import numpy as np

import pandas as pd

from sklearn.impute import SimpleImputer

from sklearn.preprocessing import LabelEncoder

desired\_width = 320

pd.set\_option('display.width',desired\_width)

pd.set\_option('display.max\_columns',14)

heart = pd.read\_csv("/content/sample\_data/heart failur classification dataset.csv")

#heart.insull()

#printing 1st 7 rows

print(heart.head(7))

#printing total rows & column

print(heart.shape)

# To get the column info with missing data

print(heart.isnull().sum())

#Handling missing values

#print("Handling missing values:")

# Check how many values are missing in the serum\_sodium column

print("Number of rows with null values in serum\_sodium column: ", heart['serum\_sodium'].isnull().sum())

heart = heart[heart['serum\_sodium'].notnull()]

# Print out the shape of the heart

print("Shape after removing null values in serum\_sodium: ", heart.shape)

# Check how many values are missing in the time column

#print("Number of rows with null values in time column: ", heart['time'].isnull().sum())

heart = heart[heart['time'].notnull()]

print("Shape after removing null values in time: ", heart.shape)

#Encoding categorical features

#print("Encoding categorical features part:")

#print(heart.info())

#print(heart['sex'].unique())

# Set up the LabelEncoder object

enc = LabelEncoder()

# Apply the encoding to the "Accessible" column

heart['sex'] = enc.fit\_transform(heart['sex'])

# Compare the two columns

#print(heart[['sex', 'sex\_enc']].head(7))

#print(heart['smoking'].unique())

enc = LabelEncoder()

# Apply the encoding to the "Accessible" column

heart['smoking'] = enc.fit\_transform(heart['smoking'])

# Compare the two columns

#print(heart[['smoking', 'smoking\_enc']].head(7))

print(heart.head(33))

print(heart.isnull().sum())

#dropping column

#print("Scaling all the values between 0-1 with proper scaling technique:")

#from sklearn.datasets import load\_breast\_cancer

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

label = heart[['DEATH\_EVENT']]

features = heart.drop(axis=1, columns='DEATH\_EVENT')

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

scaler.fit(features)

feature\_scaled = scaler.transform(features)

feature\_scaled\_df = pd.DataFrame(feature\_scaled, columns=list(heart.columns)[:-1])

print(feature\_scaled\_df.head())

print(label.head())

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

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

#heart = heart\_scaling

#print(heart.head())

#Prepare the training set

# Perform classification and calculate accuracy using logistic regression

#heart = dataset

# X = feature values, all the columns except the   last column

X = heart.iloc[:,:-1].values

# y = target values,  last column of the data frame

y = heart.iloc[:,-1].values

#Split the data into 80% training and 20% testing

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

#Train the model

model = LogisticRegression()

model.fit(x\_train, y\_train) #Training the model

predictions = model.predict(x\_test)

print("prediction:",predictions)# printing predictions

accuracy\_score\_of\_regression = accuracy\_score( predictions,y\_test)

print("logistic regression accuracy:", accuracy\_score\_of\_regression)

#Perform classification and calculate accuracy using decision tree

from sklearn.tree import DecisionTreeClassifier

#from sklearn.metrics import accuracy\_score

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

X = heart.iloc[:, :-1].values

y = heart.iloc[:,-1].values

print(X.shape)

print("y",y.shape)

x\_train,x\_test,y\_train,y\_test = train\_test\_split(X,y,test\_size=0.2,random\_state=1)

clf = DecisionTreeClassifier(criterion='entropy',random\_state=1)

clf.fit(x\_train,y\_train)

y\_pred = clf.predict(x\_test)

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

print("Decission tree accuracy:",score)

#graphical representation

import matplotlib.pyplot as plt

#colors = {'logistic regression':'r', 'decision tree':'g'}

fig,ax = plt.subplots()

ax.bar(['logistic regression','decision tree'],[accuracy\_score\_of\_regression,score])

ax.set\_title('Comparison of logistic regression and decision tree')

ax.set\_xlabel("Method name")

ax.set\_ylabel('Accuracy')

plt.show()

print("svc:")

from sklearn.svm import SVC

svc = SVC(kernel="linear")

svc.fit(x\_train, y\_train)

svc\_predictions\_pre = svc.predict(x\_test)

print("prediction:",svc\_predictions\_pre)# printing predictions

accuracy\_score\_of\_pre\_svc = accuracy\_score(svc\_predictions\_pre,y\_test)

print("svc accuracy:", accuracy\_score\_of\_pre\_svc)

print("Training accuracy of the model is {:.2f}".format(svc.score(x\_train, y\_train)))

print("Testing accuracy of the model is {:.2f}".format(svc.score(x\_test, y\_test)))

print("### Neural Network Classifier/MLPClassifier")

from sklearn.neural\_network import MLPClassifier

nnc=MLPClassifier(hidden\_layer\_sizes=(7), activation="relu", max\_iter=10000)

nnc.fit(x\_train, y\_train)

nnc\_predictions\_pre = nnc.predict(x\_test)

print("predictions:",nnc\_predictions\_pre)

accuracy\_score\_of\_pre\_nnc = accuracy\_score(nnc\_predictions\_pre,y\_test)

print(accuracy\_score\_of\_pre\_nnc)

print("The Training accuracy of the model is {:.2f}".format(nnc.score(x\_train, y\_train)))

print("The Testing accuracy of the model is {:.2f}".format(nnc.score(x\_test, y\_test)))

#predictions = nnc.predict(x\_test)

#print(predictions)

print("RandomForestClassifier")

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n\_estimators=50)

rfc.fit(x\_train, y\_train)

rfc\_predictions\_pre = rfc.predict(x\_test)

print("predictions:",rfc\_predictions\_pre)

accuracy\_score\_of\_pre\_rfc = accuracy\_score(rfc\_predictions\_pre,y\_test)

print(accuracy\_score\_of\_pre\_rfc)

print("The Training accuracy of the model is {:.2f}".format(rfc.score(x\_train, y\_train)))

print("The Testing accuracy of the model is {:.2f}".format(rfc.score(x\_test, y\_test)))

print("pca")

from sklearn.decomposition import PCA

pca = PCA(n\_components = features.shape[1] // 2)

principal\_components= pca.fit\_transform(features.values)

print(principal\_components)

print("sum",sum(pca.explained\_variance\_ratio\_))

principal\_df = pd.DataFrame(data=principal\_components, columns=["component 1", "component 2","component 3","component 4","component 5","component 6"])

print(principal\_df.head())

main\_df = pd.concat([principal\_df, label], axis=1)

print("main\_df:",main\_df.head())

print("After implementing PCA")

X=principal\_df.values

y=label.values

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X,y, test\_size=0.2, random\_state=42)

print("svcpost")

from sklearn.svm import SVC

svc = SVC(kernel="linear")

svc.fit(x\_train, y\_train.ravel())

svc\_predictions\_post = svc.predict(x\_test)

print("prediction:",svc\_predictions\_post)# printing predictions

accuracy\_score\_of\_post\_svc = accuracy\_score(svc\_predictions\_post,y\_test)

print("svc accuracy:", accuracy\_score\_of\_post\_svc)

print("Training accuracy of the model is {:.2f}".format(svc.score(x\_train, y\_train)))

print("Testing accuracy of the model is {:.2f}".format(svc.score(x\_test, y\_test)))

print("### Neural Network Classifier/MLPClassifier")

from sklearn.neural\_network import MLPClassifier

nnc=MLPClassifier(hidden\_layer\_sizes=(7), activation="relu", max\_iter=10000)

nnc.fit(x\_train, y\_train)

nnc\_predictions\_post = nnc.predict(x\_test)

print("predictions:",nnc\_predictions\_pre)

accuracy\_score\_of\_post\_nnc = accuracy\_score(nnc\_predictions\_post,y\_test)

print("The Training accuracy of the model is {:.2f}".format(nnc.score(x\_train, y\_train)))

print("The Testing accuracy of the model is {:.2f}".format(nnc.score(x\_test, y\_test)))

#predictions = nnc.predict(x\_test)

#print(predictions)

print("RandomForestClassifier:")

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n\_estimators=50)

rfc.fit(x\_train, y\_train.ravel())

rfc\_predictions\_post = rfc.predict(x\_test)

print("predictions:",rfc\_predictions\_post)

accuracy\_score\_of\_post\_rfc = accuracy\_score(rfc\_predictions\_post,y\_test)

print("The Training accuracy of the model is {:.2f}".format(rfc.score(x\_train, y\_train)))

print("The Testing accuracy of the model is {:.2f}".format(rfc.score(x\_test, y\_test)))

import matplotlib.pyplot as plt

#colors = {'logistic regression':'r', 'decision tree':'g'}

fig,ax = plt.subplots()

ax.bar(['accuracy score of pre-pca svc','accuracy score of post-pca svc'],[accuracy\_score\_of\_pre\_svc,accuracy\_score\_of\_post\_svc])

ax.set\_title('Comparison of pre-PCA and post-PCA result of SVC part')

ax.set\_xlabel("Method name")

ax.set\_ylabel('Accuracy')

plt.show()

import matplotlib.pyplot as plt

#colors = {'logistic regression':'r', 'decision tree':'g'}

fig,ax = plt.subplots()

ax.bar(['accuracy score of pre-pca nnc','accuracy score of  post-pca nnc'],[accuracy\_score\_of\_pre\_nnc,accuracy\_score\_of\_post\_nnc])

ax.set\_title('Comparison of pre-PCA and post-PCA result of MLPClassifier part')

ax.set\_xlabel("Method name")

ax.set\_ylabel('Accuracy')

plt.show()

import matplotlib.pyplot as plt

#colors = {'logistic regression':'r', 'decision tree':'g'}

fig,ax = plt.subplots()

ax.bar(['accuracy score of pre-pca rfc','accuracy score of post-pca rfc'],[accuracy\_score\_of\_pre\_rfc,accuracy\_score\_of\_post\_rfc])

ax.set\_title('Comparison of pre-PCA and post-PCA result of RandomForestClassifier part')

ax.set\_xlabel("Method name")

ax.set\_ylabel('Accuracy')

plt.show()