HEART DISEASE ANALYSIS AND PREDICTION

By

K. LOKESH BABU

Table of Contents

ABSTRACT

In this project, we will be dealing in the field of analysis and prediction of diseases. The branch of diseases dealt with in our project are the heart diseases in which the diseases related to the heart and other related diseases will be analyzed and predicted so as to prevent further developments.

Primarily, we have planned to use various data analysis techniques like Exploratory Data Analysis, and appropriate Machine Learning Algorithms in this project, but the implementation can be modified and extended further as per development in the project.

Currently we are planning to move ahead with the **"Heart Disease Prediction UCI"** dataset available on Kaggle, and accessible through the following link:
https://www.kaggle.com/priyanka841/heart-disease-prediction-uci

Although, we tend to stick to this dataset, it is subjected to change depending on further developments in the project.

INTRODUCTION

Nowadays, Cardiovascular diseases are one of the causes for reduction the lifespan of human beings. Each year millions of people are dying due to cardiovascular diseases. Thus, to reduce these diseases, an estimate of a person's risk for coronary heart disease is important for many aspects of health promotion and clinical medicine. So that researches on the cure/prevention of cardiovascular diseases could be done in much more efficient manners.

Since there are huge amount of data on the disease are present on online databases of healthcare systems and hospitals, the analysis becomes more complex and lengthier. This is where the exploratory data analysis comes in to provide an estimate of the overall diseases in the form of a moderately small data file which can be analyzed in a much better manner.

The data includes only the necessary attributes such as Age, Sex, Fasting Blood Pressure, Chest Pain type, Resting ECG(test that measures the electrical activity of the heart), Number of major vessels colored by fluoroscopy, Rest Blood Pressure (high blood pressure), Serum Cholesterol (determine the risk for developing heart disease), Thalach (maximum heart rate achieved), ST depression (finding on an electrocardiogram, Trace in the ST segment is abnormally low below the baseline), Painloc (chest pain location (substernal=1, otherwise=0)), Fasting blood sugar, Exang (exercise included angina).

LITERATURE SURVEY

Numerous works have been done related to disease prediction systems using different data mining techniques and machine learning algorithms in medical centers.

K. Polaraju et al[1], proposed Prediction of Heart Disease using Multiple Regression Model and it proves that Multiple Linear Regression is appropriate for predicting heart disease chance. The work is performed using training data set, which consists of 3000 instances with 13 different attributes which have been mentioned earlier. The data set is divided into two parts that is 70% of the data are used for training and 30% used for testing. Based on the results, it is clear that the classification accuracy of Regression algorithm is better compared to other algorithms.

Akhand Pratap Singh, Dr. Bhupal Singh et al[2], developed heart disease prediction using KStar, j48, SMO, and Bayes Net and Multilayer perceptron using WEKA software. Based on performance from different factor SMO and Bayes Net achieve optimum performance than KStar, Multilayer perceptron and J48 techniques using k-fold cross validation. The accuracy performances achieved by those algorithms are still not satisfactory. Therefore, the accuracy's performance is improved more to give better decision to diagnose diseases.

Siddharth Mundra, Kiran Manjrekar et al[3], focuses on techniques that can predict chronic disease by mining the data containing in historical health records using Naïve Bayes, Decision tree, Support Vector Machine(SVM) and Artificial Neural Network(ANN). A comparative study is performed on classifiers to measure the better performance on anaccurate rate. From this experiment, SVM gives highest accuracy rate, whereas for diabetes Naïve Bayes gives the highest accuracy.

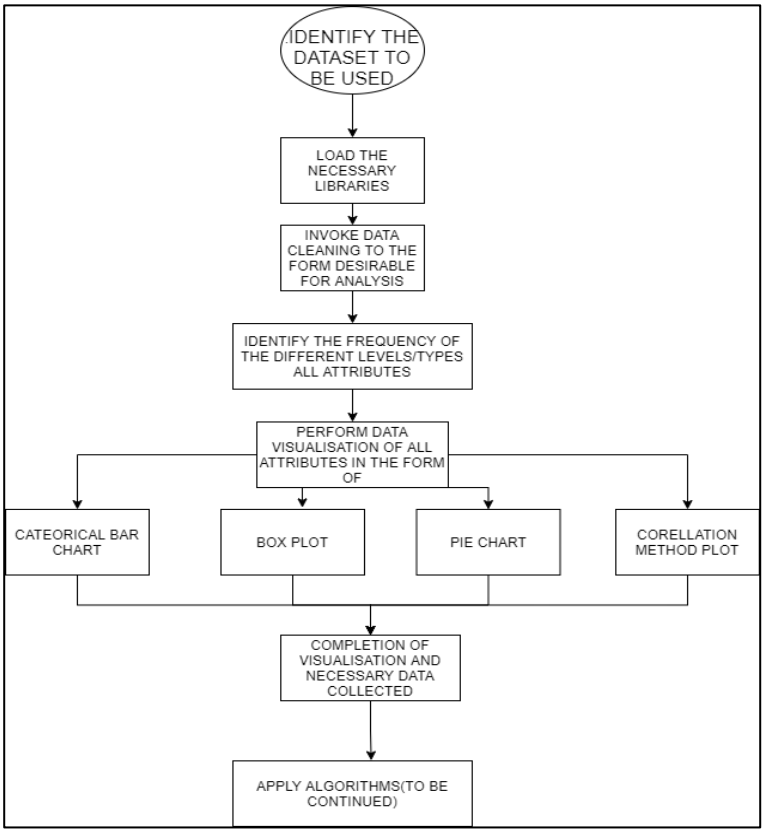
M.Marimuthu, M.Abinaya, K.S.Hariesh, K.Madhankumar, V.Pavithra etal[4], recommended different algorithms like Naive Bayes, Classification Tree, KNN,Logistic Regression, SVM and ANN. The Logistic Regression gives better accuracy compared to other algorithms.

Chala Beyene et al[5], recommended Prediction and Analysis of the occurrence of Heart Disease Using Data Mining Techniques. The main objective is to predict the occurrence of heart disease for early automatic diagnosis of the disease with result in short time. The proposed methodology is also critical in healthcare organisation with experts that have no more knowledge and skill. It uses different medical attributes such as blood sugar and heart rate, age, sex is some of the attributes are included to identify if the person has heart disease or not. Analyses of dataset are computed using WEKA software.

METHODOLOGY

- Identifying and loading dataset
- Including necessary libraries
- Data Cleaning, formatting and categorising
- Identifying frequencies of different levels/types of different attributes (Exploring Data Distribution)
- Data Visualization and Exploratory Data Analysis, using categorical bar graphs, piecharts, box plots, correlation plots.

- Application of algorithms for prediction of heart disease.



EXPERIMENTS AND RESULTS

Experimental Setup and Hyperparameters:

- The Heart-Disease-UCI dataset named “heart.csv” from Kaggle, is used for the project.
- There are 13 independent variables, and one target/dependent variable, on which extensive Exploratory Data Analytics is performed before proceeding with any of the prediction algorithms.
- Some research is done on the variable names in the dataset, which are later renamed to define the feature they are representing, using appropriate words.

Results and discussions:

- The necessary libraries are loaded at the beginning of the project.
- The “heart.csv” dataset is read

age <int>	sex <int>	cp <int>	trestbps <int>	chol <int>	fbs <int>	restecg <int>	thalach <int>	exang <int>	oldpeak <dbl>
63	1	3	145	233	1	0	150	0	2.3
37	1	2	130	250	0	1	187	0	3.5
41	0	1	130	204	0	0	172	0	1.4
56	1	1	120	236	0	1	178	0	0.8
57	0	0	120	354	0	1	163	1	0.6
57	1	0	140	192	0	1	148	0	0.4
56	0	1	140	294	0	0	153	0	1.3
44	1	1	120	263	0	1	173	0	0.0
52	1	2	172	199	1	1	162	0	0.5
57	1	2	150	168	0	1	174	0	1.6

- Number of rows (303) and columns (14 are found) are found.
- The initial column names are viewed.

```
 [1] "age"      "sex"      "cp"       "trestbps" "chol"     "fbs"      "restecg"  "thalach"  "exang"    "oldpeak"  "slope"    "ca"
[13] "thal"     "target"
```

- Glimpse of the dataset

```
Rows: 303
Columns: 14
$ age      <int> 63, 37, 41, 56, 57, 57, 56, 44, 52, 57, 54, 48, 49, 64, 58, 50, 58, 66, 43, 69, 59, 44, 42, 61, 40, 71, 59, 51, 65, 53, 41, 65…
$ sex      <int> 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, …
$ cp       <int> 3, 2, 1, 1, 0, 0, 1, 1, 2, 2, 0, 2, 1, 3, 3, 2, 2, 3, 0, 3, 0, 2, 0, 2, 3, 1, 2, 2, 2, 2, 1, 0, 1, 2, 3, 2, 2, 2, 2, 2, 2, 1, …
$ trestbps <int> 145, 130, 130, 120, 120, 140, 140, 120, 172, 150, 140, 130, 130, 110, 150, 120, 120, 150, 150, 140, 135, 130, 140, 150, 140, 1…
$ chol     <int> 233, 250, 204, 236, 354, 192, 294, 263, 199, 168, 239, 275, 266, 211, 283, 219, 340, 226, 247, 239, 234, 233, 226, 243, 199, 3…
$ fbs      <int> 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, …
$ restecg  <int> 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, …
$ thalach  <int> 150, 187, 172, 178, 163, 148, 153, 173, 162, 174, 160, 139, 171, 144, 162, 158, 172, 114, 171, 151, 161, 179, 178, 137, 178, 1…
$ exang    <int> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, …
$ oldpeak  <dbl> 2.3, 3.5, 1.4, 0.8, 0.6, 0.4, 1.3, 0.0, 0.5, 1.6, 1.2, 0.2, 0.6, 1.8, 1.0, 1.6, 0.0, 2.6, 1.5, 1.8, 0.5, 0.4, 0.0, 1.0, 1.4, 0…
$ slope    <int> 0, 0, 2, 2, 2, 1, 1, 2, 2, 2, 2, 2, 1, 2, 1, 2, 0, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 0, 2, 2, 2, 0, 2, 0, 2, 2, 2, 2, 2, 1, …
$ ca       <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 2, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, …
$ thal     <int> 1, 2, 2, 2, 2, 1, 2, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2, 3, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2, 2, 2, 3, 2, 2, 2, 2, …
$ target   <int> 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, …
```

- Summary of the dataset

```
      age             sex               cp            trestbps         chol            fbs             restecg          thalach     
 Min.   :29.00   Min.   :0.0000   Min.   :0.000   Min.   : 94.0   Min.   :126.0   Min.   :0.0000   Min.   :0.0000   Min.   : 71.0  
 1st Qu.:47.50   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:120.0   1st Qu.:211.0   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:133.5  
 Median :55.00   Median :1.0000   Median :1.000   Median :130.0   Median :240.0   Median :0.0000   Median :1.0000   Median :153.0  
 Mean   :54.37   Mean   :0.6832   Mean   :0.967   Mean   :131.6   Mean   :246.3   Mean   :0.1485   Mean   :0.5281   Mean   :149.6  
 3rd Qu.:61.00   3rd Qu.:1.0000   3rd Qu.:2.000   3rd Qu.:140.0   3rd Qu.:274.5   3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:166.0  
 Max.   :77.00   Max.   :1.0000   Max.   :3.000   Max.   :200.0   Max.   :564.0   Max.   :1.0000   Max.   :2.0000   Max.   :202.0  
     exang           oldpeak         slope             ca              thal            target      
 Min.   :0.0000   Min.   :0.00   Min.   :0.000   Min.   :0.0000   Min.   :0.000   Min.   :0.0000  
 1st Qu.:0.0000   1st Qu.:0.00   1st Qu.:1.000   1st Qu.:0.0000   1st Qu.:2.000   1st Qu.:0.0000  
 Median :0.0000   Median :0.80   Median :1.000   Median :0.0000   Median :2.000   Median :1.0000  
 Mean   :0.3267   Mean   :1.04   Mean   :1.399   Mean   :0.7294   Mean   :2.314   Mean   :0.6898  
 3rd Qu.:1.0000   3rd Qu.:1.60   3rd Qu.:2.000   3rd Qu.:1.0000   3rd Qu.:3.000   3rd Qu.:1.0000  
 Max.   :1.0000   Max.   :6.20   Max.   :2.000   Max.   :4.0000   Max.   :3.000   Max.   :1.0000  
```

EXPLORING DATA DISTRIBUTION:

- Identifying the number of values in each level of dependent variable - "target"

target	n
0	94
1	209

- Identifying the different Sex and their frequency

sex	n
0	96
1	207

- Identifying the different levels of Fasting_Blood_Sugar and their frequency

fbs	n
0	258
1	45

- Identifying the different levels of Exercise_Induced_Angina and their frequency

exang	n
0	204
1	99

- Identifying the different Chest_Pain_Type and their frequency

cp	n
0	143
1	50
2	87
3	23

- Identifying the different levels of Resting_ECG and their frequency

restecg	n
0	147
1	152
2	4

- Identifying the different levels of Thalassemia and their frequency

thal	n
0	2
1	18
2	166
3	117

- From the above, we get a notion about the different levels of each attribute and their corresponding frequencies.
- It's difficult to visualise the data (which is being done later) with numerical values. Thus, the different levels of the attributes are labelled with suitable words.

DATA CLEANING:

- Checking if "NA" value is present in the dataset.

```
  {r}
sum(is.na(data))
```

```
[1] 0
```

- We don't have any "NA" values in our dataset.

DATA TRANSFORMATION:

The data is categorised for all the different levels/frequencies of the different parameters as displayed previously, and are converted to factors. A glimpse of the data after performing this task is as follows:

```
Rows: 303
Columns: 14
$ target   <fct> YES, YES, NO, YES, YES, YES, NO, YES, YES, YES, YES, YES, YES, YES, YES, NO, YES, NO, YES, YES, YES, YES, YES, YES, YES, YES, …
$ sex      <fct> MALE, MALE, FEMALE, MALE, FEMALE, MALE, FEMALE, MALE, MALE, MALE, MALE, FEMALE, MALE, MALE, FEMALE, FEMALE, FEMALE, FEMALE, MA…
$ fbs      <fct> >120, <=120, <=120, <=120, <=120, <=120, <=120, <=120, >120, <=120, <=120, <=120, <=120, <=120, >120, <=120, <=120, <=120, <=1…
$ exang    <fct> NO, NO, NO, NO, YES, NO, NO, NO, NO, NO, NO, NO, NO, YES, NO, NO, NO, NO, NO, NO, NO, YES, NO, YES, YES, NO, NO, NO, NO, NO, N…
$ cp       <fct> ASYMPTOMATIC, NON-ANGINAL PAIN, ATYPICAL ANGINA, ATYPICAL ANGINA, ASYMPTOMATIC, ASYMPTOMATIC, ATYPICAL ANGINA, ATYPICAL ANGINA…
$ restecg  <fct> NORMAL, ABNORMALITY, NORMAL, ABNORMALITY, ABNORMALITY, ABNORMALITY, NORMAL, ABNORMALITY, ABNORMALITY, ABNORMALITY, ABNORMALITY…
$ slope    <fct> 0, 0, 2, 2, 2, 1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 1, 2, 0, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 0, 2, 2, 2, 0, 2, 0, 2, 2, 2, 2, 2, 1, …
$ ca       <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 2, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, …
$ thal     <fct> 1, 2, 2, 2, 2, 1, 2, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2, 3, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2, 2, 2, …
$ age      <int> 63, 37, 41, 56, 57, 57, 56, 44, 52, 57, 54, 48, 49, 64, 58, 50, 58, 66, 43, 69, 59, 44, 42, 61, 40, 71, 59, 51, 65, 53, 41, 65…
$ trestbps <int> 145, 130, 130, 120, 120, 140, 140, 120, 172, 150, 140, 130, 130, 110, 150, 120, 120, 150, 150, 140, 135, 130, 140, 150, 140, 1…
$ chol     <int> 233, 250, 204, 236, 354, 192, 294, 263, 199, 168, 239, 275, 266, 211, 283, 219, 340, 226, 247, 239, 234, 233, 226, 243, 199, 3…
$ thalach  <int> 150, 187, 172, 178, 163, 148, 153, 173, 162, 174, 160, 139, 171, 144, 162, 158, 172, 114, 171, 151, 161, 179, 178, 137, 178, 1…
$ oldpeak  <dbl> 2.3, 3.5, 1.4, 0.8, 0.6, 0.4, 1.3, 0.0, 0.5, 1.6, 1.2, 0.2, 0.6, 1.8, 1.0, 1.6, 0.0, 2.6, 1.5, 1.8, 0.5, 0.4, 0.0, 1.0, 1.4, 0…
```

DATA VISUALISATION AND EXPLORATORY DATA ANALYTICS:

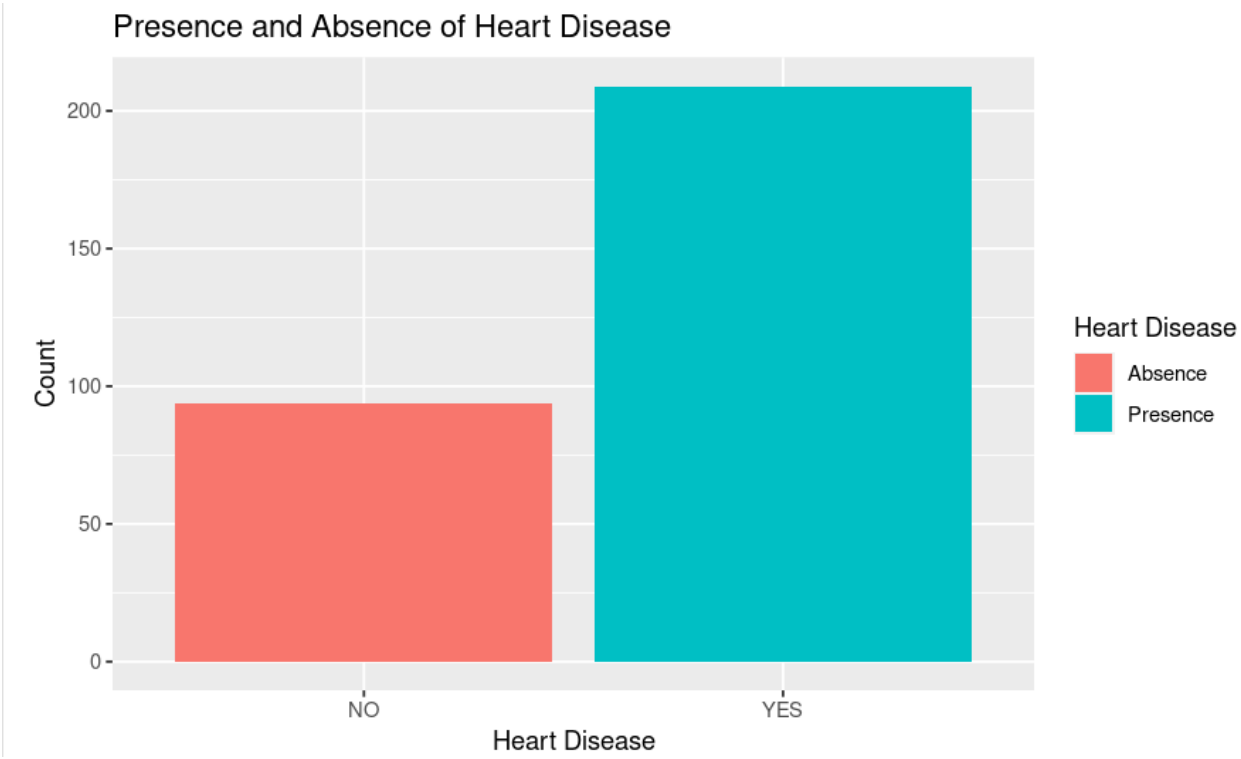
- Distribution of whether a person have heart disease present(YES) or heart disease not present(NO)

```
       NO       YES
0.310231 0.689769
```

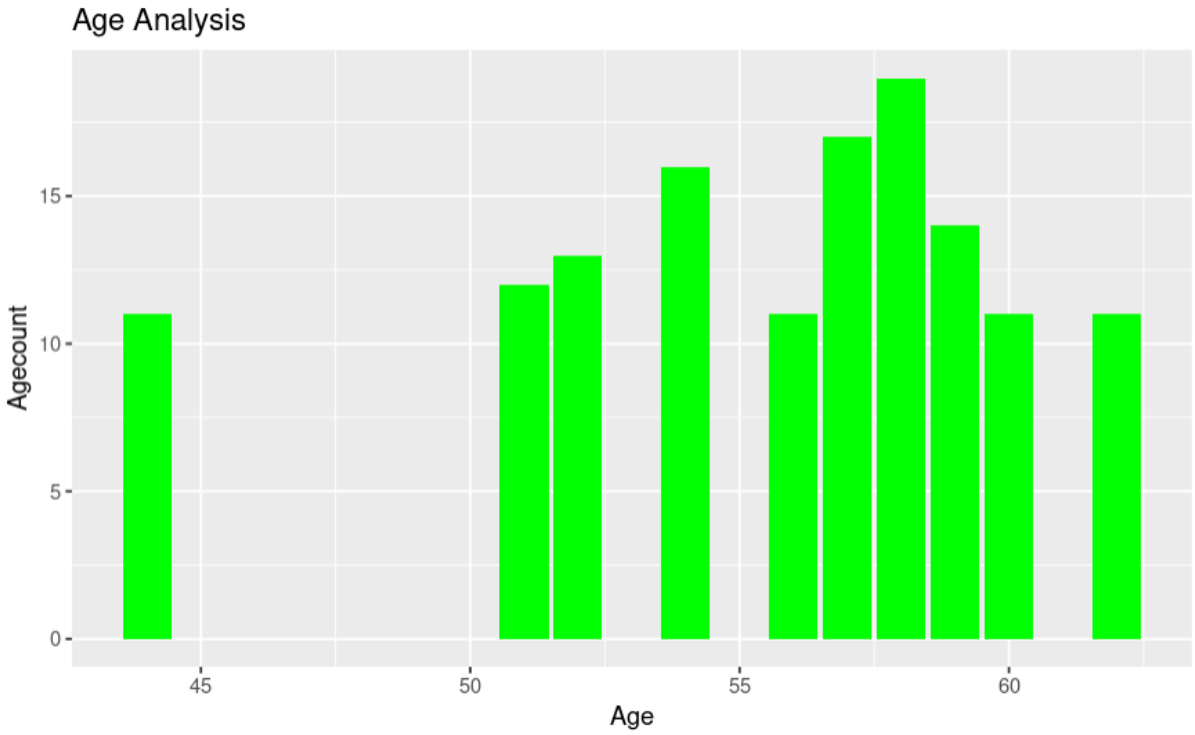
- Count of whether person is healthy (heart disease not present) or unhealthy (heart disease present)

```
NO YES
94 209
```

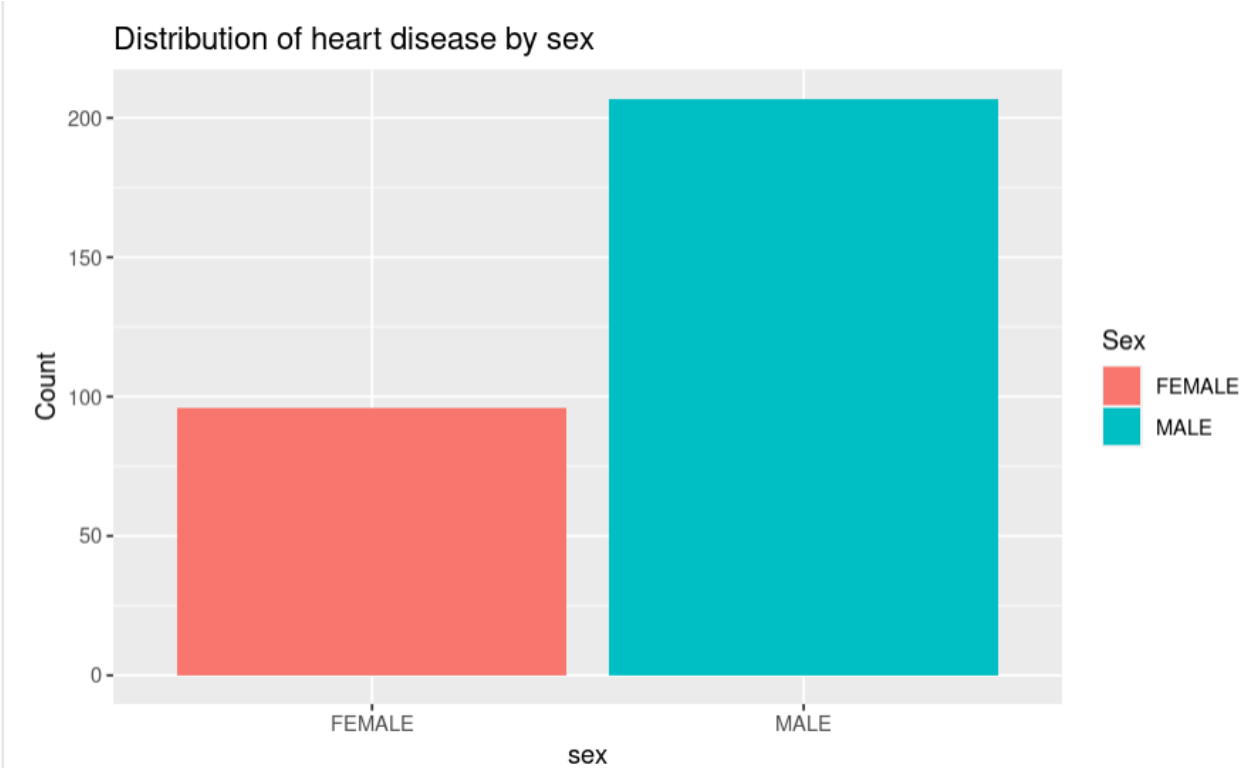
- Bar Plot for "**target**" variable



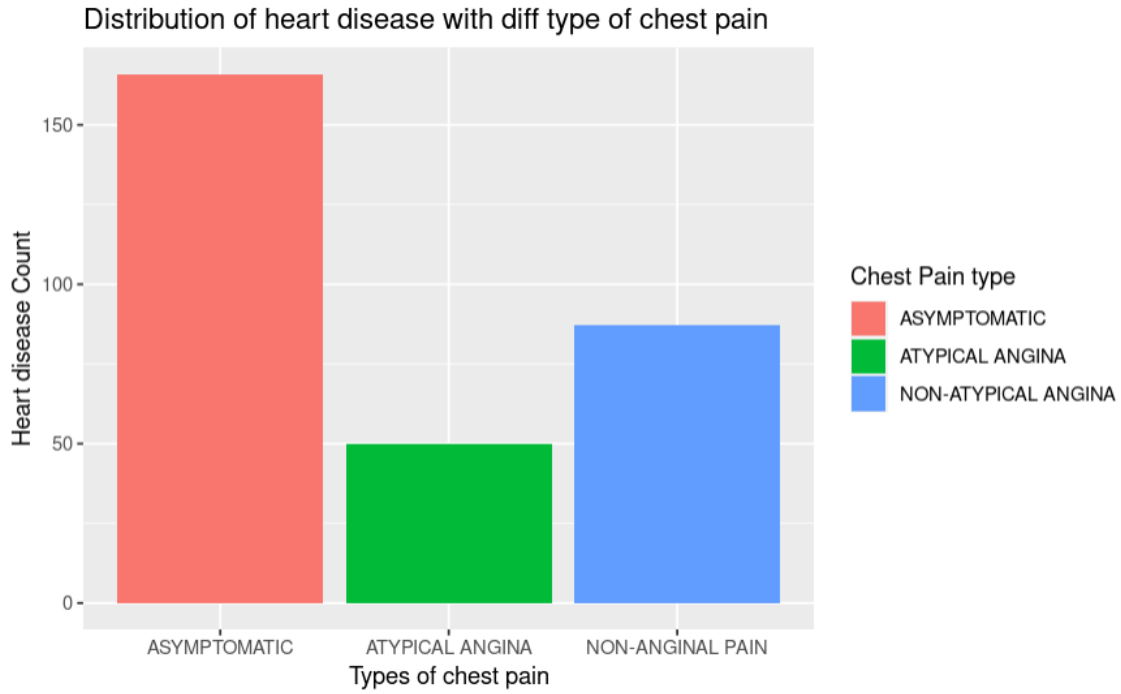
- Frequency of presence of heart disease by age



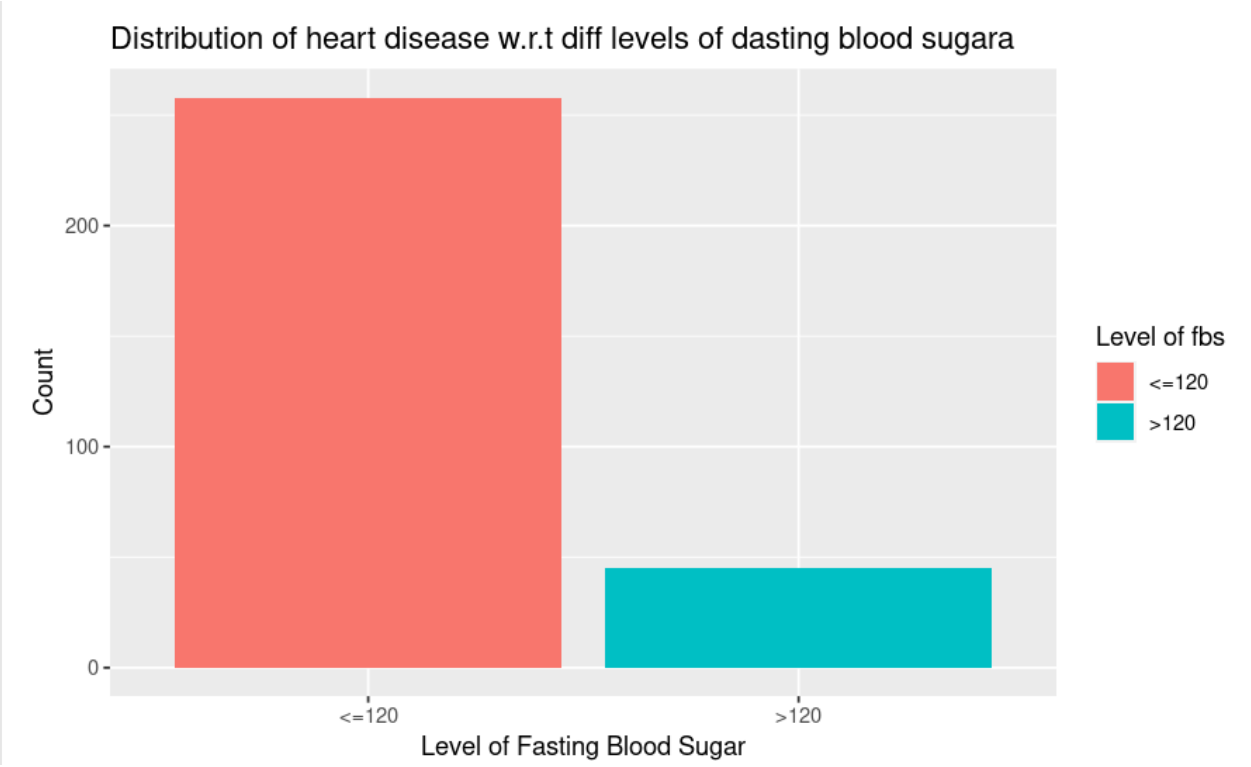
- Frequency of presence of heart disease by Sex



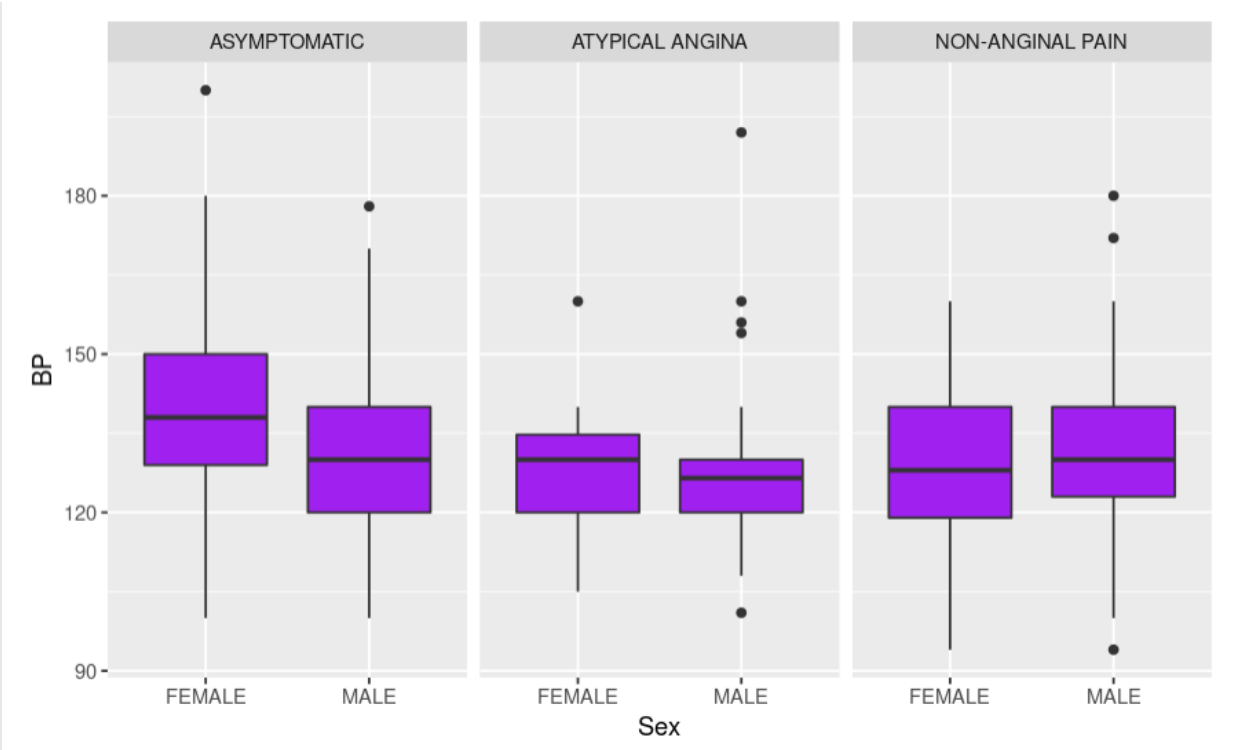
- Frequency of presence of heart disease w.r.t. different types of chest pains



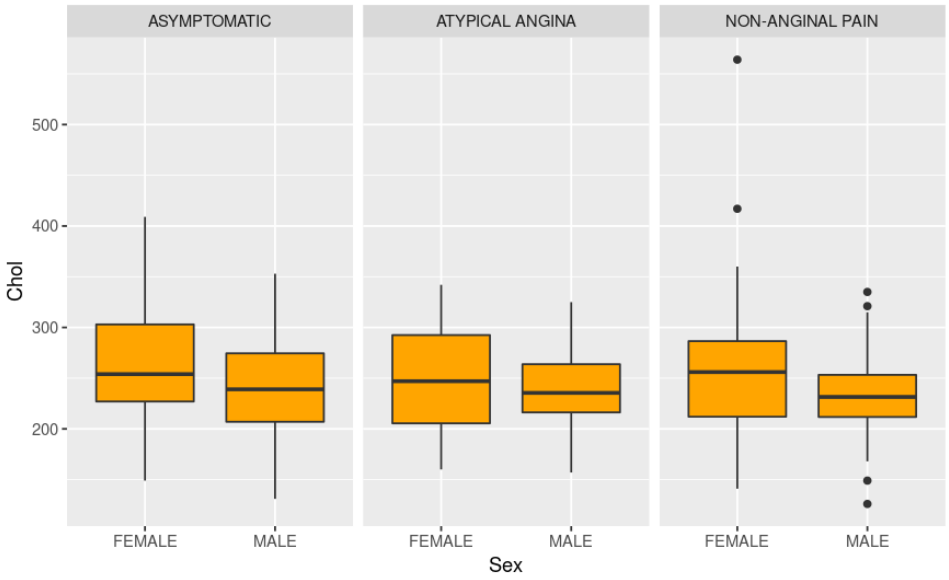
- Frequency of presence of heart disease w.r.t. different levels of fasting blood sugar



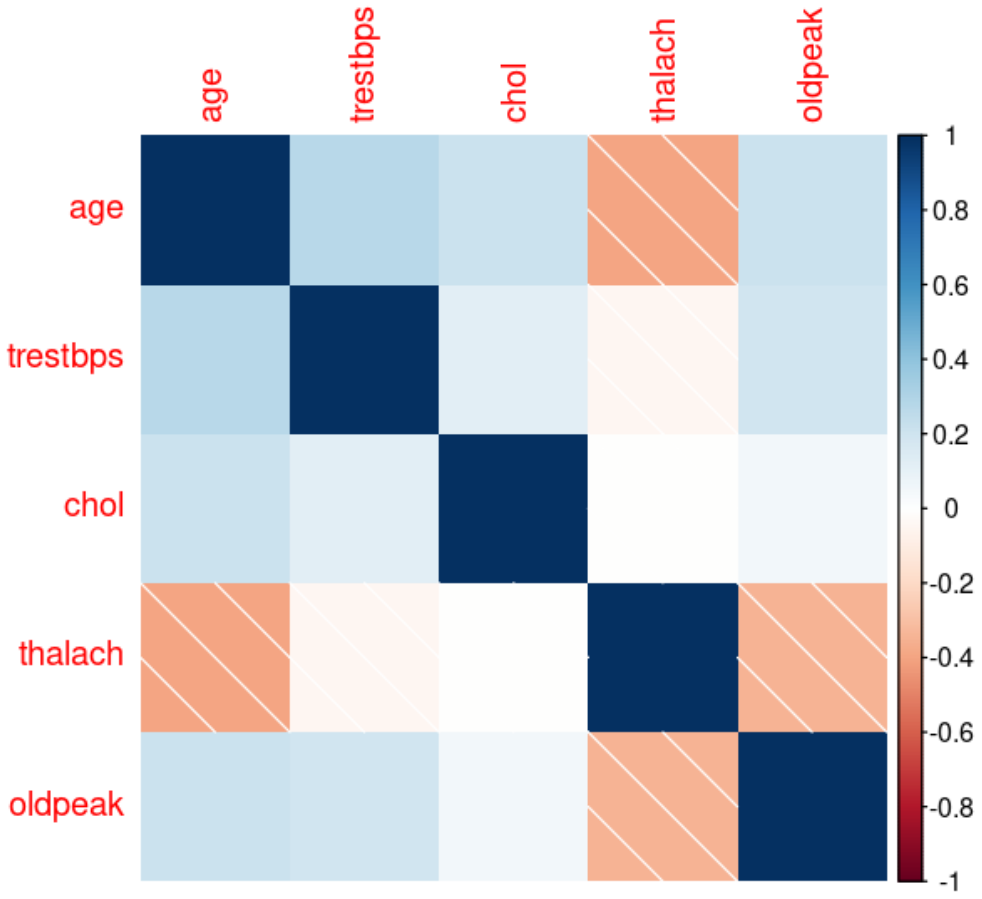
- Comparing Blood Pressure across types of pain in males and females using box plot



- Comparing Cholesterol levels across types of pain in males and females using box plot



- Correlation (method = 'number')
    - The same correlation on methods like “shade”, “square”, “circle”, and type like “upper” and “lower” are also implemented to view the correlation in different visual formats.



LOGISTIC REGRESSION
(A LINEAR MODEL FOR BINARY CLASSIFICATION AND PREDICTIVE MODELING)

We Verify if the data is imbalanced or not - The data would be imbalanced if a particular level/type of an attribute has only 1 or 2 values, which might get in the way of a smoother regression.

Some quality control is done by making sure all of the factor levels are represented by people with and without heart disease (target).

We also try to exclude variables that only have 1 or 2 samples in a category since +/- one or two samples can have a large effect on the odds/log(odds).
For example, for the attribute “Sex”, we need to make sure that healthy and diseased samples come from each gender (male and female). If only male/female samples have heart disease, the other samples must be removed. This is done using the xtabs() function (used for cross tabulation), to which the dataset is passed, and the "model syntax" to select the columns in the dataset from which a table has to be built.

The above-mentioned procedure is applied to all the boolean and categorical variables that are being used to predict heart disease.

- The output of the above procedure is as follows:

```
      sex
target FEMALE MALE
   NO      39   55
   YES     57  152
      cp
target ASYMPTOMATIC ATYPICAL ANGINA NON-ANGINAL PAIN
   NO            69              11               14
   YES           97              39               73
      fbs
target <=120 >120
   NO     74   20
   YES   184   25
      restecg
target ABNORMALITY NORMAL PROBABLE OR DEFINITE
   NO           35     56                    3
   YES         117     91                    1
      exang
target  NO YES
   NO   47  47
   YES 157  52
      slope
target   0   1   2
   NO   10  58  26
   YES  11  82 116
      ca
target   0   1   2   3   4
   NO   39  20  21  14   0
   YES 136  45  17   6   5
```

Logistic Regression model using "Sex" as the only parameter

At first, a simple model is built using "Sex" as the only parameter, before the actual implementation of Logistic Regression Using all the parameters, to see if gender (female/male) is a good predictor.

- The raw data regarding heart disease and sex

	sex	
target	FEMALE	MALE
NO	39	55
YES	57	152

From the above table…

- Most of the males have more chances of getting heart disease when compared to females.
- Being female is likely to decrease the odds in being unhealthy. In other words, if a sample is female, the odds are against it that it will be unhealthy.
- Being male is likely to increase the odds in being unhealthy. In other words, if a sample is male, the odds are for it being unhealthy.

The Logistic Regression model using "Sex" as the only parameter is now built.

- The glm() function is used which performs Generalized Linear Models.
  Here, we specify the binomial family of Generalized Linear Models, which makes glm() do Logistic Regression.
  Lastly, we summarize the Logistic Regression variable

```
Call:
glm(formula = target ~ sex, family = "binomial", data = data2)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6281  -1.3422   0.7859   0.7859   1.0211

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   0.3795     0.2078   1.826   0.0678 .
sexMALE       0.6371     0.2607   2.444   0.0145 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 375.29  on 302  degrees of freedom
Residual deviance: 369.37  on 301  degrees of freedom
AIC: 373.37

Number of Fisher Scoring iterations: 4
```

From the above summary…

- It is evident that the Deviance Residuals return good results as they are close to being centred on 0 and are roughly symmetrical. Deviance residuals represent the contributions of individual samples to the deviance D.

- The coefficients correspond to the following model:
  target = 0.3795 + 0.6371 * the patient is male (0 if female, otherwise 1)

  Therefore, if we are predicting heart disease for a female patient using the above formula, the log(odds), which is the measure of the likeliness of a female having a heart disease, is 0.3795.
  Similarly, prediction of heart disease for a male patient using the above formula, gives the answer as 0.3795 + 0.6371.

- In the column "Estimate" of the Coefficient output…
  Since, the first term, i.e. "Intercept" (0.3795) is the log(odds) of a female having heart disease, the second term, i.e. "SexMALE" (0.6371) indicates the increase in the log(odds) that a male has of having a heart disease.
  In other words, the second term is the log(odds ratio) of the odds that a male will have a heart disease over the odds that a female will have a heart disease.
  Both log(odds) and log(odds ratio) are calculated exclusively and separately in the subsequent steps.

- The "Std. Error" and "z value" columns of the Coefficient output shows how the Wald's test was computed for both ("Intercept" and "SexMALE") the coefficients. The Wald test works by testing the null hypothesis that a set of parameters is equal to some value. In the model being tested here, the null hypothesis is that the two coefficients of interest are simultaneously equal to zero. After running the logistic regression model, the Wald test can be used.

- Also, both p-values are well below 0.05, and thus, the log(odds) and log(odds ratios) are statistically significant.
  The p-value for each term tests the null hypothesis that the coefficient is equal to zero (no effect). A low p-value (< 0.05) indicates that you can reject the null hypothesis.

- The Default Dispersion Parameter is what is listed next in the Logistic Regression output.
  In Logistic Regression the mean of the data is estimated, from which the variance is derived. This is unlike Normal Linear Regression, where both the mean and the variance are estimated.
  Since, in the case of Logistic Regression, the variance is derived from the mean and not estimated from the data, it is possible that it is underestimated. This can be corrected, by adjusting the dispersion parameter in the above “summary()” command.

- Next comes the Null Deviance and the Residual Deviance.
  These are used to compare models, compute R-squared value and an overall p-value.
  The R-squared value (more correctly known as Pseudo R-squared value or McFadden's Pseudo R-squared value in Logistic Regression) is primarily used for comparing multiple models that fit to the same dataset.

- The AIC (Akaike Information Criterion), which, in this context, is the Residual Deviance adjusted for the number of parameters in the model. The AIC can be used to compare one model to another.

- Lastly, the Number of Fisher Scoring iterations tells how quickly the glm() function converged on the Maximum Likelihood Estimates for the coefficients.
  Maximum Likelihood Estimation is a probabilistic framework for estimating the parameters of a model.
  In Maximum Likelihood Estimation, we wish to maximize the conditional probability of observing the data (X), given a specific probability distribution and its parameters (theta).

All the above outputs from the summary of the Logistic Regression variable, can be used further in advanced data analytics using regression.

- Next, the log(odds), i.e the "Intercept" value and the log(odds ratio), i.e. the "SexMALE" value are calculated exclusively and separately, which come out to be the same as those obtained in the summary of the Logistic Regression variable. This suggests, that our model is working fine till now.

- Now, we calculate the overall "Pseudo R-squared" and its p-value
  Since we are doing logistic regression…
  Null devaince = 2*(0 - LogLikelihood(null model))
  = -2*LogLikihood(null model)
  Residual deviacne = 2*(0 - LogLikelihood(proposed model))
  = -2*LogLikelihood(proposed model)
  For **McFadden's Pseudo R^2**

  - First, we calculate the log-likelihood of the null model using the logistic variable by getting the value of the null deviance and dividing it by -2.

  - Then, we calculate the log-likelihood of the proposed model (The model we are building) using the logistic variable by getting the value of the residual deviance and dividing it by -2.

- Now, we calculate McFadden's Pseudo R^2, which is given by
  (ll.null - ll.proposed) / ll.null
  This Pseudo R^2 value can be interpreted as the overall effect size.
  For the given model, it is found out to be **0.05812569**.

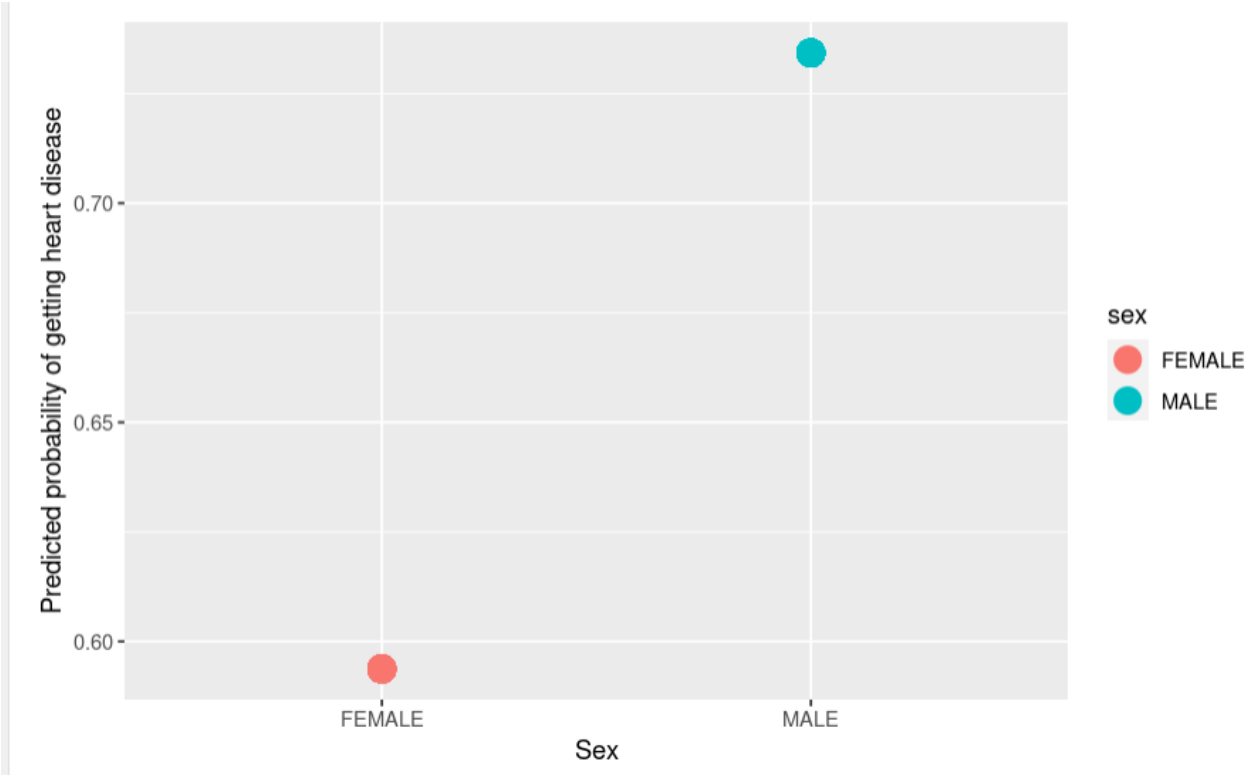
- The same log-likelihoods can be used to calculate a p-value for the R^2 value computed above, using a Chi-square distribution.
  chi-square value = 2*(LL(Proposed) - LL(Null)) = 0.01498277

  p-value = 1 - pchisq(chi-square value, df = 1) = 0.01498277

- Now, the logistic regression predicts whether a patient is having a heart disease or not, given that a patient is either female or male (and no other data about them).
  Here, we create a new data-frame that contains the probabilities of having a heart

disease along with the actual actual heart disease status (the value of the target attribute in the given dataset).

Lastly, we plot the graph for the probability of a person having a heart disease or not, with “Sex” as the only parameter.



As is evident from the above graph, females have a much lower probability of getting a heart disease as compared to males.

- Since there are only two probabilities (one for females and one for males), a table can be used to summarize the predicted probabilities.

	sex	
probability.of.target	FEMALE	MALE
0.59375	96	0
0.734299516907657	0	207

Logistic Regression model using all attributes as parameters

Data Splicing is done , where the data is split into 70% training data and 30% testing data.

Now, a model is built with all of the data available to predict heart disease.
The description of the code and the outputs for this model are similar to that for the model built for predicting heart disease using Sex as the only parameter.

The summary of the Logistic Regression variable for this model is as follows:

```
Call:
glm(formula = target ~ ., family = "binomial", data = train_data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2721  -0.3192   0.3599   0.7044   2.3517

Coefficients:
                            Estimate Std. Error z value Pr(>|z|)
(Intercept)                -1.771e+01  1.624e+03  -0.011  0.99130
sexMALE                     1.844e+00  5.864e-01   3.145  0.00166 **
fbs>120                    -1.410e+00  6.348e-01  -2.222  0.02628 *
exangYES                   -1.130e+00  4.913e-01  -2.300  0.02144 *
cpATYPICAL ANGINA          -1.010e+00  6.464e-01  -1.562  0.11830
cpNON-ANGINAL PAIN          8.792e-01  5.596e-01   1.571  0.11618
restecgNORMAL              -4.530e-01  4.072e-01  -1.113  0.26586
restecgPROBABLE OR DEFINITE -1.679e+01  1.587e+03  -0.011  0.99155
slope1                     -8.408e-01  1.051e+00  -0.800  0.42390
slope2                     -1.561e-01  1.129e+00  -0.138  0.89009
ca1                         1.463e-01  5.524e-01   0.265  0.79117
ca2                        -2.494e-01  6.706e-01  -0.372  0.70994
ca3                        -2.137e+00  9.309e-01  -2.296  0.02169 *
ca4                         1.179e+01  1.696e+03   0.007  0.99446
thal1                       1.768e+01  1.624e+03   0.011  0.99132
thal2                       1.780e+01  1.624e+03   0.011  0.99125
thal3                       1.668e+01  1.624e+03   0.010  0.99181
age                         5.173e-03  2.753e-02   0.188  0.85096
trestbps                   -2.807e-03  1.318e-02  -0.213  0.83132
chol                       -1.467e-04  3.751e-03  -0.039  0.96881
thalach                     1.348e-02  1.001e-02   1.347  0.17812
oldpeak                    -1.304e-01  2.275e-01  -0.573  0.56671
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 236.33  on 195  degrees of freedom
Residual deviance: 167.39  on 174  degrees of freedom
AIC: 211.39

Number of Fisher Scoring iterations: 15
```

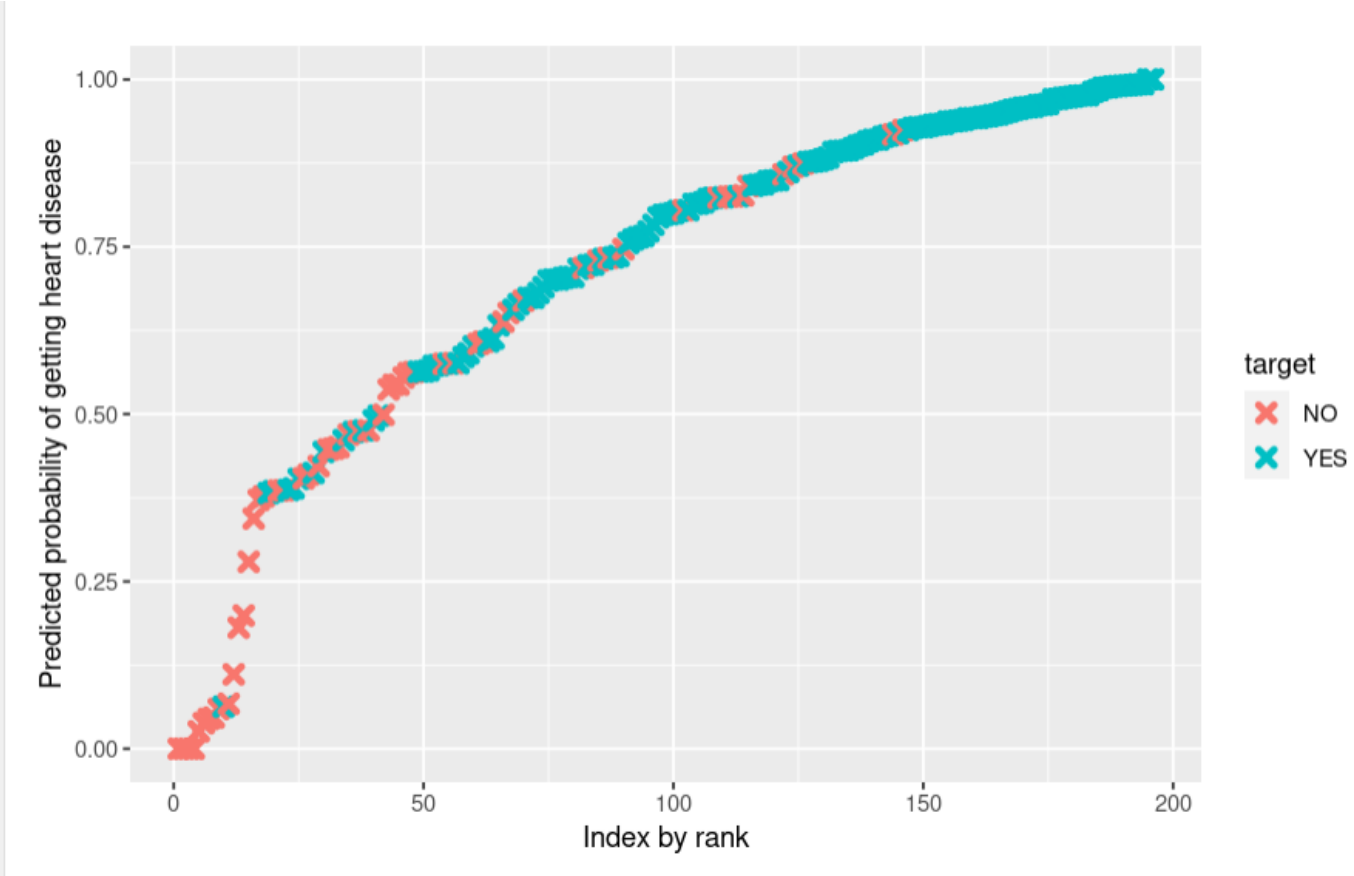
In the above summary output, the description of most of the parameters are similar to that of the previous model.

However, it can be inferred that,

- We can see that age is not a good predictor as it has a large p-value.
- Gender/Sex is Still a very good predictor, with a small p-value, as already evident from the previous model.

- The Residual Deviance and the AIC values are both much smaller for this model, than they were for the simple model, which was built using Sex as the only parameter to predict heart disease.

- Now, we calculate the overall "Pseudo R-squared" (McFadden's Pseudo R^2 = [LL(Null) - LL(Proposed)] / LL(Null)) and its p-value from the previously defined formulas. It comes out to be **0.01577058.**
- The p-value is very small (~0) as the R^2 value is large.
- Now, the logistic regression predicts whether a patient is having a heart disease or not, with all the different data about the person being provided to the model.
  - Here, we create a new data-frame that contains the probabilities of having a heart disease along with the actual heart disease status (the value of the Target attribute in the given dataset).
  - Then the data-frame is sorted from low probability to higher probability.
- Lastly, we add a new column named 'rank' to the data-frame, that has the rank of each sample, from low probability to high probability, as obtained by sorting them in the last command.
- Lastly, we plot the predicted probabilities for each sample having heart disease, and colorize the plot by whether or not they actually had heart disease.



The above graph shows the predicted probabilities of whether a patient has heart disease or not, along with their actual heart disease status.

- The 'Index by rank' label on the x-axis gives the position/rank of the given data point in the sorted data-frame, which is being plotted here along with the probabilities.
- Most of the people with heart disease (ones in blue), are predicted to have a high probability of having a heart disease.
- Most of the people without heart disease (ones in red), are predicted to have a low probability of having a heart disease.

We also made a confusion matrix for predicted and actual values for testing dataset

Actual_value	predicted_value FALSE	TRUE
NO	17	20
YES	6	64

And the **ACCURACY** is **80%**

Thus, it can be said that our Logistic Regression Model has worked successfully well.

KNN CLASSIFICATION ALGORITHM

- First, we convert all variables to integers, as KNN works only with numerical values, and create a new data-frame named 'heart_knn' for this. A glimpse of the data-frame is as follows

```
Rows: 303
Columns: 14
$ age      <int> 63, 37, 41, 56, 57, 57, 56, 44, 52, 57, 54, 48, 49, 64, 58, 50, 58, 66, 43, 69, 59, 44, 42, 61, 40, 71, 59, 51, 65, 53, 41, 65…
$ sex      <int> 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, …
$ cp       <int> 3, 2, 1, 1, 0, 0, 1, 1, 2, 2, 0, 2, 1, 3, 3, 2, 2, 3, 0, 3, 0, 2, 0, 2, 3, 1, 2, 2, 2, 2, 1, 0, 1, 2, 3, 2, 2, 2, 2, 2, 1, …
$ trestbps <int> 145, 130, 130, 120, 120, 140, 140, 120, 172, 150, 140, 130, 130, 110, 150, 120, 120, 150, 150, 140, 135, 130, 140, 150, 140, 1…
$ chol     <int> 233, 250, 204, 236, 354, 192, 294, 263, 199, 168, 239, 275, 266, 211, 283, 219, 340, 226, 247, 239, 234, 233, 226, 243, 199, 3…
$ fbs      <int> 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, …
$ restecg  <int> 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, …
$ thalach  <int> 150, 187, 172, 178, 163, 148, 153, 173, 162, 174, 160, 139, 171, 144, 162, 158, 172, 114, 171, 151, 161, 179, 178, 137, 178, 1…
$ exang    <int> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, …
$ oldpeak  <int> 2, 3, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 2, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, …
$ slope    <int> 0, 0, 2, 2, 2, 1, 1, 2, 2, 2, 2, 2, 1, 2, 1, 2, 0, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 0, 2, 2, 2, 0, 2, 0, 2, 2, 2, 2, 2, 1, …
$ ca       <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 2, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, …
$ thal     <int> 1, 2, 2, 2, 2, 1, 2, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2, 3, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2, 2, 2, 3, 2, 2, 2, 2, 2, …
$ target   <int> 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, …
```

- Next, the data is cleaned, in the same way as was done for Logistic Regression.

- Next, the data is normalized so that the output remains unbiased. Also, the 'Target' variable is removed from the normalized data since it'sthe response variable that needs to be predicted. A glimpse of the normalized data:

	age <dbl>	sex <dbl>	cp <dbl>	trestbps <dbl>	chol <dbl>	fbs <dbl>	restecg <dbl>	thalach <dbl>
1	0.7083333	1	1.0000000	0.4811321	0.2442922	1	0.0	0.6030534
2	0.1666667	1	0.6666667	0.3396226	0.2831050	0	0.5	0.8854962
3	0.2500000	0	0.3333333	0.3396226	0.1780822	0	0.0	0.7709924
4	0.5625000	1	0.3333333	0.2452830	0.2511416	0	0.5	0.8167939
5	0.5833333	0	0.0000000	0.2452830	0.5205479	0	0.5	0.7022901
6	0.5833333	1	0.0000000	0.4339623	0.1506849	0	0.5	0.5877863

- The structure of the normalized data-frame is viewed, and it is found out that all the variables are converted to the numerical type.

```
'data.frame':   303 obs. of  13 variables:
 $ age     : num  0.708 0.167 0.25 0.562 0.583 ...
 $ sex     : num  1 1 0 1 0 1 0 1 1 1 ...
 $ cp      : num  1 0.667 0.333 0.333 0 ...
 $ trestbps: num  0.481 0.34 0.34 0.245 0.245 ...
 $ chol    : num  0.244 0.283 0.178 0.251 0.521 ...
 $ fbs     : num  1 0 0 0 0 0 0 0 1 0 ...
 $ restecg : num  0 0.5 0 0.5 0.5 0.5 0 0.5 0.5 0.5 ...
 $ thalach : num  0.603 0.885 0.771 0.817 0.702 ...
 $ exang   : num  0 0 0 0 1 0 0 0 0 0 ...
 $ oldpeak : num  0.333 0.5 0.167 0 0 ...
 $ slope   : num  0 0 1 1 1 0.5 0.5 1 1 1 ...
 $ ca      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ thal    : num  0.333 0.667 0.667 0.667 0.667 ...
```

- Data Splicing is done next, where the normalized data is split into 80% training data and 20% testing data, and a separate data-frame is created for the ‘Target’ variable so that the final outcome can be compared withthe actual value.

Now, we build a Machine Learning model using KNN.

- We use the ‘class’ package for KNN.
- The number of observations in the training data is found out to be 242.
    - This is required for initializing the value of 'K' in the model, as one of the ways to find the optimal 'K' value is to calculate the square root of the total number of observations in the dataset. This square root will give the ‘K’ value.
- Since, there are 237 observations, we take two values of ‘K’ as 15 and 16, as the square root of 237 comes out to be 15.394.
- Now, we evaluate the model using the accuracy score, where the accuracy of the model using K = 15 is found out to be 68.852, K = 16 is found out to be 65.573.

- Checking the prediction of the model against actual value in tabular form for k = 15

```
      test.heart_labels
knn_15  0  1
     0  5  4
     1 15 37
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1
Levels: 0 1
```

- Checking the prediction of the model against actual value in tabular form for k = 16

```
      test.heart_labels
knn_16  0  1
     0  2  3
     1 18 38
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1
Levels: 0 1
```

- Using the confusion matrix to calculate the accuracy of the KNN model with K value set to 15. Here, the Accuracy = 68.85%.

```
Confusion Matrix and Statistics

      test.heart_labels
knn_15  0  1
     0  5  4
     1 15 37

               Accuracy : 0.6885
                 95% CI : (0.5571, 0.801)
    No Information Rate : 0.6721
    P-Value [Acc > NIR] : 0.45198

                  Kappa : 0.1774

 Mcnemar's Test P-Value : 0.02178

            Sensitivity : 0.25000
            Specificity : 0.90244
         Pos Pred Value : 0.55556
         Neg Pred Value : 0.71154
             Prevalence : 0.32787
         Detection Rate : 0.08197
   Detection Prevalence : 0.14754
      Balanced Accuracy : 0.57622

       'Positive' Class : 0
```

- Using the confusion matrix to calculate the accuracy of the KNN model with K value set to 16. Here, the Accuracy = 65.573%

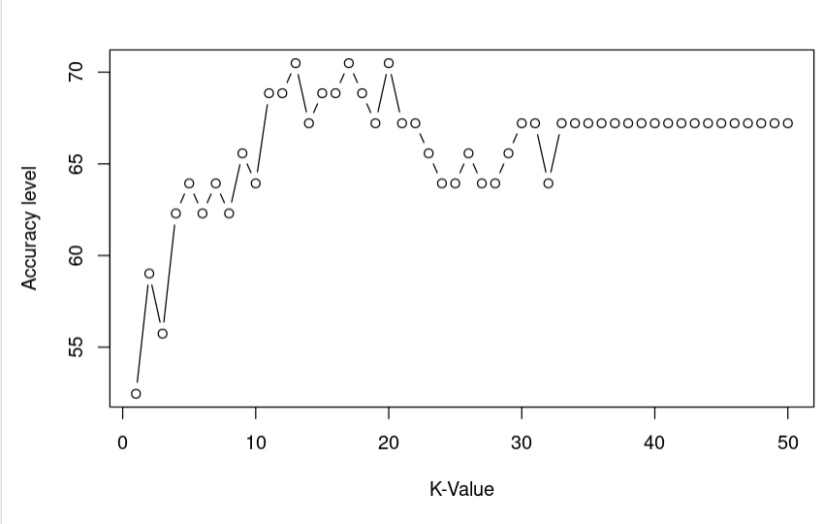
```
Confusion Matrix and Statistics

          test.heart_labels
knn_16     0  1
         0  2  3
         1 18 38

                  Accuracy : 0.6557
                    95% CI : (0.5231, 0.7727)
       No Information Rate : 0.6721
       P-Value [Acc > NIR] : 0.66339

                     Kappa : 0.0332

    Mcnemar's Test P-Value : 0.00225

               Sensitivity : 0.10000
               Specificity : 0.92683
            Pos Pred Value : 0.40000
            Neg Pred Value : 0.67857
                Prevalence : 0.32787
            Detection Rate : 0.03279
      Detection Prevalence : 0.08197
         Balanced Accuracy : 0.51341

          'Positive' Class : 0
```

- In order to optimize and improve the accuracy of the model, at first, we create a loop that calculates the accuracy of the KNN model for ‘K’ values ranging from 1 to 50, and then plot the maximum percentage accuracy graph. This way we can check which ‘K’ value will result in the most accurate model. The first 30 values found are:

```
1 = 52.45902
2 = 59.01639
3 = 55.7377
4 = 62.29508
5 = 63.93443
6 = 62.29508
7 = 63.93443
8 = 62.29508
9 = 65.57377
10 = 63.93443
11 = 68.85246
12 = 68.85246
13 = 70.4918
14 = 67.21311
15 = 68.85246
16 = 68.85246
17 = 70.4918
18 = 68.85246
19 = 67.21311
20 = 70.4918
21 = 67.21311
22 = 67.21311
23 = 65.57377
24 = 63.93443
25 = 63.93443
26 = 65.57377
27 = 63.93443
28 = 63.93443
29 = 65.57377
30 = 67.21311
31 = 67.21311
32 = 63.93443
33 = 67.21311
```

The accuracy value is the highest (70.4918) for k = 13, 17 and 20, which is a very good accuracyscore, as the given dataset is very small.

Representing the above data graphically:



We thus create a model with k = 13

- The prediction of the model against the actual values in tabular form for k = 13

```
     test.heart_labels
knn_13  0  1
     0  5  3
     1 15 38
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 0 1 0 1 1 1 1 1
Levels: 0 1
```

- The confusion matrix to calculate the accuracy of the model with K value set to 13:

```
Confusion Matrix and Statistics

      test.heart_labels
knn_13  0  1
     0  5  3
     1 15 38

               Accuracy : 0.7049
                 95% CI : (0.5743, 0.8148)
    No Information Rate : 0.6721
    P-Value [Acc > NIR] : 0.346273

                  Kappa : 0.2089

 Mcnemar's Test P-Value : 0.009522

            Sensitivity : 0.25000
            Specificity : 0.92683
         Pos Pred Value : 0.62500
         Neg Pred Value : 0.71698
             Prevalence : 0.32787
         Detection Rate : 0.08197
   Detection Prevalence : 0.13115
      Balanced Accuracy : 0.58841

       'Positive' Class : 0
```

As expected, the model gives an accuracy of **70.49%** , which is very goodsince we worked with a small dataset.

SVM CLASSIFICATION ALGORITHM

SVM (Support Vector Machine) is a supervised machine learning algorithm which is mainly used to classify data into different classes. Unlike most algorithms, SVM makes use of a hyperplane which acts like a decision boundary between the various classes. SVM can be used to generate multiple separating hyperplanes such that the data is divided into segments and each segment contains only one kind of data.

Here, we store the heart_knn data-frame variable into a new variable, heart_svm. This way, we don't have to clean the data another time, and we get apre-processed data.

age <int>	sex <int>	cp <int>	trestbps <int>	chol <int>	fbs <int>	restecg <int>	thalach <int>	exang <int>	oldpeak <int>
63	1	3	145	233	1	0	150	0	2
37	1	2	130	250	0	1	187	0	3
41	0	1	130	204	0	0	172	0	1
56	1	1	120	236	0	1	178	0	0
57	0	0	120	354	0	1	163	1	0
57	1	0	140	192	0	1	148	0	0
56	0	1	140	294	0	0	153	0	1
44	1	1	120	263	0	1	173	0	0
52	1	2	172	199	1	1	162	0	0
57	1	2	150	168	0	1	174	0	1

- Structure of the data-frame

```
'data.frame':   303 obs. of  14 variables:
 $ age     : int  63 37 41 56 57 57 56 44 52 57 ...
 $ sex     : int  1 1 0 1 0 1 0 1 1 1 ...
 $ cp      : int  3 2 1 1 0 0 1 1 2 2 ...
 $ trestbps: int  145 130 130 120 120 140 140 120 172 150 ...
 $ chol    : int  233 250 204 236 354 192 294 263 199 168 ...
 $ fbs     : int  1 0 0 0 0 0 0 0 1 0 ...
 $ restecg : int  0 1 0 1 1 1 0 1 1 1 ...
 $ thalach : int  150 187 172 178 163 148 153 173 162 174 ...
 $ exang   : int  0 0 0 0 1 0 0 0 0 0 ...
 $ oldpeak : int  2 3 1 0 0 0 1 0 0 1 ...
 $ slope   : int  0 0 2 2 2 1 1 2 2 2 ...
 $ ca      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ thal    : int  1 2 2 2 2 1 2 3 3 2 ...
 $ target  : int  1 1 0 1 1 1 0 1 1 1 ...
```

- Summary of the data-frame

```
      age              sex              cp            trestbps          chol            fbs              restecg          thalach
 Min.   :29.00   Min.   :0.0000   Min.   :0.000   Min.   : 94.0   Min.   :126.0   Min.   :0.0000   Min.   :0.0000   Min.   : 71.0
 1st Qu.:47.50   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:120.0   1st Qu.:211.0   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:133.5
 Median :55.00   Median :1.0000   Median :1.000   Median :130.0   Median :240.0   Median :0.0000   Median :1.0000   Median :153.0
 Mean   :54.37   Mean   :0.6832   Mean   :0.967   Mean   :131.6   Mean   :246.3   Mean   :0.1485   Mean   :0.5281   Mean   :149.6
 3rd Qu.:61.00   3rd Qu.:1.0000   3rd Qu.:2.000   3rd Qu.:140.0   3rd Qu.:274.5   3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:166.0
 Max.   :77.00   Max.   :1.0000   Max.   :3.000   Max.   :200.0   Max.   :564.0   Max.   :1.0000   Max.   :2.0000   Max.   :202.0
     exang           oldpeak          slope             ca              thal           target
 Min.   :0.0000   Min.   :0.0000   Min.   :0.000   Min.   :0.0000   Min.   :0.000   Min.   :0.0000
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:1.000   1st Qu.:0.0000   1st Qu.:2.000   1st Qu.:0.0000
 Median :0.0000   Median :0.0000   Median :1.000   Median :0.0000   Median :2.000   Median :1.0000
 Mean   :0.3267   Mean   :0.7657   Mean   :1.399   Mean   :0.7294   Mean   :2.314   Mean   :0.6898
 3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:2.000   3rd Qu.:1.0000   3rd Qu.:3.000   3rd Qu.:1.0000
 Max.   :1.0000   Max.   :6.0000   Max.   :2.000   Max.   :4.0000   Max.   :3.000   Max.   :1.0000
```

- Now, data splicing is done where we split the data into 80% training set (the train_heart_svm variable) and a 20% testing set (the test_heart_svm variable).
    - Here, we use the caret package provides a method createDataPartition(), which is basically used for partitioning the data into the training and testing set.
    - 3 parameters are passed to the createdatapartition() function:
        - The “y” parameter takes the value of the variable (target variable, i.e. Target) according to which data needs to be partitioned.
        - The “p” parameter holds a decimal value in the range of 0-1, which represents the percentage of the split. Here, p=0.8, meaning that data split is done in 80:20 ratio. So, 80% of the data is used for training and the remaining 20% is for testing the model.
        - The “list” parameter is for whether to return a list or matrix - set to FALSE for not returning a list.
        - The createDataPartition() method returns a matrix “svm_spliced”, which contains the training dataset, which in turn is stored in the ‘train_heart_svm’, and the remaining 20% of the data is stored in the 'test_heart_svm' variable.

- Dimensions of the training data-frame is found out to be [243, 14], and that of the testing data-frame is found out to be [60, 14].

- From the output of the summary() command, we found out that the value of the 'Target' variable is not standardized, and holds only '0' or '1'.Instead, 'Target' should be a categorical variable. Thus, it is standardized (i.e., changed to a factor) in the next step.

- Before the actual training, the trainControl() method is implemented, which controls all the computational overheads so that the train() function provided by the caret package can be uesd. The training method trains the data on different algorithms.
    - The trainControl() method takes 3 parameters:
        - The “method” parameter (set to repeatedcv i.e., repeated cross-validation method) defines the resampling method.
        - The “number” parameter (set to 10), which holds the number of resampling iterations.

        - The “repeats” parameter (set to 3) contains the sets to compute for the repeated cross-validation.

- The trainControl() method returns a list, which is passed on to the train() method.
    - The train() method should be passed with the “method” parameter as “svmLinear”. Here, all attributes are used as parameters in the classifier, and Target is the target variable.
        - The “trControl” parameter should be passed with results from the trianControl() method.
        - The “preProcess” parameter is for preprocessing the training data, which is passed with 2 values i.e., “center,” and “scale” parameters. These two values help in centering and scaling the data. After pre-processing, the train_heart_svm data is converted with mean value as approximately “0” and standard deviation as “1”.
        - The “tuneLength” parameter holds an integer value, used for tuning the algorithm.

- The result of the train() method is saved in the svm_Linear variable.
  The result of the train() method, as stored in the svm_Linear variable (since, it is a linear model, it was just tested at value “C” = 1):

```
Support Vector Machines with Linear Kernel

243 samples
 13 predictor
  2 classes: '0', '1'

Pre-processing: centered (13), scaled (13)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 219, 218, 218, 218, 220, 220, ...
Resampling results:

  Accuracy   Kappa
  0.7213647  0.296347

Tuning parameter 'C' was held constant at a value of 1
```

- Now, the model is ready to predict the classes for the test set, for which the predict() method of the caret package is used.
  The predict() method returns a list, which is saved in the test_pred variable.

```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 0 1 0 1
[39] 0 1 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0 0 1 0 1 0
Levels: 0 1
```

- Accuracy of our model using the confusion matrix

```
Confusion Matrix and Statistics


test_pred  0  1
        0  7  8
        1 11 34

               Accuracy : 0.6833
                 95% CI : (0.5504, 0.7974)
    No Information Rate : 0.7
    P-Value [Acc > NIR] : 0.6692

                  Kappa : 0.2083

 Mcnemar's Test P-Value : 0.6464

            Sensitivity : 0.3889
            Specificity : 0.8095
         Pos Pred Value : 0.4667
         Neg Pred Value : 0.7556
             Prevalence : 0.3000
         Detection Rate : 0.1167
   Detection Prevalence : 0.2500
      Balanced Accuracy : 0.5992

       'Positive' Class : 0
```

Ther, the Accuracy = 68.33%
The svmLinear classifier can be build using the above procedure.

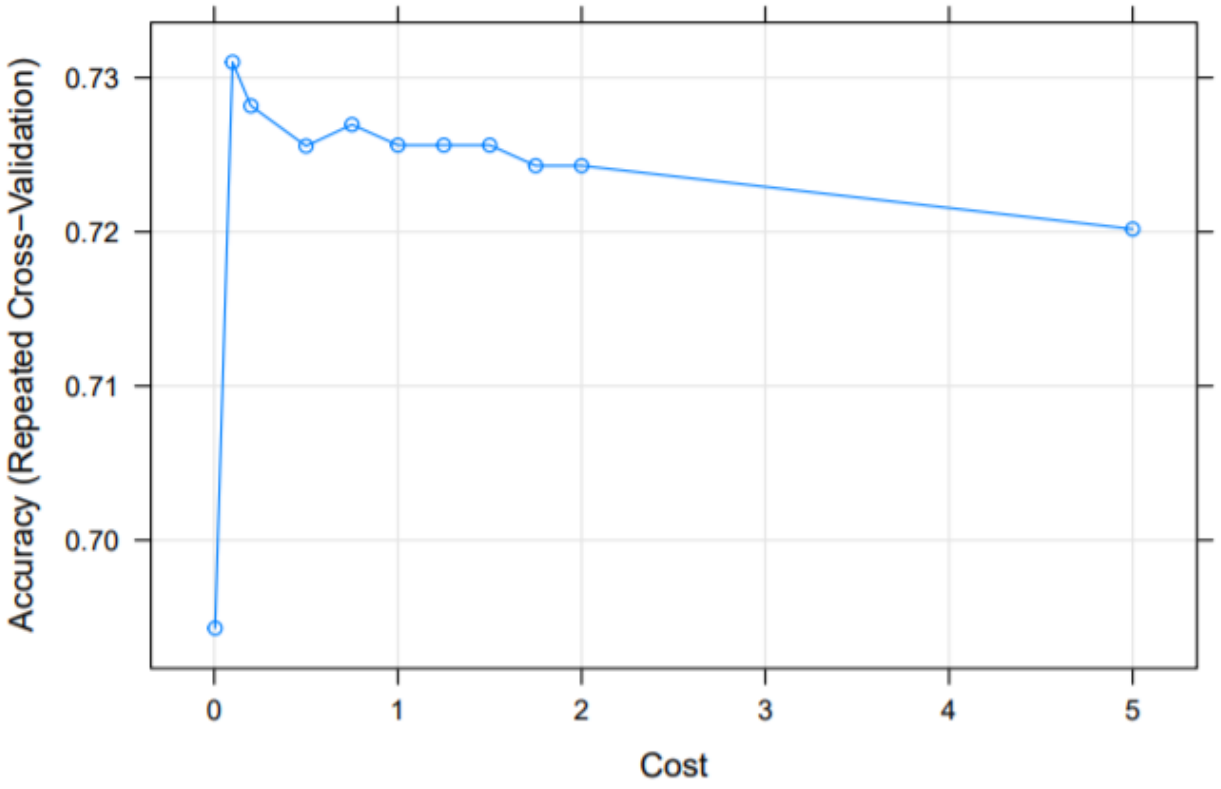
Now, some customization for selecting C (Cost) value in the Linear classifier is done by inputting values in grid search. Thus, we build & tune the SVM classifier with different values of C.

- A few values of C are put into the "grid" data-frame using expand.grid() function.
- Then, this data-frame is used for testing the classifier at specific C values (as opposed to the previous build, where the model was tested for only C = 1), by putting it in the train() method with the "tuneGrid" parameter. The output of this procedure is as follows:

```
Support Vector Machines with Linear Kernel

243 samples
 13 predictor
  2 classes: '0', '1'

Pre-processing: centered (13), scaled (13)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 219, 218, 218, 218, 219, 218, ...
Resampling results across tuning parameters:
```

```
C      Accuracy   Kappa
0.005  0.6942874  0.0298571
0.100  0.7310097  0.3086085
0.200  0.7281715  0.3014922
0.500  0.7255652  0.2984450
0.750  0.7269541  0.3019082
1.000  0.7256208  0.2995446
1.250  0.7256208  0.2995446
1.500  0.7256208  0.2995446
1.750  0.7242874  0.2973484
2.000  0.7242874  0.2973484
5.000  0.7201763  0.2905990

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was C = 0.1.
```

- The graphical output for the above-mentioned procedure:



The above plot is showing that our classifier is giving best accuracy at C = 0.1.

Now, we optimize our model by making predictions using the above model for the test set.

- Accuracy of the above model using the confusion-matrix:

```
Confusion Matrix and Statistics


test_pred_grid  0  1
             0  8  7
             1 10 35

               Accuracy : 0.7167
                 95% CI : (0.5856, 0.8255)
    No Information Rate : 0.7
    P-Value [Acc > NIR] : 0.4514

                  Kappa : 0.2917

 Mcnemar's Test P-Value : 0.6276

            Sensitivity : 0.4444
            Specificity : 0.8333
         Pos Pred Value : 0.5333
         Neg Pred Value : 0.7778
             Prevalence : 0.3000
         Detection Rate : 0.1333
   Detection Prevalence : 0.2500
      Balanced Accuracy : 0.6389

       'Positive' Class : 0
```

The results of the confusion matrix show that this time the accuracy on the test set is **71.67%,** which is way more accurate than our previous result.

COMPARATIVE STUDY WITH THE EXISTING TECHNIQUES (IN TERMS OF PERFORMANCE METRICS CONSIDERED IN THE RELATED ARTICLES IN THE LITERATURE)

In the first paper we referred (by K. Polaraju et al[1]), the method used for analysis was linear regression with a 70/30 division of training /testing performed in C# language. One con to this is the amount of data used which could be unreliable especially during the training/testing division and data exploration wasn't also done vastly and there is a lack of machine learning algorithms used.
This is overcome in our model.

In the third paper we referred, (by Siddharth Mundra, Kiran Manjrekar et al[3]), various algorithm have been discussed, majority of them being of data mining algorithms including, KNN algorithm, Neural networks, Support vector machines, etc. Although the analysis was not performed in this paper.
However, we have performed two of these algorithms (KNN and SVM), and found out that SVM gives an accuracy of up to **71.67%.**

In the fourth paper being referred (by M.Marimuthu, M.Abinaya, K.S.Hariesh, K.Madhankumar, V.Pavithra et al [4]), the algorithms used for analysis were Gaussian Naive Bayes Classifier in which only a marginal success is achieved in the creation of predictive model for heart disease patients and hence there is a need for combinational and more complex models to increase the accuracy of the model. However, in the project we have made, we have been able to reach a maximum accuracy of **80%**, with a comparatively small dataset. So, it can be concluded, that if our work is applied on bigger datasets, then a much better accuracy in predicting a person's heart disease can be obtained, which in turn can be applied to real world problems.

CONCLUSION

In this project we successfully analyzed the dataset for heart disease prediction by performing data exploratory and data visualization in various forms such as bar plot, box plot, shader, pie chart etc. on all the attributes present in the dataset which represents the data in a very explanatory manner.

Further we did Logistic Regression on the entire dataset. At the end, the model could predict with a high accuracy if person has a heart disease or not.

We also implemented KNN and SVM Classification Algorithms, and obtained satisfactory results with accuracy rates, which is a good a performance for a dataset with relatively lesser data points.

As a future work, we plan on improving the already implemented models by manipulating various parameters until the best result is obtained.

REFERENCES

1. K.Polaraju, D. Durga Prasad, “Prediction of Heart Disease using Multiple Linear Regression Model”, International Journal of Engineering Development and Research Development, Volume 5 Issue 4. Number of pages – 7.

2. Akhand Pratap Singh, Dr. Bhupal Singh, “A Review on Heart Disease Prediction using Machine Learning”, Journal of Xi'an University of Architecture & Technology, Volume 12 Issue 3. Number of pages – 5.

3. Siddharth Mundra, Kiran Manjrekar, “Review on prediction of heart disease using data mining”, International Journal of advanced research ideas and innovations in technology, Volume 5 Issue 6. Number of pages – 3.

4. M.Marimuthu, M.Abinaya, K.S.Hariesh, K.Madhankumar, V.Pavithra, “A Review on Heart Disease Prediction using Machine Learning and Data Analytics Approach”, International Journal of Computer Applications, Volume 181 Issue 18. Number of pages – 3.

5. Mr. Chala Beyene, Prof. Pooja Kamat, “Survey on Prediction and Analysis the Occurrence of Heart Disease Using Data Mining Techniques”, International Journal of Pure and Applied Mathematics, Volume 118 Issue 8. Number of pages – 11.