Project Group – 3

April 28th 2020

Heart Disease Prediction using Machine Learning

By Pranav Pulkundwar & Mugdha Parbat

Contents

Problem Setting and Definition

- 659,000 people die every year due to heart disease (1 in every 4 deaths)

- Leading cause of deaths irrespective of gender

- Why is it so hard to detect? Several factors contributing

- Here's when data mining and machine learning steps in

- Applied and tested models on the database provided by UCI

Data Description

- 303 rows and 14 columns
- Encoded the categorical variables
- 14 columns –

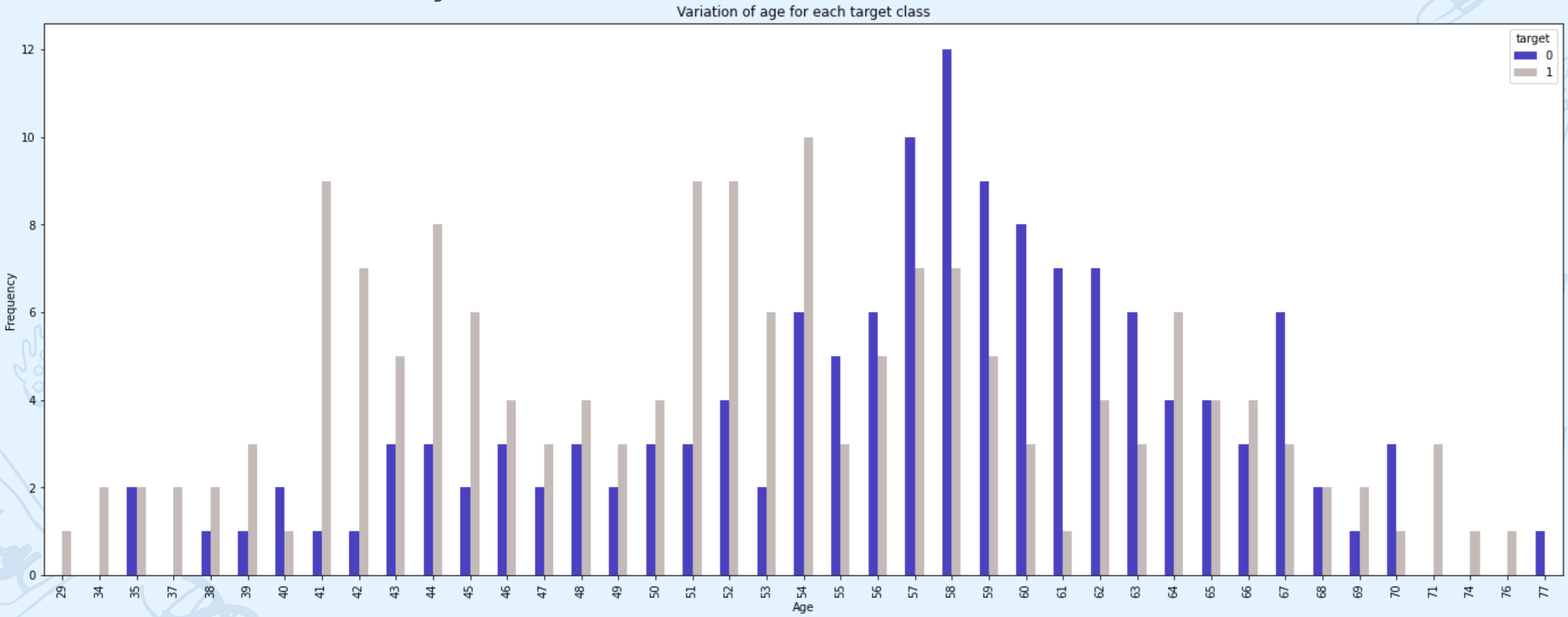
Age	Max Heart Rate Achieved
Sex	Exercise Induced Angina
Chest-pain Type	St Depression Induced By Exercise Relative To Rest
Resting Blood Pressure	Peak Exercise St Segment
Serum Cholesterol	Number Of Major Vessels Colored By Fluoroscopy
Fasting Blood Sugar	Thalassemia
Resting ECG	Diagnosis Of Heart Disease

Libraries Used

- Numpy
- Pandas
- Matplotlib
- Seaborn
- LogisticRegression
- train_test_split
- confusion_matrix
- classification_report
- roc_curve, auc
- accuracy_score
- RandomForestClassifier

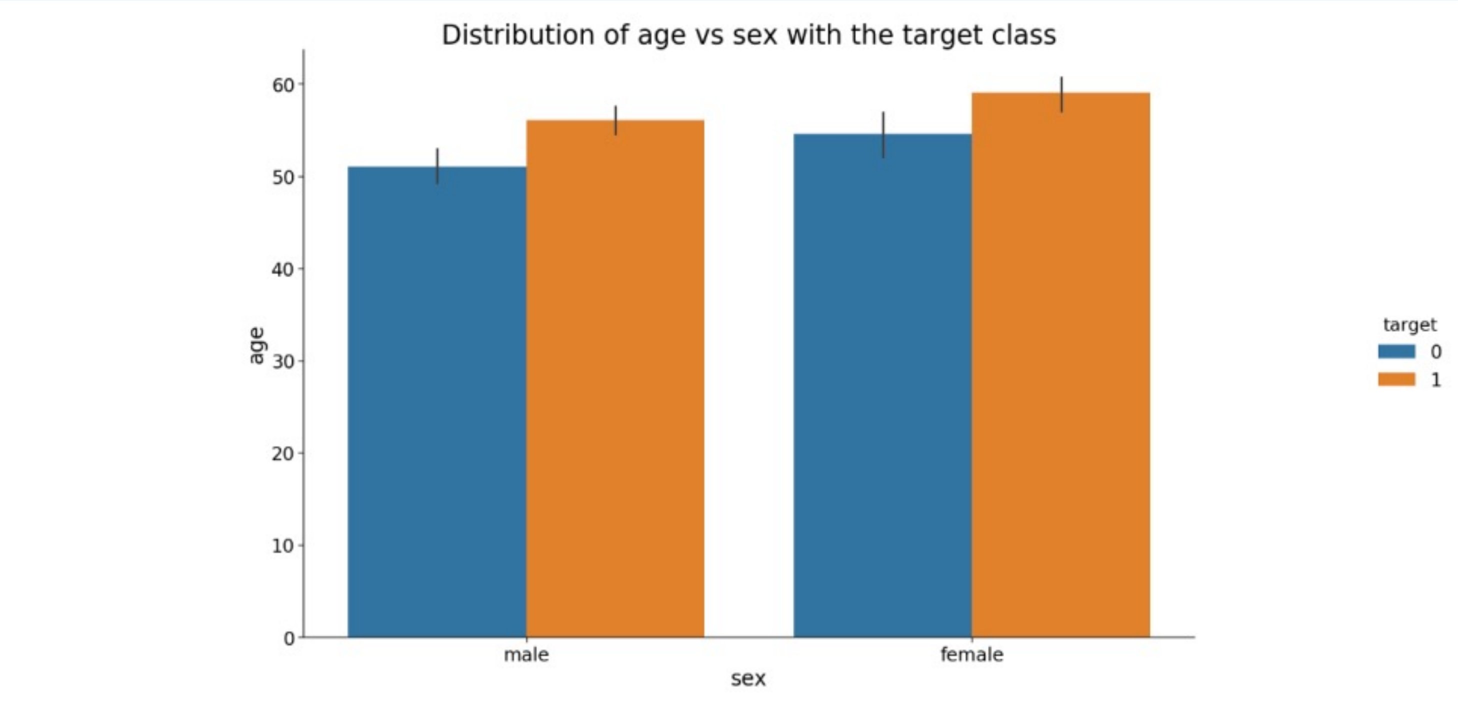
Data Exploration

- Target = 1 has heart disease and 0 does not
- Majority - 58 years old, followed by 57 years old
- Overall, 50+ years old



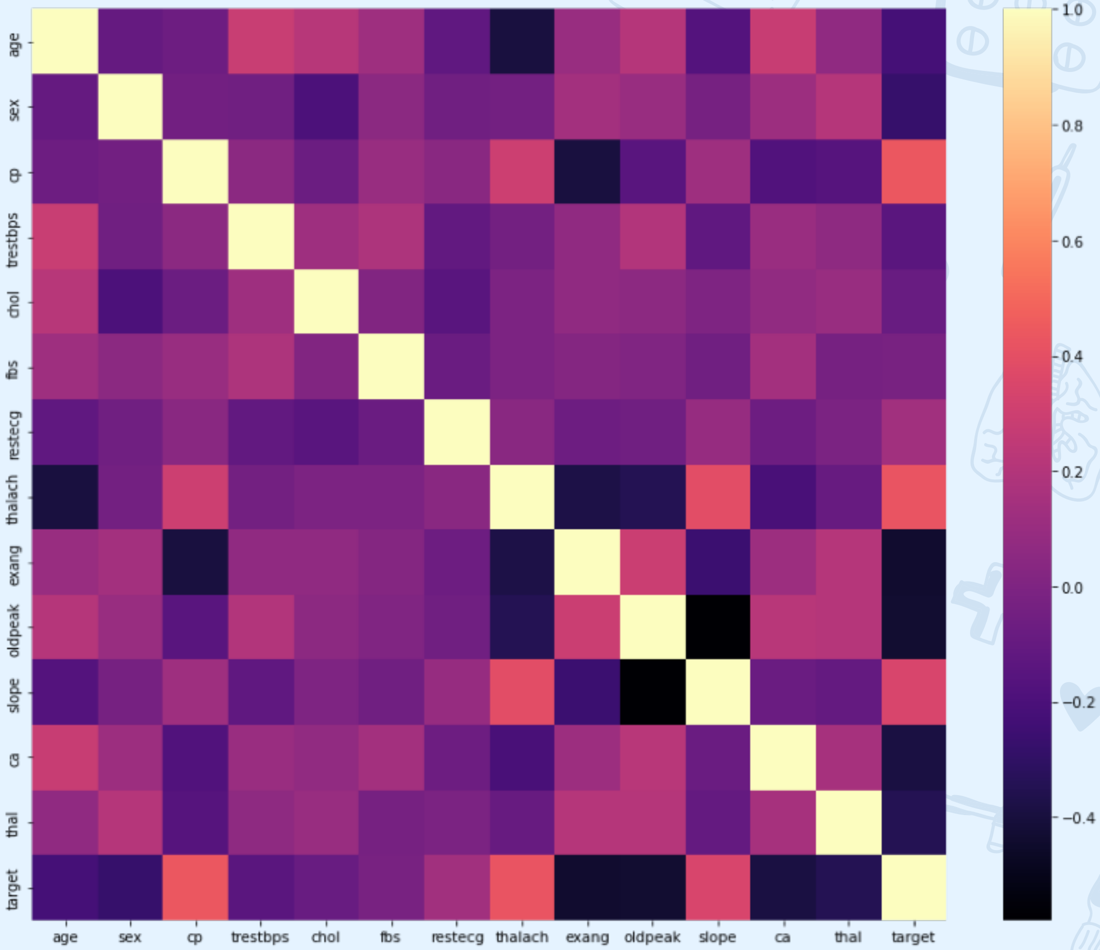
Data Exploration

- Age of males who are suffering from the heart disease are younger than the females.



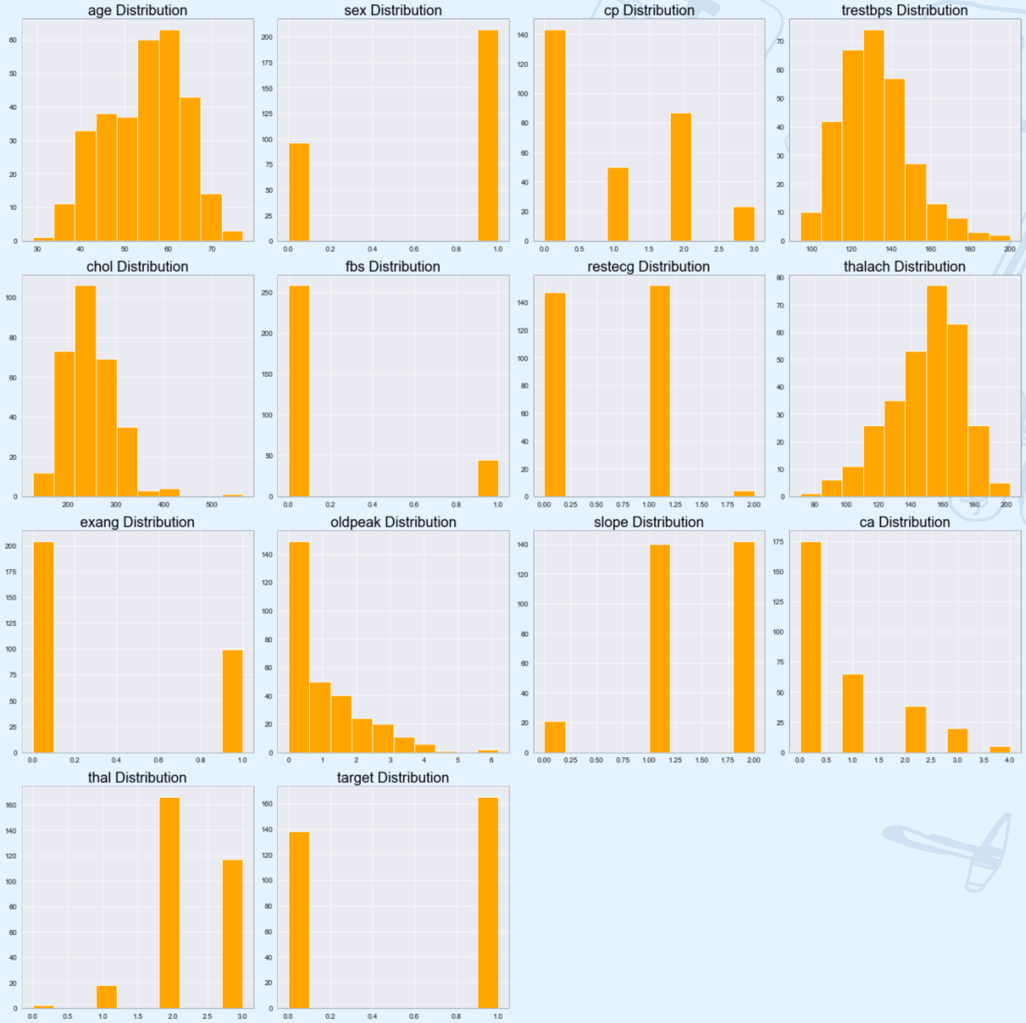
Data Exploration

- Some of the features have a positive and some have negative correlation with the target values but no strong correlation as such



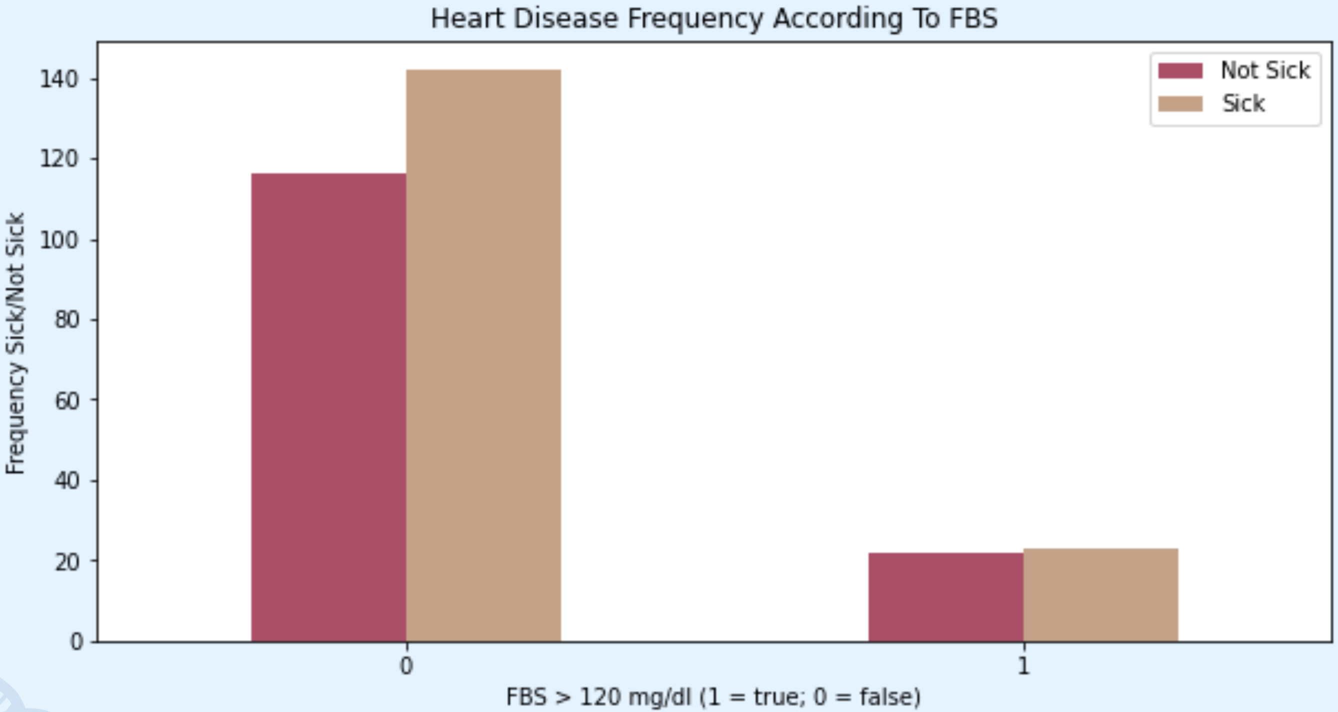
Data Exploration

- The histogram plots show how each feature and label is distributed along different ranges.
- Needs Scaling



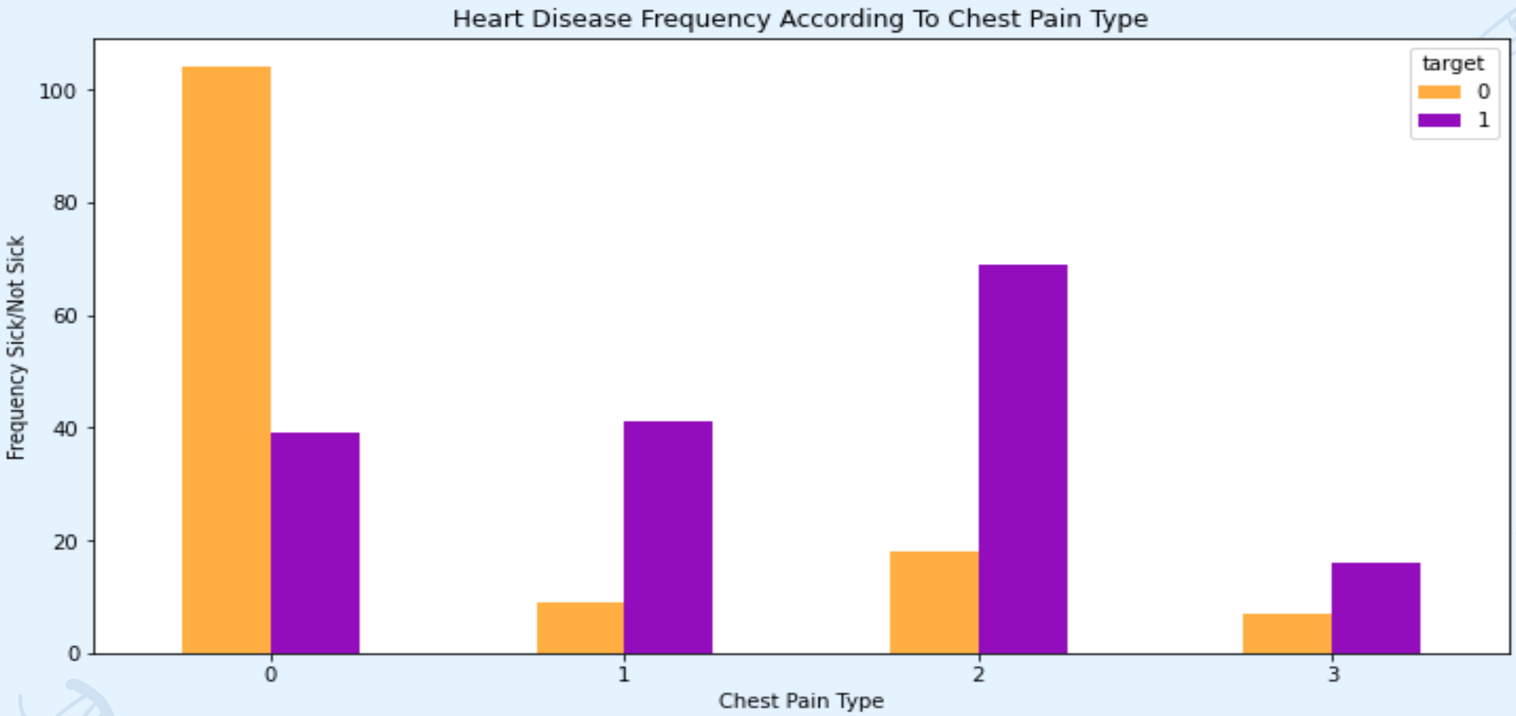
Data Exploration

- The number of patients who have / do not have heart disease is higher in those who do not do fasting blood sugar.



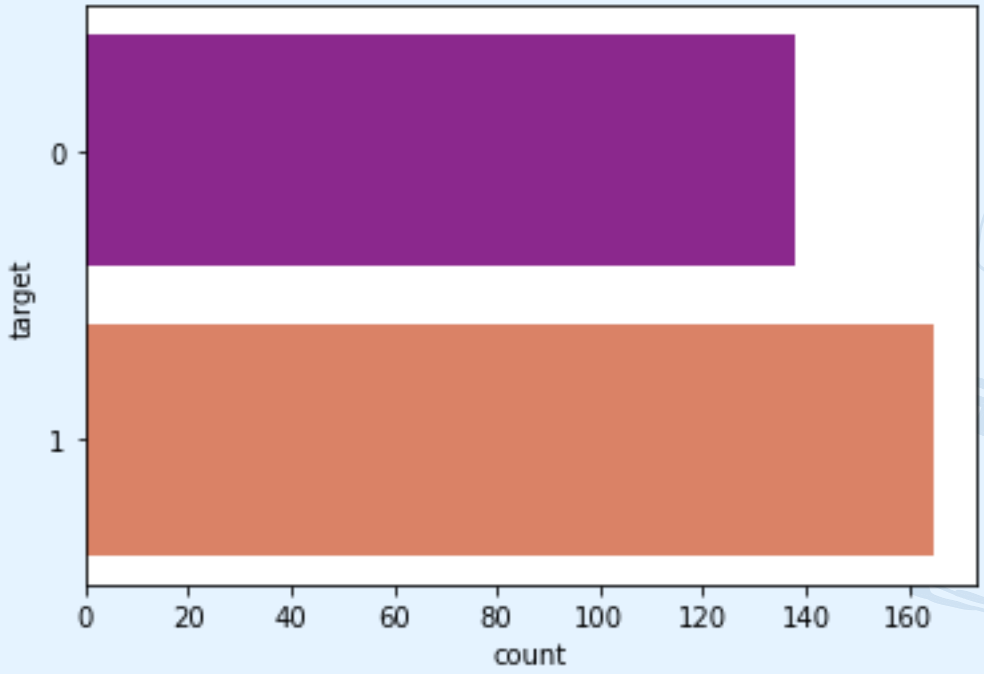
Data Exploration

- Level 0 has higher number of patients that don't have heart disease.
- Level 2 has higher number of patients that have heart disease.



Data Exploration

- The classes are balanced, and ready to go to the next step, data pre-processing.

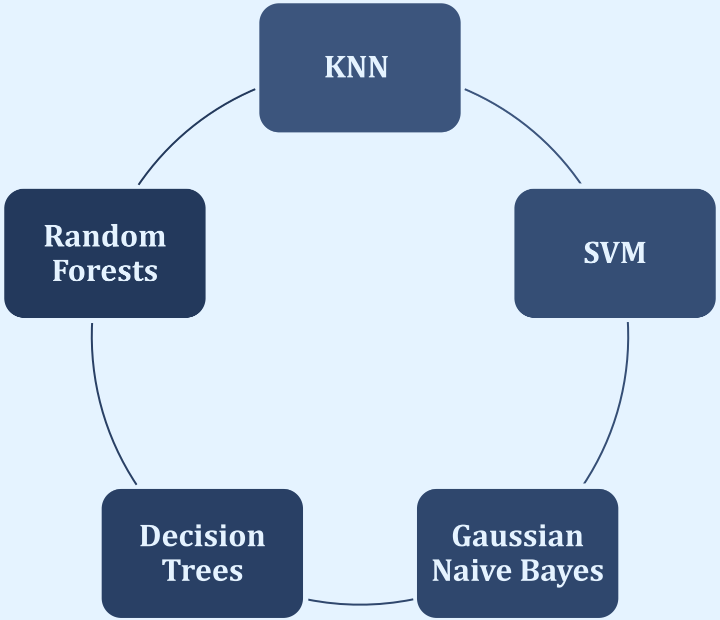


Data Preprocessing

- Converting categorical column into dummy columns with 1s and 0s using get_dummies()
- Normalised the data to bring every column in the same range
- Dropped ‘Thal’, ‘ST segment sloping’, ‘Chest pain type’

Data Mining Tasks

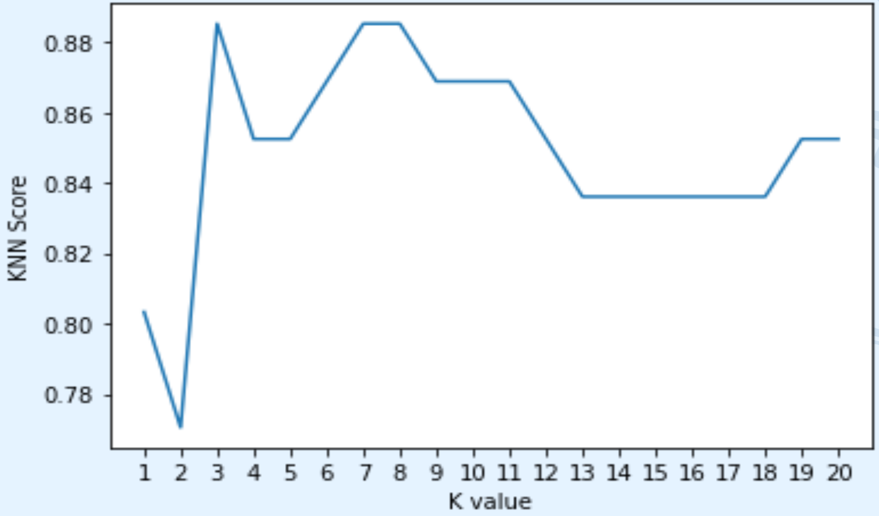
- Split data into 80% training data and 20% test data
- 5 algorithms taken by varied their parameters and compared models



Model Exploration

KNN

- This classifier looks for the classes of K nearest neighbors of a given data point and based on the majority class, it assigns a class to this data point.
- Varied them from 1 to 20 neighbors and achieved highest accuracy of 88.52%



Model Exploration

SVM

- Forms a hyperplane that can separate the classes as much as possible by adjusting the distance between the data points and the hyperplane.
- Tried four kernels viz linear, poly, rbf, and sigmoid.
- 'rbf' performed the best for highest score of 88.52%.

```
In [19]: svc_scores = []
         kernels = ['linear', 'poly', 'rbf', 'sigmoid']
         for i in range(len(kernels)):
             svc_classifier = SVC(kernel = kernels[i])
             svc_classifier.fit(x_train.T, y_train.T)
             svc_scores.append(svc_classifier.score(x_test.T,y_test.T))

In [20]: svc_scores

Out[20]: [0.8524590163934426, 0.819672131147541, 0.8852459016393442, 0.7868852459016393]

In [21]: print("Test Accuracy of SVM: {}%".format(max(svc_scores)*100))

         Test Accuracy of SVM: 88.52459016393442%
```

Model Exploration

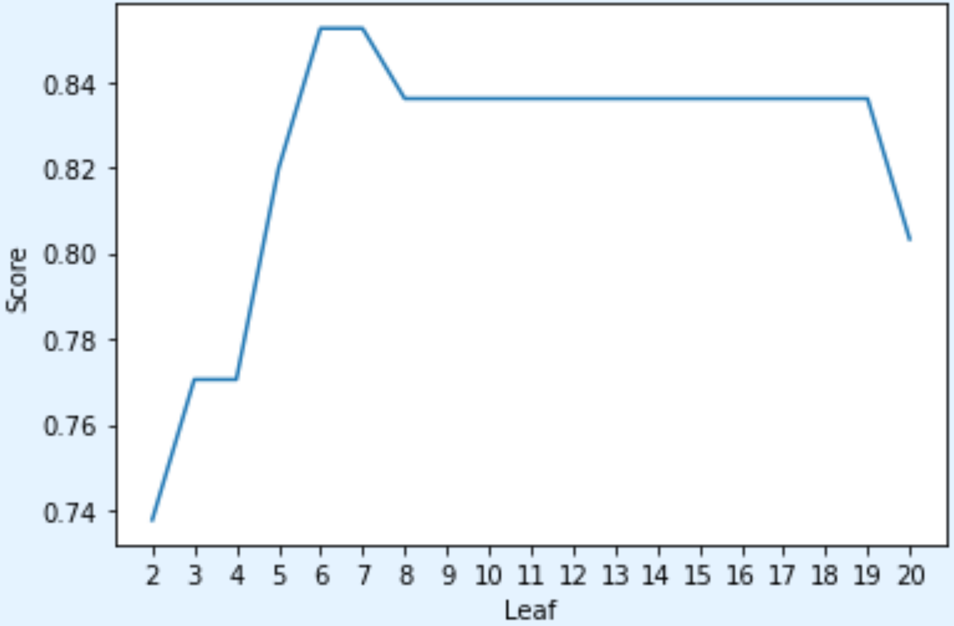
Gaussian Naïve Bayes

- Gaussian Naive Bayes is used when the data is continuous and assumed to have gaussian distribution
- Test accuracy is 86.89 %

Model Exploration

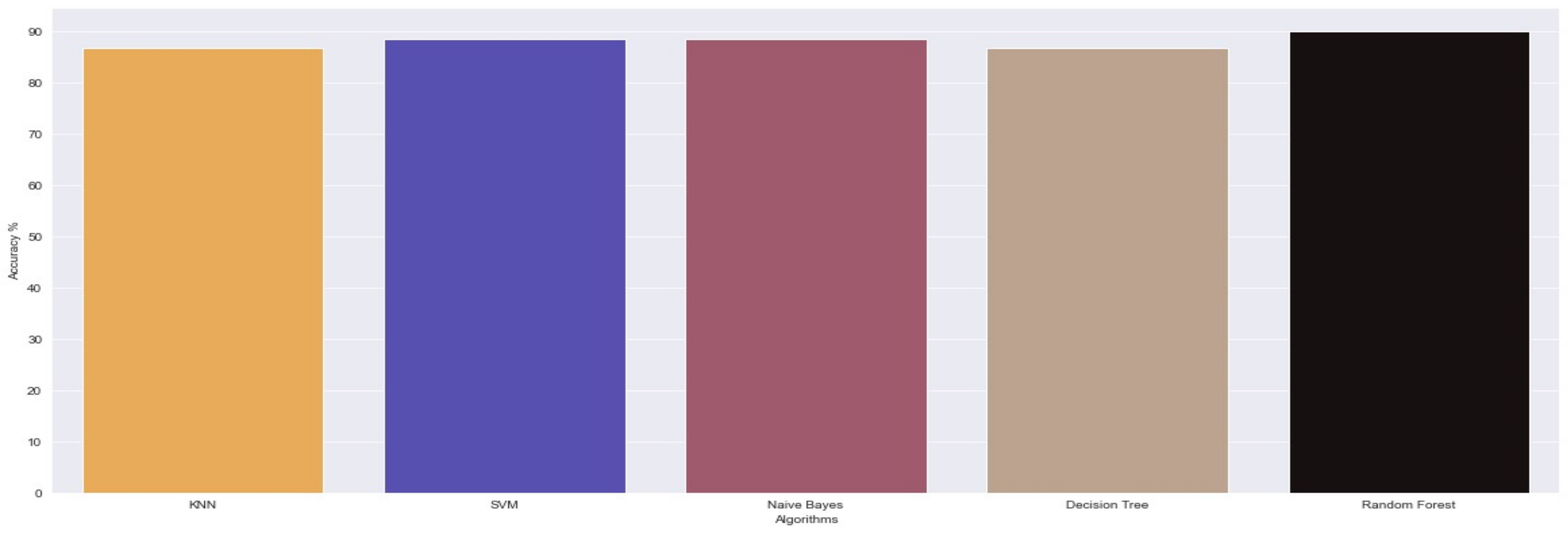
Decision Tree

- We range the features from 1 to 30 (30 is the total number of features in the dataset after creating dummy variables).
- We got an accuracy of 85.24 %



Model Exploration

- Five models selected, were executed to find the highest accuracy for each model



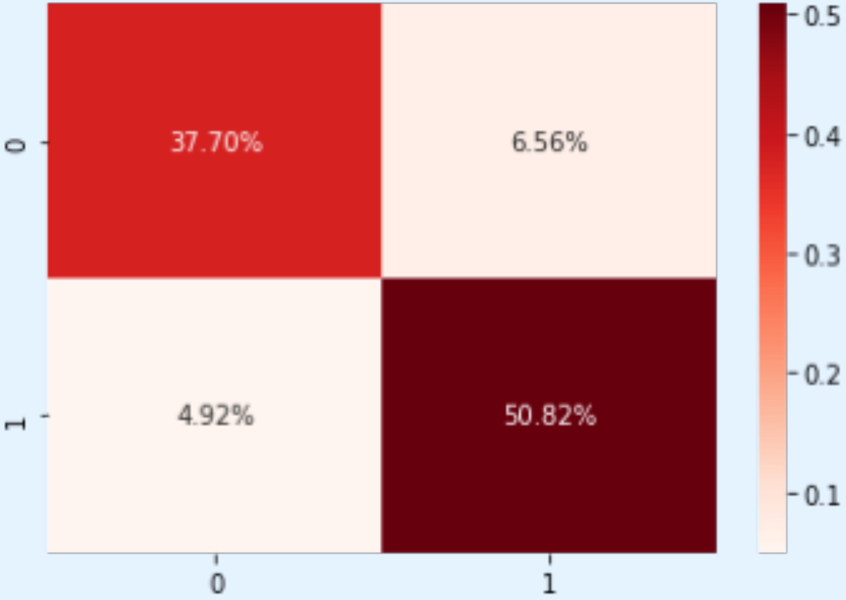
88.52 % 88.52 % 86.89 % 85.24 % 90.16 %

Model Selection – Random Forest

- Random forest model was chosen for implementation since the accuracy is found to be highest
- Although the accuracy is one factor, Random Forest has other advantages like –
    - Random Forest Is Based On The Bagging Algorithm And Uses Ensemble Learning Technique
    - Works Well With Both Categorical And Continuous Variables
    - Can Automatically Handle Missing Values
    - No Feature Scaling Required
    - Handles Non-linear Parameters Efficiently
    - Usually Robust To Outliers And Can Handle Them Automatically
    - Algorithm Is Very Stable
    - Comparatively Less Impacted By Noise

Performance Evaluation

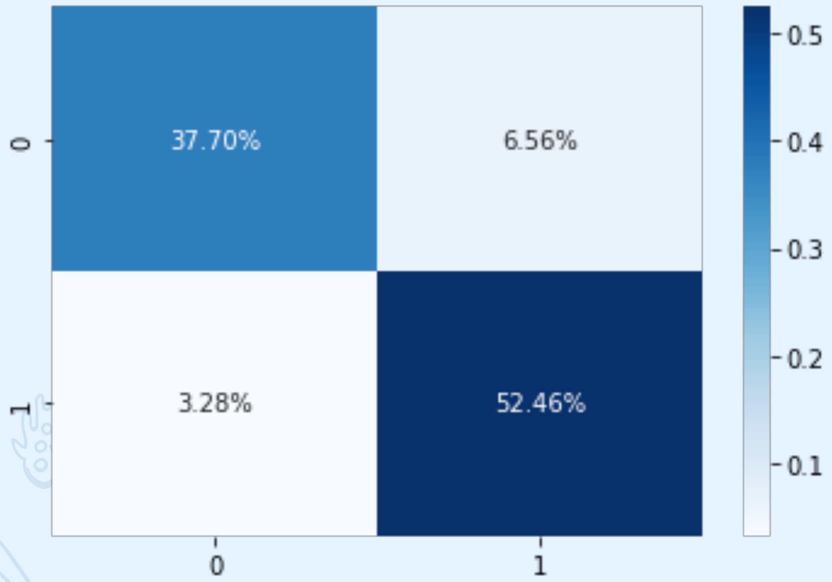
- For model execution, first we fit the model without changing the ‘max_leaf_nodes’, which gave us the accuracy for model as 88.52 %.
- >> rf = RandomForestClassifier(n_estimators = 1000, random_state = 2)
- >> rf.fit(x_train.T, y_train.T)
- [23 4]
  [3 31]

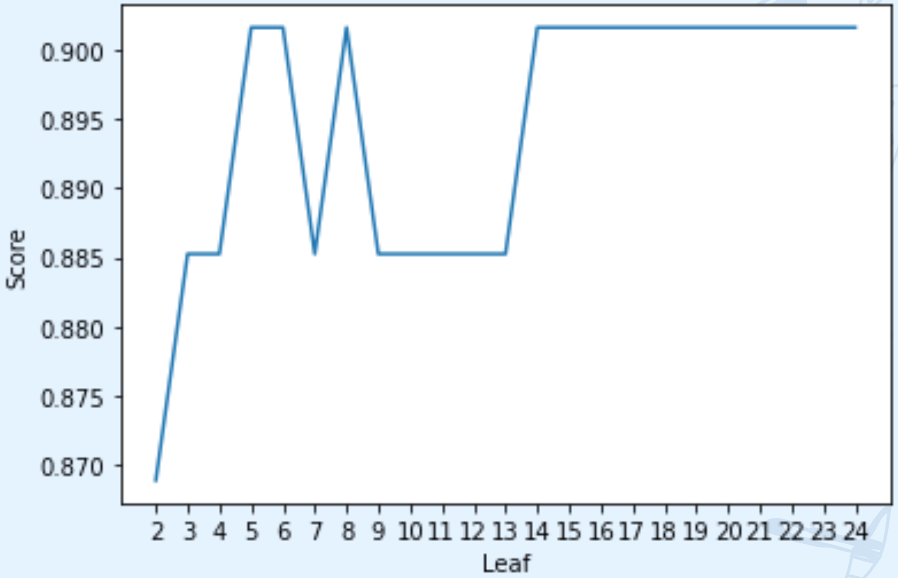


Performance Evaluation

- 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))
- [23 4]
  [2 32]

Performance Evaluation



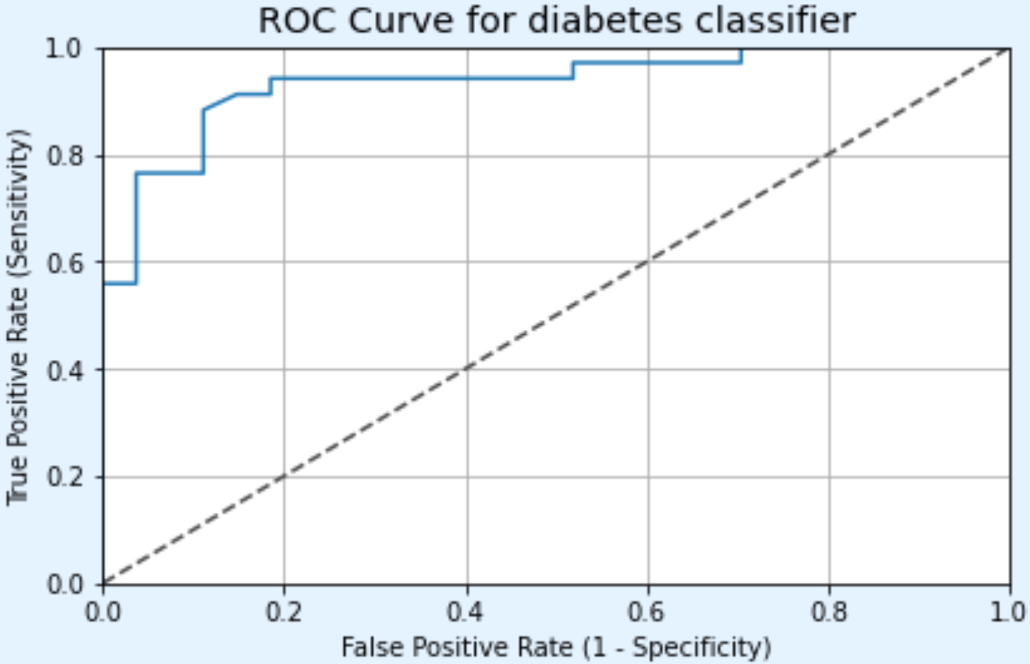


Metrics Comparison

Metric	1st Execution (%)	Re-execution (%)
Error	11.48	9.83
Accuracy	88.52	90.16
Sensitivity	91.17	94.11
Specificity	85.18	85.18

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.



Classification Report – Insights for Decision Making

- Examining other metrics like
  - Precision
  - Recall
  - F1 score

  to get even more detailed insight into the model's performance.

```
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
```

Future Scope

- More detailed data can be used for unknown patterns in prediction
- Prediction of heart disease with higher accuracy can be achieved
- Reducing time taken for prediction using more refined algorithms
- Finding the risk factor percentage so that high risk patients are tend to with urgency

THANK YOU!

Any Questions?