Sri Lanka Institute of Information Technology

Department of Software Engineering

Assignment 01 - Report

Predicting Heart Diseases using Random Forest Classifier Algorithm

Machine Learning

2020

Submitted by

IT17100076

(Silva S. R. R. M.)

Table of Contents

List of Tables

List of Figures

1. Introduction

1.1. Background and Problem Statement

Nowadays heart diseases are highly trendy topic in the worldwide and also term "Cardiovascular Diseases" reference to heart diseases. World Health Organization has stated in year 2016, an estimated 17.9 million people died from heart diseases. Hence, cardiovascular diseases are the number 1 root trigger of death globally [1]. Range of conditions that can be affected or narrow down the structures or a function of human heart can be identified as heart or cardiovascular diseases and also this heart diseases can be act as umbrella term which can include heart rhythm problem, blood vessel disease, stroke, cardiomyopathy which is known as heart muscle disease and heart failure etc.

Heart disease symptoms can be differing on some key facts such as sex and also those symptoms are depended on type of heart diseases. Nevertheless, there are different varieties of risk factors accompanying with heart diseases and age, sex, high blood cholesterol level, high blood pressure, stress can be identified as risk factors of heart diseases.

This documentation contains implementation of machine learning classifier model utilizing the Random Forest Classifier Algorithm to predict frequently of a person can be affected with heart diseases base on given dataset.

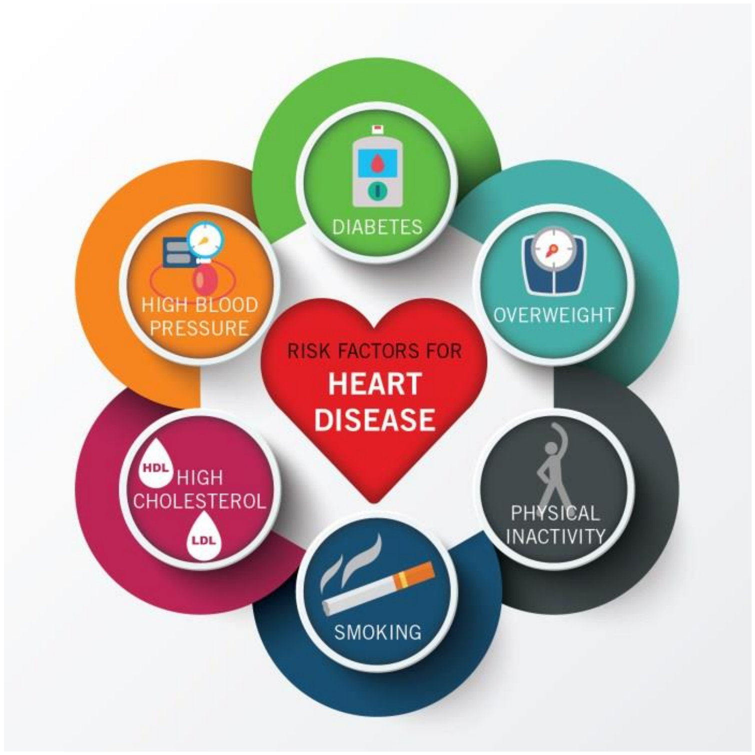


FIGURE 1 : RISK FACTORS FOR HEART DISEASE

2. Methodology

2.1. Collection of data

2.1.1. Dataset

In order to get the usage of Random Forest Classifier algorithm to predict heart diseases, there is a requirement to have a dataset to train the model. “Heart Disease UCI” is the dataset which has taken to use in this scenario. This dataset is taken from Kaggle.com which is allowed users to find out public datasets under variety of classifications and domains also it is a site which grant users to announce, explore and build up model in web base data science background [3]. The “Heart Disease UCI” dataset has stored in comma separated value (CSV) file. This dataset is available at [2].

Characteristics of dataset	Multivariate
Characteristics of attributes	Real, Integer, Categorical
Associate tasks	Classification
Number of instances	303
Number of attributes	14

Table 2. 1 : Information of Dataset

2.1.2. Dataset description

Below table describes attributes which are used in “Heart Disease UCI” dataset and description of each attribute. Description of attributes has taken from [4].

Attribute	**Description of the attribute**
Age	Age in years

Sex	Male = 1
	Female = 0

Cp	Chest pain • 0: chest pain related decrease blood supply to the heart • 1: chest pain not related to heart • 2: typically, esophageal spasms (non-heart related) • 3: chest pain not showing signs of disease
Trestbps	Resting blood pressure
Chol	Serum cholesterol in mg/dl (above 200 is trigger for concern)

Fbs	Fasting blood sugar
	True = 1
	False = 0

Restecg	Resting electrocardiographic result • 0: Nothing to note • 1: ST-T Wave abnormality • 2: Possible or definite left ventricular hypertrophy
Thalach	Maximum heart rate achieved

Exang	Exercise induced angina
	Yes = 1
	No = 0

Oldpeak	ST depression induced by exercise relative to rest Unhealthy heart will stress more
Slope	ST segment of peak exercises • 0: Up sloping: healthier heart rate with exercise (unusual) • 1: Flat sloping: minimal change (normal healthy heart) • 2: Down sloping: signs of unhealthful heart
Ca	Number of foremost vessels colored by fluoroscopy (0-3)
Thal	Thallium stress result • 3: normal • 6: fixed defect • 7: reversable defect

Target	Predicted attribute
	True = 1
	False = 0

TABLE 2. 2 : DESCRIPTION OF ATTRIBUTES

In the given dataset attribute called “target” outlines whether a patient is positive with heart disease or not. Attribute “target” may encompass two values. If the value of target = 1 which illustrates patient is positive with heart disease, and if the target value = 0, it defines patient is not having heart diseases.

2.2. Random Forest Classifier Algorithm

Machine learning is a self-driving science of gaining computers to perform deprived of being unambiguously programmed. Furthermore, Alpaydin, E. has stated that machine learning is not only a problem of database, but it is subpart of Artificial Intelligent (AI)[5]. How even, machine learning to be intelligent, a system which is in an altering environment ought to be have capability to learn [5]. Below figure 2 evident up very basic conception of machine learning. Algorithms which are used in machine learning can be categorized under below.

- Supervised Learning Algorithm
- Unsupervised Learning Algorithm
- Reinforcement Learning Algorithm

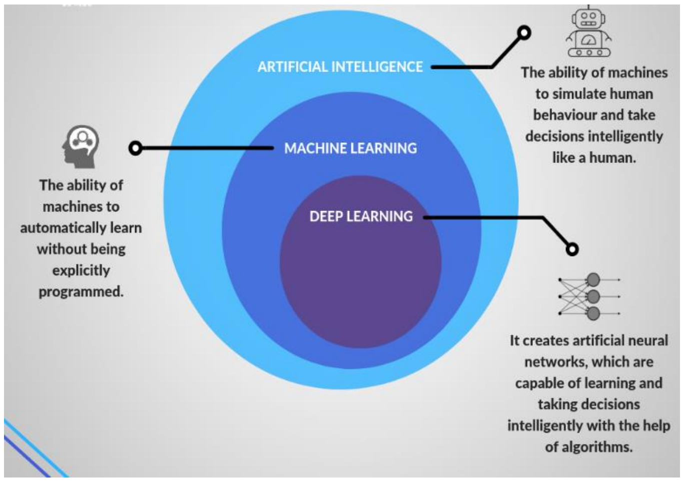

ARTIFICIAL INTELLIGENCE
The ability of machines to simulate human behaviour and take decisions intelligently like a human.
MACHINE LEARNING
The ability of machines to automatically learn without being explicitly programmed.
DEEP LEARNING
It creates artificial neural networks, which are capable of learning and taking decisions intelligently with the help of algorithms.

FIGURE 2 : MACHINE LEARNING CONCEPT

Random Forest is a formidable machine learning classifier algorithm which is grouped under supervised learning algorithm and which comprises high classification accuracy, nature of non-parametric and also competence to ascertain variable importance. How even, Random forest can be deemed as black box type classifier due for to its unspecified split rules used for classification. This algorithm flexible and straightforward to apply and further than this can be utilized both regression and classification.

Random forest is highly accurate algorithm and bulky number of decision trees are participating for this algorithm so that it can be considered as robust method and also this model does not contain overfitting problematic, it is proficient with handling missing values and also random forest, which benefits in opt for the furthermost contributing features for the classifier. Selection of the algorithm has been accomplished by means of deliberation of above-mentioned gains of Random forest classifier algorithm.

Creation of random forest and prediction using random forest can be named as two phases of random forest algorithm. Over this algorithm, ‘x’ number of features are randomly taken from the given dataset and amongst of the selected ‘x’ features node (d) or best split point is calculated and using the best split point it spilt in to children nodes and repeat same steps until it returns the ‘1’ number of nodes. By repeating above all steps for ‘n’ number of times, will create ‘n’ number of trees which is resulted in creation of random forest.
As the next phase of random forest algorithm, from the given dataset test features are taken and get the usage of the rules of each decision trees which have been formed randomly to make the prediction and each and very predicted values are calculated it’s votes and among of those votes, high voted practiced outcome ponder as concluding prediction from the algorithm.

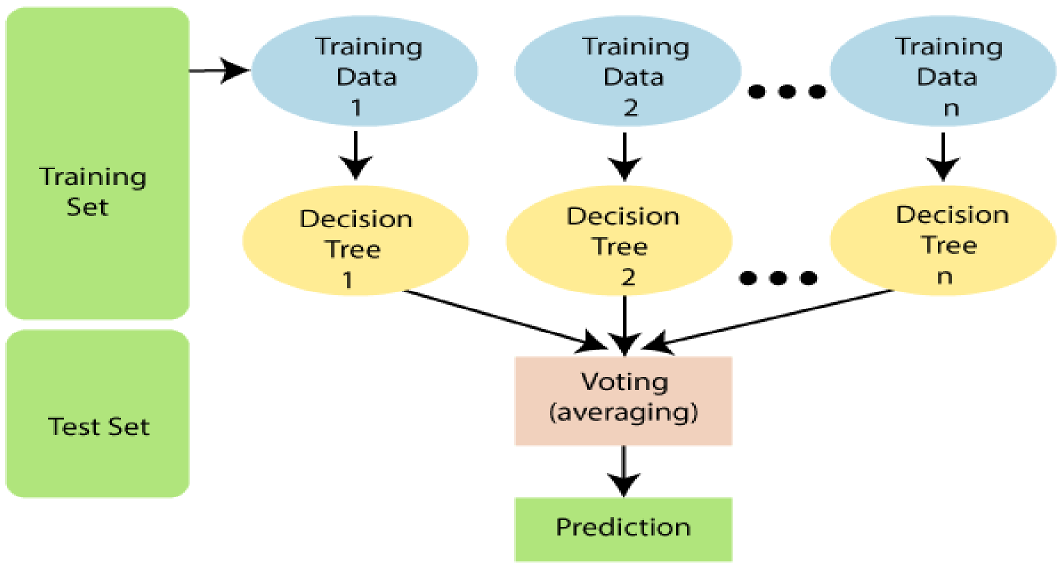
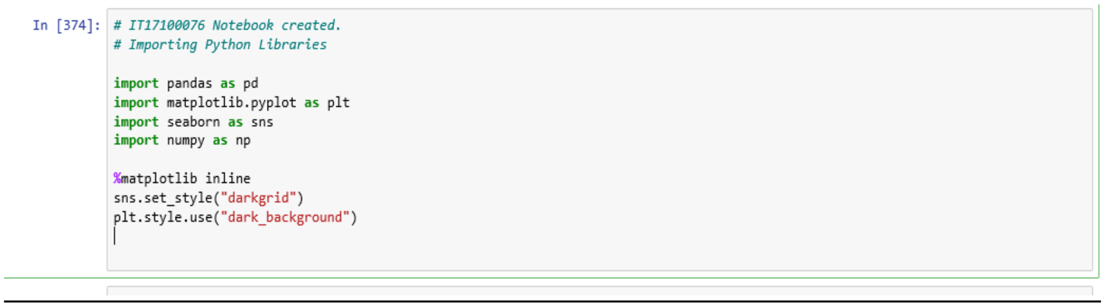


FIGURE 3 : SIMPLE PROCESS OF RANDOM FOREST ALGORITHM

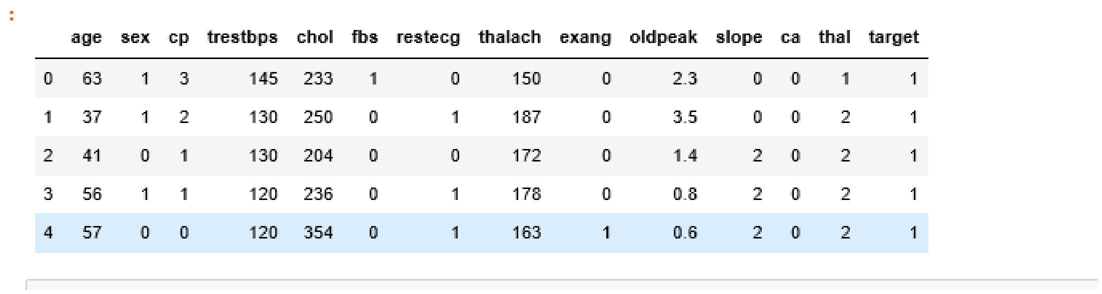
2.3. Implementation



```
In [374]: # IT17100076 Notebook created.
          # Importing Python Libraries

          import pandas as pd
          import matplotlib.pyplot as plt
          import seaborn as sns
          import numpy as np

          %matplotlib inline
          sns.set_style("darkgrid")
          plt.style.use("dark_background")
```

FIGURE 4 : IMPORTING LIBRARIES

Code segment included in figure 4 indicates importing all libraries which is a basic requirement in order for execution of the entire code. In that case importing libraries are considered as first step of the implementation. Importing 'pandas' is use for data manipulation and numpy is used for covert arrays.



	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

FIGURE 5 : LOADED DATA

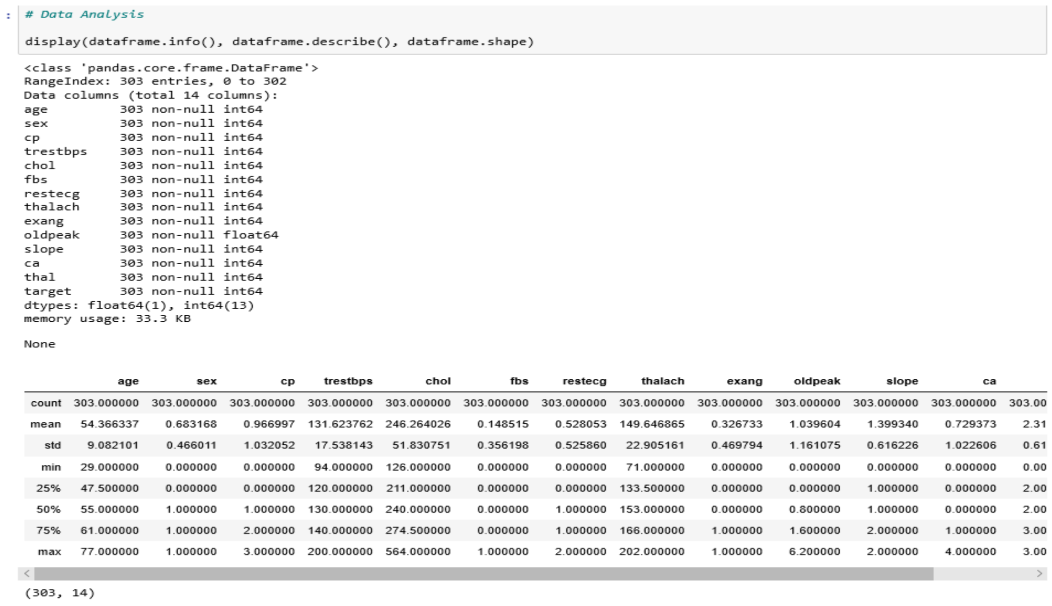
Below code is resulted in figure 5 output.

```
# Loading data

dataframe = pd.read_csv("D:\\SLIIT\\4th Year\\1st Semester\\ML\\Assignments\\Assignment -
01\\IT17100076\\it17100076_heart_disease_dataset.csv")

# In[376]:


# All data of first 5 rows

dataframe.head()
```

This code is supported to load data from given dataset by directing to given path, read and it prints data of first five rows.

```
# Data Analysis

display(dataframe.info(), dataframe.describe(), dataframe.shape)
```

This code is supported in data analysis and it prints all the information related to data set and describes the descriptive statistics for the given dataset and datatypes of the attributes. Below figure 6 is illustrated the output of above code.



```
: # Data Analysis

  display(dataframe.info(), dataframe.describe(), dataframe.shape)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
age         303 non-null int64
sex         303 non-null int64
cp          303 non-null int64
trestbps    303 non-null int64
chol        303 non-null int64
fbs         303 non-null int64
restecg     303 non-null int64
thalach     303 non-null int64
exang       303 non-null int64
oldpeak     303 non-null float64
slope       303 non-null int64
ca          303 non-null int64
thal        303 non-null int64
target      303 non-null int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB

None
```

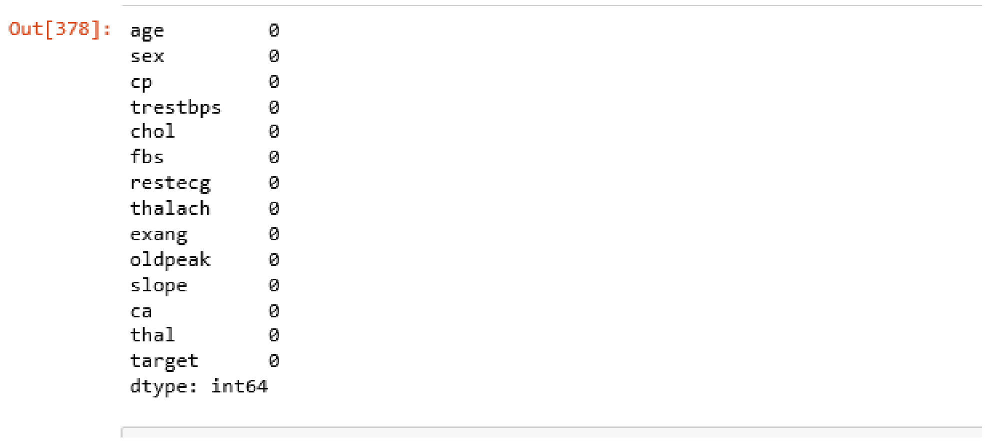
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.00
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.31
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.61
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.00
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.00
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.00
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.00
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.00

```
(303, 14)
```

FIGURE 6 : OUTPUT INFO DATASET

```
# checking for null values

dataframe.isna().sum()
```

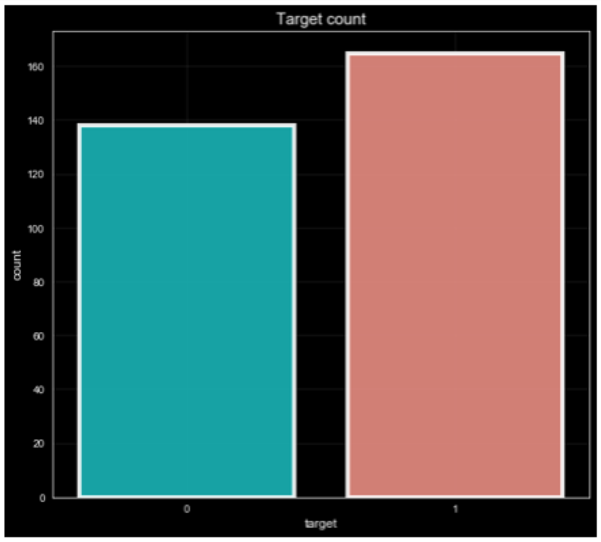
Above code is used to check whether the given dataset contain null values or not. In this case by considering the output in figure 7, it represented that there is no null values found in dataset which has been taken on this task.



```
Out[378]: age         0
          sex         0
          cp          0
          trestbps    0
          chol        0
          fbs         0
          restecg     0
          thalach     0
          exang       0
          oldpeak     0
          slope       0
          ca          0
          thal        0
          target      0
          dtype: int64
```

FIGURE 7 : CODE OUTPUT

```
In [52]: # Getting target count

         colors = ['darkturquoise', 'salmon']
         plt.style.use('dark_background')
         plt.rcParams['figure.figsize']=(9,8)

         axis = sns.countplot(x='target', data=dataframe, palette=colors, alpha=0.9, edgecolor=('white'), linewidth=4)
         axis.set_ylabel('count', fontsize=12)
         axis.set_xlabel('target', fontsize=12)
         axis.grid(b=True, which='major', color='grey', linewidth=0.2)
         plt.title('Target count', fontsize=15)
         plt.show()

         target_0 = len(dataframe[dataframe.target == 0])
         target_1 = len(dataframe[dataframe.target == 1])

         print("Percentage of negative Heart Disease: {:.2f}%".format((target_0 / (len(dataframe.target))*100)))
         print("Percentage of  positive Heart Disease: {:.2f}%".format((target_1 / (len(dataframe.target))*100)))
         dataframe.target.value_counts()
```



```
Percentage of negative Heart Disease: 45.54%
Percentage of  positive Heart Disease: 54.46%

Out[52]: 1    165
         0    138
         Name: target, dtype: int64
```

FIGURE 8 : TARGET COUNT

Moving attention to code in figure 8 is resulted in getting target count. In this case 138 patients are not falling under heart diseases and on the other hand 165 number of patients are having heart diseases. Which mean selected dataset has approximately equivalent number of target count, having this kind of dataset can be considered as good in training a model in machine learning. Outcome of bar chat represent more than 50% of distributing is with heart diseases.

```
# Creating correlation Metrix

plt.rcParams['figure.figsize'] = 15, 15
plt.style.use('dark_background')
plt.matshow(dataframe.corr())
plt.yticks(np.arange(dataframe.shape[1]), dataframe.columns)
plt.xticks(np.arange(dataframe.shape[1]), dataframe.columns)
plt.colorbar()
```

Data visualization is generating data information in graphical/ in visual way which is easy to identify and understand the patterns or trends of data rather going through thousand of row of data. Code segment has shown above support on data visualization and it is generated correlation matrix. Correlation matrix demonstrates correlation coefficients amongst set of variables in a table of format and which make available to summarize and in advance analysis of data.

```
# In[382]:


dataframe.hist()
plt.style.use('dark_background')
```

Histogram is another way of data visualization, and in here by using above line of code exemplifies dissemination of each attribute as in an array under different scope of distribution.

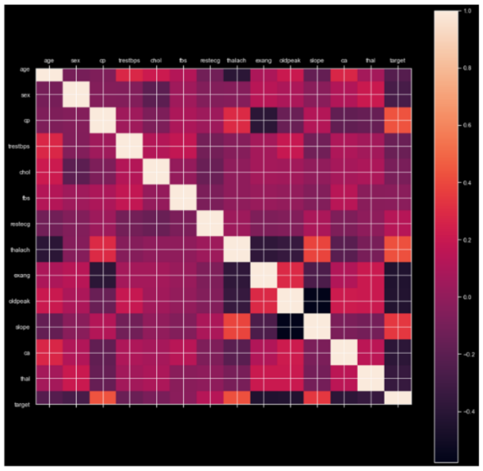


Figure 9 : Correlation matrix

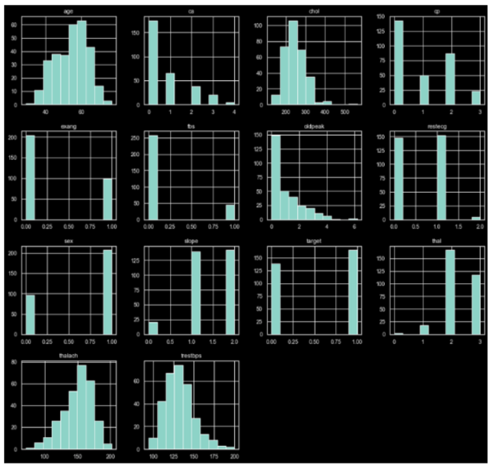


Figure 10 : Hist

```
# In[383]:
# Create another figure to expose max heart rate for age
plt.figure(figsize=(10, 8))
# Scatter with negative and postivie examples
plt.scatter(dataframe.age[dataframe.target==0],
            dataframe.thalach[dataframe.target==0],
            c="lightblue")
plt.scatter(dataframe.age[dataframe.target==1],
            dataframe.thalach[dataframe.target==1],
            c="salmon")
plt.title("Heart Disease in function in range of Age and Max Heart Rate")
plt.xlabel("Age")
plt.ylabel("Max Heart Rate")
plt.legend(["positive", "Negative"]);
```

Above code segment resulted in below figure which represent data visualization on maximum heart rate according to the age value.

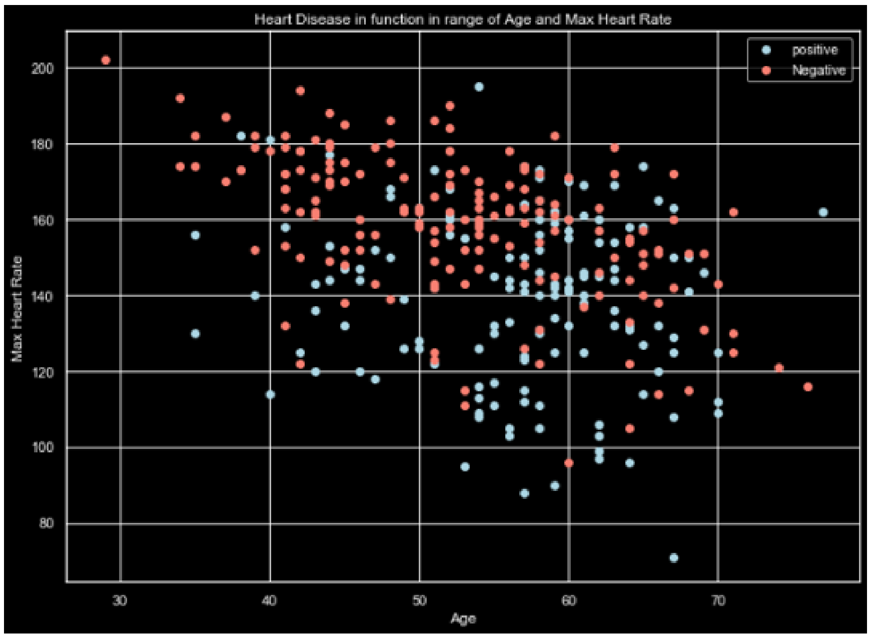


FIGURE 11 : SCATTER GRAPH

```
# Filtering values

categorical_values = []
continous_values = []
for column in dataframe.columns:
    if len(dataframe [column].unique()) <= 10:
        categorical_values.append(column)
    else:
        continous_values.append(column)


# In[385]:

# Getting categorical values from the dataset
categorical_values


# In[386]:

# Creating dummy columns for categorical values

categorical_values.remove('target')
dataset = pd.get_dummies(dataframe, columns = categorical_values)
dataset.head()
```

Whenever after investigating on the dataset, there is a requirement where some categorical attributes ought to transform in to dummy values and also should scale all the variable value prior to proceed with train the model in machine learning. In the above used code segment as a very first step value filtering has done. It has filtered out all the categorical and continuous values and from the 'In[385]' it prints all the categorical values and then attribute 'target' which is a predicted value is removed from categorical values and dummy columns are created for the other remaining categorical values, by using 'get_dummies'. And then scaling up the provide columns has done using below code segment. And also, another library has imported which is used when scale the features.

```
# Scaling provied columns
from sklearn.preprocessing import StandardScaler

ssc = StandardScaler()
columns_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
dataset[columns_to_scale] = ssc.fit_transform(dataset[columns_to_scale])
```

Below code segment represents very first importing a library which is applied in splitting, training and testing a dataset. And then it is divided entire dataset into two separate datasets which are names as training dataset and testing dataset when train a model it will use training dataset and testing dataset take a place where trained model place to test for accuracy.

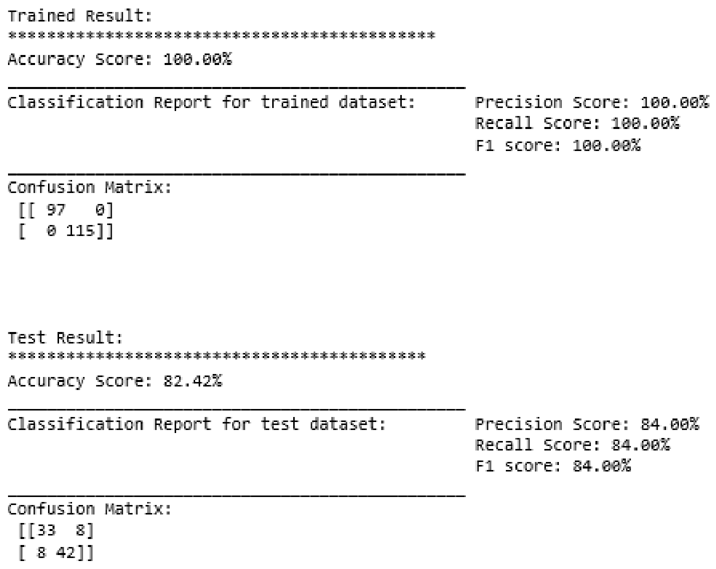
```
#Split dataset into 2 separate datasets
from sklearn.model_selection import train_test_split

X = dataset.drop('target', axis=1)
y = dataset.target

X_trainingdataset, X_testingdataset, y_trainingdataset, y_testingdataset = train_test_split(X,
y, test_size=0.3, random_state=42)
```

Code segment which has shown below, embodies how to apply random forest classifier algorithm to predict heart diseases. Thus, Random forest classifier machine learning algorithm is imported. In this case number of decision trees used are 1000. It indicates from the parameter called ‘n_estimators’ simply it demonstrate to predict heart disease using random forest, this scenario has used 1000 of decision trees in the forest.

```
# In[391]:


# Apply Random Forest Classifier Algorithm

from sklearn.ensemble import RandomForestClassifier


random_forest = RandomForestClassifier(n_estimators=1000, random_state=42)
random_forest.fit(X_trainingdataset, y_trainingdataset)


print_score(random_forest, X_trainingdataset, y_trainingdataset, X_testingdataset,
y_testingdataset, train=True)
print_score(random_forest, X_trainingdataset, y_trainingdataset, X_testingdataset,
y_testingdataset, train=False)
```

3. Results and Discussion

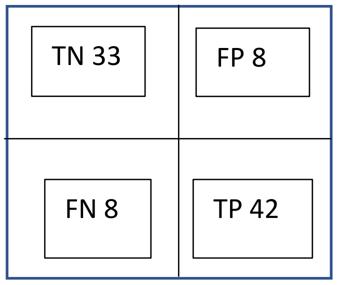
Classification report for trained and tested data has generated as a result and by using random classifier algorithm.



```
Trained Result:
********************************************
Accuracy Score: 100.00%
_______________________________________________
Classification Report for trained dataset:      Precision Score: 100.00%
                                                Recall Score: 100.00%
                                                F1 score: 100.00%
_______________________________________________
Confusion Matrix:
 [[ 97   0]
 [  0 115]]



Test Result:
*******************************************
Accuracy Score: 82.42%
_______________________________________________
Classification Report for test dataset:         Precision Score: 84.00%
                                                Recall Score: 84.00%
                                                F1 score: 84.00%
_______________________________________________
Confusion Matrix:
 [[33  8]
 [ 8 42]]
```

FIGURE 12 : REPORT OF CLASSIFICATION

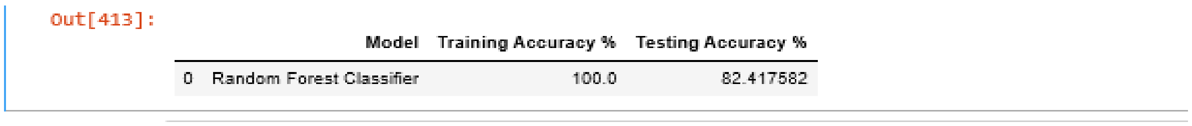
Above figure 11 has shown the main classification metrics precisions, recall, f1-score for each trained and test results. Based on true, false values, the report is calculated. Precision illustrates, the ratio of true positives to sum of true and false positives. In simply way this precision of the report shows accuracy of positive prediction whereas recall is the fraction of positives that were correctly identified. Further it defines the ratio of true positives to sum of true and false negatives [6]. Simply as recall is an ability of a classifier to discover positive occurrences. F1 score is not global accuracy but it is utilized to compare classifier models. Confusion matrix is a particular layout of the table which is make available visualization of performance of an algorithm. And it also known as error matrix. The usage of compute confusion matrix is to assess the accuracy of a classification which is resulted in returns an array of actual values and predicted values for each feature label. When consider about confusion matrix for test result,



41 instances correspond as ‘No heart diseases’ whereas actually 33 instances negative with heart diseases and model also able to identify it correctly. Though 50 of instances considered that have heart diseases and but actually 42 are having and model also accurately predicted it.

Base on confusion matrix test result accuracy calculate as per given format.

Testing Accuracy = (TN + TP)/ FN + TN + FP + TP



Out[413]:

	Model	Training Accuracy %	Testing Accuracy %
0	Random Forest Classifier	100.0	82.417582

FIGURE 13 : TRAINING AND TESTING ACCURACY

From the result obtained in figure 12 it can be determined that random forest has given some considerable high accuracy in this task.

Moving further pay attention of limitation of used algorithm complexity can be main limitation of random forest classifier algorithm due training huge number of deep trees may use lot of memory usage also much more power of computation and also time taken for training a model is time consuming.

4. References

[1].Who.int. 2020. *Cardiovascular Diseases (Cvds)*. [online] Available at: <https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)> [Accessed 18 April 2020].

[2].Kaggle.com. 2020. *Heart Disease UCI*. [online] Available at: <https://www.kaggle.com/ronitf/heart-disease-uci> [Accessed 19 April 2020].

[3].En.wikipedia.org. 2020. *Kaggle*. [online] Available at: <https://en.wikipedia.org/wiki/Kaggle> [Accessed 19 April 2020].

[4].Archive.ics.uci.edu. 2020. *UCI Machine Learning Repository: Heart Disease Data Set*. [online] Available at: <http://archive.ics.uci.edu/ml/datasets/Heart+Disease> [Accessed 20 April 2020].

[5].Alpaydin, E., 2020. *Introduction To Machine Learning*. [online] Google Books. Available at: <https://books.google.lk/books?hl=en&lr=&id=tZnSDwAAQBAJ&oi=fnd&pg=PR7&dq=machine+learning+&ots=F2ZW8Z5vzg&sig=ukn-3X2iwD6qt71L_kWYIRVMmqs&redir_esc=y#v=onepage&q=machine%20learning&f=false> [Accessed 20 April 2020].

[6].Scikit-yb.org. 2020. *Classification Report — Yellowbrick V1.1 Documentation*. [online] Available at: <https://www.scikit-yb.org/en/latest/api/classifier/classification_report.html> [Accessed 21 April 2020].

5. Appendices

Appendix A – List of Codes

```
#!/usr/bin/env python
# coding: utf-8
# In[374]:

# IT17100076 Notebook created.
# Importing Python Libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
get_ipython().run_line_magic('matplotlib', 'inline')
sns.set_style("darkgrid")
plt.style.use("dark_background")
#pd.set_option('display.max_columns', 500)

# In[375]:
# Loading data
dataframe = pd.read_csv("D:\\SLIIT\\4th Year\\1st Semester\\ML\\Assignments\\Assignment -
01\\IT17100076\\it17100076_heart_disease_dataset.csv")

# In[376]:
# All data of first 5 rows
dataframe.head()

# In[377]:
# Data Analysis
display(dataframe.info(), dataframe.describe(), dataframe.shape)

# In[378]:
# checking for null values
dataframe.isna().sum()

# In[379]:
# Getting target count
colors = ['darkturquoise', 'salmon']
plt.style.use('dark_background')
plt.rcParams['figure.figsize']=(9,8)
axis = sns.countplot(x='target', data=dataframe, palette=colors, alpha=0.9, edgecolor=('white'),
linewidth=4)
axis.set_ylabel('count', fontsize=12)
axis.set_xlabel('target', fontsize=12)
axis.grid(b=True, which='major', color='grey', linewidth=0.2)
plt.title('Target count', fontsize=15)
plt.show()
target_0 = len(dataframe[dataframe.target == 0])
target_1 = len(dataframe[dataframe.target == 1])

print("Percentage of negative Heart Disease: {:.2f}%".format((target_0 /
(len(dataframe.target))*100)))
print("Percentage of  positive Heart Disease: {:.2f}%".format((target_1 /
(len(dataframe.target))*100)))
dataframe.target.value_counts()

# In[380]:
# Creating correlation metrix
plt.rcParams['figure.figsize'] = 15, 15
plt.style.use('dark_background')
plt.matshow(dataframe.corr())
plt.yticks(np.arange(dataframe.shape[1]), dataframe.columns)
plt.xticks(np.arange(dataframe.shape[1]), dataframe.columns)
plt.colorbar()

# In[381]:
# Correlation matrix
plt.style.use('dark_background')
f, (axis1, axis2) = plt.subplots(1,2,figsize =(15, 8))
corr = dataframe.corr()
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
heatmaps’ = dicot(linewidths=0.1)
sns.heatmap((dataframe[dataframe['target'] ==1]).corr(), vmax = .8, square=True, ax = axis1, cmap =
'YlGnBu', mask=mask, **heatmapkws);
```

```
sns.heatmap((dataframe[dataframe['target'] ==0]).corr(), vmax = .8, square=True, ax = axis2, cmap =
'afmhot', mask=mask,**heatmapkws);
axis1.set_title('Healthy Chart', fontsize=14)
axis2.set_title('Disease Chart', fontsize=14)
plt.show()

# In[382]:
dataframe.hist()
plt.style.use('dark_background')

# In[383]:
# Create another figure to expose max heart rate for age
plt.figure(figsize=(10, 8))
# Scatter with negative and postivie examples

plt.scatter(dataframe.age[dataframe.target==0],
            dataframe.thalach[dataframe.target==0],
            c="lightblue")

plt.scatter(dataframe.age[dataframe.target==1],
            dataframe.thalach[dataframe.target==1],
            c="salmon")
plt.title("Heart Disease in function in range of Age and Max Heart Rate")
plt.xlabel("Age")
plt.ylabel("Max Heart Rate")
plt.legend(["positive", "Negative"]);

# In[384]:
# Filtering values
categorical_values = []
continous_values = []
for column in dataframe.columns:
    if len(dataframe[column].unique()) <= 10:
        categorical_values.append(column)
    else:
        continous_values.append(column)

# In[385]:
# Getting categorical values from the dataset
categorical_values

# In[386]:
# Creating dummy columns for categorical values
categorical_values.remove('target')
dataset = pd.get_dummies(dataframe, columns = categorical_values)
dataset.head()

# In[387]:
# Scalling provied columns
from sklearn.preprocessing import StandardScaler
ssc = StandardScaler()
columns_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
dataset[columns_to_scale] = ssc.fit_transform(dataset[columns_to_scale])

# In[388]:
dataset.head()
# In[389]:
# Applying machine learning algorithm
# Data processing modeling and metrics
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score, f1_score
def print_score(clf, X_trainingdataset, y_trainingdataset, X_testingdataset, y_testingdataset,
train=True):
    if train:
        pred = clf.predict(X_trainingdataset)
        print("Trained Result:\n**********************************************")
        print(f"Accuracy Score: {accuracy_score(y_trainingdataset, pred) * 100:.2f}%")
        print("_______________________________________________")
        print("Classification Report for trained dataset:", end='')
        print(f"\tPrecision Score: {precision_score(y_trainingdataset, pred) * 100:.2f}%")
        print(f"\t\t\t\t\t\tRecall Score: {recall_score(y_trainingdataset, pred) * 100:.2f}%")
        print(f"\t\t\t\t\t\tF1 score: {f1_score(y_trainingdataset, pred) * 100:.2f}%")
        print("_______________________________________________")
        print(f"Confusion Matrix: \n {confusion_matrix(y_trainingdataset, pred)}\n")
    elif train==False:
        pred = clf.predict(X_testingdataset)
        print("\n\n\nTest Result:\n*******************************************")
        print(f"Accuracy Score: {accuracy_score(y_testingdataset, pred) * 100:.2f}%")
        print("_______________________________________________")
        print("Classification Report for test dataset:", end=' ')
        print(f"\tPrecision Score: {precision_score(y_testingdataset, pred) * 100:.2f}%")
        print(f"\t\t\t\t\t\tRecall Score: {recall_score(y_testingdataset, pred) * 100:.2f}%")
        print(f"\t\t\t\t\t\tF1 score: {f1_score(y_testingdataset, pred) * 100:.2f}%")
        print("_______________________________________________")
        print(f"Confusion Matrix: \n {confusion_matrix(y_testingdataset, pred)}\n")




# In[390]:
#Split dataset into 2 separte datasets
from sklearn.model_selection import train_test_split
X = dataset.drop('target', axis=1)
y = dataset.target
X_trainingdataset, X_testingdataset, y_trainingdataset, y_testingdataset = train_test_split(X, y, 
test_size=0.3, random_state=42)

# In[391]:
# Apply Random Forest Classifier Algorithm
from sklearn.ensemble import RandomForestClassifier
random_forest = RandomForestClassifier(n_estimators=1000, random_state=42)
random_forest.fit(X_trainingdataset, y_trainingdataset)
print_score(random_forest, X_trainingdataset, y_trainingdataset, X_testingdataset, y_testingdataset, 
train=True)
print_score(random_forest, X_trainingdataset, y_trainingdataset, X_testingdataset, y_testingdataset, 
train=False)

# In[392]:
# Finalize the training and testing accuracy in Random Forest
test_score = accuracy_score(y_testingdataset, random_forest.predict(X_testingdataset)) * 100
train_score = accuracy_score(y_trainingdataset, random_forest.predict(X_trainingdataset)) * 100
results_dataframe = pd.DataFrame(data=[["Random Forest Classifier", train_score, test_score]], 
                          columns=['Model', 'Training Accuracy %', 'Testing Accuracy %'])
results_dataframe

# In[393]:
feature_importance = pd.DataFrame({'feature': list(X_trainingdataset.columns),
                   'importance': random_forest.feature_importances_}).\
                    sort_values('importance', ascending = False)
# Display importance of feature
feature_importance.head()
```