import numpy as np

from pandas.\_typing import F

import pandas as pd

import pickle

from scipy import stats

from sklearn.linear\_model import Lasso

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

from sklearn.experimental import enable\_iterative\_imputer

from sklearn.impute import IterativeImputer

df = pd.read\_csv('originalData.csv')

X=df.drop('num',axis=1)

y=df['num']

print("check whether the data has null values" , X.isnull().sum())

lr = Lasso()

imp=IterativeImputer(estimator=lr,verbose=2,max\_iter=330,tol=1e-10,imputation\_order='roman')

imp.fit(X)

a=imp.transform(X)

my\_array = np.array(a)

X = pd.DataFrame(my\_array)

print("as a dataframe", X)

print("check whether the data has null values" , X.isnull().sum())

XY=X.drop(0,1)

XY=XY.drop(1,1)

XY=XY.drop(6,1)

print("check whether the data has null values" , XY.isnull().sum())

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

def print\_score(clf, X\_train, y\_train, X\_test, y\_test, train=True):

if train:

pred = clf.predict(X\_train)

clf\_report = pd.DataFrame(classification\_report(y\_train, pred, output\_dict=True))

print("Train Result:\n================================================")

print(f"Accuracy Score: {accuracy\_score(y\_train, pred) \* 100:.2f}%")

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

print(f"CLASSIFICATION REPORT:\n{clf\_report}")

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

print(f"Confusion Matrix: \n {confusion\_matrix(y\_train, pred)}\n")

elif train==False:

pred = clf.predict(X\_test)

clf\_report = pd.DataFrame(classification\_report(y\_test, pred, output\_dict=True))

print("Test Result:\n================================================")

print(f"Accuracy Score: {accuracy\_score(y\_test, pred) \* 100:.2f}%")

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

print(f"CLASSIFICATION REPORT:\n{clf\_report}")

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

print(f"Confusion Matrix: \n {confusion\_matrix(y\_test, pred)}\n")

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaled\_data = scaler.fit\_transform(XY)

print(XY)

print("scaled data" , scaled\_data)

scaled\_data=pd.DataFrame(scaled\_data)

print("check whether scaled data has null values",scaled\_data.isnull().sum())

scaled\_data.mean(axis = 0)

from sklearn.model\_selection import train\_test\_split

X=XY

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X,y,test\_size=0.3, stratify=y, random\_state=876)

from imblearn.over\_sampling import SMOTE

import collections

counter = collections.Counter(Y\_train)

print('Before', counter)

smt=SMOTE()

X\_train\_sm, Y\_train\_sm = smt.fit\_resample(X\_train,Y\_train)

print("x",type(X\_train\_sm))

print("y",type(Y\_train\_sm))

counter = collections.Counter(Y\_train\_sm)

print('After', counter)

print('y train sm',Y\_train\_sm)

########----MODEL BUILDING----########

print("########----MODEL BUILDING----########")

print("########----MODEL BUILDING----########")

print("---------------------------------------")

print("----------------------------------------")

print("########----LogisticRegression----########")

print("########-----⇩⇩⇩⇩⇩⇩⇩---########")

from sklearn.linear\_model import LogisticRegression

model = LogisticRegression()

model.fit(X\_train\_sm, Y\_train\_sm)

X\_train\_prediction = model.predict(X\_train\_sm)

from sklearn.metrics import accuracy\_score

train\_data\_accuracy = accuracy\_score(X\_train\_prediction,Y\_train\_sm)

print("LogisticRegression training data accuracy" , train\_data\_accuracy)

X\_test\_prediction = model.predict(X\_test)

test\_data\_accuracy = accuracy\_score(X\_test\_prediction, Y\_test)

print("LogisticRegression test data accuracy", test\_data\_accuracy)

print\_score(model, X\_train\_sm, Y\_train\_sm, X\_test, Y\_test, train=True)

print\_score(model, X\_train, Y\_train, X\_test, Y\_test, train=False)

print("---------------------------------------")

print("----------------------------------------")

print("########----RandomForestClassifier----########")

print("########-----⇩⇩⇩⇩⇩⇩⇩---########")

from sklearn.ensemble import RandomForestClassifier

modelrf= RandomForestClassifier(n\_estimators= 10, criterion="entropy")

modelrf.fit(X\_train\_sm, Y\_train\_sm)

X\_train\_prediction = modelrf.predict(X\_train\_sm)

train\_data\_accuracy = accuracy\_score(X\_train\_prediction,Y\_train\_sm)

print("RandomForestClassifier training data accuracy" , train\_data\_accuracy)

X\_test\_prediction = modelrf.predict(X\_test)

test\_data\_accuracy = accuracy\_score(X\_test\_prediction, Y\_test)

print("RandomForestClassifier test data accuracy" , test\_data\_accuracy)

print\_score(modelrf, X\_train\_sm, Y\_train\_sm, X\_test, Y\_test, train=True)

print\_score(modelrf, X\_train, Y\_train, X\_test, Y\_test, train=False)

print("---------------------------------------")

print("----------------------------------------")

print("########----AdaBoostClassifier----########")

print("########-----⇩⇩⇩⇩⇩⇩⇩---########")

from sklearn.ensemble import AdaBoostClassifier

model= AdaBoostClassifier(random\_state=96)

model.fit(X\_train\_sm, Y\_train\_sm)

X\_train\_prediction = model.predict(X\_train\_sm)

train\_data\_accuracy = accuracy\_score(X\_train\_prediction,Y\_train\_sm)

print("AdaBoostClassifier training data accuracy", train\_data\_accuracy)

X\_test\_prediction = model.predict(X\_test)

test\_data\_accuracy = accuracy\_score(X\_test\_prediction, Y\_test)

print("AdaBoostClassifier test data accuracy", test\_data\_accuracy)

print\_score(model, X\_train\_sm, Y\_train\_sm, X\_test, Y\_test, train=True)

print\_score(model, X\_train, Y\_train, X\_test, Y\_test, train=False)

print("---------------------------------------")

print("----------------------------------------")

print("########----SVM----########")

print("########-----⇩⇩⇩⇩⇩⇩⇩---########")

from sklearn import svm

model = svm.SVC(kernel='linear')

model.fit(X\_train\_sm, Y\_train\_sm)

X\_train\_prediction = model.predict(X\_train\_sm)

######

train\_data\_accuracy = accuracy\_score(X\_train\_prediction,Y\_train\_sm)

print("SVM training data accuracy", train\_data\_accuracy)

X\_test\_prediction = model.predict(X\_test)

test\_data\_accuracy = accuracy\_score(X\_test\_prediction, Y\_test)

print("SVM test data accuracy", test\_data\_accuracy)

print\_score(model, X\_train\_sm, Y\_train\_sm, X\_test, Y\_test, train=True)

print\_score(model, X\_train, Y\_train, X\_test, Y\_test, train=False)

filename = 'heart-disease-prediction-knn-model.pkl'

pickle.dump(modelrf, open(filename, 'wb'))