19기 정규세션

ToBig's 18기 강의자

강효은

KNN

K-Nearest Neighbor

Contents

Unit 01 | KNN

모델 학습 방법

1 Model-based learning (모델 기반)

데이터로부터 모델을 생성하여 분류 / 예측 진행
Ex) Linear Regression, Logistic Regression

2 Instance-based learning (사례 기반)

별도의 모델 생성 없이 인접 데이터를 분류 / 예측에 사용
모델을 만들지 않고, 새로운 데이터가 들어오면 계산 시작
Ex) KNN, Naïve Bayes

Unit 01 | KNN

K K개의

N Nearest 가까운

N Neighbors 이웃

➔ K개의 가까운 이웃을 찾자 !

Unit 01 | KNN

< KNN 알고리즘 >

특정 공간 내에서 입력과 제일 근접한 k개의 요소를 찾아,
분류 혹은 수치를 예측하는 알고리즘

Instance-based Learning

각각의 관측치(instance)만을 이용하여 새로운 data에 대한 예측을 진행

Memory-based Learning

모든 학습 데이터를 메모리에 저장한 후, 이를 바탕으로 예측

Lazy Learning

모델을 미리 만들지 않고,
새로운 데이터가 들어온 후 학습

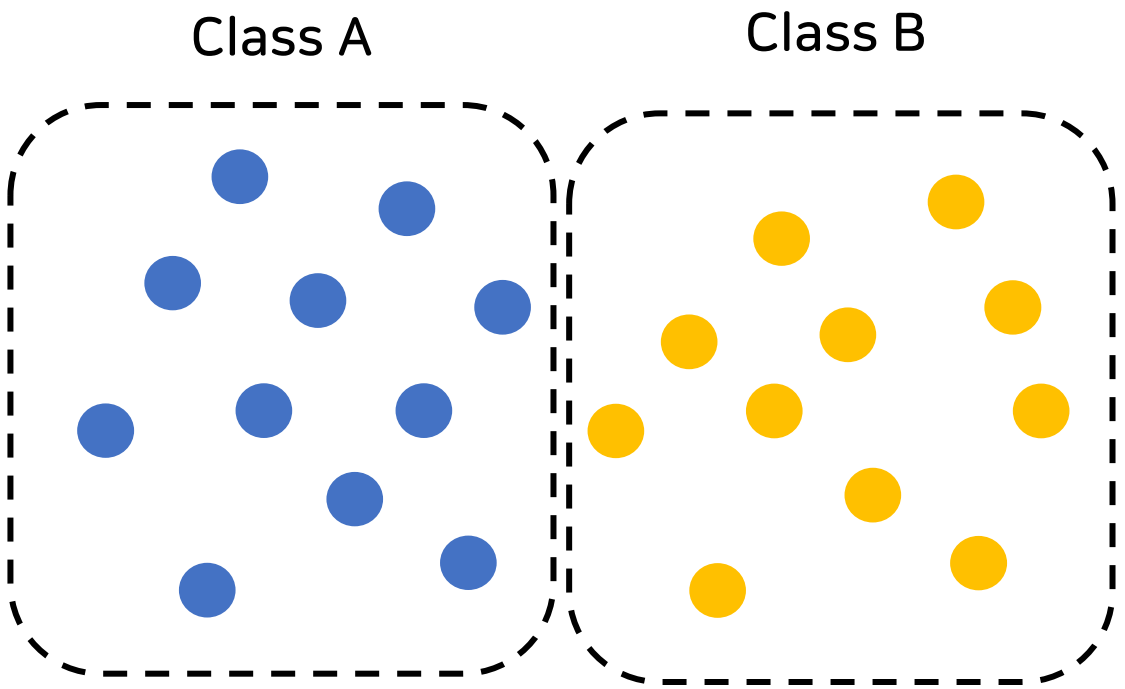
Unit 01 | KNN

K-NN Classification

1. 새로운 데이터 X 추가
2. X로부터 인접한 K개의 학습 데이터 선택
3. 선택된 k개 학습 데이터의 majority class C를 탐색
4. C를 X의 분류결과로 반환

Unit 01 | KNN

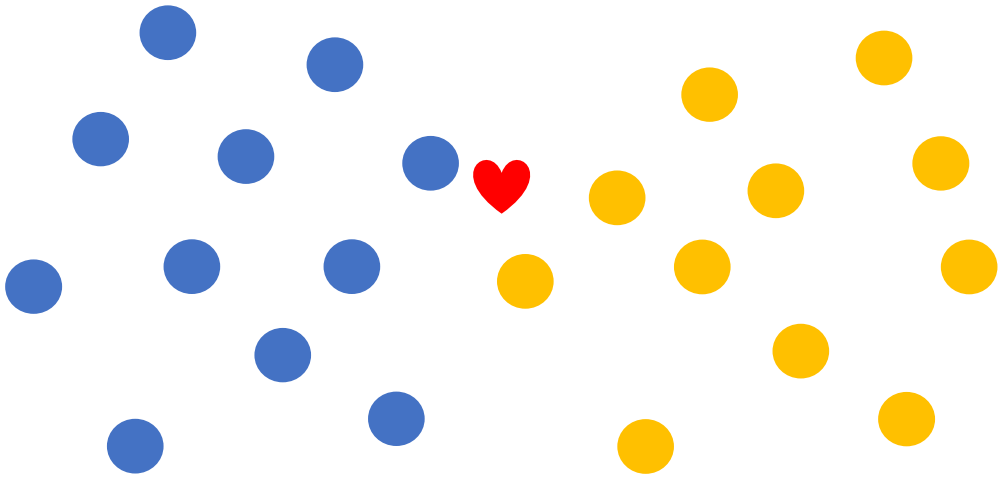
Classification



2개의 class로 분류된 데이터

Unit 01 | KNN

Classification

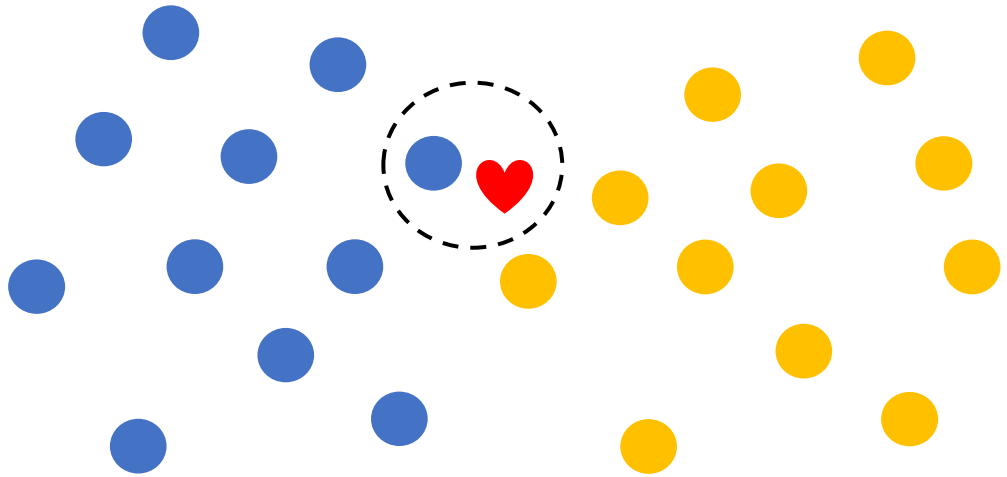


새로운 데이터가 추가됐을 때

Unit 01 | KNN

Classification

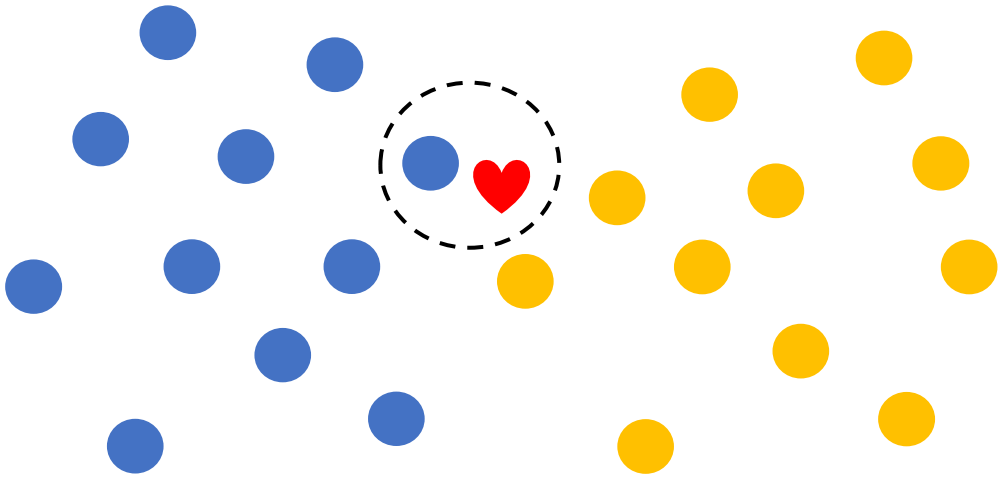
가장 가까운 k개의 이웃 중
빈도가 가장 높은 class를
new data의 class로 예측



1 – NN

Unit 01 | KNN

Classification

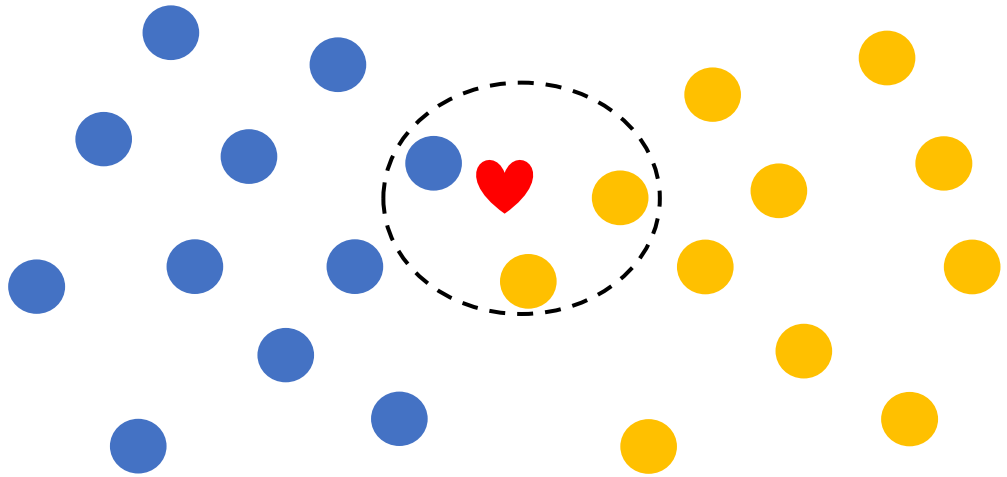


New Data는 class A로 분류

Unit 01 | KNN

Classification

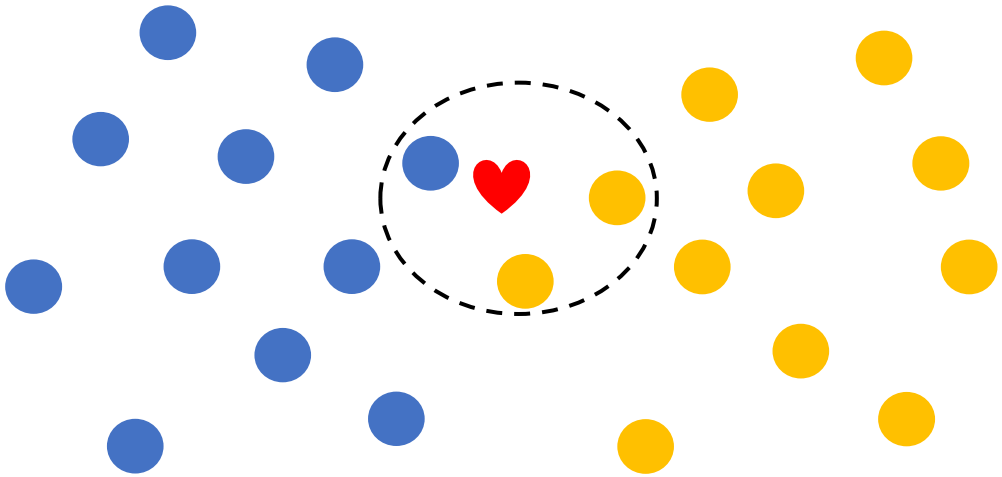
가장 가까운 k개의 이웃 중
빈도가 가장 높은 class를
new data의 class로 예측



3 - NN

Unit 01 | KNN

Classification



New Data는 class B로 분류

Unit 01 | KNN

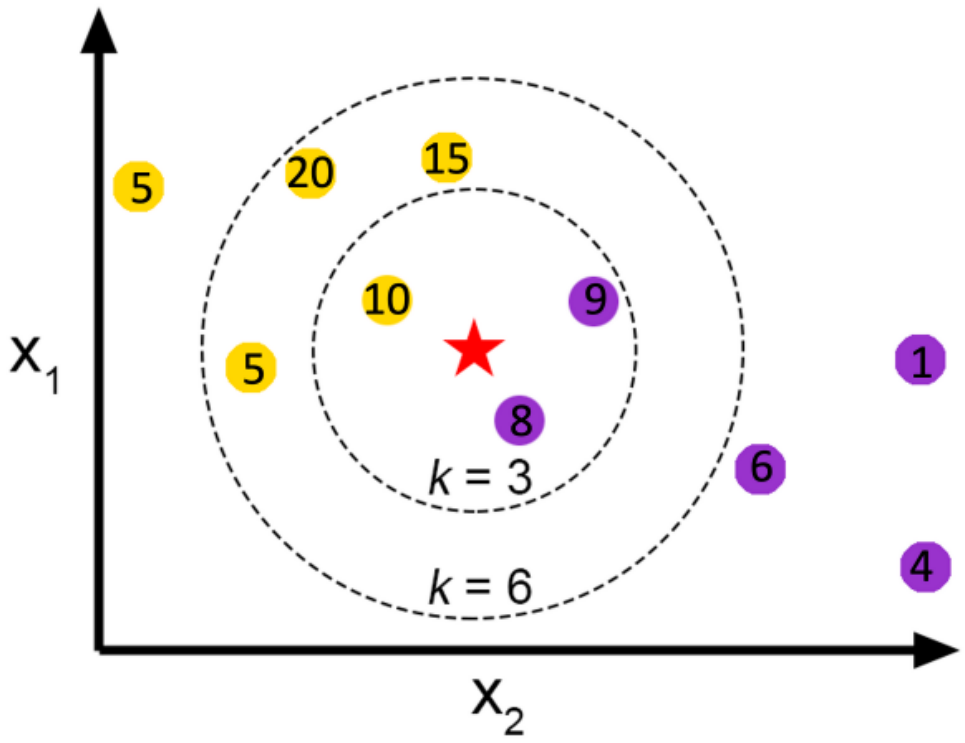
K-NN Regression

1. 새로운 데이터 X 추가
2. X로부터 인접한 K개의 학습 데이터 선택
3. 선택된 k개 학습 데이터의 평균값을 계산
4. 계산된 평균값을 X의 예측값으로 사용

Unit 01 | KNN

Regression

가장 가까운 k개의 샘플을
통해 값을 예측 (ex. 평균)



Predict = (10+9+8)/3

Contents

Unit 02 | KNN Hyperparameter

KNN Hyperparameter 결정

※ Hyperparameter : 어떤 임의의 모델을 학습시킬 때 컴퓨터가 아닌 사람이 직접 튜닝하는 변수

1. **인접한 학습 데이터의 개수 - K**
2. **두 데이터가 얼마나 유사한가 -** Distance Measure 조정

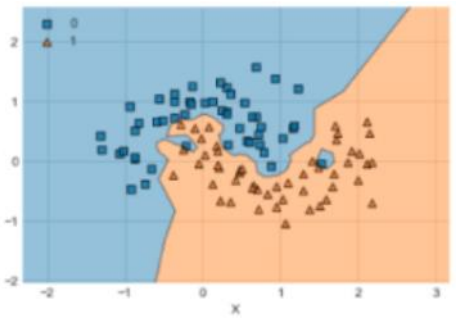
Unit 02 | KNN Hyperparameter

KNN Hyperparameter K 결정

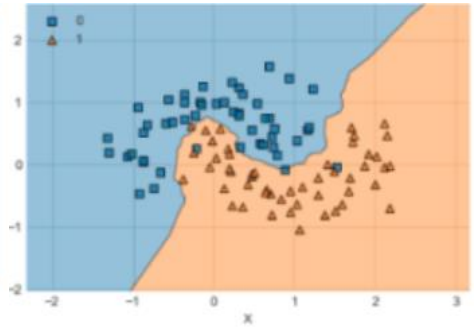
Unit 02 | KNN Hyperparameter

K값에 따른 결과 (결정경계)

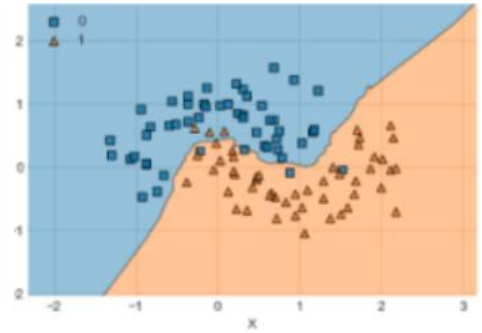
KNN with K=1



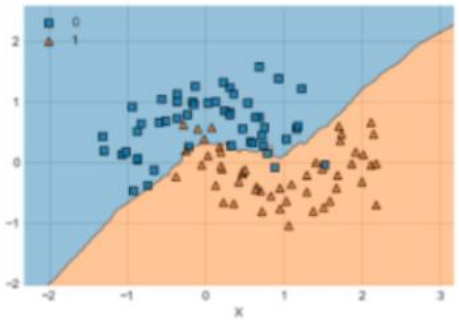
KNN with K=5



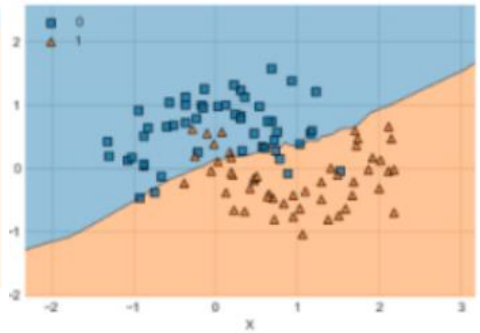
KNN with K=20



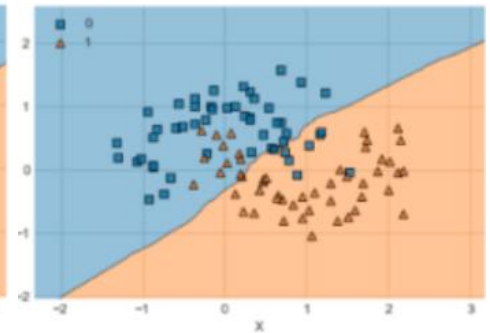
KNN with K=30



KNN with K=40



KNN with K=60



K값이 너무 작으면

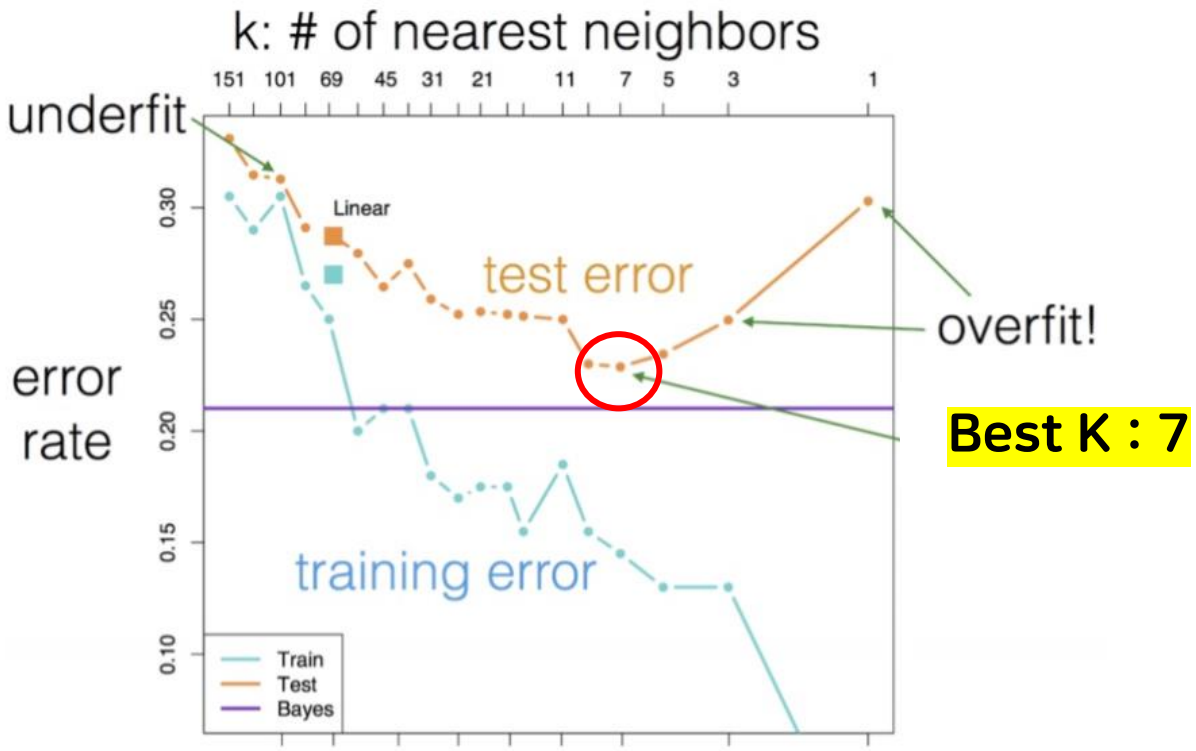
- 데이터의 지역적 특성을 지나치게 반영
- 분류 경계면이 noise에 민감하게 반응
- Overfitting

K값이 너무 크면

- 다른 범주의 개체를 너무 많이 포함
- KNN의 의미가 없어짐
- Underfitting

Unit 02 | KNN Hyperparameter

Hyperparameter K 결정



Best K : 7

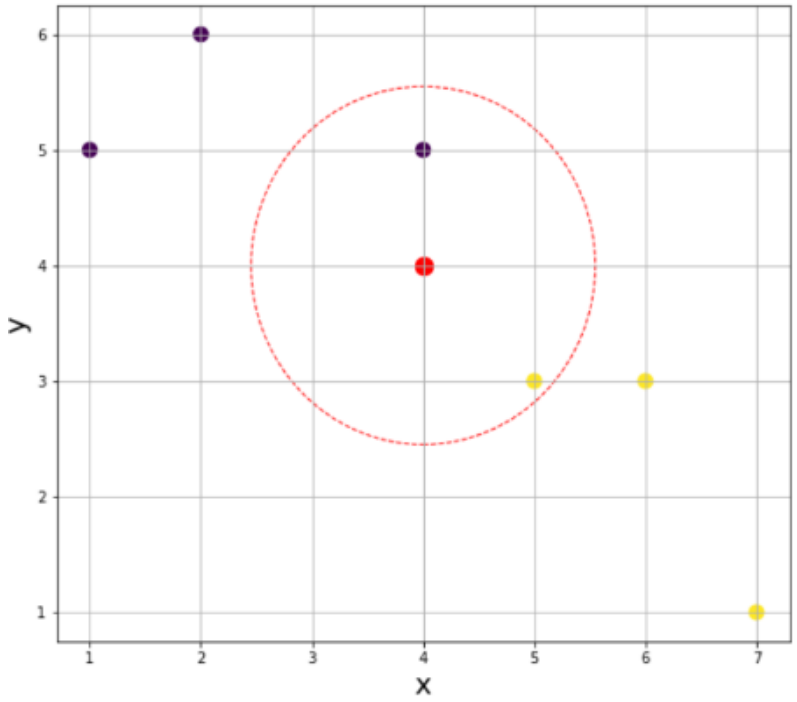
Unit 02 | KNN Hyperparameter

<최적의 K값 선택 방법>

1. 일반적인 규칙은 없음 (분석가의 주관에 의존)
2. 노이즈가 없고 잘 구조화된 데이터의 경우 K값이 작을수록 Good 👍
3. 보통 1~20 사이의 값으로 설정
4. 보통 홀수를 사용

Unit 02 | KNN Hyperparameter

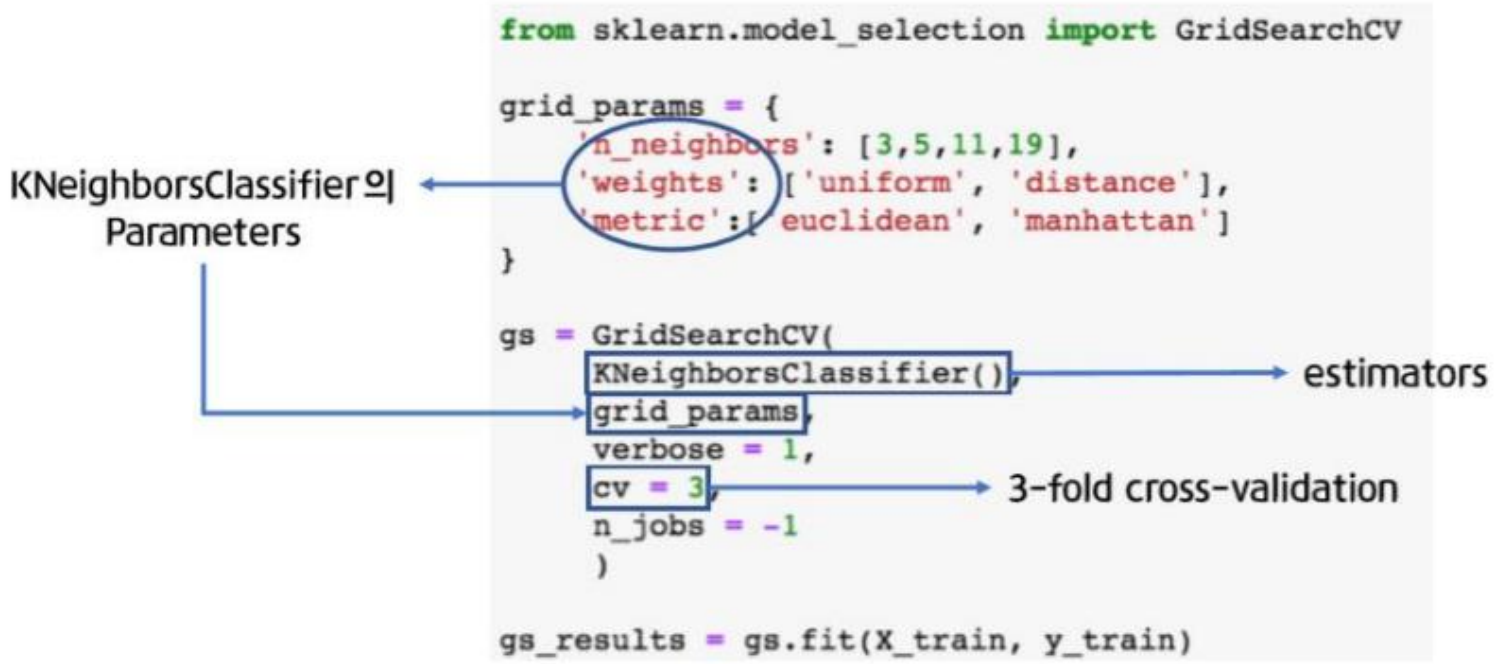
Hyperparameter K 결정



보편적으로 K값을 정할 때, 동률이 나오지 않도록
홀수로 지정

Unit 02 | KNN Hyperparameter

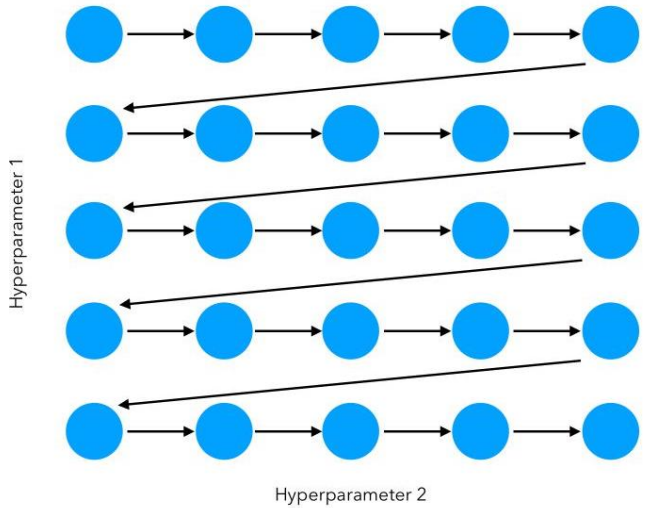
Grid Search & K-fold cross-validation code




```
from sklearn.model_selection import GridSearchCV

grid_params = {
    'n_neighbors': [3,5,11,19],
    'weights': ['uniform', 'distance'],
    'metric': ['euclidean', 'manhattan']
}

gs = GridSearchCV(
    KNeighborsClassifier(),
    grid_params,
    verbose = 1,
    cv = 3,
    n_jobs = -1
    )

gs_results = gs.fit(X_train, y_train)
```

Unit 02 | KNN Hyperparameter

Grid Search(그리드 서치)

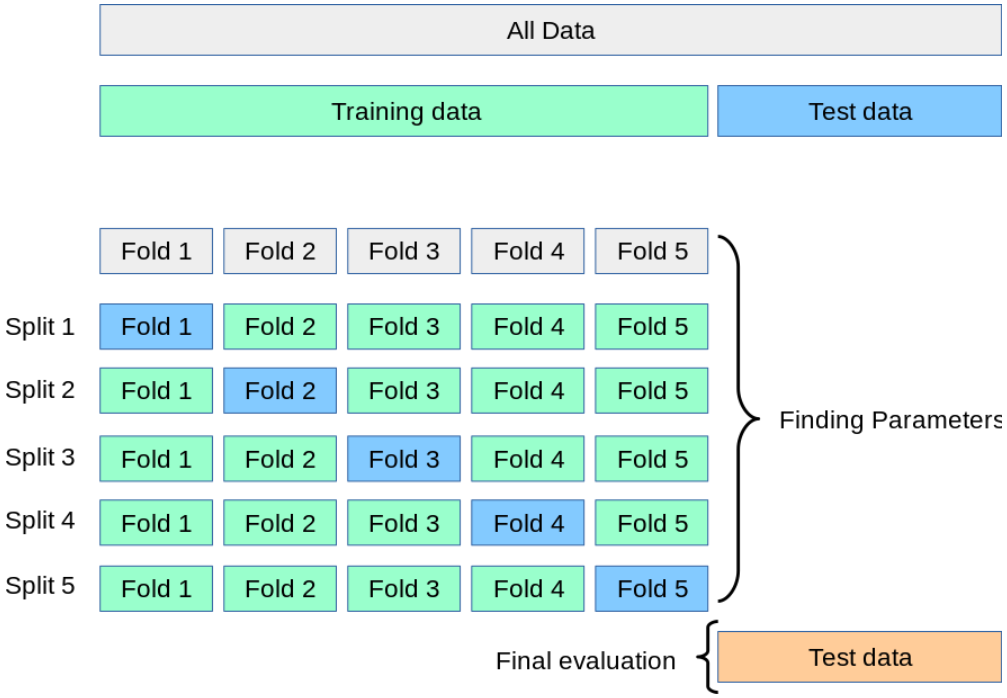


- 격자 무늬로 Hyperparameter를 탐색(Search)
- 모든 파라미터의 경우의 수에 대해 cross-validation 결과가 가장 좋은 파라미터를 고르는 방법

- 장점 : 주어진 공간 내에서 가장 좋은 결과를 얻을 수 있음
- 단점 : 시간이 정말 오래 걸림

Unit 02 | KNN Hyperparameter

K-fold cross-validation (K겹 교차 검증)



- 모든 데이터가 최소 한번은 test-set으로 쓰이도록 함
- 5-fold cross-validation에서 총 5개의 성능 평가지표가 생기게 되는데, 보통 이 값들을 평균을 내어 모델 성능을 평가

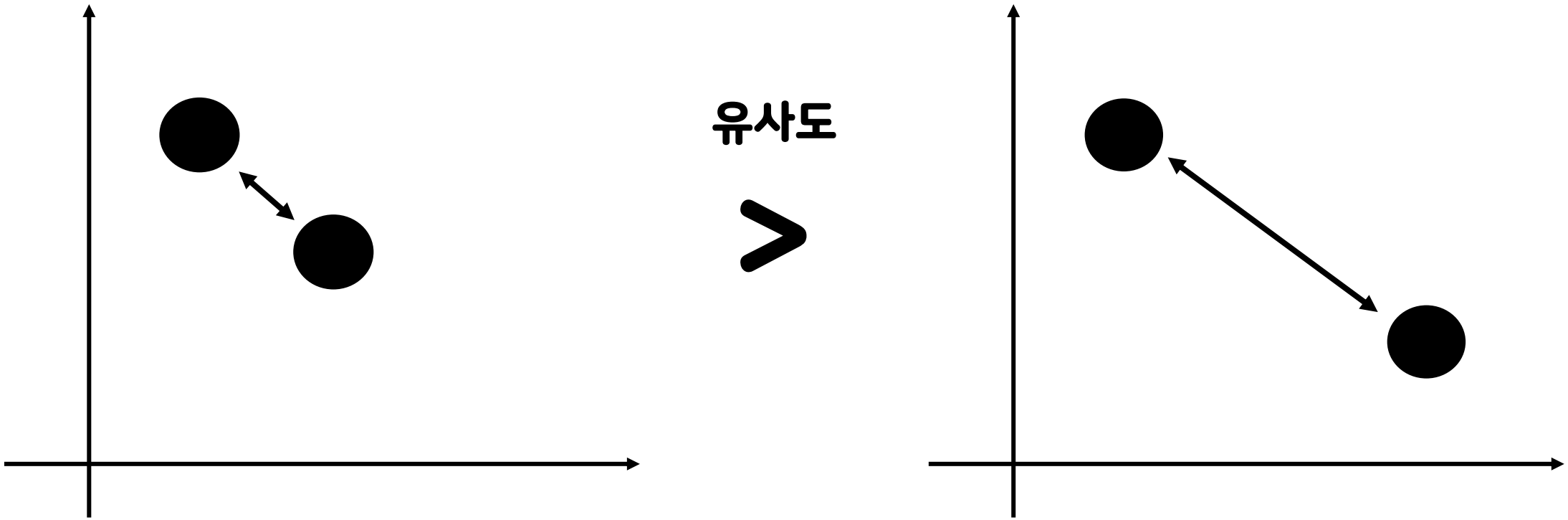
교차 검증을 통한 성능 평가의 목적

- 더 좋은 모델을 선택하기 위해
- Hyperparameter tuning을 위해

Unit 02 | KNN Hyperparameter

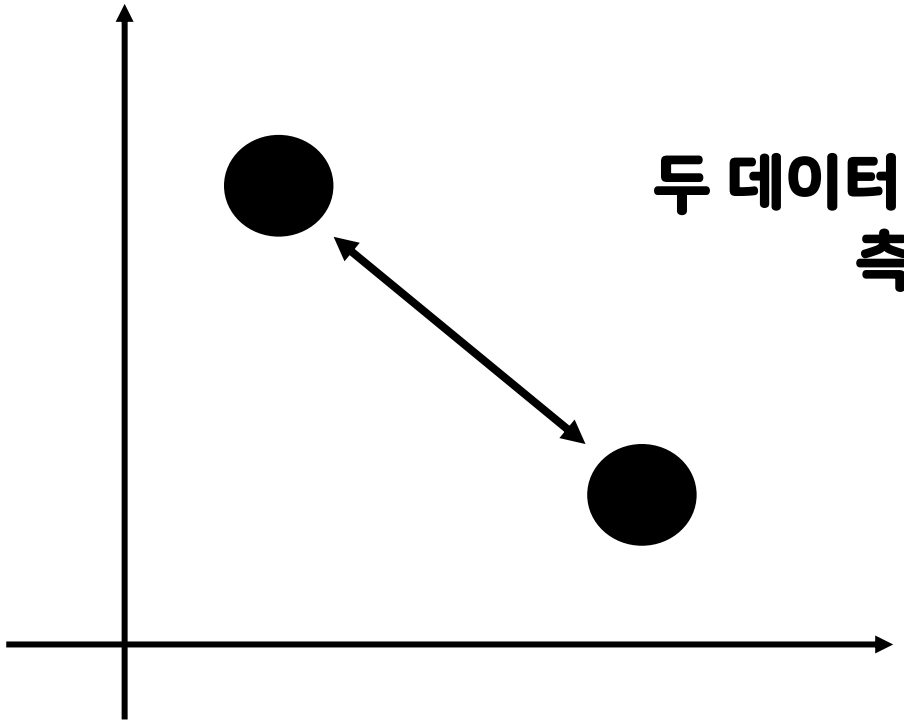
KNN Hyperparameter Distance 결정

Unit 02 | KNN Hyperparameter



거리가 가까울수록 유사도가 높다

Unit 02 | KNN Hyperparameter



두 데이터간 거리는 어떻게 측정할까 ?

Unit 02 | KNN Hyperparameter

Distance Measures

Euclidean Distance
유클리드 거리

Manhattan Distance
맨해튼 거리

Mahalanobis Distance
마할라노비스 거리

데이터 내 각기 다른 데이터 범위, 분산을 가질 수 있으므로,
데이터 **정규화(or 표준화)**를 통해 범위와 분산을 맞추는 것이 중요!

- 거리를 계산할 때, 단위가 큰 특정 변수가 거리에 영향을 크게 미치는 것을 방지

Unit 02 | KNN Hyperparameter

Distance Measure 1 : Euclidean Distance

- 가장 흔히 사용하는 거리 척도
- 두 관측치 사이의 최단거리(직선거리)를 의미

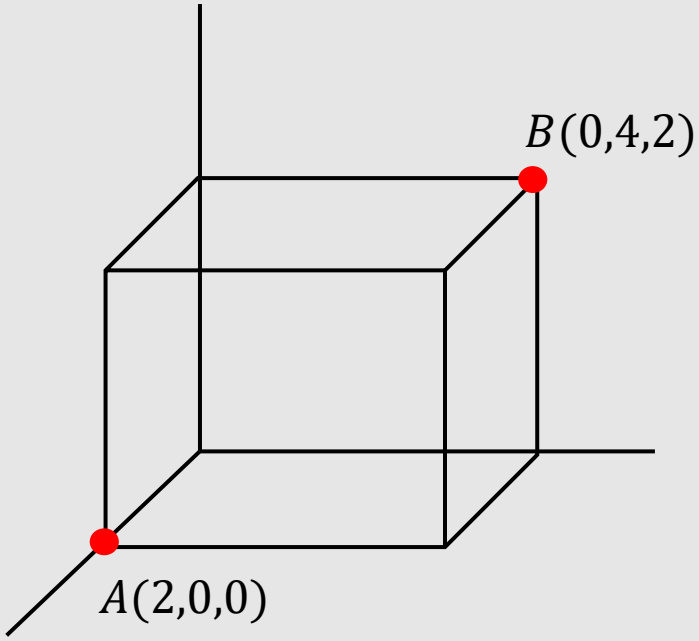
$$X = (x_1, x_2, \dots, x_n)$$
$$Y = (y_1, y_2, \dots, y_n)$$

$$Euclidean\ Distance = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Unit 02 | KNN Hyperparameter

Distance Measure 1 : Euclidean Distance

- 가장 흔히 사용하는 거리 척도
- 두 관측치 사이의 최단거리(직선거리)를 의미

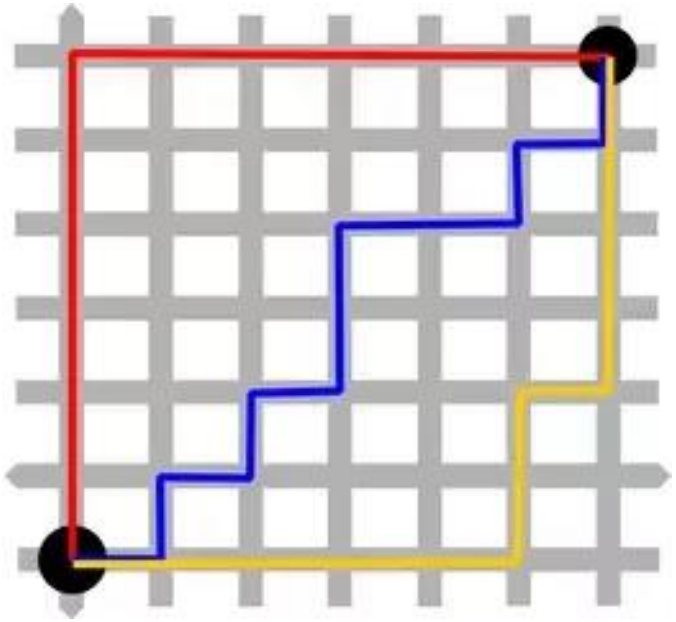


$$d_{(A,B)} = \sqrt{(0-2)^2 + (4-0)^2 + (2-0)^2} = 2\sqrt{6}$$

Unit 02 | KNN Hyperparameter

Distance Measure 2 : Manhattan Distance

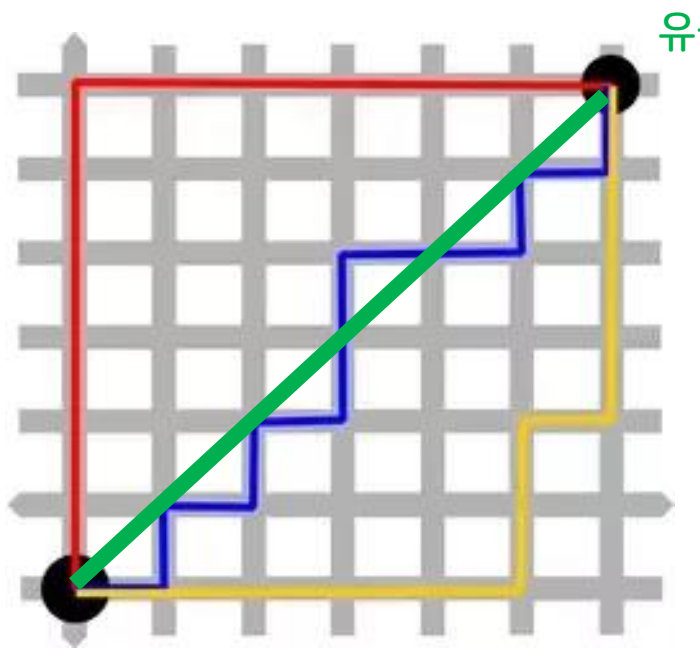
- 각 좌표축 방향으로만 이동할 경우에 계산되는 거리
- 항상 유클리드 거리보다 같거나 크다는 성질이 있음



Unit 02 | KNN Hyperparameter

Distance Measure 2 : Manhattan Distance

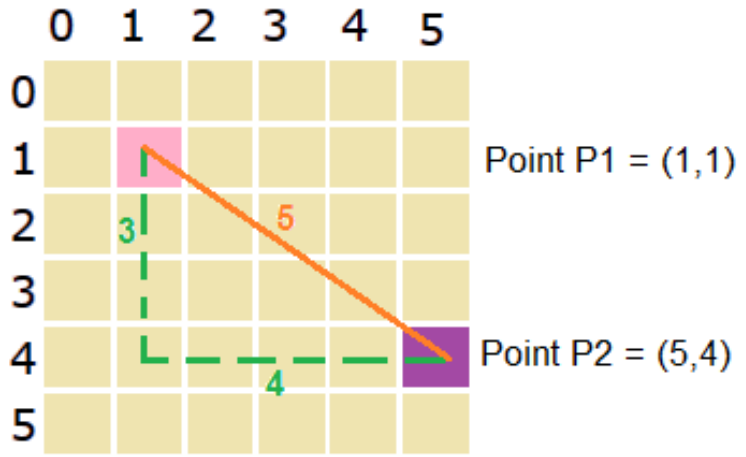
- 각 좌표축 방향으로만 이동할 경우에 계산되는 거리
- 항상 유클리드 거리보다 같거나 크다는 성질이 있음



Unit 02 | KNN Hyperparameter

Distance Measure 2 : Manhattan Distance

- 각 좌표축 방향으로만 이동할 경우에 계산되는 거리
- 항상 유클리드 거리보다 같거나 크다는 성질이 있음



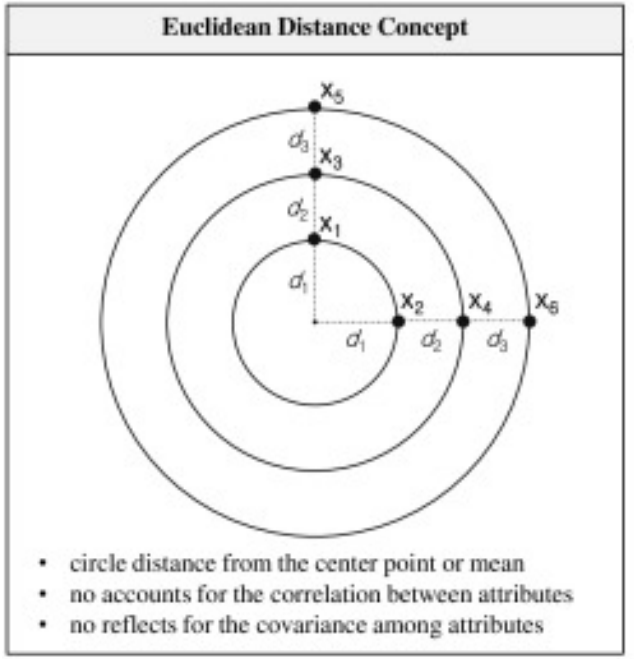
Euclidean distance = $\sqrt{(5-1)^2+(4-1)^2} = 5$

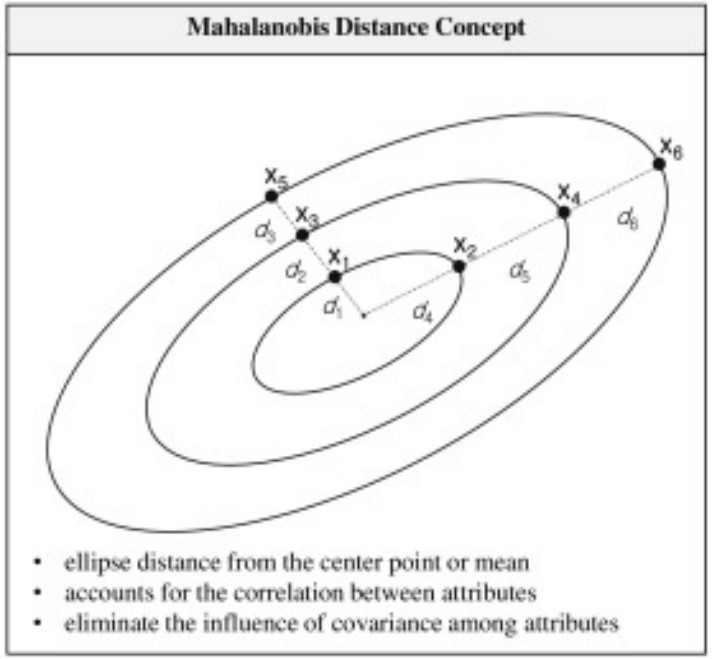
Manhattan distance = $|5-1| + |4-1| = 7$

Unit 02 | KNN Hyperparameter

Distance Measure 3 : Mahalanobis Distance

- 데이터가 평균과 표준편차를 고려했을 때 얼마나 중심에서 멀리 떨어져 있는지를 측정 (밀도를 고려한 거리 척도)
- 두 변수 사이에 상관관계가 거리에 영향을 미치기 때문에 변수들 간에 상관관계가 존재하는 경우 사용하면 유용





Contents

Unit 03 | 고려사항 & weighted KNN

1. 다수 클래스, 더 좋은 방법 ?
2. Distance 기반 알고리즘

Unit 03 | 고려사항 & weighted KNN

다수 클래스? ➔ Weighted KNN

Unit 03 | 고려사항 & weighted KNN

Weighted KNN

- 거리가 가까운(유사도가 높은) 이웃의 정보에 좀 더 가중치를 주는 방법

➔ 단순 평균, 다수결로 값을 정하지 않고 거리에 따라서 영향력을 달리주고 싶을 때 사용

$$\text{유사도} = \frac{1}{\text{거리}^2}$$

$$\text{가중치} = \frac{\text{유사도}}{\text{모든 유사도의 합}}$$

Unit 03 | 고려사항 & weighted KNN

Weighted KNN

- 거리가 가까운(유사도가 높은) 이웃의 정보에 좀 더 가중치를 주는 방법

➔ 단순 평균, 다수결로 값을 정하지 않고 거리에 따라서 영향력을 달리주고 싶을 때 사용

예측모델

$$\hat{y}_{new} = \frac{\sum_{i=1}^{k} w_i y_i}{\sum_{i=1}^{k} w_i} \quad where \;\; w_i = \frac{1}{d_{(new,x_i)}{}^2}$$

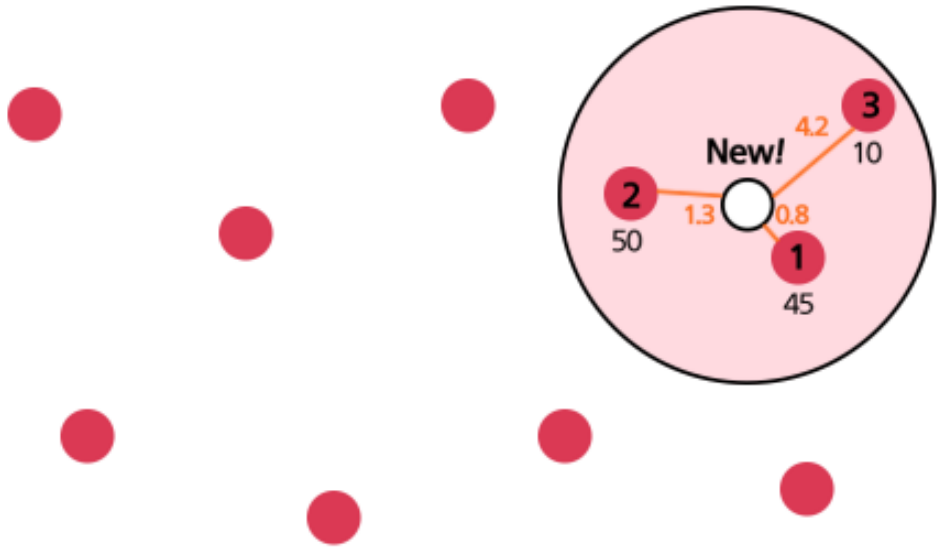
분류모델

$$\hat{c}_{new} = max_c \sum_{i=1}^{k} w_i I(w_i \in c) \quad where \;\; w_i = \frac{1}{d_{(new,x_i)}{}^2}$$

Unit 03 | 고려사항 & weighted KNN

Weighted KNN

- Weighted 3-NN 예측모델 예제



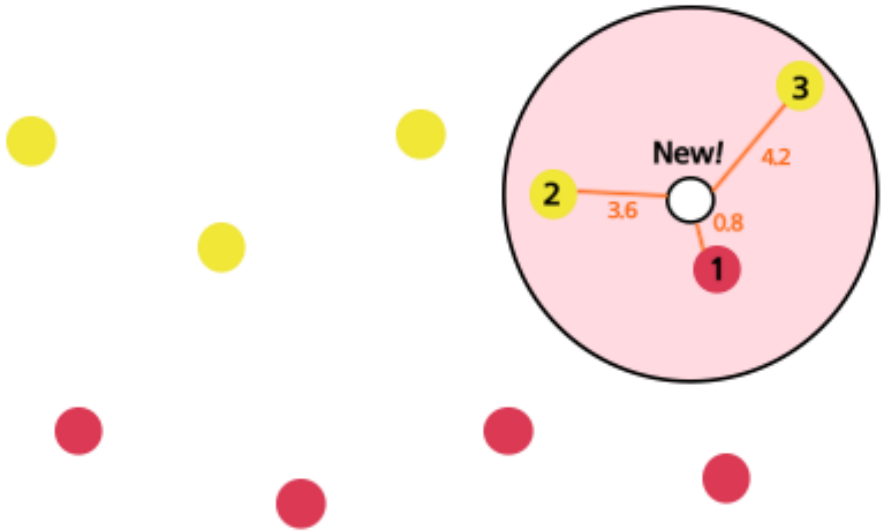
New : $\frac{10+50+45}{3} = 35$

New_weighted : $\frac{\left(\frac{1}{0.8^2} \times 45 + \frac{1}{1.3^2} \times 50 + \frac{1}{4.2^2} \times 10\right)}{\left(\frac{1}{0.8^2} + \frac{1}{1.3^2} + \frac{1}{4.2^2}\right)}$

Unit 03 | 고려사항 & weighted KNN

Weighted KNN

- Weighted 3-NN 분류모델 예제



New : Yellow

New_weighted : Red

Yellow = $\frac{1}{3.6^2} + \frac{1}{4.2^2} \cong 0.13$

Red = $\frac{1}{0.8^2} \cong 1.56$

Unit 03 | 고려사항 & weighted KNN

거리 기반 알고리즘?

- 변수들의 단위(Scale)에 민감 ➔ Feature Scaling
- Categorical은? ➔ One-hot Encoding

Unit 03 | 고려사항 & weighted KNN

Feature Scaling

Min-Max Normalization

- 데이터를 일반적으로 0~1 사이의 값으로 반환

$$x = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Standardization

- 표준화를 사용하여 데이터의 평균이 0, 표준편차가 1이 되도록 변환

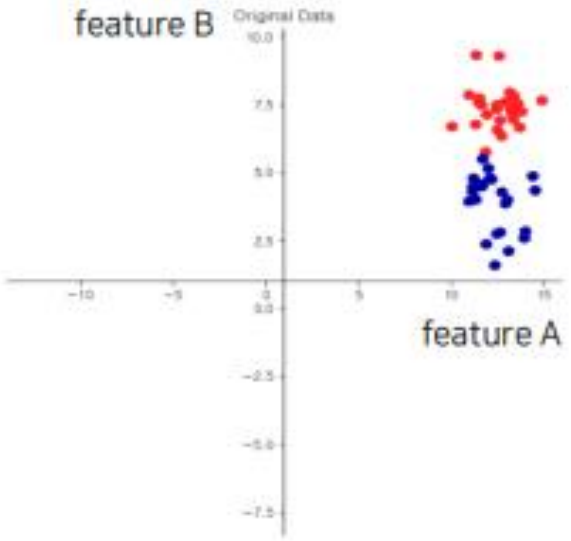
$$x = \frac{x - x_{mean}}{x_{std}}$$

- train 데이터와 test 데이터의 scale을 따로 조정하면 안됨
- train 데이터의 scale을 조정하고자 구한 정규화 parameter(최대최소, 평균, 표준편차 등) 을 기억하여 사용하여 test 데이터도 변환해야함

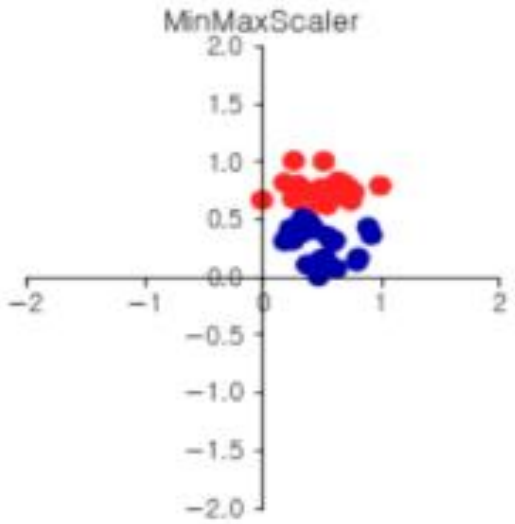
Unit 03 | 고려사항 & weighted KNN

Feature Scaling

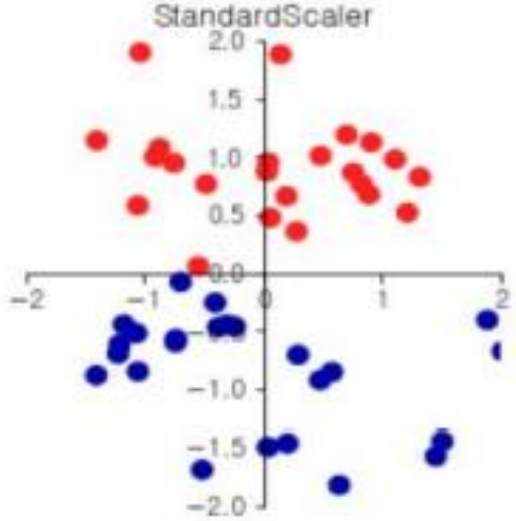
Original data



Min-Max Normalization



Standardization



Unit 03 | 고려사항 & weighted KNN

One-Hot Encoding

- Categorical 값을 feature로 만든 후 1 또는 0으로 지정하는 방법
- 1개만 Hot(1)이고 나머지는 Cold(0)
- KNN은 거리기반 ➔ input에 numerical형이 와야함

color
Red
Green
Blue
Red

color_Red	color_Green	color_Blue
1	0	0
0	1	0
0	0	1
1	0	0

Unit 03 | KNN 고려사항

One-Hot Encoding

- Categorical 값을 feature로 만든 후 1 또는 0으로 지정하는 방법
- 1개만 Hot(1)이고 나머지는 Cold(0)
- KNN은 거리기반 ➔ input에 numerical형이 와야함

color
Red
Green
Blue
Red

color
1
2
3
1

Red + Green = Blue라는

잘못된 관계를 형성할 수 있음 !!

Contents

Unit 04 | KNN 실습 – iris data

기본 KNN : 파라미터튜닝(GridSearchCV) – fit - predict

```
[ ] #knn 시작
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.model_selection import GridSearchCV

    knn = KNeighborsClassifier()
```

```
[ ] #그리드서치를 위한 파라미터 설정
    params_1 = {
        "n_neighbors": [i for i in range(1, 20, 2)],
        "p": [1, 2], #1은 맨하탄, 2는 유클리드
        "weights": ['uniform']
    }
```

```
[ ] grid_cv.best_params_

    {'n_neighbors': 7, 'p': 1, 'weights': 'uniform'}
```

```
[ ] knn_1 = KNeighborsClassifier(n_neighbors = 7, p = 1, weights = 'uniform')
    knn_1.fit(train_data, train_target)

    KNeighborsClassifier(n_neighbors=7, p=1)
```

```
[ ] #예측 진행
    test_pred = knn_1.predict(test_data)
```

```
[ ] from sklearn.metrics import accuracy_score
    from sklearn.metrics import confusion_matrix

    print("test_data accuracy : ", accuracy_score(test_target, test_pred))
    print("* confusion matrix *")

    cm = confusion_matrix(test_target, test_pred)
    cm_ = pd.DataFrame(cm).rename(index={0:'target(0)', 1:'target(1)', 2:'target(2)'}, columns={0:'pred(0)', 1:'pred(1)', 2:'pred(2)'})
    print(cm_)
```

```
test_data accuracy :  0.9777777777777777
* confusion matrix *
           pred(0)  pred(1)  pred(2)
target(0)       17        0        0
target(1)        0       17        0
target(2)        0        1       10
```

Unit 04 | KNN 실습 – iris data

weighted KNN : 파라미터튜닝(GridSearchCV) – fit - predict

```
[ ] #가중치를 부여하는 경우 파라미터 조정 (그리드서치)
    params_2 = {
        "n_neighbors": [i for i in range(1, 20, 2)],
        "p": [1, 2]  #1은 맨하탄, 2는 유클리드
        "weights": ['distance']
    }

[ ] grid_cv = GridSearchCV(knn, param_grid = params_2, cv = 3)

[ ] grid_cv.fit(train_data, train_target)
    grid_cv.best_params_

    {'n_neighbors': 7, 'p': 2, 'weights': 'distance'}

[ ] knn_2 = KNeighborsClassifier(n_neighbors = 7, p = 2, weights = 'distance')
    knn_2.fit(train_data, train_target)

    KNeighborsClassifier(n_neighbors=7, weights='distance')
```

```
[ ] knn_2 = KNeighborsClassifier(n_neighbors = 7, p = 2, weights = 'distance')
    knn_2.fit(train_data, train_target)

    KNeighborsClassifier(n_neighbors=7, weights='distance')

[ ] #예측 진행
    test_pred = knn_2.predict(test_data)

[ ] print("test_data accuracy : ", accuracy_score(test_target, test_pred))
    print("* confusion matrix *")

    cm = confusion_matrix(test_target, test_pred)
    cm_ = pd.DataFrame(cm).rename(index={0:'target(0)', 1:'target(1)', 2:'target(2)'}, columns={0:'pred(0)', 1:'pred(1)', 2:'pred(2)'})
    print(cm_)

    test_data accuracy :  1.0
    * confusion matrix *
              pred(0)  pred(1)  pred(2)
    target(0)      17        0        0
    target(1)       0       17        0
    target(2)       0        0       11
```

Unit 04 | KNN 실습 – iris data

KNN + scaling : scaling - 파라미터튜닝(GridSearchCV) – fit - predict

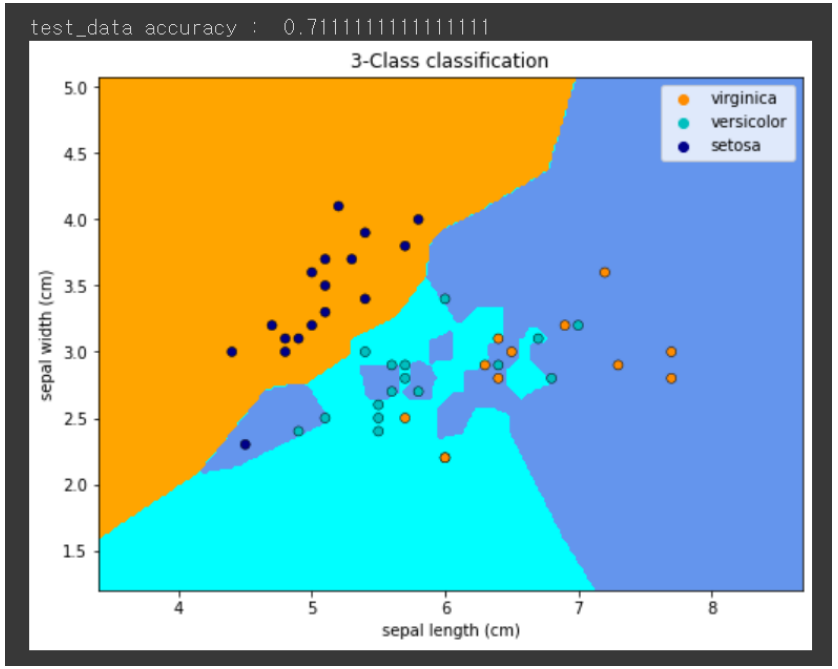
```
[ ] #변수 스케일링을 사용하는 경우
    from sklearn.preprocessing import StandardScaler
    scaler = StandardScaler()

[ ] #train과 test 모두 스케일링 진행
    scaler.fit(train_data)

    scaled_train_data = scaler.transform(train_data)
    scaled_test_data = scaler.transform(test_data)

[ ] params_3 = {
        "n_neighbors": [i for i in range(1, 20, 2)],
        "p": [1, 2], #1은 맨하탄, 2는 유클리드
        "weights": ['uniform','distance']
    }
```

```
[ ] knn_3 = KNeighborsClassifier(n_neighbors = 5, p = 2, weights = 'distance')
    knn_3.fit(scaled_train_data, train_target)

    KNeighborsClassifier(weights='distance')

[ ] #예측 진행
    test_pred = knn_3.predict(scaled_test_data)

[ ] print("test_data accuracy : ", accuracy_score(test_target, test_pred))
    print("* confusion matrix *")

    cm = confusion_matrix(test_target, test_pred)
    cm_ = pd.DataFrame(cm).rename(index={0:'target(0)', 1:'target(1)', 2:'target(2)'}, columns={0:'pred(0)', 1:'pred(1)', 2:'pred(2)'})
    print(cm_)
```

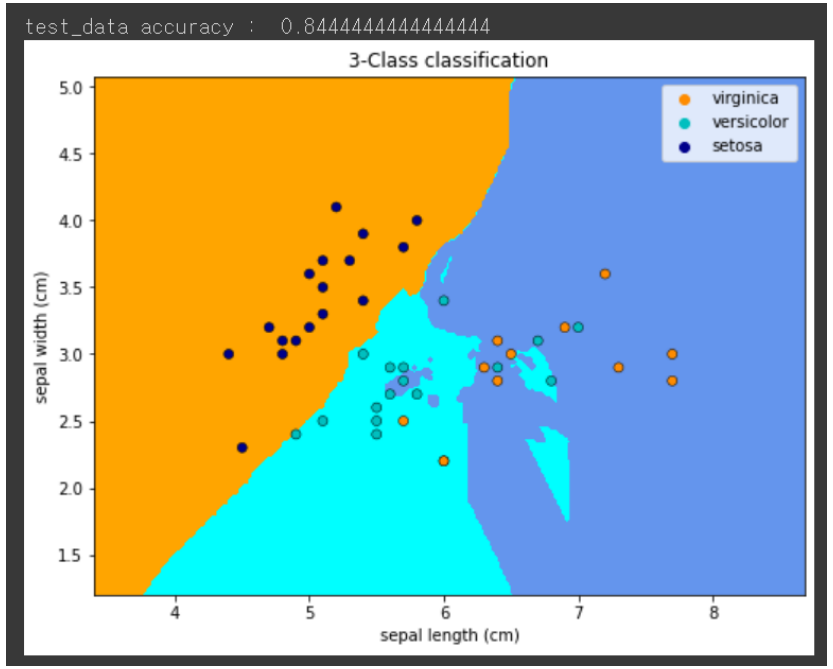
```
test_data accuracy :  0.9777777777777777
* confusion matrix *
          pred(0)  pred(1)  pred(2)
target(0)       17        0        0
target(1)        0       17        0
target(2)        0        1       10
```

Unit 04 | KNN 실습 - iris data

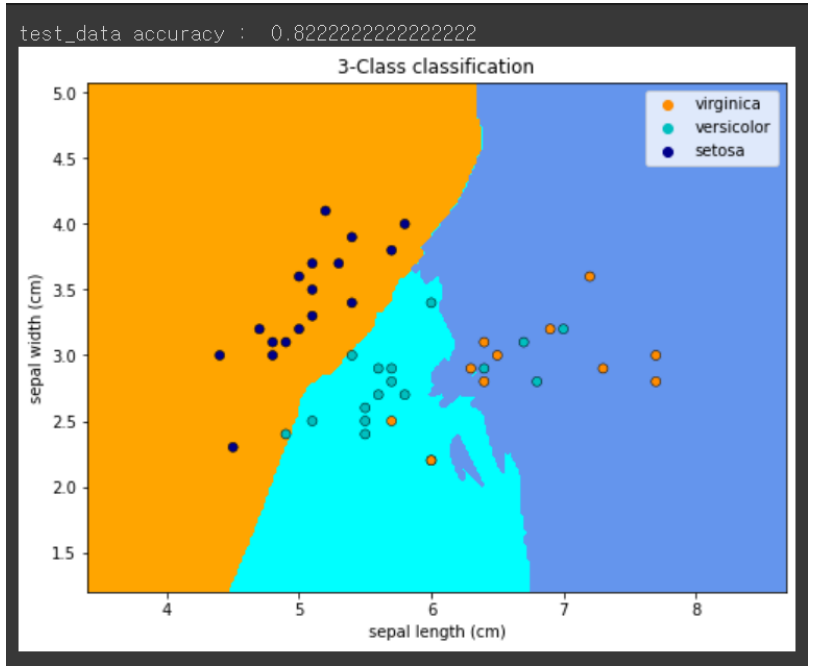
K값 조정에 따른 결정 경계



K=1



K=7



K=19

Contents

Unit 05 | KNN 장단점

Good

1. 이해하기 매우 쉬운 모델이다.
2. 데이터 내 노이즈 영향을 크게 받지 않으며, 특히 Mahalanobis distance와 같이 데이터의 분산을 고려할 경우 강건하다.
3. 학습데이터의 수가 많을 경우 효과적이다.

Unit 05 | KNN 장단점

Bad

1. 하나의 데이터를 예측할 때마다 전체 데이터와의
   거리를 계산하기 때문에 연산속도가 다른 알고리즘에 비해 느리다
2. 어떤 거리 척도가 분석에 적합한지 불분명하기에 데이터의 특성에
   맞는 거리측도를 임의로 선정해야한다.

과제

KNN 구현해보기

주어진 데이터로 주석과 함께 자유롭게 과제를 진행해주세요 ☺

- Preprocessing / EDA
- KNN & Hyperparameter tuning
- Evaluation

데이터 : https://www.kaggle.com/llopesolivei/blackfriday

Reference

참고자료

- 투빅스 17기 이지수님 강의자료
- 투빅스 16기 박한나님 강의자료
- 투빅스 15기 김현지님 강의자료
- 투빅스 14기 김민경님 강의자료
- https://velog.io/@cleansky/K-Nearest-Neighbor-KNN-%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98
- 고려대학교 김성범 교수님의 핵심 머신러닝 K-nearest neighbors & Distance Measures 강의
  https://www.youtube.com/watch?v=W-DNu8nardo

들어주셔서 감사합니다.