교차검증
(Cross Validation)

- 여러 번 성능을 평가한다
- 검증세트
- K-Fold CV, Stratified K-fold CV
- LOOCV

훈련 데이터 / 검증 데이터 / 테스트 데이터

훈련 데이터(training set): 모형 적합

모수의 적합과 모수의 추정에 사용

검증 데이터(validation set): 모형 선택

parameter tuning, 변수 선택, 모형 선택

테스트 데이터(test set): 최종 평가

모형 적합과 모형 선택이 끝난 후 최종 모형의 오류를 측정

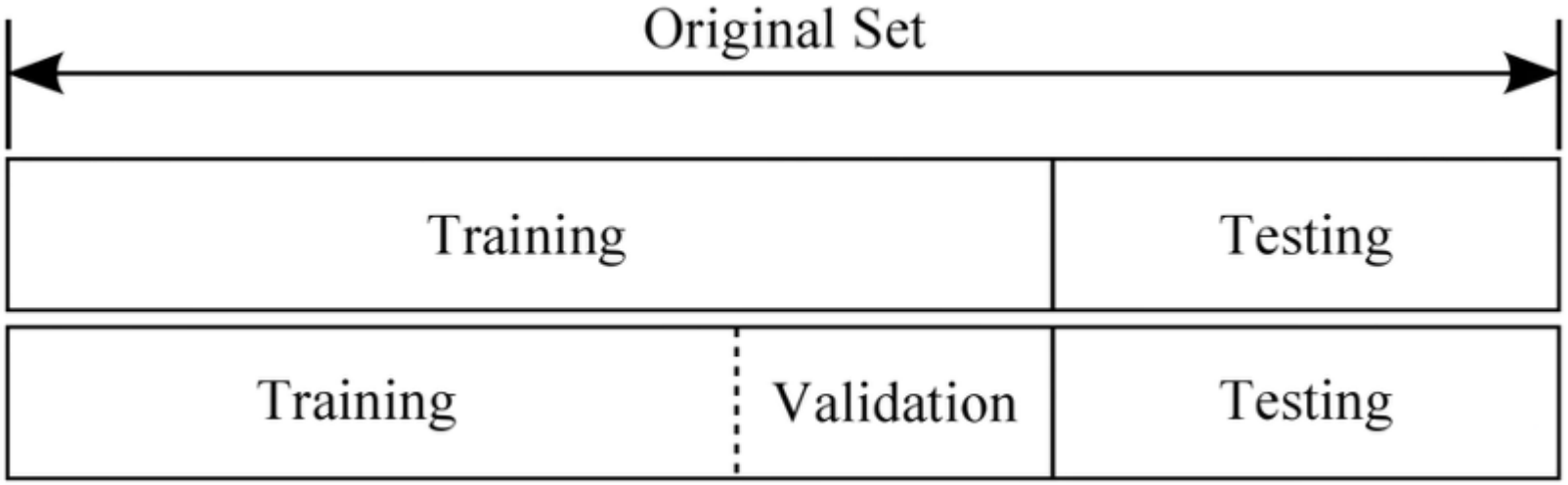
훈련 데이터 / 검증 데이터 / 테스트 데이터

지도 학습기(Supervised Learner)가 하는 작업은 **훈련 데이터**로부터 주어진 데이터에 대해 예측하고자 하는 값을 올바로 추측해내는 것이다.

이 목표를 달성하기 위해서는 훈련용 데이터 맞춤으로 만들어진 모형이 기존의 **훈련 데이터에 나타나지 않던 상황까지도 일반화**하여 처리할 수 있어야 한다.

앞 예시에서는 학습하는 데 사용된 훈련 데이터를 가지고 모형의 성능을 평가했기 때문에 공정하지 않다.

훈련 데이터 / 검증 데이터 / 테스트 데이터

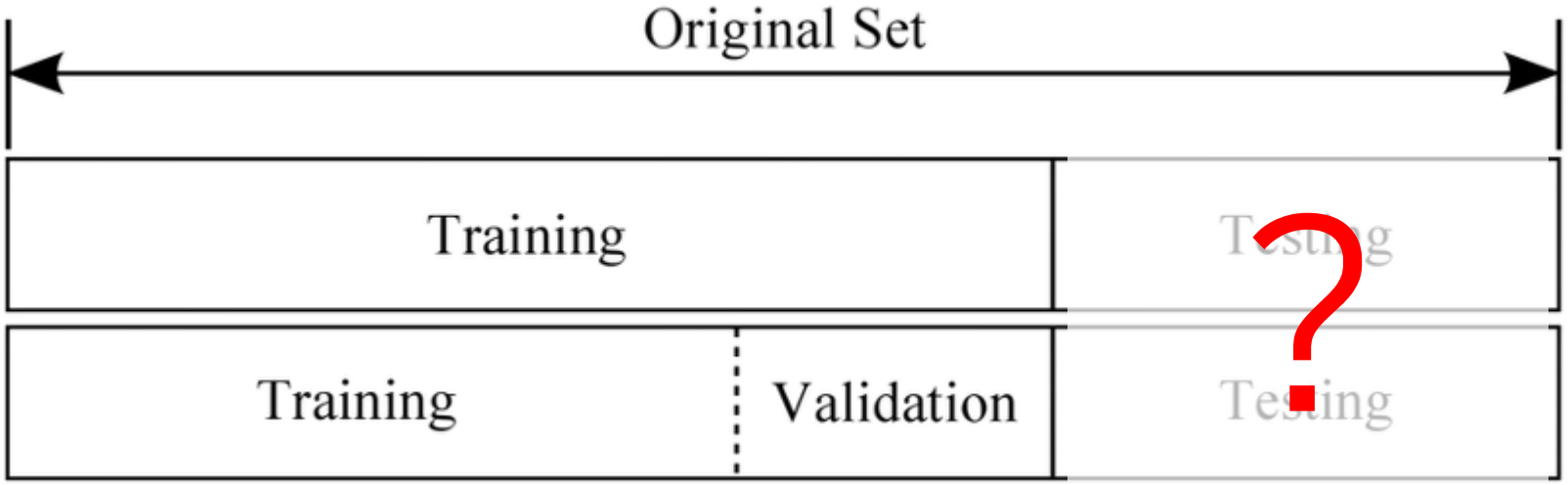


데이터를 나누는 비율은 50:25:25 또는 60:20:20이 많이 쓰인다.

모형 선택을 하지 않는 경우 validation set을 생략하고
70:30이나 80:20 등의 비율로 나누기도 한다.

Source : http://cfile22.uf.tistory.com/image/9951E5445AAE1BE0258820

훈련 데이터 / 검증 데이터 / 테스트 데이터



Kaggle과 같은 분석 대회 플랫폼에서는 평가의 공정성을 위하여 참가자에게 test dataset을 공개하지 않는 경우도 있다.

K-fold 교차검정(K-fold cross validation)

원 데이터를 k개의 데이터로 등분한 후 k-1개는 학습용, 1개는 검증(테스트)용으로 학습과 성능 추정을 k번 반복한 뒤 평균적인 값을 사용

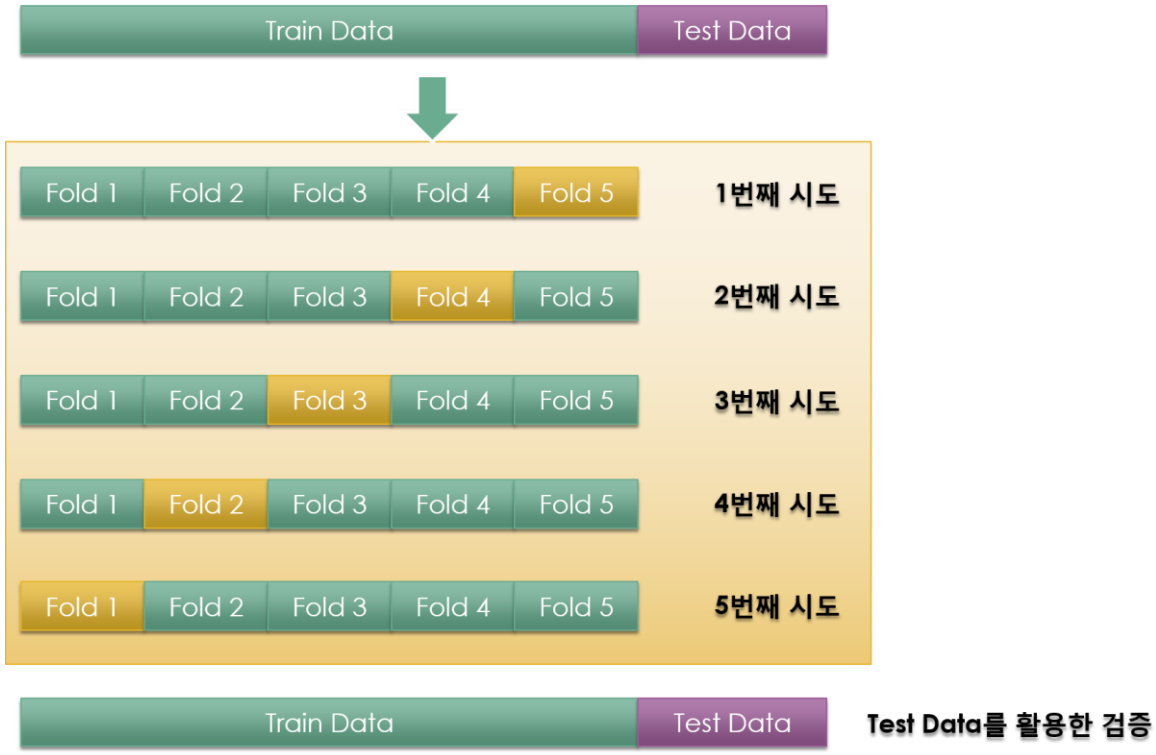
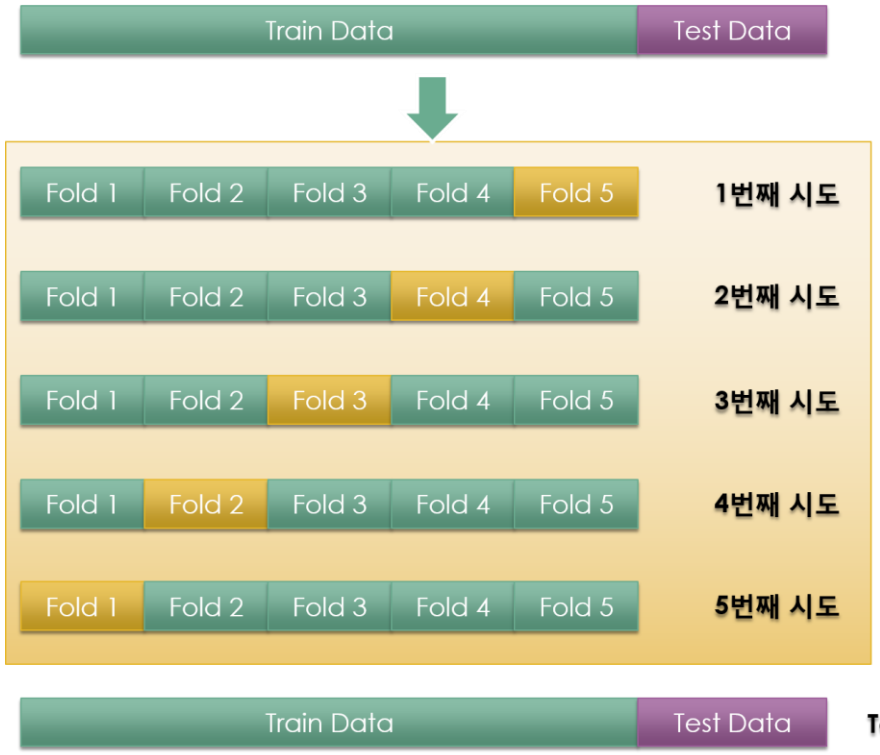


Image Source : http://cinema4dr12.tistory.com/1275, http://issactoast.com/89?category=248085

K-fold 교차검정(K-fold cross validation)

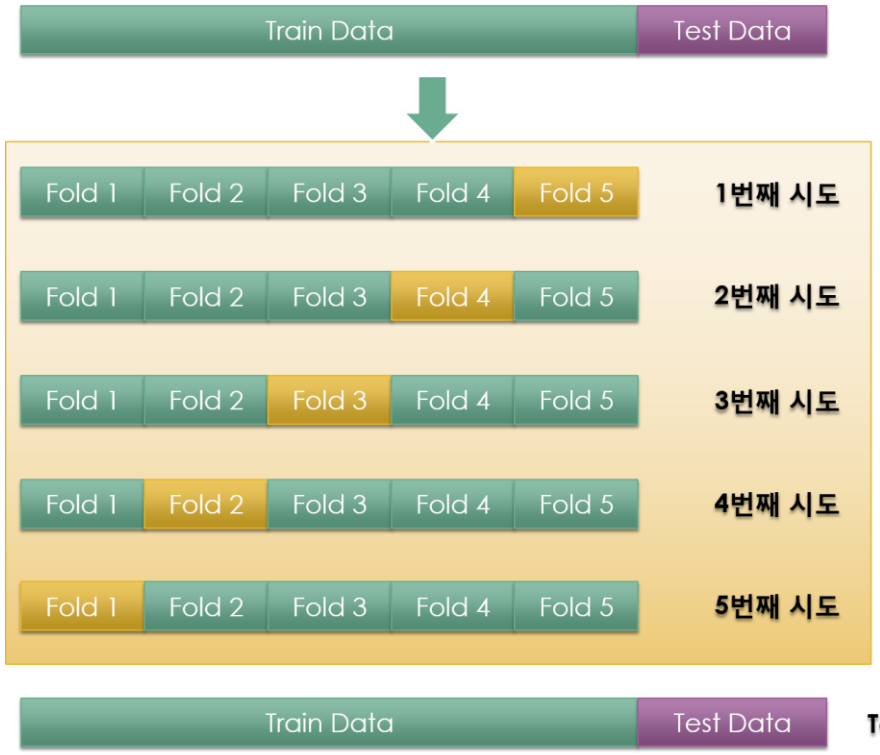
원 데이터를 k개의 데이터로 등분한 후 k-1개는 학습용, 1개는 검증(테스트)용으로 학습과 성능 추정을 k번 반복한 뒤 평균적인 값을 사용



시간이 오래 걸리나
과적합에 대한 우려가 적음

K-fold 교차검정(K-fold cross validation)

원 데이터를 k개의 데이터로 등분한 후 k-1개는 학습용, 1개는 검증(테스트)용으로 학습과 성능 추정을 k번 반복한 뒤 평균적인 값을 사용 또는 1개 선택



시간이 오래 걸리나
과적합에 대한 우려가 적음

k= 10또는 20이 많이 쓰임

training set, validation set, test set 요약

데이터를 크게 training set, validation set, test set으로 나눌 수 있다.

- Training set(학습 데이터): 원래 주어진 데이터, 모형 수립용으로 사용

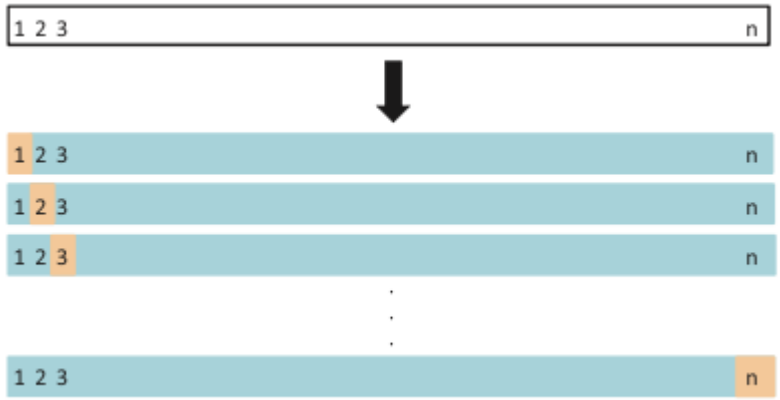
- Validation set(검증 데이터): 모형의 성능을 개선하는 데 사용
  모의 test set으로 생각하면 편하다.

- Test set(시험 데이터): 모형의 성능(정확도)를 평가

LOOCV(Leave-One-Out Cross-Validation)

- Cross-Validation 방법 중 한 번에 한 개의 표본만 validation set으로 이용 -> 데이터의 개수가 n개이면 n번 모형을 만들어 평균을 낸다.

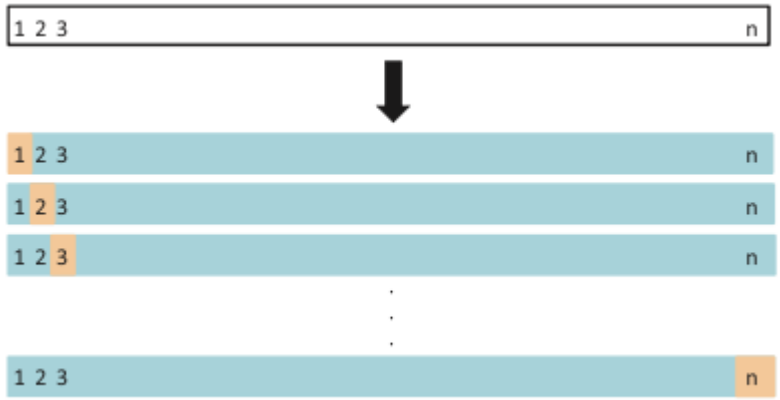
- k-fold CV을 하고 싶으면 cv.knn()을 이용하면 된다.



LOOCV(Leave-One-Out Cross-Validation)

- 장점
  - 모든 표본에 대해 테스트를 하기 때문에 randomness가 없다.
  - (일반적인) k-fold CV보다 안정적이다.

- 단점
  - k-fold CV에 비해 모형의 다양성이 떨어진다.
  - LOOCV는 n-fold CV이므로 계산이 오래 걸린다.
  (단, 선형회귀분석에서는 계산시간을 줄여주는 공식이 존재)



첨언

테스트셋으로 한번만 성능평가(검증)하는 것보다 평가를 여러 번 하기 위한 아이디어이다

회귀 문제에서 성능을 평가할 때는 MSE, MAE, MAPE 등을 사용했다.

분류 문제에서는 분류 문제에 맞는 평가 방법들이 따로 있는데,
특히, 두 개의 범주가 있는 이진 분류 문제에서 평가방법을 다룬다.
분류 문제의 성능 평가 방법은 Precision, Recall 등이 있다.

Stratified K-fold CV (층화 K-fold cross validation)

불균형한 DataSet 을 위한 KFold 방법.

클래스의 비율에 맞추어 추출

150개의 데이터에 0번, 1번, 2번 클래스가 30개, 50개, 70개가 포함되어 있다고 가정하면, 5 fold로 나눈 테스트 셋(한 세트당 30개 데이터)에는 0, 1, 2번 클래스가 각 6개, 10개, 14개가 포함되어야 이상적.

Stratified K-fold는 3:5:7의 비율로 추출

분할 과정에서 라벨 분포의 정보도 받아와야 하므로 for 문 내에서 kf.split(X) 대신 kf.split(X, y) 형태로 사용

출처 : https://jimmy-ai.tistory.com/178

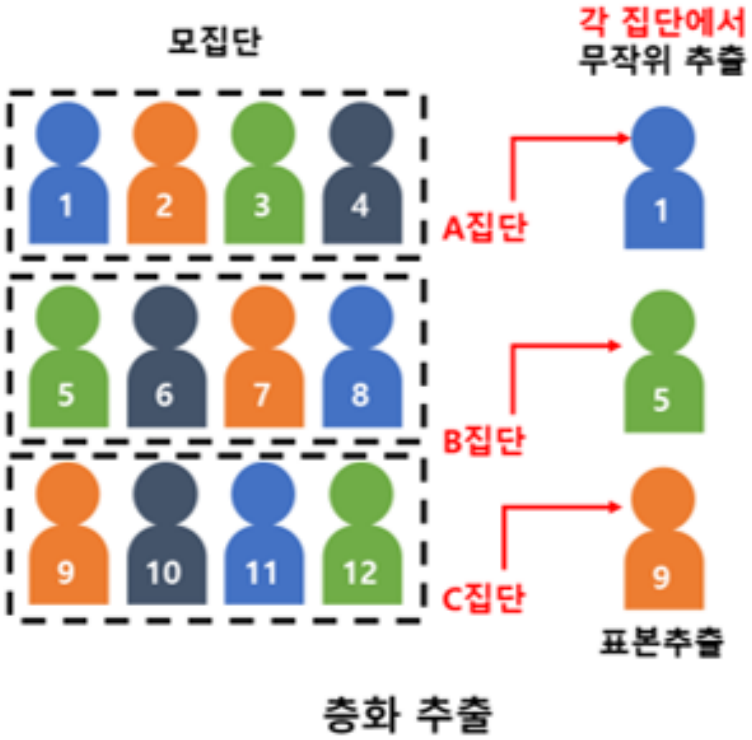
표본 추출법

단순 임의 추출

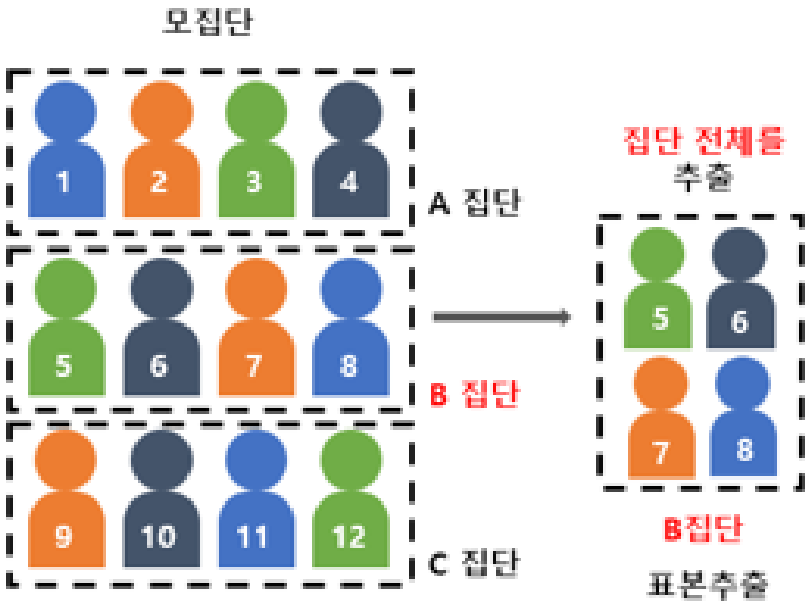
계통추출

출처 :https://datascientistforai.github.io/DataScienceStudy/TimeSeriesData/1_Data_Analysis_Cycle.html

표본 추출법



층화 추출



집락 추출

그리드 서치
(Grid Search)

- 하이퍼파라미터 자동 조정

Grid Search

Grid search (격자 탐색) 은 모델 파라미터에 넣을 수 있는 값들을
순차적으로 입력한 뒤에 가장 높은 성능을 보이는 파라미터들을 찾는 탐색 방법

하이퍼 파라미터 (hyper parameter, 초매개변수)란
 모델을 생성할 때, 사용자가 직접 설정하는 변수가 사전적 정의이다.
랜덤 포레스트 모델에서는 트리의 개수, 트리의 깊이는 몇까지 할 것인지
딥러닝 모델에서는 layer의 개수, 에폭(학습횟수)의 수 등이 된다.
딥러닝에서 파라미터는 가중치이므로 기존의 파라미터를 하이퍼파라미터라 한다.

dt = DecisionTreeClassifier( ) 일 때

```
criterion='gini',
splitter='best',
max_depth=None,
min_samples_split=2,
min_samples_leaf=1,
min_weight_fraction_leaf=0.0,
max_features=None,
random_state=None,
max_leaf_nodes=None,
min_impurity_decrease=0.0,
min_impurity_split=None,
class_weight=None,
presort='deprecated',
ccp_alpha=0.0,
```

이런 파라미터들을 조정하는 것

Grid Search의 단점

- 모든 하이퍼 파라미터를 일률적으로 한번씩 실행. 시간이 많이 걸린다

Feature Scaling

1. 정규화 Normalization

2. 표준화 Standardization

정규화 vs 표준화

- k-인접 이웃 분석은 '거리'의 개념을 사용하므로, 특별한 이유가 없는 경우엔 원 데이터를 그대로 쓰기보다는 데이터를 적당히 가공하는 것이 좋다.

- 이 때 데이터를 가공하는 방법에는 여러 가지가 있는데, 여기서 소개할 것은

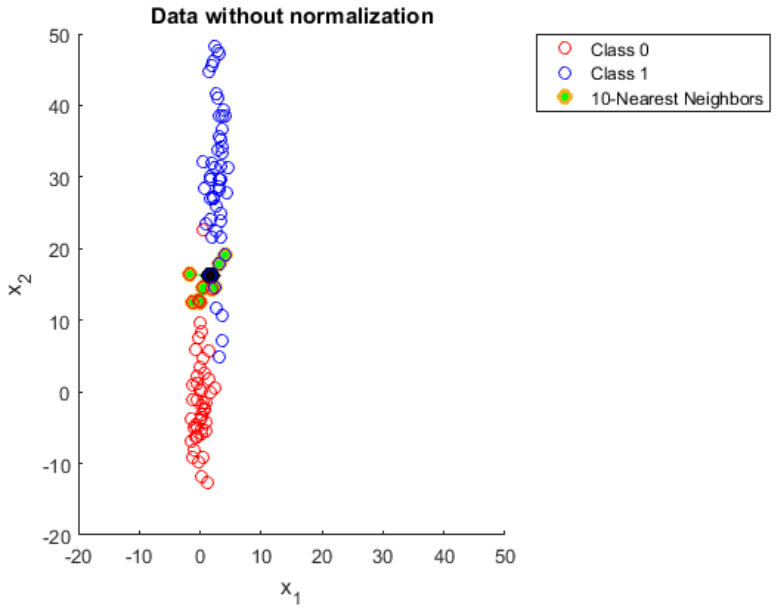
1) 정규화(Normalization)
2) 표준화(Standardization)

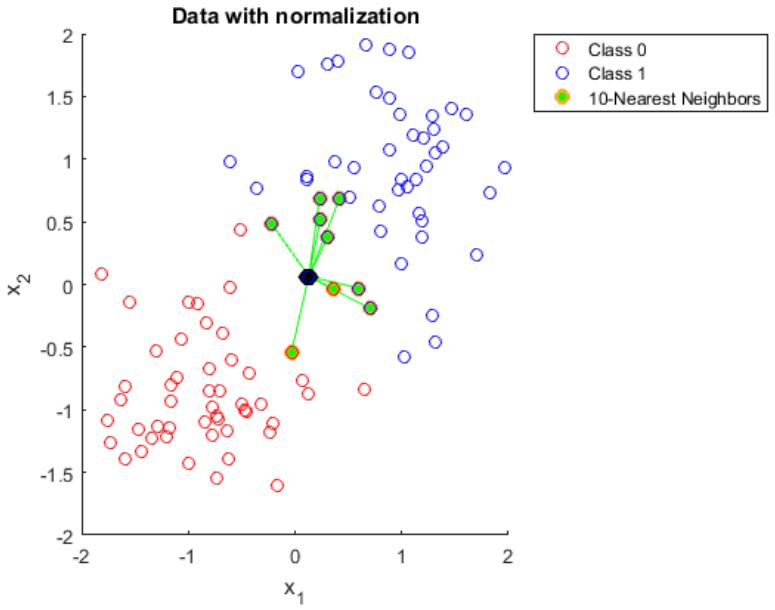
이다.

- 예시를 보면 알겠지만, 두 방법 모두 크기 순서는 바뀌지 않는다.

정규화 vs 표준화

- k-인접 이웃 분석은 '거리'의 개념을 사용하므로, 특별한 이유가 없는 경우엔 원 데이터를 그대로 쓰기보다는 데이터를 적당히 가공하는 것이 좋다.





표준화가 필요한 경우는?

LASSO, ridge regression뿐 아니라 다른 알고리즘 중에서도 크기(norm), 거리(distance)가 사용되는 경우 표준화가 필요할 수 있다.

예를 들면, 군집 분석(Cluster Analysis), 주성분 분석(Principal Component Analysis), K-nearest neighbors 등은 표준화를 하지 않은 경우 잘못된 결과가 도출될 수 있다.

반대로 선형회귀분석(linear regression), 의사결정나무(decision tree) 등은 표준화가 영향을 끼치지 않는다. 예를 들어 선형회귀분석은 표준화를 해도 MSE, R^2 등은 그대로이다.

Source : https://www.listendata.com/2017/04/how-to-standardize-variable-in-regression.html

1) 정규화(Normalization)

scale을 조정할 때 정규화(normalization)는 보통 최솟값이 0, 최댓값이 1이 되도록 scale을 조정하는 것을 의미한다.

함수를 직접 작성하거나, MixMax Scaler 에 의해 Scaling 한다

```
from sklearn.preprocessing import MinMaxScaler
# MinMaxScaler객체 생성
scaler = MinMaxScaler()
# MinMaxScaler 로 데이터 셋 변환. fit() 과 transform() 호출
scaler.fit(iris_df)
iris_scaled = scaler.transform(iris_df)
```

```
print(iris_scaled.max(axis = 0))
print(iris_scaled.min(axis = 0))
```

```
[1. 1. 1. 1.]
[0. 0. 0. 0.]
```

2) 표준화(Standardization)

scale을 조정할 때 표준화(Standardization)는 데이터의 (표본)평균, 분산을 구해 평균 0, 분산 1(표준편차 1)이 되도록 조정하는 것을 의미한다.

평균을 기준으로 얼마나 떨어져 있는지를 나타내게 되고, 2개 이상의 대상이 단위가 다를 때 대상 데이터를 같은 기준으로 볼 수 있게 한다.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(iris_df)
iris_scaled = scaler.transform(iris_df)

print(iris_scaled.mean(axis = 0, ))
print(iris_scaled.var(axis = 0))
```

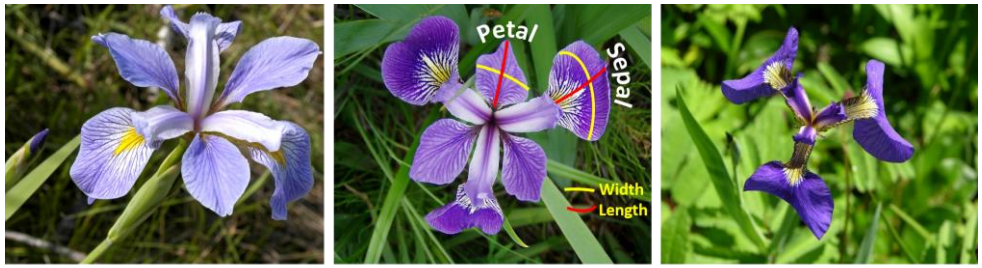
```
[-1.69031455e-15 -1.84297022e-15 -1.69864123e-15 -1.40924309e-15]
[1. 1. 1. 1.]
```

data set - iris

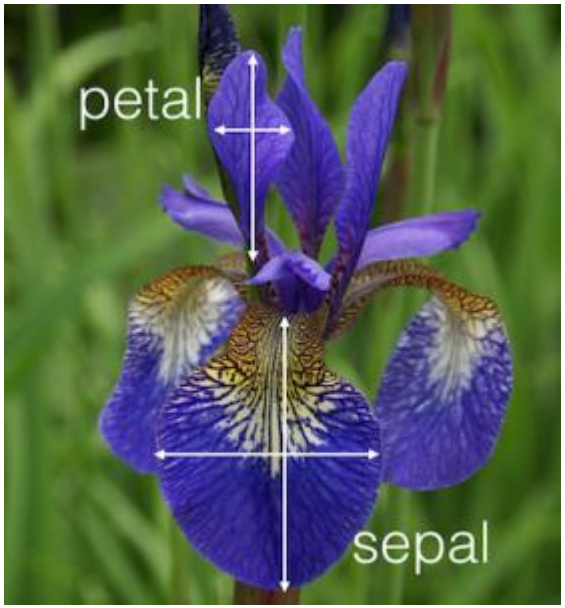
- 통계학자 R. A. Fisher가 소개한 붓꽃(iris) 데이터로, 붓꽃의 3가지 종 (setosa, versicolor, virginica)을 각 종별로 50개, 총 150개의 데이터 수집

  Sepal.Width(꽃받침의 너비), Sepal.Length(꽃받침의 길이), Petal.Width(꽃잎의 너비), Petal.Length(꽃잎의 길이)이다.

- R에 내장되어 있으며, 통계학/데이터 마이닝/ 기계학습 등 여러 분야에서 사용되는 가장 대 표적인 데이터 셋 중 하나이다.



Iris Virginica | Iris Versicolor | Iris Setosa



Source : http://www.socr.umich.edu/people/dinov/2017/Spring/DSPA_HS650/images/IrisDataPic_SepalsPetals_LengthWidth.png
http://blog.kaggle.com/2015/04/22/scikit-learn-video-3-machine-learning-first-steps-with-the-iris-dataset/