CSE506: Introduction to Data Mining
Assignment-1: Report

Nimisha Gupta (2019315)
Vani Sikka (2019395)

(A) TRAINING

Importing the dataset values
We were given 2 datasets, ‘data_1.csv’ and ‘data_2.csv’ in the form of comma separated files. We imported these using the pandas.read_csv() function.

Pre-processing of data
Pre-processing of data is the modification of data that is done to enhance its performance. Pre-processing was done to remove all the null values and duplicate values.

Splitting the DataSet
We were required to split the dataset 1 in 80-20 ratio while maintaining equal class distribution in both train and test set.
For this, the following syntax was used:
X_train, X_test, y_train, y_test= train_test_split(X,y,test_size=0.2, stratify=y)
Here, X is the array of feature names:
y is the target class column (”fetal_health” was given as the target class column)
test_size=0.2 ensures the required 80-20 ratio
stratify=y maintains equal class distribution in both train and test set

Answer 1)
Classifying the dataset using Decision Tree classifier:
For this step, we need to import DecisionTreeClassifier from sklearn.tree
An instance of this model was made using the DecisionTreeClassifier function.
Fit function was used to train the model on the data and predict the data.

Visualizing and Saving the Decision Tree
We plotted the decision tree using plot_tree function of matplotlib and displayed it using the plt.show() function. The tree was saved as 'DT_A_1.png'.

Calculating:

1. Precision: Precision is calculated using the precision_score function.
   precision = precision_score(y_test, predicted, labels=[1,2,3], average='macro')
   Precision: 0.92181377122597496

2. Recall: Recall is calculated using the recall_score function.
   recall = recall_score(y_test, predicted, labels=[1,2,3], average='macro') print('Recall: %.17f' % recall)
   Recall: 0.92072829131652656

3. Accuracy: Accuracy is calculated using the accuracy_score function.
   accuracy = metrics.accuracy_score(y_test, predicted)
   Accuracy: 0.9513888888888888

4. AUC-ROC curve:
   Area Under Class-1 : 0.92
   Area Under Class-2 : 0.90
   Area Under Class-3 : 0.95

Answer 2)
To train the Decision Tree classifier on different depths, we modified the DecisionTreeClassifier() function by changing the value of max_depth.
We then used the metrics.accuracy_score(y_test, predicted) function to calculate the accuracy at all the different values of max_depth.
We then stored all the max_depth values in an array called x1 and their corresponding values of accuracy (which we just calculated) in an array called y1.
x1= [2,3,4,5,6,8,10]
y1=[0.9344827586206896,0.9482758620689655,0.9620689655172414,0.9620689655172414,0.9724137931034482,0.9793103448275862,0.9793103448275862]

We then used the plot function and passed the 2 arrays x1 and y1 as attributes to get the accuracy vs depth graph.

Answer 3)
Changing the HyperParameters

1. Criterion

Decision Tree formation:
Here, we again plotted the decision tree like we did in Q1. The additional feature was that now, the criterion was ”entropy”.

Observations:
This plot led to the following observations:

a) Accuracy: 0.9513888888888888 =
b) Precision: 0.94135802469135799 >
c) Recall: 0.88606442577030808 <
d) AUC-ROC curve:
    i) Area Under Class-1 : 0.90 <
    ii) Area Under Class-2 : 0.88 <
    iii) Area Under Class-3 : 0.95 =

Reasoning:
Accuracy **remains the same** with respect to the accuracy of the base model.
Precision **has increased** with respect to the precision of the base model.
Recall **has decreased** with respect to the recall of the base model.
Area Under Class-1 **has decreased**.
Area Under Class-2 **has decreased**.
Area Under Class-3 **remains the same**.
We see such a decrease in recall and AUC values because Entropy varies over 0 to 1 , since the variability of entropy value is higher we see such a decrease.

2. Splitter= “random”

Decision Tree formation:
Here, we again plotted the decision tree like we did in Q1. The additional feature was that splitter is now kept ‘random’.

Observations:
This plot led to the following observations:

a) Accuracy: 0.9270833333333334 <
b) Precision: 0.7966502066920476 7 <
c) Recall: 0.82352941176470595 <
d) AUC-ROC curve:
    i) Area Under Class-1 : 0.91 <
    ii) Area Under Class-2 : 0.83 <
    iii) Area Under Class-3 : 0.89 <

Reasoning:
Accuracy **has decreased** with respect to the accuracy of the base model.
Precision **has decreased** with respect to the precision of the base model.
Recall **has decreased** with respect to the recall of the base model.
Area Under Class-1 **has decreased**.

Area Under Class-2 **has decreased**.
Area Under Class-3 **has decreased**.

We see an overall negative performance on changing splitter hyperparameters as compared to the base model.This because when the splitter value is set to randan , then the randon best split is used. Since we get the random best split as a split that would split the tree in the best way , that is probabilistic. Since this probability we see a decreased performance.

3. Min Samples Split = 8

Decision Tree formation:
Here, we again plotted the decision tree like we did in Q1. The additional feature was that now, the min_samples_split=8.

Observations:
This plot led to the following observations:

a) Accuracy: 0.9444444444444444 <
b) Precision: 0.92217573221757332 <
c) Recall: 0.89019607843137250 <
d) AUC-ROC curve:
    i) Area Under Class-1 : 0.94 >
    ii) Area Under Class-2 : 0.92 >
    iii) Area Under Class-3 : 0.95 =

Reasoning:
Accuracy **has decreased** with respect to the accuracy of the base model.
Precision **has decreased** with respect to the precision of the base model.
Recall **has decreased** with respect to the recall of the base model.
Area Under Class-1 **has increased**.
Area Under Class-2 **has increased**.
Area Under Class-3 **remains the same**.

By setting the Min Sample Split to 8 we are telling the classifier that minimum 8 samples are required for spitting the internal node. Using minimum 8 samples but in base model minimum 2 samples are used, since we are considering 8 incase of 2 (in the minimum scenario) our samples increase, which increases the variability due to which we see the Accuracy , Precision and recall decrease.

4. Max Depth= 6

Decision Tree formation:
Here, we again plotted the decision tree like we did in Q1. The additional feature was that max_depth=6.

Observations:
This plot led to the following observations:

a) Accuracy: 0.9583333333333334 >
b) Precision: 0.95198902606310021 >
c) Recall: 0.89579831932773113 <
d) AUC-ROC curve:
    i) Area Under Class-1 : 0.94 >
    ii) Area Under Class-2 : 0.93 >
    iii) Area Under Class-3 : 0.95 =

Reasoning:
Accuracy **has increased** with respect to the accuracy of the base model.
Precision **has increased** with respect to the precision of the base model.
Recall **has decreased** with respect to the recall of the base model.
Area Under Class-1 **has increased**.
Area Under Class-2 **has increased**.
Area Under Class-3 **remains the same**.

The max depth 6 is lesser that of the base model, but this model has a better performance, this is because as the depth increase, the train error reduce but test error increase due to overfitting and the base model is overfitted , but since this model has a controlled depth hence it is to a some extent reducing overfitting. Because of this reason we see that there is a positive performance from the base model.

5. Min Samples Leaf

Decision Tree formation:
Here, we again plotted the decision tree like we did in Q1. The additional feature was that min_samples_leaf=3.

Observations:
This plot led to the following observations:

a) Accuracy: 0.9409722222222222 <
b) Precision: 0.91259361450532117 <
c) Recall: 0.89572829131652654 <
d) AUC-ROC curve:

i) Area Under Class-1 : 0.94 >
ii) Area Under Class-2 : 0.92 >
iii) Area Under Class-3 : 0.95 =

Reasoning:
Accuracy **has decreased** with respect to the accuracy of the base model.
Precision **has decreased** with respect to the precision of the base model.
Recall **has decreased** with respect to the recall of the base model.
Area Under Class-1 **has increased**.
Area Under Class-2 **has increased**.
Area Under Class-3 **remains the same**.

We see a decreased performance when we set this parameter as 3. We are telling the model that the minimum number of samples required to be at a leaf node are 3 but the default is 1. So in the case of the base model when only one sample is left that decision node is changed to leaf node, so this model is bound to predict better than the model where there are 3 samples left and the node becomes a leaf node.

6. Max Features

Decision Tree formation:
Here, we again plotted the decision tree like we did in Q1. The additional feature was that max_features="log2".

Observations:
This plot led to the following observations:

a) Accuracy: 0.9270833333333334 <
b) Precision: 0.83026437847866408 <
c) Recall: 0.80966386554621861 <
d) AUC-ROC curve:
    i) Area Under Class-1 : 0.87 <
    ii) Area Under Class-2 : 0.81 <
    iii) Area Under Class-3 : 0.89 <

Reasoning:
Accuracy **has decreased** with respect to the accuracy of the base model.
Precision **has decreased** with respect to the precision of the base model.
Recall **has decreased** with respect to the recall of the base model.

Area Under Class-1 **has decreased**.
Area Under Class-2 **has decreased**.
Area Under Class-3 **has decreased**.

We see a decreased performance because when max feature is set to log we are telling the model that The number of features to consider when looking for the best split should be logarithmic.This reduces the number of features to consider than the features are are considered by the base model, because of which the performance decrease.

7. Class Weight

Decision Tree formation:
Here, we again plotted the decision tree like we did in Q1. The additional feature was that class_weight='balanced'.

Observations:
This plot led to the following observations:

a) Accuracy: 0.9340277777777778 <
b) Precision: 0.90342924714229411 <
c) Recall: 0.88599439775910360 <
d) AUC-ROC curve:
    i) Area Under Class-1 : 0.90 <
    ii) Area Under Class-2 : 0.88 <
    iii) Area Under Class-3 : 0.95 =

Reasoning:
Accuracy **has decreased** with respect to the accuracy of the base model.
Precision **has decreased** with respect to the precision of the base model.
Recall **has decreased** with respect to the recall of the base model.
Area Under Class-1 **has decreased**.
Area Under Class-2 **has decreased**.
Area Under Class-3 **remains the same**.

The decreased performance is due to The "balanced" mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as n_samples / (n_classes * np.bincount(y)). Due to this relation we observe a decreased performance.

8. Max Leaf Nodes

Decision Tree formation:
Here, we again plotted the decision tree like we did in Q1. The additional feature was that max_leaf_nodes=10.

Observations:
This plot led to the following observations:

a) Accuracy: 0.9548611111111112 >
b) Precision: 0.94393567120839850 >
c) Recall: 0.89439775910364139 <
d) AUC-ROC curve:
    i) Area Under Class-1 : 0.93 >
    ii) Area Under Class-2 : 0.91 >
    iii) Area Under Class-3 : 0.91 <

Reasoning:
Accuracy **has increased** with respect to the accuracy of the base model.
Precision **has increased** with respect to the precision of the base model.
Recall **has decreased** with respect to the recall of the base model.
Area Under Class-1 **has increased**.
Area Under Class-2 **has increased**.
Area Under Class-3 **has decreased**.

Here we see that the accuracy and precision is decreased because we have set Max Leaf Nodes as 10, whereas in the base model it is node.Grow a tree with max_leaf_nodes in best-first fashion. Because of which we relax in obtaining the correct prediction. Due to this trade, we can say that there is a decreased performance.

BEST OBTAINED HYPERPARAMETERS ARE:
class_weight: None
criterion: ‘gini’
max_depth: 6
max_features: None
max_leaf_nodes: None
min_samples_leaf: 3
min_samples_split: 8

splitter: ‘best’

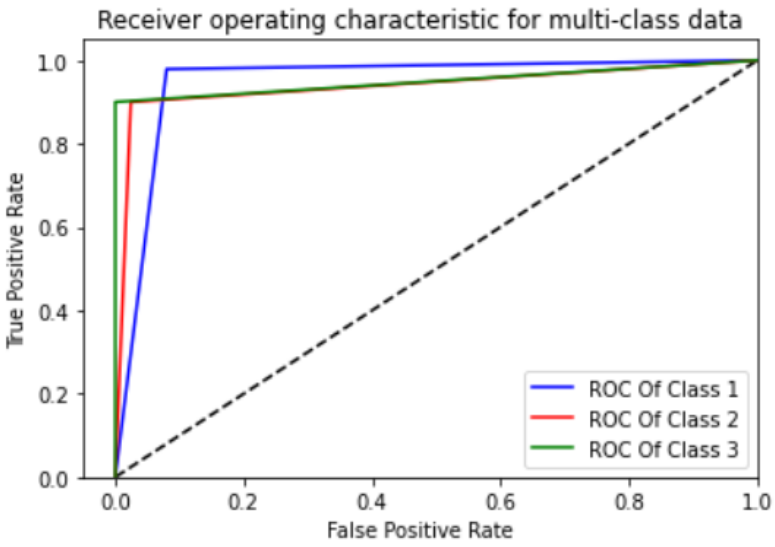
(B) POST PRUNING

Answer 1)
After preprocessing and splitting the dataset values (like what was done in Part A)
We classified the dataset using the Decision Tree classifier.
For this step, we need to import DecisionTreeClassifier from sklearn.tree
An instance of this model was made using the DecisionTreeClassifier function.
Fit function was used to train the model on the data and predict the data and predicted was calculated using the predict function.

Removing a node from the Decision Tree:
For this, TREE_LEAF was imported from sklearn.tree._tree
The following syntax was used to delete a particular node:

```
clf.tree_.children_left[2] = TREE_LEAF
```

tree.plot_tree(fig) was then used to plot this new tree
This tree was saved as an image (DT_B_1.png).



Observations:

This plot led to the following observations:

a) Accuracy: 0.9652777777777778
b) Precision: 0.94675507333735176
c) Recall: 0.92633053221288508
d) AUC-ROC Curve:
    i) Area Under Class-1 : 0.95
    ii) Area Under Class-2 : 0.94
    Area Under Class-3 : 0.95

Reasoning:

Accuracy **has increased** with respect to the accuracy of the base model.
Precision **has increased** with respect to the precision of the base model.
Recall **has increased** with respect to the recall of the base model.
Area Under Class-1 **has increased**.
Area Under Class-2 **has increased**.
Area Under Class-3 **remains the same**.

Answer 2)

Cost Complexity Pruning:

THis was done using complexity_pruning_path function of Decision Tree Classifier to get various ccp_alpha values.

Next we created different Decision Tree classifiers on different values of ccp_alphas and calculated Accuracy of Each Decision Tree.

THen we plotted ccp_alphas vs Accuracy graph for both Tain Accuracy and Test Accuracy. This was done using the set, line plot, and xticks functions.
The following graph is what we got.

From the above graph, we found the value of alpha as the intersecting point of the 2 graphs.
We then took an optimal value of cpp_alphas and built a tree from those values.

Observations:

This plot led to the following observations:

a) Accuracy: 0.9444444444444444
b) Precision: 0.92226832641770395
c) Recall: 0.82633053221288522
d) AUC-ROC Curve:
    i) Area Under Class-1 : 0.89

ii) Area Under Class-2 : 0.86
iii) Area Under Class-3 : 0.85

Hyperparameter Tuning:
We wanted to find a set of optimal hyperparameters. Thus we did hyperparameter tuning.
So we set various hyperparameters and all the possible values of each hyperparameter.

```
"criterion": ["gini", "entropy"],
"splitter":["best","random"],
"max_depth": [1,2,3,4,5,6,7,None],
"min_samples_leaf": [2,3,4,5,6,7,8,1],
"min_samples_split": [3,4,5,6,7,8,9,2],
"max_features": ["auto", "sqrt", "log2",None],
"class_weight": ["balanced",None],
"max_leaf_nodes": [1,2,3,4,5,6,7,None]
```

We then used grid search to train the data and finally used the grid.best_params_ command for Hyperparameter Tuning.
We finally get all the best values of the given hyperparameters.
These are:

```
{'class_weight': None,
 'criterion': 'gini',
 'max_depth': 6,
'max_features': None,
'max_leaf_nodes': None,
'min_samples_leaf': 3,
'min_samples_split': 8,
'splitter': 'best'}
```

Using this information, we made a pre-pruned tree found out the following:

(i) Accuracy: 0.9652777777777778
(ii) Precision: 0.93850627486991123
(iii) Recall: 0.91246498599439774
(iv) AUC-ROC graph:
Area Under Class-1 : 0.93
Area Under Class-2 : 0.92
Area Under Class-3 : 1.00

(C) EXPERIMENTS

Answer 1:

We used skmultiflow.trees.ExtremelyFastDecisionTreeClassifier of the scikit- multiflow library to obtain an extremely fast Decision Tree.This type of decision tree is extremely useful in handling large data sets.
So in our code we first make an extremely fast Decision Tree classifier, then we train the classifier on that data_set 1. We use the function partial_fit, which is useful to incrementally make the model learn new rules as in when new data is fed. Once the initial model is formed then all the required statistics are also displayed.
Then we again feed the same model with data_2, and the model changes accordingly as we again use the partial_fit function. Then we again calculate the required statistics.

Learnings:

- In depth knowledge of Decision trees
- The conceptual understanding of AUC-ROC curve
- Understanding of SLIQ
- Introduction to Extremely Fast Decision Tree

References

- https://scikit-multiflow.readthedocs.io/en/stable/api/generated/skmultiflow.trees.ExtremelyFastDecisionTreeClassifier.html
- https://towardsdatascience.com/data-pre-processing-techniques-you-should-know-8954662716d6
- https://towardsdatascience.com/finding-and-removing-duplicate-rows-in-pandas-dataframe-c6117668631f
- https://towardsdatascience.com/visualizing-decision-trees-with-python-scikit-learn-graphviz-matplotlib-1c50b4aa68dc
- Source code provided in the classroom
- https://stackoverflow.com/questions/51378105/plot-multi-class-roc-curve-for-decisiontree classifier
- https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html
- https://scikit-multiflow.readthedocs.io/en/stable/api/generated/skmultiflow.data.DataStream.html#skmultiflow.data.DataStream
- https://medium.com/analytics-vidhya/post-pruning-and-pre-pruning-in-decision-tree-561f3df73e65
- https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
-