Sri Lanka Institute of Information Technology

Predicting Presence of Heart Diseases Using Random Forest Classifier Algorithm

Machine Learning

(SE-4060)

Assignment

Report

IT16178700 – Rajapakshe D.I.K

Date of submission
27/04/2020

TABLE OF CONTENTS

TABLE OF CONTENTS 2

1. INTRODUCTION 3

2. METHODOLOGY 5

   2.1. Dataset 5

   2.2. Attribute Information 5

   2.3. Data Preprocessing 6

   2.4. Random Forest Classifier 7

   2.5. Decision Tree Classifier 8

3. IMPLEMENTATION 9

   3.1. Understanding data 9

   3.2. Generating Test data 9

   3.3. Random Forest Classifier and Predictions 10

   3.4. Accuracy for test data 10

   3.5. Importance of the features 11

   3.6. Errors made by classifier 12

4. EVALUATION 13

   4.1. Result 13

   4.2. Possible Limitation 13

   4.3. Future Work 13

REFERENCE 14

APPENDIX – A 15

APPENDIX – B 24

APPENDIX – C 25

1. INTRODUCTION

Heart disease refers to various conditions that affect a person's heart. Cardiovascular diseases include diseases of the blood vessels and circulatory systems, as well as coronary artery disease and arrhythmias. You have congenital heart disease (congenital heart defects) and others. Heart disease is a problem and distortions in the heart.

The phrase "heart disease" is often used interchangeably with the word "heart disease". Cardiovascular disease is usually associated with narrow or blocked blood vessels that can lead to heart attacks, chest pain (angina) or arrhythmias. Other heart conditions, primarily those affecting a person's heart muscles, valves, or heart rhythm, are considered as types of heart disease.

Many terms of heart disease can be prevented or treated with healthy lifestyle preferences. Major risk factors for heart disease include age, depression, diabetes, alcoholism, family history of heart disease, high blood pressure, high cholesterol, high stress, obesity, poor nutrition or eating habits, sedentary lifestyle, smoking.

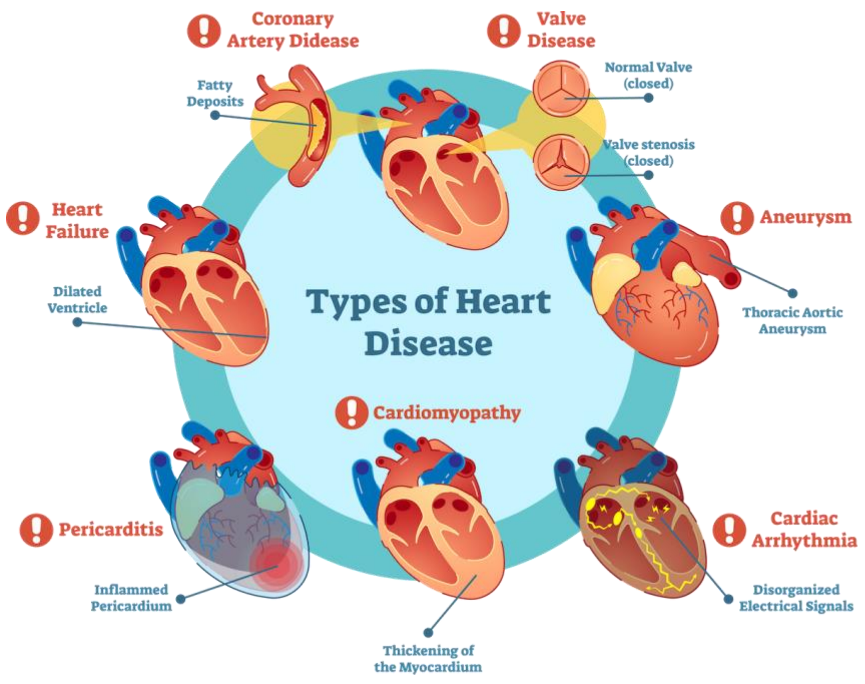


Figure 1.1: Type of Heart Disease [2].

Machine learning is more widely used in many fields around the world than it is today. [1] The health sector is also no exception to e-commerce and many other commercial applications. Predictability is one aspect of our theme of learning this machine: predicting cardiovascular disease by preparing the data set and patient data needed to predict the likelihood of heart disease.

Our topic can play an essential role in learning the mechanics of predicting cardiovascular disease by preparing the patient's dataset and the patient data needed to predict the likelihood of heart disease. To that end, we can use predictive algorithms. There we predict the use of forest random classification and decision tree classification algorithm. The dataset was retrieved from Kaggle. The results of heart disease have shown accurate results.

2. METHODOLOGY

2.1. Dataset

This dataset starts from 1988 and includes four databases: Cleveland, Hungary, Switzerland and Long Beach VA. It contains 76 attributes, including the expected attribute, but 14 of all published experiments mention the use of a subset. [3]. The "target" field is the presence of heart disease in the patient. Its full value is 0 = no disease and 1 = disease [4].

The link of the dataset has shown below,

https://www.kaggle.com/johnsmith88/heart-disease-dataset

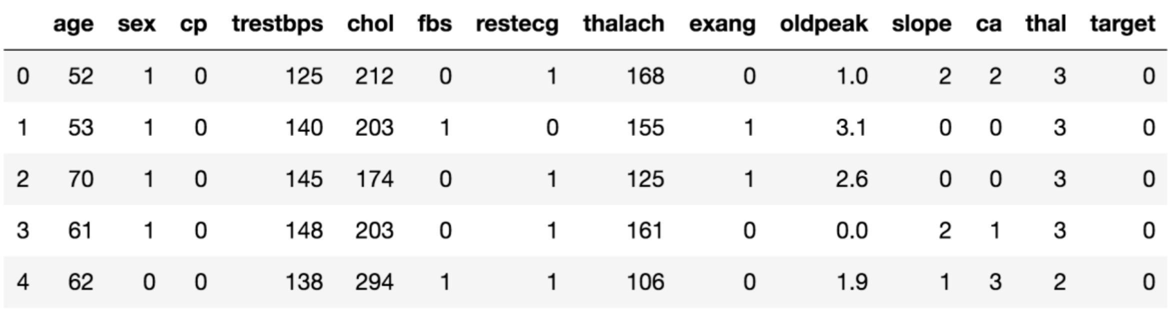
2.2. Attribute Information

Table 2.2.1: Attributes context.

Attributes	Domain
age	age in time
sex	(1=male, 0=female)
cp	chest pain type
trestbps	resting blood pressure (in mm Hg on admission to the hospital)
chol	serum cholesterol in mg/dl
fbs	(fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
restecg	resting electrocardiographic results
thalach	maximum heart rate achieved
exang	exercise-induced angina (1 = yes; 0 = no)
old peak	ST depression induced by exercise relative to rest
slope	the slope of the peak exercise ST segment
ca	number of major vessels (0-3) colored by fluoroscopy
thal	1 = normal; 2 = fixed defect; 3 = reversible defect
target	1=have disease or 0= have not to disease

2.3. Data Preprocessing

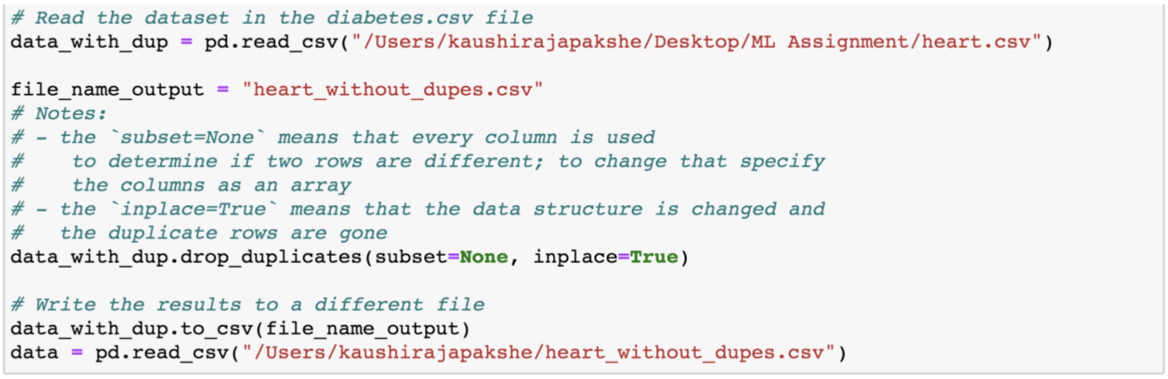
This dataset contains 702 duplicate rows when removed only left with the original UCI dataset of 303. The below figure shows given heart disease data set which had duplication with the features.



	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

Figure 2.3.1: Original heart disease data set before preprocess.

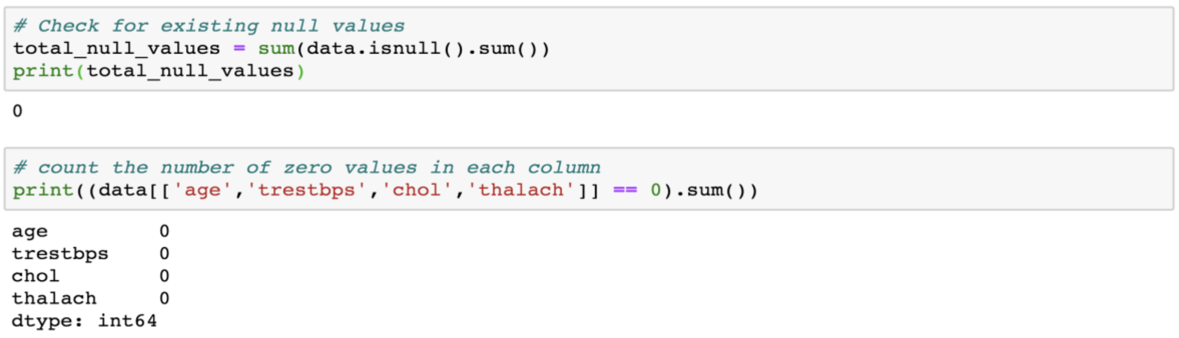
Removing duplicates and save without duplicates value set to a new csv file for the prediction.



```
# Read the dataset in the diabetes.csv file
data_with_dup = pd.read_csv("/Users/kaushirajapakshe/Desktop/ML Assignment/heart.csv")

file_name_output = "heart_without_dupes.csv"
# Notes:
# - the `subset=None` means that every column is used
#     to determine if two rows are different; to change that specify
#     the columns as an array
# - the `inplace=True` means that the data structure is changed and
#    the duplicate rows are gone
data_with_dup.drop_duplicates(subset=None, inplace=True)

# Write the results to a different file
data_with_dup.to_csv(file_name_output)
data = pd.read_csv("/Users/kaushirajapakshe/heart_without_dupes.csv")
```

Figure 2.3.2: Code for removing duplicates and save the new dataset.

This dataset does not have any null values, as well as no 0 values for age, trestbps, chol, thalach fields. The below figure shows values for the dataset with coding.



```
# Check for existing null values
total_null_values = sum(data.isnull().sum())
print(total_null_values)
```

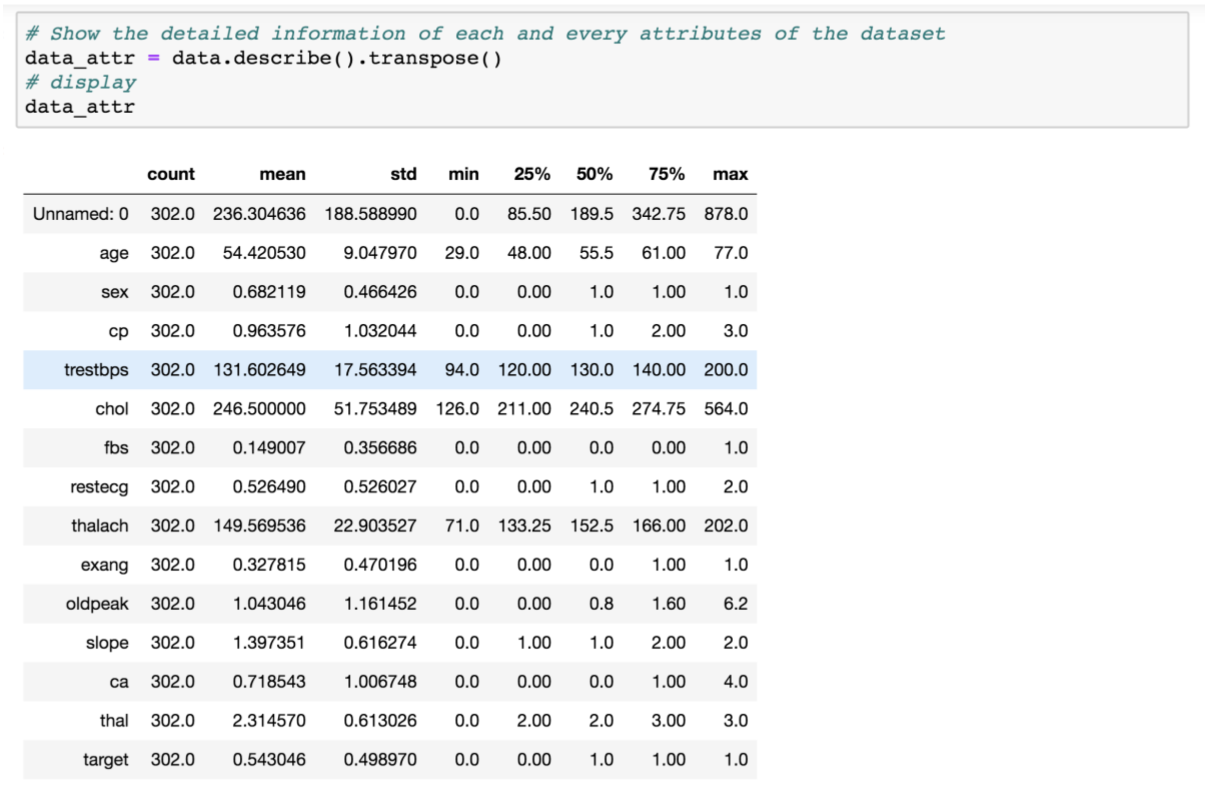
```
0
```

```
# count the number of zero values in each column
print((data[['age','trestbps','chol','thalach']] == 0).sum())
```

```
age         0
trestbps    0
chol        0
thalach     0
dtype: int64
```

Figure 2.3.3: Dataset null and 0 values for the column.

According to Figure 2.3.4 will be shown the preprocessed data in heart disease.



```
# Show the detailed information of each and every attributes of the dataset
data_attr = data.describe().transpose()
# display
data_attr
```

	count	mean	std	min	25%	50%	75%	max
Unnamed: 0	302.0	236.304636	188.588990	0.0	85.50	189.5	342.75	878.0
age	302.0	54.420530	9.047970	29.0	48.00	55.5	61.00	77.0
sex	302.0	0.682119	0.466426	0.0	0.00	1.0	1.00	1.0
cp	302.0	0.963576	1.032044	0.0	0.00	1.0	2.00	3.0
trestbps	302.0	131.602649	17.563394	94.0	120.00	130.0	140.00	200.0
chol	302.0	246.500000	51.753489	126.0	211.00	240.5	274.75	564.0
fbs	302.0	0.149007	0.356686	0.0	0.00	0.0	0.00	1.0
restecg	302.0	0.526490	0.526027	0.0	0.00	1.0	1.00	2.0
thalach	302.0	149.569536	22.903527	71.0	133.25	152.5	166.00	202.0
exang	302.0	0.327815	0.470196	0.0	0.00	0.0	1.00	1.0
oldpeak	302.0	1.043046	1.161452	0.0	0.00	0.8	1.60	6.2
slope	302.0	1.397351	0.616274	0.0	1.00	1.0	2.00	2.0
ca	302.0	0.718543	1.006748	0.0	0.00	0.0	1.00	4.0
thal	302.0	2.314570	0.613026	0.0	2.00	2.0	3.00	3.0
target	302.0	0.543046	0.498970	0.0	0.00	1.0	1.00	1.0

Figure 2.3.4: Preprocessed Data.

2.4. Random Forest Classifier

Random forest algorithm is one of the most popular and powerful supervised machine learning algorithms, capable of performing regression and classification tasks. Therefore, the accuracy of modeling multiple decision trees for the forest is high.

In random forests, the court grows a few trees instead of a single tree model to classify a new object based on the attributes that each tree classifies. The results are normalized to different trees in the regression case.

Advantages of Random Forest Classifier:

- It can be used for both classification and regression tasks.
- Won't over fit the model.
- Handle the missing values and maintains accuracy for missing data.
- Power to handle large data set with higher dimensionality.

2.5. Decision Tree Classifier

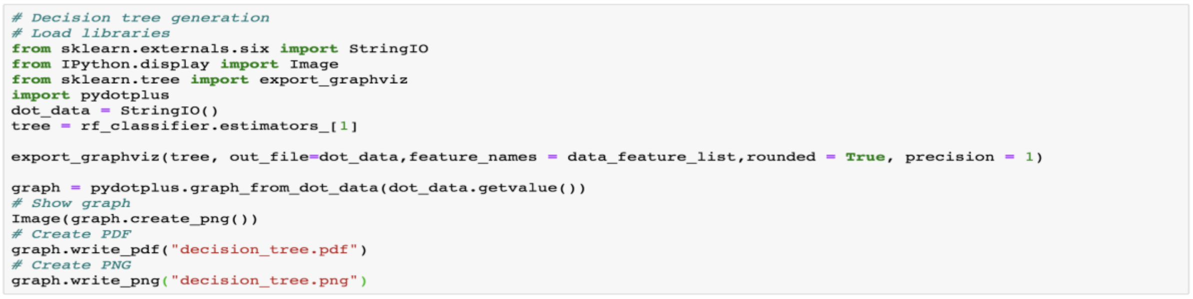
The decision tree is a supervised learning algorithm. It has been used primarily for classification problems. It has many similarities in life and seems to have been influenced by a large area of machine learning, covering both taxonomic and regressive trees.

The decision tree is a flow log structure that checks the function of each internal node. Each branch represents the result of a test and each sheet or technical node has a class label. The top node of a tree is the main node. Uncertainty analysis of a critical tree can be used to visualize and make clear decisions, and the decision-making process uses a name-like decision tree model.

All the nodes, except the leaf nodes, those are parts in there:

- Questions about a dataset based on each attribute of a feature's attributes. There is a true or false answer to every question. Based on the answer of the main node to the query, a data point moves down the tree.
- Gini- The nasty Gini of the node. When we come down from the tree, the average weight of the fire is reduced.
- Samples- Number of node considerations.
- Value- Number of samples in each class. For example, the upper node has two class 0 samples and 1 class 4 samples.
- Class- Multiple classification of nodes. For leaf nodes, this is the forecast for all the samples in the node.

Appendix- B as sown in to narrow down the problem for heart disease using decision tree classification. According to figure 2.5.1 display how to implement a decision tree algorithm for heart disease.



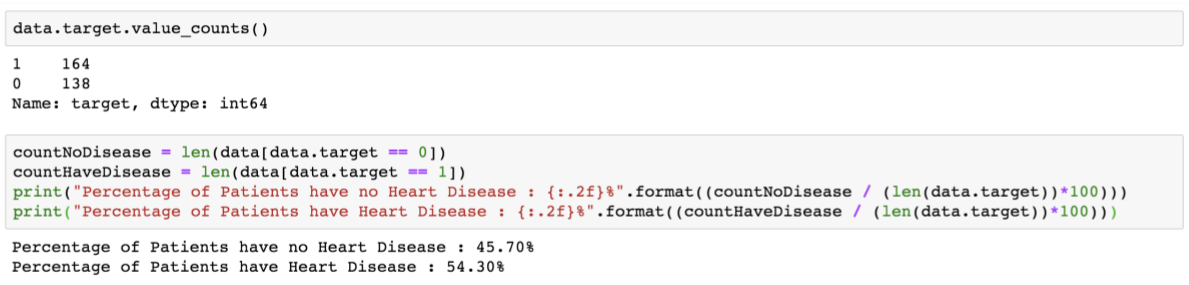
```
# Decision tree generation
# Load libraries
from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
dot_data = StringIO()
tree = rf_classifier.estimators_[1]

export_graphviz(tree, out_file=dot_data,feature_names = data_feature_list,rounded = True, precision = 1)

graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
# Show graph
Image(graph.create_png())
# Create PDF
graph.write_pdf("decision_tree.pdf")
# Create PNG
graph.write_png("decision_tree.png")
```

Figure 2.5.1: Decision Tree Implementation

3. IMPLEMENTATION

3.1. Understanding data

As the first step for implementation, we will get an understanding of the data set in Figure 3.1.1. For that we generate some graphs using Matplotlib library as Appendix – C and show histogram in Appendix – D.

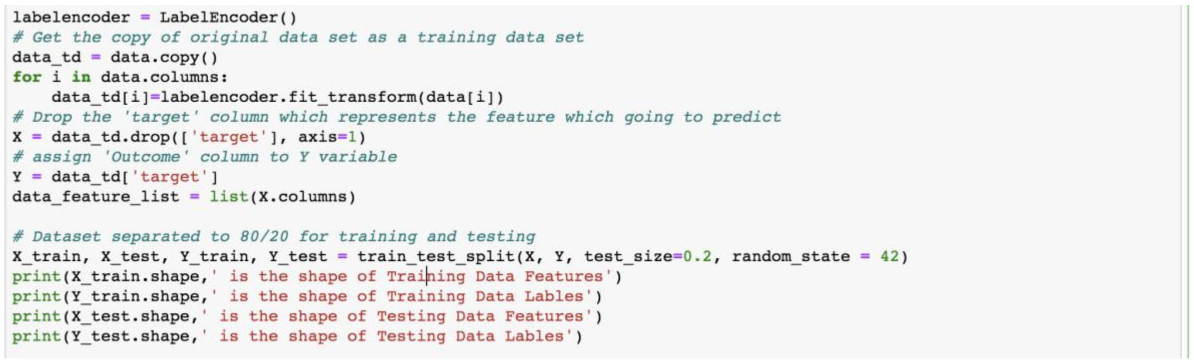


```
data.target.value_counts()

1    164
0    138
Name: target, dtype: int64

countNoDisease = len(data[data.target == 0])
countHaveDisease = len(data[data.target == 1])
print("Percentage of Patients have no Heart Disease : {:.2f}%".format((countNoDisease / (len(data.target))*100)))
print("Percentage of Patients have Heart Disease : {:.2f}%".format((countHaveDisease / (len(data.target))*100)))

Percentage of Patients have no Heart Disease : 45.70%
Percentage of Patients have Heart Disease : 54.30%
```

Figure 3.1.1: Heart Disease target count.

3.2. Generating Test data

To generate a test data, we get a copy of the original data set as a training data set and delete the 'target' attribute which represents the feature which is predict and assign the 'target' column to Y variable. Figure 3.2.1 will show the generating test data and training for the test heart disease dataset.

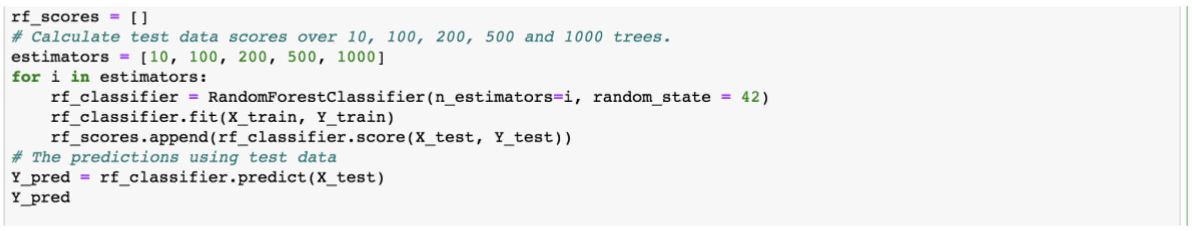


```
labelencoder = LabelEncoder()
# Get the copy of original data set as a training data set
data_td = data.copy()
for i in data.columns:
    data_td[i]=labelencoder.fit_transform(data[i])
# Drop the 'target' column which represents the feature which going to predict
X = data_td.drop(['target'], axis=1)
# assign 'Outcome' column to Y variable
Y = data_td['target']
data_feature_list = list(X.columns)

# Dataset separated to 80/20 for training and testing
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state = 42)
print(X_train.shape,' is the shape of Training Data Features')
print(Y_train.shape,' is the shape of Training Data Lables')
print(X_test.shape,' is the shape of Testing Data Features')
print(Y_test.shape,' is the shape of Testing Data Lables')
```

Figure 3.2.1: Generate test data and training.

3.3. Random Forest Classifier and Predictions

Calculate test data scores over 10, 100, 200, 500, and 1000 trees for predictions using test data. Figure 3.3.1 will show the predictions for the test data.

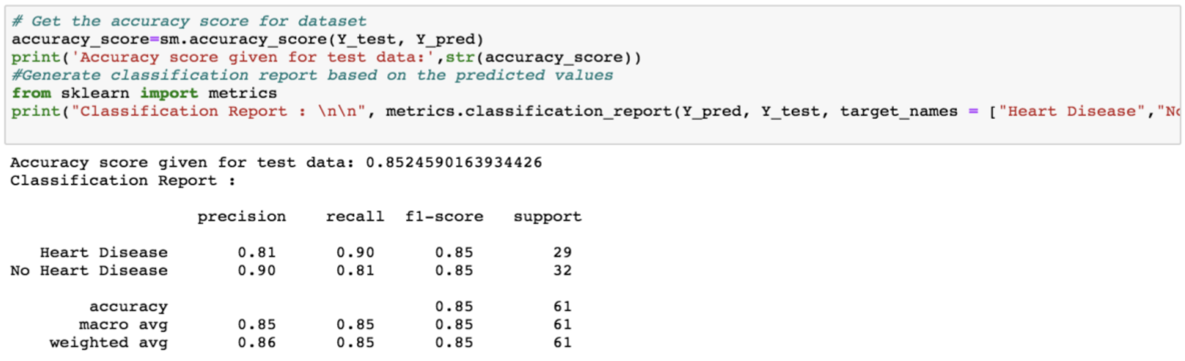


```
rf_scores = []
# Calculate test data scores over 10, 100, 200, 500 and 1000 trees.
estimators = [10, 100, 200, 500, 1000]
for i in estimators:
    rf_classifier = RandomForestClassifier(n_estimators=i, random_state = 42)
    rf_classifier.fit(X_train, Y_train)
    rf_scores.append(rf_classifier.score(X_test, Y_test))
# The predictions using test data
Y_pred = rf_classifier.predict(X_test)
Y_pred
```

Figure 3.3.1: Random forest classifier.

3.4. Accuracy for test data

Inputting label prediction data and label test data, we get the accuracy score for the dataset. For that generate a classification report based on the predicted values. Figure 3.4.1 will show the displaying the result of the accuracy for test data and generating a classification report.



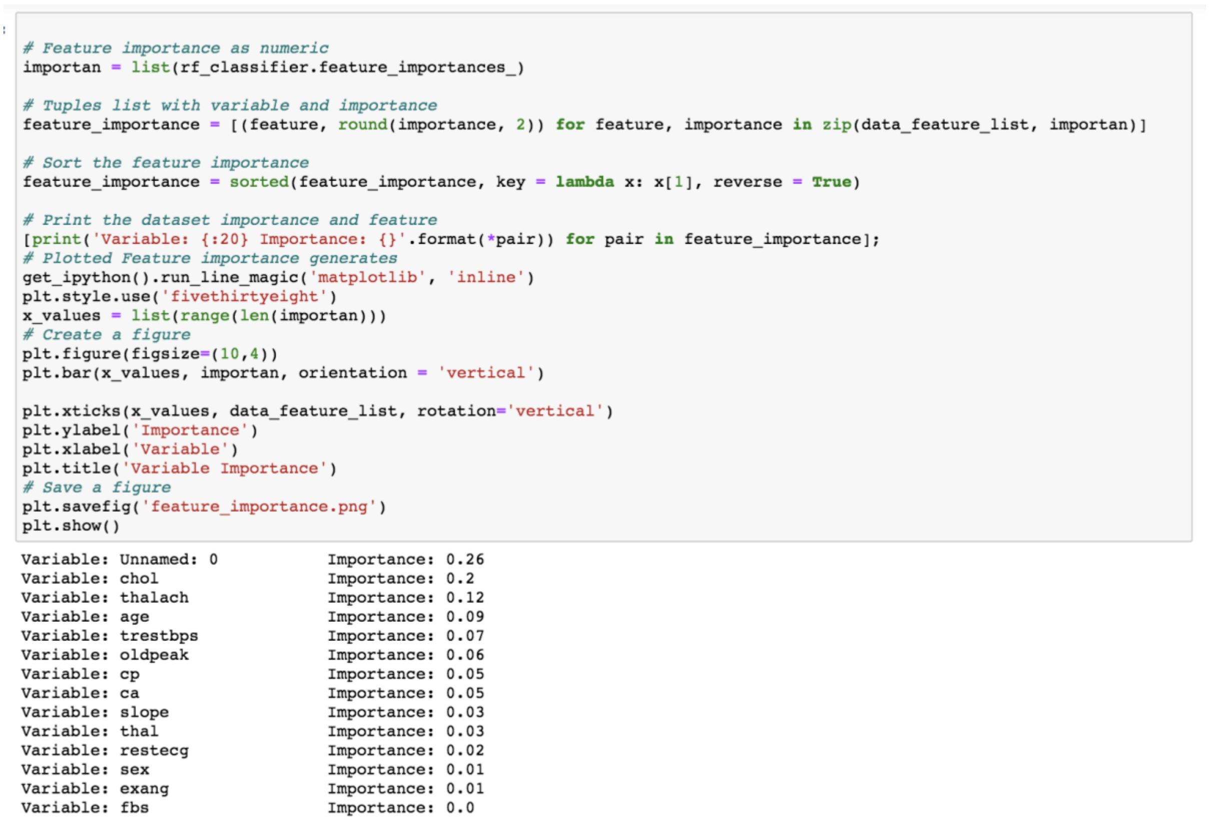
```
# Get the accuracy score for dataset
accuracy_score=sm.accuracy_score(Y_test, Y_pred)
print('Accuracy score given for test data:',str(accuracy_score))
#Generate classification report based on the predicted values
from sklearn import metrics
print("Classification Report : \n\n", metrics.classification_report(Y_pred, Y_test, target_names = ["Heart Disease","N
```

```
Accuracy score given for test data: 0.8524590163934426
Classification Report :

                  precision    recall  f1-score   support

   Heart Disease       0.81      0.90      0.85        29
No Heart Disease       0.90      0.81      0.85        32

        accuracy                           0.85        61
       macro avg       0.85      0.85      0.85        61
    weighted avg       0.86      0.85      0.85        61
```

Figure 3.4.1: Accuracy for test data.

3.5. Importance of the features

To get the importance of features, we inputting the list of random Forest Classifiers to predict data. Figure 3.5.1 will show the displaying importance of the features.



```
# Feature importance as numeric
importan = list(rf_classifier.feature_importances_)

# Tuples list with variable and importance
feature_importance = [(feature, round(importance, 2)) for feature, importance in zip(data_feature_list, importan)]

# Sort the feature importance
feature_importance = sorted(feature_importance, key = lambda x: x[1], reverse = True)

# Print the dataset importance and feature
[print('Variable: {:20} Importance: {}'.format(*pair)) for pair in feature_importance];
# Plotted Feature importance generates
get_ipython().run_line_magic('matplotlib', 'inline')
plt.style.use('fivethirtyeight')
x_values = list(range(len(importan)))
# Create a figure
plt.figure(figsize=(10,4))
plt.bar(x_values, importan, orientation = 'vertical')

plt.xticks(x_values, data_feature_list, rotation='vertical')
plt.ylabel('Importance')
plt.xlabel('Variable')
plt.title('Variable Importance')
# Save a figure
plt.savefig('feature_importance.png')
plt.show()
```

```
Variable: Unnamed: 0           Importance: 0.26
Variable: chol                 Importance: 0.2
Variable: thalach              Importance: 0.12
Variable: age                  Importance: 0.09
Variable: trestbps             Importance: 0.07
Variable: oldpeak              Importance: 0.06
Variable: cp                   Importance: 0.05
Variable: ca                   Importance: 0.05
Variable: slope                Importance: 0.03
Variable: thal                 Importance: 0.03
Variable: restecg              Importance: 0.02
Variable: sex                  Importance: 0.01
Variable: exang                Importance: 0.01
Variable: fbs                  Importance: 0.0
```

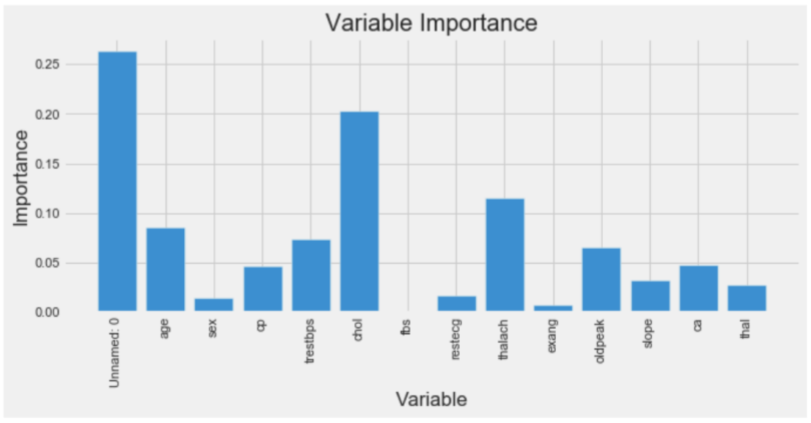


Figure 3.5.1: Importance of the features.

3.6. Errors made by classifier

To decide which algorithm works best with our data wasted by dividing the data into training and testing sets. Creating a confusion Matrix for every algorithm, we can summarize how our implemented algorithm performed on the testing data. The rows in the confusion matrix, correspond to what the machine learning algorithm predicted and the column corresponds to the known truth since there are only two categories to choose from heart disease or does not have heart disease.

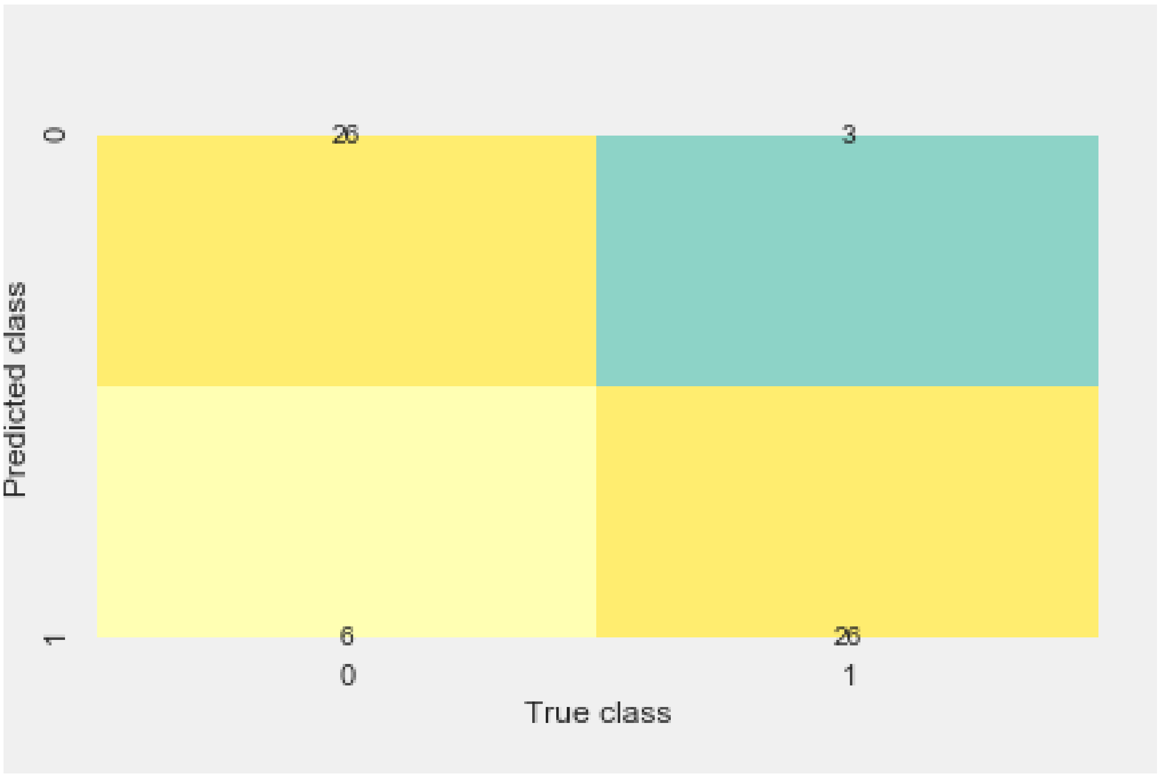


Figure 3.6.1: Confusion Metrix.

4. EVALUATION

4.1. Result

After implementing and training dataset to random forest classifier algorithm shows as accuracy score given for test data is 0.8360655737704918. Giving accuracy is 83.60%.

4.2. Possible Limitation

Although this model can provide good accuracy predictions, the problems raised by the accuracy of several decision nodes can be considered a limitation of this implemented machine learning model.

4.3. Future Work

Future works may further address the issues not giving a 99% accuracy, so want to apply to the complexity and best machine learning algorithm and implementation and want to test with more data to and want to get predictions with the test data.

REFERENCE

[1]. Towards Data Science, “Predicting presence of Heart Diseases using Machine Learning”, 2019, [online]. Available: https://towardsdatascience.com/predicting-presence-of-heart-diseases-using-machine-learning-36f00f3edb2c/, [Accessed: April 17, 2020].

[2]. UDMI, “Types of Cardiovascular Disease”, [online]. Available: https://www.udmi.net/cardiovascular-disease-risk/, [Accessed: April 23, 2020].

[3]. UDMI, “Heart Disease Data Set”, [online]. Available: https://archive.ics.uci.edu/ml/datasets/heart+Disease, [Accessed: April 24, 2020].

[4]. Kaggle, “Heart Disease Dataset”, 2019, [online]. Available: https://www.kaggle.com/johnsmith88/heart-disease-dataset, [Accessed: April 15, 2020].

APPENDIX – A

```
# IT16178700 - D.I.K. Rajapakshe
# Heart Diseases using Random Forest Classifier Algorithm
# Importing all required libraries to work with the dataset
import numpy as np # linear algebra
# Matplotlib library all graph types for dataset
import matplotlib.pyplot as pylt
import sklearn.metrics as skm
# sklearn.metrics includes score functions, performance metrics and pairwise metrics
 and distance computations
import pandas as pds
# visualization of statistical data
import seaborn as sns
# Python Interface to Graphviz’s Dot language.
import pydotplus
# LabelEncoder use to converting the labels into the numeric form so as to convert it
into the machine-readable form
from sklearn.preprocessing import LabelEncoder
# Split arrays or matrices into random train and test subsets
from sklearn.model_selection import train_test_split
# create charts using pyplot, and color them with
from matplotlib.cm import rainbow
from sklearn import metrics
from sklearn.metrics import confusion_matrix
# Impliment the Random Forest Classifier Algorithm
from sklearn.ensemble import RandomForestClassifier
# Decision tree generation
# Load libraries
from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
# Implementation of the Ada Boost Classifier
from sklearn.ensemble import AdaBoostClassifier

%matplotlib inline
# Read the dataset heart.csv from the file location
data_with_dup = pds.read_csv("/Users/kaushirajapakshe/Desktop/ML Assignment/he
art.csv")
# Display the data with column
data_dup_top = data_with_dup.head()
# display
data_dup_top
file_name_output = "heart_without_dupes.csv"
# removing duplicates value sets
data_with_dup.drop_duplicates(subset=None, inplace=True)

# Write the results to a different file
data_with_dup.to_csv(file_name_output)
data = pds.read_csv("/Users/kaushirajapakshe/heart_without_dupes.csv")
# Display the data with column
data_top = data.head()
# display
data_top
# Show the detailed information of each and every attributes of the dataset
data_attr = data.describe().transpose()
# display
data_attr
# Print null values
total_null_values = sum(data.isnull().sum())
```

```
print(total_null_values)
print(data.target.value_counts())
# count the number of zero values in each column
print((data[['age','trestbps','chol','thalach']] == 0).sum())

sns.countplot(x="target", data=data, palette="icefire")
# Save a figure
pylt.savefig('target.png')
pylt.show()
count_no_diseases = len(data[data.target == 0])
count_have_diseases = len(data[data.target == 1])
print("Percentage of Patients have no Heart Disease : {:.2f}%".format((count_no_dise
ases / (len(data.target))*100)))
print("Percentage of Patients have Heart Disease : {:.2f}%".format((count_have_dise
ases / (len(data.target))*100)))
sns.countplot(x='sex', data=data, palette="cubehelix")
pylt.xlabel("Sex (0 = female, 1= male)")
# Save a figure
pylt.savefig('sex.png')
pylt.show()
# convert strings to indicators (only numeric for correlation)
dummies = pds.get_dummies(data, columns=['sex', 'target'])
corr = dummies.corr()
pylt.figure(figsize=(10,8))
sns.heatmap(corr,
          vmin=-1,
          cmap='Set3',
          annot=True,
          mask=np.tri(corr.shape[0], k=0))
# Save a figure
```

```
pylt.savefig('sex_and_target_heatmap.png')
pylt.show()
pds.crosstab(data.sex,data.target).plot(kind="bar",figsize=(15,6),color=['#ffb3b3','#66
0066' ])
pylt.title('Heart Disease Frequency for Type of Sex')
pylt.xlabel('Sex Types : (0 = Female, 1 = Male)')
pylt.xticks(rotation=0)
pylt.legend(["No Disease", "Disease"])
pylt.ylabel('Frequency')
# Save a figure
pylt.savefig('sex_and_target_crosstab.png')
pylt.show()
data.groupby('target').mean()
pds.crosstab(data.age,data.target).plot(kind="bar",figsize=(26,9))
pylt.title('Frequency of heart disease for Ages')
pylt.xlabel('Age')
pylt.ylabel('Frequency')
# Save a figure
pylt.savefig('heart_disease_and_ages.png')
pylt.show()
# Create a figure
pylt.figure(figsize=(12,8))
pylt.scatter(x=data.age[data.target==1], y=data.thalach[(data.target==1)], c="red")
pylt.scatter(x=data.age[data.target==0], y=data.thalach[(data.target==0)])
pylt.legend(["Disease", "No Disease"])
pylt.xlabel("Age")
pylt.ylabel("Maximum Heart Rate")
# Save a figure
pylt.savefig('age_target_and_thalach_target_scatter.png')
pylt.show()
```

```
pds.crosstab(data.slope,data.target).plot(kind="bar",figsize=(15,6),color=['#ccf5ff','#b
3ffb3' ])
pylt.title('Heart Disease Frequency for Slope')
pylt.xlabel('The Slope of The Peak Exercise ST Segment ')
pylt.xticks(rotation = 0)
pylt.ylabel('Frequency')
# Save a figure
pylt.savefig('slop_and_target_crosstab.png')
pylt.show()
pds.crosstab(data.fbs,data.target).plot(kind="bar",figsize=(15,6),color=['#ff99e6','#cc
ccff' ])
pylt.title('Heart Disease Frequency According To Fasting Blood Suger')
pylt.xlabel('Fasting Blood Sugar > 120 mg/dl (FBS) (1 = true; 0 = false;)')
pylt.xticks(rotation = 0)
pylt.legend(["No Disease", "Disease"])
pylt.ylabel('Frequency of Disease or Not')
# Save a figure
pylt.savefig('fbs_and_target.png')
pylt.show()
pds.crosstab(data.cp,data.target).plot(kind="bar",figsize=(15,6),color=['#740b63', '#0
e878b'])
pylt.title('Heart Disease Frequency According To Chest Pain Type')
pylt.xlabel('Chest Pain Type')
pylt.xticks(rotation = 0)
pylt.legend(["No Disease", "Disease"])
pylt.ylabel('Frequency of Disease or No Disease')
# Save a figure
pylt.savefig('sp_and_target.png')
pylt.show()
labelencoder = LabelEncoder()
```

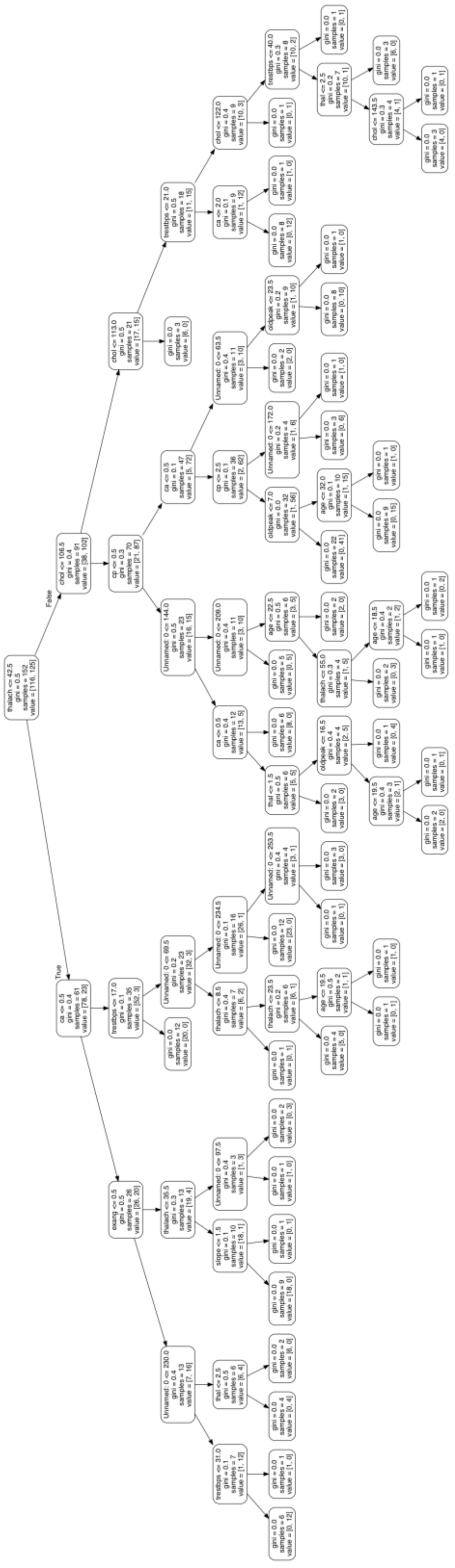
```
# Get the copy of original data set as a training data set
data_td = data.copy()
for i in data.columns:
    data_td[i]=labelencoder.fit_transform(data[i])
# Drop the 'target' column which represents the feature which predict
X = data_td.drop(['target'], axis=1)
# assign 'target' column to Y variable
Y = data_td['target']
data_feature_list = list(X.columns)

# Seperated dataset to 80/20 for testing and training
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state =
42)
print(X_train.shape,' is shape of training data features : ')
print(Y_train.shape,' is shape of training data lables : ')
print(X_test.shape,' is shape of testing data features : ')
print(Y_test.shape,' is shape of testing data lables : ')

rf_score_arr = []
# Calculate test data scores over 10, 100, 200, 500 and 1000 trees.
estimator_arr = [10, 100, 200, 500, 1000]
for i in estimator_arr:
    rndf_classifier = RandomForestClassifier(n_estimators=i, random_state = 42)
    rndf_classifier.fit(X_train, Y_train)
    rf_score_arr.append(rndf_classifier.score(X_test, Y_test))
# The predictions using test data
Y_pred = rndf_classifier.predict(X_test)
Y_pred

clr = rainbow(np.linspace(0, 1, len(estimator_arr)))
```

```
# Create a figure
pylt.figure(figsize=(12,4))
pylt.bar([i for i in range(len(estimator_arr))], rf_score_arr, color = clr, width = 0.8)
for i in range(len(estimator_arr)):
   pylt.text(i, rf_score_arr[i], rf_score_arr[i])
pylt.xticks(ticks = [i for i in range(len(estimator_arr))], labels= [str(estimator) for estim
ator in estimator_arr])
pylt.xlabel('Number of estimators')
pylt.ylabel('Scores')
pylt.title('Random forest classifier scores for different no of estimators')
# Save a figure
pylt.savefig('rfc_scor_over_estimators.png')
pylt.show()

# The predictions using test dataset
Y_pred = rndf_classifier.predict(X_test)
Y_pred
# Get the accuracy score for dataset
accuracy_score=skm.accuracy_score(Y_test, Y_pred)
print('Accuracy score given for test data:',str(accuracy_score))
# Generate classification report based on the predicted valuesprint("Classification Re
port : \n\n", metrics.classification_report(Y_pred, Y_test, target_names = ["Heart Dise
ase","No Heart Disease"]))
sns.set()
get_ipython().run_line_magic('matplotlib', 'inline')

mat = confusion_matrix(Y_test, Y_pred)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False, cmap='Set3')
pylt.xlabel('True class')
pylt.ylabel('Predicted class')
```

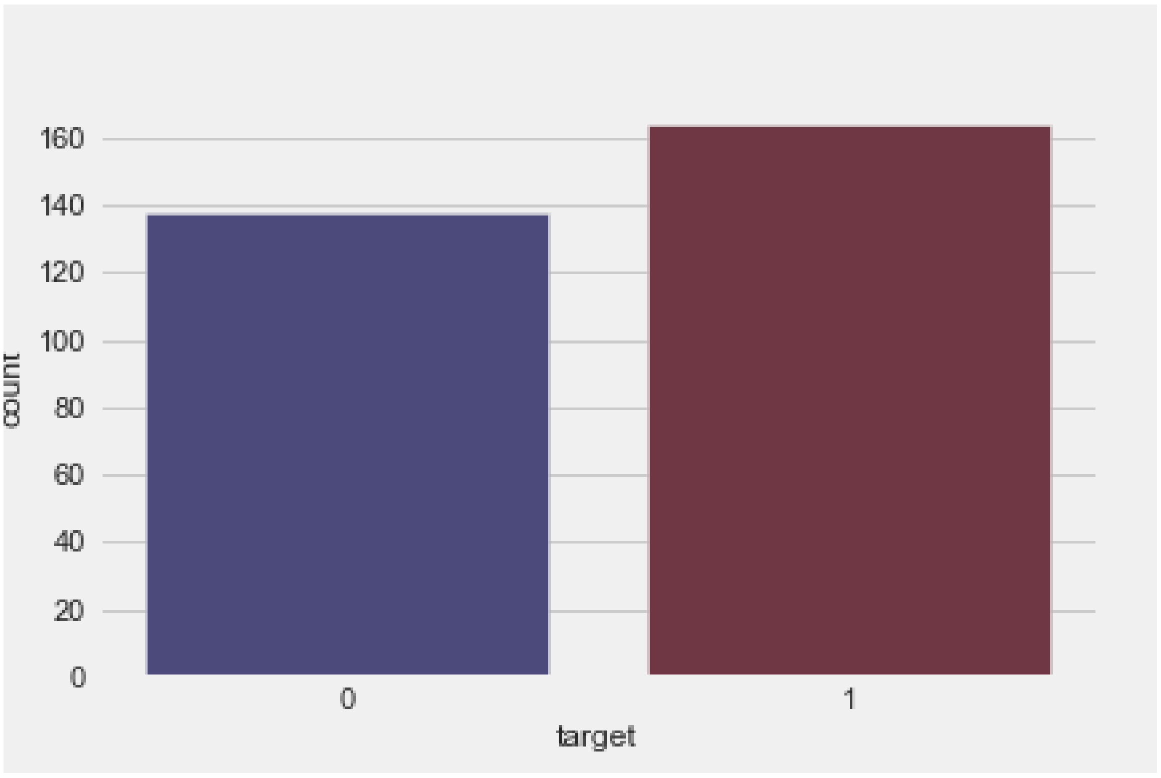
```
# Save a figure
pylt.savefig('metrix_heatmap.png')
pylt.show()
# Feature importance as numeric
important = list(rndf_classifier.feature_importances_)
# Tuples list with variable and importance
feature_importance = [(feature, round(importance, 2)) for feature, importance in zip(d
ata_feature_list, important)]

# Sort the feature importance
feature_importance = sorted(feature_importance, key = lambda x: x[1], reverse = Tru
e)

# Print the dataset importance and feature
[print('Variable: {:20} Importance: {}'.format(*pair)) for pair in feature_importance]
# Plotted Feature importance generates
get_ipython().run_line_magic('matplotlib', 'inline')
pylt.style.use('fivethirtyeight')
x_values = list(range(len(important)))
# Create a figure
pylt.figure(figsize=(10,4))
pylt.bar(x_values, important, orientation = 'vertical')

pylt.xticks(x_values, data_feature_list, rotation='vertical')
pylt.ylabel('Importance of data')
pylt.xlabel('Variable of data')
pylt.title('Variable Importance of data')
# Save a figure
pylt.savefig('feature_importance.png')
pylt.show()
```

```
dot_df = StringIO()
tree_data = rndf_classifier.estimators_[1]

export_graphviz(tree_data, out_file=dot_df,feature_names = data_feature_list,rounde
d = True, precision = 1)

graph = pydotplus.graph_from_dot_data(dot_df.getvalue())
# Show graph
Image(graph.create_png())
# Create PDF
graph.write_pdf("decision_tree.pdf")
# Create PNG
graph.write_png("decision_tree.png")

fig = pylt.figure(figsize = (12,12))
ax = fig.gca()
data.hist(ax=ax)

# Save a figure
pylt.savefig('data_hist.png')
pylt.show()

rndf_classifier = AdaBoostClassifier(n_estimators=1000)
rndf_classifier.fit(X_train, Y_train)
predictions = rndf_classifier.predict(X_test)
confusion_matrix(Y_test, predictions)
```

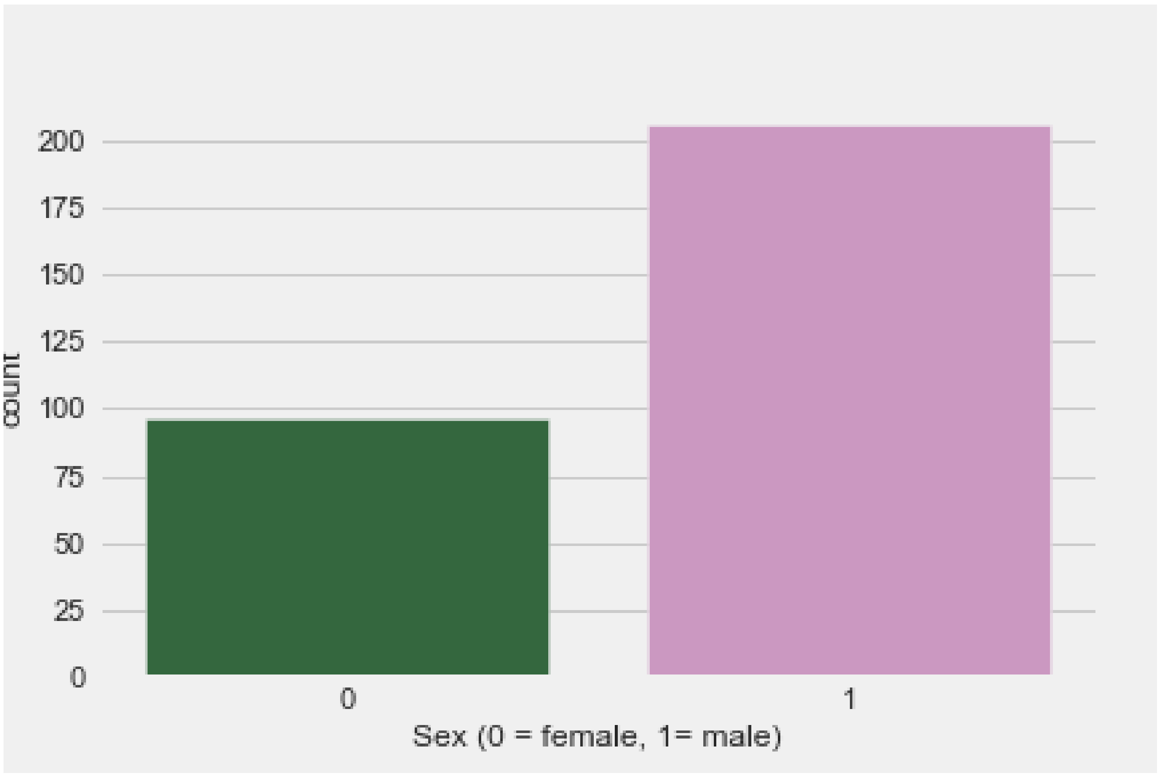
APPENDIX – B



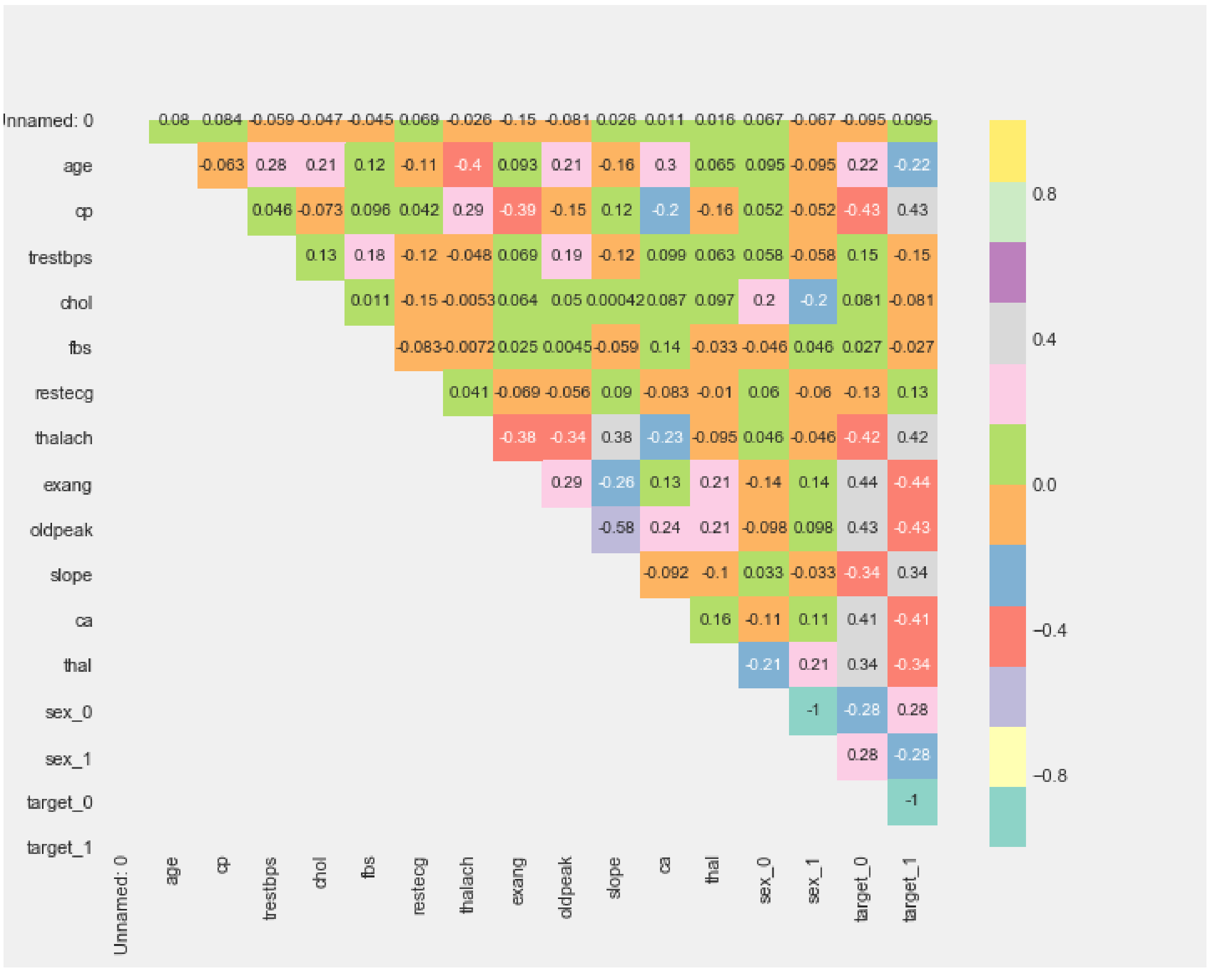
APPENDIX – C

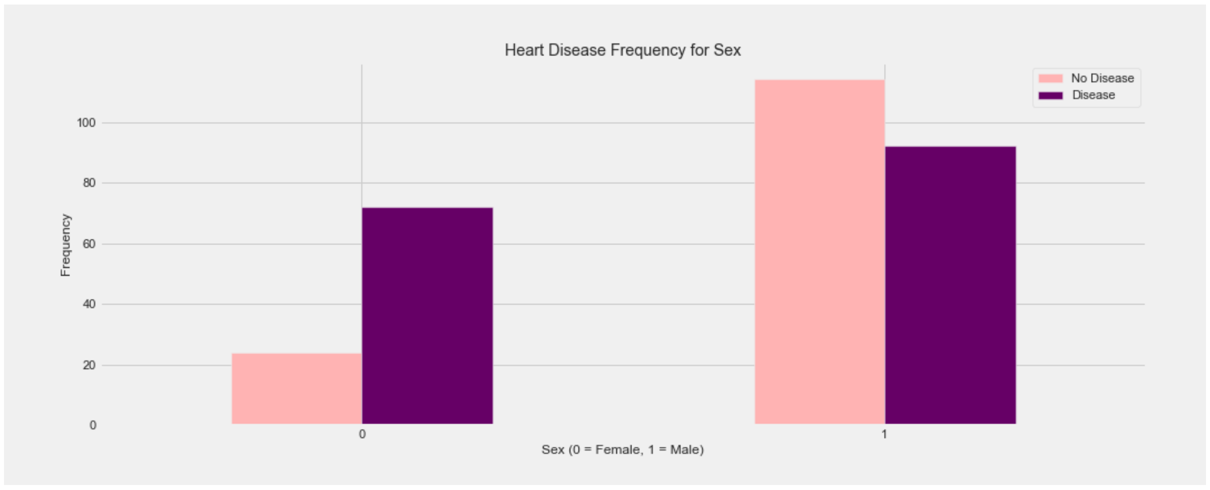


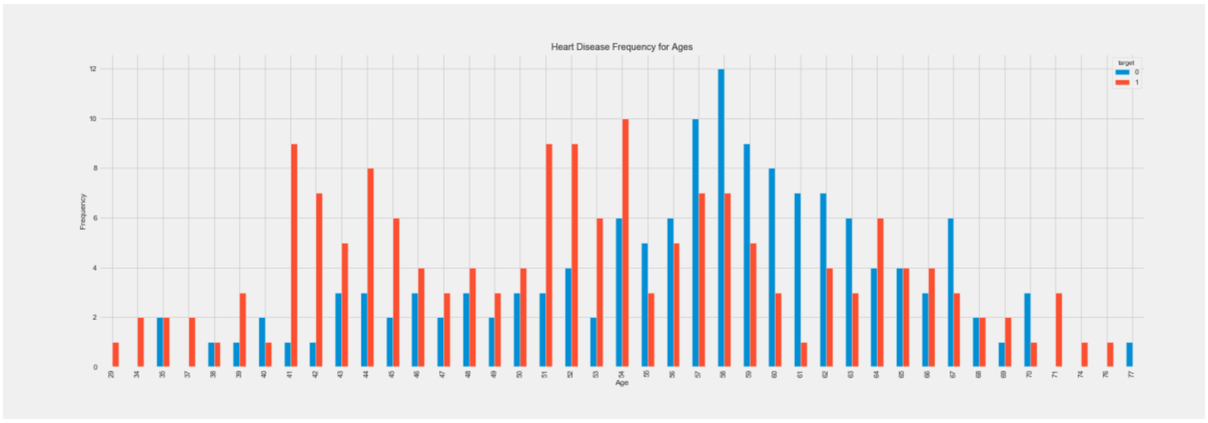
count
160
140
120
100
80
60
40
20
0
0
1
target

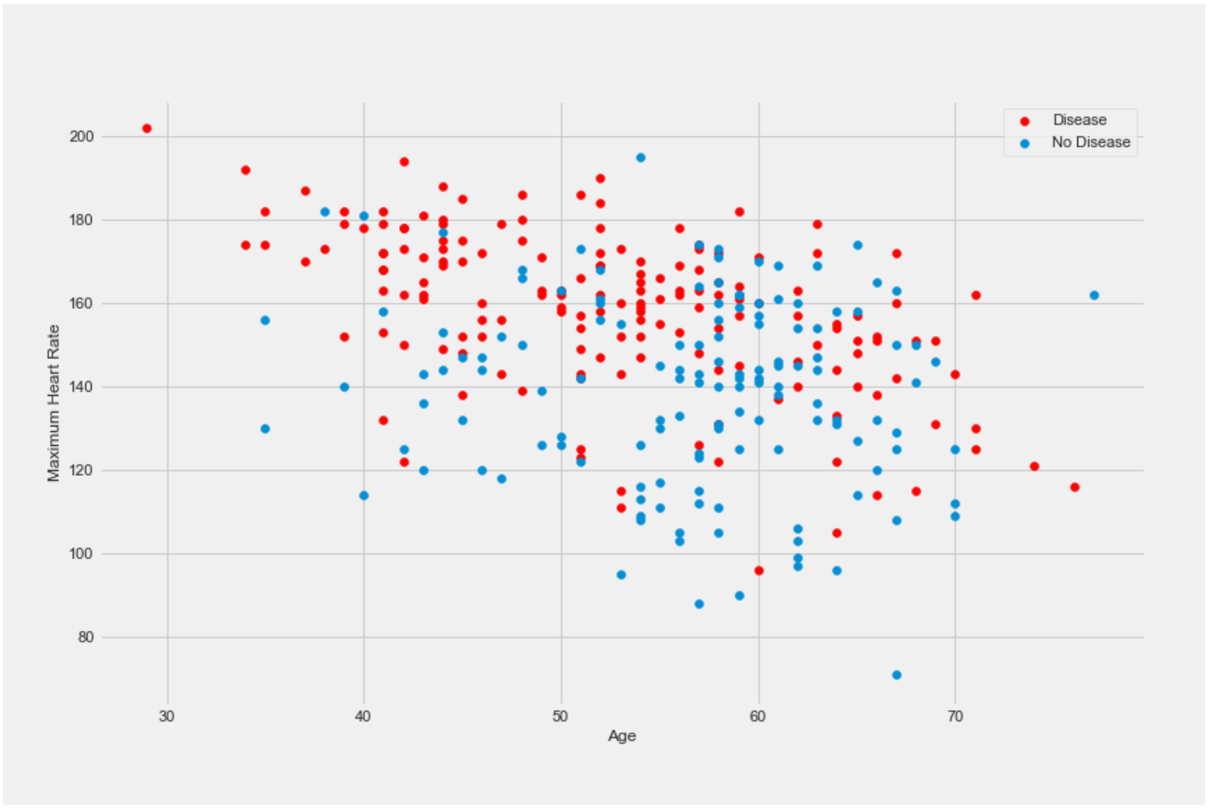


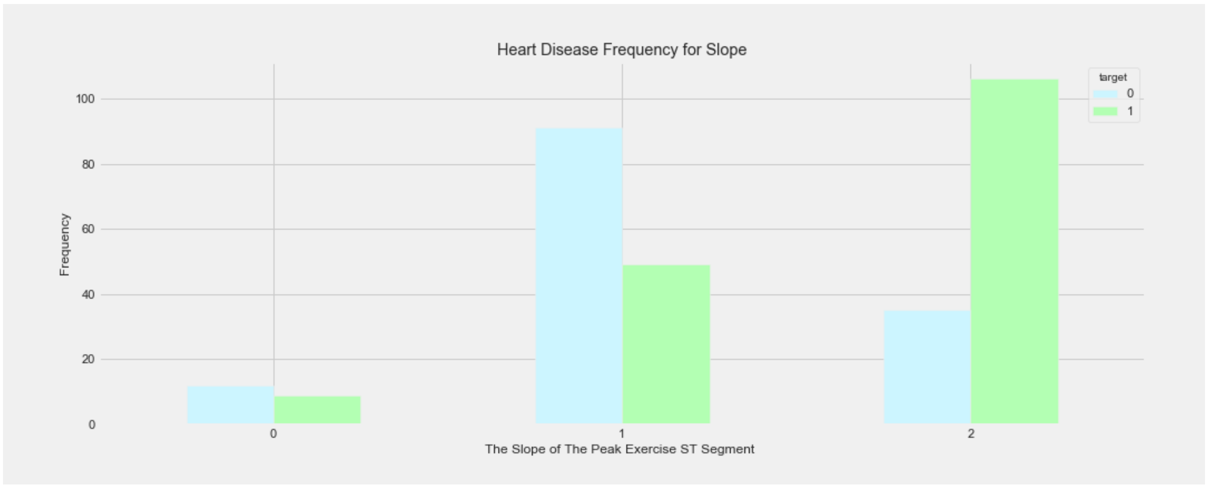
count
200
175
150
125
100
75
50
25
0
0
1
Sex (0 = female, 1= male)

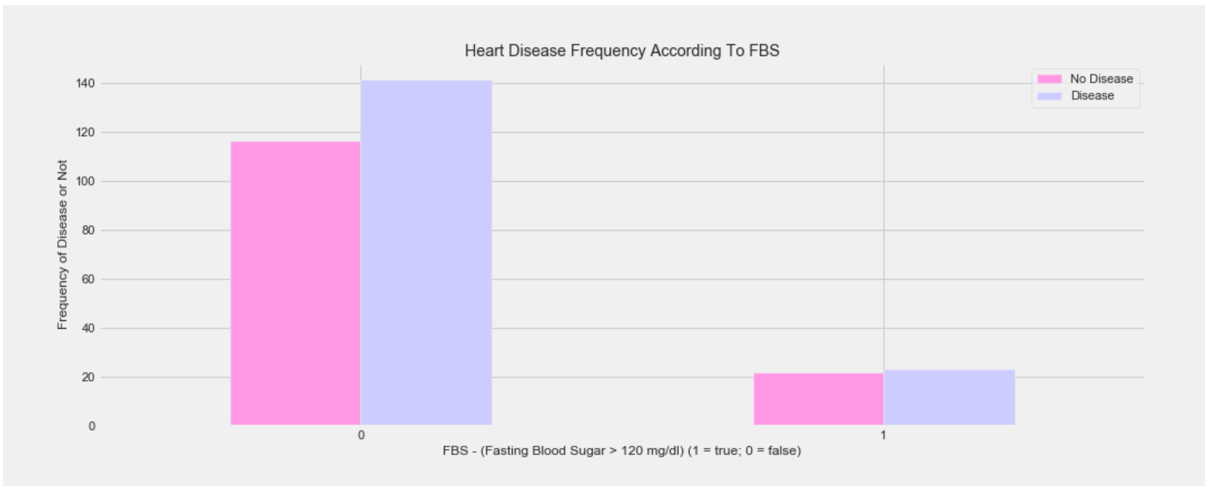

Unnamed: 0
age
cp
trestbps
chol
fbs
restecg
thalach
exang
oldpeak
slope
ca
thal
sex_0
sex_1
target_0
target_1
0.8
0.4
0.0
−0.4
−0.8

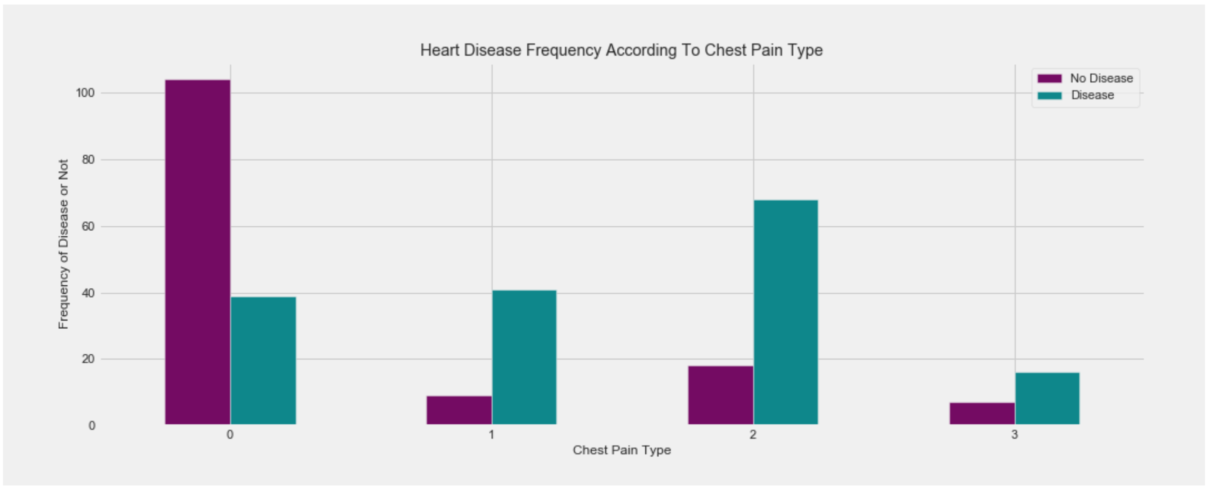

Heart Disease Frequency for Sex
No Disease
Disease
Frequency
0
20
40
60
80
100
0
1
Sex (0 = Female, 1 = Male)

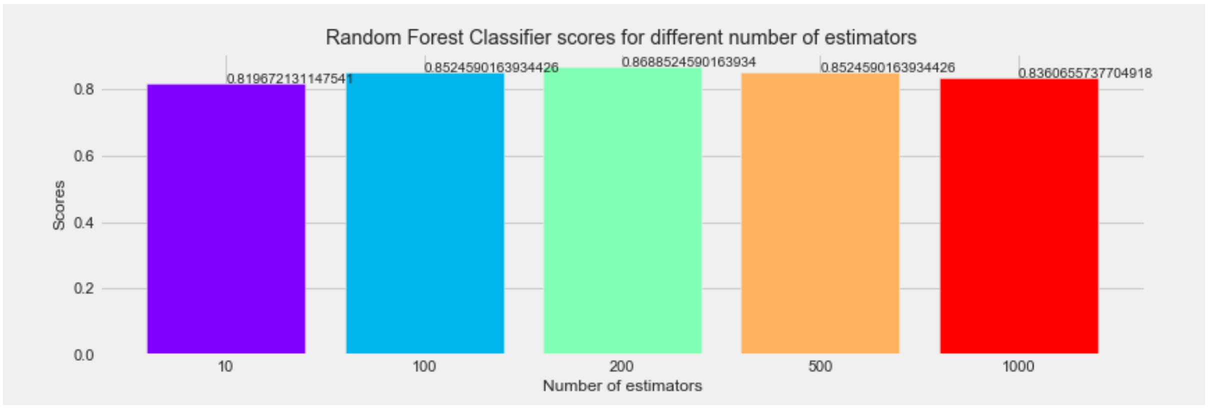

Heart Disease Frequency for Ages
Frequency
Age
target
0
1


Maximum Heart Rate
Age
Disease
No Disease

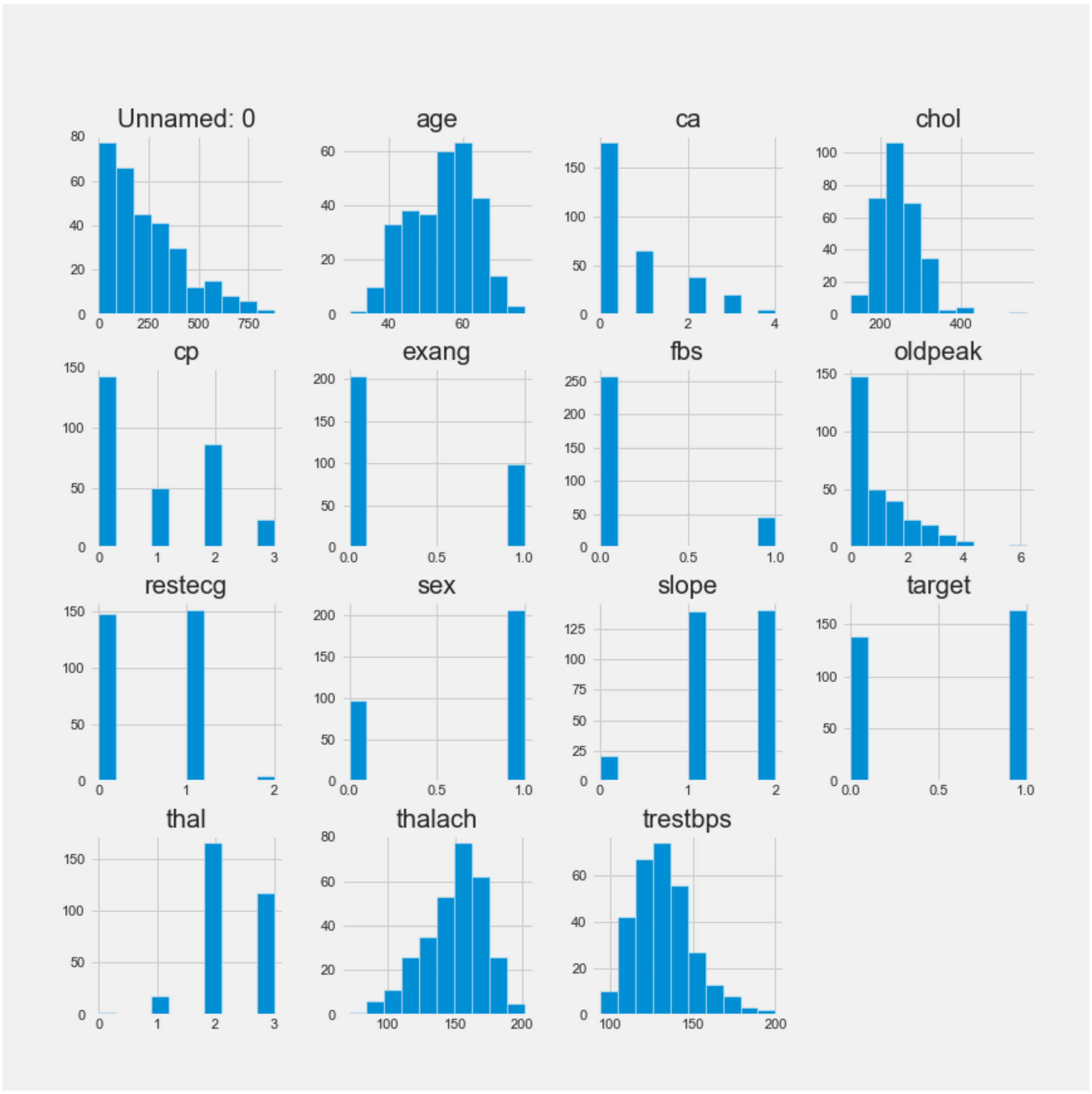

Heart Disease Frequency for Slope
Frequency
The Slope of The Peak Exercise ST Segment
target
0
1


Heart Disease Frequency According To FBS
Frequency of Disease or Not
FBS - (Fasting Blood Sugar > 120 mg/dl) (1 = true; 0 = false)
No Disease
Disease
0
1
0
20
40
60
80
100
120
140


Heart Disease Frequency According To Chest Pain Type
Frequency of Disease or Not
Chest Pain Type
No Disease
Disease
0
1
2
3
0
20
40
60
80
100


Random Forest Classifier scores for different number of estimators
Scores
Number of estimators
0.819672131147541
0.8524590163934426
0.8688524590163934
0.8524590163934426
0.8360655737704918
10
100
200
500
1000
0.0
0.2
0.4
0.6
0.8

APPENDIX – D


Unnamed: 0
age
ca
chol
cp
exang
fbs
oldpeak
restecg
sex
slope
target
thal
thalach
trestbps