ASSIGNMENT - 1

Importing libraries
Preparing the dataset

- Importing the data_1 and data_2
- Checking dimensions of the dataset
- Preview dataset
- Check missing values in variables
- Feature vector and target variable declaration (X and y)
- Check target variable distribution - unbalanced distribution

(A) Q1

Step1: Splitting X and y into training and test set with a split ratio of 80-20.
For maintaining equal distribution use stratify=y.
Step2: Instantiate the decision tree classifier model tr_model with the random state as 0 that will control the shuffling applied before each split.
All the default parameters are taken:

criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0
Step3: Fit the model and predict test set results
Step4: Precision, recall, accuracy, AUC-ROC curve are reported.
Step5: Decision tree DT visualization
Save the Decision tree DT as DT_A_1

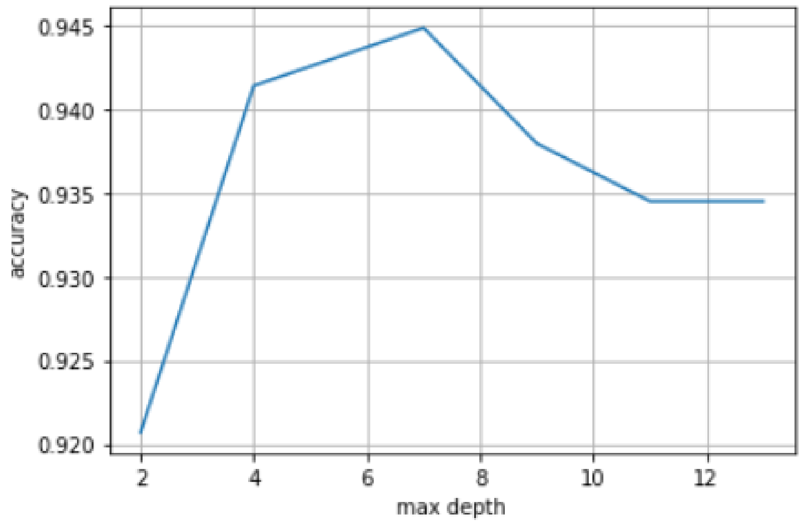
(A) Q2

Step1: Training the decision tree classifier on 6 different depths
Step2: Calculate the accuracy for each depth (ta

- max_depth=2 accuracy= 0.9206896551724137
- max_depth=4 accuracy= 0.9413793103448276
- max_depth=7 accuracy= 0.9448275862068966
- max_depth=9 accuracy= 0.9379310344827586
- max_depth=11 accuracy= 0.9344827586206896
- max_depth=13 accuracy= 0.9344827586206896

Step3: Plot the accuracy versus depth graph taking accuracy on the y-axis and max depth in X-axis
From the obtained graph we can observe that we are getting the maximum accuracy for max depth as 7.

```
Text(0.5, 0, 'max depth')
```



(A) Q3

Experiment 1: If we **vary Criterion as entropy** taking other hyperparameters as default, we observe a positive performance score in the precision of classes, recall, and AUC_ROC score. Though the Gini criterion provides the best features and is faster to compute but here since the class1 labels are more and we aim for the more balanced tree so selecting entropy is a better choice.

```
precision    recall  f1-score   support

           1       0.96      0.97      0.97       240
           2       0.79      0.78      0.78        40
           3       0.89      0.80      0.84        10

    accuracy                           0.94       290
   macro avg       0.88      0.85      0.86       290
weighted avg       0.94      0.94      0.94       290
```

Experiment 2: varying **splitter as random** and taking other hyperparameters as default. It is used for choosing the technique to split each node. we observed that it reduces all the accuracy, precision, and recall among others such as random. So we picked **splitter to be best**, which was the default one.

```
precision    recall  f1-score   support

           1       0.97      0.96      0.97       240
           2       0.74      0.85      0.79        40
           3       0.71      0.50      0.59        10

    accuracy                           0.93       290
   macro avg       0.81      0.77      0.78       290
weighted avg       0.93      0.93      0.93       290
```

Experiment 3:varying **min sample split as 2** and taking other hyperparameters as default, min sample split is for deciding a minimum number of samples required to split the node. accuracy, precision, recall, and auc_roc score all increased. So-min sample split as 2 is taken as the best choice.

```
precision    recall  f1-score   support

           1       0.96      0.97      0.97       240
           2       0.82      0.78      0.79        40
           3       1.00      0.90      0.95        10

    accuracy                           0.94       290
   macro avg       0.93      0.88      0.90       290
weighted avg       0.94      0.94      0.94       290
```

Experiment 4:varying **max_depth as 7** and taking other hyperparameters as default. Max depth indicates the maximum depth that the tree is allowed to grow after which we take the majority class label and make it as a leaf(if impure). We observed that among various depths, 7 is giving good results. So it is being chosen.

```
precision    recall  f1-score   support

           1       0.97      0.97      0.97       240
           2       0.80      0.80      0.80        40
           3       0.90      0.90      0.90        10

    accuracy                           0.94       290
   macro avg       0.89      0.89      0.89       290
weighted avg       0.94      0.94      0.94       290
```

Experiment 5: varying **min_samples_leaf as 2** and taking other hyperparameters as default. The min_samples_leaf indicates the minimum number of samples to be needed at the leaf node. By experimenting, we found that min_samples_leaf as 2 gives the best results. So it is being chosen.

```
precision    recall  f1-score   support

           1       0.96      0.97      0.96       240
           2       0.78      0.78      0.78        40
           3       1.00      0.70      0.82        10

    accuracy                           0.93       290
   macro avg       0.91      0.82      0.85       290
weighted avg       0.93      0.93      0.93       290
```

Experiment 6: varying **max features as sqrt** and taking other hyperparameters as default. It indicates the number of features needed to look at while determining the best split. We have taken the **default that is None** as all the accuracy, precision, recall and auc_roc score decreased.

```
precision    recall  f1-score   support

           1       0.96      0.97      0.97       240
           2       0.83      0.75      0.79        40
           3       0.60      0.60      0.60        10

    accuracy                           0.93       290
   macro avg       0.80      0.78      0.79       290
weighted avg       0.93      0.93      0.93       290
```

Experiment 7: varying **class weight as balanced** and taking other hyperparameters as default. It is the weight that is associated with the class. **Default** performs better in terms of all scores in comparison to balanced.

```
precision    recall  f1-score   support

           1       0.97      0.98      0.97       240
           2       0.84      0.80      0.82        40
           3       0.89      0.80      0.84        10

    accuracy                           0.95       290
   macro avg       0.90      0.86      0.88       290
weighted avg       0.95      0.95      0.95       290
```

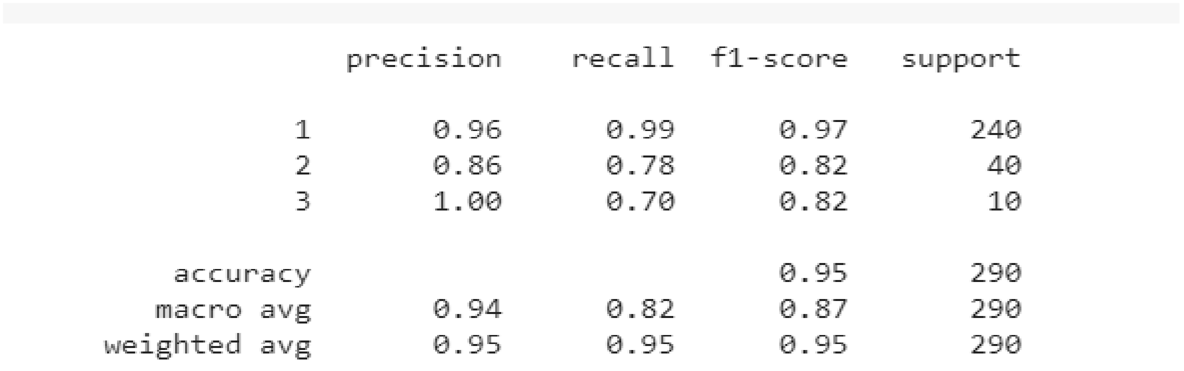
Experiment 8: varying **max-leaf nodes to 20** and taking other hyperparameters as default. Max leaf nodes put the limit on the maximum number of leaves that can be in the decision tree. By experimenting with different values, I found out that 20 gives good results.

```
precision    recall  f1-score   support

           1       0.97      0.97      0.97       240
           2       0.80      0.82      0.81        40
           3       1.00      0.80      0.89        10

    accuracy                           0.95       290
   macro avg       0.93      0.87      0.89       290
weighted avg       0.95      0.95      0.95       290
```

After all the experiments the **best parameters** derived are:
criterion='entropy', splitter='best',max_depth=7, min_samples_split=3, min_samples_leaf=5, max_leaf_nodes=20, random_state=0



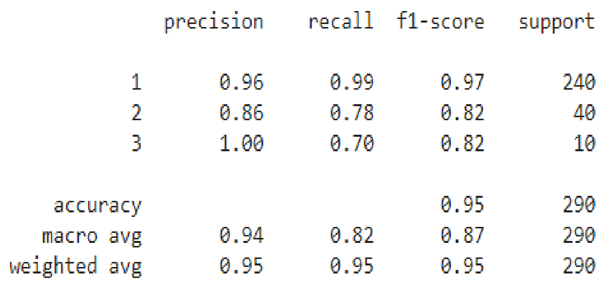
```
              precision    recall  f1-score   support

           1       0.96      0.99      0.97       240
           2       0.86      0.78      0.82        40
           3       1.00      0.70      0.82        10

    accuracy                           0.95       290
   macro avg       0.94      0.82      0.87       290
weighted avg       0.95      0.95      0.95       290
```

(B) Q1

Step1: Take DT_B_1 from (A) for the computation
Step2: After fitting the model, check the tree structure using id of left and right children
Total node count=39
Step3: Select a random node that needs to be deleted, in our experiment we took node 10. Now, making both the children of node 10 as -1.
We saw the optimization of some performance measures like roc score but the overall accuracy is not changed.



```
              precision    recall  f1-score   support

           1       0.96      0.99      0.97       240
           2       0.86      0.78      0.82        40
           3       1.00      0.70      0.82        10

    accuracy                           0.95       290
   macro avg       0.94      0.82      0.87       290
weighted avg       0.95      0.95      0.95       290
```

The optimization of accuracy could be seen if we remove nodes starting from leaf nodes.
These high values of accuracy might be the result of overfitting that can be removed using pruning in further questions.

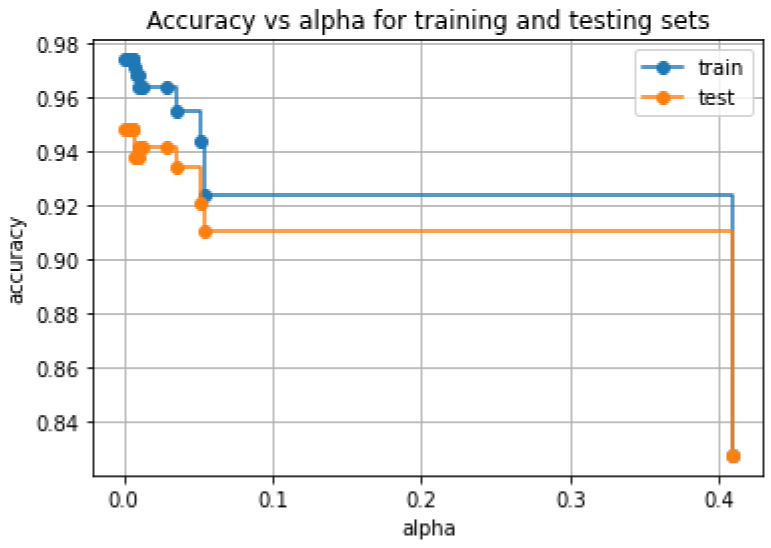
(B) Q2

Cost Complexity pruning technique(Post pruning)
Cost complexity pruning helps to control the size of a tree.Cost complexity pruning technique is parameterized by the cost complexity parameter,ccp_alpha.Greater values of ccp_alpha, more the number of nodes that are pruned.

We use the DT-A that we got from the end of question (A) and generate all ccp_alpha values based on x_train and y_train.

Then for each ccp_alpha, we fit the model and keep on appending them.

Next we plot the accuracy vs ccp_alpha graph where we can observe the training and testing accuracy with increasing ccp_alpha values.

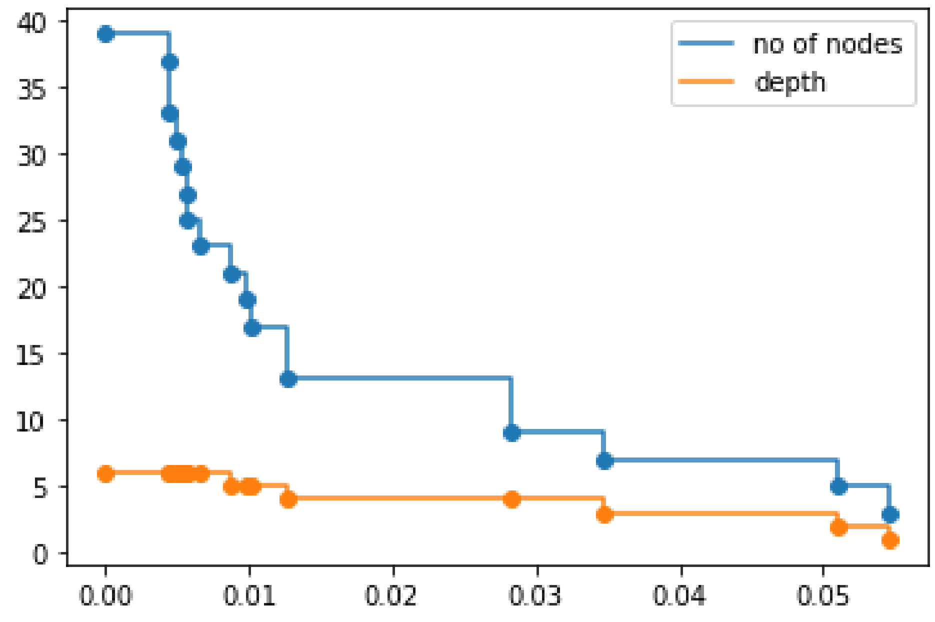


In the above graph, we can observe that at ccp_aplha value of 0.025, we get maximum accuracy for both training data and testing data simultaneously. So, we choose ccp_alpha value as 0.025.

Then we fit the tree by giving the ccp_alpha hyperparameter value as 0.025. The scores after this post pruning technique are observed as follows.

Accuracy score: 0.9413793103448276
Precision score: 0.9413793103448276
Recall score: 0.9413793103448276
AUC_ROC score: 0.9543027777777778

With increase in value of alpha, the number of nodes and depth is decreasing. This can be visualized from the following scatter plot.



Other pruning technique - Pre pruning

We can use pre pruning before the construction of the decision tree using hyper parameter tuning that will help us to overcome the overfitting issue that we observed in the previous question.

Again taking the same `DT_A = tr_model`

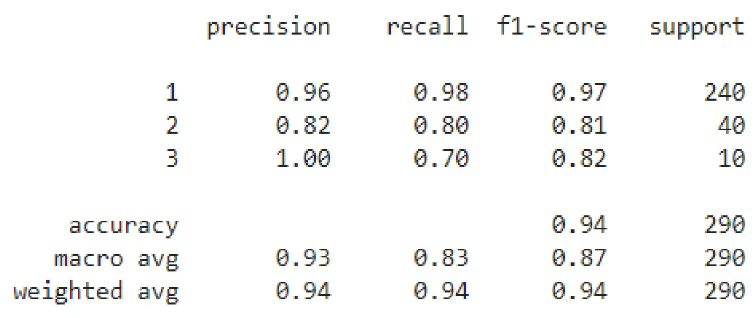
For selecting the best parameters for hyperparameters tuning we have used GrisSearchCV that results in the best possible parameters in the given range to feed into the classifier.

```
{'criterion': 'entropy', 'max_depth': 7, 'min_samples_leaf': 4,
'min_samples_split': 13, 'splitter': 'best'}
```

Now, using this result in the pre pruning operation.

Test results after pre pruning:

```
Accuracy score:  0.9448275862068966
Precision score:  0.9448275862068966
Recall score:  0.9448275862068966
AUC_ROC score:  0.94145
```



```
              precision    recall  f1-score   support

           1       0.96      0.98      0.97       240
           2       0.82      0.80      0.81        40
           3       1.00      0.70      0.82        10

    accuracy                           0.94       290
   macro avg       0.93      0.83      0.87       290
weighted avg       0.94      0.94      0.94       290
```

Observations:
Though pre pruning computes faster and helps in trade-off but here the overall accuracy decreased.
The decision tree that we got is definitely more interpretable as compared to part A.
Also, using pruning we are able to limit overfitting as best parameters are selected so the pruned tree fits the data well.

(B) Q3

(C) Q1

Method1:
Step1: Split both the datasets 80-20 ratio.
Step2 Train both the models
Step3: Compare the performances
Method2:
Use VFDT using Hoeffding Tree
Define the dataset complete and another for streaming
Stream the data one after another
Plot the performances after each sample added to the dataset.
So, the performance is increasing with each data sample added to the sample. If more streaming data is added, we will get better performance.

(C) Q2

Decision surface boundaries

Here we train the decision tree trained on pairs of features. For each possible pair, the decision tree learns the decision boundaries. We have plotted surface boundaries by taking two features at a time.

Learnings: We have learnt the following.

- How to fit the decision tree using the decision tree classifier present in sklearn.
- Calculate different metrics such as precision, recall, auc_roc curve and many more.
- Equal distribution of classes in testing and training data sets.
- Pre pruning and post pruning techniques such as cost complexity pruning technique.
- Different hyperparameters for decision tree classifiers.
- Implementation of decision surface boundaries.

References:

- Scikit-learn documentation
- https://www.kaggle.com/prashant111/decision-tree-classifier-tutorial
- Towards data science - Medium
- Analytics Vidya