MACHINE LEARNING GROUP-40 ASSIGNMENT-1 REPORT

Neha Dalmia 19CS30055 **Hritaban Ghosh 19CS30053**

Task:

Dataset Link : https://archive.ics.uci.edu/ml/datasets/Avila

The Avila data set has been extracted from 800 images of the 'Avila Bible', an XII century giant Latin copy of the Bible. The prediction task consists in associating each pattern to a copyist.

The task requires us to classify a sample into one of the classes A, B, C, D, E, F, G, H, I, W, X, Y based on the following attributes:

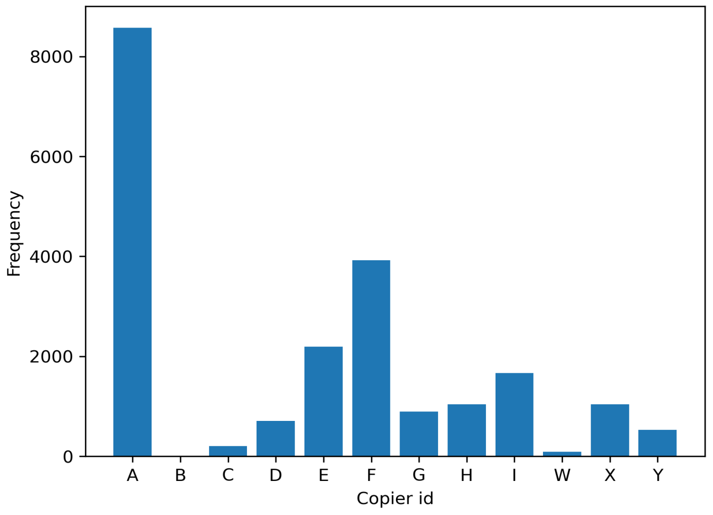
F1: intercolumnar distance ,
F2: upper margin,
F3: lower margin,
F4: exploitation,
F5: row number,
F6: modular ratio,
F7: interlinear spacing,
F8: weight,
F9: peak number,
F10: modular ratio/ interlinear spacing.

Data Analysis and Visualisation

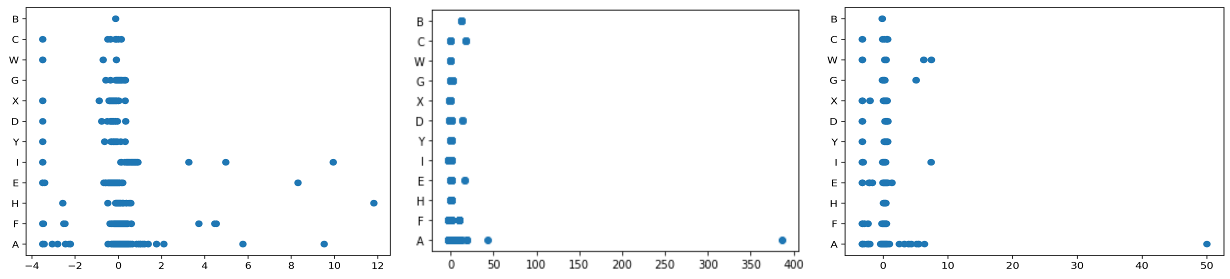
We combined the training and test sets available on the website into a single dataset. We first printed the values which can be NULL or missing in any of the samples. We did not obtain any such values hence no preprocessing was required. There was also no attribute with the same value for all classes hence we did not need to remove any attribute either.

```
 >>print(df.isnull().sum())
intercolumnar distance              0
upper margin                        0
lower margin                        0
exploitation                        0
row number                          0
modular ratio                       0
interlinear spacing                 0
weight                              0
peak number                         0
modular ratio/ interlinear spacing  0
Class                               0
 dtype: int64
```

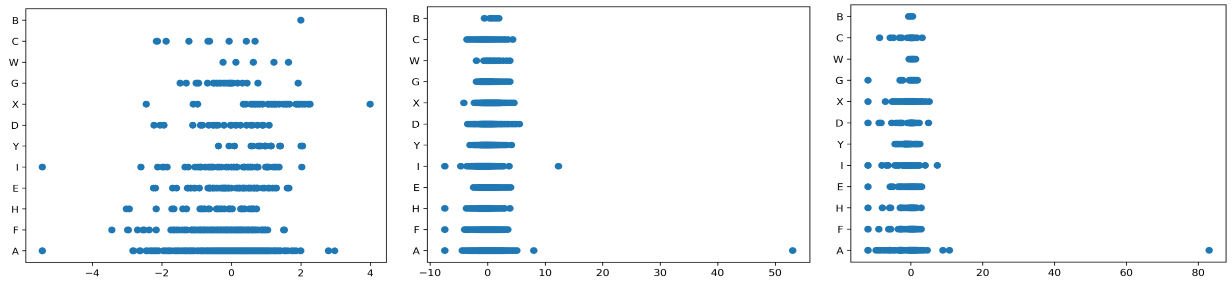
CLASS DISTRIBUTION of the combined set



As we can see , the dataset is extremely **biased** towards class A and some classes like B and W appear a negligible amount of times so classifying them correctly will be harder. We also obtained scatter plots for the different attributes to estimate their ranges and relation with the classes, a few have been shown below.



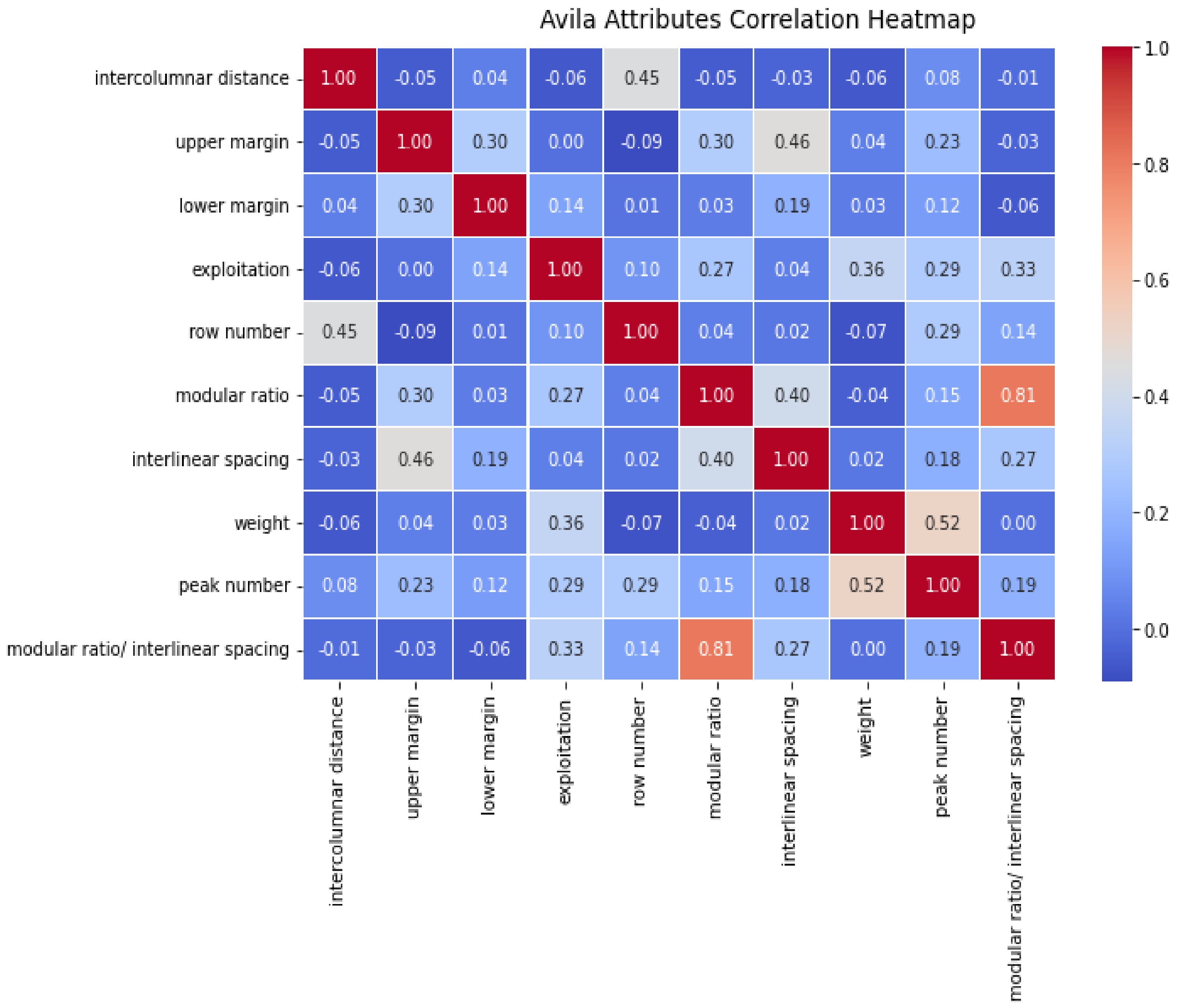
F1: intercolumnar distance **F2: upper margin** **F3: lower margin**



F4: exploitation **F6: modular ratio** **F7: interlinear spacing**

It is clear that there is some **noise** in the dataset as there are stray points for class A with values much higher than the mean range for that attribute, but the proportion of noise to the dataset is very less as our dataset is considerably large. Hence there might **not** be much **over-fitting happening** here.

The next graph that we make is the **heatmap**, this lets us know the correlation between any two attributes. This knowledge enables us to drop any columns that have high correlation with another column, so that the model doesn't have any redundancy in it.



	intercolumnar distance	upper margin	lower margin	exploitation	row number	modular ratio	interlinear spacing	weight	peak number	modular ratio/ interlinear spacing
intercolumnar distance	1.00	-0.05	0.04	-0.06	0.45	-0.05	-0.03	-0.06	0.08	-0.01
upper margin	-0.05	1.00	0.30	0.00	-0.09	0.30	0.46	0.04	0.23	-0.03
lower margin	0.04	0.30	1.00	0.14	0.01	0.03	0.19	0.03	0.12	-0.06
exploitation	-0.06	0.00	0.14	1.00	0.10	0.27	0.04	0.36	0.29	0.33
row number	0.45	-0.09	0.01	0.10	1.00	0.04	0.02	-0.07	0.29	0.14
modular ratio	-0.05	0.30	0.03	0.27	0.04	1.00	0.40	-0.04	0.15	0.81
interlinear spacing	-0.03	0.46	0.19	0.04	0.02	0.40	1.00	0.02	0.18	0.27
weight	-0.06	0.04	0.03	0.36	-0.07	-0.04	0.02	1.00	0.52	0.00
peak number	0.08	0.23	0.12	0.29	0.29	0.15	0.18	0.52	1.00	0.19
modular ratio/ interlinear spacing	-0.01	-0.03	-0.06	0.33	0.14	0.81	0.27	0.00	0.19	1.00

As you can see from the heat map that there are no two distinct columns whose correlation is greater than 0.9, therefore we avoid any redundancies in our Decision Tree Classifier.

Decision Tree Construction

We performed 10 random **80:20** splits of the data into training and test sets. The training split is further split by **80:20** ratio into training and validation sets (for the purpose of pruning later on). We create the tree using the **ID3** algorithm where at every step we decide the best attribute for branching.
Since our decision tree has continuous attribute values, we first **sort** the data by the attribute values at every node for the attribute we are currently considering. We then iterate over all possible lengths of the left and right splits. The threshold is chosen as the mean of the rightmost value of the left split and leftmost value of the right split. The gain is found at every possible split and we choose the threshold giving us the maximum gain. Values <= the threshold are sent to the left child and others to the right child of the tree.

We obtain the test accuracy for each of the 10 random splits and choose the decision tree giving the best test accuracy. Since our decision tree deals with categorical data, the accuracy measure is:

$$test\ accuracy\ = \frac{Number\ of\ test\ samples\ in\ which\ (h(x) == c(x))}{Total\ number\ of\ test\ samples}$$

In our **initial approach (Constant Depth Binary Decision Tree Approach),** we removed an attribute once it was considered and did not allow it to be reused for the descendants of that node. However, since we are dealing with continuous values of attributes, after performing a split we might want to split one of the splits further using the same attribute. To incorporate this, we allowed the same attribute to be considered for the descendants of a node as well in the **final approach (Variable Depth Binary Decision Tree Approach).**

The following pseudo code summarises our final process for decision tree construction:

```
Tree Construction:

ConstructTree(samples, target_values, attributes_left,
impurity_measure, max_depth):
    # samples are training examples.
    # target_values is the attribute whose value is to be predicted
    by the tree.
    # attributes_left is a list of attributes that may be tested by
    the learned decision tree.
    # impurity measure is the metric used for choosing the best
    attribute with the highest gain.
    # max_depth is a restriction imposed on the depth of the tree

  • Create a Root node for the tree
```

- If all examples belong to a unique target class, then Return the single-node tree Root, with label = that unique target_value.
- If attribute_left is empty, then Return the single-node tree Root with label = most frequent class in target_values.
- If samples are empty, then Return the single-node tree Root with label = most frequent class in target_values.
- If depth has reached max_depth , then Return the single-node tree Root with label = most frequent class in target_values.
- Otherwise Begin
    - A ← the attribute from attribute_left that gives the best gain
    - The decision attribute for Root ← A
    - For each partition p_i of A,
        - Add a new tree branch below root corresponding to $v(A) \in p_i$.
        - Get subset of samples for $p_i(sub(p_i))$ and call *ConstructTree(sub(p_i), target_values(sub(p_i)), attribute_left, impurity_measure, max_depth)*
- End
- Return Root

We have a node class which contains the required data members to perform the above mentioned calculations.

To choose the best attribute:

For all attributes in list of possible attributes for a node repeat the following:

- Sort the samples based on the values of this attribute
- Get all possible left and right splits where left consists of samples from [1..i] and right from [i..n] where n is the number of samples in that node
- Calculate the gini gain for all the possible splits
- Choose the split that gives maximum gini gain.
- Choose the attribute whose maximum gain split's gain value is maximum.
- Threshold for this node = (leftsplit[-1]+rightsplit[0])/2. Samples with attribute values less than or equal to the threshold are sent to the left child otherwise to the right child.

For predictions:

For a given sample x at each node we look at the attribute value at the node and the threshold value. If the x[attribute_value] <= threshold, we go to the left child of the node otherwise we go to the right child of the node. On encountering a leaf node, we return the "vote" which is the class of that node.

For printing the tree:

We use a library called graphviz and traverse the tree in a Breadth First Search manner to print it in that manner.

This process is repeated for both i) Entropy ii) Gini as impurity measures.

Results

The results obtained in this section are after performing 10 random splits (80/20) on the data set and constructing the decision tree using those samples. We take note of the maximum accuracy and average accuracy over the 10 splits. We then continue to use the tree that gave us the maximum accuracy.

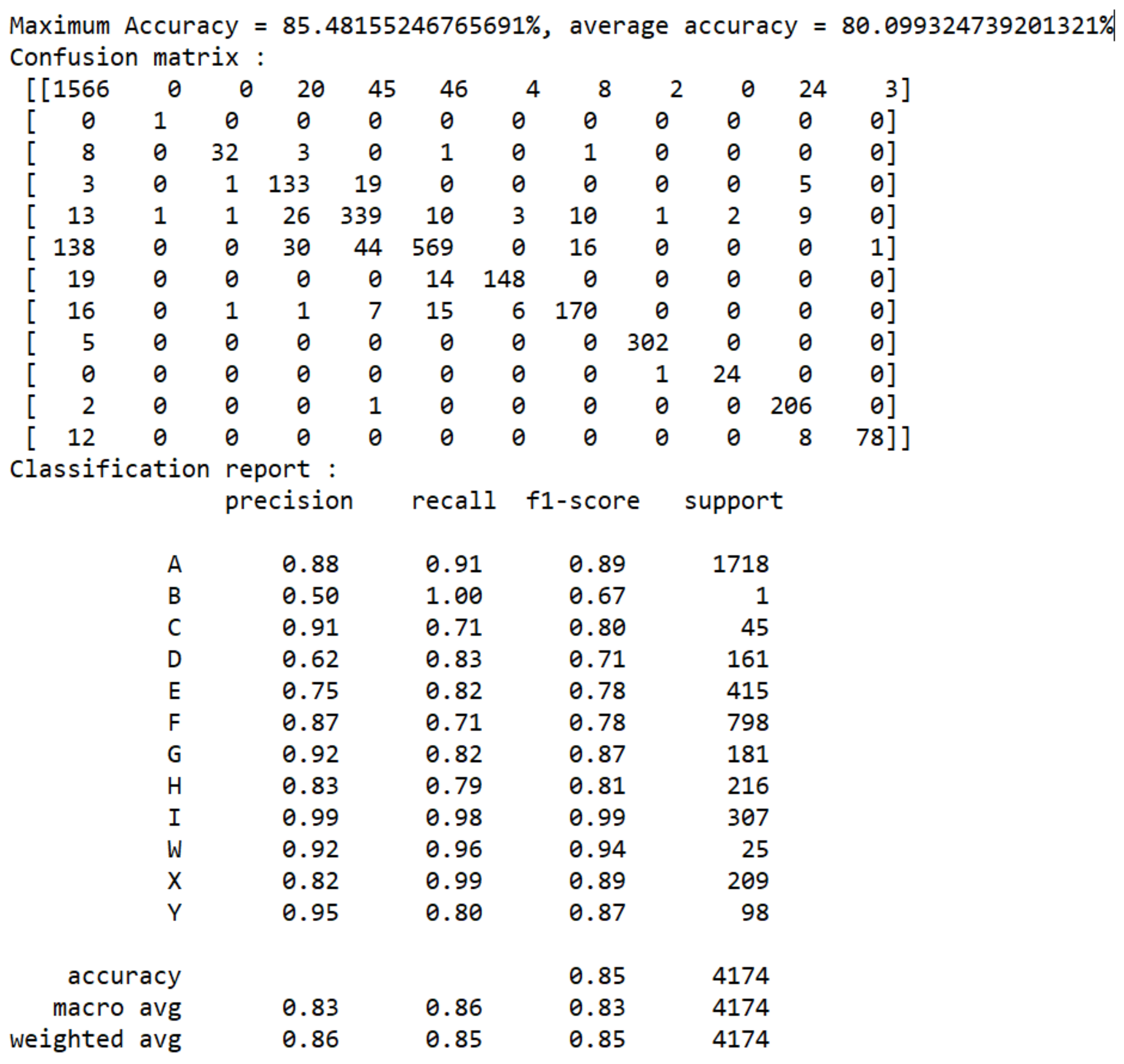
- Initially we used the **Constant Depth Binary Decision Tree approach** which gave some results that we were not satisfied with (using both gini and entropy) even after pruning.
- We then changed our approach to **Variable Depth Binary Decision Tree approach**, which gave incredibly good results (using both gini and entropy).
- We then continue our depth, nodal and pruning analysis using the trees obtained in the second approach.

Gini as impurity measure:

Initial Approach (Constant Depth Binary Decision Tree approach):

Maximum test Accuracy = 78.70148538572113%
Average test accuracy = 75.6851940584571%

Final Approach (Variable Depth Binary Decision Tree approach):



```
Maximum Accuracy = 85.48155246765691%, average accuracy = 80.099324739201321%
Confusion matrix :
 [[1566    0    0   20   45   46    4    8    2    0   24    3]
 [   0    1    0    0    0    0    0    0    0    0    0    0]
 [   8    0   32    3    0    1    0    1    0    0    0    0]
 [   3    0    1  133   19    0    0    0    0    0    5    0]
 [  13    1    1   26  339   10    3   10    1    2    9    0]
 [ 138    0    0   30   44  569    0   16    0    0    0    1]
 [  19    0    0    0    0   14  148    0    0    0    0    0]
 [  16    0    1    1    7   15    6  170    0    0    0    0]
 [   5    0    0    0    0    0    0    0  302    0    0    0]
 [   0    0    0    0    0    0    0    0    1   24    0    0]
 [   2    0    0    0    1    0    0    0    0    0  206    0]
 [  12    0    0    0    0    0    0    0    0    0    8   78]]
Classification report :
              precision    recall  f1-score   support

           A       0.88      0.91      0.89      1718
           B       0.50      1.00      0.67         1
           C       0.91      0.71      0.80        45
           D       0.62      0.83      0.71       161
           E       0.75      0.82      0.78       415
           F       0.87      0.71      0.78       798
           G       0.92      0.82      0.87       181
           H       0.83      0.79      0.81       216
           I       0.99      0.98      0.99       307
           W       0.92      0.96      0.94        25
           X       0.82      0.99      0.89       209
           Y       0.95      0.80      0.87        98

    accuracy                           0.85      4174
   macro avg       0.83      0.86      0.83      4174
weighted avg       0.86      0.85      0.85      4174
```

Maximum test accuracy = 85.481%
Average test accuracy = 80.10%

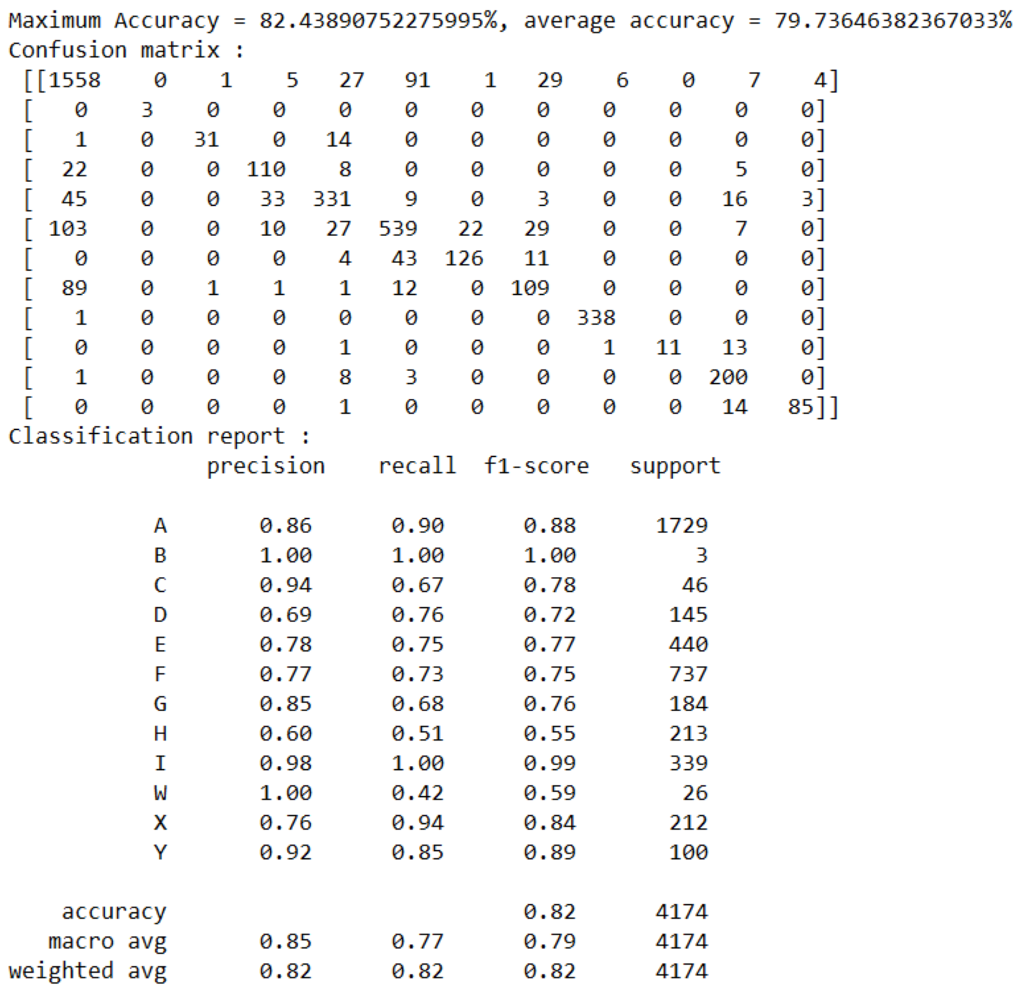
Number of nodes in the tree with Maximum Accuracy (not pruned) = 977

Entropy as impurity measure:

Initial Approach (Constant Depth Binary Decision Tree approach):

Maximum test accuracy = 79.65979875419262%
Average test accuracy = 76.72975563009105%

Final Approach (Variable Depth Binary Decision Tree approach):



```
Maximum Accuracy = 82.43890752275995%, average accuracy = 79.73646382367033%
Confusion matrix :
 [[1558    0    1    5   27   91    1   29    6    0    7    4]
 [   0    3    0    0    0    0    0    0    0    0    0    0]
 [   1    0   31    0   14    0    0    0    0    0    0    0]
 [  22    0    0  110    8    0    0    0    0    0    5    0]
 [  45    0    0   33  331    9    0    3    0    0   16    3]
 [ 103    0    0   10   27  539   22   29    0    0    7    0]
 [   0    0    0    0    4   43  126   11    0    0    0    0]
 [  89    0    1    1    1   12    0  109    0    0    0    0]
 [   1    0    0    0    0    0    0    0  338    0    0    0]
 [   0    0    0    0    1    0    0    0    1   11   13    0]
 [   1    0    0    0    8    3    0    0    0    0  200    0]
 [   0    0    0    0    1    0    0    0    0    0   14   85]]
Classification report :
              precision    recall  f1-score   support

           A       0.86      0.90      0.88      1729
           B       1.00      1.00      1.00         3
           C       0.94      0.67      0.78        46
           D       0.69      0.76      0.72       145
           E       0.78      0.75      0.77       440
           F       0.77      0.73      0.75       737
           G       0.85      0.68      0.76       184
           H       0.60      0.51      0.55       213
           I       0.98      1.00      0.99       339
           W       1.00      0.42      0.59        26
           X       0.76      0.94      0.84       212
           Y       0.92      0.85      0.89       100

    accuracy                           0.82      4174
   macro avg       0.85      0.77      0.79      4174
weighted avg       0.82      0.82      0.82      4174
```

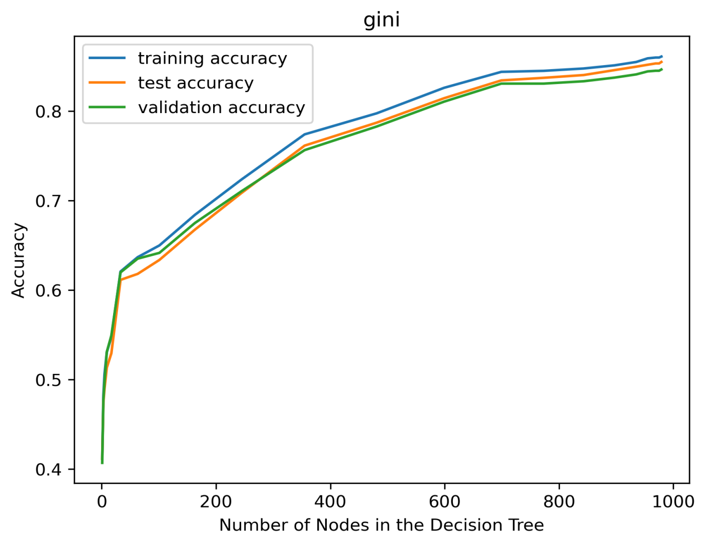
Maximum test accuracy = 82.439%
Average test accuracy = 79.736%
Number of nodes in the tree with Maximum Accuracy (not pruned) = 723

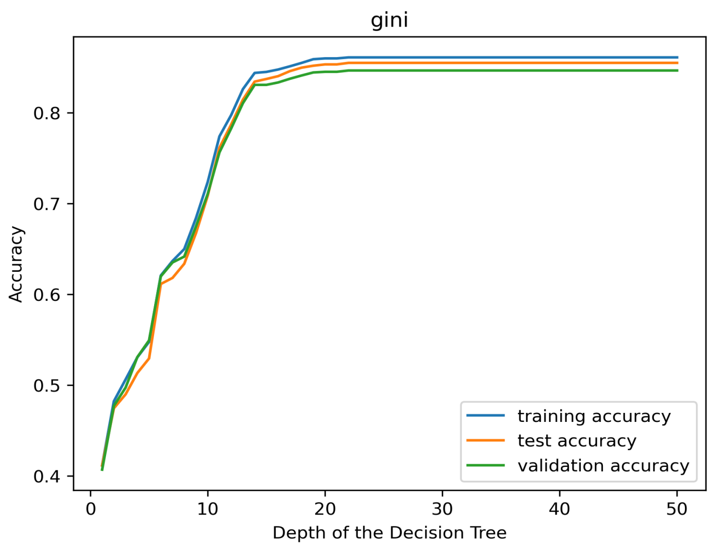
Comments: As most of the values are along the diagonal of the matrix, and the f1-scores are high as well, we can see that the high accuracy is not due to some biased class and also that overfitting does not seem like a likely scenario. It is also even able to classify classes like B (it has a perfect f-1 score) which have very samples correctly so the bias towards A is not affecting our model much. Since most of the values in the confusion matrix are along the diagonal, a large number of correct classifications for each class are happening. The precision of our predictions is also very high and is even 1 for some classes. Also, our final approach shows a significant improvement from the initial approach as we have a much improved accuracy and a fewer number of misclassifications.

Step 3: Depth analysis

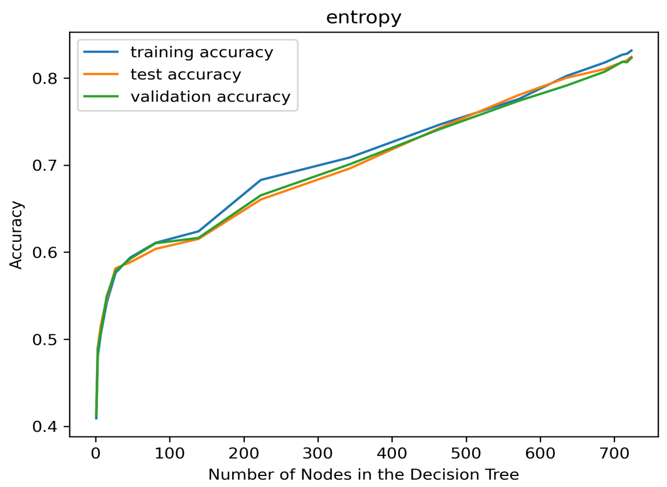
We varied the depths for the decision tree for both impurity measures obtained using our final approach **(Variable Depth Binary Decision Tree approach)**. We did this by setting a `maxdepth` parameter such that if a node is at the maximum depth, we return its most frequent vote and do not explore further. Thus for a given max depth only nodes with depths<=maxdepth will be explored. We varied the maximum depth from 1 to 55. As we varied the max depth the number of nodes in the decision tree also varied. The plots are as follows:

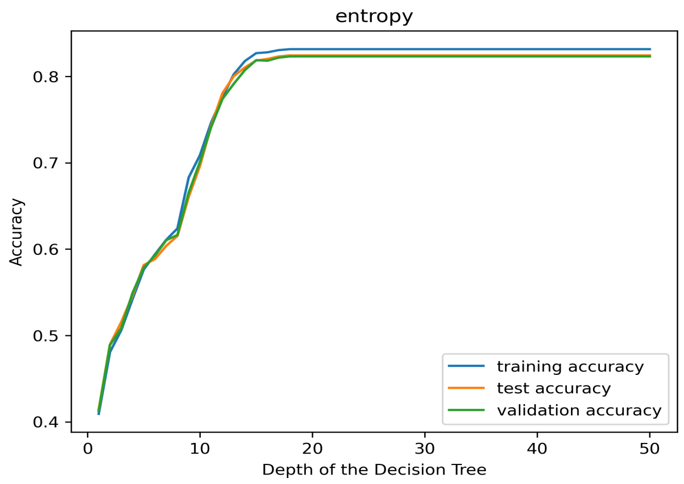
Gini as impurity measure:





Entropy as impurity measure:





Comments: As we can see, the test accuracy more or less stabilises around a depth of 20 in both cases so the **best depth range for our tree will be 19-20.** Lower depths will give lower test as well as training accuracy and also underfit the training data and higher depths may lead to overfitting in the future without any increase in the test accuracy as it becomes stable at higher depths. As the number of nodes increases, the test as well as training accuracies increase which is a strong indicator that for our dataset, a deeper tree with more nodes is not overfitting and larger number of nodes are needed because of the large number of classes to classify into. This can also mean that the number of nodes at very large depths are less and only these nodes are likely to get pruned with a potential slight increase in test accuracy.

Step 4: Pruning

We perform **Reduced Error Pruning**, for which we divide the data further into **80:20** for train and validation respectively.

- First we construct the decision tree with a given height (the best height obtained earlier). Pruning means removing all the children of a particular node, i.e. making a node a leaf node.
- At each step, for every node still in the tree, we get the accuracy on the validation set if that node was to be pruned next.
- Out of all the nodes we choose the one which gives the maximum validation accuracy when pruned.
- We continue this as long as the validation accuracy increases.

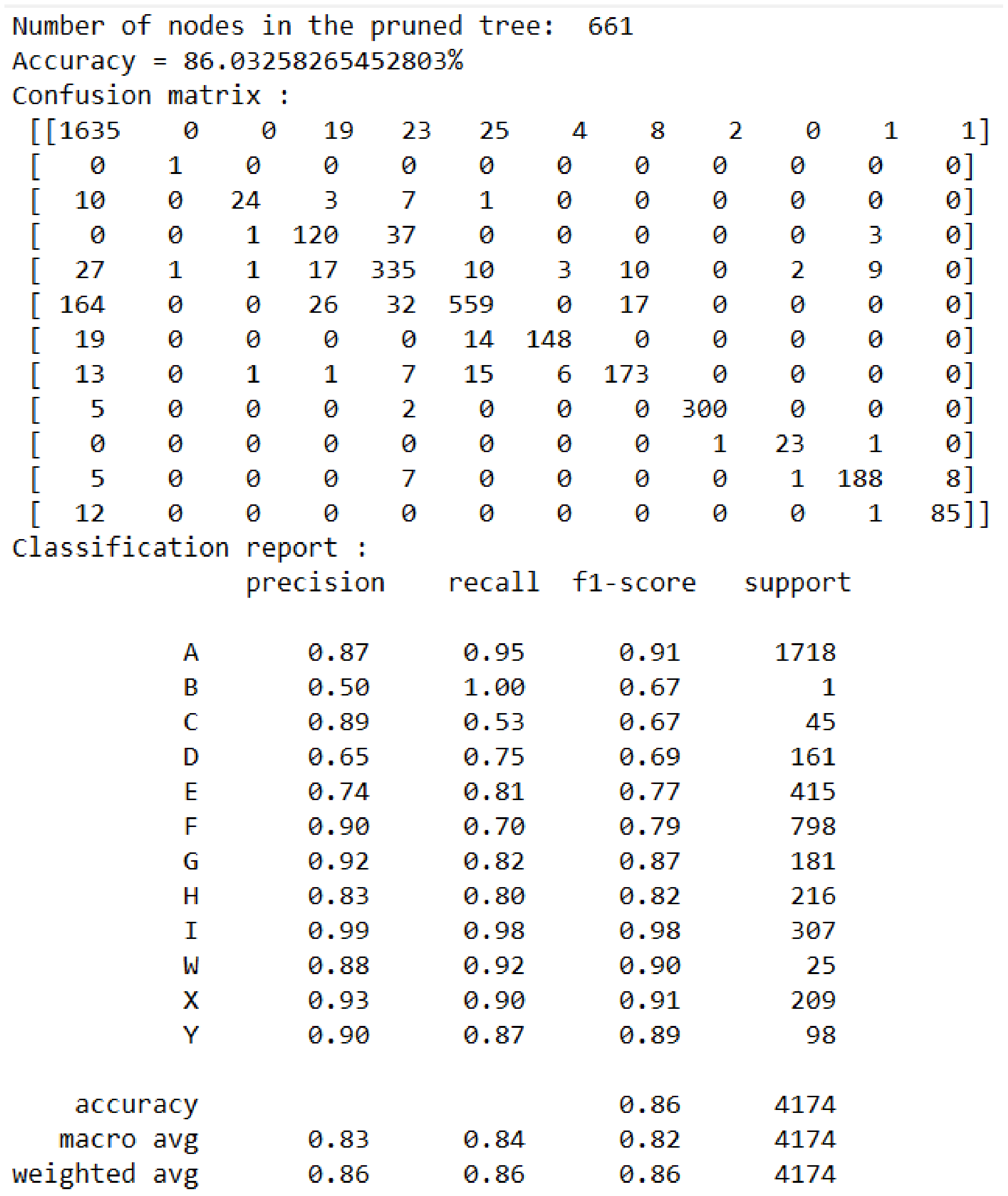
Additionally, we have stopping_rounds, to give some control over pruning to the data analyst. Our pruning algorithm prunes until there is no further increase in validation accuracy. Then it uses the number of stopping rounds provided by the user to continue pruning for that many number of rounds (i.e. if stopping_rounds = 3, then there will be three more rounds of pruning irrespective of whether or not validation accuracy increases). Then we check whether pruning had caused an increase in validation accuracy in the last round, if yes then there is scope for further improvement if we continue pruning (because slope is positive), and if no then we stop pruning.

Gini as impurity measure:

Results of Pruning:

Initial method (Constant Depth Binary Decision Tree approach):
Test Accuracy before pruning: 78.7 %
Test Accuracy after pruning = 79.15%
Number of nodes in the pruned tree: 315

Final method (Variable Depth Binary Decision Tree approach):
Test Accuracy obtained on the test set **before pruning** the tree is: **85.482%**
Test Accuracy obtained on the test set **after pruning** the tree is: **86.033%**
Number of nodes pruned: **977- 661= 316**
Number of nodes in the pruned tree = **661**

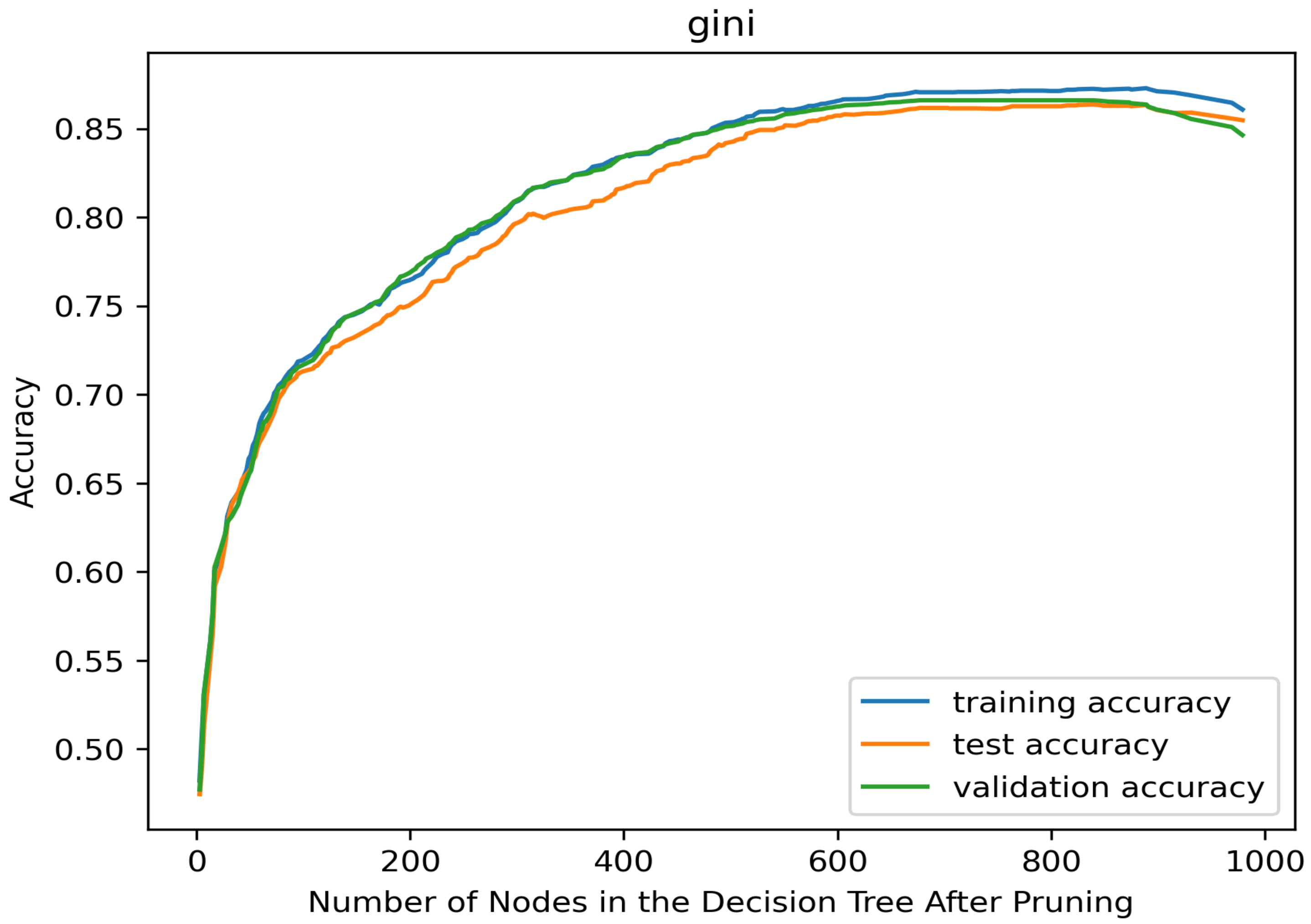


```
Number of nodes in the pruned tree:  661
Accuracy = 86.03258265452803%
Confusion matrix :
 [[1635    0    0   19   23   25    4    8    2    0    1    1]
 [   0    1    0    0    0    0    0    0    0    0    0    0]
 [  10    0   24    3    7    1    0    0    0    0    0    0]
 [   0    0    1  120   37    0    0    0    0    0    3    0]
 [  27    1    1   17  335   10    3   10    0    2    9    0]
 [ 164    0    0   26   32  559    0   17    0    0    0    0]
 [  19    0    0    0    0   14  148    0    0    0    0    0]
 [  13    0    1    1    7   15    6  173    0    0    0    0]
 [   5    0    0    0    2    0    0    0  300    0    0    0]
 [   0    0    0    0    0    0    0    0    1   23    1    0]
 [   5    0    0    0    7    0    0    0    0    1  188    8]
 [  12    0    0    0    0    0    0    0    0    0    1   85]]
Classification report :
               precision    recall  f1-score   support

           A       0.87      0.95      0.91      1718
           B       0.50      1.00      0.67         1
           C       0.89      0.53      0.67        45
           D       0.65      0.75      0.69       161
           E       0.74      0.81      0.77       415
           F       0.90      0.70      0.79       798
           G       0.92      0.82      0.87       181
           H       0.83      0.80      0.82       216
           I       0.99      0.98      0.98       307
           W       0.88      0.92      0.90        25
           X       0.93      0.90      0.91       209
           Y       0.90      0.87      0.89        98

    accuracy                           0.86      4174
   macro avg       0.83      0.84      0.82      4174
weighted avg       0.86      0.86      0.86      4174
```

To obtain a statistical view of pruning vs accuracy, we ran a separate pruning algorithm which continues to prune the tree till only one node is left to see if our pruning algorithm stops pruning at an ideal depth.



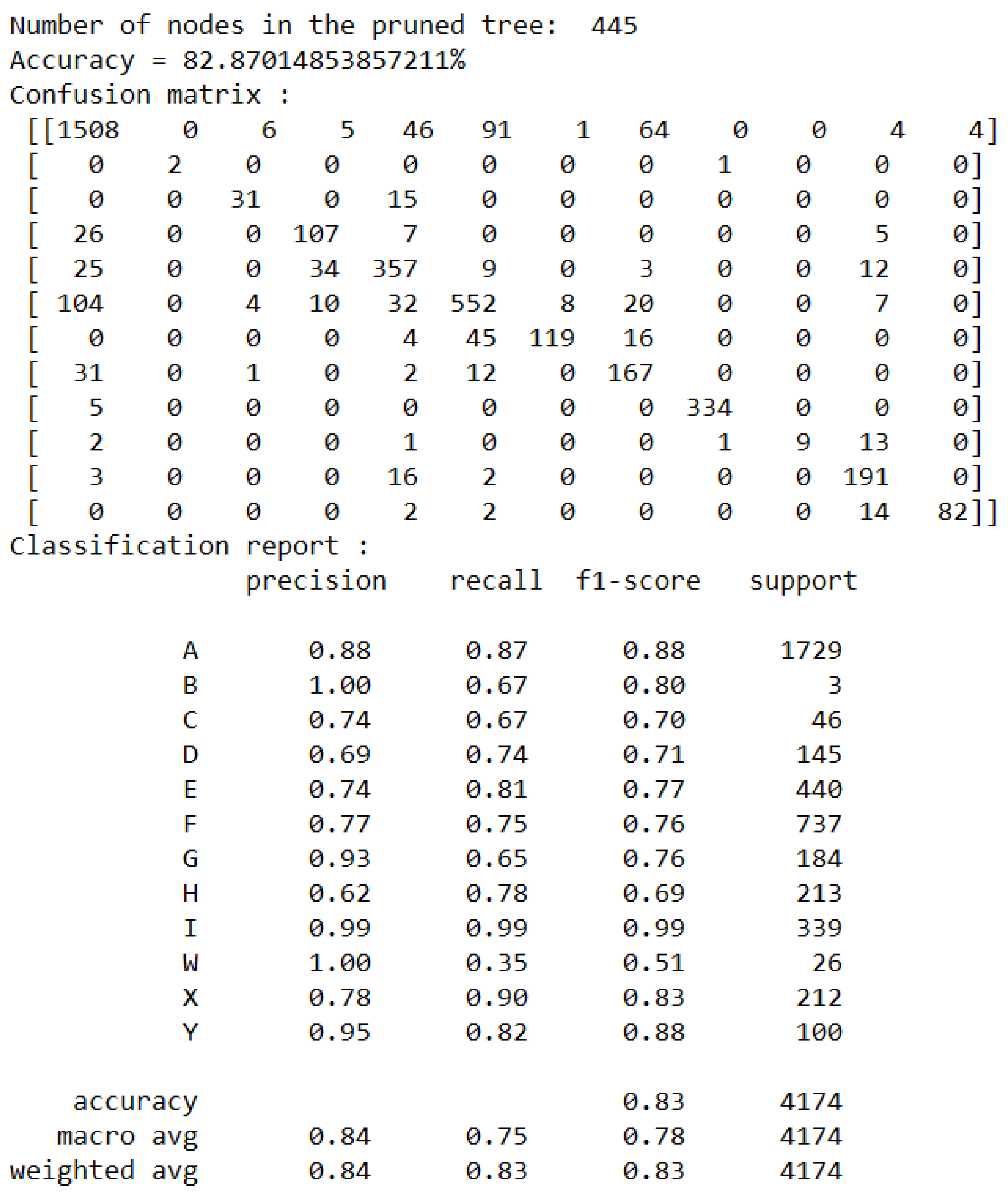
Based on this, our pruning algorithm should stop pruning around 650 nodes as the accuracies fall significantly before that. This matches with the result given by the algorithm as well. There is not much increase in accuracy after pruning because as we had predicted, for our dataset overfitting was not a very likely scenario. The slight increase in accuracy after pruning can be due to fewer misclassifications due to class A (at very large depths) as can be seen in the confusion matrix.

Entropy as impurity measure:

Results of Pruning:

Initial method (Constant Depth Binary Decision Tree approach):
Number of nodes in the pruned tree: 351
Test Accuracy before pruning = 79.72544321993293%
Test Accuracy after pruning = 78.72544321993293%

Final method (Variable Depth Binary Decision Tree approach):
Test Accuracy obtained on the test set **before pruning** the tree is: **82.439%**
Test Accuracy obtained on the test set **after pruning** the tree is: **82.870%**
Number of nodes pruned: **723-445 = 278**
Number of nodes in the pruned tree = **445**

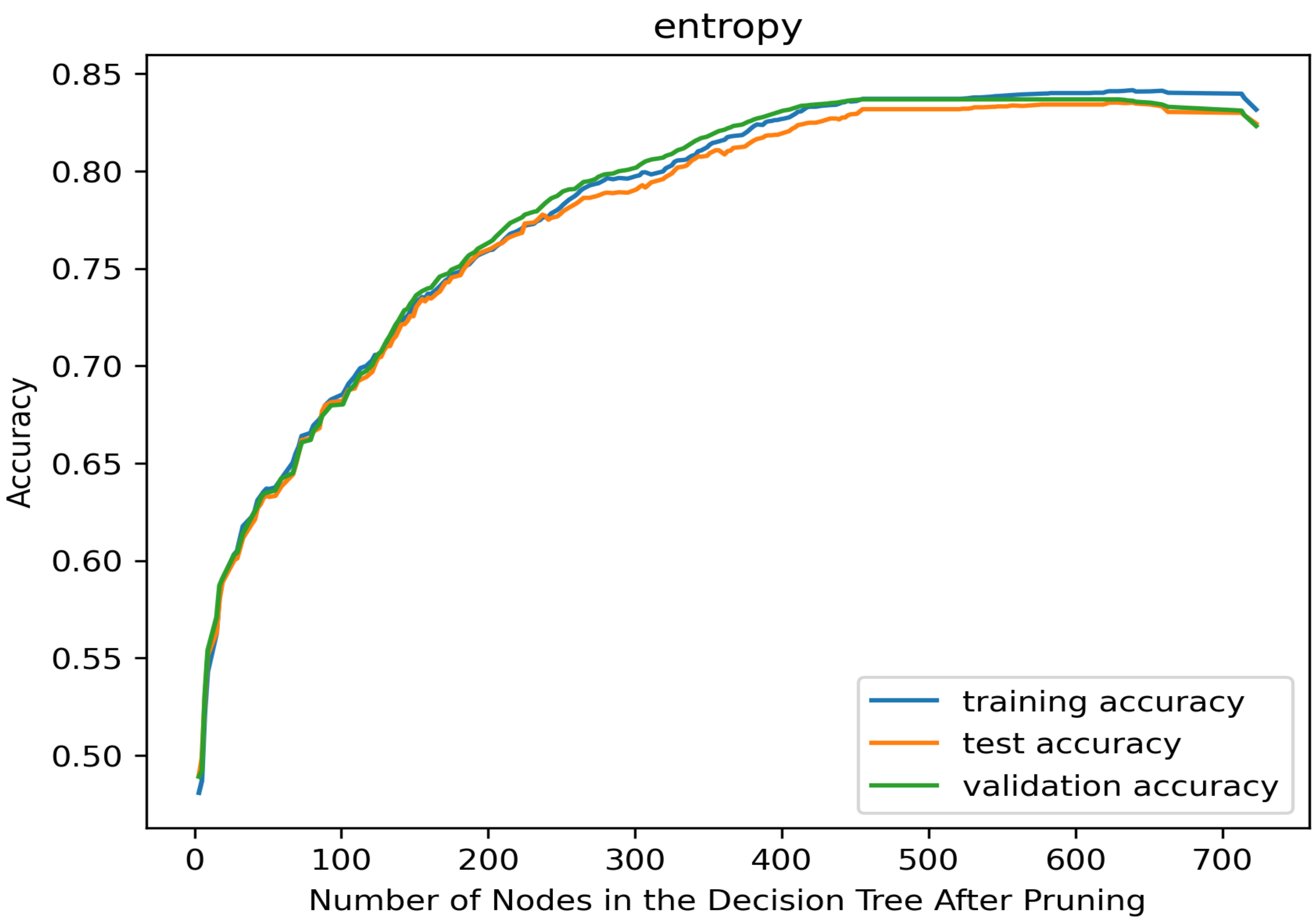


```
Number of nodes in the pruned tree:  445
Accuracy = 82.87014853857211%
Confusion matrix :
 [[1508    0    6    5   46   91    1   64    0    0    4    4]
 [   0    2    0    0    0    0    0    0    1    0    0    0]
 [   0    0   31    0   15    0    0    0    0    0    0    0]
 [  26    0    0  107    7    0    0    0    0    0    5    0]
 [  25    0    0   34  357    9    0    3    0    0   12    0]
 [ 104    0    4   10   32  552    8   20    0    0    7    0]
 [   0    0    0    0    4   45  119   16    0    0    0    0]
 [  31    0    1    0    2   12    0  167    0    0    0    0]
 [   5    0    0    0    0    0    0    0  334    0    0    0]
 [   2    0    0    0    1    0    0    0    1    9   13    0]
 [   3    0    0    0   16    2    0    0    0    0  191    0]
 [   0    0    0    0    2    2    0    0    0    0   14   82]]
Classification report :
               precision    recall  f1-score   support

           A       0.88      0.87      0.88      1729
           B       1.00      0.67      0.80         3
           C       0.74      0.67      0.70        46
           D       0.69      0.74      0.71       145
           E       0.74      0.81      0.77       440
           F       0.77      0.75      0.76       737
           G       0.93      0.65      0.76       184
           H       0.62      0.78      0.69       213
           I       0.99      0.99      0.99       339
           W       1.00      0.35      0.51        26
           X       0.78      0.90      0.83       212
           Y       0.95      0.82      0.88       100

    accuracy                           0.83      4174
   macro avg       0.84      0.75      0.78      4174
weighted avg       0.84      0.83      0.83      4174
```

To obtain a statistical view of pruning vs accuracy, we ran a separate pruning algorithm which continues to prune the tree till only one node is left to see if our pruning algorithm stops pruning at an ideal depth.



Based on this, our pruning algorithm should stop pruning around 450 nodes as the accuracies fall significantly before that. This matches with the result given by the algorithm as well. There is not much increase in accuracy after pruning because as we had predicted, for our dataset overfitting was not a very likely scenario. The slight increase in accuracy after pruning can be due to fewer misclassifications due to class A as can be seen in the confusion matrix.

Summary

- We as data analysts always want the best from the models we built, that is why when we were not satisfied with our **Constant Depth Binary Decision Tree approach** we switched over to **Variable Depth Binary Decision Tree approach.** In our first approach we were following an approach similar to the ID3 algorithm whereas in our second approach we thought that why not consider an attribute again for making decisions as we grow the tree, as that would mean being able to incorporate multiple splits and hence thresholds based on the same attribute at different levels.

- And we were right, we found a tremendous increase in accuracy as we started reusing attributes. A short summary of the comparison of both the approaches is given below.

	Initial Approach: Constant Depth Binary Decision Tree approach		**Final Approach: Variable Depth Binary Decision Tree approach**	
Impurity Measure	Gini	Entropy	Gini	Entropy
Average Test Accuracy before pruning	75.685%	76.730%	**80.10%**	**79.736%**
Maximum Test Accuracy before pruning	78.701%	79.660%	**85.481%**	**82.439%**
Test Accuracy after pruning using the best tree.	79.156%	78.725%	**86.033%**	**82.870%**

- Then we moved on to performing **depth**, **nodal** and **pruning** analysis on the best trees obtained from approach two which is **Variable Depth Binary Decision Tree approach.** In order to prune the tree we used reduced error pruning with the help of a validation set. Here's a short summary of the results obtained.

Final Approach: Variable Depth Binary Decision Tree approach

	GINI	**ENTROPY**
Most Optimal Depth w.r.t. accuracy	Around 19-20	Around 19-20
Number of nodes in the tree before pruning	977	723
Number of nodes in the tree after pruning	661	445
Accuracy on the training set before pruning (approx.)	89%	84%
Accuracy on the training set after pruning (approx.)	87%	83%
Accuracy on the test set before pruning	85.481%	82.439%
Accuracy on the test set after pruning	86.033%	82.870%

❖ As you can clearly see from the above analysis that gini proves to be a much better measure of impurity than entropy and that our second approach **Variable Depth Binary Decision Tree approach** has proved to be a winner. Gini gives a better result because it is a measure of how often a randomly chosen element from the set would be incorrectly labeled**.** Since our dataset was highly biased towards certain classes, an impurity measure which takes into account incorrect classifications to these classes should perform better than one which does not. This matches with the result as well where gini outperformed gain as an impurity measure.

Statistical Measures used:

- Entropy : Performed the splitting using gain in entropy as heuristic to determine the best possible split for a given attribute and for determining best attribute

$$Entropy \;=\; -\,\Sigma(p * log2(p)) \;\; where\ p\ is\ the\ proportion\ of\ a\ class\ in\ the\ samples.$$
$$Gain\ in\ entropy \;=\; Entropy(S) \;-\; \Sigma\frac{|Sv|}{|S|}Entropy(Sv)\ where\ Sv\ is\ a\ split$$

- Gini index: Performed the splitting using gini gain as heuristic to determine the best possible split for a given attribute and for determining best attribute

$$Gini \;=\; \Sigma(p * (1 - p)) \;\; where\ p\ is\ the\ proportion\ of\ a\ class\ in\ the\ samples.$$
$$Gini\ Gain \;=\; Gini(S) \;-\; \Sigma\frac{|Sv|}{|S|}Gini(Sv)\ where\ Sv\ is\ a\ split$$

- Validation Set accuracy: We use the accuracy obtained in the validation set to determine when to stop pruning the tree.

Some remarks about the approaches we used and an introduction to a new way of looking at decision trees:

Constant Depth Binary Decision Tree Approach (Initial Approach) :

In this approach, we did not make any reuse of attributes and the continuous valued attributes were split to two regions only(thus a binary decision tree). This approach is similar to the standard ID3 algorithm where an attribute once chosen at a node is never reused of making a decision in it's children and with the discretisation of the attributes is binary (two regions). The depth of the tree was therefore upper bounded by the number of attributes (10) and the number of nodes in the tree was upper bounded by $2^{\text{maximum depth+1}}$ (2^{11}). This approach did not yield good results and thus we moved to a better approach.

Variable Depth Binary Decision Tree Approach (Final Approach) :

In this approach, we made reuse of attributes and the continuous valued attributes were split to two regions only(thus a binary decision tree). The depth of the tree was therefore unbounded and we had to use a parameter of the DecisionTreeClassifier model to bound the depth, but with the discretisation of the attributes being binary (two regions), the number of nodes in the tree was upper bounded by $2^{\text{maximum depth+1}}$. This approach deviates from the ID3 algorithm but provides much better results. There was nearly 15% increase in test accuracy in both gini and entropy as impurity measurement metrics.

New approaches one could try which are more computationally expensive:

We have not tried the following approaches due to lack of time but they do provide a food for thought to our readers.

Constant Depth k-ary Decision Tree Approach :

In this approach, we do not make any reuse of attributes and the continuous valued attributes were split to k regions (thus a k-ary decision tree). This approach is similar to standard ID3 algorithm where an attribute once chosen at a node is never reused of making a decision in it's children and with the discretisation of the attributes is k-ary (k regions). The depth of the tree was therefore upper bounded by the number of attributes (10) and the number of nodes in the tree was upper bounded by $k^{\text{maximum depth+1}}$ (k^{11}).

Variable Depth k-ary Decision Tree Approach :

In this approach, we make reuse of attributes and the continuous valued attributes were split to k regions (thus a k-ary decision tree). The depth of the tree will be unbounded and we would have to use a parameter of the DecisionTreeClassifier model to bound the depth, but with the discretisation of the attributes is k-ary (k regions), the number of nodes in the tree was upper bounded by $k^{\text{maximum depth+1}}$.