

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](#_Toc38822268)

[1. INTRODUCTION 3](#_Toc38822269)

[2. METHODOLOGY 5](#_Toc38822270)

[2.1. Dataset 5](#_Toc38822271)

[2.2. Attribute Information 5](#_Toc38822272)

[2.3. Data Preprocessing 6](#_Toc38822273)

[2.4. Random Forest Classifier 7](#_Toc38822274)

[2.5. Decision Tree Classifier 8](#_Toc38822275)

[3. IMPLEMENTATION 9](#_Toc38822276)

[3.1. Understanding data 9](#_Toc38822277)

[3.2. Generating Test data 9](#_Toc38822278)

[3.3. Random Forest Classifier and Predictions 10](#_Toc38822279)

[3.4. Accuracy for test data 10](#_Toc38822280)

[3.5. Importance of the features 11](#_Toc38822281)

[3.6. Errors made by classifier 12](#_Toc38822282)

[4. EVALUATION 13](#_Toc38822283)

[4.1. Result 13](#_Toc38822284)

[4.2. Possible Limitation 13](#_Toc38822285)

[4.3. Future Work 13](#_Toc38822286)

[REFERENCE 14](#_Toc38822287)

[APPENDIX – A 15](#_Toc38822288)

[APPENDIX – B 23](#_Toc38822289)

[APPENDIX – C 24](#_Toc38822290)

# 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.

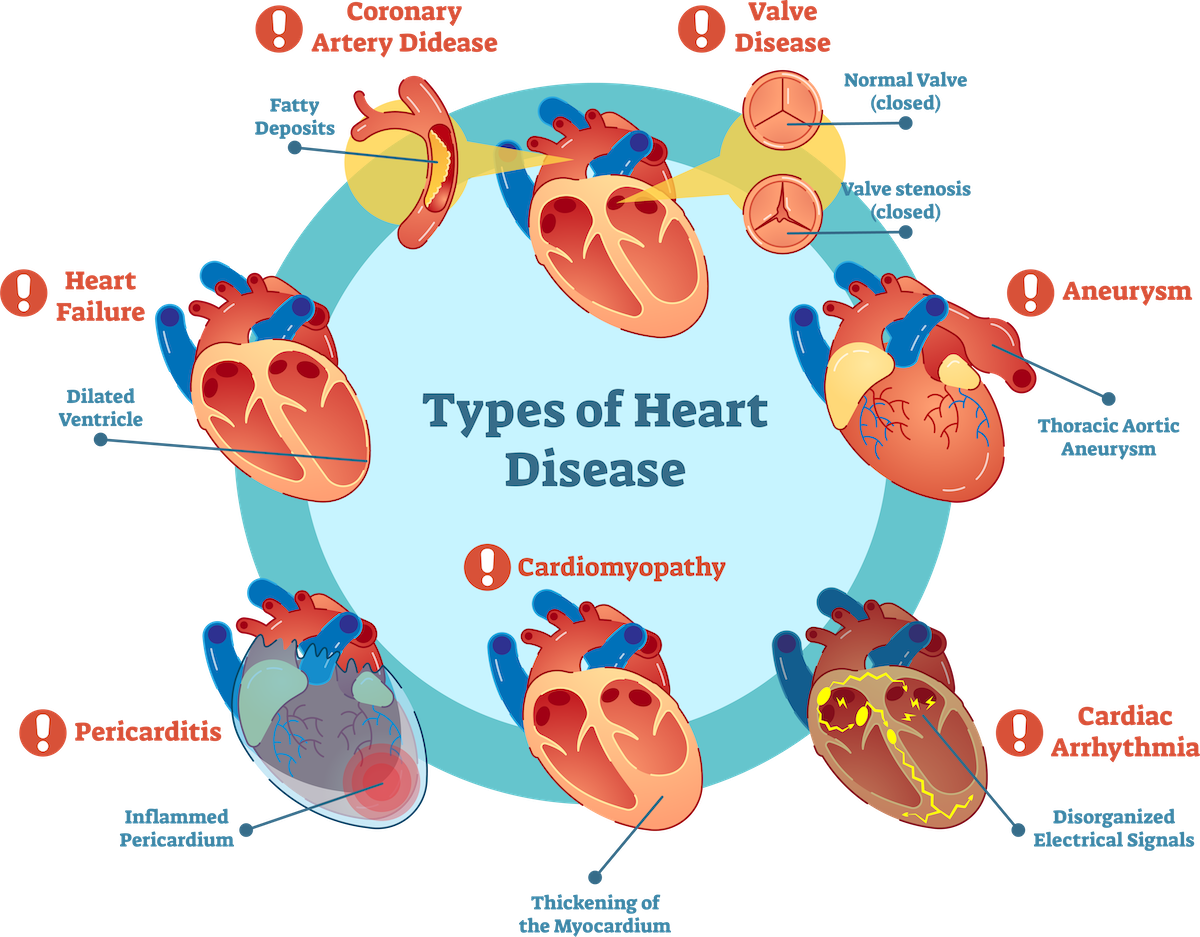


Figure 1.1: Type of Heart Disease [2].

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.

# METHODOLOGY

## 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>

## 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 |

## 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.

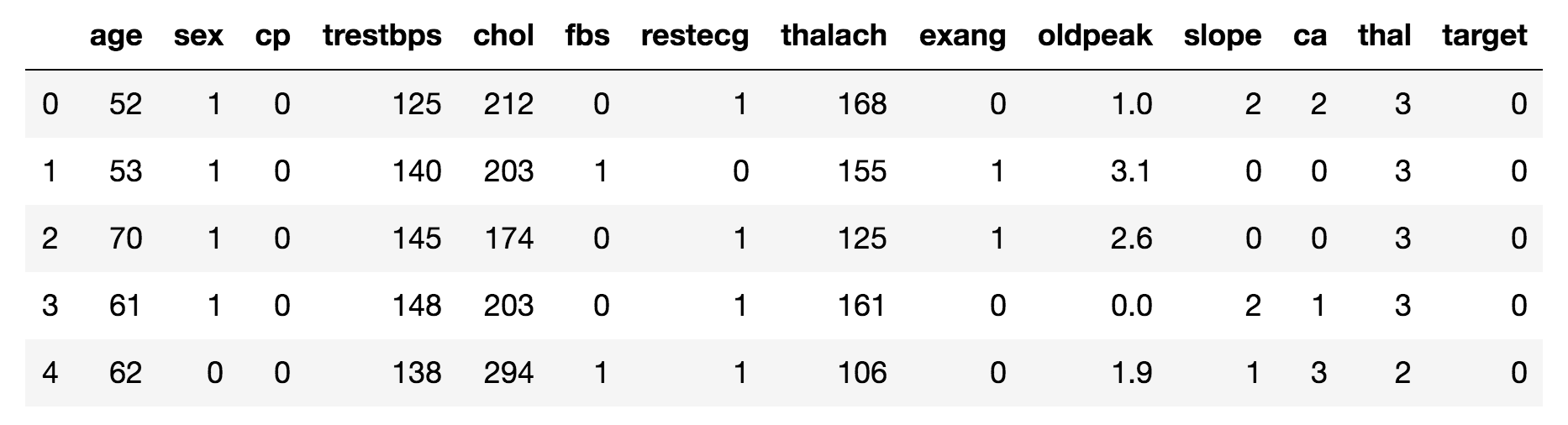


Figure 2.3.1: Original heart disease data set before preprocess.

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

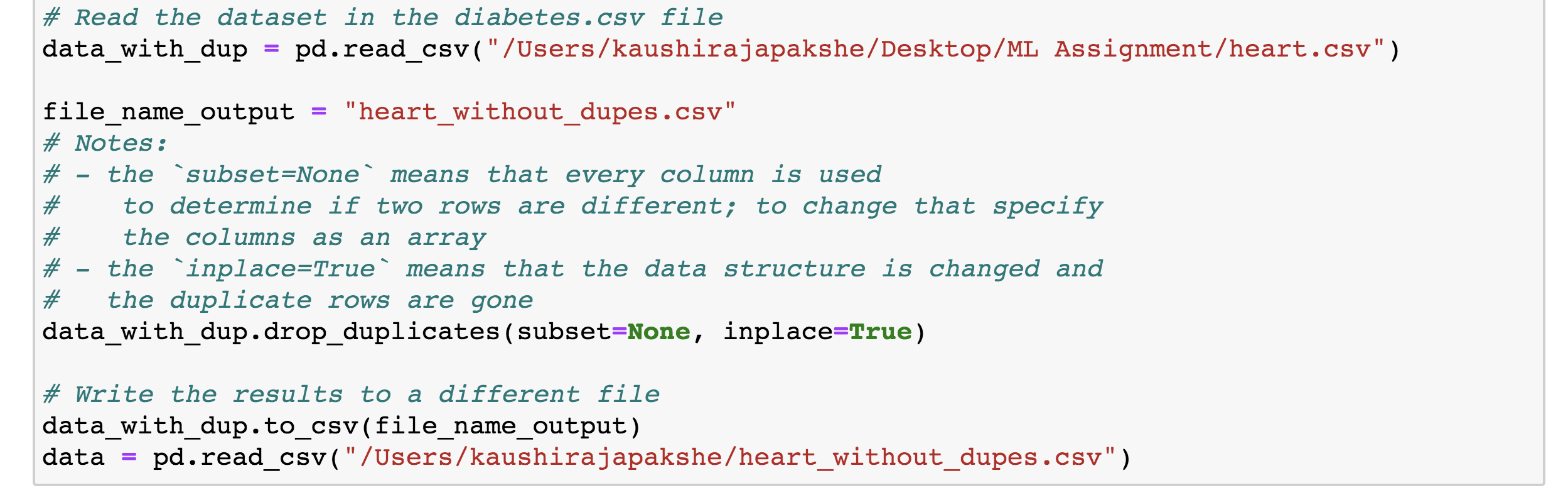


Figure 2.3.2: Code for removing duplicates and save the new dataset.

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.

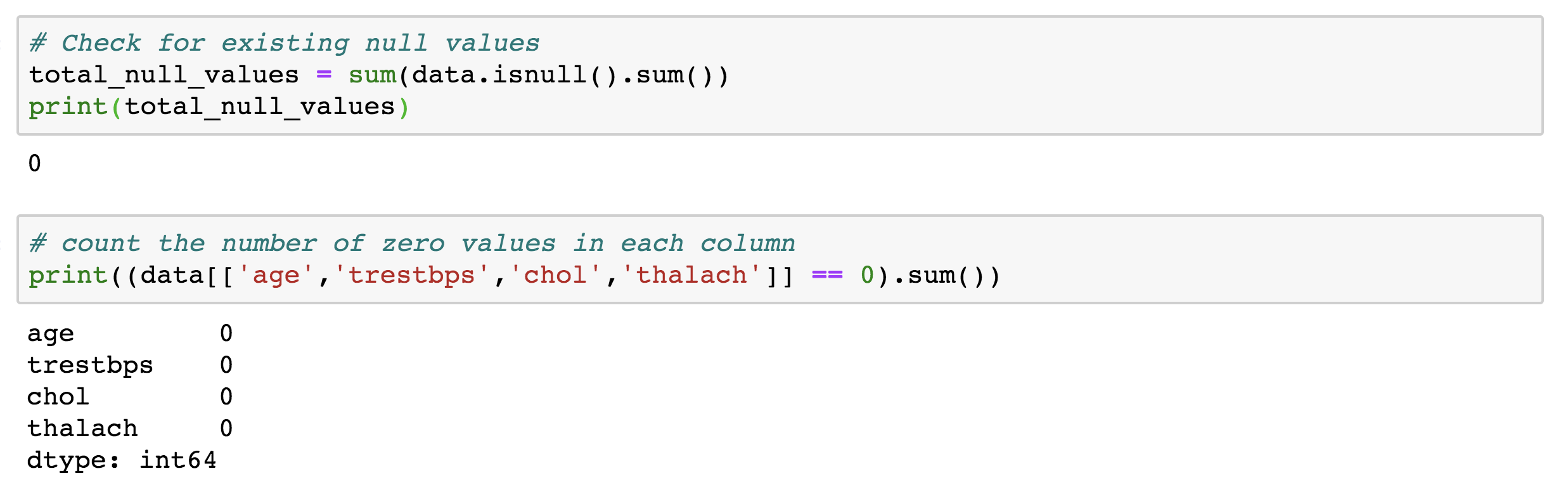
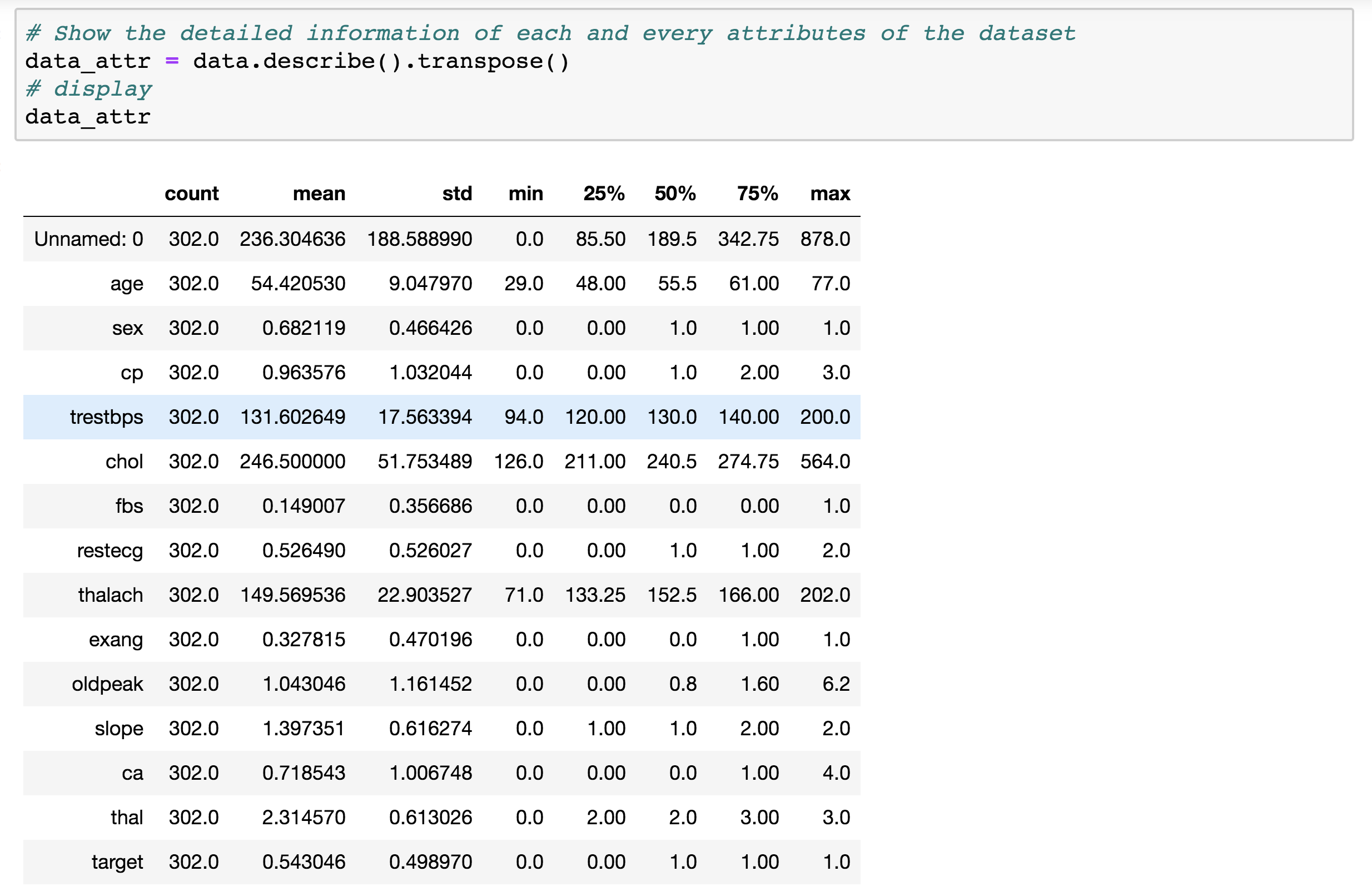


Figure 2.3.3: Dataset null and 0 values for the column.

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

Figure 2.3.4: Preprocessed Data.

## 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.

## 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.

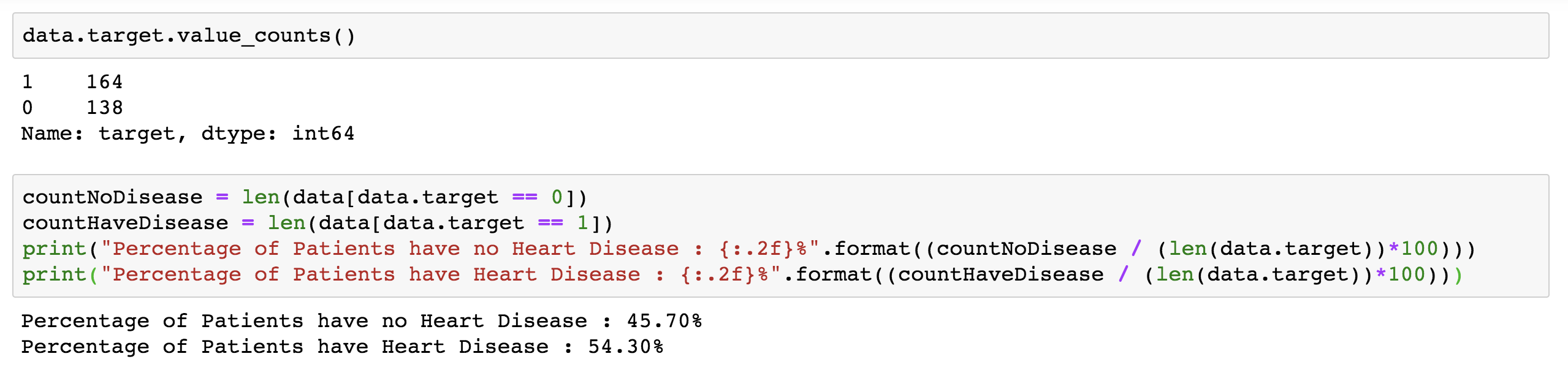
Appendix- B as sown in to narrow down the problem for heart disease using decision tree classification. According to figure 2.5.1 display how to implement a decision tree algorithm for heart disease.

Figure 2.5.1: Decision Tree Implementation

# IMPLEMENTATION

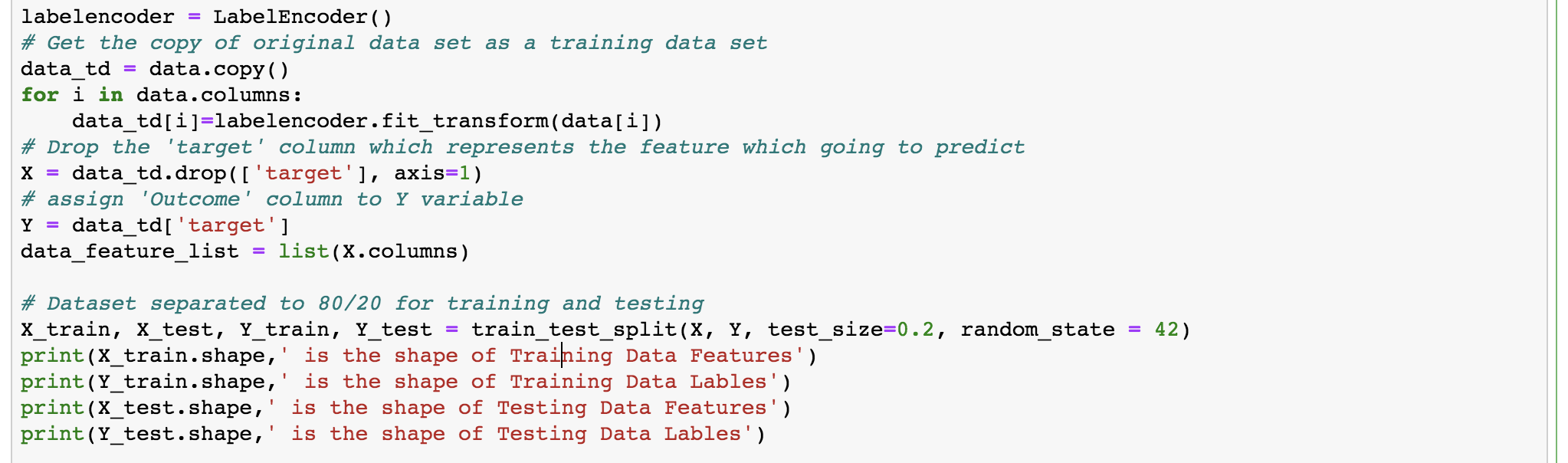
## Understanding data

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.

 Figure 3.1.1: Heart Disease target count.

## 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.

Figure 3.2.1: Generate test data and training.

## 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.

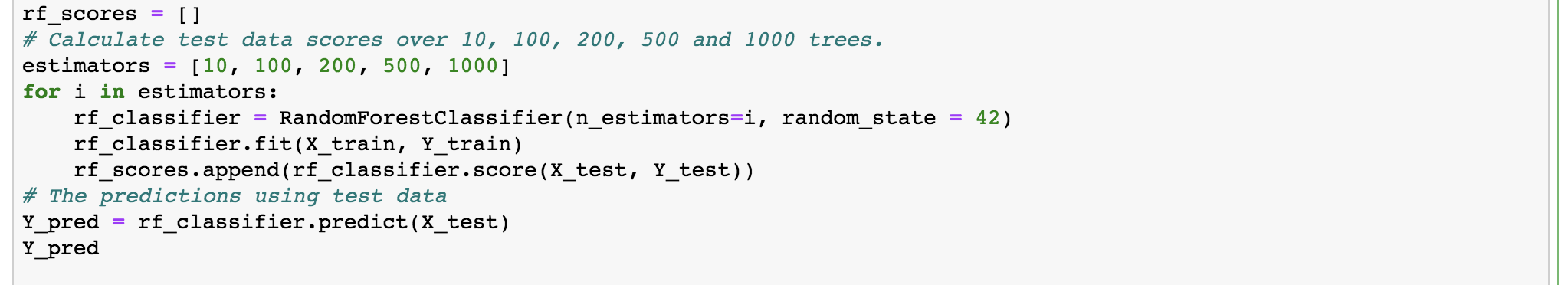
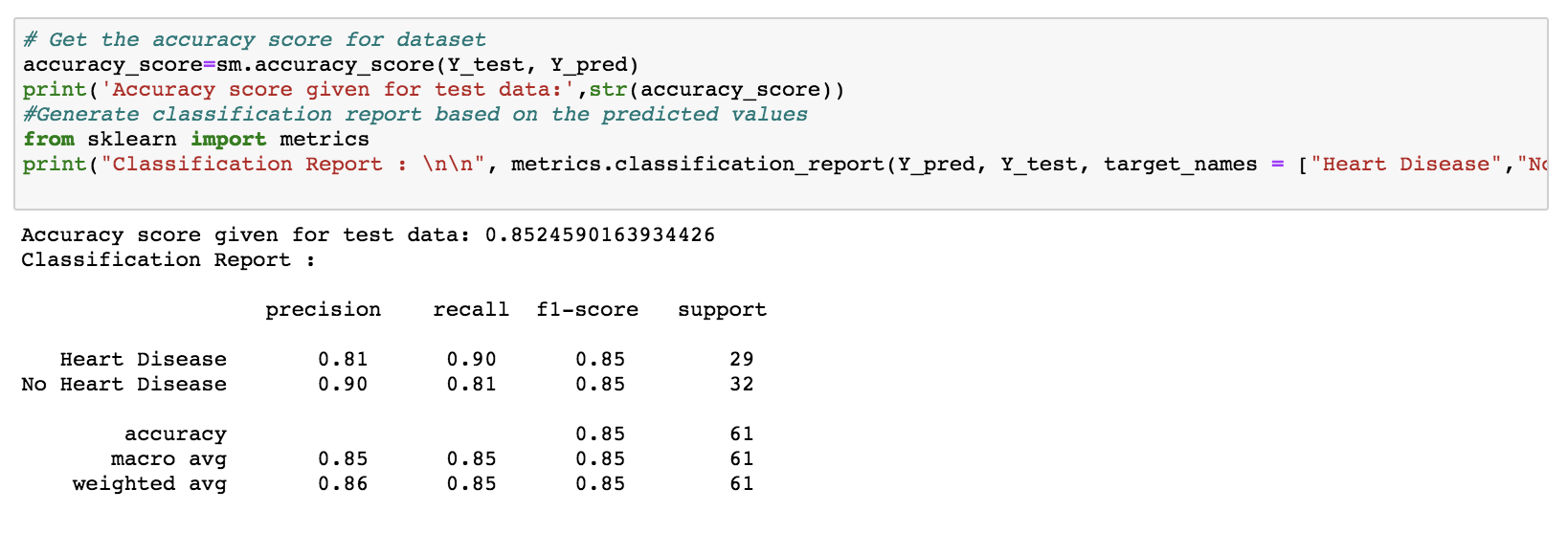


Figure 3.3.1: Random forest classifier.

## 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.

Figure 3.4.1: Accuracy for test data.

## 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.

Figure 3.5.1: Importance of the features.

## 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.

# EVALUATION

## 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%.

## 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.

## 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/heart.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\_diseases / (len(data.target))\*100)))

print("Percentage of Patients have Heart Disease : {:.2f}%".format((count\_have\_diseases / (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','#660066' ])

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','#b3ffb3' ])

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','#ccccff' ])

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', '#0e878b'])

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 estimator 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 Report : \n\n", metrics.classification\_report(Y\_pred, Y\_test, target\_names = ["Heart Disease","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(data\_feature\_list, important)]

# 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')

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,rounded = 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()

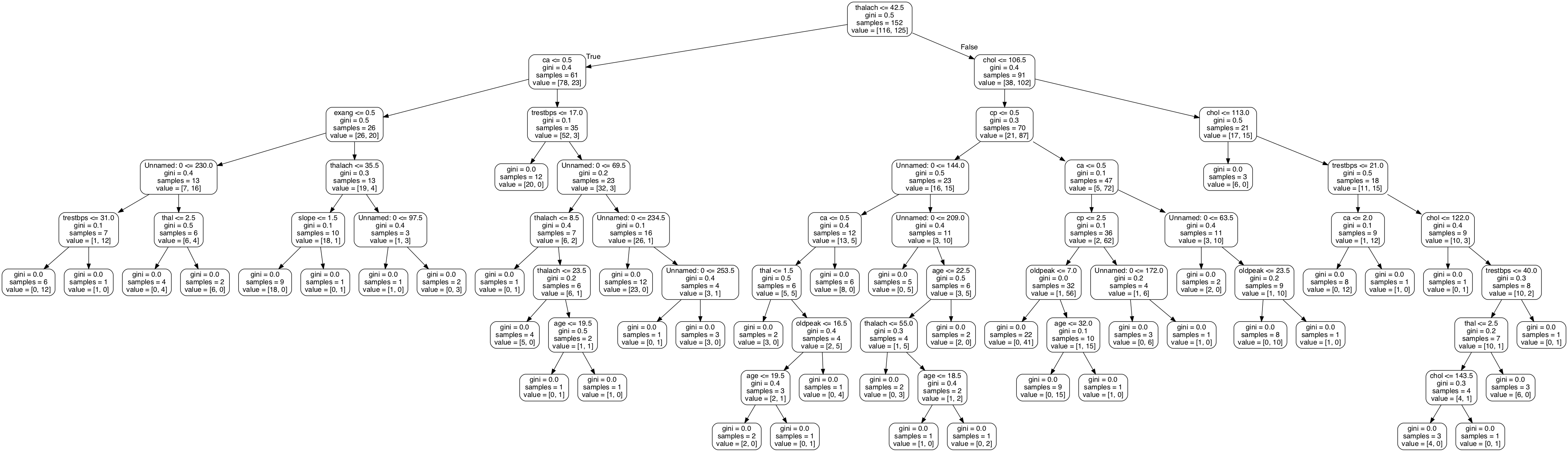
rndf\_classifier = AdaBoostClassifier(n\_estimators=1000)

rndf\_classifier.fit(X\_train, Y\_train)

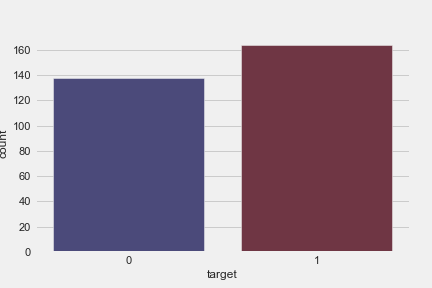
predictions = rndf\_classifier.predict(X\_test)

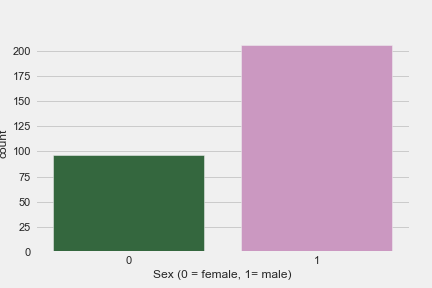
confusion\_matrix(Y\_test, predictions)

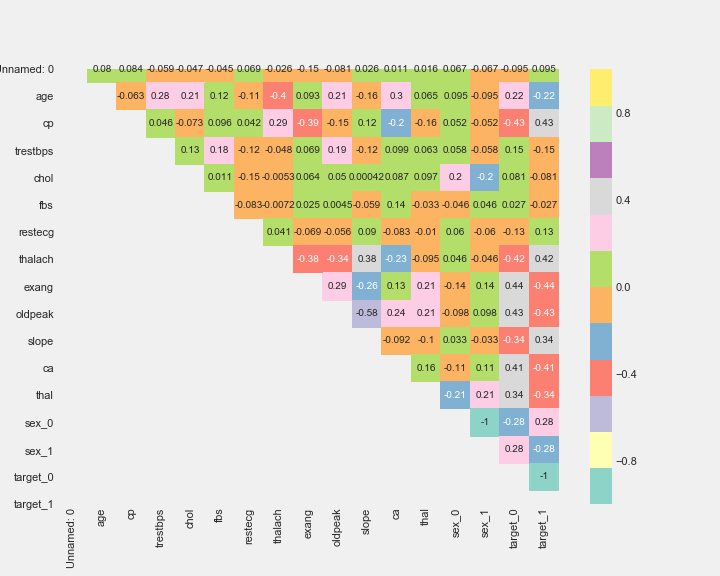
**APPENDIX – B**

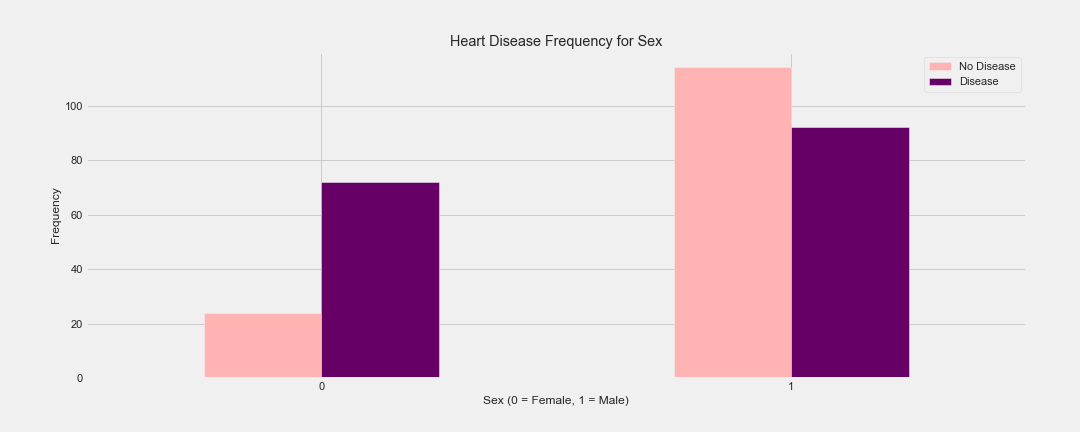
****

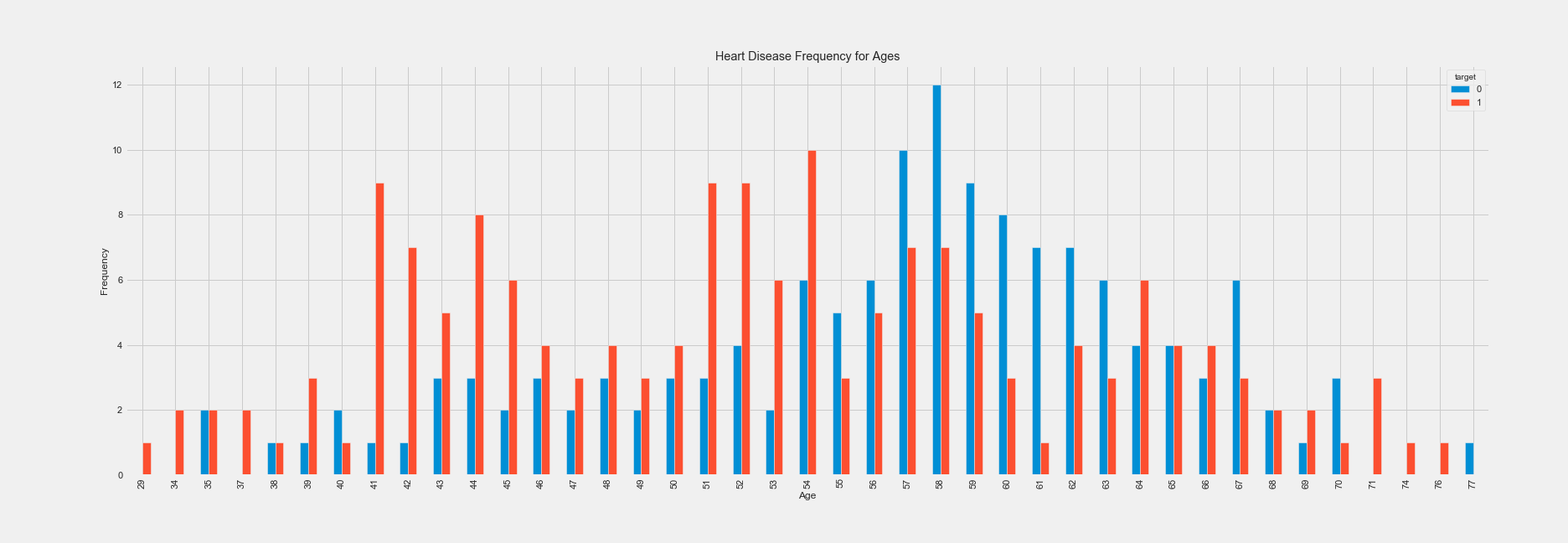
**APPENDIX – C**

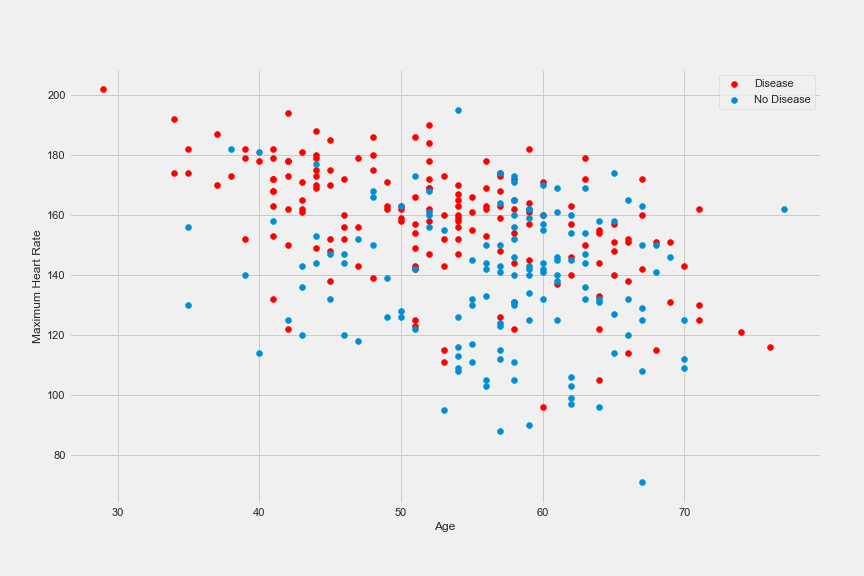
****

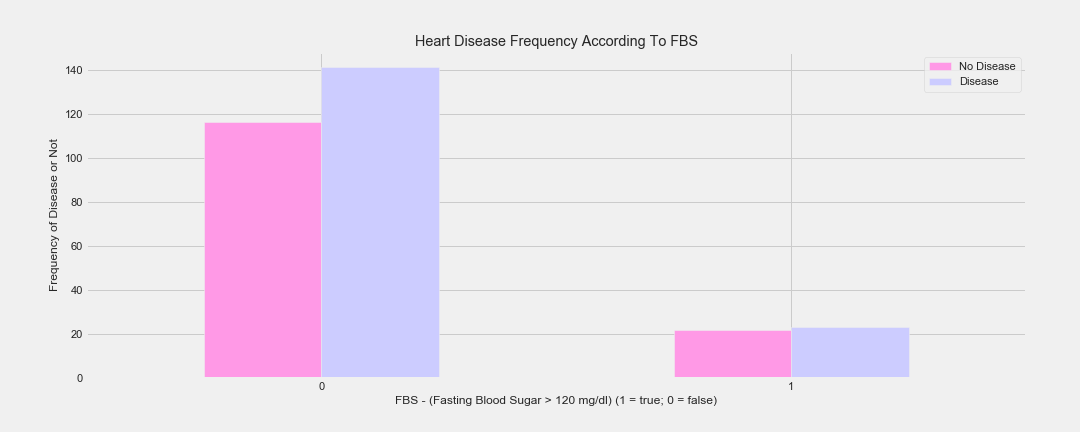
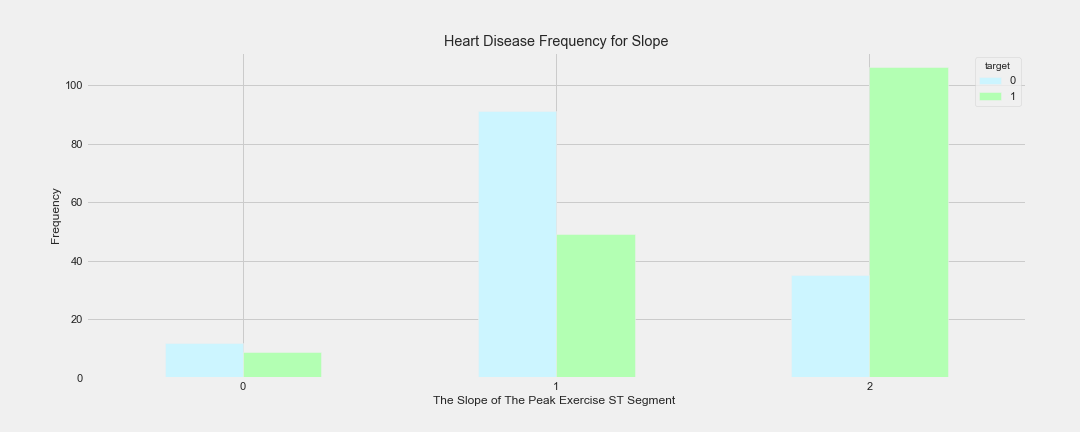
****

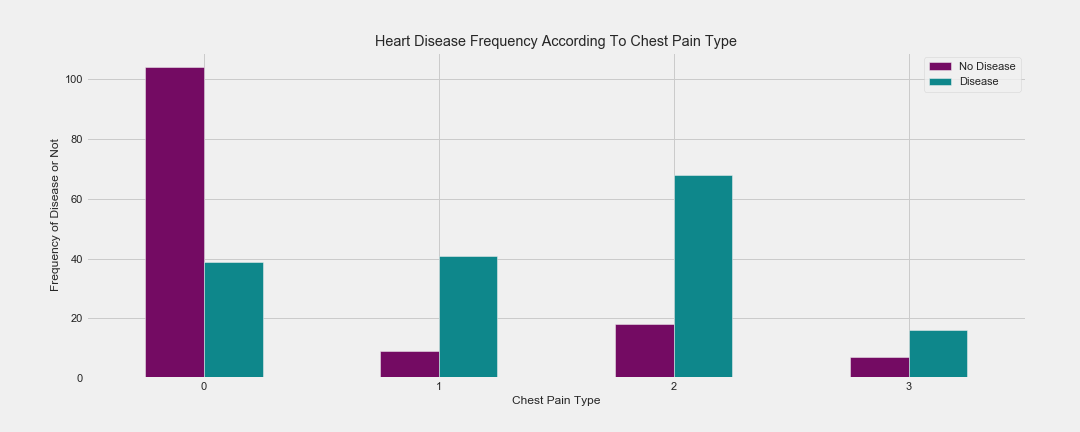
****

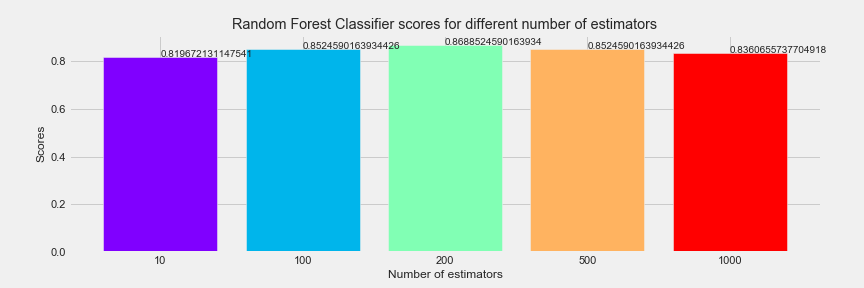
****

****

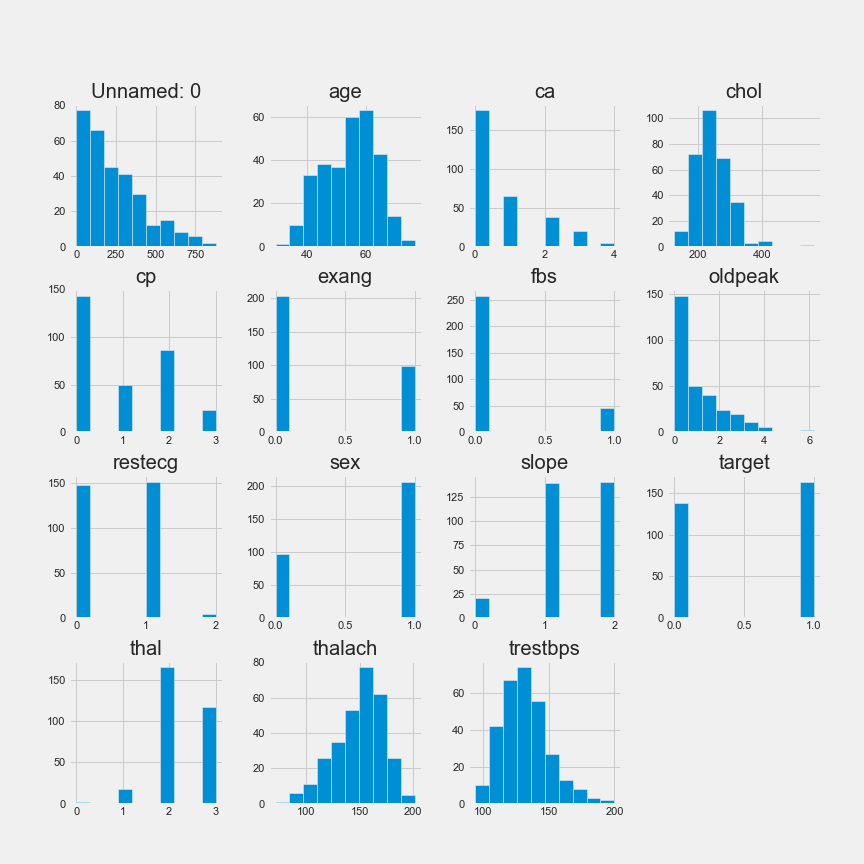
****

****

****

****

**APPENDIX – D**

****