Heart Disease Prediction

Milestone: Project Report

Group 3

Mugdha Sanjay Parbat - 002142372

Pranav Chandrakant Pulkundwar - 002121679

Telephone

+1 (617) 901-8417

+1 (617) 901-8418

Email ID

parbat.m@northeastern.edu

pulkundwar.p@northeastern.edu

Percentage of contribution by Student 1: 50%

Percentage of contribution by Student 2: 50%

Signature of Student 1:

Signature of Student 2:

Submission Date: 1st May 2022

Table of Contents

1. Problem Setting 3

2. Problem Definition 4

3. Data Sources 5

4. Data Description 6

5. Data Exploration 8

6. Data Mining Tasks 13

6.1. Data Preprocessing 13

7. Data Mining Models and Methods 14

7.1. K Nearest Neighbors (KNN) 14

7.2. Support Vector Machine (SVM) 15

7.3. Gaussian Naïve Bayes 16

7.4. Decision Tree 16

7.5. Random Forest 17

8. Performance Evaluation 18

8.1. Evaluations 19

9. Project Results 21

9.1. Classification Report 21

9.2. ROC 23

10. Project Outcome & Future Scope 24

10.1. Future Scope 24

1. Problem Setting

About 659,000 people in the United States die from heart disease each year. That is 1 in every 4 deaths. The conditions of narrowing and blockage in blood vessels that can lead to a heart-attack, angina or even stroke is referred to as a Cardiovascular disease. It has been discovered that half the people who suffer from it have regular cholesterol levels. It suggests that there are various other factors that contribute to heart disease. This is what makes heart disease difficult to detect, as there are various contributory risk factors, such as blood pressure, cholesterol levels, pulse rate, unhealthy diet habits, obesity, lack or extreme physical activity, and many other.

There are numerous types of heart diseases like one of the most common types is Coronary Artery disease killing 360,900 people in 2019. These numbers are alarming, and this is where machine learning and data mining plays a vital role. The Machine Learning methods assist in building algorithms which can make effective predications from large quantity of records that is being produced every day.

2. Problem Definition

In healthcare data analysis prediction of cardiovascular disease is one of the most critical and highly important subjects. Recent studies show that heart disease is one of the major causes of death regardless of the gender of a person.

In this project, different Machine Learning approaches are to be applied and compared to determine whether a person is suffering from heart disease or not. Data mining turns a large collection of raw data into something that can be used to gain information using various data mining processes which can assist us in taking informed decisions and predictions. For the purpose of heart disease prediction project various models were explored.

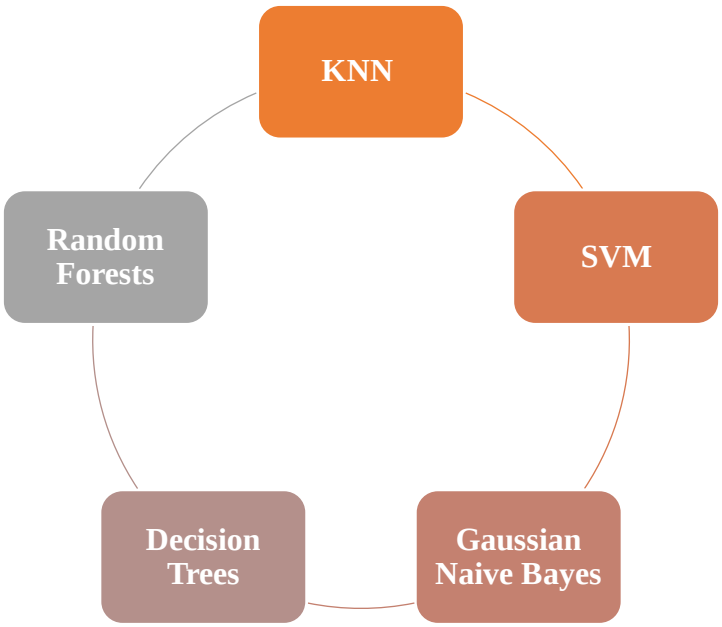

Figure 2.1: Prediction Models

3. Data Sources

This database contains 76 attributes out of which only 14 of them are being used. Sensitive information such as names and social security numbers of the patients are deleted from the database, replaced with dummy values.

The dataset has been obtained from the following website :-

https://archive.ics.uci.edu/ml/datasets/heart+disease

The following are the creators and donors:-Creators:

1. Hungarian Institute of Cardiology. Budapest: Andras Janosi, M.D.

2. University Hospital, Zurich, Switzerland: William Steinbrunn, M.D.

3. University Hospital, Basel, Switzerland: Matthias Pfisterer, M.D.

4. V.A. Medical Center, Long Beach, and Cleveland Clinic Foundation: Robert Detrano, M.D., Ph.D.

Donor: David W. Aha (aha '@' ics.uci.edu) (714) 856-8779

4. Data Description

This data is the collection of information from the Cleveland Heart Disease dataset taken from the UCI repository, formed by taking into consideration some of the information of 779 individuals. The dataset consists of 303 rows and 14 columns of data. The 14 features or columns considered in this analysis are as follows,

1. Age: Age of a person
2. Sex: Sex of a person

   Male =1

   Female = 0
3. Chest-pain type: Type of chest pain

   Typical angina = 1

   Atypical angina = 2

   Non - anginal pain = 3

   Asymptotic = 4
4. Resting Blood Pressure: Resting blood pressure in mmHg
5. Serum Cholesterol: Cholesterol level in mg/dl
6. Fasting Blood Sugar:

   Greater than 120 mg/dl = 1 (True)

   Less than 120 mg/dl = 0 (False)
7. Resting ECG
8. Max heart rate
9. Exercise induced Angina

   Yes = 1

   No = 0
10. ST depression induced by exercise relative to rest
11. Peak exercise ST segment

    1 = Up-sloping

    2 = Flat

    3 = Down-sloping
12. Number of major vessels

13. Thal (Thalassemia)

    Normal = 3

    Fixed defect = 6

    Reversible defect = 7

14. Diagnosis of heart disease

    Absence = 0

    Present = 1, 2, 3, 4

5. Data Exploration

First step of this project was performing exploratory data analysis on the records to find out key insights and patterns. The plot below uses pd.crosstab(), data visualization was created which is a cross tabulation of age and the target variable. It tells us how age Is a factor in variation of number of people who have heart disease and who don't. The graph shows that majority of people with 57 years of age to 61 years of age do not have heart disease. The age group between of 54 years and 41 years are the ones who suffer from heart disease the most.

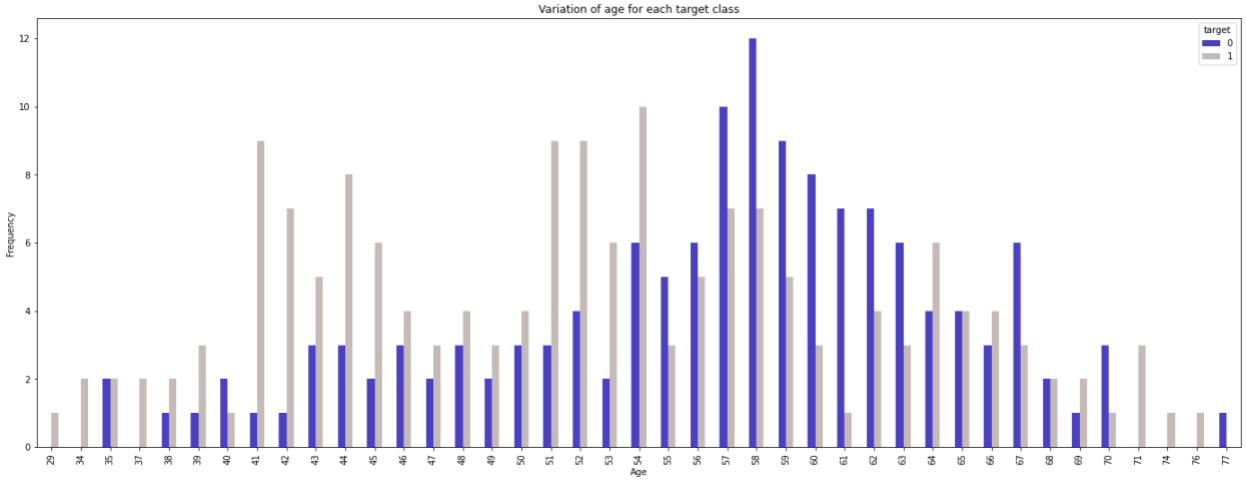

Figure 5.1: Variation of Age by Target Class

In data visualization shown below, the relationship between numerical (age) and categorical (sex) is shown by utilizing sns.catplot(). It is derived from the visualization, that males who suffer from heart disease are younger than the females who suffer from the heart disease.

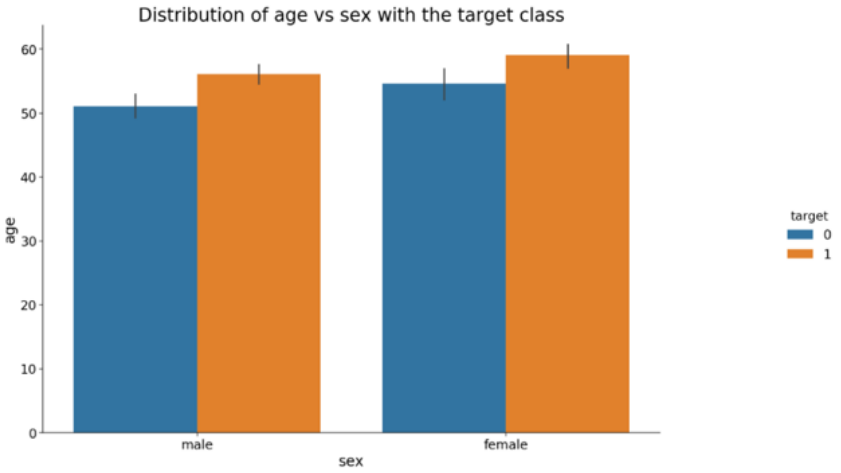

Figure 5.2: Distribution of Age vs Sex with Target Class

Plotting a correlation matrix helps to check if there are any correlations with the target variable. Seaborn's sns.heatmap() took in parameters for data, axes in which the heatmap is to be plotted in, and color combination to create the above correlation matrix. There are no strong correlations with the target variables. Some of the features have a positive and negative correlation with the target variable, but there are no features who have strong correlations with the target variable.

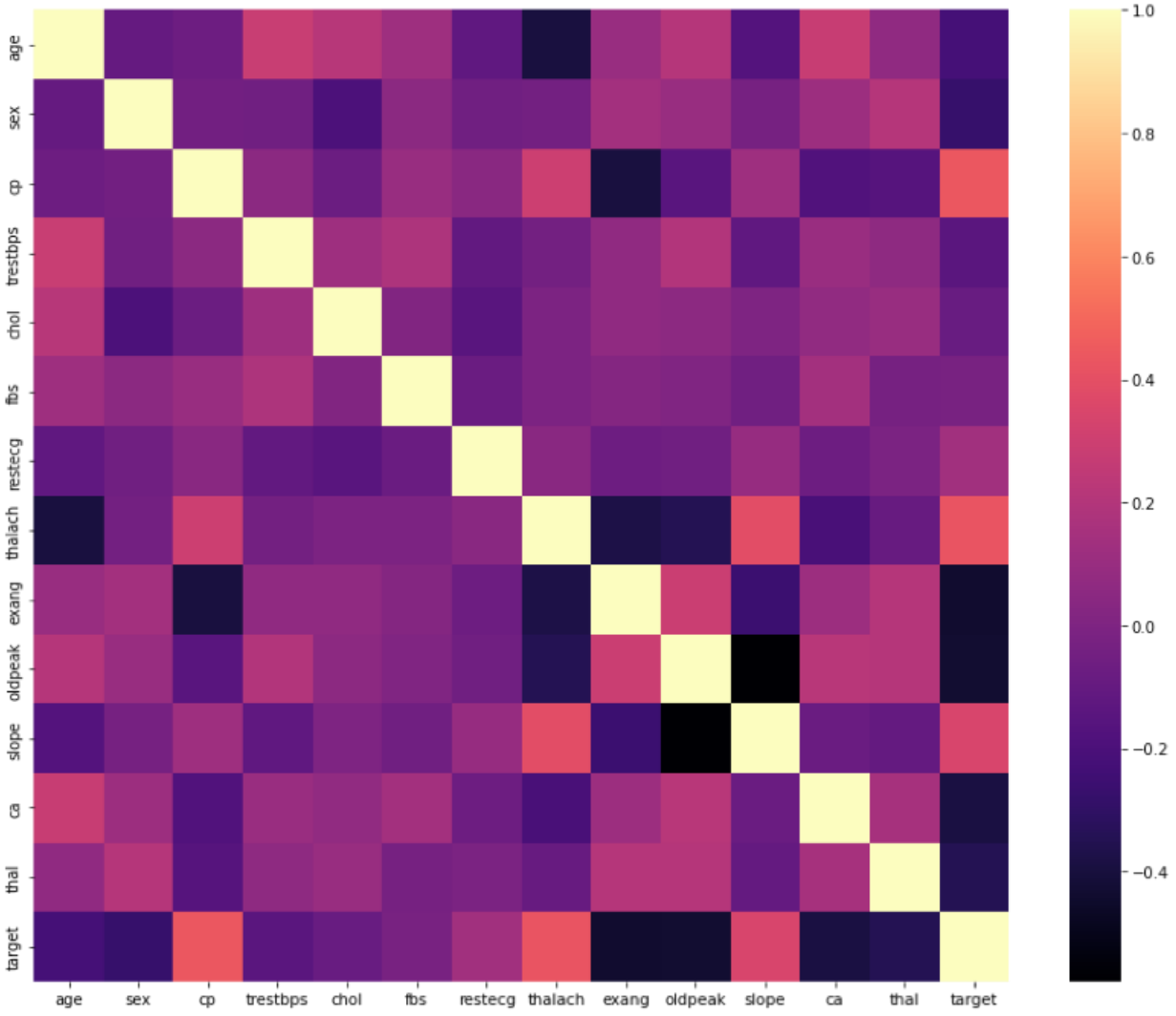

Figure 5.3: Correlation Heatmap

Histogram plots for all the features with a function and dataframe.hist() were created. This is one of the most important visualizations that can be created in fewer lines of code but gives significant information. For instance, it can be analyzed quickly that all the features and labels are distributed in different ranges, and hence, the data needs to be scaled before proceeding to the net step. There are some discrete bar plots, which are nothing but categorical variables, and there are various ways to handle categorical variables before building machine learning models.

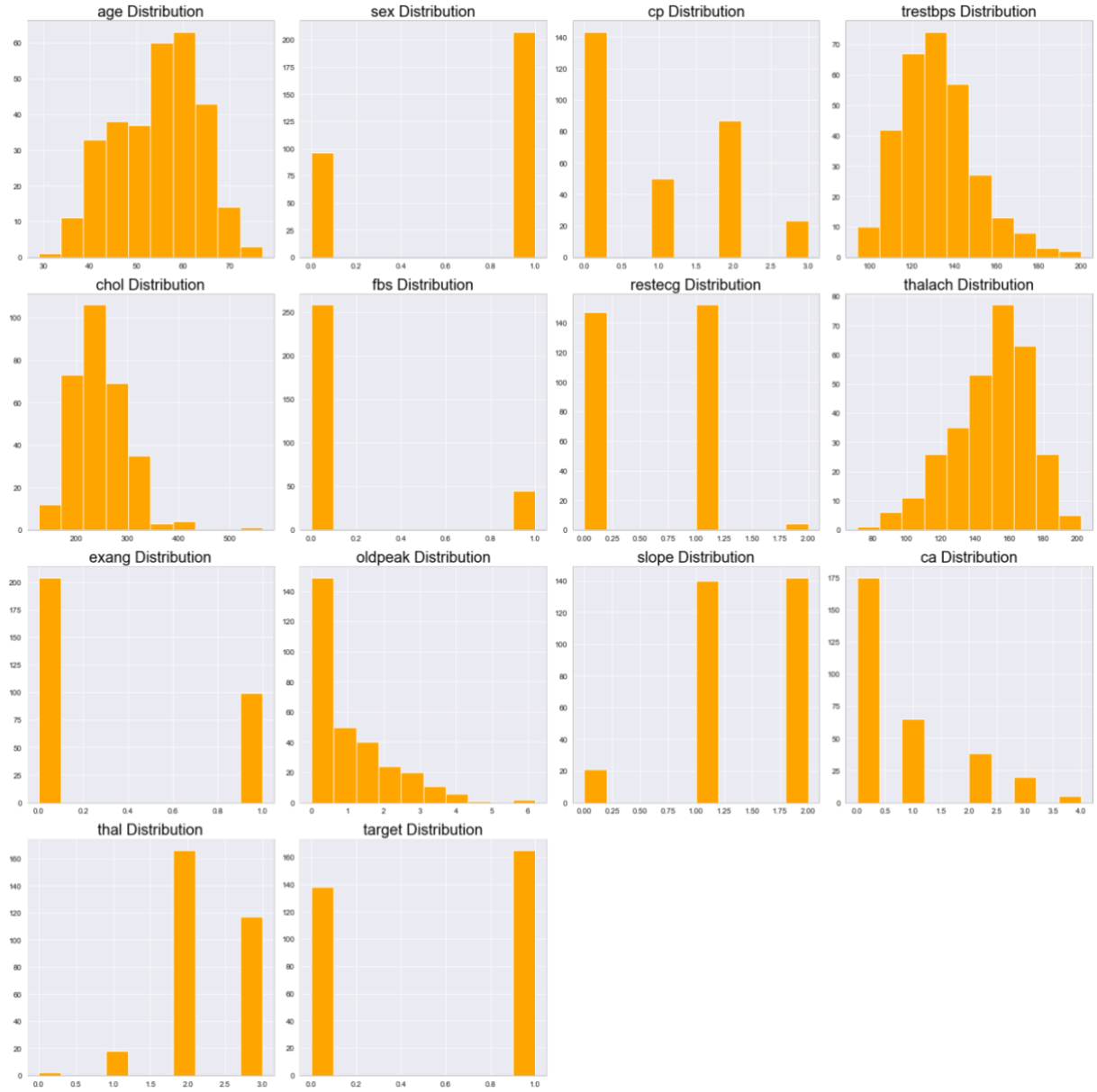

Figure 5.4: Histogram Plot of Data

The function pd.crosstab() helped build this visualization throwing light upon the fact that people who do not have fasting blood sugar have higher chances of being sick than people who have fasting blood sugar.

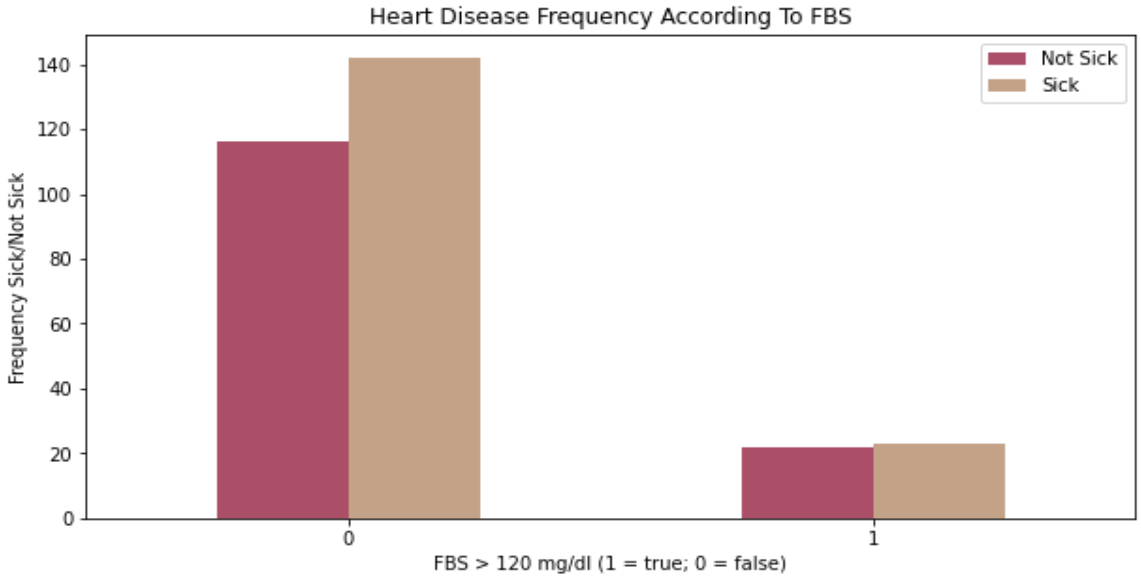

Figure 5.5: Heart Disease Frequency by FBS

It is also used to show a relationship between chest pain types and the target variable. There are four levels in chest pain 0, 1, 2 and 3. Level 0 chest pain type has majority of healthy individuals, whereas other levels have comparatively higher number of sick individuals. Level 2 chest pain type has high number of sick people.

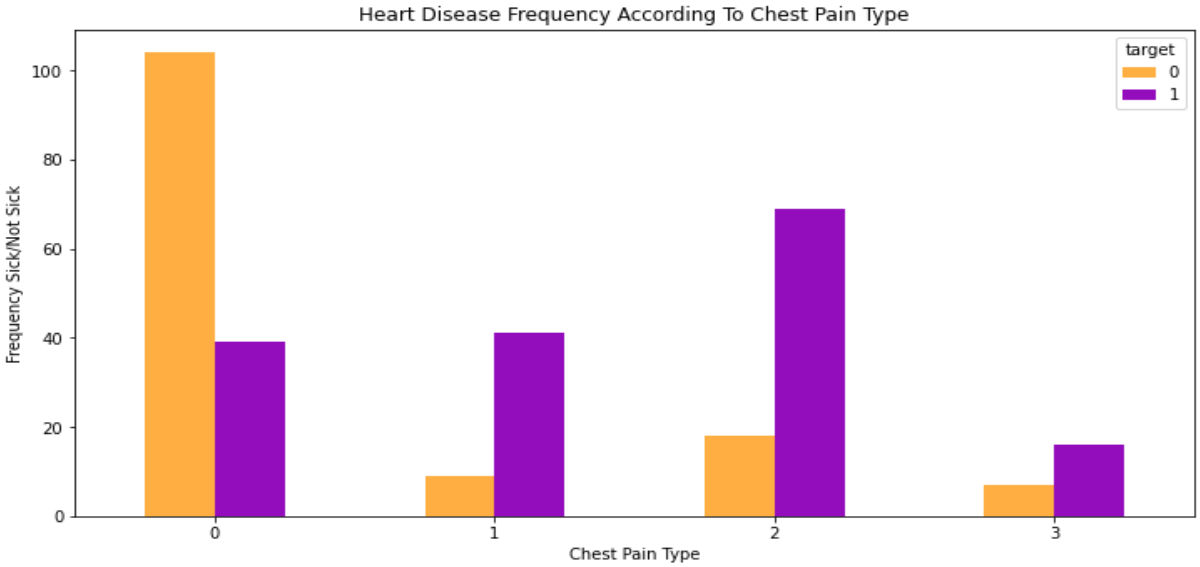

Figure 5.6: Heart Disease Frequency by Chest Pain Type

The function unique() values are used from the target variable column for y-axis. For x-axis, value_count() is used to get the total count for each class. X_ticks was used to name 0, 1. It is vital to check if the dataset is balanced or not. If not, then the whole model will be of no benefit, as it can create a bias towards a particular outcome even if the accuracy is found to be very high. Here, the classes are almost balanced and therefore the data can be further used for preprocessing.

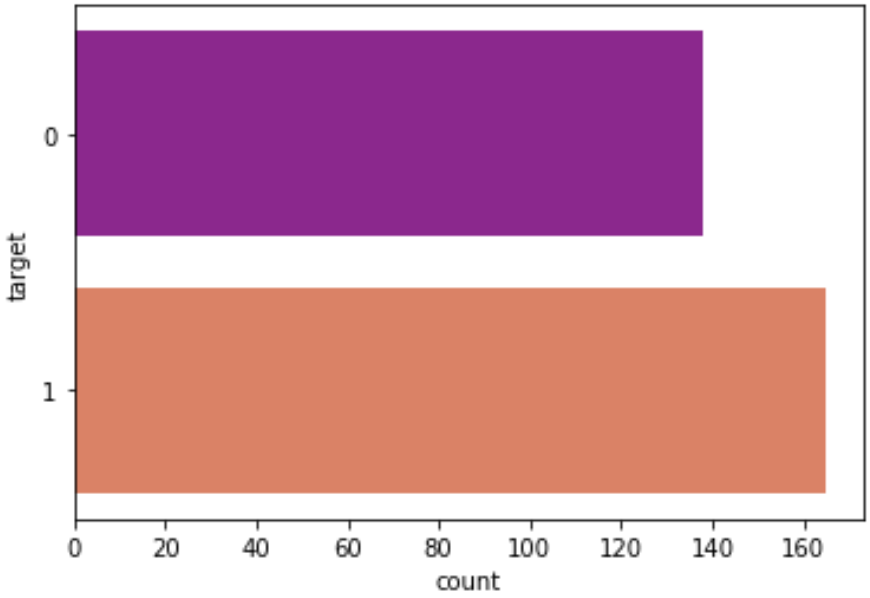

Figure 5.7: Target Count Plot

6. Data Mining Tasks

6.1. Data Preprocessing

303 records are used, and in data preprocessing phase to check if there are any null values. Features 'Chest Pain (CP)', 'Thalassemia (Thal)' have two and four null values respectively. Either imputation of these variables is performed, or columns are dropped. Along with other unwanted features, it is decided to drop the columns.

In the histogram plots there are a lot of categorical features, get dummies() is used to break them down into dummy variables i.e. 1s and 0s.

As the histogram also tells that there is need for scaling, it is achieved by scaling data and performing normalization on the dataset. Normalization was performed using formula

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

The next step was to split the data into training and testing datasets. Using train_test_split(), the data is split into 80% training and 20% testing data. Next, 5 algorithms were chosen to build and evaluate the model by varying the parameters to select the best performing model.

```
In [15]: y = df.target.values
         x_data = df.drop(['target'], axis = 1)


         x = (x_data - np.min(x_data)) / (np.max(x_data) - np.min(x_data)).values #normalising data
         x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.2,random_state=0)

         x_train = x_train.T
         y_train = y_train.T
         x_test = x_test.T
         y_test = y_test.T
```

Figure 6.1: Train Test Split Code

7. Data Mining Models and Methods

7.1. K Nearest Neighbors (KNN)

K Neighbors Classifier checks classes of the k nearest neighbors and classifies the new records as to what the majority of k nearest neighbors are. The number of neighbors can be varied, and in this model, it is varied from 1 to 20.

```
In [17]: score_list = []
         for i in range(1,21):
             knn2 = KNeighborsClassifier(n_neighbors = i)
             knn2.fit(x_train.T, y_train.T)
             score_list.append(knn2.score(x_test.T, y_test.T))

         plt.plot(range(1,21), score_list)

         plt.xticks(np.arange(1,21,1))

         plt.xlabel("K value")
         plt.ylabel("KNN Score")

         plt.show()

         print("KNN Score Max {}%".format((max(score_list)) * 100))
```

Figure 7.1: KNN Model Code

The above code plots a line graph of number of neighbors versus the KNN score. The highest accuracy achieved for this model is 88.52% for neighbors 3, 7, and 8.

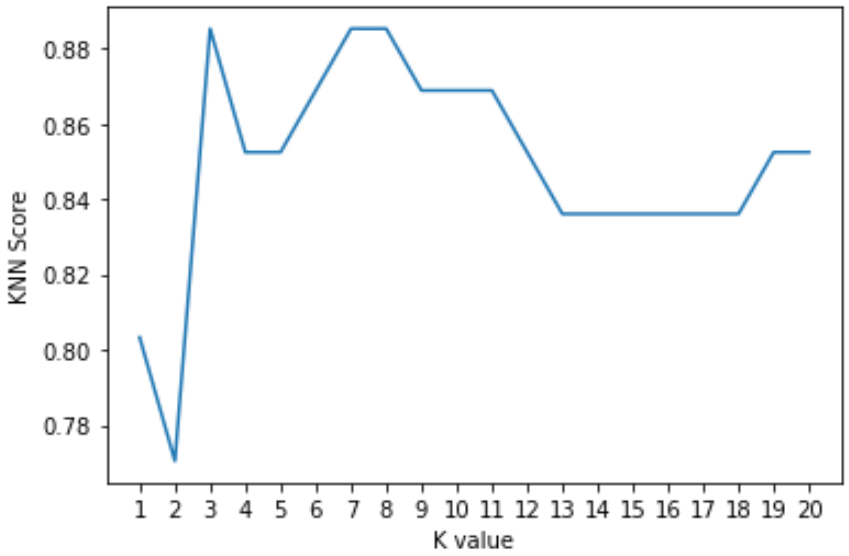

Figure 7.2: KNN Score Level

7.2. Support Vector Machine (SVM)

SVM maps data points to higher dimension space to classify the data points, even when the data is overlapping or is not linearly separable. A separator between the two classes is found, then the data are transformed in such a way that the separator could be drawn as a hyperplane. It forms a hyperplane that can separate the classes as much as possible by adjusting the distance between the data points and the hyperplane. There are many kernels based on which the hyperplane is decided, following are the four kernels tried:-

- Linear
- Poly
- RBF
- Sigmoid

The accuracies for all the four kernels were found, and the best accuracy was for RBF (Radial Basis Function) at 88.52%.

```
In [18]: from sklearn.svm import SVC

         svm = SVC(random_state = 5)
         svm.fit(x_train.T, y_train.T)

Out[18]: SVC(random_state=5)

In [19]: svc_scores = []
         kernels = ['linear', 'poly', 'rbf', 'sigmoid']
         for i in range(len(kernels)):
             svc_classifier = SVC(kernel = kernels[i])
             svc_classifier.fit(x_train.T, y_train.T)
             svc_scores.append(svc_classifier.score(x_test.T,y_test.T))

In [20]: svc_scores

Out[20]: [0.8524590163934426, 0.819672131147541, 0.8852459016393442, 0.7868852459016393]
```

Figure 7.3: SVM Model Code

7.3. Gaussian Naïve Bayes

This algorithm is generally used when the data is continuous and has a lot of different input variables. Here an accuracy of 86.89% is achieved.

```
In [22]: from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(x_train.T, y_train.T)
print("Test Accuracy of Gaussian Naive Bayes: {}%".format(gnb.score(x_test.T,y_test.T) * 100))
```

Figure 7.4: Naïve Bayes Model Code

7.4. Decision Tree

Decision tree classifier creates decision trees and then it assigns the class values to each data point. The max_leaf_nodes are varied from 1 to 30, because maximum number of features can be varied when creating the model. There are 30 features in the dataset after data preprocessing.

```
score_list_DT = []
for i in range(2,31):
    dt2 = DecisionTreeClassifier(max_leaf_nodes = i)
    dt2.fit(x_train.T, y_train.T)
    score_list_DT.append(dt2.score(x_test.T, y_test.T))

plt.plot(range(2,31), score_list_DT)

plt.xticks(np.arange(2,31,1))

plt.xlabel("Leaf")
plt.ylabel("Score")

plt.show()

print("Decision Tree Max Score: {}%".format((max(score_list_DT)) * 100))
```

Figure 7.5: Decision Tree Model Code

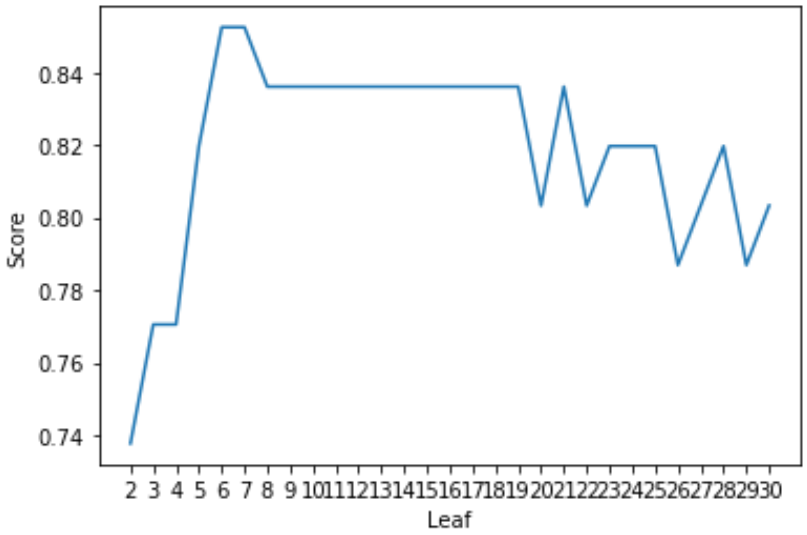

Figure 7.6: Decision Tree Accuracy graph

After building the model, line graph was created to check scores for each node and is found out that the best accuracy of 85.24% for maximum features being selected is equal to 7 and 8.

7.5. Random Forest

In Random Forest Classifier, it creates a random forest of trees where features can be randomly selected from all the available parameters present for each tree. In the model, n_estimators that is number of trees set to 1000 and max_leaf_nodes are varied. As a routine, a line graph is created of scores and number of max_leaf_nodes. The highest accuracy of 90.16% is achieved in Random Forest Classifier.

```
In [26]: score_list_RF = []
         for i in range(2,25):
             rf2 = RandomForestClassifier(n_estimators = 1000, random_state = 2, max_leaf_nodes = i)
             rf2.fit(x_train.T, y_train.T)
             score_list_RF.append(rf2.score(x_test.T, y_test.T))

         plt.plot(range(2,25), score_list_RF)
         plt.xticks(np.arange(2,25,1))
         plt.xlabel("Leaf")
         plt.ylabel("Score")
         plt.show()
         print("RF Score Max {}%".format((max(score_list_RF)) * 100))
```

Figure 7.7: Random Forest Model Code

8. Performance Evaluation

The Heart Disease Data was first imported in Jupyter Notebook, which is then cleaned and visualized to understand which factors show better relationship in finding out if a person has a heart disease or not.

After calculating accuracy using different models available, the ‘Random Forest’ model was selected for further evaluation as it has the best accuracy among all and has several advantages over other models.

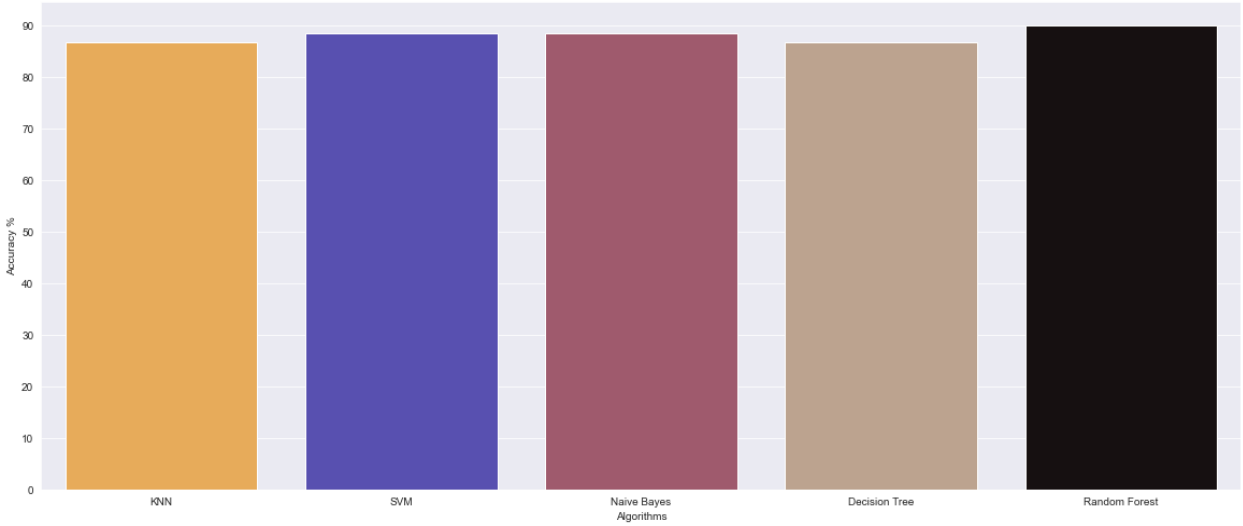

Figure 8.1: Model Accuracy Comparison

8.1. Evaluations

The Random Forest model is then evaluated twice to see if we get better accuracy by increasing the number of trees or 'max_leaf_nodes'. This gave a better fitted model and a better accuracy for the data model.

1. Confusion Matrix of the dataset without changing the maximum leaf node number

Calculating the error percentage, accuracy, sensitivity, specificity. For model execution, first model is fit without changing the 'max_leaf_nodes', which gives the accuracy for model as 88.52 %.

Table 8.1: 1st Evaluation Confusion Matrix

	0 (Predicted Negative)	**1 (Predicted Positive)**
0 (Actual Negative)	23	4
1 (Actual Positive)	3	31

Error Percentage = (3 + 4)/61 = **11.48 %**

Accuracy = (23 + 31)/61 = **88.52 %**

Sensitivity = 31/(31 + 3) = **91.17 %**

Specificity = 23/(23 + 4) = **85.18 %**

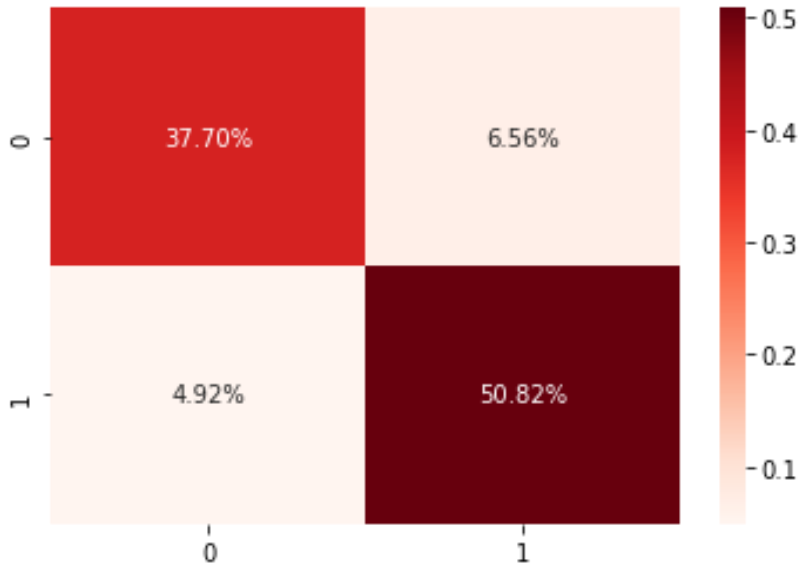

Figure 8.2: 1st Evaluation Confusion Matrix

2. Confusion Matrix of the dataset after looping for the maximum leaf node number

Now calculate the error percentage, accuracy, sensitivity, specificity. The model is fitted and re-evaluated again for better accuracy using a loop for 'max_lead_nodes' values between (2,25) which in turn performed well, to give highest accuracy of 90.16 %.

Table 8.2: Re-evaluation Confusion Matrix

	0 (Predicted Negative)	**1 (Predicted Positive)**
0 (Actual Negative)	23	4
1 (Actual Positive)	2	32

Error Percentage = (2 + 4)/61 = **9.83 %**

Accuracy = (23 + 31)/61 = **90.16 %**

Sensitivity = 32/(32 + 2) = **94.11 %**

Specificity = 23/(23 + 4) = **85.18 %**

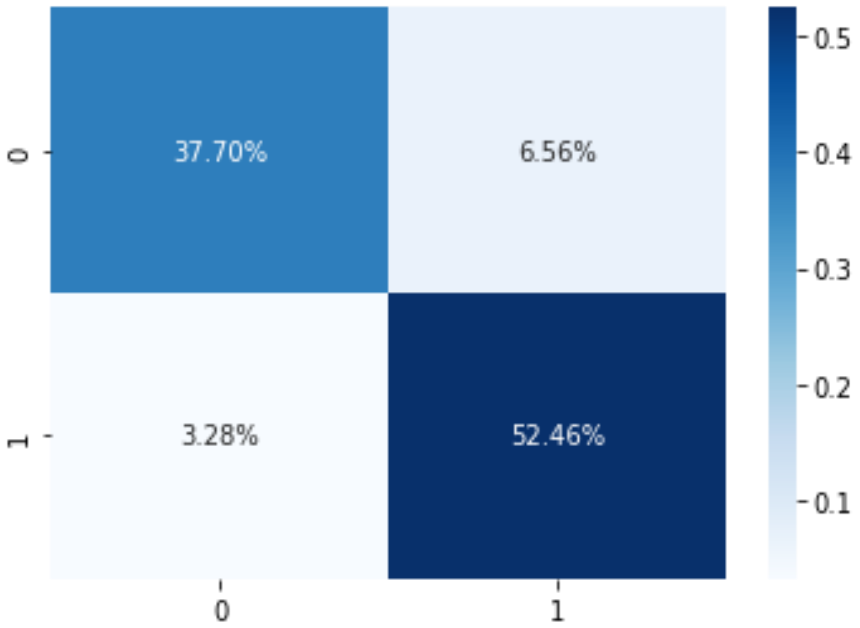

Figure 8.3: Re-evaluation Confusion Matrix

9. Project Results

By comparing the two models , parameters, and their confusion matrices, It can be seen that the model is predicting most of the true cases correctly i.e. 52.46 %. The most important part about this model being selected is that it predicts only 3.28% of patients have no heart disease when in fact they have heart disease. This percentage value is comparatively very low and having a low mis-classification value is very crucial in any clinical setting where there is a need for diagnosis of any disease.

9.1. Classification Report

After examining other metrics like precision, recall, and F1 score for re-evaluated model even more detailed insight into the model's performance can be obtained.

```
In [22]: from sklearn.metrics import classification_report
         print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.92      0.85      0.88        27
           1       0.89      0.94      0.91        34

    accuracy                           0.90        61
   macro avg       0.90      0.90      0.90        61
weighted avg       0.90      0.90      0.90        61
```

Figure 9.1: Classification Report Code

1. Precision
   - Precision is the ratio of number of correctly identified members of a class to the times the model predicted that class.
   - In this case, the precision score is the number of instances that are correctly identified as having heart disease divided by the number of instances of times the model predicts having heart diseases correctly or incorrectly
   - Precision in this model is 88.89%

2. Recall
   - Recall is the ratio members of a class that the classifier identified correctly to the total number of members in that class
   - Here, this would be the number of times correctly identifying having heart disease
   - Precision is 94.11% for this model
3. F1 score
   - F1 score combines precision and recall into one metric
   - F1 is a quick way to analyze if the model is good at identifying members of a class, or if it is taking shortcuts (e.g. just identifying all members as majority class)
   - An F1 score is considered perfect when it's 1, while the model is a total failure when it's 0. In this model, F1 score is 0.91.

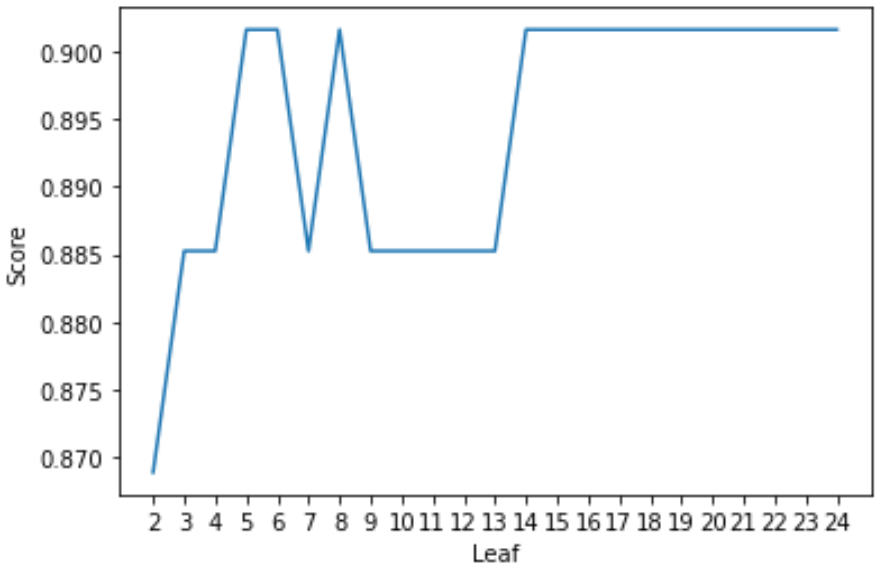

Figure 9.2: Accuracy graph for max_leaf_nodes

The re-evaluated random forest classifier has an accuracy score of 90.16% on the test data. It looks like a great accuracy score, but it must be kept in mind that it's not the best measure of classifier performance when the classes are not balanced. The important question is do the model perform equally well for each class? Are there any pairs of classes that were hard to distinguish? To answer all these questions, we make a confusion matrix.

9.2. ROC

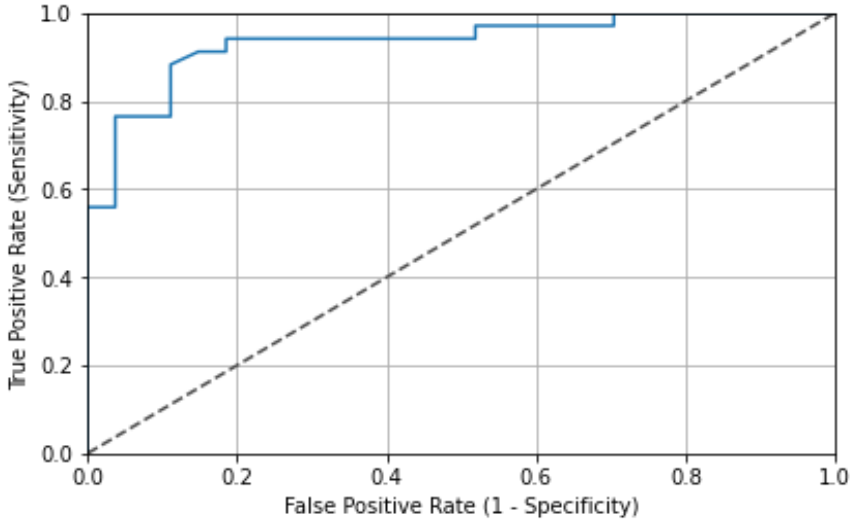

Figure 9.3: ROC Graph

An ROC curve was then plotted, which is a plot of 'sensitivity' vs '1 – specificity'. The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve. The higher the AUC, the better the performance of the model is at distinguishing between the positive and negative classes. Here the model AUC is 0.93, which shows that the model is performing well.

10. Project Outcome & Future Scope

The results of this project show exceptional performance with high accuracy. The models created can be used as a reference in various organizations for research and improvement of several heart disease prediction models. This project helps analyze predictor variables and find patterns which show direct relation to the prediction of heart diseases. Since the target class is balanced the performance of the model on this dataset can be trusted. This project has also helped us as professional students to learn and implement the Machine Learning algorithms in real life application projects. As famous as it is the Random Forest model continued to perform best for this project.

10.1. Future Scope

The project outcome for this project can be improved in several ways.

1. A more detailed data can be used to predict the heart disease which may have hidden patterns with new predictor variables, which may not have been identified in this model.
2. Higher prediction accuracy can be achieved by optimizing the model parameters. The model can also be optimized to reduce the computation time since the huge amount of data may need to be predicted.
3. A different approach for optimization can be used, like cost optimization, risk factor calculation. This project topic is cost-sensitive and here each class can be assigned a misclassification cost. The model can then be used to reduce misclassification costs instead of focusing on reducing error percentage. Risk factor can help prioritize the potential patients to take an early action against any type of disease.