Capstone Project

Topic :- Cardiovascular Risk Prediction Project

Akash Choudhury

Table Of Content

Introduction

- Cardiovascular disease is a general classification of various infections that affect the heart and veins. The most important behavioral risk factors for heart disease and stroke are physical inactivity and tobacco/alcohol use. This raises blood pressure, blood sugar ,obesity, etc.
- This dataset is from an ongoing cardiovascular study of residents of the town of Framingham, Massachusetts. This dataset provides pieces of information to the patient. It contains over 3390 records and 17 attributes.
- Database contains patients in the age group from 32 to 70 years. This project used the machine learning (supervised) classification algorithm.

Problem Statement

- To provide an overview of prediction models for risk of cardiovascular disease (CVD) in the general population.
- The classification goal is to predict whether the patient has a 10-year risk of future coronary heart disease (CHD).

Data Summary

Demographic:

- Sex: male or female("M" or "F")
- Age: Age of the patient;(Continuous - Although the recorded ages have been truncated to whole numbers, the concept of age is continuous)

Behavioral :

- is_smoking: whether or not the patient is a current smoker ("YES" or "NO")
- Cigs Per Day

Medical (history)

- **BP Meds: whether or not the patient was on blood pressure medication (Nominal)**
- **Prevalent Stroke: whether or not the patient had previously had a stroke (Nominal)**
- **Prevalent Hyp: whether or not the patient was hypertensive (Nominal)**
- **Diabetes: whether or not the patient had diabetes (Nominal)**

Medical(current)

- Tot Chol: total cholesterol level (Continuous)
- Sys BP: systolic blood pressure (Continuous)
- Dia BP: diastolic blood pressure (Continuous)
- BMI: Body Mass Index (Continuous)
- Heart Rate: heart rate (Continuous - In medical research, variables such as heart rate though in fact discrete, yet are considered continuous because of large number of possible values.)
- Glucose: glucose level (Continuous)

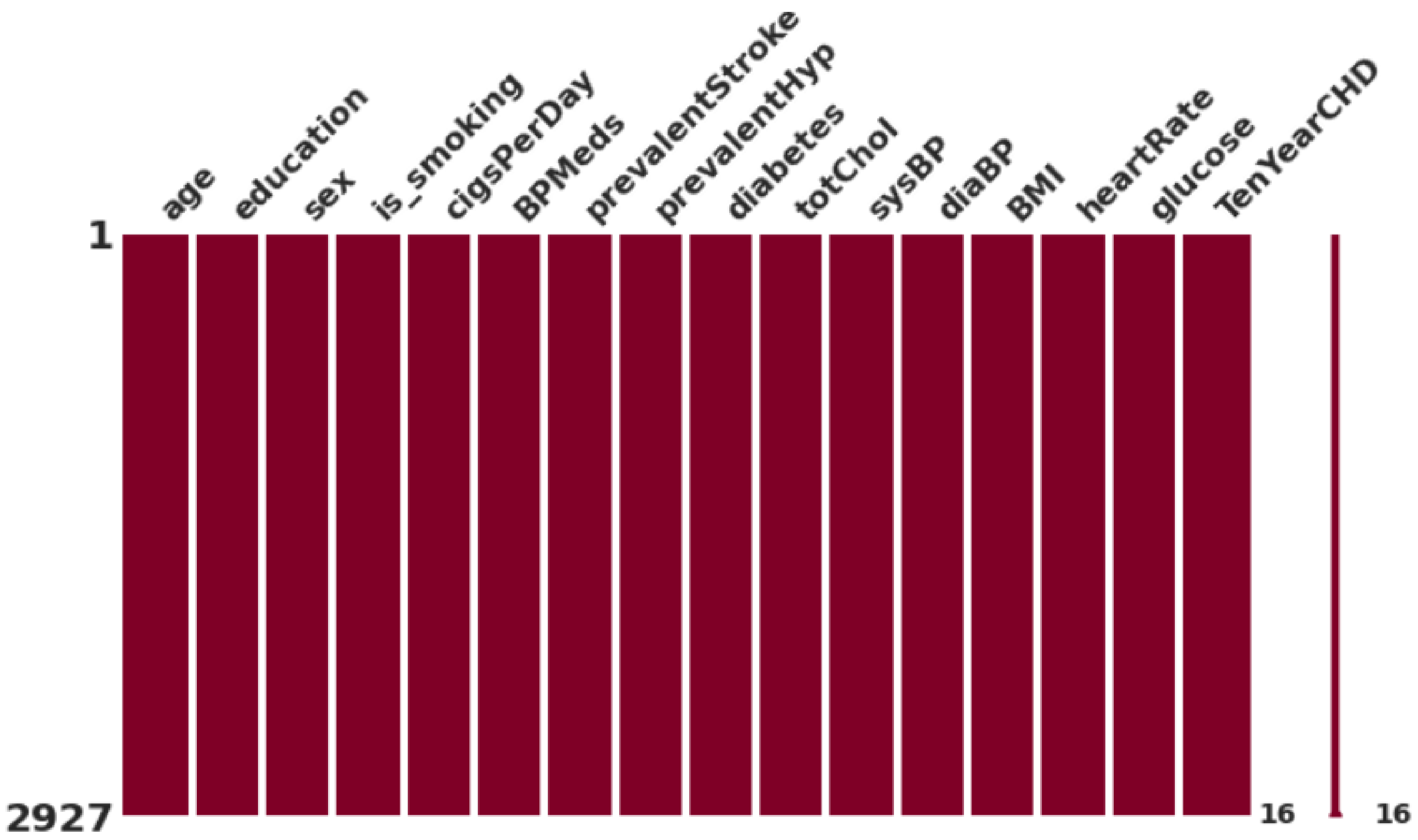
Predict variable (desired target)

10-year risk of coronary heart disease CHD(binary: “1”, means “Yes”, “0” means “No”) - Dv

Data Wrangling

- Columns and their unique values to understand what they contain
- Data Cleaning
- Handling missing Null values.

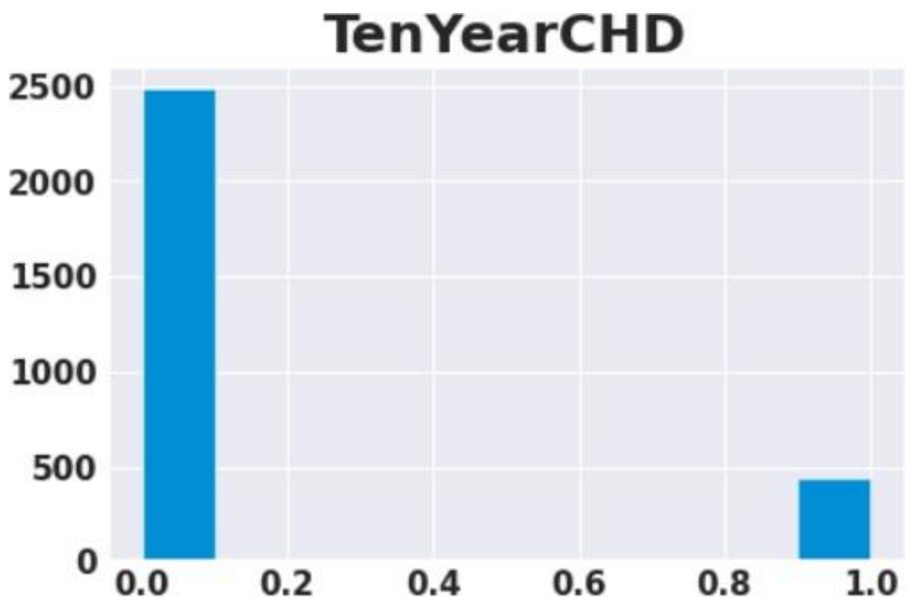
Visulisation of Data Cleaning



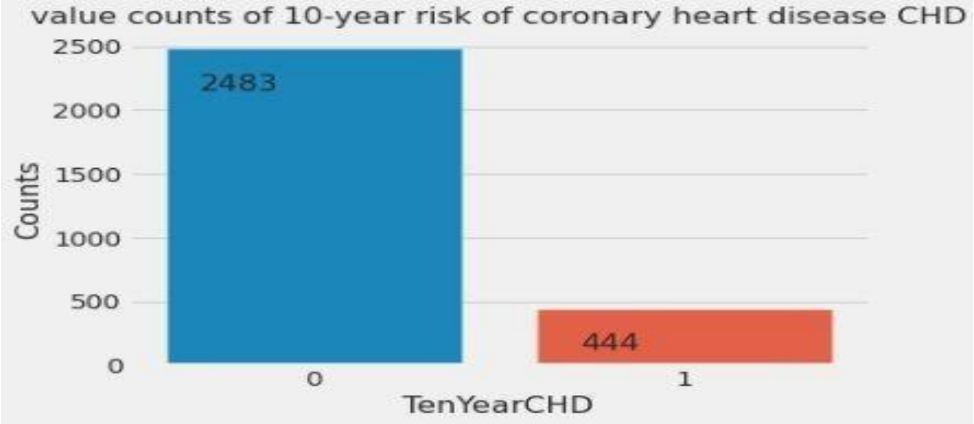
Exploratory Data Analysis

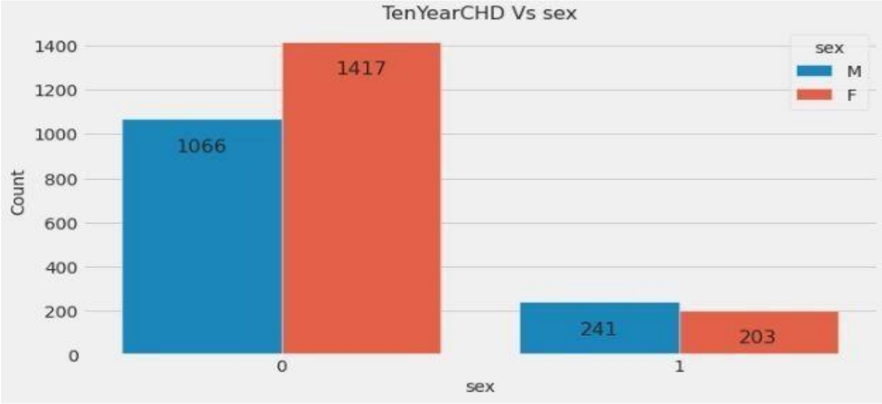
Exploratory Data Analysis

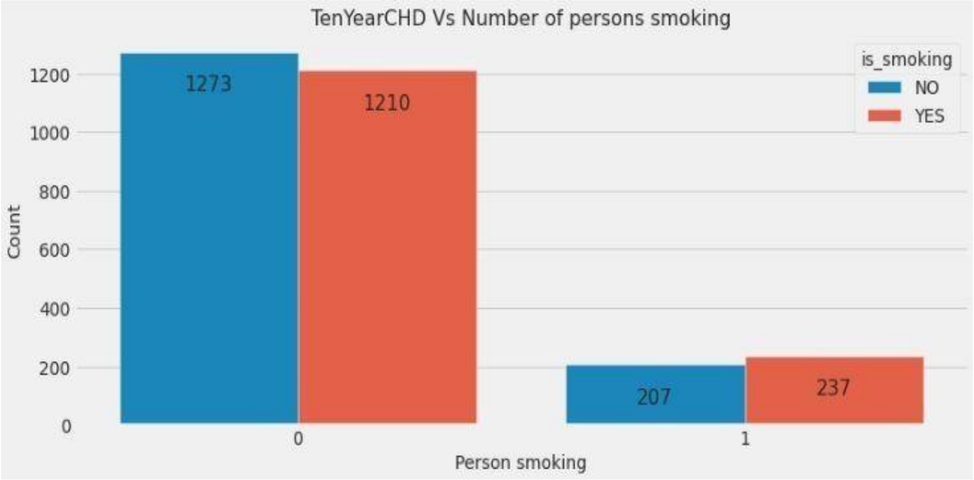
- Let's take a look at the plots. It shows how each feature and label is distributed along different ranges, which further confirms the need for scaling.
- Next, wherever you see discrete bars, it basically means that each of these is actually a categorical variable.
- We will need to handle these categorical variables before applying Machine Learning. Our target labels have two classes, 0 for no disease and 1 for disease.

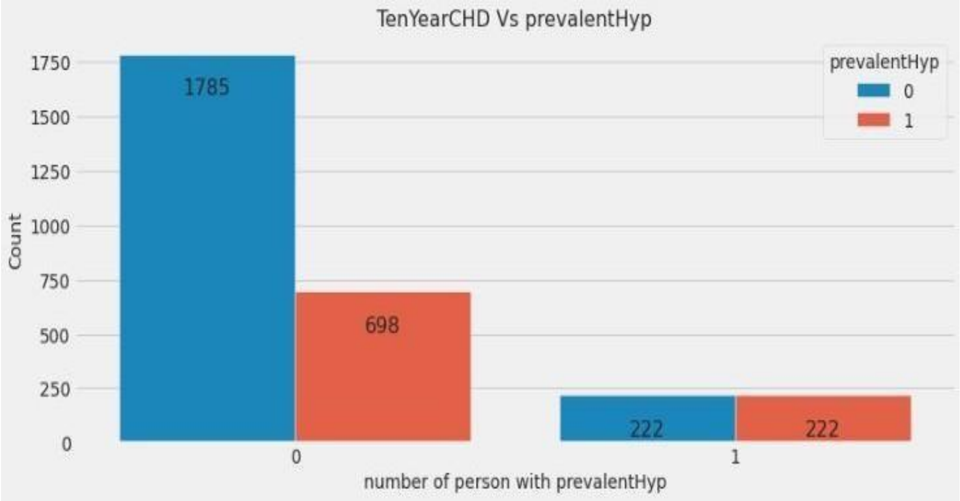


Visualization on Dependent and Independent Variables

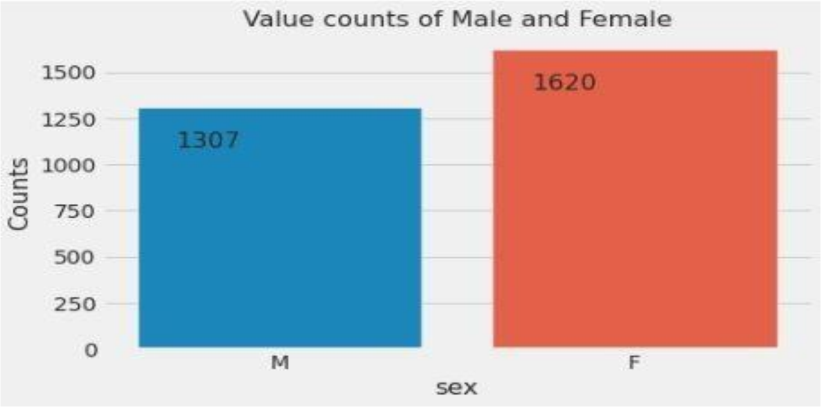


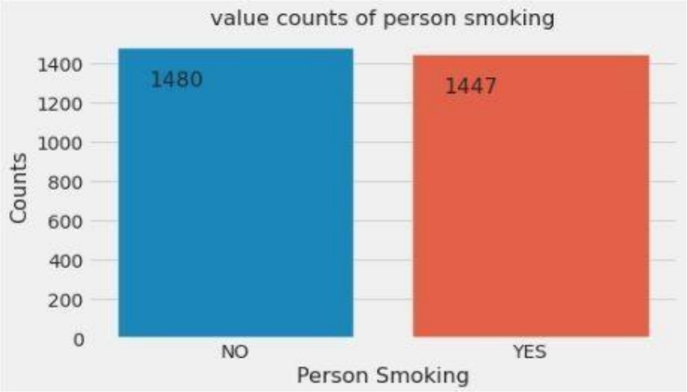


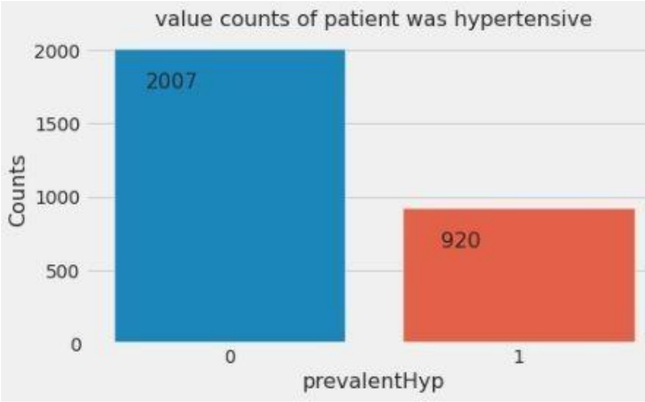


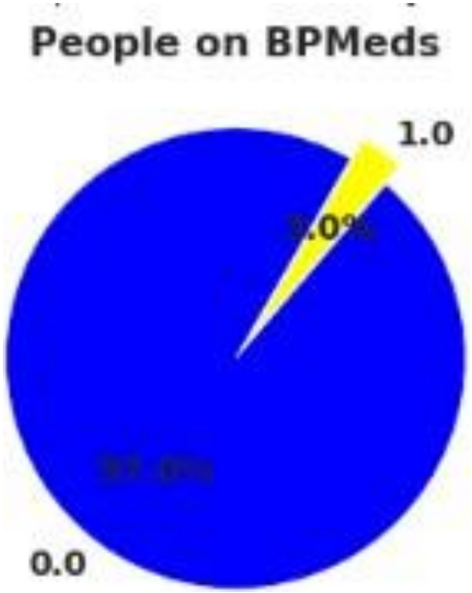


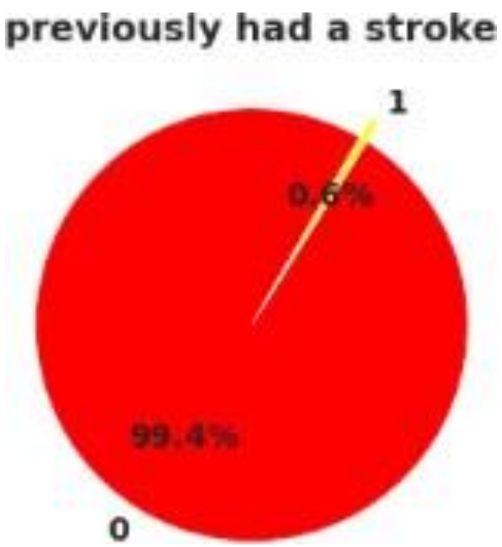
Analysis by Value Counts of some features

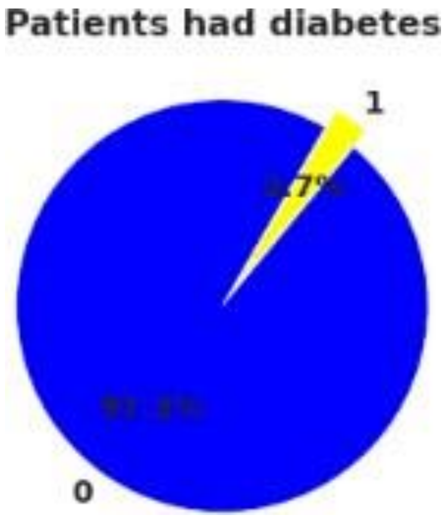




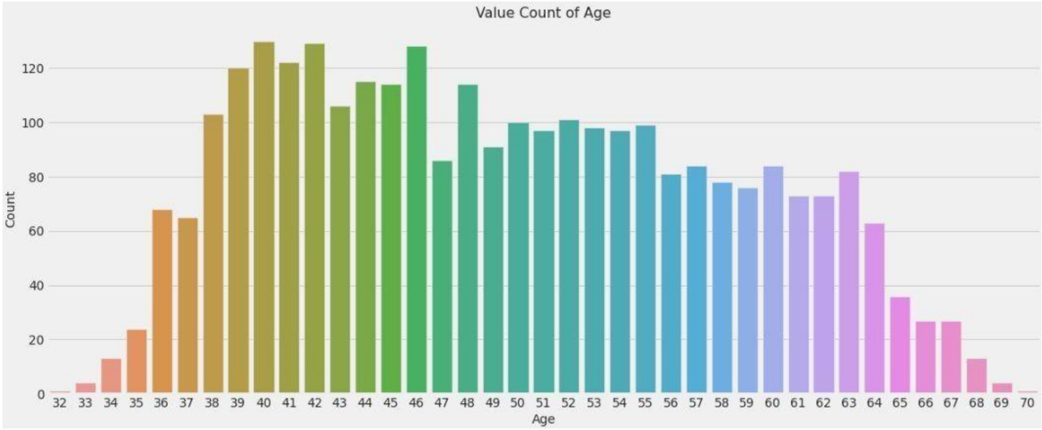


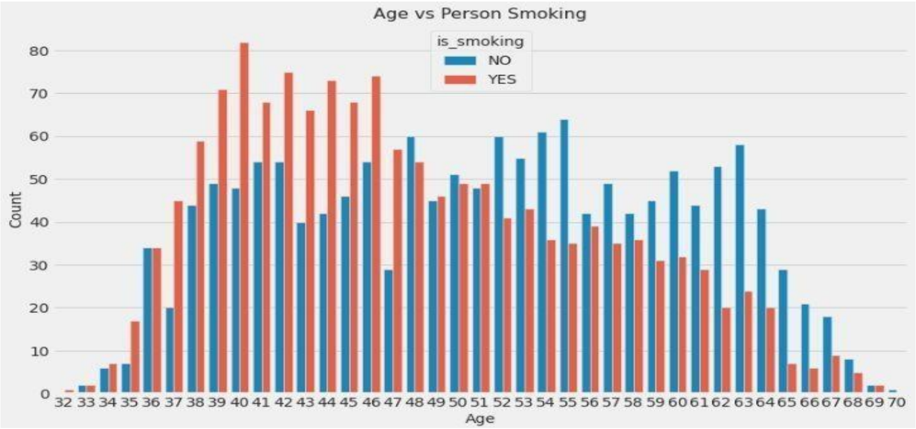


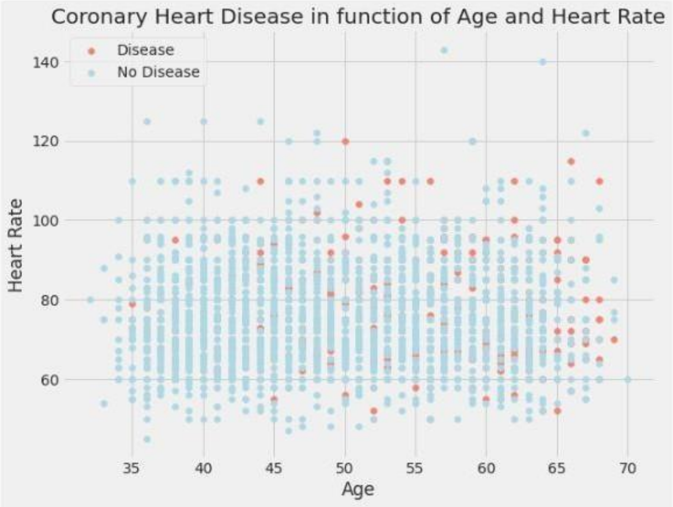




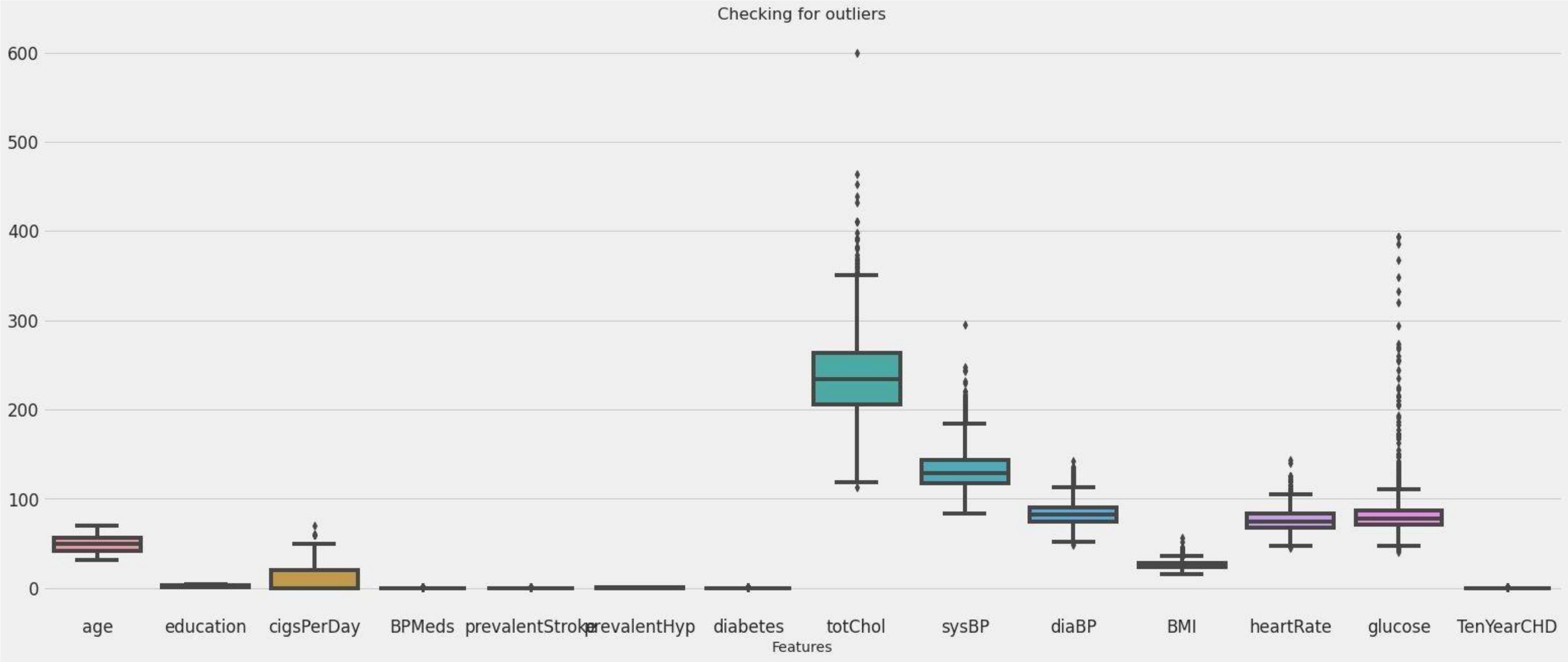
Bar plot and Scatter Plot for important variables







Feature Selection
Outliers



Heatmap

- Some of the features have a negative correlation with the target value and some have positive.

- Heart Rate and Prevalent Stroke are the lowest correlated with the target variable.

Data Preparation

- After exploring the dataset, we observed that we need to convert some categorical variables into dummy variables and scale all the values before training the Machine Learning models.
- First, we will use the get_dummies method to create dummy columns for categorical variables

```
# Adding pulse pressure as a column
df['pulsePressure'] = df['sysBP'] - df['diaBP']
# Dropping the systolic and diastolic BP columns
df.drop(['sysBP','diaBP'], axis = 1, inplace = True)
# Dropping the 'is_smoking' column
df.drop('is_smoking', axis = 1, inplace = True)
```

```
# To get the Categorical Variables
categorical_val = []
continous_val = []
for column in df.columns:
    if len(df[column].unique()) <= 10:
        categorical_val.append(column)
    else:
        continous_val.append(column)
categorical_val
```

One Hot Encoding

```
# Creating dummy variables-
categorical_val.remove('TenYearCHD')
df=pd.get_dummies(df, columns = categorical_val)
```

Synthetic Minority Oversampling Technique(SMOTE)

```
# Importing SMOTE
from imblearn.over_sampling import SMOTE
...#Synthetic Minority Oversampling Technique
# transform the dataset
# Creating an instance for SMOTE oversample
= SMOTE()
X = df.drop('TenYearCHD', axis=1)   y =
df.TenYearCHD
# The rows and columns of X and y
print(f'X has {X.shape[ 0]} rows and
print(f'y has {y.shape[0]} rows')    # Using
SMOTE to oversample
X, y = oversample.fit_resample(X, y)
```

As there exits a clear imbalance in the classes. Hence, we used SMOTE to oversample the classes which are in less number.

Model Implementation

Standard Scaler Transformation

```
from sklearn.preprocessing import StandardScaler

s_sc = StandardScaler()
col_to_scale = ['age', 'cigsPerDay', 'totChol', 'BMI',

'heartRate' , 'glucose' , 'pulsePressure' ]
df[col_to_scale] = s_sc.fit_transform(df[col_to_scale])
df.head()
```

Train and Test data sets

```
# Importing packages to split data into train and test
from sklearn.model_selection  import train_test_split
X = df.drop('TenYearCHD', axis=1)
y = df.TenYearCHD

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=
```

LogisticRegression

```
Train Result:
================================================
Accuracy Score: 85.94%
_______________________________________________
CLASSIFICATION REPORT:
                     0           1  accuracy    macro avg  weighted avg
precision     0.860534    0.769231  0.859375     0.814882      0.847070
recall        0.996564    0.066225  0.859375     0.531394      0.859375
f1-score      0.923567    0.121951  0.859375     0.522759      0.805360
support    1746.000000  302.000000  0.859375  2048.000000   2048.000000
_______________________________________________
Confusion Matrix:
 [[1740    6]
 [ 282   20]]

Test Result:
================================================
Accuracy Score: 84.87%
_______________________________________________
CLASSIFICATION REPORT:
                    0           1  accuracy   macro avg  weighted avg
precision    0.848730    0.846154  0.848692    0.847442      0.848314
recall       0.997286    0.077465  0.848692    0.537376      0.848692
f1-score     0.917031    0.141935  0.848692    0.529483      0.791816
support    737.000000  142.000000  0.848692  879.000000    879.000000
_______________________________________________
Confusion Matrix:
 [[735   2]
 [131  11]]
```

K-Nearest Neighbors

```
Train Result:
================================================
Accuracy Score: 86.67%
_______________________________________________
CLASSIFICATION REPORT:
                     0           1  accuracy    macro avg  weighted avg
precision     0.876341    0.659341  0.866699     0.767841      0.844342
recall        0.982245    0.198675  0.866699     0.590460      0.866699
f1-score      0.926276    0.305344  0.866699     0.615810      0.834713
support    1746.000000  302.000000  0.866699  2048.000000   2048.000000
_______________________________________________
Confusion Matrix:
 [[1715   31]
 [ 242   60]]

Test Result:
================================================
Accuracy Score: 82.82%
_______________________________________________
CLASSIFICATION REPORT:
                    0           1  accuracy   macro avg  weighted avg
precision    0.845519    0.354839  0.828214    0.600179      0.766251
recall       0.972863    0.077465  0.828214    0.525164      0.828214
f1-score     0.904732    0.127168  0.828214    0.515950      0.779119
support    737.000000  142.000000  0.828214  879.000000    879.000000
_______________________________________________
Confusion Matrix:
 [[717  20]
 [131  11]]
```

Support Vector Machine

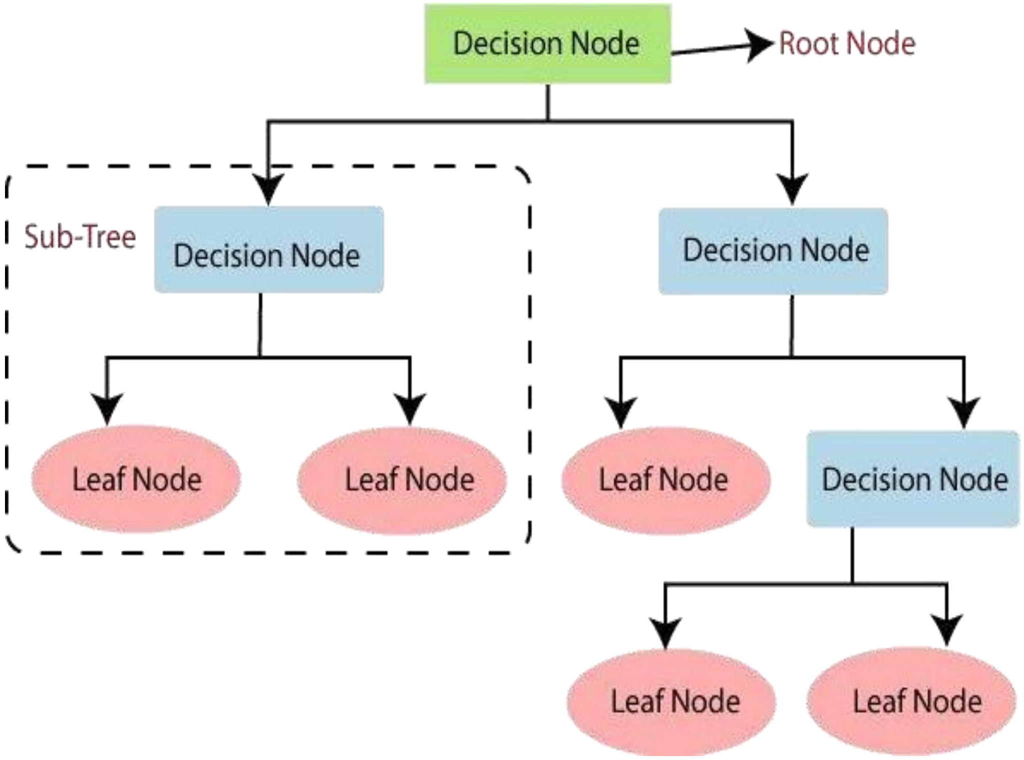
```
Train Result:
================================================
Accuracy Score: 85.99%
_______________________________________________
CLASSIFICATION REPORT:
                     0           1  accuracy    macro avg  weighted avg
precision     0.858829    1.000000  0.859863     0.929415      0.879646
recall        1.000000    0.049669  0.859863     0.524834      0.859863
f1-score      0.924054    0.094637  0.859863     0.509346      0.801747
support    1746.000000  302.000000  0.859863  2048.000000   2048.000000
_______________________________________________
Confusion Matrix:
 [[1746    0]
 [ 287   15]]

Test Result:
================================================
Accuracy Score: 83.85%
_______________________________________________
CLASSIFICATION REPORT:
                    0      1  accuracy   macro avg  weighted avg
precision    0.838453    0.0  0.838453    0.419226      0.703003
recall       1.000000    0.0  0.838453    0.500000      0.838453
f1-score     0.912129    0.0  0.838453    0.456064      0.764777
support    737.000000  142.0  0.838453  879.000000    879.000000
_______________________________________________
Confusion Matrix:
 [[737   0]
 [142   0]]
```

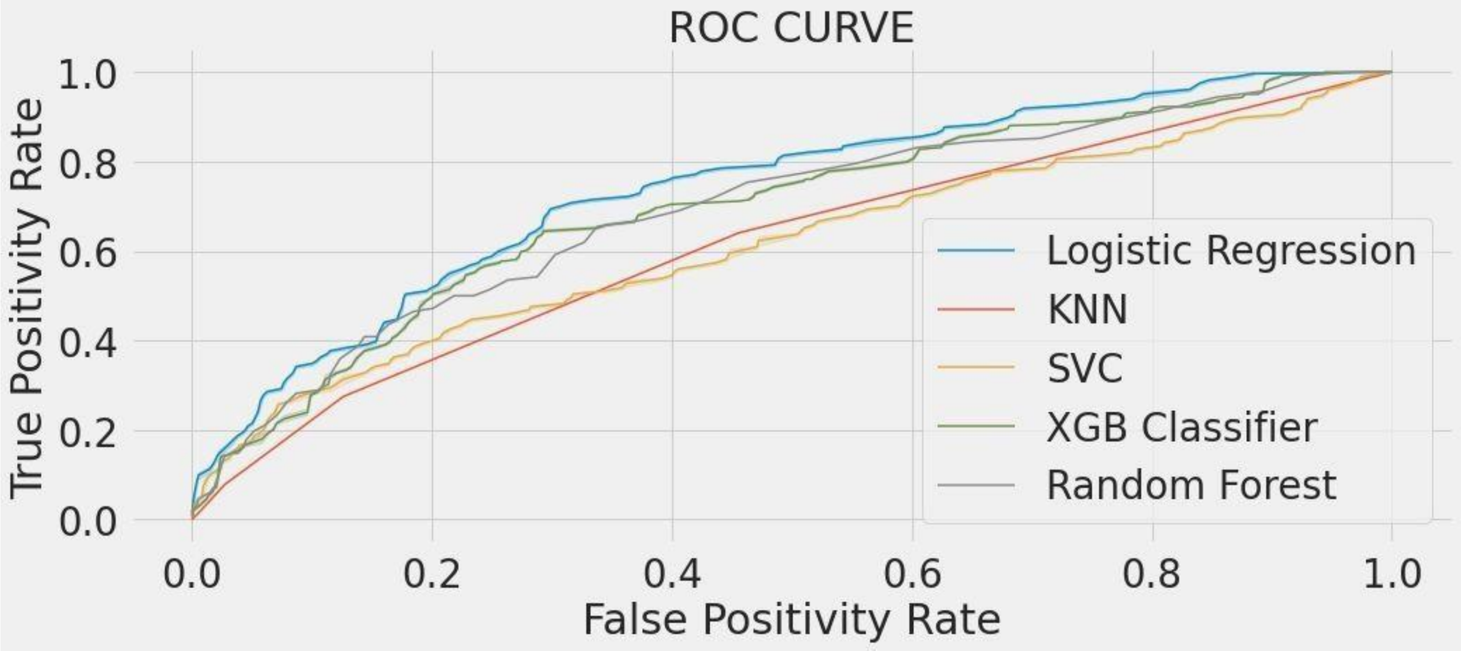
Decision Tree Classifier

```
Train Result:
================================================
Accuracy Score: 100.00%
_______________________________________________
CLASSIFICATION REPORT:
                0      1  accuracy  macro avg  weighted avg
precision     1.0    1.0       1.0        1.0           1.0
recall        1.0    1.0       1.0        1.0           1.0
f1-score      1.0    1.0       1.0        1.0           1.0
support    1746.0  302.0       1.0     2048.0        2048.0
_______________________________________________
Confusion Matrix:
 [[1746    0]
 [   0  302]]

Test Result:
================================================
Accuracy Score: 77.36%
_______________________________________________
CLASSIFICATION REPORT:
                    0           1  accuracy   macro avg  weighted avg
precision    0.862534    0.291971  0.773606    0.577252      0.770361
recall       0.868385    0.281690  0.773606    0.575038      0.773606
f1-score     0.865450    0.286738  0.773606    0.576094      0.771960
support    737.000000  142.000000  0.773606  879.000000    879.000000
_______________________________________________
Confusion Matrix:
 [[640  97]
 [102  40]]
```



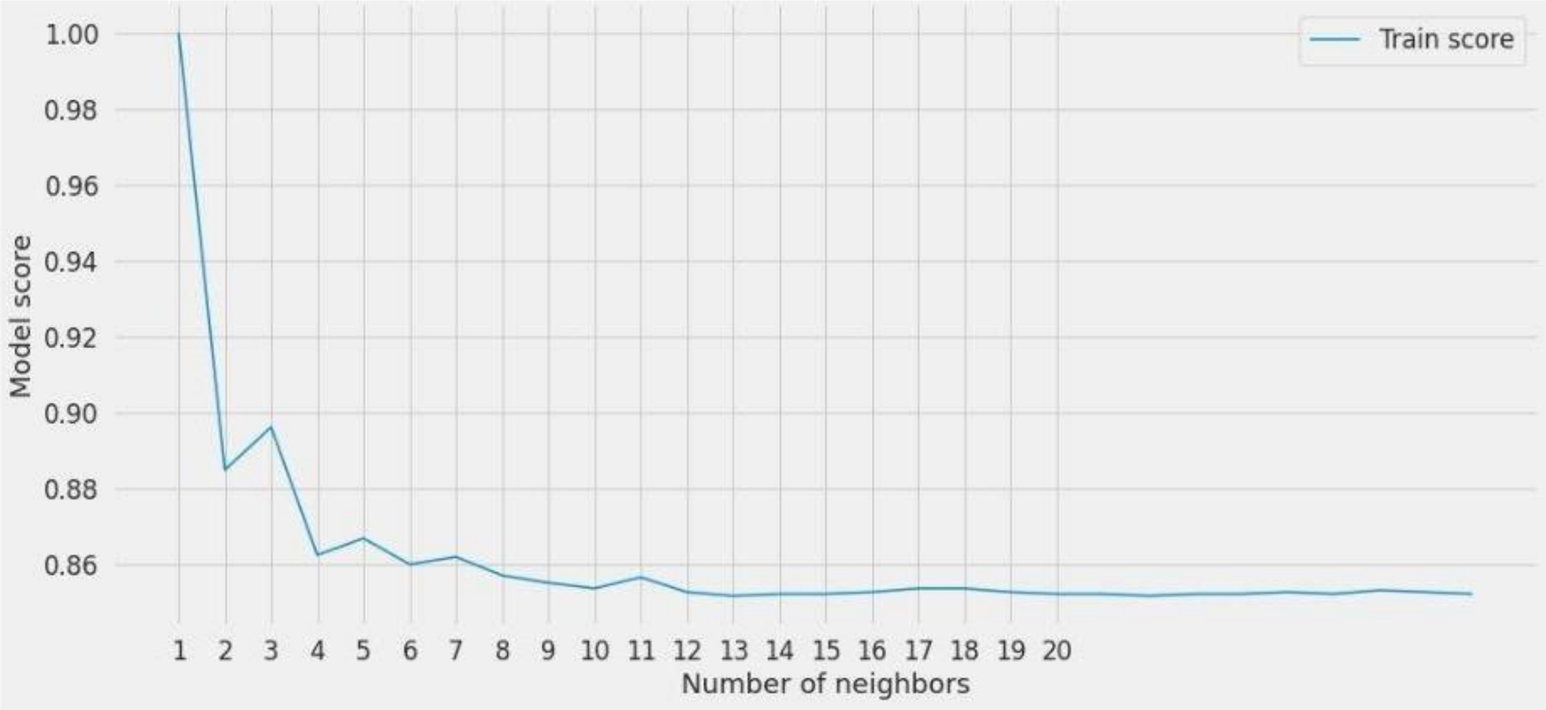
Diagramatic Representation of Models-ROC Curve



Receiver Operating Classifier curve of a purely random classifier; a good classifier stays as far away from that line as possible (toward the top-left corner).

The more that the curve hugs the top left corner of the plot, the better the model does at classifying the data into categories

Hyperparameter Tuning (K-Nearest Neighbors)



Test Accuracy using

Hyperparameter Tuning

= 84.07

Challenges

- Remove The Null Values
- Feature Selection Outliners
- Accuracy of Confusion Matrix.
- Handling Class Imbalance.
- Model training, tuning and performance improvement.

Future Improvement

- For future improvement in the model fitting for Coronary Heart Disease, we can perform the Random Forest Classifier, XGBoost models also.
- Consulting medical people we can analyze the feature in proper and required manner to approach the disease cause and effects

Conclusion

	Model	Training Accuracy %	Testing Accuracy %
0	Logistic Regression	85.937500	84.869170
1	K-nearest neighbors	86.669922	82.821388
2	Support Vector Machine	85.986328	83.845279
3	Decision Tree Classifier	100.000000	77.360637

	Model	Training Accuracy %	Testing Accuracy %
0	Tuned K-nearest neighbors	85.302734	84.07281

- **Logistic Regression, K Nearest Neighbors, Support Vector Machines, and Decision Tree Classification Models have been implemented. From these above models, we found ANN to be the best model compared to the other model**
- **In Hyperparameter tuning ,we observed that K-Nearest Neighbors accuracy has improved which shows that KNN (with Hyperparameter Tuning) is the best fitted model for Coronary Heart Disease dataset.**
- **Training Accuracy = 85.30 & Testing Accuracy = 84.07**
- **We can also run random forest classifiers and XGBoost models for improved future coronary artery disease model fitting. By consulting the medical staff, the characteristics can be analyzed in an appropriate and necessary way to address the causes and consequences of the disease.**

Thank You