Part A

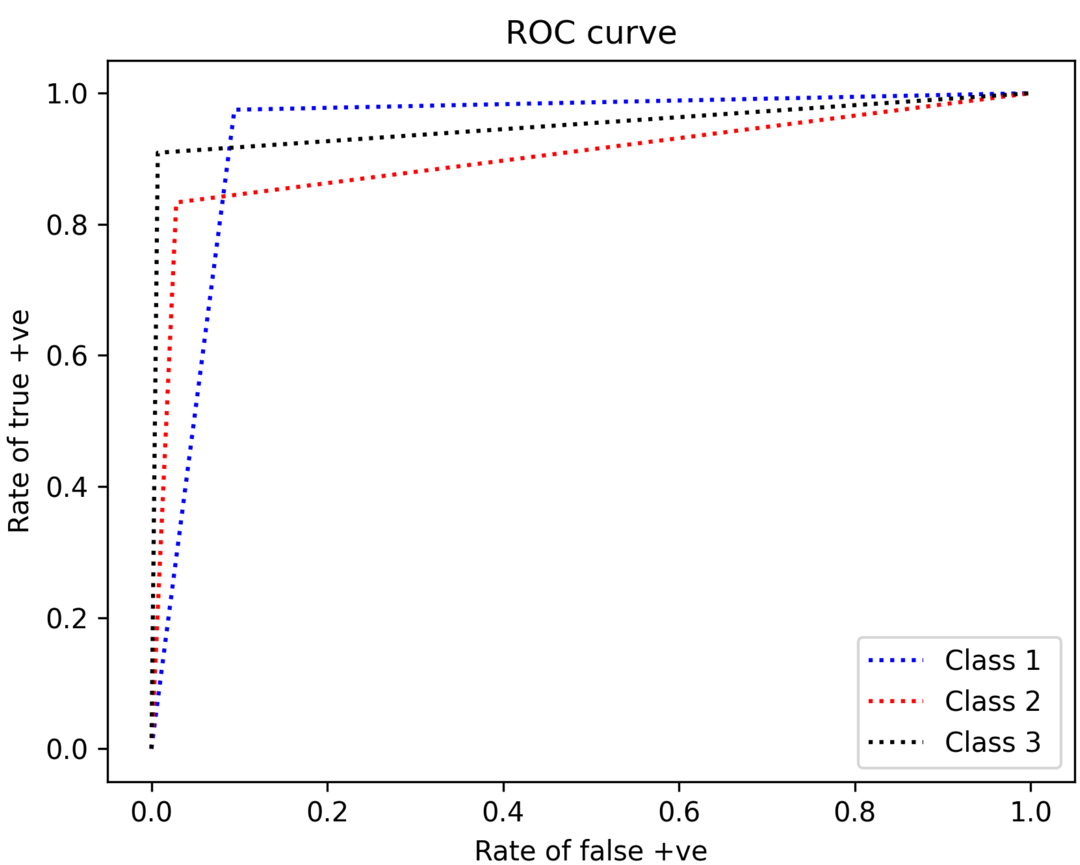
Q1: (5 points) Classify the dataset using Decision Tree classifier and report precision, recall, accuracy and AUC-ROC curve on the test set. Visualize the Decision Tree (DT) and save the visualization as image (image name: DT_A_1.png/pdf/jpg).

```
classifier = DecisionTreeClassifier(criterion='entropy', random_state=0)
```

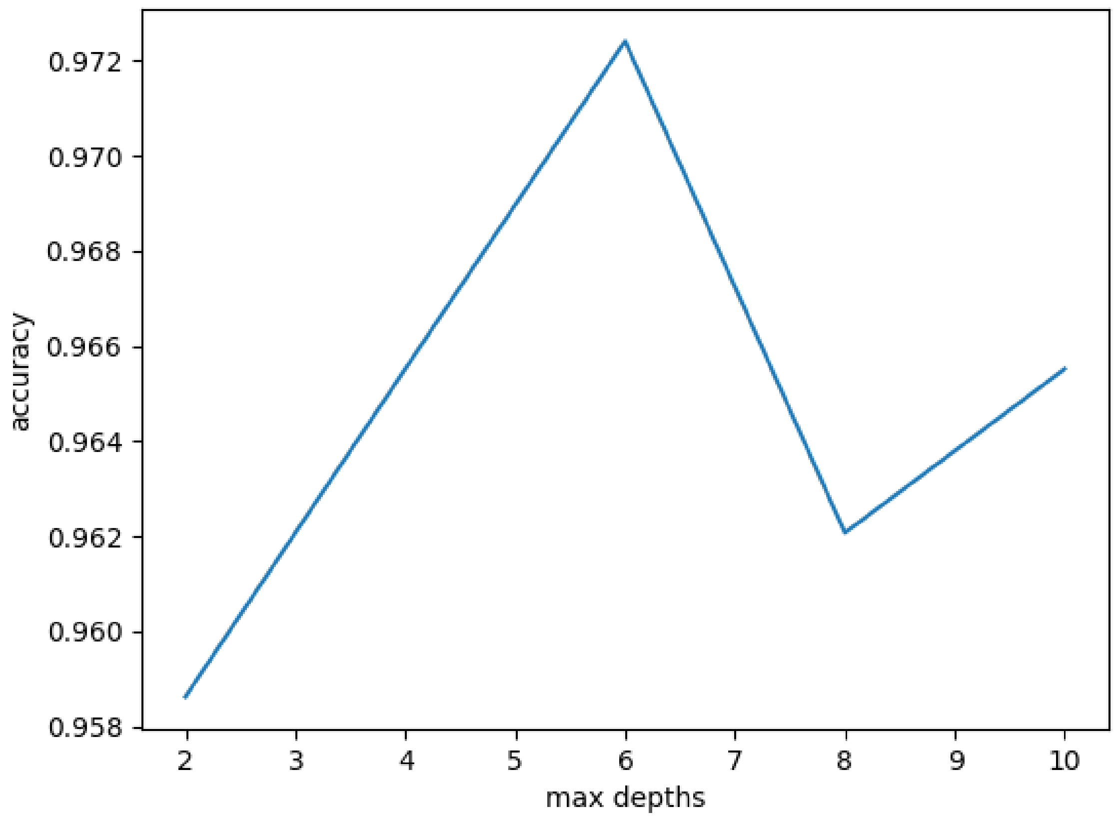
Taking entropy as the default criterion.
Used the Decision tree classifier of scikit learn library
Visualization of Decision tree

```
------------- Decision Tree Results -------------
Precision:  0.9522257938827197
Recall   :  0.9517241379310345
Accuracy :  0.9517241379310345
auc score:  0.9351330560627579

Process finished with exit code 0
```



Note - Decision tree of q1 is large so its present in the visualization folder as DT_A_1.png
Q2: (5 points) Train the Decision Tree classifier on different depths (Minimum 5 different depths) of tree and plot the accuracy vs depth graph.



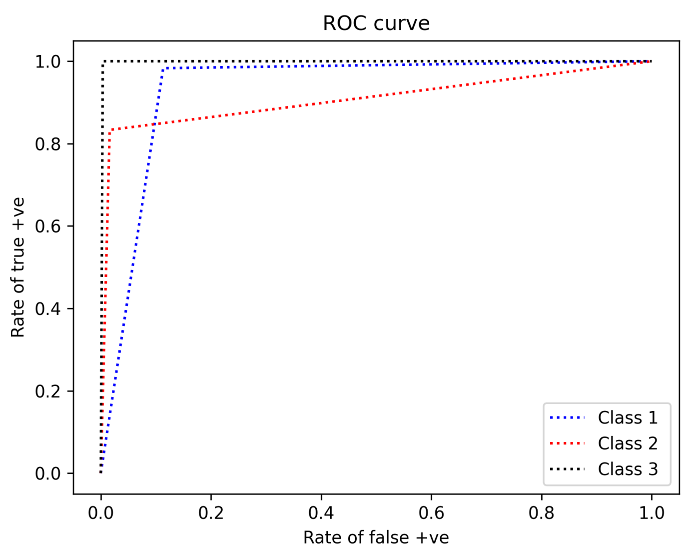

```
depth=[2,4,6,8,10]
```

Trained the decision tree classifier on 5 different random depths and recorded the accuracy for each depth in a list. The corresponding values of depths and accuracy were plotted.The attached graph was thus obtained.

Q3: (20 points) Taking the Decision Tree formulated in Q1, vary the following hyperparameters:- Criterion, Splitter, min samples split, max depth, min samples leaf, max features (sqrt/log2), class weight and max leaf nodes and report the observations (precision, recall, accuracy and AUC-ROC curve) on each. You just have to change one hyperparameter in one experiment keeping others fixed and observe the performance. Minimum observations to take should be 8. Report your best intuition/reasoning behind positive/negative performance scores against the base model (Q1). Keep the best obtained hyperparameters (DT referred as DT-A) from Q3 and carry on to Part B. Do not influence class weight in further experiments and set it as default ("none").

Curve 1:

```
DecisionTreeClassifier(criterion='gini', random_state=0)
```



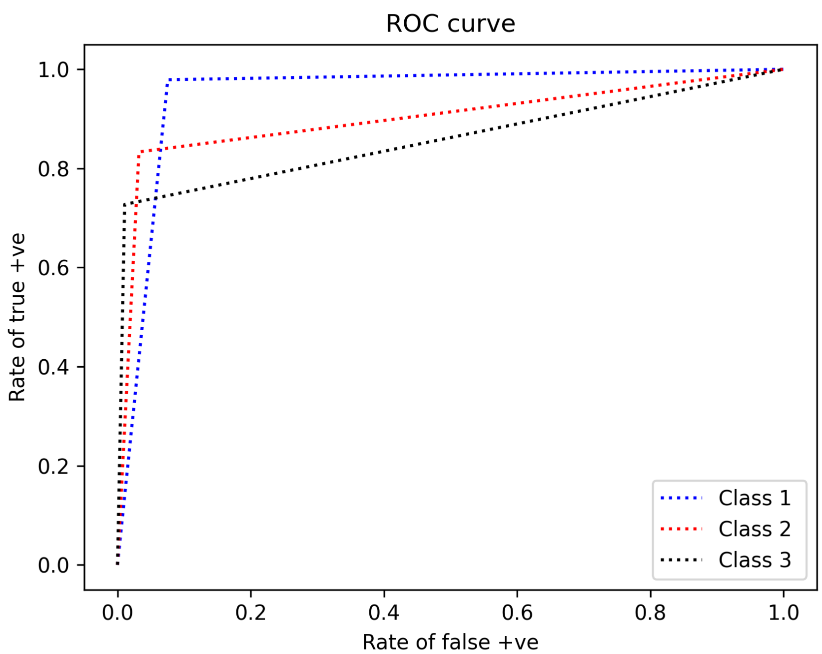
------------- Decision Tree Results -------------
Criterion changed to gini
Precision: 0.9614684490712482
Recall   : 0.9620689655172414
Accuracy : 0.9620689655172414
auc score: 0.9335395955907215

Generally ginni gives a better accuracy than entropy as the range of ginni is from [0,0.5] while the range of entropy[0,1] so if u observe the graph it would be observed that entropy first increases from 0 to 1 and then it decreases but ginni increases from 0 to 0.5

and then it decreases . So it actually needs less computation power than entropy hence our accuracy is increased.

Curve 2:

```
DecisionTreeClassifier(criterion='entropy',random_state=0,splitter='random')
```

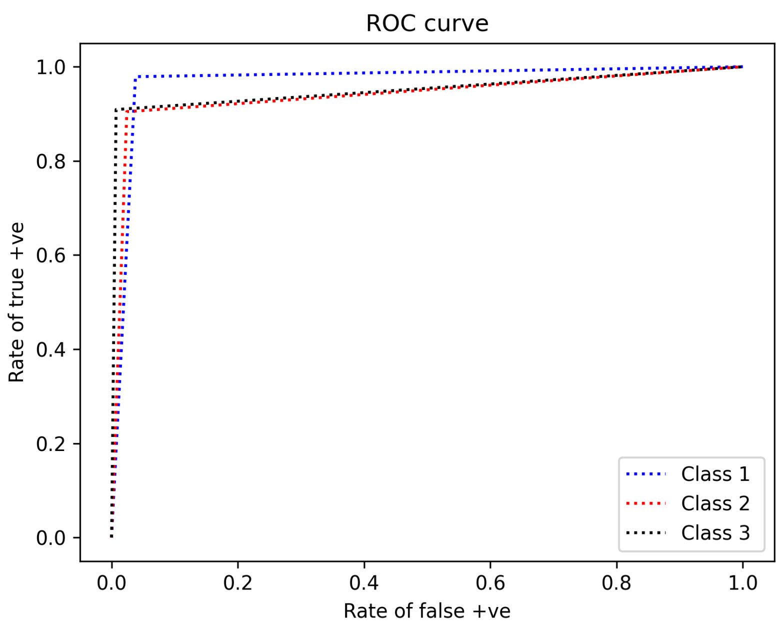


------------- Decision Tree Results -------------
splitter changed to random
Precision: 0.9488589564106398
Recall   : 0.9482758620689655
Accuracy : 0.9482758620689655
auc score: 0.9407587745993102

This is how the decision tree looks for a split in the features. "Best" is the default setting. That is, the algorithm analyses all of the characteristics for each node and selects the optimal split. If you use "random" for the splitter option, a random subset of characteristics will be examined.As we are using random splitter we can observe that the precision and accuracy has decreased as compared to the default 'best'.

Curve 3:

```
DecisionTreeClassifier(criterion='entropy', random_state=0,max_depth=10)
```

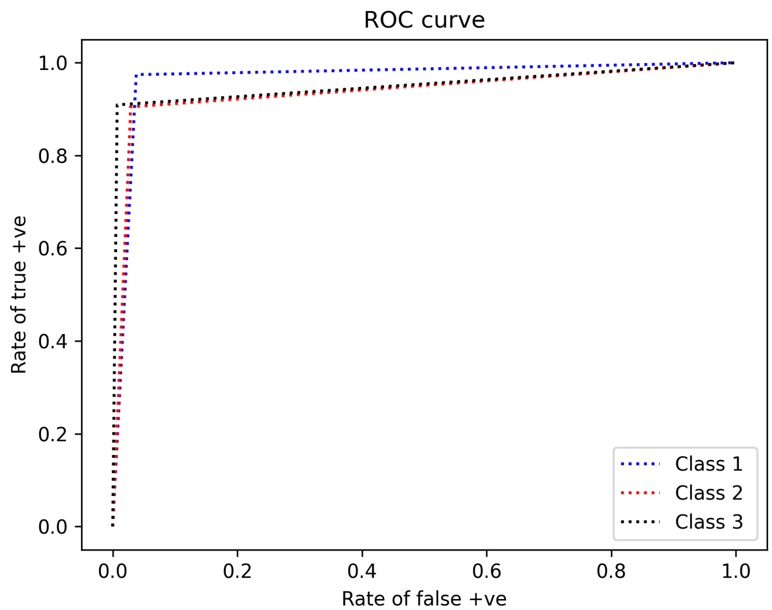


------------- Decision Tree Results -------------
maxdepth changed to 10
Precision: 0.9669439755646653
Recall   : 0.9655172413793104
Accuracy : 0.9655172413793104
auc score: 0.9653582014024434

Here we have increased the max depth hence we can have more nodes so it increases the accuracy also this can be observed in the Q2 (A) part that accuracy increases for depth 10

Curve 4:

```
DecisionTreeClassifier(criterion='entropy', random_state=0,min_samples_split=5)
```

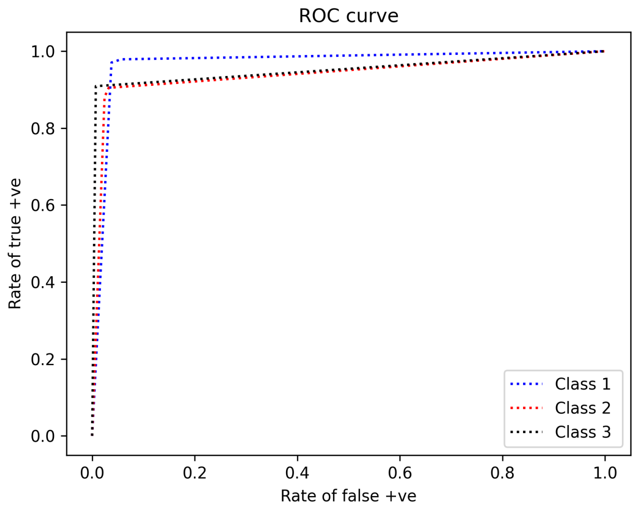


------------- Decision Tree Results -------------
min Sample Split changed to 5
Precision: 0.9601599549875414
Recall   : 0.9586206896551724
Accuracy : 0.9586206896551724
auc score: 0.9631910922455185

The bare minimum of samples that a node must have in order to be considered for splitting. This option may be used to make your tree more regular. As two is the default value by changing the parameter to five we could see changes in the precision recall and accuracy. thus in this case the node must have at least 5 sample for a split.

Curve 5:

```
DecisionTreeClassifier(criterion='entropy', random_state=0,min_samples_leaf=2)
```

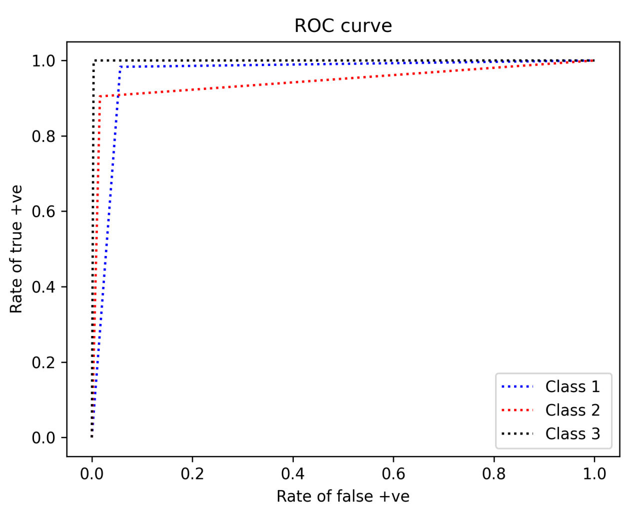


------------- Decision Tree Results -------------
min Sample leaf changed to 2
Precision: 0.963036791846531
Recall   : 0.9620689655172414
Accuracy : 0.9620689655172414
auc score: 0.9649819972342901

Minimum sample leaf is the number of samples used to declare a node as leaf node. Here we have considered it as 2 so it requires only 2 samples to be regarded as a leaf node

Curve 6:

```
DecisionTreeClassifier(criterion='entropy', random_state=0,max_features='sqrt')
```



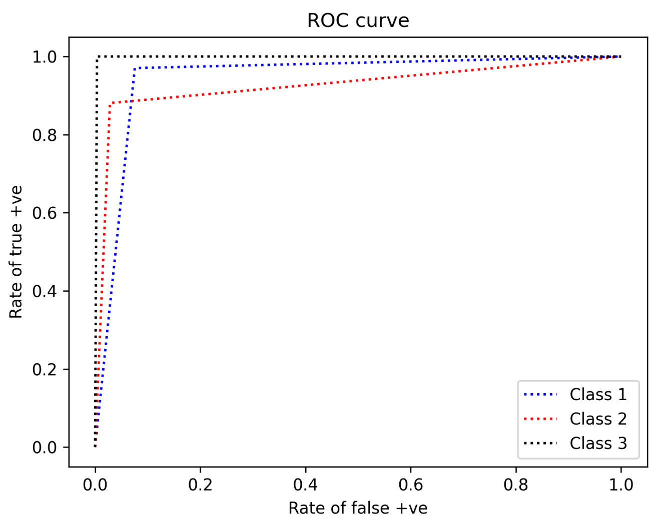
------------- Decision Tree Results -------------
max features changed to sqrt

Precision: 0.9726573154100914
Recall : 0.9724137931034482
Accuracy : 0.9724137931034482
auc score: 0.9618414823831745

When looking for the optimum split, there are a lot of factors to consider. If this value is not specified, the decision tree will use all available characteristics to determine the optimal split. But in this case we have specified the max features as sqrt. So we take the square root of all the features and among them we find the best split possible.We can observe our precision recall and accuracy improved.So tuning the max feature allowed can help us improve our performance against the best model.

Curve 7:

```
DecisionTreeClassifier(criterion='entropy', random_state=0,class_weight='balanced')
```

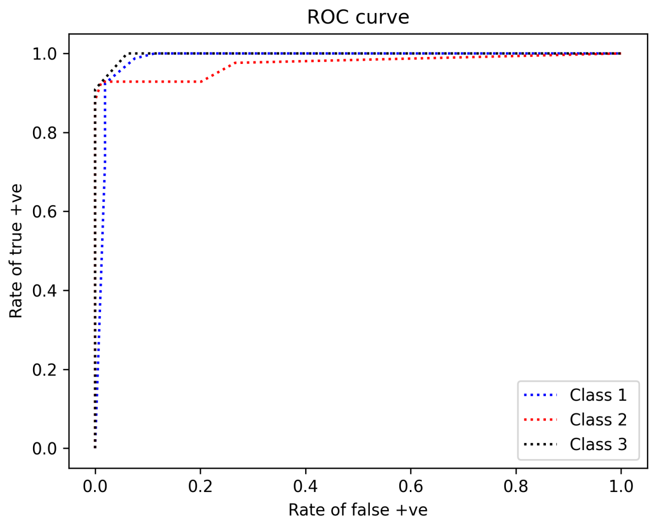


------------- Decision Tree Results -------------
class weight changed to balanced
Precision: 0.9598283900008038
Recall : 0.9586206896551724
Accuracy : 0.9586206896551724
auc score: 0.9463591330222493

Class weight is actually used to give preference to features as we have used to balance class weight hence each feature would be given equal weight and hence we observe almost negligible change in this section.

Curve 8:

```
DecisionTreeClassifier(criterion='entropy', random_state=0,max_leaf_nodes=8)
```



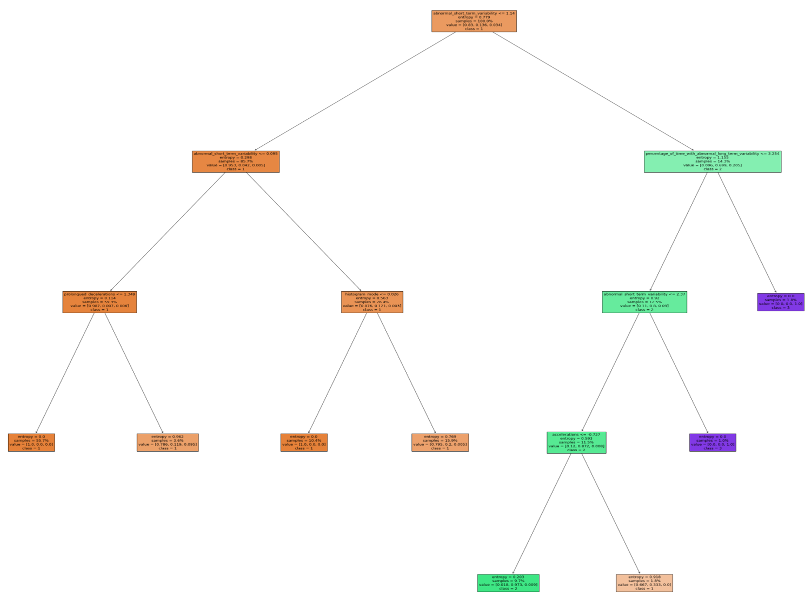
------------- Decision Tree Results -------------
max leaf node changed to 8
Precision: 0.9798212005108557
Recall   : 0.9793103448275862
Accuracy : 0.9793103448275862
auc score: 0.9839883821530097

In simple words Max leaf nodes in the decision tree are those nodes which don't have any further childs. So by limiting the leaf nodes of a decision tree we limit the growth of the tree.We can observe a gain in the performance score of our tree as compared to the base model.this helps our Decision tree from overfitting and thus choosing a right set of value can improve the precision recall and accuracy of model.

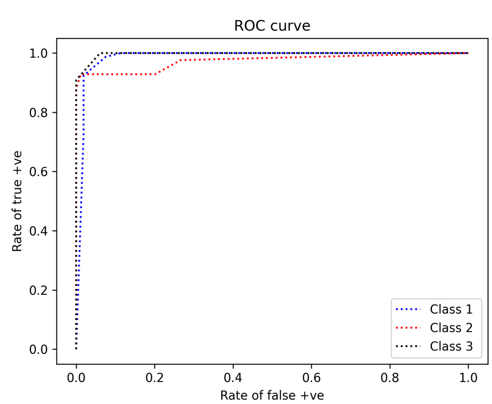
Best performing model is the model with max leaf nodes = 8.

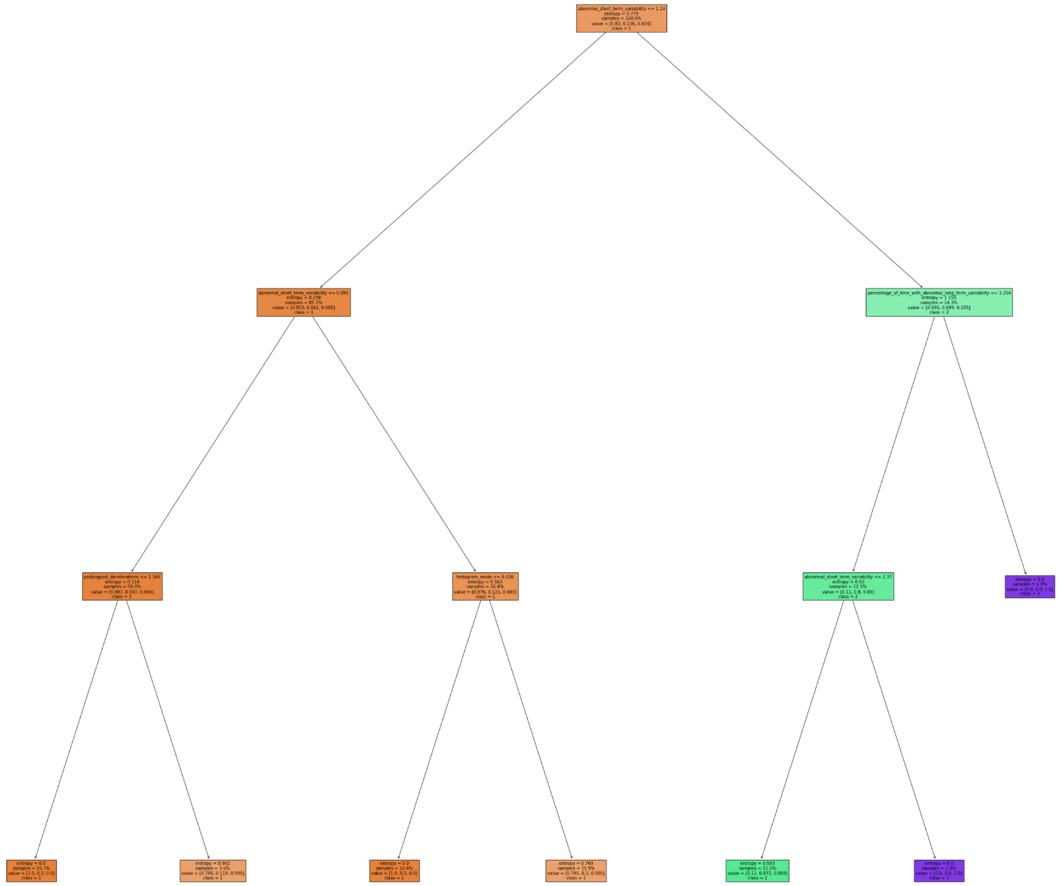
PART B

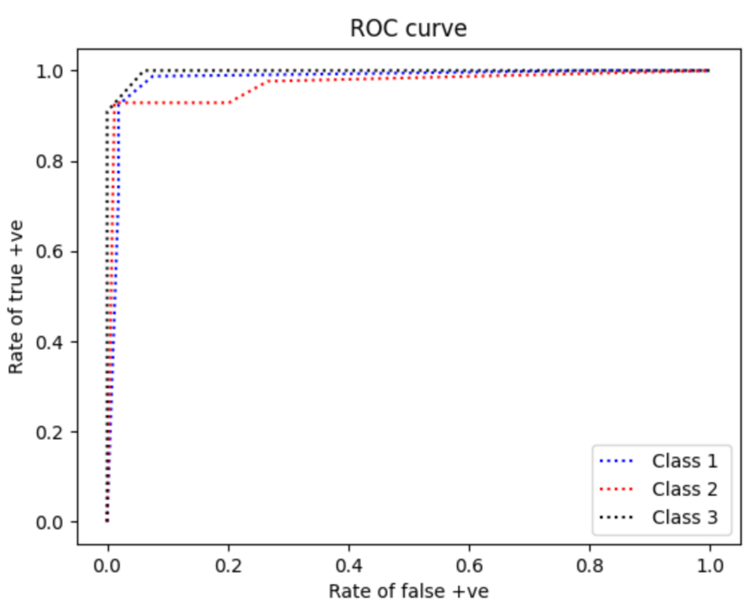
Q1: (5 points) Remove a random node from the DT-A and observe the changes. Report your obser-vations in terms of performance (precision, recall, AUC-ROC curve and accuracy) along with the tree diagram. Save the diagram as DT_B_1.png/pdf/jpg.



------------- Decision Tree Results -------------
max leaf node changed to 8
Precision: 0.9798212005108557
Recall   : 0.9793103448275862
Accuracy : 0.9793103448275862
auc score: 0.9839883821530097






ROC curve
1.0
0.8
0.6
0.4
0.2
0.0
Rate of true +ve
0.0
0.2
0.4
0.6
0.8
1.0
Rate of false +ve
Class 1
Class 2
Class 3

------------- Decision Tree Results -------------
Precision: 0.9798212005108557
Recall   : 0.9793103448275862
Accuracy : 0.9793103448275862
auc score: 0.9796057610016254

Results after a random node is deleted from DT-A. We can observe that there is not very significant change in the performance measure such as precision, recall and accuracy as compared to the best model we obtained from part A. But we can observe that by removing a random node from the tree the AUC score of the tree has slightly decreased. Although the change is very minute.

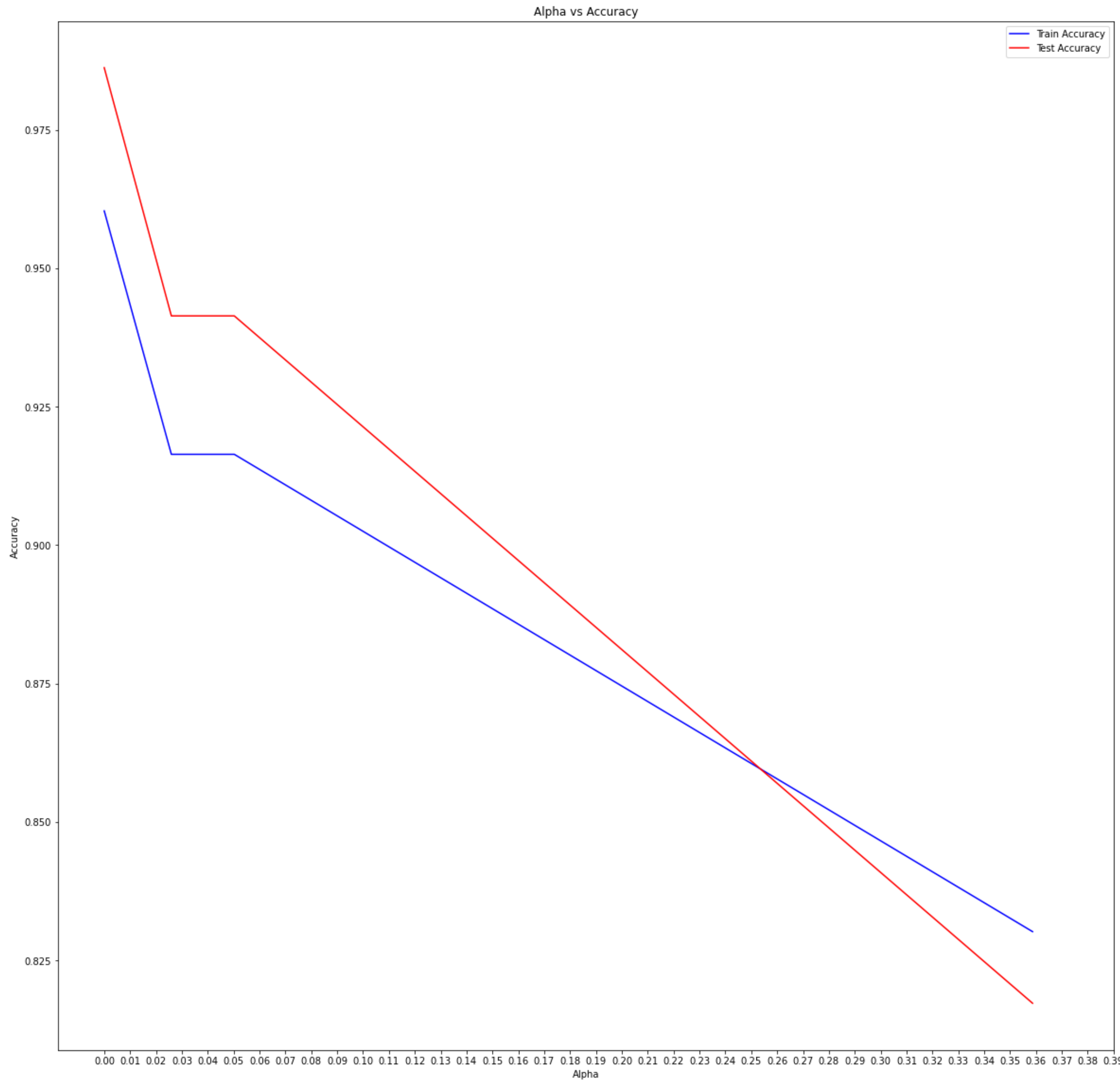
Q2: (15 points) Apply the Cost Complexity pruning technique and any other pruning technique of your choice on the DT obtained from part A (DT-A). Report the value of alpha for Cost Complexity Pruning and parameter values for the second pruning technique (ex. alpha, beta values if using Alpha-Beta pruning) and precision, recall and accuracy between the DT-A and the pruned trees. Visualize the DT-B-2-CC and DT-B-2-XX and save the image as DT_B_2_CC.png/pdf/jpg and DT_B_2_XX.png/pdf/jpg.

Taking the model as the best model obtained from part A.
Thus the hyperparameter value of max_leaf_node is set to eight for further models.

```
classifier_ccp=DecisionTreeClassifier(criterion='entropy',max_leaf_nodes=8,
random_state=0,ccp_alpha=alpha)
```

------------- Decision Tree Results -------------
DT-A
Precision: 0.9798212005108557
Recall   : 0.9793103448275862
Accuracy : 0.9793103448275862

------------- Decision Tree Results -------------
Cost Complexity pruning
Precision: 0.9798212005108557
Recall   : 0.9793103448275862
Accuracy : 0.9793103448275862



Alpha is max in the range 0.00 and 0.01.

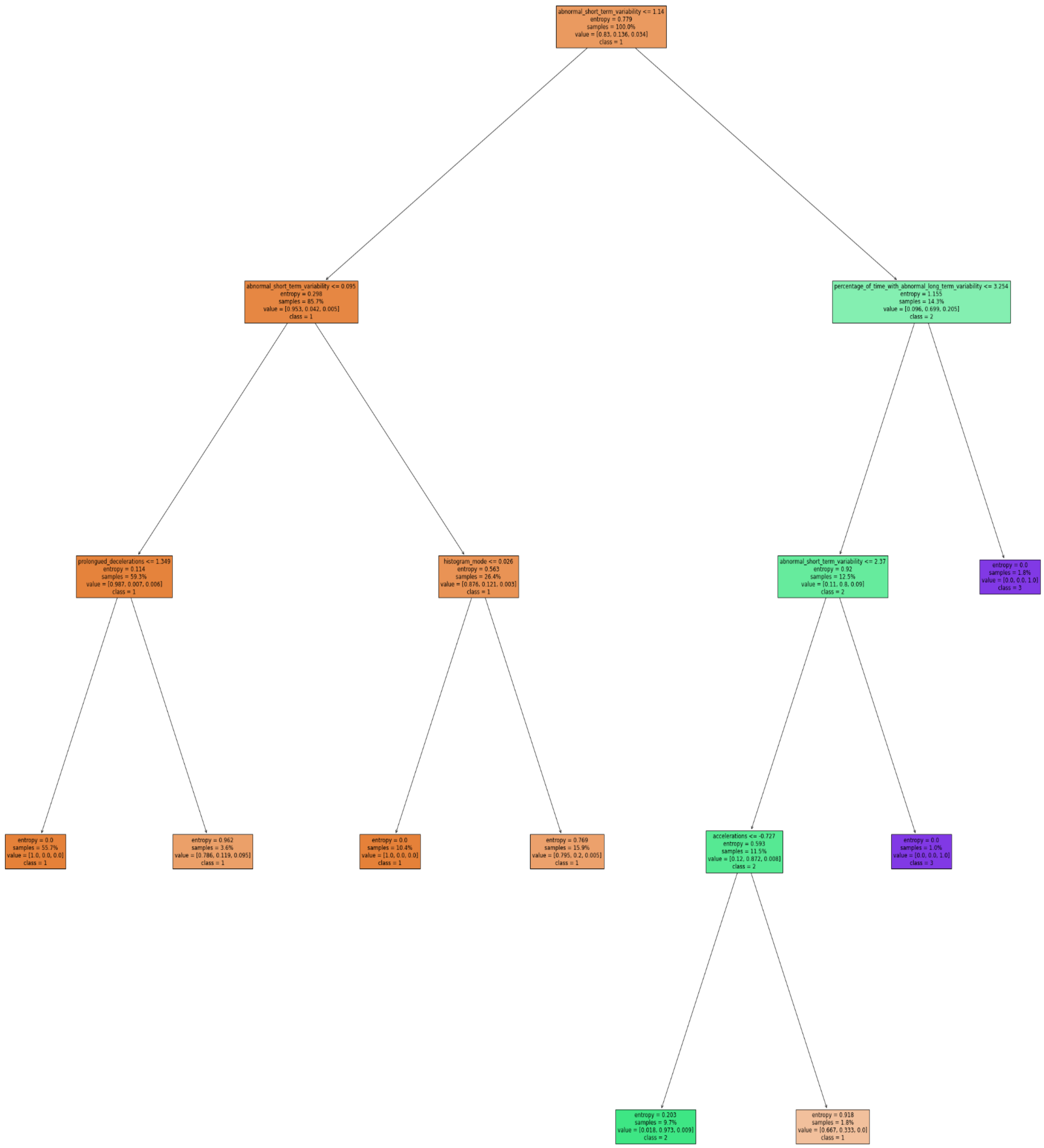
On observing this accuracy vs alpha graph it is observed that our max accuracy occurs for the value of alpha = 0.0 hence we will not observe any significant change in the performance of our cost pruned model because the default value of alpha is also 0.0.

------------- Decision Tree Results -------------
Pre Pruning
Precision: 0.9863508161885444
Recall   : 0.9862068965517241
Accuracy : 0.9862068965517241
Pre Pruning Parameters {'criterion': 'gini', 'max_depth': 5, 'max_features': None, 'max_leaf_nodes': 8, 'splitter': 'best'}

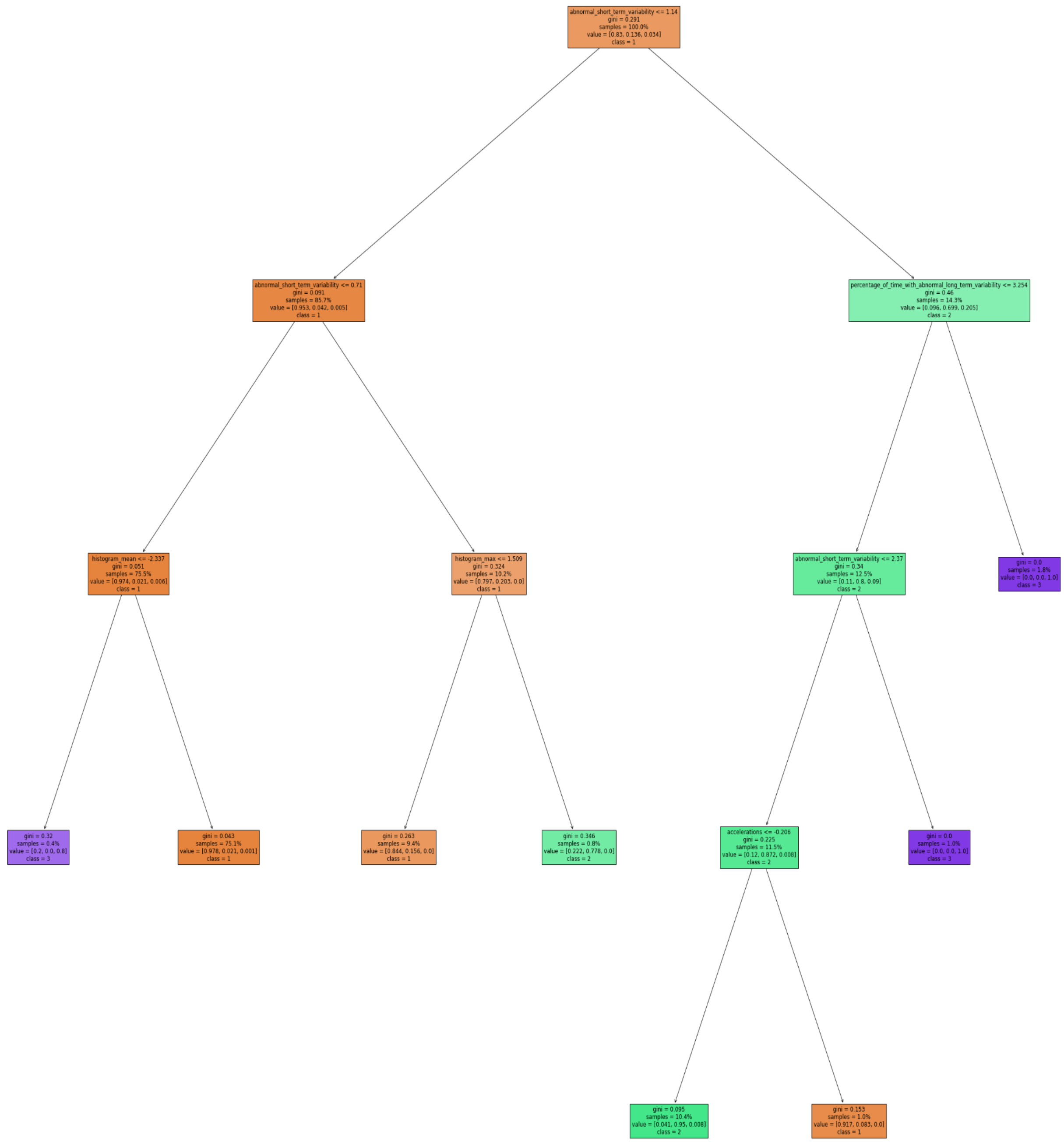
```
GridSearchCV(estimator=classifier,param_grid=parameter)
```

In the pre pruning technique we did a grid search over the dataset and computed the value of best parameters for the above example which would improve the performance of our decision tree model. As the Grid Search finds the best possible combination among the given parameters the recall precision and accuracy of our model has increased.
The Decision Tree for the cost Complexity pruning as well as the Pre pruning Technique as attached below.Also max_leaf_node were fixed to eight as obtained from our base model in part A.

Decision tree for Cost Complexity pruning technique



Decision tree for Pre pruning technique.


abnormal_short_term_variability <= 1.14
gini = 0.291
samples = 100.0%
value = [0.83, 0.136, 0.034]
class = 1
abnormal_short_term_variability <= 0.71
gini = 0.091
samples = 85.7%
value = [0.953, 0.042, 0.005]
class = 1
percentage_of_time_with_abnormal_long_term_variability <= 3.254
gini = 0.46
samples = 14.3%
value = [0.096, 0.699, 0.205]
class = 2
histogram_mean <= -2.337
gini = 0.051
samples = 75.5%
value = [0.974, 0.021, 0.006]
class = 1
histogram_max <= 1.509
gini = 0.324
samples = 10.2%
value = [0.797, 0.203, 0.0]
class = 1
abnormal_short_term_variability <= 2.37
gini = 0.34
samples = 12.5%
value = [0.11, 0.8, 0.09]
class = 2
gini = 0.0
samples = 1.8%
value = [0.0, 0.0, 1.0]
class = 3
gini = 0.32
samples = 0.4%
value = [0.2, 0.0, 0.8]
class = 3
gini = 0.043
samples = 75.1%
value = [0.978, 0.021, 0.001]
class = 1
gini = 0.263
samples = 9.4%
value = [0.844, 0.156, 0.0]
class = 1
gini = 0.346
samples = 0.8%
value = [0.222, 0.778, 0.0]
class = 2
accelerations <= -0.206
gini = 0.225
samples = 11.5%
value = [0.12, 0.872, 0.008]
class = 2
gini = 0.0
samples = 1.0%
value = [0.0, 0.0, 1.0]
class = 3
gini = 0.095
samples = 10.4%
value = [0.041, 0.95, 0.008]
class = 2
gini = 0.153
samples = 1.0%
value = [0.917, 0.083, 0.0]
class = 1

Part C

Q1: (15 points)Train your decision tree on the training data provided ”data_1” and obtain the classifier DT-C-1. You are provided with additional data ”data_2” similar to the training data ”data_1”.You have to make DT-C-1 augment the new data and formulate DT-C-1-X which will be a new DT for both data_1 and data_2. You can not train the DT from scratch on both the datasets. Obtain the DT-C-1-X and compare its performance (precision, recall, accuracy and AUC-ROC curve) with the DT-C-1 on ”data_1” test set and ”data_2” test set (test set is the remaining 20% split from each data source) and report precision, recall, accuracy and AUC-ROC curve.

VFDT Very Fast Decision Tree (Hoeffding Tree Classifier) is used to implement the incremental decision tree.The model was first trained on the data_1.csv and then the model was augmented to formulate the new incoming data from data_2.csv .

```
ht = HoeffdingTreeClassifier()
y_pred = ht.predict(X_train)
ht = ht.partial_fit(X_train, Y_train)
```

------------- Decision Tree Results -------------

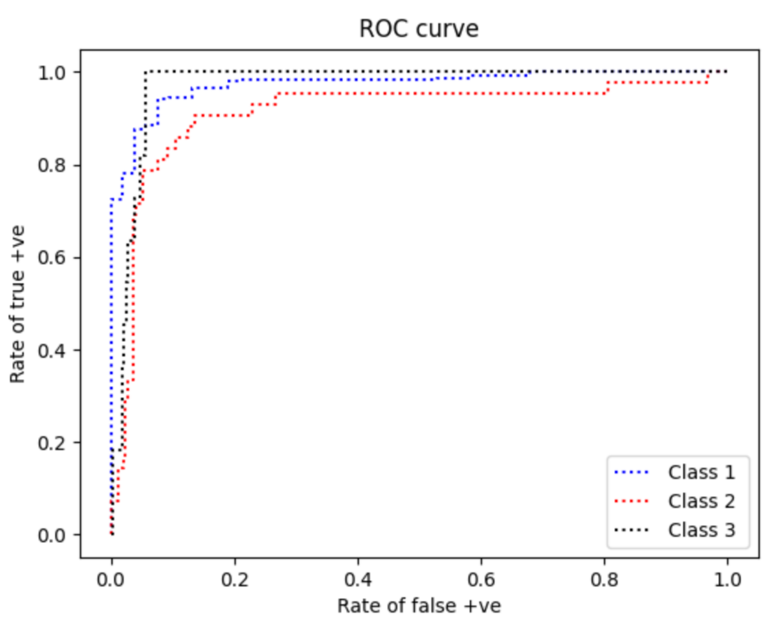
performance of testset of data1 on dc-1

Precision: 0.8416843501326261

Recall   : 0.8620689655172413

Accuracy : 0.8620689655172413

auc score: 0.9644524081031101



------------- Decision Tree Results -------------

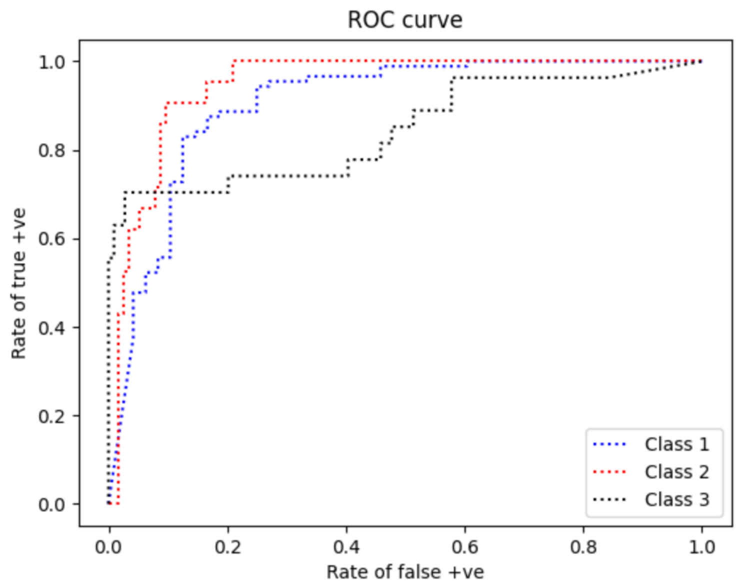
Performance of testset of data2 on dc-1

Precision: 0.8215125518337347

Recall   : 0.8235294117647058

Accuracy : 0.8235294117647058

auc score: 0.8985008398054075



------------- Decision Tree Results -------------

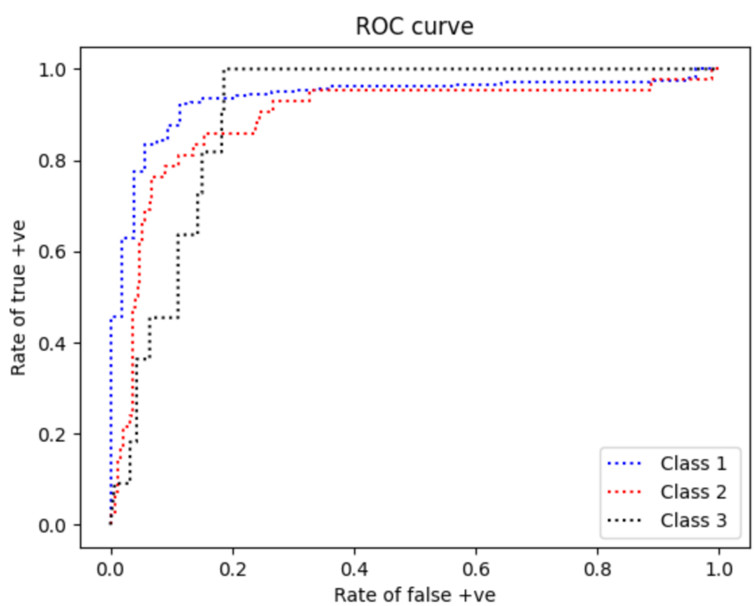
Performance of testset of data1 on dc-2

Precision: 0.8105141966580313

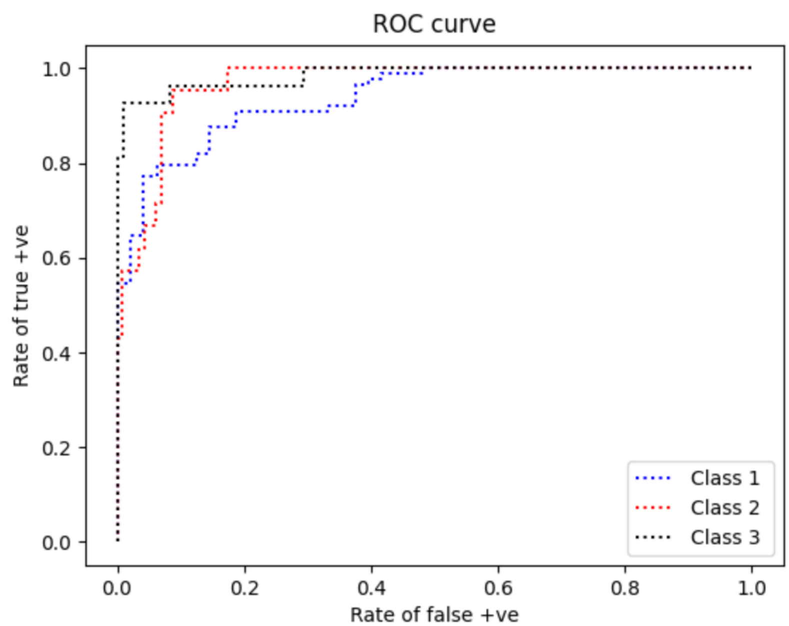
Recall   : 0.8310344827586207

Accuracy : 0.8310344827586207

auc score: 0.928692199345651



------------- Decision Tree Results -------------
Performance of testset of data2 on dc-2
Precision: 0.8585004969729828
Recall   : 0.8382352941176471
Accuracy : 0.8382352941176471
auc score: 0.951871194478988



It is observed that after augmenting the new dataset our accuracy has decreased but the auc score has increased this is because accuracy is calculated for a threshold value of 0.5 on the other hand auc score gives us how good the model will perform while classifying. UC score is the adding up of all accuracies for each threshold value.Also accuracy performs well for balanced splitting of data into test and train but we have done random splitting in 80:20 ratio .

Learnings

- Formation of decision tree using gini and entropy impurities
- Calculating the performance scores of our trained models
- Significance of performance scores
- Effect of different hyperparameters on decision tree
- Visualization of decision trees
- Traversing and removing nodes in decision tree
- Cost complexity alpha pruning technique
- Pre pruning technique using grid_search
- Incremental Decision tree model which can improve its performance by fitting data sets in real time.
- learned about the use of sklearn .library.

REFERENCES:

https://towardsdatascience.com/scikit-learn-decision-trees-explained-803f3812290d
https://www.analyticsvidhya.com/blog/2020/03/beginners-guide-random-forest-hyperparameter-tuning/
https://colab.research.google.com/drive/1hVR7TBruaxyjXtDWkidUtfl2RnhWrHwQ#scrollTo=eSBDKys2Wu_A
https://scikit-multiflow.readthedocs.io/en/stable/api/generated/skmultiflow.trees.HoeffdingTreeClassifier.html#skmultiflow.trees.HoeffdingTreeClassifier
https://www.cs.waikato.ac.nz/~eibe/pubs/thesis.final.pdf
https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/
https://stackoverflow.com/questions/49428469/pruning-decision-trees