Import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings("ignore")

df = pd.read\_csv("/content/heart.csv")

print("The shape of the dataset is : ", df.shape)

df.head()

dict = {}

for i in list(df.columns):

    dict[i] = df[i].value\_counts().shape[0]

pd.DataFrame(dict,index=["unique count"]).transpose()

cat\_cols = ['sex','exng','caa','cp','fbs','restecg','slp','thall']

con\_cols = ["age","trtbps","chol","thalachh","oldpeak"]

target\_col = ["output"]

print("The categorial cols are : ", cat\_cols)

print("The continuous cols are : ", con\_cols)

print("The target variable is :  ", target\_col)

df[con\_cols].describe().transpose()

df.isnull().sum()

df\_corr = df[con\_cols].corr().transpose()

df\_corr

fig = plt.figure(figsize=(10,10))

gs = fig.add\_gridspec(1,1)

gs.update(wspace=0.3, hspace=0.15)

ax0 = fig.add\_subplot(gs[0,0])

color\_palette = ["#5833ff","#da8829"]

mask = np.triu(np.ones\_like(df\_corr))

ax0.text(1.5,-0.1,"Correlation Matrix",fontsize=22, fontweight='bold', fontfamily='serif', color="#000000")

df\_corr = df[con\_cols].corr().transpose()

sns.heatmap(df\_corr,mask=mask,fmt=".1f",annot=True,cmap='YlGnBu')

plt.show()

# Scaling

from sklearn.preprocessing import RobustScaler

# Train Test Split

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

# Models

import torch`

import torch.nn as nn

from sklearn.svm import SVC

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import GradientBoostingClassifier

# Metrics

from sklearn.metrics import accuracy\_score, classification\_report, roc\_curve

# Cross Validation

from sklearn.model\_selection import cross\_val\_score

from sklearn.model\_selection import GridSearchCV

print('Packages imported...')

# creating a copy of df

df1 = df

# define the columns to be encoded and scaled

cat\_cols = ['sex','exng','caa','cp','fbs','restecg','slp','thall']

con\_cols = ["age","trtbps","chol","thalachh","oldpeak"]

# encoding the categorical columns

df1 = pd.get\_dummies(df1, columns = cat\_cols, drop\_first = True)

# defining the features and target

X = df1.drop(['output'],axis=1)

y = df1[['output']]

# instantiating the scaler

scaler = RobustScaler()

# scaling the continuous featuree

X[con\_cols] = scaler.fit\_transform(X[con\_cols])

print("The first 5 rows of X are")

X.head()

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

print("The shape of X\_train is      ", X\_train.shape)

print("The shape of X\_test is       ",X\_test.shape)

print("The shape of y\_train is      ",y\_train.shape)

print("The shape of y\_test is       ",y\_test.shape)

# instantiating the object and fitting

clf = SVC(kernel='linear', C=1, random\_state=42).fit(X\_train,y\_train)

# predicting the values

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

# printing the test accuracy

print("The test accuracy score of SVM is ", accuracy\_score(y\_test, y\_pred))

# instantiating the object

svm = SVC()

# setting a grid - not so extensive

parameters = {"C":np.arange(1,10,1),'gamma':[0.00001,0.00005, 0.0001,0.0005,0.001,0.005,0.01,0.05,0.1,0.5,1,5]}

# instantiating the GridSearchCV object

searcher = GridSearchCV(svm, parameters)

# fitting the object

searcher.fit(X\_train, y\_train)

# the scores

print("The best params are :", searcher.best\_params\_)

print("The best score is   :", searcher.best\_score\_)

# predicting the values

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

# printing the test accuracy

print("The test accuracy score of SVM after hyper-parameter tuning is ", accuracy\_score(y\_test, y\_pred))

# instantiating the object

logreg = LogisticRegression()

# fitting the object

logreg.fit(X\_train, y\_train)

# calculating the probabilities

y\_pred\_proba = logreg.predict\_proba(X\_test)

# finding the predicted valued

y\_pred = np.argmax(y\_pred\_proba,axis=1)

# printing the test accuracy

print("The test accuracy score of Logistric Regression is ", accuracy\_score(y\_test, y\_pred))