# Project 3: Applying Your Skills

Submitted by : Gmon Kuzhiynaikkal
NU ID : 002724506

## **INTRODUCTION:**

For this project, I have used the heart disease data set from the kaggle. It has following dataset statistics

| | |
|---|---|
| **Number of variables** | 12 |
| **Number of observations** | 918 |

It has 5 numeric variables, 6 categorical variables and 1 boolean variable.

I have implemented the  pre-processing data, including filling missing values, PCA, plotting the data and standardising data to a specific range. In addition, I have tried to  identify the best features for a dataset using various feature selection methods, and have recognized significant signals present in independent variables.

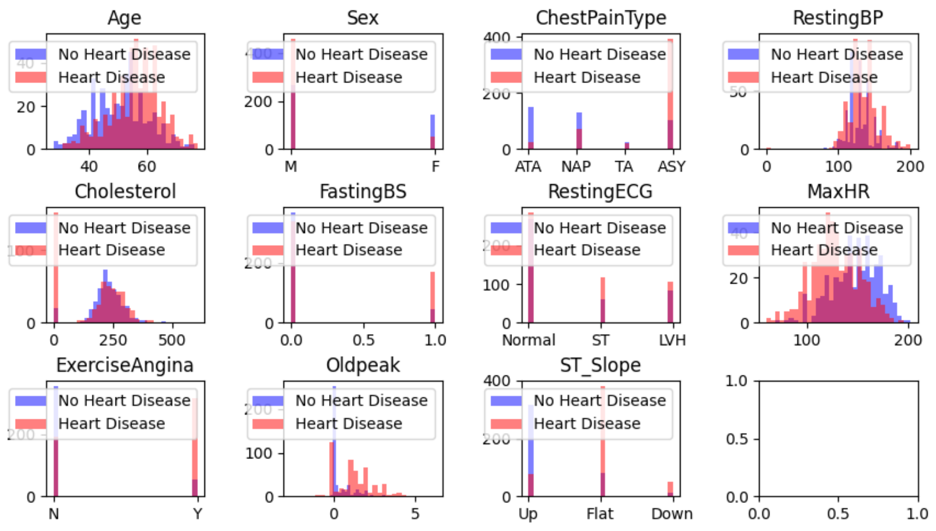
To improve the accuracy of my machine learning models, I have implemented several classic ML models, such as logistic regression, decision tree, and random forest. I evaluated these models using various metrics, including confusion matrix, F1 value, bias, and variance, and determined which classifier performed best.

Furthermore, I have learned how to use Receiver Operator Curve (ROC) for the model and performed additional iterations on the random forest to improve its efficiency. I also became proficient in tuning hyper-parameters and criteria on the random forest to improve its overall performance.As a part of extensions, I have done a fourth iteration to improve its efficiency too.

As part of my second extensions to extend my knowledge and skills, I have implemented a support vector machine and compared its efficiency with the other models I had previously implemented. Through this process, I have gained a deeper understanding of machine learning and how to effectively apply it to real-world problems.

## 1.Pre-processing, Data Mining, and Visualization:

I have tried to plot the histogram plot for each of the feature with the dependent variable heart disease and output of the files are given below:



Program to run this is 1a.py

### B)What pre-processing (if any) did you execute on the variables?

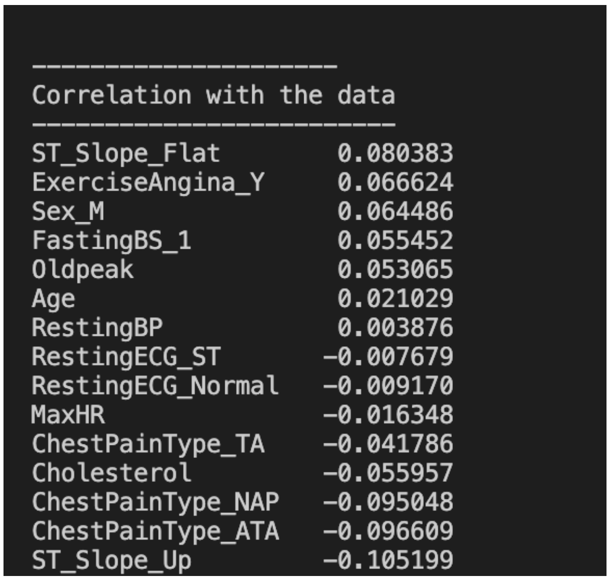
The pre processing, I have done on the variables are

1) I have tried to fill the missing values, if there are any
2) Converted all the categorical data to numerical data using the hot-encoding technique
3) Standardise the data into a particular range.

PS: this program can be found in 1b.py

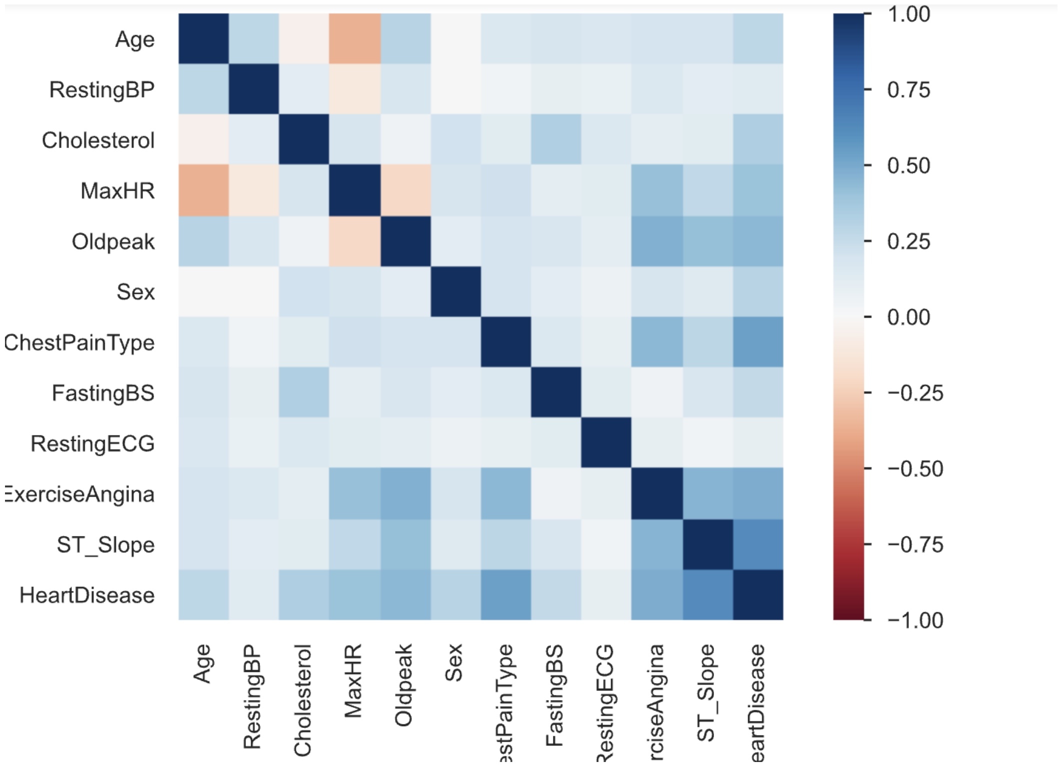
### C)Which independent variables are strongly correlated (positively or negatively)?

I have tried two method for this, first i tired to Use linear regression to identify features that are strongly correlated with heart disease and the output of the program is given below:



```
---------------------
Correlation with the data
-------------------------
ST_Slope_Flat         0.080383
ExerciseAngina_Y      0.066624
Sex_M                 0.064486
FastingBS_1           0.055452
Oldpeak               0.053065
Age                   0.021029
RestingBP             0.003876
RestingECG_ST        -0.007679
RestingECG_Normal    -0.009170
MaxHR                -0.016348
ChestPainType_TA     -0.041786
Cholesterol          -0.055957
ChestPainType_NAP    -0.095048
ChestPainType_ATA    -0.096609
ST_Slope_Up          -0.105199
```

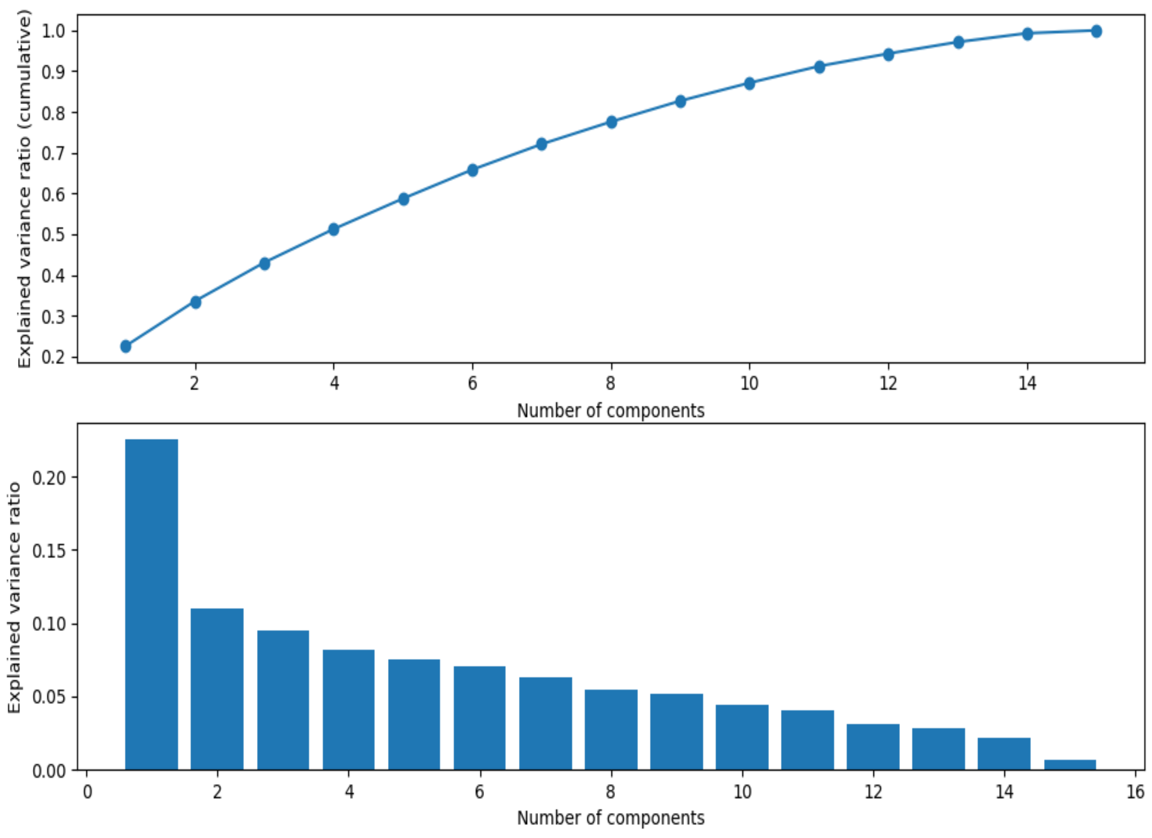
ST slope(flat) is showing a strong correlation with heart disease.
PS:program to run this is in 1b.py. In addition to that, I tried to plot the heat map of the pandas profiling report



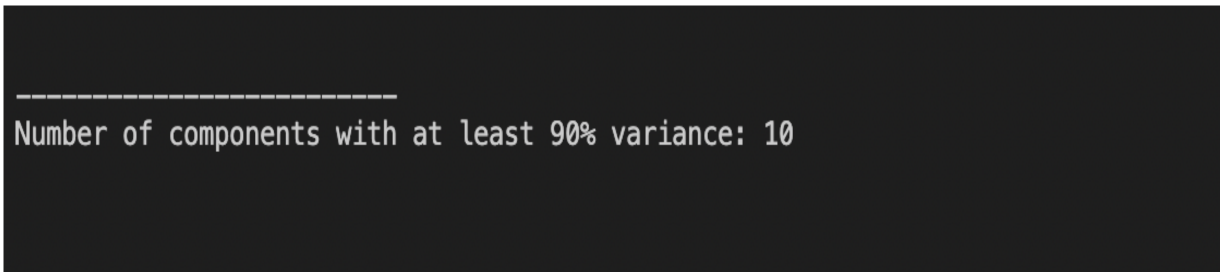
From both the experiment, I can conclude that,
1)ChestPainType is highly overall correlated with HeartDisease
2)ST_Slope is highly overall correlated with HeartDisease

PS: pandas profiling report is also attached along with the submission, file to run is pandas_profiling.py

**D: How many significant signals exist in the independent variables?**
Initially i tried to perform the PCA on the significant signals, I tired to plot the graph between the number of components and explained variance ratio and the output of the graph is shown below:



In addition to that, I tried to find the minimum number of components to have at least 90% variance and output for came out to be 10 components. The output of the program is given below:



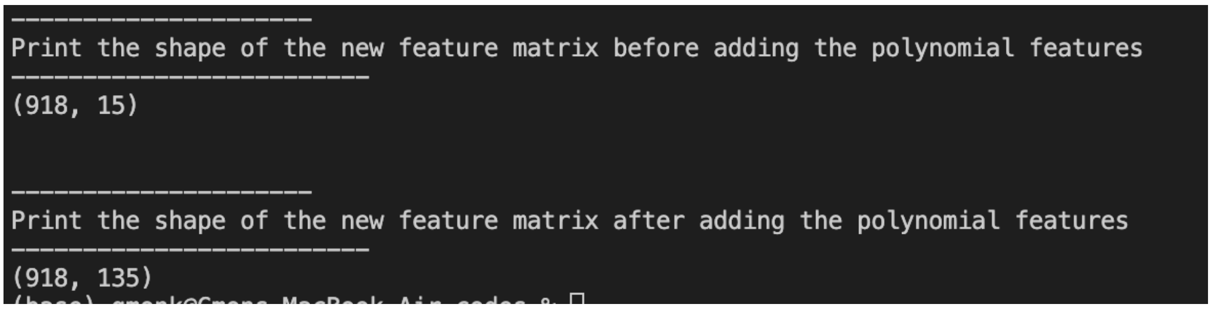
```
-------------------------
Number of components with at least 90% variance: 10
```

PS: program to run this is 1b.py

**E)What derived or alternative features might be useful for analysis (e.g. polynomial features)?**

I have tried to add the polynomial features to the data. I have then created a PolynomialFeatures object with a degree of 2, which will add squared terms for each of the original independent variables, as well as interaction terms between them. The include_bias=False argument ensures that an intercept term is not included.

From printing the feature matrix before and after adding the polynomial feature, I could understand that there is an increase in the number column significantly because of adding the polynomial matrix.



```
---------------------
Print the shape of the new feature matrix before adding the polynomial features
-------------------------
(918, 15)


---------------------
Print the shape of the new feature matrix after adding the polynomial features
-------------------------
(918, 135)
(base) gmonk@Gmons-MacBook-Air codes % 
```

PS: python program to run this is 1b.py

## <u>2.Classification:</u>

In all of the 3 ML model, I tried to use the below preprocessing techniques and they are:

1) I have tried to fill the missing values, if there are any
2) Converted all the categorical data to numerical data using the hot-encoding technique
3) Standardise the data into a particular range.
4) Applied PCA on the data
5) Applied polynomial feature of degree 2

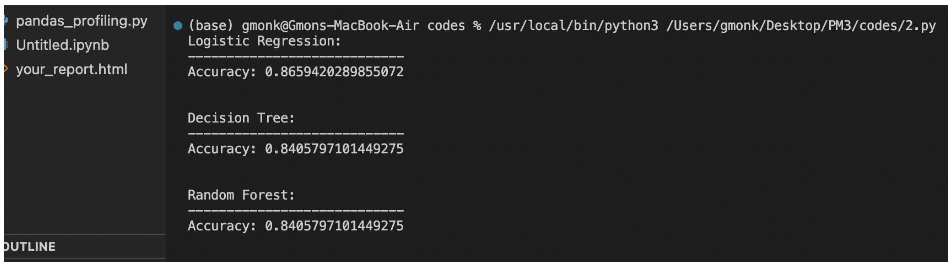
After doing all the above mentioned pre-processing,  I have mainly used 3 ML classification technique and they are

1)Logistic Regression: Logistic regression is a classification method that models the probability of a binary outcome (in this case, the presence or absence of heart disease) based on a set of independent variables. The basic idea is to fit a linear model to the data, and then transform the output of the linear model using a logistic function to obtain a probability value between 0 and 1.

2)Decision Tree:A decision tree is a classification method that recursively partitions the data into subsets based on the values of the independent variables, and assigns a class label to each subset based on the majority class of the training examples that belong to that subset.

3)Random Forest:A random forest is an ensemble method that combines multiple decision trees trained on different subsets of the data and different subsets of the independent variables, and averages their predictions to improve the generalisation performance.

The accuracy report of all these classifiers produced and output of the program is given below and Logistic regression has higher accuracy from the output.

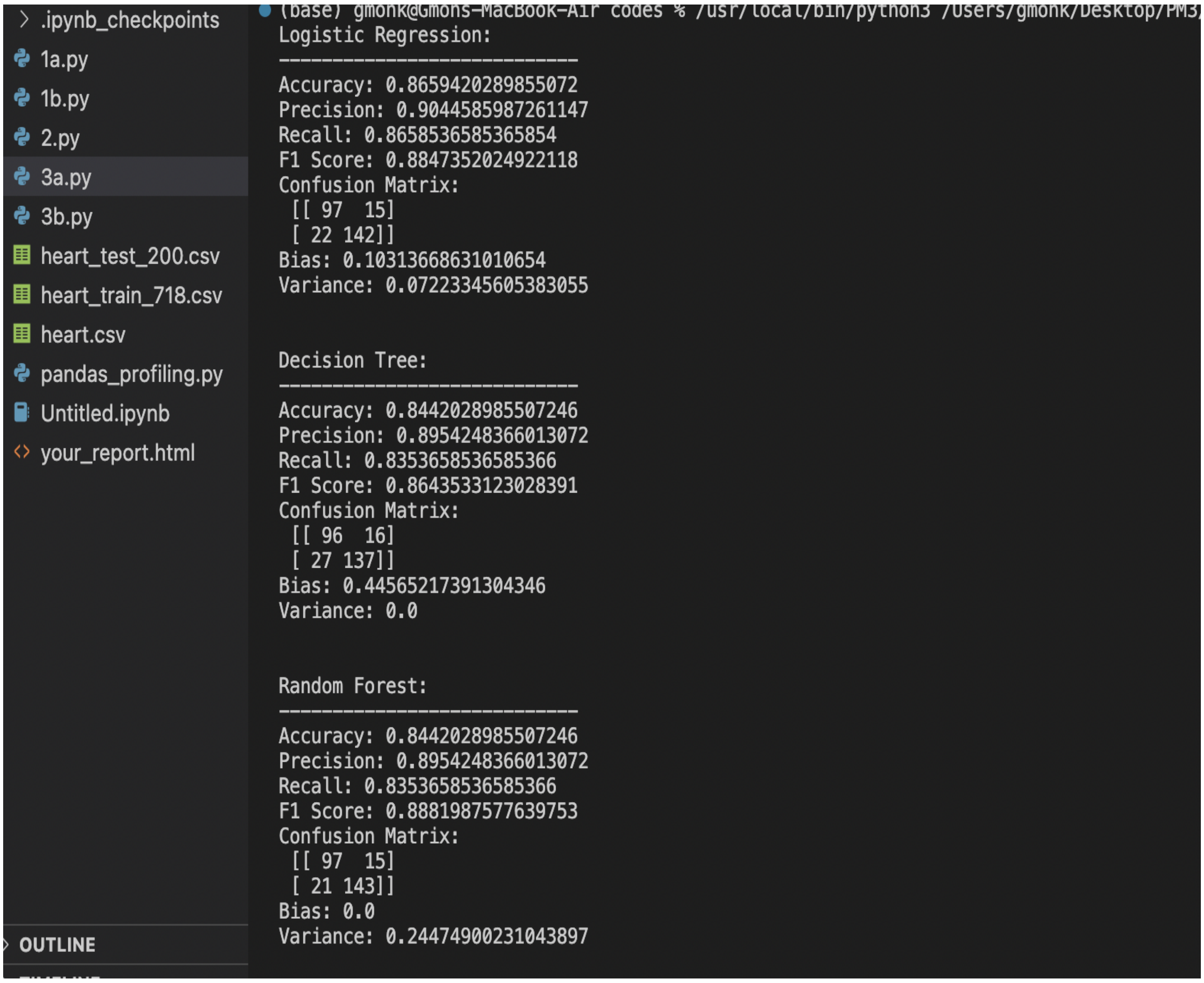


```
(base) gmonk@Gmons-MacBook-Air codes % /usr/local/bin/python3 /Users/gmonk/Desktop/PM3/codes/2.py
Logistic Regression:
------------------------------
Accuracy: 0.8659420289855072


Decision Tree:
------------------------------
Accuracy: 0.8405797101449275


Random Forest:
------------------------------
Accuracy: 0.8405797101449275
```

PS: Python file to run this ‘2.py’

## 3.Evaluation:

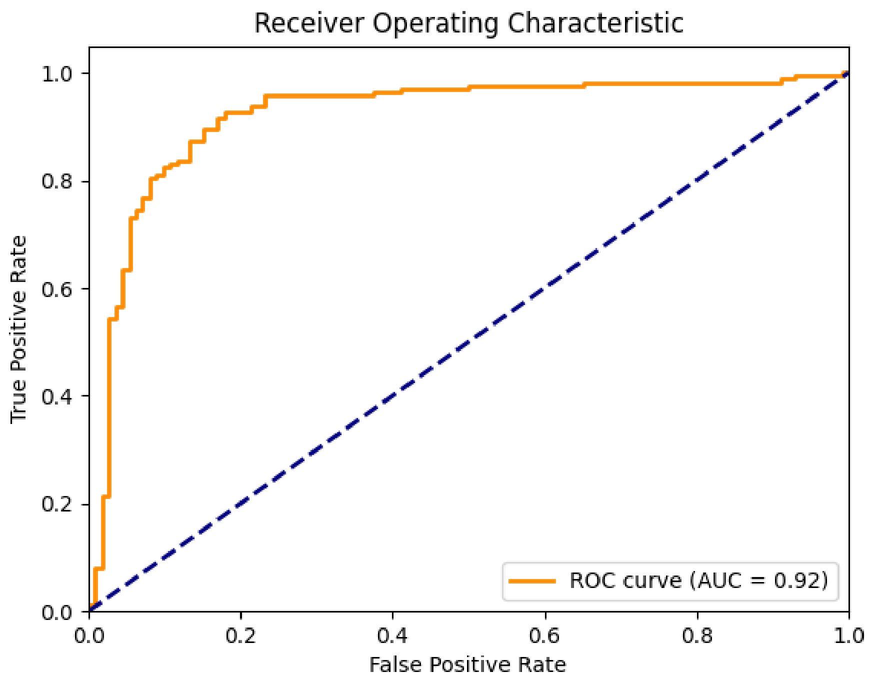
I have tried to compute the accuracy,precision,confusion matrix, the F1 value, the bias, and the variance for the default operating point for all of the 3 ML models and output of the program is as given below:



```
(base) gmonk@Gmons-MacBook-Air codes % /usr/local/bin/python3 /Users/gmonk/Desktop/PM3
Logistic Regression:
----------------------------
Accuracy: 0.8659420289855072
Precision: 0.9044585987261147
Recall: 0.8658536585365854
F1 Score: 0.8847352024922118
Confusion Matrix:
 [[ 97  15]
 [ 22 142]]
Bias: 0.10313668631010654
Variance: 0.07223345605383055


Decision Tree:
----------------------------
Accuracy: 0.8442028985507246
Precision: 0.8954248366013072
Recall: 0.8353658536585366
F1 Score: 0.8643533123028391
Confusion Matrix:
 [[ 96  16]
 [ 27 137]]
Bias: 0.44565217391304346
Variance: 0.0


Random Forest:
----------------------------
Accuracy: 0.8442028985507246
Precision: 0.8954248366013072
Recall: 0.8353658536585366
F1 Score: 0.8881987577639753
Confusion Matrix:
 [[ 97  15]
 [ 21 143]]
Bias: 0.0
Variance: 0.24474900231043897
```

PS: python program to run this '3a.py'

From the different values, I can see that Logistic Regression is performing better and hence I tried to generate a Receiver Operator Curve [ROC] by varying the operating point of the classifier. The output of the file is given below:



PS: python program to run this is ‘3b.py’

Now for the comparison of the models

**A)Which classifier did the best?**
Based on the given output, Logistic Regression performed the best among the three classifiers. Because it has higher accuracy, precision, and F1 value. The confusion matrix, bias, variance are almost equal to all the other 2 models. Thus from my model, Logistic Regression performed better than decision tree and Random Forest.

**B)What statistic are you using to decide which classifier is the best performer?**
I have used F1 Score to decide which classifier is the best performer. F1 Score takes into account both precision and recall, which makes it a good metric for evaluating the overall performance of a classifier.

**C)What does the bias and variance indicate as to what the best next steps to take would be to improve performance?**
Bias and variance provide insight into the generalisation error of the model.it helps in identifying the overfitting and underfitting in a model. Bias is an indication of the systematic error, while variance represents the degree of fluctuation of the model's prediction for different sets of training data. In general, we want a model with low bias and low variance, but there is often a trade-off between the two. The decision tree model has high bias and low variance, while the random forest has low bias but high variance. To improve the performance of the decision tree, we may need to consider a more complex model with more features, while for the random forest, we

may need to reduce the complexity of the model by tuning the hyperparameters or reducing the number of features. These are some of the steps which can be taken into account to improve the performances.

**D)What would be a good operating point for the classifier for which you generated the ROC curve?**

A good operating point on an ROC curve is at the point that maximises the trade-off between true positives and false positives. This is often achieved by selecting a point that is closest to the top-left corner of the ROC curve, which represents the ideal classifier with 100% true positive rate and 0% false positive rate. From the graph the true positive rate should be around 0.8 and false positive rate around close to 0.0 from the graph.

## 4.Iteration:

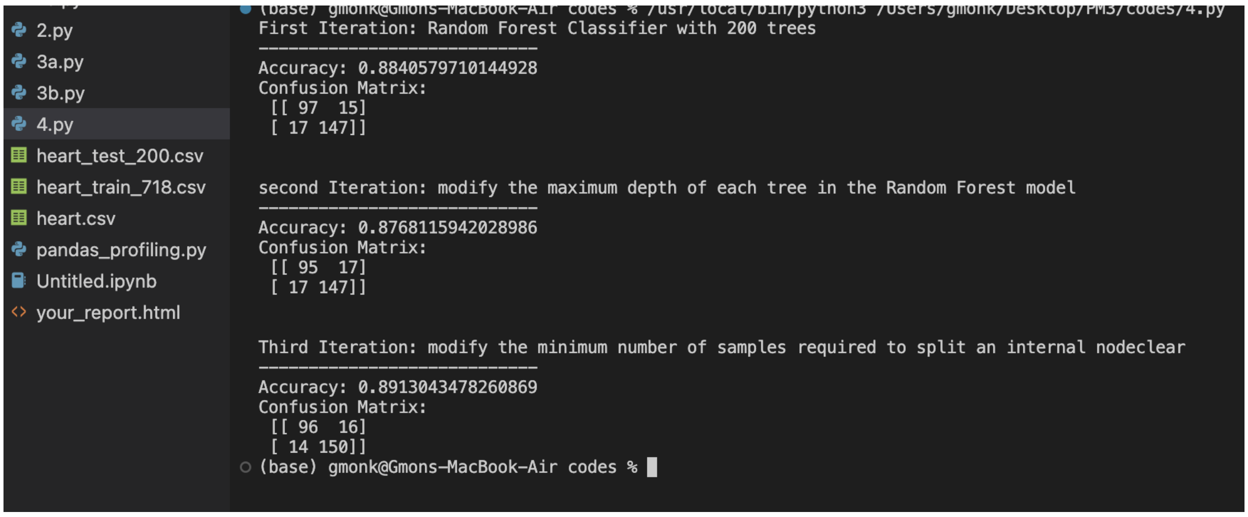
To improve the performance of one of the classifiers, I have selected the Random Forest classifier and will go through three iterations of making modifications to it to see if I can improve its accuracy. I will use the confusion matrix and the accuracy score as the performance metrics to optimise.

**First Iteration:** I have tried to improve the number of trees in the model from 100 to 200 and the accuracy of the model has increased from 0.84.4 to 0.884. There is better improvement in the confusion matrix, true positive value too.

**Second Iteration:**I have increased the number of trees from 100 to 200. In addition to that, I have increased the depth of the tree from the default value to 7. From this, I could see a decrease in the accuracy of the model and the confusion matrix too. That is it has reduced from 0.884 to 0.876

**Third Iteration:** In the third iteration. I have increased the number of trees from 100 to 200. In addition to that, I have increased the depth of the tree from the default value to 7 and I will modify the minimum number of samples required to split an internal node in the Random Forest model from the default value of 2 to 7.From this accuracy has increased from the 0.876 to 0.8913. There is better improvement in the confusion matrix, true positive value to false negative.

By the three iterations, I was able to change the accuracy of the model from 0.844 to 0.8913. The output of the program is given below



```
(base) gmonk@Gmons-MacBook-Air codes % /usr/local/bin/python3 /Users/gmonk/Desktop/PM3/codes/4.py
First Iteration: Random Forest Classifier with 200 trees
------------------------------
Accuracy: 0.8840579710144928
Confusion Matrix:
 [[ 97  15]
 [ 17 147]]


second Iteration: modify the maximum depth of each tree in the Random Forest model
------------------------------
Accuracy: 0.8768115942028986
Confusion Matrix:
 [[ 95  17]
 [ 17 147]]


Third Iteration: modify the minimum number of samples required to split an internal nodeclear
------------------------------
Accuracy: 0.8913043478260869
Confusion Matrix:
 [[ 96  16]
 [ 14 150]]
(base) gmonk@Gmons-MacBook-Air codes %
```

Program to run this is 4.py

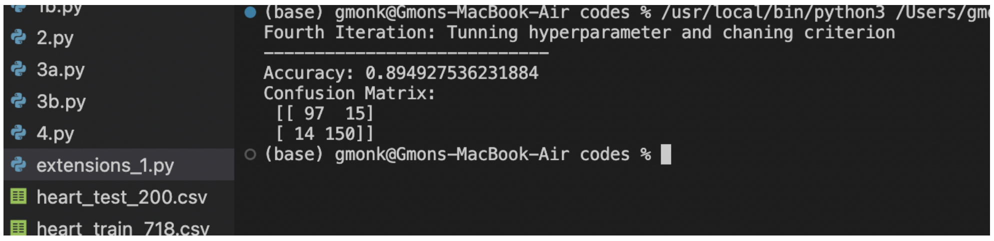
## First Extensions:

I am trying to attempt additional iterations to make your system better. Next, I am changing the criterion used to split nodes from 'gini' to 'entropy'. The intuition behind this change is that 'entropy' is a more complex measure of impurity that takes into account the distribution of class labels at a node, so it may result in a more accurate model. In addition to that, I am further tune hyperparameter the data, that is

1)changing the Criterion

2) Tuning the hyperparameter.

By combining both this approach, I was able to increase the accuracy of the model from 0.84 to 0.894



```
(base) gmonk@Gmons-MacBook-Air codes % /usr/local/bin/python3 /Users/gm
Fourth Iteration: Tunning hyperparameter and chaning criterion
------------------------------
Accuracy: 0.894927536231884
Confusion Matrix:
 [[ 97  15]
 [ 14 150]]
(base) gmonk@Gmons-MacBook-Air codes %
```

Python program to run this ‘extensions_1.py’

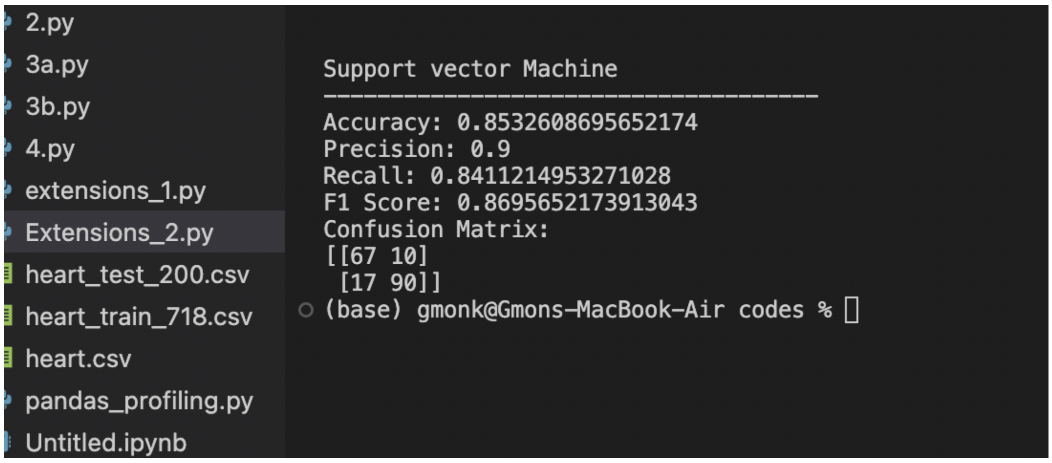
In addition to that, I was able to increase one more true positive value in the confusion matrix, thus I am able to increase the accuracy.

## **Second Extensions:**

As a part of second extensions, I have tried to implement the support vector machine.SVMs are a powerful and versatile machine learning method that can be used for both classification and regression tasks. They work by finding the optimal hyperplane that separates the different classes in the data. SVMs can be tuned using different kernels and regularisation parameters to find the optimal decision boundary for the specific dataset.

I tried to use the below preprocessing techniques and they are:

1) I have tried to fill the missing values, if there are any
2) Converted all the categorical data to numerical data using the hot-encoding technique
3) Standardise the data into a particular range.
4) Applied PCA on the data
5) Applied polynomial feature of degree 2



```
Support vector Machine
-------------------------------------
Accuracy: 0.8532608695652174
Precision: 0.9
Recall: 0.8411214953271028
F1 Score: 0.8695652173913043
Confusion Matrix:
[[67 10]
 [17 90]]
(base) gmonk@Gmons-MacBook-Air codes % 
```

Its accuracy is better than the random forest and Decision Tree but not better than the logistic Regression. Its F1 score is less than both the logistic regression and random forest.

Python program to run this file is ‘extensions_2.py’

## **Summary(short reflection of what you learned):**

Some of the key points that i have learned from the projects are:

- [ ] Learned to do Pre-processing on the data
- [ ] Learned to fill the missing data in the dataset and also how to standardise the dataset into a particular range.
- [ ] Learned how to find the best features from the dataset, using different feature selection methods.
- [ ] Identified significant signals exist in the independent variables.
- [ ] Applied different classic ML models, I have implemented logistic Regression, decision tree, Random forest
- [ ] Evaluation of the models using their confusion matrix, the F1 value, the bias, and the variance.
- [ ] Identified which classifier did the best.
- [ ] Learned about the Receiver Operator Curve [ROC] for the model
- [ ] Learned to do further iteration on the model(I have done on random forest) for further improving the model
- [ ] Learned how to tune hyperparameter and criterion on random forest to improve his efficiency.
- [ ] As a part of the first extensions, I have done one more iteration on the random forest to improve its efficiency.
- [ ] I have implemented a support vector machine and further compared its efficiency with the other models I have implemented before.

## Bibliography:

Some of the importance links and materials i have used for this projects are:

1. https://scikit-learn.org/stable/

2. Textbook by Muller and Guido

3. https://youtu.be/ukzFI9rgwfU