HEART DISEASE PREDICTION

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Heart disease is the most common type of disease in today's world occurring due to changes in an individual's lifestyle. This includes but not limited to stress, prolonged works, lack of physical activity, smoking, alcohol consumption, and other existing medical history. Early prediction of the symptoms can help a person to avoid this dangerous disease by leading a proper physical and dietary lifestyle.

Here in our analysis, we selected a dataset that contains all the key factors that promotes for heart disease. These factors are not the only ones that contribute but are the major ones. We are going to feed these factors to a predictive model that provides a binary output i.e., 0 (No) or 1 (Yes). We will perform the analysis using various predictive models like Trees and Regression. Finally, will provide a performance comparison among all the models using performance metrics. Also, we will perform dimensionality reduction on the data to obtain the important components from the dataset.

However as this is a preliminary analysis, we allude to the need for further investigation to improve the predictive and explanatory models.

1. INTRODUCTION

In this project, we analyze and predict heart disease based on the demographics, behavioral, previous, and current medical history of a person. The dataset is from kaggle.com, including the key details of gender, age, cholesterol, glucose, blood pressure, and smoking. We took a total of 16 variables including the dependent variable from the heart disease dataset to predict the heart disease.

World Health Organization (WHO) has estimated that about 12 million deaths occur worldwide every year is due to various heart diseases. Half of the deaths in the United States and other developed countries are due to cardiovascular diseases. The early diagnosis of such diseases can help in making decisions on lifestyle changes in high-risk patients and in turn reduce the complications. The idea for choosing this project was to pinpoint the most relevant or risk factors of heart disease as well as predict the overall risk using various predictive models like Logistic Regression, Decision Tree, Random Forest, and Boosted Tree. This project also performs clustering to identify the relation between each cluster and finally performs a principal component analysis to find out the important variables that contributes the model better. Finally, we will suggest a model that best predicts the heart disease based on our study.

The outcome of this project would be a classification model that predicts the 10-year risk of coronary heart disease using the provided predictors. As our predictive models, we have used Logistic Regression, Neural Networks, Decision Tree, Random Forest, and Boosted Tree to analyze the relation between output and input variables. Also, we have performed dimensionality reduction using Principal Component Analysis technique to find the important components of the dataset considered for our use case.

2. DATA DESCRIPTION AND UNDERSTANDING

The dataset is publicly available on the kaggle website, and it is from an ongoing cardiovascular study on residents of the town of Framingham, Massachusetts. The classification goal is to predict whether the patient has 10-year risk of future coronary heart disease (CHD). This dataset provides the patients' information which includes over 4,000 records, 15 independent variables, and 1 dependent variable each of which is a potential risk factor. This data contains demographic, behavioral, and medical risk factors. The following are the variables we have from the selected dataset.

Attribute Name	Type	Description
male	Binary	Male or Female
age	Numerical	Age of the patient
education	Numerical	Education of the patient
currentSmoker	Numerical	Whether or not the patient is a current smoker
cigsPerDay	Numerical	The number of cigarettes that the person smoked on average in a day
BPMeds	Numerical	Whether or not the patient was on blood pressure medication
prevalentStroke	Numerical	Whether or not the patient had previously had a stroke
prevalentHyp	Numerical	Whether or not the patient was hypertensive
diabetes	Numerical	Whether or not the patient had diabetes
totChol	Numerical	Total cholesterol level
sysBP	Numerical	Systolic blood pressure
diaBP	Numerical	Diastolic blood pressure
BMI	Numerical	Body Mass Index
heartRate	Numerical	Heart rate of the patient
glucose	Numerical	Glucose level of the patient
TenYearCHD	Binary	10-year risk of coronary heart disease (Yes/No)

3. EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis refers to the critical process of performing initial investigations on data to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations. It is a good practice to understand the data first and try to gather as many insights as possible from it. EDA is all about making sense of data in hand, before getting them dirty with it.

3.1. Dimensions of the data

Finding out the total number of Rows and Columns in the dataset using 'dim' function.

```
> dim(dataset)
[1] 4238   16
```

Dataset comprises of 4238 observations and 16 attributes. Out of which one is dependent variable and rest 15 are independent variables.

3.2. Output variable distribution

Finding out the dependent variable value distribution in the dataset using 'table' function.

```
> table(dataset$TenYearCHD)

   0    1
3594  644
```

There are 3594 records having 0 as a value and 644 records having 1 as value in the whole dataset. There exists a class imbalance in dependent variable as 0's are more than 1's. It can also be said that the output variable is binary in nature which indicates Yes or No.

3.3. Datatypes of the attributes

Finding out the datatypes of all the attributes in the dataset using 'str' function.

```
> str(dataset)
'data.frame':   4238 obs. of  16 variables:
 $ male           : int  1 0 1 0 0 0 0 0 1 1 ...
 $ age            : int  39 46 48 61 46 43 63 45 52 43 ...
 $ education      : int  4 2 1 3 3 2 1 2 1 1 ...
 $ currentSmoker  : int  0 0 1 1 1 0 0 1 0 1 ...
 $ cigsPerDay     : int  0 0 20 30 23 0 0 20 0 30 ...
 $ BPMeds         : num  0 0 0 0 0 0 0 0 0 0 ...
 $ prevalentStroke: int  0 0 0 0 0 0 0 0 0 0 ...
 $ prevalentHyp   : int  0 0 0 1 0 1 0 0 1 1 ...
 $ diabetes       : int  0 0 0 0 0 0 0 0 0 0 ...
 $ totChol        : num  195 250 245 225 285 228 205 313 260 225 ...
 $ sysBP          : num  106 121 128 150 130 ...
 $ diaBP          : num  70 81 80 95 84 110 71 71 89 107 ...
 $ BMI            : num  27 28.7 25.3 28.6 23.1 ...
 $ heartRate      : int  80 95 75 65 85 77 60 79 76 93 ...
 $ glucose        : num  77 76 70 103 85 99 85 78 79 88 ...
 $ TenYearCHD     : int  0 0 0 1 0 0 1 0 0 0 ...
```

Data has only the numeric datatypes.

3.4. Summary statistics

Finding out the Minimum, Maximum, Mean, Median, 1st and 4th quartiles of the dataset using 'summary' function. This also gives the information about missing values.

```
> summary(dataset)
      male              age          education     currentSmoker      cigsPerDay
 Min.   :0.0000   Min.   :32.00   Min.   :1.000   Min.   :0.0000   Min.   : 0.000
 1st Qu.:0.0000   1st Qu.:42.00   1st Qu.:1.000   1st Qu.:0.0000   1st Qu.: 0.000
 Median :0.0000   Median :49.00   Median :2.000   Median :0.0000   Median : 0.000
 Mean   :0.4292   Mean   :49.58   Mean   :1.979   Mean   :0.4941   Mean   : 9.003
 3rd Qu.:1.0000   3rd Qu.:56.00   3rd Qu.:3.000   3rd Qu.:1.0000   3rd Qu.:20.000
 Max.   :1.0000   Max.   :70.00   Max.   :4.000   Max.   :1.0000   Max.   :70.000
                                  NA's   :105                      NA's   :29
     BPMeds        prevalentStroke     prevalentHyp       diabetes          totChol
 Min.   :0.00000   Min.   :0.000000   Min.   :0.0000   Min.   :0.00000   Min.   :107.0
 1st Qu.:0.00000   1st Qu.:0.000000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:206.0
 Median :0.00000   Median :0.000000   Median :0.0000   Median :0.00000   Median :234.0
 Mean   :0.02963   Mean   :0.005899   Mean   :0.3105   Mean   :0.02572   Mean   :236.7
 3rd Qu.:0.00000   3rd Qu.:0.000000   3rd Qu.:1.0000   3rd Qu.:0.00000   3rd Qu.:263.0
 Max.   :1.00000   Max.   :1.000000   Max.   :1.0000   Max.   :1.00000   Max.   :696.0
 NA's   :53                                                             NA's   :50
     sysBP            diaBP             BMI           heartRate         glucose          TenYearCHD
 Min.   : 83.5   Min.   : 48.00   Min.   :15.54   Min.   : 44.00   Min.   : 40.00   Min.   :0.000
 1st Qu.:117.0   1st Qu.: 75.00   1st Qu.:23.07   1st Qu.: 68.00   1st Qu.: 71.00   1st Qu.:0.000
 Median :128.0   Median : 82.00   Median :25.40   Median : 75.00   Median : 78.00   Median :0.000
 Mean   :132.4   Mean   : 82.89   Mean   :25.80   Mean   : 75.88   Mean   : 81.97   Mean   :0.152
 3rd Qu.:144.0   3rd Qu.: 89.88   3rd Qu.:28.04   3rd Qu.: 83.00   3rd Qu.: 87.00   3rd Qu.:0.000
 Max.   :295.0   Max.   :142.50   Max.   :56.80   Max.   :143.00   Max.   :394.00   Max.   :1.000
                                  NA's   :19      NA's   :1        NA's   :388
```

Here it can be noticed that the mean value is always larger than the median for all the attributes. Also, we can observe the existence of missing values in the dataset that needs to be taken care of before performing modeling.

3.5. Missing values detection

Finding out the columns containing missing values and their sum using 'is.na' and 'colSums' functions.

```
> colSums(is.na(dataset))
          male            age      education   currentSmoker      cigsPerDay         BPMeds
             0              0            105              0             29             53
prevalentStroke   prevalentHyp       diabetes        totChol          sysBP          diaBP
             0              0              0             50              0              0
           BMI      heartRate        glucose     TenYearCHD
            19              1            388              0
> sum(is.na(dataset))
[1] 645
```

The education, cigsPerDay, BPMeds, totChol, BMI, heartrate, and glucose are having null values with their respective count. There are a total of 645 null values in the whole dataset.

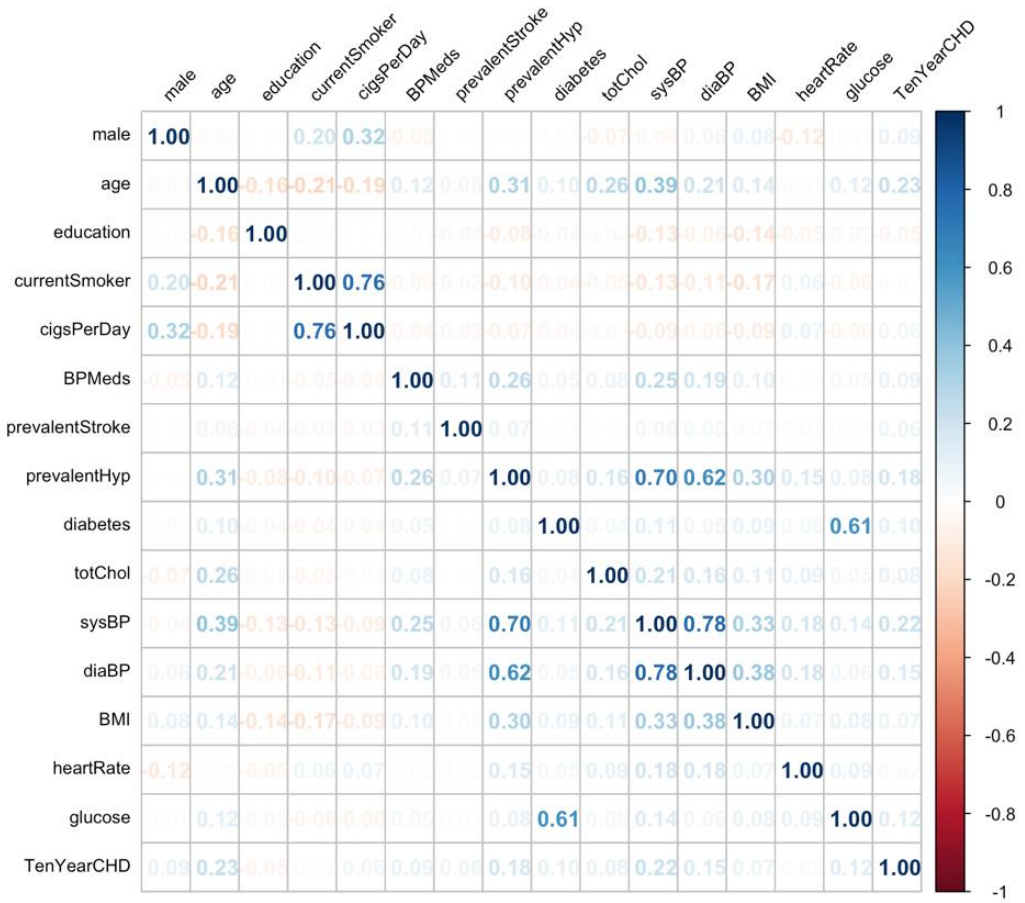
3.6. Missing value imputation

Null values can be imputed using mean or median for numerical variables, and mode can be used for categorical variables. However, one can also impute a desired value in place of missing value. For our use case, we used median imputation to fill the null values

```
> colSums(is.na(dataset))
          male            age      education   currentSmoker      cigsPerDay         BPMeds
             0              0              0              0              0              0
prevalentStroke   prevalentHyp       diabetes        totChol          sysBP          diaBP
             0              0              0              0              0              0
           BMI      heartRate        glucose     TenYearCHD
             0              0              0              0
> sum(is.na(dataset))
[1] 0
```

There are no missing values after the imputation.

3.7. Correlation matrix

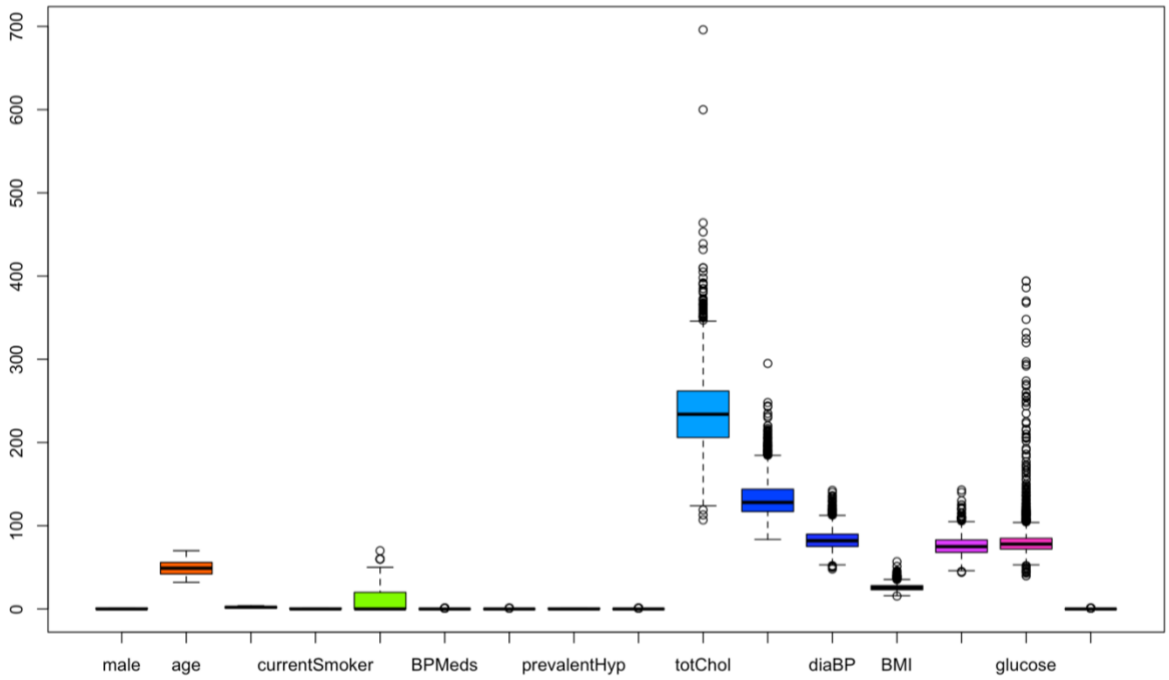


Dark shades represent positive correlation while lighter shades represent negative correlation. It is a good practice to remove correlated variables during feature selection.

Here we can infer that 'cigsPerDay' has strong positive correlation with 'male', while 'heartRate' has strong negative correlation.
'sysBP' has strong positive correlation with 'diaBP' and 'prevalentHyp' has strong positive correlation with 'sysBP'.
'age' is stongly negatively correlated with 'currentSmoker' and 'cigsPerDay'.

3.8. Outlier detection

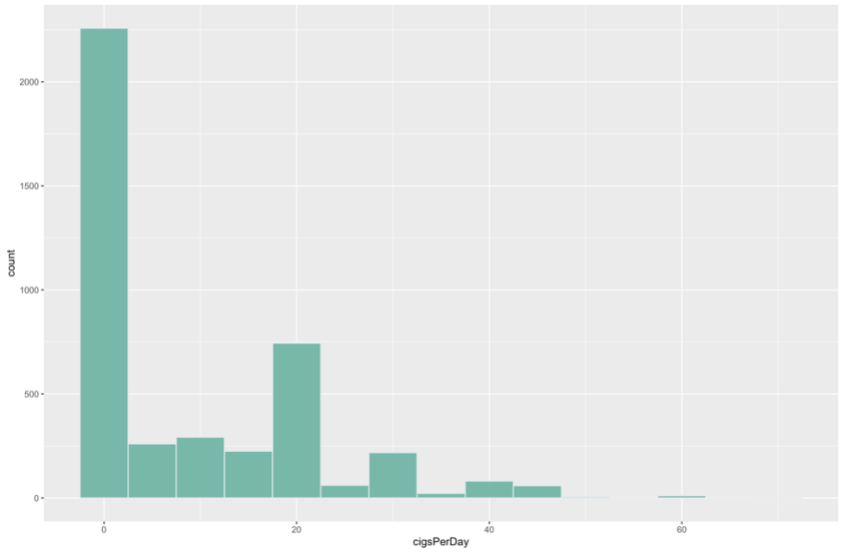
Box plot is used to detect the outliers in the selected dataset.



Here, we can observe that there are more outliers in totChol, sysBP, diaBP, BMI, heartrate, and glucose. But we thought of not to omit these outliers as this play key role in our use case i.e., heart disease prediction. Heart disease can occur only when one or more of these values are at their extreme ranges.

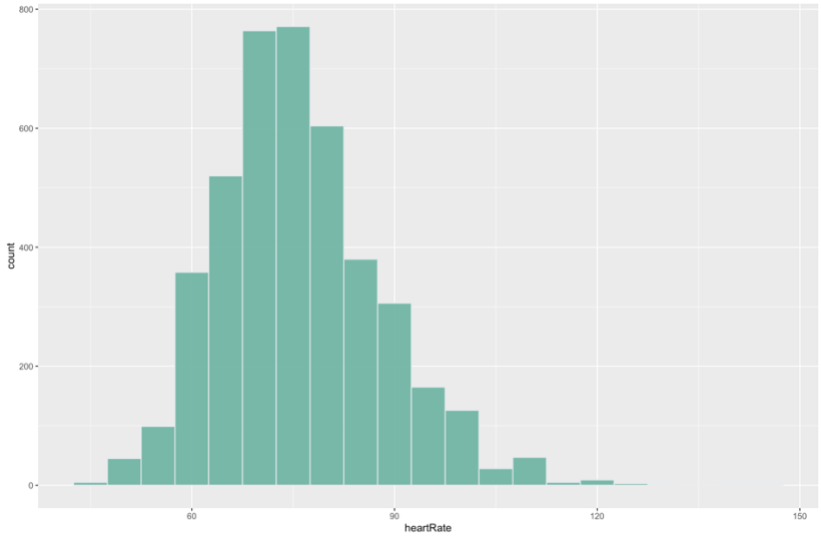
3.9. Data distribution

Cigarettes per day



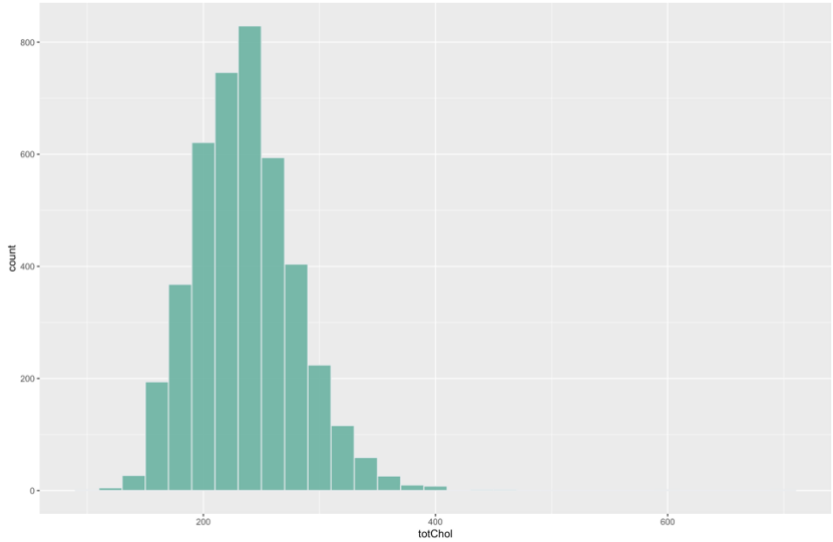
We can observe that the more than 2000 people are smoking 0 to 5 cigarettes per day. As per the data distribution, it is skewed towards left.

Heart Rate



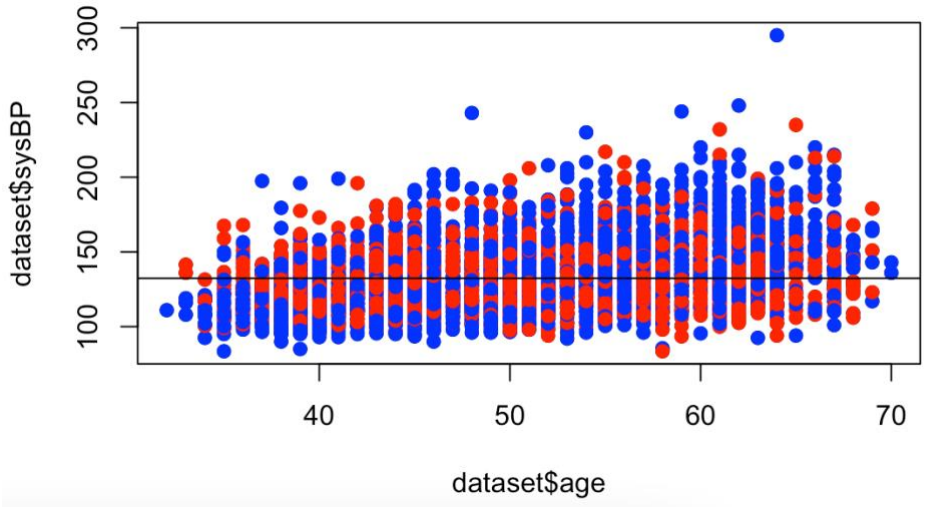
It can be observed that the heart rate is normally distributed with majority ranging between 70 to 80.

Total Cholesterol



The data here is left skewed as there are few outliers in the data that is falling under 700 range. Majority of the data is distributed between 100 to 400.

Plotting Age and Systolic Blood Pressure to understand the relation.



We can see the proportion of people falling below and above the mean of systolic blood pressure.

3.10. Skewness

```
> skewness(dataset)
          male            age      education   currentSmoker      cigsPerDay          BPMeds
    0.28603354     0.22806502     0.69692452     0.02359768     1.25702713     5.58637274
prevalentStroke   prevalentHyp       diabetes        totChol          sysBP          diaBP
    12.90449278     0.81898791     5.99225605     0.87840731     1.14495671     0.71384941
           BMI      heartRate        glucose      TenYearCHD
    0.98509274     0.64438001     6.53916060     1.93905463
```

From the box plot and distribution chart, we can say that there exists a skewness in data. From the above calculation, we can evidently say how much extent the data is skewed.

4. DATA PREPARATION

While performing exploratory data analysis, we took take care of removing the missing value by data imputation, performed the relationship and distribution studies. For modelling, we performed a split on the data in two parts. The first part, used in training our model, will be majority of the dataset. The second part will be used for evaluating the accuracy of trained model. We used 70:30 rule to split the data into train and test (validation) data using 'sample' function to ensure randomness in data split.

5. PREDICTIVE MODELLING

5.1. Logistic Regression

Logistic regression is a useful analysis method for classification problems, is a process of modeling the probability of a discrete outcome given an input variable. The most common logistic regression models a binary outcome, something that can take two values such as true/false, yes/no, and so on. This is exactly the type of logistic model that we used for our use case.

Training the model

```
logit_model <- glm (TenYearCHD ~., train, family = "binomial")
summary(logit_model)
```

Here shown in this code snippet, we used 'glm' function that is used for generalized linear modelling. This function is then passed with all the input variables and output variable to train the model based on the provided dataset. We also specified that this model is a binomial.

Testing the model

```
logit.reg.pred <- predict(logit_model, test, type = "response")
```

After training the model on training data, we used the 'predict' function on test data to predict the output variable. Below are the top 5 actual and predicted values using the logistic model.

	actual	predicted
3	0	0.1493401
4	1	0.3620714
5	0	0.1048231
7	1	0.1745915
8	0	0.0666842

Since the predicted values are probabilistic values, we have defined a rule to convert into value i.e., if the predicted value is more than 0.6, then we are converting it to 1, otherwise 0.

After performing all the steps, finally we have arrived at a confusion matrix and other statistical parameters for the test dataset.

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1073  189
         1    5    4

               Accuracy : 0.8474
                 95% CI : (0.8264, 0.8667)
    No Information Rate : 0.8482
    P-Value [Acc > NIR] : 0.5502

                  Kappa : 0.0264

 Mcnemar's Test P-Value : <0.0000000000000002

            Sensitivity : 0.99536
            Specificity : 0.02073
         Pos Pred Value : 0.85024
         Neg Pred Value : 0.44444
             Prevalence : 0.84815
         Detection Rate : 0.84422
   Detection Prevalence : 0.99292
      Balanced Accuracy : 0.50804

       'Positive' Class : 0
```

We have also performed step wise elimination using forward elimination, backward elimination, and both and obtained below formula from each method.

Backward elimination

```
> formula(backwards)
TenYearCHD ~ male + age + cigsPerDay + prevalentStroke + prevalentHyp +
    sysBP + glucose
```

AIC = 2277.5

Forward elimination

```
> formula(forwards)
TenYearCHD ~ age + sysBP + cigsPerDay + glucose + male + prevalentStroke +
    prevalentHyp
```

AIC = 2277.46

Both direction elimination

```
> formula(stepwise)
TenYearCHD ~ age + sysBP + cigsPerDay + glucose + male + prevalentStroke +
    prevalentHyp
```

AIC = 2277.46

Lower the AIC, better the model performance.

5.2. Neural Networks

The name and structure of neural networks are inspired by the human brain, mimicking the way that biological neurons signal to one another. These are comprised of node layers, containing an input layer, one or more hidden layers, and an output layer. Each node connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.

$\sum w_i x_i$ + bias = $w_1x_1 + w_2x_2 + . + . + w_nx_n$ + bias

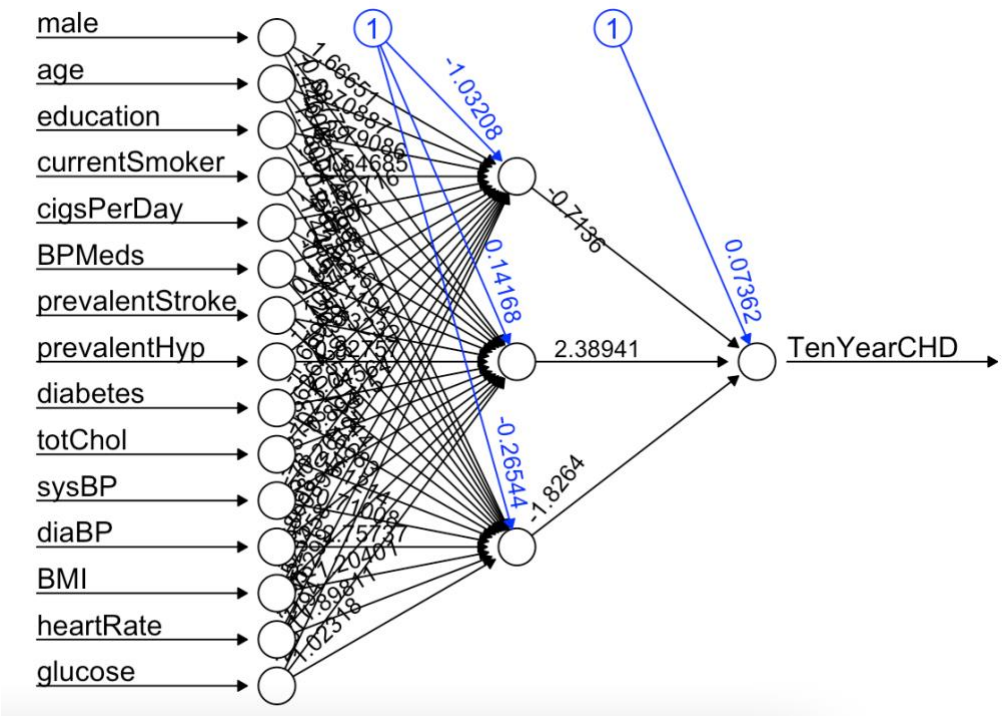
output = f(x) = 1 if $\sum w_1x_1 + w_2x_2 + . + . + w_nx_n$ + bias >= Threshold

0 if $\sum w_1x_1 + w_2x_2 + . + . + w_nx_n$ + bias < Threshold

Training the model

```
nn <- neuralnet (TenYearCHD ~., data = train, linear.output = F,
hidden = 3)
```

Here we used 'neuralnet' function to train the neural network model for which we passed the independent and dependent variables, dataset, and number of hidden layers as input.



Testing the model

```
nn.pred <- predict(nn, test, type = "response")
```

After training the model on training data, we used the 'predict' function on test data to predict the output variable. Converting the predicted value to 1 if predicted value is greater than 0.6, else to 0. We have arrived at a confusion matrix and other statistical parameters for the test dataset.

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1067  185
         1   11    8

               Accuracy : 0.8458
                 95% CI : (0.8247, 0.8652)
    No Information Rate : 0.8482
    P-Value [Acc > NIR] : 0.611

                  Kappa : 0.0496

 Mcnemar's Test P-Value : <0.0000000000000002

            Sensitivity : 0.98980
            Specificity : 0.04145
         Pos Pred Value : 0.85224
         Neg Pred Value : 0.42105
             Prevalence : 0.84815
         Detection Rate : 0.83950
   Detection Prevalence : 0.98505
      Balanced Accuracy : 0.51562

       'Positive' Class : 0
```

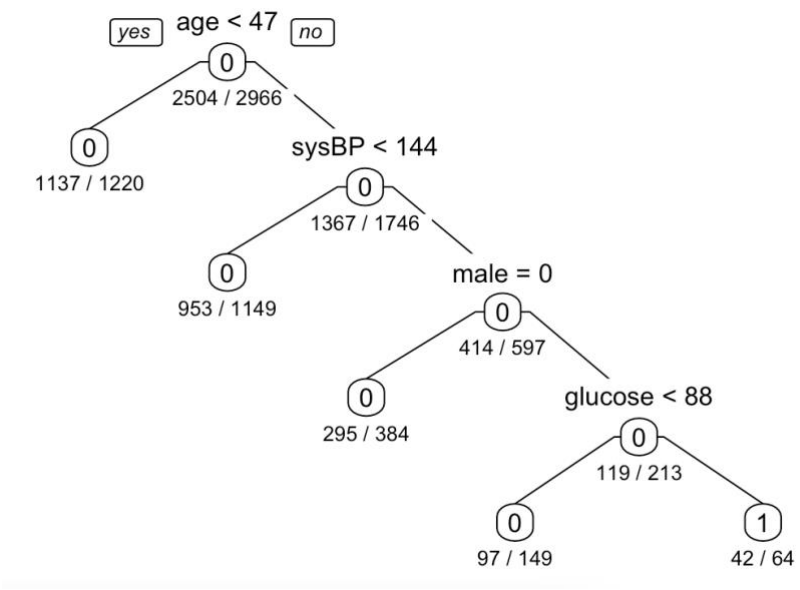
5.3. Decision Tree

Decision Tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

Training the model

```
default.ct <- rpart(TenYearCHD ~ ., data = train.df ,method = "class")
```

Here we used ‘rpart’ function to train the decision tree model for which we passed the independent and dependent variables, dataset, and method as input.



Here we have arrived at 5 leaf nodes started from the root node as ‘age < 47’.

Testing the model

```
default.ct.point.pred.valid <- predict(default.ct, valid.df,
type = "class")
```

After training the model on training data, we used the ‘predict’ function on test data to predict the output variable. We have arrived at a confusion matrix and other statistical parameters for the test dataset.

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1075  177
         1   15    5

               Accuracy : 0.8491
                 95% CI : (0.8282, 0.8683)
    No Information Rate : 0.8569
    P-Value [Acc > NIR] : 0.8006

                  Kappa : 0.0218

 Mcnemar's Test P-Value : <0.0000000000000002

            Sensitivity : 0.98624
            Specificity : 0.02747
         Pos Pred Value : 0.85863
         Neg Pred Value : 0.25000
             Prevalence : 0.85692
         Detection Rate : 0.84513
   Detection Prevalence : 0.98428
      Balanced Accuracy : 0.50686

       'Positive' Class : 0
```

5.4. Decision Tree – Deeper Tree

A deeper version of decision tree that includes more branches and leaves.

Training the model

```
deeper.ct <- rpart(TenYearCHD ~ ., data = train.df, method = "class",
cp = -1, minsplit = 1)
```

Here we used 'rpart' function, which is the same function we used for decision to train the deeper decision tree model by passing additional input parameters like cp, and minsplit.

Testing the model

```
deeper.ct.point.pred.valid <- predict(deeper.ct, valid.df,
type = "class")
```

After training the model on training data, we used the 'predict' function on test data to predict the output variable. We have arrived at a confusion matrix and other statistical parameters for the test dataset.

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 925 137
         1 165  45

               Accuracy : 0.7626
                 95% CI : (0.7382, 0.7857)
    No Information Rate : 0.8569
    P-Value [Acc > NIR] : 1.0000

                  Kappa : 0.0901

 Mcnemar's Test P-Value : 0.1203

            Sensitivity : 0.8486
            Specificity : 0.2473
         Pos Pred Value : 0.8710
         Neg Pred Value : 0.2143
             Prevalence : 0.8569
         Detection Rate : 0.7272
   Detection Prevalence : 0.8349
      Balanced Accuracy : 0.5479

       'Positive' Class : 0
```

5.5. Random Forest

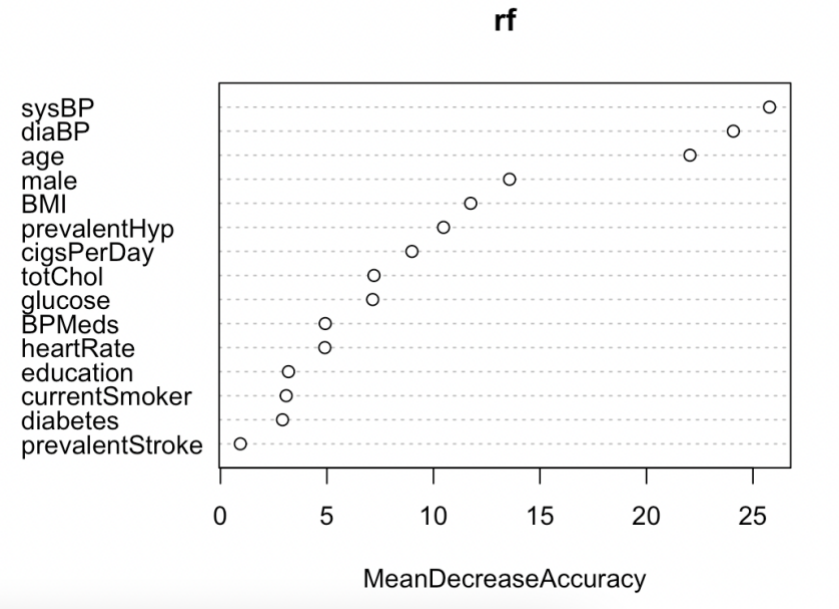
Random Forest is a supervised machine learning algorithm that is used widely in classification and regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.
One of the most important features of the Random Forest algorithm is that it can handle the data set containing continuous variables as in the case of regression and categorical variables as in the case of classification. It performs better results for classification problems.

Training the model

```
rf <- randomForest(as.factor(TenYearCHD) ~ ., data = train.df,
ntree = 500, mtry = 4, nodesize = 5, importance = TRUE,
na.action=na.exclude)
```

Here we used 'randomForest' function to train the Random Forest model for which we passed the key parameters like independent and dependent variables, dataset, number of trees, and node size.

Variable importance plot



Training the model

```
rf.pred <- predict(rf, valid.df)
```

After training the model on training data, we used the 'predict' function on test data to predict the output variable. We have arrived at a confusion matrix and other statistical parameters for the test dataset.

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 920 145
         1  13   6

               Accuracy : 0.8542
                 95% CI : (0.8318, 0.8747)
    No Information Rate : 0.8607
    P-Value [Acc > NIR] : 0.7466

                  Kappa : 0.0407

 Mcnemar's Test P-Value : <0.0000000000000002

            Sensitivity : 0.98607
            Specificity : 0.03974
         Pos Pred Value : 0.86385
         Neg Pred Value : 0.31579
             Prevalence : 0.86070
         Detection Rate : 0.84871
   Detection Prevalence : 0.98247
      Balanced Accuracy : 0.51290

       'Positive' Class : 0
```

5.6. Boosted Tree

Boosting is an ensemble method in which boosted trees are created with a group of decision trees. Boosting transforms weak decision trees (weak learners) into strong learners. Each new tree is built considering the errors of previous trees. Boosting is an iterative process in which each tree is dependent on the previous one. The trees modified from the boosting process are called Boosted Trees.

Training the model

```
boost <- boosting (TenYearCHD ~., data = train.df)
```

Here we used 'boosting' function to train the boosted tree model for which we passed the independent and dependent variables, and train dataset.

Testing the model

```
pred <- predict (boost, valid.df)
```

After training the model on training data, we used the 'predict' function on test data to predict the output variable. We have arrived at a confusion matrix and other statistical parameters for the test dataset.

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1064  162
         1   26   20

               Accuracy : 0.8522
                 95% CI : (0.8315, 0.8713)
    No Information Rate : 0.8569
    P-Value [Acc > NIR] : 0.701

                  Kappa : 0.1249

 Mcnemar's Test P-Value : <0.0000000000000002

            Sensitivity : 0.9761
            Specificity : 0.1099
         Pos Pred Value : 0.8679
         Neg Pred Value : 0.4348
             Prevalence : 0.8569
         Detection Rate : 0.8365
   Detection Prevalence : 0.9638
      Balanced Accuracy : 0.5430

       'Positive' Class : 0
```

5.7. Principal Component Analysis

Principal component analysis (PCA), a technique for reducing the dimensionality of large data sets, works by condensing a large collection of variables into a smaller set that retains most of the information in the larger set.
The quality of a data set inherently suffers when the number of variables is reduced, but the idea in dimensionality reduction is to trade a little accuracy for simplicity. Smaller data sets are easier to examine and display, and since there are fewer unnecessary variables to handle, machine learning algorithms can analyze data much more quickly and easily.

Modeling

```
pcs <- prcomp(dataset)
```

Here we used 'prcomp' function which takes dataset as an input to perform the dimensionality reduction using Principal Component Analysis technique. This helped us to understand the importance of components.

Importance of components

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9
Standard deviation	44.8324	24.7195	21.7805	12.52234	11.16397	7.91781	6.21872	3.71285	0.98134
Proportion of Variance	0.5753	0.1749	0.1358	0.04489	0.03568	0.01795	0.01107	0.00395	0.00028
Cumulative Proportion	0.5753	0.7502	0.8860	0.93092	0.96660	0.98454	0.99561	0.99956	0.99983

	PC10	PC11	PC12	PC13	PC14	PC15	PC16
Standard deviation	0.45608	0.34263	0.32446	0.31870	0.16116	0.1255	0.07568
Proportion of Variance	0.00006	0.00003	0.00003	0.00003	0.00001	0.0000	0.00000
Cumulative Proportion	0.99989	0.99993	0.99996	0.99999	0.99999	1.0000	1.00000

From this output, it can be observed that almost 98% of information in the actual dataset is covered in 6 principal components i.e., PC1 to PC6. By using this limited data dimensions, we can achieve a greater response in machine learning models with a little inaccuracy comparatively.

6. MODEL COMPARISON

Model Name	Accuracy	Error
Random Forest	85.40%	14.60%
Boosted Tree	85.20%	14.80%
Decision Tree	84.90%	15.10%
Logistic Regression	84.70%	15.30%
Neural Networks	84.50%	15.50%
Decision Tree (Deeper)	76.20%	23.80%

Among the models that we have performed, all models performed equally better except the deeper decision tree based on the model accuracy. Random Forest outperformed all the models with 85.4%, while the Boosted Tree and Decision Tree stood at second and third positions respectively. From this it can be observed that a performance metric alone cannot be a decision factor while considering the best performing model. We should consider the model that best suits our data and business use case along with the other performance metrics like Confusion Matrix, Accuracy, Precision, AUC/ROC, Recall, and F1 Score.

7. CONCLUSION

We have performed predictive analysis using Decision Tree, Random Forest, Boosted Tree, Logistic Regression, and Neural Networks. We extracted the accuracy of all the models and Random Forest has highest accuracy, while the Deeper Decision Tree has the lowest among all the models that we have used. We can evidently say that the heart disease prediction works well with Random Forest followed by Boosted Tree and Decision Tree. Although we are concluding that the Random Forest is best for this use case, there might be chances of deviations in the performance due to several factors like poor outlier handling, class imbalance problem, incorrect performance metric, lack of monitoring, bias variance trade-off, etc. This recommended model is only for our use case and does not applicable to all the use cases and datasets. Finally, it is not completely possible to define a model that works better for a use case. Even after considering various performance metrics to rule out the models, one should carefully consider the business requirement and purpose of the modeling to pick the best performing among the available options.

8. REFERENCES

https://www.kaggle.com/datasets/dileep070/heart-disease-prediction-using-logistic-regression

https://www.sciencedirect.com/topics/computer-science/logistic-regression

https://www.ibm.com/cloud/learn/neural-networks

https://www.geeksforgeeks.org/decision-tree/

https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/

https://towardsdatascience.com/introduction-to-boosted-trees-2692b6653b53