[Machine Learning]

[2023-1]

Homework 2

[Due Date] 2023.04.26

Student ID : 2018112007

Name : 이승현

Professor : Juntae Kim

1. Describe the two Impurity measures for Decision Tree Learning (including math formula) and their meaning. (10 pts)

Your Answer
1. Entropy - 불확실성의 정도를 측정한다. - Entropy 가 낮을수록 많은 정보량을 가지고 있으므로, 분류 정확도가 높아진다. $I(S) = \sum_{i=1}^{m} p_i \left(-\log_2 p_i\right) = -\sum_{i=1}^{m} \frac{s_i}{s} \left(\log_2 \frac{s_i}{s}\right)$ 2. Gini 계수 - 분포에 따라 무작위로 레이블이 지정된 경우 세트에서 무작위로 선택된 요소가 잘못된 레이블인 경우의 빈도를 측정한다. - 계수가 낮을수록 노드 내에 같은 클래스의 샘플이 많아지므로, 분류 정확도가 높아진다. $I(S) = \sum_{i=1}^{m} p_i \left(1 - p_i\right) = \sum_{i=1}^{m} p_i - p_i^2 = 1 - \sum_{i=1}^{m} p_i^2$

2. Following 1-D dataset is not linearly separable (we can not find decision boundary x = c). Show how we make them linearly separable by transform the dataset to higher dimensional space. (10 pts)

Your Answer

입력 데이터 X를 고차원으로 확장하여 사용하여 linear boundary 를 구해야한다. 그런데 차원이 높아지면서 연산이 매우 복잡해지기 때문에 최적화를 해야 할 필요성을 느끼게 되었고, 이에 커널 함수를 적용하게 되었다.

커널 함수는 이러한 특성을 띄고 있다.

$$K(x_i, x) = \varphi(x_i) * \varphi(x)$$

고차원으로 확장된 support vector $\varphi(x_i)$ 를 사용하여 w를 이렇게 구할 수 있다.

$$w = \sum c_i y_i \varphi(x_i)$$

입력 데이터 X 에 대하여 prediction 을 수행하면

$$\text{y\_hat} = w^T * \varphi(x) + b = \sum c_i y_i \varphi(x_i) * \varphi(x) + b = \sum c_i y_i K(x_i, x) + b$$

이렇게 커널 함수를 적용함으로써, 차원은 늘어남에도 내적 연산은 더 적어졌기 때문에 수월하기 linear boundary 를 구할 수 있다.

이렇게 커널 함수를 이용하여 linear boundary 를 구하는 방법을 kernal trick 이라고 한다.

3. Explain what 'Feature Selection' and 'Dimensionality Reduction' are, and why they are needed in machine learning. (10 pts)

Your Answer
1. Feature Selection 은 입력 데이터에서 가장 유용한 특징들을 선택하는 것이다. 즉, 모델 성능에 영향을 미치지 않거나 오히려 성능을 향상시키는 특징들만 선택하고, 불필요한 특징들은 제거한다. 이렇게 선택된 특징만을 사용하여 모델을 학습하면, 모델이 더 간단하고 빠르게 학습되며, overfitting 을 방지할 수 있다. 2. Dimensionality Reduction 은 입력 데이터의 특징 수를 줄이는 방법 중 하나다. 입력 데이터의 차원이 너무 높으면, 모델이 복잡해져 학습이 어렵고, 모델의 성능이 떨어질 수 있다. 이를 해결하기 위해, 입력 데이터의 차원을 축소하여

데이터의 구조를 유지하면서 불필요한 정보를 제거한다. 이 방법을 사용하면, 모델의 성능을 향상시키면서도 모델의 복잡성을 낮출 수 있다.

Feature Selection 과 Dimensionality Reduction 은 모델의 성능을 향상시키기 위해 필요하다. 불필요한 특징들이나 차원이 높은 입력 데이터는 모델의 성능을 저하시키는 원인이 될 수 있으므로, 이를 제거하여 모델을 보다 간단하고 정확하게 만들며 데이터 분석의 효율성을 더욱 높일 수 있다.

4. Explain what 'Bias' and 'Variances' are and how they affect the performance of machine learning models. (10 pts)

Your Answer
1. Bias 는 모델을 통한 예측 값과 실제 값의 차이를 나타낸 값이다. 다양한 dataset 에 대한 평균적인 error 의 값이며, bias 가 높을수록 underfitting 이 되기 때문에 feature 의 수를 늘리고 regularization 정도를 줄여서 예측 값과 실제 값의 차이를 줄여야 한다. 2. Variance 는 예측 값의 다양성에 대한 정보이다. 다양한 dataset 에 대한 error 의 변화 정도를 살펴보며, variance 가 높을수록 overfitting 이 되기 때문에 feature 의 수를 줄이고, regularization 정도를 늘려서 overfitting 이 되지 않도록 해야 한다.

5. Apply Logistic Regression, Decision Tree, k-Nearest Neighbor, and SVM on Wine Dataset to predict the origin of wines. Describe the learned model, and compare the accuracies. (20 pts)

- Dataset

https://archive.ics.uci.edu/ml/datasets/Wine

The dataset is the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the 3 types of wines.

- Use downloaded raw data or scikit-learn library

https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_wine.html

```
from sklearn.datasets import load_wine

wine = load_wine()
X = wine.data
y = wine.target
```

- Check test accuracies (use 30% for test)

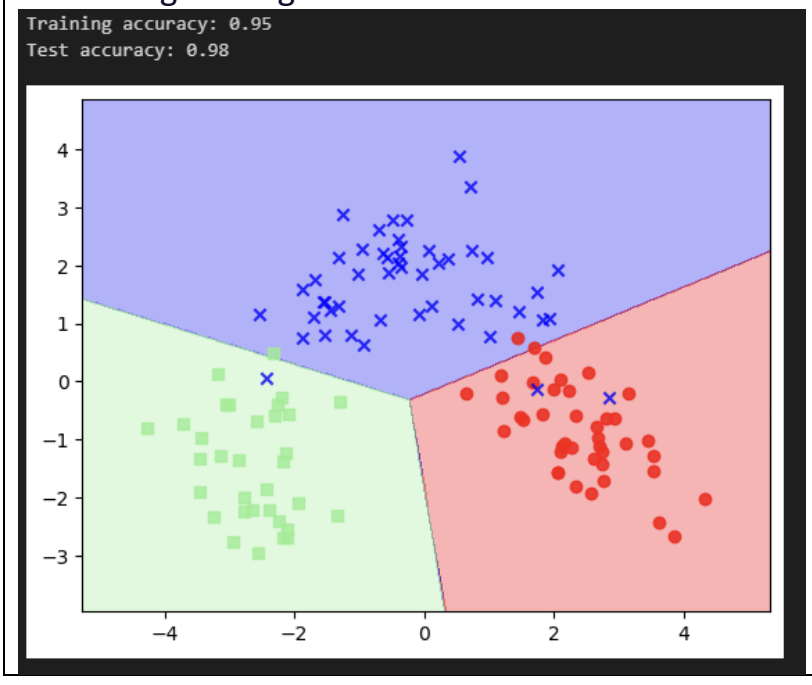
Code
from sklearn.datasets import load_wine from sklearn.model_selection import train_test_split import numpy as np wine = load_wine() X = wine.data y = wine.target X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y) from sklearn.preprocessing import StandardScaler scaler = StandardScaler() X_train = scaler.fit_transform(X_train) X_test = scaler.transform(X_test) from sklearn.decomposition import PCA pca = PCA(n_components=2) X_train = pca.fit_transform(X_train)

```
X_test = pca.transform(X_test)

from matplotlib.colors import ListedColormap
import matplotlib.pyplot as plt

def plot_decision_regions(X, y, classifier, resolution=0.02):

    # setup marker generator and color map
    markers = ('o', 'x', 's', '^', 'v')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])

    # plot the decision surface
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
                           np.arange(x2_min, x2_max, resolution))
    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)
    plt.contourf(xx1, xx2, Z, alpha=0.3, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())

    # plot class samples
    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],
                    alpha=0.8, c=colors[idx], marker=markers[idx], label=cl)

from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(multi_class='ovr', random_state=1)
lr.fit(X_train, y_train)

plot_decision_regions(X_train, y_train, lr)

print('Training accuracy: %.2f' % lr.score(X_train, y_train))
print('Test accuracy: %.2f' % lr.score(X_test, y_test))

from sklearn.tree import DecisionTreeClassifier
```

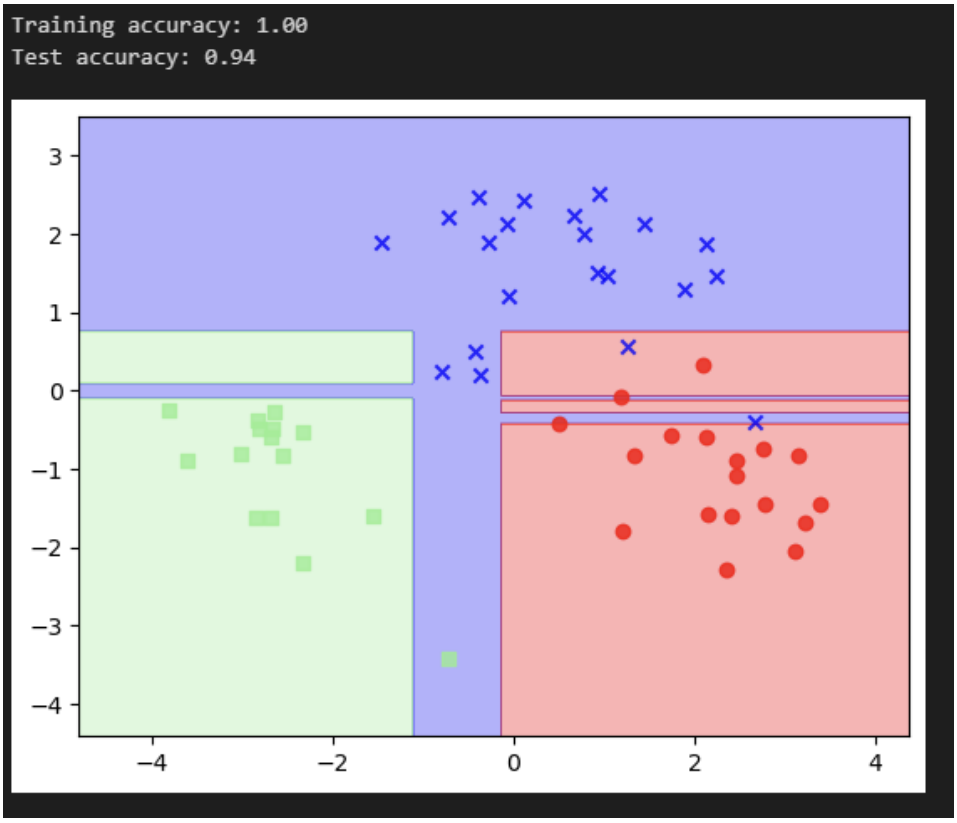
```
tree = DecisionTreeClassifier(random_state=1)
tree.fit(X_train, y_train)
plot_decision_regions(X_test, y_test, tree)

print('Training accuracy: %.2f' % tree.score(X_train, y_train))
print('Test accuracy: %.2f' % tree.score(X_test, y_test))

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5, p=2)
knn.fit(X_train, y_train)
plot_decision_regions(X_test, y_test, knn)
print('Training accuracy: %.2f' % knn.score(X_train, y_train))
print('Test accuracy: %.2f' % knn.score(X_test, y_test))

from sklearn.svm import SVC
svm = SVC(kernel = 'rbf', random_state=1, gamma=0.2, C=1.0)
svm.fit(X_train, y_train)
plot_decision_regions(X_test, y_test, svm)
print('Training accuracy: %.2f' % svm.score(X_train, y_train))
print('Test accuracy: %.2f' % svm.score(X_test, y_test))
```

Result(Captured images)

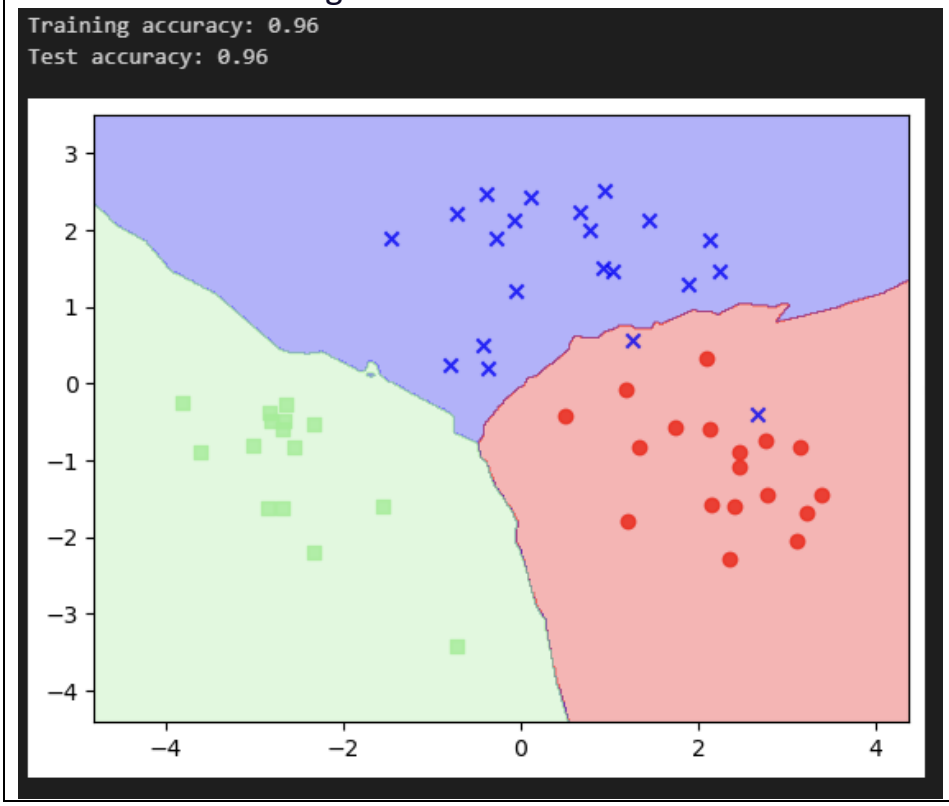
1. Logistic regression



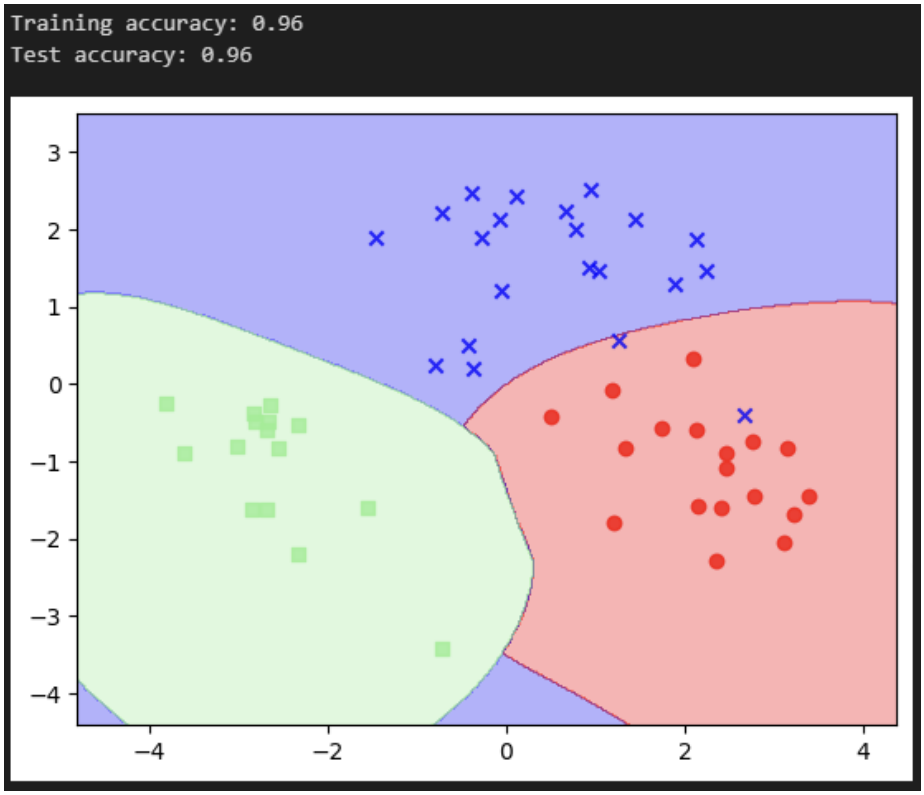
2. Decision tree



3. K-nearest neighbor



4. SVM



Description

학습된 모델을 표현하기 위해, PCA 라이브러리를 사용하여 입력 데이터의 차원을 줄인 뒤 prediction 을 수행하고, decision region 을 화면에 출력하였다.

Logistic regression 모델은 3 가지 클래스를 예측해야 하므로 multi_class = 'ovr' 옵션을 파라미터로 전달하여 one vs rest 방법을 사용하였다.

Decision tree 모델은 max depth 의 제한 없이 예측을 진행하였다. 참고로 depth 를 임의로 설정해도 예측을 진행해도 test accuracy 가 동일하게 나온다.

k-nearest neighbor 모델은 neighbor 의 개수는 5, p 는 2 로 설정하여 Euclidean distance 로 클래스를 예측하였다.

SVM 모델은 kernel 은 rbf, gamma 는 0.2, c 는 1.0 으로 설정하여 prediction 을 진행하였다.

Test accuracy 를 기준으로 모델 간 정확성을 비교하면 logistic regression 이 0.98 로 가장 높았고, 그 다음으로 k-nearest neighbor 와 svm 이 0.96 으로 높았고, decision tree 가 0.94 로 가장 낮았다.

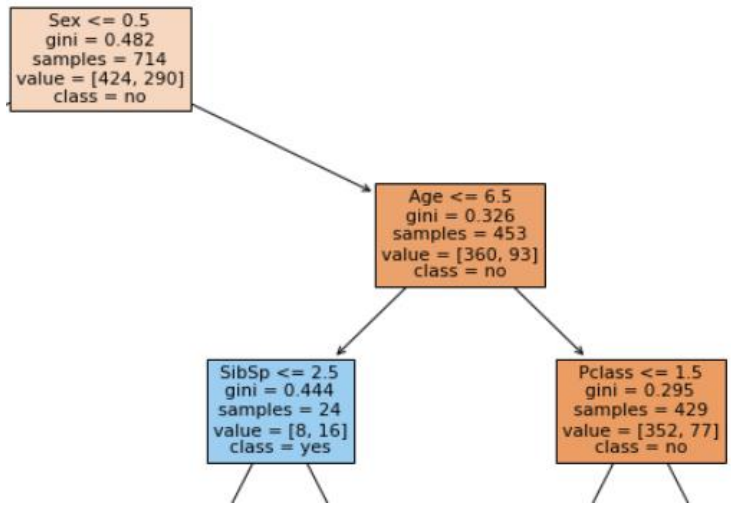
(Logistic regression > k-nearest neighbor = svm > decision tree)

6. By using Decision Tree learning, find out who survived from Titanic accident. Use only relevant features, and present the result as interpretable rules. (20 pts)

- Dataset : titanic.csv

- Preprocessing :
    1. Remove irrelavant features
        - PassengerId, Name, Ticket, Cabin, Fare, Embarked
    2. Remove data with missing values
    3. Convert feature 'Sex' to numerical (0/1)

- Analysis :
    1. Plot the decision tree with depth=3
    2. Describe who survived the accident by rules from the tree. Choose all rules that satisfy following condition: gini < 0.3 and samples > 20
       Ex>

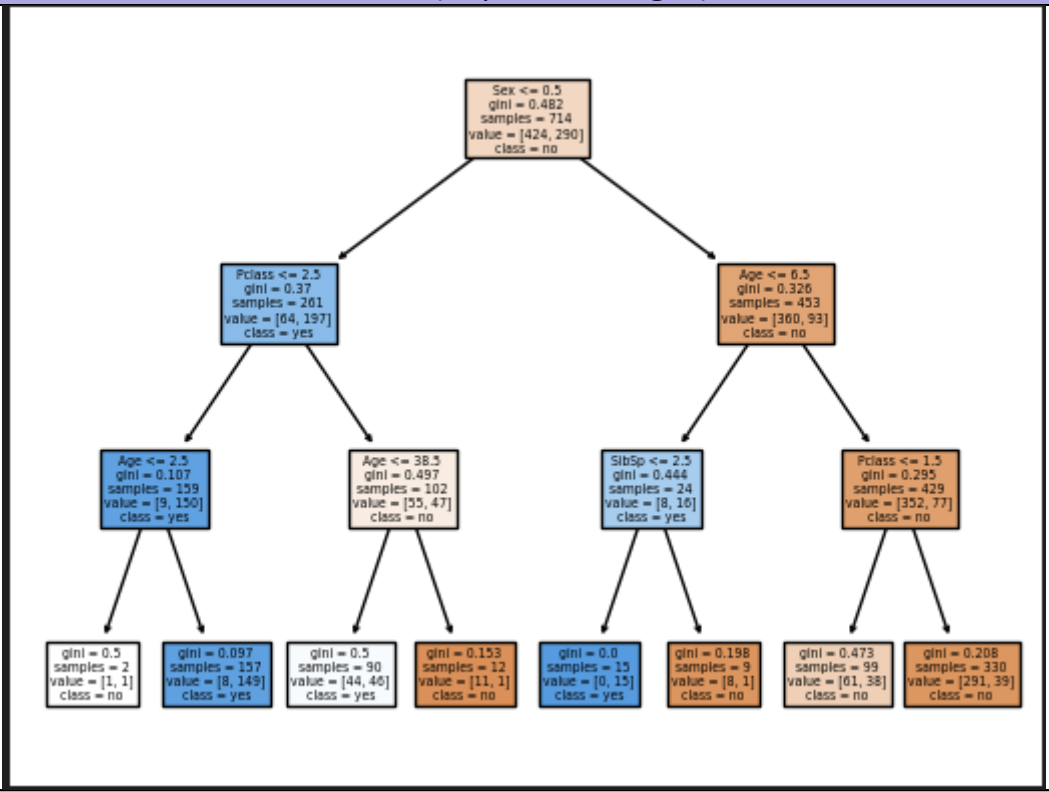


--> If (Sex == male and Age >= 7) then survived = No

Code

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder

df = pd.read_csv("titanic.csv")
df_select = df.drop(columns=['PassengerId', 'Name', 'Ticket', 'Cabin', 'Fare',
'Embarked'])
df_select = df_select.dropna(axis=0)
enc = LabelEncoder()
df_select["Sex"] = enc.fit_transform(df_select['Sex'])

X = df_select.values[:, 1:]
y = df_select.values[:, 0]

from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree

tree = DecisionTreeClassifier(criterion = 'gini', max_depth=3, random_state=1)
tree.fit(X, y)

plot_tree(tree, feature_names=df_select.columns[1:], class_names=['no', 'yes'],
filled=True)
plt.show()
```

Result(Captured images)



Description

gini < 0.3 and samples > 20 를 만족하는 규칙은

1. If (Sex == female and Pclass <= 2.5) then survived = Yes
2. If (Sex == female and Pclass <= 2.5 and Age >= 3) then survived = Yes
3. If (Sex == male and Age >= 7) then survived = No
4. If (Sex == male and Age >= 7 and Pclass >= 2) then survived = No

7. Apply K Nearest Neighbor classifier to heart_disease.csv dataset as follows. (20 pts)

- Setting X, y

```
hd = pd.read_csv('heart_disease.csv')

# set num values to 0 and 1 (if it is > 0)
hd['num'] = np.where(hd['num'] > 0, 1, 0)
```

```
# Make X, y using all features
X = hd.values[:, :-1]
y = hd.values[:, -1].astype(np.int32)
```

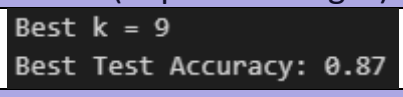
- Split the dataset as 70% train and 30% test
- Make a pipeline with StandardScaler and KNeighborsClassifier
- Plot validation curve with k values 1 ~ 20
- Find the best k value and test accuracy for that value

Code

```
import pandas as pd
import numpy as np

hd = pd.read_csv('heart_disease.csv')

# set num values to 0 and 1 (if it is > 0)
hd['num'] = np.where(hd['num'] > 0, 1, 0)

# Make X, y using all features
X = hd.values[:, :-1]
y = hd.values[:, -1].astype(np.int32)

from sklearn.model_selection import train_test_split

# train / test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, stratify=y,
random_state=1)

from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import make_pipeline

k = 1
Accuracy = 0
for i in range(1, 21) :
   pipe_lr = make_pipeline(StandardScaler(), KNeighborsClassifier(n_neighbors=i,
p=2))
   pipe_lr.fit(X_train, y_train)
```

```
    if Accuracy < pipe_lr.score(X_test, y_test) :
        k = i
        Accuracy = pipe_lr.score(X_test, y_test)

print('Best k = %d' % k)
print('Best Test Accuracy: %.2f' % Accuracy)
```

Result(Captured images)



Description

파이프라인을 사용하여 standardization 과 k-nearest neighbor 를 동시에 수행한 모습이다. For 문을 이용하여 k-nearest neighbor 의 n_neighbors 파라미터를 1 부터 20 까지 변화시켜 정확도가 최대가 되는 k 의 값을 찾고, k 에 상응하는 정확도를 출력하였다. 가장 정확도가 높은 k 값은 9 이며, 정확도는 0.87 이 나왔다.

Note

1. Summit the file to e-class as pdf.
2. Specify your file name as “hw2_<StudentID>_<Name>.pdf”