REPORT

KNU 강원대학교
KANGWON NATIONAL UNIVERSITY

과목명	에너지데이터전처리및분석
담당교수	권기현 교수님
학과	소프트웨어미디어융합전공
학년	4
학번	201921321
이름	이승욱
제출일	2024/11/05

실행환경 : 아나콘다 + 파이참

소스코드, 실행 결과, 설명은 아래의 이미지에 모두 포함되어 있음

1. 파이썬 기반 분류와 회귀

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
import numpy as np
import matplotlib.pyplot as plt
```

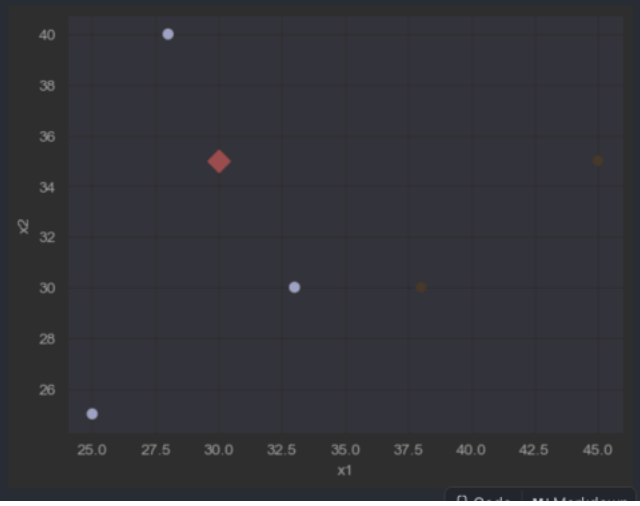
✓ [2] < 10 ms

```
# 학습용 데이터 (기존 개체)
# 입력
X_train = np.array([[25, 25],
                    [33, 30],
                    [38, 30],
                    [45, 35],
                    [28, 40]])

# 라벨
y_train = np.array([0, 0, 1, 1, 0])

# 테스트용 데이터 (새로운 개체)
# 입력
X_test = np.array([[30, 35]])
```

✓ [3] < 10 ms

```
# 산포도
# 학습용 데이터
plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train)
# 테스트용 데이터
plt.scatter(X_test[:, 0], X_test[:, 1], c='red', marker='D', s=100)
plt.xlabel('x1')
plt.ylabel('x2')
plt.show()
```

✓ [4] 122ms



```python
# 피처 스케일링: 학습용 데이터
scalerX = StandardScaler()  # 표준화 스케일러 객체 생성
scalerX.fit(X_train)        # 표준화 스케일러를 학습용 데이터에 맞춤
X_train_std = scalerX.transform(X_train)  # 학습용 데이터의 표준화
print(X_train_std)
```

✓ [5] < 10 ms

```
[[-1.23272999 -1.37281295]
 [-0.11206636 -0.39223227]
 [ 0.58834841 -0.39223227]
 [ 1.56892908  0.58834841]
 [-0.81248113  1.56892908]]
```

```python
# 피처 스케일링: 테스트용 데이터
X_test_std = scalerX.transform(X_test)  # 테스트 데이터의 변환
print(X_test_std)
```

✓ [6] < 10 ms

```
[[-0.53231522  0.58834841]]
```

```python
# 모델화
knn = KNeighborsClassifier(n_neighbors=3, metric='euclidean')

# 학습
knn.fit(X_train_std, y_train)

# 예측
pred = knn.predict(X_test_std)
print(pred)
```

✓ [8] < 10 ms

```
[0]
```

```python
# 클래스별 확률 값을 반환
print(knn.predict_proba(X_test_std))
```

✓ [9] < 10 ms

```
[[0.66666667 0.33333333]]
```

```python
# 인접한 K개의 개체들에 대한 거리와 색인을 반환
dist, index = knn.kneighbors(X_test_std)
print(dist)
print(index)
```

✓ [10] < 10 ms

```
[[1.0198193  1.06683999 1.48910222]]
[[4 1 2]]
```

2. 파이썬 기반 분류와 회귀(새로운 개체)

```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.preprocessing import StandardScaler
import numpy as np
import matplotlib.pyplot as plt
```

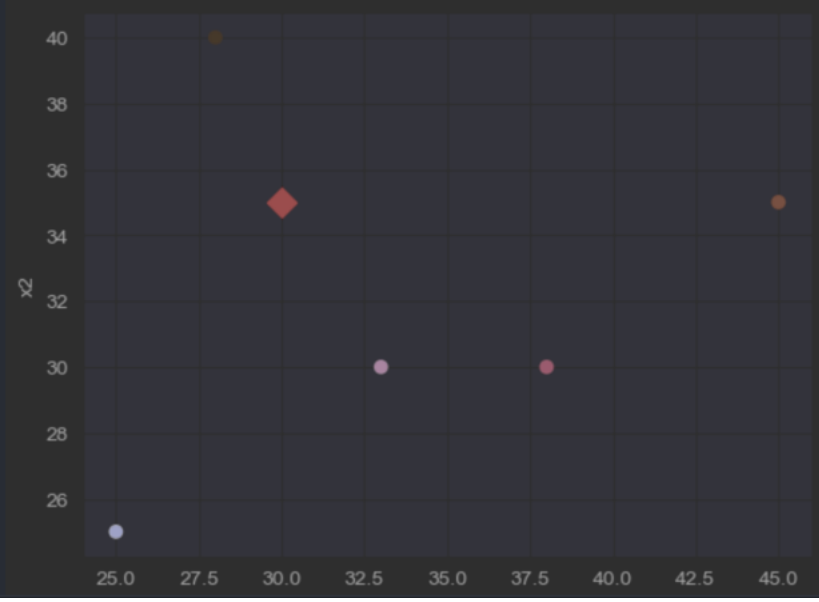
✓ [11] < 10 ms

```
# 학습용 데이터(기존 개체)
# 입력 값
X_train = np.array([[25, 25],
                    [33, 30],
                    [38, 30],
                    [45, 35],
                    [28, 40]])

```

✓ [12] < 10 ms

```
# 목표 값
y_train = np.array([[10], [20], [30], [40], [50]])

# 테스트용 데이터(새로운 개체)
# 입력
X_test = np.array([[30, 35]])

# 산포도
# 학습용 데이터
plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train)
# 테스트용 데이터
plt.scatter(X_test[:, 0], X_test[:, 1], c='red', marker='D', s=100)
plt.xlabel('x1')
plt.ylabel('x2')
plt.show()
```

✓ [13] 69ms



```python
# 피처 스케일링: 학습용 데이터
# 입력 값
scalerX = StandardScaler()
scalerX.fit(X_train)
X_train_std = scalerX.transform(X_train)
print("X_train_std:", X_train_std)
```

✓ [17] < 10 ms

```
X_train_std: [[-1.23272999 -1.37281295]
 [-0.11206636 -0.39223227]
 [ 0.58834841 -0.39223227]
 [ 1.56892908  0.58834841]
 [-0.81248113  1.56892908]]
```

```python
# 목표 값
scalerY = StandardScaler()
scalerY.fit(y_train)
y_train_std = scalerY.transform(y_train)
print("y_train_std:", y_train_std)
```

✓ [18] < 10 ms

```
y_train_std: [[-1.41421356]
 [-0.70710678]
 [ 0.        ]
 [ 0.70710678]
 [ 1.41421356]]
```

```python
# 피처 스케일링: 테스트용 데이터
X_test_std = scalerX.transform(X_test)
print("X_test_std:", X_test_std)
```

✓ [19] < 10 ms

```
X_test_std: [[-0.53231522  0.58834841]]
```

```python
# 모형화
knn = KNeighborsRegressor(n_neighbors=3, metric='euclidean', weights="uniform")

# 학습
knn.fit(X_train_std, y_train_std)

# 예측
y_pred = knn.predict(X_test_std)
print("y_pred:", y_pred)
```

✓ [20] < 10 ms

```
y_pred: [[0.23570226]]
```

```python
# 예측 값의 역변환
y_pred_inverse = scalerY.inverse_transform(y_pred)
print("y_pred_inverse:", y_pred_inverse)
```

✓ [21] < 10 ms

```
y_pred_inverse: [[33.33333333]]
```

3. 유방암 진단

```python
# 필요한 라이브러리 임포트
from sklearn.datasets import load_breast_cancer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# 데이터 불러오기
data = load_breast_cancer(as_frame=True)
```

```python
# 데이터 프레임 출력
print(data.frame)
```

✓ [22] 40ms

```
     mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
0          17.99         10.38          122.80     1001.0          0.11840
1          20.57         17.77          132.90     1326.0          0.08474
2          19.69         21.25          130.00     1203.0          0.10960
3          11.42         20.38           77.58      386.1          0.14250
4          20.29         14.34          135.10     1297.0          0.10030
..           ...           ...             ...        ...              ...
564        21.56         22.39          142.00     1479.0          0.11100
565        20.13         28.25          131.20     1261.0          0.09780
566        16.60         28.08          108.30      858.1          0.08455
567        20.60         29.33          140.10     1265.0          0.11780
568         7.76         24.54           47.92      181.0          0.05263
```

{} Code | M↓ Markdown

```python
# 학습용과 테스트 데이터 분리
X_train, X_test, y_train, y_test = train_test_split(data.data, data.target, test_size=0.3,
 random_state=1234)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

✓ [27] < 10 ms

```
(398, 30)
(171, 30)
(398,)
(171,)
```

```python
# 입력 부분과 목표 값 출력
print(data.data)
print(data.target)
```

✓ [23] < 10 ms

```
1                 0.05667  ...         24.990          25.41
2                 0.05999  ...         23.570          25.53
3                 0.09744  ...         14.910          26.50
4                 0.05883  ...         22.540          16.67
..                    ...  ...            ...            ...
564               0.05623  ...         25.450          26.40
565               0.05533  ...         23.690          38.25
566               0.05648  ...         18.980          34.12
567               0.07016  ...         25.740          39.42
568               0.05884  ...          9.456          30.37

     worst perimeter  worst area  worst smoothness  worst compactness  \
0             184.60      2019.0           0.16220            0.66560
```

```python
# 데이터 세트 개요
print(data.DESCR)
```

✓ [24] < 10 ms

```
.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset
--------------------------------------------

**Data Set Characteristics:**
                        양각

:Number of Instances: 569

:Number of Attributes: 30 numeric, predictive attributes and the class

:Attribute Information:
```

```python
# 학습용과 테스트 데이터 분리
X_train, X_test, y_train, y_test = train_test_split(data.data, data.target, test_size=0.3,
 random_state=1234)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

✓ [27] < 10 ms

```
(398, 30)
(171, 30)
(398,)
(171,)
```

```python
# 데이터 구조 특성 이름과 목표 변수 이름
print(data.feature_names)
print(data.target_names)
```

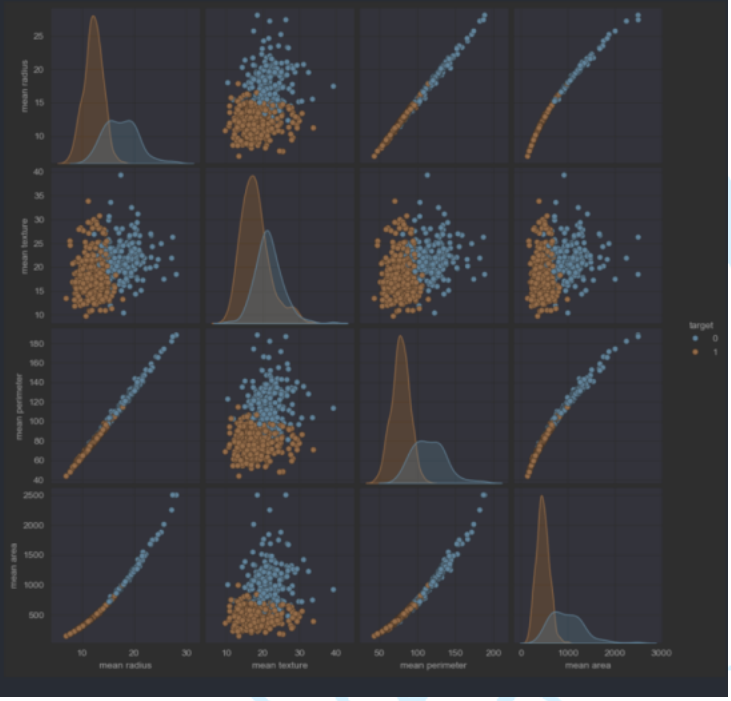
✓ [25] < 10 ms

```
['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
['malignant' 'benign']
```

```python
# 변수 간 산포도
data_mean = data.frame[['mean radius', 'mean texture', 'mean
 perimeter', 'mean area', 'target']]
sns.pairplot(data_mean, hue='target')
```

✓ [26] 1s 401ms

```
<seaborn.axisgrid.PairGrid at 0x146031250>
```



```python
# 학습용과 테스트 데이터 분리
X_train, X_test, y_train, y_test = train_test_split(data.data, data.target, test_size=0.3,
 random_state=1234)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

✓ [27] < 10 ms

```
(398, 30)
(171, 30)
(398,)
(171,)
```

```python
# 피처 스케일링: 학습 데이터
scalerX = StandardScaler()
scalerX.fit(X_train)
X_train_std = scalerX.transform(X_train)
print(X_train_std)
```

✓ [28] < 10 ms

```
   0.22129607]
 [-0.79609663 -0.38603656 -0.81356785 ... -0.43011095  0.08970515
  -0.36303452]
 [ 0.21752653 -0.38603656  0.18557689 ...  0.76443594  0.80894448
  -0.67502531]
 ...
 [-0.48269225 -0.14686262 -0.46083202 ... -0.21253919  0.1565732
   0.16129784]
 [ 1.14079887 -0.12364185  1.14739725 ...  0.25197353  0.1679897
  -0.23677737]
 [-0.41210568 -1.26610378 -0.43253113 ... -0.78299078 -0.89537548
  -0.79241315]]
```

```python
# 피처 스케일링: 테스트 데이터
X_test_std = scalerX.transform(X_test)
print(X_test_std)
```

✓ [29] < 10 ms

```
   0.26564259]
 [-0.95421055 -2.21118915 -0.9661466  ... -1.04371727 -1.35040445
  -0.38703381]
 [-0.48833918 -0.6553975  -0.38864423 ...  0.27744681  0.51048462
   0.99083859]
 ...
 [-0.45163416 -0.19330417 -0.51251192 ... -1.60200162 -0.67356925
  -1.04857951]
 [-0.45728109 -0.037725   -0.42678891 ... -0.34515008 -1.29984567
  -0.65363464]
 [ 0.58740016  0.61477866  0.62239509 ... -0.16458949 -0.29682483
  -0.25451598]]
```

```python
# 최근접 이웃 수 결정
train_accuracy = []
test_accuracy = []

# 최근접 이웃의 수: 1~15
neighbors = range(1, 16)
for k in neighbors:
    # 모델화
    knn = KNeighborsClassifier(n_neighbors=k)
    # 학습
    knn.fit(X_train_std, y_train)
    # 학습 데이터의 분류 정확도
    score = knn.score(X_train_std, y_train)
    train_accuracy.append(score)  # K에 따른 학습 데이터의 분류 정확도 추가
    # 테스트 데이터의 분류 정확도
    score = knn.score(X_test_std, y_test)
    test_accuracy.append(score)  # K에 따른 테스트 데이터의 분류 정확도 추가
```

```python
# 테스트 데이터의 분류 정확도
print(test_accuracy)
```

✓ [31] < 10 ms

```
[0.9239766081871345, 0.9239766081871345, 0.9298245614035088, 0.935672514619883, 0.
.9239766081871345, 0.9415204678362573, 0.9298245614035088, 0.9415204678362573, 0.
.9239766081871345, 0.9298245614035088, 0.9181286549707602, 0.9181286549707602, 0.
.9181286549707602, 0.9239766081871345, 0.9181286549707602]
```

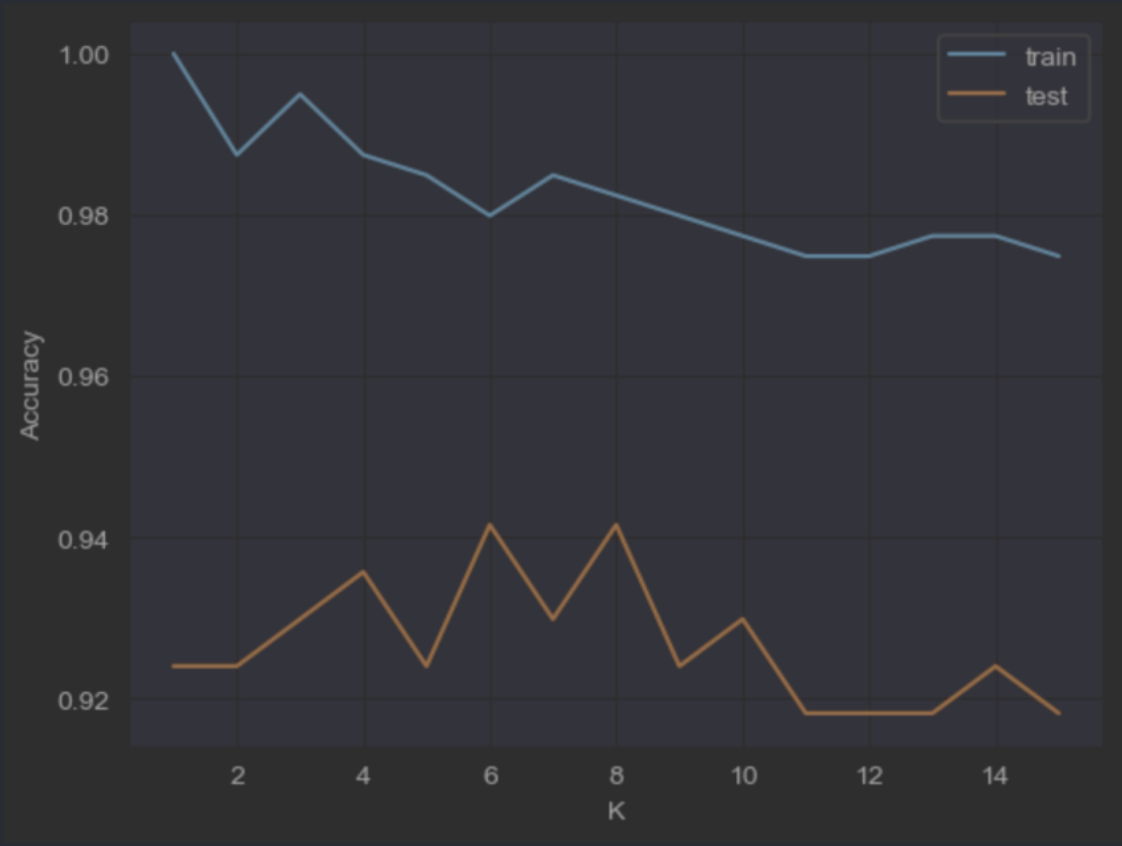
```python
# 모델화
K = 6
knn = KNeighborsClassifier(n_neighbors=K)

# 학습
knn.fit(X_train_std, y_train)

# 예측
y_pred = knn.predict(X_test_std)
print(y_pred)
```

✓ [32] < 10 ms

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

```python
# K의 크기에 따른 분류 정확도 변화
plt.plot(neighbors, train_accuracy, label="train")
plt.plot(neighbors, test_accuracy, label="test")
plt.xlabel("K")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```

✓ [30] 212ms




```python
# confusion matrix
cf = confusion_matrix(y_test, y_pred)
print(cf)
```

✓ [33] < 10 ms

```
[[ 56  10]
 [  0 105]]
```

{} Code M↓ Markdown

```python
# 테스트 데이터에 대한 정확도
accuracy = knn.score(X_test_std, y_test)
print("Test Accuracy:", accuracy)
```

✓ [34] < 10 ms

```
Test Accuracy: 0.9415204678362573
```

{} Code M↓ Markdown