Data Science

Hands on machine learning
with Scikit-learn

Yueh-Lin Tsai
2019 / 07

About me

- Yueh-Lin Tsai

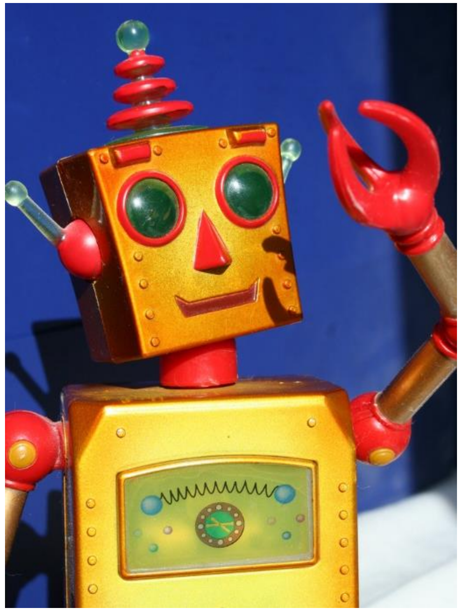
- Education
  - National Cheng Kung University, M.S., Psychology (2013-2015)
  - National Cheng Kung University, B.S., Psychology (2009-2013)

- Present
  - AI Engineer in Taiwan AI Academy

What is machine learning ?

- Extract relations/patterns from data automatically

- Apply those rules to unseen data



$$f(x) = y$$

Machine learning workflow

- Problem definition
- Data exploration / preprocessing
- Build model
- Model evaluation

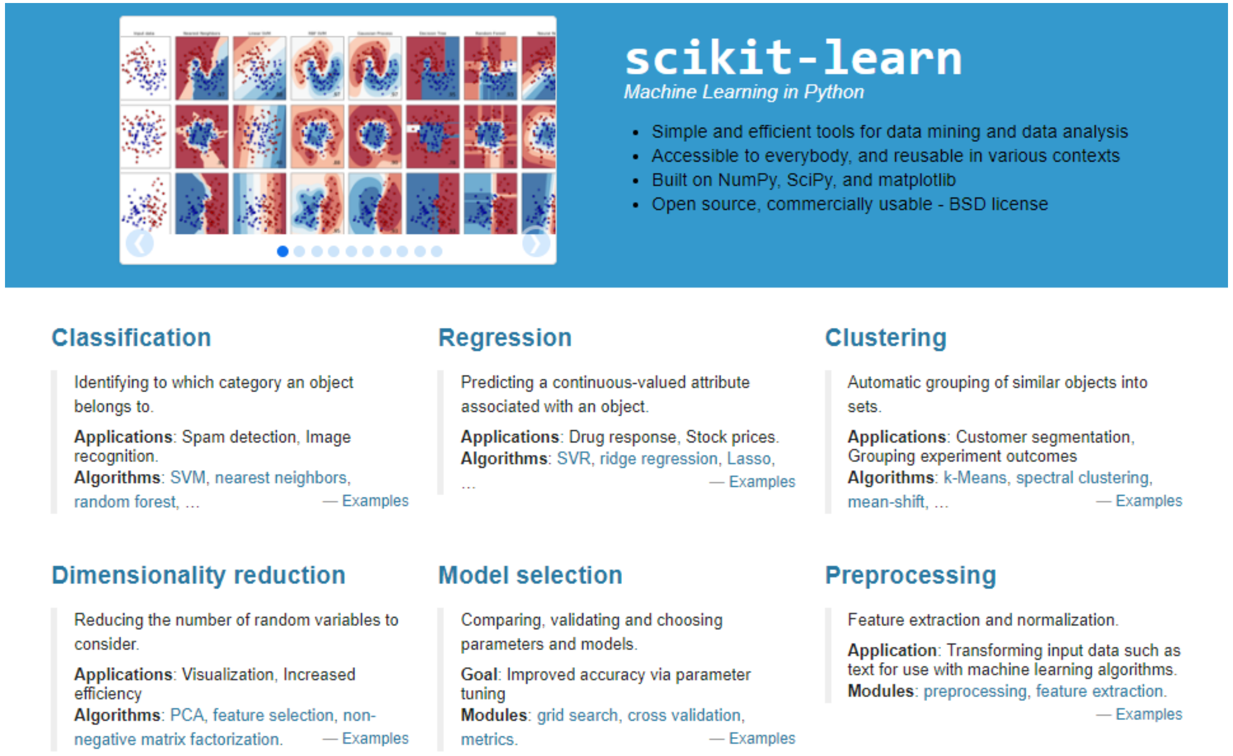
[Cheat sheet – scikit learn]

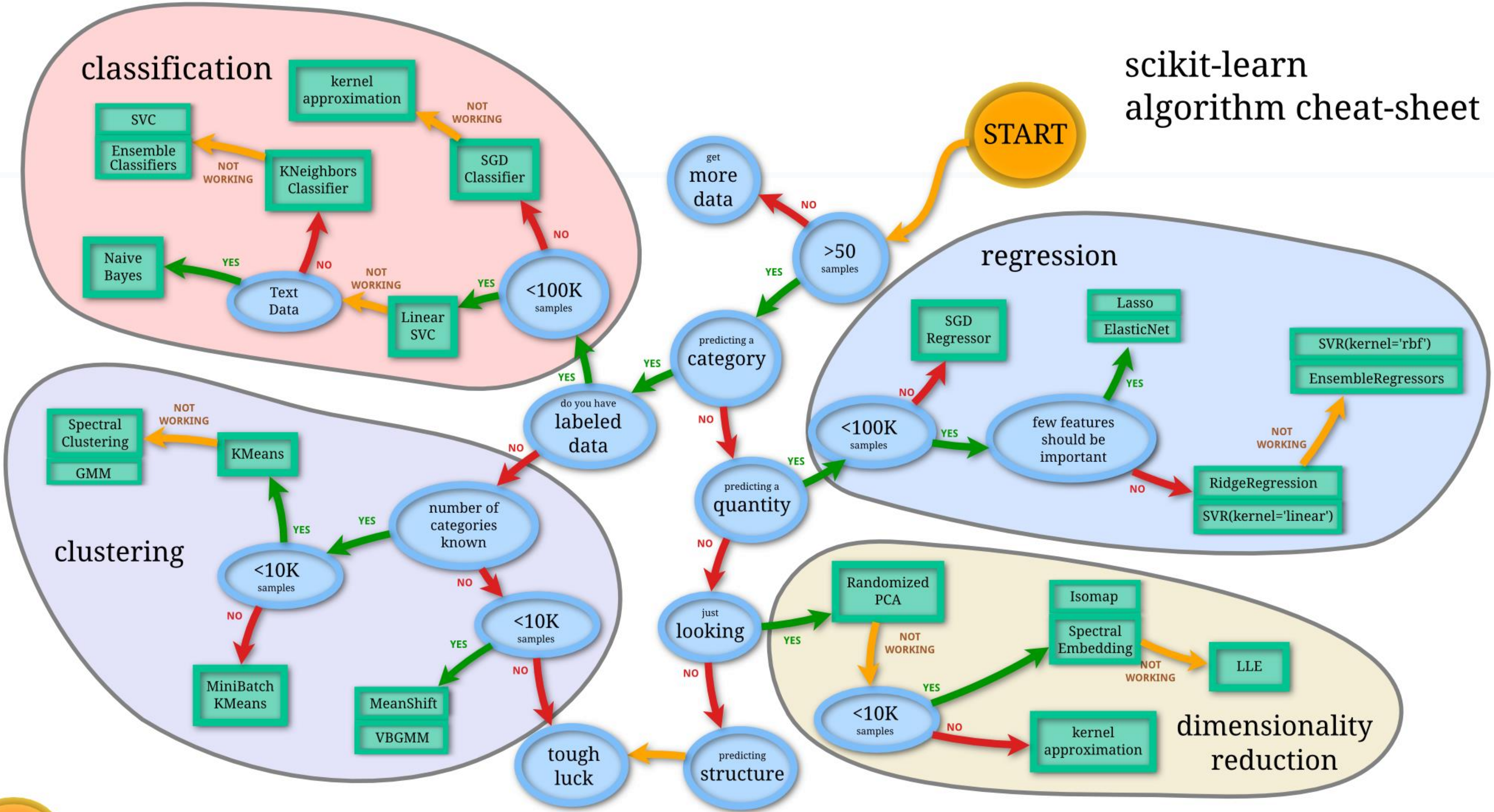
Type of machine learning

- Supervised learning
  - Regression
  - Classification
- Unsupervised learning
  - Cluster
  - Dimension reduction
- Reinforcement learning

Scikit - learn

- Package for machine learning in python
- What can sklearn do
  - Preprocess
  - Ml models
  - Evaluation metrics





scikit-learn
algorithm cheat-sheet
START
>50 samples
get more data
predicting a category
do you have labeled data
<100K samples
Linear SVC
Text Data
Naive Bayes
KNeighbors Classifier
SVC
Ensemble Classifiers
SGD Classifier
kernel approximation
classification
number of categories known
<10K samples
KMeans
Spectral Clustering
GMM
MiniBatch KMeans
MeanShift
VBGMM
clustering
tough luck
predicting a quantity
regression
SGD Regressor
few features should be important
Lasso
ElasticNet
RidgeRegression
SVR(kernel='linear')
SVR(kernel='rbf')
EnsembleRegressors
just looking
predicting structure
Randomized PCA
Isomap
Spectral Embedding
LLE
kernel approximation
dimensionality reduction
YES
NO
NOT WORKING

Back

Source

Build up your ML model in one slide

```python
import pandas as pd
from sklearn import preprocessing, linear_model, model_selection, metrics

data = pd.read_csv('example_data.csv')

data_y = data['target']
data = data.drop('target', axis = 1, inplace = True)

one_hot_data = pd.get_dummies(data)

ss = preprocessing.StandardScaler()
scale_data = ss.fit_transform(data)

train_x, test_x, train_y, test_y = model_selection.train_test_split(data, data_y, test_size = 0.2, random_state = 99)

model = linear_model.LinearRegression() # LogisticRegression()
model.fit(train_x, train_y)

test_prediction = model.predict(test_x)
print('r-square of linear regression : {:.3f}'.format(metrics.r2_score(test_prediction, test_y)))
```

Supervised learning

Machine learning workflow

- Problem definition
- Data exploration / preprocessing
- Build model
- Model evaluation

Cheat sheet – scikit learn

Data preprocessing

- Handle missing data
  - Delete data which have missing values (row or column)
  - Missing imputation
- Handle outliers
  - Distribution transformation
  - Replace outliers

Data preprocessing

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

- Convert categorical data to numerical data
  - Label encoding
  - One-hot encoding

Name	Score
Amy	78
Bob	90
Chris	65
Amy	86
Chris	67

Name_label	Score
1	78
2	90
3	65
1	86
3	67

Label encoding

Amy_oh	Bob_oh	Chris_oh	Score
1	0	0	78
0	1	0	90
0	0	1	65
1	0	0	86
0	0	1	67

One-hot encoding

Data preprocessing

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
```

- Normalize data
  - Standard scale
  - Min-max scale

$$x_{standard} = \frac{x - \mu}{\sigma}$$

$$x_{minmax} = \frac{x - Min(x)}{Max(x) - Min(x)}$$

Name	Score
Amy	78
Bob	90
Chris	65
Amy	86
Chris	67

Name	Score
Amy	0.0719
Bob	1.1509
Chris	-1.0969
Amy	0.7912
Chris	-0.9171

Standard scale

Name	Score
Amy	0.48
Bob	0
Chris	1
Amy	0.16
Chris	0.92

Min-max scale

Data preprocessing

```
from sklearn.preprocess import train_test_split
```

- Data splitting
  - Training set
  - Validation set
  - Testing set

- Cross validation

Model selection

```
from sklearn.linear_model import LinearRegression, LogisticRegression
```

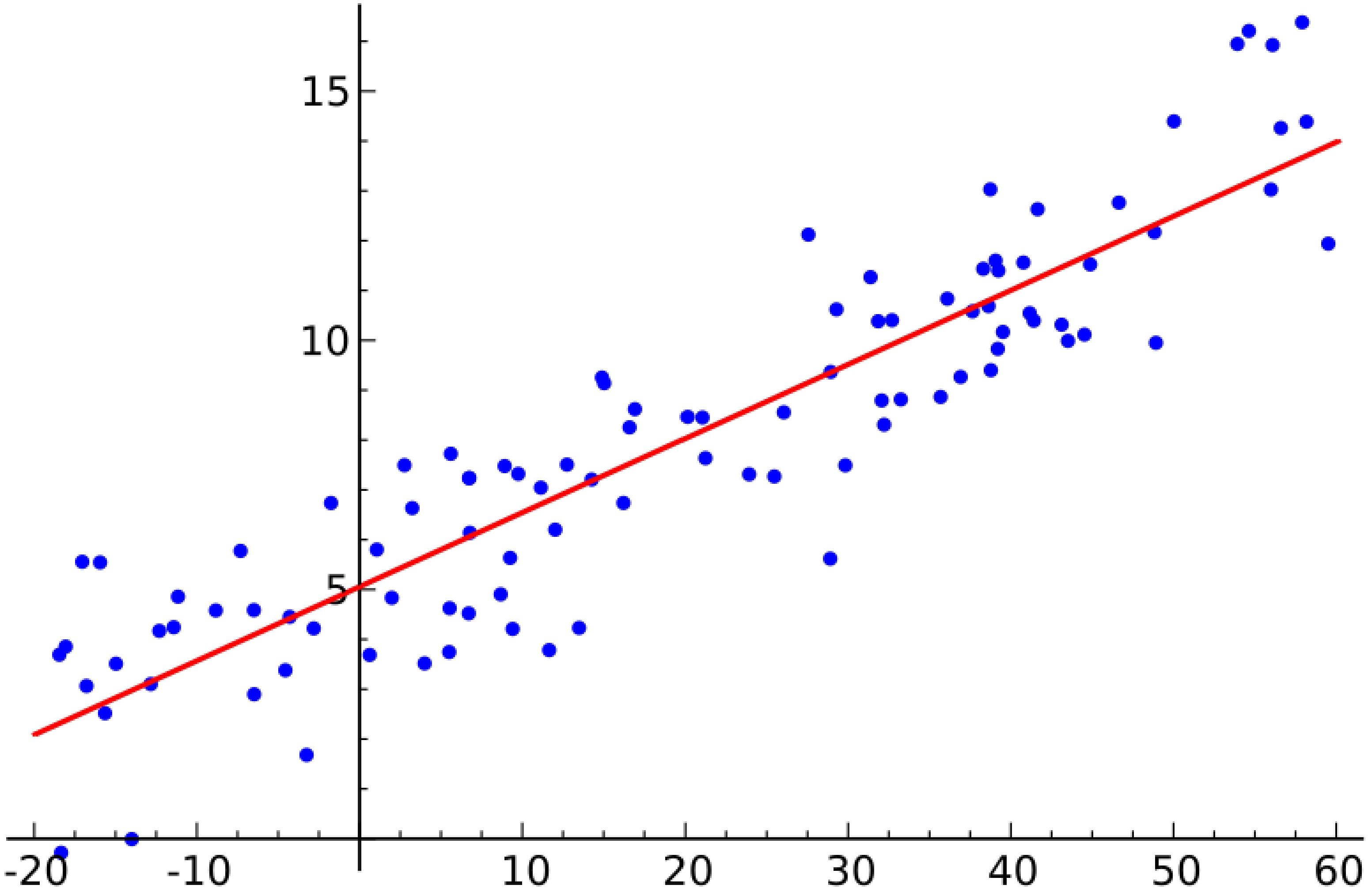
- Linear model
    - Linear Regression
    - Logistic Regression
    - Ridge regression
    - Lasso regression

$$y = f(a_0 + a_1 x_1 + a_2 x_2 + \ldots)$$

The aim of linear model is to find a line which minimize the distance (error) between data and the line

Linear model

- Linear regression

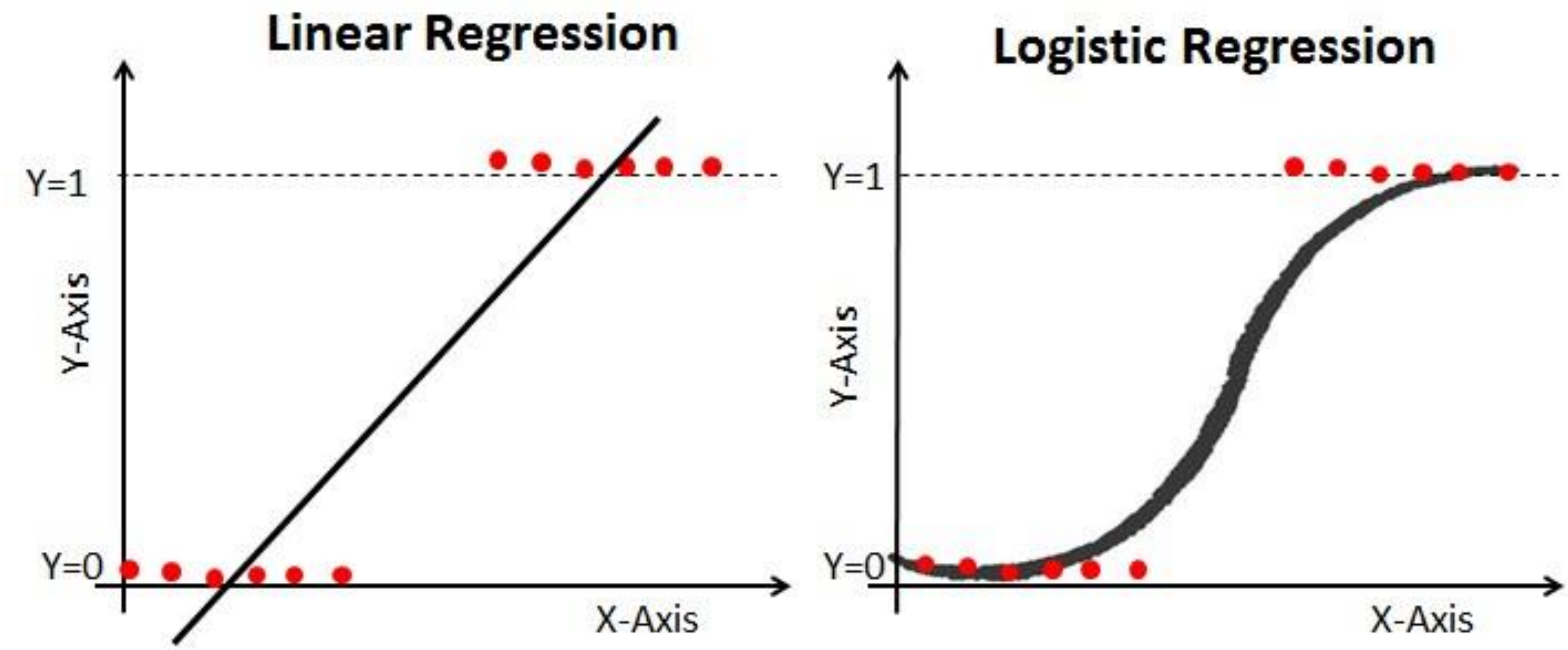


Source

Linear model

$$f(x) = \frac{1}{1 + e^{-(x)}}$$

- Logistic regression

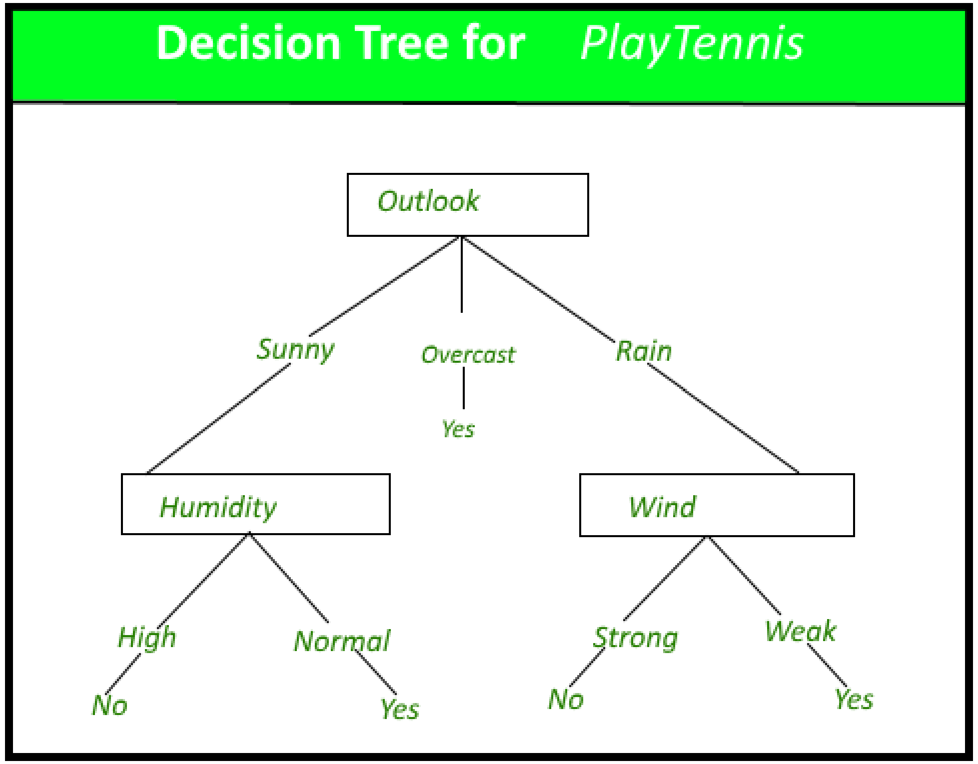


Source

Model selection

```
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
```

- Tree-based model
  - Decision Tree
  - Random Forest



The aim of tree-based model is to find good cut-off points to split data repetitively.

Source

Model selection

- Model comparison
  - Linear model
    - Focus on global information
    - Have data hypothesis

    Encoding matters

    Normalization is needed
  - Tree-based model
    - Clear rules provided by model
    - Focus on local information

    No need to normalize data

Model selection

- Other machine learning models
  - Support Vector Machine
  - K-Nearest Neighbor
  - Naïve-bayes
  - Neural network

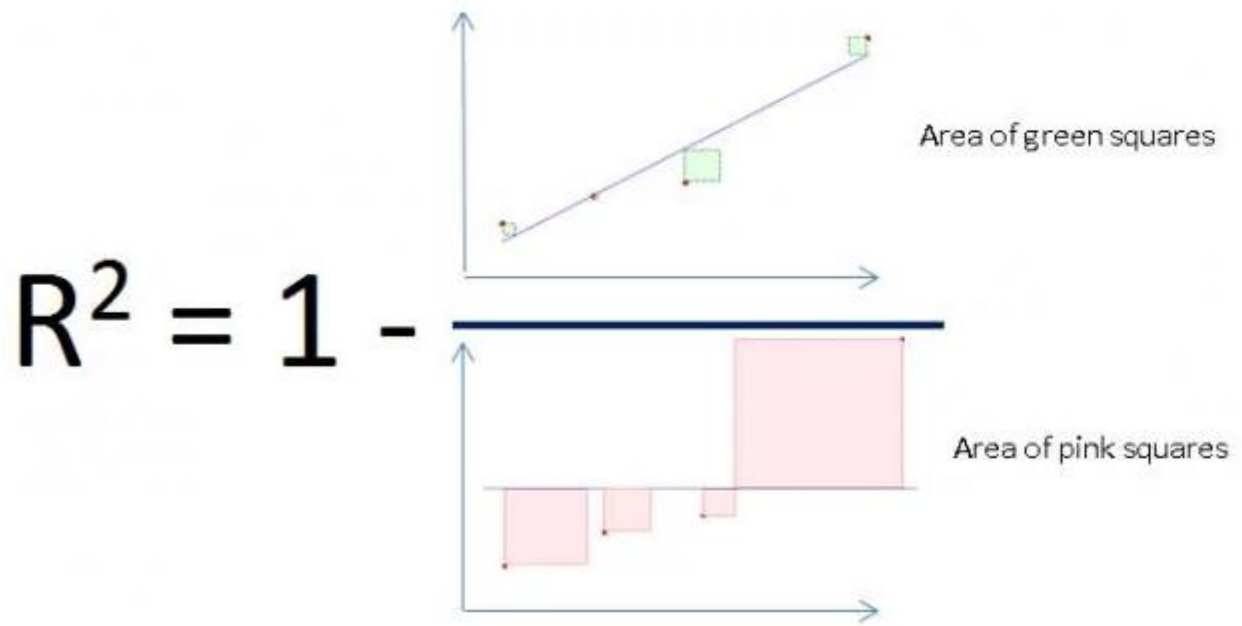
Model evaluation

```
from sklearn.metrics import r2_score, mean_squared_error,
mean_absolute_error
```

- Regression problem
  - R - square, R^2
  - Mean Squared Error, MSE
  - Mean Absolute Error, MAE

Model evaluation

- R-square
  - Range from 0 to 1
  - More is better



Source

Model evaluation

- Mean Square Error

$$MSE = \frac{1}{n}\sum \underbrace{\left(y - \hat{y}\right)^2}_{\text{The square of the difference between actual and predicted}}$$

- Mean Absolute Error
    - Less is better

$$MAE = \overset{\text{Divide by the total number of data points}}{\boxed{\frac{1}{n}}} \underset{\text{Sum of}}{\sum} \underbrace{\left| \overset{\text{Actual output value}}{\boxed{y}} - \overset{\text{Predicted output value}}{\boxed{\hat{y}}} \right|}_{\text{The absolute value of the residual}}$$

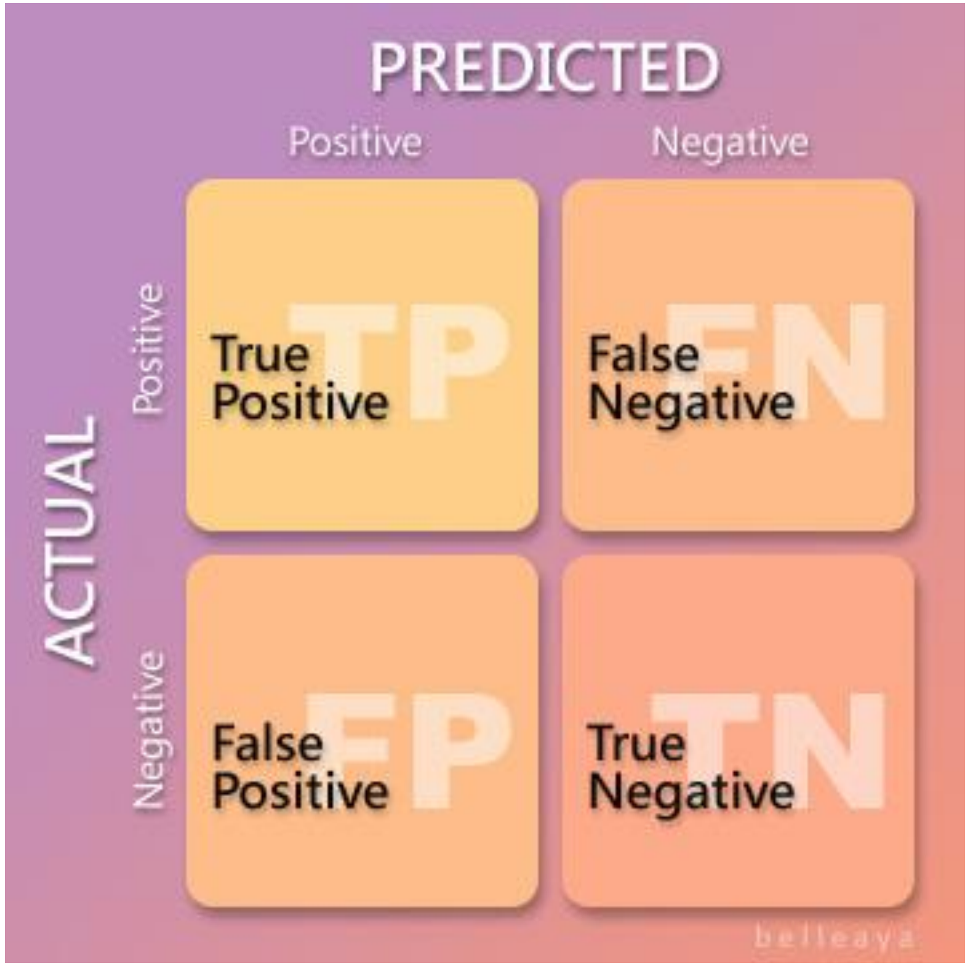
Source

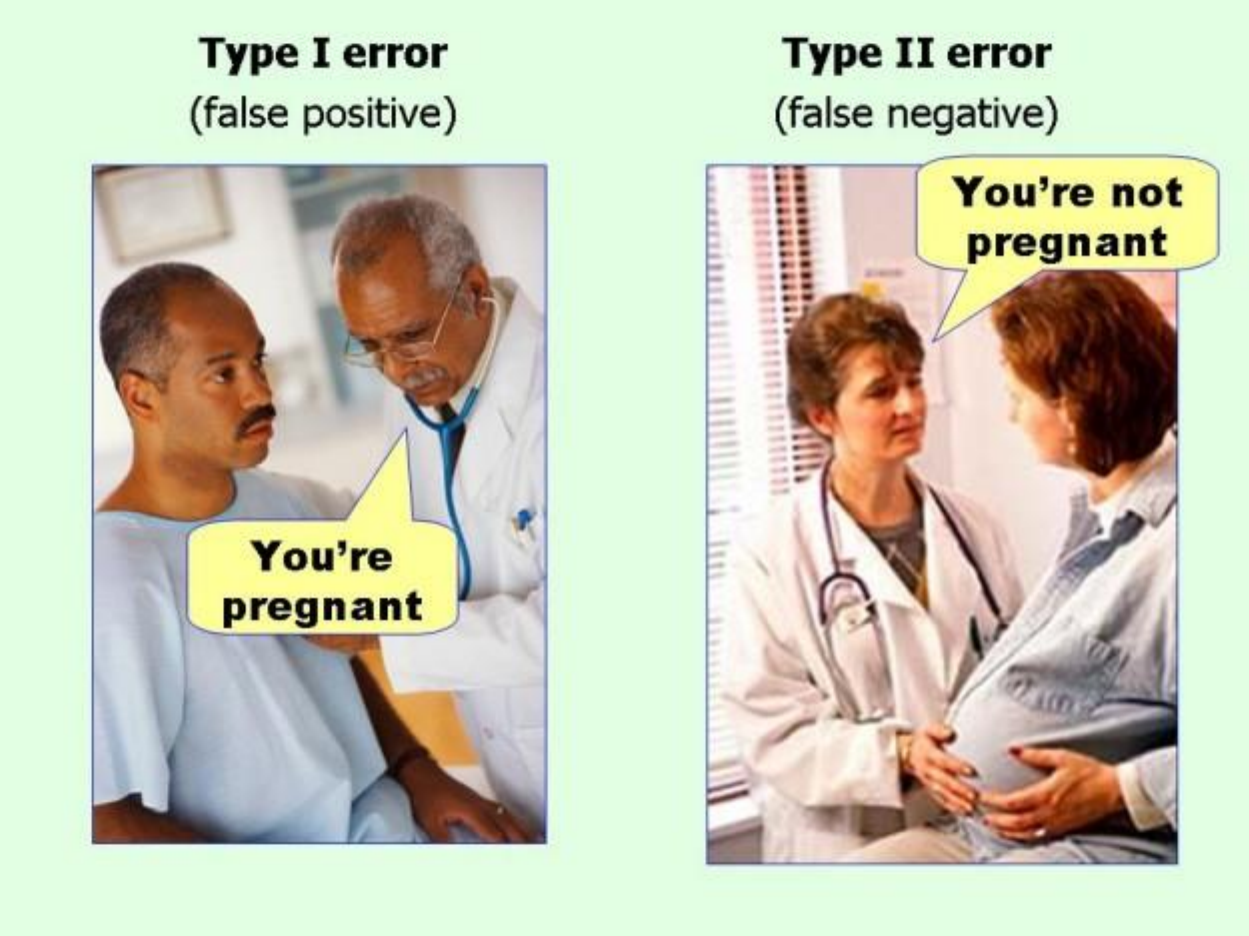
Model evaluation

```
from sklearn.metrics import confusion_matrix, accuracy_score,
precision_score, recall_score
```

- Classification problem
  - Confusion matrix
  - Accuracy
  - Precision, recall
  - Area under curve (AUC), f1 score

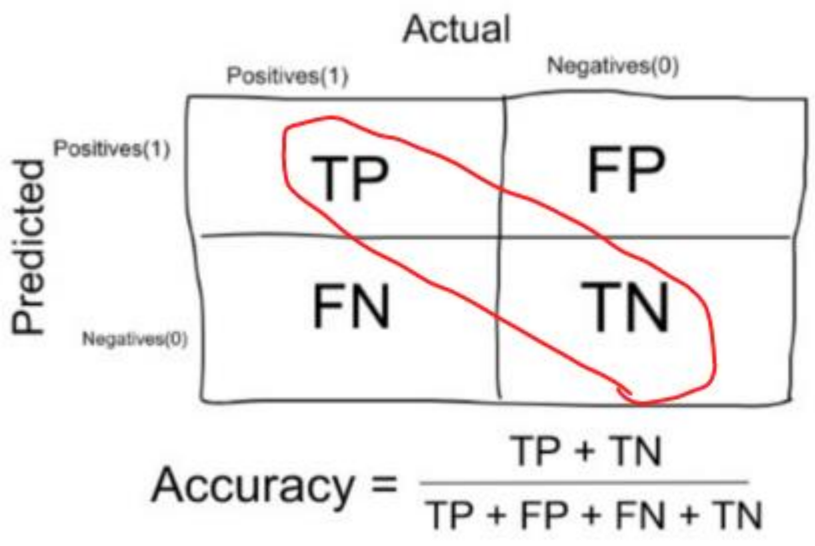
Model evaluation

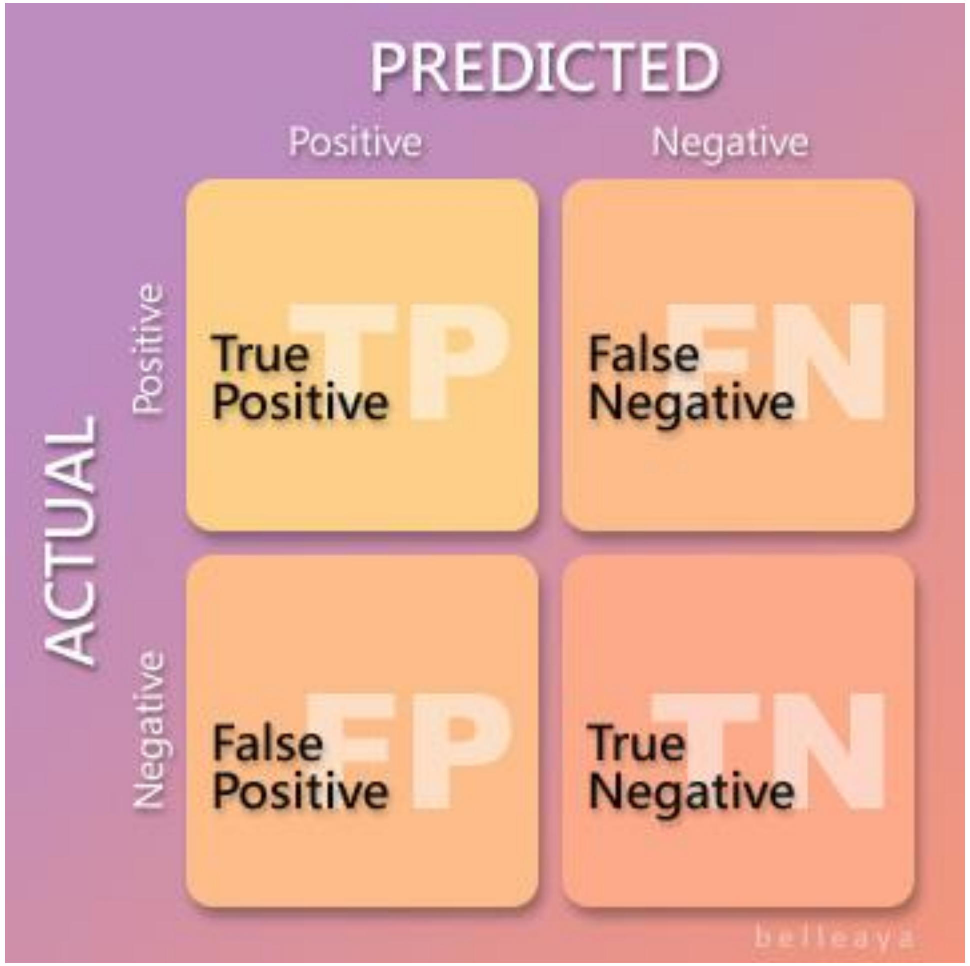




Source

Model evaluation





Source

Source

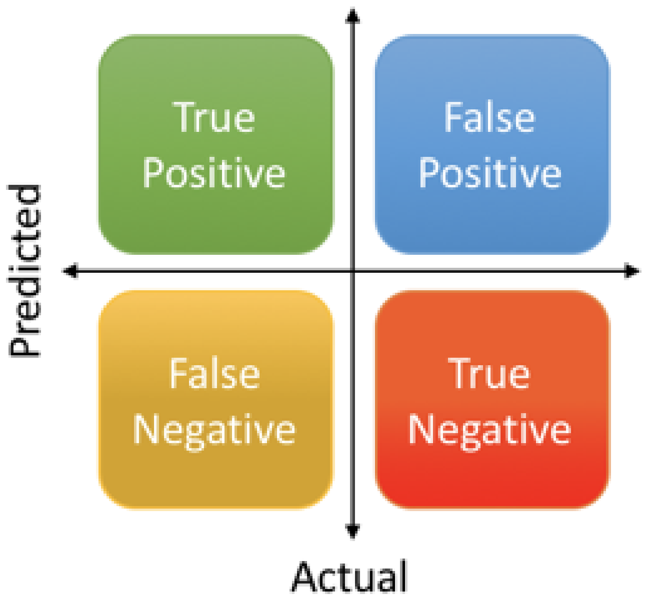
Model evaluation

- Precision & recall

$$\textbf{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \text{ or } \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

$$\textbf{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \text{ or } \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$

$$\textbf{Accuracy} = \frac{\text{True Positive + True Negative}}{\text{Total}}$$



Other topics – supervised learning

- Feature engineering and feature selection
- Overfitting and generalizations
- Parameter selection

Unsupervised learning

Machine learning workflow

- Problem definition
- Data exploration / preprocessing
- Build model
- Model evaluation

Cheat sheet – scikit learn

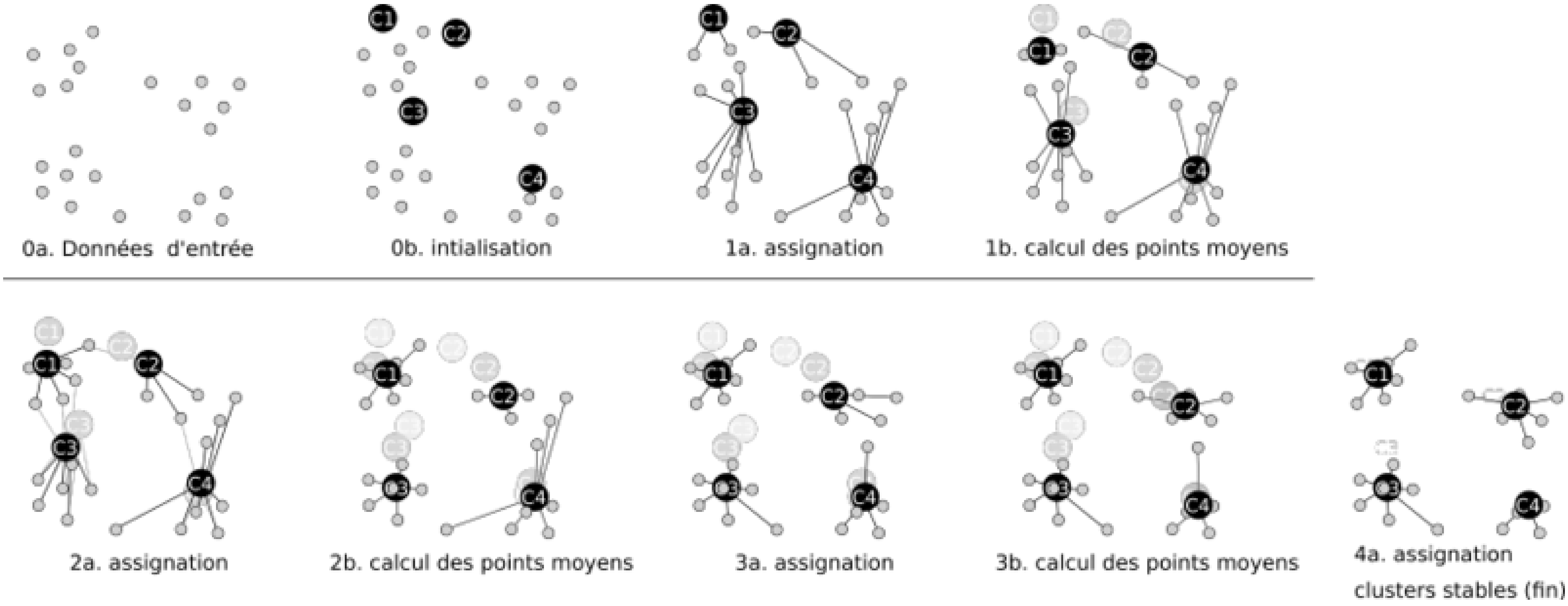
Problem definition and model selection

- clustering
  - K-means clustering
  - Hierarchical clustering
  - DBSCAN

The aim of clustering is to categorize data base on their similarity

K-means

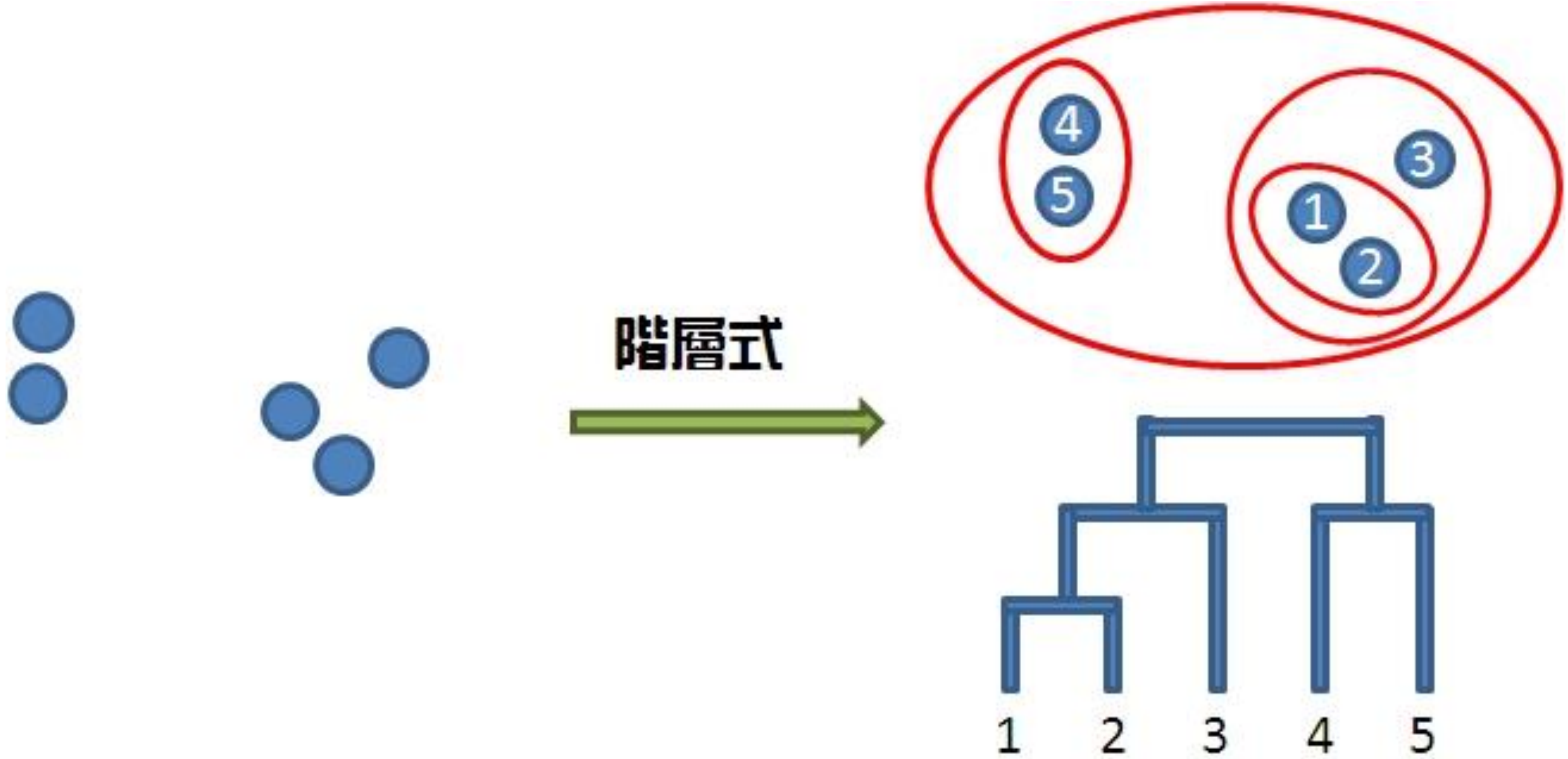
```
from sklearn.cluster import KMeans
```



Source

Hierarchical clustering

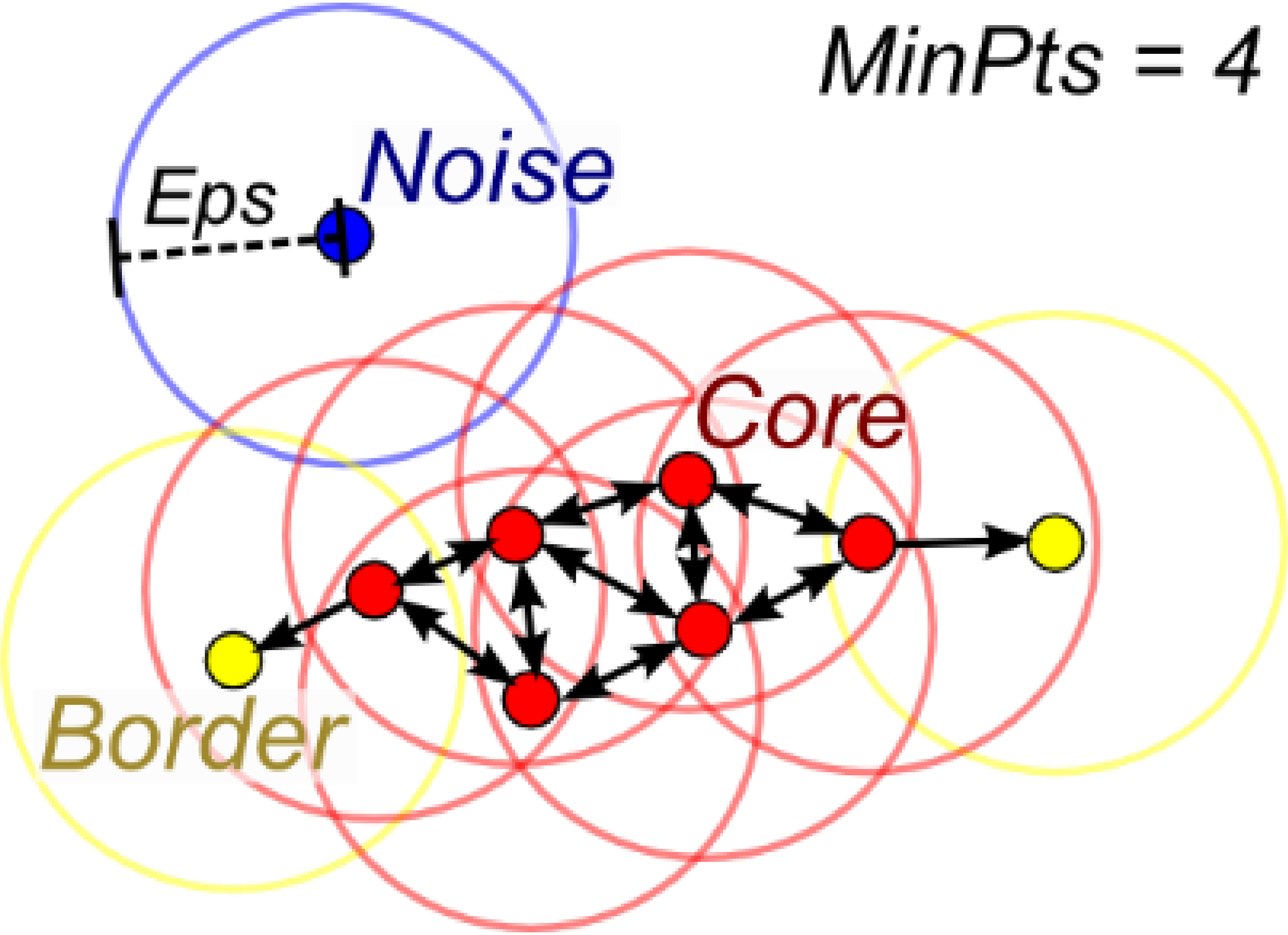
```
from sklearn.cluster import AgglomerativeClustering
```



Source

DBSCAN

```
from sklearn.cluster import DBSCAN
```



Source

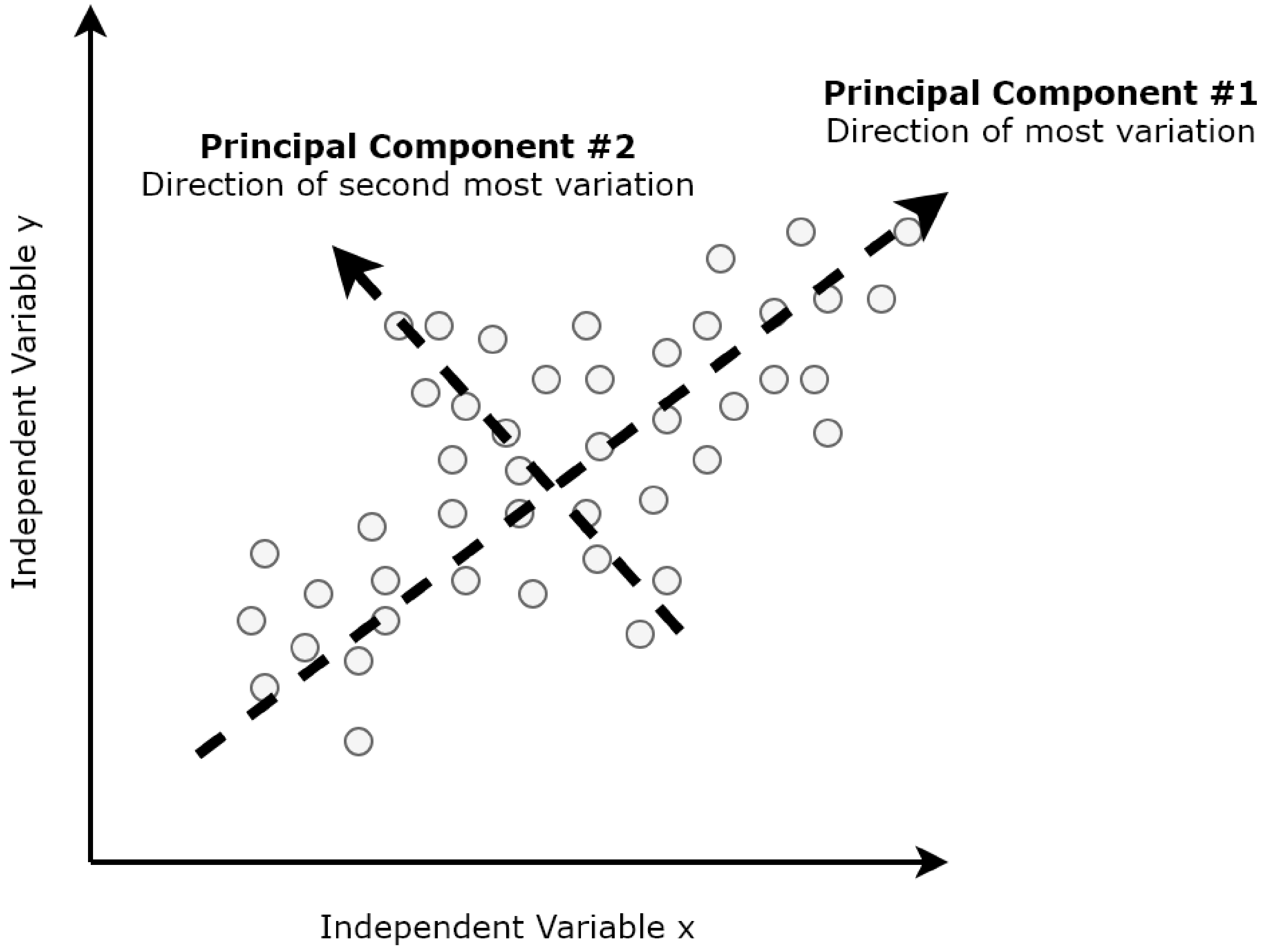
Problem definition and model selection

- dimension reduction
  - Principal component analysis, PCA

The aim of dimension reduction is to describe high-dimension data with fewer dimension

```
from sklearn.decomposition import PCA
```

PCA



Principal Component #1
Direction of most variation
Principal Component #2
Direction of second most variation
Independent Variable y
Independent Variable x

Source

Model evaluation

- Cluster
  - Homogeneity scores
- Dimension reduction
  - Information loss ratio

Supplementary

Feature engineering / feature selection

- Feature generation
  - From domain knowledge
  - By exploring data
- Feature selection
  - Based on correlation
  - Based on coefficients in lasso / ridge regression model
  - Based on feature importance index in tree-based models

Further reading : feature selection with sklearn

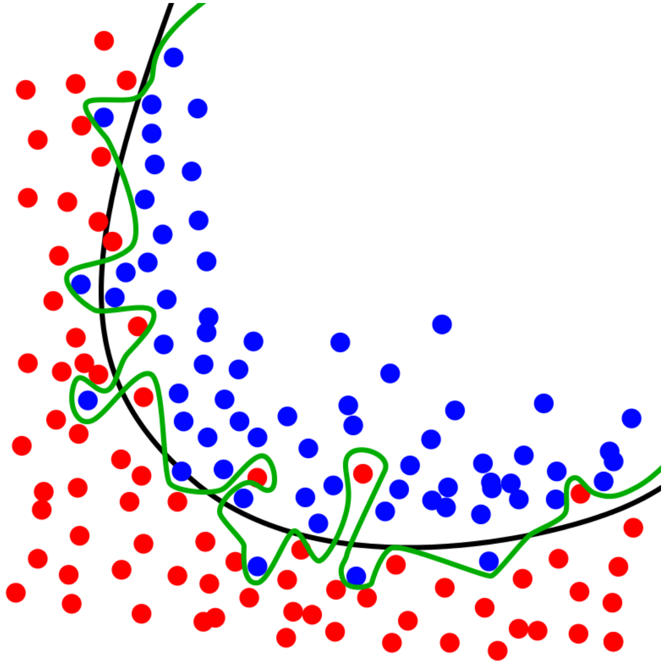
Other ML models

- Ml models that usually showed on ml competitions
  - Bagging ： Random Forest
  - Boosting ： XGBoost, LightGBM, CatBoost
  - Neural Network
  - Stacking model

Overfitting and generalization

We can always get 100% accuracy on training set with a powerful model, but …

- How to detect overfitting ?
- How to resolve / avoid overfitting ?
    - Decrease the power of present model
    - Ensemble methods



Parameter selection

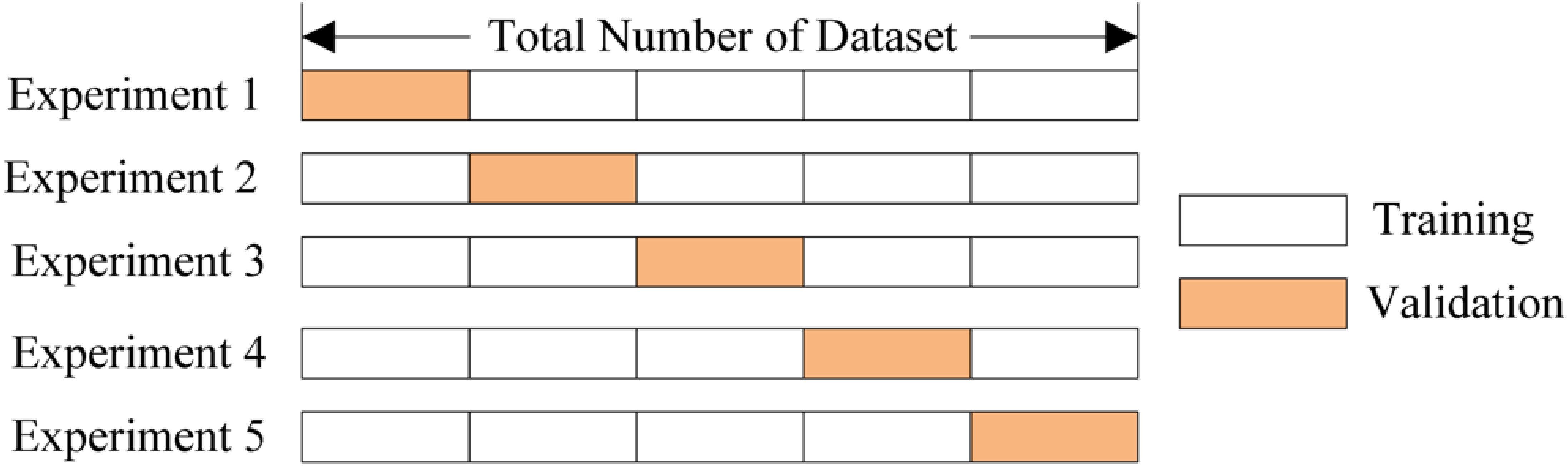
```
from sklearn.model_selection import GridSearchCV
```

- Select a proper set of parameters would affect the power of model hence make a better prediction

- How to decide which parameter set should be used ?

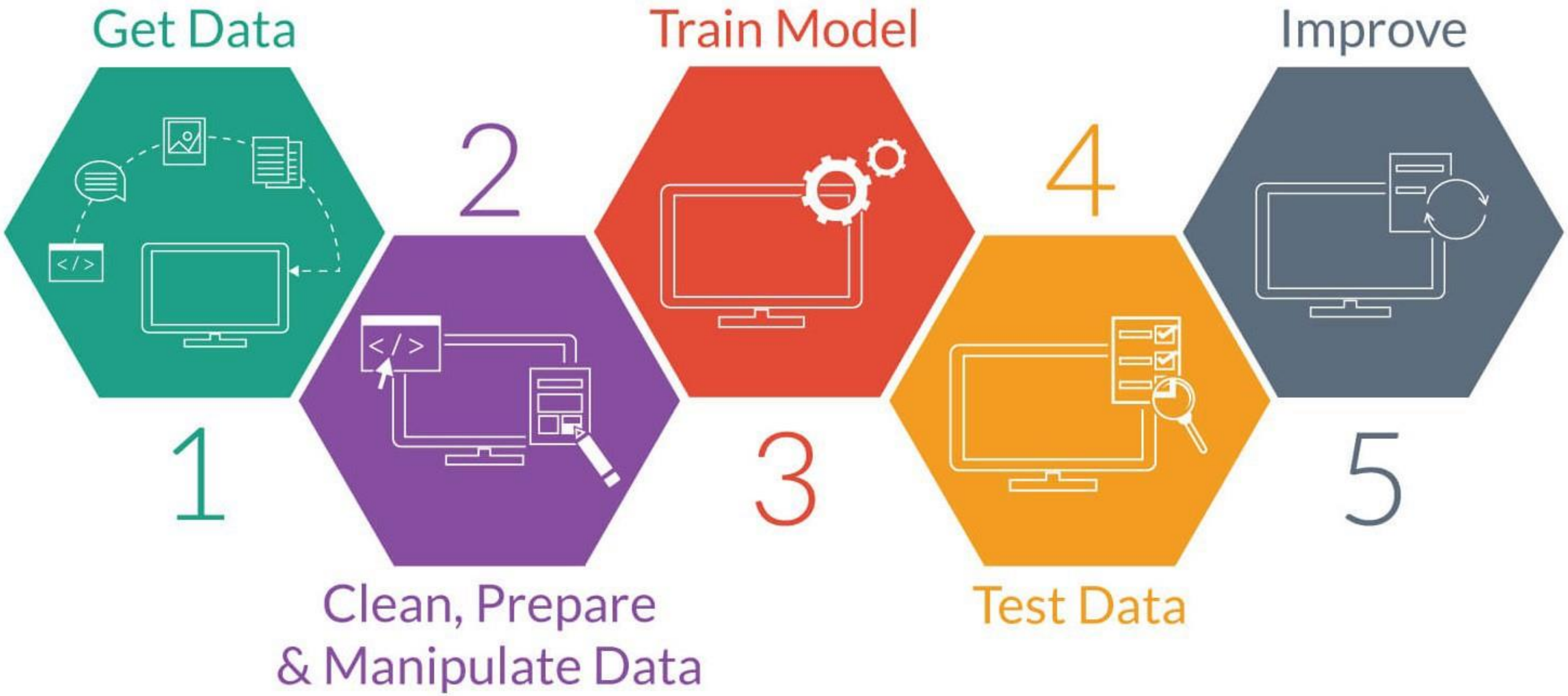
Parameter selection

Cross Validation



Source

Summary of Machine learning



Source

How to build a great model ?

- Before model fitting
  - Preprocessing
  - Data exploration
  - Data engineering
- After preliminary model
  - Parameter tuning
  - Cross validation