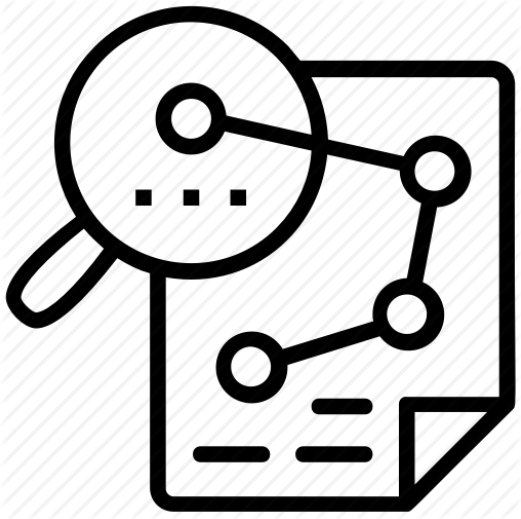
Workshop
Data Science Learning Strategy

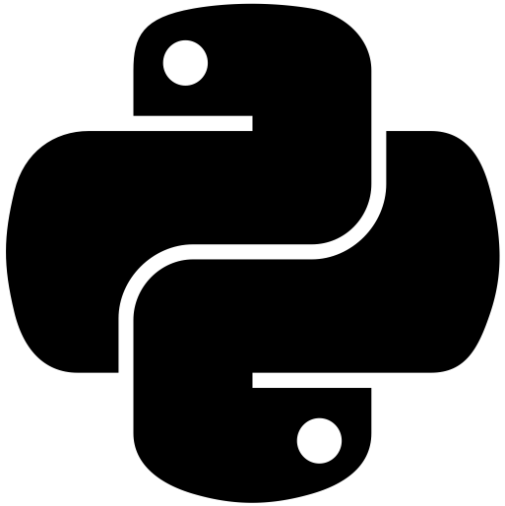
PS. How to survive in this competition

Muhammad Sifa'ul Rizky

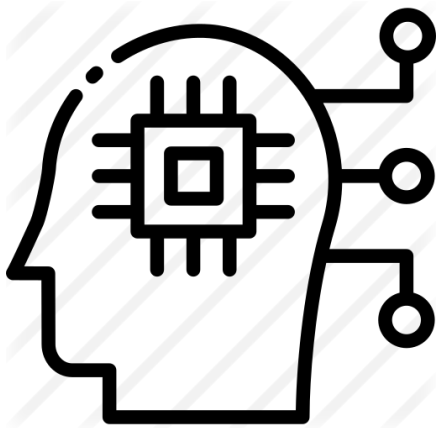
Content



Intro to Data Science



Python 101



Sneak Peek
of Machine Learning

About Me:

Lead Data Science Instructor
(PT. Renom Infrastruktur Indonesia – Make.ai)
Experience:

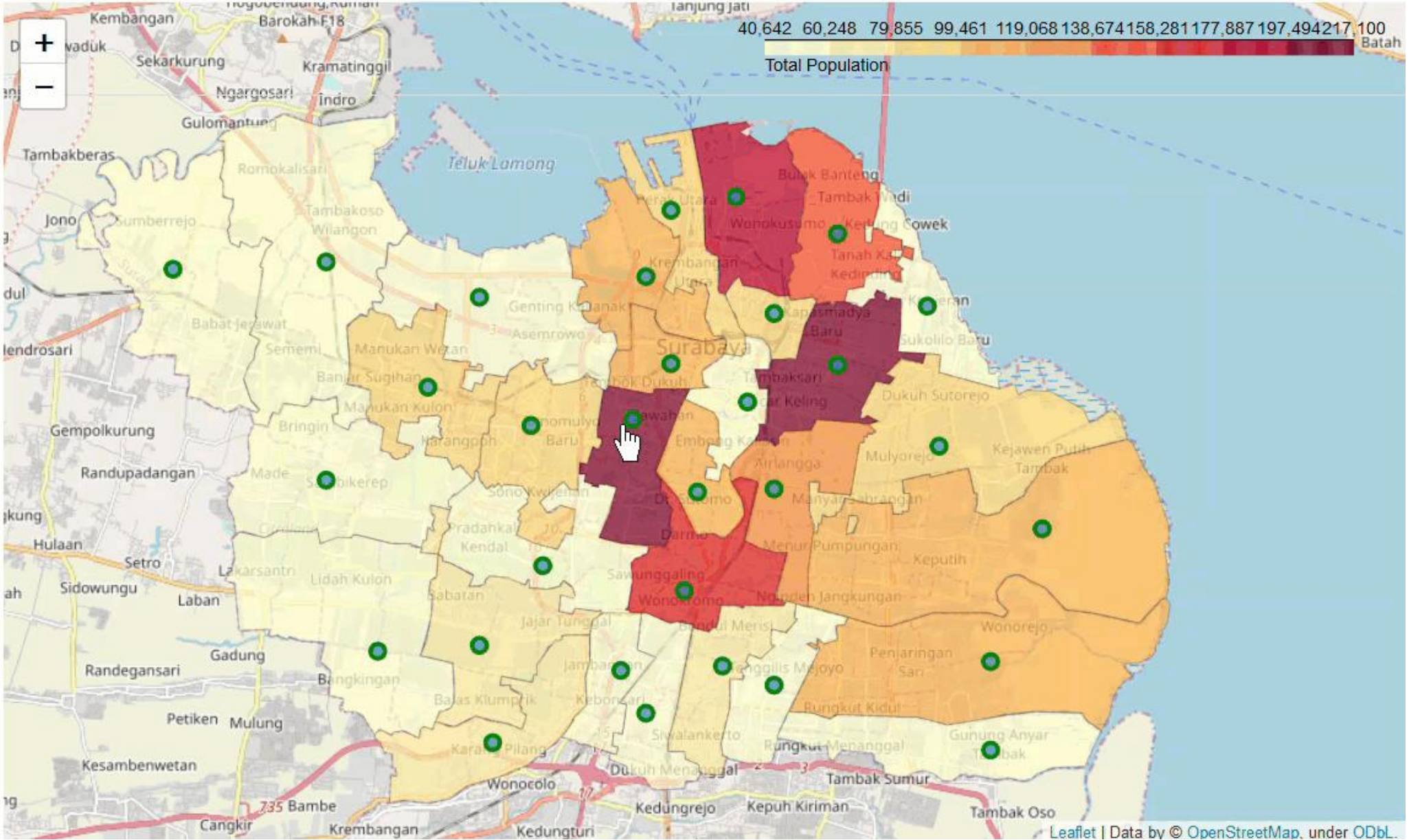
- Mentor and Reviewer at Udacity 2019-present
- Bertelsmann Data Science Scholarship from Udacity 2018-2019
- IBM Data Science Professional Certificate Course Coursera 2019
- Bertelsmann Data Scholarship Cloud Track from Udacity 2019-2020

… and many more

Muhammad Sifa'ul Rizky

rizkysifaul

Exploring Places and Area Regions in Surabaya (2019)




Muhammad Sifa'ul Rizky
Visualize Your Population and Density of Surabaya
based on Dianna with Folium
Lead of Data Science Instructor
(PT. Rasan Infrastruktur Indonesia)
PYCON

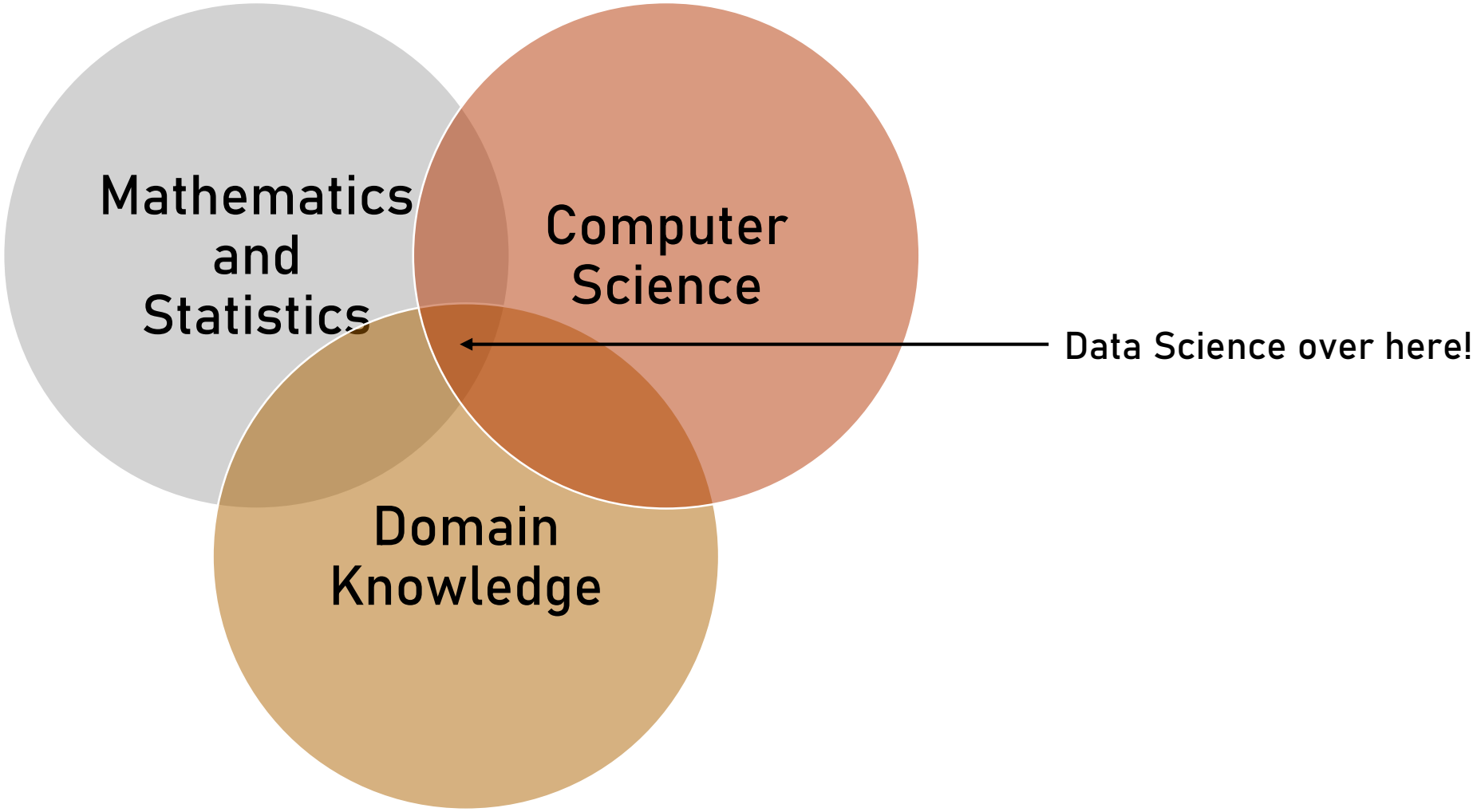
Intro to Data Science

I am sure that you are knowing about this side, so why you join in this workshop?

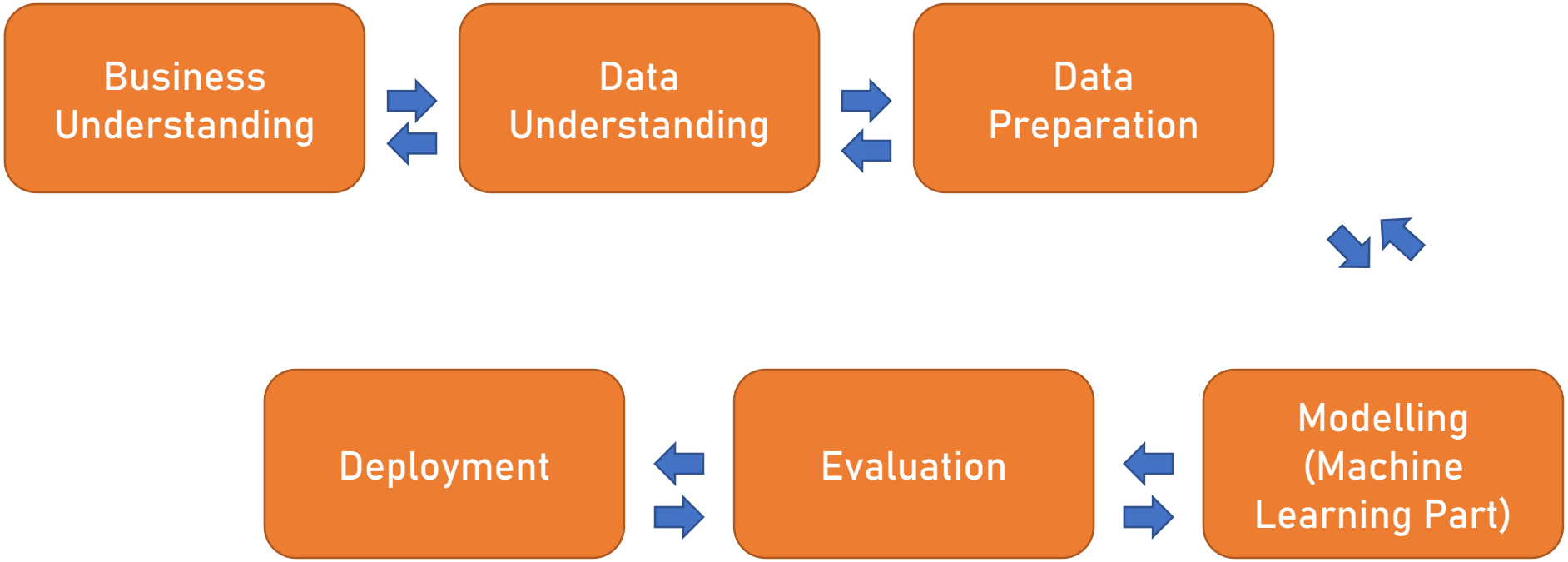
Definition

Data science is the study of data. It involves developing methods of recording, storing, and analyzing data to effectively extract useful information. (Tech Terms)

Familiar with this?



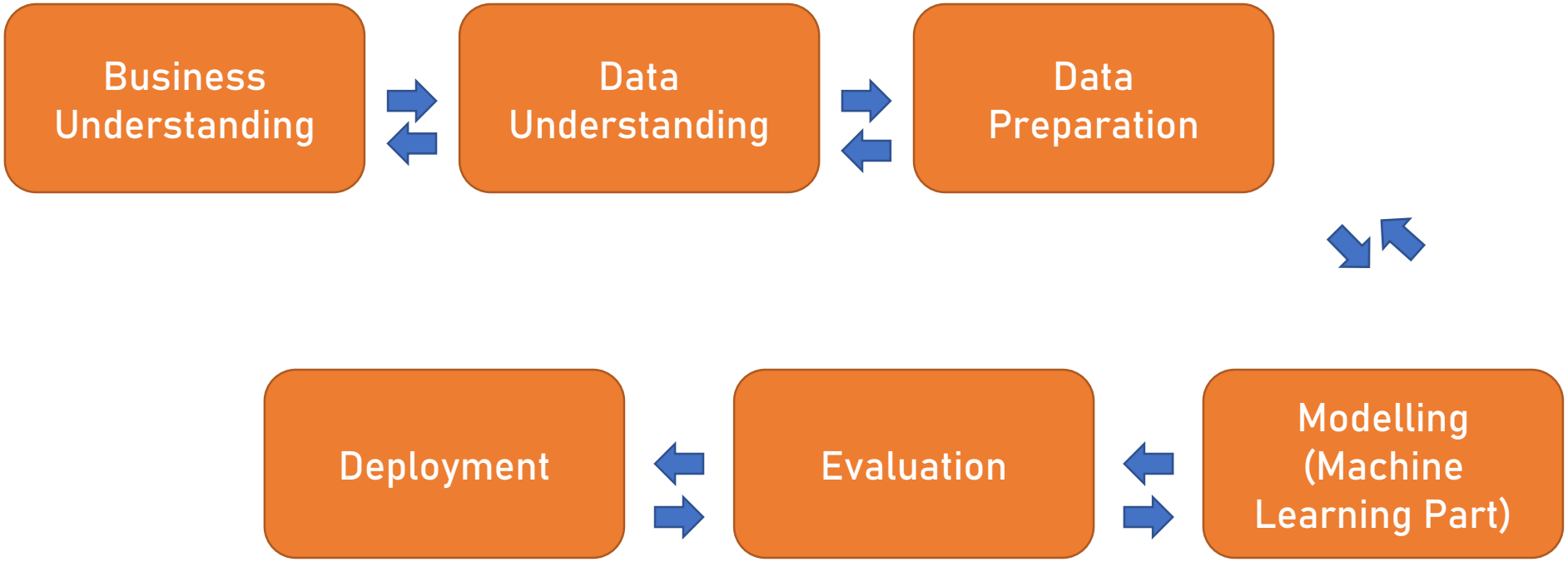
The Process of Data Science



Example

How our sales projection on next year?

Data on last five years

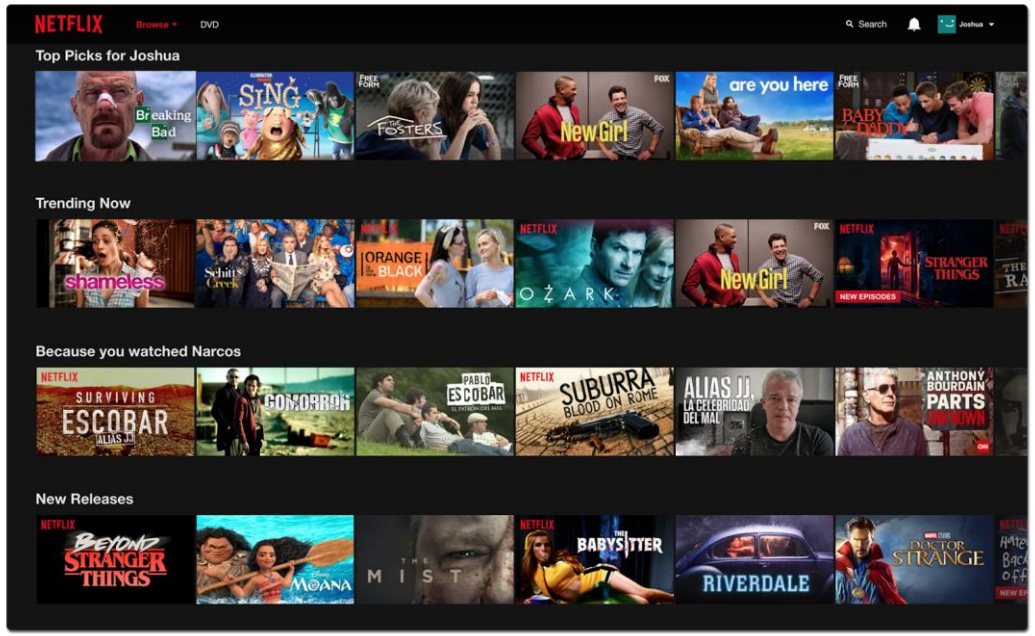


Create projection based on pattern

Hmm, seems that is too good to be true

It can be an app or report for next meeting what should we do

What can you do?



Series/Video Recommendation

Fintech Credit Scoring

Tools for your life

SQL

python™

R

tableau

Power BI

scikit learn

TensorFlow

PYTORCH

aws

Google Cloud Platform

APACHE Spark™

Python 101

How many people that still struggling with this language?

Do you think you are beginner, intermediate, or even advanced?

Starting with a simple things

```
print("Hello world")
```

Know your variable

```
#input
money = 100000
cash  = 50000.0
money + cash
----------------
#output
150000.0
```

```
#input
type(money)
print("---")
type(cash)
----------------
#output
int
---
float
```

Data type is important too

Containers is needed!

```
#input
name_single = "Faul"
name_multi = ["Faul", "Arga", "Clara"]
type(name_single)
print("---")
type(name_multi)
--------------
#output
str
---
list
```

Using index for access

```
#input
print(name_multi[0])
print(name_multi[1])
print(name_multi[2])
--------------
#output
"Faul"
"Arga"
"Clara"
```

So many things inside

```
#input
gpa = 3.01
if gpa > 3.5:
    print("Great result!")
elif gpa > 3:
    print("Good one")
else:
    print("Don't give up, get better!")
--------------
#output
"Good one"
```

Iterative?
No Problem

```
#input
#for
for number in range(5):
    print(number)

#while
number = 0
while number < 5:
    print(number)
    number = number + 1
-------------
#output(for and while return same output)
0
1
2
3
4
```

More details?
Function is your solution

```
#input
def hitung_bulan(tahun):
    return tahun * 12

hitung_bulan(25)
--------------
#output
300
```

Sneak Peek of Machine Learning

So, actually after we learn some basics of Python, now how would we applied machine learning to this?

Predicting House Prices (Simple but Annoying)

IDR 300 Million

IDR 450 Million

?

First, always import your package

```
#input
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
--------------
#output
#nothing, it just imported your package
```

What is that?

NumPy

pandas

Seaborn

matplotlib

Your data
should be ready

```
#input
data = pd.read_csv('data.csv')
data.head()
--------------
#output
#hmm what's is the output?
```

Output

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement
0	2014-05-02 00:00:00	313000.0	3.0	1.50	1340	7912	1.5	0	0	3	1340	0
1	2014-05-02 00:00:00	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5	3370	280
2	2014-05-02 00:00:00	342000.0	3.0	2.00	1930	11947	1.0	0	0	4	1930	0
3	2014-05-02 00:00:00	420000.0	3.0	2.25	2000	8030	1.0	0	0	4	1000	1000
4	2014-05-02 00:00:00	550000.0	4.0	2.50	1940	10500	1.0	0	0	4	1140	800

```
#checking name of columns
print(data.columns)
#checking how long column we have
print(len(data.columns)
```

Check your columns

Output

- Index(['date', 'price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot','floors', 'waterfront', 'view', 'condition', 'sqft_above', 'sqft_basement', 'yr_built', 'yr_renovated', 'street', 'city', 'statezip', 'country'], dtype='object')
- 18

How much is the price range?

```
data[['price']].describe()
```

Name	Price
count	4600.00
mean	551962.99
std	563834.70
min	0.00
25%	322875.00
50%	460943.46
75%	654962.50
max	26590000.00

Seems strange!

Anomalies(?)

```python
#check what house that dont have price (price=0)
data[data['price'] == 0]
```

Price = 0?

So much!

Out[14]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition
4354	2014-05-05 00:00:00	0.00	3.00	1.75	1490	10125	1.00	0	0	4
4356	2014-05-05 00:00:00	0.00	4.00	2.75	2600	5390	1.00	0	0	4
4357	2014-05-05 00:00:00	0.00	6.00	2.75	3200	9200	1.00	0	2	4
4358	2014-05-06 00:00:00	0.00	5.00	3.50	3480	36615	2.00	0	0	4
4361	2014-05-07 00:00:00	0.00	5.00	1.50	1500	7112	1.00	0	0	5
4362	2014-05-07 00:00:00	0.00	4.00	4.00	3680	18804	2.00	0	0	3

49 rows

```python
#check what house that dont have price (price=26590000)
data[data['price'] == 26590000]
```

Price = 26590000

So few!

Out[16]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition
4350	2014-07-03 00:00:00	26590000.00	3.00	2.00	1180	7793	1.00	0	0	4

1 row

What should you do?

Just go on

Analyze another row,
maybe ignoring or
do at the end of project

Do something

Delete price rows, or
just delete missing
value etc

I can't do this!
Why everything is so
heavy.

Do something

- Looking for upper and lower bounds

$$Lower\ bound = Q1\ - 1.5 * (Q3 - Q1)$$

$$Upper\ bound = Q3 + 1.5 * (Q3 - Q1)$$

```python
#what should you do?
#in this case I will do something, drop value that not on range
iqr = data['price'].describe()['75%'] - data['price'].describe()['25%']
lower_bound = data['price'].describe()['25%'] - (1.5*iqr)
upper_bound = data['price'].describe()['75%'] + (1.5*iqr)

print("IQR equals {}".format(iqr))
print("Lower bound of price is {}".format(lower_bound))
print("Upper bound of price is {}".format(upper_bound))
```

How-to

Output

- IQR equals 332087.5
- Lower bound of price is -175256.25
- Upper bound of price is 1153093.75

How if Lower bound is minus?

So that all the values on lower should be count,

starting from > 0.

```
#just go on with data itself
data_clean = data.copy()
data_clean = data[(data.price > 0) & (data.price <= upper_bound)]
data_clean.shape

--------------
#output
(4311, 18)
```

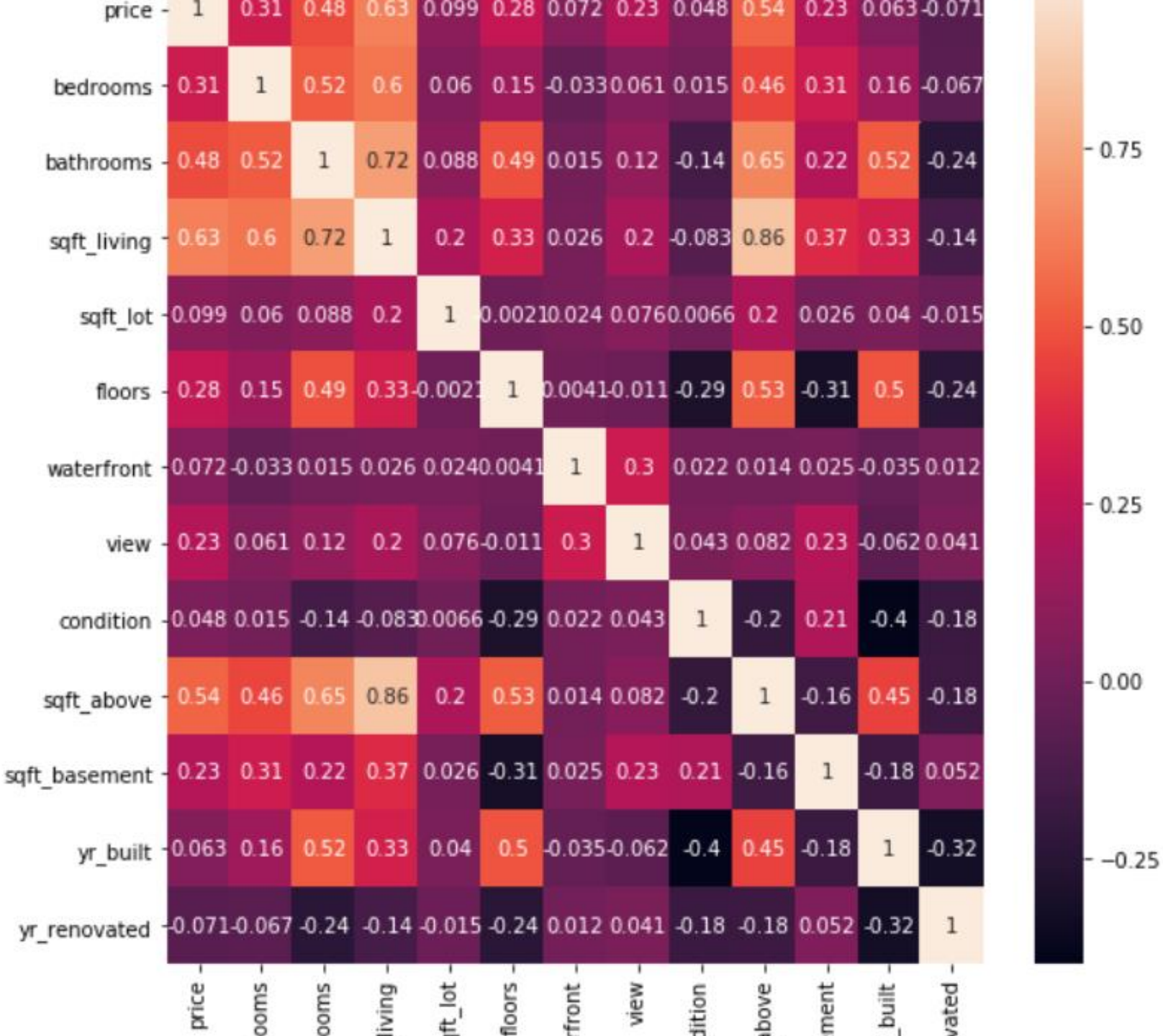
Drop data price

col date

Run Inte

date
price
bedrooms
bathrooms
sqft_living
sqft_lot
floors
waterfront
view
condition
sqft_above
sqft_basement
yr_built
yr_renovated

```
@int
def
(col)))
alse);
```


price 1 0.31 0.48 0.63 0.099 0.28 0.072 0.23 0.048 0.54 0.23 0.063 -0.071
bedrooms 0.31 1 0.52 0.6 0.06 0.15 -0.033 0.061 0.015 0.46 0.31 0.16 -0.067
bathrooms 0.48 0.52 1 0.72 0.088 0.49 0.015 0.12 -0.14 0.65 0.22 0.52 -0.24
sqft_living 0.63 0.6 0.72 1 0.2 0.33 0.026 0.2 -0.083 0.86 0.37 0.33 -0.14
sqft_lot 0.099 0.06 0.088 0.2 1 0.0021 0.024 0.076 0.0066 0.2 0.026 0.04 -0.015
floors 0.28 0.15 0.49 0.33 -0.0021 1 0.0041 -0.011 -0.29 0.53 -0.31 0.5 -0.24
waterfront 0.072 -0.033 0.015 0.026 0.024 0.0041 1 0.3 0.022 0.014 0.025 -0.035 0.012
view 0.23 0.061 0.12 0.2 0.076 -0.011 0.3 1 0.043 0.082 0.23 -0.062 0.041
condition 0.048 0.015 -0.14 -0.083 0.0066 -0.29 0.022 0.043 1 -0.2 0.21 -0.4 -0.18
sqft_above 0.54 0.46 0.65 0.86 0.2 0.53 0.014 0.082 -0.2 1 -0.16 0.45 -0.18
sqft_basement 0.23 0.31 0.22 0.37 0.026 -0.31 0.025 0.23 0.21 -0.16 1 -0.18 0.052
yr_built 0.063 0.16 0.52 0.33 0.04 0.5 -0.035 -0.062 -0.4 0.45 -0.18 1 -0.32
yr_renovated -0.071 -0.067 -0.24 -0.14 -0.015 -0.24 0.012 0.041 -0.18 -0.18 0.052 -0.32 1
price
ooms
ooms
living
ft_lot
floors
front
view
dition
above
ment
built
vated
0.75
0.50
0.25
0.00
−0.25

Simple selecting

```
#selecting bigger and/or smaller correlation plot
data_clean = data_clean[['sqft_living','bathrooms','bedrooms','floors','price']]
data_clean.head()
```

sqft_living	bathrooms	bedrooms	floors	price
1340	1.5	3.00	1.5	313,000.00
1930	2.00	3.00	1.00	342,000.00
2000	2.25	3.00	1.00	420,000.00
1940	2.50	4.00	1.00	550,000.00
880	1.00	2.00	1.00	490,000.00

```python
#Import addionpackage
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

X = data_clean.drop('price', axis=1)
y = data_clean['price']
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=.2, random_state=45)
```

Split your data

Modelling your data

```
#import your model
model=LinearRegression()

#fitting and predict your model
model.fit(X_train, y_train)
model.predict(X_test)
```

```
array([ 485354.69817553,
441835.3076801 ,
453789.98520846,
        415049.11850156,
473551.53971655,
550880.38328669,
        413405.93671759,
743722.90407821,
484311.20643497,
        376271.3433367 ,
761396.97759918,
303086.79083996,
        328005.09496555,
813657.50395897,
303050.37988799,…
```

Weight of each columns

```python
#coeficient for each column
print(model.coef_)
#bias of model
print(model.intercept_)
```

$$
\begin{aligned}
&price \\
&= 179.3 * sqft_{living} \\
&+ 5527.7 * bathrooms \\
&- 28018.6 * bedrooms \\
&+ 23832.5 * floors \\
&+ 170132
\end{aligned}
$$

Scoring

```
#metrics scoring
from sklearn.metrics import mean_squared_error, r2_score
print("Root MSE of this model is {}".format(np.sqrt(mean_squared_error(y_test,model.predict(X_test)))))
print("R2 of this model is {}".format((r2_score(y_test,model.predict(X_test)))))
```

```
Root MSE of this model is 161854.5292971274
R2 of this model is 0.3675004503026602
```

How to understand

ID	Y_test	prediction
1591	350,000.00	485,354.00
2164	240,000.00	441,835.31
1890	532,000.00	453,789.99

On the range

Range of price: 291,935.4 - 615,644.5

Error: -135,534

Error: -201,835.31

Error: 78,210.01

RMSE makes range between your prediction and error itself.

End Product

“Without big data analytics, companies are blind and deaf, wandering out onto the web like deer on a freeway.” (Gary King-Harvard)

Danke schoen!

Terima kasih!