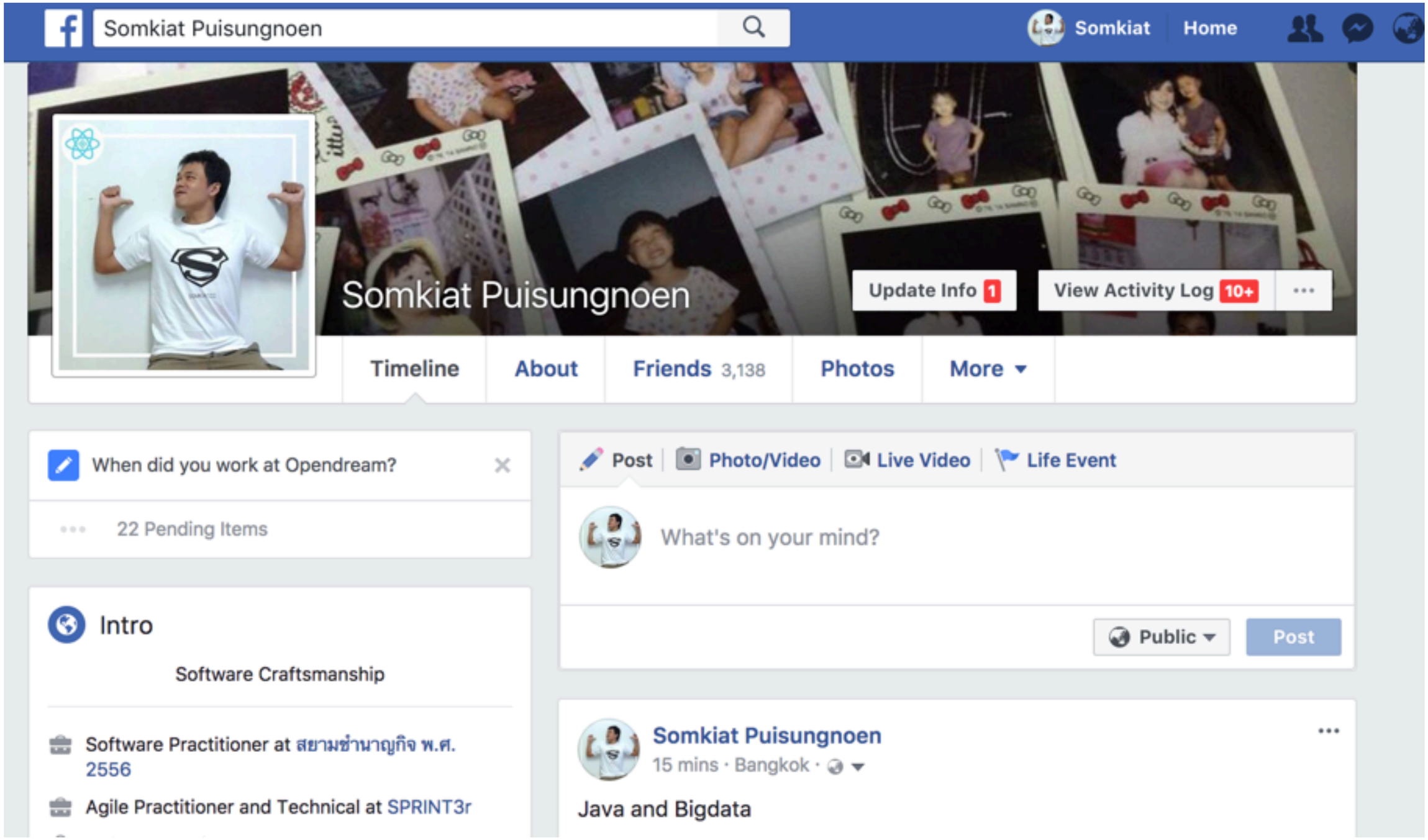
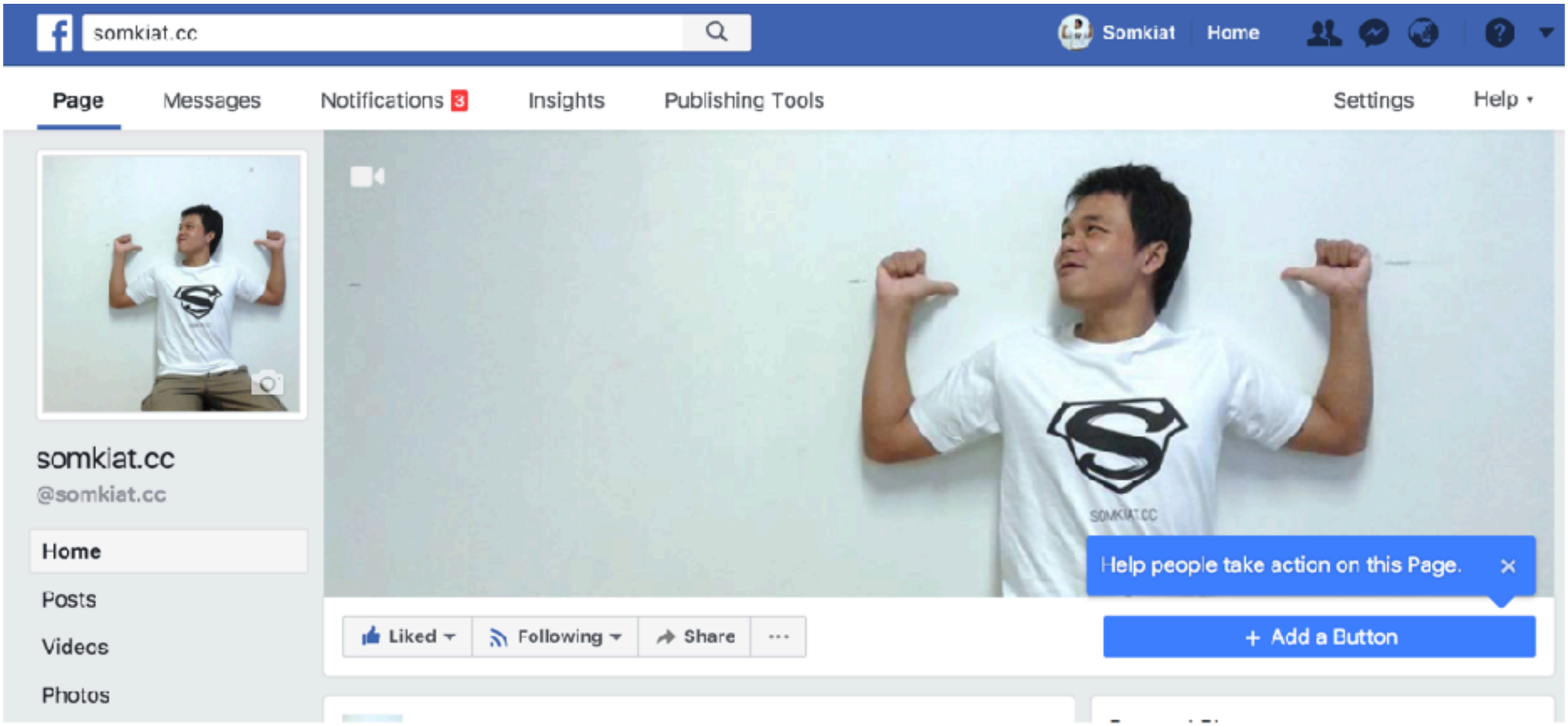
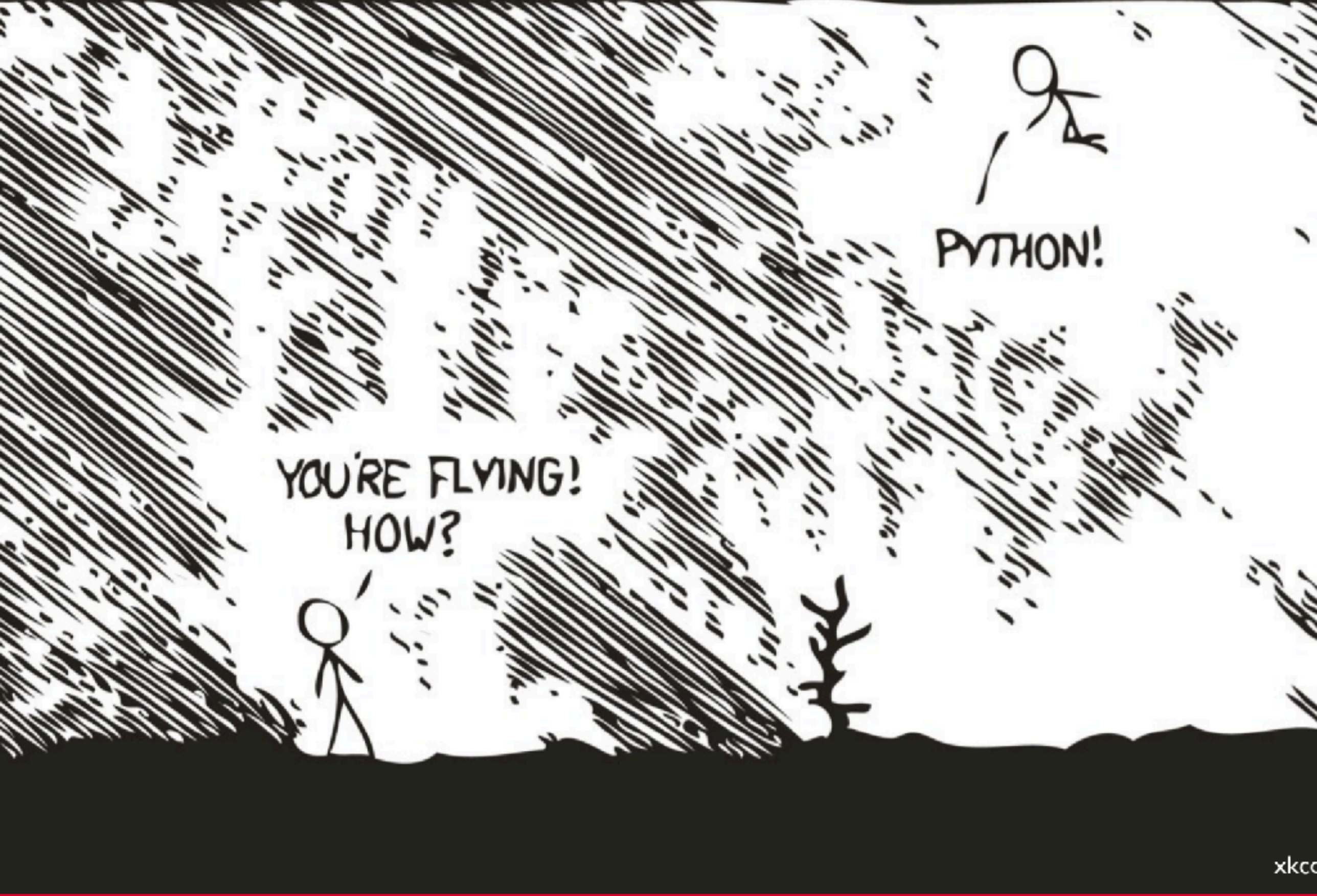
สยามชำนาญกิจ
SIAM CHAMNAN KIT

Python for Data Science





Advance Python for Data Science



Agenda

TDD for Python

Setup your computer => Python 3 and Jupyter

Summary of Python

List comprehensive

Workshop

Agenda

Data Science

Data Science with Python

Numpy, Pandas and Matplotlib/Seaborn

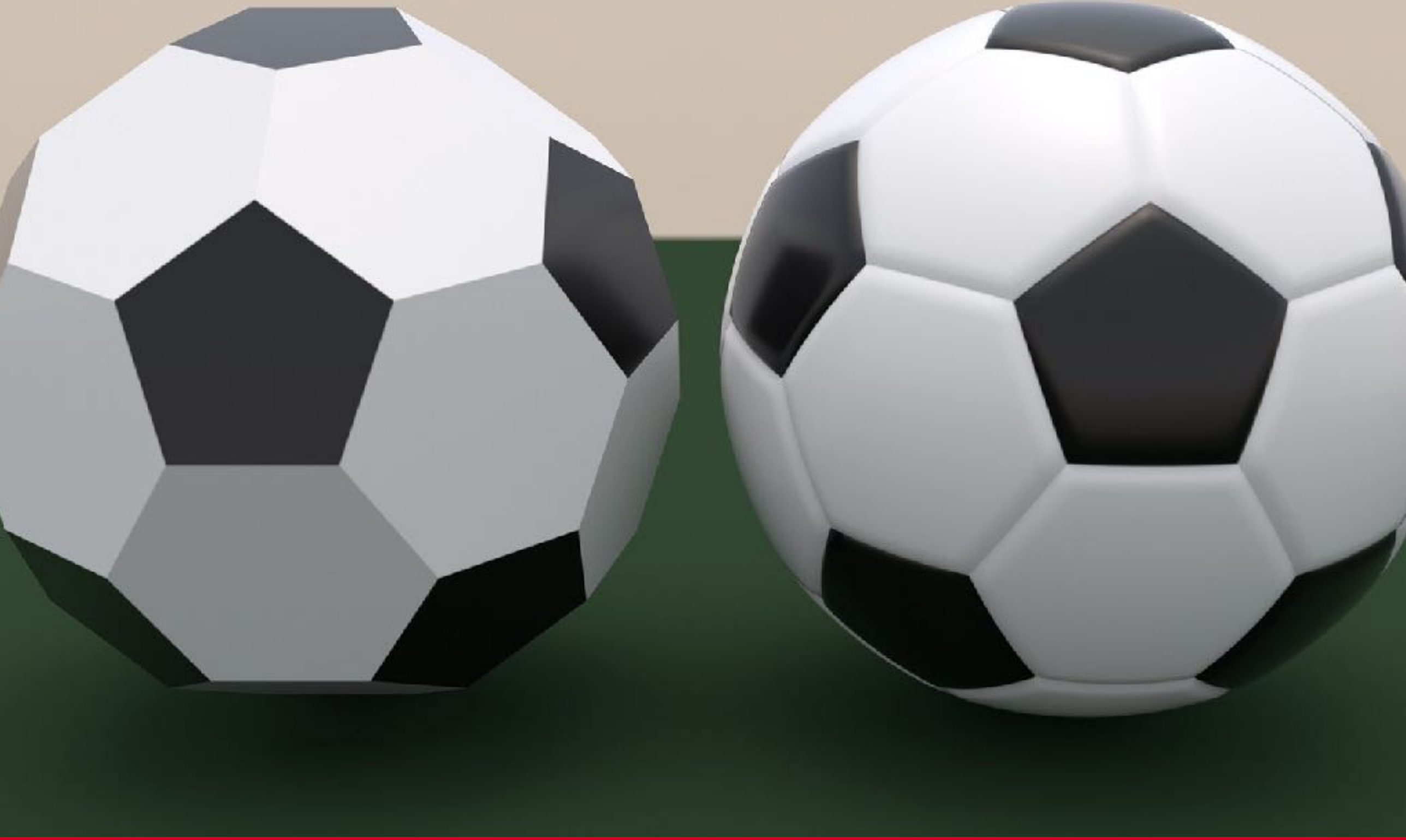
Scikit-learn

Kaggle :: Home for Data Science

RSpark for Big Data

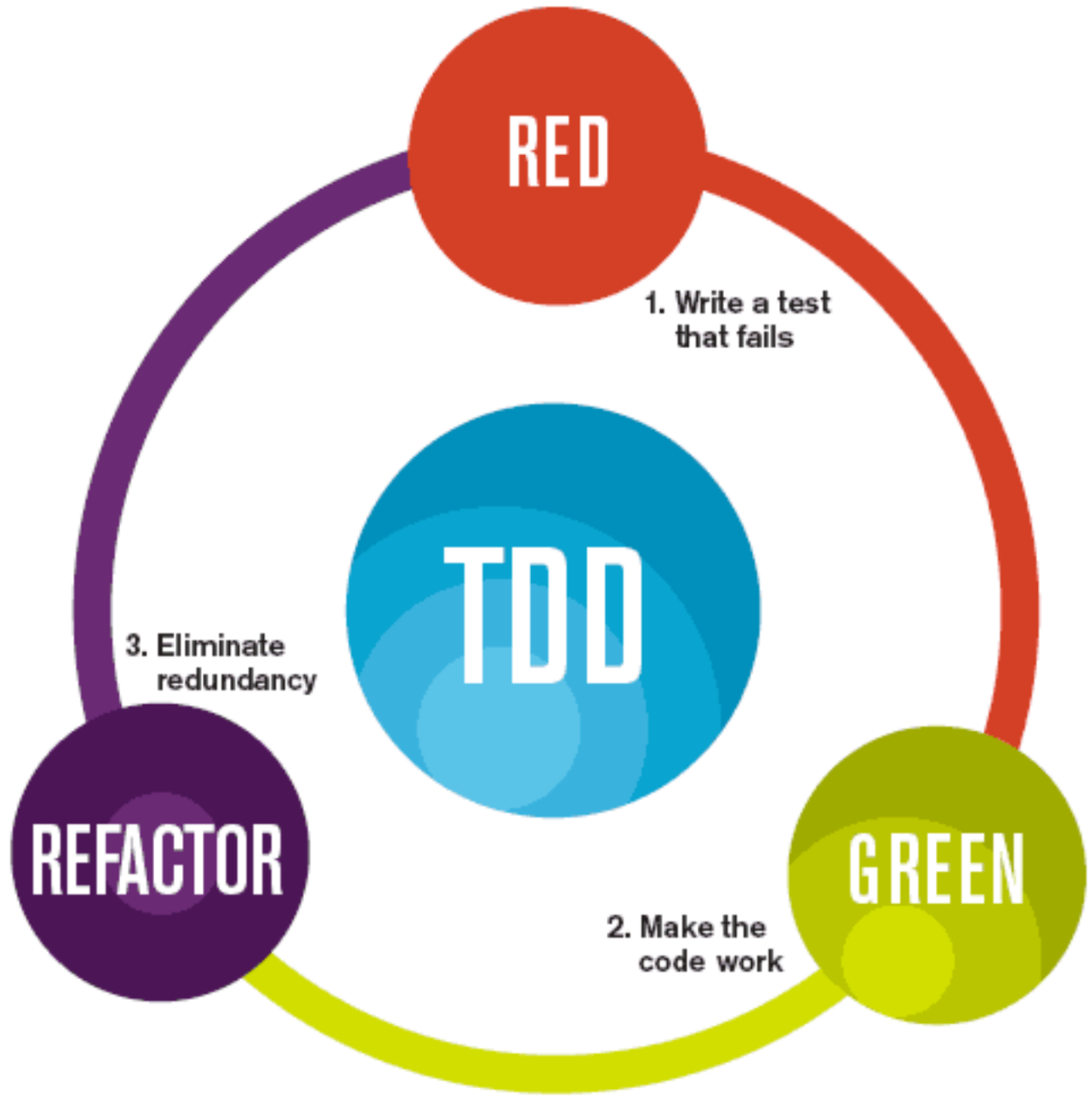
Workshop and Homework

https://github.com/up1/course-python-for-data-science



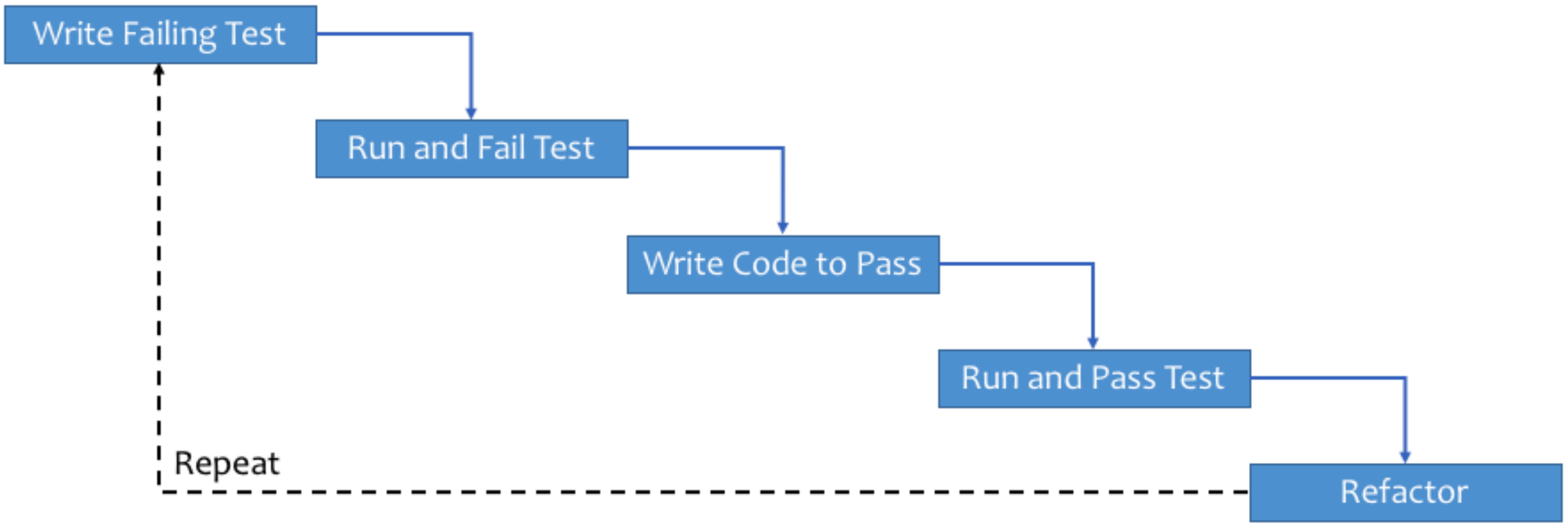
TDD for Data Science (Python)

TDD for Data Science (Python)



The mantra of Test-Driven Development (TDD) is "red, green, refactor."

Workflow



Write first test case

```
import unittest
from hello import *

class MyFirstTests(unittest.TestCase):

    def test_hello(self):
        self.assertEqual(hello_world(), 'hello world')

if __name__ == '__main__':
    unittest.main()
```

Run your test

$python hellotests.py

Write your code

Basic Data Types

int - Integer value
float - Decimal value
bool - True/False
complex - imaginary
NoneType - null value

Iterable data types

Type	Meaning
str	String immutable value
list	Collection of elements
tuple	Immutable list
dict	Unordered key-value pairs
set	Unordered collection of unique elements

Iterable data types

Type	How to use ?	Example
str	Defined with quotes	‘ab’
list	Defined with brackets	[‘a’, ‘b’]
tuple	Defined with parentheses	(‘a’, ‘b’)
dict	Defined with braces	{‘a’: 1, ‘b’: 2}
set	Defined with braces	{‘a’, ‘b’}

Control Flows

If-else statements
While loops
For loops

List comprehensive

Use for creating new list from another iterables
Introduced in Python 2.0
Python 3.0 comes with Dict and Set

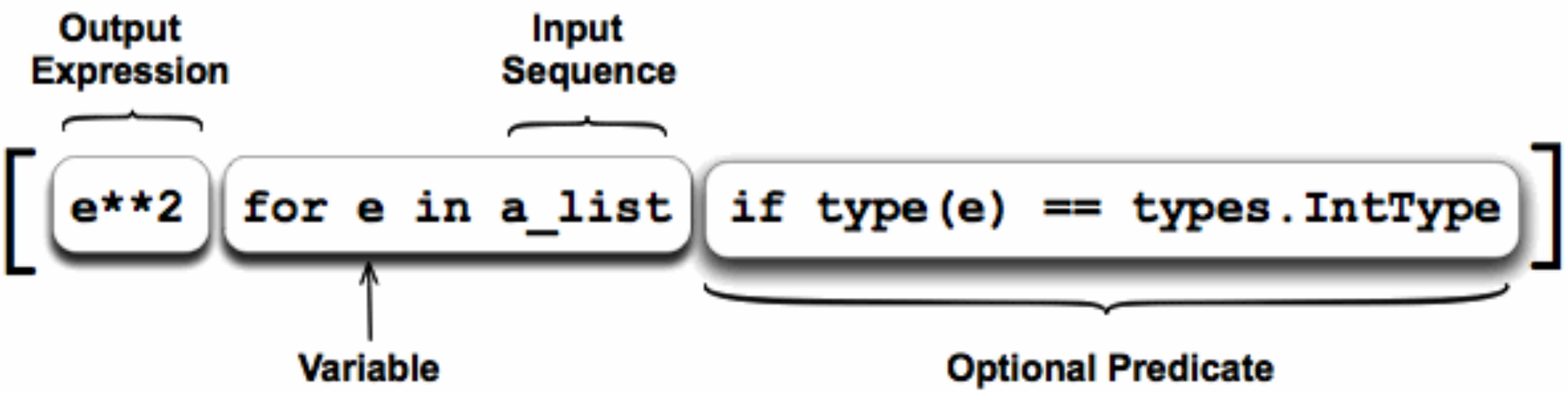
List comprehensive

Try to replace for loops and map(), filter(), reduce()

In Data Science working with List too much !!

List comprehensive

1. Input sequence
2. Variable of input sequence
3. Optional predicate expression
4. Output expression



Example 1

Square of number

```
def calculate():
    numbers = [1, 2, 3, 4, 5]
    results = []
    for number in numbers:
        results.append(number**2)
    print(results)

if __name__== "__main__":
    calculate()
```

Rewrite with List comprehensive

Square of number

```
def calculate():
    numbers = [1, 2, 3, 4, 5]
    result = [number**2 for number in numbers]
    print(result)

if __name__== "__main__":
    calculate()
```

Example 2

Find the same number in 2 lists

```
def process():
    list1 = [1, 2, 3, 4, 5]
    list2 = [3, 4, 5, 6, 7]
    results = []
    for x in list1:
        for y in list2:
            if x == y:
                results.append(x)

    print(results)

if __name__== "__main__":
    process()
```

Rewrite with List comprehensive

Find the same number in 2 lists of number

```
def process():
    list1 = [1, 2, 3, 4, 5]
    list2 = [3, 4, 5, 6, 7]
    results = [x for x in list1 for y in list2 if x==y]
    print(results)

if __name__== "__main__":
    process()
```

Example 3

Replace number with string (Even and Odd)

```
def process():
    numbers = [1, 2, 3, 4, 5]
    results = []
    for number in numbers:
        if number%2 == 0:
            results.append("Even")
        else:
            results.append("Odd")
    print(results)

if __name__== "__main__":
    process()
```

Rewrite with List comprehensive

Replace number with string (Even and Odd)

```
def process():
    numbers = [1, 2, 3, 4, 5]
    results = ["Even" if number%2 == 0 else "Odd" for number in numbers]
    print(results)

if __name__== "__main__":
    process()
```

Example 4

Remove vowels from sentence

```
def process(sentence):
    vowels = 'aeiou'
    results = []
    for c in sentence:
        if c not in vowels:
            results.append(c)
    return ''.join(results)

if __name__== "__main__":
    print(process('Hello World'))
```

Rewrite with List comprehensive

Remove vowels from sentence

```
def process(sentence):
    vowels = 'aeiou'
    return ''.join([c for c in sentence if c not in vowels])

if __name__== "__main__":
    print(process('Hello World'))
```

Try to practices

Practice python skills

Project Euler.net

Weekly Python Exercise

Project Euler (616 problems)

https://projecteuler.net/archives

ID	Description / Title	Solved By
1	Multiples of 3 and 5	749135
2	Even Fibonacci numbers	602067
3	Largest prime factor	431141
4	Largest palindrome product	383883
5	Smallest multiple	392340
6	Sum square difference	394673
7	10001st prime	337853
8	Largest product in a series	285474
9	Special Pythagorean triplet	288258
10	Summation of primes	264489

1. Multiples of 3 and 5

https://projecteuler.net/problem=1

Problem 1

If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23.

Find the sum of all the multiples of 3 or 5 below 1000.

2. Even Fibonacci Numbers

https://projecteuler.net/problem=2

Problem 2

Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 2, the first 10 terms will be:

1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

By considering the terms in the Fibonacci sequence whose values do not exceed four million, find the sum of the even-valued terms.

6. Sum square difference

https://projecteuler.net/problem=6

Problem 6

The sum of the squares of the first ten natural numbers is,

$$1^2 + 2^2 + \ldots + 10^2 = 385$$

The square of the sum of the first ten natural numbers is,

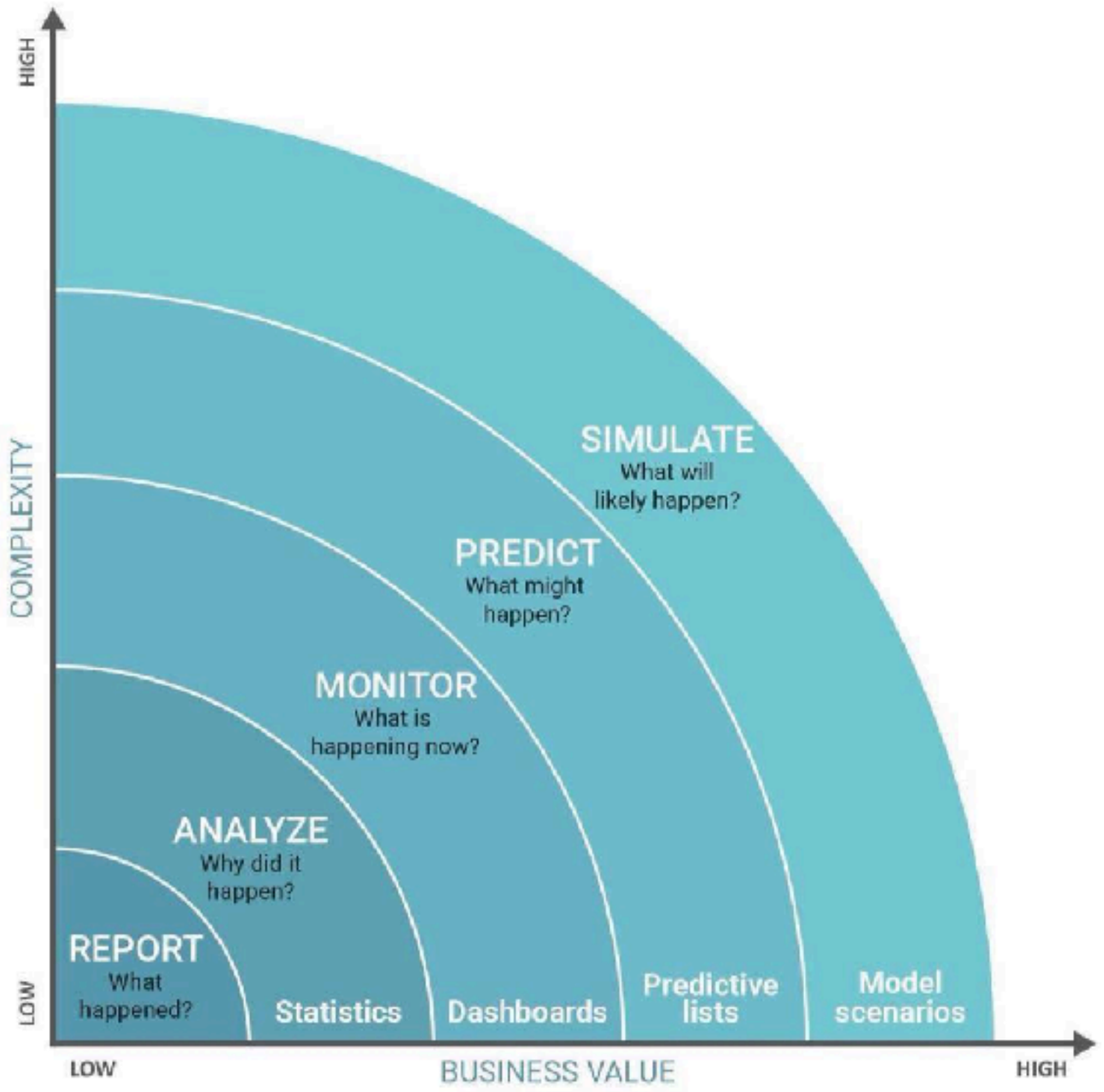
$$(1 + 2 + \ldots + 10)^2 = 55^2 = 3025$$

Hence the difference between the sum of the squares of the first ten natural numbers and the square of the sum is $3025 - 385 = 2640$.

Find the difference between the sum of the squares of the first one hundred natural numbers and the square of the sum.

Data Science

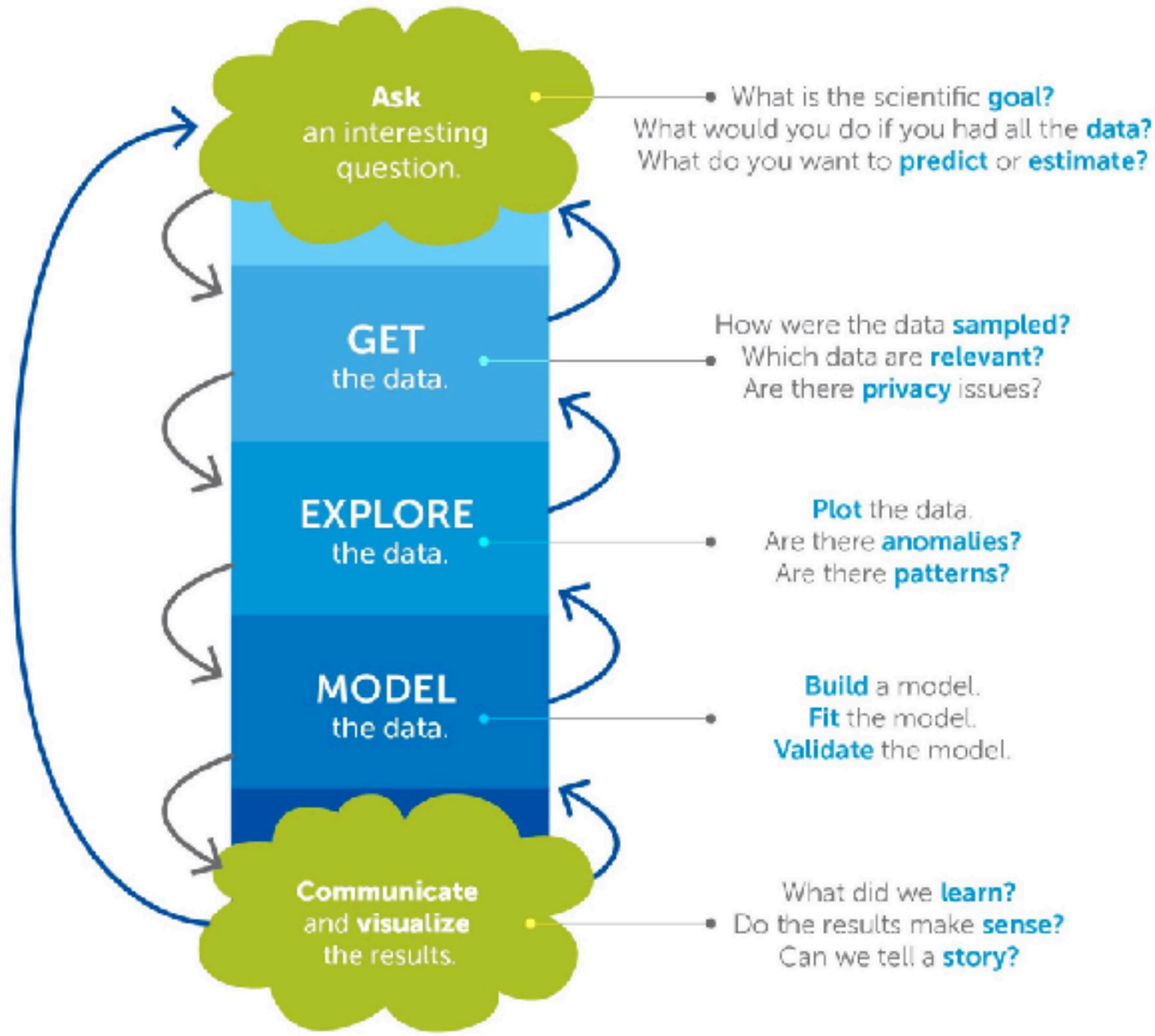
Levels of Data Science



Data Science Process

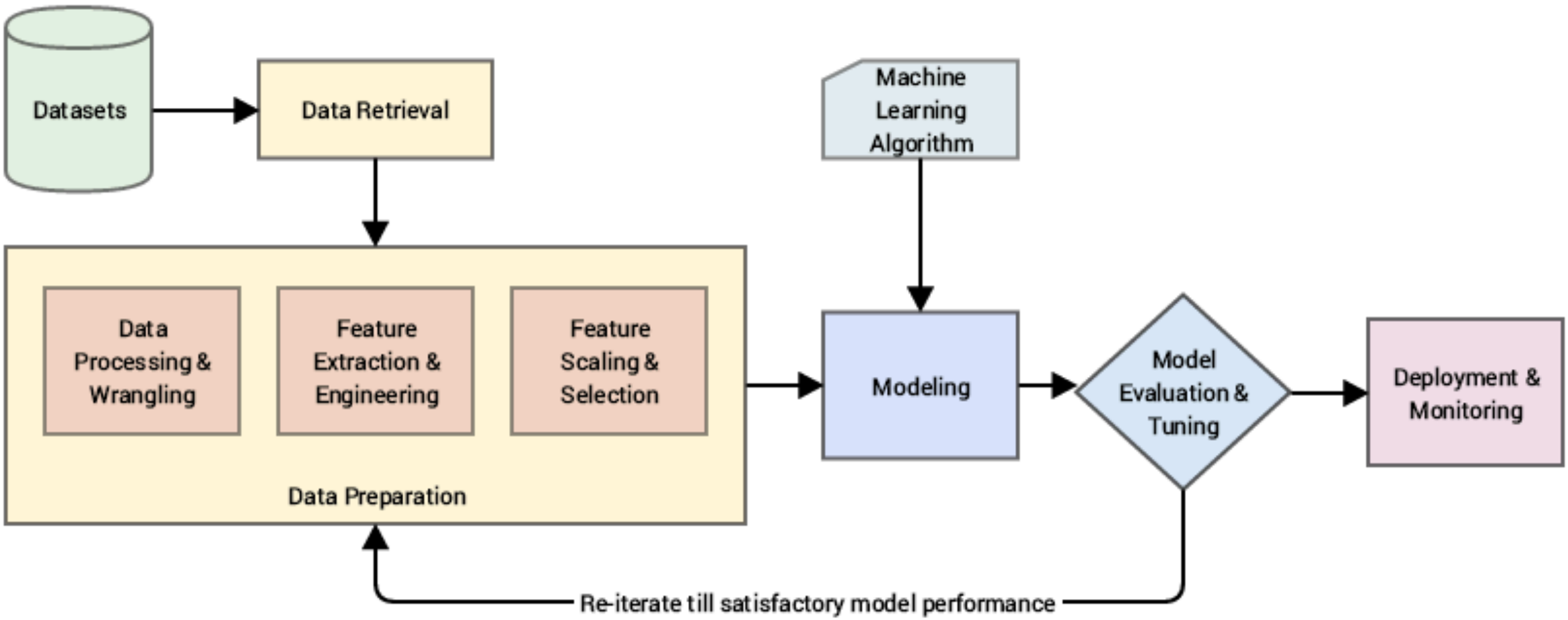
1. Collect the raw data needed to solve problem
2. Process the data (data wrangling)
3. Explore the data (data visualization)
4. Perform in-depth analysis (ML, Statistic, Algorithm)
5. Communicate result of the analysis

The Data Science Process

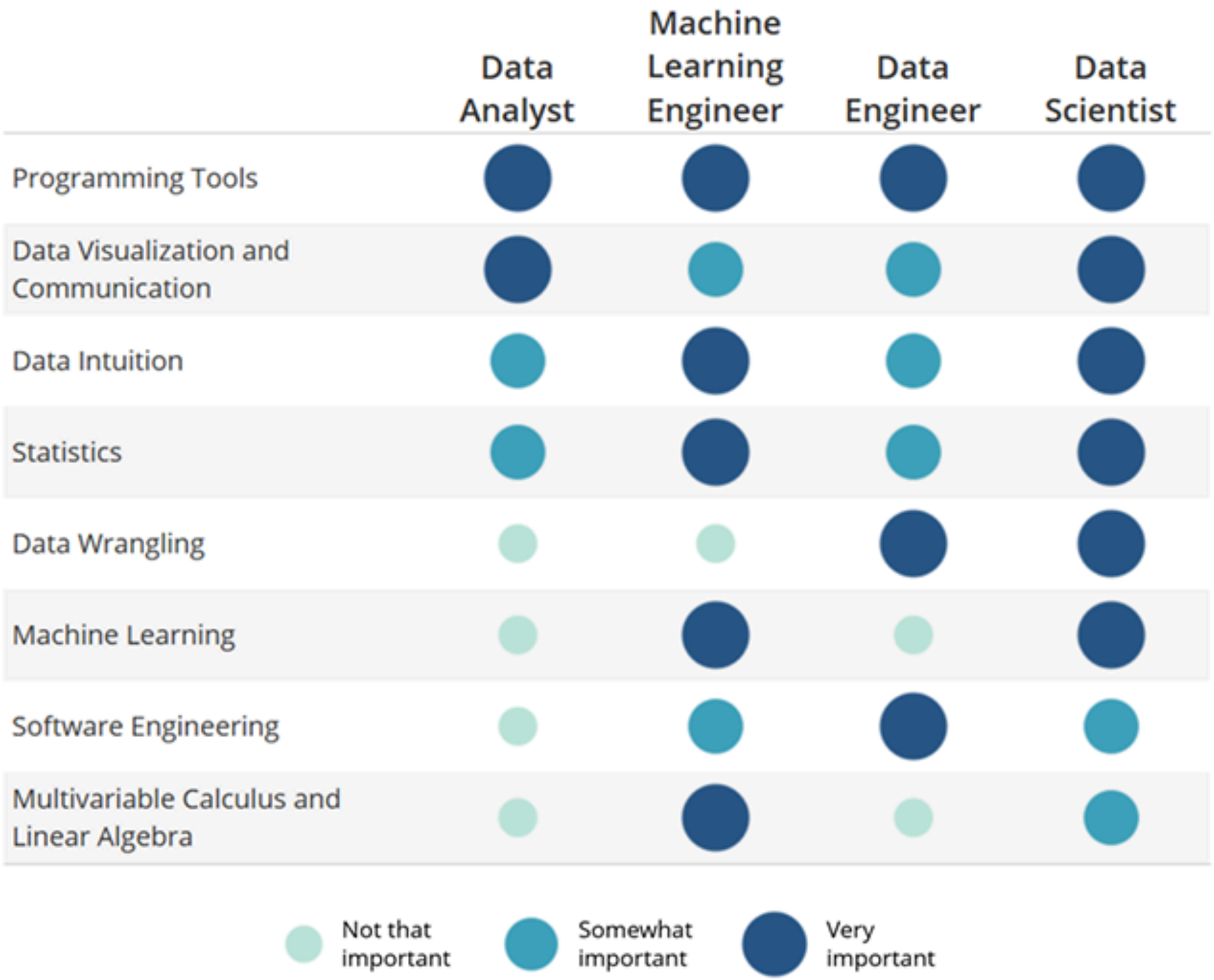


Derived from the work of Joe Blitzstein and Hanspeter Pfister, originally created for the Harvard data science course http://cs109.org/.

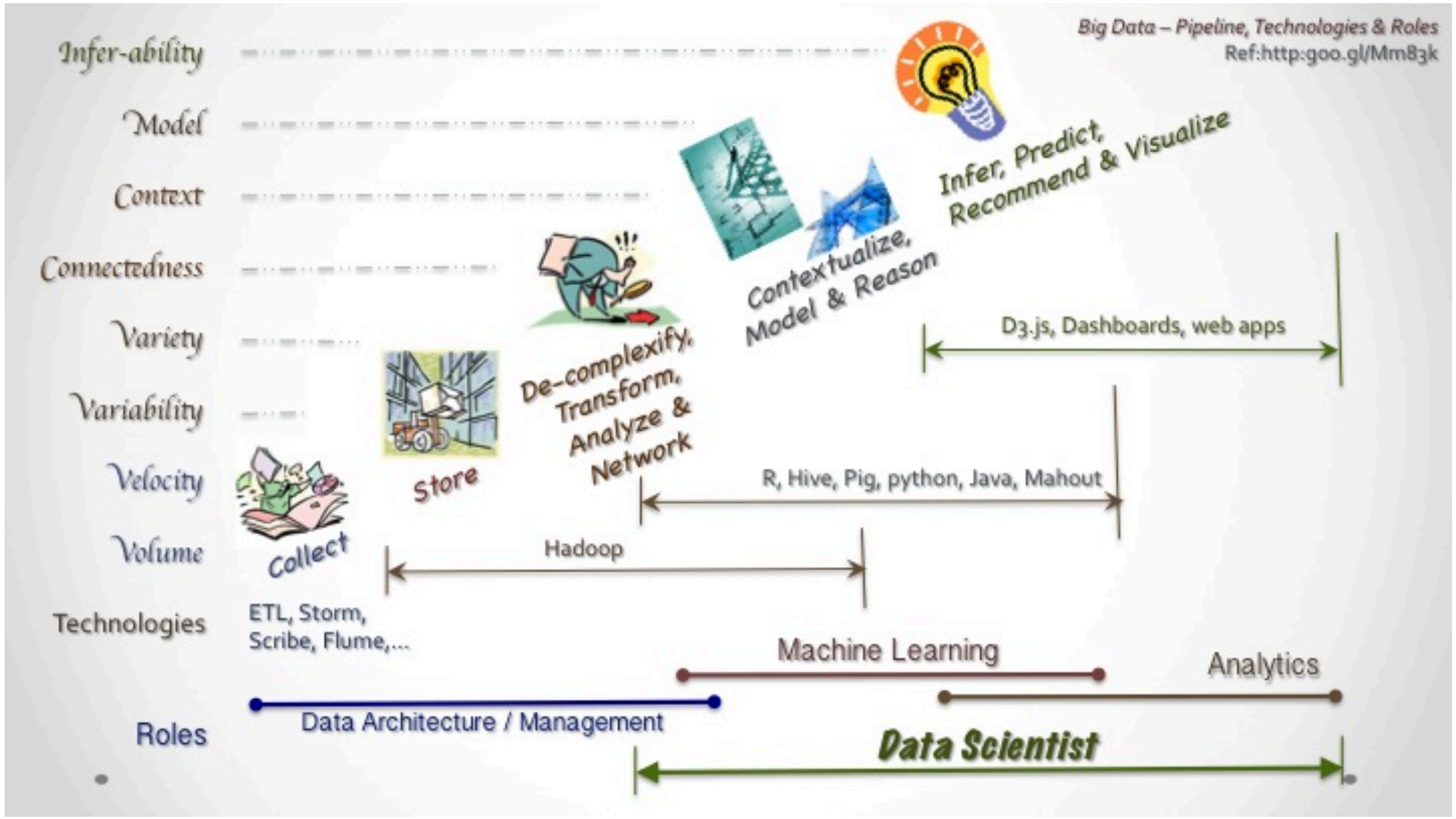
Data Science Process



Data Science Skills



Operation under Data Science



Exploratory Data Analysis

Check data how it is scattered
Data dimension
Column name
Unique and grouping values
Missing values

Feature Engineering

Create additional relevance features from the existing features in the raw data.

Try to increase the predictive power of the learning algorithm.

Data Manipulation

The process of changing data in an effort to make it easier to read and organize.

Exploratory Data Analysis (EDA)

Seeing what the data can tell us beyond the formal modeling or hypothesis testing tasks.

Exploratory Data Analysis (EDA)

The approach to analyzing datasets to summarize their main characteristics, **often with visual methods**.

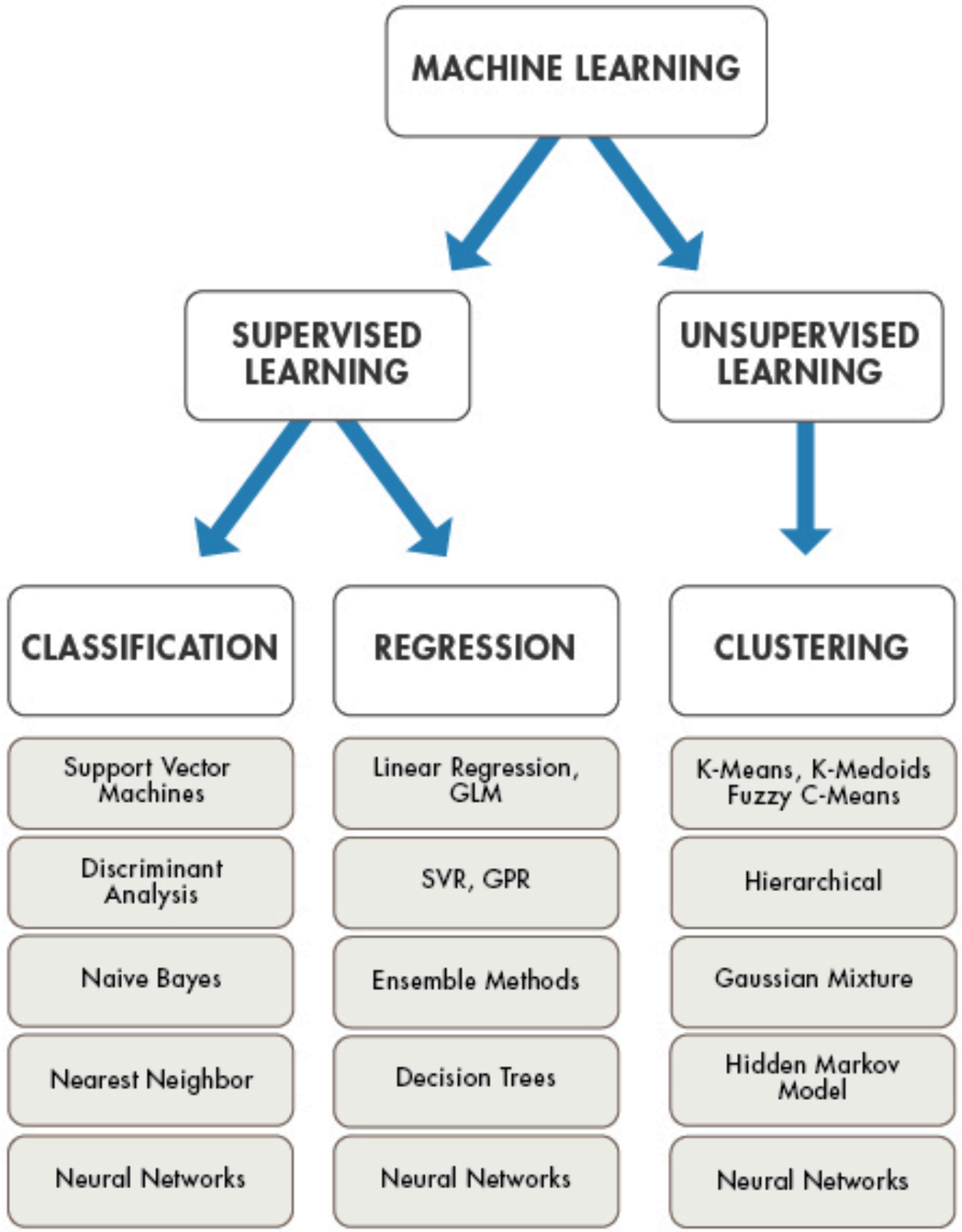
Machine Learning (ML)

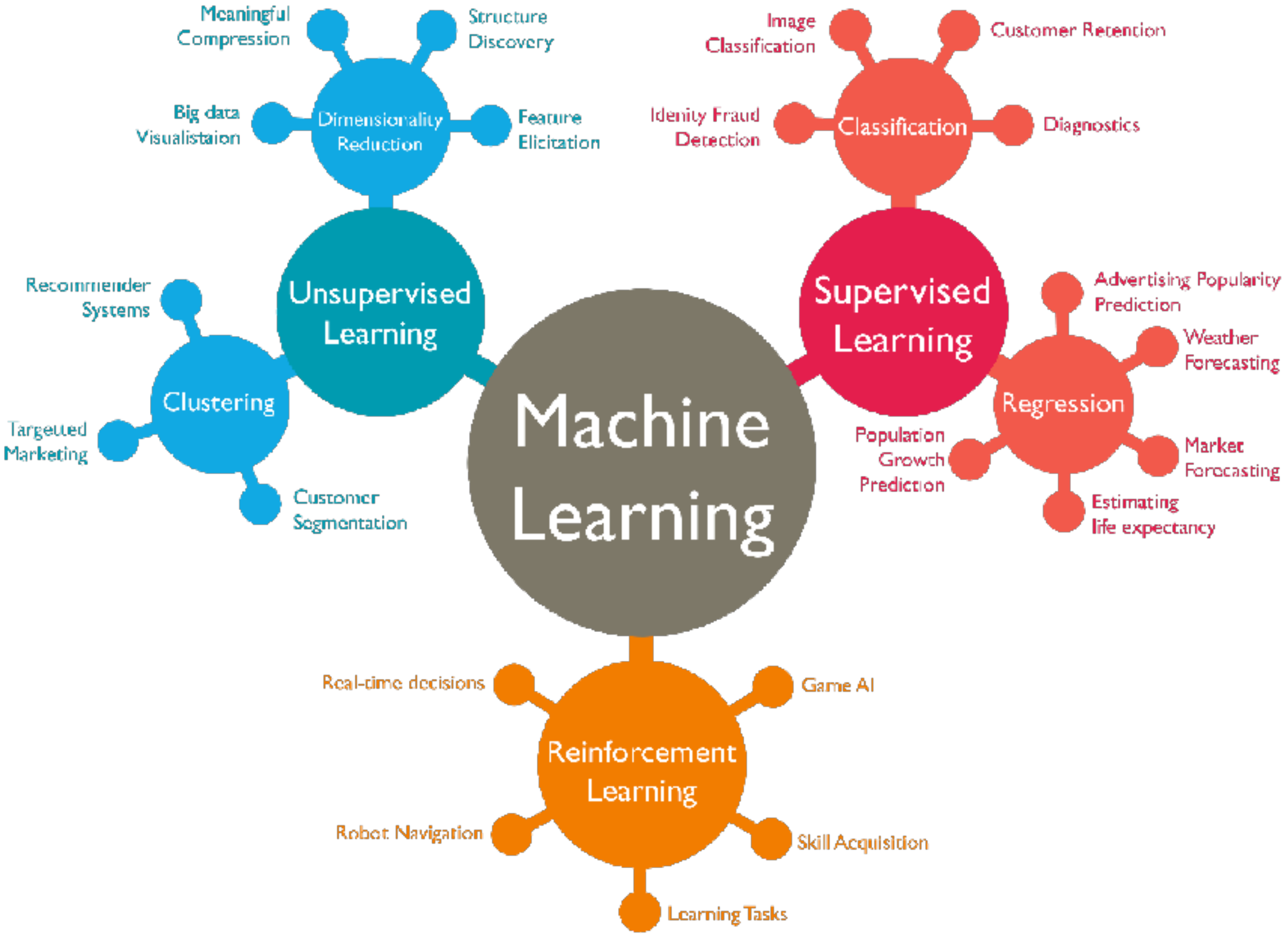
Machine Learning

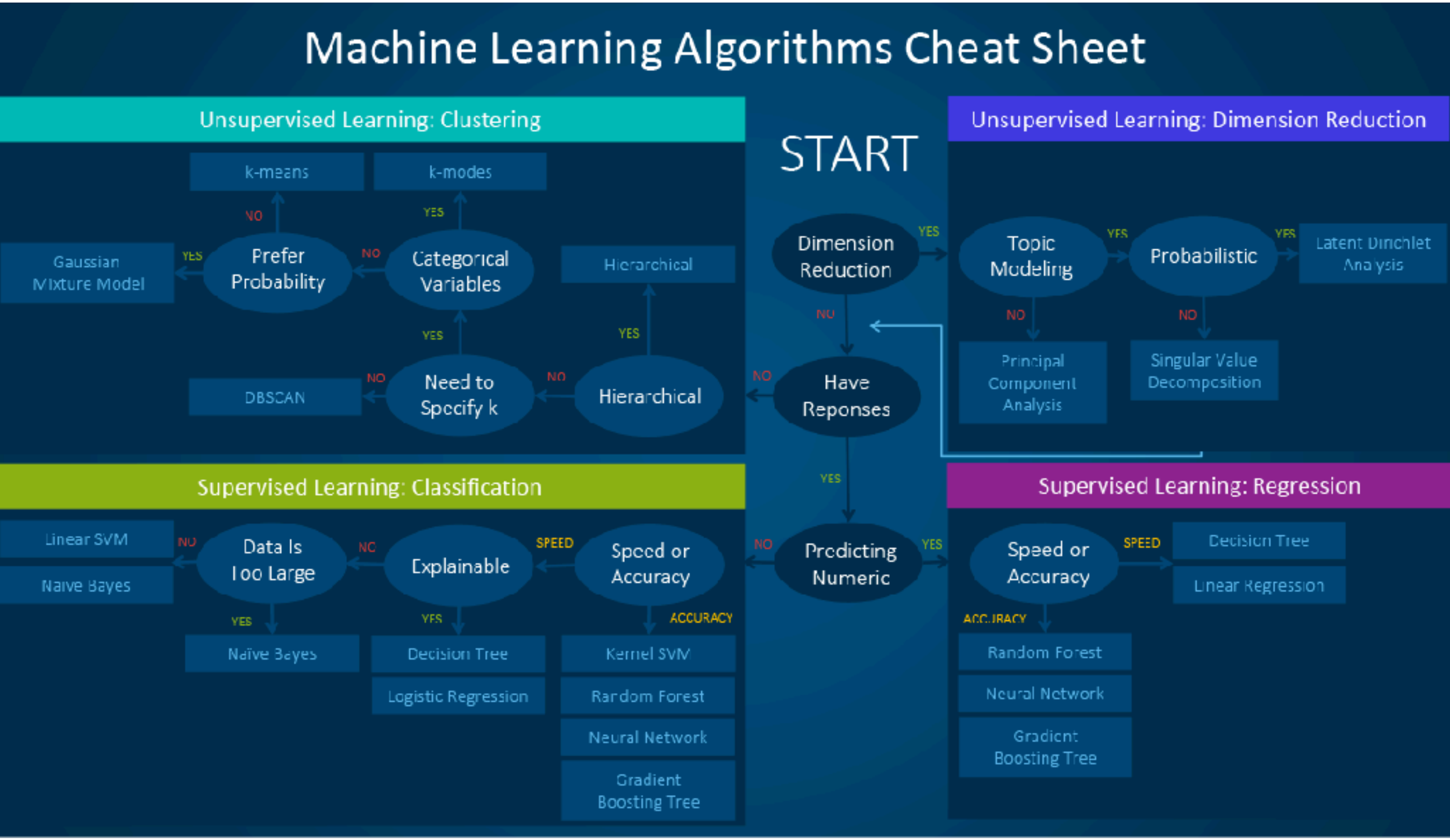
The application of AI that provides system the ability to **automatically learn and improve** experience without explicitly programmed.

Machine Learning

Focus on the development of computer program that can access data and use it to learn themselves.





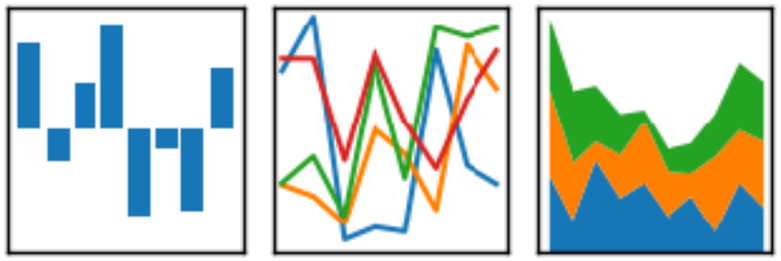


Libraries for Data Science

NumPy

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

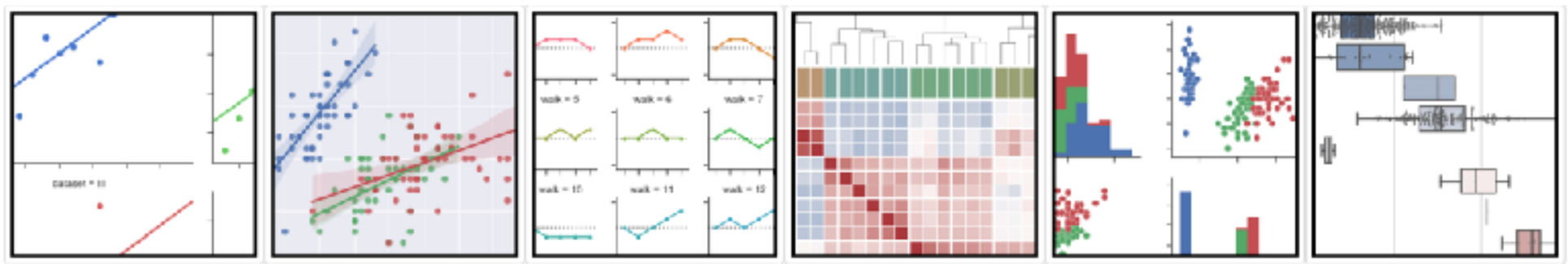


matplotlib

scikits learn

machine learning in Python

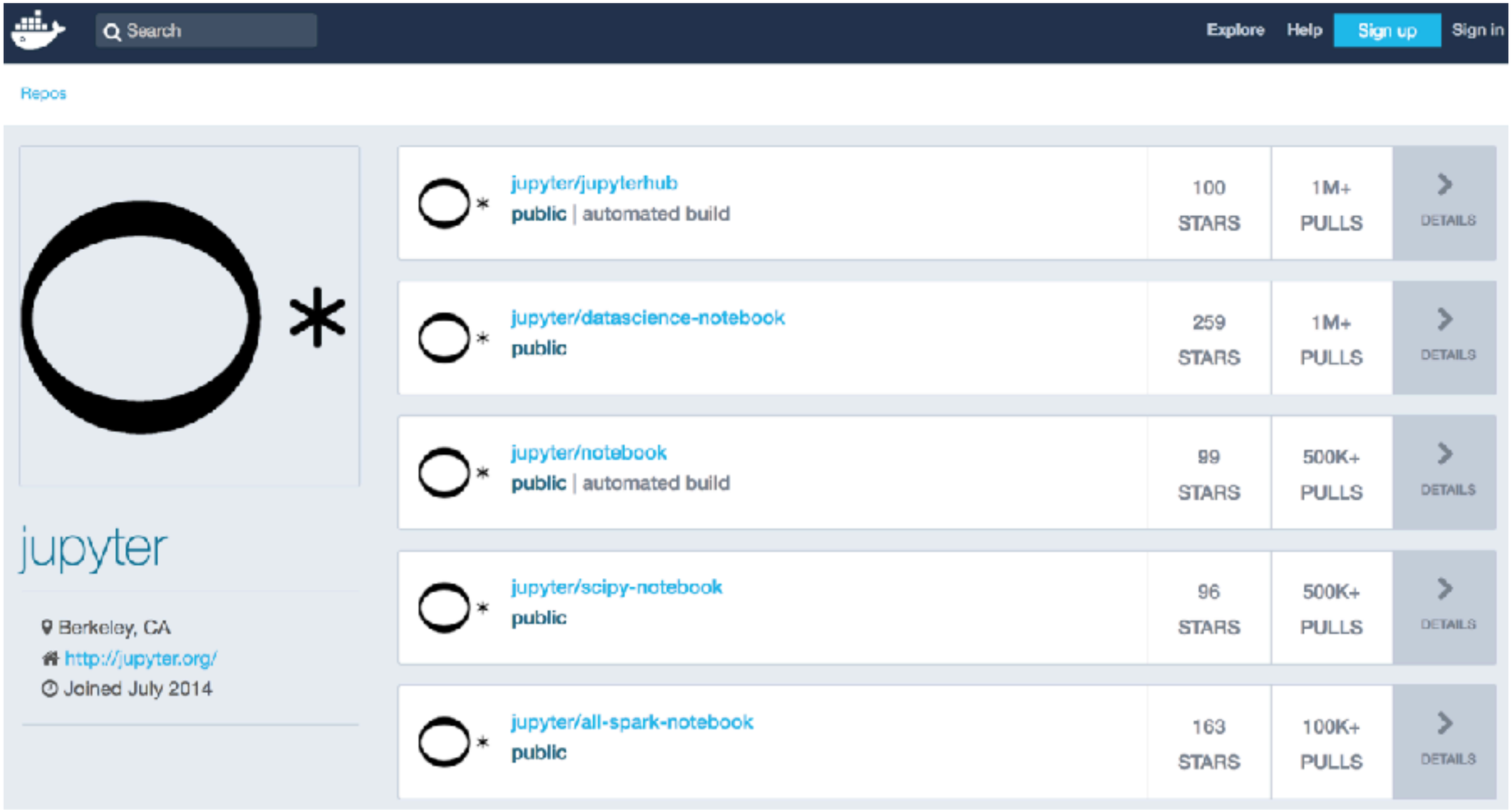
seaborn: statistical data visualization



Tools

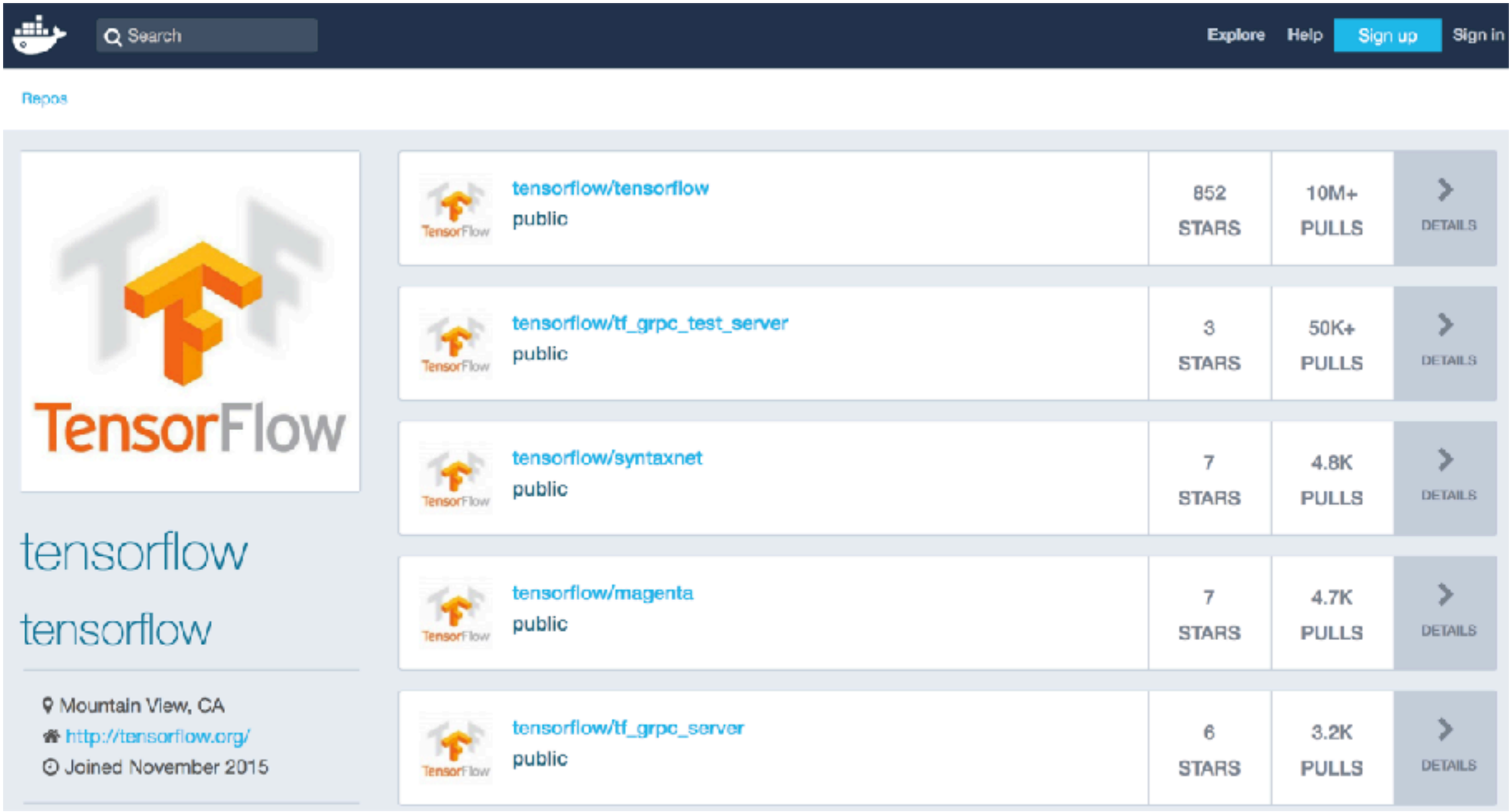
Docker for Data Science

Jupyter Images



https://hub.docker.com/u/jupyter/

TensorFlow Images



https://hub.docker.com/u/tensorflow/

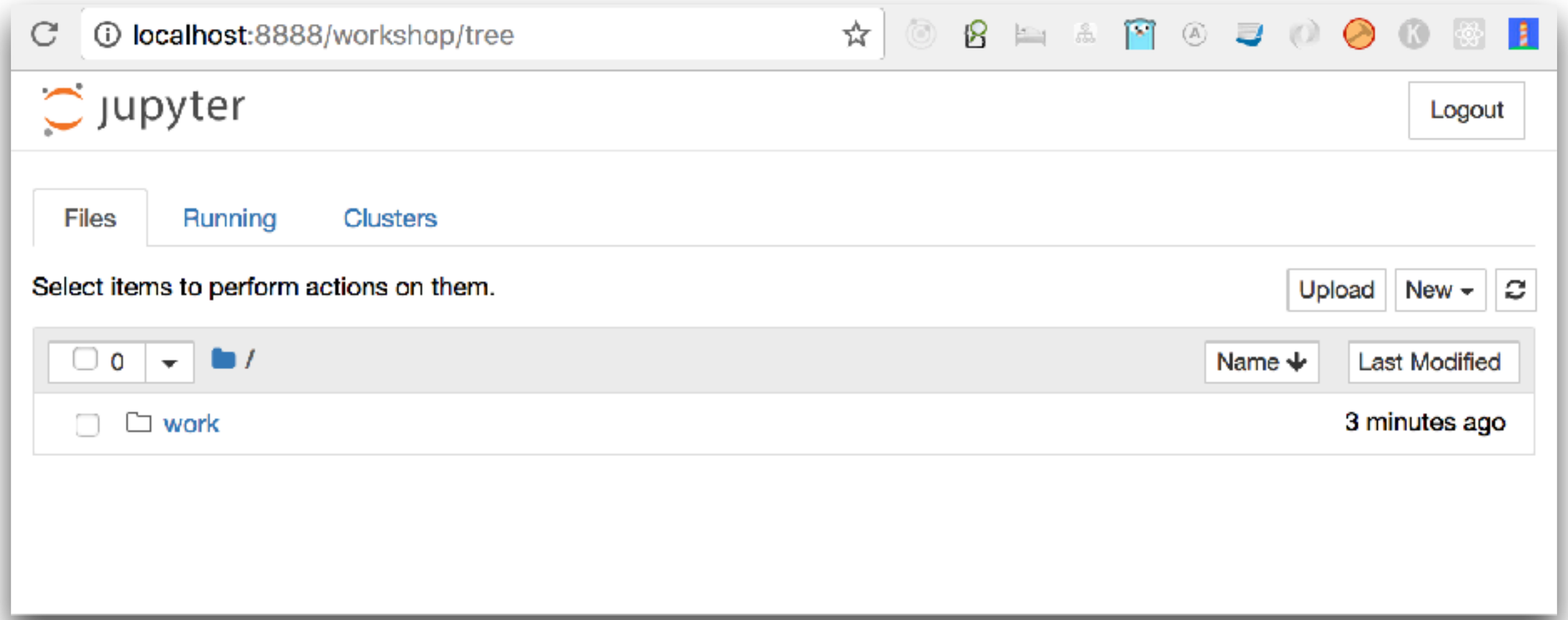
Install Jupyter with Docker

$docker pull jupyter/datascience-notebook

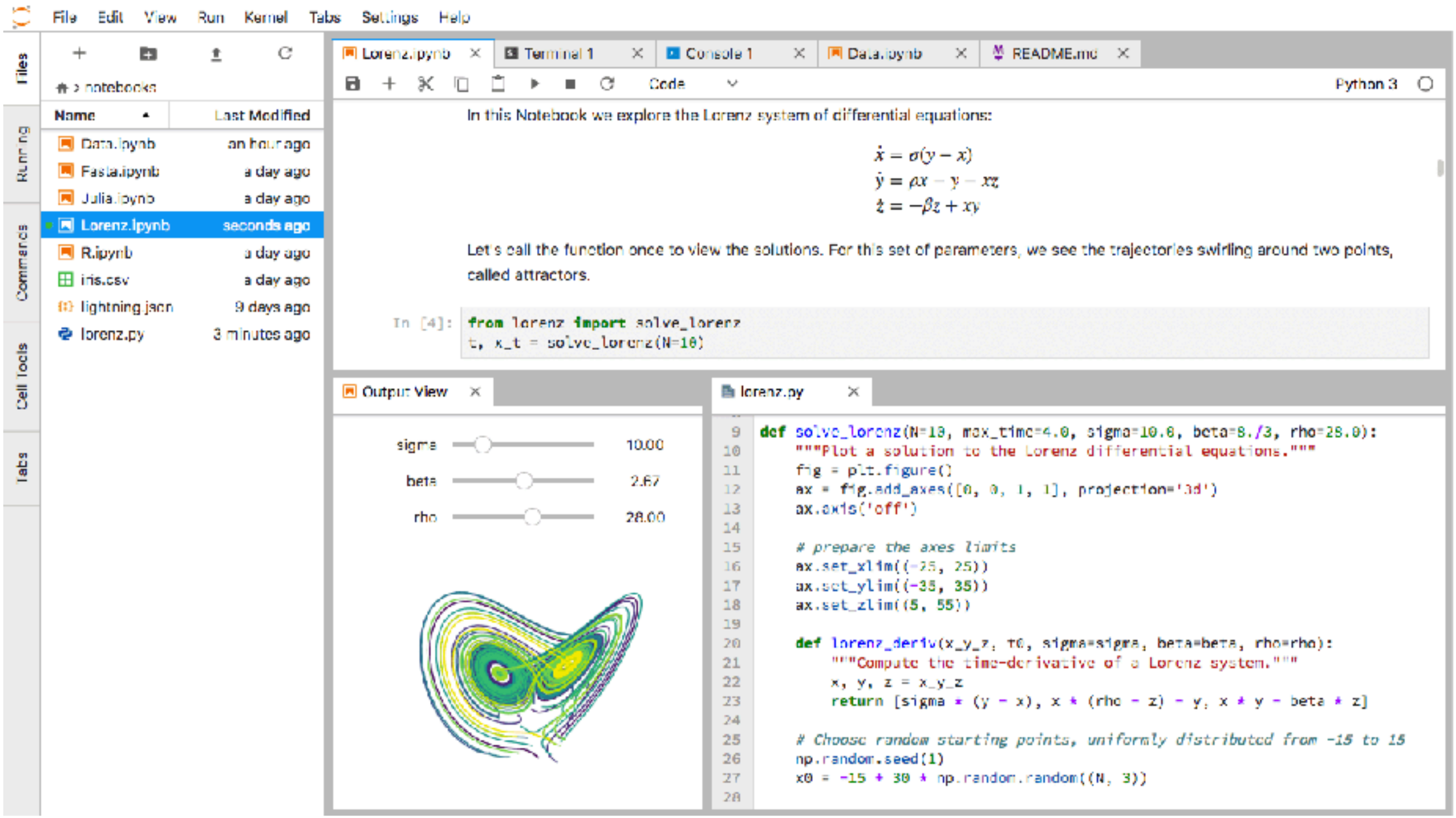
Install Jupyter with Docker

```
$docker container run -d -p 8888:8888
    -v $(pwd):/home/jovyan/work
    jupyter/datascience-notebook
    start-notebook.sh
    --NotebookApp.base_url=/workshop/
```

Hello Jupyter



JupyterLab



http://jupyterlab.readthedocs.io/en/stable/

Working with data

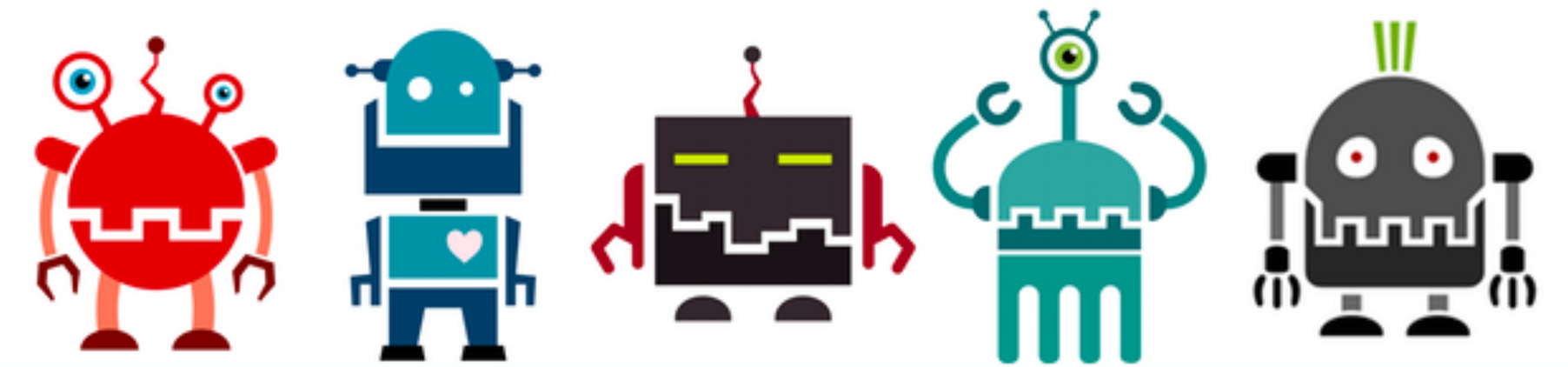
Python for Data Science Cheat Sheet

https://s3.amazonaws.com/assets.datacamp.com/blog_assets/PythonForDataScience.pdf

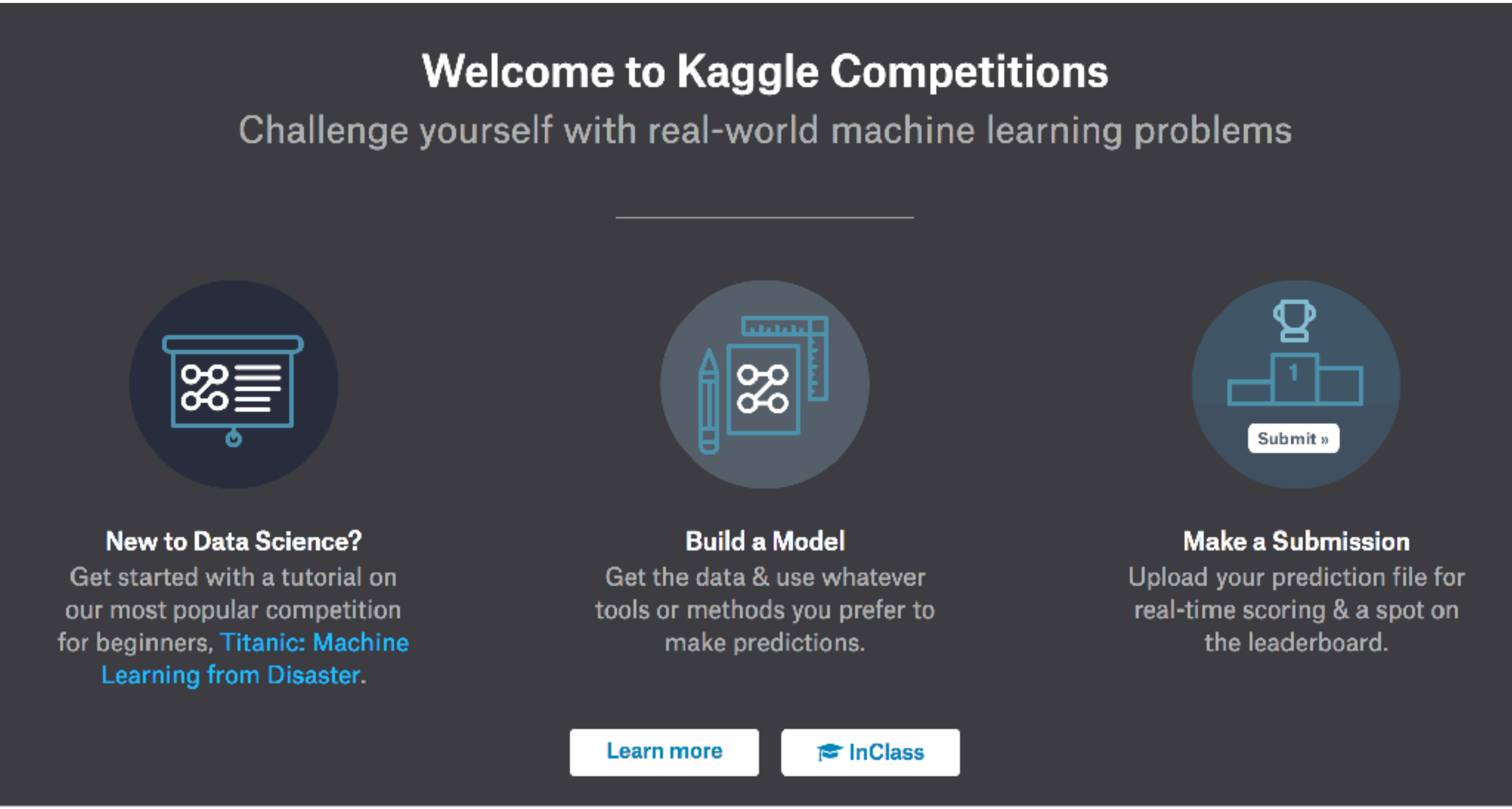
Choosing the good charts

http://extremepresentation.typepad.com/blog/files/choosing_a_good_chart.pdf

kaggle



Home of Data Science



Working with Data

https://www.kaggle.com/c/titanic/

Exploratory Data Analysis

Exploratory Data Analysis (EDA)

Seeing what the data can tell us beyond the formal modeling or hypothesis testing tasks.

Exploratory Data Analysis (EDA)

Data extraction
Data cleaning
Data plotting
Make assumption

Read data

Using Pandas for read data to data frame

```
import pandas as pd

titanic_df = pd.read_csv(“train.csv")
```

```
print(titanic_df)
print(titanic_df.head())
print(titanic_df.info())
```

Understanding data

Variable name	Description	Possible value
Survived		0 = No 1 = Yes
Pclass	Passenger class	1 = First 2 = Second 3 = Third
Name	Prefix + name	
Sex	Sex	male female
Age	Age	
SibSp	Number of sibling/spouse	

Understanding data

Variable name	Description	Possible value
Parch	Number of parent/children	
Ticket	Ticket number	
Fare	Passenger fare	
Cabin	Cabin	
Embarked	Port of embarkation	C = Cherbourg Q = Queenstown S = Southampton

Understanding data

The overall chance of survival for a passenger

```
In [4]: titanic_df['Survived'].mean()

Out[4]: 0.3838383838383838381
```

38% of passengers survived

Understanding data

Class of passenger vs. % of survived ?

```
titanic_df.groupby('Pclass').mean()
```

Pclass	PassengerId	Survived	Age	SibSp	Parch	Fare
1	461.597222	0.629630	38.233441	0.416667	0.356481	84.154687
2	445.956522	0.472826	29.877630	0.402174	0.380435	20.662183
3	439.154786	0.242363	25.140620	0.615071	0.393075	13.675550

First class has a 62% chance of survival

Understanding data

Class of passenger vs. % of survived ?

```
titanic_df[['Pclass', 'Survived']].groupby(
    ['Pclass'], as_index=False).mean()
```

	Pclass	Survived
0	1	0.629630
1	2	0.472826
2	3	0.242363

First class has a 62% chance of survival

Understanding data

Class of passenger vs. % of survived ?

```
titanic_df.groupby('Pclass').mean()
```

Pclass	PassengerId	Survived	Age	SibSp	Parch	Fare
1	461.597222	0.629630	38.233441	0.416667	0.356481	84.154687
2	445.956522	0.472826	29.877630	0.402174	0.380435	20.662183
3	439.154786	0.242363	25.140620	0.615071	0.393075	13.675550

Lower classes consisted of younger people

Understanding data

Grouping data of **Pclass** and **Sex**

```
pclass_sex = titanic_df.groupby(["Pclass", "Sex"]).mean()
pclass_sex
```

Pclass	Sex	PassengerId	Survived	Age	SibSp	Parch	Fare
1	female	469.212766	0.968085	34.611765	0.553191	0.457447	106.125798
	male	455.729508	0.368852	41.281386	0.311475	0.278689	67.226127
2	female	443.105263	0.921053	28.722973	0.486842	0.605263	21.970121
	male	447.962963	0.157407	30.740707	0.342593	0.222222	19.741782
3	female	399.729167	0.500000	21.750000	0.895833	0.798611	16.118810
	male	455.515850	0.135447	26.507589	0.498559	0.224784	12.661633

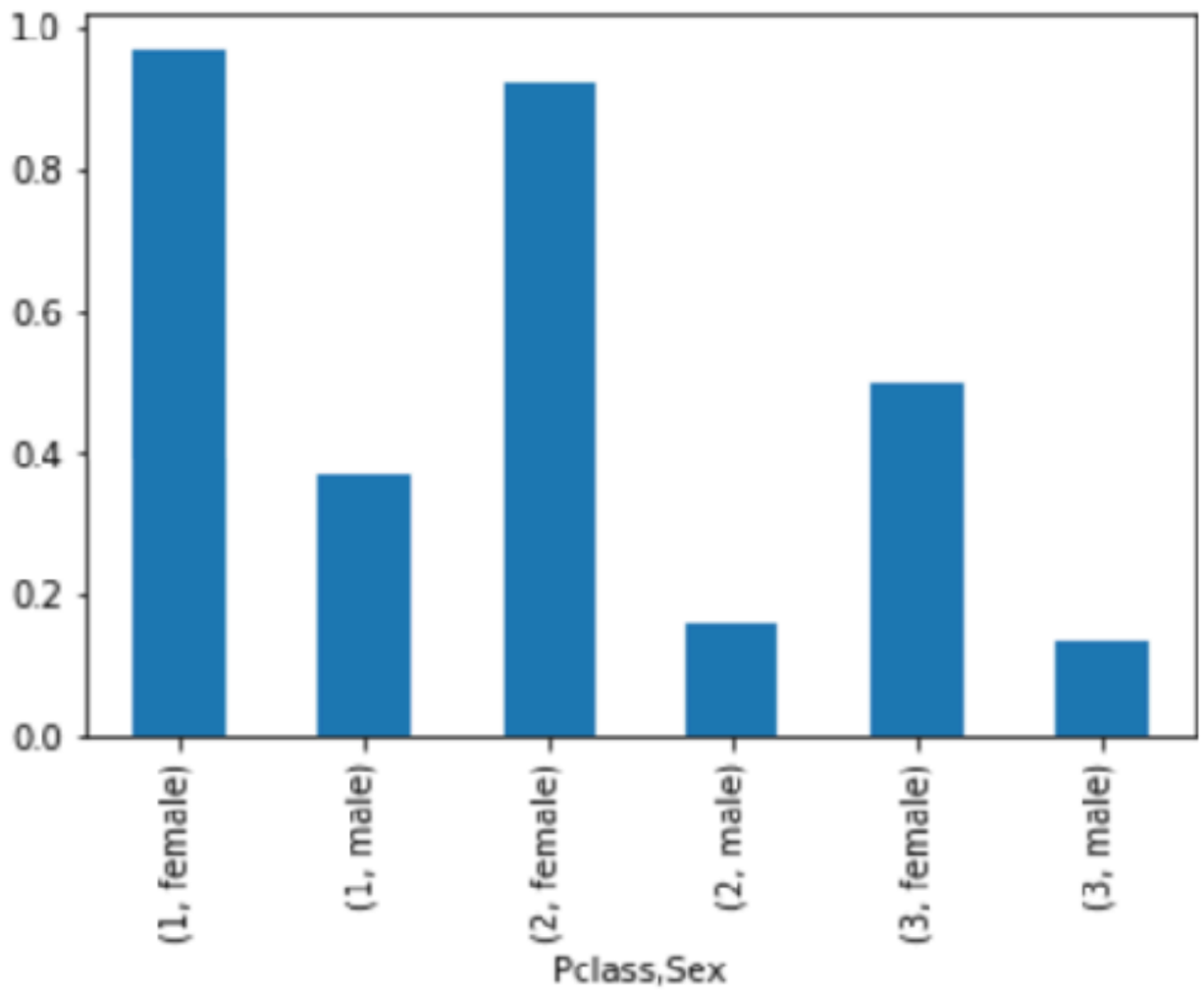
Female > Male of survival

Understanding data

Grouping data of **Pclass** and **Sex**

```
pclass_sex["Survived"].plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcd54c68710>
```



Understanding data

Female and children first ?

```
pclass_sex = titanic_df.groupby(["Pclass", "Sex"]).mean()
pclass_sex
```

Pclass	Sex	PassengerId	Survived	Age	SibSp	Parch	Fare
1	female	469.212766	0.968085	34.611765	0.553191	0.457447	106.125798
	male	455.729508	0.368852	41.281386	0.311475	0.278689	67.226127
2	female	443.105263	0.921053	28.722973	0.486842	0.605263	21.970121
	male	447.962963	0.157407	30.740707	0.342593	0.222222	19.741782
3	female	399.729167	0.500000	21.750000	0.895833	0.798611	16.118810
	male	455.515850	0.135447	26.507589	0.498559	0.224784	12.661633

Understanding data

Female and children first ?

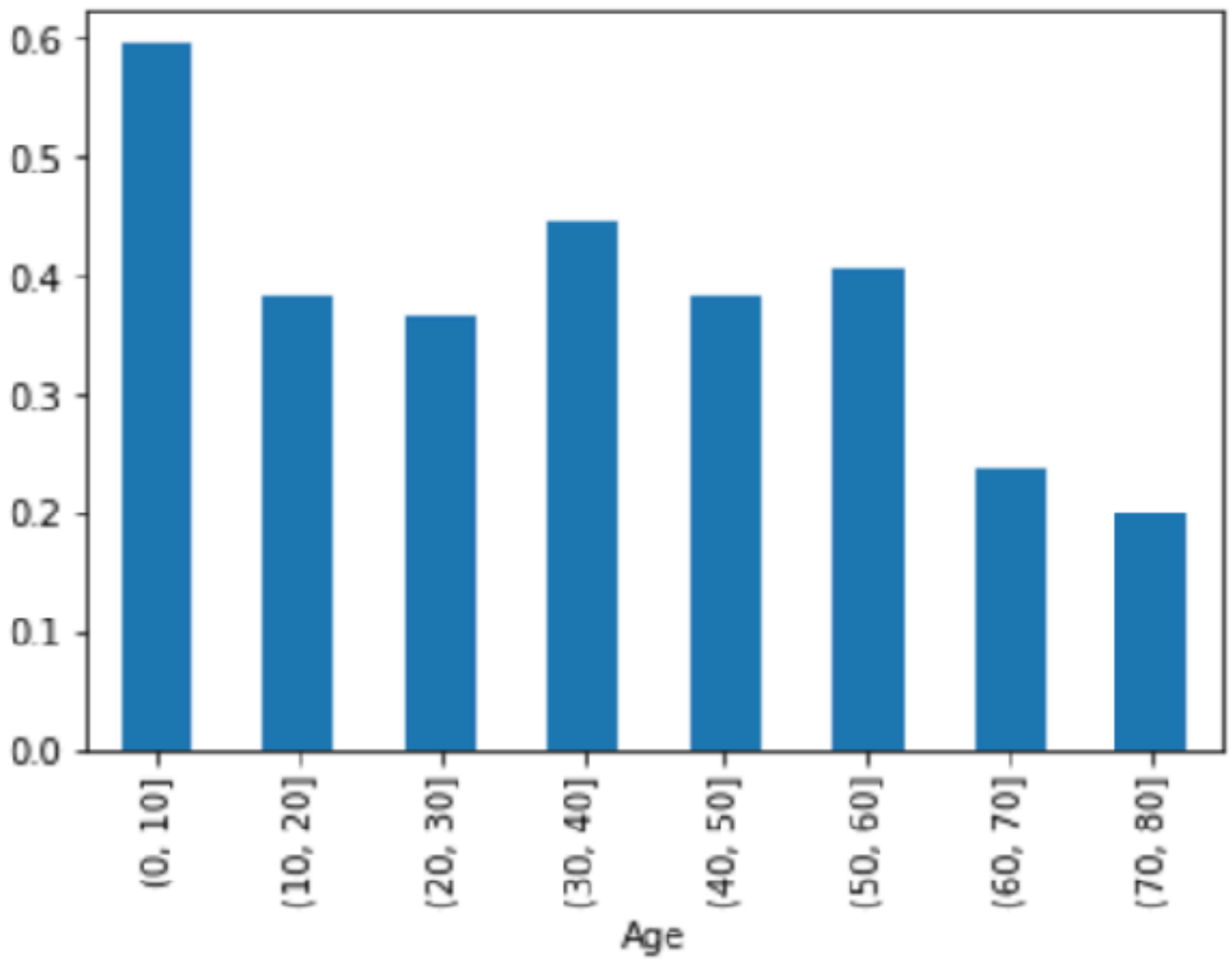
```
group_age = pd.cut(titanic_df["Age"],
                   np.arange(0, 90, 10))
age_grouping = titanic_df.groupby(group_age).mean()
```

Age	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
(0, 10]	430.843750	0.593750	2.640625	4.268281	1.843750	1.421875	30.434439
(10, 20]	447.660870	0.382609	2.530435	17.317391	0.591304	0.391304	29.529531
(20, 30]	428.682609	0.365217	2.386957	25.423913	0.321739	0.239130	28.306719
(30, 40]	468.690323	0.445161	2.090323	35.051613	0.374194	0.393548	42.496100
(40, 50]	483.500000	0.383721	1.918605	45.372093	0.372093	0.430233	41.163181
(50, 60]	449.809524	0.404762	1.523810	54.892857	0.309524	0.309524	44.774802
(60, 70]	430.882353	0.235294	1.529412	63.882353	0.176471	0.352941	45.910782
(70, 80]	438.200000	0.200000	1.800000	73.300000	0.000000	0.000000	25.936680

Understanding data

```
age_grouping["Survived"].plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcd1ea9c8d0>
```



See your data

```
titanic_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
```

Missing value

Missing value

Pclass

No missing value, numeric value

```
titanic_df[['Pclass', 'Survived']].groupby(
    ['Pclass'],
    as_index=False).mean()
```

	Pclass	Survived
0	1	0.629630
1	2	0.472826
2	3	0.242363

Sex

No missing value

```
titanic_df[['Sex', 'Survived']].groupby(
    ['Sex'],
    as_index=False).mean()
```

	Sex	Survived
0	female	0.742038
1	male	0.188908

Handling Missing Values

See your data

```
titanic_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
```

Missing value

Missing value

Age

Age have missing value

```
titanic_df.Age.unique()
```

Outlier ?

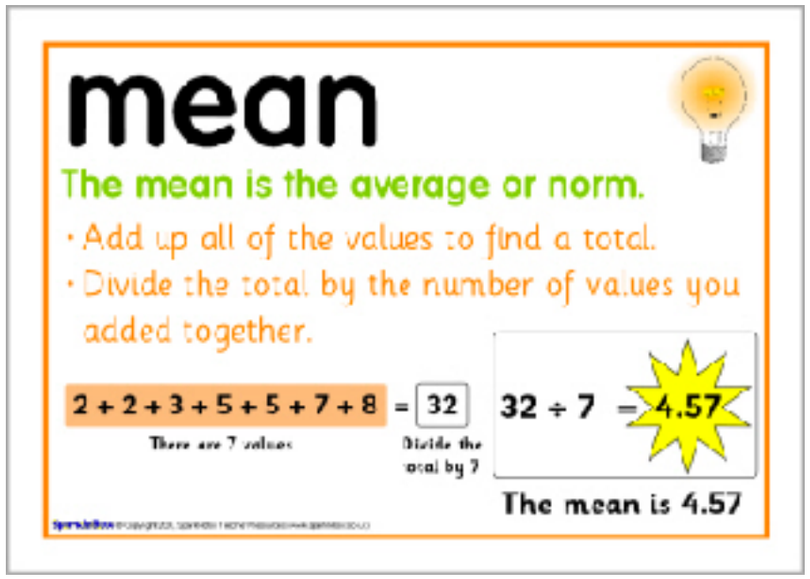
```
array([ 22.  ,  38.  ,  26.  ,  35.  ,     nan,  54.  ,   2.  ,  27.  ,
        14.  ,   4.  ,  58.  ,  20.  ,  39.  ,  55.  ,  31.  ,  34.  ,
        15.  ,  28.  ,   8.  ,  19.  ,  40.  ,  66.  ,  42.  ,  21.  ,
        18.  ,   3.  ,   7.  ,  49.  ,  29.  ,  65.  ,  28.5 ,   5.  ,
        11.  ,  45.  ,  17.  ,  32.  ,  16.  ,  25.  ,   0.83,  30.  ,
        33.  ,  23.  ,  24.  ,  46.  ,  59.  ,  71.  ,  37.  ,  47.  ,
        14.5 ,  70.5 ,  32.5 ,  12.  ,   9.  ,  36.5 ,  51.  ,  55.5 ,
        40.5 ,  44.  ,   1.  ,  61.  ,  56.  ,  50.  ,  36.  ,  45.5 ,
        20.5 ,  62.  ,  41.  ,  52.  ,  63.  ,  23.5 ,   0.92,  43.  ,
        60.  ,  10.  ,  64.  ,  13.  ,  48.  ,   0.75,  53.  ,  57.  ,
        80.  ,  70.  ,  24.5 ,   6.  ,   0.67,  30.5 ,   0.42,  34.5 ,  7
4.   ])
```

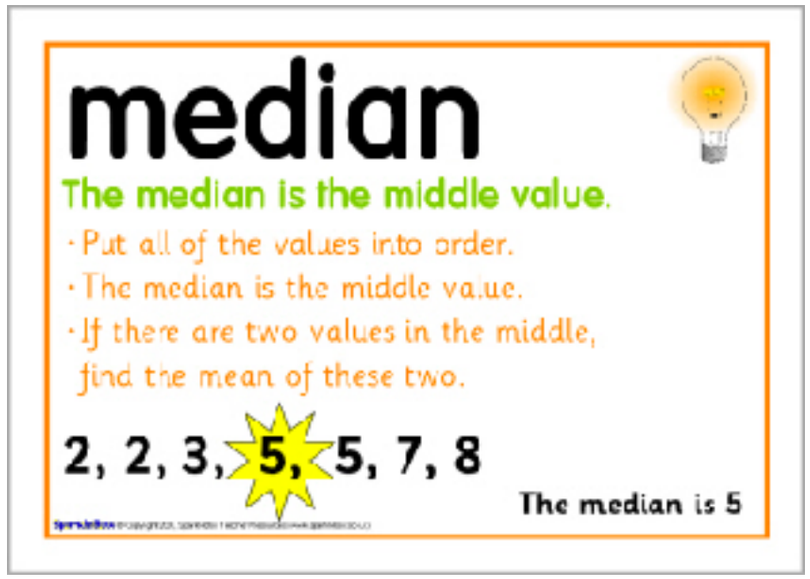
How to handling missing value of Age ?

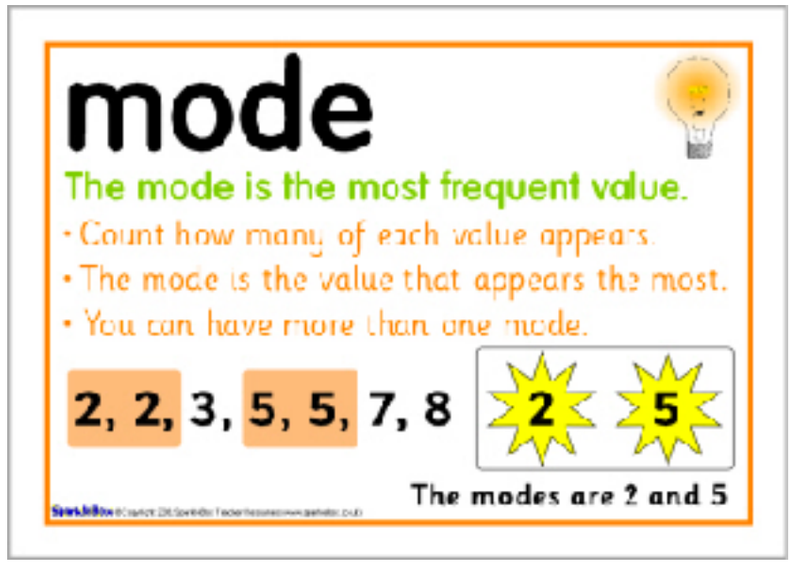
Age

Q: How to handling missing value ?

A: Replace with **mean**, **medium**, **mode** value







Age

Replace missing value with median

```
titanic_df['NewAge'] =
        titanic_df.Age.fillna(titanic_df.Age.median())

titanic_df[['Age', 'NewAge']].describe()
```

	Age	NewAge
count	714.000000	891.000000
mean	29.699118	29.361582
std	14.526497	13.019697
min	0.420000	0.420000
25%	20.125000	22.000000
50%	28.000000	28.000000
75%	38.000000	35.000000
max	80.000000	80.000000

Embarked

Embarked have missing value

```
print('Survived =>', titanic_df.Survived.unique())
print('Pclass =>', titanic_df.Pclass.unique())
print('Sex =>', titanic_df.Sex.unique())
print('SibSp =>', titanic_df.SibSp.unique())
print('Parch =>', titanic_df.Parch.unique())
print('Embarked =>', titanic_df.Embarked.unique())
```

```
Survived => [0 1]
Pclass => [3 1 2]
Sex => ['male' 'female']
SibSp => [1 0 3 4 2 5 8]
Parch => [0 1 2 5 3 4 6]
Embarked => ['S' 'C' 'Q' nan]
```

Embarked

How to handling missing value ?

```
titanic_df[['Embarked']].describe()
```

	Embarked
count	889
unique	3
top	S
freq	644

Replace with popular value ?

Embarked

Replace missing value with D

```
titanic_df['Embarked'] = titanic_df['Embarked'].fillna('S')
titanic_df[['Embarked']].describe()
```

	Embarked
count	891
unique	3
top	S
freq	646

Cabin

TODO

Feature Engineering

Feature Engineering

Create additional relevance features from the existing features in the raw data.

Try to increase the predictive power of the learning algorithm.

Name

Try to see data of Name

```
titanic_df.Name.tail()
```

```
886                                  Montvila, Rev. Juozas
887                           Graham, Miss. Margaret Edith
888               Johnston, Miss. Catherine Helen "Carrie"
889                                  Behr, Mr. Karl Howell
890                                    Dooley, Mr. Patrick
Name: Name, dtype: object
```

Bad data have Title and Name

Name

Working with regular expression !!

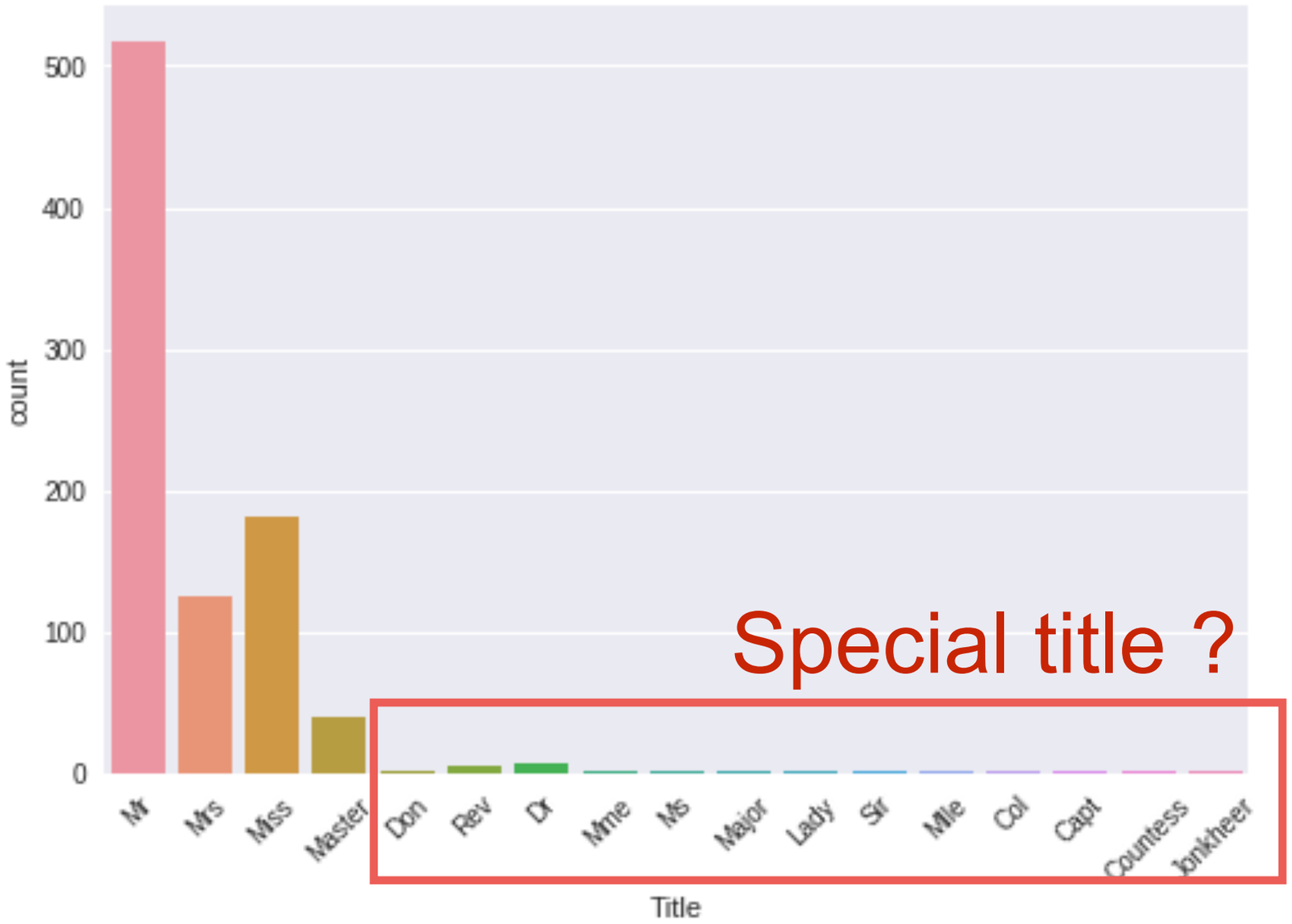
```
import seaborn as sns
import re
```

```
titanic_df['Title']
    = titanic_df.Name.apply(
        lambda x: re.search(' ([A-Z][a-z]+)\.', x).group(1))
```

```
sns.countplot(x='Title', data=titanic_df);
plt.xticks(rotation=45);
```

Name

Try to grouping title of name



Name

Extract to new function, make easier to read

```
import pandas as pd
import re

def get_title(name):
    title = re.search(' ([A-Za-z]+)\.', name)
    if title:
        return title.group(1)
    return ""

if __name__== "__main__":
    titanic_df = pd.read_csv("train.csv")
    titanic_df['Title'] = titanic_df['Name'].apply(get_title)
    print(pd.crosstab(titanic_df['Title'], titanic_df['Sex']))
```

Name

Try to grouping title of name

```
pd.crosstab(titanic_df['Title'], titanic_df['Sex'])
```

Sex	female	male
Title		
Capt	0	1
Col	0	2
Countess	1	0
Don	0	1
Dr	1	6
Jonkheer	0	1
Lady	1	0
Major	0	2
Master	0	40
Miss	182	0
Mlle	2	0
Mme	1	0
Mr	0	517
Mrs	125	0
Ms	1	0
Rev	0	6
Sir	0	1

Name

Try to grouping title of name

Special title ?

Capt	1
Col	2
Countess	1
Don	1
Dr	7
Jonkheer	1
Lady	1
Major	2
Master	40
Miss	182
Mlle	2
Mme	1
Mr	517
Mrs	125
Ms	1
Rev	6
Sir	1

Special title ?

Name

Replace title with new title ?

```
pd.crosstab(titanic_df['Title'], titanic_df['Sex'])
```

Sex	female	male
Title		
Capt	0	1
Col	0	2
Countess	1	0
Don	0	1
Dr	1	6
Jonkheer	0	1
Lady	1	0
Major	0	2
Master	0	40
Miss	182	0
Mlle	2	0
Mme	1	0
Mr	0	517
Mrs	125	0
Ms	1	0
Rev	0	6
Sir	0	1

Mlle

Abbreviation of Mademoiselle

Noun [edit]

Mademoiselle *f* (*plural* **Mesdemoiselles** *or* **Mademoiselles**)

1. *A title or form of address for a girl or a young woman*, Miss.

https://en.wiktionary.org/wiki/Mademoiselle#French

Ms

Ms.

From Wikipedia, the free encyclopedia

This article is about the title. For the magazine, see Ms. (magazine). For other uses, see MS (disambiguation).

"**Ms**" or "**Ms.**" (normally /mɪz/, but also /məz/, or /məs/ when unstressed)[1][2] is an English honorific used with the last name or full name of a woman, intended as a default form of address for women regardless of their marital status.[3] Like "Miss" and "Mrs.", the term "Ms." has its origins in the female English title once used for all women, "Mistress". It has its origin in the 17th century and was revived into mainstream usage in the 20th century.[4] In the UK and the majority of Commonwealth countries, a full stop (period) is usually not used with the title; in the United States and Canada a period is usually used (see Abbreviation).[1][5]

Contents [hide]

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

Mme

Madam, a married woman

MME may stand for:

- *M*^{me} or *Mme*, the French abbreviation for *Madame*
- MME, the IATA code for Durham Tees Valley Airport, England
- MME (psychedelic), 2,4-dimethoxy-5-ethoxyamphetamine, a p
- Madam, a married woman or a female brothel manager
- Magyar Madártani és Természetvédelmi Egyesület, Hungarian
- Mass mortality event, the naturally occurring death of large nu

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

Name

Replace **Mlle** with **Miss**

Replace **Ms** with **Miss**

Replace **Mme** with **Mrs**

Replace **Special** with **rare titles !!**

Name

Replace title with new title

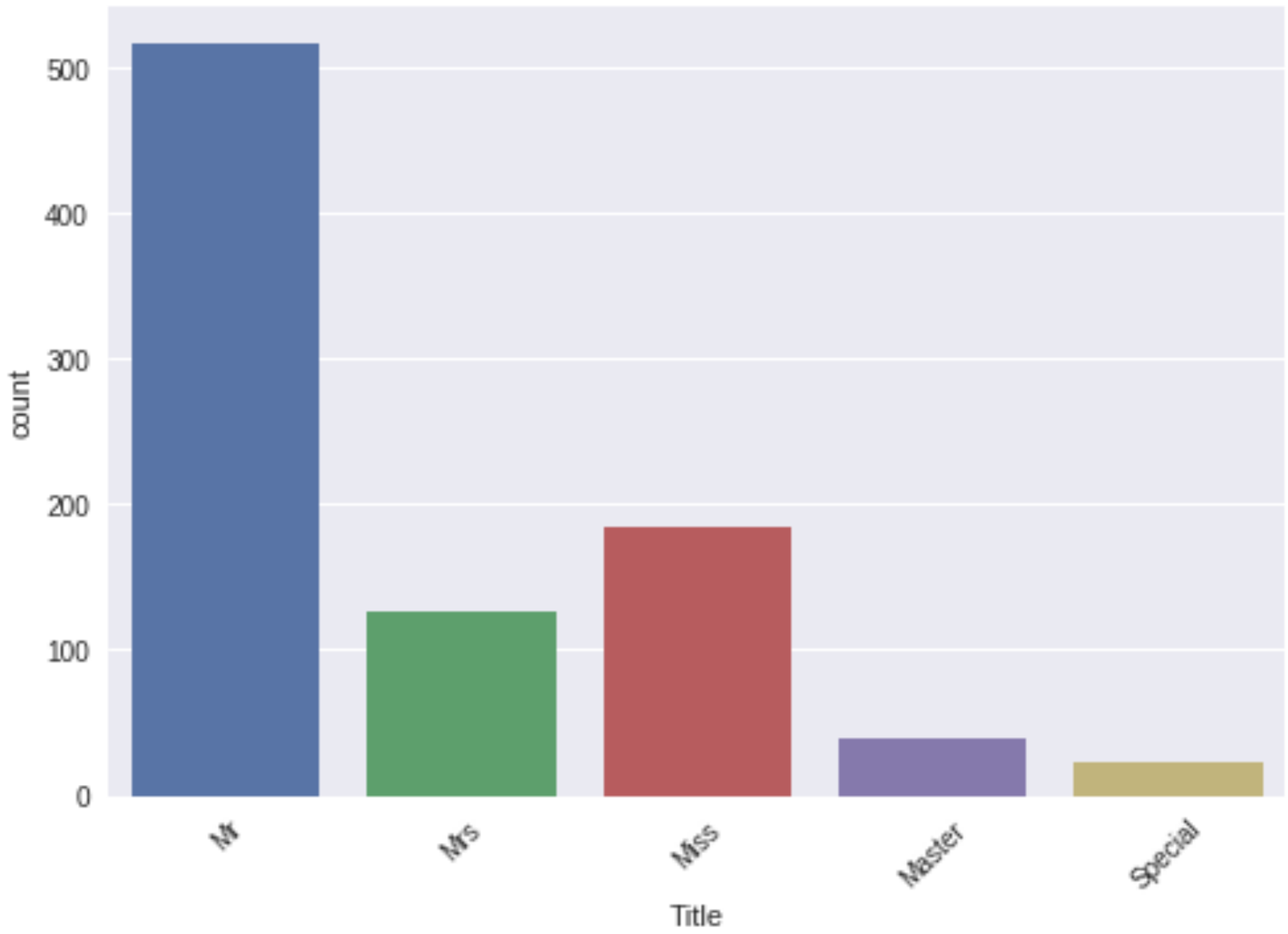
```
titanic_df['Title']
    = titanic_df['Title'].replace({'Mlle':'Miss', 'Mme':'Mrs', 'Ms':'Miss'})

titanic_df['Title']
    = titanic_df['Title'].replace(['Don', 'Dona', 'Rev', 'Dr',
                                   'Major', 'Lady', 'Sir', 'Col',
                                   'Capt', 'Countess', 'Jonkheer'],'Special')

sns.countplot(x='Title', data=titanic_df);
plt.xticks(rotation=45);
```

Name

Replace title with new title



SibSp and Parch

We can create new feature !!

```
titanic_df['FamilySize']
      = titanic_df['SibSp'] + titanic_df['Parch'] + 1

titanic_df[['FamilySize', 'Survived']].groupby(
          ['FamilySize']
).mean()
```

Size of family ?

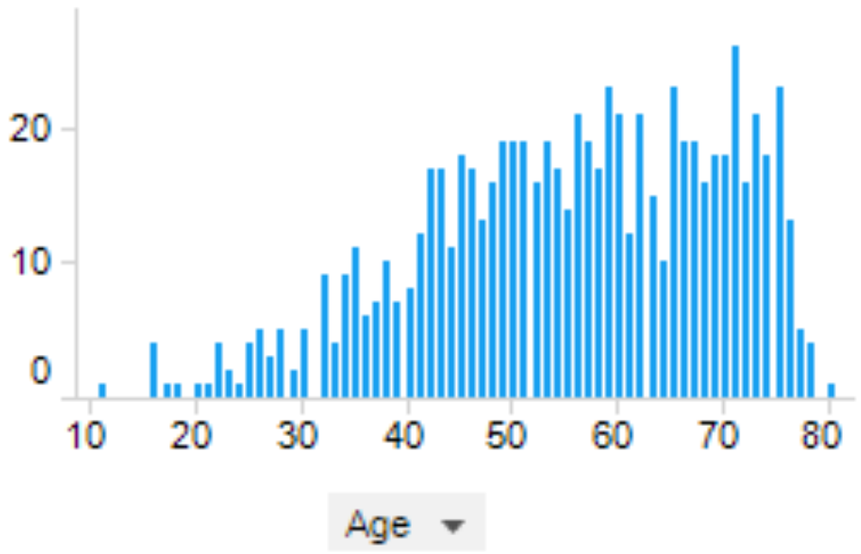
SibSp and Parch

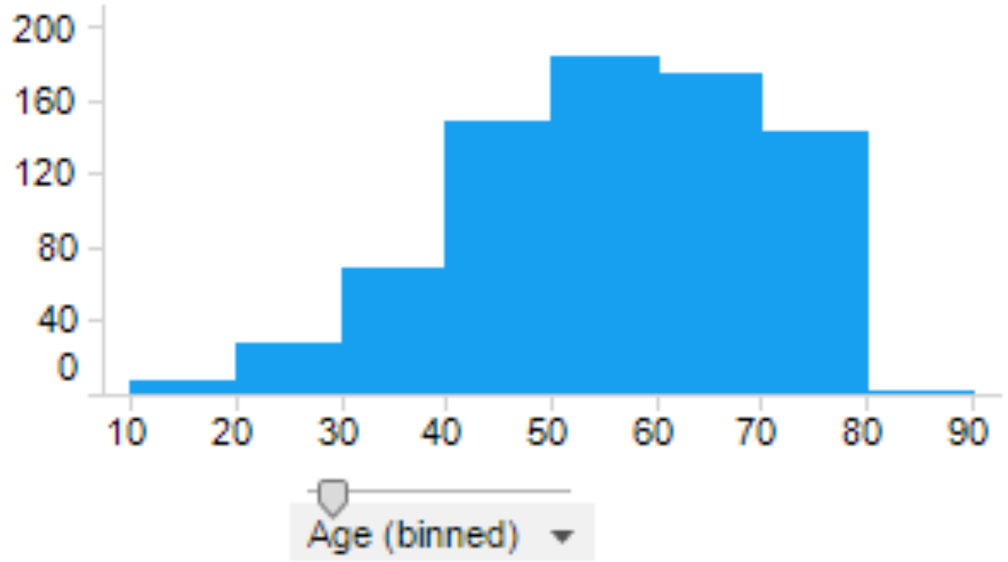
Size of family vs survival ?

```
titanic_df['FamilySize'] = titanic_df['SibSp'] + titanic_df['Parch'] + 1
titanic_df[['FamilySize', 'Survived']].groupby(
      ['FamilySize']
).mean()
```

FamilySize	Survived
1	0.303538
2	0.552795
3	0.578431
4	0.724138
5	0.200000
6	0.136364
7	0.333333
8	0.000000
11	0.000000

Bin numerical data





Fare and Age

Range of fare and age ?

Using function **qcut()** of pandas library

```
titanic_df['CategoricalFare']
                = pd.qcut(titanic_df['Fare'], 4, labels=False)
```

```
titanic_df['CategoricalAge']
                = pd.qcut(titanic_df['Age'], 4, labels=False)
```

	CategoricalFare	Survived
0	0	0.197309
1	1	0.303571
2	2	0.454955
3	3	0.581081

	CategoricalAge	Survived
0	0	0.424242
1	1	0.331169
2	2	0.437037
3	3	0.382488

https://pandas.pydata.org/pandas-docs/stable/generated/pandas.qcut.html

Transform datas to numerical

Transform data to numerical

1. Missing value
2. Binned numerical data
3. Transform all variables to numerical

Transform data to numerical

Using get_dummies() from pandas library

```
data_dummy = pd.get_dummies(titanic_df, drop_first=True)
data_dummy.head()
```

	Survived	Pclass	SibSp	Parch	FamilySize	NewAge	CategoricalFare	CategoricalAge
0	0	3	1	0	2	22.0	0	0
1	1	1	1	0	2	38.0	3	3
2	1	3	0	0	1	26.0	1	1
3	1	1	1	0	2	35.0	3	2
4	0	3	0	0	1	35.0	1	2

More …

End up with More features !!

Delete useless informations

Delete useless informations

```
titanic_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId     891 non-null int64
Survived        891 non-null int64
Pclass          891 non-null int64
Name            891 non-null object
Sex             891 non-null object
Age             714 non-null float64
SibSp           891 non-null int64
Parch           891 non-null int64
Ticket          891 non-null object
Fare            891 non-null float64
Cabin           204 non-null object
Embarked        889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
```

Delete useless informations

Name, PassengerID, Ticket, Cabin

```
titanic_df.drop(['Cabin', 'Name', 'PassengerId', 'Ticket'], axis=1, inplace
titanic_df.head()
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	FamilySize	Title	Prefix
0	0	3	male	22.0	1	0	7.2500	S	2	Mr	Mr
1	1	1	female	38.0	1	0	71.2833	C	2	Mrs	Mrs
2	1	3	female	26.0	0	0	7.9250	S	1	Miss	Miss
3	1	1	female	35.0	1	0	53.1000	S	2	Mrs	Mrs
4	0	3	male	35.0	0	0	8.0500	S	1	Mr	Mr

End up with More features !!

Feature selection

Feature Selection

After feature engineering process is done,
Dimension of data should be decreased
by selecting the **right features**

Benefits

Decrease redundancy of data

Speed up the training process

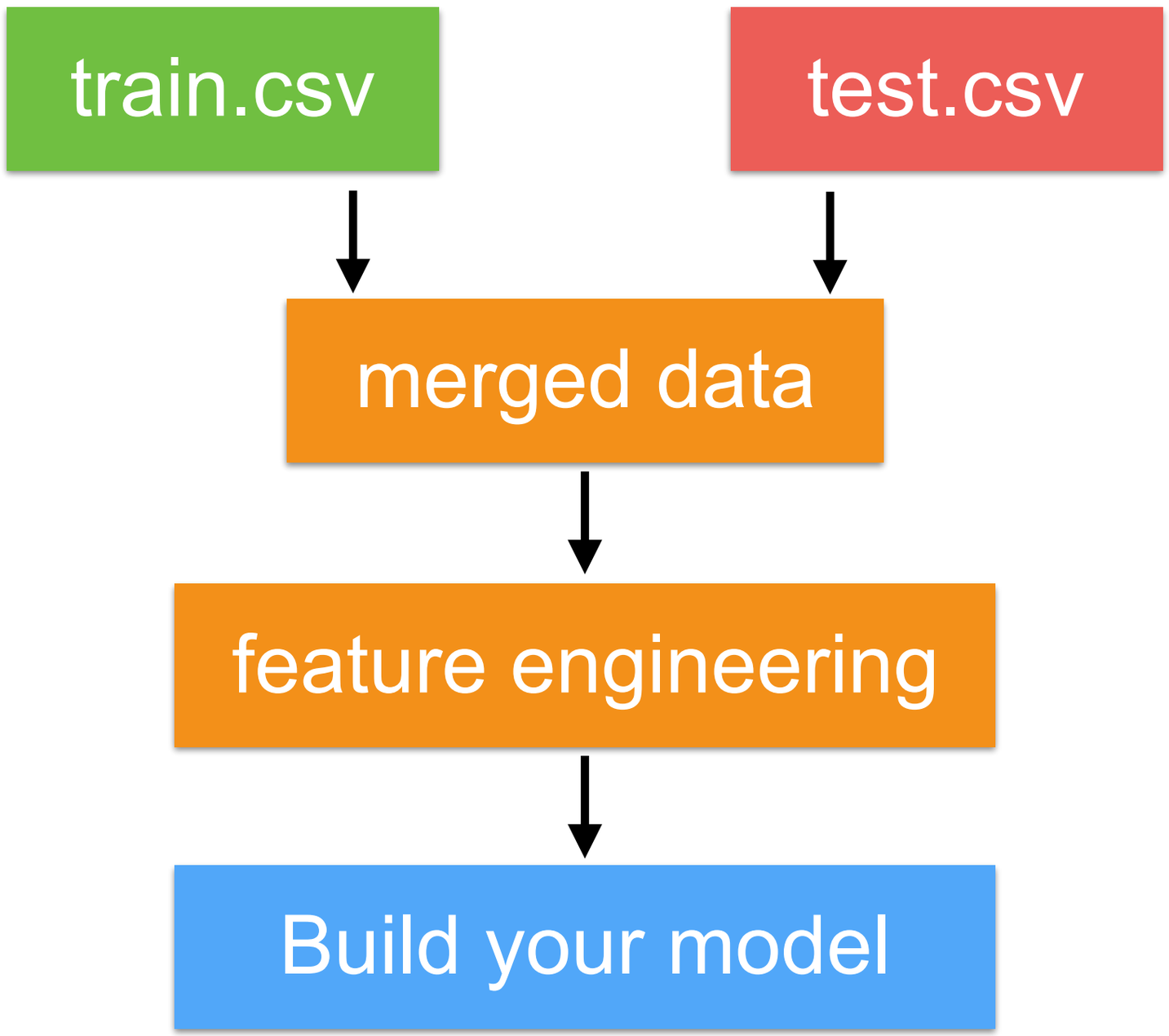
Reduce overfit

https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/

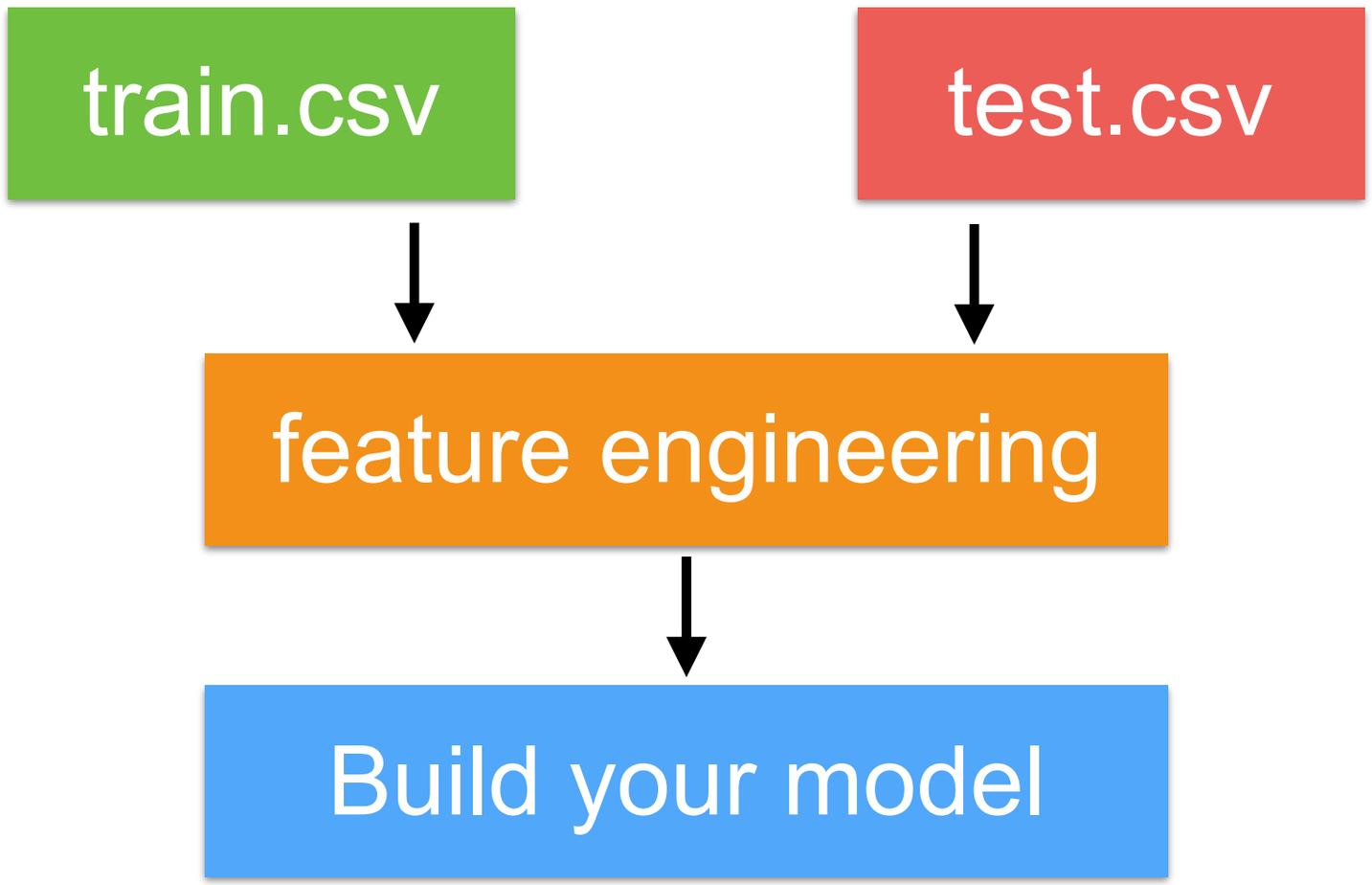
Build your model

Build simple Machine Learning Model

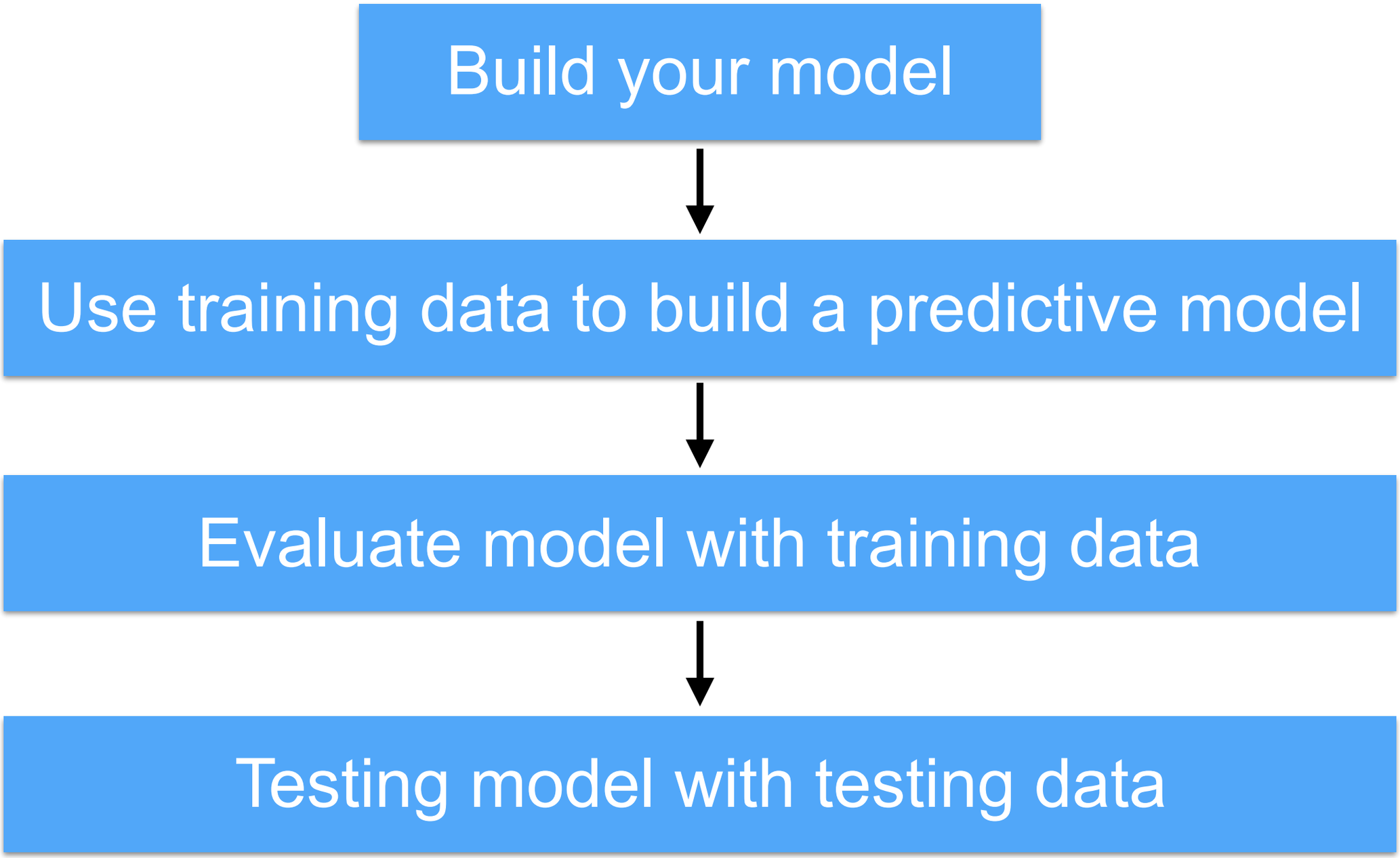
Build your model



Build your model



Build your model



Try to modeling

1. Combine data

Training and testing data for easy to manage

```
import pandas as pd

def combine_data():
    train_data = pd.read_csv('train.csv')
    testing_data = pd.read_csv('test.csv')

    train_data.drop('Survived', inplace=True, axis=1)

    data = pd.concat([train_data, testing_data], sort=False)
    data.drop(['PassengerId'], inplace=True, axis=1)
    return data

if __name__== "__main__":
    data = combine_data()
```

2. Feature engineering

Try to clean data and create new features from existing features

```
if __name__== "__main__":
    data = process_with_name(data)
    data = process_with_sex(data)
data = process_with_age(data)
data = process_with_fare(data)
data = process_with_embarked(data)
data = process_with_cabin(data)
data = process_with_pclass(data)
data = process_with_family(data)
data = process_with_ticket(data)
```

3. Create Machine Learning Model

Using some algorithm to build first model

```
from sklearn.ensemble import RandomForestClassifier

def create_model(data, survived):
    training_data = data.iloc[:891].copy()
    testing_data = data.iloc[891:].copy()

    parameters = {'bootstrap': False, 'max_depth': 8,
                  'max_features': 'log2', 'min_samples_leaf': 3,
                  ‘min_samples_split': 3, 'n_estimators': 10}
    model = RandomForestClassifier(**parameters)

    model.fit(training_data, survived)
    prediction_results = model.predict(testing_data)

    return prediction_results
```

4. Compare with other algorithms

Using many algorithm to build model

```
from sklearn.neural_network import MLPClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
```

4. Compare with other algorithms

Using many algorithm to build model

```
models = [
        KNeighborsClassifier(3),
        SVC(kernel="linear", C=0.025),
        SVC(gamma=2, C=1),
        DecisionTreeClassifier(max_depth=10),
        RandomForestClassifier(n_estimators=100),
        MLPClassifier(),
        AdaBoostClassifier(),
        GaussianNB(),
        QuadraticDiscriminantAnalysis()
    ]
    for model in models:
        value = cross_val_score(model, training_data, survived, cv=5,
scoring='accuracy')
        print(model.__class__ , " have score ", np.mean(value))
```

4. Compare with other algorithms

Using many algorithm to build model

```
<class 'sklearn.neighbors.classification.KNeighborsClassifier'>  have scor
<class 'sklearn.svm.classes.SVC'>  have score  0.8316596962820242
<class 'sklearn.svm.classes.SVC'>  have score  0.639703991826333
<class 'sklearn.tree.tree.DecisionTreeClassifier'>  have score  0.79018003
<class 'sklearn.ensemble.forest.RandomForestClassifier'>  have score  0.81
<class 'sklearn.neural_network.multilayer_perceptron.MLPClassifier'>  have
<class 'sklearn.ensemble.weight_boosting.AdaBoostClassifier'>  have score
<class 'sklearn.naive_bayes.GaussianNB'>  have score  0.7891324770275202
<class 'sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis'>  hav
```

Tuning parameters of algorithm

Tuning parameters

Called hyperparameters tuning

Example How to tuning parameters for Random Forest algorithm ?

http://scikit-learn.org/stable/modules/grid_search.html

Tuning parameters with GridSearchCV

```
parameter_grid = {
                  'max_depth' : [4, 6, 8],
                  'n_estimators': [50, 10],
                  'max_features': ['sqrt', 'auto', 'log2'],
                  'min_samples_split': [2, 3, 10],
                  'min_samples_leaf': [1, 3, 10],
                  'bootstrap': [True, False],
                  }

forest = RandomForestClassifier()
cross_validation = StratifiedKFold(n_splits=5)

grid_search = GridSearchCV(forest,
                           scoring='accuracy',
                           param_grid=parameter_grid,
                           cv=cross_validation,
                           verbose=1
                          )

grid_search.fit(training_data, survived)
```

http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.GridSearchCV

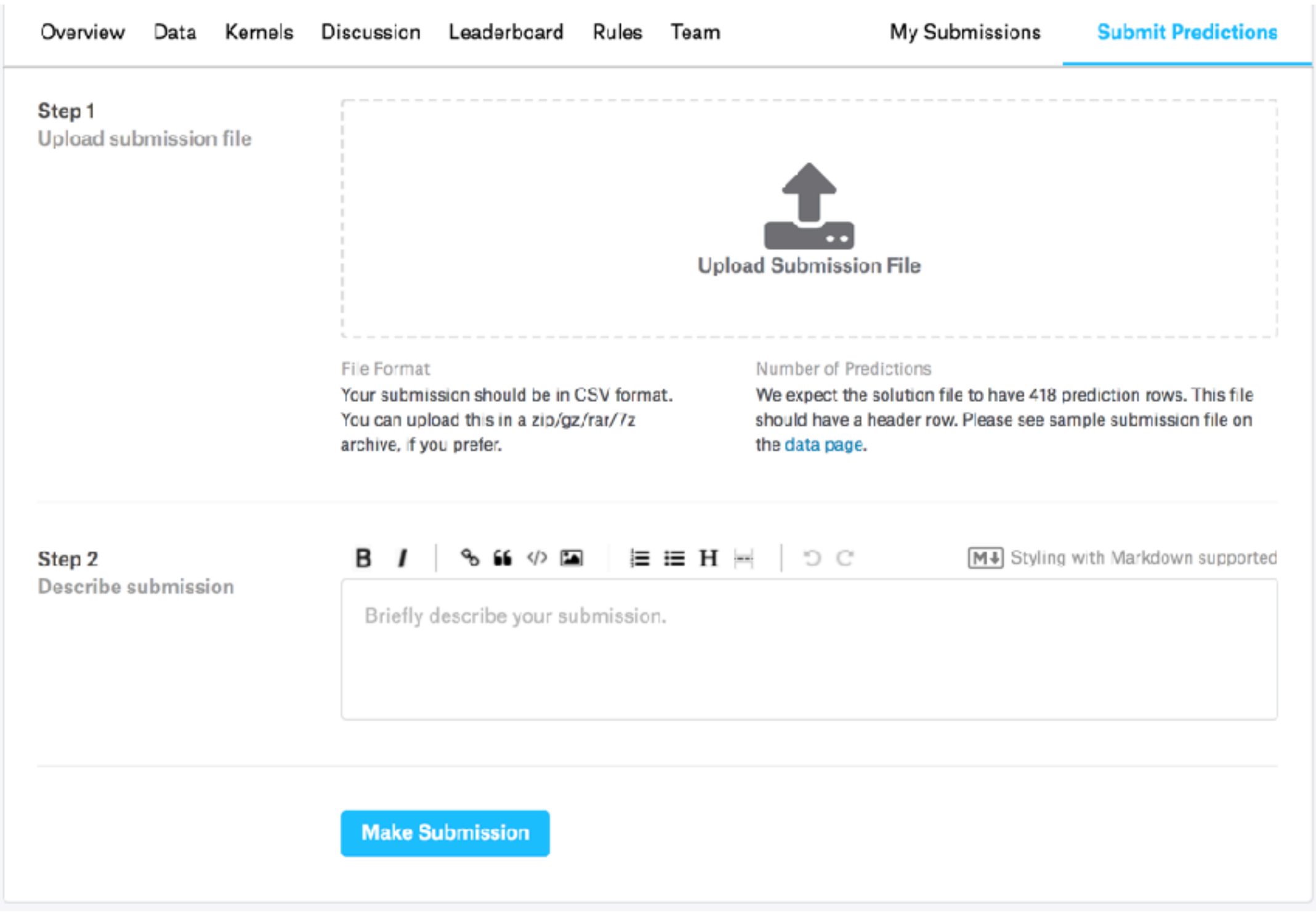
Tuning parameters

Take long time to proceed

```
[Parallel(n_jobs=1)]: Done 1620 out of 1620 | elapsed:  1.9min finished
Fitting 5 folds for each of 324 candidates, totalling 1620 fits
Best score: 0.8383838383838383
Best parameters: {'bootstrap': False, 'max_depth': 8, 'max_features': 'sqrt',
'min_samples_leaf': 3, 'min_samples_split': 3, 'n_estimators': 50}
```

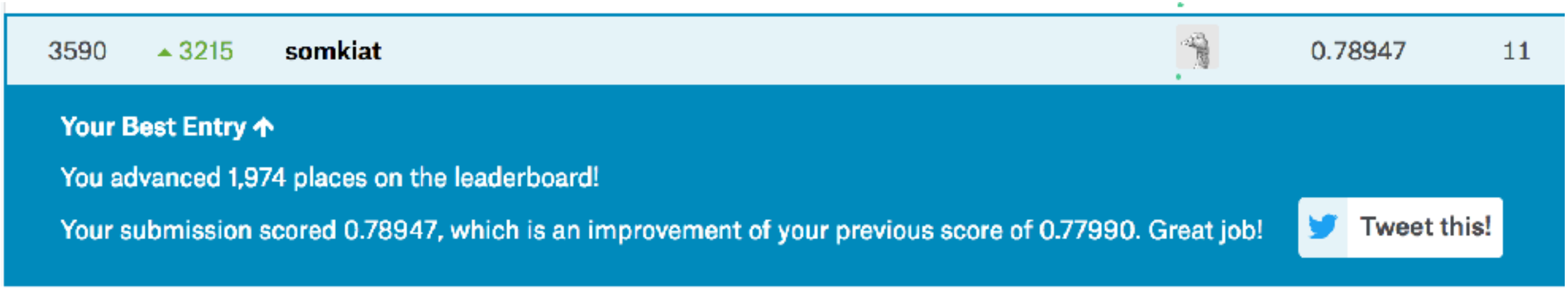
http://scikit-learn.org/stable/modules/grid_search.html

Submit your result to Kaggle



https://www.kaggle.com/c/titanic/submit

Share your ranking



https://www.kaggle.com/c/titanic/leaderboard