Deklaracje	`my_dict = {`
	`  'key1': 10,`
	`  'key2': 20,`
	`  'key3': 30`
	`}`
		`{'key1': 10, 'key2': 20, 'key3': 30}`

Indeksowanie	`# unsafe, may throw error`
	`my_dict['key1']`
	
	`# safer option`
	`my_dict.get('key1', 'no such value!')`
		10

Dodawanie elementów	`my_dict['keyX'] = 111`	`{'key1': 10, 'key2': 20, 'key3': 30, 'keyX': 111}`
Sprawdzanie zawartości	`'key1' in my_dict` `'key1' in my_dict`	True False

PODSTAWY – STRUKTURY STEROWANIA

warunkowe: if statement

```
if some_condition:
    # action if true
else:
    # action if false
```

```
x = 10
y = 5

if x <  y:
    print("x is lower than y")
else:
    print("y is bigger")
```

warunkowe: multiple if-else

```
if condition1:
    # action 1
elif condition2:
    # action 2
else:
    # default action
```

```
x = 10
y = 5
z = 1

if x <  y:
    print("x is lower than y")
elif x < z:
    print("x is lower then z")
else:
   print("x is the biggest one :) ")

    print("y is bigger")
```

PODSTAWY – STRUKTURY STEROWANIA

iteracja: for loop

```
for element in collection:
    # process element
```

```
lst = [1, 2, 3]
for elem in lst:
    print(elem)
```

iteracja: for with index

```
for idx, element in enumerate(collection):
    # process idx, process elem
```

```
lst = [ "A" , "B", "C" ]
for idx, elem in enumerate(lst):
    print(idx)
    print(elem)
    print("----")
```

PODSTAWY – STRUKTURY STEROWANIA

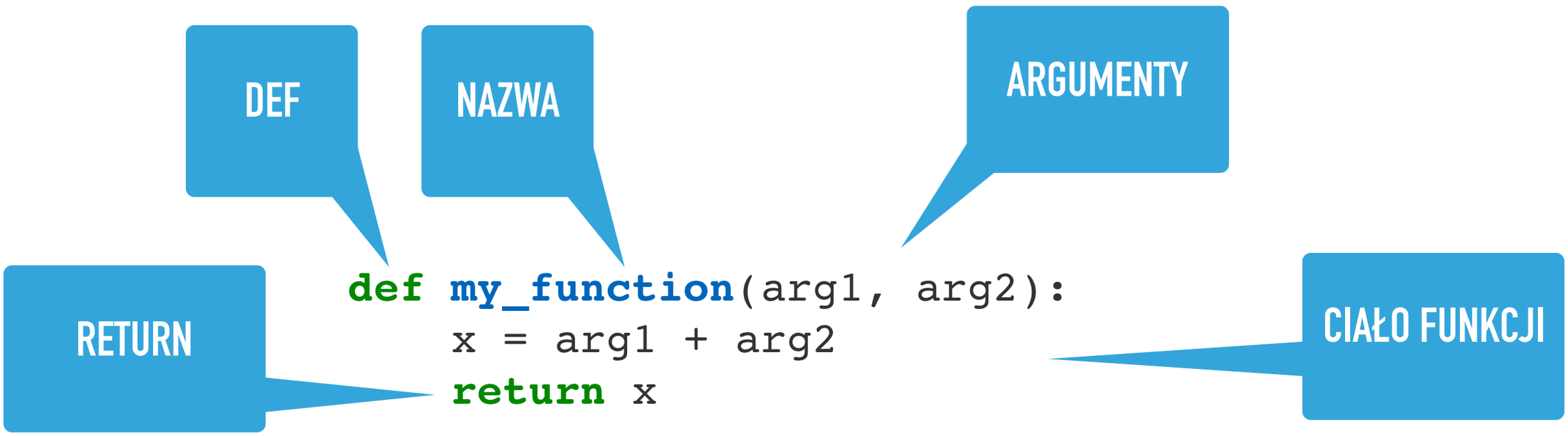
iteracja: klucze w słowniku

```
for key, value in my_dict.items():
    # process key, process value
```

```
my_dict = { 'key1': 1, 'key2': 2, 'key3': 3}
total = 0
for key, value in my_dict.items():
    print("key= ", key)
    total += value
print(total)
```

PODSTAWY – FUNKCJE

- “First class citizens” - zachowują się jak obiekty
- Można przekazywać je jak zmienne
- Przyjmują argumenty i zwracają wartości
- UWAGA: obiekty przekazywane są **przez referencję** i mogą być zmieniane w ciele funkcji

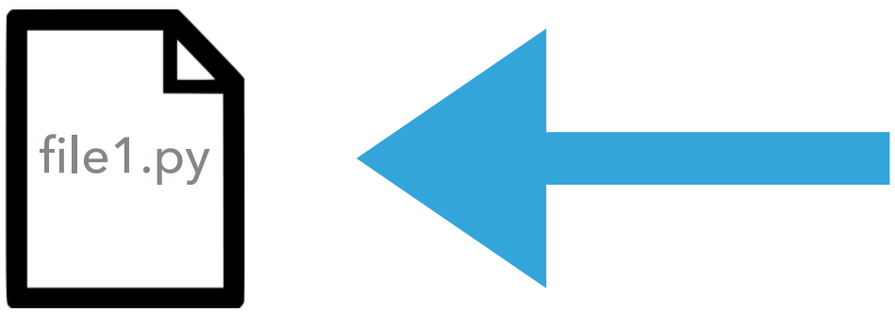


PODSTAWY – TYPY DANYCH I DEKLARACJE

- Ciała funkcji i modułów definiują wcięcia (nie jak w C#/ Java)

```
1. variable1 = 1
2. def some_function(x):
3.     # inner block - inside function
4.       return x + 3
5. some_function(variable1)
```

- Pliki Pythona są modułami, które można importować:




```
def add_numbers(x, y):
    return x + y
```

```
import file1 as f1

f1.add_numbers(1, 2)
```

```
from file1 import add_numbers
add_numbers(1, 2)
```

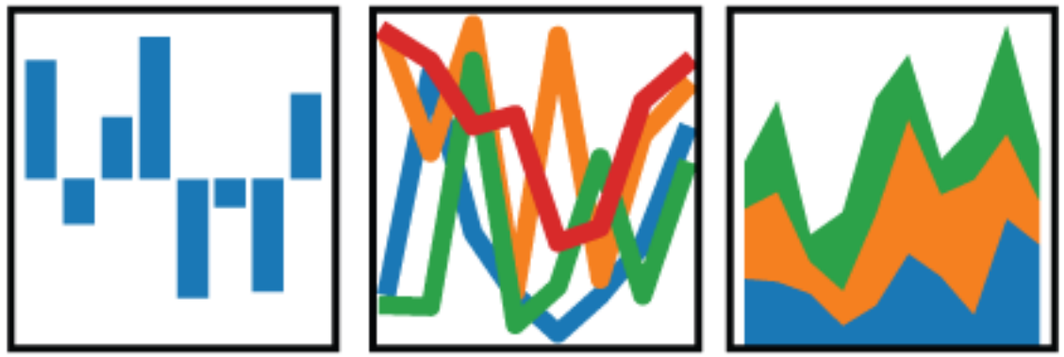
Data science, ZIF, MNZ 2, sem 1 Filip Wójcik, filip.wojcik@outlook.com

CODING TIME =)

<title>code ninja</title>

pandas

$$y_i t = \beta' x_{it} + \mu_i + \epsilon_{it}$$



CZĘŚĆ 3

WPROWADZENIE DO PANDAS

Data science, ZIF, MNZ 2, sem 1 Filip Wójcik, filip.wojcik@outlook.com

DATAFRAME – BAZOWA STRUKTURA DANYCH

- Data frame - podstawowa struktura danych dla Pandasa i innych bibliotek analitycznych
- Przypomina arkusz Excela albo tabelę w SQL
- Kolumnowa struktura danych - optymalizacja analityczna (pobieranie danych w kolumnach zamiast w wierszach):
  - łatwiejsze wyliczanie statystyk cechy
  - łatwiejsze wyszukiwanie korelacji pomiędzy zmiennymi

DATAFRAME – BAZOWA STRUKTURA DANYCH

INDEX: PRZYPOMINA KLUCZ GŁÓWNY

TEXT/STRING

NUMERICAL

DATE -TIME

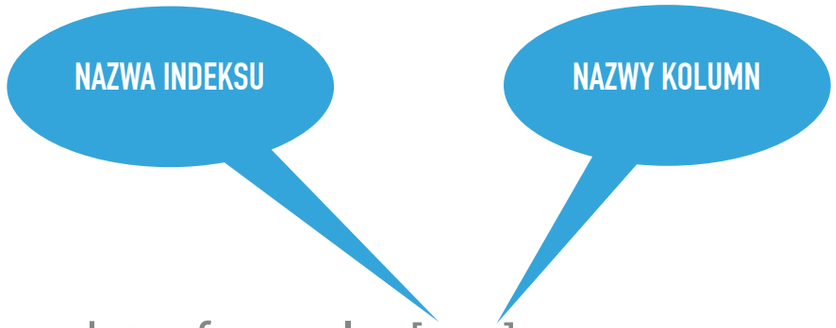
1. Index służy pobieraniu wierszy
2. Index może być dowolnym typem
3. Waże by wiedzieć, po czym się odwołujemy :)

index	Surname	Age	Inserted at
	Smith	40	20.10.2016
	Bean	45	02.01.2017
	Bond	NULL	04.06.2011
	Wayne	42	01.01.2010

Data science, ZIF, MNZ 2, sem 1

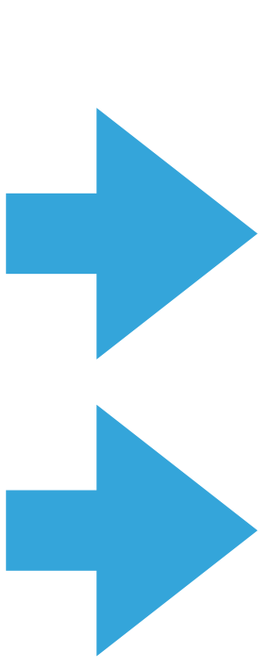
Filip Wójcik, filip.wojcik@outlook.com

DATA FRAME – ODWOŁANIE PO INDEKSIE



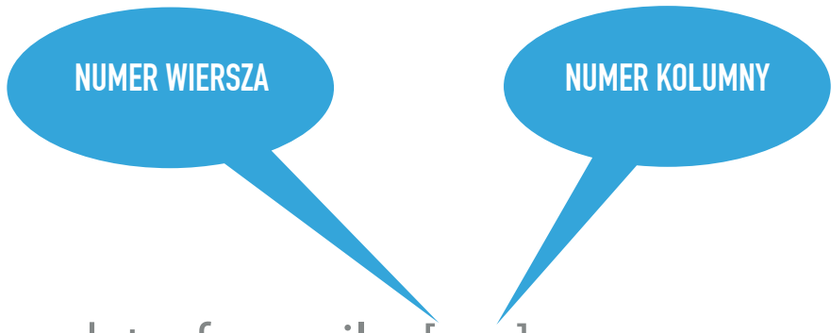
my_data_frame.loc[,]

my_data_frame.loc[['a', 'c'], ['Name', 'Surname']]



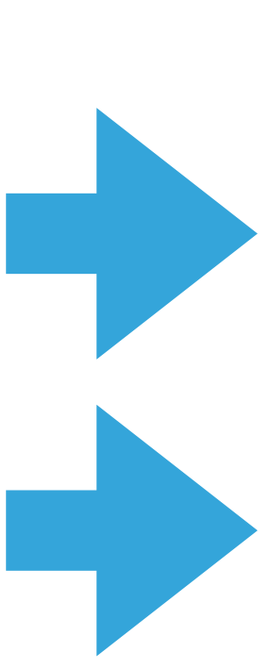
index\column	Name	Surname	Age	Inserted at
a	Agent	Smith	40	20.10.2016
b	Mr.	Bean	45	02.01.2017
c	James	Bond	NULL	04.06.2011
d	John	Wayne	42	01.01.2010

DATA FRAME – DOWOŁANIE PO POZYCJI



my_data_frame.iloc[,]

my_data_frame.iloc[[0, 2], [0,1]]



index\column	Name	Surname	Age	Inserted at
a	Agent	Smith	40	20.10.2016
b	Mr.	Bean	45	02.01.2017
c	James	Bond	NULL	04.06.2011
d	John	Wayne	42	01.01.2010

DATA FRAME – ODWOŁANIE PO PRZEDZIALE

PRZEDZIAŁY NUMERÓW WIERSZY

PRZEDZIAŁY NUMERÓW KOLUMN

OD-DO OTWARTY PRZEDZIAŁ

my_data_frame.iloc[,]

my_data_frame.iloc[0:2, 1:3]

index\column	Name	Surname	Age	Inserted at
a	Agent	Smith	40	20.10.2016
b	Mr.	Bean	45	02.01.2017
c	James	Bond	NULL	04.06.2011
d	John	Wayne	42	01.01.2010

DATA FRAME – POBIERANIE KOLUMNY

NAZWY KOLUMN

my_data_frame[ ]

my_data_frame[[“Name”, “Age”]]

index\column	Name	Surname	Age	Inserted at
a	Agent	Smith	40	20.10.2016
b	Mr.	Bean	45	02.01.2017
c	James	Bond	NULL	04.06.2011
d	John	Wayne	42	01.01.2010

Data science, ZIF, MNZ 2, sem 1

Filip Wójcik, filip.wojcik@outlook.com

DATA FRAME – OPERACJE NA KOLUMNACH

my_data_frame[“Age”] + 10

index\column	Name	Surname	Age	Inserted at
a	Agent	Smith	40 + 10	20.10.2016
b	Mr.	Bean	45 + 10	02.01.2017
c	James	Bond	NULL	04.06.2011
d	John	Wayne	42 + 10	01.01.2010

DATA FRAME – OPERACJE NA KOLUMNACH

my_data_frame[“Name”] + “ “ + my_data_frame[“Surname”]

index\column	Name	Surname	Age	Inserted at
a	Agent	Smith	40	20.10.2016
b	Mr.	Bean	45	02.01.2017
c	James	Bond	NULL	04.06.2011
d	John	Wayne	42	01.01.2010

Agent Smith
Mr. Bean
James Bond
John Wayne

Data science, ZIF, MNZ 2, sem 1

Filip Wójcik, filip.wojcik@outlook.com

DATA FRAME – OPERACJA NA WIERSZACH I KOLUMNACH

Axis 1

Axis 0

index\column	Age	Salary	Experience years
a	20	2500	2
b	25	5000	4
c	35	9500	7
d	31	9000	4

Data science, ZIF, MNZ 2, sem 1

Filip Wójcik, filip.wojcik@outlook.com

DATA FRAME – OPERACJA NA WIERSZACH I KOLUMNACH

Axis 1

Axis 0

index\column	Age	Salary	Experience years
a	20	2500	2
b	25	5000	4
c	35	9500	7
d	31	9000	4

my_data_frame.sum(axis=0)

age	111
experience	17
salary	26000

my_data_frame.sum(axis=1)

a	2522
b	5029
c	9542
d	9035

Data science, ZIF, MNZ 2, sem 1

Filip Wójcik, filip.wojcik@outlook.com

DATA FRAME – OPERACJA NA WIERSZACH I KOLUMNACH

Axis 1

Axis 0

index\column	Name	Surname	
a	Thomas	Anderson	Thomas Anderson
b	Agent	Smith	Agent Smith
c	The	Oracle	The Oracle
d	The	Architect	The Architect

```
my_data_frame.apply(lambda row: row['name'] + ' ' + row['surname'],  axis=1)
```

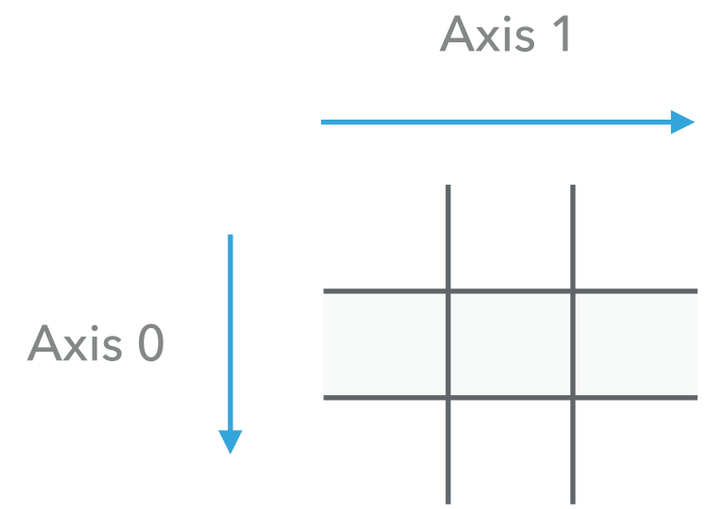
Data science, ZIF, MNZ 2, sem 1

Filip Wójcik, filip.wojcik@outlook.com

DATA FRAME – OPERACJA NA WIERSZACH I KOLUMNACH

Porady, kiedy używać określonych operacja

- Używanie operacji na wierszach "axis 1":
  - kiery przeprowadzamy operacje na wielu kolumnach :) - interakcja pomiędzy kolumnami
  - kiedy funkcja musi operować na wielu wartościach z różnych kolumn w ramach wiersza



- Używanie operacji na kolumnach "axis 0":
  - używanie niestandardowych funkcji
  - usuwanie kolumn

Data science, ZIF, MNZ 2, sem 1

Filip Wójcik, filip.wojcik@outlook.com

DATA FRAME – ŁĄCZENIE TABEL

pandas.merge(df1, df2, how=METHOD, on=KEY)

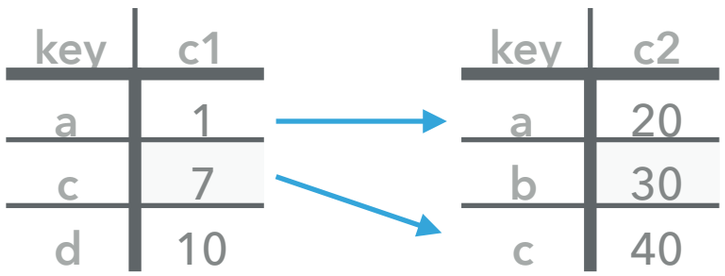
key	c1
a	1
c	7
d	10

key	c2
a	20
b	30
c	40

LEFT

Zachowaj wiersze z lewej i próbuj

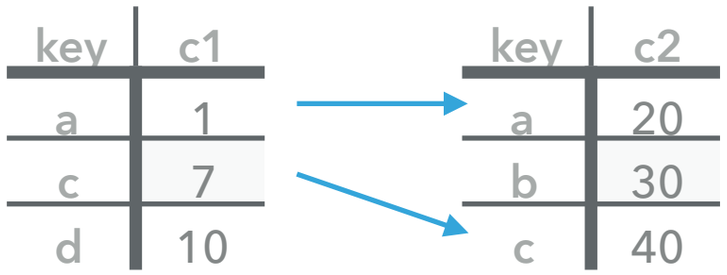
dopasować wiersze z prawej



key	c1	c2
a	1	20
c	7	30
d	10	NULL

INNER

Zachowaj tylko dopasowane wiersze

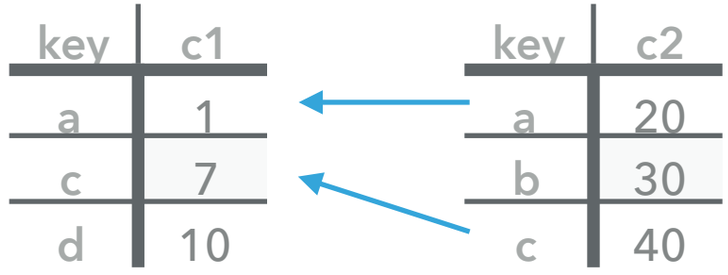


key	c1	c2
a	1	20
c	7	40

RIGHT

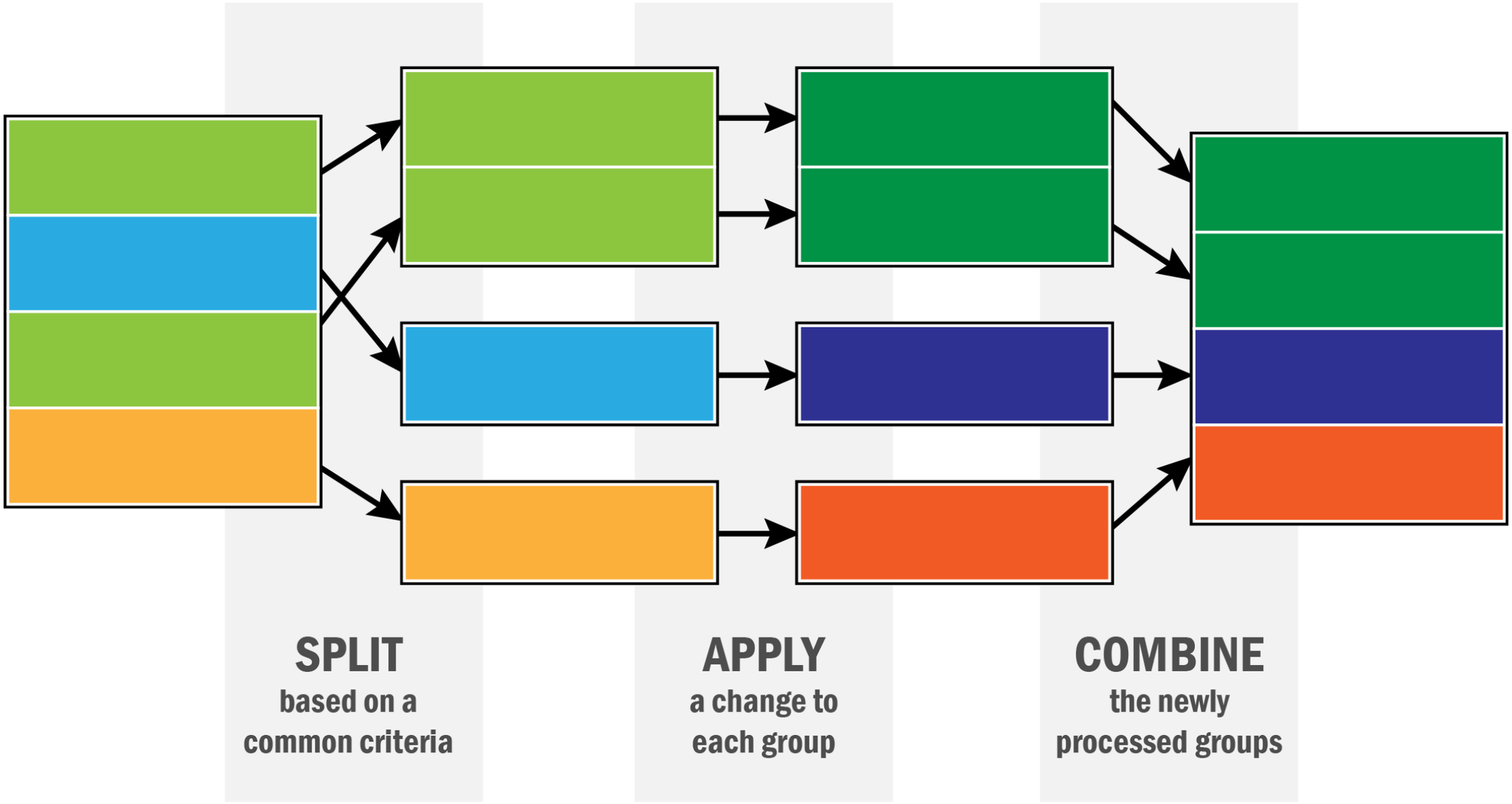
Zachowaj wiersze z prawej i próbuj

dopasować wiersze z lewej



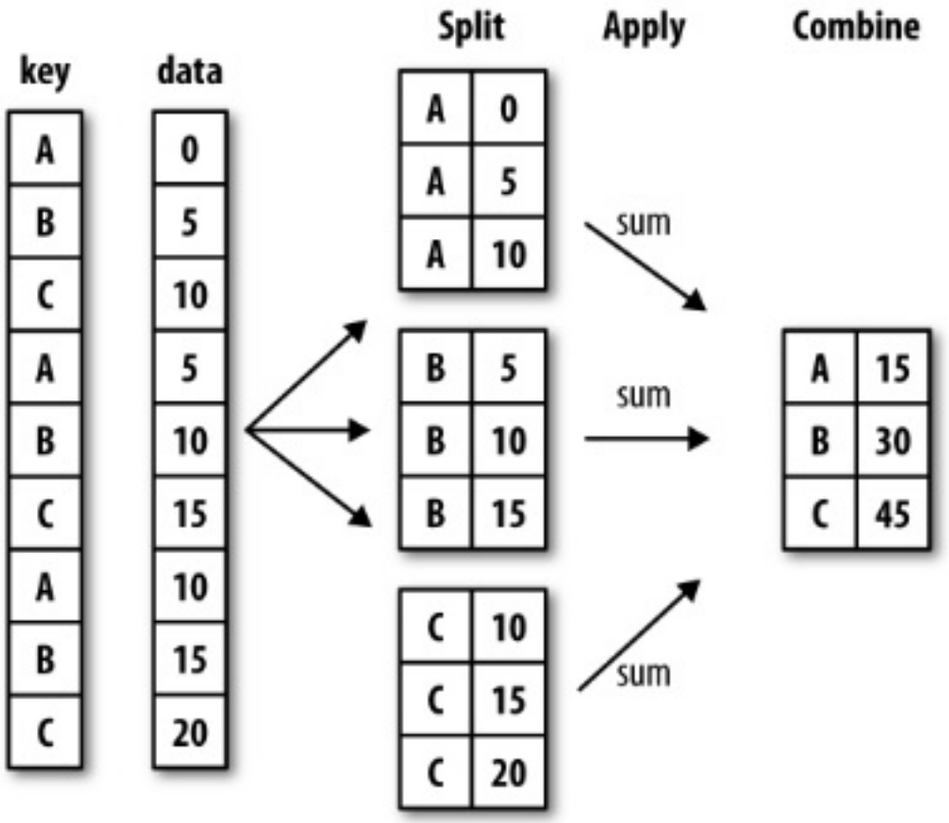
key	c1	c2
a	1	20
b	NULL	30
c	7	40

SPLIT-APPLY-COMBINE



Source: https://programminghistorian.org/lessons/text-mining-with-extracted-features

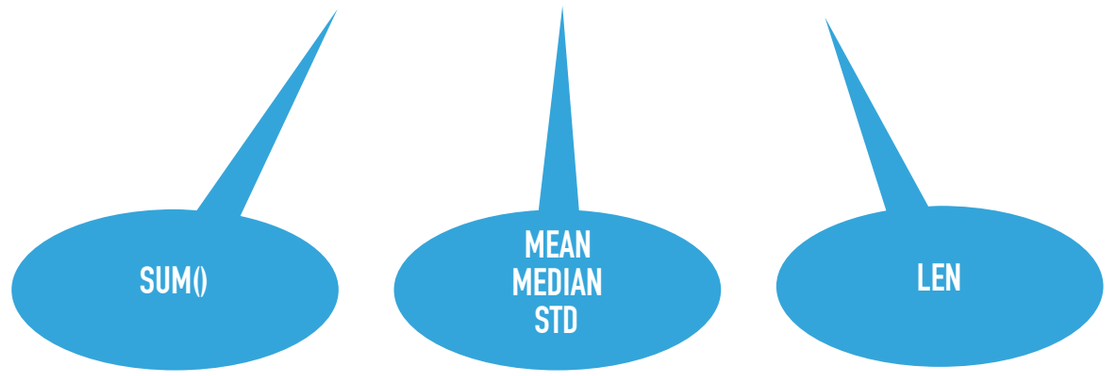
SPLIT-APPLY-COMBINE PATTERN: AGREGACJA



my_data_frame[“VALUE_COLUMN_HERE”]

.groupby(my_data_frame[“KEY_COLUMN_HERE”])

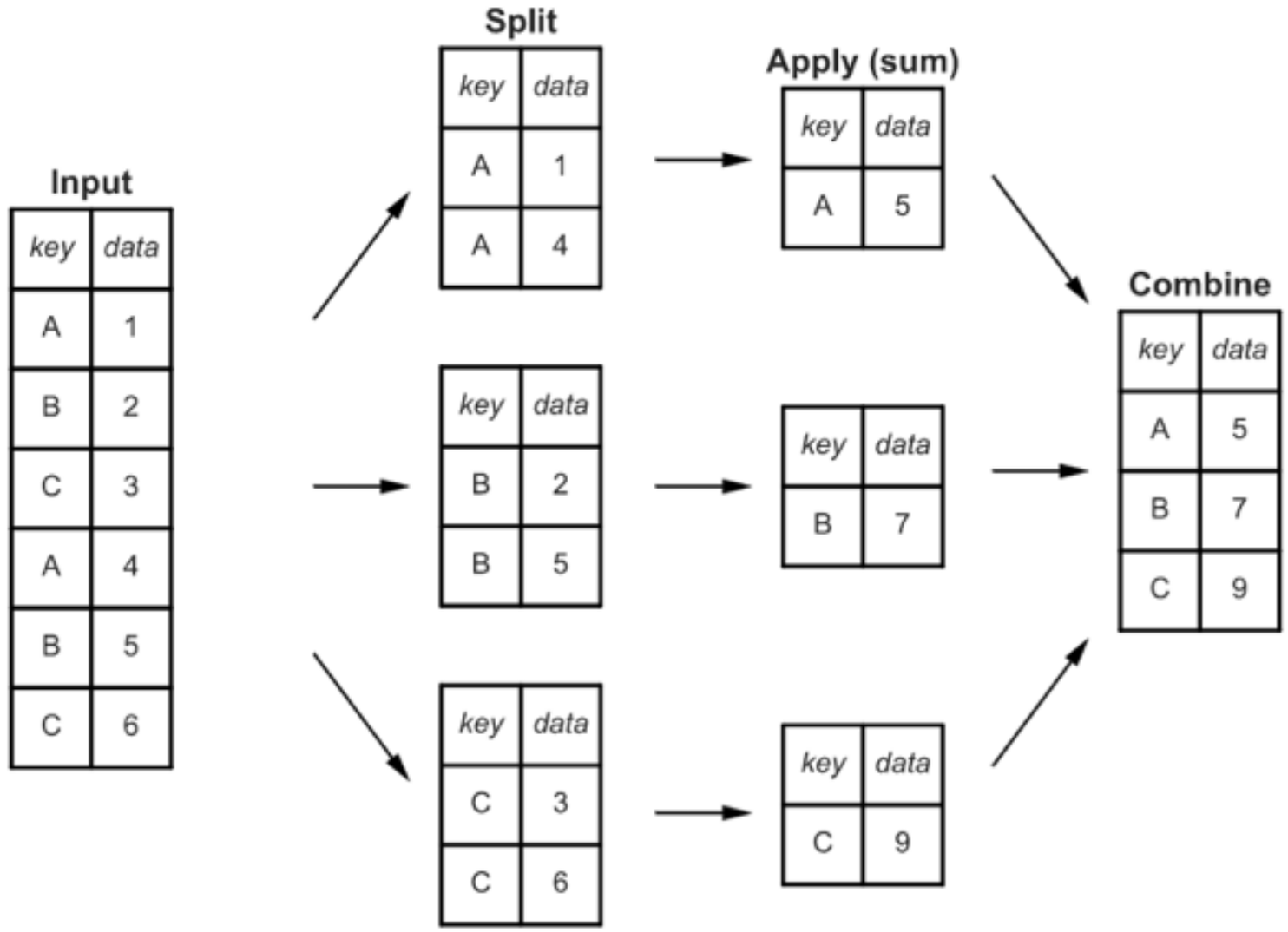
.AGGREGATION_FUNCTION_HERE



Source: William McKinney, Python for Data Analysis, 2nd Edition

SPLIT-APPLY-COMBINE PATTERN: AGREGACJA

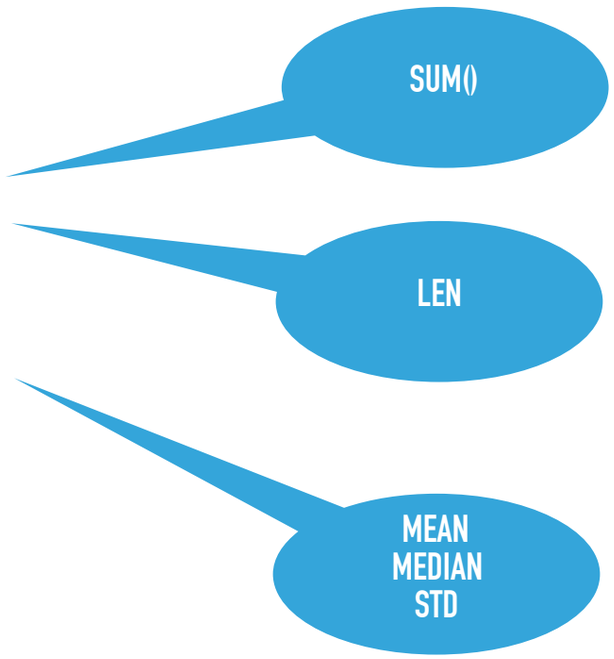
input_df[“data”].groupby(‘key”).sum()



Source: William McKinney, Python for Data Analysis, 2nd Edition

SPLIT-APPLY-COMBINE PATTERN: AGREGACJA

```
my_data_frame[  “VALUE_COLUMN_HERE” ]
   .groupby(my_data_frame[ “KEY_COLUMN_HERE”])
   .agg({
      “column1”: [ FUNCITONS ],
      “column2”: [ FUNCTIONS ],
      …
      “columnN": [ FUNCTIONS ]
})
```



Source: William McKinney, Python for Data Analysis, 2nd Edition

SPLIT-APPLY-COMBINE PATTERN: AGREGACJA

key	c1	c2	c3
a	1	2	3
b	4	5	6
a	7	8	9
b	10	11	12

key	c1	c2	c3
a	1	2	3
a	7	8	9

key	c1	c2	c3
b	4	5	6
b	10	11	12

key	c1		c2	c3	
OPER	sum	mean	sum	std	sum
a	8	4	10	3	12

C1: [sum, mean]

C2: [sum]

C3: [std, sum]

key	c1		c2	c3	
OPER	sum	mean	sum	std	sum
b	14	7	16	3	18

key	c1		c2	c3	
OPER	sum	mean	sum	std	sum
a	8	4	10	3	12
b	14	7	16	3	18

Data science, ZIF, MNZ 2, sem 1 Filip Wójcik, filip.wojcik@outlook.com

TABELE PRZESTAWNE

- Idea jest identyczna jak w Excelu
- Zamiana miejscami wierszy/kolumn i agregacja
- Przydatne podczas inspekcji danych z różnych punktów widzenia
- Tzw. drilldown znany z Business Intelligence albo data mining'u

TABELE PRZESTAWNE

Sex	Survived	Passenger Class
male	TRUE	1
male	FALSE	1
female	TRUE	2
female	TRUE	3
male	FALSE	3

Zadanie: policzyć pasażerów, którzy przetrwali

i pogrupować ich wg. płci i klasy

```
pd.pivot_table(
        titanic_data_small,
        values='survived',
        index='passenger class',
        columns='sex',

aggfunc=np.sum)
```

aggfunc = func(Values)

Columns

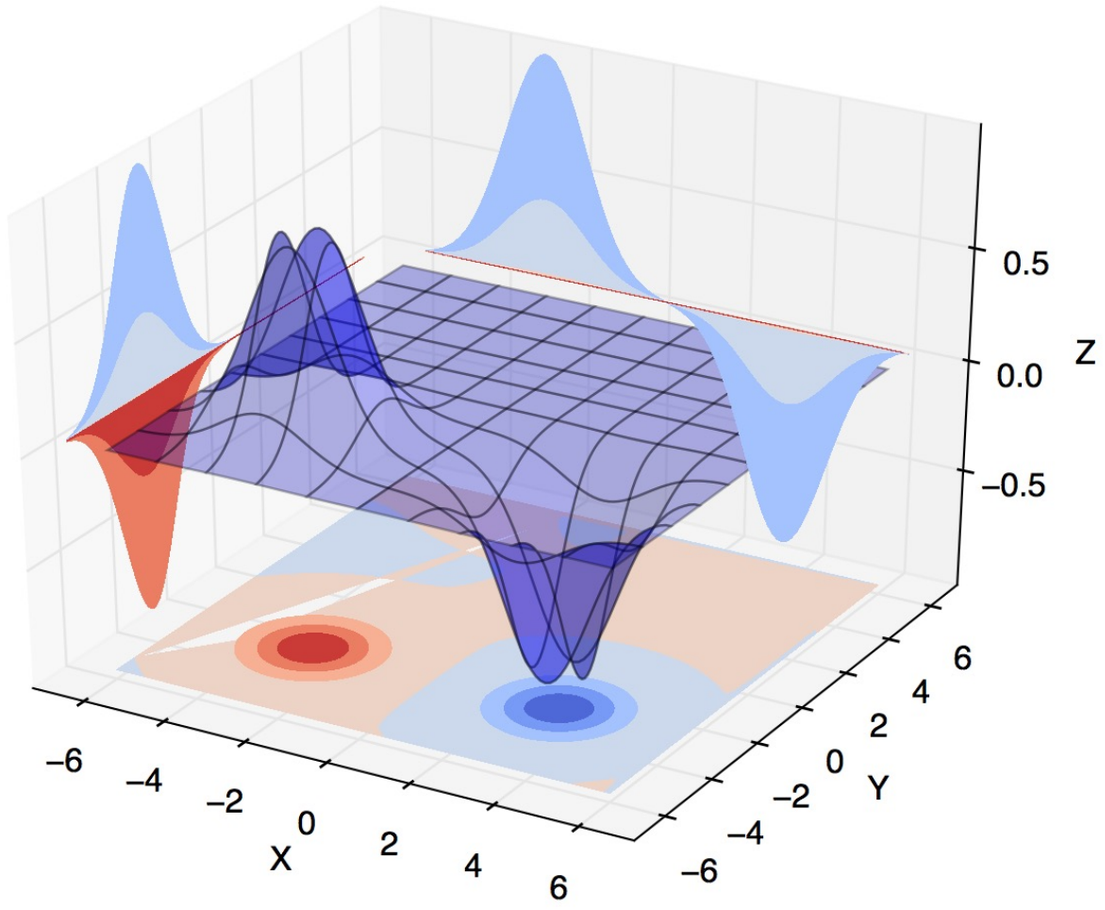
Index

Pclass\Sex	felame	male
1	-	1
2	1	-
3	1	0

Data science, ZIF, MNZ 2, sem 1 Filip Wójcik, filip.wojcik@outlook.com

CODING TIME =)

<title>code ninja</title>



CZĘŚĆ 4

WIZUALIZACJA DANYCH

Data science, ZIF, MNZ 2, sem 1 Filip Wójcik, filip.wojcik@outlook.com

API

- Istnieje kilka głównych silników wizualizacji w Pythonie
- Niektóre z nich są bardziej zaawansowane, niektóre służą tylko do zgrubnego rysowania
- Nie trzeba znać każdego API z osobna na pamięć :)
  Wystarczy ogólna znajomość ich cech i właściwości

Data science, ZIF, MNZ 2, sem 1

Filip Wójcik, filip.wojcik@outlook.com

API

	PANDAS	MATPLOTLIB
Typ	Wbudowany w bibliotekę	Zewnętrzna biblioteka
Przypadki użycia	Wizualizacja Data Frame'ów	Biblioteka ogólnego przeznaczenia
Złożoność	Bardzo prosta	Złożona
Elastyczność	Mała	Duża

Data science, ZIF, MNZ 2, sem 1 Filip Wójcik, filip.wojcik@outlook.com

PANDAS PLOTTING

- Oparty na Data Frame'ach i kolumnach
- Może tworzyć tylko wizualizacje per kolumna
- Ograniczona liczba predefiniowanych wizualizacji
- Proste sposoby dostosowywania wykresów

Data science, ZIF, MNZ 2, sem 1 Filip Wójcik, filip.wojcik@outlook.com

PANDAS PLOTTING

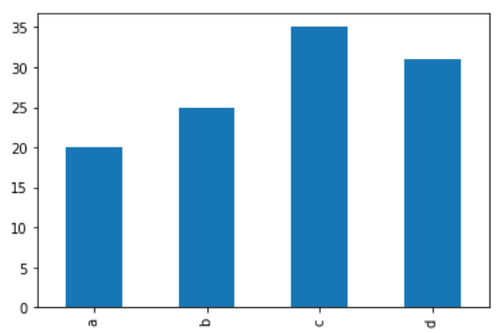
my_data_frame[COLUMN(s)].plot(SETTINGS)

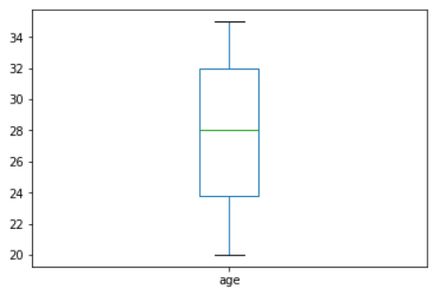
typ

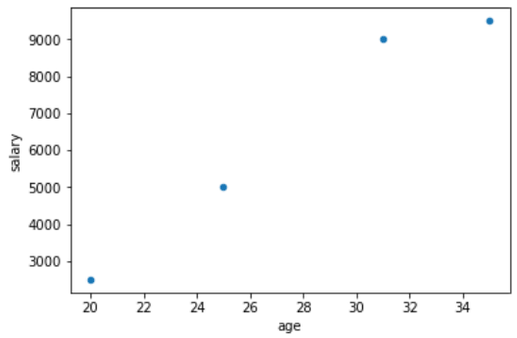
- bar
- histogram
- box
- scatter
- pie

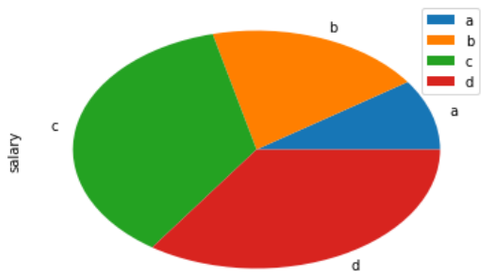
title

labels







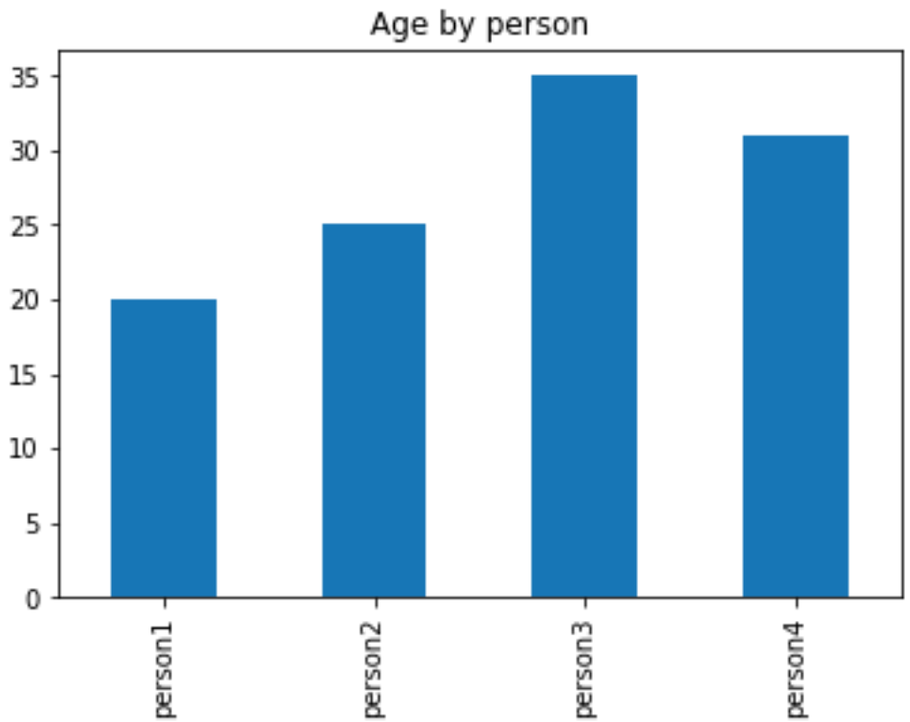


Data science, ZIF, MNZ 2, sem 1 Filip Wójcik, filip.wojcik@outlook.com

PANDAS PLOTTING

example_df[“age”].plot(kind=‘bar', title='Age by person')

key	age	experience	salary	position
person1	20	2	2500	tester
person2	25	4	5000	tester
person3	35	7	9500	developer
person4	31	4	9000	developer

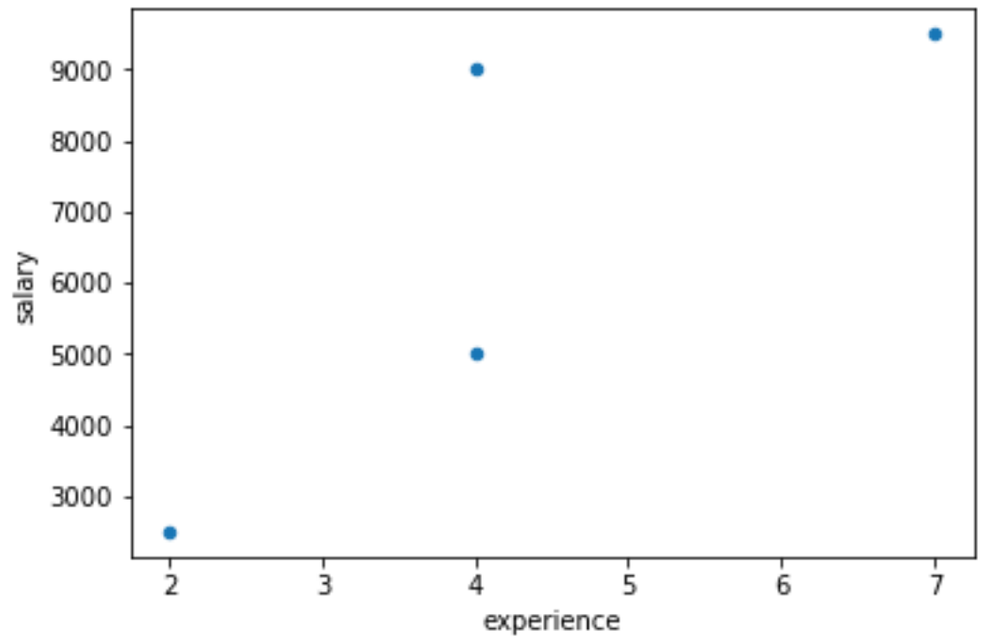


Data science, ZIF, MNZ 2, sem 1 Filip Wójcik, filip.wojcik@outlook.com

PANDAS PLOTTING

df[['experience', 'salary']].plot(kind="scatter", x='experience', y='salary')

key	age	experience	salary	position
person1	20	2	2500	tester
person2	25	4	5000	tester
person3	35	7	9500	developer
person4	31	4	9000	developer

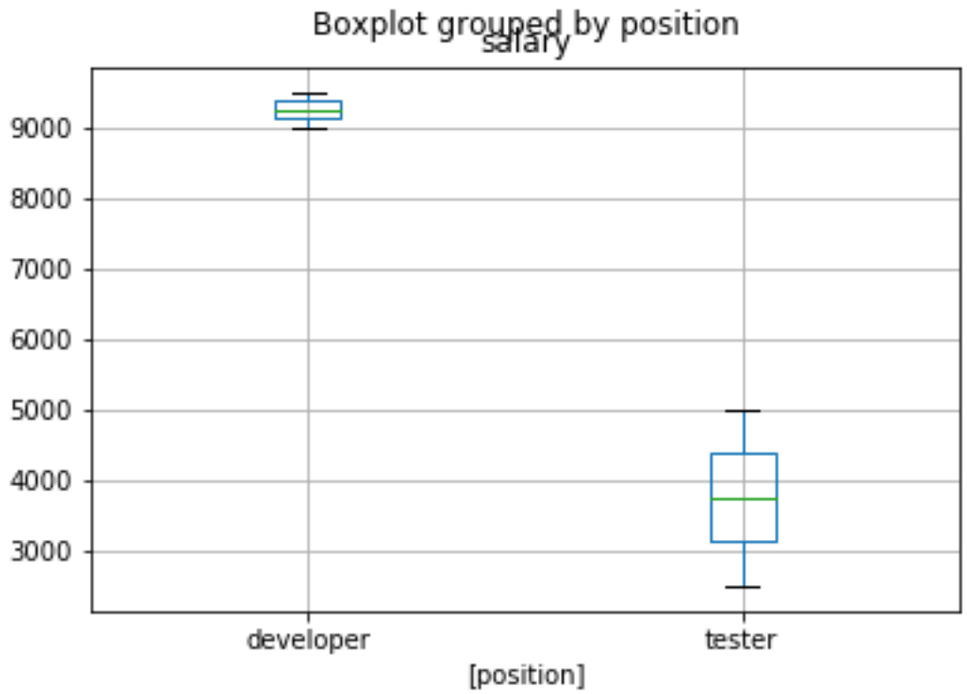


Data science, ZIF, MNZ 2, sem 1 Filip Wójcik, filip.wojcik@outlook.com

PANDAS PLOTTING

df[['position', 'salary']].boxplot(by='position')

key	age	experience	salary	position
person1	20	2	2500	tester
person2	25	4	5000	tester
person3	35	7	9500	developer
person4	31	4	9000	developer

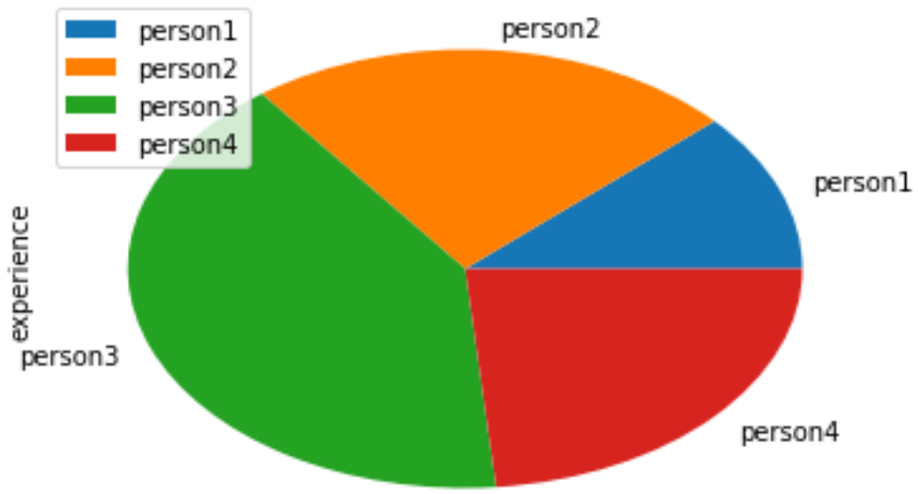


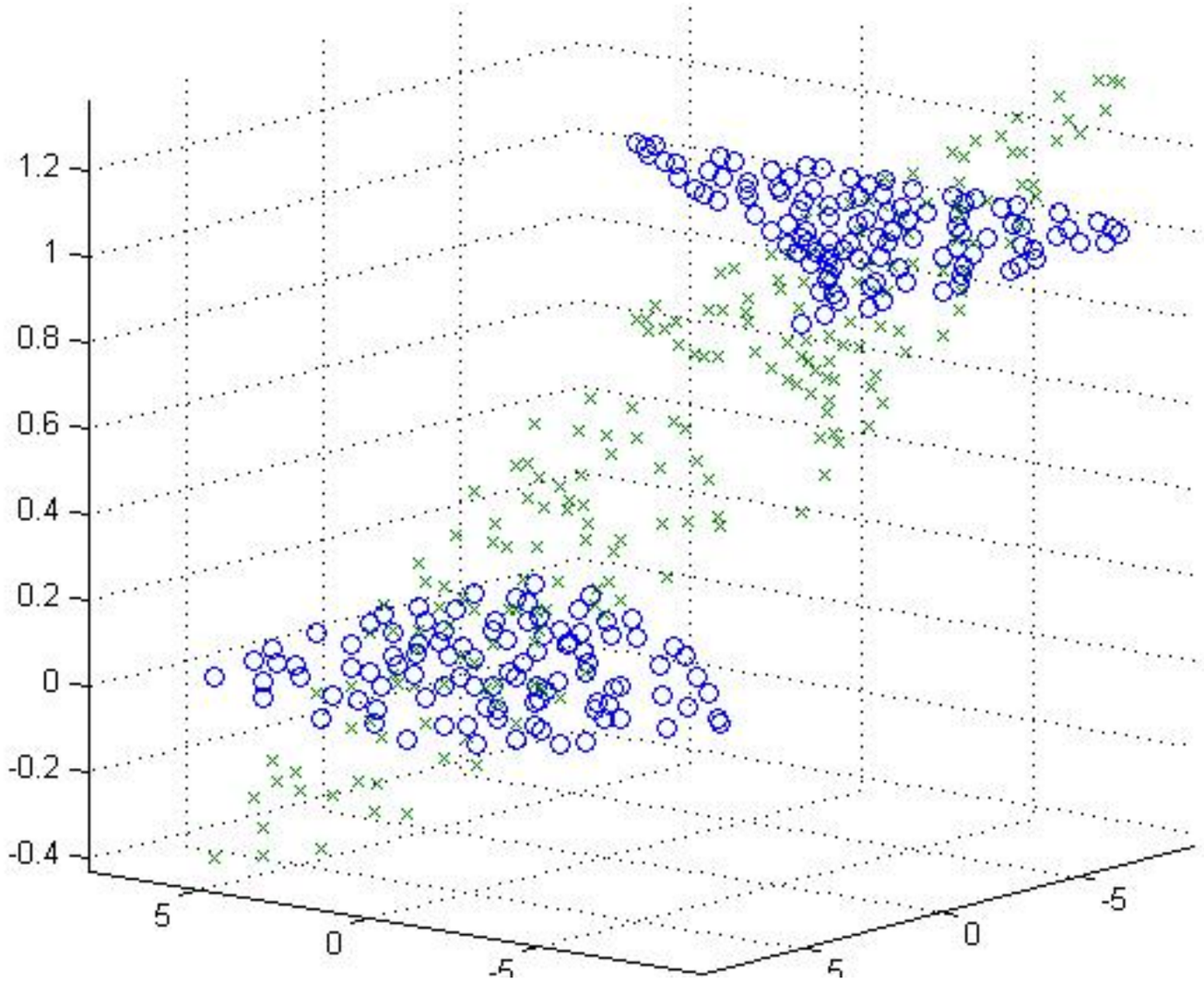
Data science, ZIF, MNZ 2, sem 1 Filip Wójcik, filip.wojcik@outlook.com

PANDAS PLOTTING

key	age	experience	salary	position
person1	20	2	2500	tester
person2	25	4	5000	tester
person3	35	7	9500	developer
person4	31	4	9000	developer

df[['experience']].plot(kind='pie')





PART 5

ANALIZA DANYCH

Data science, ZIF, MNZ 2, sem 1 Filip Wójcik, filip.wojcik@outlook.com

BIBLIOTEKI I API

- Problemy statystyczne mogą być zaadresowane w Pythonie na różne sposby, z innym rozłożeniem akcentów

- Każda biblioteka skupia się na innych aspektach i pozwala uwypuklić pewne zagadnienia

- Powinno się znać możliwości każdej z bibliotek

Data science, ZIF, MNZ 2, sem 1 Filip Wójcik, filip.wojcik@outlook.com

BIBLIOTEKI I API

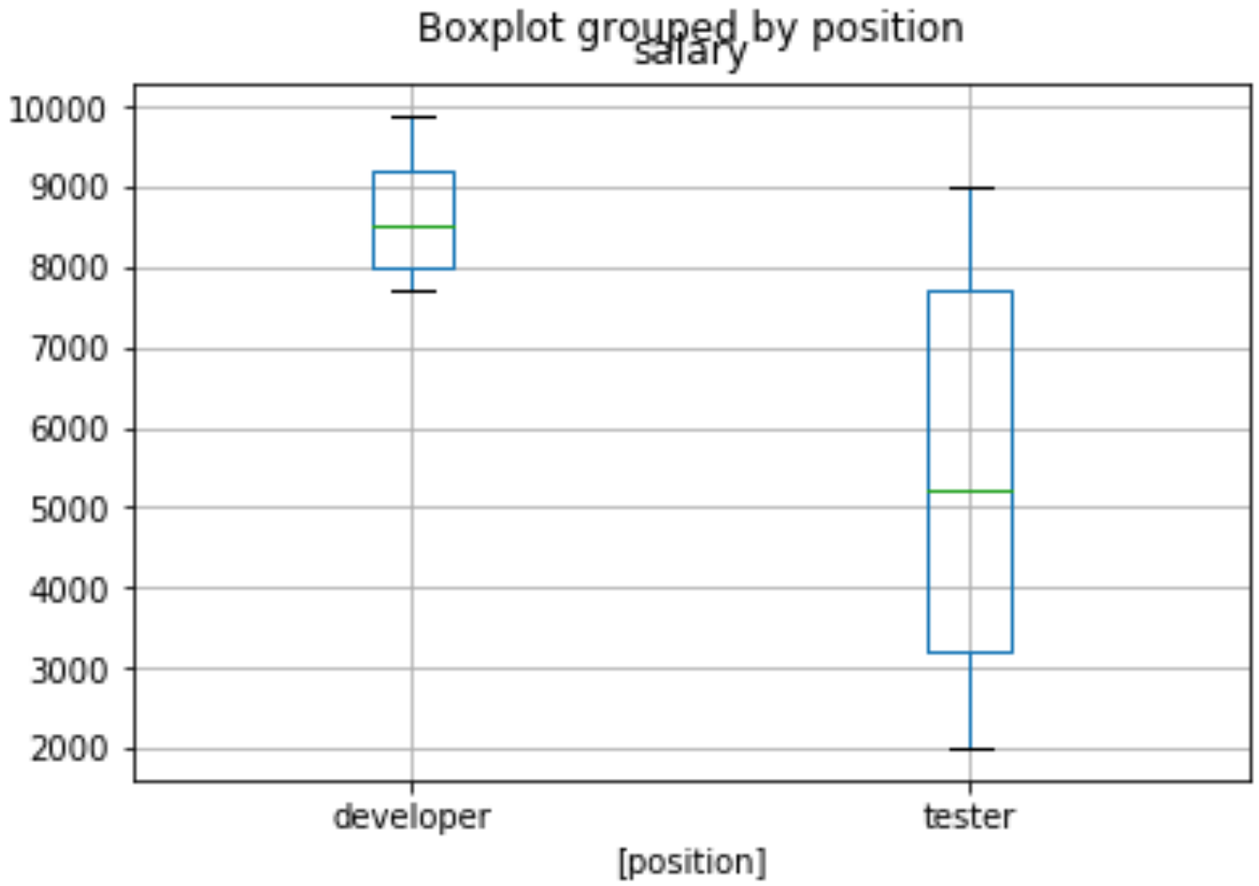
	Scipy	Statsmodels	Sklearn
Przypadki użycia	Szereoko rozumiane nauki ścisłe	Staystyka matematyczna	Uczenie maszynowe
Orientacja	Pełen przekrój operacji matematycznych z różnych działów	Testy statystyczne i modelowanie	Pełen proces uczenia maszynowego
Złożoność API	Średnie	Średnie	Bardzo proste
Dokumentacja	Bardzo dobrze utrzymana i z przykładami	Fragmentaryczna i ciężka do znalezienia	Doskonała :)

Data science, ZIF, MNZ 2, sem 1 Filip Wójcik, filip.wojcik@outlook.com

T-TEST PORÓWNYWANIA ŚREDNICH

key	position	salary
0	developer	8500
1	developer	8000
2	developer	9200
3	developer	7700
4	developer	9900
5	tester	5200
6	tester	3200
7	tester	2000
8	tester	9000
9	tester	7700

GROUP	MEAN	STD
developer	8660	896,1026727
tester	5420	2944,825971



Data science, ZIF, MNZ 2, sem 1

Filip Wójcik, filip.wojcik@outlook.com

T-TEST PORÓWNYWANIA ŚREDNICH

key	position	salary
0	developer	8500
1	developer	8000
2	developer	9200
3	developer	7700
4	developer	9900
5	tester	5200
6	tester	3200
7	tester	2000
8	tester	9000
9	tester	7700

Scipy API

```
import scipy.stats as st

st.ttest_ind(
        df2.salary[df2.position == 'developer'],
        df2.salary[df2.position == ‘tester’],

        equal_var=True
        )
```

```
Ttest_indResult(statistic=2.3536419878265598, pvalue=0.046416618432144868)
```

OPCJE KONFIGURACJI

Statsmodels API

```
from statsmodels.stats import weightstats as wst

wst.ttest_ind(
        df2.salary[df2.position == 'developer'],
        df2.salary[df2.position == 'tester'],


        alternative='two-sided',
        usevar='pooled')
```

```
(2.3536419878265598, 0.046416618432144868, 8.0)
```

Data science, ZIF, MNZ 2, sem 1 Filip Wójcik, filip.wojcik@outlook.com

T-TEST PORÓWNYWANIA ŚREDNICH

	High School	Bachelors	Masters	Ph.d.	Total
Female	60	54	46	41	201
Male	40	44	53	57	194
Total	100	98	99	98	395

Scipy API

```
import scipy.stats as st

chi2_stat, pval, deg_fr, expected_counts = st.chi2_contingency(df3)
print("Chi2 stat: {0} \nPval: {1}\nDegr.free: {2}".format(
        chi2_stat, pval, deg_fr))

"""
Chi2 stat: 8.006066246262538
Pval: 0.045886500891747214
Degr.free: 3
"""
```

Statsmodels API

```
from statsmodels.stats.proportion import proportions_chisquare

chi2_stat, pval = proportions_chisquared(df3)
print("Chi2 stat: {0} \nPval: {1}".format(
        chi2_stat, pval))
"""
Chi2 stat: 8.006066246262538
Pval: 0.045886500891747214
Degr.free: 3
“"""
```

Data science, ZIF, MNZ 2, sem 1 Filip Wójcik, filip.wojcik@outlook.com

REGRESJA LINIOWA

- Dopasowywanie linii regresji metodą najmniejszych kwadratów - Oridany Least Squares
- Klasyczna procedura statystyczna
- Można ją wykonywać na co najmniej dwa sposoby:
  - Statsmodels - zorientowanie czysto matematyczne. Bardzo dokładny wgląd w parametry i wyniki
  - Sklearn - zorientowanie na cel - dobrą predykcję. Brak szczegółów parametrycznych, ale za to szybsze i lepsze modelowanie

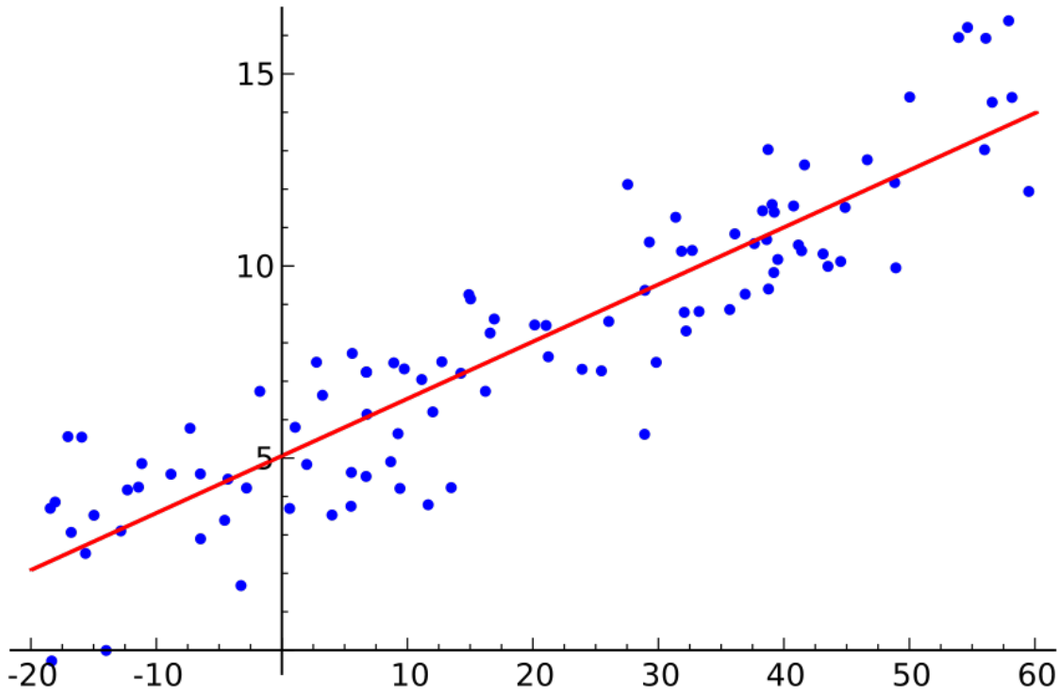
Data science, ZIF, MNZ 2, sem 1 Filip Wójcik, filip.wojcik@outlook.com

REGRESJA LINIOWA

- Główne elementy regresji:
  - Macierz X - zmienne niezależne
  - Y - zmienne zależne
  - Estymacja parametrów regresji na podstawie danych
  - Predykcja

Data science, ZIF, MNZ 2, sem 1 Filip Wójcik, filip.wojcik@outlook.com

REGRESJA LINIOWA



Założenia:

LINE

Linear relationship

Independent error terms

Normal distribution of errors

Equal variances of errors

$$y_i = \beta_0 1 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^\top \boldsymbol{\beta} + \varepsilon_i, \qquad i = 1, \ldots, n,$$

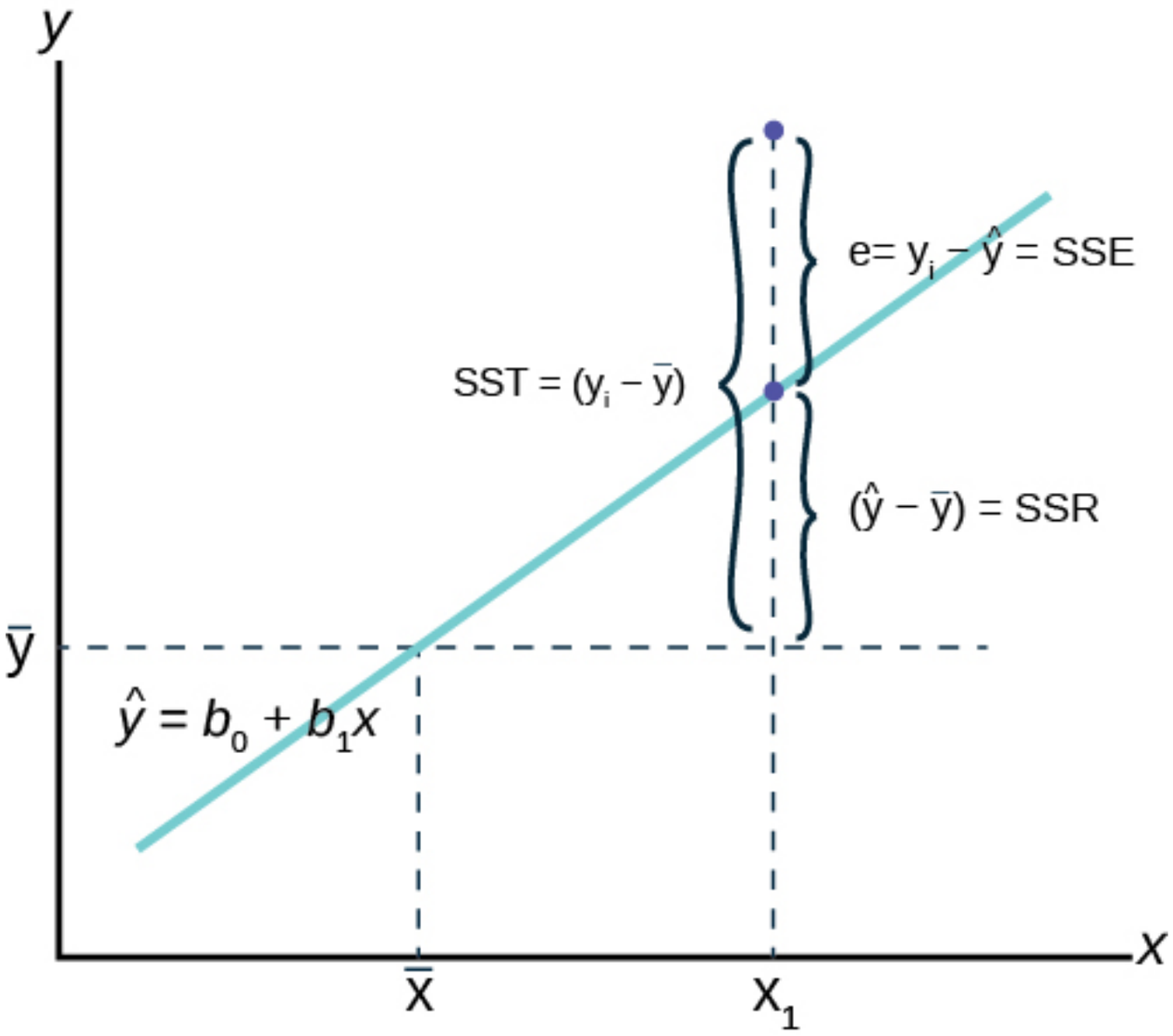
$$\mathbf{y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

$$\beta_1 = \frac{\sum(x_i - \bar{X})(Y_i - \bar{Y})}{\sum(x_i - \bar{X})^2}$$

$$\beta_0 = \bar{Y} - \beta_1 \bar{X}$$

Data science, ZIF, MNZ 2, sem 1 Filip Wójcik, filip.wojcik@outlook.com

REGRESJA LINIOWA



$$\text{SSTO} = \sum (y_i - \bar{y})$$

$$\text{SSR} = \sum (\bar{y_i} - \hat{y_i})^2$$

$$SSE = \sum (y_{ij} - \hat{y_{ij}})^2$$

$$\text{SSTO} = \text{SSR} + \text{SSE}$$

$$\sum (\text{Y}_\text{i} - \bar{Y})^2 = \sum (\hat{Y_i} - \bar{Y})^2 + \sum (Y_i - \hat{Y_i})^2$$

$$= \sum [\, (\hat{Y_i} - \bar{Y}) + (Y_i - \hat{Y_i}) \,]^2$$

$$= \sum (\hat{Y_i} - \bar{Y})^2 + (Y_i - \hat{Y_i})^2 + 2(\hat{Y_i} - \bar{Y})(Y_i - \hat{Y_i})$$

$$= \sum (\hat{Y_i} - \bar{Y})^2 + (Y_i - \hat{Y_i})^2 + 0$$

$$= \text{SSR} + \text{SSE}$$

Source: Tiobe https://stats2analytics.wordpress.com/2016/09/09/how-to-find-regression-equation/

Data science, ZIF, MNZ 2, sem 1 Filip Wójcik, filip.wojcik@outlook.com

REGRESJA LINIOWA

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

$$Y = X\beta + \varepsilon$$

X			Y
Company	Popularity	Core products count	Revenue Q1 2016 [Billion USD]
Apple	7	12	51
Alphabet	10	4	20
Microsoft	6.5	8	21

Data science, ZIF, MNZ 2, sem 1 Filip Wójcik, filip.wojcik@outlook.com

REGRESJA LINIOWA

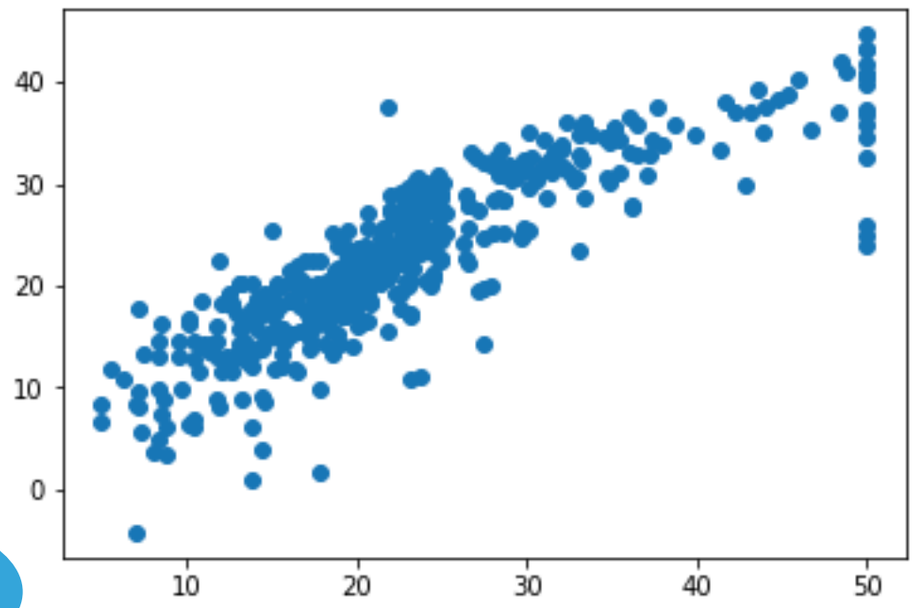
Sklearn API

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import  regression

lr = LinearRegression(normalize=True)

fitted = lr.fit(boston_df, y)

predicted = fitted.predict(boston_df)

regression.mean_squared_error(y, predicted)
>> 27.4

regression.r2_score(y, predicted)
>> 0.74

plt.plot(x=y, y=predicted)
```

UTWORZENIE OBIEKTU MODELU

PRZEWIDYWANIE NOWYCH OBSERWACJI

MANUALNE SPRAWDZANIE METRYK PREDYKCJI



Data science, ZIF, MNZ 2, sem 1

Filip Wójcik, filip.wojcik@outlook.com

REGRESJA LINIOWA

Statsmodels API

```
import statsmodels.api as stm

model = stm.OLS(
   y,
   stm.add_constant(boston_df)
   )

results = model.fit()
print(results.summary())
```

MIARY TRAFNOŚCI

ZŁOŻONOŚC MODELU

WSPÓŁCZYNNIKI MODELU I ICH ISTOTNOŚĆ

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.741
Model:                            OLS   Adj. R-squared:                  0.734
Method:                 Least Squares   F-statistic:                     108.1
Date:                Wed, 16 Aug 2017   Prob (F-statistic):          6.95e-135
Time:                        21:01:31   Log-Likelihood:                -1498.8
No. Observations:                 506   AIC:                             3026.
Df Residuals:                     492   BIC:                             3085.
Df Model:                          13
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         36.4911      5.104      7.149      0.000      26.462      46.520
CRIM          -0.1072      0.033     -3.276      0.001      -0.171      -0.043
ZN             0.0464      0.014      3.380      0.001       0.019       0.073
INDUS          0.0209      0.061      0.339      0.735      -0.100       0.142
CHAS           2.6886      0.862      3.120      0.002       0.996       4.381
NOX          -17.7958      3.821     -4.658      0.000     -25.302     -10.289
RM             3.8048      0.418      9.102      0.000       2.983       4.626
AGE            0.0008      0.013      0.057      0.955      -0.025       0.027
DIS           -1.4758      0.199     -7.398      0.000      -1.868      -1.084
RAD            0.3057      0.066      4.608      0.000       0.175       0.436
TAX           -0.0123      0.004     -3.278      0.001      -0.020      -0.005
PTRATIO       -0.9535      0.131     -7.287      0.000      -1.211      -0.696
B              0.0094      0.003      3.500      0.001       0.004       0.015
LSTAT         -0.5255      0.051    -10.366      0.000      -0.625      -0.426
==============================================================================
Omnibus:                      178.029   Durbin-Watson:                   1.078
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              782.015
Skew:                           1.521   Prob(JB):                    1.54e-170
Kurtosis:                       8.276   Cond. No.                     1.51e+04
==============================================================================
```

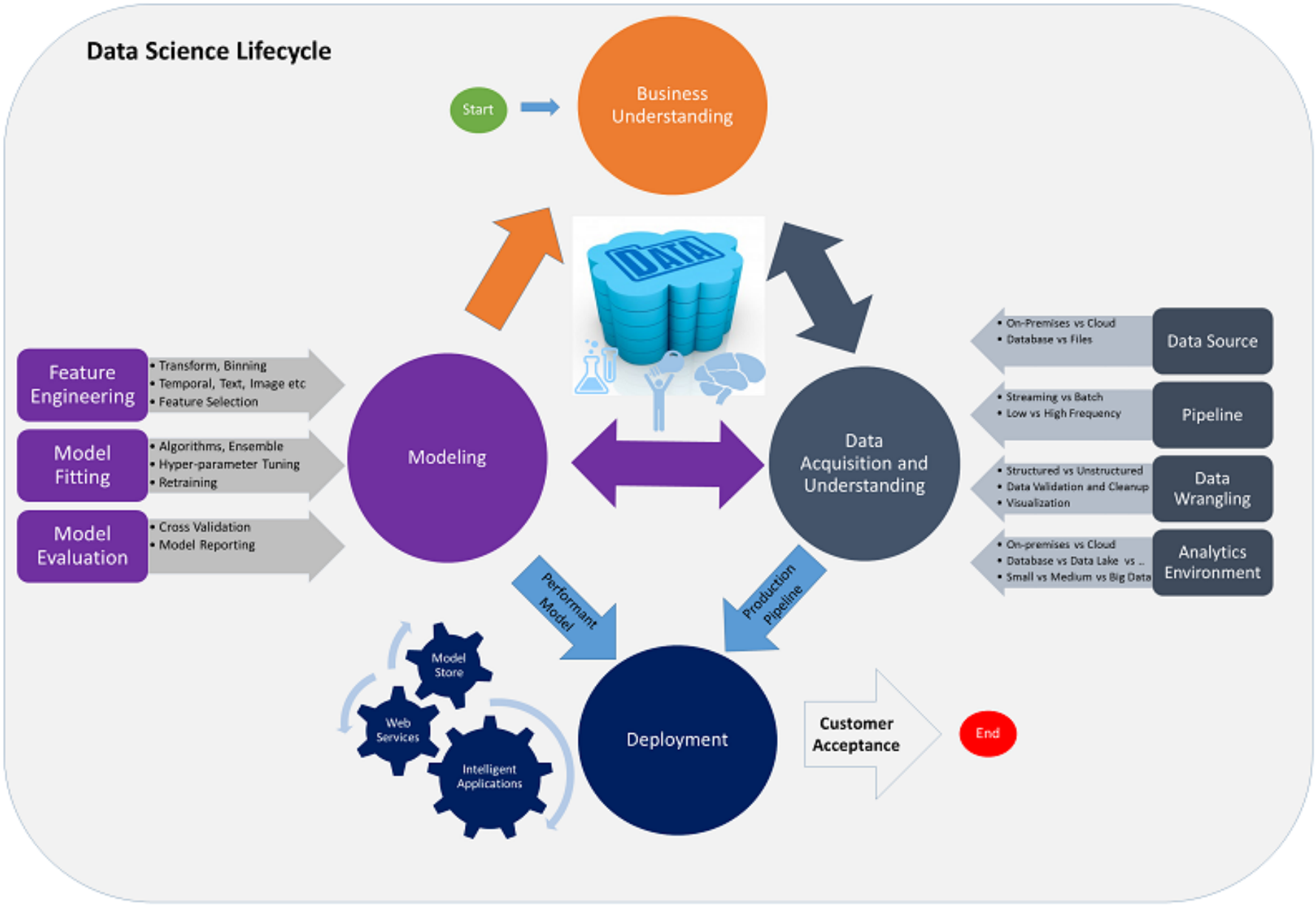
Data science, ZIF, MNZ 2, sem 1 Filip Wójcik, filip.wojcik@outlook.com

BUDOWANIE MODELI PREDYKCYJNYCH

- Sklearn definiuje standardową procedurę budowania modeli
- Kolejne kroki:
  - Podział danych na część uczącą i testową
  - Szkolenie modelu na danych uczących
  - Sprawdzenie trafności predykcji na danych testowych
  - Powtórka procesu, jeśli wyniki nie są zadowalające

Filip Wójcik, filip.wojcik@outlook.com

PROCES BUDOWANIA MODELI PREDYKCYJNYCH



Source: https://docs.microsoft.com/en-us/azure/machine-learning/data-science-process-overview

Dziękuje za uwagę!

Dziękuję za uwagę!