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('E:\SEMESTER\Summer- 21\CSE422 ARTIFICIAL INTELLIGENCE\Lab\lab-05\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'] = 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'] = 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

heart = heart.drop(['sex'], axis = 1)

heart = heart.drop(['smoking'], axis = 1)

#print("After dropping time column the shape is: ",heart.shape)

#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

heart\_scaling = heart

X\_train, X\_test, y\_train, y\_test = train\_test\_split(heart\_scaling, heart\_scaling.values, random\_state=1)

print(X\_train.shape)

print(X\_test.shape)

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

scaler.fit(X\_train)

# transform data

X\_train\_scaled = scaler.transform(X\_train)

# transform test data

X\_test\_scaled = scaler.transform(X\_test)

#print(X\_test\_scaled)

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

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

#Prepare the training set

# Perform classification and calculate accuracy using logistic regression

#heart = dataset

# X = feature values, all the columns except the 3rd last column

X = heart.iloc[:, :-3].values

# y = target values, 3rd last column of the data frame

y = heart.iloc[:, -3].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(y\_test, predictions)

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[:, :-3].values

y = heart.iloc[:,-3].values

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()