CARDIOVASCULAR DISEASE PREDICTION

By

ABHISHEK TYAGI

HARSHWARDHAN SINGH

POST-GRADUATION DIPLOMA IN BIG DATA ANALYTICS

PRN: - 230320525003

PRN: - 230320525015

CENTER FOR DEVELOPMENT OF ADVANCED COMPUTING (NOIDA)

CERTIFICATE

I hereby certify that the project work titled "**Cardiovascular Disease Prediction Using Machine Learning**" has been satisfactorily completed by Abhishek Tyagi, Harshwardhan Singh Thakur, Khushboo Yadav, and Prankur, under the guidance of **Mrs. Priti Bhardwaj**, as part of the PG-DBDA program at CDAC Noida.

To the best of my knowledge, this report has not been submitted for any other examination and does not form a part of any other course undertaken by the candidate. I have no doubts about their excellent research potential.

I wish them all the best for their future endeavours.

Project Guide

Priti Bhardwaj

ACKNOWLEDGEMENT

We express our sincere gratitude to all those who contributed to the successful completion of our project, "**Cardiovascular Disease Prediction Using Machine Learning**", under the guidance of Mrs. Priti Bhardwaj. we are immensely grateful to her for inspiring and assisting us in the project, personally reviewing our work, and providing encouragement throughout. Additionally, we extend our heartfelt thanks to all the faculty members of the PG-DBDA department for their unwavering guidance and support throughout the project, both in the challenging and smooth phases. We also appreciate the assistance provided by CDAC, Noida. We take this moment to thank our peer for their support and appreciation, which. we value the most.

ABSTRACT

In modern times, health issues are on the rise due to various factors such as lifestyle and genetics. Heart disease has become more prevalent and poses a serious threat to people's well-being. The normal values for blood pressure, cholesterol, and pulse rate differ for everyone. Medically verified standards indicate that normal blood pressure is 120/90, cholesterol should be between 100-129 mg/dL, pulse rate should be 72, fasting blood sugar level is 100 mg/dL, heart rate should be between 60-100 bpm, ECG results should be normal, and the width of major vessels should range from 25 mm (1 inch) in the aorta to only 8 mm in the capillaries. This report examines various classification techniques that are used to predict the risk level of each person based on their age, gender, blood pressure, cholesterol, and pulse rate.

The “Disease Prediction” system employs predictive modelling to determine a user’s potential illness based on the symptoms and provide input. the system evaluates the input symptoms and provides the probability of the disease. six techniques- Naïve Bayes, KNN, Decision tree, Logistic regression, Random Forest, and support vector Machine Algorithm are used for the prediction, these techniques compute the probability of the disease,

INDEX

1) INTRODUCTION

Numerous daily factors have the potential to impact our cardiac health. Given the discovery of new heart conditions, it is critical to prioritize the care of this vital organ responsible for blood circulation throughout our bodies. Both personal and occupational habits, as well as inherited genetic predispositions, can influence our heart's well-being. Unfortunately, millions worldwide fall victim to cardiovascular ailments annually. Cardiovascular disease is an umbrella term encompassing various conditions that specifically impact the heart and arteries. Even those in their 20s and 30s may be susceptible to cardiac issues due to unhealthy dietary habits, insufficient sleep, stress, depression, obesity, hypertension, high blood pressure, high blood cholesterol, or smoking. Maintaining a healthy heart requires proactive measures such as regular physical activity, a balanced diet, and avoiding harmful habits.

Heart disease symptoms can vary depending on the type of discomfort experienced by an individual. Some symptoms may not be easily recognizable, but common symptoms include chest pain, breathlessness, and heart palpitations. Angina, also known as angina pectoris, is a common type of chest pain in many forms of heart disease and is caused by a lack of oxygen in a part of the heart. Physical exertion or stressful events can trigger angina, which usually lasts less than ten minutes. Sometimes, heart attack symptoms may resemble indigestion, including heartburn, stomach-ache, and a heavy feeling in the chest. Other heart attack symptoms include pain that travels through the body, such as from the chest to the arms, neck, back, abdomen, or jaw, light-headedness and dizziness, profuse sweating, nausea, and vomiting.

Data Mining is a powerful technique for extracting crucial decision-making information from records for future analysis or prediction. It can unearth hidden patterns that are otherwise unidentifiable. Medical data mining integrates classification techniques and computerized training on datasets to discover patterns in medical data sets. This aids in predicting a patient's future state and analysing their medical history, which is essential for clinical analysis.

The objective of our project is to develop a robust model for predicting the occurrence of heart disease. To achieve this, a comparative analysis was conducted using six commonly used machine learning algorithms, namely Naïve Bayes, Decision Tree, KNN, Logistic Regression, Support Vector Machine, and Random Forest, XgBoost. However, since the accuracy of heart disease prediction is of utmost importance, all the algorithms were evaluated at various levels and using different evaluation strategies. By doing so, the project aims to provide valuable insights to researchers and medical practitioners, thereby enabling them to gain a better understanding of the most reliable and accurate method for predicting heart disease.

1.1 What is heart disease?

Heart disease is a complex medical condition that refers to a range of conditions that affect the heart and its associated blood vessels. These conditions may include coronary artery disease, arrhythmias, congenital heart defects, and other related blood vessel diseases. Cardiovascular disease is a term that is often used interchangeably with heart disease and generally refers to conditions that involve narrowed or blocked blood vessels that can lead to chest pain, heart attack, or stroke. Other heart conditions that affect the heart's muscle, valves, or rhythm are also considered forms of heart disease. Heart failure is a severe and prevalent condition that affects a significant percentage of the adult population in developed countries and is the leading cause of admission to healthcare professionals. It is a chronic and progressive condition that occurs when the heart muscle is unable to pump enough blood to meet the body's needs. Heart failure can cause a range of symptoms, including shortness of breath, fatigue, and swelling in the legs and ankles. It is associated with high medical costs, reaching up to 2% of the total health costs in developed countries.

2) DATA COLLECTION AND DATA VISULIZATION: -

2.1) View of Dataset:

```
df= pd.read_csv('data_cardiovascular_risk.csv', sep= ',' , encoding= ' utf-8')
df.head(15)
```

	id	age	education	sex	is_smoking	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	CHD
0	0	64	2.0	F	YES	3.0	0.0	0	0	0	221.0	148.0	85.0	NaN	90.0	80.0	1
1	1	36	4.0	M	NO	0.0	0.0	0	1	0	212.0	168.0	98.0	29.77	72.0	75.0	0
2	2	46	1.0	F	YES	10.0	0.0	0	0	0	250.0	116.0	71.0	20.35	88.0	94.0	0
3	3	50	1.0	M	YES	20.0	0.0	0	1	0	233.0	158.0	88.0	28.26	68.0	94.0	1
4	4	64	1.0	F	YES	30.0	0.0	0	0	0	241.0	136.5	85.0	26.42	70.0	77.0	0
5	5	61	3.0	F	NO	0.0	0.0	0	1	0	272.0	182.0	121.0	32.80	85.0	65.0	1
6	6	61	1.0	M	NO	0.0	0.0	0	1	0	238.0	232.0	136.0	24.83	75.0	79.0	0
7	7	36	4.0	M	YES	35.0	0.0	0	0	0	295.0	102.0	68.0	28.15	60.0	63.0	0
8	8	41	2.0	F	YES	20.0	NaN	0	0	0	220.0	126.0	78.0	20.70	86.0	79.0	0
9	9	55	2.0	F	NO	0.0	0.0	0	1	0	326.0	144.0	81.0	25.71	85.0	NaN	0
10	10	61	1.0	F	NO	0.0	0.0	0	1	0	NaN	185.0	121.0	35.22	80.0	NaN	0
11	11	53	2.0	F	NO	0.0	0.0	0	0	0	210.0	138.0	86.5	22.49	88.0	87.0	0
12	12	43	2.0	F	NO	0.0	0.0	0	0	0	213.0	96.0	62.0	19.38	74.0	80.0	0
13	13	44	1.0	M	YES	40.0	0.0	0	0	0	227.0	146.5	97.0	26.92	80.0	67.0	0
14	14	58	3.0	F	NO	0.0	0.0	0	1	0	188.0	160.0	120.0	35.58	88.0	85.0	0

2.2) Describing the attribute of the dataset

a) Age: This attribute represents the age of the individual in your dataset.

b) Education: This attribute could represent the education level of individuals.

c) Sex: This attribute signifies the gender of the individual.

d) Is smoking: This attribute indicates whether an individual is a smoker or not.

e) BPMeds: This attribute indicates whether an individual is taking blood pressure medication or not.

f) prevalentStroke: This attribute indicates whether an individual has had a stroke or not.

g) prevalentHyp: This attribute represents the prevalence of hypertension in individual.

.

h) Totchol: This attribute represents the total cholesterol level in an individual's blood. High levels are a risk factor for heart.

i) sysBP and diaBP: This attribute denotes systolic and diastolic blood pressure.

j) BMI: Represent an individual's body mass relative to their height. It is used to assess the impact of weight on cardiovascular issues.

k) Heart Rate: This attribute indicates an individual heart rate. Abnormal heart rate associated with cardiovascular disease.

l) Glucose: This attribute represents the blood glucose level. A high level can be indicative of diabetes.

m) CHD: This attribute indicates the presence or absence of coronary disease.

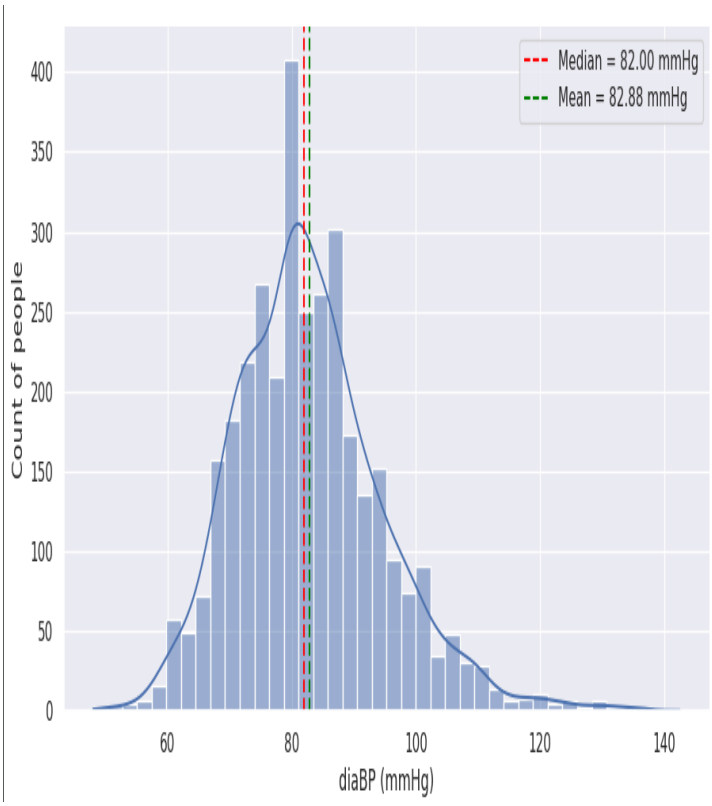
2.3) SHAPE OF DATASET

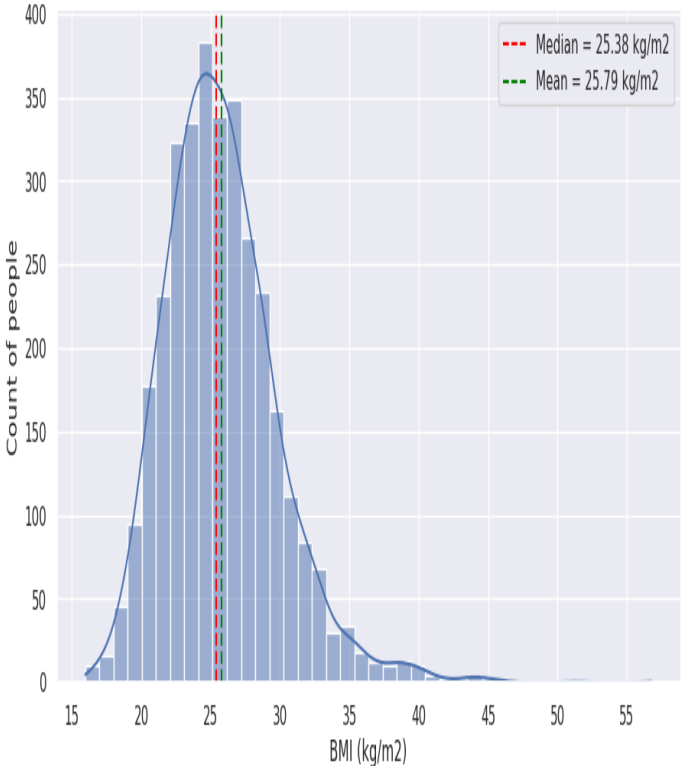
```
[ ]  print(f'Number of rows in the dataset: {df.shape[0]}')
     print(f'Number of columns  in the dataset: {df.shape[1]}')

     Number of rows in the dataset: 3390
     Number of columns  in the dataset: 17
```

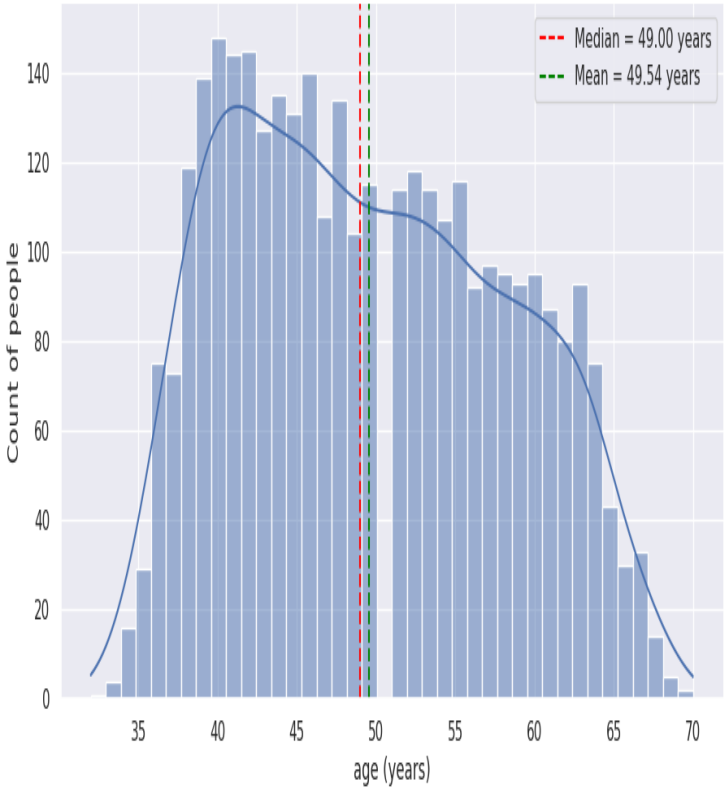
df. shape: It returns the number of rows and columns in the dataset.

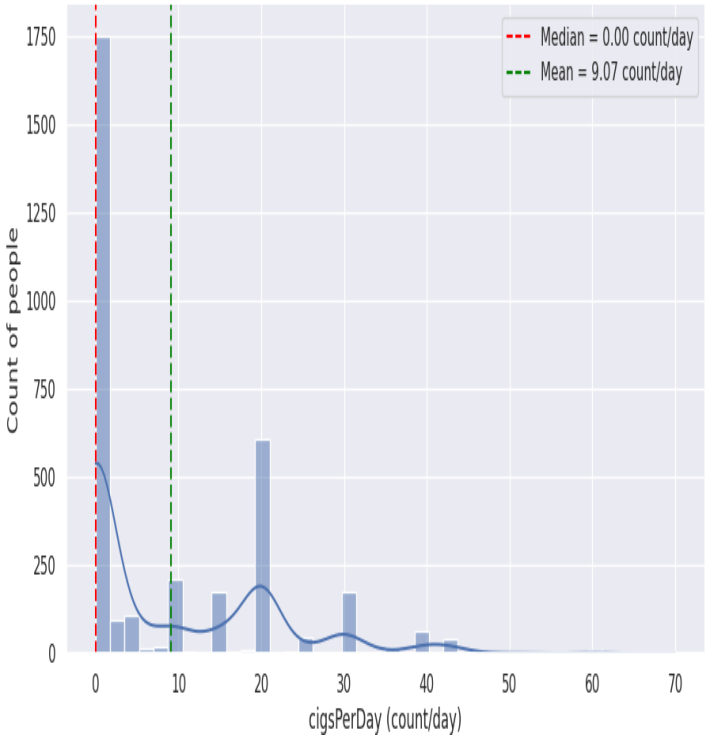
2.4) DATA VISULIZATION

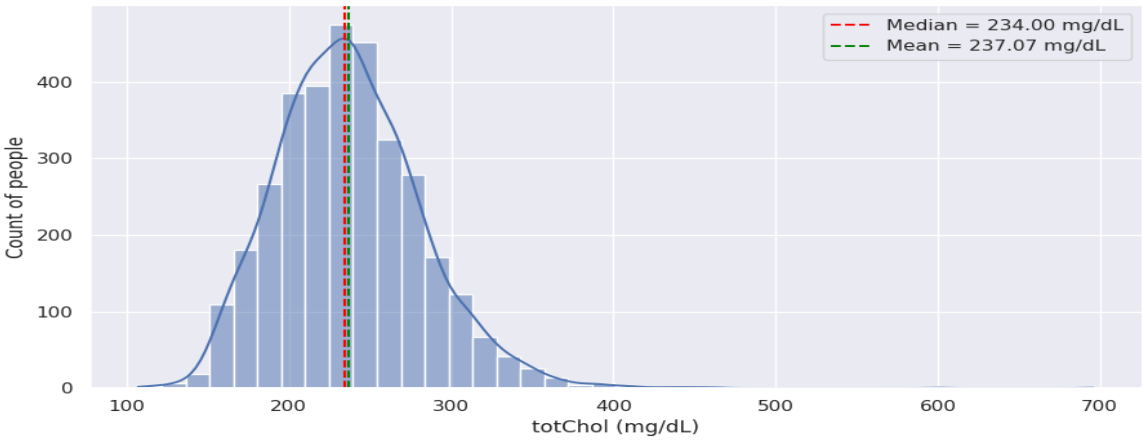
```
# Function for Plotting the continuous variable distribution
def displot_with_median(dataset, variable, median = False, mean = False, unit = None):
  '''A displot with median and mean and the appropriate units (if any) as the inputs'''
  sns.displot(dataset[variable], height = 5, aspect = 11/6, bins = 40, kde = True)
  if median == True:
    plt.axvline(dataset[variable].median(), color = 'red', linestyle = '--', label = f'Median = {dataset[variable].median():.2f} {unit}')
  if mean == True:
    plt.axvline(dataset[variable].mean(), color = 'green', linestyle = '--', label = f'Mean = {dataset[variable].mean():.2f} {unit}')
  plt.ylabel('Count of people')
  plt.xlabel(var + f' ({unit})')
  plt.legend()
  plt.show()


units = ['years', 'count/day', 'mg/dL', 'mmHg', 'mmHg', 'kg/m2', 'beats/min', 'mg/dL']
cont_var_units = dict(zip(cont_vars, units))


for var in cont_var_units:
  displot_with_median(df, var, median = True, mean = True, unit = cont_var_units[var])
```

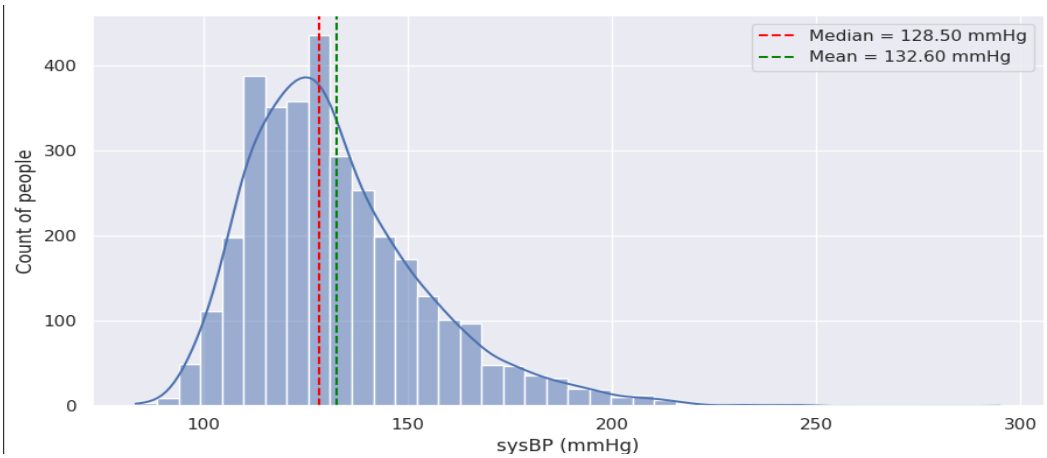

Median = 82.00 mmHg
Mean = 82.88 mmHg
Count of people
diaBP (mmHg)

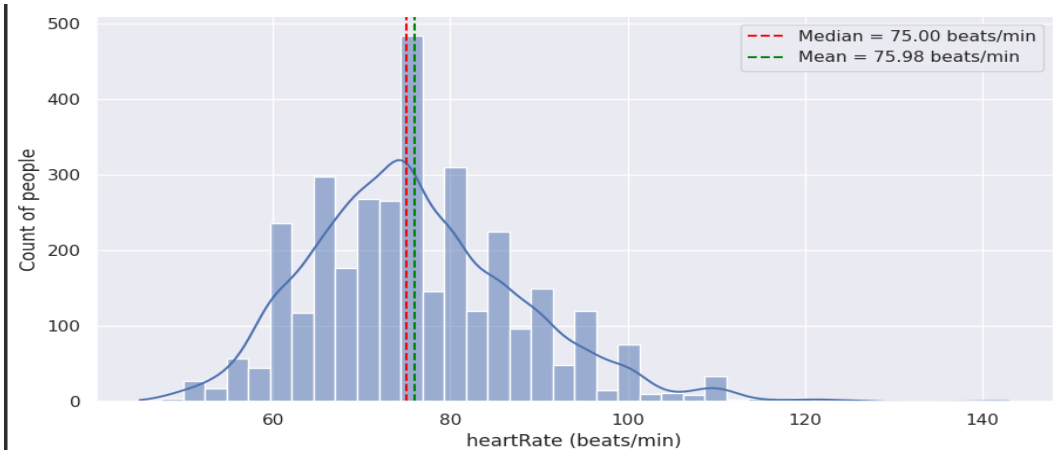

Median = 25.38 kg/m2
Mean = 25.79 kg/m2
Count of people
BMI (kg/m2)

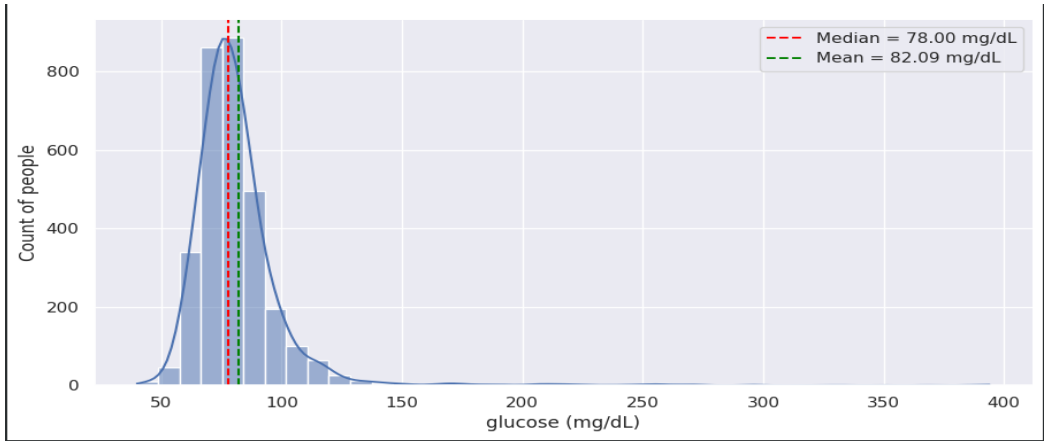

Median = 49.00 years
Mean = 49.54 years
Count of people
age (years)


Median = 0.00 count/day
Mean = 9.07 count/day
Count of people
cigsPerDay (count/day)


Median = 234.00 mg/dL
Mean = 237.07 mg/dL
Count of people
totChol (mg/dL)


Median = 128.50 mmHg
Mean = 132.60 mmHg
Count of people
sysBP (mmHg)


Median = 75.00 beats/min
Mean = 75.98 beats/min
Count of people
heartRate (beats/min)


Median = 78.00 mg/dL
Mean = 82.09 mg/dL
Count of people
glucose (mg/dL)

3) Data Preprocessing and Train Test Split: -

Data preprocessing: This involves cleaning, transforming, and organizing raw data into a format that is suitable for model training.

3.1) Importing libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
[ ] from statsmodels.stats.proportion import proportions_ztest
    from scipy.stats import ttest_1samp, shapiro
    from sklearn.feature_selection import chi2
```

```
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import StandardScaler, MinMaxScaler

from sklearn.metrics import recall_score, make_scorer, roc_auc_score, confusion_matrix, classification_report, ConfusionMatrixDisplay

from sklearn.model_selection import GridSearchCV, RepeatedStratifiedKFold

from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
import xgboost as xgb

import warnings
warnings.filterwarnings('ignore')
```

3.2) Load data

Dataset Loading

```
[ ] from google.colab import files
    files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current b

```
Saving data_cardiovascular_risk(1).csv to data_cardiovascular_risk(1).csv
{'data_cardiovascular_risk(1).csv':
b'id,age,education,sex,is_smoking,cigsPerDay,BPMeds,prevalentStroke,prevalentHyp,diabetes,totChol,
```

3.3) Check the shape of the dataset.

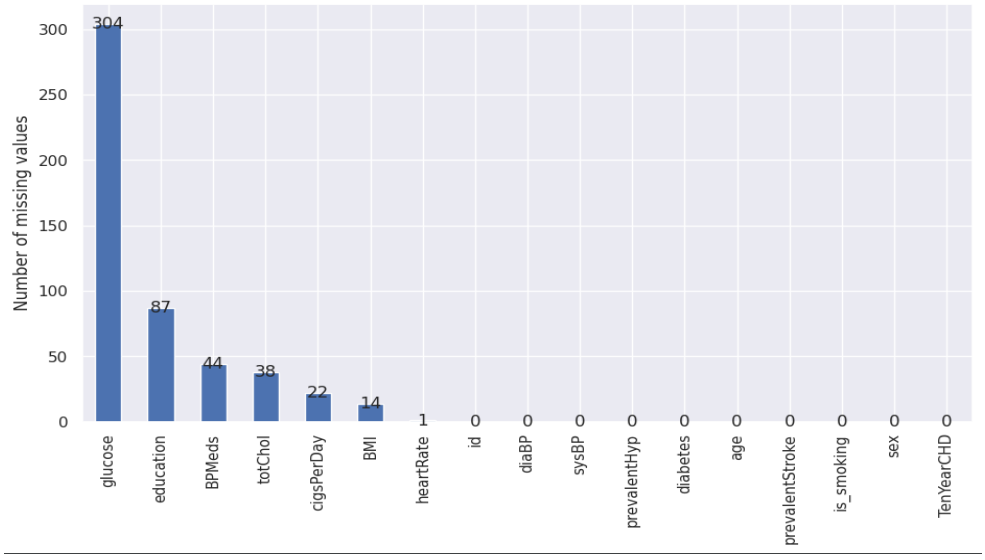
```
print(f'Number of rows in the dataset: {df.shape[0]}')
print(f'Number of columns  in the dataset: {df.shape[1]}')
```

```
Number of rows in the dataset: 3390
Number of columns  in the dataset: 17
```

3.4) Check for the missing values.

```
print(f'There are {df.isna().sum().sum()} missing values in the dataset\n')
df.isna().sum()
```

```
There are 510 missing values in the dataset

id                  0
age                 0
education          87
sex                 0
is_smoking          0
cigsPerDay         22
BPMeds             44
prevalentStroke     0
prevalentHyp        0
diabetes            0
totChol            38
sysBP               0
diaBP               0
BMI                14
heartRate           1
glucose           304
CHD                 0
dtype: int64
```



3.4) Check the correlation.

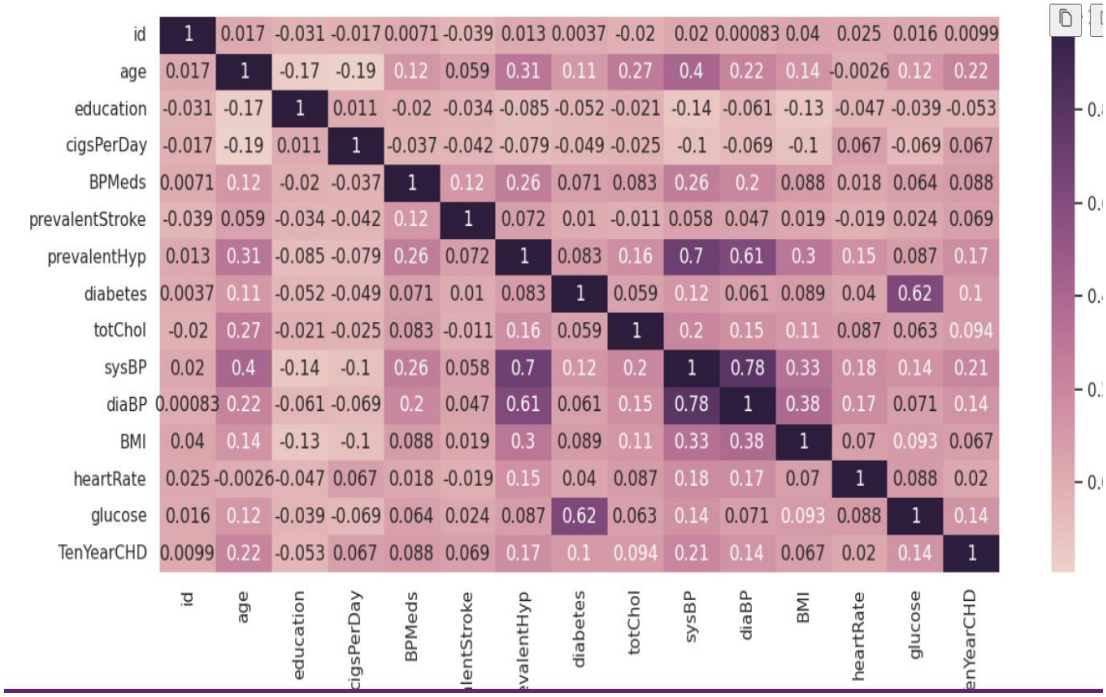
```
df.corr()
```

Python

	id	age	education	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI
id	1.000000	0.016759	-0.030573	-0.017249	0.007080	-0.038873	0.013236	0.003690	-0.019889	0.019676	0.000832	0.039849
age	0.016759	1.000000	-0.172559	-0.192335	0.124376	0.059038	0.308826	0.107875	0.274028	0.404845	0.221896	0.138176
education	-0.030573	-0.172559	1.000000	0.011126	-0.019920	-0.034194	-0.084817	-0.052013	-0.020781	-0.137195	-0.060801	-0.133313
cigsPerDay	-0.017249	-0.192335	0.011126	1.000000	-0.037080	-0.042057	-0.079312	-0.049016	-0.024703	-0.100834	-0.068785	-0.100996
BPMeds	0.007080	0.124376	-0.019920	-0.037080	1.000000	0.119402	0.259548	0.071316	0.083299	0.263729	0.201217	0.087932
prevalentStroke	-0.038873	0.059038	-0.034194	-0.042057	0.119402	1.000000	0.071652	0.010115	-0.010832	0.057568	0.047235	0.018602
prevalentHyp	0.013236	0.308826	-0.084817	-0.079312	0.259548	0.071652	1.000000	0.082565	0.159680	0.699285	0.612897	0.300464
diabetes	0.003690	0.107875	-0.052013	-0.049016	0.071316	0.010115	0.082565	1.000000	0.059080	0.124011	0.061165	0.089112
totChol	-0.019889	0.274028	-0.020781	-0.024703	0.083299	-0.010832	0.159680	0.059080	1.000000	0.199159	0.154974	0.114305
sysBP	0.019676	0.404845	-0.137195	-0.100834	0.263729	0.057568	0.699285	0.124011	0.199159	1.000000	0.781908	0.333864
diaBP	0.000832	0.221896	-0.060801	-0.068785	0.201217	0.047235	0.612897	0.061165	0.154974	0.781908	1.000000	0.380498
BMI	0.039849	0.138176	-0.133313	-0.100996	0.087932	0.018602	0.300464	0.089112	0.114305	0.333864	0.380498	1.000000
heartRate	0.024684	-0.002596	-0.046980	0.066759	0.018107	-0.019184	0.150815	0.039742	0.087458	0.178262	0.174904	0.069655
glucose	0.015517	0.116135	-0.038732	-0.068797	0.064490	0.023607	0.086658	0.620211	0.062723	0.143090	0.071499	0.093454
TenYearCHD	0.009866	0.224927	-0.052751	0.066778	0.088020	0.068627	0.166544	0.103681	0.094306	0.212703	0.135979	0.066947

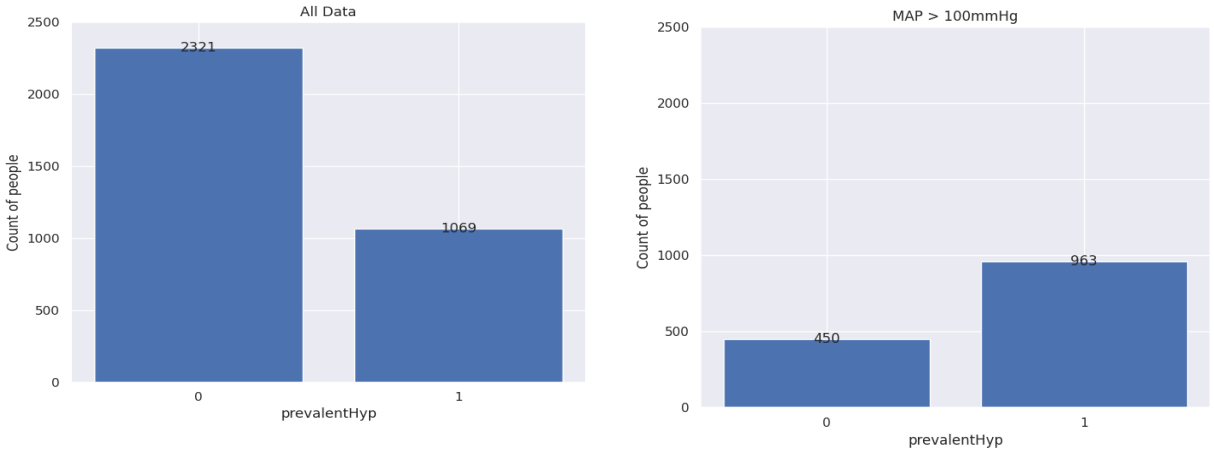
```
plt.figure(figsize = (14, 6))
sns.heatmap(df.corr(), annot = True, cmap = sns.cubehelix_palette(as_cmap=True))
```

<Axes: >



3.5) Feature Engineering

```
# Checking distribution of prevalentHyp

filter_counts = data[data['MAP'] > 100]['prevalentHyp'].value_counts().sort_index()
all_counts = data['prevalentHyp'].value_counts().sort_index()
fig, ax = plt.subplots(nrows = 1, ncols = 2, figsize = (14, 6))
ax[0].bar(all_counts.index, all_counts.values)
ax[0].set_title('All Data')
ax[1].bar(filter_counts.index, filter_counts.values)
ax[1].set_title('MAP > 100mmHg')
for i in range(2):
  ax[i].set(xticks = [0, 1], ylim = [0, 2500], ylabel = 'Count of people', xlabel = 'prevalentHyp')
  display_vals(ax[i])
plt.tight_layout()
plt.show()
```




```
[ ]  # Defining the function for Diabetes grades
     def diabetes_grades(df):
       if df['glucose'] >= 126:
         return 4 #Diabetes
       elif df['glucose'] > 100:
         return 3 #Pre-diabetes
       elif df['glucose'] > 70:
         return 2 #Normal
       elif df['glucose'] < 71:
         return 1 #Hypoglycemia
```

```
# Dropping the redundant columns
data = data.drop(['sysBP', 'diaBP', 'glucose', 'diabetes', 'prevalentHyp'], axis = 1)
cont_vars = [var for var in cont_vars if var not in ['sysBP', 'diaBP', 'glucose']]
cont_vars+=['MAP']
categ_vars = [var for var in categ_vars if var not in ['is_smoking', 'diabetes', 'prevalentStroke', 'prevalentHyp']]
categ_vars+=['diabetes_grade']

print(f'The categorical variables are: {categ_vars}')
print(f'The continuous variables are: {cont_vars}')
```

```
The categorical variables are: ['sex', 'BPMeds', 'diabetes_grade']
The continuous variables are: ['age', 'cigsPerDay', 'totChol', 'BMI', 'heartRate', 'MAP']
```

3.5) Handling Imbalance Dataset:

SMOTE: Smote, which stands for synthetic minority over-sampling technique. Smote helps in balancing the class distribution by creating synthetic examples for the minority class.

SMOTE for Handling Imbalanced data

```
smote = SMOTE(random_state = 8)
X_smote, Y_train_final = smote.fit_resample(X_train, Y_train)
```

3.6) Splitting the data:

```
X = data.drop(['TenYearCHD', 'id'], axis = 1)
Y = data['TenYearCHD']

# Visualising the input data
X.head()
```

Python

	age	sex	cigsPerDay	BPMeds	totChol	BMI	heartRate	MAP	diabetes_grade
0	64	0	3.0	0.0	221.0	25.38	90.0	106.000000	2
1	36	1	0.0	0.0	212.0	29.77	72.0	121.333333	2
2	46	0	10.0	0.0	250.0	20.35	88.0	86.000000	2
3	50	1	20.0	0.0	233.0	28.26	68.0	111.333333	2
4	64	0	30.0	0.0	241.0	26.42	70.0	102.166667	2

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state = 8, stratify = Y, shuffle = True)
```

Python

```
Y_train.value_counts()
```

Python

```
0    2303
1     409
Name: TenYearCHD, dtype: int64
```

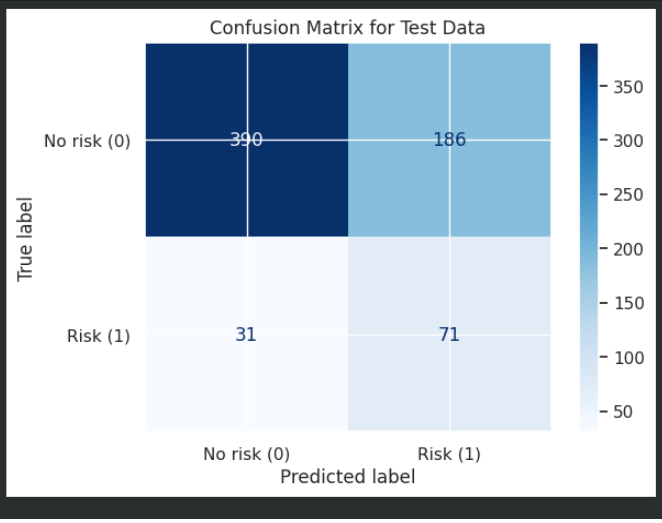
Splitting the data refers to dividing the dataset into two or more subsets for different purposes.
First, the training dataset is the largest portion of the dataset that is used to train the machine learning model.
Second, the test set is a separate portion of the dataset it is used to assess the final performance of the trained machine-learning model.

4) MODELING AND PREDICTON: -

4.1) Logistic Regression:

It is a supervised machine learning algorithm that accomplishes binary classification tasks by predicting the probability of an outcome.

```
lr_scores, lr_test_preds, lr_model = model_train_test(LogisticRegression(), X_train_final, Y_train_final, X_test_final, Y_test)
```




```
# Classification Report
print(classification_report(Y_test, lr_test_preds, target_names=['class-0', 'class-1']))
```

```
              precision    recall  f1-score   support

     class-0       0.93      0.68      0.78       576
     class-1       0.28      0.70      0.40       102

    accuracy                           0.68       678
   macro avg       0.60      0.69      0.59       678
weighted avg       0.83      0.68      0.72       678
```

```
# Printing the train and test Recalls and ROC-AUC scores
def print_scores(model_name, model_scores):
  '''Function to print the scores of a given model'''
  print(f"The train and test recalls of the {model_name} Model are: {round(model_scores['Train Recall'] * 100, 2)}%
  print(f"The train and test ROC-AUC scores of the {model_name} Model are: {round(model_scores['Train ROC-AUC'] * 1

print_scores(model_name = model_names[0], model_scores = lr_scores)
```

```
The train and test recalls of the Logistic Regression Model are: 67.0% and 69.61% respectively
The train and test ROC-AUC scores of the Logistic Regression Model are: 65.85% and 68.66% respectively
```

4.2) CONFUSION MATRIX

A confusion matrix is a table or a matrix that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.

A confusion matrix typically consists of four values:

1. **True Positives (TP):** These are cases in which the model predicted a positive class (e.g., "Yes" or "1") correctly, and the actual class is also positive.
2. **True Negatives (TN):** These are cases in which the model predicted a negative class (e.g., "No" or "0") correctly, and the actual class is also negative.
3. **False Positives (FP):** These are cases in which the model predicted a positive class incorrectly when the actual class is negative. Also known as a "Type I error."
4. **False Negatives (FN):** These are cases in which the model predicted a negative class incorrectly when the actual class is positive. Also known as a "Type II error."

4.2.1) The confusion matrix is often used to calculate various performance metrics for a classification model, including:

- **Accuracy:** The ratio of correctly predicted instances to the total instances. It measures the overall correctness of the model.
  Accuracy = (TP + TN) / (TP + TN + FP + FN)

- **Precision:** Also called Positive Predictive Value, it measures the accuracy of positive predictions. It's the ratio of true positives to the sum of true positives and false positives.
  Precision = TP / (TP + FP)

- **Recall:** Also called Sensitivity or True Positive Rate, it measures the model's ability to identify all relevant instances. It's the ratio of true positives to the sum of true positives and false negatives.
  Recall = TP / (TP + FN)

- **F1-Score:** The harmonic mean of precision and recall. It provides a balance between precision and recall.
  F1-Score = 2 * (Precision * Recall) / (Precision + Recall)

- **Specificity:** Also called True Negative Rate, it measures the model's ability to identify all negative instances.
  Specificity = TN / (TN + FP)

- **False Positive Rate (FPR):** The ratio of false positives to the sum of false positives and true negatives.
  FPR = FP / (FP + TN)

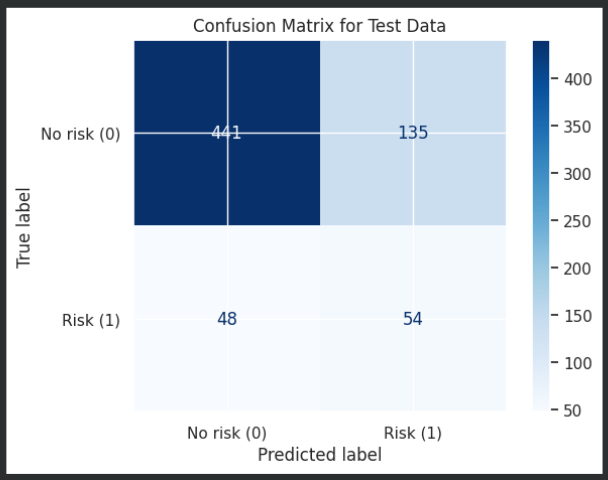
4.3) NAÏVE BAYES:

Naive Bayes is a family of probabilistic machine learning algorithms used for classification and sometimes regression. It's based on Bayes' theorem, which is a fundamental theorem in probability theory.

```
[ ]  # Defining the Hyperparameters and scoring metric
     params_nb = {'var_smoothing': np.logspace(0, -9, num = 50)}
     cv = RepeatedStratifiedKFold(n_splits = 5, n_repeats = 3, random_state = 42)
     scorer = make_scorer(recall_score, average = 'binary')

     nb_models = GridSearchCV(GaussianNB(), params_nb, cv = cv, scoring = scorer)
```

```
# Training the model
nb_scores, nb_test_preds, nb_model = model_train_test(nb_models, X_train_final, Y_train_final, X_test_final, Y_test, gs = True)
```

```
Best model parameters are: {'var_smoothing': 0.1842069969326716}
Best model score is: 0.5102144047282217
```




```
[ ]  # Classification Report
     print(classification_report(Y_test, nb_test_preds, target_names=['class-0', 'class-1']))
```

```
                precision    recall  f1-score   support

       class-0       0.90      0.77      0.83       576
       class-1       0.29      0.53      0.37       102

      accuracy                           0.73       678
     macro avg       0.59      0.65      0.60       678
  weighted avg       0.81      0.73      0.76       678
```

```
[ ]  print_scores(model_name = model_names[1], model_scores = nb_scores)
```

```
The train and test recalls of the Naive Bayes Model are: 51.58% and 52.94% respectively
The train and test ROC-AUC scores of the Naive Bayes Model are: 63.48% and 64.75% respectively
```

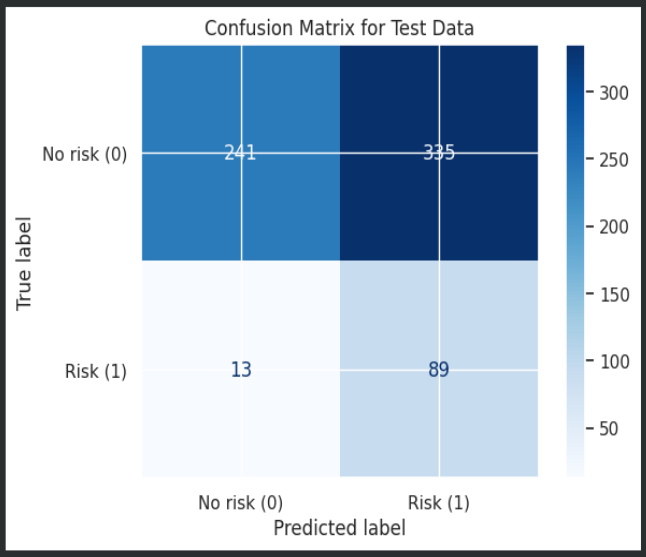
4.4) DECISION TREE:

A decision tree is a popular supervised machine learning algorithm used for both classification and regression tasks. It's a tree-like model that is constructed by recursively partitioning the dataset into subsets based on the values of input features.

```
[ ]  # Defining the Hyperparameters
     params_dt = {
                  'max_depth' : [3, 4, 5],
                  'min_samples_split':[10, 20, 25, 30],
                  'min_samples_leaf':[10, 20, 25, 30]
                  }

     dt_model = DecisionTreeClassifier(criterion= 'entropy', random_state = 42)
     dt_models = GridSearchCV(dt_model, params_dt, cv = cv, scoring = scorer)
```

```
dt_scores, dt_test_preds, dt_model = model_train_test(dt_models, X_train_final, Y_train_final, X_test_final, Y_test, gs =True)
```

```
Best model parameters are: {'max_depth': 4, 'min_samples_leaf': 30, 'min_samples_split': 10}
Best model score is: 0.8323807727372757
```




```
# Classification Report
print(classification_report(Y_test, dt_test_preds, target_names=['class-0', 'class-1']))
```

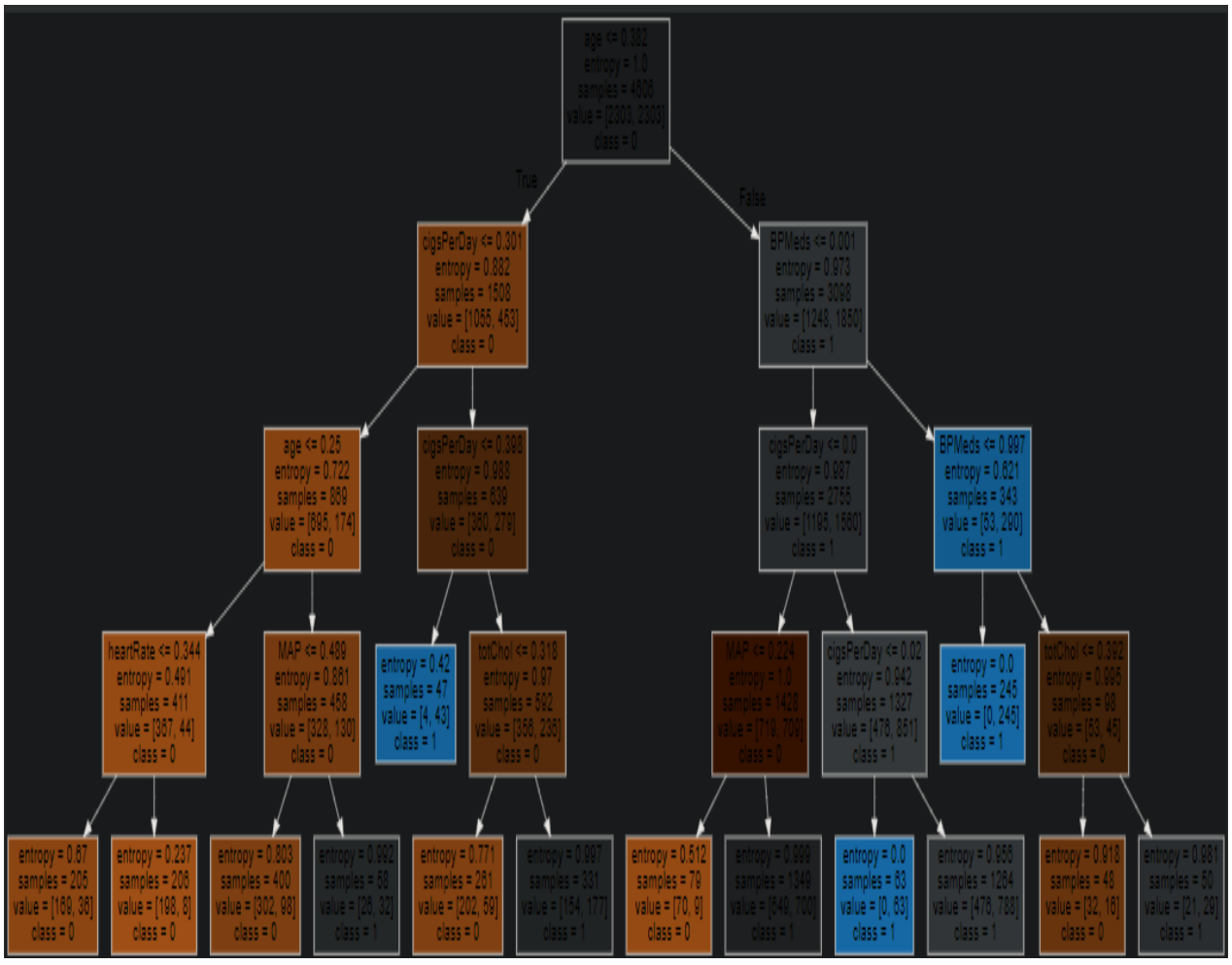
```
              precision    recall  f1-score   support

     class-0       0.95      0.42      0.58       576
     class-1       0.21      0.87      0.34       102

    accuracy                           0.49       678
   macro avg       0.58      0.65      0.46       678
weighted avg       0.84      0.49      0.54       678
```

```
[ ]  # Printing the model scores
     print_scores(model_name = model_names[2], model_scores = dt_scores)
```

```
The train and test recalls of the Decision Tree Model are: 90.19% and 87.25% respectively
The train and test ROC-AUC scores of the Decision Tree Model are: 66.22% and 64.55% respectively
```

4.4.1) DECISION TREE VISUALISATION

```
# Visualising the decision tree
graph = Source(tree.export_graphviz(dt_model, out_file = None, feature_names = X.columns, class_names=['0', '1'] , filled = True))
display(SVG(graph.pipe(format = 'svg')))
```

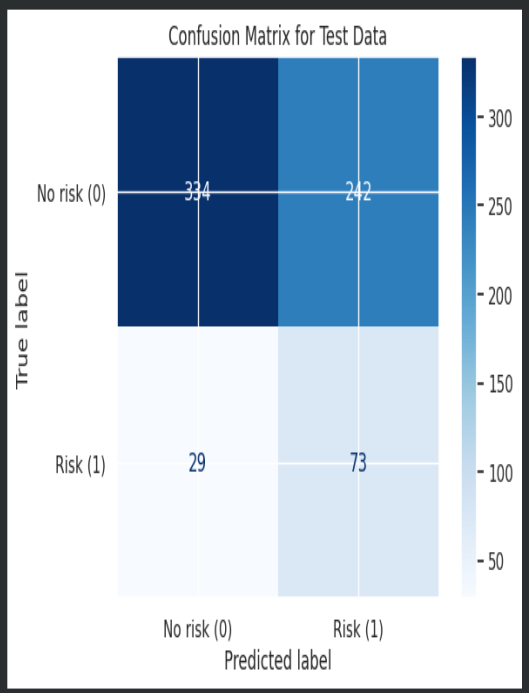


4.5) K-NEAREST NEIGHBOURS:

K-Nearest Neighbours (KNN) is a simple and widely used supervised machine learning algorithm for classification and regression tasks. It's known as a non-parametric, instance-based learning algorithm because it makes predictions based on the majority class or average value of the k-nearest data points in the training dataset.

```
[ ] # Fitting the knn model for various values of k

    knn_train_recalls = []
    knn_test_recalls = []
    for k in range(1, 51):
      knn_scores = model_train_test(KNeighborsClassifier(n_neighbors = k), X_train_final, Y_train_final, X_test_final, Y_test, confusion = False)[0]
      knn_train_recalls.append(knn_scores['Train Recall'])
      knn_test_recalls.append(knn_scores['Test Recall'])
```

```
[ ] # Fitting the knn model with the optimum k
    knn_scores, knn_test_preds, knn_model = model_train_test(KNeighborsClassifier(n_neighbors = k_optimum), X_train_final, Y_train_final, X_test_final, Y_test, gs = False)
```



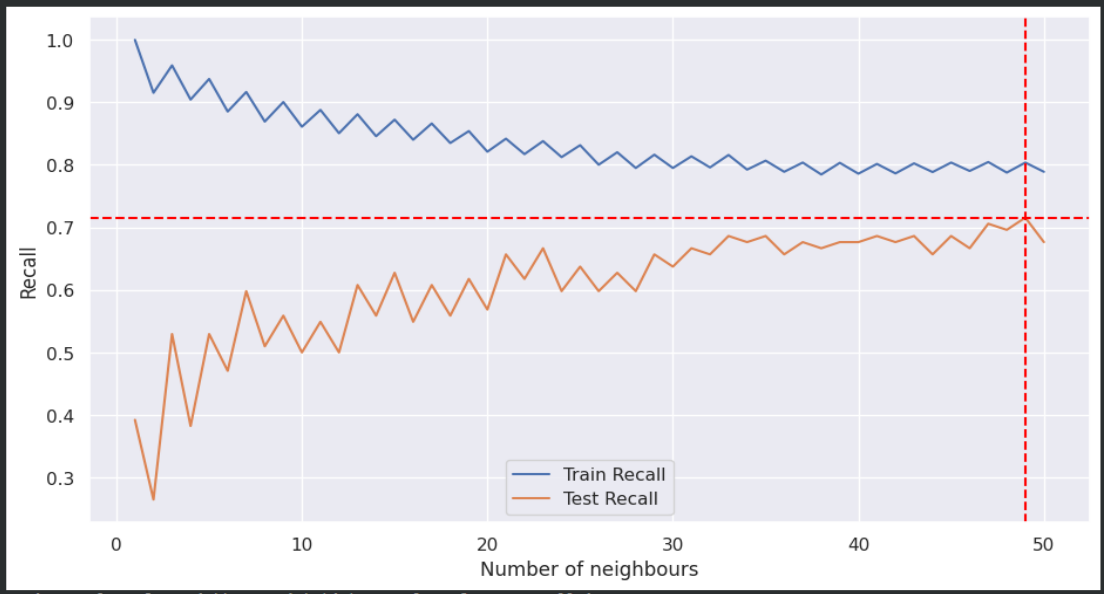
```
[ ] print(classification_report(Y_test, knn_test_preds, target_names=['class-0', 'class-1']))
```

```
              precision    recall  f1-score   support

     class-0       0.92      0.58      0.71       576
     class-1       0.23      0.72      0.35       102

    accuracy                           0.60       678
   macro avg       0.58      0.65      0.53       678
weighted avg       0.82      0.60      0.66       678
```

```
print_scores(model_name = model_names[3], model_scores = knn_scores)
```

```
The train and test recalls of the KNN Model are: 80.37% and 71.57% respectively
The train and test ROC-AUC scores of the KNN Model are: 70.32% and 64.78% respectively
```

```
# Plotting the train and test recalls
k_optimum = knn_test_recalls.index(max(knn_test_recalls)) + 1
plt.figure(figsize=(11, 6))
plt.plot(range(1, 51), knn_train_recalls, label='Train Recall')
plt.plot(range(1, 51), knn_test_recalls, label='Test Recall')
plt.axhline(max(knn_test_recalls), color = 'red', linestyle = '--')
plt.axvline(k_optimum, color = 'red', linestyle = '--')
plt.xlabel('Number of neighbours')
plt.ylabel('Recall')
plt.legend()
plt.show()

print(f'Optimum value of n_neighbours with highest value of test recall is {k_optimum}')
```

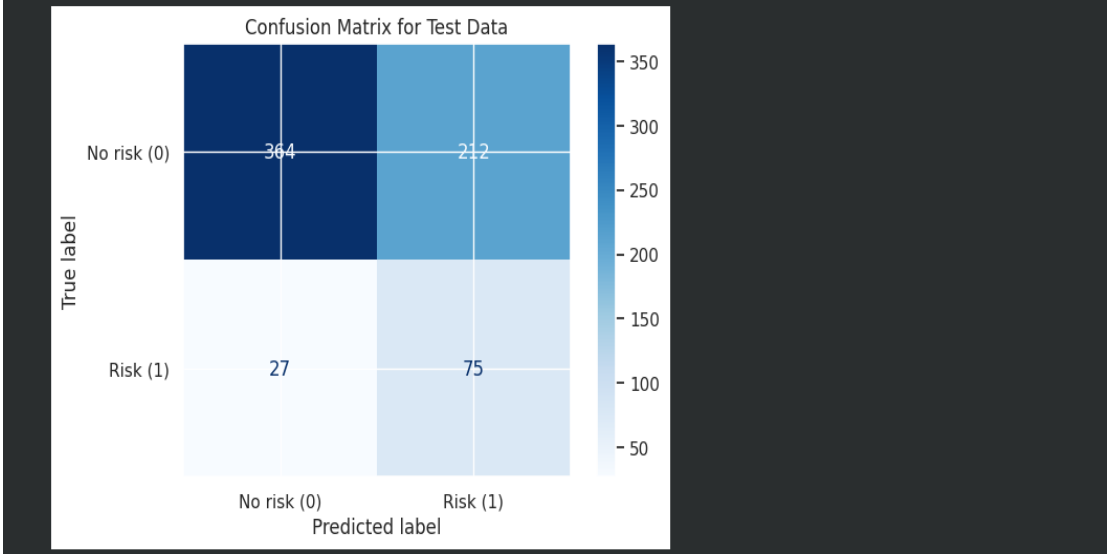



```
Optimum value of n_neighbours with highest value of test recall is 49
```

4.6) SUPPORT VECTOR MACHINE:

Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks. SVM is particularly effective for classification tasks and is known for its ability to handle high-dimensional data, find complex decision boundaries, and work well in both linear and nonlinear scenarios.

```
svc_scores, svc_test_preds, svc_model = model_train_test(svc_models, X_train_final, Y_train_final, X_test_final, Y_test, gs = True)

Best model parameters are: {'C': 10, 'gamma': 0.01, 'kernel': 'rbf'}
Best model score is: 0.701117922600522
```




```
[ ] # Classification Report
    print(classification_report(Y_test, svc_test_preds, target_names=['class-0', 'class-1']))
```

	precision	recall	f1-score	support
class-0	0.93	0.63	0.75	576
class-1	0.26	0.74	0.39	102
accuracy			0.65	678
macro avg	0.60	0.68	0.57	678
weighted avg	0.83	0.65	0.70	678

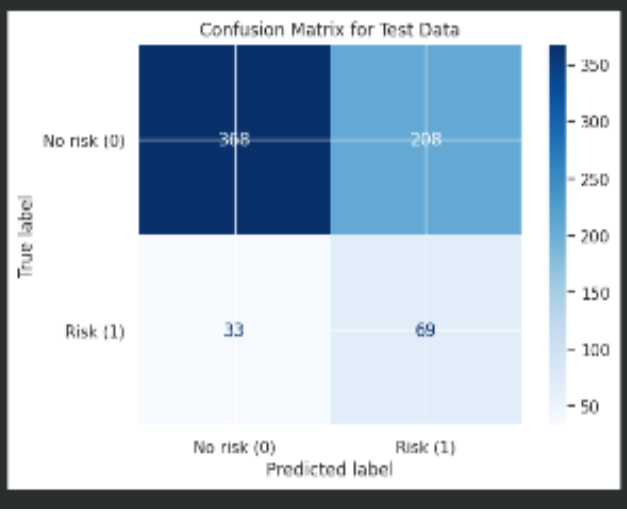
```
[ ] # Printing the scores
    print_scores(model_name = model_names[4], model_scores = svc_scores)

    The train and test recalls of the SVM Model are: 70.78% and 73.53% respectively
    The train and test ROC-AUC scores of the SVM Model are: 65.85% and 68.36% respectively
```

4.7) RANDOM FOREST:

Random Forest is an ensemble machine learning algorithm used for both classification and regression tasks. It is a versatile and powerful algorithm known for its accuracy and robustness. Random Forest is an ensemble of decision trees, where multiple decision trees are trained on different subsets of the data and their predictions are combined to make a final prediction.

```
[ ]  # Defining the Hyperparameters
     params_rf = {
                   'n_estimators':[50, 100, 200],
                   'max_depth':[3, 4, 5],
                   'min_samples_split':[10, 20, 25],
                   'min_samples_leaf':[10, 20, 25]
                  }

     rf_model = RandomForestClassifier(criterion= 'entropy', random_state = 42)
     rf_models = GridSearchCV(rf_model, params_rf, cv = cv, scoring = scorer)
```

```
# Training the model

rf_scores, rf_test_preds, rf_model = model_train_test(rf_models, X_train_final, Y_train_final, X_test_final, Y_test, gs = True)

Best model parameters are: {'max_depth': 5, 'min_samples_leaf': 10, 'min_samples_split': 25, 'n_estimators': 100}
Best model score is: 0.7027503380349209
```




```
[ ]  print(classification_report(Y_test, rf_test_preds, target_names=['class-0', 'class-1']))
```

	precision	recall	f1-score	support
class-0	0.92	0.64	0.75	576
class-1	0.25	0.68	0.36	102
accuracy			0.64	678
macro avg	0.58	0.66	0.56	678
weighted avg	0.82	0.64	0.69	678

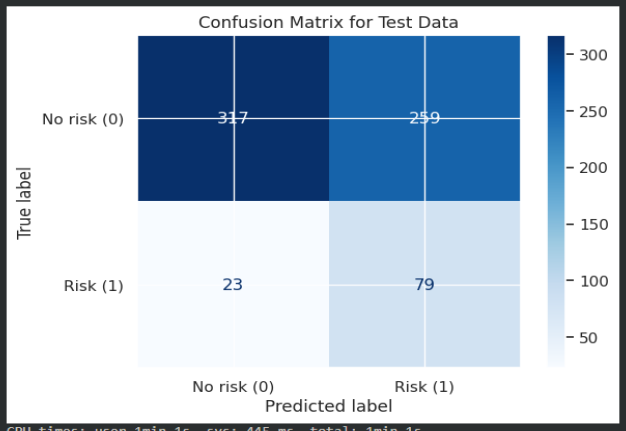
```
[ ]  # Printing the model scores
     print_scores(model_name = model_names[5], model_scores = rf_scores)

     The train and test recalls of the Random Forest Model are: 78.98% and 67.65% respectively
     The train and test ROC-AUC scores of the Random Forest Model are: 72.28% and 65.77% respectively
```

4.8) XGBOOST:

XGBoost, short for "Extreme Gradient Boosting," is a popular and powerful machine learning algorithm that belongs to the gradient boosting family. It is known for its efficiency, speed, and high predictive accuracy, making it a popular choice for both classification and regression tasks.

```
[ ]  # Defining the Hyperparameters
     params_xgb = {
                   'n_estimators':[50, 100],
                   'max_depth':[3, 4],
                   'learning_rate':[0.01, 0.02]
                   }

     xgb_model = xgb.XGBClassifier(random_state = 42)
     xgb_models = GridSearchCV(xgb_model, params_xgb, cv = cv, scoring = scorer)
```

```
# Training the model
%%time
xgb_scores, xgb_test_preds, xgb_model = model_train_test(xgb_models, X_train_final, Y_train_final, X_test_final, Y_test, gs = True)

Best model parameters are: {'learning_rate': 0.01, 'max_depth': 3, 'n_estimators': 50}
Best model score is: 0.788684648998711
```




```
CPU times: user 1min 1s, sys: 445 ms, total: 1min 1s
Wall time: 39 s
```

```
[ ]  # Classification Report
     print(classification_report(Y_test, xgb_test_preds, target_names=['class-0', 'class-1']))
```

	precision	recall	f1-score	support
class-0	0.93	0.55	0.69	576
class-1	0.23	0.77	0.36	102
accuracy			0.58	678
macro avg	0.58	0.66	0.53	678
weighted avg	0.83	0.58	0.64	678

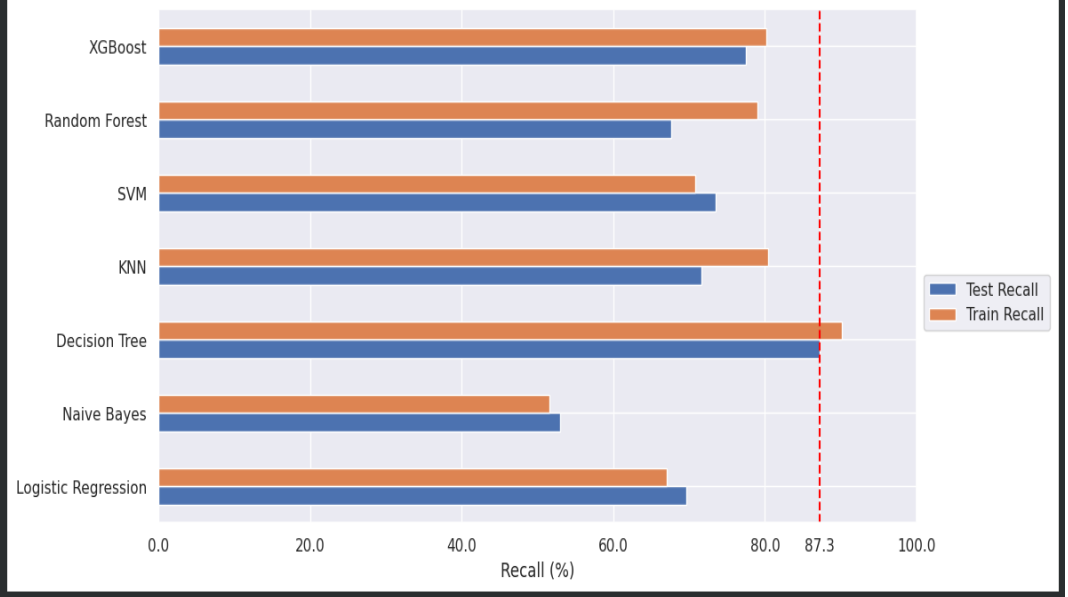
```
[ ]  # Printing model scores
     print_scores(model_name = model_names[6], model_scores = xgb_scores)

     The train and test recalls of the XGBoost Model are: 80.2% and 77.45% respectively
     The train and test ROC-AUC scores of the XGBoost Model are: 67.39% and 66.24% respectively
```

4.9) MODEL EVALUATION:

```
# Comparing the evaluation metrics from each model
results_df = pd.DataFrame(scores, index = model_names)
styler_list = [
                {'selector': 'th','props': [('border', '2px solid black')]},
                {'selector': 'td','props': [('border', '2px solid black')]}
              ]
styler = results_df.style.set_table_styles(styler_list)
display(styler)
```

	Train Recall	Test Recall	Train ROC-AUC	Test ROC-AUC
Logistic Regression	66.999566	69.607843	65.848893	68.658088
Naive Bayes	51.584889	52.941176	63.482414	64.751838
Decision Tree	90.186713	87.254902	66.217977	64.547590
KNN	80.373426	71.568627	70.321320	64.777369
SVM	70.777247	73.529412	65.848893	68.361928
Random Forest	78.983934	67.647059	72.275293	65.767974
XGBoost	80.199739	77.450980	67.390360	66.242851

```
# Plotting the train and test Recalls and choosing optimum model
results_df[['Test Recall', 'Train Recall']].plot(kind = 'barh', figsize = (11, 6))
plt.xlabel('Recall (%)')
plt.xlim((0, 100))
plt.xticks([0, 20, 40, 60, 80, 100, results_df['Test Recall'].max()])
plt.axvline(results_df['Test Recall'].max(), color = 'red', linestyle = '--')
plt.legend(bbox_to_anchor = (1, 0.5))
plt.show()

best_model = results_df[results_df['Test Recall'] == results_df['Test Recall'].max()].index[0]
print(f'\nThe model with maximum Test Recall is the {best_model} model')
```




```
The model with maximum Test Recall is the Decision Tree model
```

5) MODEL PREDICTION:

```
[ ] # Picking any particular observation for local analysis
    obs = 8
    print(f'The actual class of TenYearCHD of this observation is {Y_test.iloc[obs]}.\nThe predicted class is {dt_test_preds[obs]}.\nThe data of the patient is:\n')
    print(X_test.iloc[obs])

    The actual class of TenYearCHD of this observation is 0.
    The predicted class is 0.
    The data of the patient is:

    age              43.00
    sex               1.00
    cigsPerDay        0.00
    BPMeds            0.00
    totChol         262.00
    BMI              24.01
    heartRate        85.00
    MAP              93.00
    diabetes_grade    2.00
    Name: 1052, dtype: float64
```

```
# Picking any particular observation for local analysis
obs = 6
print(f'The actual class of TenYearCHD of this observation is {Y_test.iloc[obs]}.\nThe predicted class is {dt_test_preds[obs]}.\nThe data of the patient is:\n')
print(X_test.iloc[obs])

The actual class of TenYearCHD of this observation is 1.
The predicted class is 1.
The data of the patient is:

age              60.000000
sex               1.000000
cigsPerDay       20.000000
BPMeds            1.000000
totChol         269.000000
BMI              29.590000
heartRate        60.000000
MAP             123.333333
diabetes_grade    2.000000
Name: 2704, dtype: float64
```

```
[118] input_data=(43,1,0,0,262,24,85,93,2)
      input_data_as_numpy_array=np.asarray(input_data)
      input_data_reshaped=input_data_as_numpy_array.reshape(1,-1)
      prediction=dt_model.predict(input_data_reshaped)
      print(prediction)

      if(prediction[0] == 0):
        print("The person is unlikely to have a heart Disease")
      else:
        print("The person is likely to have a heart Disease")

      [1]
      The person is likely to have a heart Disease
```

6) USER INTERFACE WEB APP:

pred_systum.py | cardio_web_app.py

```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""

import numpy as np
import pickle


loaded_model=pickle.load(open("C:/Users/harsh/OneDrive/Desktop/final project/INTERFACE_PROJECT/INTERFACE_PROJECT/Final_model.sav",'rb'))

input_data=(43,1,0,0,262,24,85,93,2)
input_data_as_numpy_array=np.asarray(input_data)
input_data_reshaped=input_data_as_numpy_array.reshape(1,-1)
prediction=loaded_model.predict(input_data_reshaped)
print(prediction)

if(prediction[0] == 0):
  print("The person is unlikely to have a heart Disease")
else:
  print("The person is likely to have a heart Disease")
```

```
import numpy as np
import pickle
import streamlit as st
from PIL import Image

loaded_model=pickle.load(open("C:/Users/harsh/OneDrive/Desktop/final project/INTERFACE_PROJECT/INTERFACE_PROJECT/Final_model.sav",'rb'))

def cardio_pred(input_data):

    input_data_as_numpy_array=np.asarray(input_data)
    input_data_reshaped=input_data_as_numpy_array.reshape(1,-1)
    prediction=loaded_model.predict(input_data_reshaped)
    print(prediction)

    if(prediction[0] == 0):
      return "The person is unlikely to have a heart Disease"
    else:
      return "The person is likely to have a heart Disease"



def main():

    st.title("Cardiovascular Disease Prediction Web App")

    image = Image.open('C:/Users/harsh/OneDrive/Desktop/final project/INTERFACE_PROJECT/INTERFACE_PROJECT/hinage.jpg5')
    st.image(image, caption='Do your part by caring for the heart')

    age=st.text_input("Enter the Age of the person")
    sex = st.radio("Select Gender [Male:1,Female:0]::",('1','0'))
    if (sex == '1'):
        st.info("male")
    else:
        st.info("Female")
    cigsPerDay=st.slider("Number of cigarettes smoked in a single day",0,25)
    BPMeds=st.text_input("Do you take Blood pressure medications?")
    totChol=st.text_input("What is your Cholestrol level?")
    BMI=st.text_input("Enter your Body_Mass Index[weight(kg)/height(m)]")
    heartRate=st.slider("Enter your Heart rate",40,130)
    MAP=st.text_input("Enter your Mean Arterial Pressure")
    diabetes_grade=st.text_input("Enter your diabetes grade")


    #code for Prediction

    diagnosis = ''


    if st.button("Cardiovascular Disease Result"):
        diagnosis=cardio_pred([age,sex,cigsPerDay,BPMeds,totChol,BMI,heartRate,MAP,diabetes_grade])

    st.success(diagnosis)
```

Cardiovascular Disease Prediction Web App

Do your part by caring for the heart

Enter the Age of the person

64

Select Gender [Male:1,Female:0]::

1

0

Female

Number of cigarettes smoked in a single day

0

0 25

Do you take Blood pressure medications?

0

What is your Cholestrol level?

305

Enter your Body_Mass Index(weight(kg)/height(m)]

25.77

Enter your Heart rate

67

40 130

Enter your Mean Arterial Pressure

86.833

Enter your diabetes grade

1

Cardiovascular Disease Result

The person is likely to have a heart Disease

7) CONCLUSION AND FUTURE SCOPE:

7.1) CONCLUSION

The development of a heart disease prediction model using machine learning techniques represents a significant step forward in the field of healthcare and medical diagnostics. The model's ability to analyze and interpret complex medical data to identify individuals at risk of heart disease has the potential to revolutionize early detection and intervention, ultimately saving lives and reducing healthcare costs.

In conclusion, the heart disease prediction model is a valuable tool for both patients and healthcare providers. Patients can benefit from early warnings and take proactive steps to improve their heart health, such as lifestyle modifications and regular check-ups. Healthcare providers can use the model to prioritize patients for further evaluation and tailor treatment plans to individual risk profiles.

7.2) FUTURE SCOPE

Looking ahead, the future scope of this model is promising. Here are some key areas for future research and development:

1. **Integration with Electronic Health Records (EHRs):** Integrating the prediction model with electronic health records can enhance its accuracy and applicability. Access to a patient's complete medical history and real-time data can lead to more precise predictions.
2. **Continuous Monitoring:** Expanding the model to support continuous monitoring of patients' health can provide ongoing risk assessments, allowing for timely interventions and personalized care plans.
3. **Multimodal Data:** Incorporating various data sources, such as genetic information, wearable device data, and patient-reported outcomes, can improve the model's predictive capabilities.
4. **Explain ability:** Enhancing the model's interpretability and providing clear explanations for predictions will be crucial for gaining trust among both patients and healthcare professionals.
5. **Global Accessibility:** Ensuring that the model is accessible and applicable to diverse populations worldwide will be essential for addressing heart disease on a global scale.
6. **Collaborative Research:** Collaboration between data scientists, medical experts, and policymakers can facilitate the development of more robust and ethical heart disease prediction models.

8) REFERENCE

8.1) Datasets

Popular sources include Kaggle, UCI Machine Learning Repository

8.2) Books

Classification Projects on Machine Learning for Beginners is a comprehensive reference guide for beginners.

8.3) Papers

Heart disease prediction using machine learning algorithms.
Harshit Jindal, Sarthak Agrawal, Rishabh Khera, Rachna Jain and Preeti Nagrath

https://www.geeksforgeeks.org/ml-introduction-data-machine-learning/?ref=lbp
https://www.altexsoft.com/blog/datascience/machine-learning-project-structure-stages-roles-and-tools/
https://www.javatpoint.com/how-to-get-datasets-for-machine-learning
https://www.researchgate.net/publication/344717762_Machine_Learning_Algorithms_-A_Review