**PHW1**

**201835474 안해빈, 202037634 윤주은, 201835503 이지민, 201835508 임윤수**

**<Code>**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings(action='ignore')

from sklearn.preprocessing import StandardScaler, MinMaxScaler, RobustScaler, MaxAbsScaler, Normalizer

from sklearn.tree import DecisionTreeClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.model\_selection import train\_test\_split, KFold, cross\_val\_score

from sklearn.metrics import accuracy\_score

# Data exploration

df = pd.read\_csv('breast-cancer-wisconsin.data', sep=',', header=None) # No column name in dataset, read first line as value

print(f'First 5 rows of data\n{df.head()}', end='\n\n')

print(f'Data shape\n{df.shape}', end='\n\n')

# Data preprocessing

df = df.replace('?',int(df.iloc[:, 6].mode())) # Cleaning dirty data in 'Bare Nuclei' with mode value

# Feature engineering

df = df.drop(df.columns[0], axis = 1) # 'Sample code number' column has no relationship with target column

# Prepare dataset

X = df.iloc[:, 0:8]

y = df.iloc[:,-1]

# Showing information of target column

ax = sns.countplot(y).set(title='Number of benign and malignant')

B, M = y.value\_counts()

print('Number of Benign: ', B)

print('Number of Malignant : ', M)

plt.show()

# Set various data scaling methods and classification models

scaler = ['StandardScaler()', 'MinMaxScaler()', 'RobustScaler()', 'MaxAbsScaler()', 'Normalizer()']

model = ['DecisionTreeClassifier(criterion="entropy")', 'DecisionTreeClassifier(criterion="gini")', 'LogisticRegression()',

         'SVC(kernel="linear", C=0.1)', 'SVC(kernel="linear", C=0.5)', 'SVC(kernel="linear", C=1)', 'SVC(kernel="linear", C=10)'] # Various values for the hyperparameters

k = [5, 8, 10] # Various numbers k for k-fold cross validation

def best\_comb(scaler, model, X, y, k):

    """Train model and find best combination of classifier and scaler, and validate model with k-fold cross validation

    Args:

        scaler: Array of data scaling methods

        model: Array of classification models

        X: Array of independent variables

        y: Array of dependent variable

        k: Hyperparameter of k-fold cross validation

    """

    combi = [] # Combination of data scaling methods and classification models

    for element in scaler:

        scaler = eval(element)

        scaled = scaler.fit\_transform(X)

        X\_train, X\_test, y\_train, y\_test = train\_test\_split(scaled, y, test\_size = 0.2, random\_state=42) # Using random\_state to fixing random rate

        print('')

        for element2 in model:

            classifier = eval(element2)

            classifier = classifier.fit(X\_train,y\_train)

            y\_pred = classifier.predict(X\_test)

            acc = accuracy\_score(y\_test, y\_pred)

            print(f'Using {classifier} in {scaler} score : {acc}')

            combi.append({'acc' : acc, 'scaler' : scaler, 'classifier' : classifier})

    def get\_acc(element):

        """Function for getting accuracy in combi

        Args:

            element: list for getting value

        Returns:

            element['acc']: Return 'acc' in list

        """

        return element['acc']

    combi.sort(key=get\_acc, reverse=True) # Sort combination by accuracy in ascending order

    print('')

    print('Best 5 scaler, model, score in classificiation :', end='\n\n')

    for i in range (0,5) :

        print('|',i + 1, 'accuracy : ', combi[i]['acc'],

              'scaler : ', combi[i]['scaler'],

              'classifier : ', [combi[i]['classifier']])

    # Validate model with k-fold CV with value of array 'k'

    res = []

    for i in k :

        kfold = KFold(n\_splits=i, shuffle=True, random\_state=7)

        for j in range(0, len(combi)):

            res.append(cross\_val\_score(combi[j]['classifier'], X, y, cv = kfold))

        print(f'\nK-fold validation with best 5 scaler, model, score in classificiation :', end='\n\n')

        for j in range (0,5) :

            print(f"Average in K-fold validation score with {combi[j]['classifier']}, {combi[j]['scaler']} in k = {i} : {res[j].mean()}")

best\_comb(scaler, model, X, y, k)

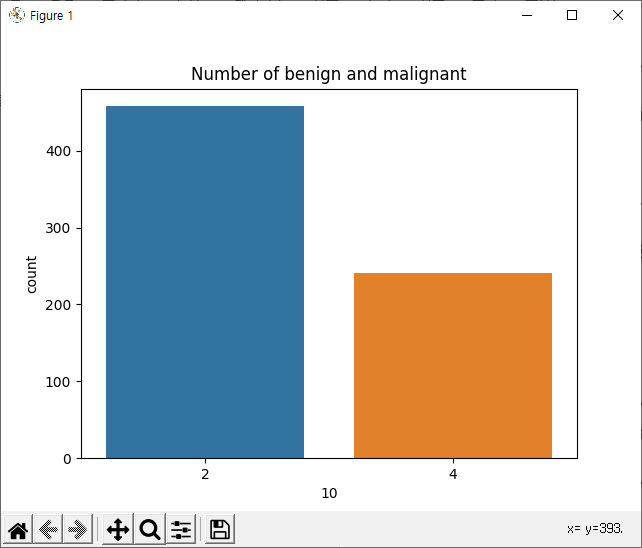
**<Result>**

# Data exploration

**텍스트이(가) 표시된 사진

자동 생성된 설명**

# Showing information of target column

****

# Try various combinations of the scalers, models, hyperparameters

**텍스트, 신문, 서류이(가) 표시된 사진

자동 생성된 설명**

# Results of k-fold cross-validation of the five most accurate combinations  
(k = 5, 8, 10) **텍스트이(가) 표시된 사진

자동 생성된 설명**

**<Contribution>**

안해빈: Dataset analysis / Algorithm Structure development / Preprocessing verification

윤주은: Dataset analysis / Algorithm Structure development / Scaler, Model verification

이지민: Dataset analysis / Algorithm Structure development / Program validation

임윤수: Dataset analysis / Algorithm Structure development / K-Fold verification

*Each members wrote their own programs first, then we verified the best part and integrated it into one program.*

|  |  |  |  |
| --- | --- | --- | --- |
| **Percentage** | | | |
| **안해빈** | **윤주은** | **이지민** | **임윤수** |
| **25%** | **25%** | **25%** | **25%** |