Heart Disease Prediction

Milestone: Performance Evaluation & Interpretation

Group 3

Mugdha Sanjay Parbat - 002142372

Pranav Chandrakant Pulkundwar - 002121679

Telephone

+1 (617) 901-8417

+1 (617) 901-8418

Email ID

parbat.m@northeastern.edu

pulkundwar.p@northeastern.edu

Percentage of contribution by Student 1: 50%

Percentage of contribution by Student 2: 50%

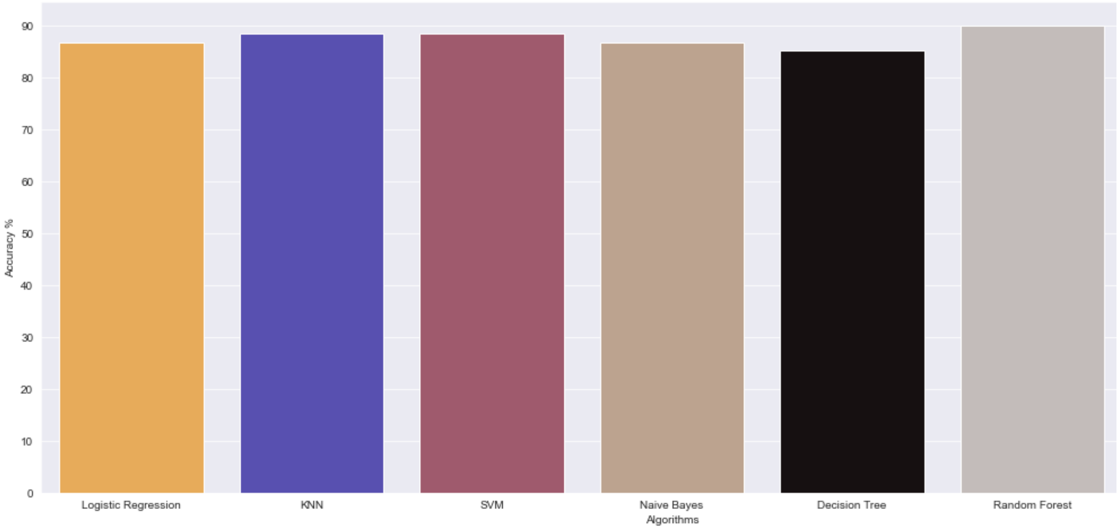
Submission Date: 18th April 2022

The Heart Disease Data for this project was first imported in Jupyter Notebook, which is then cleaned and visualized to understand which factors that show better relationship in finding out if a person has a heart disease or no.

After this analysis for model preparation first data was split into training and testing data. The division of training and testing data was decided to be 80-20%.

```
>> x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 0)
```

After calculating accuracy of the model using different methods available, we then picked ‘Random Forest’ as it has the best accuracy among all.



The Random Forest model is then performed twice to see if we get better accuracy if we increase the number of trees or ‘max_leaf_nodes’. This gave us a better fit and a better accuracy for the data model.

1. Confusion Matrix of the dataset without changing the maximum leaf node number.

We now calculate the Error Percentage, Accuracy, Sensitivity, specificity
For model execution, first we fit the model without changing the ‘max_leaf_nodes’, which gave us the accuracy for model as 88.52 %.

```
>> rf = RandomForestClassifier(n_estimators = 1000, random_state = 2)
>> rf.fit(x_train.T, y_train.T)
```

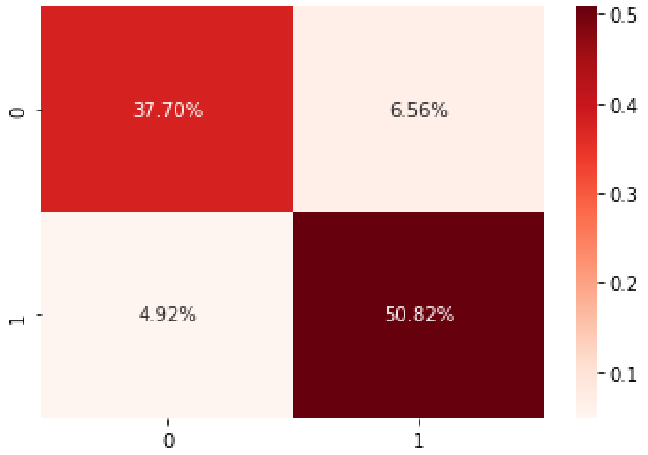
	0 (Predicted Negative)	1 (Predicted Positive)
0 (Actual Negative)	23	4
1 (Actual Positive)	3	31

Error Percentage = (3 + 4)/61 = **11.48 %**

Accuracy = (23 + 31)/61 = **88.52 %**

Sensitivity = 31/(31 + 3) = **91.17 %**

Specificity = 23/(23 + 4) = **85.18 %**



2. **Confusion Matrix of the dataset after looping for the maximum leaf node number.**

We now calculate the Error Percentage, Accuracy, Sensitivity, specificity
The again for better accuracy the model was run on a loop for 'max_lead_nodes' values between (2,25), which in turn performed well, to give highest accuracy of 90.16 %

```
>> score_list_RF = [] for i in range(2,25):

>> rf2 = RandomForestClassifier(n_estimators = 1000, random_state = 2, >>
max_leaf_nodes = i) rf2.fit(x_train.T, y_train.T)
>> score_list_RF.append(rf2.score(x_test.T, y_test.T))
```

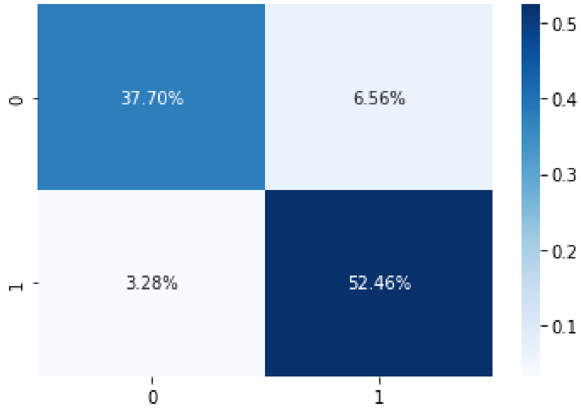
	0 (Predicted Negative)	**1 (Predicted Positive)**
0 (Actual Negative)	23	4
1 (Actual Positive)	2	32

Error Percentage = (2 + 4)/61 = **9.83 %**

Accuracy = (23 + 31)/61 = **90.16 %**

Sensitivity = 32/(32 + 2) = **94.11 %**

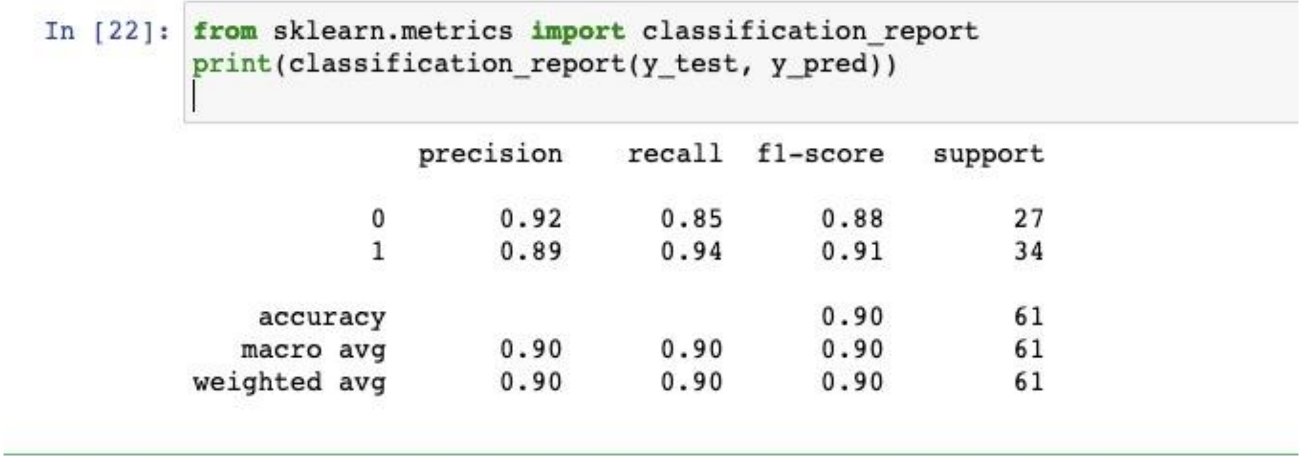
Specificity = 23/(23 + 4) = **85.18 %**



There are 2 confusion matrices:- one with the random forest classifier and another with the random classifier model where we adjusted the parameters that improved the accuracy to 90.16%. We varied the number of max_leaf_nodes from 2 to 25, and number of trees = 1000.

From the confusion matrix of the second model, we can say that For instance, we can see the model is predicting most of the true cases correctly (52.46%). The most important about this model being selected is that it predicts only 3.28% of patients as no heart disease when they have heart disease. This percentage is very low, and very crucial in any clinical setting where there is a diagnosis for any disease.

Classification report



```
In [22]: from sklearn.metrics import classification_report
         print(classification_report(y_test, y_pred))

              precision    recall  f1-score   support

           0       0.92      0.85      0.88        27
           1       0.89      0.94      0.91        34

    accuracy                           0.90        61
   macro avg       0.90      0.90      0.90        61
weighted avg       0.90      0.90      0.90        61
```

We examined other metrics like precision, recall, and F1 score to get even more detailed insight into the model's performance.

1. Precision

- Precision is the number of correctly identified members of a class divided by all the times the model predicted that class.

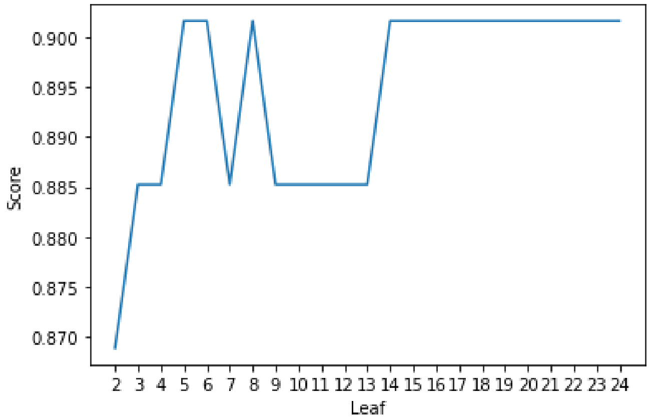
- In our case, the precision score would be the number of correctly identified having heart disease divided by the total number of times the classifier predicted having heart diseases rightly or wrongly.
- Precision in this model is 88.89%.

2. Recall

- Recall is the number of members of a class that the classifier identified correctly divided by the total number of members in that class.
- Here, this would be the number of having heart disease that the classifier correctly identified as such.
- Precision is 94.11% for this model.

3. F1 score

- F1 score combines precision and recall into one metric.
- If precision and recall are both high, F1 will be high, too. If they are both low, F1 will be low. If one is high and the other low, F1 will be low.
- F1 is a quick way to tell whether the classifier is good at identifying members of a class, or if it is taking shortcuts (e.g., just identifying everything as a member of a majority class).
- An F1 score is considered perfect when it's 1, while the model is a total failure when it's 0. In this model, F1 score is 0.91.

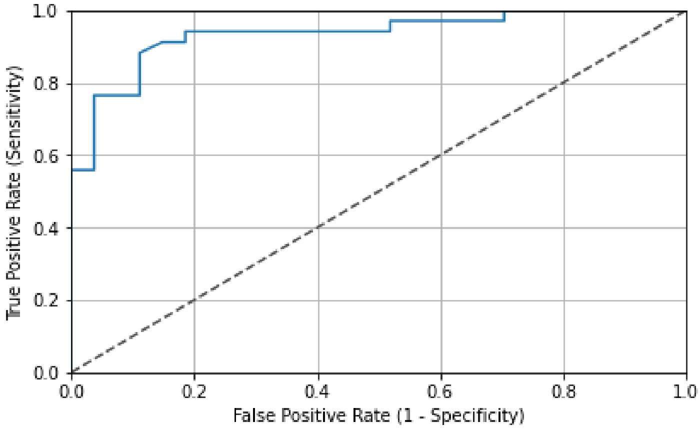


Our random forest classifier has an accuracy score of 91.06% on the test data.

It looks like a great accuracy score, but we have to keep in mind that it's not the best measure of classifier performance when the classes are not balanced.

The important question is doing the model perform equally well for each class? Are there any pairs of classes that were hard to distinguish? To answer all these questions, we make a confusion matrix.

3. ROC Curve



We then plotted the ROC curve, which is a plot of 'sensitivity' vs '1 – Specificity'. The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve. The higher the AUC, the better the performance of the model is at distinguishing between the positive and negative classes. Here the model AUC is 0.93, which shows that the model is performing well.