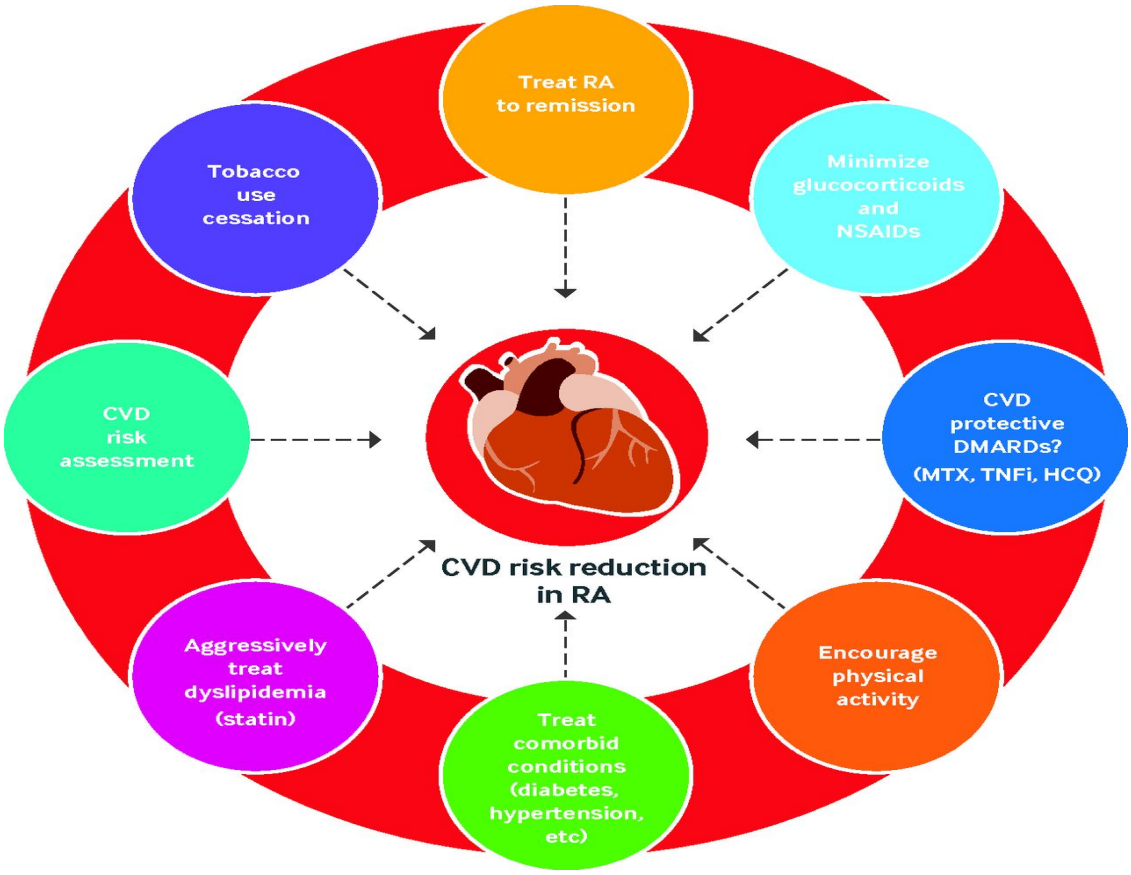
Classification Analysis on Cardiovascular Risk Prediction



Rohan Jagadale
Data science trainee at Almabetter

Abstract

Cardiovascular disease (CVD) is the leading cause of death worldwide. Early prediction of CVD is urgently important for timely prevention and treatment. Incorporation or modification of new risk factors that have an additional independent prognostic value of existing prediction models is widely used for improving the performance of the prediction models. This paper is to investigate the physiological parameters that are used as risk factors for the prediction of cardiovascular events, as well as summarizing the current status on the medical devices for physiological tests and discuss the potential implications for promoting CVD prevention and treatment in the future. The results show that measures extracted from blood pressure, electrocardiogram, arterial stiffness, ankle-brachial blood pressure index (ABI), and blood glucose carry valuable information for the prediction of both long-term and near-term cardiovascular risk.

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).

Introduction

Cardiovascular disease (CVD) is increasing daily in this modern world. According to the World Health Organization (WHO), an estimated 17 million people die each year from cardiovascular disease, particularly heart attacks and strokes [1]. It is, therefore, necessary to record the most important symptoms and health habits that contribute to CVD. Various tests are performed prior to diagnosis of CVD, including auscultation, ECG, blood pressure, cholesterol and blood sugar. These tests are often long and long when a patient's condition may be critical and he or she must start taking medication immediately, so it becomes important to prioritize the tests. Several health habits contribute to CVD. Therefore, it is also necessary to know which health habits contribute to CVD. Machine learning is now an emerging field due to the increasing amount of data. Machine learning makes it possible to acquire knowledge from a massive amount of data, which is very heavy

for man and sometimes impossible. The objective of this paper is to prioritize the diagnostic test and to see some of the health habits that contribute to CVD. Moreover, and above all, the different machine learning algorithms are compared using intelligent optimization algorithms. In this project, manually classified data is used. Manual classification is healthy or unhealthy. Based on a machine learning technique called classification, 80% of the data is supervised or trained and 20% is tested as part of this project.

Objective

The main objective is to build a model that can predict the patient's risk of developing Coronary Heart Disease **(CHD)** in the next **10 years.**

Dataset Peeping

- ➔ Owing to the size of the dataset, extensive cleaning of the dataset is not needed.
- ➔ Column names in the dataset are short but we already explain the meaning of these features. We did not rename the columns.
- ➔ Dataset contains NaN values.

Data Description

Attribute Description :

Each attribute is a potential risk factor. There are both demographic, behavioral, and medical risk factors.

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: the number of cigarettes that the person smoked on average in one day.(can be considered continuous as one can have any number of cigarettes, even half a cigarette.)

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 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

Challenges Faced

- Data set is very simple so I don't face any challenges during data pepping.

Approach

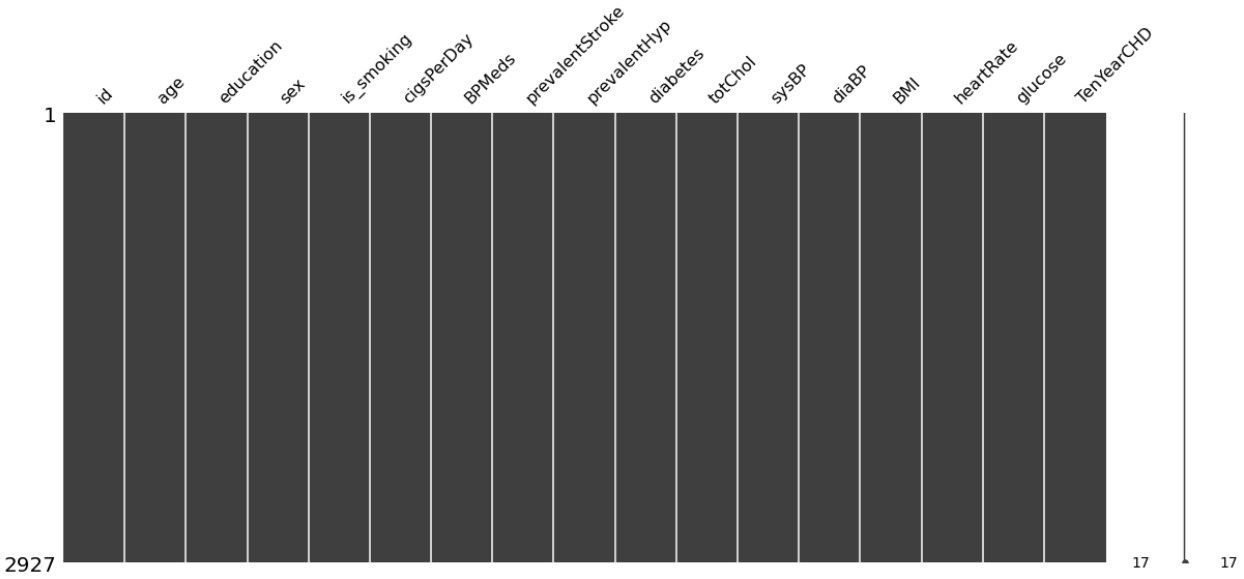
As the problem statement says the main objective is to predict whether the patient has a 10-year risk of future coronary heart disease (CHD). I have used the Decision Tree Classifier,Logistic Regression, Random Forest Classifier, K Near Neighbor, Naive Bayes Classifier, SVM Classifier, Gradient Boosting Classifier to train the model.

Tools Used

The whole project was done using python, in google collaborator. Following libraries were used for analyzing the data and visualizing:

- Pandas: Extensively used to load and wrangle with the dataset.
- Matplotlib: Used for visualization.
- Seaborn: Used for visualization.
- Warnings: For filtering and ignoring warnings.
- Numpy: For some math operations in predictions.
- Sklearn: For analysis and prediction.
- Stats models: For outliers influence.

Visualizing the presence of NaN values



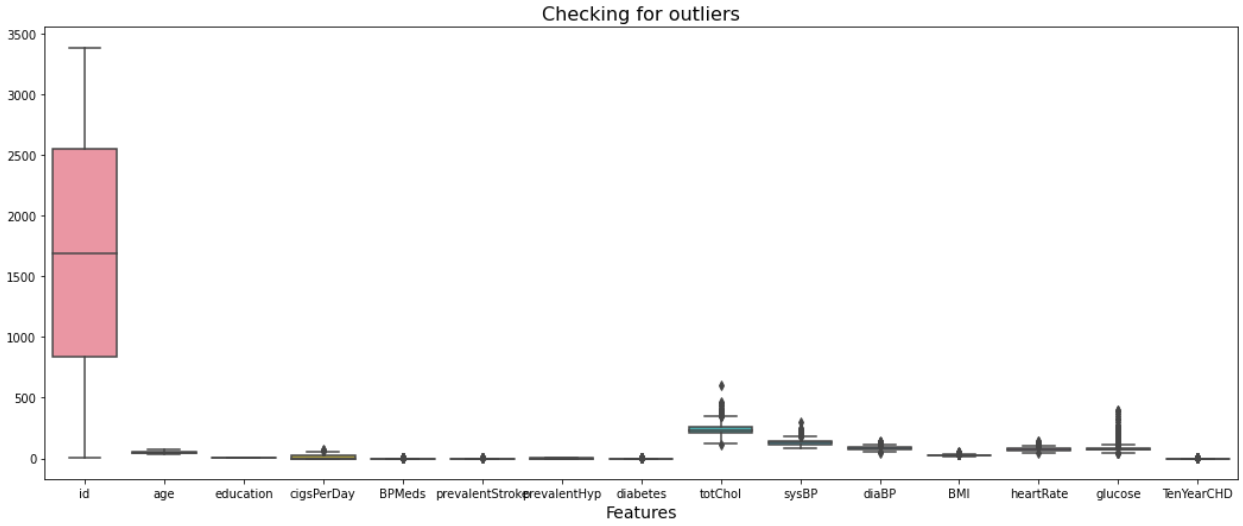
The above figure shows that there are no NaN (Not a Number) values in the given dataset.

Pandas DataFrame

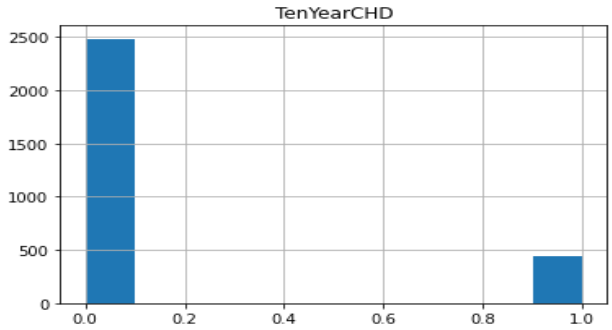
id	age	education	sex	is_smoking	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp
0	64	2.0	F	YES	3.0	0.0	0	0
1	36	4.0	M	NO	0.0	0.0	0	1
2	46	1.0	F	YES	10.0	0.0	0	0
3	50	1.0	M	YES	20.0	0.0	0	1
4	64	1.0	F	YES	30.0	0.0	0	0

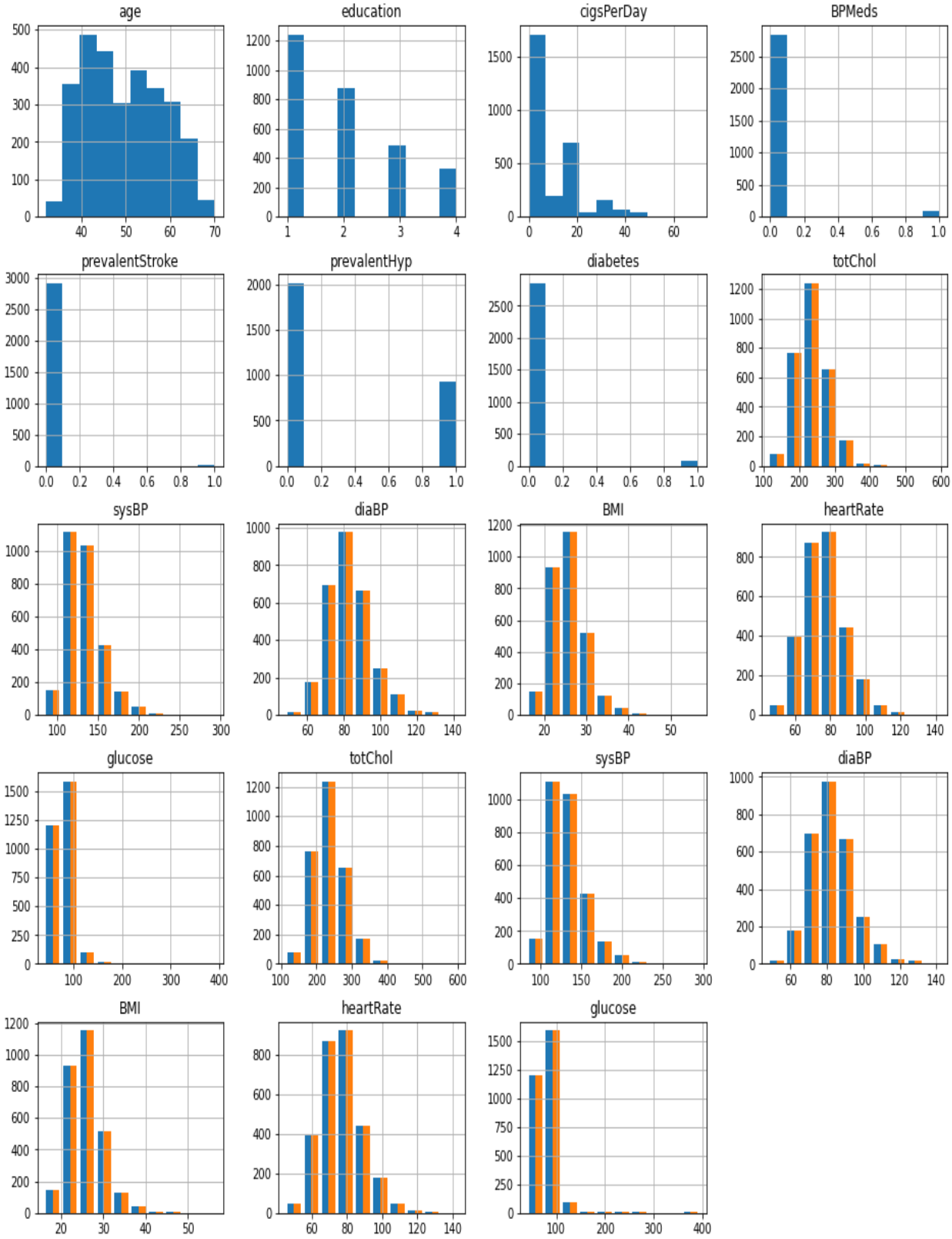
The table shows the dataset in the form of Pandas DataFrame. The dataset has 3391 rows and 17 columns.

Checking for Outliers

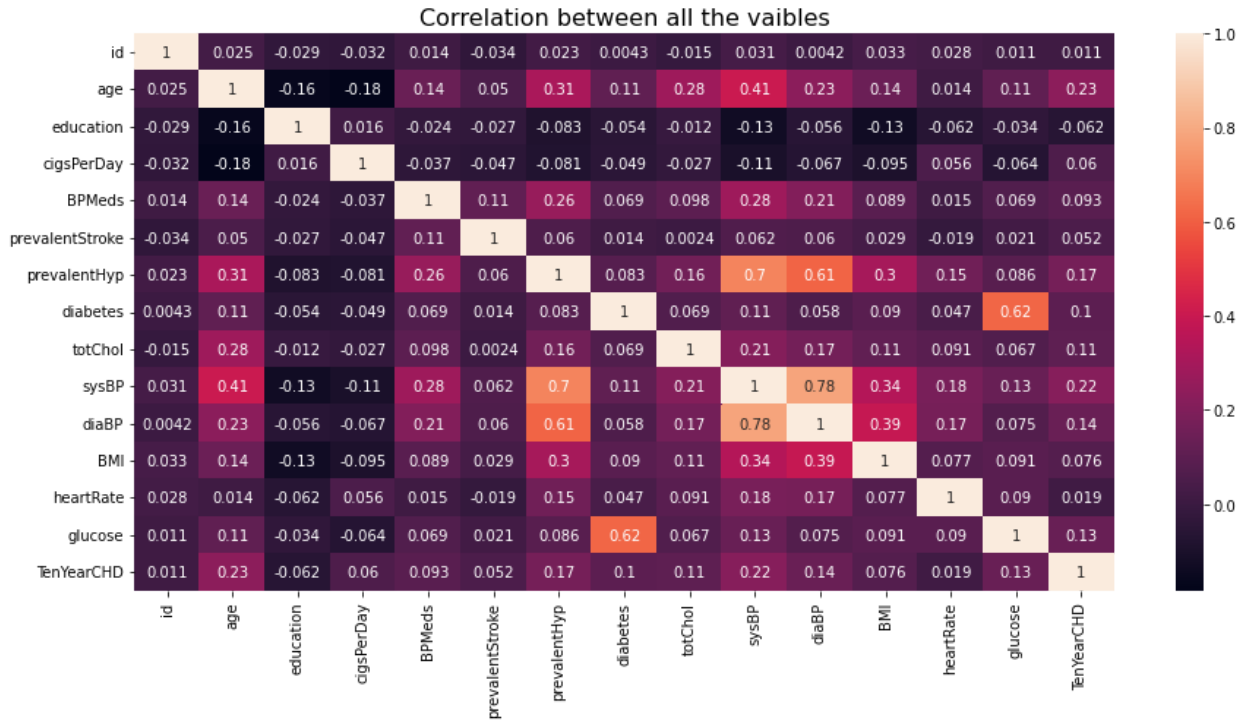


Histogram representation of the data




age
education
cigsPerDay
BPMeds
prevalentStroke
prevalentHyp
diabetes
totChol
sysBP
diaBP
BMI
heartRate
glucose
totChol
sysBP
diaBP
BMI
heartRate
glucose

Correlation Heatmap



Checking for Multi Correlation

Variance Inflation Factor

The Variance Inflation Factor is used when we have the multi-collinearity between the features. The factor helps in reducing the inflation between the features by dropping some of the features which are having a high correlation among them.

In the given dataset the correlation between the features is very high and some of the features are dropped to reduce the correlation among them. Mainly bill amount features.

Data Modeling

After the data preparation is completed it is ready for the purpose of analysis. Only numerical valued features are taken into consideration. The data were combined and labeled as X and y as independent and dependent variables respectively. The open, high and low columns are taken as independent variables (X) and the closing price is taken as dependent variable (y).

Splitting the data

The train_test_split was imported from the sklearn.model_selection. The data is now divided into 80% and 20% as train and test splits respectively. 80% of the data is taken for training the model and 20% is for a test and the random state was taken as 0.

Scaling the data

To normalize the data minmaxscaler was used from sklearn.preprocessing. It scales the data in the form of the standard deviation of the feature multiplied with the difference of maximum and minimum, again it was added to a minimum. At first, the training data was made fit into the scaling function and test data is transformed now. The output we get is X_train, X_test, y_train, y_test.

```
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 Evaluators(Metrics)

Classification Report

A Classification report is used to measure the quality of predictions from a classification algorithm. How many predictions are True and how many are False. More specifically, True Positives, False Positives, True negatives and False Negatives are used to predict the metrics of a classification report.

The report shows the main classification metrics precision, recall and f1-score on a per-class basis. The metrics are calculated by using true and false positives, true and false negatives.

Confusion matrix

In machine learning and statistical classification, a confusion matrix is a table in which predictions are represented in columns and actual status is represented by rows. Sometimes this is reversed, with actual instances in rows and predictions in columns. The table is an extension of the confusion matrix in predictive analytics and makes it easy to see whether mislabeling has occurred and whether the predictions are more or less correct.

A confusion matrix is also known as an error matrix, and it is a type of contingency table.

Model Implementation

1. Decision Tree Classifier

Decision tree build classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.

Train data accuracy

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

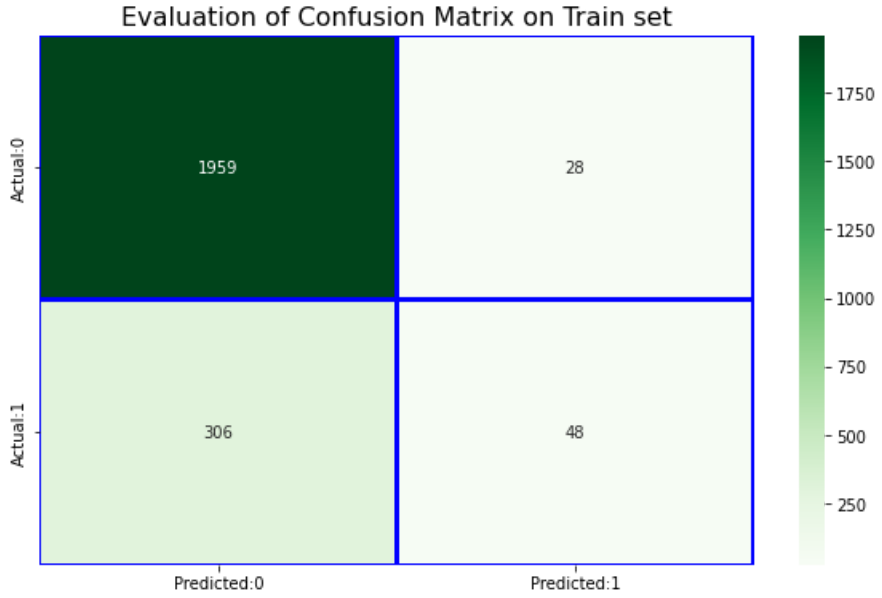
```
0.8573259290901324
```

Test data accuracy

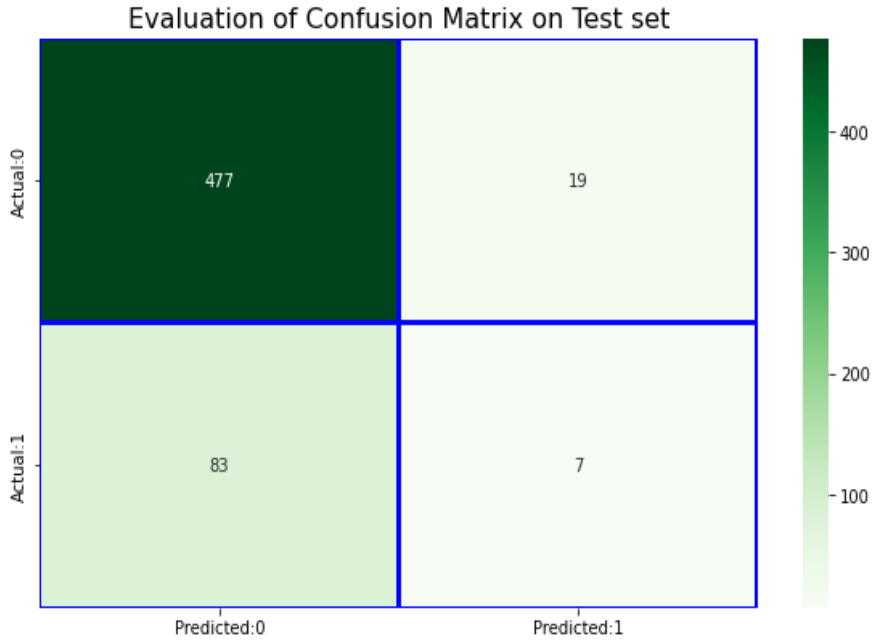
```
accuracy = accuracy_score(y_test,pred_test)
accuracy
```

```
0.8276450511945392
```

Confusion matrix on Train dataset



Confusion matrix on Test dataset



Classification Report on Train dataset

```
Classification Report is:

              precision    recall  f1-score   support

           0       0.86      0.99      0.92      1987
           1       0.63      0.14      0.22       354

    accuracy                           0.86      2341
   macro avg       0.75      0.56      0.57      2341
weighted avg       0.83      0.86      0.82      2341
```

Classification Report on Test Dataset

```
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
```

2. Logistic Regression

A logistic regression is a type of statistical procedure. It is used to refer specifically to the problem in which the dependent variable is binary, that is the number of available categories is two, while the problem with more than two categories is referred to as multi logistic regression.

Train data and Test data accuracy

```
#Accuracy Test
Train_accuracy = accuracy_score(y_train,pred_train)
Train_accuracy
```

```
0.8521999145664246
```

```
#Accuracy Test
Test_accuracy = accuracy_score(y_test,pred_test)
Test_accuracy
```

```
0.8532423208191127
```

Cross validation on Logistic Regression

```
scoring = ['accuracy']
scores = cross_validate(lr_clf, X_train, y_train, scoring=scoring, cv=5,
                        return_train_score = True, return_estimator = True, verbose=10)
```

```
[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 on train data and test data after cross validation

```
#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 ])
```

3. K-Neighbor Classifier

The K Nearest Neighbor algorithm falls under the Supervised Learning category and is used for classification (most commonly) and regression. It is a versatile algorithm also used for imputing missing values and resampling datasets.

Train data accuracy

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

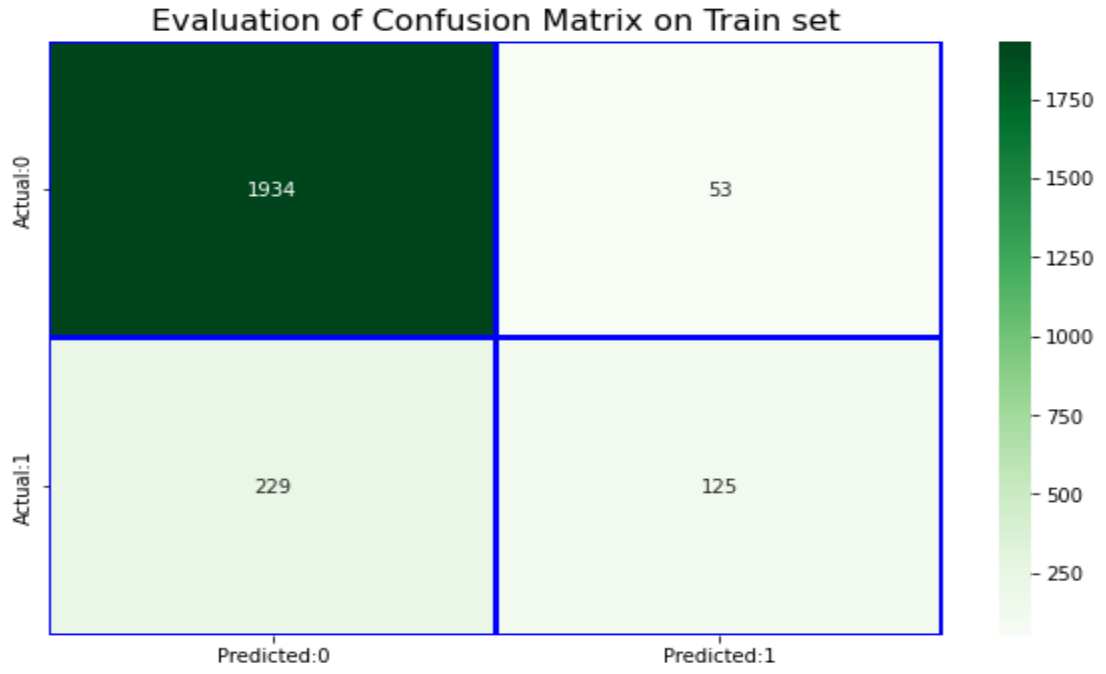
```
0.8795386586928663
```

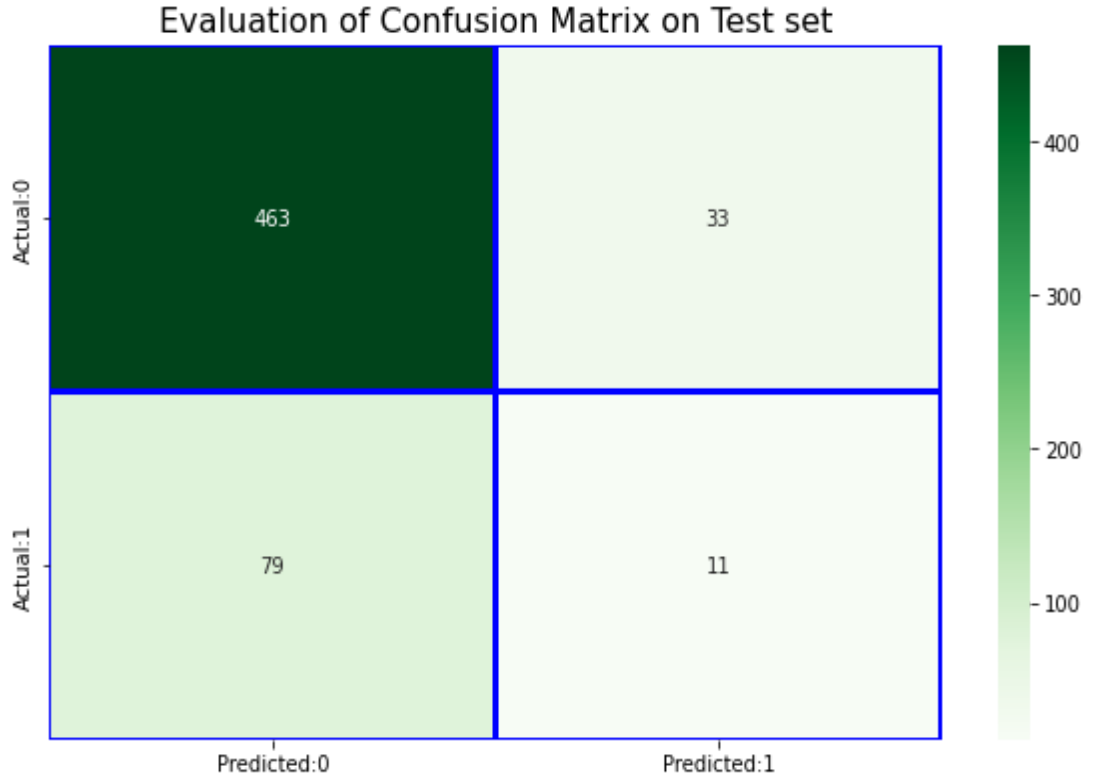
Test data accuracy

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

```
0.8088737201365188
```

Confusion matrix on Train and Test dataset





Classification Report on Train dataset

```
Classification Report is:

              precision    recall  f1-score   support

           0       0.89      0.97      0.93      1987
           1       0.70      0.35      0.47       354

    accuracy                           0.88      2341
   macro avg       0.80      0.66      0.70      2341
weighted avg       0.87      0.88      0.86      2341
```

Classification Report on Test dataset

```
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
```

4. Naive Bayes Classifier

Naive Bayes is a simple and powerful algorithm for predictive modeling. ... Naive Bayes is called naive because it assumes that each input variable is independent. This is a strong assumption and unrealistic for real data; however, the technique is very effective on a large range of complex problems.

Accuracy on Train and Test data set

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

```
0.8278513455788125
```

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

```
0.8344709897610921
```

5. SVM Classifier

Support Vector Machine (SVM) is a classification technique used for the classification of linear as well as non-linear data. SVM is the margin based classifier. It selects the maximum margin. ... This model is further used to perform classification of testing data.

Accuracy on Train and Test data set

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

```
0.8487825715506194
```

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

```
0.8464163822525598
```

6.Gradient Boosting Classifier

Gradient boosting classifiers are a group of machine learning algorithms that combine many weak learning models together to create a strong predictive model. Decision trees are usually used when doing gradient boosting.

Accuracy on Train and Test data set

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

```
0.8735583084152072
```

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

```
0.8395904436860068
```

Gradient Boosting Classifier (TUNING)

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

Accuracy on Train and Test data set (Tuning)

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

```
0.8624519436138403
```

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

```
0.8481228668941979
```

7. Random Forest Classifier

Random Forest is a supervised learning algorithm that uses ensemble learning methods for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes(classification).

Accuracy on Train and Test data set

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

```
1.0
```

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

```
0.8344709897610921
```

Random Forest Classifier (hyperparameter Tuning)

```
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]
```

Cross validation on Random Forest Classifier

```
from sklearn.model_selection import RandomizedSearchCV
rf_grid = RandomizedSearchCV(estimator= rf_model , param_distributions= random_grid , cv = 5
```

```
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]},
```

Accuracy on Train and Test data set (Tuning)

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

Comparing all the models

	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

Conclusion :

- ➢ Defining dependent variables and EDA on Dataset.
- ➢ We have patients from the 32 to 70 age group. Number of patients from the 38 to 46 age group is high with smoking habits.
- ➢ Number of female patients is higher than male patients.
- ➢ There are 1307 male patients in the dataset out of which 809 male patients smoke cigarettes.
- ➢ There are 1620 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.
- ➢ We used the Decision Tree,Logistic Regression,Random Forest Classifier, KNear Neighbor, Naive Bayes,SVM, Gradient Boosting Classifier 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.
- ➢ If we want to choose only the best one model it is better to train the model with Logistic Regression which has 85.32% accuracy on test data.