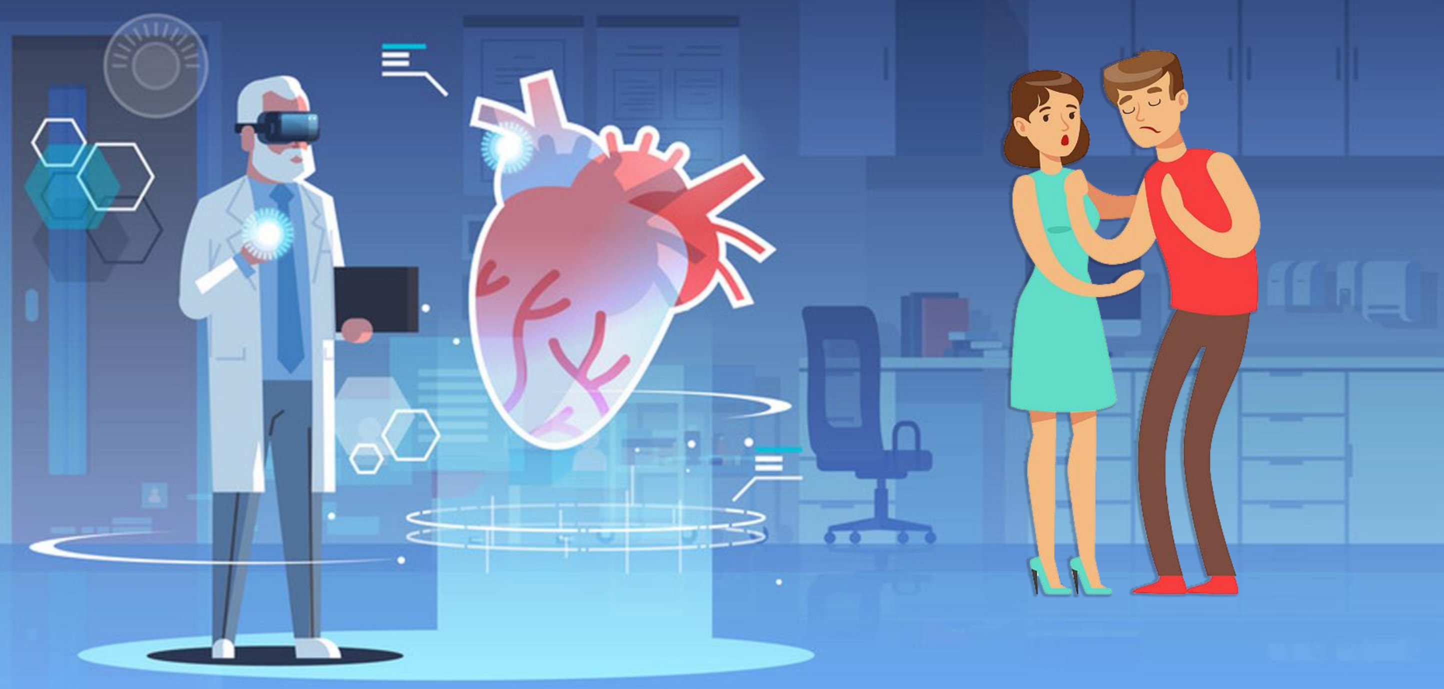
Sri Lanka Institute of Information Technology

Assignment 2

Year 4, Semester 1 (2021)

Heart Attack Risk Prediction Using Random Forest Classifier



Pathirana K.P.A.K.
IT18148046
it18148046@my.sliit.lk

Kariyapperuma.K.A.D.R.L.
IT18121766
it18121766@my.sliit.lk

Amarasinghe R.M.G.H
IT18172560
it18172560@my.sliit.lk

Machine Learning - IT4060

B.Sc. (Hons) in Information Technology Specialized in Software Engineering

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1. Description

Studies have showed that heart disease is one of the major causes of death throughout the world. As it is a difficult task which demands expertise and higher knowledge for prediction it cannot be easily predicted by the medical specialists. Procedures such as bypass surgery and angioplasty can help blood and oxygen to flow more easily to the heart, but still the arteries will remain damaged, which means it is still more likely to have a heart attack. A computerized system in medical diagnosis would enhance medical efficiency and reduce costs in prevention from this disease. In this fast-moving world people want to live a very luxurious life and to live a comfortable life, therefore in this rat race people forget to take care of themselves, because of this the food habits change and entire lifestyle change. In this type of lifestyle, people are more tensed to have blood pressure, sugar at a very young age. As a result of all this small negligence it will lead to a major threat called heart attack.

Risk factors are conditions or habits that make a person more likely to develop a heart disease. They can also increase the probabilities that an existing disease will get worse. Important risk factors for heart diseases that can be controlled are high blood pressure, high blood cholesterol, physical inactivity, overweight, and diabetes. Recent research indicates that more than 95 percent of patients those who die from heart attacks have at least one of these general risk factors.

Much research has been conducted in attempt to pinpoint the most powerful factors of heart disease as well as accurately predict the overall risk. Heart Attack is even highlighted as a silent killer which leads to the death of the person without noticeable symptoms. Most of the heart patients are treated for the heart diseases but they are not well controlled. The early diagnosis of heart disease plays a vital role in making changes to lifestyle of high-risk patients and in turn to reduce the complications that occur. Overall system aims to calculate risk of future heart attack by analyzing data of heart patients which classifies whether they have the risk of a heart attack or not using machine-learning algorithms [1].

1.2. Problem Addressed

Heart is a one of the most important organs of the human body. Over the last decade heart attack is the main reason for death in the world. The main reason for this is the busy lifestyle of the humans. Being very busy is avoiding good healthy habits of the humans. In Sri Lanka during the past 40 years, circulatory diseases including heart diseases and stroke have been the leading cause of death. It has increased from 3% in 1945 to 24% by 2018. According to the national representative study in Sri Lanka which has been conducted throughout the country has revealed that the mean age of heart disease victims was 58.2 years and highest number of heart patients were males (61.8%), because of smoking and excess of alcohol [2].

At the same time, several other medical conditions and life choices such as lack of exercise, high mental stress and fast-food habits can also put people at high risk for heart disease, including diabetes, overweight and obesity, unhealthy diet, and physical inactivity. Most of the patients are treated for the disease but the conditions have not been properly controlled, so there will be a risk of having it again. It is not possible to monitor patients every day in all cases correctly and consultation of a patient for 24 hours by a doctor is not available since it requires more wisdom, time, and expertise. Non-availability of medical diagnosing tools and medical experts specifically in undeveloped countries like Sri Lanka, diagnosis and cure of heart disease are very complex. So, a reasonable and accurate risk prediction of heart related diseases and guidance for keeping a healthy heart is necessary for a heart patient.

2. Data Collection

2.1. Dataset

The dataset required for this assignment was obtained from a **Kaggle** website. **Heart Attack Analysis & Prediction Dataset** was used for heart attack classification. It is the dataset at the top of most popular dataset in Kaggle. Dataset is containing CSV format (**heart.csv**). This data set is created by **Rashik Rahman**. This is a dataset related to the health care sector [4].

Dataset available here:
https://www.kaggle.com/rashikrahmanpritom/heart-attack-analysis-prediction-dataset

2.2. Description of Attributes

The description of each attribute is mentioned below. Three hundred three (303) instances with fourteen attributes (14) having the shape of (303 * 14) [4].

#	Feature	Description
1	**age**	age in years
2	**sex**	sex 1 = male 0 = female
3	**cp**	chest pain type Value 1: typical angina Value 2: atypical angina Value 3: non-anginal pain Value 4: asymptomatic
4	**trtbps**	resting blood pressure (in mm Hg on admission to the hospital)
5	**chol**	serum cholestoral in mg/dl
6	**fbs**	(fasting blood sugar > 120 mg/dl) 1 = true 0 = false
7	**restecg**	resting electrocardiographic results Value 0: normal Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV) Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria
8	**thalach**	maximum heart rate achieved
9	**exng**	exercise induced angina 1 = yes 0 = no
10	**oldpeak**	ST depression induced by exercise relative to rest
11	**slp**	the slope of the peak exercise ST segment Value 1: upsloping Value 2: flat Value 3: downsloping
12	**caa**	number of major vessels (0-3) colored by flourosopy
13	**thall**	thal rate normal =3 fixed defect =6 reversable defect = 7
14	**output**	target variable 0 = No Heart Attack Risk 1 = Heart Attack Risk

Table 1 - Description of Attributes

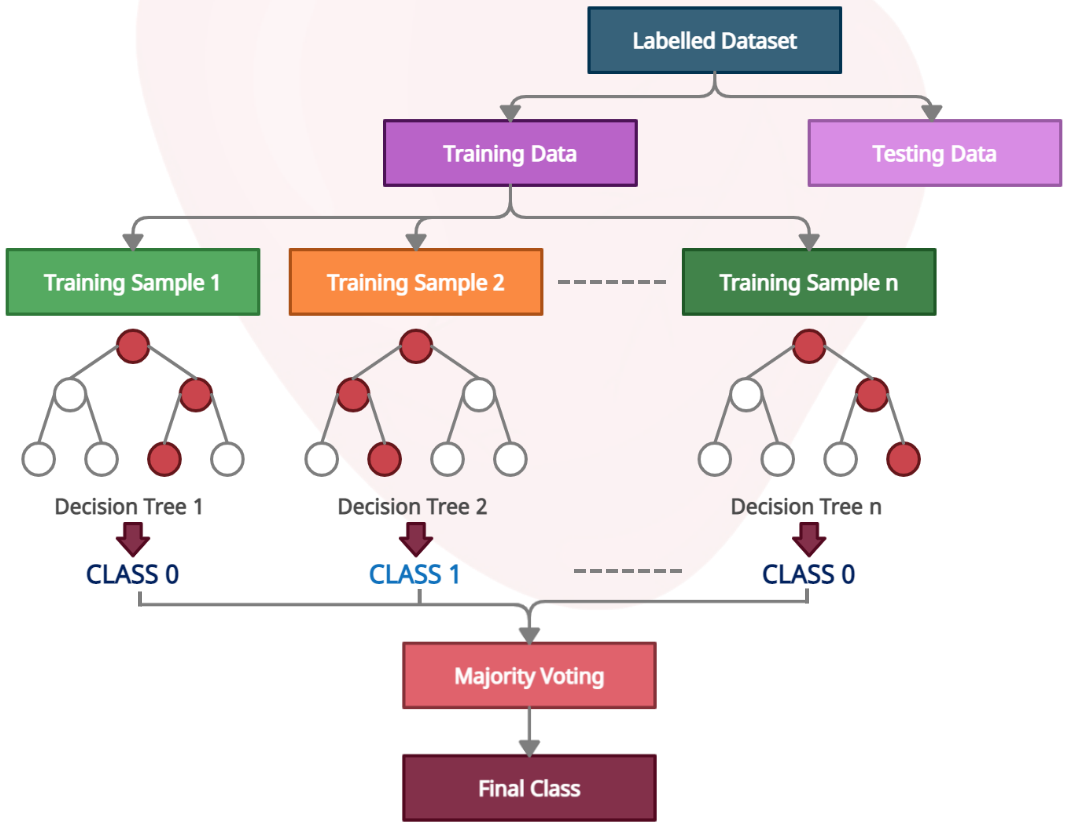
3. Methodology

3.1. Machine learning algorithm selection

To predict the presence of the heat attack, a classifier is needed to reduce the overfitting. So Random forest classification algorithm is suitable for that as it will obtain results from many decision trees. And Random forest can be used for regression problems and classification problems. This is binary classification problem, and those two classes are ***Heart Attack Risk***, ***No Heart Attack Risk***. Random forest classification can deal with the missing values. As it is very speedy it will provide a high accuracy for the classification problem [3].

3.2. Random Forest

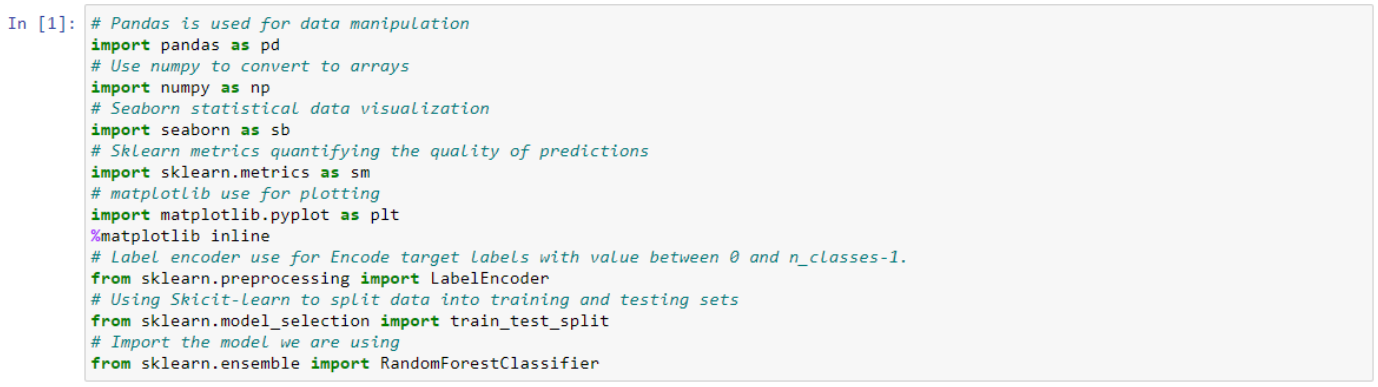
Random Forest is a supervised learning algorithm. In this algorithm most of the time train with the bagging method. Random forest makes many decision trees and merge them together to get the more accuracy prediction. Random forest is a flexible, easy to use that generates, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because of its simplicity and diversity (it can be used for both classification and regression problems). In this project it explains how the random forest algorithm works, how it differs from other algorithms and how we use it. Form the attributes in the data set it will select the subset of the attributes randomly to split the node [5]. In Random forest it is easy to predict the matching attributes. One big advantage of random forest is that it can be used for both classification and regression problems, which form most current machine learning systems. classification is sometimes deemed the building block of machine learning.



4. Implementation

4.1. Importing Libraries and packages

First, we need to import the required libraries and packages to implement the prediction model. Pandas is a package for data analysis and manipulation. Pandas help to explore, clean, and process datasets. Numpy is a library that utilized for working with arrays. Seaborn is data visualization library. It uses for drawing desirable and informative statistical graphs. Sklearn metrics use for measuring the quality of predictions. Matplotlib use for plotting data. LabelEncoder is package for encoding the levels of categorical features into numeric values. train_test_split is tool for split dataset into random train and test subsets. RandomForestClassifier is the machine learning model we use for this prediction.

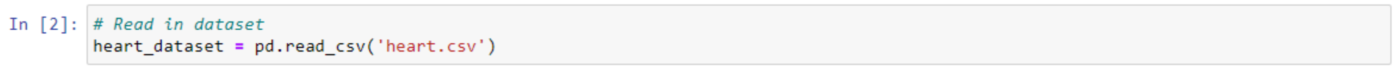


```
In [1]: # Pandas is used for data manipulation
        import pandas as pd
        # Use numpy to convert to arrays
        import numpy as np
        # Seaborn statistical data visualization
        import seaborn as sb
        # Sklearn metrics quantifying the quality of predictions
        import sklearn.metrics as sm
        # matplotlib use for plotting
        import matplotlib.pyplot as plt
        %matplotlib inline
        # Label encoder use for Encode target labels with value between 0 and n_classes-1.
        from sklearn.preprocessing import LabelEncoder
        # Using Skicit-learn to split data into training and testing sets
        from sklearn.model_selection import train_test_split
        # Import the model we are using
        from sklearn.ensemble import RandomForestClassifier
```

Figure 1 - Import Libraries and packages

4.2. Reading the dataset

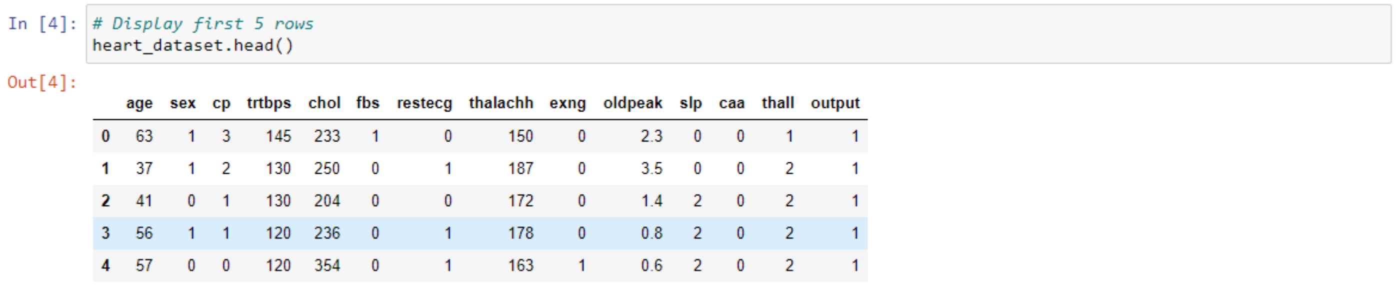
Heart Attack Analysis & Prediction Dataset is store in the heart.csv data source. Pandas library read_csv() method use for read the dataset. The read dataset was assigned to a variable called heart_dataset.



```
In [2]: # Read in dataset
        heart_dataset = pd.read_csv('heart.csv')
```

Figure 2 - Read the dataset

head() function used to get first 5 rows in dataset for quickly testing if right type of data in our dataset.



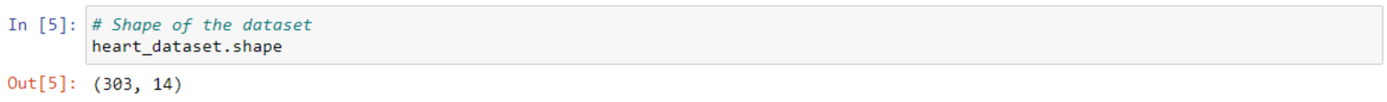
```
In [4]: # Display first 5 rows
        heart_dataset.head()
```

Out[4]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
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 3 - First five rows in the dataset

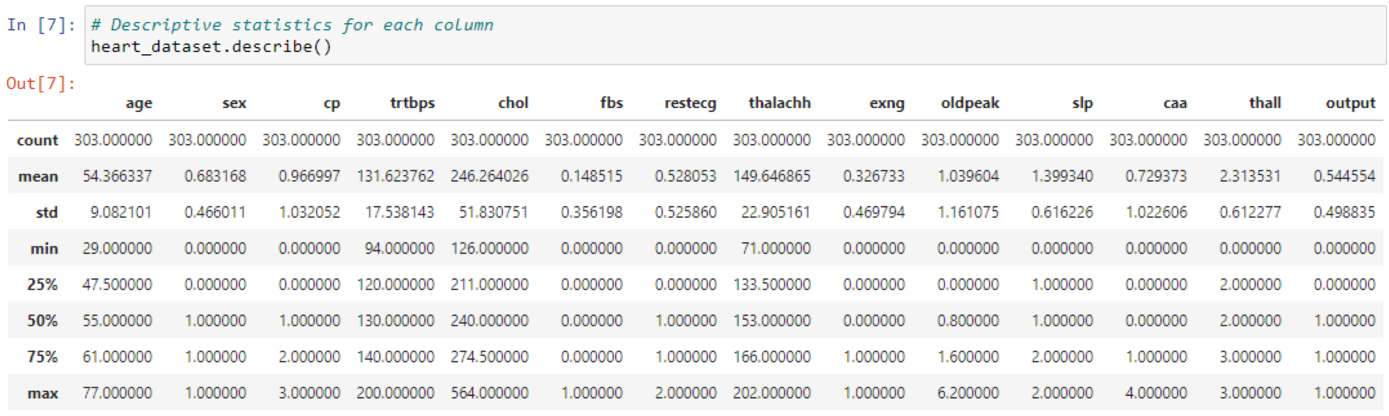
shape() return the shape of our dataset. Our dataset contains 303 rows and 14 columns.



```
In [5]: # Shape of the dataset
        heart_dataset.shape

Out[5]: (303, 14)
```

Figure 4 - Shape of the dataset

describe() function use for get summary of statistics in our dataset. It gives count, mean, standard deviation(std), and IQR values.



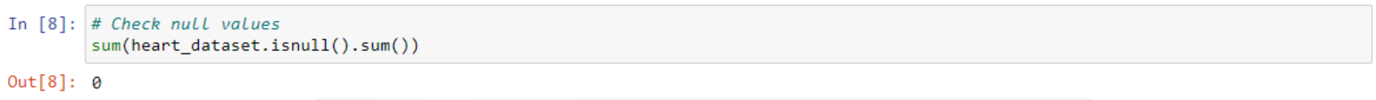
```
In [7]: # Descriptive statistics for each column
        heart_dataset.describe()

Out[7]:
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
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	303.000000	303.000000
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	2.313531	0.544554
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	0.612277	0.498835
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	0.000000	0.000000
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	2.000000	0.000000
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	2.000000	1.000000
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	3.000000	1.000000
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	3.000000	1.000000

Figure 5 - Summary of statistics in dataset

4.3. Data Preprocessing

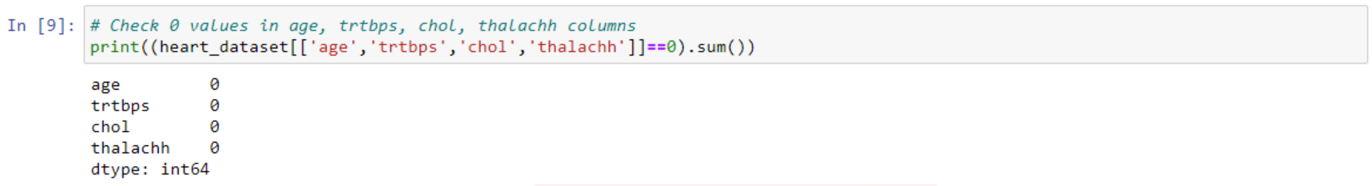
Data preprocessing is a very important step. Real-world datasets mostly contain missing values, noises, and maybe in an unusable format. We check null values in our dataset. There are no null values in our heart dataset.



```
In [8]: # Check null values
        sum(heart_dataset.isnull().sum())

Out[8]: 0
```

Figure 6 - Check null values in dataset

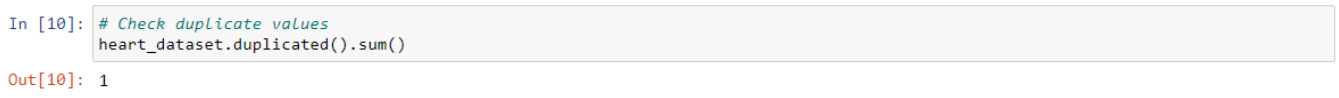
In our dataset cannot have 0 for age, trtbps, chol, thalachh columns. There are no 0 values in that columns.



```
In [9]: # Check 0 values in age, trtbps, chol, thalachh columns
        print((heart_dataset[['age','trtbps','chol','thalachh']]==0).sum())

        age         0
        trtbps      0
        chol        0
        thalachh    0
        dtype: int64
```

Figure 7 - Check 0 values in dataset

Our dataset has only one duplicate record. We delete that record to increase the accuracy of prediction.



```
In [10]: # Check duplicate values
         heart_dataset.duplicated().sum()

Out[10]: 1
```

Figure 8 - Check duplicate values

```
In [11]: # Display duplicate values
         heart_dataset[heart_dataset.duplicated()]

Out[11]:
           age  sex  cp  trtbps  chol  fbs  restecg  thalachh  exng  oldpeak  slp  caa  thall  output
      164   38    1   2     138   175    0        1       173     0      0.0    2    4      2       1
```

Figure 9 - Display duplicate values

```
In [12]: # Delete duplicate values
         heart_dataset.drop_duplicates(inplace=True)
```

Figure 10 - Delete duplicate values

After deleting that record, a dataset contains 302 rows and 14 columns.

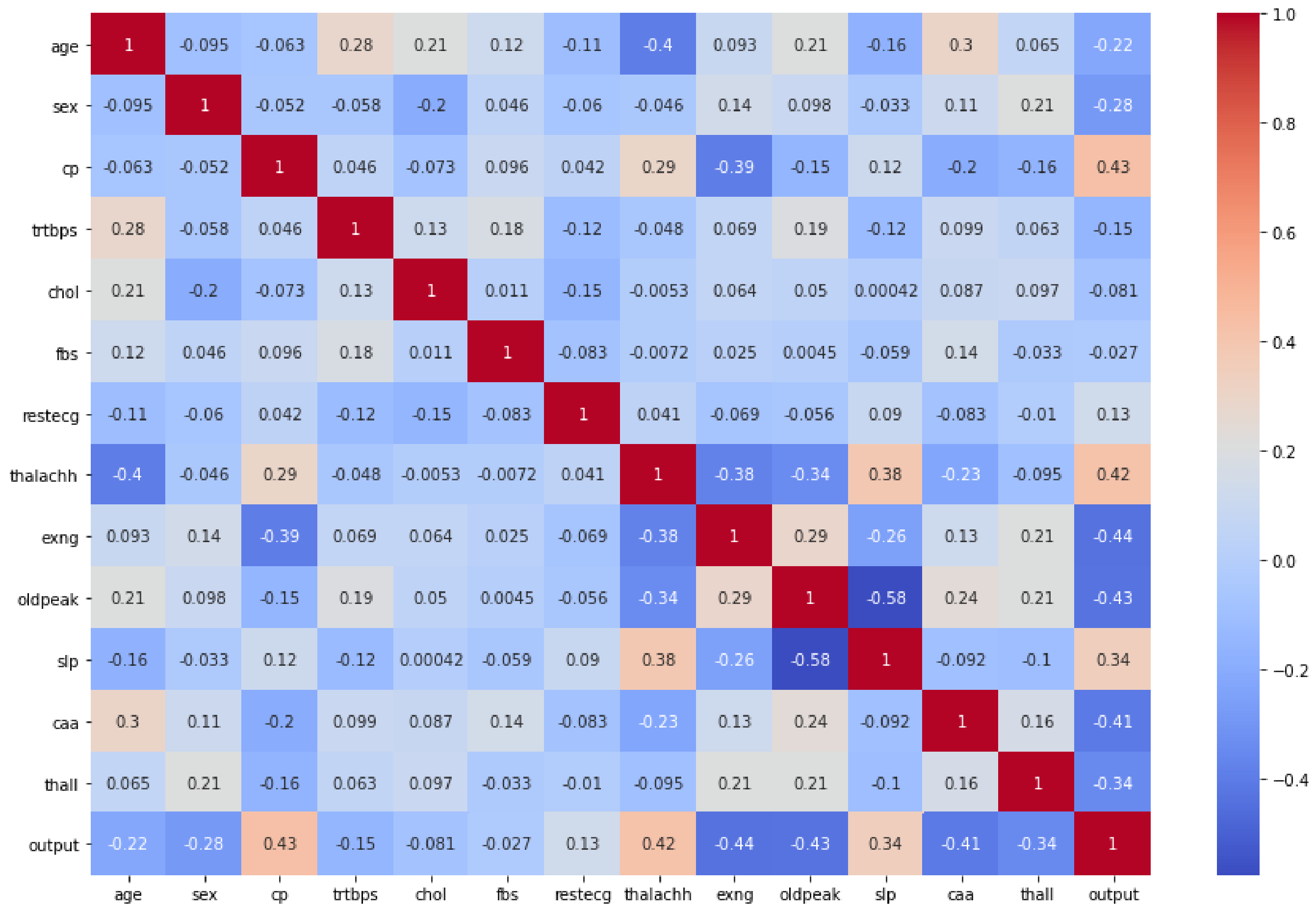
```
In [13]: # Shape of the dataset after data pre processing
         heart_dataset.shape

Out[13]: (302, 14)
```

Figure 11 - Shape of the dataset after data preprocessing

4.4. Correlation

This table visualize correlation coefficients between variables. corr() method use for get correlation in our dataset.

```
In [14]: # Display correlation coefficients between variables
         plt.figure(figsize=(15,10))
         sb.heatmap(heart_dataset.corr(),annot=True,cmap='coolwarm')

Out[14]: <AxesSubplot:>
```



	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
age	1	-0.095	-0.063	0.28	0.21	0.12	-0.11	-0.4	0.093	0.21	-0.16	0.3	0.065	-0.22
sex	-0.095	1	-0.052	-0.058	-0.2	0.046	-0.06	-0.046	0.14	0.098	-0.033	0.11	0.21	-0.28
cp	-0.063	-0.052	1	0.046	-0.073	0.096	0.042	0.29	-0.39	-0.15	0.12	-0.2	-0.16	0.43
trtbps	0.28	-0.058	0.046	1	0.13	0.18	-0.12	-0.048	0.069	0.19	-0.12	0.099	0.063	-0.15
chol	0.21	-0.2	-0.073	0.13	1	0.011	-0.15	-0.0053	0.064	0.05	0.00042	0.087	0.097	-0.081
fbs	0.12	0.046	0.096	0.18	0.011	1	-0.083	-0.0072	0.025	0.0045	-0.059	0.14	-0.033	-0.027
restecg	-0.11	-0.06	0.042	-0.12	-0.15	-0.083	1	0.041	-0.069	-0.056	0.09	-0.083	-0.01	0.13
thalachh	-0.4	-0.046	0.29	-0.048	-0.0053	-0.0072	0.041	1	-0.38	-0.34	0.38	-0.23	-0.095	0.42
exng	0.093	0.14	-0.39	0.069	0.064	0.025	-0.069	-0.38	1	0.29	-0.26	0.13	0.21	-0.44
oldpeak	0.21	0.098	-0.15	0.19	0.05	0.0045	-0.056	-0.34	0.29	1	-0.58	0.24	0.21	-0.43
slp	-0.16	-0.033	0.12	-0.12	0.00042	-0.059	0.09	0.38	-0.26	-0.58	1	-0.092	-0.1	0.34
caa	0.3	0.11	-0.2	0.099	0.087	0.14	-0.083	-0.23	0.13	0.24	-0.092	1	0.16	-0.41
thall	0.065	0.21	-0.16	0.063	0.097	-0.033	-0.01	-0.095	0.21	0.21	-0.1	0.16	1	-0.34
output	-0.22	-0.28	0.43	-0.15	-0.081	-0.027	0.13	0.42	-0.44	-0.43	0.34	-0.41	-0.34	1

Figure 12 - Correlation in dataset

4.5. Distribution of Data

These histograms visualize distribution of our dataset.

```
In [15]: # Display distribution of data
         heart_dataset.hist(figsize=(20,20))
         plt.show()
```

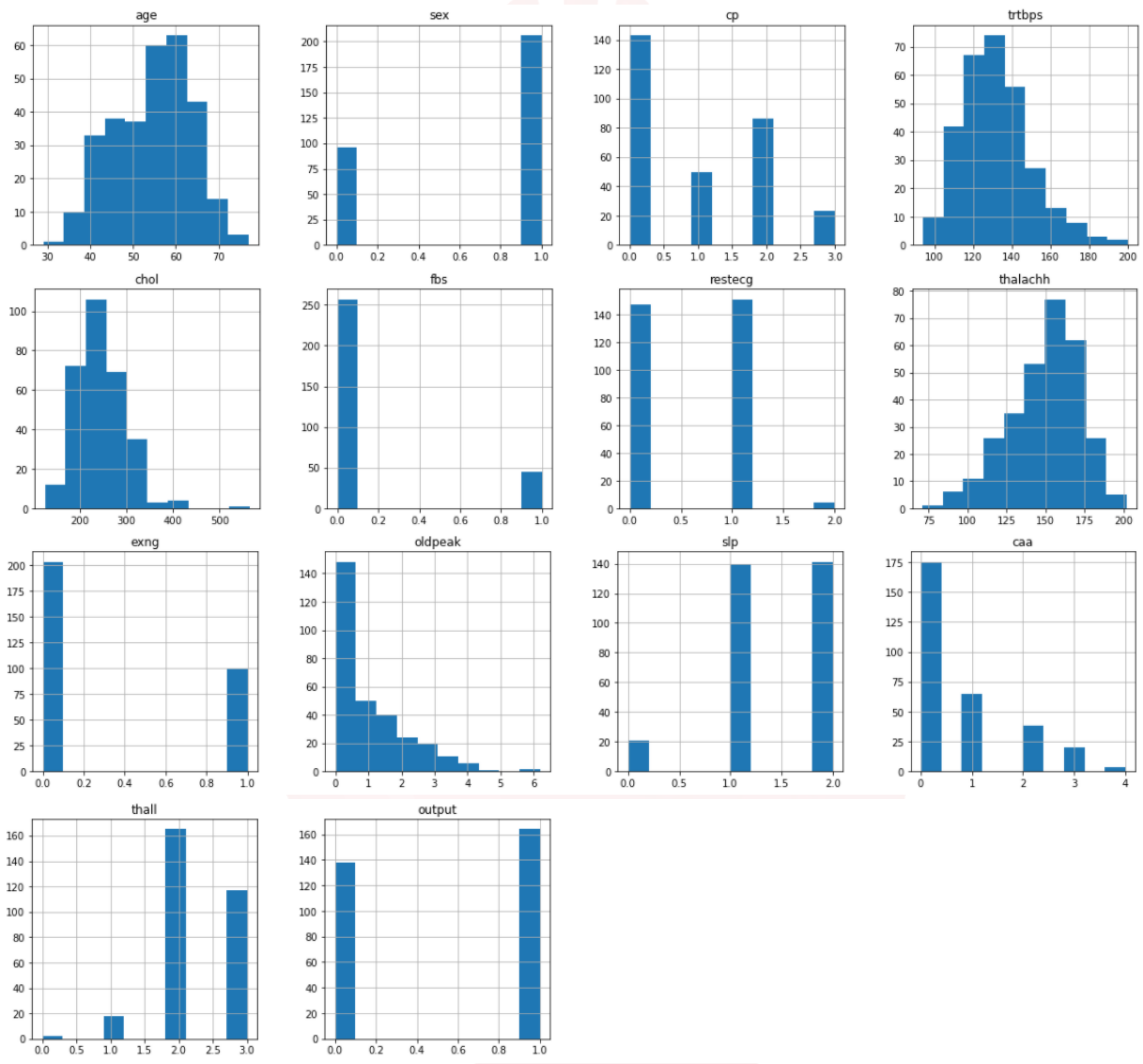
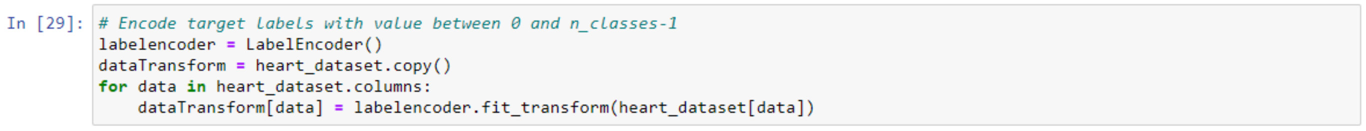


Figure 13 - Distribution of data

4.6. Process features and labels

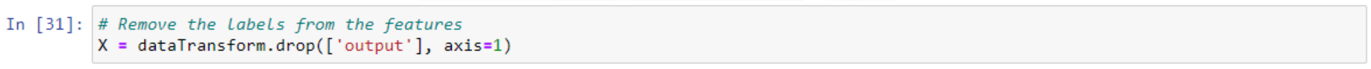
We use LabelEncoder() for encoding the levels of categorical features into numeric values and It Encode target labels with value between 0 and n_classes-1.



```
In [29]: # Encode target labels with value between 0 and n_classes-1
         labelencoder = LabelEncoder()
         dataTransform = heart_dataset.copy()
         for data in heart_dataset.columns:
             dataTransform[data] = labelencoder.fit_transform(heart_dataset[data])
```

Figure 14 - Encode target labels

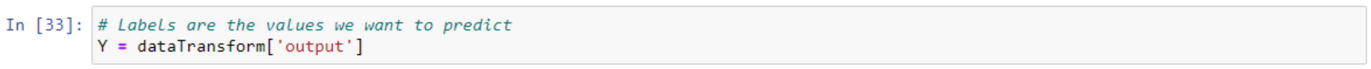
We remove labels and assign features data to X variable.



```
In [31]: # Remove the labels from the features
         X = dataTransform.drop(['output'], axis=1)
```

Figure 15 - Remove the labels from the features

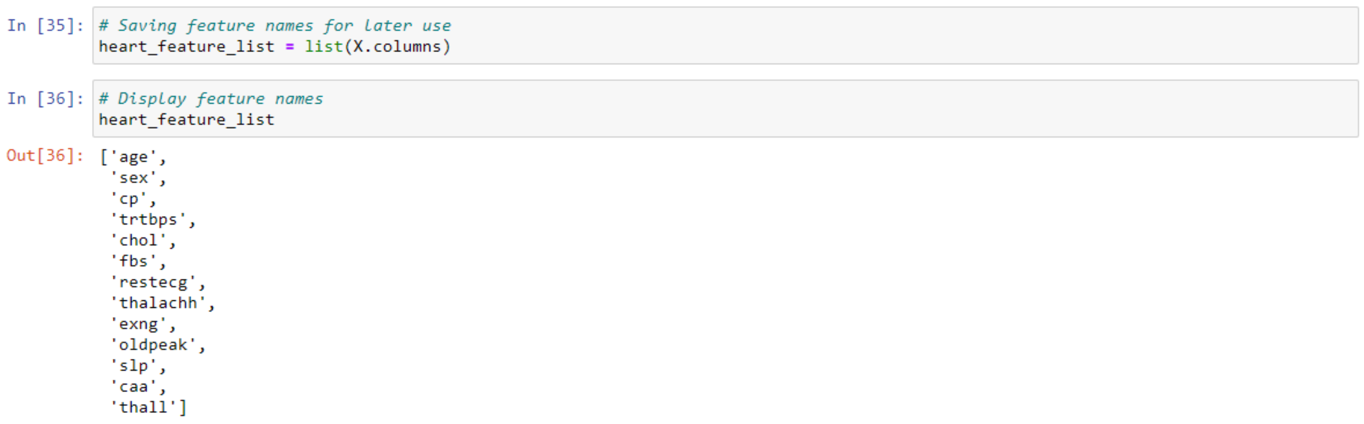
Assign labels values to Y variable.



```
In [33]: # Labels are the values we want to predict
         Y = dataTransform['output']
```

Figure 16 - Labels in dataset

Save feature names for later use.

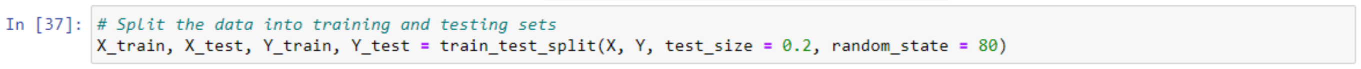


```
In [35]: # Saving feature names for later use
         heart_feature_list = list(X.columns)

In [36]: # Display feature names
         heart_feature_list

Out[36]: ['age',
          'sex',
          'cp',
          'trtbps',
          'chol',
          'fbs',
          'restecg',
          'thalachh',
          'exng',
          'oldpeak',
          'slp',
          'caa',
          'thall']
```

Figure 17 - Feature list

4.7. Split the data into training and testing sets

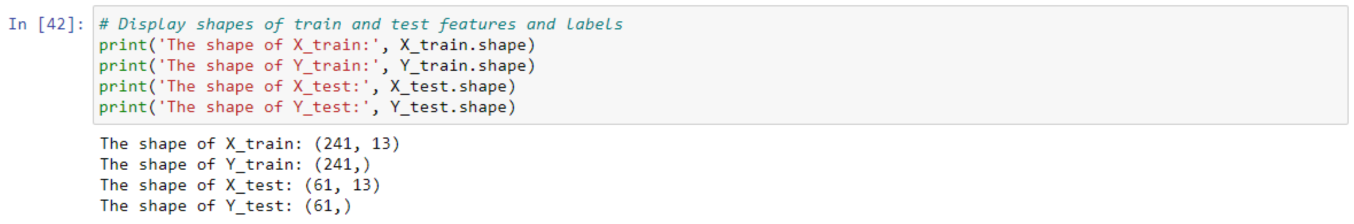
Sklearn train_test_split tool split dataset into random train and test subsets. We use 0.2 test size for proportion of the dataset to include in the test split. We use 80 as a random_state and it Controls the shuffling applied to the data before applying the split. We use default values for train_size, shuffle, stratify parameters.



```
In [37]: # Split the data into training and testing sets
         X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state = 80)
```

Figure 18 - Split the data into training and testing sets

Now our dataset splite 2 subsets as a train and test. Train subset contain 241 rows. Test subset contain 61 rows. X_train is training feature subset. Y_train is a labels use for testing model. X_test is feature subset use for test model. Y_test is labels for test model.

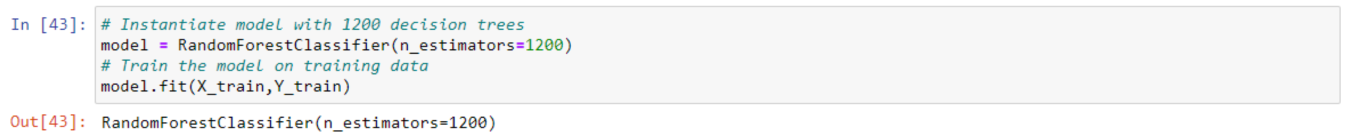


```
In [42]: # Display shapes of train and test features and labels
         print('The shape of X_train:', X_train.shape)
         print('The shape of Y_train:', Y_train.shape)
         print('The shape of X_test:', X_test.shape)
         print('The shape of Y_test:', Y_test.shape)

         The shape of X_train: (241, 13)
         The shape of Y_train: (241,)
         The shape of X_test: (61, 13)
         The shape of Y_test: (61,)
```

Figure 19 - Shapes of train and test features and labels

4.8. Prediction Model

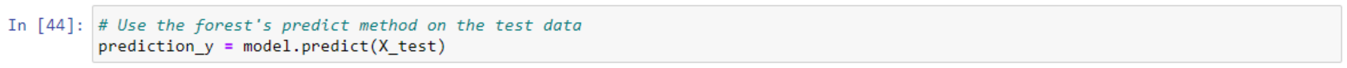
Instantiate model with 1200 decision trees. Then train our model using training dataset.



```
In [43]: # Instantiate model with 1200 decision trees
         model = RandomForestClassifier(n_estimators=1200)
         # Train the model on training data
         model.fit(X_train,Y_train)

Out[43]: RandomForestClassifier(n_estimators=1200)
```

Figure 20 - Instantiate and train the model

we get prediction result for test features subset.



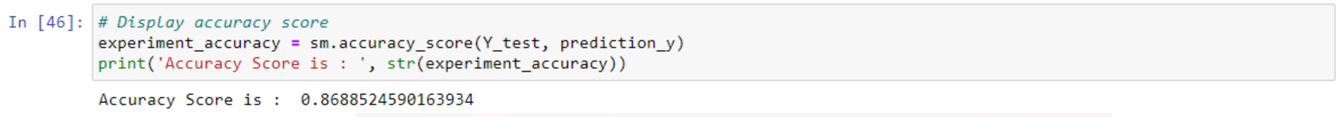
```
In [44]: # Use the forest's predict method on the test data
         prediction_y = model.predict(X_test)
```

Figure 21 - Predict on the test data

4.9. Accuracy score

Sklearn metrics accuracy_score use for computes accuracy of prediction model. This prediction model has 0.8688524590163934 accuracy. If $\hat{y}_i$ is the predicted value of the i -th sample and y_i is the corresponding true value, then the fraction of correct predictions over $n_{samples}$ is defined as [6],

$$\mathit{accuracy}(y,\hat{y}) = \frac{1}{n_{samples}}\sum_{i=0}^{n_{samples}-1} 1\,(\hat{y}_i = y_i)$$

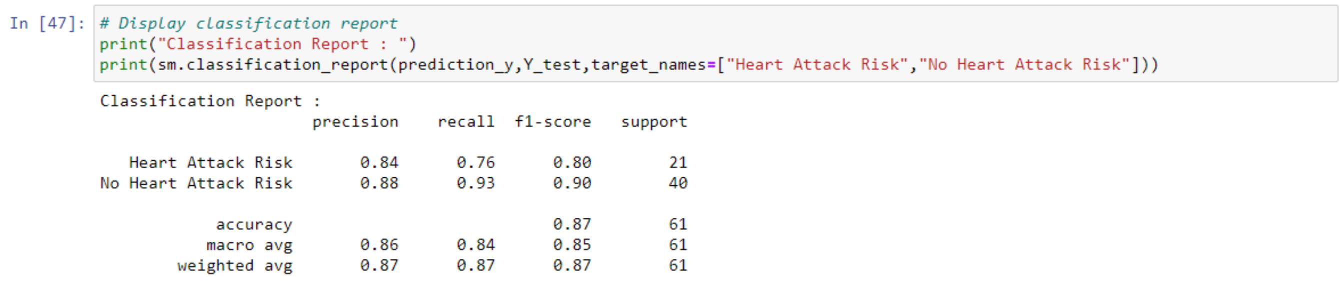


```
In [46]: # Display accuracy score
         experiment_accuracy = sm.accuracy_score(Y_test, prediction_y)
         print('Accuracy Score is : ', str(experiment_accuracy))

         Accuracy Score is :  0.8688524590163934
```

Figure 22 - Accuracy score

4.10. Classification Report

Classification report is a text report indicating the main classification metrics. this function use to calculate precision, recall, f1-score, and accuracy. According to our classification report Heart Attack Risk in the testing data set there were 21 data observation and 40 observations labeled as 1. (21 + 40 = 61) Testting data set have total 61 rows according to figure 19.



```
In [47]: # Display classification report
         print("Classification Report : ")
         print(sm.classification_report(prediction_y,Y_test,target_names=["Heart Attack Risk","No Heart Attack Risk"]))

         Classification Report :
                               precision    recall  f1-score   support

            Heart Attack Risk       0.84      0.76      0.80        21
         No Heart Attack Risk       0.88      0.93      0.90        40

                     accuracy                           0.87        61
                    macro avg       0.86      0.84      0.85        61
                 weighted avg       0.87      0.87      0.87        61
```

Figure 23 - Classification report

4.11. Confusion matrix

Compute confusion matrix to evaluate the accuracy of a classification by computing the confusion matrix with each row corresponding to the true class. confusing matrix is basically use for double check what are the actual class labels of the testing data set and what was the predicted labels of the testing data set, then we are going to create a matrix it shows what are the correctly classified observations and how much as mis classified. In X axis have true labels and Y axis have predicted values. when the true labels are 0 it has correctly classified them as 0 in 16 observations. Only 3 observations predict true class 0 as 1. likewise, when the true label is 1 it has correctly classified 37 times but one time it is identify the label as 0 in 5 times.

```
In [48]: # Display confusion matrix
         sb.set()
         get_ipython().run_line_magic('matplotlib','inline')
         confusionmt = sm.confusion_matrix(Y_test,prediction_y)
         sb.heatmap(confusionmt.T, square=True, annot=True, fmt='d', cbar=False)
         plt.xlabel('true class axis')
         plt.ylabel('predicted class axis')

Out[48]: Text(89.18, 0.5, 'predicted class axis')
```

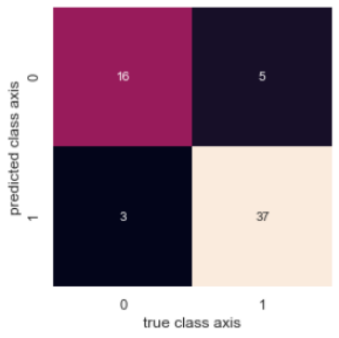


Figure 24 - Confusion matrix

4.12. Prediction result for new patient data

Finally, we predict heart attack risk for new input values using our trained prediction model.

```
In [49]: # Input values for prediction
         age = 50
         sex = 0
         cp = 3
         trtbps = 110
         chol = 264
         fbs = 1
         restecg = 1
         thalachh = 300
         exng = 0
         oldpeak = 1.2
         slp = 1
         caa = 0
         thall = 3

In [50]: # Predict heart attack risk using trained model
         predict_value = model.predict([[age,sex,cp,trtbps,chol,fbs,restecg,thalachh,exng,oldpeak,slp,caa,thall]])
         print('Predict Output : ', predict_value)

         Predict Output :  [1]
```

Figure 25 - Prediction result for new patient data

5. Critical analysis and Discussion

This application is very useful for hospitals. This application can easily predict the risk of heart attack in patients coming to the hospitals. This application helps a person to diagnose and treat a heart attack at an early stage before it occurs.

For future improvements we can suggest increasing the accuracy of the application by increasing the number of features and to make this application applicable for heart patients to detect their heart attack risk. We can also increase the number of category levels of the heart attack risk status to improve the application. Also, can suggest a heart attack risk status providing them in which stage they are and can predict the heart attack risk and give them proper guidance on meals, exercises according to the heart attack risk.

For furthermore development by using IOT technologies, can create a wearable device like watch to identify the real time patient's data and can train a model with those data to predict the heart attack risk. Also, it is better to alert the patient using the smart device wherever possible before a high-risk situation. A mobile app associated with the device can make it even easier. If further improved, we can connect the device to the hospital system and let the hospital doctor know before the patient has a heart attack. Then your personal doctor can prepare the necessary treatment before you have a heart attack. The doctor will then take you to your current location and take you to the hospital in an ambulance before you have a heart attack. Then when that person has a heart attack, his life risk is reduced because he is with the doctor.

REFERENSES

[1] Nichenametla, Rajesh & Maneesha, T. & Hafeez, Shaik & Krishna, Hari. (2018). “Prediction of Heart Disease Using Machine Learning Algorithms”. International Journal of Engineering and Technology (UAE). 7. 363-366. 10.14419/ijet.v7i2.32.15714.

[2] (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 10, No. 12, 2019 25 8 |” Cardiovascular Disease Diagnosis: A Machine Learning Interpretation Approach”Hossam Meshref Associate Professor, Computer Science Department College of Computers and Information Technology Taif University, Taif, Saudi Arabia

[3] “Prediction of Heart Disease by Clustering and Classification Techniques” Reetu Singh1, E. Rajesh2 1 Department of Computing Science and Engineering, Galgotias University, Greater Noida, India 2 Department of Computing Science and Engineering, Galgotias University, Greater Noida. Accepted: 16/May/2019, Published: 31/May/2019.

[4] Heart Attack Analysis & Prediction Dataset. [Online]. Available: https://www.kaggle.com/rashikrahmanpritom/heart-attack-analysis-prediction-dataset [Accessed: 02-April-2021].

[5] sklearn.ensemble.RandomForestClassifier. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html [Accessed: 10- April -2021].

[6] Metrics and scoring: quantifying the quality of predictions. [Online]. Available: https://scikit-learn.org/stable/modules/model_evaluation.html#accuracy-score [Accessed: 10- April -2021].

APENDIX

Source code

```
# Pandas is used for data manipulation
import pandas as pd
# Use numpy to convert to arrays
import numpy as np
# Seaborn statistical data visualization
import seaborn as sb
# Sklearn metrics quantifying the quality of predictions
import sklearn.metrics as sm
# matplotlib use for plotting
import matplotlib.pyplot as plt
%matplotlib inline
# Label encoder use for Encode target labels with value between 0 and n_classes-1.
from sklearn.preprocessing import LabelEncoder
# Using Skicit-learn to split data into training and testing sets
from sklearn.model_selection import train_test_split
# Import the model we are using
from sklearn.ensemble import RandomForestClassifier
```

```
# Read in dataset
heart_dataset = pd.read_csv('heart.csv')
```

```
# Display dataset
heart_dataset
```

```
# Display first 5 rows
heart_dataset.head()
```

```
# Shape of the dataset
heart_dataset.shape
```

```
# Columns of the dataset
heart_dataset.columns
```

```
# Descriptive statistics for each column
heart_dataset.describe()
```

```
# Check null values
sum(heart_dataset.isnull().sum())
```

```
# Check 0 values in age, trtbps, chol, thalachh columns
print((heart_dataset[['age','trtbps','chol','thalachh']]==0).sum())
```

```
# Check duplicate values
heart_dataset.duplicated().sum()
```

```
# Display duplicate values
heart_dataset[heart_dataset.duplicated()]
```

```
# Delete duplicate values
heart_dataset.drop_duplicates(inplace=True)
```

```
# Shape of the dataset after data pre processing
heart_dataset.shape
```

```
# Display correlation coefficients between variables
plt.figure(figsize=(15,10))
sb.heatmap(heart_dataset.corr(),annot=True,cmap='coolwarm')
```

```
# Display distribution of data
heart_dataset.hist(figsize=(20,20))
plt.show()
```

```
# Plot prevalence of heart attack by cp
sb.countplot(data=heart_dataset,hue=heart_dataset['output'],x=heart_dataset['cp'],
palette="mako")
plt.title('Prevalence of Heart attack by cp',fontsize=15)
```

```
# Plot prevalence of heart attack by age
plt.figure(figsize = (20, 8))
sb.countplot(data=heart_dataset,hue=heart_dataset['output'],x=heart_dataset['age'],
palette="mako")
plt.title('Prevalence of Heart attack by age',fontsize=15)
```

```
# Plot prevalence of heart attack by sex
sb.countplot(data=heart_dataset,hue=heart_dataset['output'],x=heart_dataset['sex'],
palette="mako")
plt.title('Prevalence of Heart attack by Sex',fontsize=15)
```

```
# Plot prevalence of heart attack by trtbps
plt.figure(figsize = (20, 8))
sb.countplot(data=heart_dataset,hue=heart_dataset['output'],x=heart_dataset['trtbps'],
palette="mako")
plt.title('Prevalence of Heart attack by trtbps',fontsize=15)

----------------------------------------------------------------------------------------------------------

# Plot prevalence of heart attack by fasting blood sugar > 120 mg/dl
sb.countplot(data=heart_dataset,hue=heart_dataset['output'],x=heart_dataset['fbs'],
palette="mako")
plt.title('Prevalence of Heart attack by fasting blood sugar > 120 mg/dl',fontsize=15)

----------------------------------------------------------------------------------------------------------

# Plot prevalence of heart attack by restecg
sb.countplot(data=heart_dataset,hue=heart_dataset['output'],x=heart_dataset['restecg'],
palette="mako")
plt.title('Prevalence of Heart attack by restecg',fontsize=15)

----------------------------------------------------------------------------------------------------------

# Plot prevalence of heart attack by exercise induced angina
sb.countplot(data=heart_dataset,hue=heart_dataset['output'],x=heart_dataset['exng'],
palette="mako")
plt.title('Prevalence of Heart attack by Exercise induced angina',fontsize=15)

----------------------------------------------------------------------------------------------------------

# Plot prevalence of heart attack by oldpeak
plt.figure(figsize = (20, 8))
sb.countplot(data=heart_dataset,hue=heart_dataset['output'],x=heart_dataset['oldpeak'],
palette="mako")
plt.title('Prevalence of Heart attack by oldpeak',fontsize=15)

----------------------------------------------------------------------------------------------------------
```

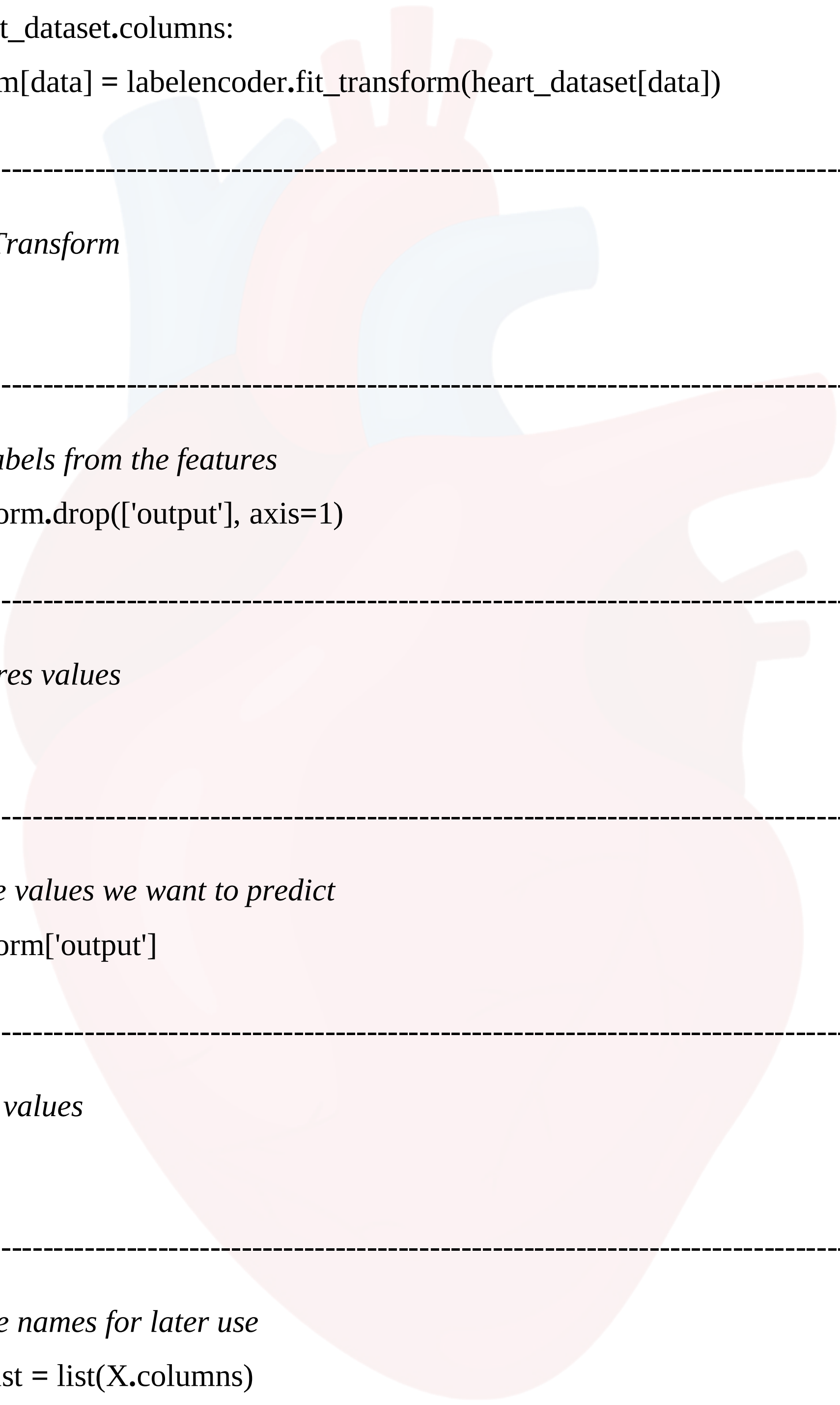
```
# Plot prevalence of heart attack by slp
sb.countplot(data=heart_dataset,hue=heart_dataset['output'],x=heart_dataset['slp'],
palette="mako")
plt.title('Prevalence of Heart attack by slp',fontsize=15)
```

```
# Plot prevalence of heart attack by number of major vessels
sb.countplot(data=heart_dataset,hue=heart_dataset['output'],x=heart_dataset['caa'],
palette="mako")
plt.title('Prevalence of Heart attack by number of major vessels',fontsize=15)
```

```
# Plot prevalence of heart attack by thall
sb.countplot(data=heart_dataset,hue=heart_dataset['output'],x=heart_dataset['thall'],
palette="mako")
plt.title('Prevalence of Heart attack by thall',fontsize=15)
```

```
# Plot distribution of chol and thalachh
plt.figure(figsize = (15, 8))
sb.scatterplot(data=heart_dataset,x='thalachh',y='chol' ,hue='output', palette="rocket")
```

```
# Plot distribution of chol and age
plt.figure(figsize = (15, 8))
sb.scatterplot(data=heart_dataset,x='age',y='chol' ,hue='output', palette="rocket")
```



```
# Encode target labels with value between 0 and n_classes-1
labelencoder = LabelEncoder()
dataTransform = heart_dataset.copy()
for data in heart_dataset.columns:
    dataTransform[data] = labelencoder.fit_transform(heart_dataset[data])
```

```
# Display dataTransform
dataTransform
```

```
# Remove the labels from the features
X = dataTransform.drop(['output'], axis=1)
```

```
# Display features values
X
```

```
# Labels are the values we want to predict
Y = dataTransform['output']
```

```
# Display label values
Y
```

```
# Saving feature names for later use
heart_feature_list = list(X.columns)
```

```
# Display feature names
heart_feature_list
```

```
# Split the data into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state = 80)
```

```
# Display train features values
X_train
```

```
# Display test features values
X_test
```

```
# Display train labels
Y_train
```

```
# Display test labels
Y_test
```

```
# Display shapes of train and test features and labels
print('The shape of X_train:', X_train.shape)
print('The shape of Y_train:', Y_train.shape)
print('The shape of X_test:', X_test.shape)
print('The shape of Y_test:', Y_test.shape)
```

```
# Instantiate model with 1200 decision trees
model = RandomForestClassifier(n_estimators=1200)
# Train the model on training data
model.fit(X_train,Y_train)
```

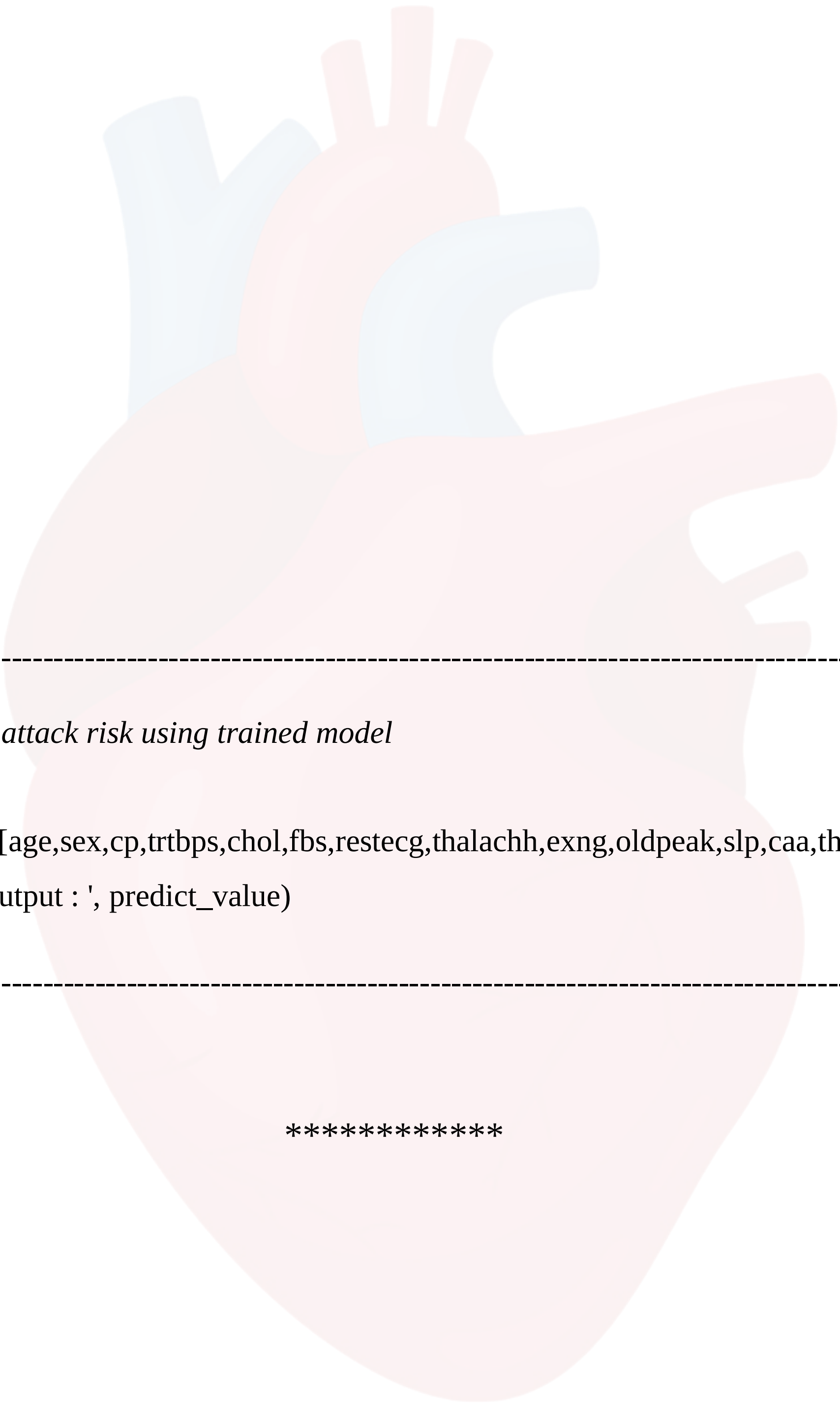
```
# Use the forest's predict method on the test data
prediction_y = model.predict(X_test)
```

```
# Display predict values
prediction_y
```

```
# Display accuracy score
experiment_accuracy = sm.accuracy_score(Y_test, prediction_y)
print('Accuracy Score is : ', str(experiment_accuracy))
```

```
# Display classification report
print("Classification Report : ")
print(sm.classification_report(prediction_y,Y_test,target_names=["Heart Attack Risk","No
Heart Attack Risk"]))
```

```
# Display confusion matrix
sb.set()
get_ipython().run_line_magic('matplotlib','inline')
confusionmt = sm.confusion_matrix(Y_test,prediction_y)
sb.heatmap(confusionmt.T, square=True, annot=True, fmt='d', cbar=False)
plt.xlabel('true class axis')
plt.ylabel('predicted class axis')
```



```
# Input values for prediction
age = 50
sex = 0
cp = 3
trtbps = 110
chol = 264
fbs = 1
restecg = 1
thalachh = 300
exng = 0
oldpeak = 1.2
slp = 1
caa = 0
thall = 3
```

```
# Predict heart attack risk using trained model
predict_value =
model.predict([[age,sex,cp,trtbps,chol,fbs,restecg,thalachh,exng,oldpeak,slp,caa,thall]])
print('Predict Output : ', predict_value)
```

****