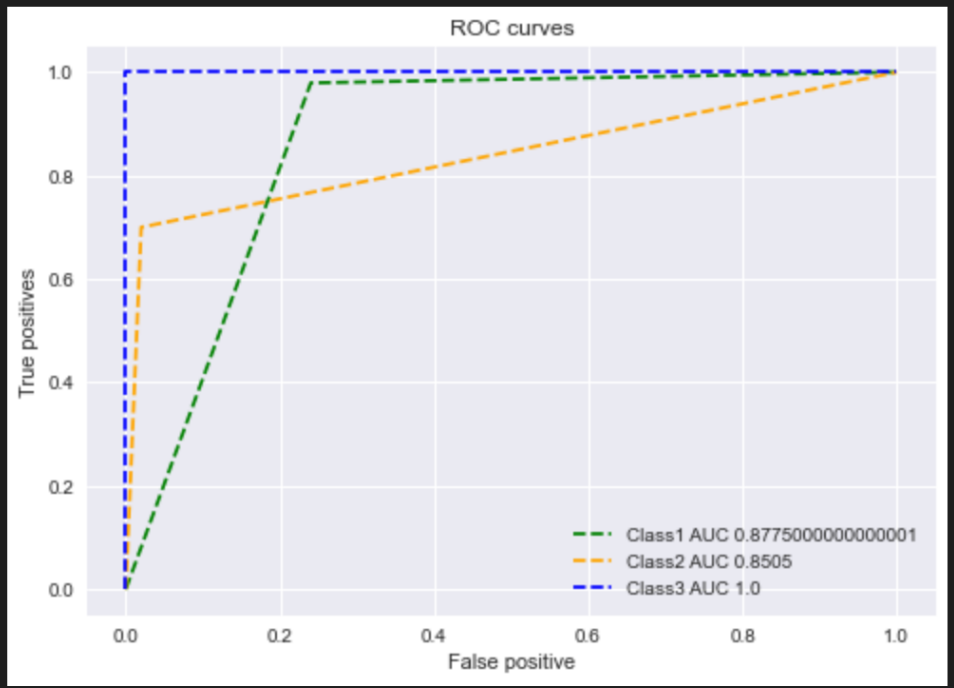
DMG Assignment 1

Shashank Chaurasia 2019105
Vaibhav Girish 2019121

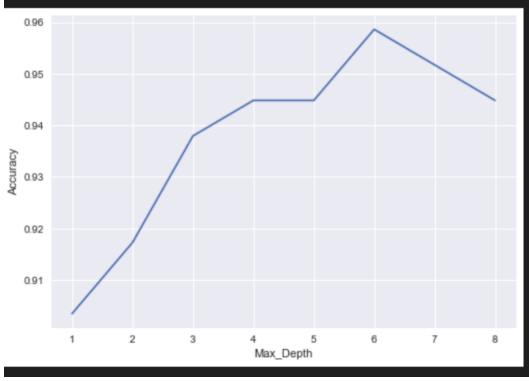
Assumptions:
Part A Q1 - The question asks to classify the dataset using Decision Tree Classifier. We used the sklearn.tree.DecisionTreeClassifier to classify the dataset. We split the dataset into train and test sets. We used the default criterion which is the 'gini index'. We then used sklearn.metrics.accuracy_score to determine the accuracy of the Decision tree and used sklearn.metrics.classification_report to determine the recall, accuracy and other scores. We then plot the roc_curve with the auc score mentioned for each class.

```
Model accuracy score: 0.9414
              precision    recall  f1-score   support

           1       0.96      0.97      0.96       240
           2       0.83      0.72      0.77        40
           3       1.00      1.00      1.00        10

    accuracy                           0.94       290
   macro avg       0.93      0.90      0.91       290
weighted avg       0.94      0.94      0.94       290
```



Part A Q2 - For this question we tested out different max_depths of the DecisionTreeClassifier ranging from values 1 to 8 and plotted their corresponding accuracy scores.

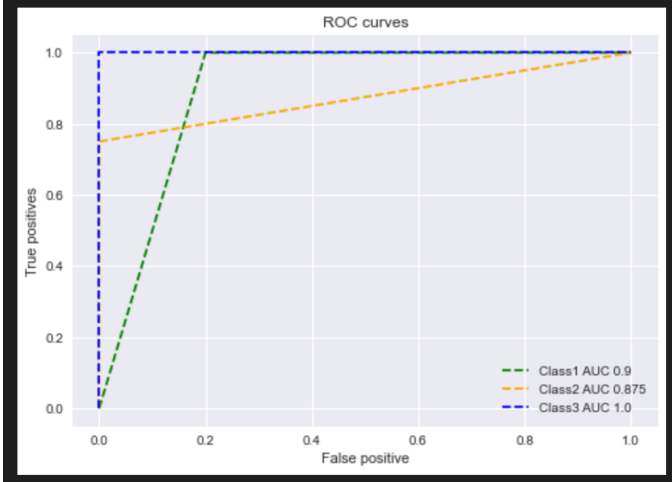
```
{1: 0.903448275862069, 2: 0.9172413793103448, 3: 0.9379310344827586, 4: 0.9448275862068966, 5: 0.9448275862068966, 6: 0.9586206896551724, 7: 0.9517241379310345, 8: 0.9448275862068966}
```



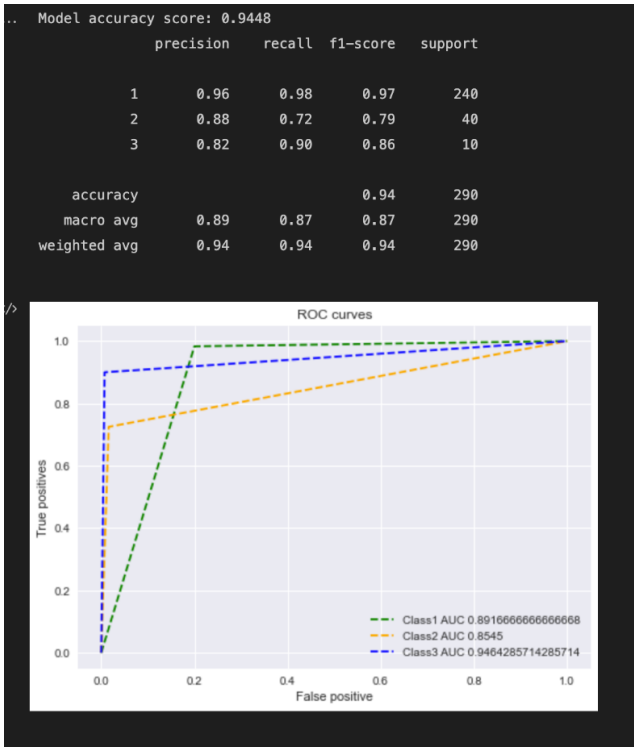
Part A Q3 - For this question we varied each of the parameters and used hit and trial to vary the values of different hyperparameters.

1. Criterion: For this hyperparameter the entropy value gave a higher accuracy, the reason for this could be because entropy values have a higher range [0,1] compared to [0,0.5] in the gini index.

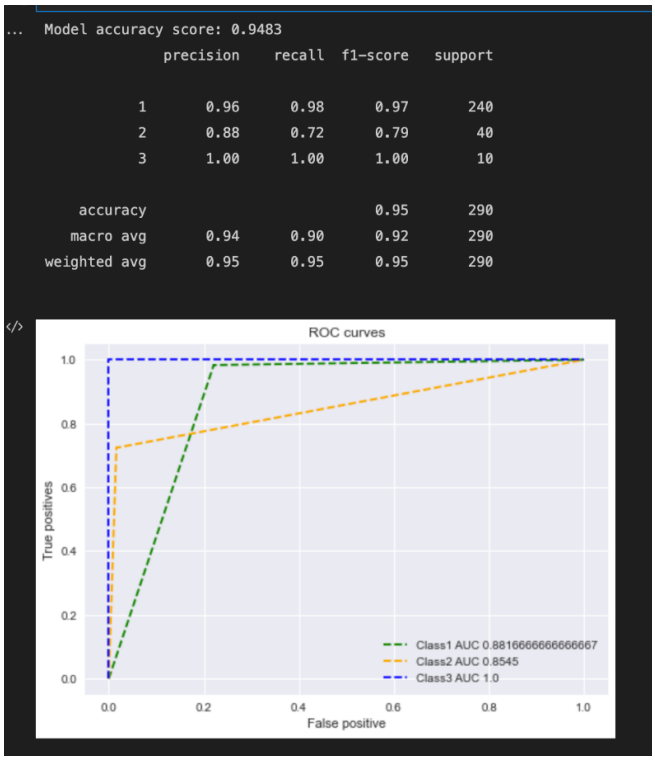
```
Model accuracy score: 0.9655
              precision    recall  f1-score   support

           1       0.96      1.00      0.98       240
           2       1.00      0.75      0.86        40
           3       1.00      1.00      1.00        10

    accuracy                           0.97       290
   macro avg       0.99      0.92      0.95       290
weighted avg       0.97      0.97      0.96       290
```



2. Splitter: For this hyperparameter the ‘best’ value gave a higher accuracy than ‘random’, the plots and scores for ‘random’ are as follows. The ‘best’ values are the default ones plotted above in Q1.

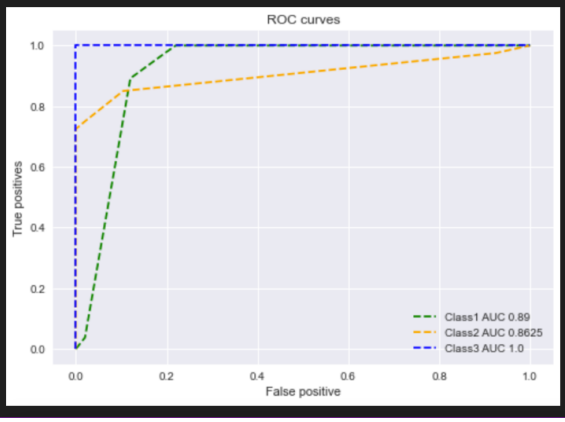


3. Min_samples_split: For this hyperparameter the value 3 gives the highest accuracy, the plots and scores are as follows.



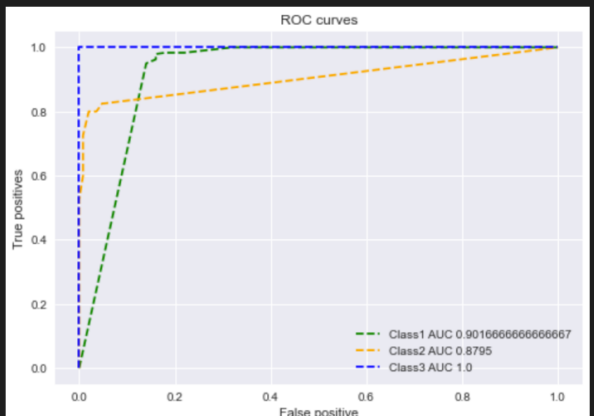
4. Max_depth: For this hyperparameter the value 6 gives the highest accuracy.The plots and scores are as follows.

```
Model accuracy score: 0.9621
              precision    recall  f1-score   support

           1       0.96      1.00      0.98       240
           2       1.00      0.72      0.84        40
           3       1.00      1.00      1.00        10

    accuracy                           0.96       290
   macro avg       0.99      0.91      0.94       290
weighted avg       0.96      0.96      0.96       290
```

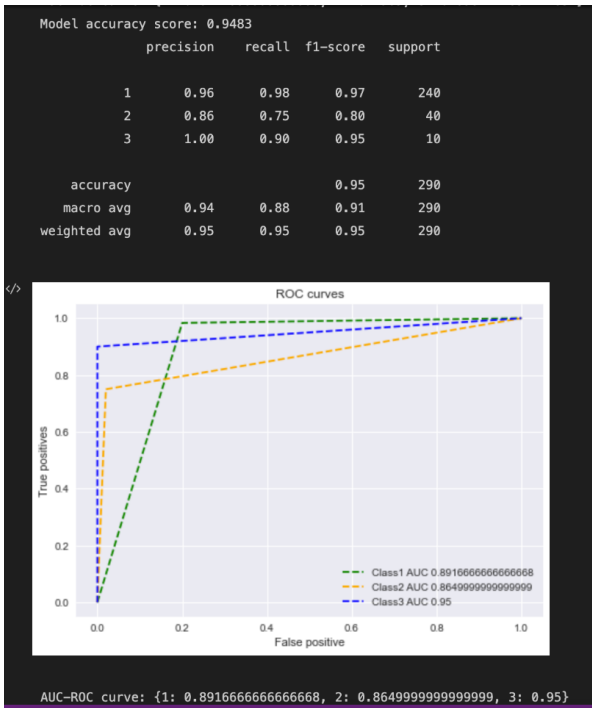


5. Min_samples_leaf: For this hyperparameter the value 6 gives the highest accuracy.The plots and scores are as follows.

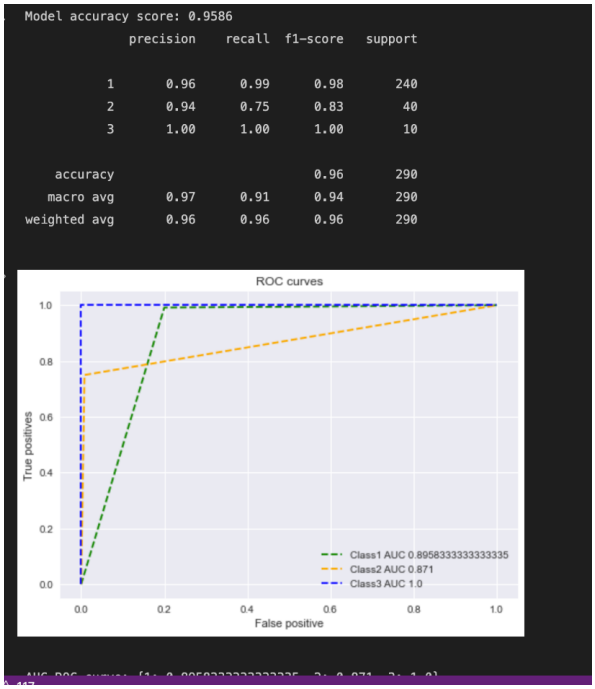
```
Model accuracy score: 0.9552
              precision    recall  f1-score   support

           1       0.96      0.98      0.97       240
           2       0.89      0.78      0.83        40
           3       1.00      1.00      1.00        10

    accuracy                           0.96       290
   macro avg       0.95      0.92      0.93       290
weighted avg       0.95      0.96      0.95       290
```



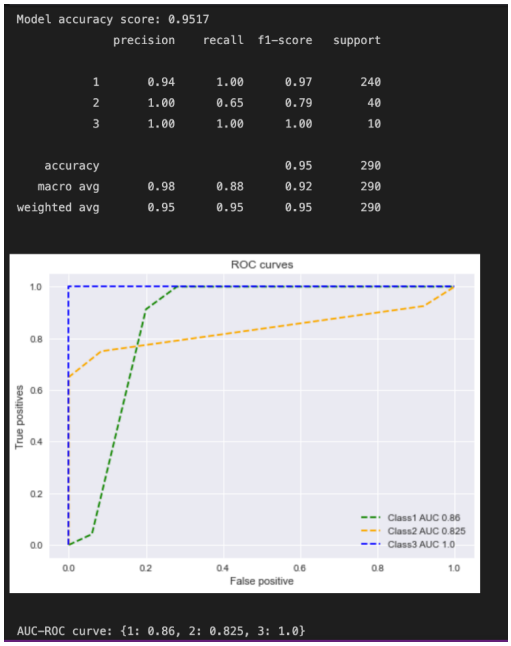
6. Max_features: For this hyperparameter the value ‘log2’ gives the highest accuracy.The plots and scores for ‘log2’’ are as follows.



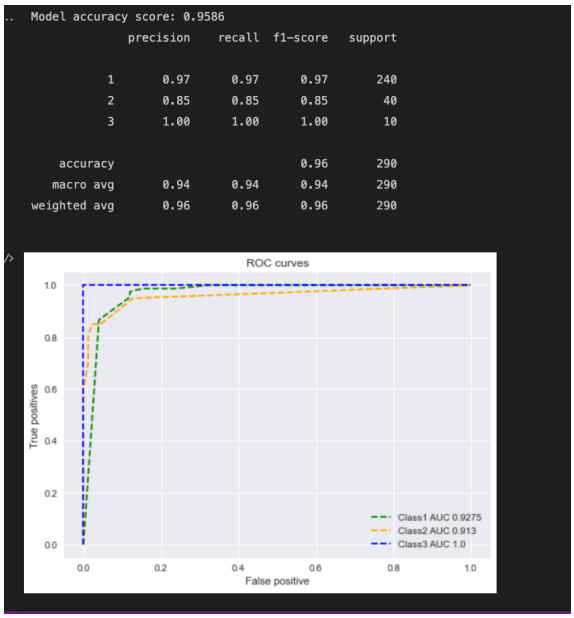
7. Class_weight: For this hyperparameter the value ‘balanced’ gives the highest accuracy.The plots and scores for ‘balanced’ are as follows.



8. Max_leaf_nodes: For this hyperparameter the value 22 gives the highest accuracy.The plots and scores are as follows.

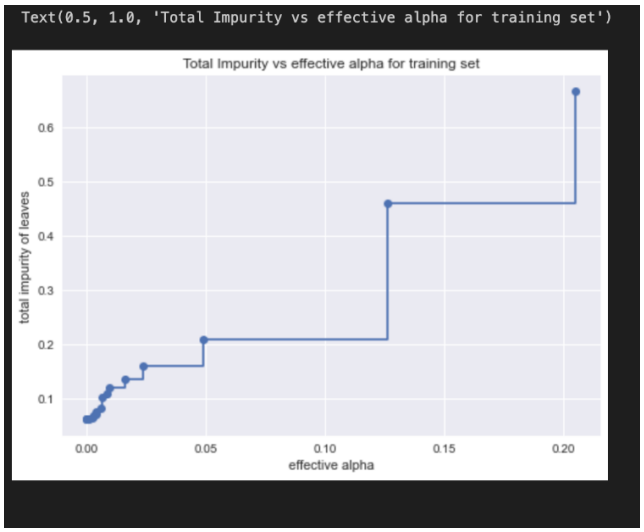


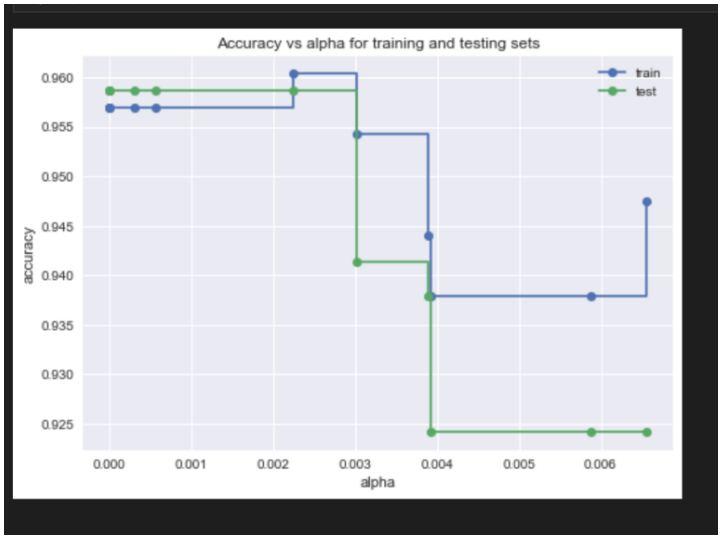
Part B Q1 - The question asks to remove one random node from the decision tree. If we remove a node, then its subtree also gets removed. In order to fix that, we needed to attach the children of the removed node to the parent of the removed node, which is not possible in sklearn's version of binary decision tree since it's a binary tree, a node cannot have 3 children. So in order to compromise, we removed the node and attached the child with more number of samples to the removed node's parents. This way, not a lot of the decision tree is lost, only the smaller child is pruned and the decision tree works.



Part B Q2 - In Cost Complexity pruning, the value of alpha with the highest test and training set accuracy was picked. Normally we pick the one with highest test set accuracy, but in our case, we picked the one with not only the highest test accuracy but also the highest training accuracy.

No improvement in accuracy was seen in regards to DT_A because DT_A is already manually tuned with good parameters.

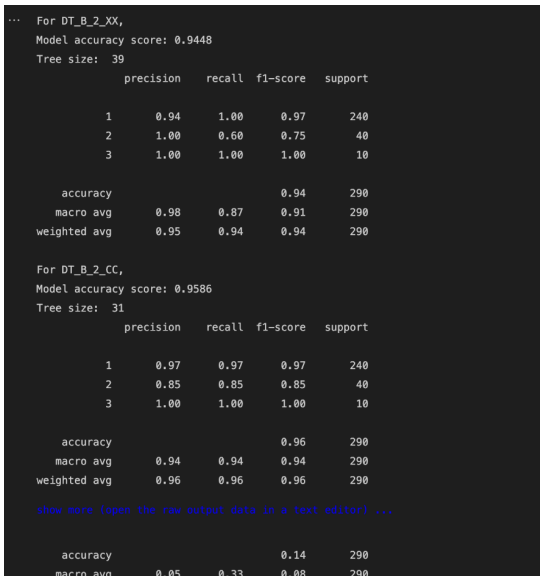




For the other pruning technique, we used pre pruning by using GridSearchCV on two parameters that are normally used to pre prune decision trees - min_sample_split and min_sample_leaf and we found that both performed worse than our original DT_A but min_sample_split with hypertuning performed better than min_sample_leaf so we chose that to be DT_B_2_XX. Our original DT_A performed better because it was already tuned selecting the best parameters manually.

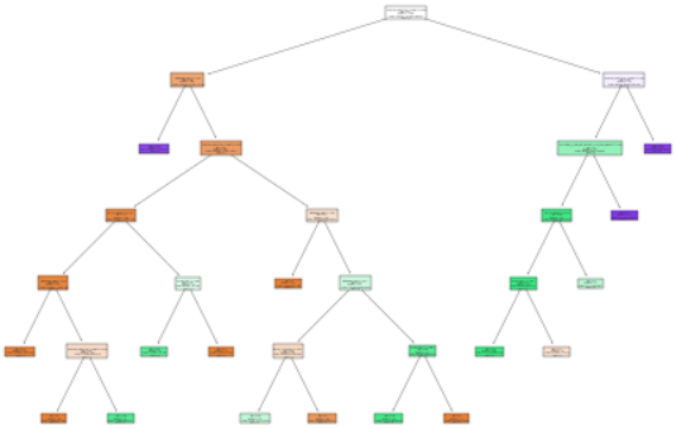
Part B Q3 - In Hybrid SLIQ pruning, we calculate the cost for every node and leaf and according to the costs, we decide whether a node should be pruned or not. According to the research paper shared, we first perform the Full pruning technique on the tree on the first pass, which will give us a smaller tree than the original. In this Full pruning, the leafs and internal nodes with 2 children are considered. If cost(parent) > cost(leaf) then the leaf is pruned. After going through the paper, it seems that no matter what, a leaf would always be pruned in Full SLIQ pruning technique. And performing more and more passes will just make the tree smaller and smaller since cost(parent) = cost(child1) + cost(child2) + encoding_cost as per the paper, the condition cost(parent) > cost(leaf) will always be true.

In Hybrid SLIQ pruning technique, we then take 2 trees (pruned and non pruned), according to the shared slides and calculate which tree has lesser cost and make our decision upon whether pruning is to be done or not. As per our understanding from the research paper and the slides, this is how the calculation was done.

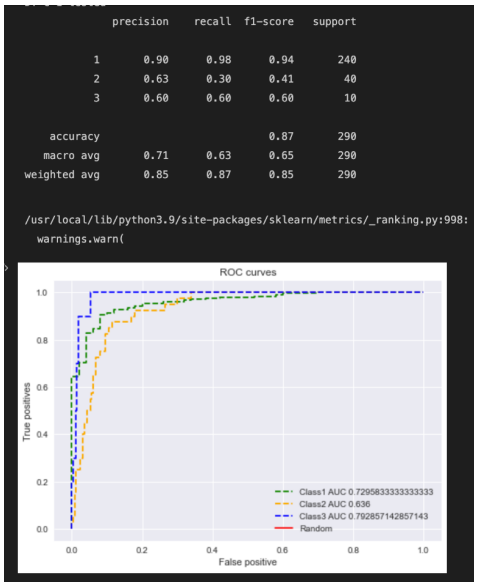


```
For DT_B_2_XX,
Model accuracy score: 0.9448
Tree size:  39
              precision    recall  f1-score   support

           1       0.94      1.00      0.97       240
           2       1.00      0.60      0.75        40
           3       1.00      1.00      1.00        10

    accuracy                           0.94       290
   macro avg       0.98      0.87      0.91       290
weighted avg       0.95      0.94      0.94       290

For DT_B_2_CC,
Model accuracy score: 0.9586
Tree size:  31
              precision    recall  f1-score   support

           1       0.97      0.97      0.97       240
           2       0.85      0.85      0.85        40
           3       1.00      1.00      1.00        10

    accuracy                           0.96       290
   macro avg       0.94      0.94      0.94       290
weighted avg       0.96      0.96      0.96       290

show more (open the raw output data in a text editor) ...


    accuracy                           0.14       290
   macro avg       0.05      0.33      0.08       290
```

```
Best perfroming model is either DT_A or DT_B_2_CC
```



Part C Q1 - In this part we had to find a way to use the DecisionTree created in Part A for a new dataset data_2. When the already created DecisionTree was tested on a portion of data_2 the accuracy was lower than normal. We then used an incremental decision tree (HoeffdingTree or VeryFastDecisionTree) to partially train the initial Decision Tree on a training set from data_2 and once trained we tested the same dataset which was used earlier and we observed that the results were significantly higher for the case after training. Through this we were able to understand how partial fitting happens on DecisionTrees and how we can create DataStreams that can be used to train DecisionTrees instead of passing all the data at once. This may be particularly useful in cases of very large datasets. The output was as follows.
Initial metrics after training and testing on data_1.



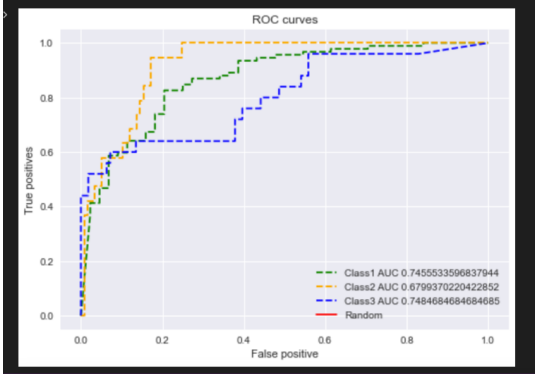
```
              precision    recall  f1-score   support

           1       0.90      0.98      0.94       240
           2       0.63      0.30      0.41        40
           3       0.60      0.60      0.60        10

    accuracy                           0.87       290
   macro avg       0.71      0.63      0.65       290
weighted avg       0.85      0.87      0.85       290

/usr/local/lib/python3.9/site-packages/sklearn/metrics/_ranking.py:998:
  warnings.warn(
```


After testing the decision tree with a portion of data_2 the results were as follows. We observe that the accuracy is lower than in the previous case.

DT-C-1 tested on data_2

	precision	recall	f1-score	support
1	0.81	0.95	0.87	92
2	0.88	0.37	0.52	19
3	0.67	0.56	0.61	25
accuracy			0.79	136
macro avg	0.78	0.62	0.67	136
weighted avg	0.79	0.79	0.78	136

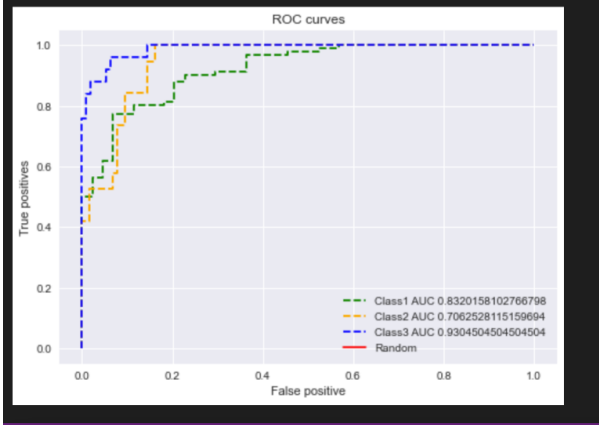
```
/usr/local/lib/python3.9/site-packages/sklearn/metrics/_ranking.py:998
  warnings.warn(
```



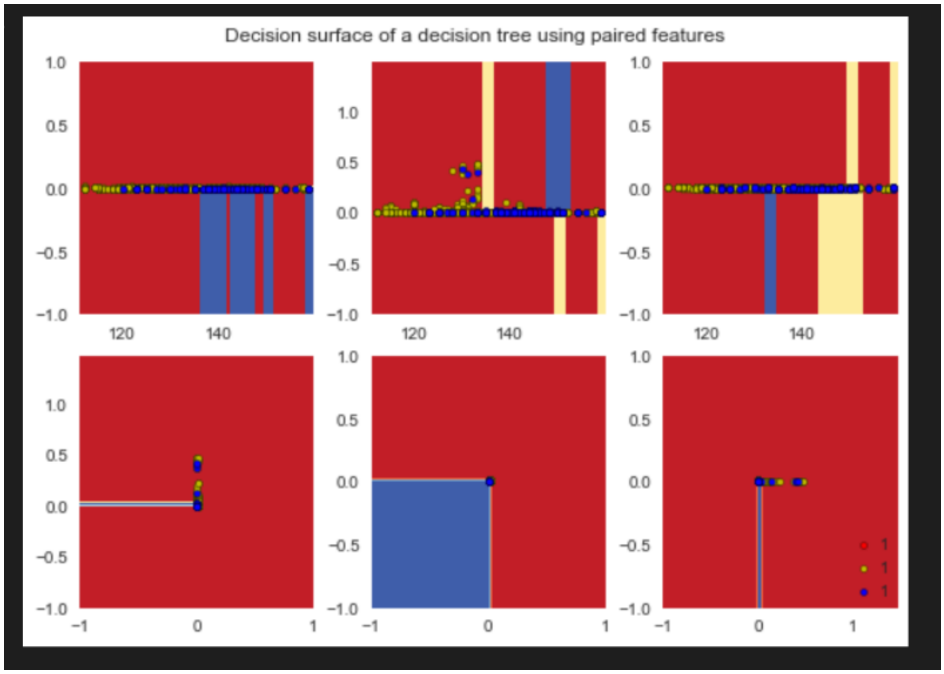
We then partially fit data_2's training set and then tested it out with the same dataset as above and the results were significantly higher as can be seen in the following screenshot.

	precision	recall	f1-score	support
1	0.89	0.89	0.89	92
2	0.89	0.42	0.57	19
3	0.69	0.96	0.80	25
accuracy			0.84	136
macro avg	0.82	0.76	0.75	136
weighted avg	0.85	0.84	0.83	136

```
/usr/local/lib/python3.9/site-packages/sklearn/metrics/_ranking.py:998:
  warnings.warn(
```



Part C Q2 - For this question we plotted the DecisionSurface boundaries for the decision tree. We used the example shown in the tut and incorporated the code with subsections of the dataset. Since surface boundaries are plotted pair wise, we plotted 6 pairs of features and the plots look as follows.



Learnings:

- Learned about the inner workings of a Decision Tree
- The different kinds of pruning
- How to tune a machine learning model by varying different parameters
- How pruning can be done using MDL and what the different types of MDL pruning are
- Learned about the SLIQ method of making a decision tree
- How to use GridSearchCV and plot for visualizing different alpha values
- Learned about incremental decision trees and how DecisionTrees can be constructed on very large datasets by using datastreams to partially train the DecisionTree for groups of samples.

References:

https://sci2s.ugr.es/keel/pdf/algorithm/congreso/SLIQ.pdf
https://www.yumpu.com/en/document/read/21978746/decision-trees-from-large-databases-sliq
https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
https://scikit-learn.org/stable/auto_examples/tree/plot_cost_complexity_pruning.html
https://scikit-multiflow.readthedocs.io/en/stable/api/generated/skmultiflow.trees.HoeffdingTreeClassifier.html