고려대학교 의료정보학과

KU MEDICINE

IMDL

Intelligent Medical Data Lab

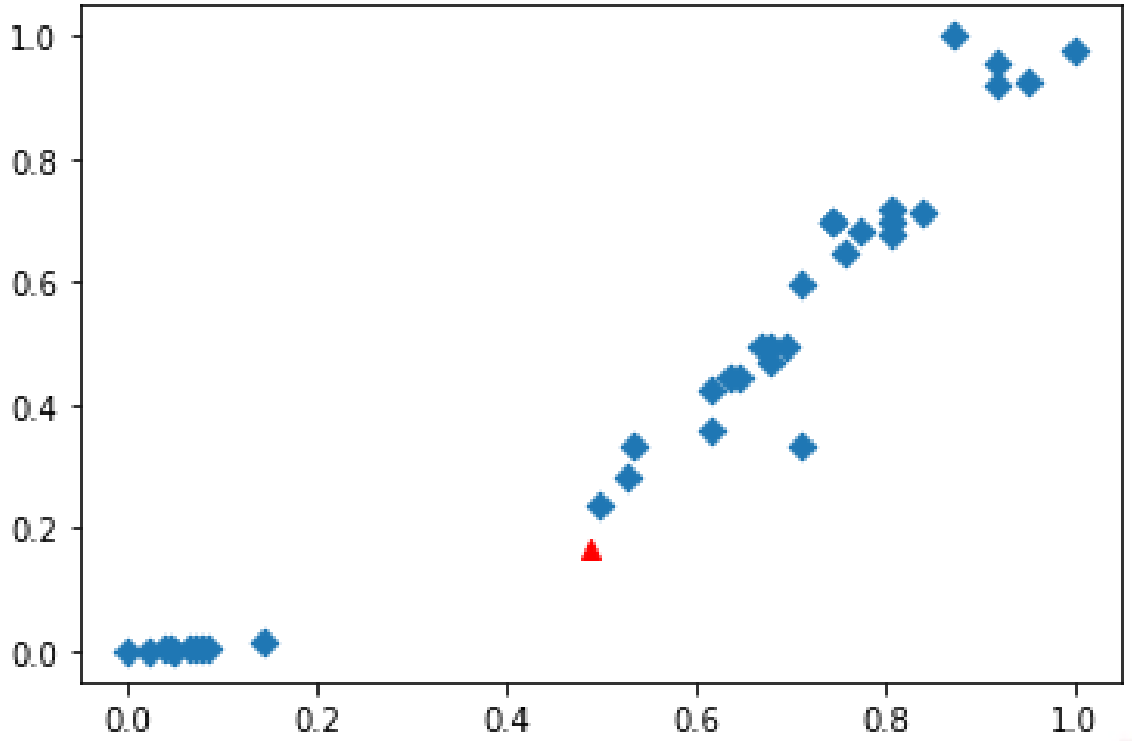
의료인공지능

머신러닝 - KNN

고려대학교 의료빅데이터연구소
채민수(minsuchae@korea.ac.kr)

1. KNN

- K-Nearest neighbors
  - 훈련 데이터셋에서 인접한 항목들을 찾아 인접한 항목 중 가장 많은 항목으로 결과를 반환
  - 사이킷런에서 제공하는 클래스는
    - KNeighborsClassifier
    - KNeighborsRegressor



1. KNN

- KNeighborsClassifier
  - 클래스 원형
    - class sklearn.neighbors.KNeighborsClassifier(n_neighbors=5, *, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None)
  - n_neighbors : 예측 시 사용될 추출할 이웃한 항목 수(홀수 사용)
  - weights
    - uniform : 이웃한 항목을 선택할 때 모든 좌표가 동일한 가중치로 설정
    - distance : 거리에 따라 가중치를 다르게 설정

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

1. KNN

- KNeighborsClassifier
  - Minkowski Distance

$$\left(\sum_{i=1}^{n}|x_i - x_i|^p\right)^{1/p}$$

  - p
    - 1 : Manhattan Distance

$$\left(\sum_{i=1}^{n}|x_i - x_i|^1\right)^{1/1} = \sum_{i=1}^{n}|x_i - y_i|$$

    - 2 : Euclidean Distance

$$\left(\sum_{i=1}^{n}|x_i - x_i|^2\right)^{1/2} = \sqrt{\sum_{i=1}^{n}|x_i - y_i|^2}$$

1. KNN

- KNeighborsClassifier
  - metric

metric	Function
‘cityblock’	metrics.pairwise.manhattan_distances
‘euclidean’	metrics.pairwise.euclidean_distances
‘haversine’	metrics.pairwise.haversine_distances
‘l1’	metrics.pairwise.manhattan_distances
‘l2’	metrics.pairwise.euclidean_distances
‘manhattan’	metrics.pairwise.manhattan_distances
‘cosine’	metrics.pairwise.cosine_distances
‘nan_euclidean’	metrics.pairwise.nan_euclidean_distances

  - n_jobs
    - ➢ 작업할 CPU의 수

1. KNN

- KNeighborsRegressor
  - 클래스 원형
    - class sklearn.neighbors.KNeighborsRegressor(n_neighbors=5, *, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None)
  - 다른 회귀 알고리즘과 달리 이웃한 항목들을 이용하여 보정하여 추론

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html

2. KNN 실습 - 물고기 분류

◦ 데이터셋 로드

```
import pandas as pd

df = pd.read_csv('fish.csv')
df.head()
```

[실행결과]

	Species	Weight	Length
0	Bream	242.0	25.4
1	Bream	290.0	26.3
2	Bream	340.0	26.5
3	Bream	363.0	29.0
4	Bream	430.0	29.0

◦ 물고기 종류 확인

```
df['Species'].value_counts()
```

[실행결과]

```
Bream     35
Smelt     14
Name: Species, dtype: int64
```

2. KNN 실습 - 물고기 분류

◦ 도미와 빙어 숫자로 라벨링 수행

```
df.loc[df['Species']=='Bream','Species'] = 0
df.loc[df['Species']=='Smelt','Species'] = 1
df['Species'] = df['Species'].astype('int32')
df.head()
```

[실행결과]

	Species	Weight	Length
0	0	242.0	25.4
1	0	290.0	26.3
2	0	340.0	26.5
3	0	363.0	29.0
4	0	430.0	29.0

◦ 학습 특징과 예측 항목 설정

```
features = df[['Weight','Length']]
outcome = df['Species']
```

2. KNN 실습 - 물고기 분류

- 도미와 빙어 숫자로 라벨링 수행

```
from sklearn.neighbors import
KNeighborsClassifier

knn = KNeighborsClassifier()
knn.fit(features, outcome)
```

- 학습 특징과 예측 항목 설정

```
print(knn.predict([[240,25]]))
```

[실행결과]
0

3. 훈련 데이터와 테스트 데이터

◦ 전체 데이터를 학습할 경우의 문제점

- 학습된 결과를 바탕으로 하였으므로 성능 평가 시 높은 성능이 나옴

- 이미 학습된 것이기 때문에 정확한 판단 기준 부재

- 이를 해결하기 위해 학습되지 않은 데이터로 평가를 해야 함

3. 훈련 데이터와 테스트 데이터

- 훈련 데이터와 테스트 데이터를 나누기1 - 임의의 인덱스

```
train_features = features[:35]
test_features = features[35:]
train_target = outcome[:35]
test_target = outcome[35:]
```

3. 훈련 데이터와 테스트 데이터

◦ 훈련 데이터와 테스트 데이터를 나누기1 - 임의의 인덱스

```
from sklearn.neighbors import
KNeighborsClassifier

knn = KNeighborsClassifier()
knn.fit(train_features, train_target)
print(knn.predict(test_features))
print(test_target.values.reshape(-1))
```

[실행결과]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]

3. 훈련 데이터와 테스트 데이터

◦ 훈련 데이터와 테스트 데이터를 나누는 기준2

- 사이킷런에서 제공하는 train_test_split 함수 사용

```
from sklearn.model_selection import train_test_split

train_features, test_featues, train_target, test_target =
train_test_split(features,outcome,random_state=42)
train_features = train_features.values
test_features = test_features.values
```

3. 훈련 데이터와 테스트 데이터

◦ 훈련 데이터와 테스트 데이터를 나누는 기준2

- 사이킷런에서 제공하는 train_test_split 함수 사용

```
train_target.value_counts()
```

```
[실행결과]
0    25
1    11
Name: Species, dtype: int64
```

```
test_target.value_counts()
```

```
[실행결과]
0    10
1     3
Name: Species, dtype: int64
```

3. 훈련 데이터와 테스트 데이터

◦ 훈련 데이터와 테스트 데이터를 나누는 기준2

- stratify 파라미터를 설정하여 데이터 비율을 유지하며 나누기

```
from sklearn.model_selection import train_test_split

train_features, test_features, train_target, test_target =
train_test_split(features,outcome,stratify=outcome,random_state=42)
train_features = train_features.values
test_features = test_features.values
```

3. 훈련 데이터와 테스트 데이터

◦ 훈련 데이터와 테스트 데이터를 나누는 기준2

- 사이킷런에서 제공하는 train_test_split 함수 사용

```
train_target.value_counts()
```

```
[실행결과]
0    26
1    10
Name: Species, dtype: int64
```

```
test_target.value_counts()
```

```
[실행결과]
0    9
1    4
Name: Species, dtype: int64
```

3. 훈련 데이터와 테스트 데이터

- 훈련데이터와 테스트 데이터를 나눈 결과를 바탕으로 수행

```
from sklearn.neighbors import
KNeighborsClassifier

knn = KNeighborsClassifier()
knn.fit(train_features, train_target)
y_pred = knn.predict(test_features)
```

3. 훈련 데이터와 테스트 데이터

- 학습 결과에 대한 성능 평가

```
print(y_pred)
```

```
[실행결과]
[0 0 0 1 1 0 1 0 1 0 0 0 0]
```

```
print(test_target.values.reshape(-1))
```

```
[실행결과]
[0 0 0 1 1 0 1 0 1 0 0 0 0]
```

```
from sklearn.metrics import accuracy_score

print('Accuracy :',accuracy_score(test_target,y_pred))
```

```
[실행결과]
Accuracy : 1.0
```

4. 데이터 스케일

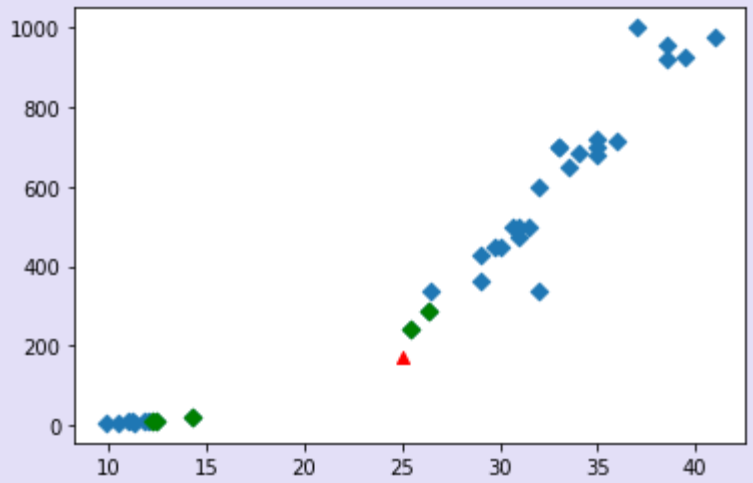
◦ 이상한 데이터?

```
print(knn.predict([[170,25]]))
```

[실행결과]
[1]

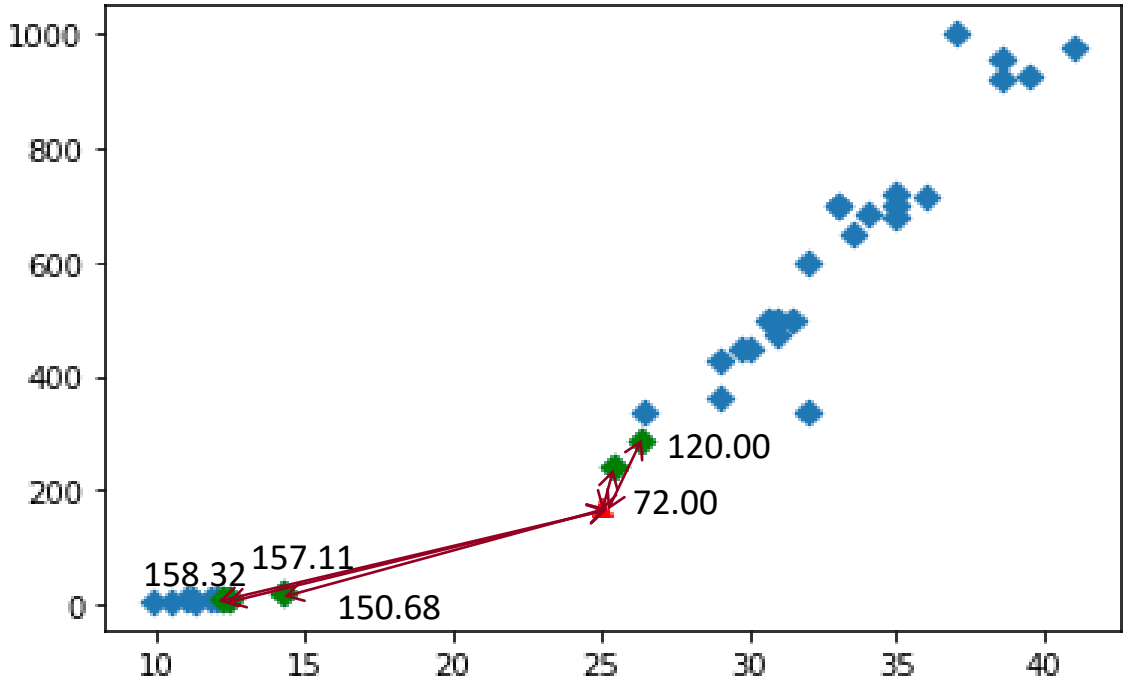
```
distances, indexes = knn.kneighbors([[170,25]])
indexes = indexes.reshape(-1)
%matplotlib inline
import matplotlib.pyplot as plt

plt.scatter(train_features[:,1],train_features[:,0],marker='D')

plt.scatter([25],[170], marker='^', color='red')
for idx in indexes :
    plt.scatter(train_features[idx][1],train_features[idx][0],
marker='D', color='green')
plt.show()
```

[실행결과]



4. 데이터 스케일

- 이상한 데이터?
  - 무게에 대한 단위 : g
  - 길이에 대한 단위 : cm
  - 무게와 길이에 대한 범위가 달라서 문제가 발생



4. 데이터 스케일

◦ 해결 방법 : 값의 범주를 줄임

- MinMaxScaler

$$z_i = \frac{x_i - mix(x_i)}{max(x_i) - mix(x_i)}$$

sklearn.preprocessing.MinMaxScaler

class sklearn.preprocessing.**MinMaxScaler**(*feature_range=(0, 1), *, copy=True, clip=False*) [source]

Transform features by scaling each feature to a given range.

This estimator scales and translates each feature individually such that it is in the given range on the training set, e.g. between zero and one.

The transformation is given by:

```
X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
X_scaled = X_std * (max - min) + min
```

where min, max = feature_range.

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

- StandardScaler

$$z_i = \frac{x_i - mean(x_i)}{standard\ deviation(x_i)}$$

sklearn.preprocessing.StandardScaler

class sklearn.preprocessing.**StandardScaler**(**, copy=True, with_mean=True, with_std=True*) [source]

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

The standard score of a sample `x` is calculated as:

z = (x - u) / s

where `u` is the mean of the training samples or zero if `with_mean=False`, and `s` is the standard deviation of the training samples or one if `with_std=False`.

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

4. 데이터 스케일

◦ 데이터 스케일하여 학습

```
from sklearn.preprocessing import MinMaxScaler

feature_scaler = MinMaxScaler()
train_features_scaled =
feature_scaler.fit_transform(train_features)
test_features_scaled =
feature_scaler.transform(test_features)

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()
knn.fit(train_features_scaled, train_target)
```

4. 데이터 스케일

◦ 데이터 스케일하여 학습 결과 확인 및 성능 평가

```
y_pred = knn.predict(test_features_scaled)
print(y_pred)
print(test_target.values.reshape(-1))
from sklearn.metrics import accuracy_score

print('Accuracy :',accuracy_score(test_target,y_pred))
```

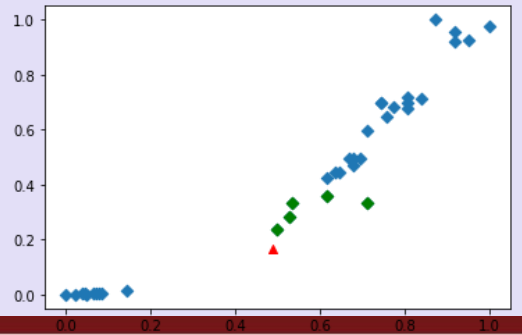
```
[실행결과]
[0 0 0 1 1 0 1 0 1 0 0 0 0]
[0 0 0 1 1 0 1 0 1 0 0 0 0]
Accuracy : 1.0
```

4. 데이터 스케일

◦ 시각화하여 확인

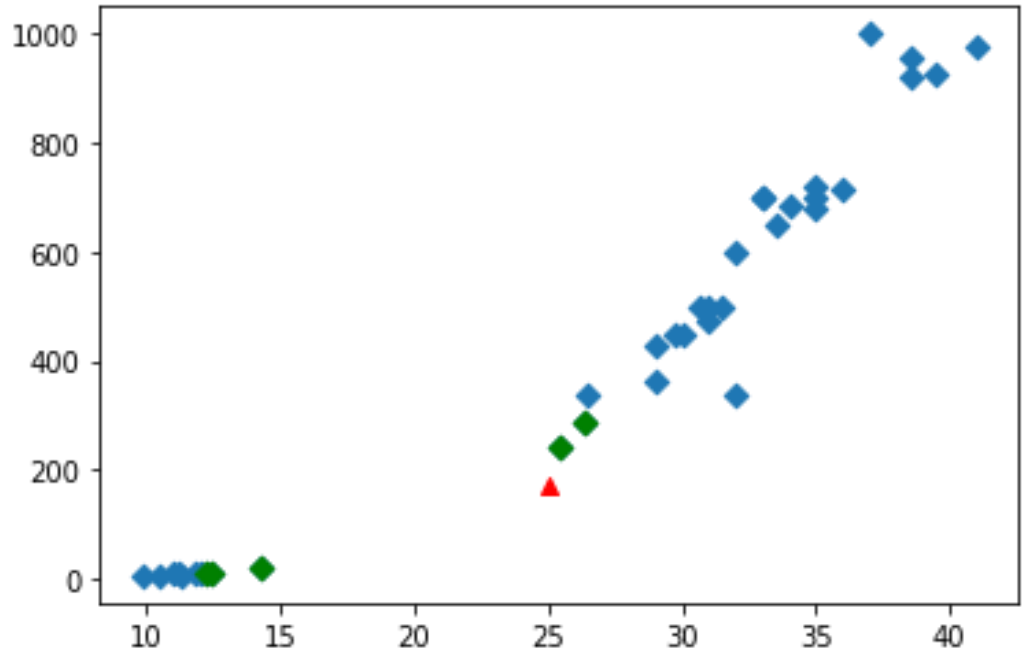
```
outlier = feature_scaler.transform([[170,25]])[0]
distances, indexes = knn.kneighbors([[outlier[0],outlier[1]]])
%matplotlib inline
import matplotlib.pyplot as plt

plt.scatter(train_features_scaled[:,1],train_features_scaled[:,0],marker='D')
outlier = feature_scaler.transform([[170,25]])[0]
plt.scatter([outlier[1]],[outlier[0]], marker='^', color='red')
for idx in indexes :
    plt.scatter(train_features_scaled[idx][1],train_features_scaled[idx][0], marker='D', color='green')
plt.show()
```

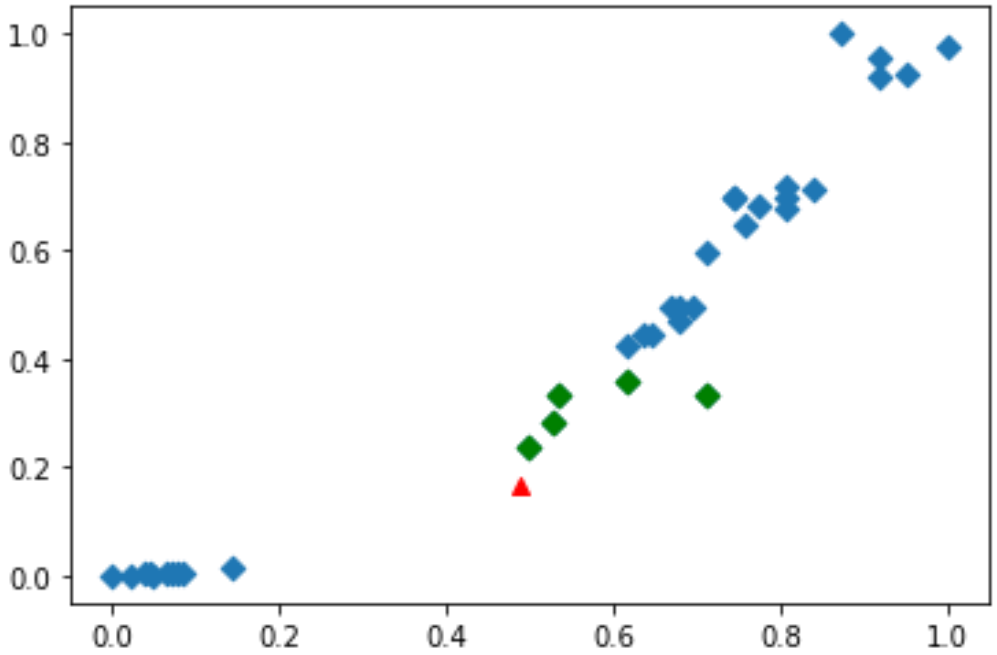
[실행결과]



4. 데이터 스케일

◦ 결과 확인





5. KNN 실습 - 시험 점수 예측

◦ 데이터셋 로드

```
import pandas as pd

df = pd.read_csv('exams.csv')
df.head()
```

[실행결과]

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	male	group A	high school	standard	completed	67	67	63
1	female	group D	some high school	free/reduced	none	40	59	55
2	male	group E	some college	free/reduced	none	59	60	50
3	male	group B	high school	standard	none	77	78	68
4	male	group E	associate's degree	standard	completed	78	73	68

5. KNN 실습 - 시험 점수 예측

- 문자열 데이터 변환

```
df.loc[df['gender']=='male','gender'] = 0
df.loc[df['gender']=='female','gender'] = 1
df['gender']=df['gender'].astype('int32')

df.loc[df['lunch']=='standard','lunch'] = 0
df.loc[df['lunch']=='free/reduced','lunch'] = 1
df['lunch']=df['lunch'].astype('int32')

df.loc[df['test preparation course']=='completed','test preparation course'] = 0
df.loc[df['test preparation course']=='none','test preparation course'] = 1
df['test preparation course']=df['test preparation course'].astype('int32')
```

5. KNN 실습 - 시험 점수 예측

- 범주형 데이터 원 핫 인코딩 수행 및 확인

```
df = pd.get_dummies(df)
df.head()
```

[실행결과]

	gender	lunch	test preparation course	math score	reading score	writing score	race/ethnicity_group A	race/ethnicity_group B	race/ethnicity_group C	race/ethnicity_group D	race/ethnicity_gro
0	0	0	0	67	67	63	1	0	0	0	
1	1	1	1	40	59	55	0	0	0	1	
2	0	1	1	59	60	50	0	0	0	0	
3	0	0	1	77	78	68	0	1	0	0	
4	0	0	0	78	73	68	0	0	0	0	

- 학습 특징과 예측 항목 설정

```
features = df[df.keys().drop(['math score','reading score','writing score'])].values
outcome = df[['math score','reading score','writing score']].values
```

5. KNN 실습 - 시험 점수 예측

◦ 훈련 데이터와 테스트 데이터 나누기

```
from sklearn.model_selection import train_test_split

train_features, test_features, train_target, test_target =
train_test_split(features,outcome,random_state=42)
```

◦ KNN 회귀 학습

```
from sklearn.neighbors import KNeighborsRegressor

knn = KNeighborsRegressor()
knn.fit(train_features, train_target)
y_pred = knn.predict(test_features)
```

5. KNN 실습 - 시험 점수 예측

◦ 성능 평가

```
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np
array_str = ["Math","Reading","Writing"]

for i in range(y_pred.shape[1]):
    print(array_str[i],"score","MAE :",mean_absolute_error(test_target[:,i], y_pred[:,i]))
    print(array_str[i],"score","RMSE :",np.sqrt(mean_squared_error(test_target[:,i], y_pred[:,i])))
```

[실행결과]
Math score MAE : 11.2672
Math score RMSE : 13.798202781521947
Reading score MAE : 11.640799999999999
Reading score RMSE : 14.18954544726504
Writing score MAE : 11.5968
Writing score RMSE : 14.26401346045355

5. KNN 실습 - 시험 점수 예측

- 데이터 스케일링 수행

```
from sklearn.preprocessing import MinMaxScaler

feature_scaler = MinMaxScaler()
train_features_scaled = feature_scaler.fit_transform(train_features)
test_features_scaled = feature_scaler.transform(test_features)
```

- 데이터 스케일링된 값을 이용하여 KNN 회귀 학습

```
from sklearn.neighbors import KNeighborsRegressor

from sklearn.neighbors import KNeighborsRegressor

knn = KNeighborsRegressor()
knn.fit(train_features_scaled, train_target)
y_pred = knn.predict(test_features)
```

5. KNN 실습 - 시험 점수 예측

◦ 성능 평가

```
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np
array_str = ["Math","Reading","Writing"]

for i in range(y_pred.shape[1]):
    print(array_str[i],"score","MAE :",mean_absolute_error(test_target[:,i], y_pred[:,i]))
    print(array_str[i],"score","RMSE :",np.sqrt(mean_squared_error(test_target[:,i], y_pred[:,i])))
```

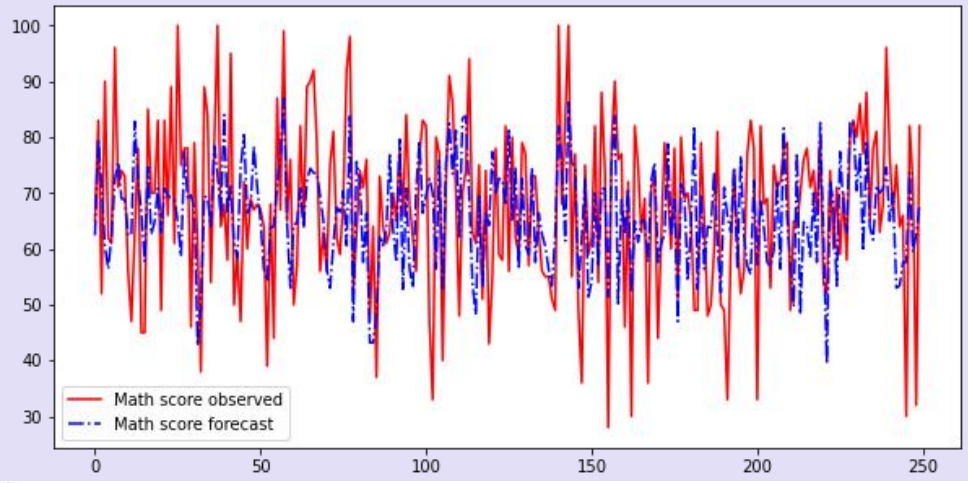
[실행결과]
Math score MAE : 11.2672
Math score RMSE : 13.798202781521947
Reading score MAE : 11.640799999999999
Reading score RMSE : 14.18954544726504
Writing score MAE : 11.5968
Writing score RMSE : 14.26401346045355

5. KNN 실습 - 시험 점수 예측

◦ 시각화를 통한 성능 확인 - 수학 점수

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.figure(figsize=(10,5))
plt.plot(test_target[:,0],linestyle='-',color='red',label='Math score observed')
plt.plot(y_pred[:,0],linestyle='-.',color='blue',label='Math score forecast')
plt.legend()
plt.show()
```

[실행결과]

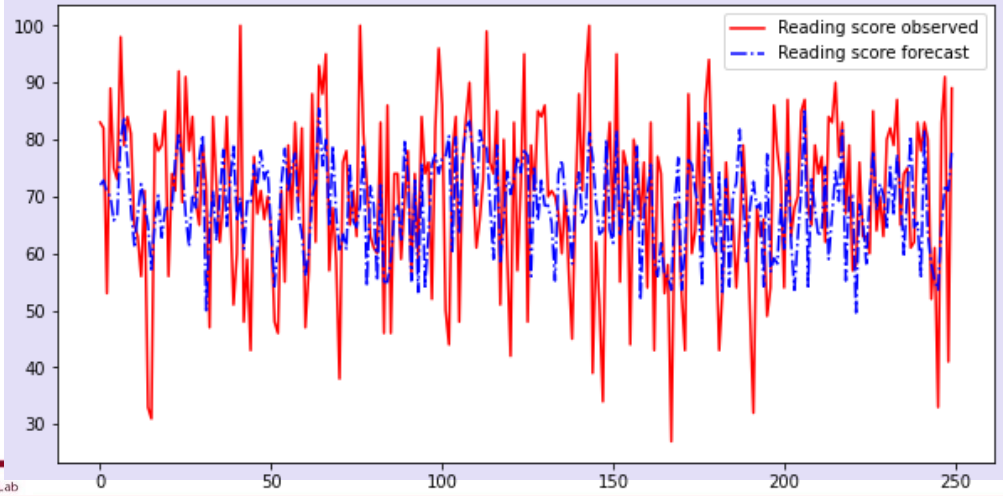


5. KNN 실습 - 시험 점수 예측

◦ 시각화를 통한 성능 확인 - 읽기 점수

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.figure(figsize=(10,5))
plt.plot(test_target[:,1],linestyle='-',color='red',label='Reading score observed')
plt.plot(y_pred[:,1],linestyle='-.',color='blue',label='Reading score forecast')
plt.legend()
plt.show()
```

[실행결과]

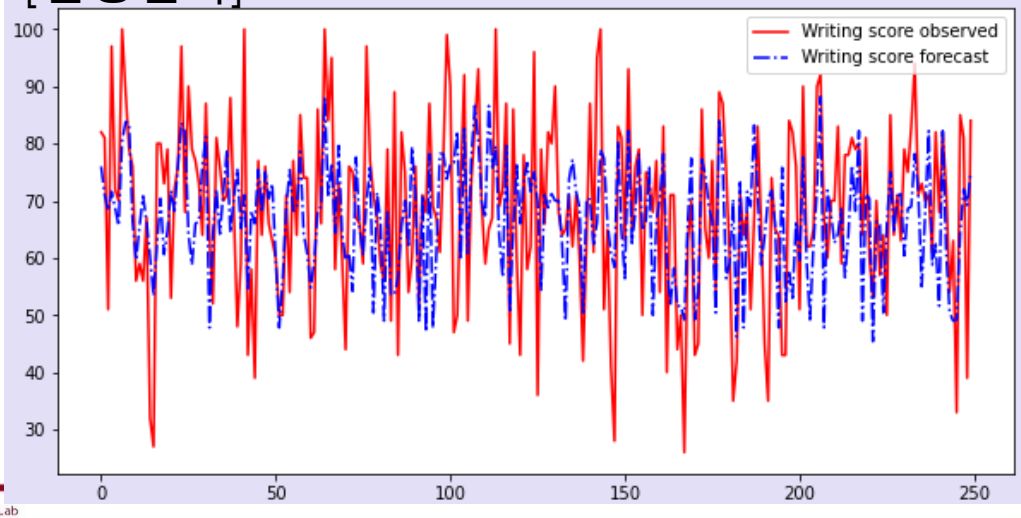


5. KNN 실습 - 시험 점수 예측

◦ 시각화를 통한 성능 확인 - 쓰기 점수

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.figure(figsize=(10,5))
plt.plot(test_target[:,1],linestyle='-',color='red',label='Reading score observed')
plt.plot(y_pred[:,1],linestyle='-.',color='blue',label='Reading score forecast')
plt.legend()
plt.show()
```

[실행결과]



6. KNN 실습 - 피마 인디언 당뇨병 예측

◦ 데이터셋 로드

```
import pandas as pd

df = pd.read_csv('Pima_Indians_Diabetes_Database.csv')
df.head()
```

[실행결과]

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

6. KNN 실습 - 피마 인디언 당뇨병 예측

◦ 당뇨병 환자 수 확인

```
df['Outcome'].value_counts()
```

[실행결과]

```
0    500
1    268
Name: Outcome, dtype: int64
```

◦ 데이터 이상치 확인

```
df.describe()
```

[실행결과]

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

6. KNN 실습 - 피마 인디언 당뇨병 예측

- 이상치 데이터 결측치로 치환

```
keys = ["Glucose","BloodPressure","SkinThickness","Insulin","BMI"]

for key in keys:
    df.loc[df[key] <= df[key].quantile(0.10), key]=None
    df.loc[df[key] >= df[key].quantile(0.90), key]=None
```

- 결측치 확인

```
df.isnull().sum()
```

```
[실행결과]
Pregnancies                 0
Glucose                   148
BloodPressure             171
SkinThickness             286
Insulin                   414
BMI                       150
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

6. KNN 실습 - 피마 인디언 당뇨병 예측

- 이상치 데이터 결측치로 치환

```
keys = ["Glucose","BloodPressure","SkinThickness","Insulin","BMI"]

for key in keys:
    df.loc[df[key] <= df[key].quantile(0.10), key]=None
    df.loc[df[key] >= df[key].quantile(0.90), key]=None
```

- 결측치 확인

```
df.isnull().sum()
```

```
[실행결과]
Pregnancies                 0
Glucose                   148
BloodPressure             171
SkinThickness             286
Insulin                   414
BMI                       150
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

6. KNN 실습 - 피마 인디언 당뇨병 예측

- 훈련 데이터와 테스트 데이터로 나누기

```
features = df[df.keys().drop('Outcome')]
outcome = df['Outcome']
from sklearn.model_selection import train_test_split

train_features, test_features, train_target, test_target =
train_test_split(features,outcome,stratify=outcome,random_state=42)
```

- 훈련 데이터를 이용한 데이터 결측치 보정

```
keys = ["Glucose","BloodPressure","SkinThickness","Insulin","BMI"]

for key in keys:
    tmp = train_features[key].median()
    train_features[key].fillna(tmp,inplace=True)
    test_features[key].fillna(tmp,inplace=True)
```

6. KNN 실습 - 피마 인디언 당뇨병 예측

◦ 데이터 보정 후 데이터 범주 확인

df.describe()

[실행결과]

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	620.000000	597.000000	482.000000	354.000000	618.000000	768.000000	768.000000	768.000000
mean	3.845052	120.103226	71.396985	26.921162	124.098870	32.081392	0.471876	33.240885	0.348958
std	3.369578	21.424225	7.989813	8.332364	63.832987	4.671923	0.331329	11.760232	0.476951
min	0.000000	86.000000	55.000000	7.000000	14.000000	23.700000	0.078000	21.000000	0.000000
25%	1.000000	102.000000	64.000000	20.000000	74.000000	28.325000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	28.000000	115.000000	32.250000	0.372500	29.000000	0.000000
75%	6.000000	136.000000	78.000000	33.000000	168.000000	35.500000	0.626250	41.000000	1.000000
max	17.000000	170.000000	86.000000	41.000000	291.000000	42.200000	2.420000	81.000000	1.000000

6. KNN 실습 - 피마 인디언 당뇨병 예측

- 데이터 스케일 수행

```
from sklearn.preprocessing import MinMaxScaler

feature_scaler = MinMaxScaler()
train_features_scaled = feature_scaler.fit_transform(train_features)
test_features_scaled = feature_scaler.transform(test_features)
```

- KNN을 이용한 학습

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()
knn.fit(train_features_scaled, train_target)
y_pred = knn.predict(test_features_scaled)
```

6. KNN 실습 - 피마 인디언 당뇨병 예측

◦ 성능 평가

```
from sklearn.metrics import accuracy_score

print('Accuracy :',accuracy_score(test_target,y_pred))
```

[실행결과]
Accuracy : 0.6666666666666666

◦ Precision, Recall, F1 Score

		Prediction	
		Disease	Non-disease
Actual	Disease	TP	FP
	Non-disease	FN	TN

```
from sklearn.metrics import accuracy_score,
precision_score,recall_score,f1_score

print('Accuracy :',accuracy_score(test_target,y_pred))
print('Precision :',precision_score(test_target,y_pred))
print('Recall :',recall_score(test_target,y_pred))
print('F1 score :',f1_score(test_target,y_pred))
```

[실행결과]
Accuracy : 0.6666666666666666
Precision : 0.5245901639344263
Recall : 0.4776119402985074
F1 score : 0.5

7. KNN 실습 - 보험료 예측

◦ 데이터셋 로드

```
import pandas as pd

df = pd.read_csv('Medical_Insurance_dataset.csv')
df.head()
```

[실행결과]

	age	sex	bmi	smoker	region	children	charges
0	21.000000	male	25.745000	no	northeast	2	3279.868550
1	36.976978	female	25.744165	yes	southeast	3	21454.494239
2	18.000000	male	30.030000	no	southeast	1	1720.353700
3	37.000000	male	30.676891	no	northeast	3	6801.437542
4	58.000000	male	32.010000	no	southeast	1	11946.625900

7. KNN 실습 - 보험료 예측

◦ 문자열 데이터 변환

```
df.loc[df['sex']=='male','sex']=0
df.loc[df['sex']=='female','sex']=1
df['sex'] = df['sex'].astype('int32')

df.loc[df['smoker']=='no','smoker']=0
df.loc[df['smoker']=='yes','smoker']=1
df['smoker'] = df['smoker'].astype('int32')
```

7. KNN 실습 - 보험료 예측

◦ 범주형 데이터 원 핫 인코딩 수행 및 확인

```
df = pd.get_dummies(df)
df.head()
```

[실행결과]

	age	sex	bmi	smoker	children	charges	region_northeast	region_northwest	region_southeast	region_southwest
0	21.000000	0	25.745000	0	2	3279.868550	1	0	0	0
1	36.976978	1	25.744165	1	3	21454.494239	0	0	1	0
2	18.000000	0	30.030000	0	1	1720.353700	0	0	1	0
3	37.000000	0	30.676891	0	3	6801.437542	1	0	0	0
4	58.000000	0	32.010000	0	1	11946.625900	0	0	1	0

◦ 학습 특징과 예측 항목 설정

```
features = df[df.keys().drop('charges')].values
outcome = df['charges'].values
```

7. KNN 실습 - 보험료 예측

- 훈련 데이터와 테스트 데이터로 나누기

```
from sklearn.model_selection import train_test_split

train_features, test_features, train_target, test_target =
train_test_split(features,outcome,random_state=42)
```

- 데이터 스케일 수행

```
from sklearn.preprocessing import MinMaxScaler

feature_scaler = MinMaxScaler()
train_features_scaled = feature_scaler.fit_transform(train_features)
test_features_scaled = feature_scaler.transform(test_features)
```

7. KNN 실습 - 보험료 예측

◦ KNN 알고리즘으로 학습

```
from sklearn.neighbors import KNeighborsRegressor

knn = KNeighborsRegressor()
knn.fit(train_features_scaled, train_target)
```

◦ 성능 평가 확인

```
y_pred = knn.predict(test_features_scaled)
from sklearn.metrics import mean_absolute_error,
mean_squared_error
import numpy as np

print("MAE :",mean_absolute_error(test_target, y_pred))
print("RMSE :",np.sqrt(mean_squared_error(test_target, y_pred)))
```

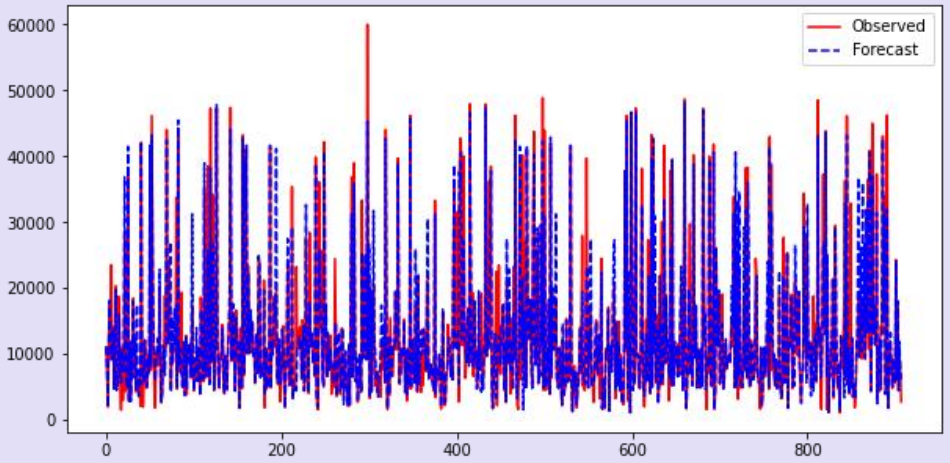
[실행결과]
MAE : 2116.379833072953
RMSE : 4339.148493303896

7. KNN 실습 - 보험료 예측

◦ 시각화하여 확인

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.figure(figsize=(10,5))
plt.plot(test_target,linestyle='-',color='red',label='Observed')
plt.plot(y_pred,linestyle='--',color='blue',label='Forecast')
plt.legend()
plt.show()
```

[실행결과]



8. KNN 실습 - 심혈관 질환 예측

◦ 데이터셋 로드

```
import pandas as pd

df = pd.read_csv('Cardiovascular_Disease_dataset.csv')
df.head()
```

[실행결과]

	id	Age	Gender	Height	Weight	Systolic blood pressure	Diastolic blood pressure	Cholesterol	Glucose	Smoke	Alcohol intake	Physical activity	Presence or absence of cardiovascular disease
0	0	18393	2	168	62.0	110	80	1	1	0	0	1	0
1	1	20228	1	156	85.0	140	90	3	1	0	0	1	1
2	2	18857	1	165	64.0	130	70	3	1	0	0	0	1
3	3	17623	2	169	82.0	150	100	1	1	0	0	1	1
4	4	17474	1	156	56.0	100	60	1	1	0	0	0	0

8. KNN 실습 - 심혈관 질환 예측

◦ 데이터 범주 확인

df.describe()

[실행결과]

	id	Age	Gender	Height	Weight	Systolic blood pressure	Diastolic blood pressure	Cholesterol	Glucose	Smoke	
count	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000
mean	49972.419900	19468.865814	1.349571	164.359229	74.205690	128.817286	96.630414	1.366871	1.226457	0.088129	0
std	28851.302323	2467.251667	0.476838	8.210126	14.395757	154.011419	188.472530	0.680250	0.572270	0.283484	0
min	0.000000	10798.000000	1.000000	55.000000	10.000000	-150.000000	-70.000000	1.000000	1.000000	0.000000	0
25%	25006.750000	17664.000000	1.000000	159.000000	65.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0
50%	50001.500000	19703.000000	1.000000	165.000000	72.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0
75%	74889.250000	21327.000000	2.000000	170.000000	82.000000	140.000000	90.000000	2.000000	1.000000	0.000000	0
max	99999.000000	23713.000000	2.000000	250.000000	200.000000	16020.000000	11000.000000	3.000000	3.000000	1.000000	1

8. KNN 실습 - 심혈관 질환 예측

- 나이 칼럼 계산

```
df['Age']=df['Age']/365
```

- 데이터 범주 확인

```
df.describe()
```

[실행결과]

	id	Age	Gender	Height	Weight	Systolic blood pressure	Diastolic blood pressure	Cholesterol	Glucose	Smoke	
count	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000
mean	49972.419900	53.339358	1.349571	164.359229	74.205690	128.817286	96.630414	1.366871	1.226457	0.088129	0
std	28851.302323	6.759594	0.476838	8.210126	14.395757	154.011419	188.472530	0.680250	0.572270	0.283484	0
min	0.000000	29.583562	1.000000	55.000000	10.000000	-150.000000	-70.000000	1.000000	1.000000	0.000000	0
25%	25006.750000	48.394521	1.000000	159.000000	65.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0
50%	50001.500000	53.980822	1.000000	165.000000	72.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0
75%	74889.250000	58.430137	2.000000	170.000000	82.000000	140.000000	90.000000	2.000000	1.000000	0.000000	0
max	99999.000000	64.967123	2.000000	250.000000	200.000000	16020.000000	11000.000000	3.000000	3.000000	1.000000	1

8. KNN 실습 - 심혈관 질환 예측

- 이상치 데이터 삭제

```
df.loc[df['Systolic blood pressure']<=0,'Systolic blood pressure']=None
df.loc[df['Diastolic blood pressure']<=0,'Diastolic blood pressure']=None
df.isnull().sum()
```

```
[실행결과]
id                                          0
Age                                         0
Gender                                      0
Height                                      0
Weight                                      0
Systolic blood pressure                     7
Diastolic blood pressure                   22
Cholesterol                                 0
Glucose                                     0
Smoke                                       0
Alcohol intake                              0
Physical activity                           0
Presence or absence of cardiovascular disease    0
dtype: int64
```

8. KNN 실습 - 심혈관 질환 예측

◦ 결측치 제거

```
df.dropna(inplace=True)
df.reset_index(inplace=True,drop=True)
```

◦ 성별 0과 1로 표현

```
df.loc[df['Gender']==2,'Gender']=0
df.head()
```

[실행결과]

	id	Age	Gender	Height	Weight	Systolic blood pressure	Diastolic blood pressure	Cholesterol	Glucose	Smoke	Alcohol intake	Physical activity	Presence or absence of cardiovascular disease
0	0	50.391781	0	168	62.0	110.0	80.0	1	1	0	0	1	0
1	1	55.419178	1	156	85.0	140.0	90.0	3	1	0	0	1	1
2	2	51.663014	1	165	64.0	130.0	70.0	3	1	0	0	0	1
3	3	48.282192	0	169	82.0	150.0	100.0	1	1	0	0	1	1
4	4	47.873973	1	156	56.0	100.0	60.0	1	1	0	0	0	0

8. KNN 실습 - 심혈관 질환 예측

◦ 학습 특징과 예측 항목 설정

```
features = df[df.keys().drop(['id','Presence or absence of cardiovascular disease'])]
outcome = df['Presence or absence of cardiovascular disease']
```

```
outcome.value_counts()
```

[실행결과]

```
0    35002
1    34969
Name: Presence or absence of cardiovascular disease, dtype: int64
```

◦ 훈련 데이터와 테스트 데이터 나누기

```
from sklearn.model_selection import train_test_split

train_features, test_features, train_target, test_target =
train_test_split(features,outcome,stratify=outcome,random_state=42)
```

8. KNN 실습 - 심혈관 질환 예측

- 데이터 스케일 수행

```
from sklearn.preprocessing import MinMaxScaler

feature_scaler = MinMaxScaler()
train_features_scaled = feature_scaler.fit_transform(train_features)
test_features_scaled = feature_scaler.transform(test_features)
```

- KNN 알고리즘 학습

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()
knn.fit(train_features_scaled, train_target)
```

8. KNN 실습 - 심혈관 질환 예측

◦ 성능 평가 확인

```
y_pred = knn.predict(test_features_scaled)
from sklearn.metrics import accuracy_score,
precision_score,recall_score,f1_score

print('Accuracy :',accuracy_score(test_target,y_pred))
print('Precision :',precision_score(test_target,y_pred))
print('Recall :',recall_score(test_target,y_pred))
print('F1 score :',f1_score(test_target,y_pred))
```

[실행결과]
Accuracy : 0.6045275252958326
Precision : 0.6041095890410959
Recall : 0.605353466026081
F1 score : 0.6047308878985258

9. Homework

◦ 스스로 해보기

- breast-cancer-Wisconsin.data 데이터셋을 이용하여 KNN 알고리즘을 사용하여 학습하여라. 훈련 데이터와 테스트 데이터를 나누어서 훈련 데이터를 통해 학습하고, 테스트 데이터로 성능 평가를 수행하여라

	A	B	C	D	E	F	G	H	I	J	K
1	1000025	5	1	1	1	2	1	3	1	1	2
2	1002945	5	4	4	5	7	10	3	2	1	2
3	1015425	3	1	1	1	2	2	3	1	1	2
4	1016277	6	8	8	1	3	4	3	7	1	2
5	1017023	4	1	1	3	2	1	3	1	1	2
6	1017122	8	10	10	8	7	10	9	7	1	4
7	1018099	1	1	1	1	2	10	3	1	1	2
8	1018561	2	1	2	1	2	1	3	1	1	2
9	1033078	2	1	1	1	2	1	1	1	5	2
10	1033078	4	2	1	1	2	1	2	1	1	2