**CARDIOVASCULAR DISEASE PREDICTION**

Submitted by

Hansa A S – 71762133015,

Giridhara Vijay R A – 71762133013

M Sc Decision and Computing Sciences III rd year

Submitted to

Dr. N. Yamuna Devi

Department of Decision and Computing Sciences



**COIMBATORE INSTITUTE OF TECHNOLOGY**

**(Government Aided Autonomous Institution)**

**Coimbatore-641014**

**December – 2023**

1. **Abstract**

The main objective of our project is to predict if a person has a chance of getting a cardiovascular disease based on the person’s medical examination readings (Age, BMI , Blood pressure , Cholesterol etc). This is a project for predicting potential cardiovascular disease. The significance of heart disease is increasing as the population ages. Existing modes of diagnosis is typically slow and may have undesirable side effects.

The data collected from patients during medical examination contain some hidden information, which is very useful for making effective decisions. For enhancing the results and making efficient decisions on this big data, some advanced ML techniques are used. We develop an efficient cardiovascular disease prediction using ML and DL models for predicting the risk level of cardiovascular disease. The model uses 11 medical parameters such as age, gender, blood pressure, cholesterol, for prediction. The model predicts the likelihood of patients getting heart disease. It derives significant patterns and relationships between the input features and learns the features. After the model is trained it is then used for predicting for the testing data. The obtained results have illustrated that the model can effectively predict the risk level of cardiovascular diseases.

1. **EDA ( dataset description)**

Exploratory Data Analysis (EDA) is a crucial step in understanding and preparing the dataset for building a machine learning model. Below is an example of how you might approach EDA for a dataset related to cardiovascular prediction.

**Dataset Overview**

- To Understand the structure of the dataset, including the number of instances and features.

- Identifying the target variable, which is likely to be the presence or absence of cardiovascular disease.

**Data Types and Missing Values:**

- Checking the data types of each feature (numeric, categorical, etc.).

- To Identify and handle missing values appropriately (imputation or removal).

**Descriptive Statistics**

- Calculating basic descriptive statistics (mean, median, standard deviation, etc.) for numerical features.

- Exploring the distribution of the target variable.

**Univariate Analysis:**

- Examine the distribution of each feature individually.

- Using histograms, box plots, or kernel density plots to visualize the spread of numerical features.

- For categorical features, plot bar charts to show the distribution of different categories.

**Bivariate Analysis:**

- The relationships between pairs of features and the target variable.

- Using scatter plots for numerical features against the target variable.

- By cross-tabulations or heatmaps for categorical features.

**Correlation Analysis:**

- To Calculate the correlation matrix for numerical features.

- Visualizing the correlation matrix using a heatmap.

- By Identifying strong correlations between features, especially with the target variable.

**Outlier Detection:**

- Identify and handle outliers in the dataset, particularly in numerical features.

- Use box plots or scatter plots to visualize outliers.

**Data Visualization**

- Utilize various plots and charts to convey insights, such as pie charts for categorical distributions, and line plots for time-based trends.

**Dimensionality Reduction**

- If dealing with high-dimensional data, explore dimensionality reduction techniques (e.g., PCA) to visualize data in lower dimensions.

By conducting a comprehensive EDA, you gain insights into the dataset's characteristics, identify potential challenges, and make informed decisions on data preprocessing steps before building the machine learning model for autism prediction.

1. **ML techniques**

**Outlier Analysis in cardiovascular prediction:**

Sorting: Arrange the cardiovascular prediction dataset based on relevant features, which can help in identifying patterns or potential outliers.

Missing Values: Examine the dataset for missing values in features crucial. Decide on appropriate strategies for imputing or handling missing data to maintain the integrity of the analysis.

Removal of Outliers: Based on the IQR analysis or other methods, consider removing outliers from the dataset to prevent them from disproportionately influencing the predictive model.

**Logistic Regression in Cardiovascular Prediction:**

Given that cardiovascular prediction is often a binary classification problem, logistic regression is suitable for estimating the probability of an individual having cardiovascular

**Support Vector Machines (SVM):**

SVM is used for classification tasks. It's effective for identifying patterns in data and separating classes. SVM can be employed to classify individuals into cardiovascular and non-cardiovascular groups based on features such as demographic information.

**Random Forest:**

Random Forest is an ensemble learning method that builds multiple decision trees and merges their predictions. It is robust and can handle complex relationships in the data. Building a Random Forest model to predict autism based on a combination of various features.

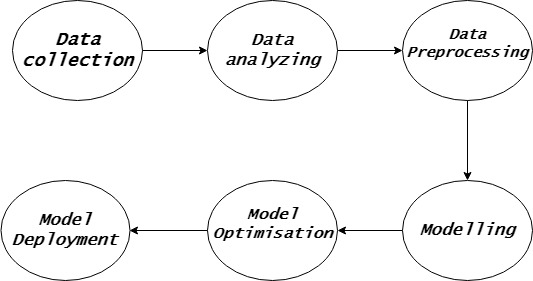
**KNN (Clustering):**

K-Nearest Neighbors (KNN) finds utility in cardiovascular disease prediction by enabling risk stratification, early detection, and personalized medicine. It contributes to decision support systems, integrates with electronic health records, aids in predictive modeling and feature selection, and facilitates continuous monitoring. Additionally, KNN supports research in population studies for public health initiatives.

**Artificial Neural Networks (ANN):**

ANNs are a class of machine learning models inspired by the human brain. They excel in capturing complex relationships in data. Training an ANN to predict cardiovascular disease using layers of interconnected nodes, with each node representing a feature or a combination of features.

1. **FLOWCHART :**



1. **Code**

import numpy as np

import pandas as pd

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

from sklearn.preprocessing import StandardScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from sklearn import svm,model\_selection

from sklearn.linear\_model import LogisticRegression

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

from sklearn.pipeline import Pipeline

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

from sklearn.neighbors import KNeighborsClassifier

from sklearn.preprocessing import StandardScaler

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import BaggingClassifier

from sklearn.neural\_network import MLPClassifier

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.metrics import average\_precision\_score,accuracy\_score,confusion\_matrix,roc\_curve,f1\_score,auc,precision\_recall\_curve

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.preprocessing import LabelEncoder,OneHotEncoder

from numpy import array

df=pd.read\_csv("C:\\Users\\19295\\Downloads\\cardio\_train.csv")

df['age']=(df['age']/365).round().astype('int')

df.drop(['id'],axis=1,inplace=True)

df['BMI'] = df['weight']/((df['height']/100)\*\*2)

df.head()

a=sns.countplot(data=df, x=df.age, hue=df.cardio)

df\_long = pd.melt(df, id\_vars=['cardio'], value\_vars=['cholesterol','gluc', 'smoke', 'alco', 'active'])

sns.catplot(x="variable", hue="value", col="cardio",

                data=df\_long, kind="count");

sns.boxplot(x='cardio',y='weight',data=df,palette='winter')

from sklearn.impute import SimpleImputer

train = df[["ap\_hi", "ap\_lo","weight","height","cardio"]]

def impute\_outliers(data, column, factor):

    q1 = data[column].quantile(0.25)

    q3 = data[column].quantile(0.75)

    iqr = q3 - q1

    lower\_bound = q1 - factor \* iqr

    upper\_bound = q3 + factor \* iqr

    data\_copy = data.copy()

    data\_copy[column] = np.where(data\_copy[column] < lower\_bound, np.nan, data\_copy[column])

    data\_copy[column] = np.where(data\_copy[column] > upper\_bound, np.nan, data\_copy[column])

    imputer = SimpleImputer(strategy="mean")

    data\_imputed = imputer.fit\_transform(data\_copy)

    return pd.DataFrame(data\_imputed, columns=data.columns)

for column in ["ap\_hi", "ap\_lo","weight","height","cardio"]:

    train = impute\_outliers(train, column, 1.5)

df[["ap\_hi", "ap\_lo","weight","height","cardio"]] = train

sns.boxplot(train)

plt.show()

df.isnull().values.any()

df.describe()

df[df.ap\_lo>df.ap\_hi]

df.sort\_values('BMI')

values = array(df['cholesterol'])

label\_encoder = LabelEncoder()

integer\_encoded = label\_encoder.fit\_transform(values)

onehot\_encoder = OneHotEncoder(sparse=False)

integer\_encoded = integer\_encoded.reshape(len(integer\_encoded), 1)

onehot\_encoded = onehot\_encoder.fit\_transform(integer\_encoded)

a=[]

b=[]

c=[]

for i in onehot\_encoded:

  a.append(i[0])

  b.append(i[1])

  c.append(i[2])

df['chol1'] = a

df['chol2']=b

df['chol3']=c

df.drop(['cholesterol'],axis=1,inplace=True)

gluc=pd.get\_dummies(df['gluc'],prefix='gluc')

df=pd.concat([df,gluc],axis=1)

df.drop(['gluc'],axis=1,inplace=True)

df['BMI']=df['BMI'].round().astype('int')

df['chol1']=df['chol1'].astype('int')

df['chol2']=df['chol2'].astype('int')

df['chol3']=df['chol3'].astype('int')

cols=df.columns.drop(['cardio'])

cols

df.drop(df[df.duplicated(cols)].index,inplace=True)

df.count()

df['gender'].replace(2,0,inplace=True)

down=df[(df.cardio==1)&(df.active==1)&(df.smoke==0)&(df.gluc\_1==1)&(df.BMI<25)&(80<df.ap\_lo)&(df.ap\_lo<90)&(120<df.ap\_hi)&(df.ap\_hi<140)&(df.chol1==1)].index

df.drop(down,inplace=True)

down2=df[(df.cardio==0)&(df.active==0)&(df.gluc\_3==1)&(df.BMI>30)&((90<df.ap\_lo)|(df.ap\_lo<80))&((140<df.ap\_hi)|(df.ap\_hi<120))&(df.chol3==1)].index

df.drop(down2,inplace=True)

df[(df.cardio==0)&(df.active==0)&(df.BMI>35)&((90<df.ap\_lo)|(df.ap\_lo<80))&((140<df.ap\_hi)|(df.ap\_hi<120))&((df.chol3==1)|(df.gluc\_3==1))]

df.drop(df[(df.BMI>40)&(df.cardio==0)].index,inplace=True)

df.groupby('gender').count()

down3=df[(df.gluc\_3==1)&(df.chol3==1)&(df.cardio==0)&((df.ap\_hi>140)|(df.ap\_hi<120))].index

df.drop(down3,inplace=True)

predictors = final.drop(["cardio",'BMI'],axis=1)

target = final['cardio']

X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(predictors,target,test\_size=0.2,random\_state=1,stratify=target)

X\_train, X\_val, Y\_train, Y\_val = train\_test\_split(X\_train, Y\_train, test\_size=0.2, random\_state=1,stratify=Y\_train)

print(Y\_train.mean(),Y\_test.mean())

from sklearn.preprocessing import StandardScaler

sc\_X=StandardScaler()

X\_train=sc\_X.fit\_transform(X\_train)

X\_val=sc\_X.fit\_transform(X\_val)

X\_test=sc\_X.fit\_transform(X\_test)

clf=LogisticRegression()

clf.fit(X\_train,Y\_train)

pred=clf.predict(X\_val)

print('f score ',f1\_score(pred,Y\_val))

print(clf.score(X\_val,Y\_val))

print(clf.score(X\_test,Y\_test))

from sklearn.metrics import f1\_score

import sklearn.metrics as met

from sklearn.svm import LinearSVC

clf\_svc = LinearSVC()

#clf\_svc = svm.SVC(verbose=1)

clf\_svc.fit(X\_train, Y\_train)

print('score for linearSVC ',clf\_svc.score(X\_train,Y\_train))

knn\_scores = []

for k in range(1,21):

    knn\_classifier = KNeighborsClassifier(n\_neighbors = k)

    knn\_classifier.fit(X\_train, Y\_train)

    knn\_scores.append(knn\_classifier.score(X\_test, Y\_test))

plt.plot([k for k in range(1, 21)], knn\_scores, color = 'red')

for i in range(1,21):

    plt.text(i, knn\_scores[i-1], (i, knn\_scores[i-1].round(3)))

plt.xticks([i for i in range(1, 21)])

plt.xlabel('Number of Neighbors (K)')

plt.ylabel('Scores')

plt.title('K Neighbors Classifier scores for different K values')

the max accuracy achieved is 83.8% at k=20

from sklearn.ensemble import RandomForestClassifier

depth = 10  #@param {type: "slider", min: 0, max: 100}

estimators = 201  #@param {type: "slider", min: 100, max: 1000}

rf = RandomForestClassifier(n\_estimators=estimators,n\_jobs=-1,random\_state=0,max\_depth=depth)

rf.fit(X\_train, Y\_train)

print("Accuracy on training set: {:.3f}".format(rf.score(X\_train, Y\_train)))

print("Accuracy on val set: {:.3f}".format(rf.score(X\_val, Y\_val)))

print("Accuracy on test set: {:.3f}".format(rf.score(X\_test, Y\_test)))

from sklearn.tree import DecisionTreeClassifier

#@title max depth

#@markdown max depth.

depth = 7  #@param {type: "slider", min: 0, max: 100}

tree = DecisionTreeClassifier(max\_depth=depth,random\_state=0)

tree.fit(X\_train, Y\_train)

print("Accuracy on training set: {:.3f}".format(tree.score(X\_train, Y\_train)))

print("Accuracy on val set: {:.3f}".format(tree.score(X\_val, Y\_val)))

print("Feature importances:\n{}".format(tree.feature\_importances\_))

dt\_scores = []

for i in range(1,10):

    dt\_classifier = DecisionTreeClassifier(max\_features = i, random\_state = 0)

    dt\_classifier.fit(X\_train, Y\_train)

    dt\_scores.append(dt\_classifier.score(X\_test, Y\_test))

plt.plot([i for i in range(1, 10)], dt\_scores, color = 'green')

for i in range(1, 10):

    plt.text(i, dt\_scores[i-1], (i, dt\_scores[i-1]))

plt.xticks([i for i in range(1, 10)])

plt.xlabel('Max features')

plt.ylabel('Scores')

plt.title('Decision Tree Classifier scores for different number of maximum features')

def plotting(true,pred):

    fig,ax=plt.subplots(1,2,figsize=(10,5))

    precision,recall,threshold = precision\_recall\_curve(true,pred[:,1])

    ax[0].plot(recall,precision,'g--')

    ax[0].set\_xlabel('Recall')

    ax[0].set\_ylabel('Precision')

    ax[0].set\_title("Average Precision Score : {}".format(average\_precision\_score(true,pred[:,1])))

    fpr,tpr,threshold = roc\_curve(true,pred[:,1])

    ax[1].plot(fpr,tpr)

    ax[1].set\_title("AUC Score is: {}".format(auc(fpr,tpr)))

    ax[1].plot([0,1],[0,1],'k--')

    ax[1].set\_xlabel('False Positive Rate')

    ax[1].set\_ylabel('True Positive Rate')

plt.figure()

plotting(Y\_test,gaussian.predict\_proba(X\_test))

# Initialize the ANN model

model = Sequential()

# Add layers to the model

model.add(Dense(128, activation='relu', input\_shape=(X\_train.shape[1],)))

model.add(Dense(64, activation='relu'))

model.add(Dense(1, activation='sigmoid'))  # For binary classification (sigmoid for output layer)

# Compile the model

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

# Train the model

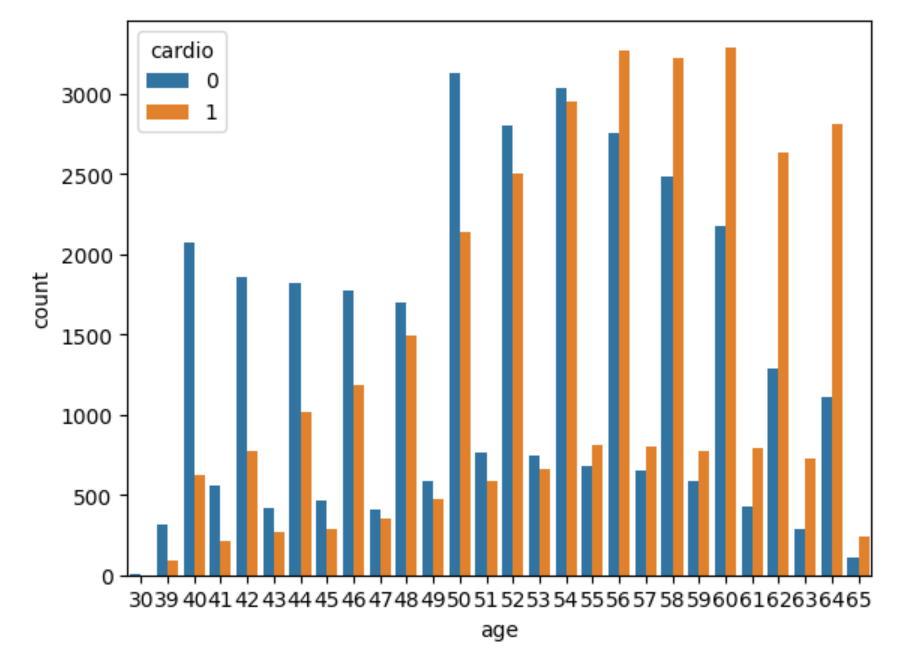
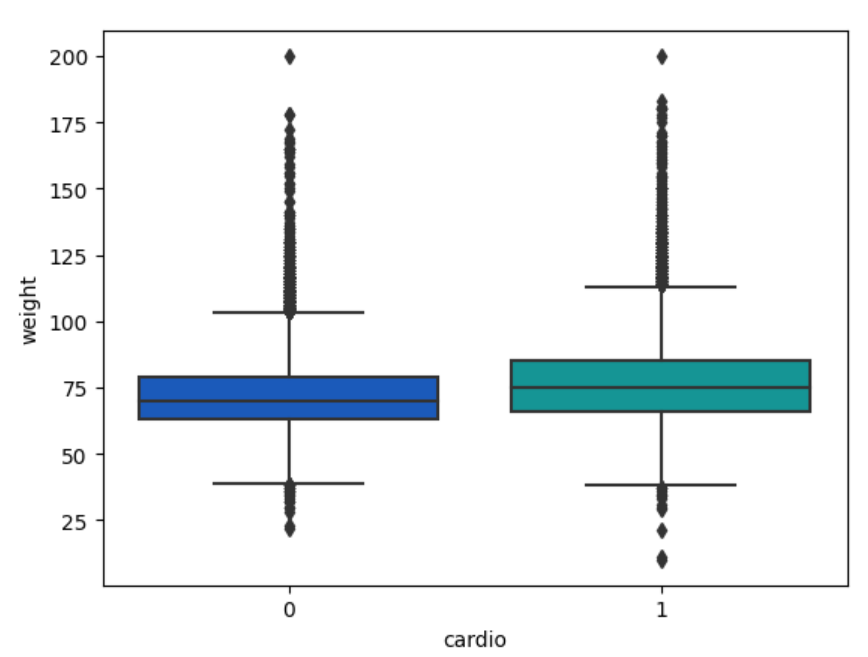
model.fit(X\_train, Y\_train, epochs=10, batch\_size=32, validation\_data=(X\_test, Y\_test))

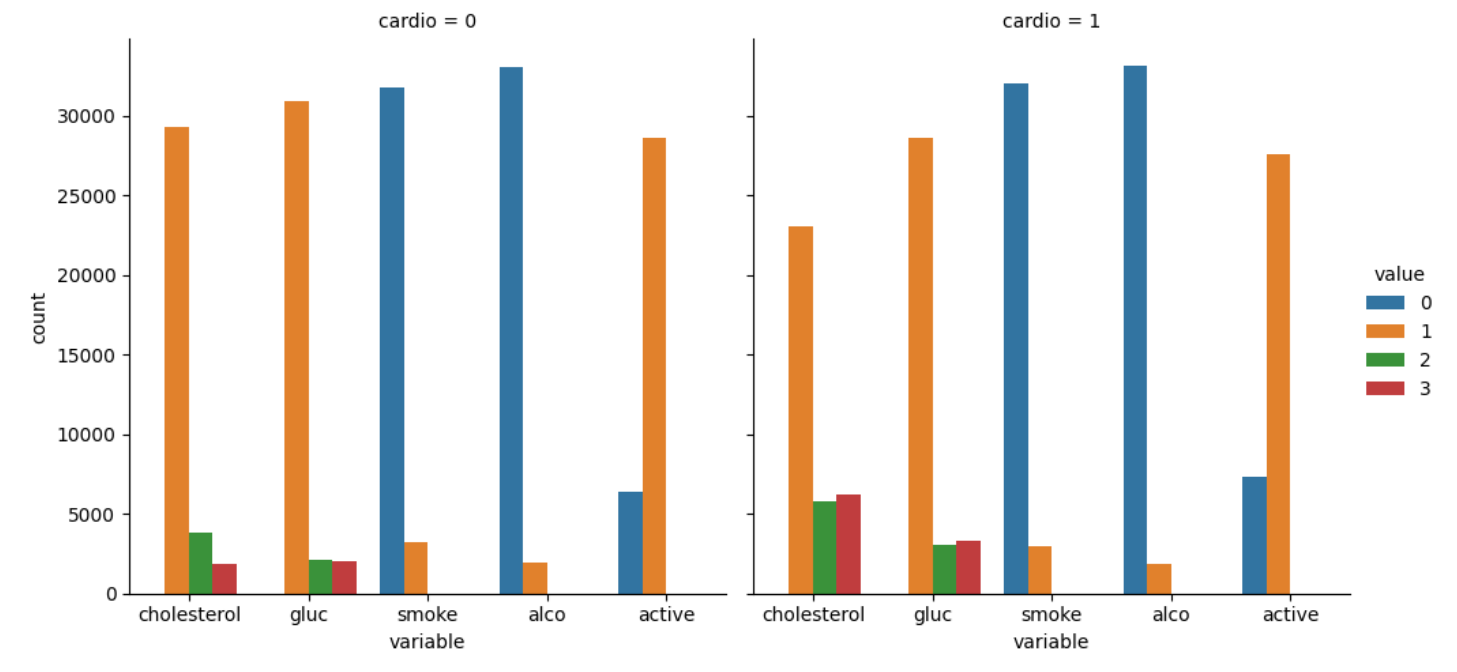
# Evaluate the model

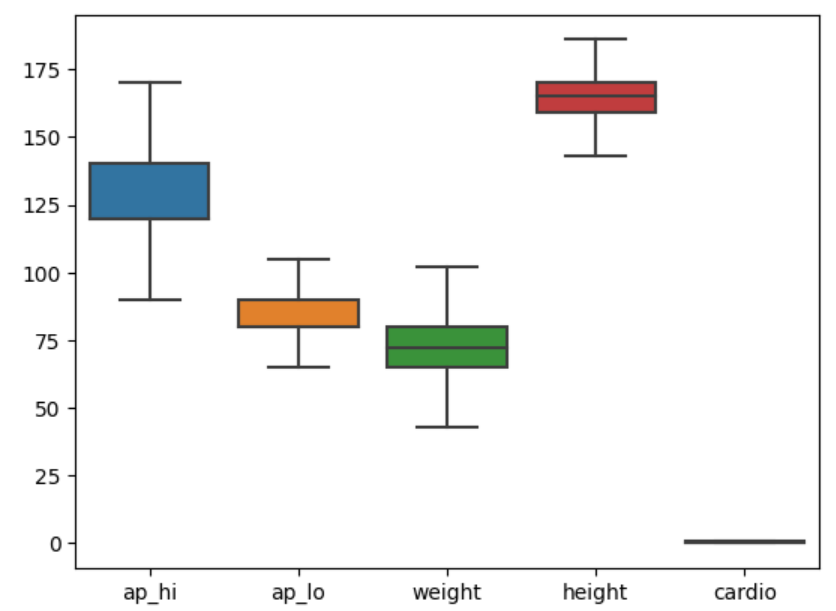
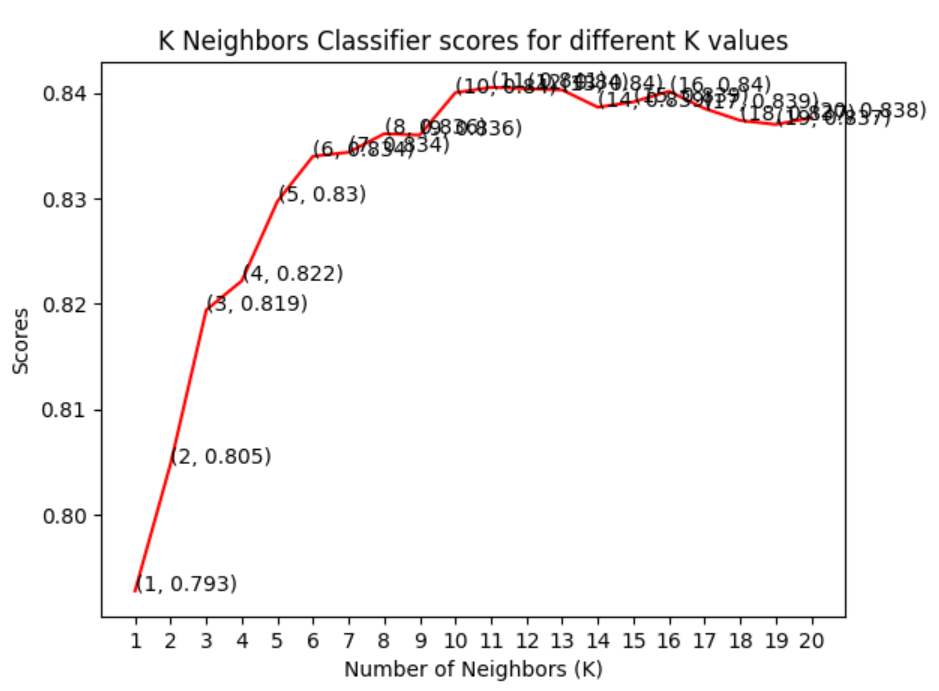
loss, accuracy = model.evaluate(X\_test, Y\_test)

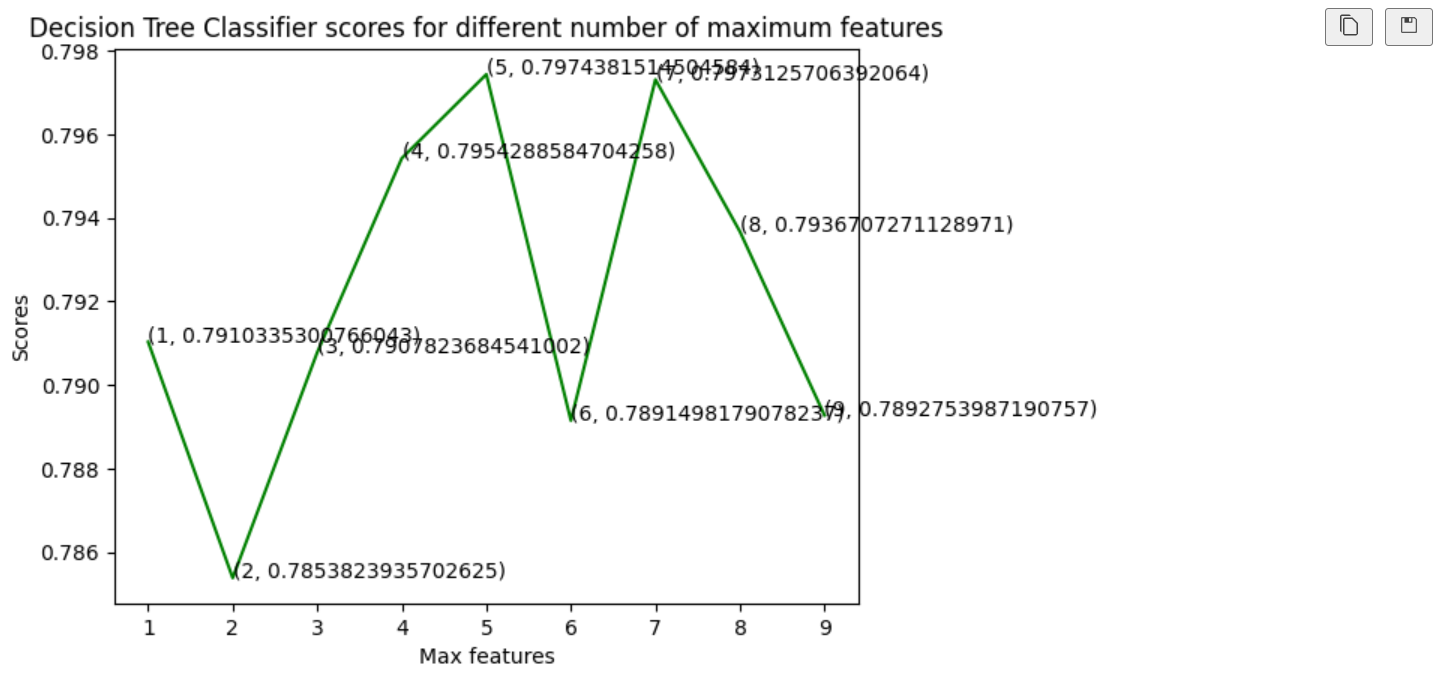
print(f'Test Accuracy: {accuracy \* 100:.2f}%')

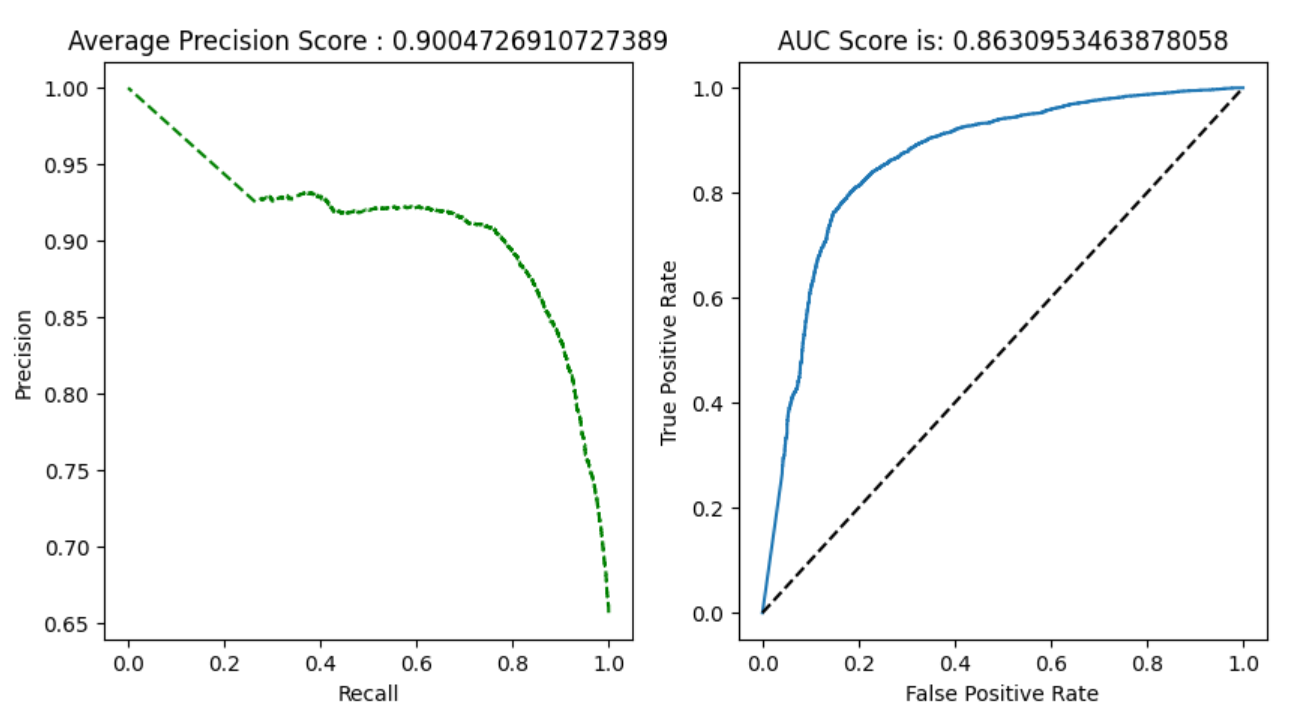
1. **Maps ( Results)**

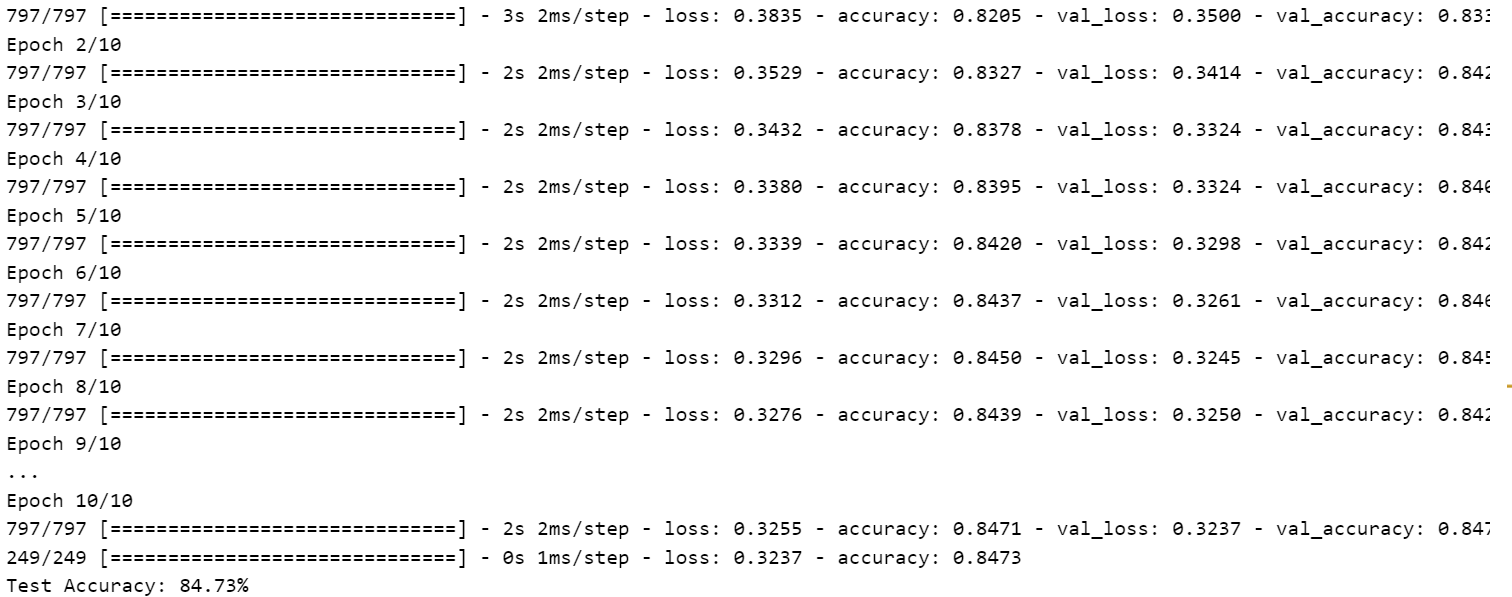
** **

****

** **

****

****

****

**6. Conclusion**

In conclusion, the machine learning predicting project for cardiovascular disease using Logistic Regression, SVM, Random Forest Classifier, Decision Tree Classifier, and Artificial Neural Networks (ANN) showcases the versatility of these algorithms. Each model has its strengths, with Logistic Regression offering interpretability, SVM providing robustness in complex datasets, Random Forest excelling in ensemble learning, Decision Tree offering simplicity, and ANN demonstrating deep learning capabilities. The comprehensive evaluation of these models enables informed choices based on performance metrics and specific project requirements, fostering advancements in cardiovascular disease prediction.