Assignment 1, Heart Disease Classification Using Random Forest Algorithm

Aswathy Kuttisseril Jewel (C0813455)

Gayathri Ravinath (C0818959)

Prasanth Moothedath Padmakumar (C0796752)

Cloud Computing for Big Data, Lambton College

CBD 3335 – Data Mining and Analysis

Parisa Naraei

May 31, 2022

Abstract

We have created a prediction machine learning model using python to predict whether a person is suffering from heart disease or not. The dataset that we have used is Cleveland Heart disease dataset from UCI directory. Since the dataset contains processed data, there were no null values or missing values. We performed some exploratory Data Analysis, Random-forest Classification. Also, the model was evaluated finally using confusion metrics to understand how effectively will our model perform.

Problem Definition

The machine learning model we build using Random Forest classifier can be used by medical professional to predict weather a patient is suffering from heart disease or not. The model takes into consideration a range of clinical parameters including but not limited to blood pressure, cholesterol, blood sugar level, heart rate etc. We achieved a fair value of accuracy of 75 percentage which proves this model can be an effective tool considering its practical implementation

Data identification

The dataset used here is the Cleveland Heart Disease dataset taken from the UCI repository. The dataset consists of 303 individuals' data. There are 14 columns in the dataset which are Age, Sex, Chest-pain type, resting blood pressure, Serum Cholesterol, Fasting blood pressure, Resting ECG, Max heart rate achieved, Exercise induced Angina, St depression, St slope, Peak exercise ST segment, Number of major vessels colored by fluoroscopy, Thal, Diagnosis of heart disease.

Data acquisition and filtering

The following model is created in the Jupiter Notebook IDE (Integrated Development Environment). We must import libraries that assist with data analysis, manipulation, mathematical functions, and data visualization. Panda, a quick, powerful, flexible, and easy-to-use data analysis and manipulation tool, is one of the libraries. It can import data from a variety of file kinds, including comma-separated values (CSV), JSON (JavaScript Object Notation), SQL, and Microsoft Excel. As the first step in this process, we import the necessary libraries and

read the data set using the read_csv function provided by the pandas library. After reading the dataset to verify the dataset, we use the head function to display the first five rows of the data set

```
# Pandas library for data manipulation and analysis
# Numpy library for some standard mathematical functions
# Matplotlib library to visualize the data in the form of different plot
# Seaborn library for visualizing statistical graphics and work on top of Matplotlib
# sklearn for random forest classifier
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
# To display plot within the document
%matplotlib inline
```

Read the dataset from csv file

```
df = pd.read_csv('heart_cleveland_upload.csv')
```

```
# Display first 5 rows of the dataset using head function
df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	condition
0	69	1	0	160	234	1	2	131	0	0.1	1	1	0	0
1	69	0	0	140	239	0	0	151	0	1.8	0	2	0	0
2	66	0	0	150	226	0	0	114	0	2.6	2	0	0	0
3	65	1	0	138	282	1	2	174	0	1.4	1	1	0	1
4	64	1	0	110	211	0	2	144	1	1.8	1	0	0	0

Figure 1. Importing the libraries and reading the data set

The columns in our original dataset were seen in shortforms which reduced the readability for developers, hence the columns were renamed to more readable forms for ease of use.

```
df.columns
```

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'condition'],
      dtype='object')
```

```
# Rename columns to readable names
df.columns = ['Age', 'Sex', 'Chest_Pain_Type', 'Resting_Blood_Pressure', 'Cholesterol', 'Fasting_Blood_Sugar', 'Rest_Ecg', 'Max_Heart_Rate_Achieved',
              'Exercise_Induced_Angina', 'St_Depression', 'St_Slope', 'Num_major_vessels', 'Thalassemia', 'Heart_Disease']
df.columns
```

```
Index(['Age', 'Sex', 'Chest_Pain_Type', 'Resting_Blood_Pressure',
       'Cholesterol', 'Fasting_Blood_Sugar', 'Rest_Ecg',
       'Max_Heart_Rate_Achieved', 'Exercise_Induced_Angina', 'St_Depression',
       'St_Slope', 'Num_major_vessels', 'Thalassemia', 'Heart_Disease'],
      dtype='object')
```

Figure 2. Renaming column names

Data validation and cleaning

This step is all about cleaning the dataset. We checked for duplicated entries, missing values and null values using the duplicate, isna and isnull functions respectively. In general, missing data leads to unbalanced observations, skewed estimations, and, in extreme circumstances, erroneous conclusions. The dataset that we use here does not contain any irrelevant data since it is a processed dataset. The overall summary of the Dataset can be known using the info () function. The describe () function returns a statistical overview of the Dataset, including mean, count, minimum, maximum etc.

```
# Checking count of duplicate entries in the data set
df.duplicated().sum()
```

```
0
```

```
# Checking missing values in data set
df.isna().sum()
```

```
Age                         0
Sex                         0
Chest_Pain_Type             0
Resting_Blood_Pressure      0
Cholesterol                 0
Fasting_Blood_Sugar         0
Rest_Ecg                    0
Max_Heart_Rate_Achieved     0
Exercise_Induced_Angina     0
St_Depression               0
St_Slope                    0
Num_major_vessels           0
Thalassemia                 0
Heart_Disease               0
dtype: int64
```

```
# Checking null values in data set
df.isnull().sum()
```

```
Age                         0
Sex                         0
Chest_Pain_Type             0
Resting_Blood_Pressure      0
Cholesterol                 0
Fasting_Blood_Sugar         0
Rest_Ecg                    0
Max_Heart_Rate_Achieved     0
Exercise_Induced_Angina     0
St_Depression               0
St_Slope                    0
Num_major_vessels           0
Thalassemia                 0
Heart_Disease               0
dtype: int64
```

Figure 3. Checking data set for duplicate, missing and null values.

Exploratory Analysis

Age and heart disease comparison.

We plotted Age of the individual against their frequency for each of the target class. It was inferred that the age which people suffer heart disease the most is 58 followed by 57

```
sns.set_context("paper", font_scale = 2, rc = {"font.size": 20,"axes.titlesize": 25,"axes.labelsize": 20})
sns.catplot(kind = 'count', data = df, x = 'Age', hue = 'Heart_Disease', order = df['Age'].sort_values().unique(), heig
plt.title('Variation of Age for each target class')
plt.show()
```

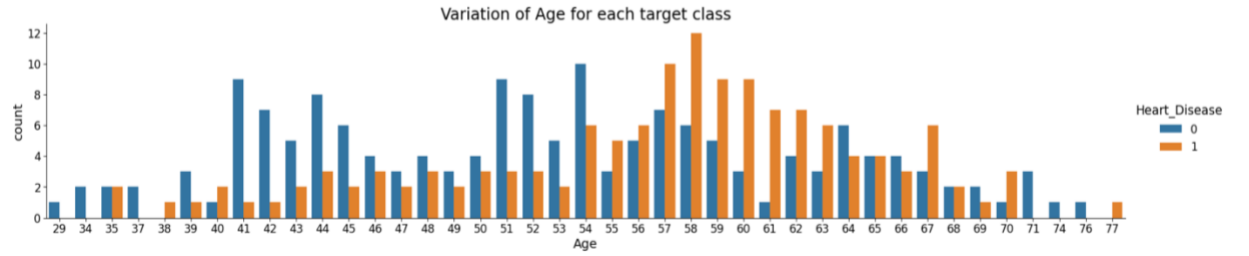

Figure 4. Bar graph for plotting Age vs heart disease frequency.

We tried to find out which age group was at most risk to heart disease. Different age groups present in the data set was found out using min, max and mean of age column. From the bar graph it can be inferred that older age group is more prone to heart disease.

```
Young = df[(df.Age>=29)&(df.Age<40)&(df.Heart_Disease==1)]
Middle = df[(df.Age>=40)&(df.Age<55)&(df.Heart_Disease==1)]
Elder = df[(df.Age>55)&(df.Heart_Disease==1)]

# Plot as bargraph to find out which age group is most affected to heart disease
plt.figure(figsize=(23,10))
sns.set_context('notebook',font_scale = 1.5)
sns.barplot(x=['young ages','middle ages','elderly ages'],y=[len(Young),len(Middle),len(Elder)])
plt.tight_layout()
```

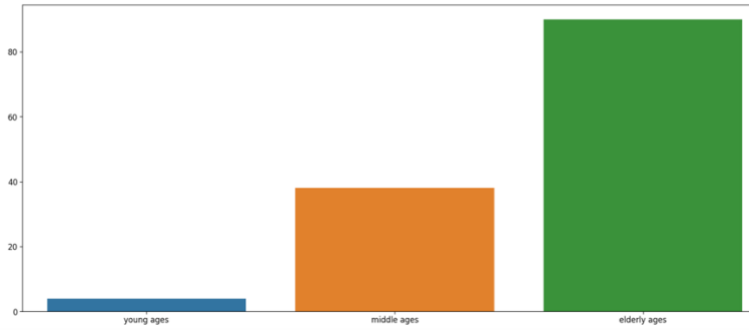

Figure 5. Bar graph for plotting Age group vs heart disease frequency.

Chest pain type and heart disease comparison

It was inferred that type 3 chest pain is the most common in heart disease patients. Type 3 chest pain is asymptomatic chest pain.

```
sns.countplot(data= df, x='Chest_Pain_Type',hue='Heart_Disease'
plt.title('Chest Pain Type v/s Heart_Disease\n')

Text(0.5, 1.0, 'Chest Pain Type v/s Heart_Disease\n')
```

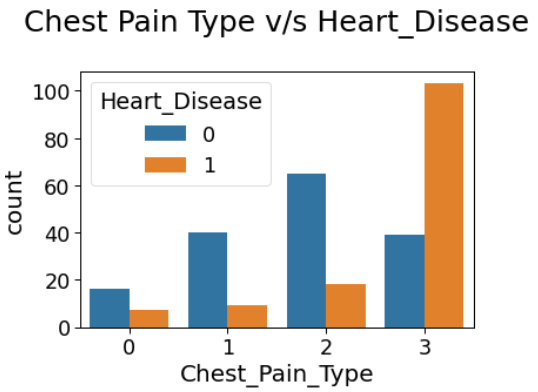

Figure 6. Bar graph for plotting Chest pain type vs heart disease frequency.

Thalassemia Types and heart disease frequency

Individuals with Thalassemia type 2 was the one with the most no of heart disease c

```
pd.crosstab(df['Thalassemia'],df['Heart_Disease']).plot(kind='bar')
plt.title("Heart Disease Frequency per Thalassemia_Types")
plt.xlabel("Thalassemia Types")
plt.ylabel("Amount")
plt.legend(['No disease','Disease'])
plt.xticks(rotation=0);
```

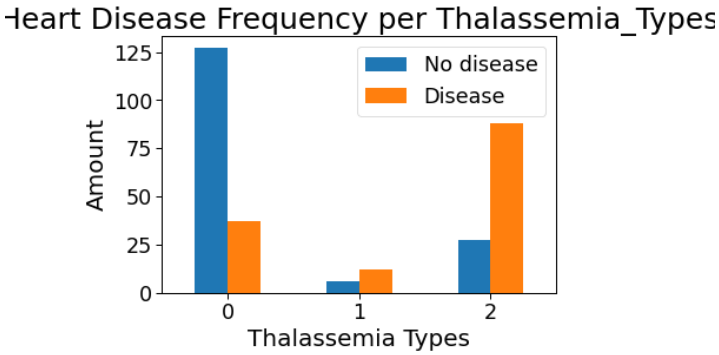

Figure 7. Bar graph for plotting Thalassemia type vs heart disease frequency.

ECG results and heart disease frequency

It was found that individuals with ECG result type 2 followed by type one has most chance of heart disease.

```
pd.crosstab(df['Rest_ecg'],df['Heart_Disease']).plot(kind='bar')
plt.title("Heart Disease Frequency per ECG Results")
plt.xlabel("ECG Result types")
plt.ylabel("Amount")
plt.legend(['No disease','Disease'])
plt.xticks(rotation=0);
```

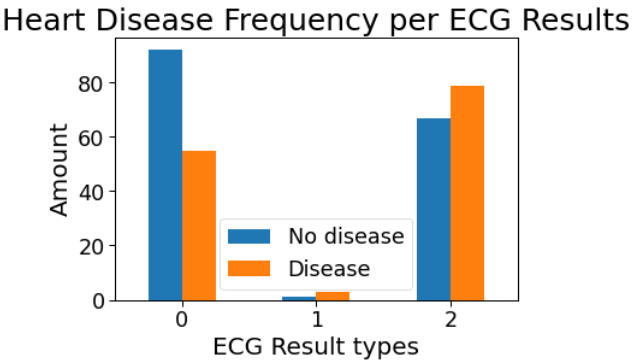

Figure 8. Bar graph for plotting ECG result type vs heart disease frequency.

Fasting blood sugar and heart disease frequency

It was found that for high and low blood sugar individuals the heart disease frequency was almost similar.

```
pd.crosstab(df['Heart_Disease'],df['Fasting_Blood_Sugar']).plot(kind="bar",figsize=(10,
plt.title("Heart Disease Frequency vs Fasting Blood Sugar")
plt.xlabel("0 = No Disease , 1 = Disease")
plt.ylabel("Amount")
plt.legend(["True","False"])
plt.xticks(rotation=0)
```

```
(array([0, 1]), [Text(0, 0, '0'), Text(1, 0, '1')])
```

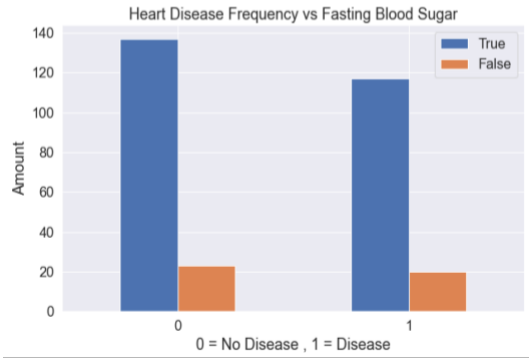

Figure 9. Bar graph for plotting Fasting blood sugar type vs heart disease frequency.

Modelling

Imports

Necessary packages for performing the modelling operation was imported

```
from sklearn.model_selection import train_test_split,cross_val_score
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.ensemble import RandomForestClassifier
```

Figure 10. Imports for modelling.

Train and Test Splitting

As the first part of modelling, we performed splitting of the data set using the train_test_split function the sklearn library. The splitting was done in such a way that 80 percentage of the data set was used for training and the rest 20 percentage was used for testing. Test size parameter for the train_test_split function was assigned a value of 0.2 to achieve the above-mentioned splitting of data set. The target feature in our data set which is 'heart_disease' column was dropped before splitting the model and applying the classification algorithm.

```
X=df.drop('Heart_Disease',axis=1)
Y=df['Heart_Disease']
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=42)
```

Figure 11. Train and test splitting.

Random Forest classifier

The machine learning classification model of choice was random forest classifier. It was performed using the bult in function called RandomForestClassifier. The model was executed on the training data set using the fit function The Prediction was performed using the predict

function and stored in Y_pred variable The accuracy score was printed using score function and it was found to be 0.75.

```
rf=RandomForestClassifier()
```

```
rf.fit(X_train,Y_train)
```

```
RandomForestClassifier()
```

Figure 12. Initializing classifier function and calling fit function.

Performance Evaluation of the model

As a starter to evaluate the classification model, We decided to compare the target feature value of Actual and Predicted Out of first 10 values 8 of them was found to be correct So it closely matches to the 75 % accuracy we calculated in the previous step.

```
# Comparing test and prediction for first 10 values
diffTable = pd.DataFrame({'Actual-Value': Y_test, 'Predicted-Value':Y_pred})
diffTable.head(10)
```

	Actual-Value	Predicted-Value
167	1	1
211	1	1
63	0	0
154	0	0
5	0	1
77	0	0
183	1	1
158	1	1
9	0	1
139	0	0

Figure 13. Actual vs predict value for first 10 values of testing data set

Confusion Matrix

To evaluate the performance of classification model we decided to use confusion matrix. The model predicted total of 60 individual for heart disease Out of those 60 cases, the classifier predicted "yes" 31 times, and "no" 29 times. In reality according the dataset, only 28 individuals have heart disease while 32 of the individuals were diagnosed as not having heart disease.

```
# Plotting Confusion matrix
sns.set(font_scale=1.5)
fig,ax=plt.subplots(figsize=(3,3))
ax=sns.heatmap(confusion_matrix(Y_test,Y_pred),annot=True,cbar=False)
plt.xlabel("True Label")
plt.ylabel("Predicted Label")
```

```
Text(-1.5, 0.5, 'Predicted Label')
```

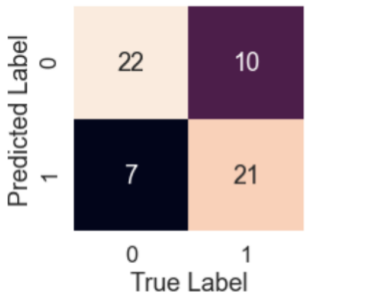

Figure 14. Confusion matrix.

Conclusion

The model gave an accuracy value of 75 % which is almost enough to take into consideration for implementing into practical use case. Calculating the chances of developing heart disease based on risk factors is difficult. However, utilizing Machine Learning, we will be able to detect whether the person has heart disease in no time.

Reference

Aman P. (2022, February 11). *Heart Disease Prediction using Machine Learning.* Analytics vidhya. https://www.analyticsvidhya.com/blog/2022/02/heart-disease-prediction-using-machine-learning/

Pulkit K. (2021, Aug 19). *Predicting Heart disease using Machine Learning.* Medium. https://medium.com/analytics-vidhya/predicting-heart-disease-using-machine-learning-ce10cfce41a6

Shubhankar R. (2019, Aug 10). *Heart Disease Prediction.* Towards Data Science. https://towardsdatascience.com/heart-disease-prediction-73468d630cfc

Hardik D. (2020, June 18). *Heart Disease UCI-Diagnosis & Prediction.* Towards Data Science. https://towardsdatascience.com/heart-disease-uci-diagnosis-prediction-b1943ee835a7