FILIP WÓJCIK

SENIOR DATA SCIENTIST

FILIP.WOJCIK@OUTLOOK.COM

DATA ANALYSIS IN PYTHON

Filip Wójcik, filip.wojcik@outlook.com

AGENDA

1. What is python and why it's so popular in data science?
2. Getting started with tools:
   - ANACONDA PACKAGE
   - JUPYTER NOTEBOOKS
   - PYCHARM
3. Python basics
4. Intro to PANDAS & data frames - basic data structures
   - Reading data from different sources
   - Data filtering and projection
   - Data aggregation
   - Calculated columns and transformations
   - Dealing with missing values
   - Using custom functions
   - Joining data frames
   - Split - Apply - Combine patterns
   - Pivoting tables
5. Intro to data visualizations
   - Different visualization engines in Python
   - Pandas built-in visualization package
   - Matplotlib & friends
   - Bokeh - user-friendly interactive dashboards
6. Basic statistical analysis
   - StatModels vs Scipy.stats
   - Basic mathematical statistics: hypothesis testing
   - T-tests, comparing variances
   - Logistic regression
7. Kaggle case study: Titanic disaster
   - Predicting, who will survive the disaster
   - Exploratory analysis
   - Dealing with missing observations
   - Hypothesis testing and inference

PART 1

INTRO

Filip Wójcik, filip.wojcik@outlook.com

WHAT IS PYTHON AND WHY IT'S SO POPULAR

- Developed in 1991
- Key principles:
  - object-oriented and functional as well
  - readability
  - clear syntax
  - fast to learn + fast to code
- Open - source language
- Very active community
- Universal and powerful:
  - Data science & analysis
  - Web applications
  - Games
  - Enterprise applications
  - Different systems compatibility

Data Science Summer School 2017
Filip Wójcik, filip.wojcik@outlook.com

WHAT IS PYTHON AND WHY IT'S SO POPULAR

Aug 2017	Aug 2016	Change	Programming Language	Ratings	Change
1	1		Java	12.961%	-6.05%
2	2		C	6.477%	-4.83%
3	3		C++	5.550%	-0.25%
4	4		C#	4.195%	-0.71%
5	5		Python	3.692%	-0.71%
6	8		Visual Basic .NET	2.569%	+0.05%
7	6		PHP	2.293%	-0.88%
8	7		JavaScript	2.098%	-0.61%
9	9		Perl	1.995%	-0.52%
10	12		Ruby	1.965%	-0.31%
11	14		Swift	1.825%	-0.16%
12	11		Delphi/Object Pascal	1.825%	-0.45%
13	13		Visual Basic	1.809%	-0.24%
14	10		Assembly language	1.805%	-0.56%
15	17		R	1.766%	+0.16%
16	20		Go	1.645%	+0.37%
17	18		MATLAB	1.619%	+0.08%
18	15		Objective-C	1.505%	-0.38%
19	22		Scratch	1.481%	+0.43%
20	26		Dart	1.273%	+0.30%

Source: Tiobe Index, https://www.tiobe.com/tiobe-index/

WHAT IS PYTHON AND WHY IT'S SO POPULAR

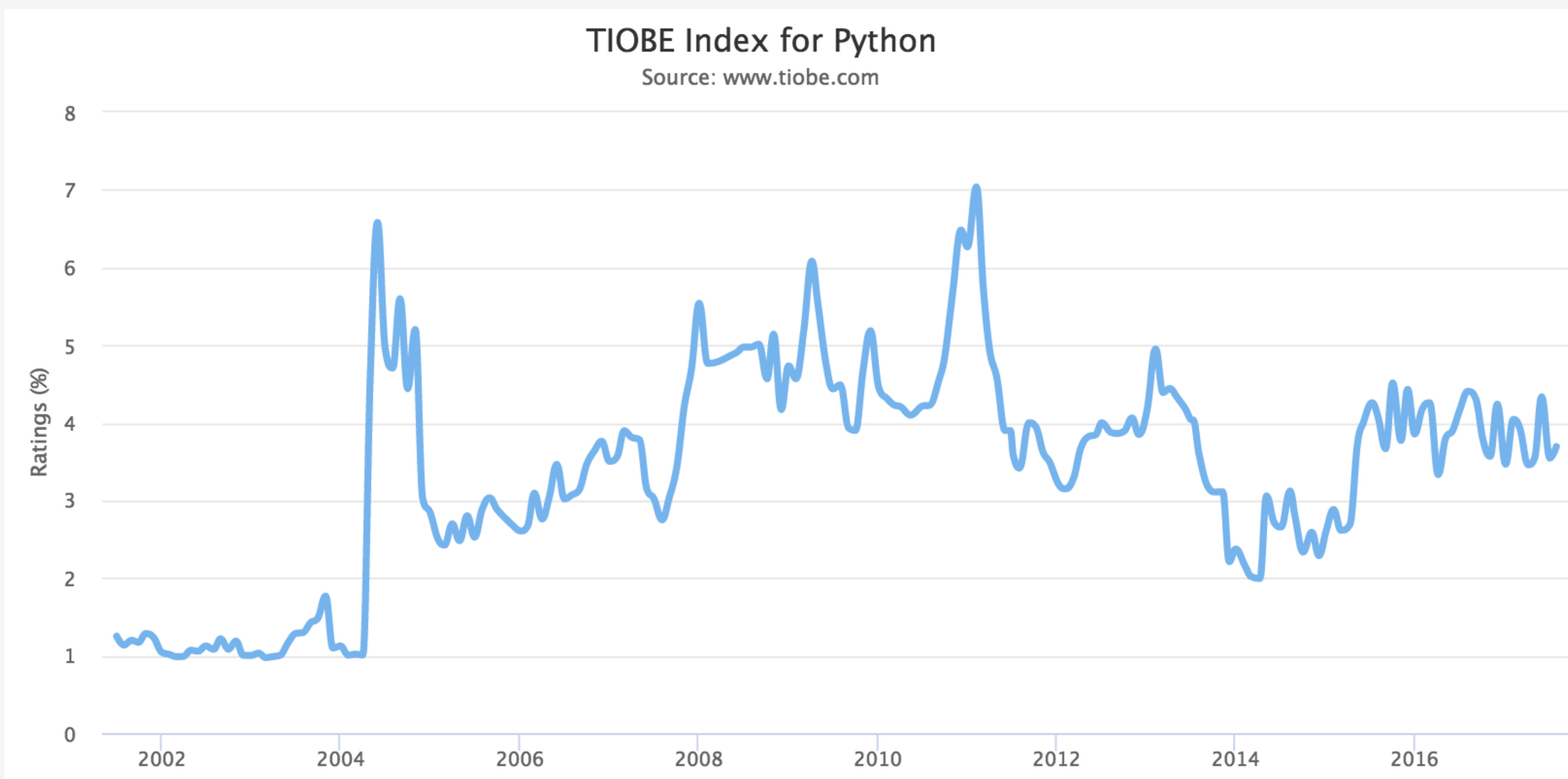
The Python Programming Language

Some information about Python:

Highest Position (since 2001): #4 in Jul 2017

Lowest Position (since 2001): #13 in Feb 2003

Language of the Year: 2007, 2010



Source: Tiobe Index, https://www.tiobe.com/tiobe-index/

 Filip Wójcik, filip.wojcik@outlook.com

KEY PYTHON TOOLS FOR DATA SCIENCE

Pandas

1. General data manipulation package
2. Data reading & importing

SciPy

1. Scientific package
2. Different mathematical functions for specific use cases (statistics, electronics, physics, etc.)
3. Wide variety of functions

Scikit-learn

1. Core machine learning library
2. A lot of different, useful algorithms
3. Clear API and interfaces
4. Considered to be state-of-the-art tool

Statsmodels

1. More "traditional" statistics-oriented than scikit-learn
2. Used mostly for statistical inference
3. API close to R - a lot of inspiration from statistical

Matplotlib

1. Core visualization package
2. One of the most useful ones in plotting data

Numpy

1. Part of the spicy package
2. Core matrix & vectors library
3. Numerical optimization engines
4. Very efficient and fast

PART 2

GETTING STARTED WITH TOOLS

ANACONDA - ULTIMATE SCIENTIFIC PYTHON ECOSYSTEM

- Developed by Continuum Analytics
- Open source, free Python and R distribution
- Multi-platform (Mac OS/Win/Linux)
- Integrated with core analytical tools:
  - Jupyter
  - Key libraries (sklearn/scipy/numpy/pandas/etc.)
  - Built-in Spyder IDE & Rstudio

Filip Wójcik, filip.wojcik@outlook.com

JUPYTER NOTEBOOKS - COMPLEX DATA SCIENCE ENVIRONMENT

- Based on older IPython notebooks
- Complex environment for Data Scientists:
  - visualizations
  - combines code + documentation
- Works with Python + R
- Very simple code assist + intellisense
- Available as an online tool - e.g. kaggle.com, community.databricks.com

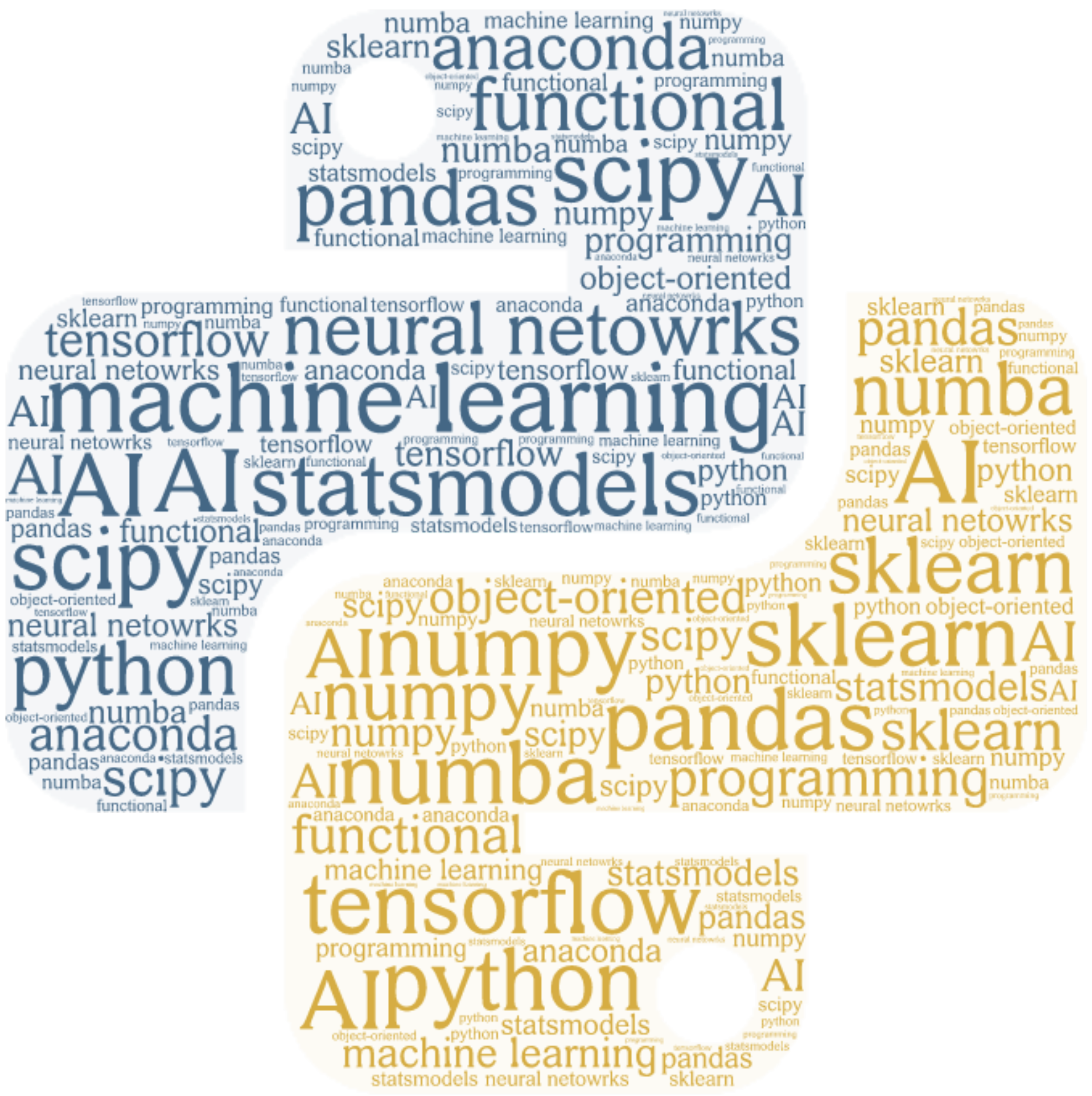
PYCHARM – COMPLEX PROGRAMMING IDE

- Full programming IDE
- Community (free)/Commercial editions
- Designed to embed different features:
  - for web developers
  - for applications developers
  - for data scientists: notebooks/visualizations/graphs

Filip Wójcik, filip.wojcik@outlook.com

SETUP TIME =)

<title>code ninja</title>



PART 3

PYTHON BASICS

BASICS - DATA TYPES & DECLARATIONS

- Declaring variables

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

- Specyfing type (3.6):

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

DATA TYPE	EXAMPLE

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`

BASICS - COLLECTIONS

- Operations on lists:

Declaration	`my_list = ['a', 'b', 'c', 'd', 'e']`	['a', 'b', 'c']

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

Appending (inplace)	`my_list.append('d')`	['a', 'b', 'c', 'd']
Concatenation	`my_list + ['e']`	['a', 'b', 'c', 'd', 'e']
Checking existence	`'a' in my_list` `'xxx' in my list`	True False

BASICS – COLLECTIONS

- Operations on dictionaries:

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

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

Adding element	`my_dict['keyX'] = 111`	`{'key1': 10, 'key2': 20, 'key3': 30, 'keyX': 111}`
Checking existence	`'key1' in my_dict` `'key1' in my_dict`	True False

BASICS – BASIC OPERATIONS

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	a + b = 31
- Subtraction	Subtracts right hand operand from left hand operand.	a – b = -11
* Multiplication	Multiplies values on either side of the operator	a * b = 210
/ Division	Divides left hand operand by right hand operand	b / a = 2.1
% Modulus	Divides left hand operand by right hand operand and returns remainder	b = 5 a = 2 b % a = 1
** Exponent	Performs exponential (power) calculation on operators	a**b =10 to the power 20

Source: Tutorialspoint, Python3, https://www.tutorialspoint.com/python3/arithmetic_operators_example.htm

BASICS – CONTROL STRUCTURES

CONDITIONALS: 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")
```

CONDITIONALS: 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")
```

BASICS – CONTROL STRUCTURES

ITERATION: simple loops

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

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

ITERATION: loop 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("----")
```

BASICS – CONTROL STRUCTURES

ITERATION: keys in dictionary

```
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)
```

BASICS - FUNCTIONS

- “First class citizens” - behave like objects
- Can be passed around like variables
- Accept arguments and return values
- WATCH OUT: mutable object (lists, instances) are passed to function **by reference -** are modified inside!

DEF KEYWORD

FUNCTION NAME

ARGUMENTS (ANY TYPE)

RETURN KEYWORD + RETURN VALUE

```
def my_function(arg1, arg2):
    x = arg1 + arg2
    return x
```

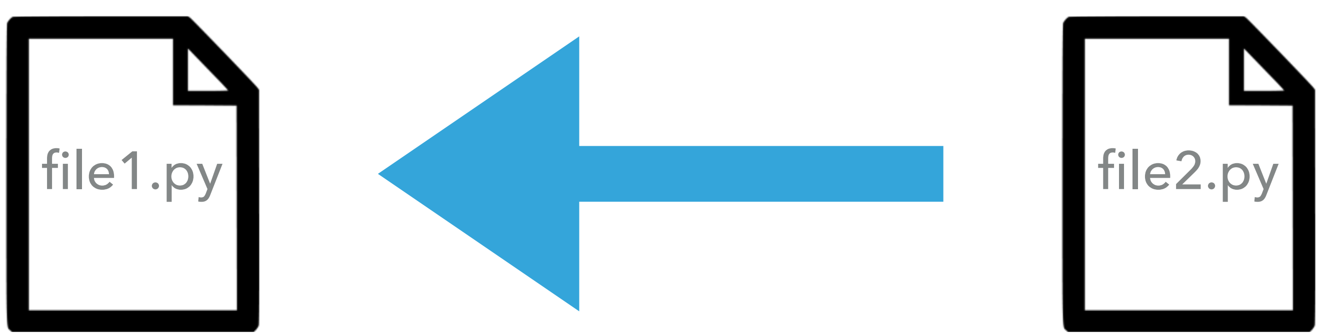
FUNCTION BODY WITH INTENDATION

BASICS - DATA TYPES & DECLARATIONS

- Organizing programs using indents not braces (Java/C#):

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

- By default python files are modules, which can be referenced:




```
import file1 as f1

f1.add_numbers(1, 2)


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

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

Filip Wójcik, filip.wojcik@outlook.com

CODING TIME =)

<title>code ninja</title>

PART 4

INTRO TO PANDAS

Filip Wójcik, filip.wojcik@outlook.com

DATA FRAME - BASIC DATA STRUCTURE

- Data frame - basic data structure for pandas and other analytical libraries
- SQL-like table or Excel Sheet
- Columnary data structure - optimized to store data in columns:
  - easier to compute per-column statistics
  - easier to find correlations between different attributes

Filip Wójcik, filip.wojcik@outlook.com

DATA FRAME - BASIC DATA STRUCTURE

INDEX COLUMN: PSEUDO-PRIMARY KEY

TEXT/STRING ATTRIBUTES

NUMERICAL ATTRIBUTE

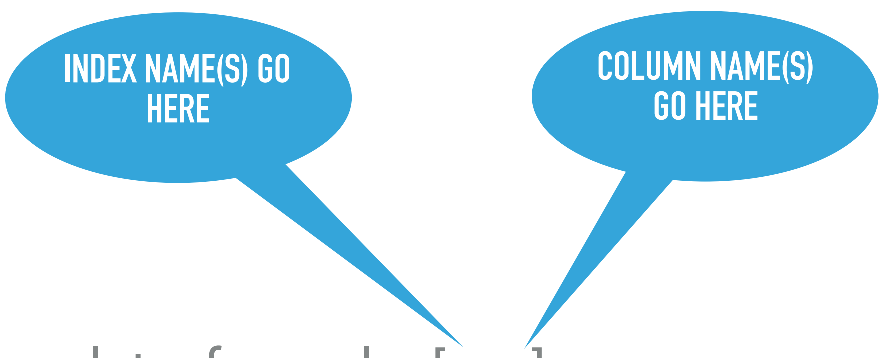
DATE -TIME ATTRIBUTE

1. Index is used to refer to particular rows.
2. It can be anything - string/number/date
3. It is important to know if we're referring by index or location

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

Filip Wójcik, filip.wojcik@outlook.com

DATA FRAME - CALL BY INDEX NAME



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

Filip Wójcik, filip.wojcik@outlook.com

DATA FRAME - CALL BY POSITION (ZERO-INDEXED)

ROW NUMBER(S) GO HERE

COLUMN NUMBER(S) GO HERE

```
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

Filip Wójcik, filip.wojcik@outlook.com

DATA FRAME – CALL BY SLICER

ROW(S) SLICERS GO HERE

COLUMN SLICERS(S) GO HERE

FROM:TO
FROM – INCLUSIVE
TO – EXCLUSIVE

```
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

Filip Wójcik, filip.wojcik@outlook.com

DATA FRAME - GET COLUMN SHORTCUT

COLUMN NAME(S) GOES HERE

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

Filip Wójcik, filip.wojcik@outlook.com

DATA FRAME - PER COLUMN OPERATIONS

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 – PER COLUMN OPERATIONS

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

Filip Wójcik, filip.wojcik@outlook.com

DATA FRAME – PER COLUMN/PER ROW OPERATIONS

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

Filip Wójcik, filip.wojcik@outlook.com

DATA FRAME - PER COLUMN/PER ROW OPERATIONS

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

Filip Wójcik, filip.wojcik@outlook.com

DATA FRAME – PER COLUMN/PER ROW OPERATIONS

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

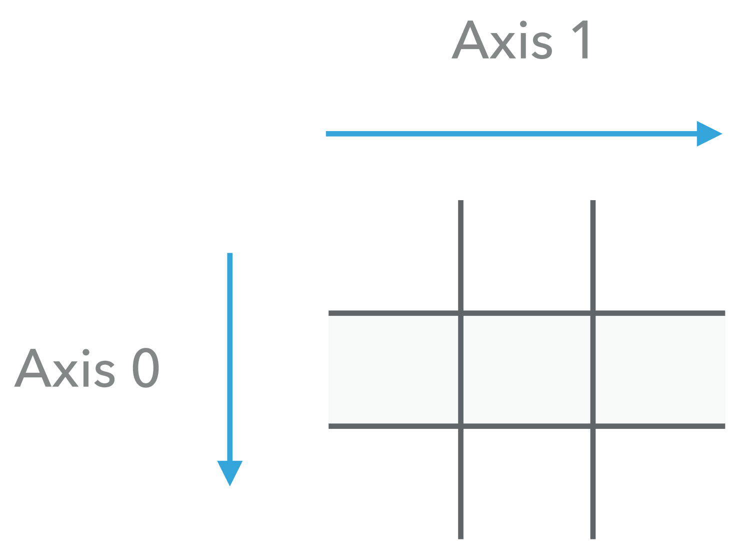
```
my_data_frame.apply(lambda row: row['name'] + ' ' + row['surname'],  axis=1)
```

Filip Wójcik, filip.wojcik@outlook.com

DATA FRAME - PER COLUMN/PER ROW OPERATIONS

Tips & tricks how to use per column/per row operations:

- Use per axis 1 (per row) operations when:
  - you want to calculate MULTIPLE COLUMNS INTERACTION
  - your function depends (needs to "know") values of multiple columns



- Use per axis 2 (per column) operations when:
  - you want to calculate something nonstandard - not included in a library
  - you want to drop columns

Filip Wójcik, filip.wojcik@outlook.com

DATA FRAME - JOINING FRAMES

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

Keep all rows from "left" data frame

and try to match rows on the right

key	c1
a	1
c	7
d	10

key	c2
a	20
b	30
c	40

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

INNER

Keep only matching rows from both sides

key	c1
a	1
c	7
d	10

key	c2
a	20
b	30
c	40

key	c1	c2
a	1	20
c	7	30

RIGHT

Keep all rows from "right" data frame

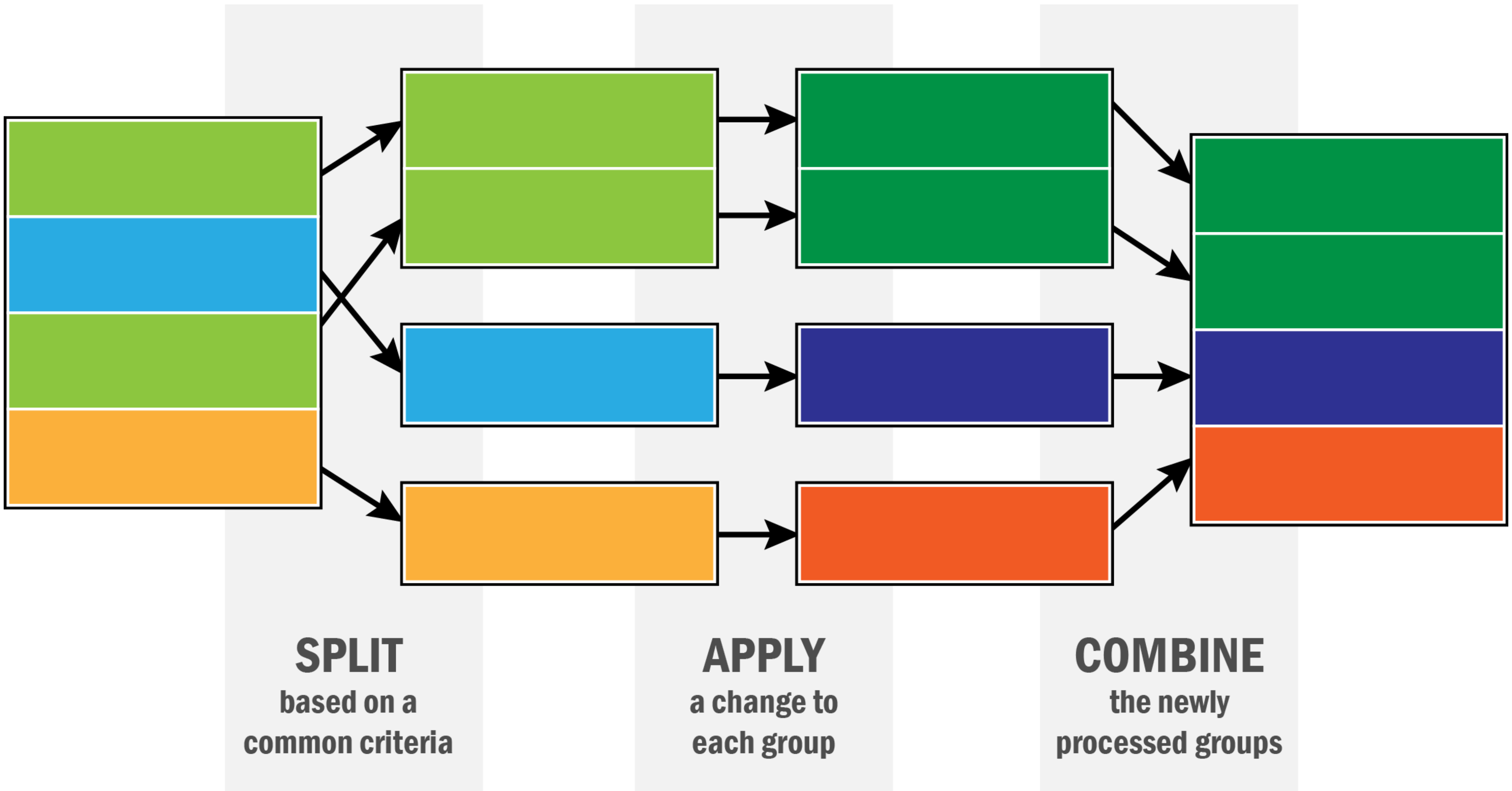
and try to match rows on the left

key	c1
a	1
c	7
d	10

key	c2
a	20
b	30
c	40

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

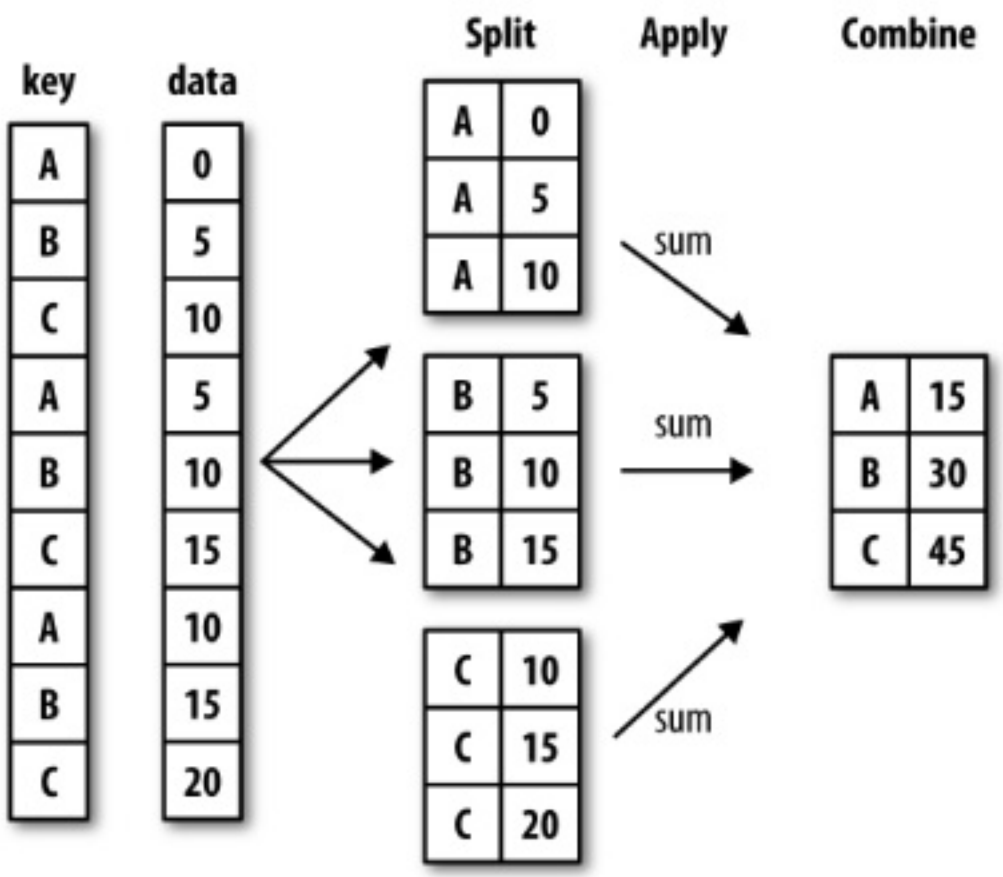
SPLIT-APPLY-COMBINE PATTERN



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

Data Science Summer School 2017 Filip Wójcik, filip.wojcik@outlook.com

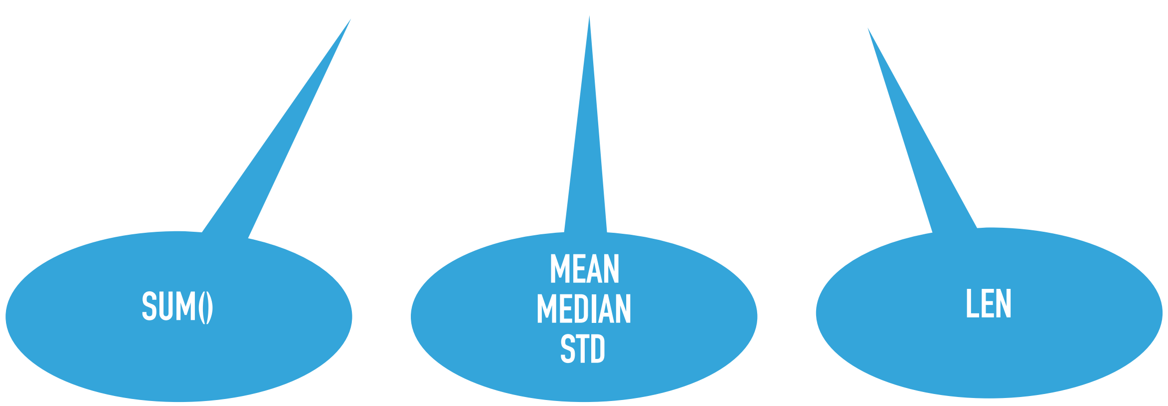
SPLIT-APPLY-COMBINE PATTERN: COMMON AGGREGATIONS



my_data_frame["VALUE_COLUMN_HERE"]

.groupby(my_data_frame["KEY_COLUMN_HERE"])

.AGGREGATION_FUNCTION_HERE

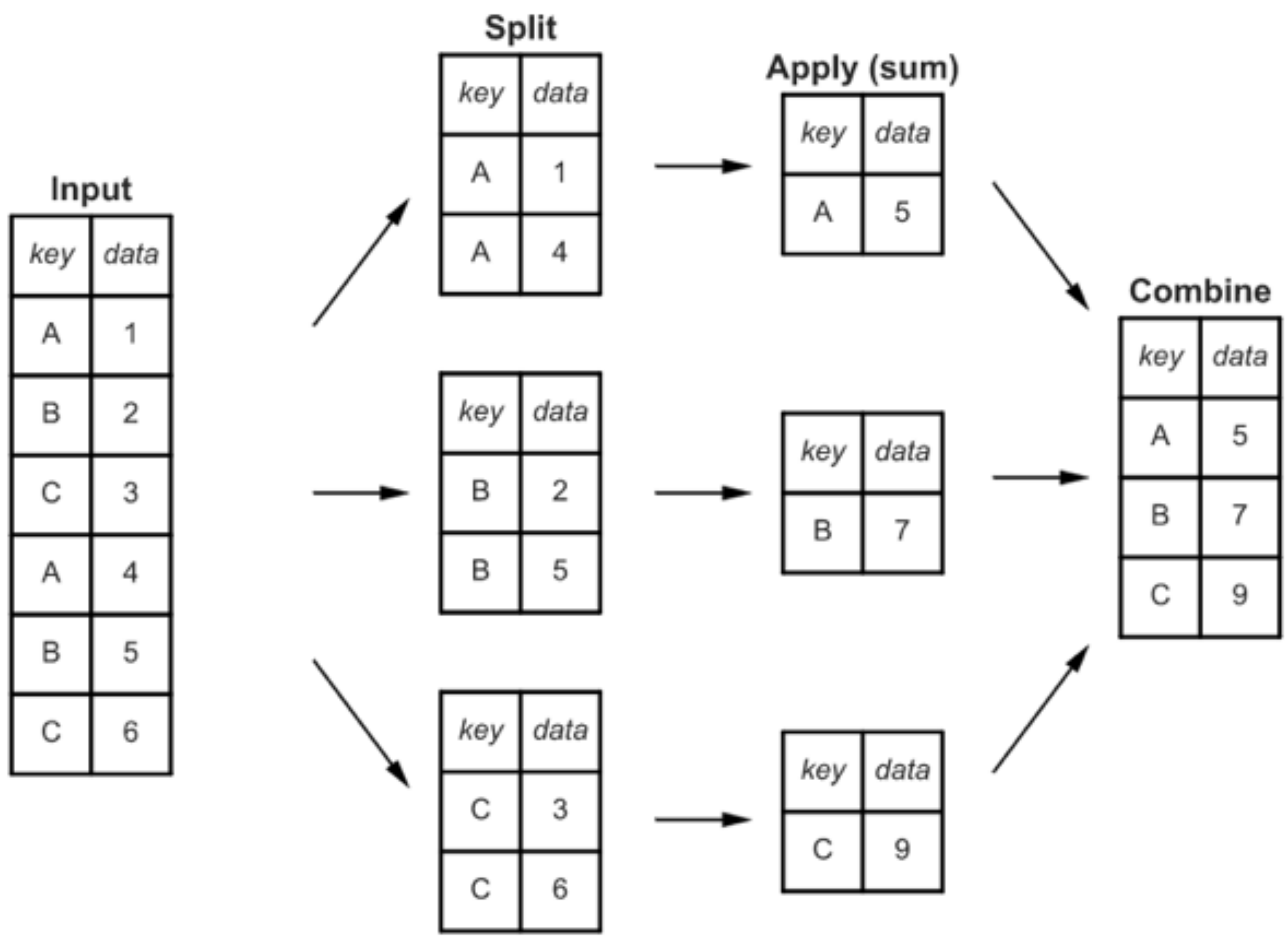


Source: William McKinney, Python for Data Analysis, 2nd Edition

Filip Wójcik, filip.wojcik@outlook.com

SPLIT-APPLY-COMBINE PATTERN: COMMON AGGREGATIONS

input_df["data"].groupby('key").sum()



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

Filip Wójcik, filip.wojcik@outlook.com

SPLIT-APPLY-COMBINE PATTERN: CUSTOM AGGREGATIONS

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

SUM()

LEN

MEAN
MEDIAN
STD

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

Filip Wójcik, filip.wojcik@outlook.com

SPLIT-APPLY-COMBINE PATTERN: MULTIPLE AGGREGATIONS

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

PIVOTING TABLES

- Pivot tables concept in Pandas is exactly the same as in MS Excel or databases
- Shifting rows/columns and values
- Allows to do multiple **drill downs** or data projections - from different perspectives
- Best explained by example

Data Science Summer School 2017

Filip Wójcik, filip.wojcik@outlook.com

PIVOTING TABLES

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

Task: count passengers who survived, and group them according to their class and sex

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

        aggfunc=np.sum)
```

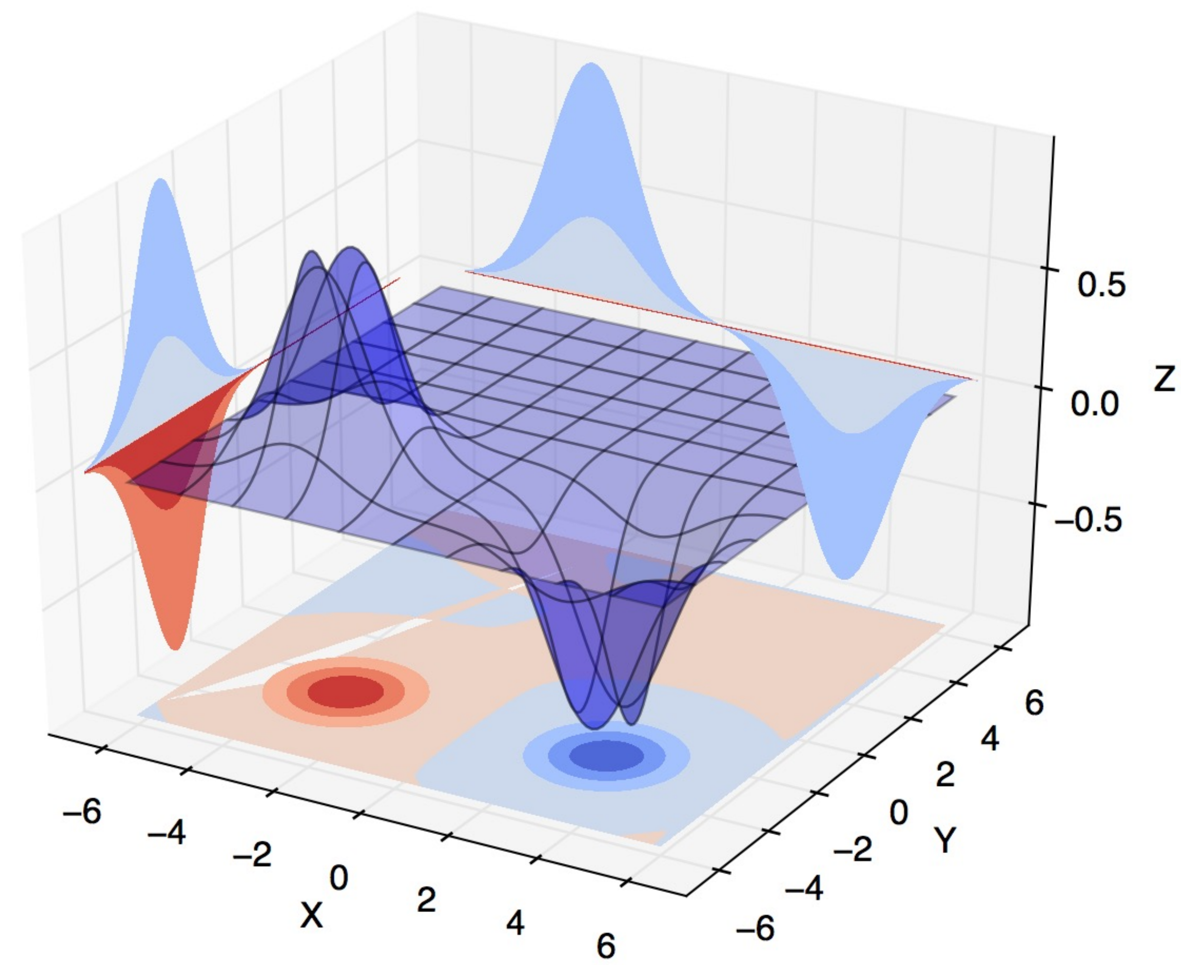
aggfunc = func(Values)

Columns

Index

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

CODING TIME =)



PART 5

DATA VISUALIZATIONS

ENGINES AND APIS

- There are different plotting engines in Python
- Some of them are simple, to be used as "sketches", some are more sophisticated
- You don't have to memorize exact specifications of each engine:
  - each engine capabilities
  - where to find good reference & documentation

Filip Wójcik, filip.wojcik@outlook.com

ENGINES AND APIS

	PANDAS PLOTTNIG API	MATPLOTLIB
Type	Built-in pandas	External library
Use cases	Data frame visualiztions	General-scientific library
Complexity	Very simple	Complex
Extensibility	Small	Large

Filip Wójcik, filip.wojcik@outlook.com

PANDAS PLOTTING ENGINE

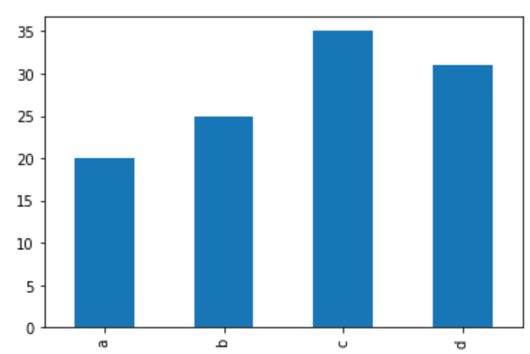
- Based on a data frame columns - can plot only numerical values
- Selecting columns to be visualized
- Number of predefined plots and charts
- User can make some simple customizations to the plot

PANDAS PLOTTING ENGINE

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

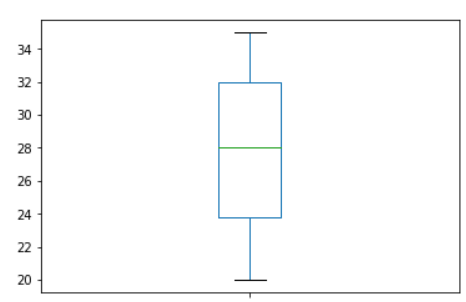
kind

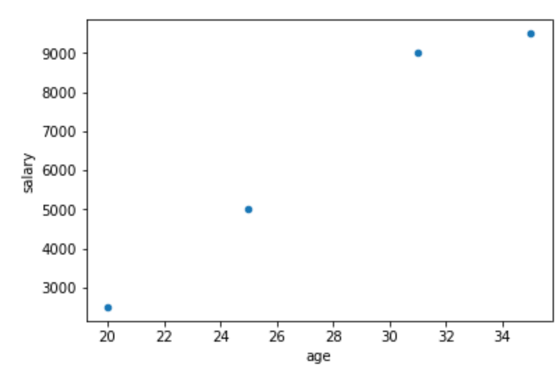
- bar
- histogram
- box
- scatter
- pie

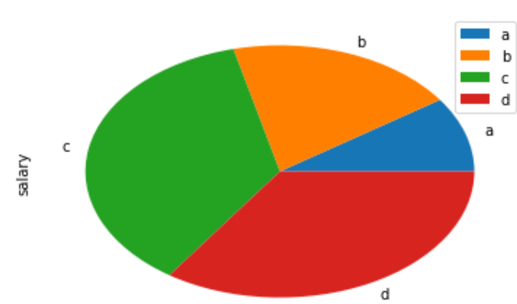


title

labels



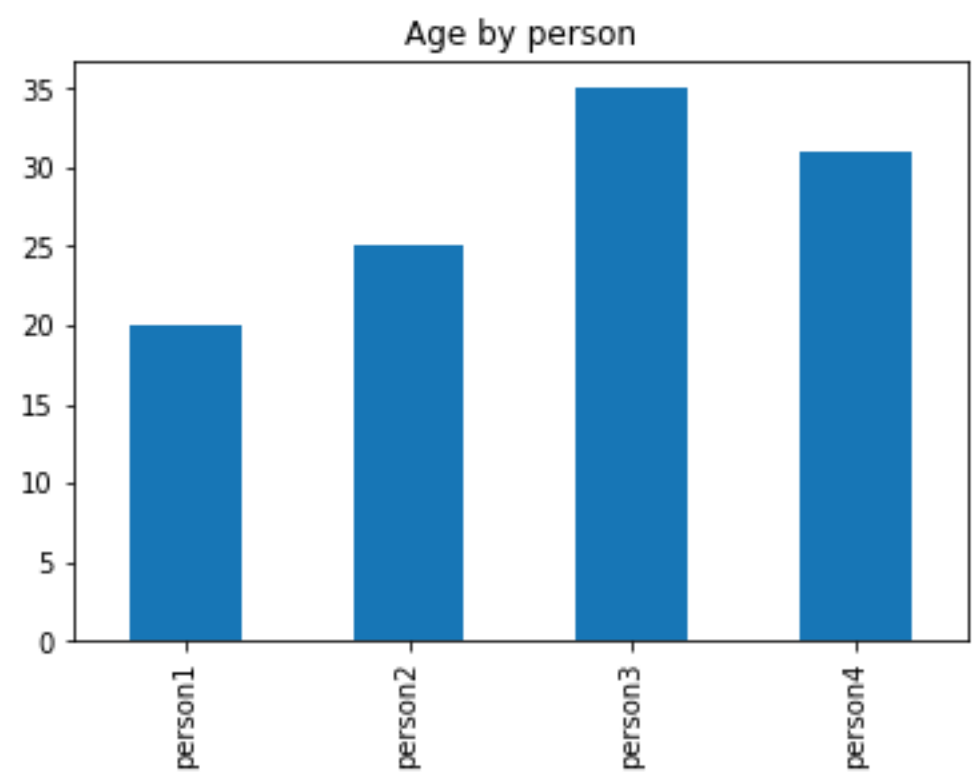




PANDAS PLOTTING ENGINE

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

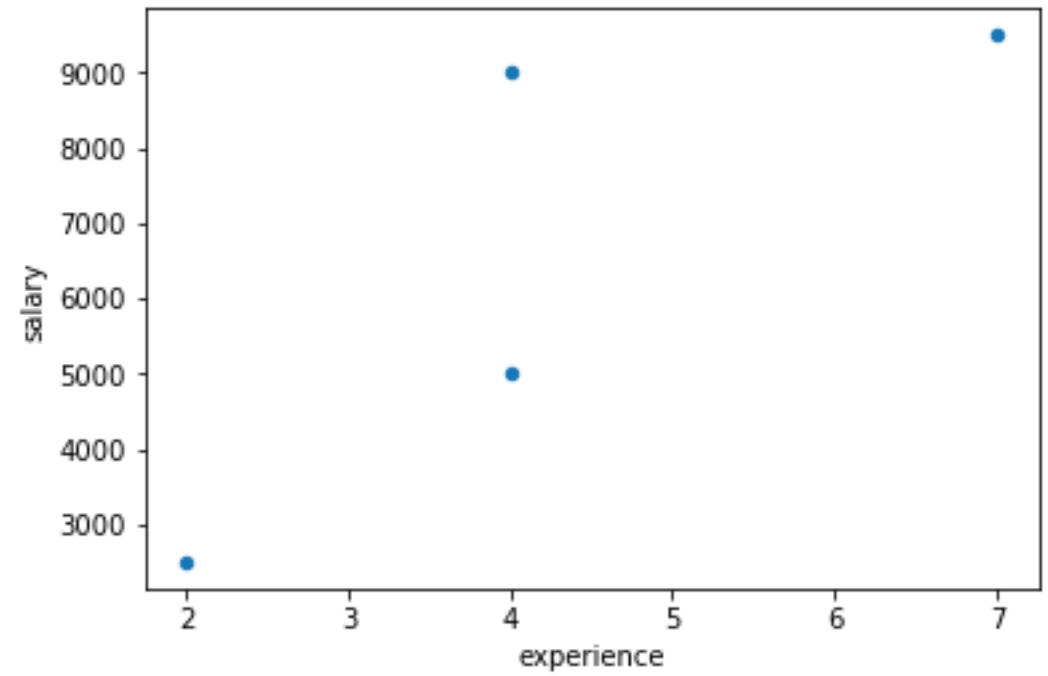


Filip Wójcik, filip.wojcik@outlook.com

PANDAS PLOTTING ENGINE

```
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

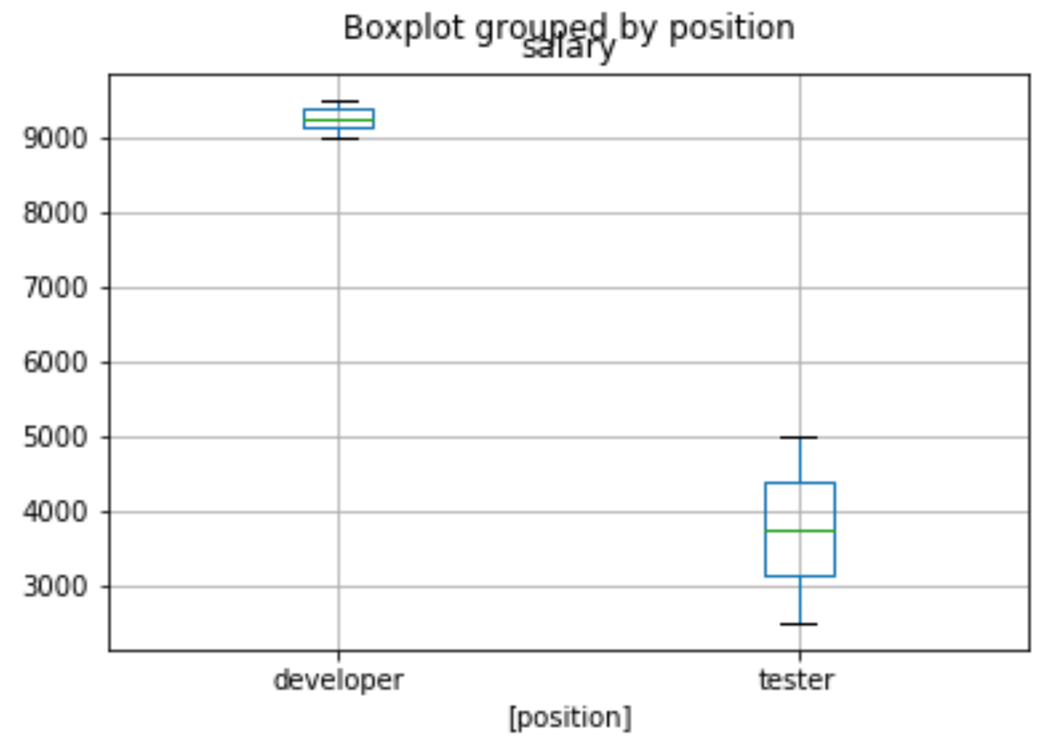


Filip Wójcik, filip.wojcik@outlook.com

PANDAS PLOTTING ENGINE

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

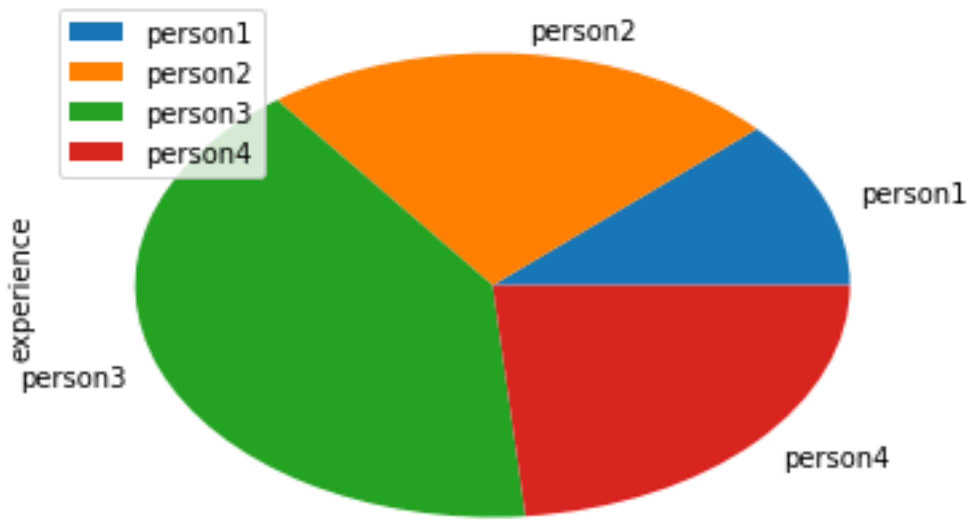
key	age	experience	salary	position
person1	20	2	2500	tester
person2	25	4	5000	tester
person3	35	7	9500	developer
person4	31	4	9000	developer

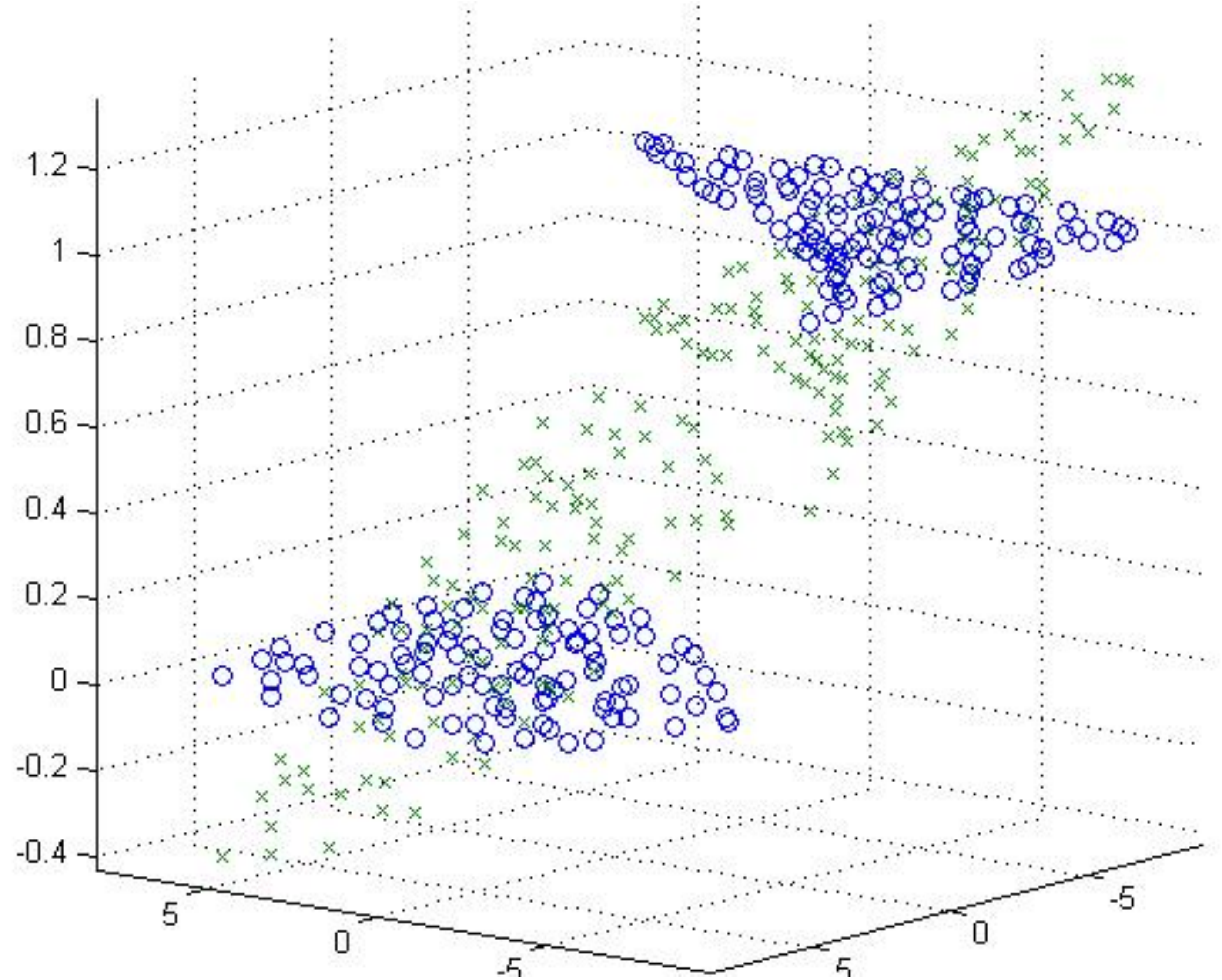


PANDAS PLOTTING ENGINE

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

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





PART 6

DATA ANALYSIS

Filip Wójcik, filip.wojcik@outlook.com

LIBRARIES AND APIS

- Statistical problems can be addressed in Python using many different libraries
- Each of this libraries concentrates on different aspects of the analysis
- One should know which tool to use in what kind of situation and use case

Filip Wójcik, filip.wojcik@outlook.com

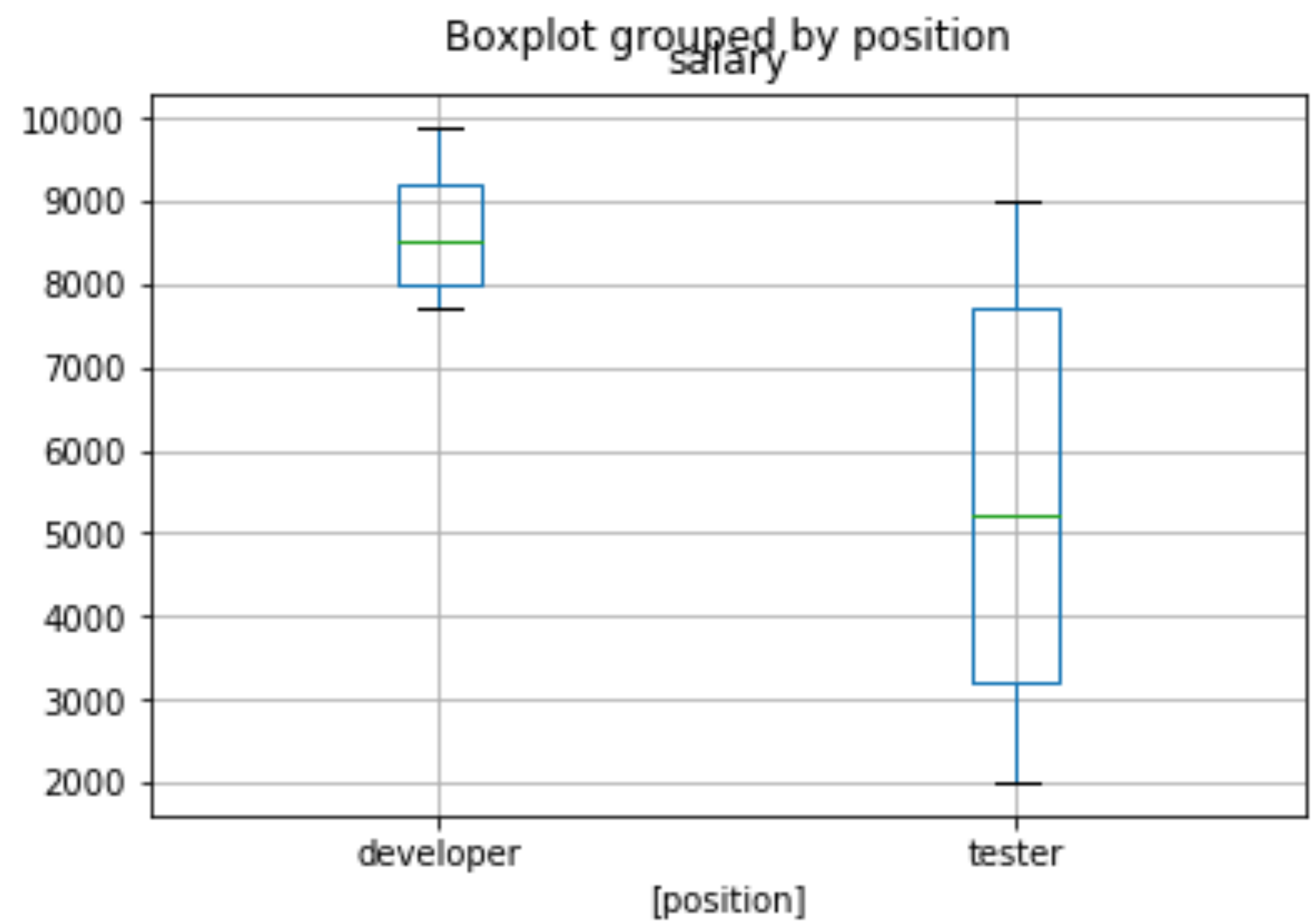
LIBRARIES AND APIS

	Scipy	Statsmodels	Sklearn
Use cases	General scientifical research (math/stats/ physics)	Mathematical statistics	Machine learning
Orientation	Elementary operations/ distributions	Statistical tests	Full machine learning process
Complexity of API	Moderate	Moderate	Simple
Documentation	Well-maintatined and precise	Fragmentary and hard to find	Very clear

T-TEST COMPARING MEANS

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



Filip Wójcik, filip.wojcik@outlook.com

T-TEST COMPARING MEANS

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

STATISTICAL TEST CONFIGURATION OPTIONS

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)
```

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)
```

T-TEST COMPARING MEANS

	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: 0.0458865008917472l4
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
"""
```

LINEAR REGRESSION

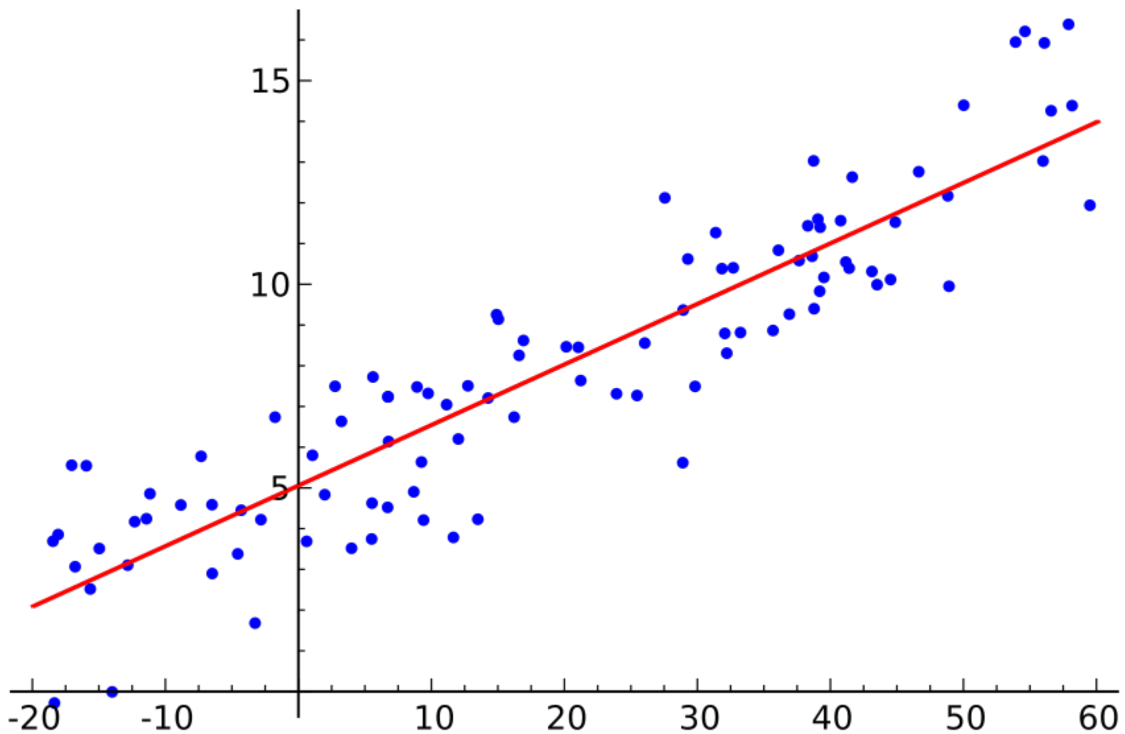
- Fitting linear model to match the data - Oridany Least Squares analysis
- "Classical" statistical procedure - well known, and well described
- It can be approached in two ways:
  - Statsmodels - more mathematically oriented. Detailed insights into fitted line properties
  - Sklearn - more task/result oriented approach. Easy to train and give results

Filip Wójcik, filip.wojcik@outlook.com

LINEAR REGRESSION

- Linear regression (OLS) in three bullet-points:
  - Take x matrix - independent features & y - expected values
  - Estimate regression line coefficients, that try to summarize the data
  - Calculate predicted y-values

LINEAR REGRESSION



Assumptions:

LINE

Linear relationship

Independent error terms

Normal distribution of errors

Equal variances of errors

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

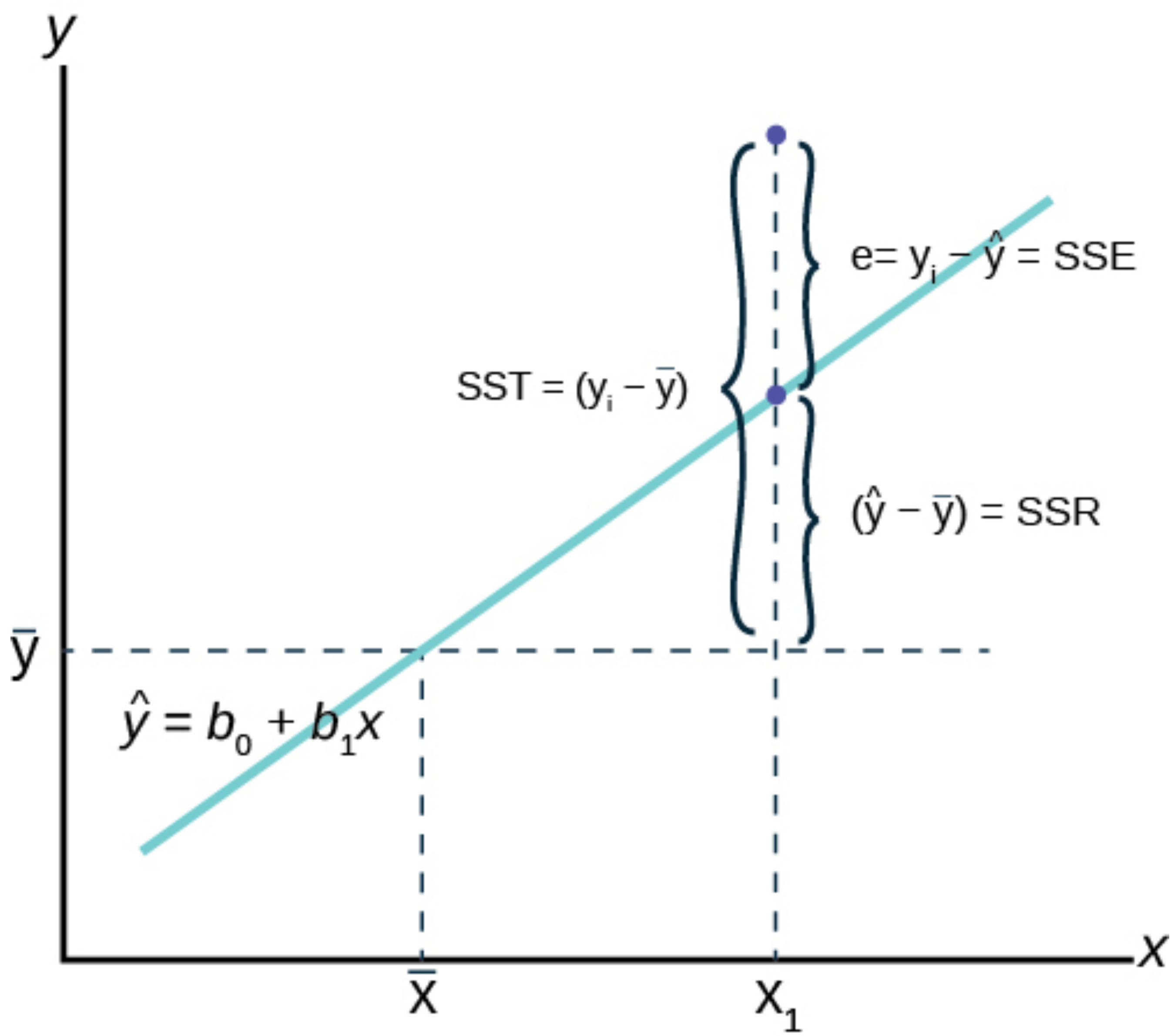
$$\beta_1 = \frac{\sum (x_i - \bar{X})(Y_i - \bar{Y})}{\sum (x_i - \bar{X})^2}$$

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

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

Filip Wójcik, filip.wojcik@outlook.com

LINEAR REGRESSION



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

$$SSR = \sum (\bar{y}_i - \hat{y}_i)^2$$

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

$$SSTO = SSR + SSE$$

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

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

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

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

$$= SSR + SSE$$

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

LINEAR REGRESSION

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

$$\boldsymbol{Y} = \boldsymbol{X}\beta + \varepsilon$$

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

LINEAR REGRESSION

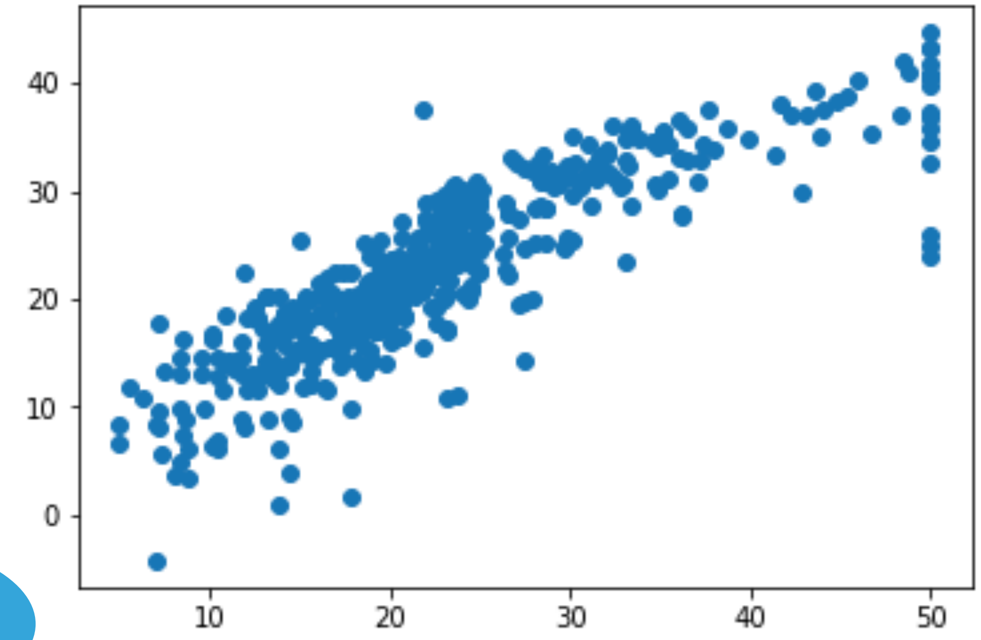
Sklearn API

CREATING PREDICTOR OBJECT

```
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)
```

PREDICTING NEW OBSERVATIONS

MANUALLY CHEKING PREDICTION METRICS



Filip Wójcik, filip.wojcik@outlook.com

LINEAR REGRESSION

Statsmodels API

```python
import statsmodels.api as stm

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

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

LINEAR FIT QUALITY

MODEL COMPLEXITY

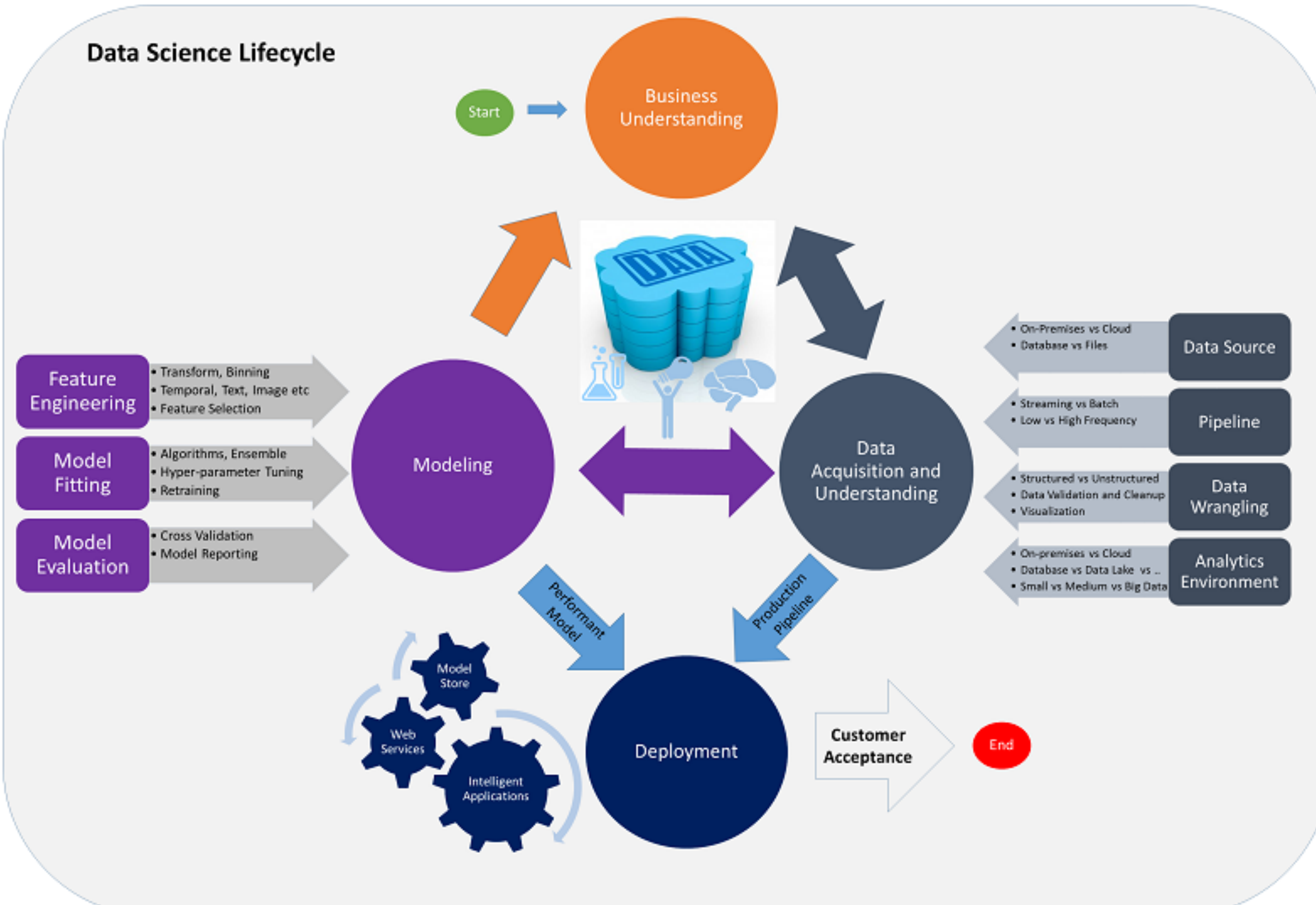
COEFFICIENTS AND THEIR IMPORTANCE

```
                            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
==============================================================================
```

PREDICTIVE MODELS BULDING

- Standard steps to build a predictive model with sklean
- Procedure includes:
  - Divide the data into training and testing
  - Train model on training data
  - Check performance on testing data
  - Reiterate if neccessary

PREDICTIVE MODELS BULDING & DATA SCIENCE GENERAL RECAP



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

THANK YOU!