CSE506 - Data Mining

REPORT

Assignment 1

Submitted By
Vidhi Sharma, 2019286
Dolly Sidar, 2019304

QUESTION A: Training

Part 1:

Step 1 - Imported the needed libraries and loaded the dataset

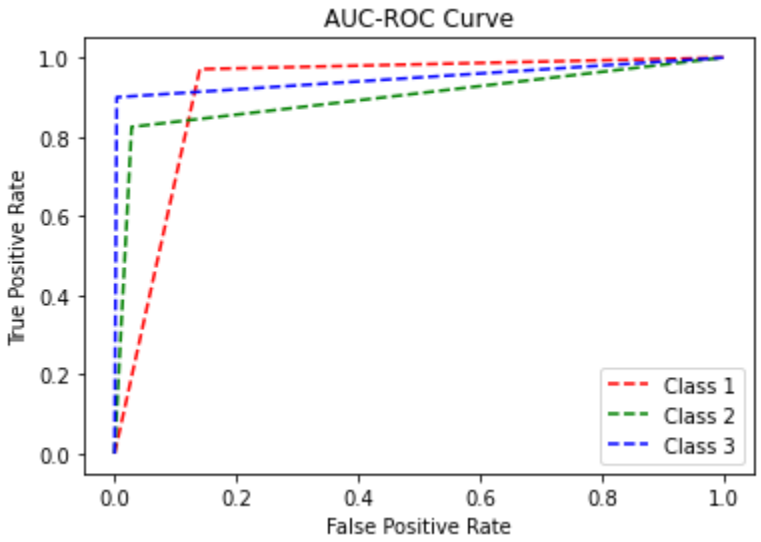
Step 2 - Preprocessing

a. Separate features and the target variable
b. Splitting the data into training and testing set of 8:2

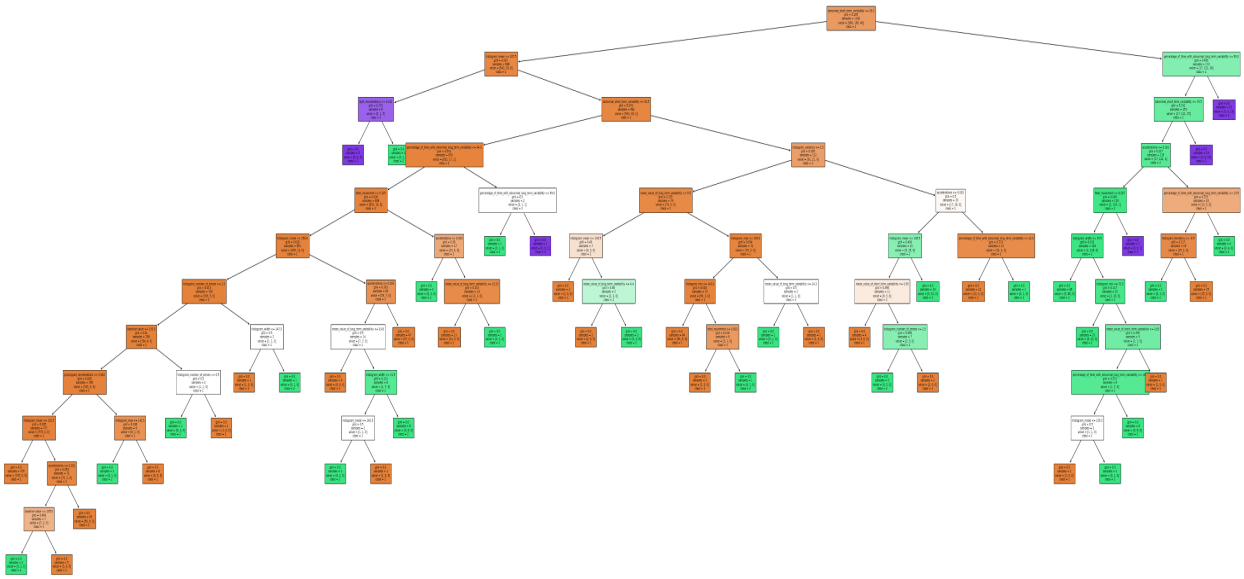
Step 3- Trained the data on the decision tree classifier
Step 4- Measured the performance of the model

	precision	recall	f1-score	support
1	0.97	0.97	0.97	240
2	0.82	0.82	0.82	40
3	0.90	0.90	0.90	10
accuracy			0.95	290
macro avg	0.90	0.90	0.90	290
weighted avg	0.95	0.95	0.95	290

Step 5- Plotted the AUC-ROC curve on the test set: AUC-ROC curve for the multi-class model includes y number of ROC curves one for each class against the rest.



Step 6- Visualization of the decision tree: (image name: DT_A_1.png).

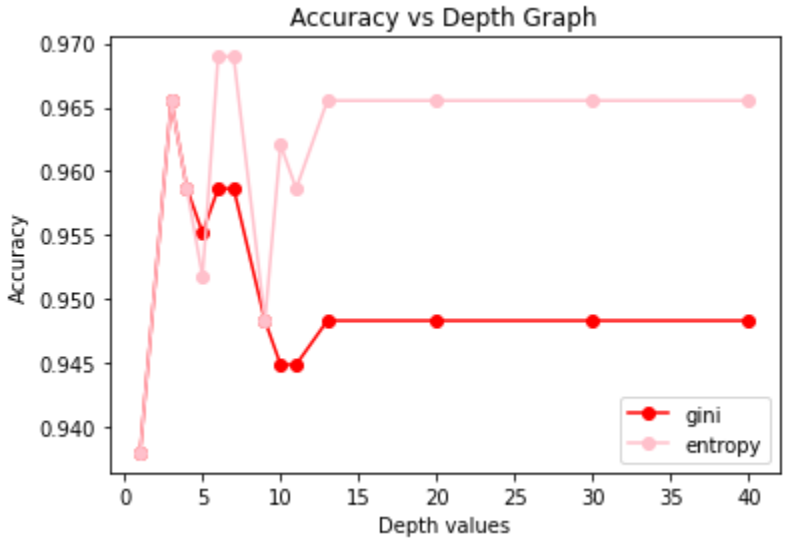


Part 2:

Step1- Trained the Decision Tree classifier on different depths
Assumption

depths = [1, 3, 4, 5, 6, 7, 9, 10, 11, 13, 20, 30, 40]

Step2- Trained for split criterion ='gini'
Step3- Trained for split criterion ='entropy'
Step4- Plotted the graph



Learning:

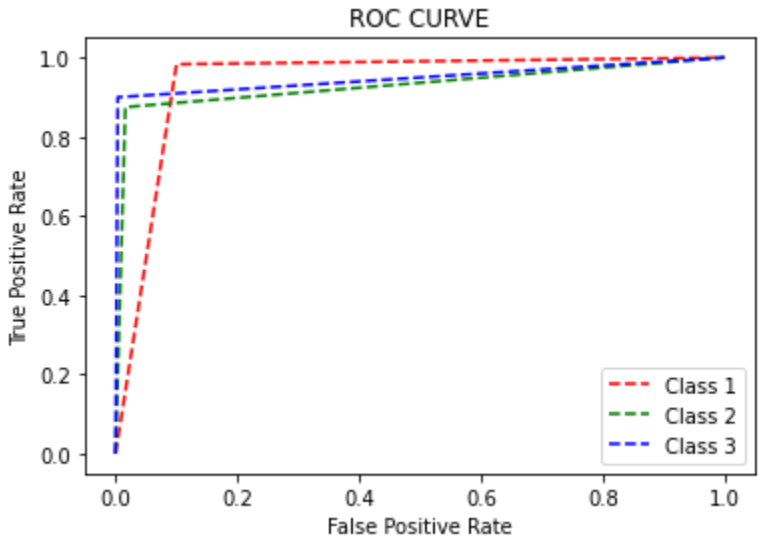
As seen in the graph, best accuracies are achieved at depth=4, 6, 7. Increasing the depth further reduces accuracy on testing data and overfits the data. So, it is better to not increase the max_depth any further. Also, a very small depth causes the underfitting of data.

Part 3:

Varying the following hyperparameters:-

Criterion: criterion = entropy, default: 'gini'

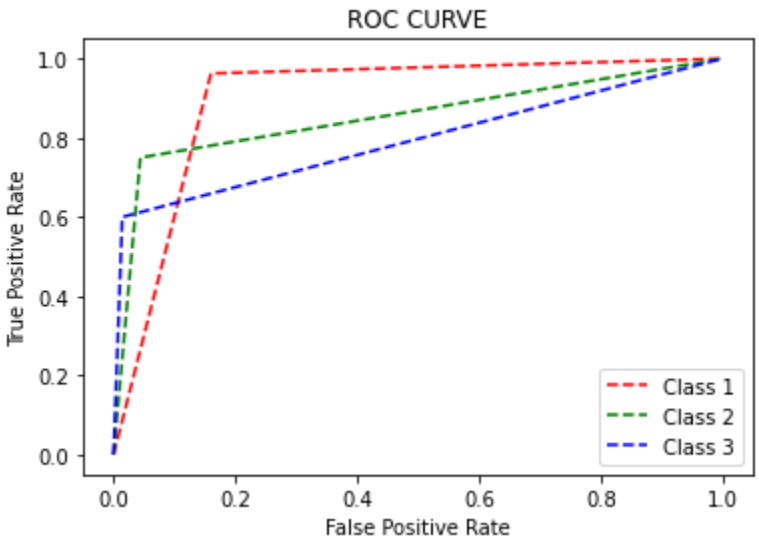
	precision	recall	f1-score	support
1	0.98	0.98	0.98	240
2	0.90	0.88	0.89	40
3	0.90	0.90	0.90	10
accuracy			0.97	290
macro avg	0.93	0.92	0.92	290
weighted avg	0.97	0.97	0.97	290



Reasoning: The Criterion of Splitting nodes can be based on gini impurity or entropy impurity(information gain). The difference in the accuracy of the entropy model is about 2% more than the accuracy model. The information gain criterion for splitting nodes is just a little more accurate but the calculation of information gain is more complex than the calculation of gini impurity. Here, we will go with Criterion = ”entropy” (2% more accurate for this dataset).

Splitter: splitter = random, default: 'best'

	precision	recall	f1-score	support
1	0.97	0.96	0.96	240
2	0.73	0.75	0.74	40
3	0.60	0.60	0.60	10
accuracy			0.92	290
macro avg	0.77	0.77	0.77	290
weighted avg	0.92	0.92	0.92	290

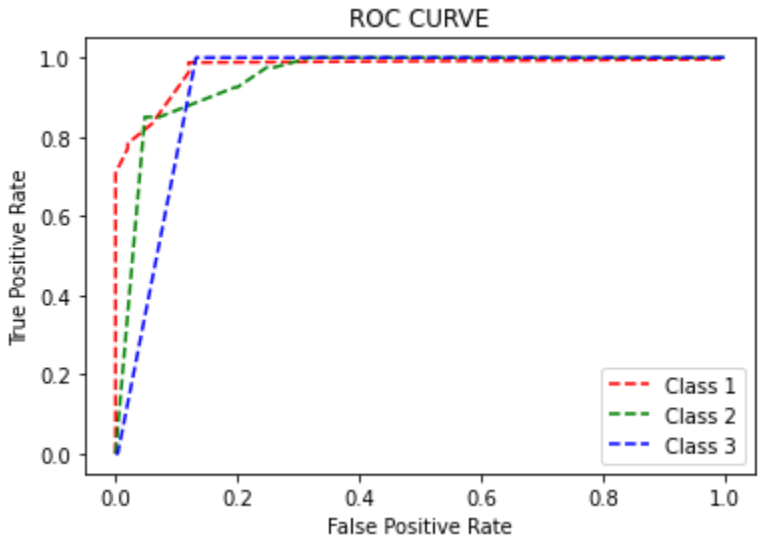


Reasoning: Splitter = 'random' chooses random best split (considers only some random features for impurity calculation). It is used when we are sure that all features are relevant. As compared to the base model the accuracy has decreased by 3%. This is because the number of features in this dataset is less. So, using a random split may lead to choosing some features that don't have much information. Therefore the tree is less accurate. So, we will choose the default values of Splitter = 'best' which gives us the best split considering all features.

Min samples spilt: min_samples_split = 200, default: 2

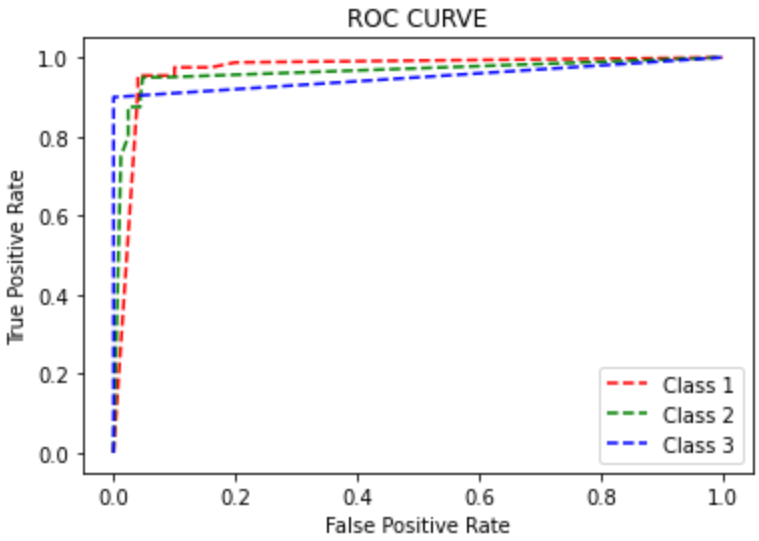
	precision	recall	f1-score	support
1	0.98	0.99	0.98	240
2	0.74	0.85	0.79	40
3	0.00	0.00	0.00	10

accuracy			0.93	290
macro avg	0.57	0.61	0.59	290
weighted avg	0.91	0.93	0.92	290



min_samples_split = 13, default: 2

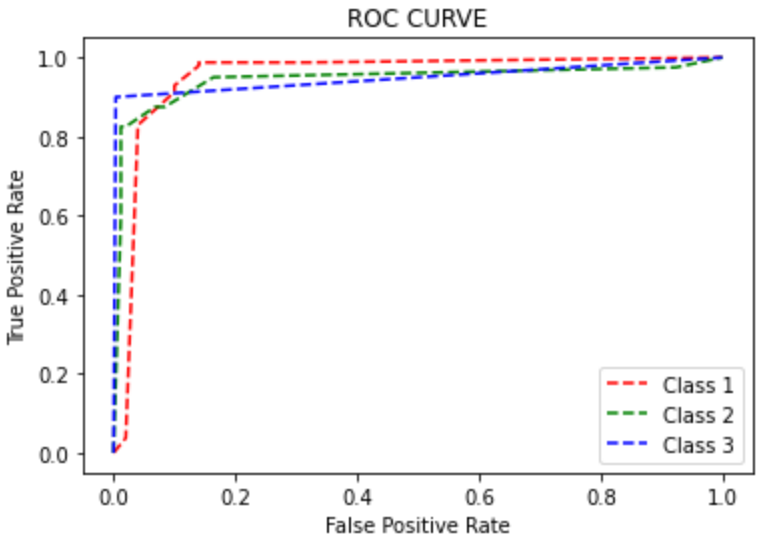
	precision	recall	f1-score	support
1	0.98	0.97	0.98	240
2	0.85	0.88	0.86	40
3	0.90	0.90	0.90	10
accuracy			0.96	290
macro avg	0.91	0.92	0.91	290
weighted avg	0.96	0.96	0.96	290



Reasoning: As we can see min_samples_split = 200 has less accuracy than the base model while min_samples_split = 13 has better accuracy than the base model. This is because for a large value (>=100) of min_samples_split underfitting of data takes place. Also, Min_samples_split if its value is a lot less than 13 then overfitting occurs.
So, min_samples_split=13 is kind of an optimal value.

Max depth: max_depth = 6

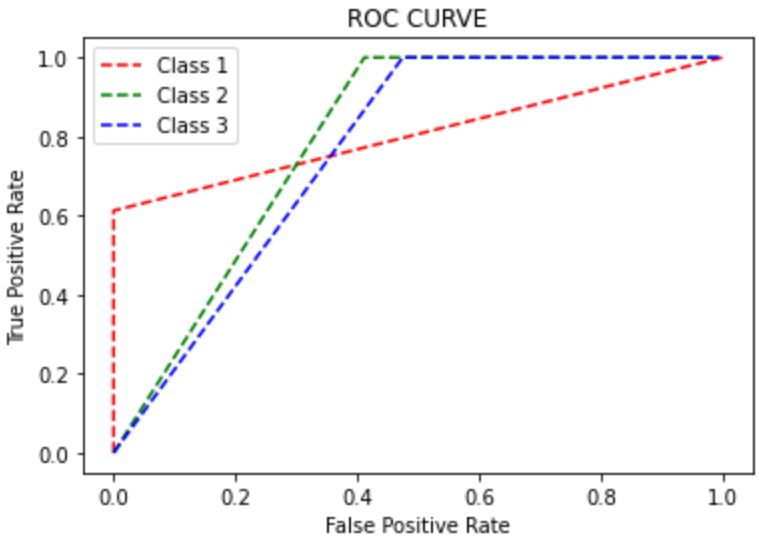
	precision	recall	f1-score	support
1	0.97	0.98	0.98	240
2	0.89	0.82	0.86	40
3	0.90	0.90	0.90	10
accuracy			0.96	290
macro avg	0.92	0.90	0.91	290
weighted avg	0.96	0.96	0.96	290



Reasoning: max_depth=6 increases the accuracy as compared to the base model. This is because pruning the tree before depth = 6 helps the model to avoid overfitting. While an optimal length such as depth= 6 ensures that the model works well on unseen data.

Min samples leaf: min_samples_leaf = 500, deafult:1

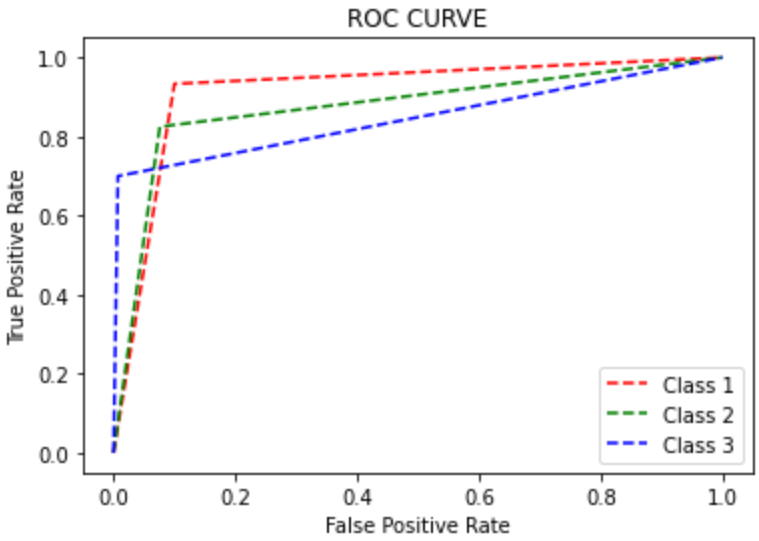
	precision	recall	f1-score	support
1	0.83	1.00	0.91	240
2	0.00	0.00	0.00	40
3	0.00	0.00	0.00	10
accuracy			0.83	290
macro avg	0.28	0.33	0.30	290
weighted avg	0.68	0.83	0.75	290



Reasoning:
Min_samples leaf is a minimum count of samples that are required to be at a leaf node. In this case Min_samples_leaf = 500 underfits the data. While very less value of min_sample_leaf can lead to overfitting.

Max features (sqrt/log): max_features='sqrt' ,deafult:***None***

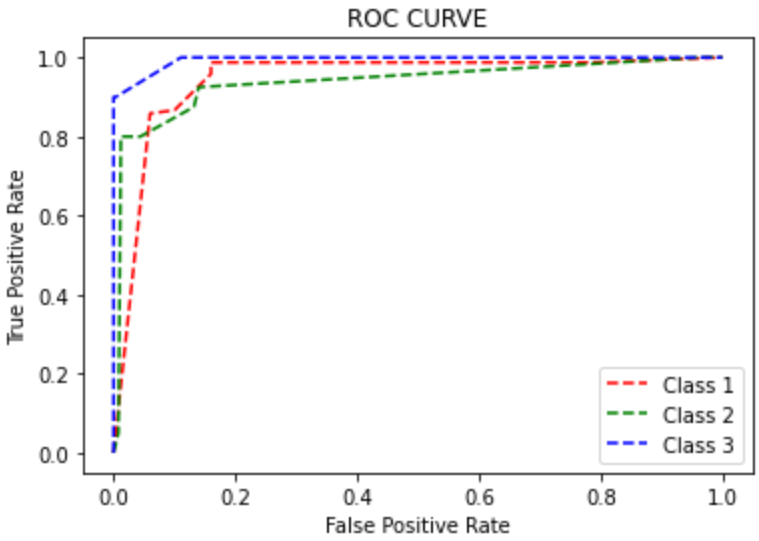
	precision	recall	f1-score	support
1	0.98	0.93	0.96	240
2	0.63	0.82	0.72	40
3	0.78	0.70	0.74	10
accuracy			0.91	290
macro avg	0.80	0.82	0.80	290
weighted avg	0.92	0.91	0.91	290



Reasoning:
Max-feature = ‘sqrt’ ensures the number of features to be considered for the best split are sqrt(no. Of features). As compared to the base model the accuracy has decreased by 4%. This is because the number of features in this dataset is less. So, using sqrt(no. Of features) may lead to choosing some features that don’t have much information. Therefore the tree is less accurate. So, we will choose the default values of Max-feature =’ None’. By using sqrt we can limit the overfitting in cases where no of features is large.

Max-leaf nodes: max_leaf_nodes=12**,** default = None

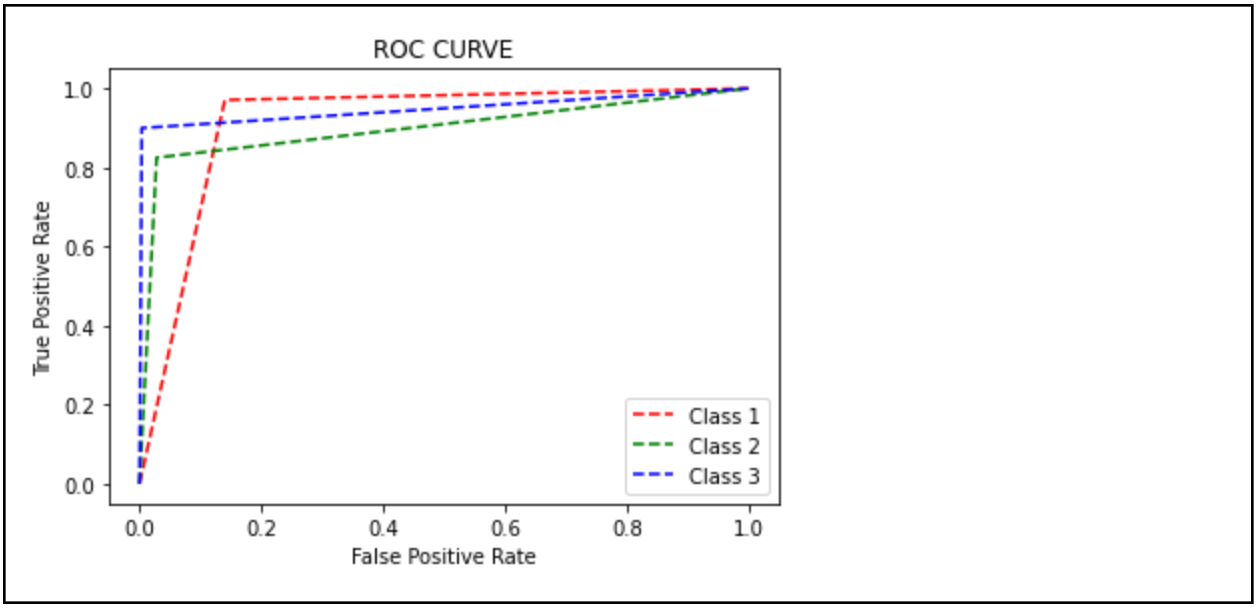
	precision	recall	f1-score	support
1	0.97	0.99	0.98	240
2	0.91	0.80	0.85	40
3	0.90	0.90	0.90	10
accuracy			0.96	290
macro avg	0.93	0.90	0.91	290
weighted avg	0.96	0.96	0.96	290



Reasoning: Max_leaf_nodes denotes the maximum number of leaves that can be there in the tree. max_leaf_nodes=12 here is one of the optimal value for max_leaf_nodes as it avoids overfitting as well as underfitting of data. Max_leaf_nodes >100 here can cause overfitting while a very less value of max_leaf_nodes can cause underfitting.

Weight: class_weight='balanced' default:None

	precision	recall	f1-score	support
1	0.97	0.97	0.97	240
2	0.82	0.82	0.82	40
3	0.90	0.90	0.90	10
accuracy			0.95	290
macro avg	0.90	0.90	0.90	290
weighted avg	0.95	0.95	0.95	290



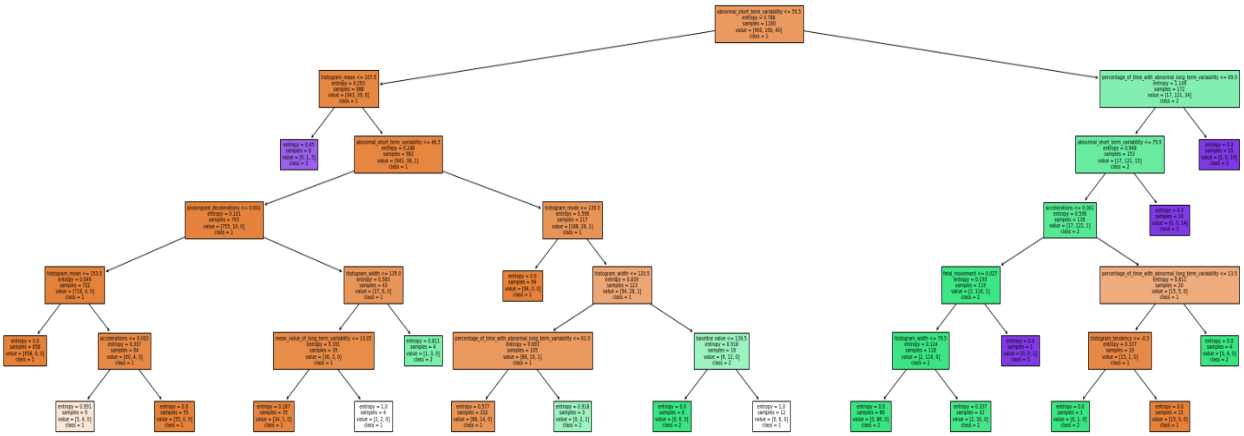
Reasoning:
class_weight is used to add weight for each output class. At the time of the splitting at a node, the resulting child nodes are weighted by the class_weight giving the child samples weights based on the class proportion. Using 'balanced' has not changed the accuracy much in this case (though classes are imbalanced), because we are calculating accuracy using classification_report which takes this into consideration.

Hyperparameters tuning DT-A

class_weight=None, criterion='entropy', min_samples_split=13, max_depth=6, random_state=123

	precision	recall	f1-score	support
1	0.97	1.00	0.98	240
2	0.97	0.82	0.89	40
3	0.90	0.90	0.90	10
accuracy			0.97	290
macro avg	0.95	0.91	0.93	290
weighted avg	0.97	0.97	0.97	290

Visualization of the decision tree: (image name: DT_A.png).



Learning:

Best Obtained Hyperparameters: criterion='entropy', min_samples_split=13, max_depth=6.

- criterion='entropy' gives 2% more accuracy as seen above. So, I use information gain as the parameter for splitting nodes.
- max_depth is taken as 6 because as we can see in the graph in (A) part 2 accuracy vs depth graph for criterion=entropy that the maximum accuracy lies at depth=6. So, there is no need to increase the depth further. Also, larger depth can cause overfitting of data.
- Min_samples_split is taken as 13 because if its value is very less then overfitting occurs (if no. of samples decrease, no of nodes increase). Also, for a large value underfitting of data can take place.

QUESTION B: Post pruning

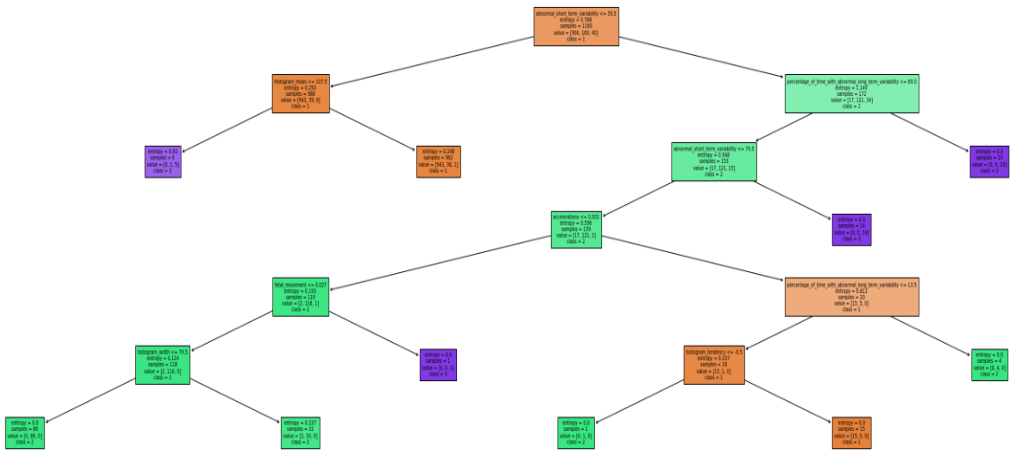
Part 1:

Assumption :

- We have removed a random node having index=3. The code is easily flexible and other random values of the index can be used in place of 3.
- We have assumed that if we have to remove a node with a random index. We will remove it and replace it with the subtree (right or left) which has a greater threshold value.

	precision	recall	f1-score	support
1	0.97	1.00	0.98	240
2	0.97	0.82	0.89	40
3	0.90	0.90	0.90	10

accuracy			0.97	290
macro avg	0.95	0.91	0.93	290
weighted avg	0.97	0.97	0.97	290

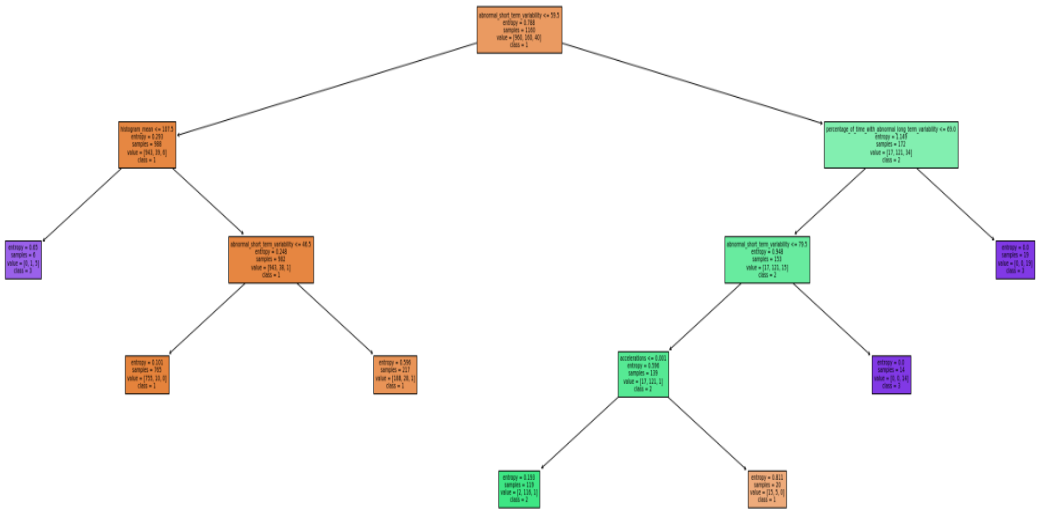


Part 2:

Step 1- Applied the Cost Complexity pruning technique
Step 2- Reported the value of alpha for Cost Complexity Pruning
Step 3- Reported precision, recall, and accuracy
Step 4- Visualization of the DT-B-2-CC

```
Value of Alpha after CCP:  {'ccp_alpha': 0.022582742436230 68}
```

	precision	recall	f1-score	support
1	0.96	1.00	0.98	240
2	0.97	0.75	0.85	40
3	0.90	0.90	0.90	10
accuracy			0.96	290
macro avg	0.94	0.88	0.91	290
weighted avg	0.96	0.96	0.96	290

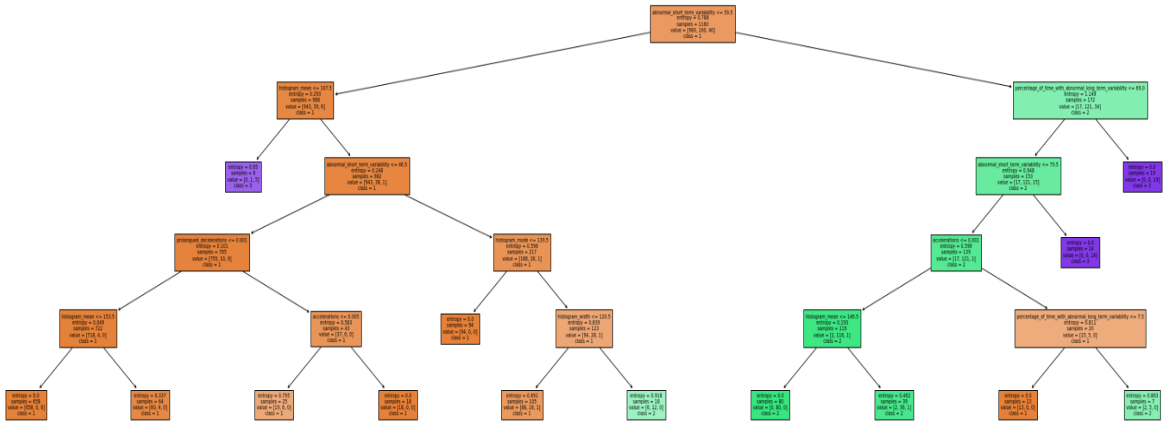


Learning: As we can see, the pruned tree has almost the same accuracy, precision, recall, and f1 score as DT_A. Also, the complexity of the pruned tree (after cost complexity pruning) has decreased a lot as compared to DT_A. This model avoids overfitting of the data.

Step 1- Applied pre-pruning technique using GridSearchCV over parameters max_depth, min_samples_leaf, min_samples_split.
Step 2- parameters: max_depth, min_samples_split, min_samples_leaf
Step 3- Reported parameters of second pruning technique
Step 4- Reported precision, recall, and accuracy
Step 5- Visualization of the DT-B-2-XX

```
Value of Parameters after pre-pruning:  {'max_depth': 5,
'min_samples_leaf': 5, 'min_samples_split': 5}
```

	precision	recall	f1-score	support
1	0.97	0.98	0.98	240
2	0.87	0.82	0.85	40
3	0.90	0.90	0.90	10
accuracy			0.96	290
macro avg	0.91	0.90	0.91	290
weighted avg	0.95	0.96	0.95	290



Learning: As we can see, the pruned tree (obtained by tuning parameters using GridSearchCV) has almost the same accuracy, precision, recall, and f1 score as DT_A. Also, the complexity of the pruned tree (after cost complexity pruning) has decreased as compared to DT_A. This model avoids overfitting of the data.

Part 3:

Learning:

QUESTION C: Experiments

Part 1:

Assumption :

To make a decision tree classifier in this part we have used - **ExtremelyFastDecisionTreeClassifier** from skmultiflow.trees model in scikit-multi-flow API.

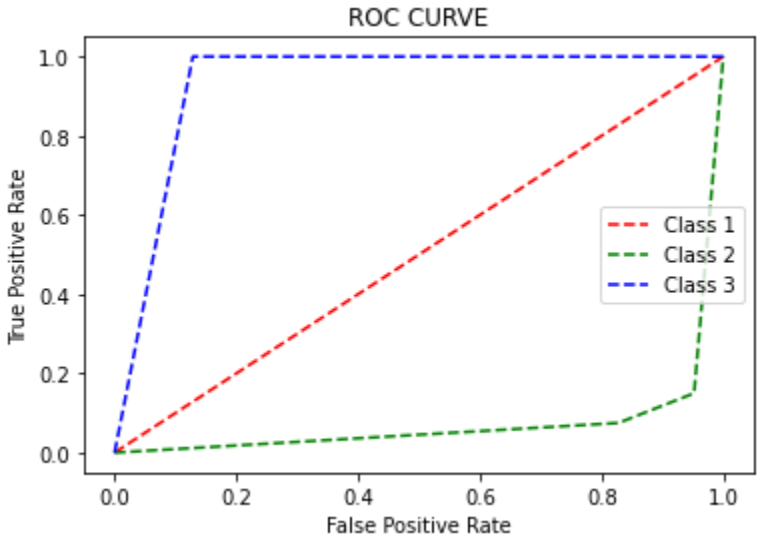
We have used it in this question as it is an **incremental learning decision** tree. Even after the tree has been trained using a dataset, this decision tree can be **updated** by training on new data points.

Step 1- trained DT_C_1 using data_1

Step 2- augmented DT-C-1 with the new data data_2 and formulated DT-C-1-X

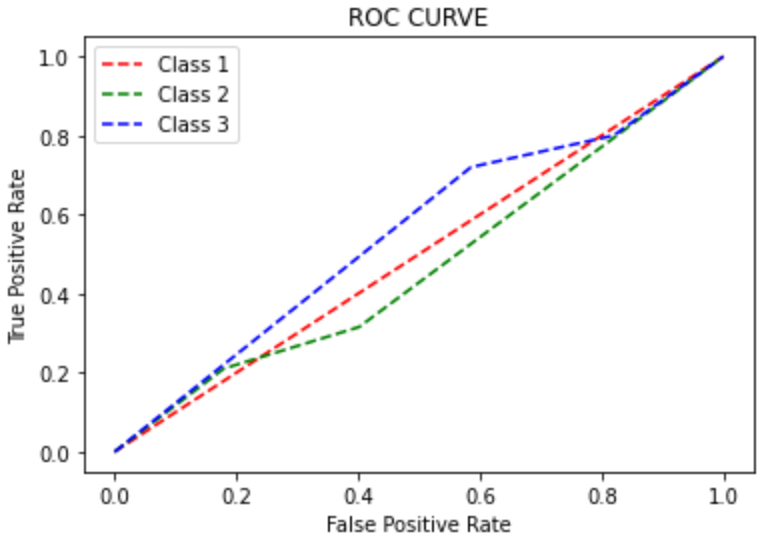
```
The classifier DT-C-1 :

 Performance of DT-C-1 on data_1 test set:
```

	precision	recall	f1-score	support
1	0.98	0.99	0.98	240
2	0.74	0.85	0.79	40
3	0.00	0.00	0.00	10
accuracy			0.94	290
macro avg	0.57	0.61	0.59	290
weighted avg	0.91	0.94	0.92	290



Performance of DT-C-1 on data_2 test set:

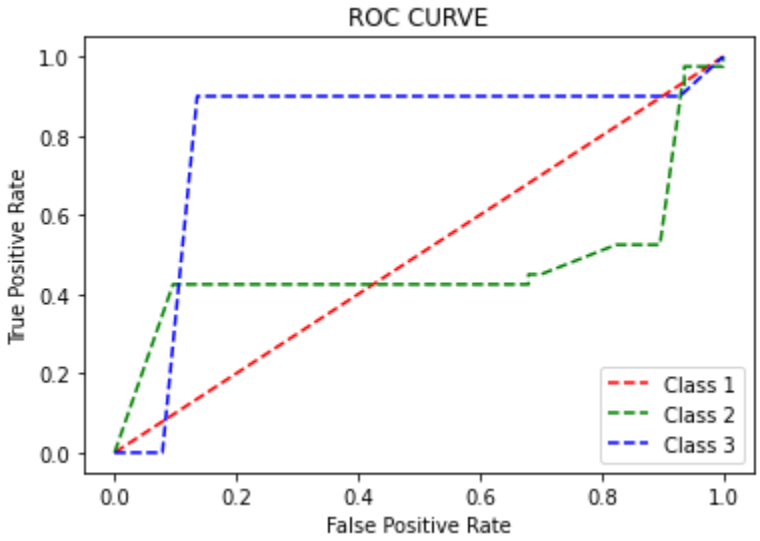
	precision	recall	f1-score	support
1	0.75	0.43	0.55	92
2	0.16	0.68	0.25	19
3	0.00	0.00	0.00	25
accuracy			0.39	136
macro avg	0.30	0.37	0.27	136
weighted avg	0.53	0.39	0.41	136



The classifier DT-C-1-X :

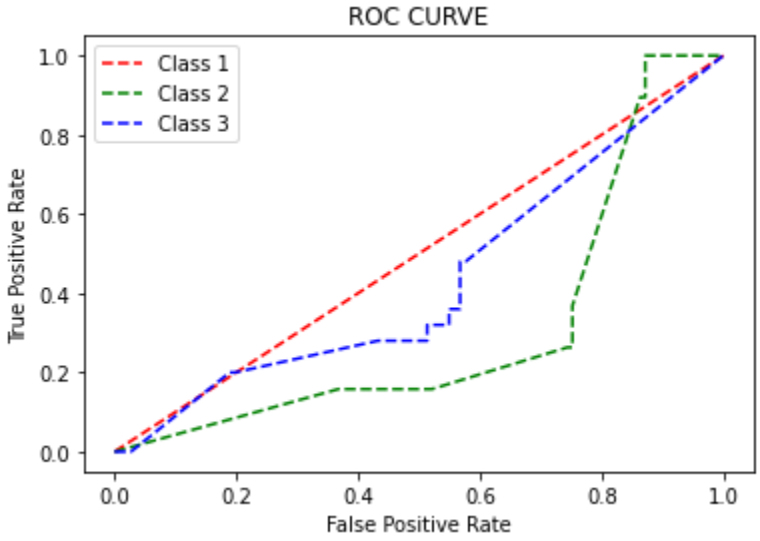
Performance of DT-C-1-X on data_1 test set:

	precision	recall	f1-score	support
1	0.91	0.90	0.90	240
2	0.38	0.45	0.41	40
3	0.00	0.00	0.00	10
micro avg	0.81	0.81	0.81	290
macro avg	0.43	0.45	0.44	290
weighted avg	0.80	0.81	0.80	290



Performance of DT-C-1-X on data_2 test set:

	precision	recall	f1-score	support
1	0.89	0.91	0.90	92
2	0.50	0.68	0.58	19
3	1.00	0.64	0.78	25
accuracy			0.83	136
macro avg	0.80	0.75	0.75	136
weighted avg	0.86	0.83	0.84	136



Learning:

As we can see, DT_C_1 which was trained only using data_1 training points works good (with an accuracy of 94%) on the data_1 testing points. But, its performance on data_2 training points is very bad (only 39% accurate). This shows that DT_C_1 needs to be trained on data_2 too in order to perform well in both cases.

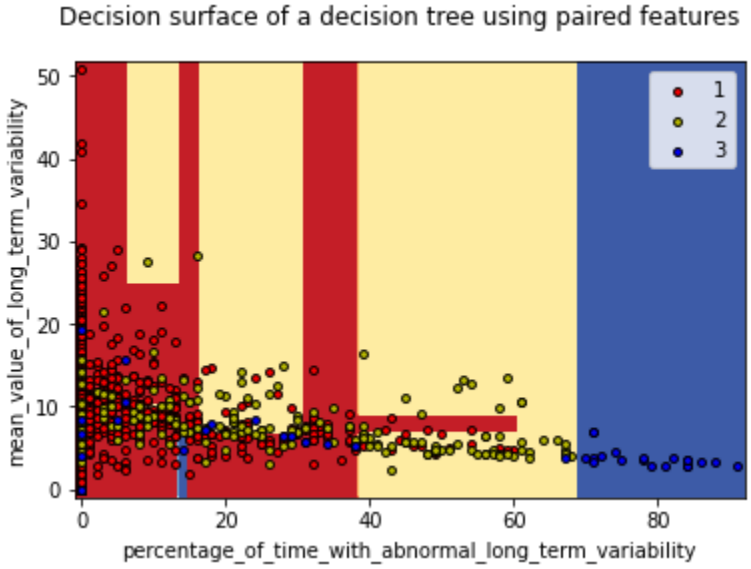
Now, DT_C_1 is an incremental decision tree. So, it is updated by training on data_2 training points as well. After its incremental learning, the accuracy for both data_1 and data_2 testing points is good and close (81%- 83%).

Part 2:

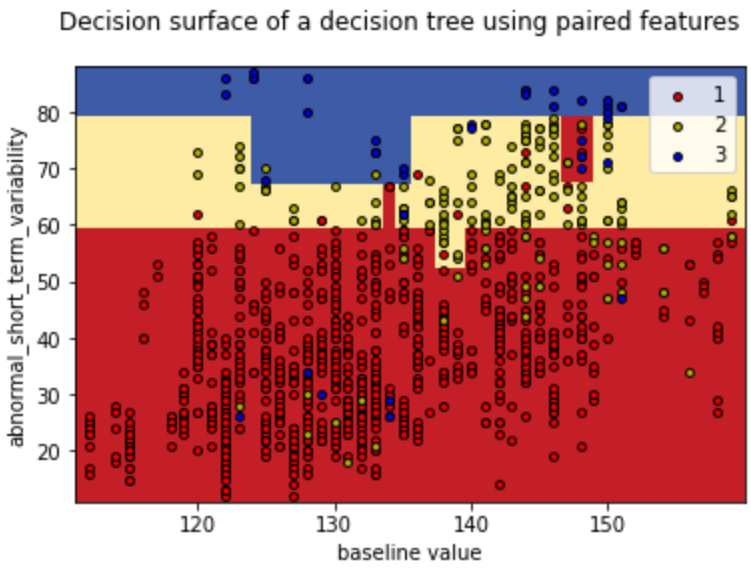
Assumption: We have Visualized the decision surface boundaries of the DT DT_A instead of DT of that in Part B Q3.We have used DT_A to find the distance of a sample to the nearest decision boundary of the decision tree as **mentioned in the comments**.

Step 1- extracted X and Y from a train set, X containing samples only with 2 features
Step 2- trained decision tree
Step 3- called sample_distance function to find the distance of the given sample to the nearest decision boundary
Step 4- found min and max values for each feature
Step 5- found predicted values
Step 6- plotted a graph of the decision surface of a decision tree using paired features

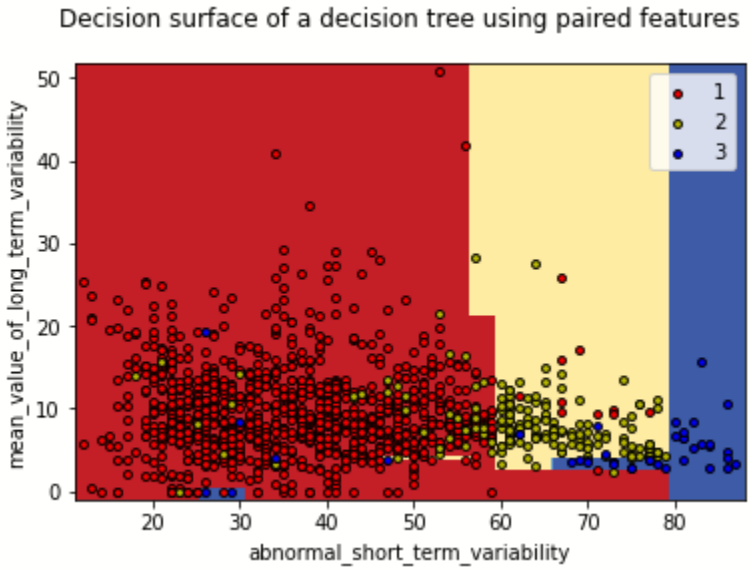
Feature : 9,10
Sample : 1,11.4
Distance of the sample to the nearest decision boundary: 4.591568537088732



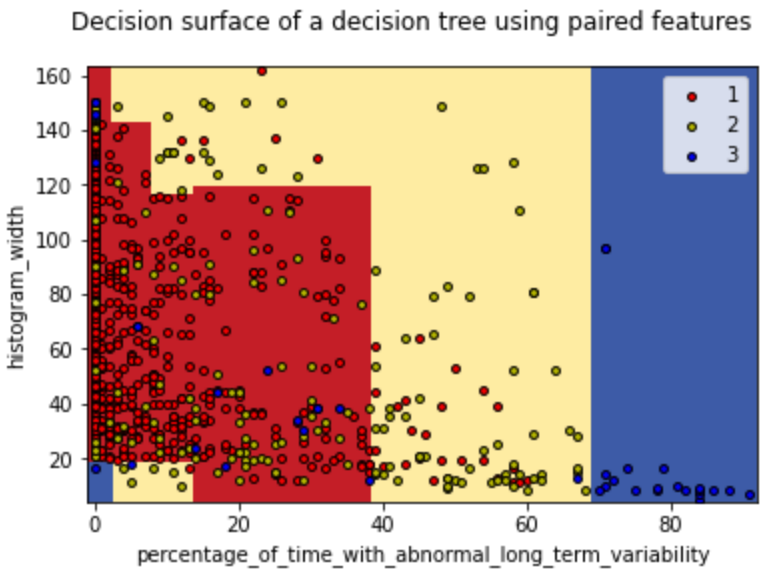
Feature : 0,7
Sample : 130,24
Distance of the sample to the nearest decision boundary: 39.45883931389772



Feature : 7,10
Sample : 19, 8.7
Distance of the sample to the nearest decision boundary: 10.76150546853627



Feature : 9,11
Sample: 18,44
Distance of the sample to the nearest decision boundary: 26.476404589747453



Learning:

As we can see from the above graph the decision boundaries made by a decision tree are parallel to the axis, the reason is, it splits the data set based on a feature value that remains constant throughout for one decision boundary. We can see in some of the graphs, the data that can be split by a single vertical or horizontal line is split by multiple decision boundaries, which implies DT are sensitive to even small data variations and can result in different tree structures.

References:

https://medium.datadriveninvestor.com/decision-tree-adventures-2-explanation-of-decision-tree-classifier-parameters-84776f39a28
https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/
https://towardsdatascience.com/build-better-decision-trees-with-pruning-8f467e73b107
https://stackoverflow.com/questions/60960692/find-distance-to-decision-boundary-in-decision-trees
https://scikit-multiflow.readthedocs.io/en/stable/api/generated/skmultiflow.trees.ExtremelyFastDecisionTreeClassifier.html
https://scikit-learn.org/stable/auto_examples/tree/plot_iris_dtc.html
https://scikit-learn.org/stable/auto_examples/tree/plot_unveil_tree_structure.html#sphx-glr-auto-examples-tree-plot-unveil-tree-structure-py
https://towardsdatascience.com/decision-tree-overview-with-no-maths-66b256281e2b