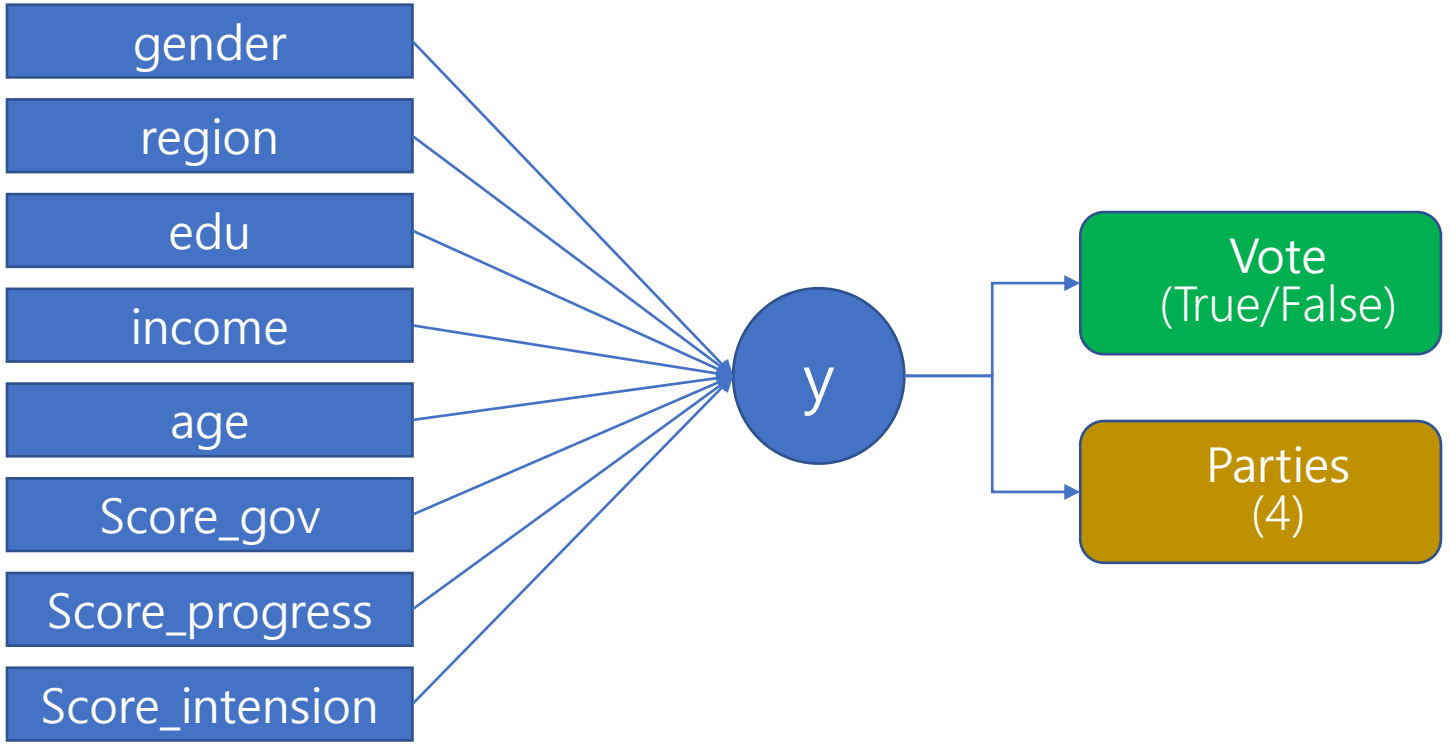
Sample Exercise for Machine Learning with Python & Scikitlearn

01 Data Description

✓ vote.csv

The effect of voters' demographic variables and political attitudes on whether or not to participate in the presidential election and supporters.



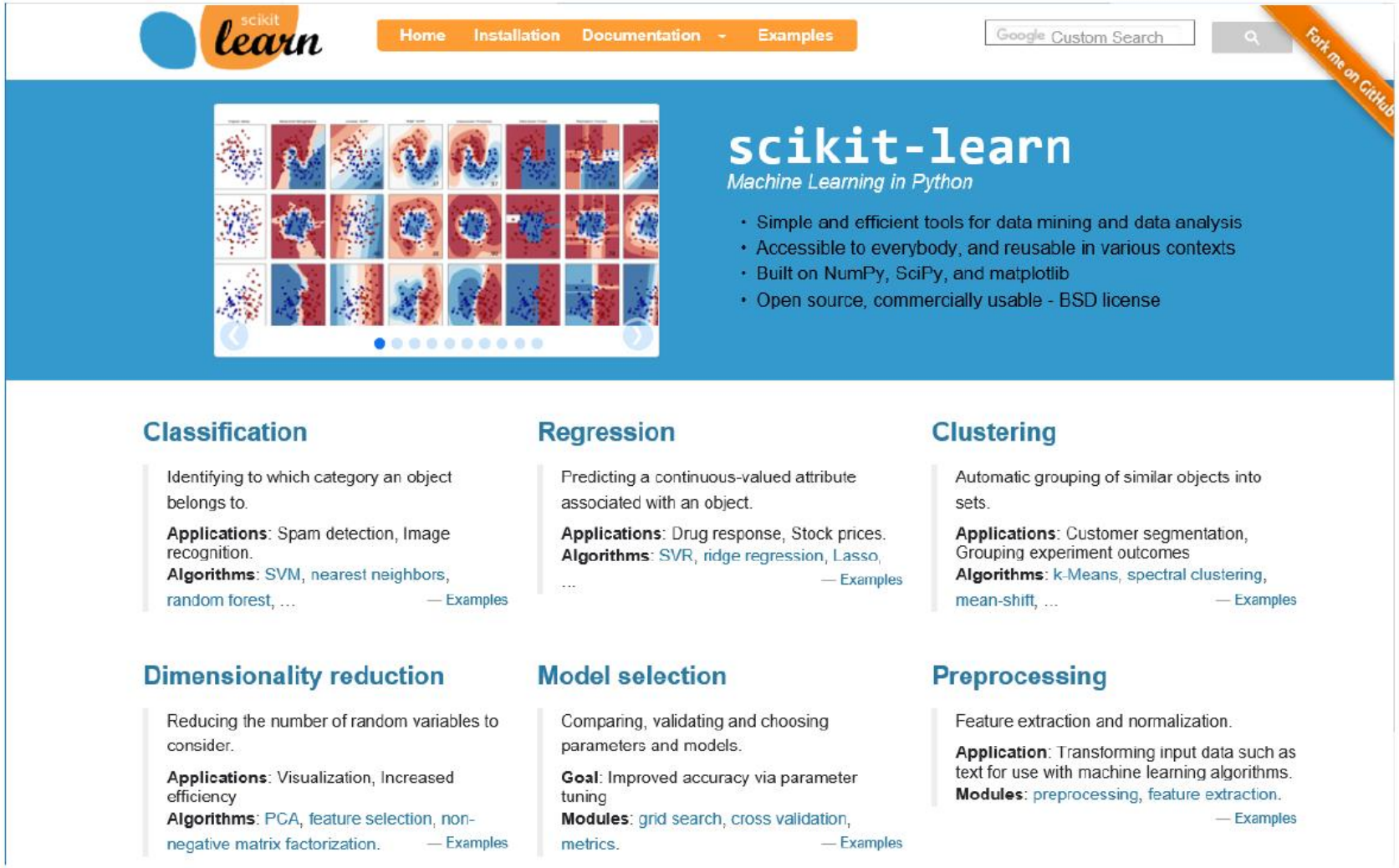
01 Data Description

✓ vote.csv (211 people)

1	gender	region	edu	income	age	score_gov	score_progress	score_intention	vote	parties
2	1	4	3	3	3	2	2	4	1	2
3	1	5	2	3	3	2	4	3	0	3
4	1	3	1	2	4	1	3	2.8	1	4
5	2	1	2	1	3	5	4	2.6	1	1
6	1	1	1	2	4	4	3	2.4	1	1
7	1	1	1	2	4	1	4	3.8	1	2
8	1	1	1	2	4	4	4	2	1	1
9	1	5	2	4	4	3	4	3.6	1	3
10	1	2	1	2	4	2	2	2	0	2
11	1	1	1	2	3	4	2	3	1	1
12	1	1	1	2	3	2	4	2.2	0	2
13	2	4	1	1	3	3	2	2.6	1	1
14	1	5	1	2	4	3	2	3	1	1
15	1	2	2	4	4	3	3	2.4	1	3
16	1	4	3	4	3	3	4	3.6	1	3
17	1	1	2	3	3	3	3	3.2	1	4
18	1	5	2	4	3	4	3	4	1	4

02 Machine learning library : Scikit-learn

*http://scikit-learn.org/stable/



02 Machine learning library : Scikit-learn

✓ KNN Library

sklearn.neighbors.KNeighborsClassifier

class sklearn.neighbors. **KNeighborsClassifier** (*n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None, **kwargs*) [source]

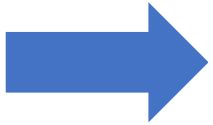
→ Parameter : n_neighbors K

Data Preprocessing

03 Data Preprocessing : One-hot encoding

✓ One-hot encoding, dummy variable

Sex	Region
1	1
2	2
1	3
2	1
1	3
2	2



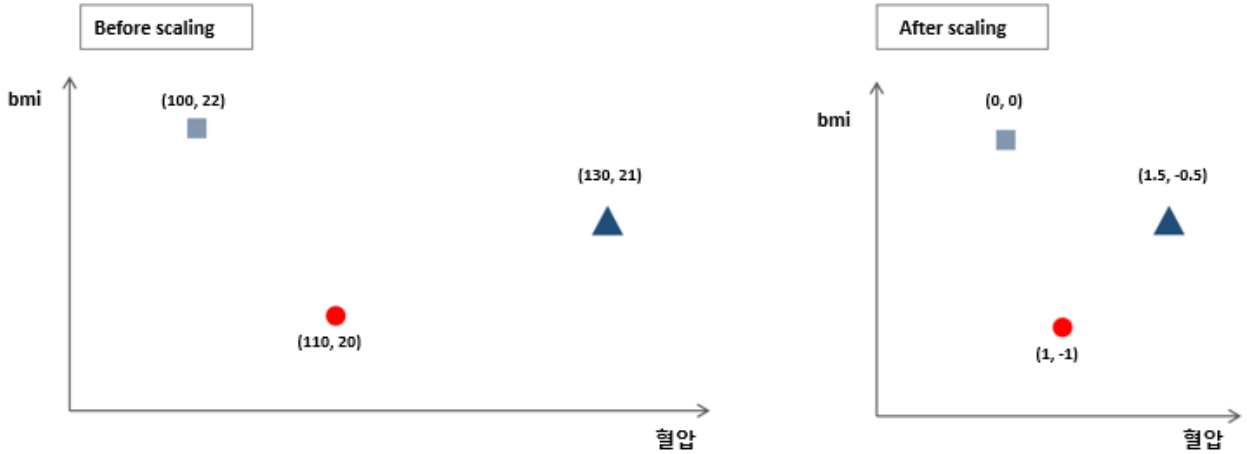
Sex	Region_1	Region_2	Region_3
0	1	0	0
1	0	1	0
0	0	0	1
1	1	0	0
0	0	0	1
1	0	1	0

03 Data Preprocessing : Scaling

✓ Min-Max Scaling

→ scales in the given range on the training set
e.g. between zero and one.

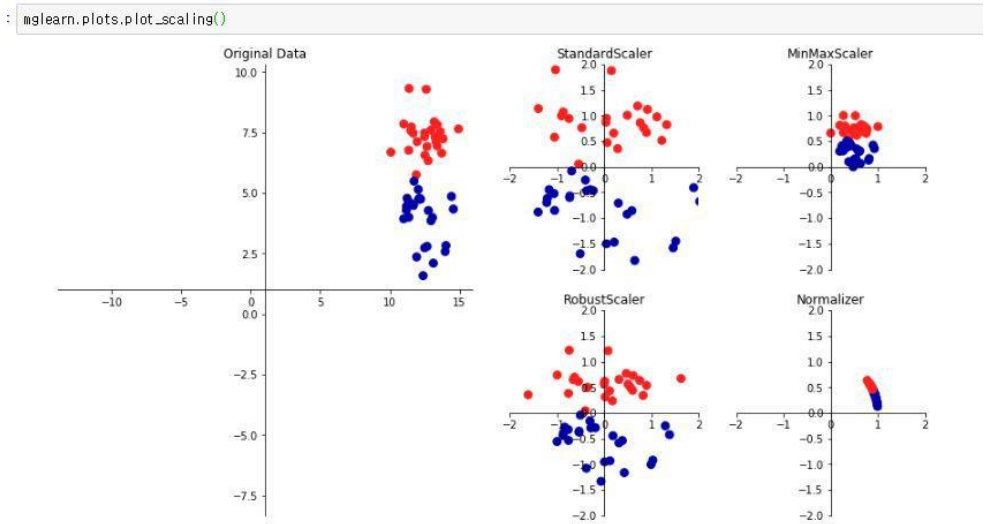
$$X_{new} = \frac{X_i - \min(X)}{\max(x) - \min(X)}$$



✓ Standardization

→ Standardize features by removing
the mean and scaling to unit variance

$$X_{new} = \frac{X_i - X_{mean}}{\text{Standard Deviation}}$$



03 Data Preprocessing : Scaling

✓ **After scaling**

gender	region	edu	income	age	score_gov	score_progress	score_intention	vote	parties
1	4	3	3	3	2	2	4	1	2
1	5	2	3	3	2	4	3	0	3
1	3	1	2	4	1	3	2.8	1	4
2	1	2	1	3	5	4	2.6	1	1
1	1	1	2	4	4	3	2.4	1	1
1	1	1	2	4	1	4	3.8	1	2
1	1	1	2	4	4	4	2	1	1

gender_female	gender_male	region_Chungcheung	region_Honam	region_Others	region_Sudo	region_Youngnam	edu	income	age	score_gov	score_progress	score_intention	vote	parties
0	1	0	0	0	0	1	1	0.666667	0.666667	0.25	0.25	0.75	1	2
0	1	0	0	1	0	0	0.5	0.666667	0.666667	0.25	0.75	0.5	0	3
0	1	0	1	0	0	0	0	0.333333	1	0	0.5	0.45	1	4
1	0	0	0	0	1	0	0.5	0	0.666667	1	0.75	0.4	1	1
0	1	0	0	0	1	0	0	0.333333	1	0.75	0.5	0.35	1	1
0	1	0	0	0	1	0	0	0.333333	1	0	0.75	0.7	1	2
0	1	0	0	0	1	0	0	0.333333	1	0.75	0.75	0.25	1	1
0	1	0	0	1	0	0	0.5	1	1	0.5	0.75	0.65	1	3

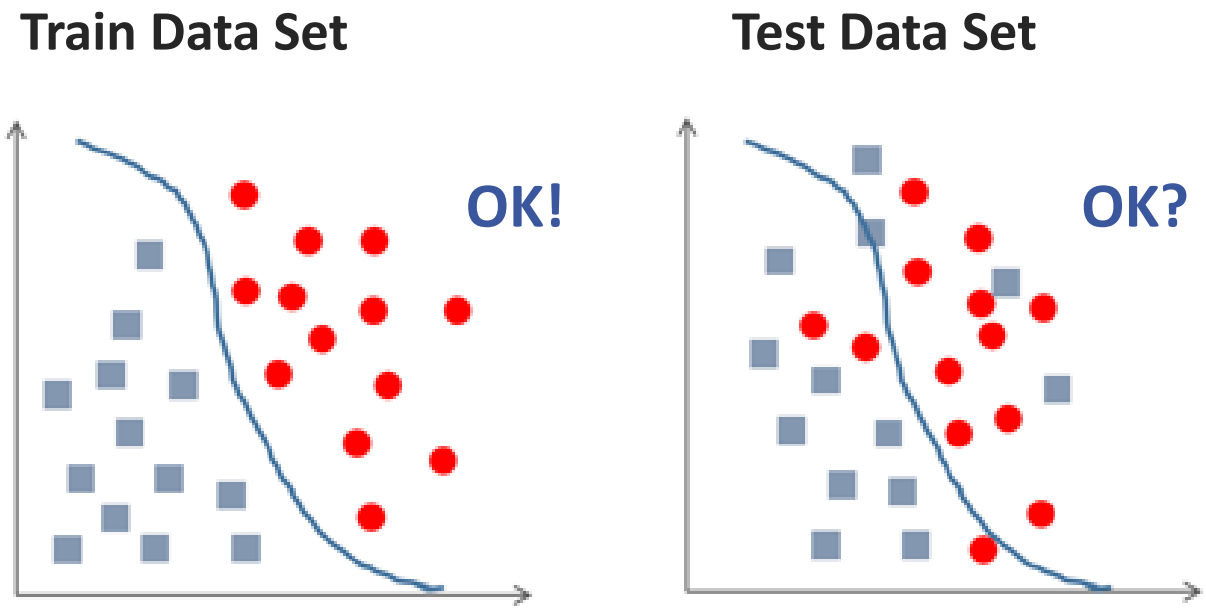
Exercise

Train/Test Data Set Split

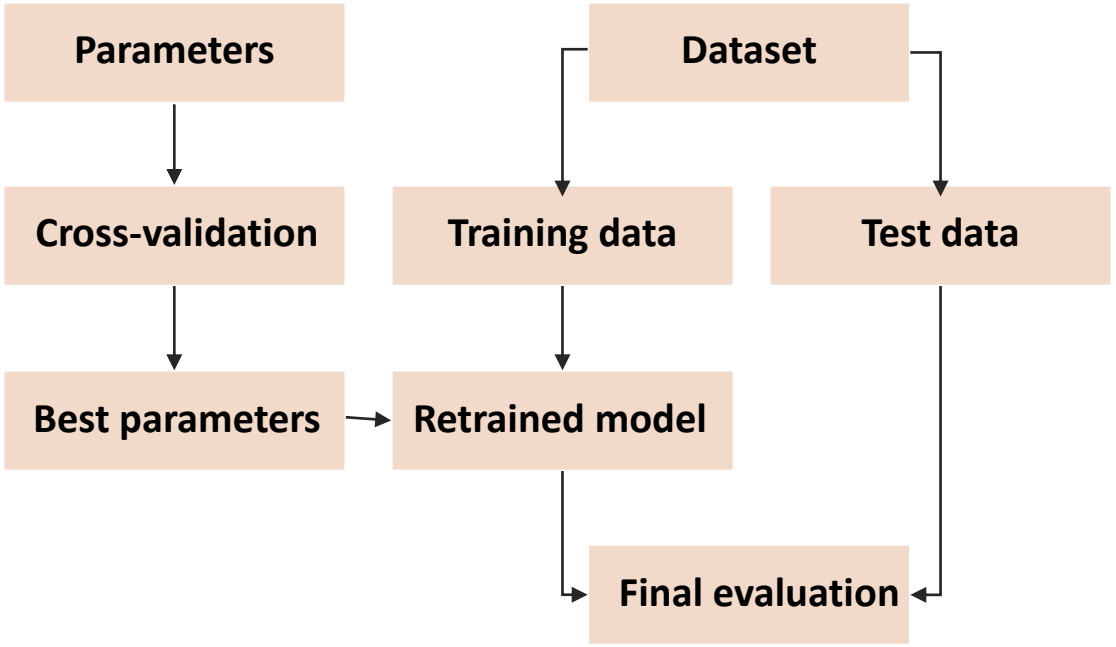
Train/Test Data Set Split

Train/Test Data Set Split

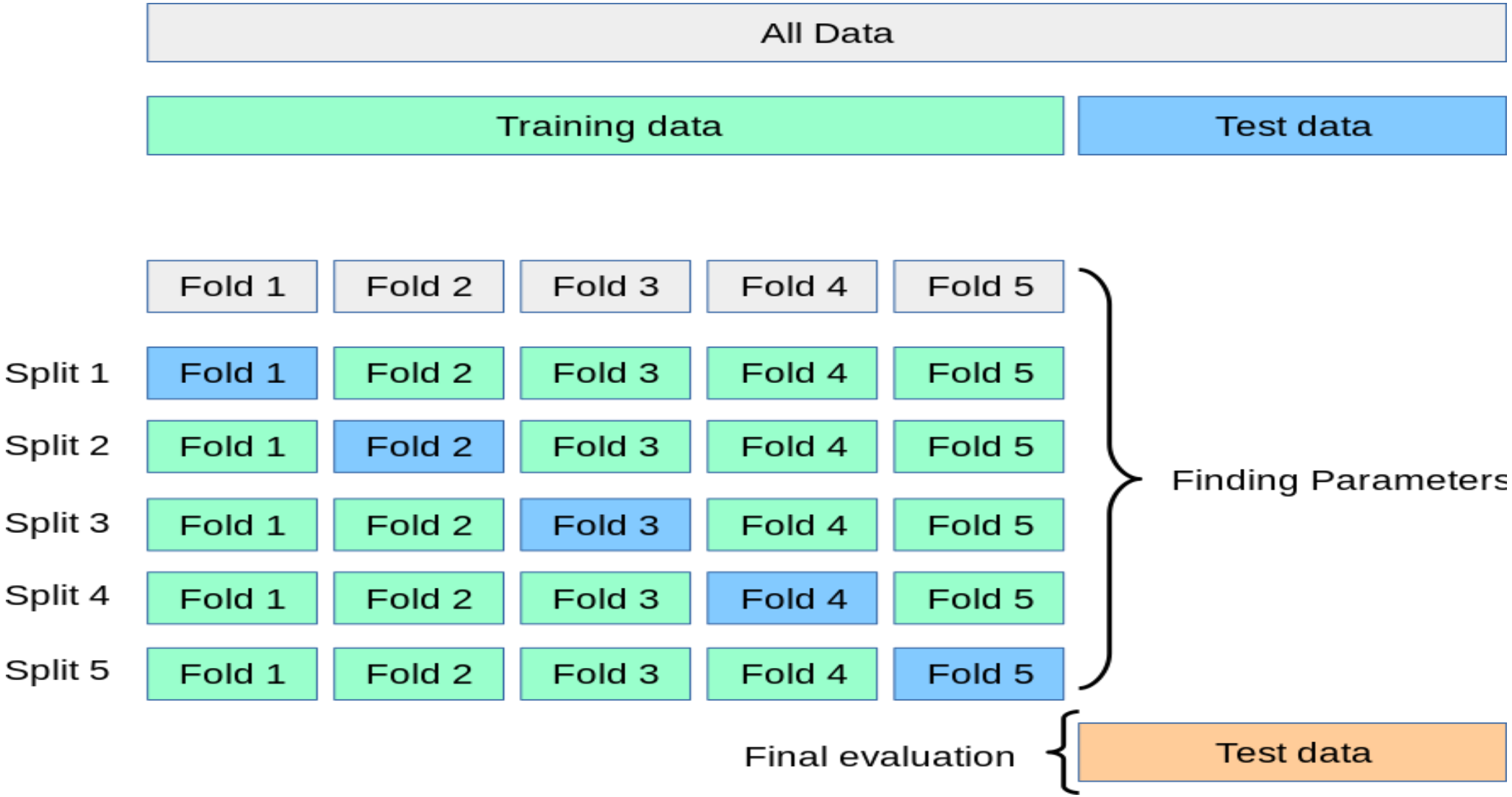
- 70~80% : train data
- 20~30% : Test Data



Overfitting VS underfitting



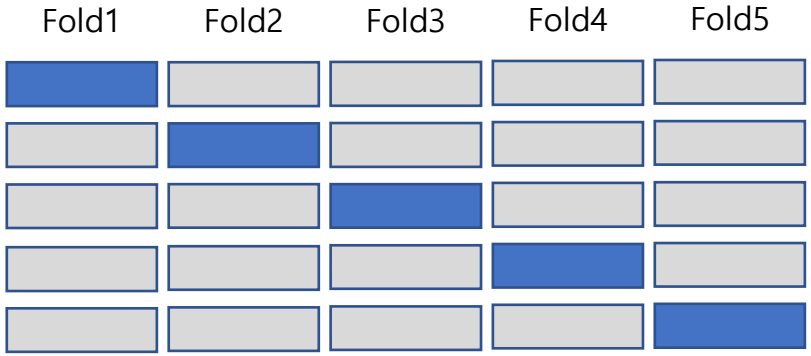
04 Train Test Set Split



04 Train Test Set Split

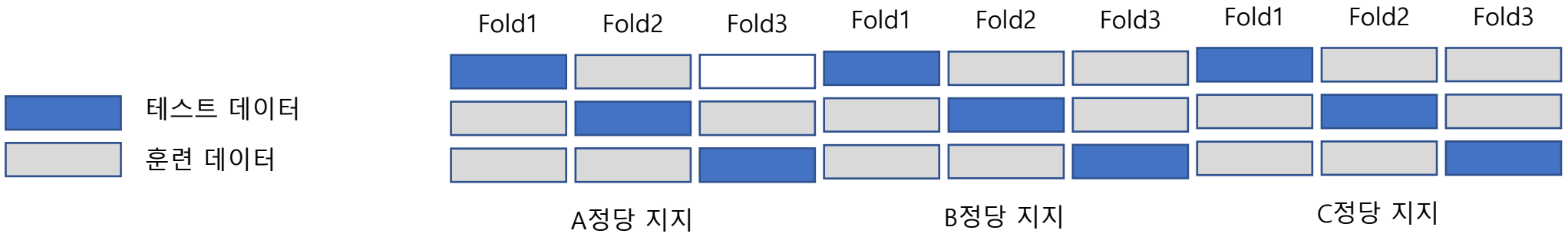
✓ cross_val_score(랜덤없는 교차검증)

→ 가장 널리 쓰이며, 데이터를 5개로 나눠서 4개 폴드는 훈련데이터로, 나머지 1개 폴더는 테스트데이터로 사용, 5회 반복



✓ KFold (랜덤있는 교차검증)

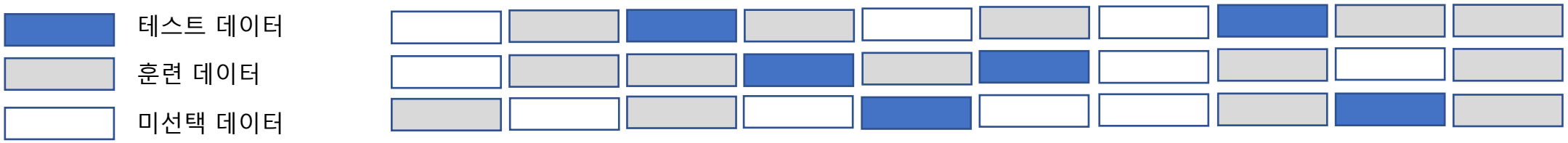
→ 일반적 교차검증은 순서에 영향을 받음, 레이블 값에 따라 폴드를 나누기 때문에 쏠림현상을 줄일 수 있음



04 Train Test Set Split

✓ Suffle-split cross_validation(임의분할 교차검증)

→ 훈련데이터와 테스트데이터를 구성할 때 다른 교차검증에 사용되었던 데이터로 랜덤으로 선택하게 함
일부 데이터는 훈련데이터와 테스트데이터 어디에도 선택되지 않을 수 있음

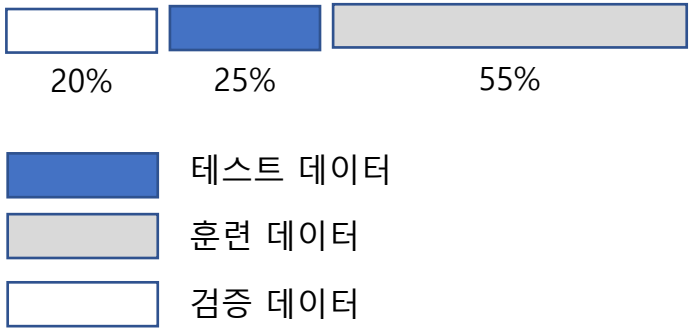


훈련사이즈 5, 테스트사이즈 2, 분할 3인 경우

✓ 훈련-검증-테스트 데이터로 나누기

→ 훈련데이터와 테스트데이터 이외에 검증데이터를 포함

→ 일반화 경향을 파악함



04 Train Test Set Split

Exercise

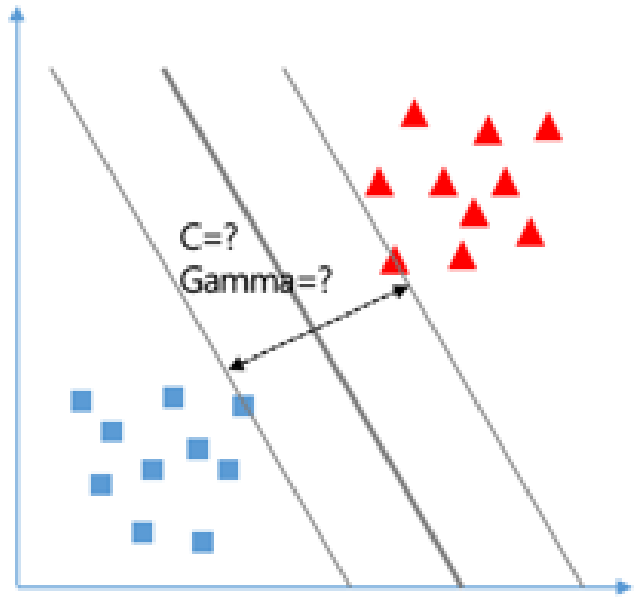
Model Selection & Parameter Search

Model Selection and Apply

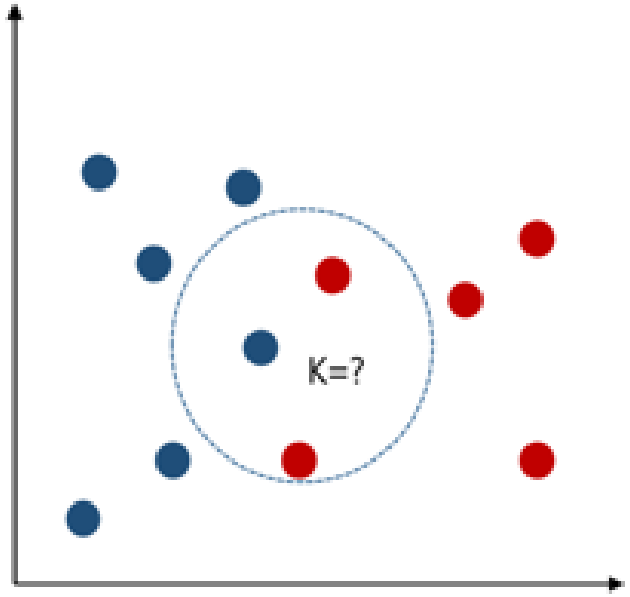
Model Selection and Apply

- to find best parameter to fit a purposed goal

SVM



KNN



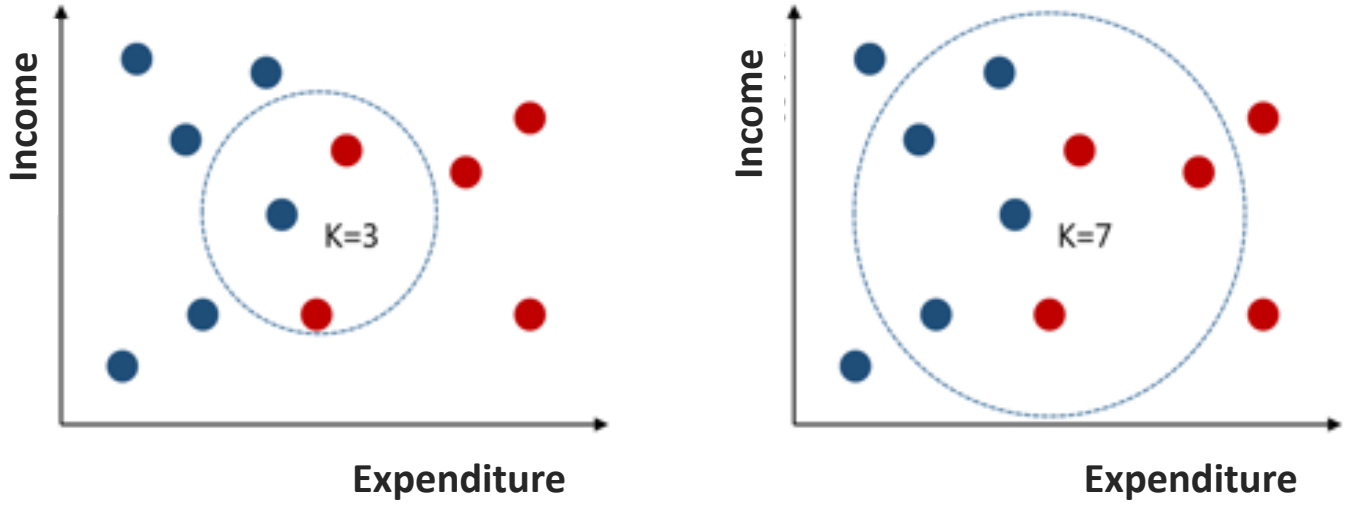
Machine Learning Model

Model Example :
KNN (K-Nearest Neighbor)

Nearest Neighbor

Concept

- Unsupervised learner for implementing neighbor searches.
- instance-based learning method
- How to measure distance between each data and classify it by referring to labels of k-difference

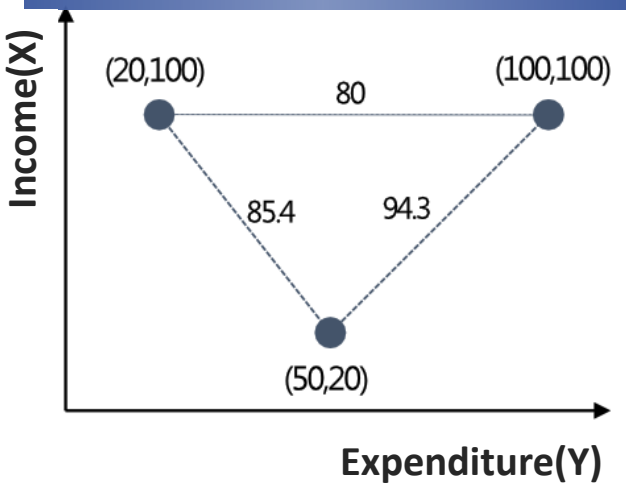


K-Nearest Neighbor

A method to calculate distance

- To apply appropriate method to find a best parameter
- basic method : Euclidean Distance

2 Mensions

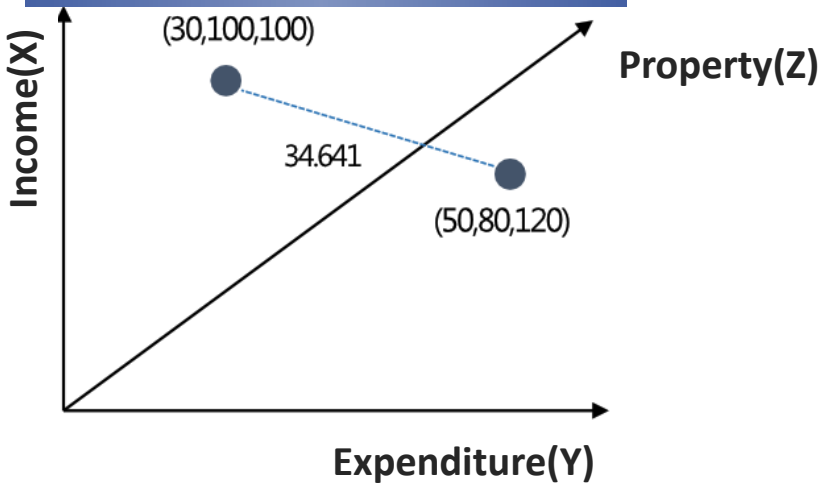


$\sqrt{(20-50)^2+(100-20)^2}$=85.4

$\sqrt{(100-50)^2+(100-20)^2}$=94.3

$\sqrt{(100-20)^2+(100-100)^2}$=80

3 Dimensions



$\sqrt{(30-50)^2+(100-80)^2+(100-120)^2}$=

$\sqrt{(20)^2+(20)^2+(20)^2}$=34.641

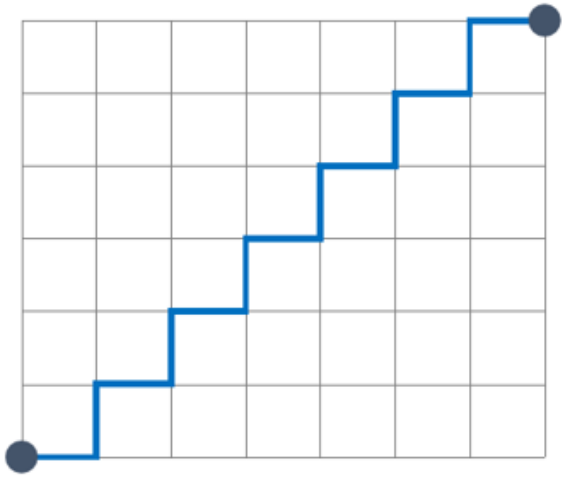
거리 $=\sqrt{(x1-x2)^2+(y1-y2)^2+(z1-z2)^2}$

K-Nearest Neighbor

A method to calculate distance

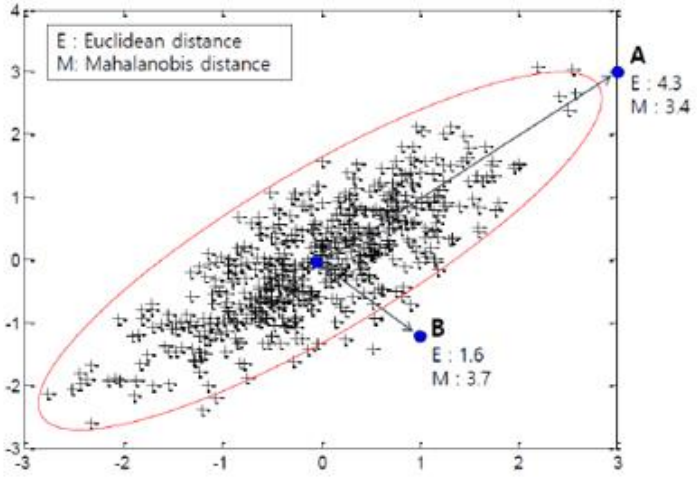
Manhattan Distance

$$d(A,B) = \sum_{i=1}^{n} |a_i - b_i|$$



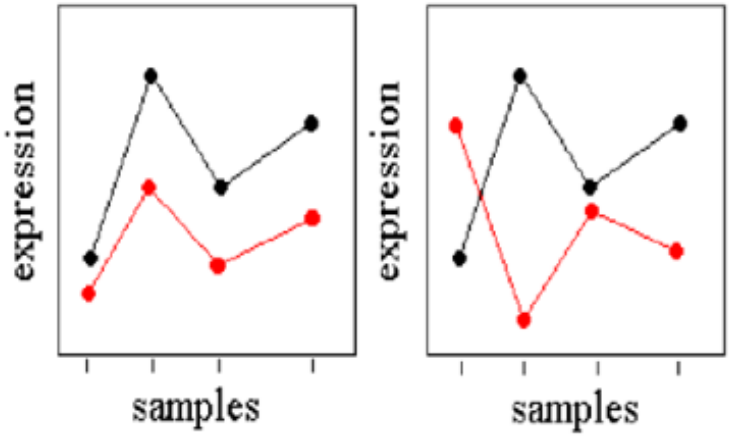
Mahalanobis Distance

$$D^2 = (x-m)^T \cdot C^{-1} \cdot (x-m)$$



Correlation Distance

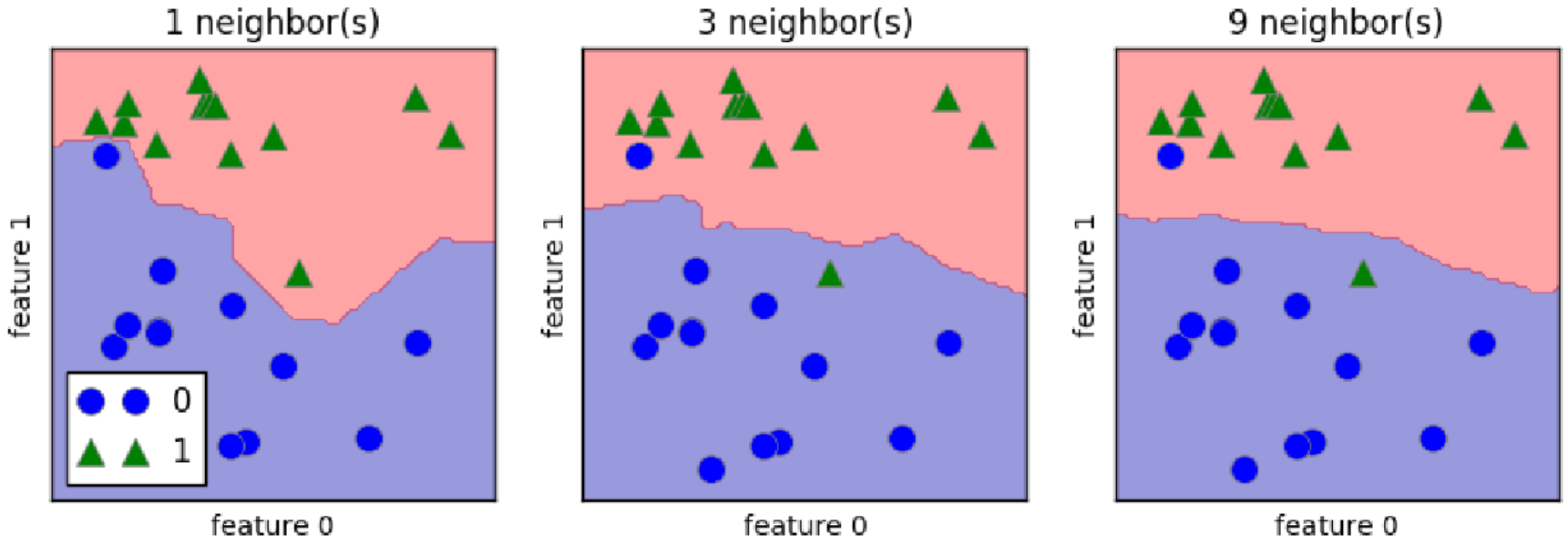
d = 1 - r **r = Z(x)·Z(y)/n**



K-Nearest Neighbor

What is the best K?

- predictability according to k
- small k : to increase the accuracy of train data, can not be suitable for validation/test data
- higher k : can be generalized, but low accuracy



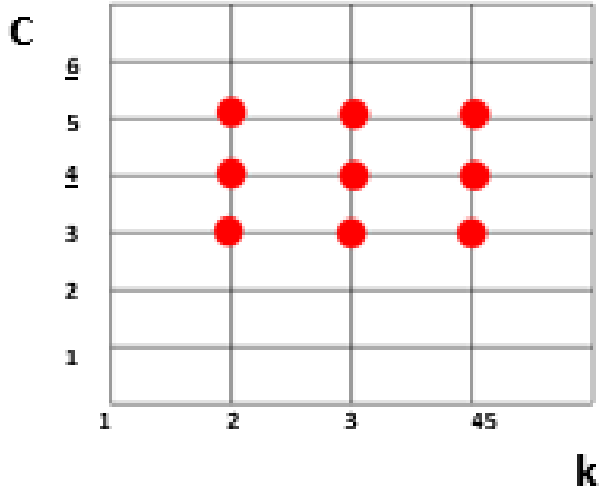
Model Selection and Apply

Model Selection and Apply

- Machine learning is to apply a variety of hyperparameters to find the optimal algorithm for the data
- Grid Search: Exhaustive search over specified parameter values for an estimator.
- Random Search: not all parameter values are tried out, but rather a fixed number of parameter settings is sampled from the specified distributions.

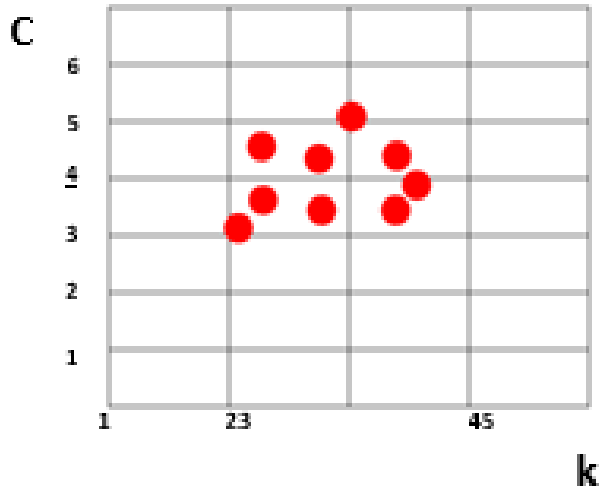
Grid Search

K={2, 3, 4}
C={3, 4, 5}



Random Search

K=2 ~ 4
C=3 ~ 5



05 Model Selection & Parameter Search

✓ Grid Search

→ exhaustively generates candidates from a grid of parameter values specified with the param_grid parameter

```
param_grid = [ {'C': [1, 10, 100, 1000], 'kernel': ['linear']},
    {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf']}
]
```

✓ Randomized Search

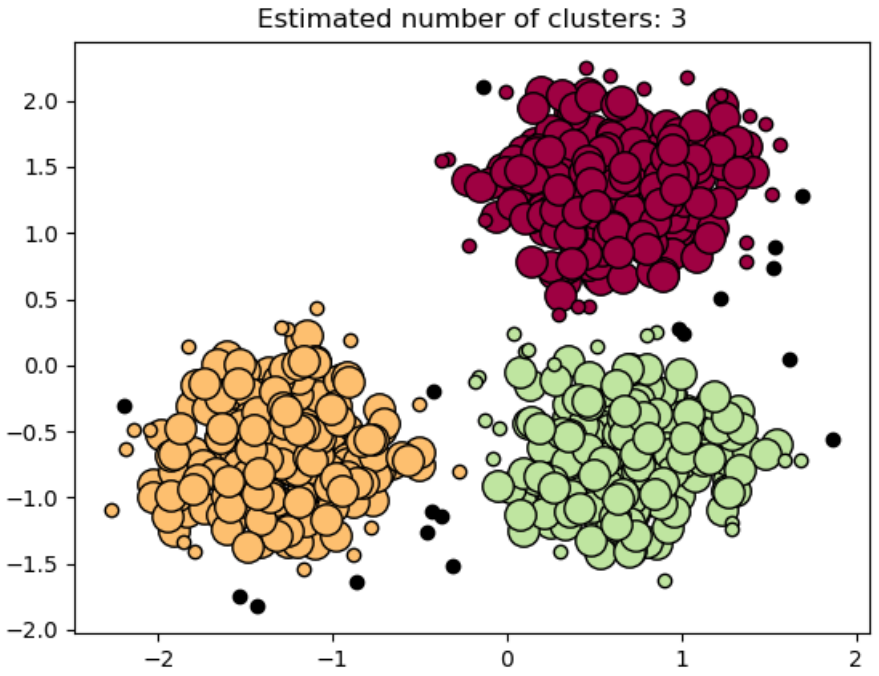
→ each setting is sampled from a distribution over possible parameter values.

```
{'C': scipy.stats.expon(scale=100), 'gamma': scipy.stats.expon(scale=.1),
    'kernel': ['rbf'], 'class_weight':['balanced', None]}
```

05 Model Selection & Parameter Search

✓ Model Training Algorithm

```
from sklearn.neigherrs import KneighborsClassifier
Knn=KNeighborsClassifier(n_neighbors=3)
Knn,fit(X_train, y_train)
```



#1. for Train Data

```
from sklearn.neigherrs import KneighborsClassifier
Knn=KNeighborsClassifier(n_neighbors=3)
Knn,fit(X_train, y_train)
```

#2. Evalluation for Training

```
Knn.score(X_train, y_train)
pd.crosstab(y_train.실제레이블변수, knn.predict(X_train))
```

#3 . Evalluation for Test

```
Knn.fit(X_test, y_test)
Knn.score(X_test, y_test)
pd.crosstab(y_test.실제레이블변수, knn.predict(X_test))
```

Model Evaluation

05 Model Evaluation

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

$$Accuracy = \frac{TruePositive + TrueNegative}{TotalSample}$$

n=165	**Predicted: NO**	**Predicted: YES**
Actual: NO	50	10
Actual: YES	5	100

$$\therefore Accuracy = \frac{100 + 50}{165} = 0.91$$

05 Model Evaluation

✓ Classification Accuracy

1) Accuracy

– the ratio of number of correct predictions to the total number of input samples.

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

2) Confusion matrix

– a matrix as output and describes the complete performance of the model.

accuracy = (90+80)/200 = 85%

	0	1
0	90	10
1	20	80

	Negative	Positive
Negative	TN	FP
Positive	FN	TP

05 Model Evaluation

- Precision : the number of correct positive results divided by the number of positive results predicted by the classifier
- Recall : the number of correct positive results divided by the number of ***all*** relevant samples
- f-score

$$F1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

	0	1
0	90	10
1	20	80

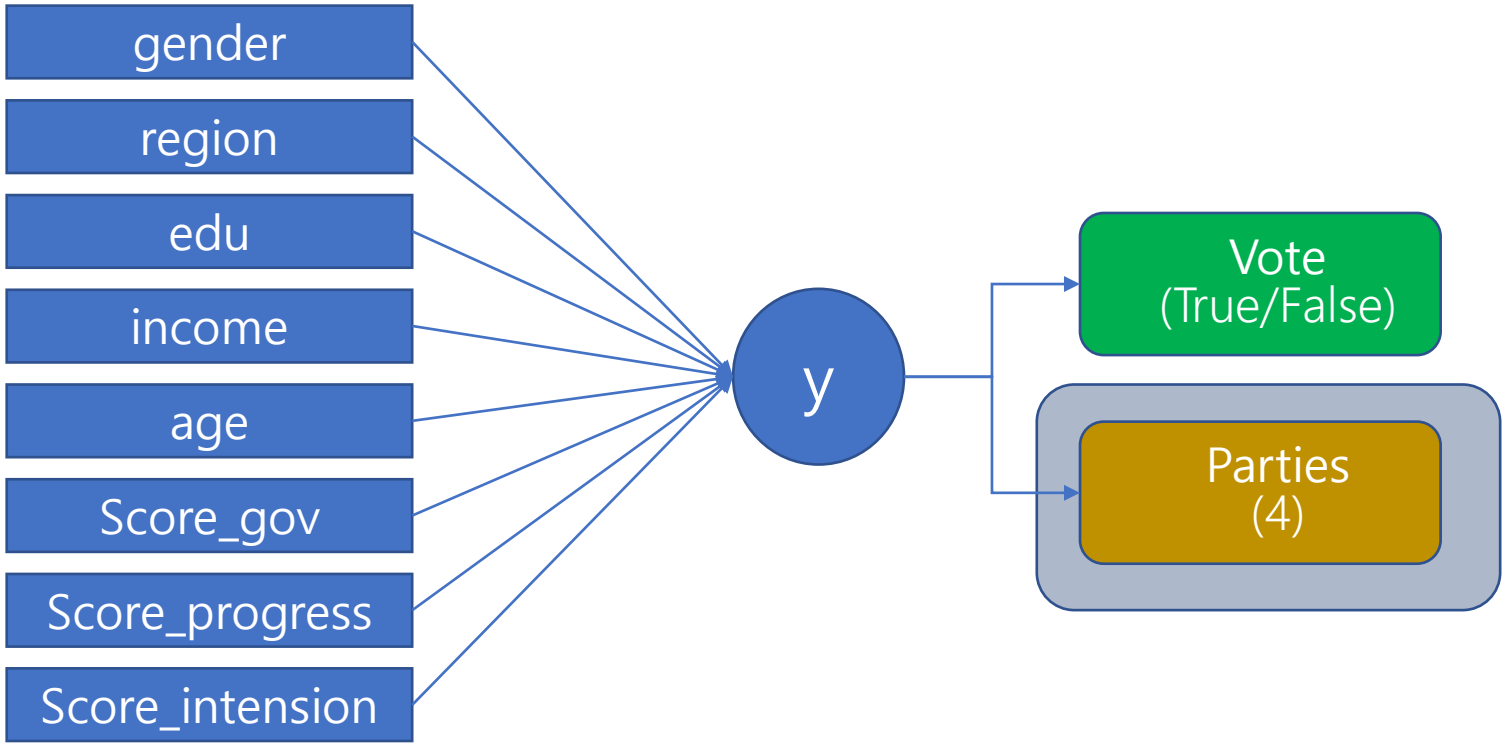
	Negative	Positive
Negative	TN	FP
Positive	FN	TP

precision = TP / (TP + FP)

recall = TP / (TP + FN)

Exercise

06 Multiple Classification



Exercise

Thank You!