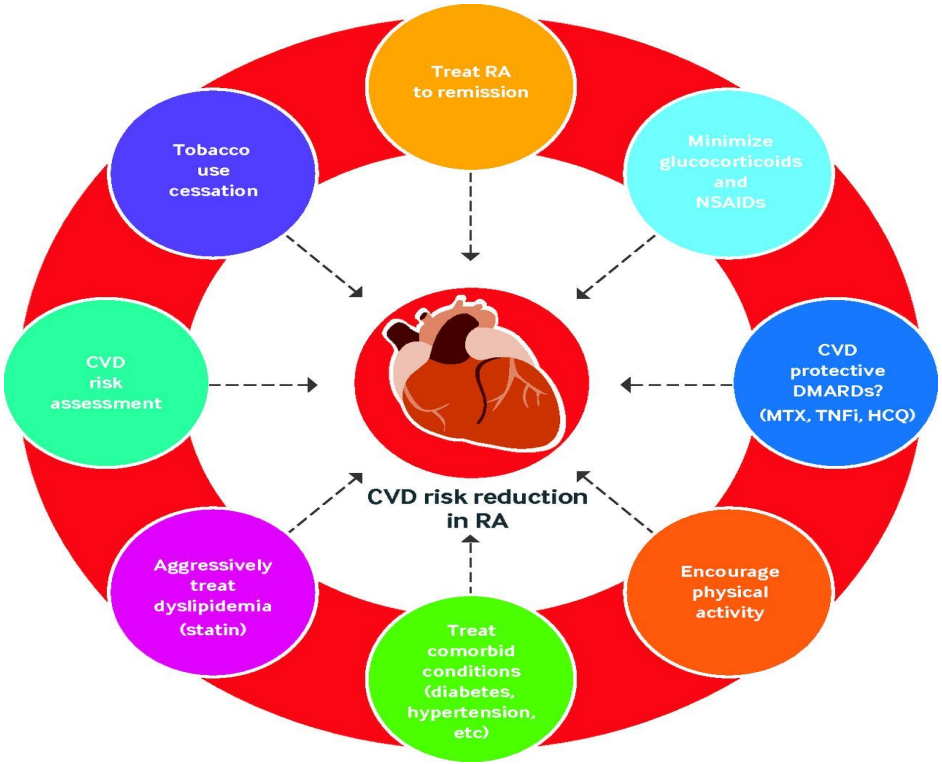
Capstone Project

Classification Analysis on Cardiovascular Risk Prediction

Rohan Jagadale
Data science trainee at Almabetter

Flow of the Presentation



- Introduction
- Problem Statement
- Exploratory Data Analysis
- Feature Selection
- Data Preparation
- Model Implementation
- Evaluation of model
- conclusion

Introduction

Cardiovascular Risk:

- The most important behavioural risk factors of heart disease and stroke are , physical inactivity, tobacco use and harmful use of alcohol. The effects of behavioural risk factors may show up in individuals as raised blood pressure, raised blood glucose, raised blood lipids, and overweight and obesity

Methodology:

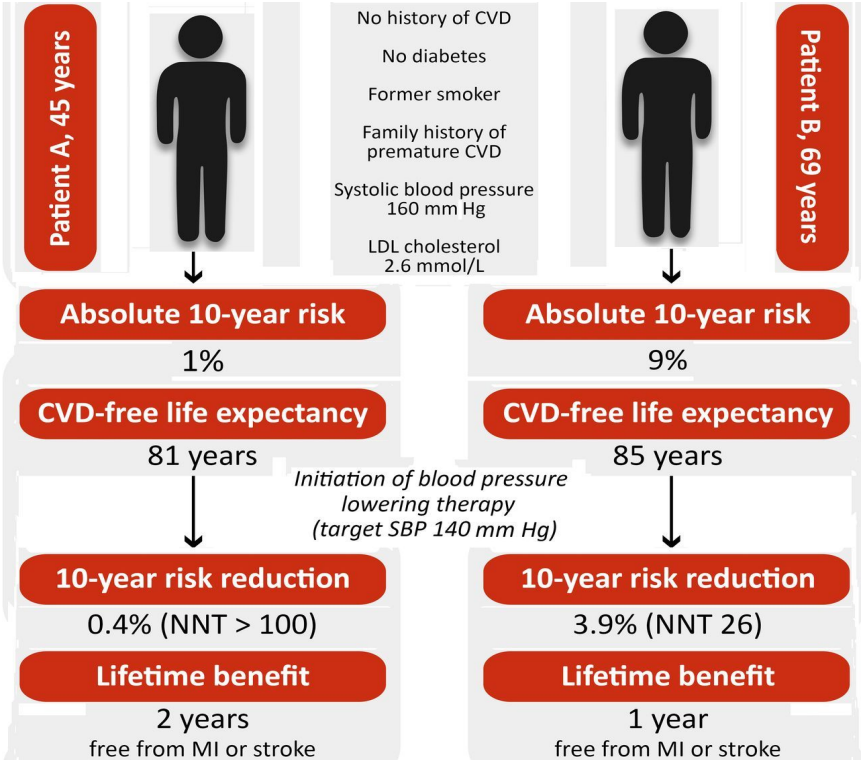
- Supervised Machine Learning (Classification)

Database:

- Cardiovascular Risk Database
- 3389 rows and 17 columns
- Patients age group 32 to 70

Problem Statement

- The dataset is from an ongoing cardiovascular study on residents of the town of Framingham, Massachusetts.
- The classification goal is to predict whether the patient has a 10-year risk of future coronary heart disease (CHD).



Data Description

Demographic:

• Sex: male or female("M" or "F")

• Age: Age of the patient

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 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 a large number of possible values.)

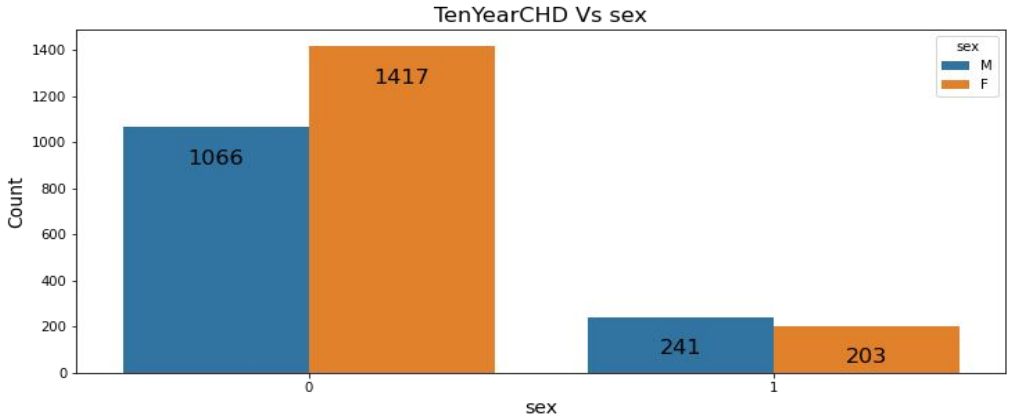
• Glucose: glucose level (Continuous)

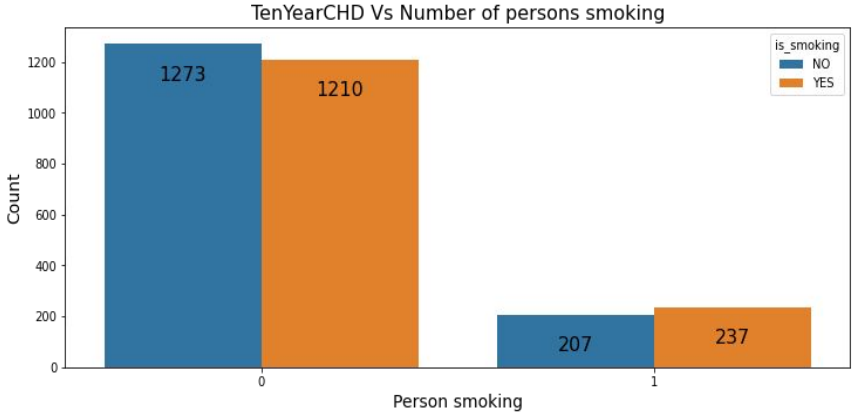
•Predict variable (desired target)

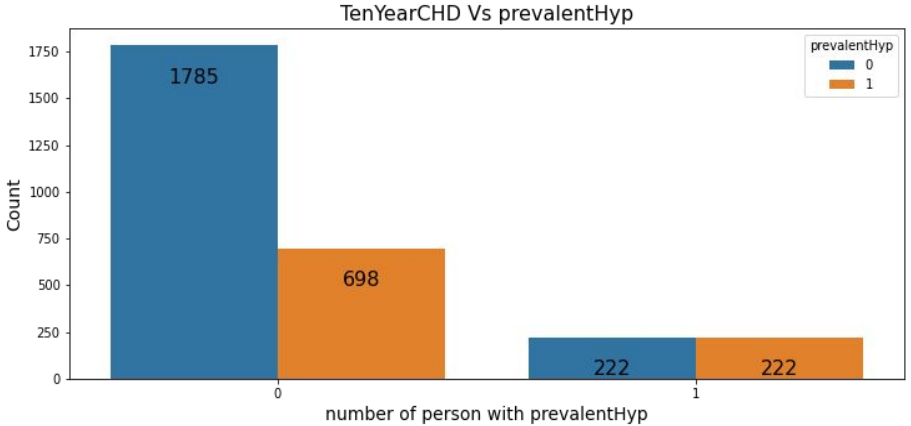
10-year risk of coronary heart disease CHD(binary: “1”, means “Yes”, “0” means “No”) - Dv

Defining the Dependent Variable

- Ten years CHD is taken as the dependent variable. 10-year risk of coronary heart disease : Contains the binary values, 1 means that the patients has a CHD and 0 means patients has no CHD

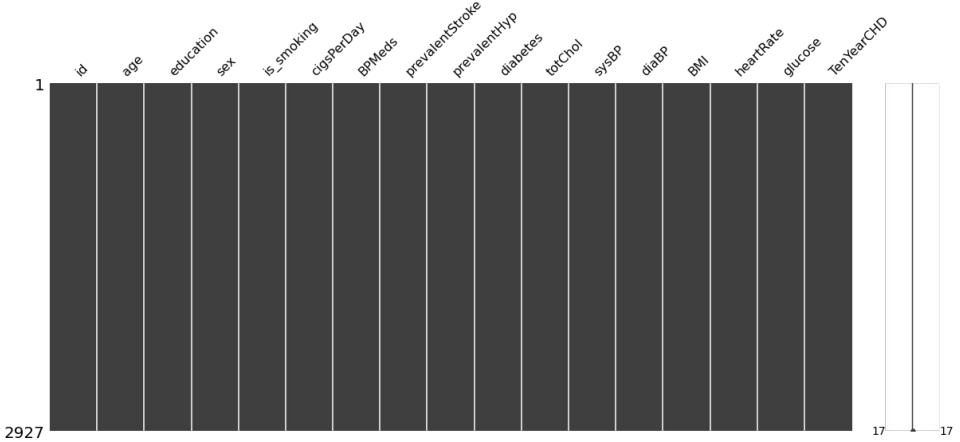


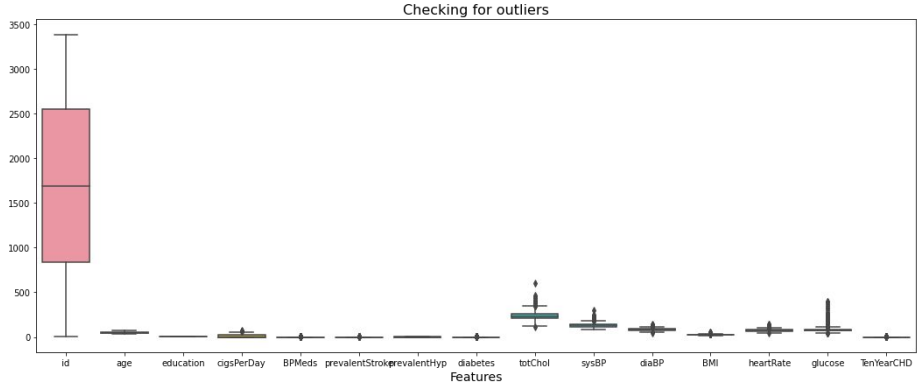


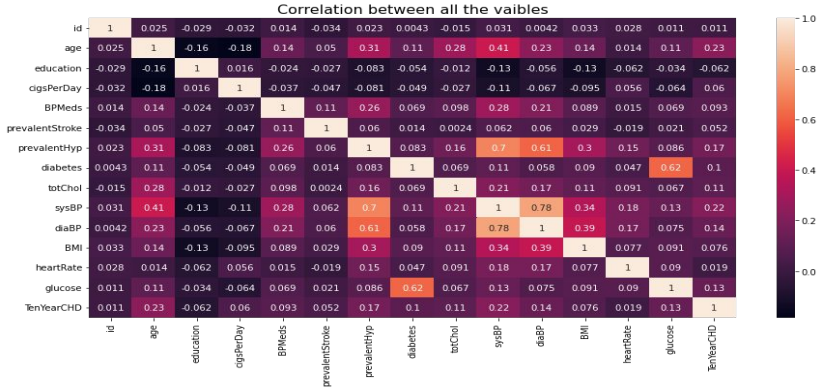


Exploratory Data Analysis

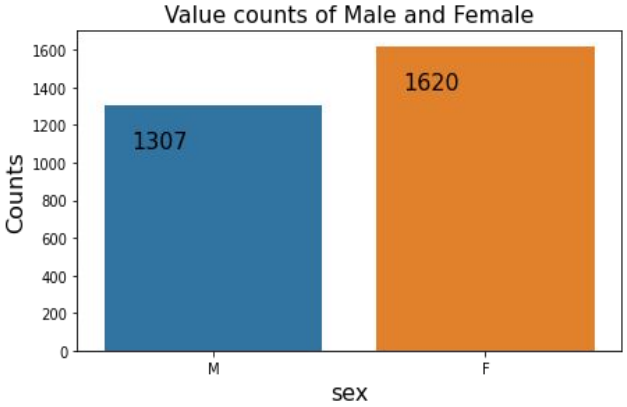
- Check Null Values
- Finding outliers
- Checking correlation
- Visualization on independent variables
- Feature selection (VIF)

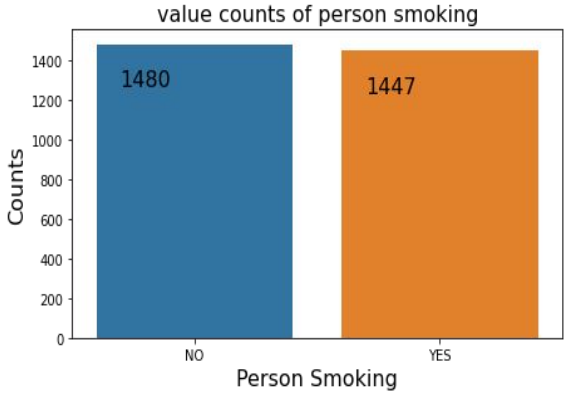


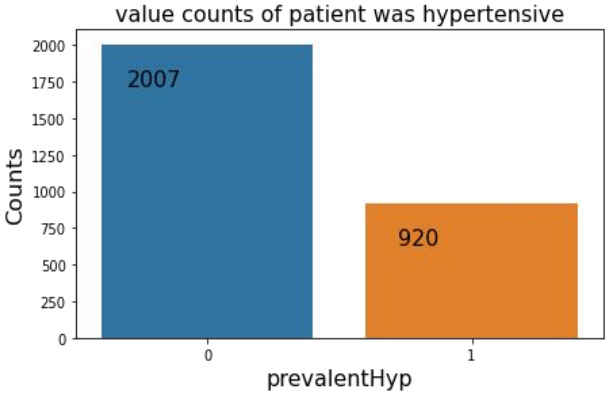


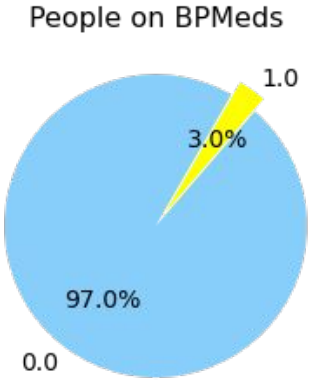


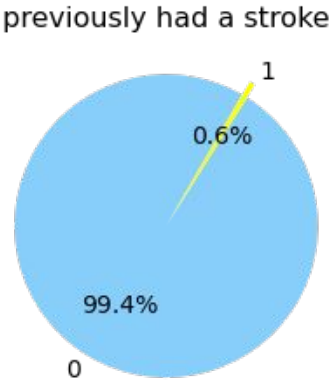
Exploratory Data Analysis

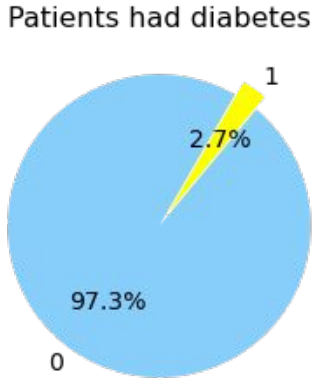




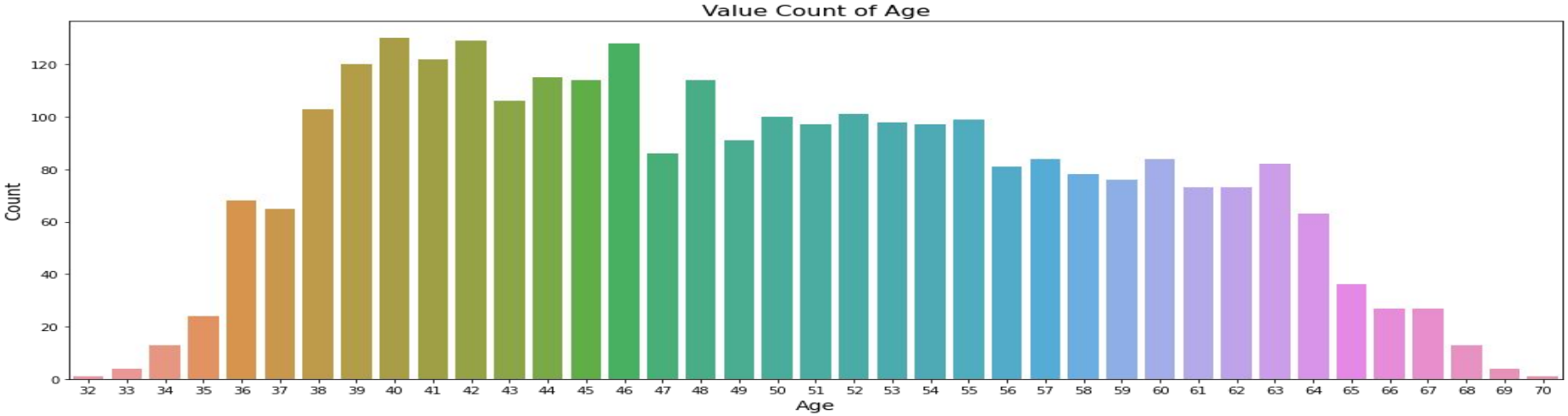


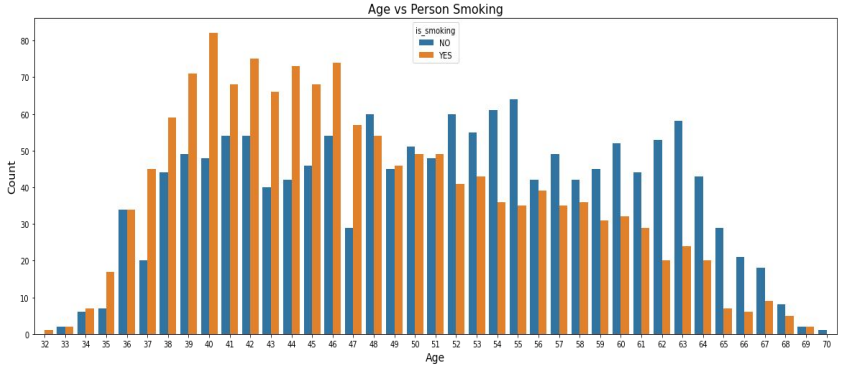


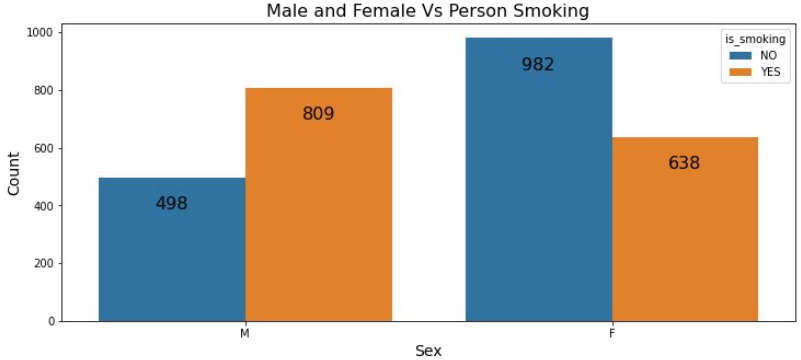




Exploratory Data Analysis







Data Preparation

- Before the model implementation the Variance Inflation Factor was used to do feature selection
- Divided the dataset into train and test set in the ratio of 80:20 with random_state as 42
- StandardScaler was used to scale the data

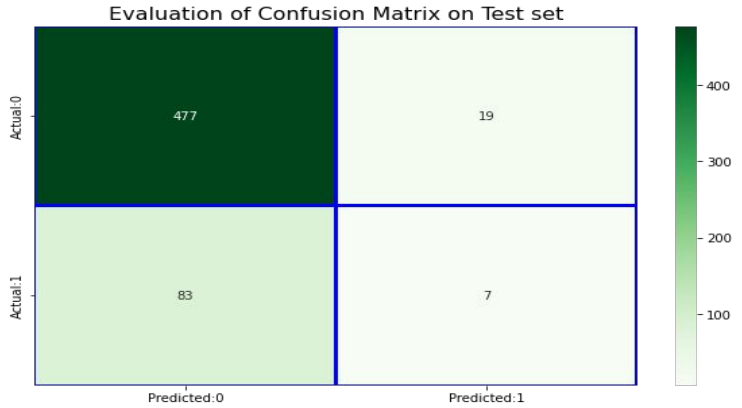
```
print(f'Size of X_train is: {X_train.shape}')
print(f'Size of X_test is: {X_test.shape}')
print(f'Size of y_train is: {y_train.shape}')
print(f'Size of y_test is: {y_test.shape}')
```

```
Size of X_train is: (2341, 8)
Size of X_test is: (586, 8)
Size of y_train is: (2341,)
Size of y_test is: (586,)
```

Model Implementation

1. Decision Tree Classifier :

- The Scaled data was used for the model implementation
- Fit the trained dataset to the model.
- The accuracy score on the test dataset is 82.76%




```
accuracy = accuracy_score(y_train,pred_train)
accuracy
```

```
0.8573259290901324
```

```
accuracy = accuracy_score(y_test,pred_test)
accuracy
```

```
0.8276450511945392
```

```
Classification Report is:
```

	precision	recall	f1-score	support
0	0.85	0.96	0.90	496
1	0.27	0.08	0.12	90
accuracy			0.83	586
macro avg	0.56	0.52	0.51	586
weighted avg	0.76	0.83	0.78	586

Model Implementation

2. Logistic Regression :

- The Scaled data was used for the model implementation
- Fit the trained dataset to the model.
- The accuracy score on the test dataset is 85.32%

```
Train_accuracy = accuracy_score(y_train,pred_train)
Train_accuracy
```

```
0.8521999145664246
```

```
Test_accuracy = accuracy_score(y_test,pred_test)
Test_accuracy
```

```
0.8532423208191127
```

```
scoring = ['accuracy']
scores = cross_validate(lr_clf, X_train, y_train, scoring=scoring, cv=5,
                        return_train_score = True, return_estimator = True,
```

```
[CV] START ....................................................................
[CV] END ..... accuracy: (train=0.850, test=0.855) total time=   0.0s
[CV] START ....................................................................
[CV] END ..... accuracy: (train=0.851, test=0.848) total time=   0.0s
[CV] START ....................................................................
[CV] END ..... accuracy: (train=0.853, test=0.853) total time=   0.0s
[CV] START ....................................................................
[CV] END ..... accuracy: (train=0.853, test=0.838) total time=   0.0s
[CV] START ....................................................................
[CV] END ..... accuracy: (train=0.851, test=0.853) total time=   0.0s
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    0.1s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:    0.1s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    0.1s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    0.1s finished
```

```
#accuracy of train set
scores['train_accuracy']
```

```
array([0.84989316, 0.85104111, 0.85317672, 0.85317672, 0.85050721])
```

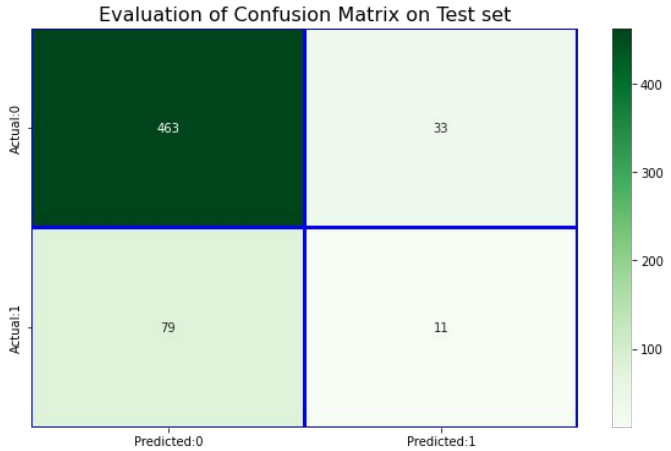
```
#accuracy of test set
scores['test_accuracy']
```

```
array([0.85501066, 0.8482906 , 0.8525641 , 0.83760684, 0.8525641 ])
```

Model Implementation

3. K-Neighbor Classifier :

- The Scaled data was used for the model implementation
- Fit the trained dataset to the model.
- The accuracy score on the test dataset is 80.88%




```
Train_accuracy = accuracy_score(y_train,pred_train)
Train_accuracy
```

```
0.8795386586928663
```

```
Test_accuracy = accuracy_score(y_test,pred_test)
Test_accuracy
```

```
0.8088737201365188
```

```
Classification Report is:
```

	precision	recall	f1-score	support
0	0.85	0.93	0.89	496
1	0.25	0.12	0.16	90
accuracy			0.81	586
macro avg	0.55	0.53	0.53	586
weighted avg	0.76	0.81	0.78	586

Model Implementation

4. Naive Bayes Classifier :

- The Scaled data was used for the model implementation
- Fit the trained dataset to the model.
- The accuracy score on the test dataset is 83.44%

Classification Report is:

	precision	recall	f1-score	support
0	0.86	0.95	0.90	1987
1	0.33	0.14	0.20	354
accuracy			0.83	2341
macro avg	0.60	0.54	0.55	2341
weighted avg	0.78	0.83	0.80	2341

```
Train_accuracy = accuracy_score(y_train,pred_train)
Train_accuracy
```

```
0.8278513455788125
```

```
Test_accuracy = accuracy_score(y_test,pred_test)
Test_accuracy
```

```
0.8344709897610921
```

Classification Report is:

	precision	recall	f1-score	support
0	0.86	0.97	0.91	496
1	0.37	0.11	0.17	90
accuracy			0.83	586
macro avg	0.61	0.54	0.54	586
weighted avg	0.78	0.83	0.79	586

Model Implementation

5. SVM Classifier :

- The Scaled data was used for the model implementation
- Fit the trained dataset to the model.
- The accuracy score on the test dataset is 84.64%

```
Train_accuracy = accuracy_score(y_train,pred_train)
Train_accuracy
```

```
0.8487825715506194
```

```
Test_accuracy = accuracy_score(y_test,pred_test)
Test_accuracy
```

```
0.8464163822525598
```

```
Classification Report is:

              precision    recall  f1-score   support

           0       0.85      1.00      0.92      1987
           1       0.00      0.00      0.00       354

    accuracy                           0.85      2341
   macro avg       0.42      0.50      0.46      2341
weighted avg       0.72      0.85      0.78      2341
```

```
Classification Report is:

              precision    recall  f1-score   support

           0       0.85      1.00      0.92       496
           1       0.00      0.00      0.00        90

    accuracy                           0.85       586
   macro avg       0.42      0.50      0.46       586
weighted avg       0.72      0.85      0.78       586
```

Model Implementation

6. Gradient Boosting Classifier :

- The Scaled data was used for the model implementation
- Fit the trained dataset to the model.
- The accuracy score on the test dataset is 81.65%

Gradient Boosting Classifier (TUNING)

```
gb_clf = ensemble.GradientBoostingClassifier(n_estimators=40)
```

```
Train_accuracy = accuracy_score(y_train,pred_train)
Train_accuracy
```

```
0.8735583084152072
```

```
Test_accuracy = accuracy_score(y_test,pred_test)
Test_accuracy
```

```
0.8395904436860068
```

```
Train_accuracy = accuracy_score(y_train,pred_train)
Train_accuracy
```

```
0.8624519436138403
```

```
Test_accuracy = accuracy_score(y_test,pred_test)
Test_accuracy
```

```
0.8481228668941979
```

Model Implementation

7. Random Forest Classifier

- The Scaled data was used for the model implementation
- Fit the trained dataset to the model.
- The accuracy score on the test dataset is 83.44%

```
n_estimators= [160,210,10]
max_depth = [25,35,1]
min_samples_split = [2,5,1]
min_samples_leaf = [1,5,1]
max_features= [4,10,1]
```

```
Train_accuracy = accuracy_score(y_train,pred_train)
Train_accuracy
```

```
1.0
```

```
Test_accuracy = accuracy_score(y_test,pred_test)
Test_accuracy
```

```
0.8344709897610921
```

Cross validation on Random Forest Classifier

```
from sklearn.model_selection import RandomizedSearchCV
rf_grid = RandomizedSearchCV(estimator= rf_model , param_distributions= random_grid
```

```
rf_grid.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 50 candidates, totalling 250 fits
RandomizedSearchCV(cv=5, estimator=RandomForestClassifier(), n_iter=50,
                   n_jobs=-1,
                   param_distributions={'max_depth': [25, 35, 1],
                                        'max_features': [4, 10, 1],
                                        'min_samples_leaf': [1, 5, 1],
                                        'min_samples_split': [2, 5, 1],
                                        'n_estimators': [160, 210, 10]},
```

```
rf_grid.best_params_
```

```
{'max_depth': 35,
 'max_features': 4,
 'min_samples_leaf': 5,
 'min_samples_split': 5,
 'n_estimators': 160}
```

```
Train Accuracy : 0.869
Test Accuracy : 0.840
```

Conclusion

- Defining dependent variables and EDA on Dataset.
- We have patients from 32 to 70 age group. Number of patients from 38 t0 46 age group is high with smoking habits.
- Number of female patients is higher than male patients.
- There are 1470 male patients in the dataset out of which 809 male patients smoke cigarettes.
- There are 1447 female patients in the dataset out of which 638 female patients smoke cigarettes.
- Number of patients with medical history like blood pressure medication, Diabetes, and patients who previously had a stroke is very low.
- Value counts of 10 year CHD in male patients is high.
- We used the 7 different models to train the model.
- Logistic Regression ,SVM classifier,Gradient Boosting Classifier(Tuning) these models give good accuracy on test data with 86%, 85%, 85% respectively.
- From this, we can say that the model should be trained with Logistic Regression which has 85.32% accuracy on test dataset

	Classifier	Train_accuracy	Test_accuracy
0	Decision Tree Classifier	0.825939	0.825939
1	Logistic Regression	0.852200	0.853242
2	K-Neighbors Classifier	0.879539	0.808874
3	Naive Bayes Classifier	0.827851	0.834471
4	SVM Classifier	0.848783	0.846416
5	Gradient Boosting Classifier	0.873558	0.839590
6	Gradient Boosting Classifier (Tuning)	0.862452	0.848123
7	Random Forest Classifier	1.000000	0.834471
8	Random Forest Classifier (Tuning)	0.868859	0.839590

Thank you