Machine Learning

IT4060

Assignment 2

Sri Lanka Institute of Information Technology

Team Members

- IT18219326 - A.D.S.Gunasekara
- IT19023656 - Imasha G.A.A
- IT19362854 - Abeygunawardana S.L
- IT19217796 - Ratnayake M.A.A.H

Table of Contents

1. 0 Address the Problem

Heart failures occur when the heart is unable to pump enough blood throughout the body. It is usually caused by a very weak or hard heart. It is also known as a heart attack. An estimated 26 million cases have been identified due to aging population, increased risk of cardiovascular risk factors, and improved risk of cardiovascular disease, while the incidence of millions more unrecognized heart attacks are increasing globally.

Heart attack is a complex clinical disease that occurs when the heart is unable to pump enough blood to meet the body's needs. It can be severe and appear quickly, or it can be a chronic, long-term illness. Symptoms include shortness of breath, cough, or shortness of breath, fatigue and lethargy, fluid retention with swelling of the legs and abdomen, and inability to work or exercise physically.

Infectious diseases such as Chagas and rheumatic heart disease; heart diseases such as coronary heart disease, valvular disease, congenital heart disease, pericardial disease and arrhythmias, including myocardial infarction; chronic lung disease; poor lifestyle choices such as high-salt diet, tobacco, alcohol or drug abuse; or failure to adhere to preventative medications are just a few of the possible causes.

People with heart disease can live longer and have a better quality of life if they are diagnosed and treated early.

Heart attacks can occur on the left or right side of the heart. Both sides of the heart may also fail simultaneously. Heart failure is also classified as diastolic or systolic.

- Left side heart failures

- Right side heart failures

- Diastolic heart failures

- Systolic heart failure

2.0 About Dataset

Cardiovascular diseases are the number one death cause of disease globally. It is around 17.9 annually (31%). Heart failure is a common serious condition of cardiovascular disease, and this dataset contains 11 features that help predict heart disease. This dataset is a combination of 5 different datasets for research purposes. The final combined dataset contains 918 observations with 11 features. [1]

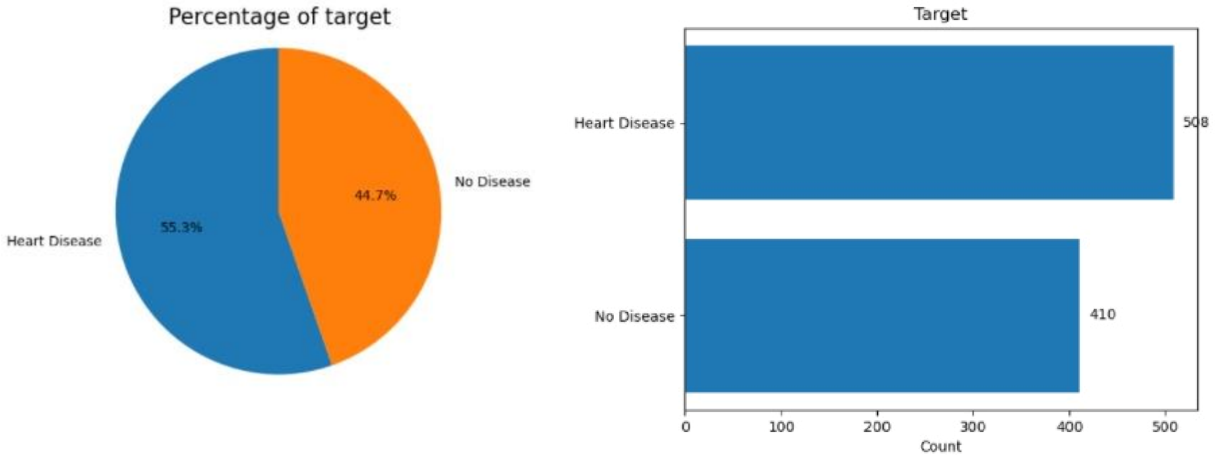
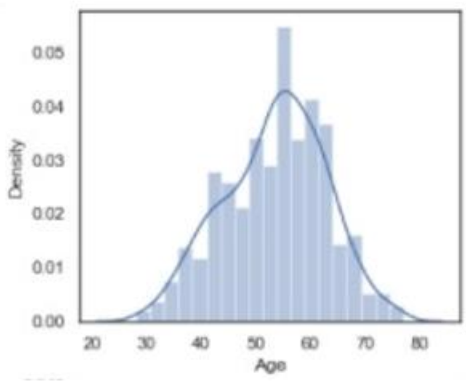


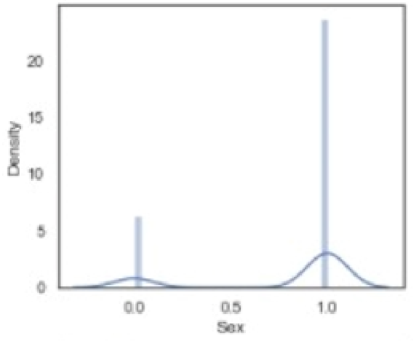
Figure 1 – Percentage of Target

2.1Describe the fields

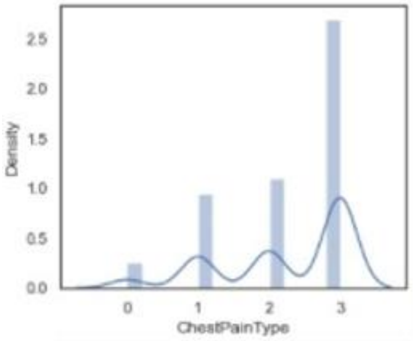
1. **Age:**
    - age of the patient [years]



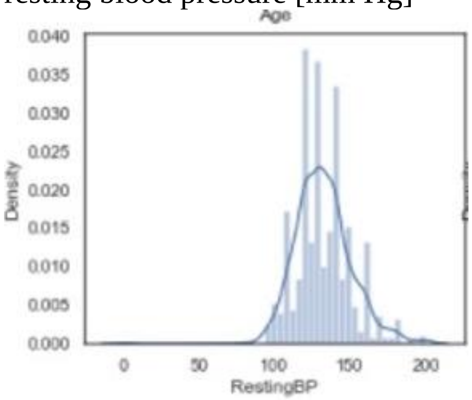
2. **Sex**:
    - sex of the patient [M: Male, F: Female]



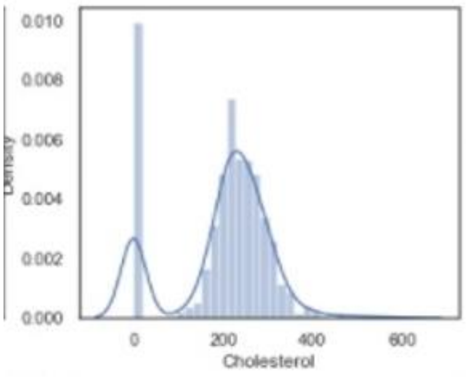
3. **ChestPainType**:
    - chest pain type [TA: Typical Angina, ATA: Atypical Angina, NAP: Non-Anginal Pain, ASY: Asymptomatic]



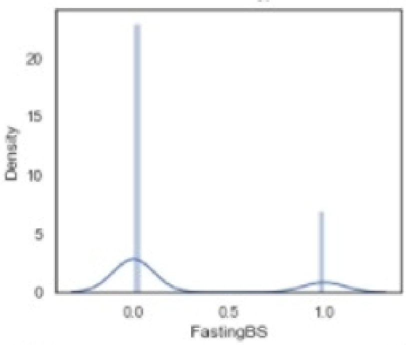
4. **RestingBP**:
    - resting blood pressure [mm Hg]



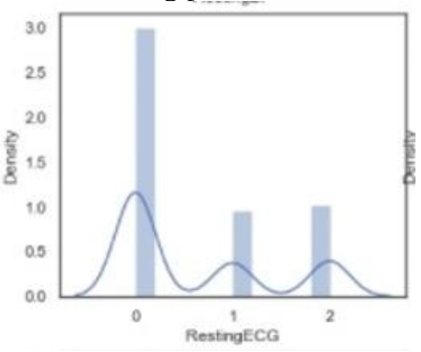
5. **Cholesterol**:
    - serum cholesterol [mm/dl]



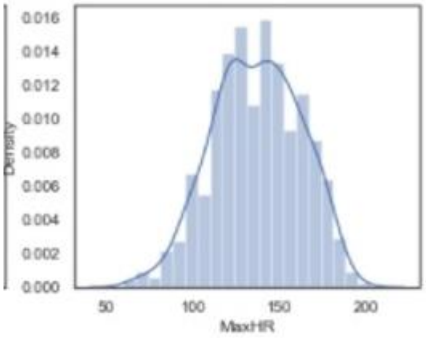
6. **FastingBS**:
    - fasting blood sugar [1: if FastingBS > 120 mg/dl, 0: otherwise]



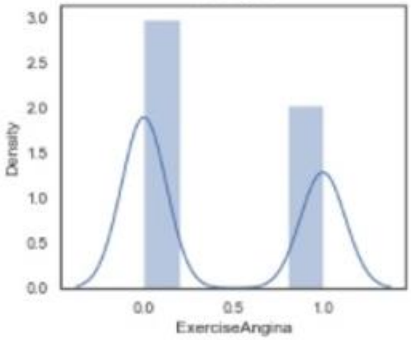
7. **RestingECG**:
    - resting electrocardiogram results [Normal: Normal, ST: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV), LVH: showing probable or definite left ventricular hypertrophy by Estes' criteria]



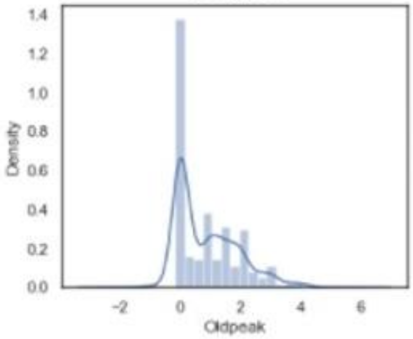
8. **MaxHR**:
    - maximum heart rate achieved [Numeric value between 60 and 202]



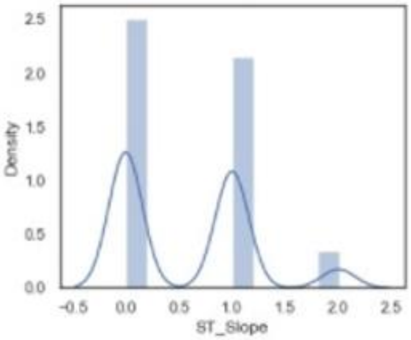
9. **ExerciseAngina**:
    - exercise-induced angina [Y: Yes, N: No]



10. **Oldpeak**:
    - oldpeak = ST [Numeric value measured in depression]



11. **ST_Slope**:
    - the slope of the peak exercise ST segment [Up: upsloping, Flat: flat, Down: downsloping]



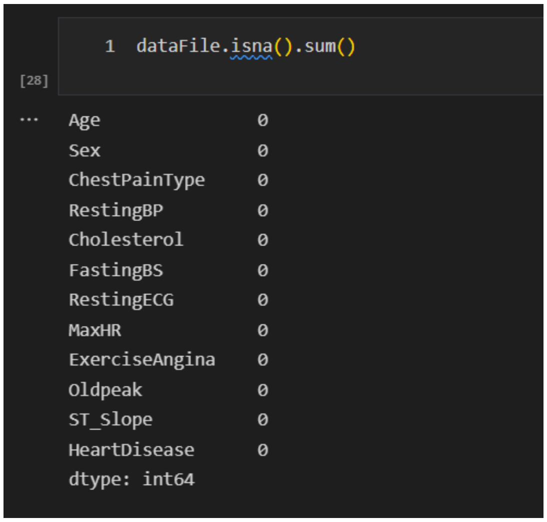
12. **HeartDisease**:

- output class [1: heart disease, 0: Normal]

3.0 Methodology

3.1 Data Preprocessing

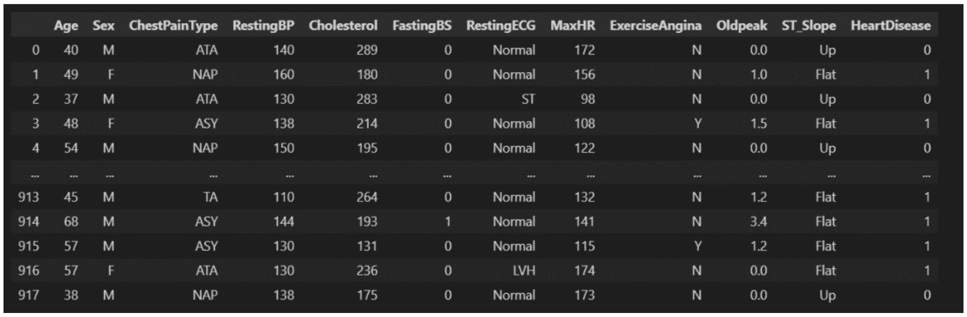
First checked the null values and there were no null values in the dataset.



```
dataFile.isna().sum()
```

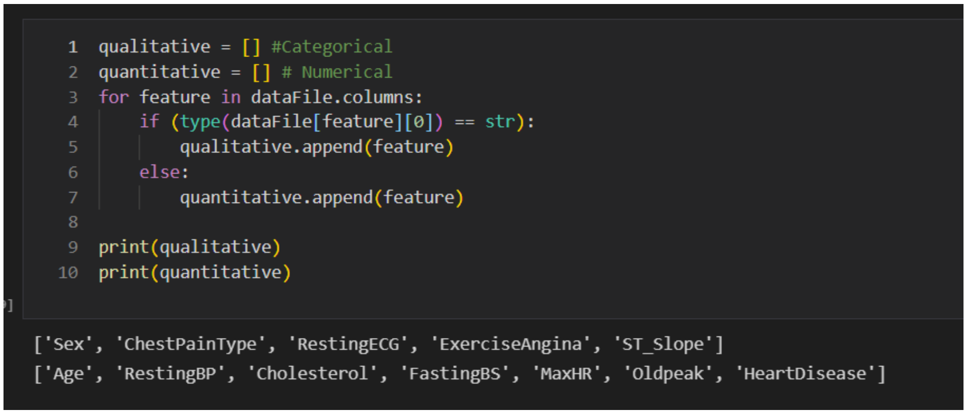
```
Age               0
Sex               0
ChestPainType     0
RestingBP         0
Cholesterol       0
FastingBS         0
RestingECG        0
MaxHR             0
ExerciseAngina    0
Oldpeak           0
ST_Slope          0
HeartDisease      0
dtype: int64
```

The selected dataset has both categorical and numerical data.



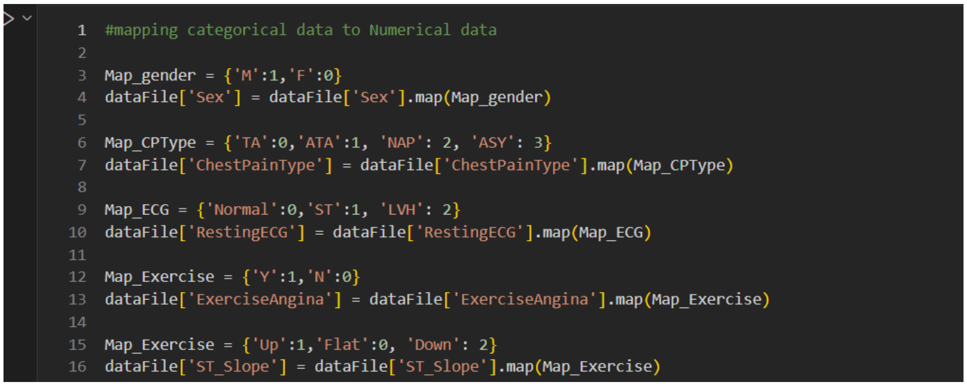
	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0
...	...	...	...	...	...	...	...	...	...	...	...	...
913	45	M	TA	110	264	0	Normal	132	N	1.2	Flat	1
914	68	M	ASY	144	193	1	Normal	141	N	3.4	Flat	1
915	57	M	ASY	130	131	0	Normal	115	Y	1.2	Flat	1
916	57	F	ATA	130	236	0	LVH	174	N	0.0	Flat	1
917	38	M	NAP	138	175	0	Normal	173	N	0.0	Up	0

Next separated the categorical data columns and mapped them to unique values to replace them with numerical values.



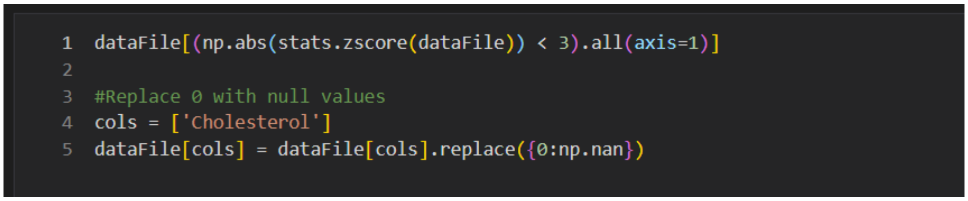
```
qualitative = [] #Categorical
quantitative = [] # Numerical
for feature in dataFile.columns:
    if (type(dataFile[feature][0]) == str):
        qualitative.append(feature)
    else:
        quantitative.append(feature)

print(qualitative)
print(quantitative)
```

```
['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina', 'ST_Slope']
['Age', 'RestingBP', 'Cholesterol', 'FastingBS', 'MaxHR', 'Oldpeak', 'HeartDisease']
```

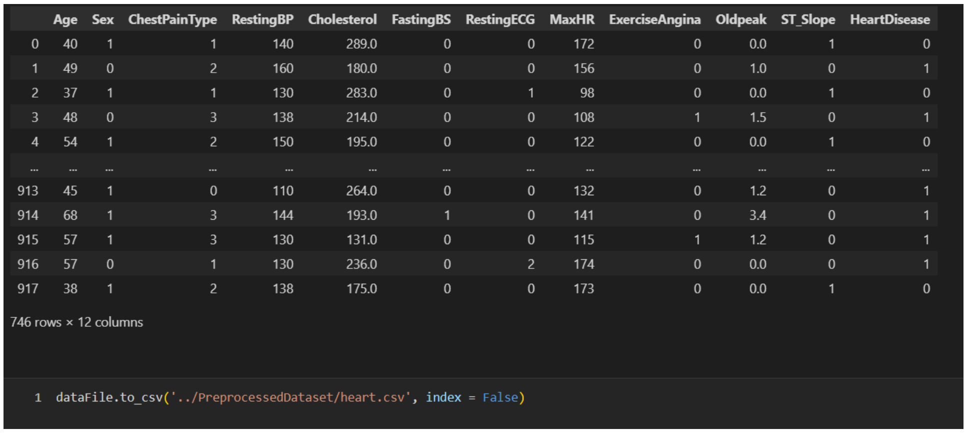


```python
#mapping categorical data to Numerical data

Map_gender = {'M':1,'F':0}
dataFile['Sex'] = dataFile['Sex'].map(Map_gender)

Map_CPType = {'TA':0,'ATA':1, 'NAP': 2, 'ASY': 3}
dataFile['ChestPainType'] = dataFile['ChestPainType'].map(Map_CPType)

Map_ECG = {'Normal':0,'ST':1, 'LVH': 2}
dataFile['RestingECG'] = dataFile['RestingECG'].map(Map_ECG)

Map_Exercise = {'Y':1,'N':0}
dataFile['ExerciseAngina'] = dataFile['ExerciseAngina'].map(Map_Exercise)

Map_Exercise = {'Up':1,'Flat':0, 'Down': 2}
dataFile['ST_Slope'] = dataFile['ST_Slope'].map(Map_Exercise)
```

Finally removed the outliers from the visualized data. The Cholesterol level column has one outlier and removed it.



```python
dataFile[(np.abs(stats.zscore(dataFile)) < 3).all(axis=1)]

#Replace 0 with null values
cols = ['Cholesterol']
dataFile[cols] = dataFile[cols].replace({0:np.nan})
```



	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	1	1	140	289.0	0	0	172	0	0.0	1	0
1	49	0	2	160	180.0	0	0	156	0	1.0	0	1
2	37	1	1	130	283.0	0	1	98	0	0.0	1	0
3	48	0	3	138	214.0	0	0	108	1	1.5	0	1
4	54	1	2	150	195.0	0	0	122	0	0.0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...
913	45	1	0	110	264.0	0	0	132	0	1.2	0	1
914	68	1	3	144	193.0	1	0	141	0	3.4	0	1
915	57	1	3	130	131.0	0	0	115	1	1.2	0	1
916	57	0	1	130	236.0	0	2	174	0	0.0	0	1
917	38	1	2	138	175.0	0	0	173	0	0.0	1	0

746 rows × 12 columns

```python
dataFile.to_csv('../PreprocessedDataset/heart.csv', index = False)
```

Then exported the preprocessed data into heart.csv file.

3.2 Selection of Algorithm – Random Forest

To prevent overfitting, a classification system is needed to predict the presence of heart failure. Small and medium-sized databases work well with random forest technology. Different decision trees also have benefits. The following database contains some misplaced values that can be manipulated by random forest and produce extremely fast and reliable results.

Random Forest is a supervised learning algorithm that monitors multiple decision tree building during training class rendering, which is the model of classes in classification problems.

The decision tree algorithm is considered a non-linear mapping classification and reaction tree (CART) relations. Populations are subdivided based on input variables. Therefore, there are two types of decision trees is used.

- Classification variable Decision tree - The target variable is categorized
- Continuous variable decision tree - The target variable is continuous

Build decision tree output based on decisions. It is usually used to identify the most important variable and the relationship between two or more attributes.
The random forest has almost the same parameters as a decision tree or bag classifier. But not necessarily combine with a tree bag classifier so the classification class can be used.

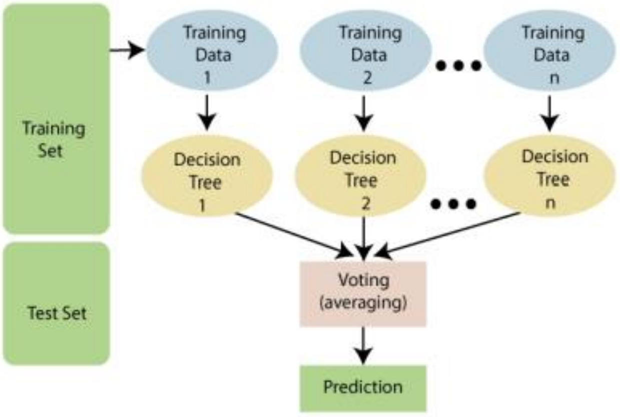


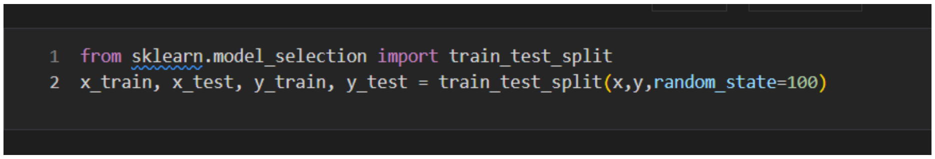
Figure – Random Forest

The random forest may be the decision tree as it combines the multiple trees mentioned in the diagram above. The output can be predicted to be accurate or not. But at the polls, everyone is predicting the right outcome. At work the process selects the random K dot point from the algorithm dataset and builds the associated referee sub points. The user can select the number of decision trees to build. Repeat the process until the training database is complete.

For a new data point, find the prediction in each decision tree and assign the new data to that category wins by voting.

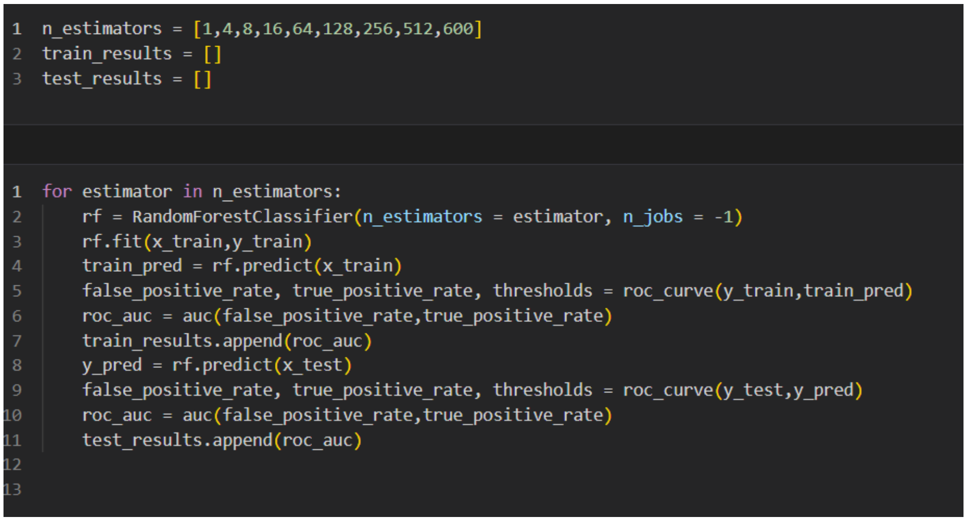
3.3 Implementation

3.3.1 Data Splitting



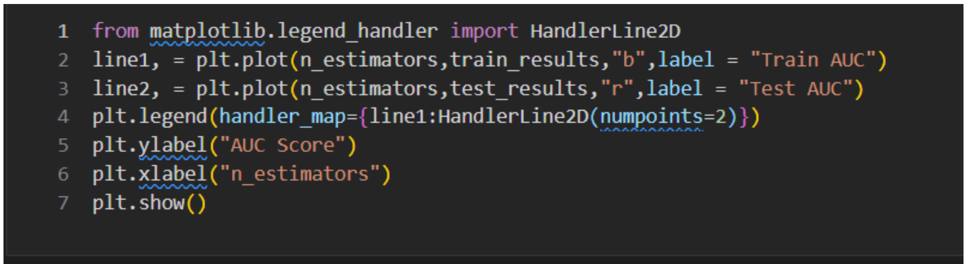
```python
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,random_state=100)
```

3.3.2 Create Random Forest

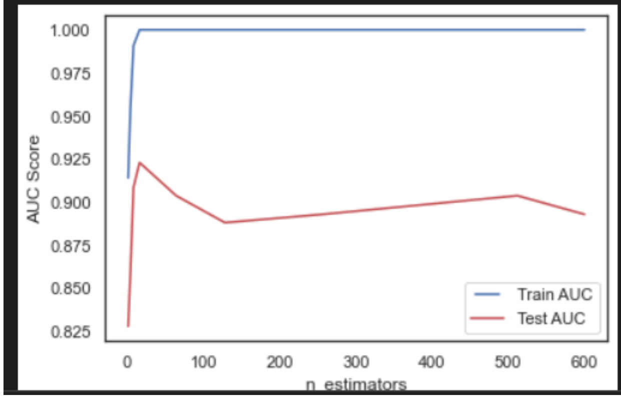


```python
n_estimators = [1,4,8,16,64,128,256,512,600]
train_results = []
test_results = []
```

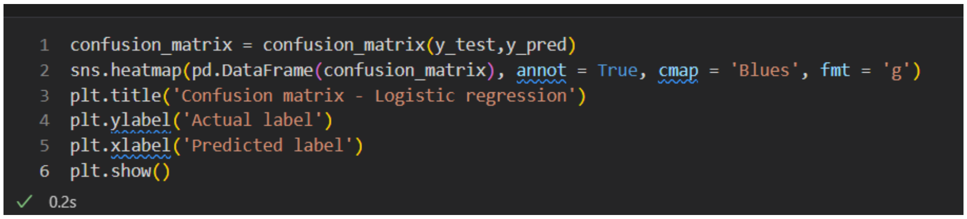
```python
for estimator in n_estimators:
    rf = RandomForestClassifier(n_estimators = estimator, n_jobs = -1)
    rf.fit(x_train,y_train)
    train_pred = rf.predict(x_train)
    false_positive_rate, true_positive_rate, thresholds = roc_curve(y_train,train_pred)
    roc_auc = auc(false_positive_rate,true_positive_rate)
    train_results.append(roc_auc)
    y_pred = rf.predict(x_test)
    false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test,y_pred)
    roc_auc = auc(false_positive_rate,true_positive_rate)
    test_results.append(roc_auc)


```



```python
from matplotlib.legend_handler import HandlerLine2D
line1, = plt.plot(n_estimators,train_results,"b",label = "Train AUC")
line2, = plt.plot(n_estimators,test_results,"r",label = "Test AUC")
plt.legend(handler_map={line1:HandlerLine2D(numpoints=2)})
plt.ylabel("AUC Score")
plt.xlabel("n_estimators")
plt.show()
```



3.3.3 Display Result



```python
confusion_matrix = confusion_matrix(y_test,y_pred)
sns.heatmap(pd.DataFrame(confusion_matrix), annot = True, cmap = 'Blues', fmt = 'g')
plt.title('Confusion matrix - Logistic regression')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
```

✓ 0.2s

4.0 Results

4.1 Accuracy and F1-Score

Can be used to evaluate the predictions made by the accuracy and F1 score format. So accuracy is one of them the clearest parametric measures all instances in which it is correctly identified.

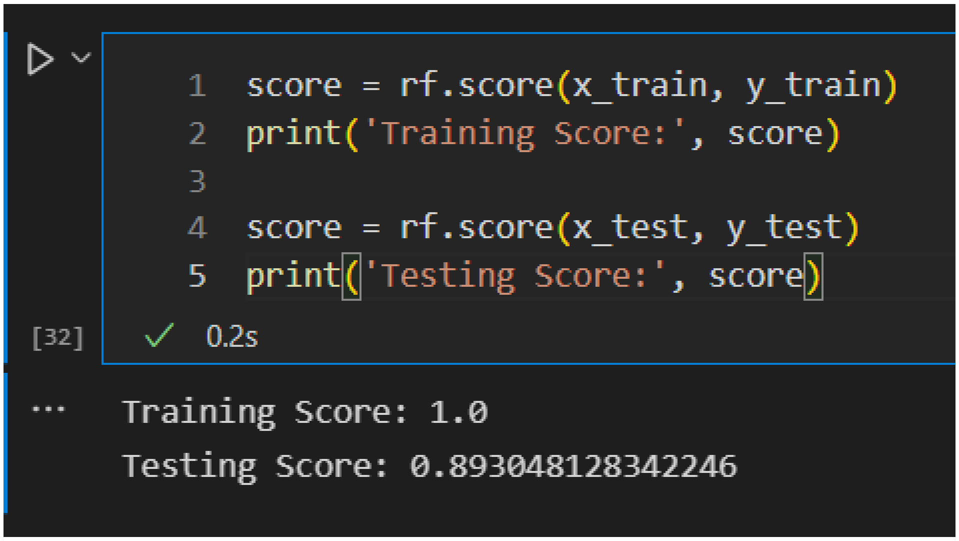
$$F1 - score \ = \left(\frac{Recall^{-1} + Precision^{-1}}{2}\right)^{-1} = 2 * \left(\frac{Recall * Precision}{Recall + Precision}\right)$$

The F1 score is the unifying point of accuracy and recall and provides a better measurement for incorrect classification chances are more than accuracy metrics.

$$Accuracy = \frac{TruePostive + TrueNegative}{(TruePostive + FalsePostive + TrueNegative + FalseNegative)}$$

F1-scores are used when false negatives and false positives are decisive, and accuracy is used when true.

The positives and the real negatives are more important.



```
score = rf.score(x_train, y_train)
print('Training Score:', score)

score = rf.score(x_test, y_test)
print('Testing Score:', score)
```

```
Training Score: 1.0
Testing Score: 0.893048128342246
```

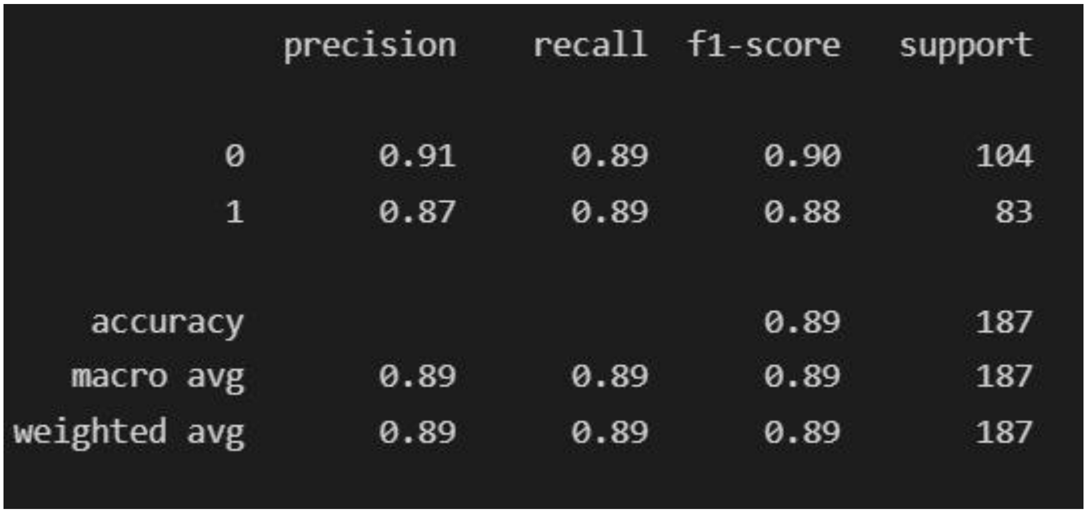
4.2 Classification Report

The classification report is used to identify the classification parameters of the training model. Accuracy, recall, F1 score and support are the parameters used in the report. It will show these values as true positive, true negative, false positive and false negative

Accuracy - Class marks for negative samples not positively labeled

Recall - Class marks for correctly identifying samples

Support - Number of samples in the test database (total dataset * 0.2)



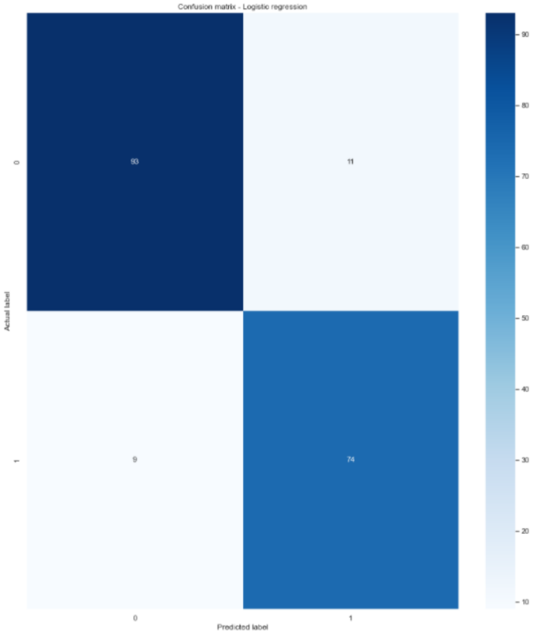
```
              precision    recall  f1-score   support

           0       0.91      0.89      0.90       104
           1       0.87      0.89      0.88        83

    accuracy                           0.89       187
   macro avg       0.89      0.89      0.89       187
weighted avg       0.89      0.89      0.89       187
```

Figure – Classification Report

4.3 Confusion Matrix Theory

Confusion matrix theory is used to calculate true and false predicted values. The training model actually gets 85 out of 154

As negative and 41 as true positive. Therefore, trainees can assume 126 out of 154 that are accurately predicted the training model.



5.0 Discussion

Algorithm	**Train Accuracy**	**Test Accuracy**
Logistic Regression	0.85	0.88
Decision Tree Classifier	0.91	0.80
Random Forest Classifier	~0.99	0.89
KNeighbors Classifier	0.77	0.65

6.0 References

[1] "Index of /ml/machine-learning-databases/heart-disease", *Archive.ics.uci.edu*, 2022. [Online]. Available: https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/. [Accessed: 29- May- 2022].

[2] https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/

7.0 Appendix

Git link

https://github.com/Sandun01/ML_Assignment2_IT18219326_IT19023656_IT19362854_IT19217796

Google drive link

https://mysliit-my.sharepoint.com/:f:/g/personal/it19362854_my_sliit_lk/EoJBBxkZ8WtMgZc6ZGgEyYoBHIq_kZU-gdoL4ygclPuzqg?e=69DlqV