1. Train a Random Forest model n_classifiers and maxdepth set to 10, min_sample_split=20 and max_features set to 11. Use the Gini index as the criterion. Report the accuracy of the training and testing data. [15 marks]

Bash command to run main.py :

```
Python3 main.py --n-classifiers=10 --criterion=gini --maxdepth=10 --min-sample-split=20
--max-features=11
Accuracy of the Training Data : 0.8713798716255643
Accuracy of the Testing Data : 0.850991953811191
```

2. Do the same experiment as 1, but this time with Information Gain as the criterion. How does this change the accuracy of training and test datasets? Discuss it in your report. [10 marks]

Bash command to run main.py :

```
Python3 main.py --n-classifiers=10 --criterion=entropy --maxdepth=10 --min-sample-split=20
—max-features=11
Train Data Accuracy : 0.8692914836767912
Test Data Accuracy: 0.8398132792825993
```

- There is not much of a significant difference between train accuracy when using Gini impurity and entropy as a criterion.
- But the test accuracy decreases when using Entropy as a criterion.
- And Entropy tends to overfit on the data since the difference between the accuracies of train and test data are greater than the difference between the accuracies of train data and test data when using Gini Impurity as a criterion.
- Observed Time taken to execute the Random Forest algorithm with Gini impurity is : 3m 50s
- Observed Time taken to execute the Random Forest algorithm with Entropy is : 4m 50s
- The Gini criterion is much faster because it is less computationally expensive.

3. In your report, explain why the accuracy of the training data is not equal to 100%. [5 marks]

- One of the many factors that play into the accuracy of training data is the model's ability to generalize patterns it sees in the data set. If the model can not find a pattern, the accuracy will suffer.
- There are many independent variables which can potentially affect the accuracy of the model. In our case, we have to take into account many factors which could influence the outcome.In our case, there are 14 independent variables which can affect the accuracy.
- Without proper feature engineering, the model will probably under fit, resulting in decreased accuracy
- In order to improve the accuracy, it is necessary to decrease the size of the training data. This is because with a smaller training data set, there is less room for error.

a) Is there any way to train a tree that yields 100% accuracy on the training dataset? How would that affect the accuracy of the test data? [5 marks]

when a model has 100% accuracy on the training dataset, this means that the model has overfit the data. This is not possible on the test dataset, but if the goal is to increase training accuracy,changing the hyperparameters will do the job but Test data accuracy decreases and model is unable to generalize on new data.

b) What parameters should you change to get a tree with perfect training accuracy? Explain if such a classification model has a high variance or bias? [10 marks]

Parameters to be changed to increase training accuracy includes

- **Increase max_depth** : If the value is higher than the optimal value, the model will tend to overfit. This means that the model will learn the training data too well and won't be able to generalize to new data. But if the value is too low, it can hamper the training process. A low depth can cause the model to underfit, which means that the model doesn't learn the training data well enough. The sweet spot is the optimal value to ensure a good accuracy
- **Increase max_features** : Increasing the maximum number of random features If we increase the maximum number of random features considered in a split, it usually decreases the bias of the model. This is because there is a higher probability that good features will be included. However, this can lead to increased variance. In addition, training speed will decrease if we include more features to test at each node.considered in a split tends to decrease the bias of the model, as there is a better chance that good features will be included, however this can come at the cost of increased variance. Also, there is a decrease in training speed when we include more features to test at each node.
- **Increase n_classifiers** : Having a higher number of trees often leads to improved performance and stability in predictions, though it comes at the price of a slower computation.
- **Decrease min-sample-split** : If we only care about accuracy, We can set those value fairly low in order to get as much node as possible and learn extensively on your training data.

Example of changing parameters to improve the accuracy of the model on training data

- 1st case : changing the max depth from 10 to 16 cause training accuracy massively increase from 86% to 98% .

  ```
  python3 main.py --n-classifiers=10 --criterion=gini --maxdepth=16 --min-sample-split=2
  ```

  ```
  Train Accuracy : 0.9844292251466479
  Test Accuracy : 0.8345310484613967
  ```

- 2nd case : changing the number of classifiers from 10 to 20 caused increase in training accuracy from 86% to 90%.

  ```
  python3 main.py --n-classifiers=20 --criterion=gini --maxdepth=8 --min-sample-split=2
  ```

  ```
  Train Accuracy : 0.9012929578329903
  Test Accuracy : 0.8493950003071065
  ```

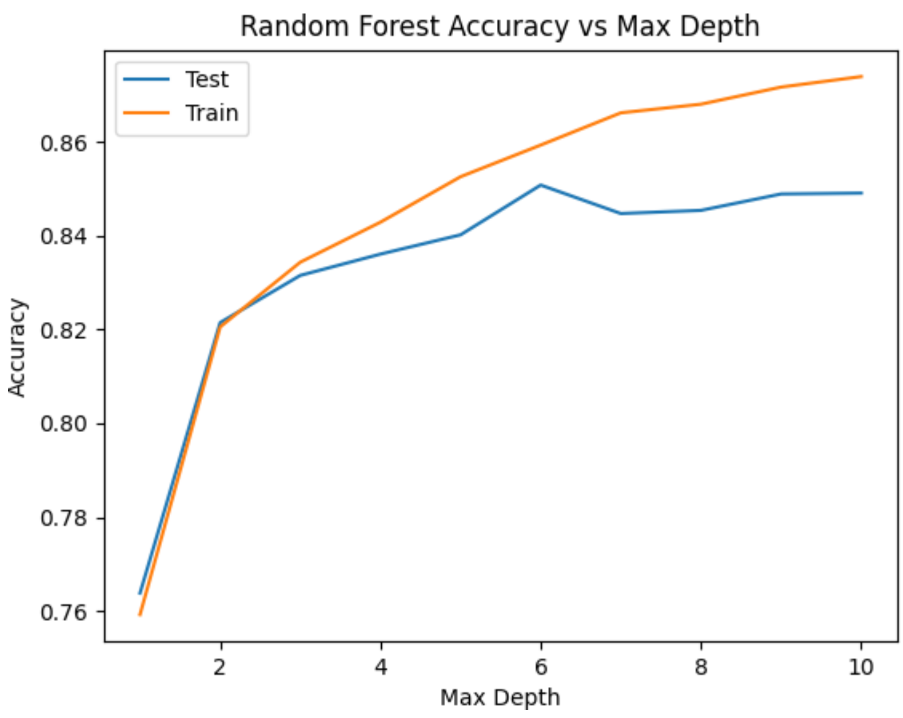
  Both the cases cause overfitting and heavily impacts on the score of testing data hence hyperparameter tuning is necessary to reduce overfitting and generalize the data better.

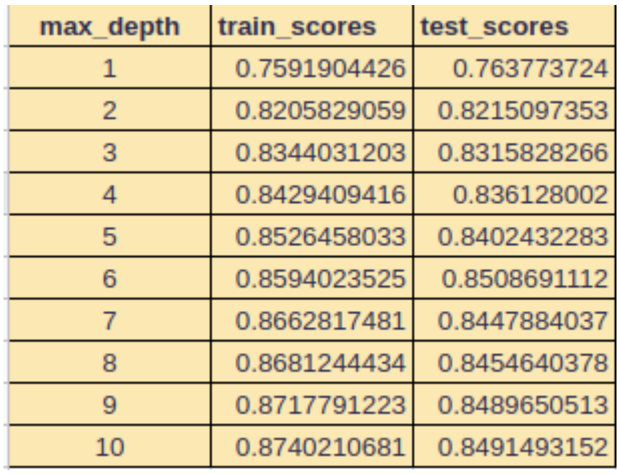
  Such type of classification model have very low bias and high variance because Random Forests are usually built of high-variance, low-bias fully grown decision trees, and their strength comes from the variance reduction that comes from the averaging of these trees. However, if the predictions of the trees are too close to each other than the variance reduction effect is limited, and they might end up overfitting.
  This can happen for example if the dataset is relatively simple, and therefore the fully grown trees perfectly learn its patterns and predict very similarly.

4. Train 10 Random Forest models with min_sample_split=max_features=n_classifiers=10 and maxdepth=[1,2,3,...,10]. Use the Gini index as the criterion. Plot the accuracy of the 10 models on the test dataset in your report. Discuss your results. [20 marks]

Bash command to run main.py :

```
python main.py --run-task4=True
```





max_depth	train_scores	test_scores
1	0.7591904426	0.763773724
2	0.8205829059	0.8215097353
3	0.8344031203	0.8315828266
4	0.8429409416	0.836128002
5	0.8526458033	0.8402432283
6	0.8594023525	0.8508691112
7	0.8662817481	0.8447884037
8	0.8681244434	0.8454640378
9	0.8717791223	0.8489650513
10	0.8740210681	0.8491493152

We can see that the performance of the model initially increases as the number of max_depth increases. But, after a certain point, the train_score keeps on increasing. But the test_score saturates and even starts decreasing towards the end, which clearly means that the model starts to overfit.