COL341 Assignment 3

Kushagra Rode

April 2023

1 Binary Classification

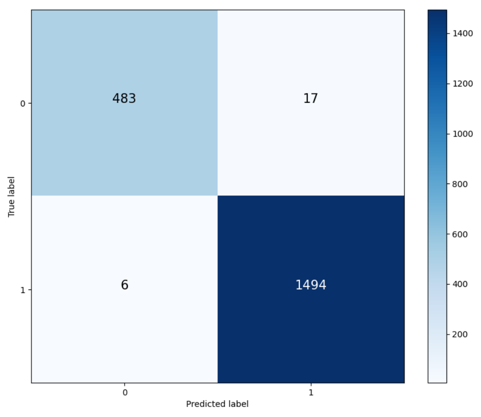
1.1 Decision Tree from Scratch

The analysis is based on the stopping criterion specified in the pdf, i.e. Max Depth (set as 10) and Minimum samples split (set as 7).

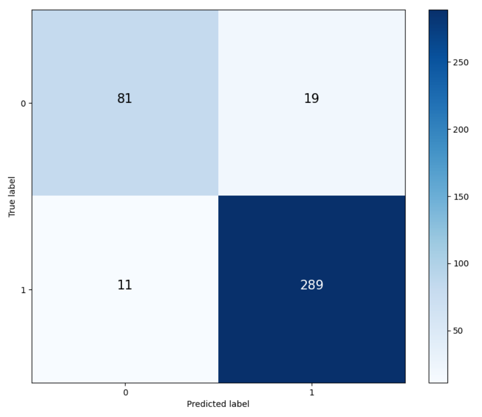
1. For Gini Index:-
   Time Taken for training - 285.00575375556946 seconds

   Training Accuracy: 0.9885
   Training Precision: 0.98773006134969 32
   Training Recall: 0.966

   Validation Accuracy: 0.925
   Validation Precision: 0.8804347826086957
   Validation Recall: 0.81
   Following are the confusion matrices for training and validation data.



(a) Training Data



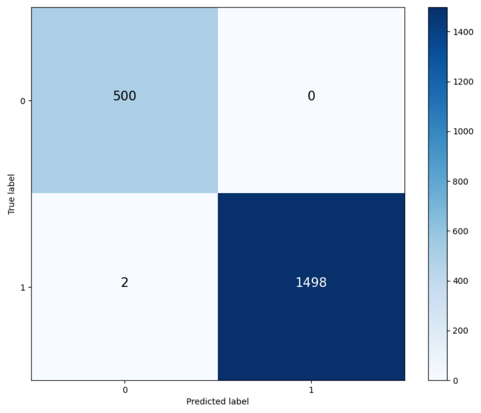
(b) Validation Data

2. For Information Gain:-
   Time Taken for training - 1035.5735049247742 seconds

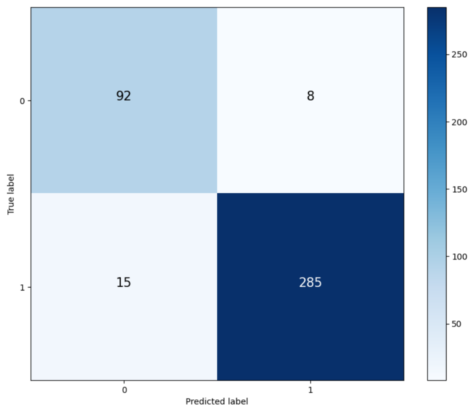
   Training Accuracy: 0.999
   Training Precision: 0.9960159362549801
   Training Recall: 1.0

   Validation Accuracy: 0.9425
   Validation Precision: 0.8598130841121495
   Validation Recall: 0.92

   Following are the confusion matrices for training and validation data.



(c) Training Data



(d) Validation Data

   As you can see, both models perform fairly well on the training and validation data. We can see that though Information Gain takes a longer time to train the decision tree, but we achieve better validation as well as training accuracies, along with the precision and recall scores are much better in the information gain case. The longer time can be attributed to the calculation of the entropy which involves logarithmic operations.

1.2 Decision Tree sklearn

1. Using the gini criterion
   Time Taken - 1.2001984119415283 seconds

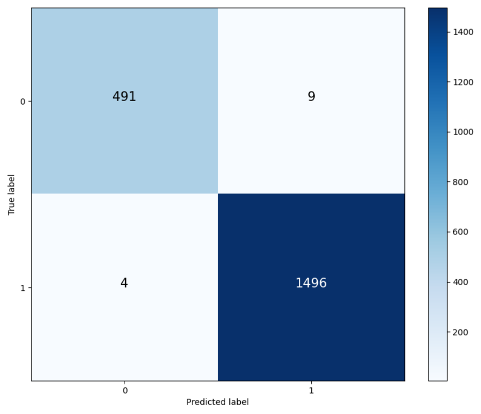
   Training Accuracy: 0.993
   Training Precision: 0.9919028340080972
   Training Recall: 0.98

   Validation Accuracy: 0.9525

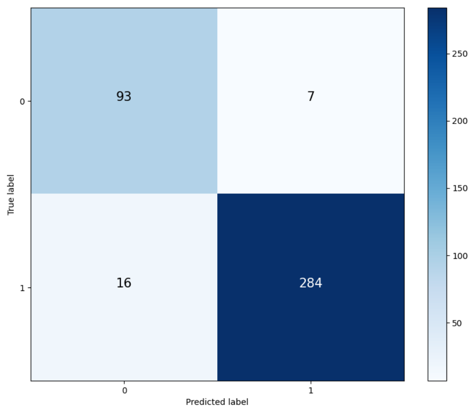
Validation Precision: 0.9090909090909091
Validation Recall: 0.9

Comparing it with my own implementation of the decision tree, we can see that the sklearn model fairs pretty well. It outperforms my own implementation in all aspects, from validation accuracy to precision and recall values. Another significant improvement is in terms of its running time. It takes very less time compared to my model for gini index.

Confusion Matrices are as follows :



(e) Training Data

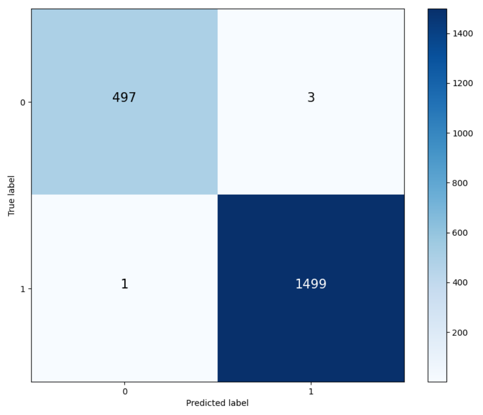


(f) Validation Data

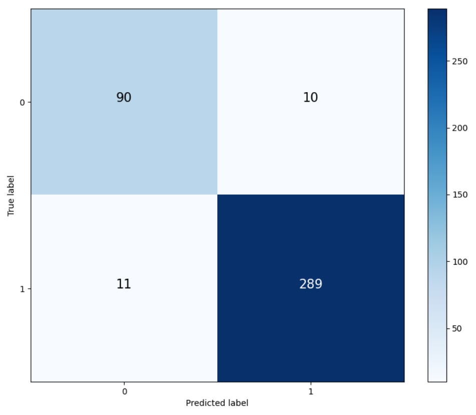
2. Using entropy as criterion
   Time Taken - 2.4041950702667236 seconds

   Training Accuracy: 0.998
   Training Precision: 0.9979919678714859
   Training Recall: 0.994

   Validation Accuracy: 0.9475
   Validation Precision: 0.8910891089108911
   Validation Recall: 0.9
   Although we used information gain in my implementation, but we can compare it with entropy criterion used in the sklearn. Similar to gini, sklearn produces better validation and training accuracies. The key factor here is the time taken, it is very less as compared to my implementation which took nearly 7 minutes to run Confusion Matrices are as follows :



(g) Training Data



(h) Validation Data

Figure 1: Validation Data

Here are graphs for comparing the running time and accuracies of parts 0.1 and 0.2.

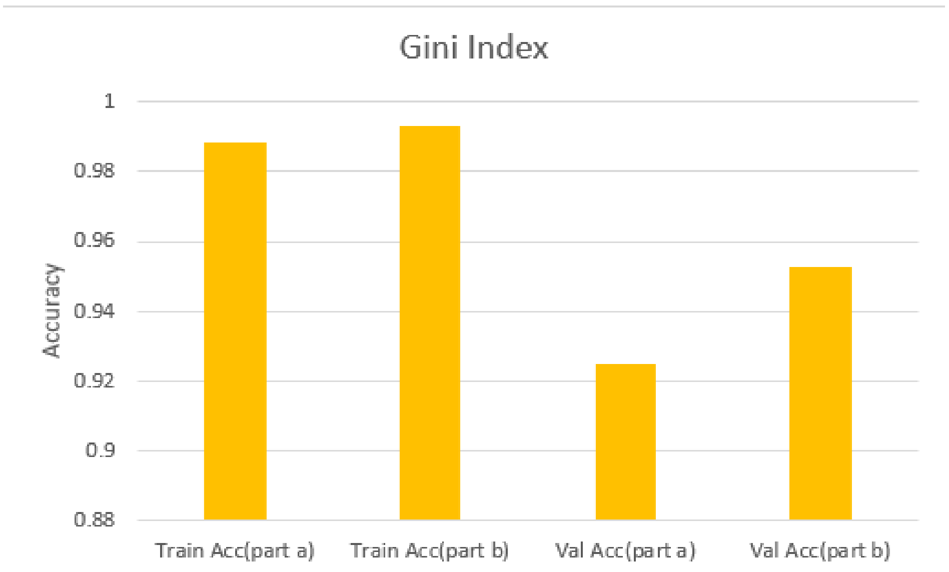


Figure 2: Training Data

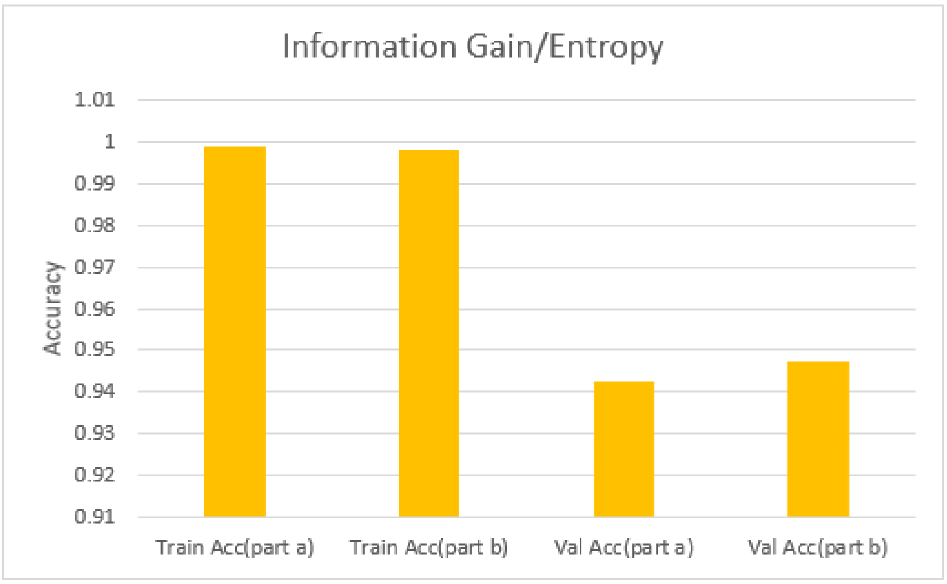


Figure 3: Validation Data

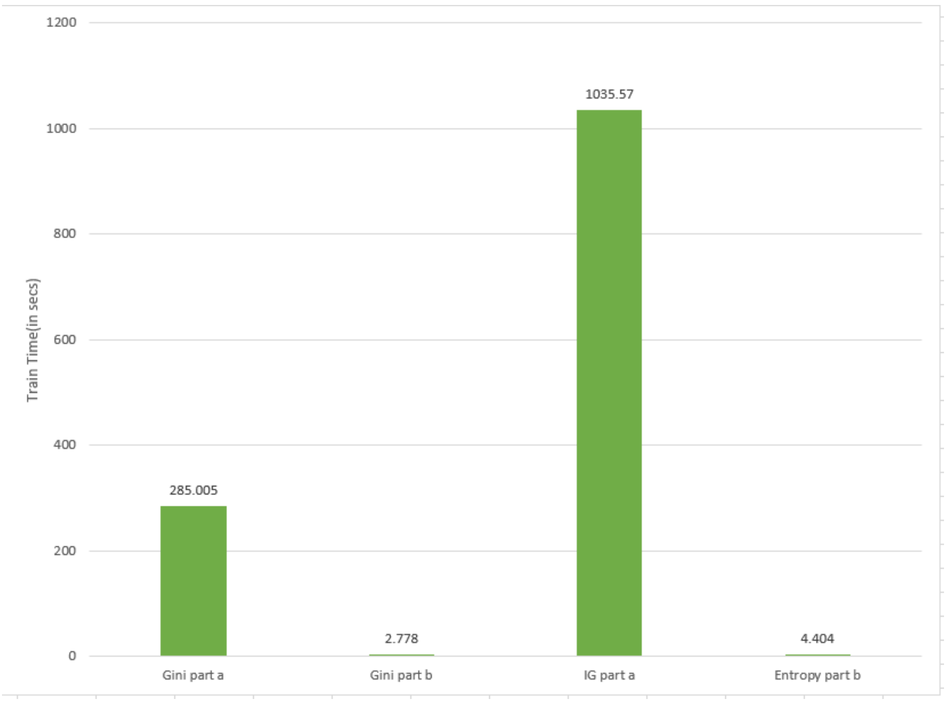


Figure 4: Training Time

1.3 Decision tree Grid Search and Visualisation

Following is the decision tree obtained on feature selection through sklearn. This has the following parameters, max-depth = 10 and minimum samples split - 7. I have made use of matplotlib and plot tree function of sklearn to create the tree.

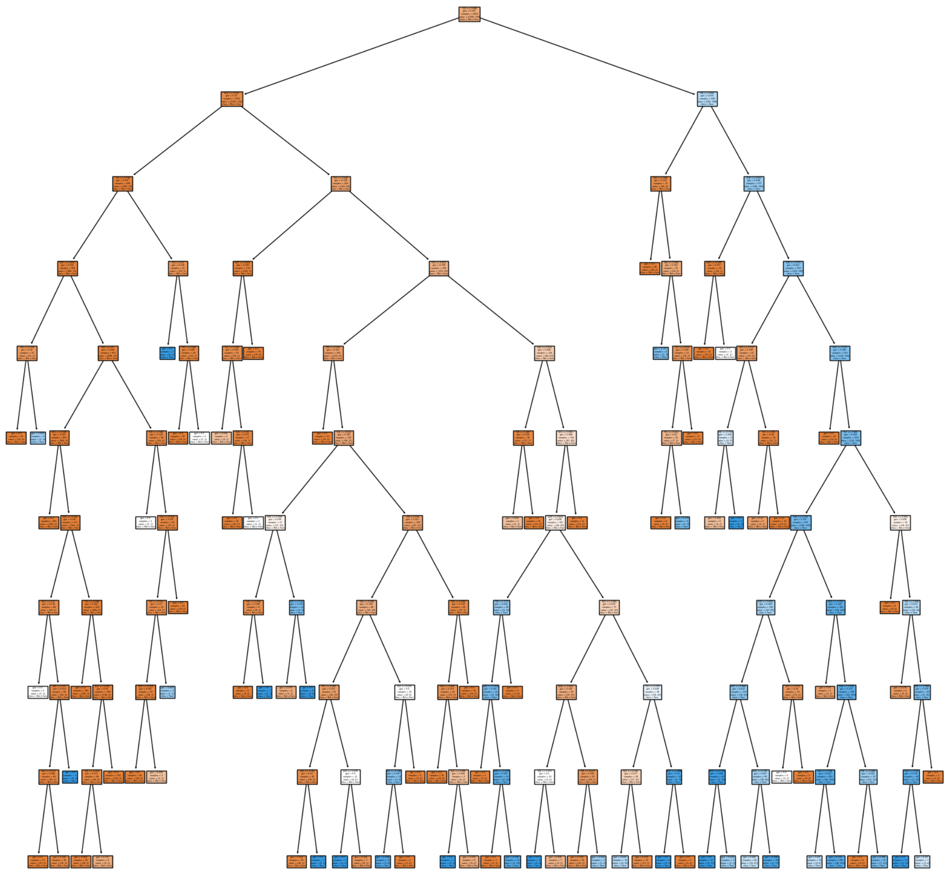
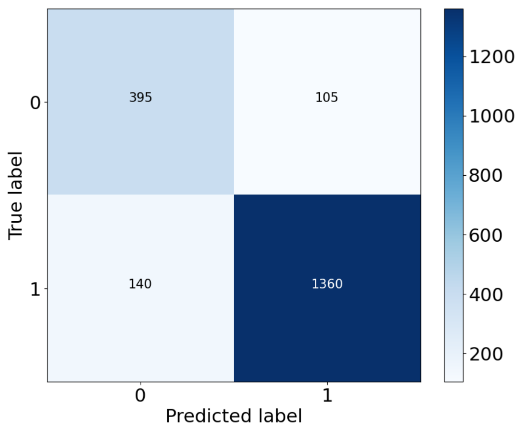


Figure 5: Decision Tree

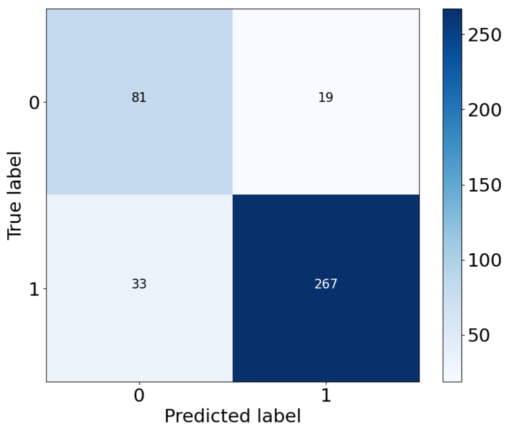
Now on running the grid search, following were the best parameters.
1. criterion: 'entropy'
2. max_depth: 5
3. min_samples_split: 2

Training Accuracy: 0.8775
Validation Accuracy: 0.8775

Confusion Matrices for the decision tree trained on the best set of parameters are as follows:-



(a) Training Data



(b) Validation Data

Observations -

Choosing a subset of features results in decrease of both training and validation accuracy as compared to the parts 0.1 and 0.2 stated above. Though, reduction of features leads to improvement in the training time and hence has it's advantage of running faster than the parts 0.1 and 0.2.
Coming to the grid search part, it does give us the best parameters although the accuracy is on the lower side as mentioned above. This is basically due to the feature selection taking place which definitely will result in absence of some useful training thus leading to lower accuracy.

1.4 Decision Tree Post Pruning with Cost Complexity Pruning

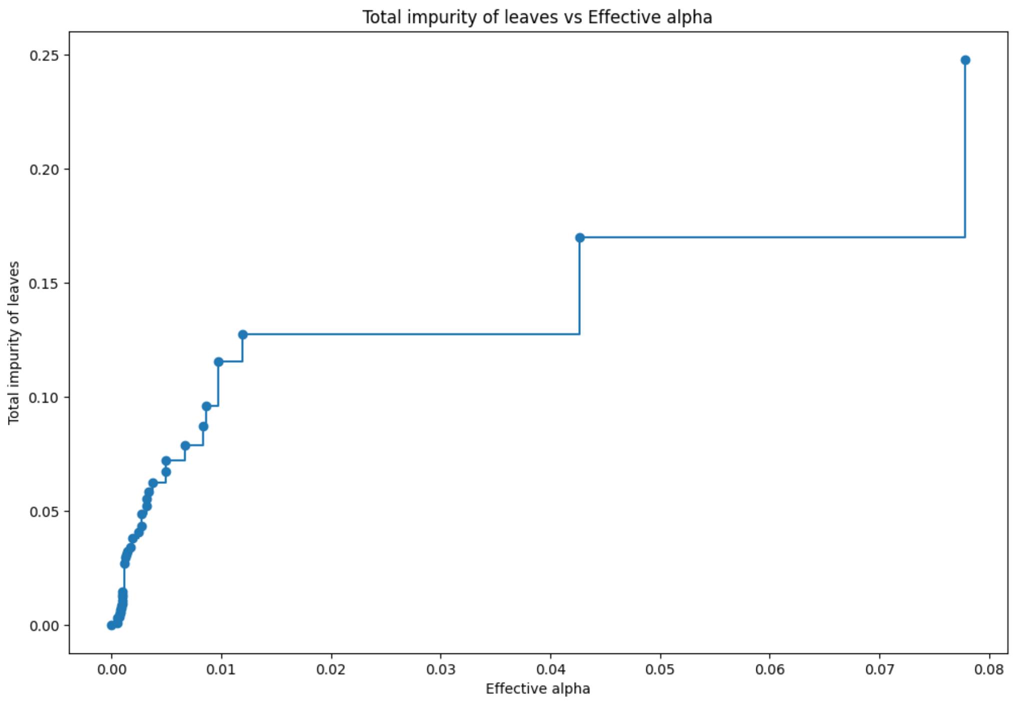


Figure 6: Total Impurity vs Effective alpha

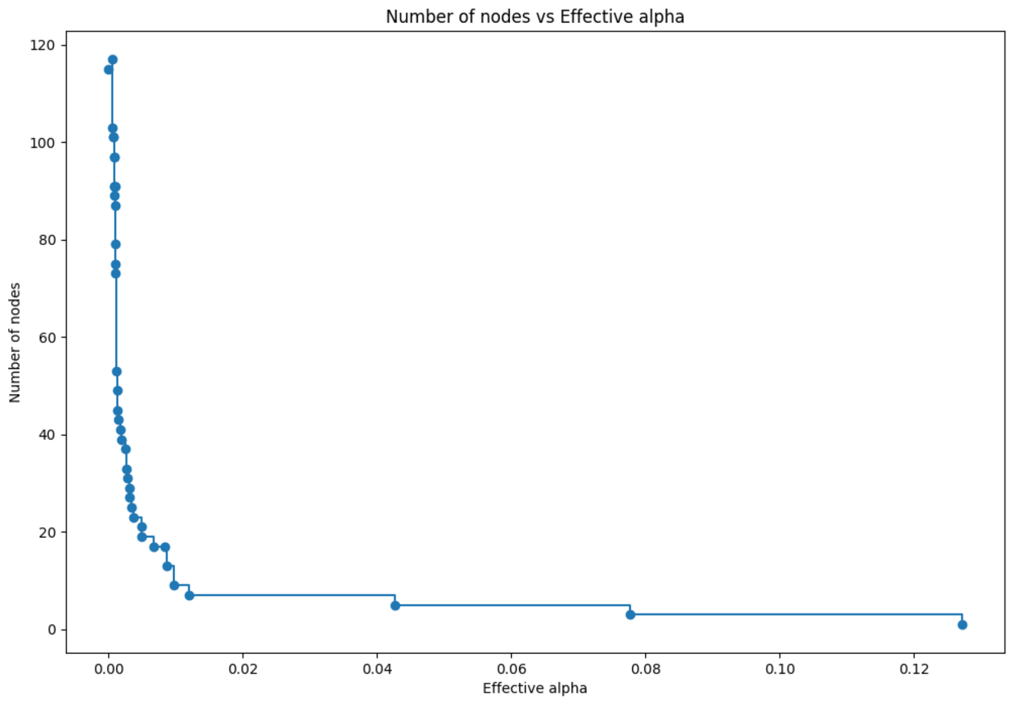


Figure 7: Number of nodes vs effective alpha

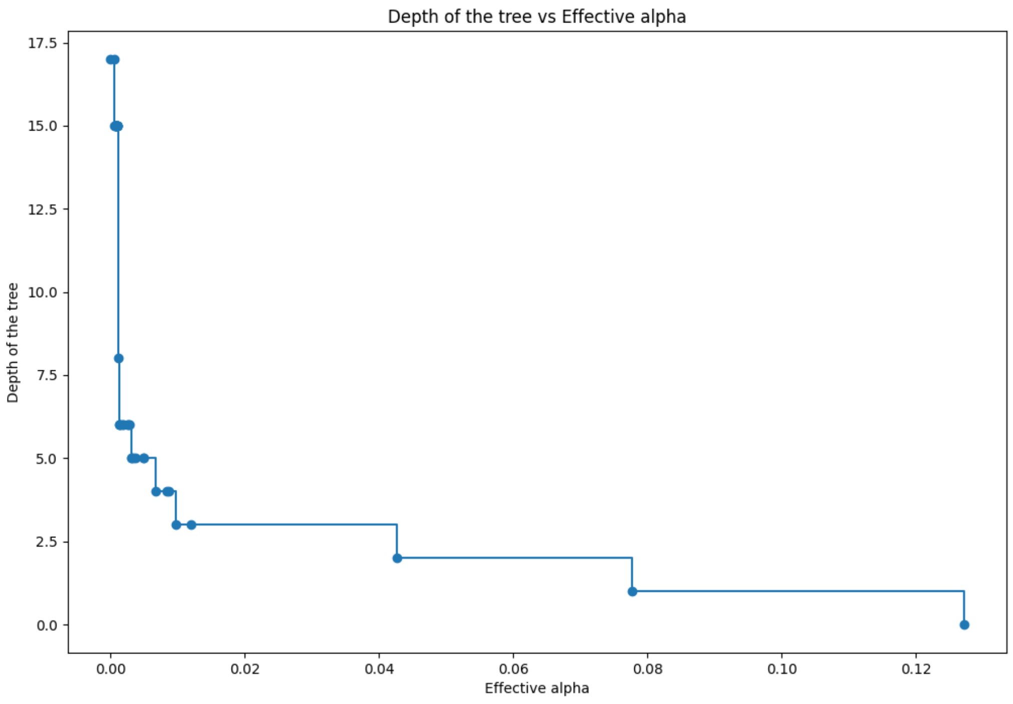


Figure 8: Depth vs effective alpha

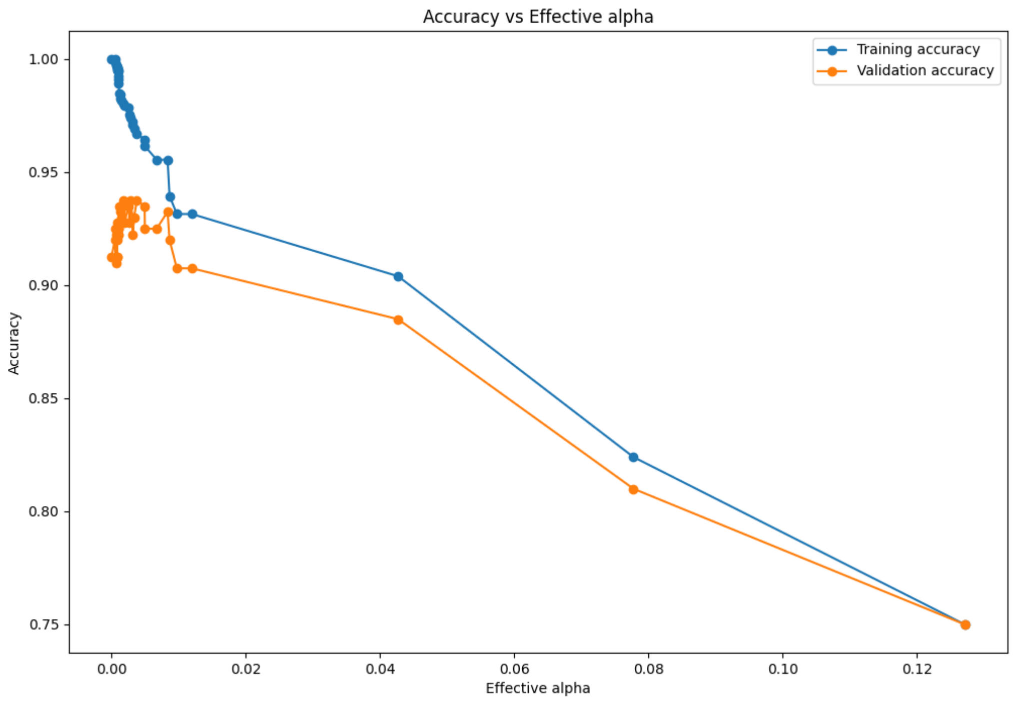


Figure 9: Training and Validation accuracies vs effective alpha

Observations -

As the effective alpha increases the value of total impurity in the leaves increases. This means that increasing alpha leads to non-homogeneity increasing at the leaf nodes. This is coupled with the decrease in the number of nodes as alpha increases, thus leading to more pruning of the tree. Both of those effects are again related to the depth of the tree. Higher alpha prunes the tree more thus leading to a decrease in the depth of the tree. For all three plots described, the values change rather slowly when the alpha is small but the change is drastic when the alpha reaches a larger value.
Coming to the Accuracy plot, we can see that the training accuracy decreases as we increase the alpha. This is due to an increase in the pruning of the tree. The validation accuracy, however, initially increases with alpha, reaches a maximum value which is the best-performing tree for us and then it decreases with alpha due to unreasonable pruning of the tree.

We use the validation data for getting the best tree. Following is the tree

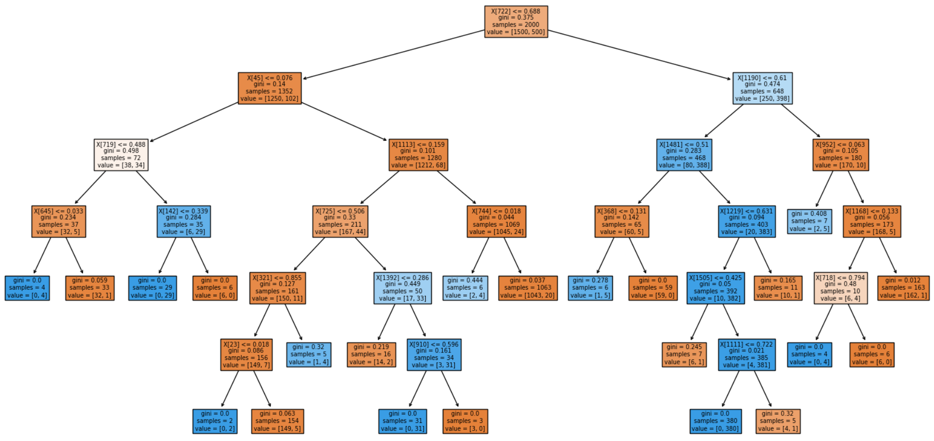
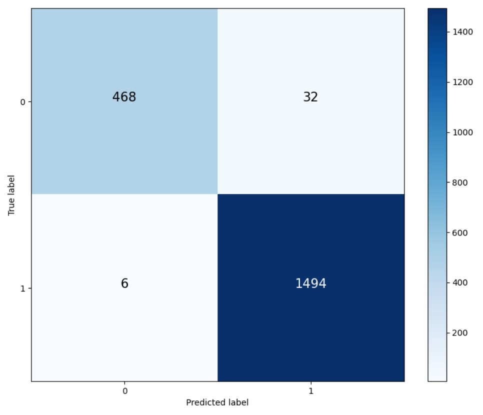


Figure 10: Best-performing tree

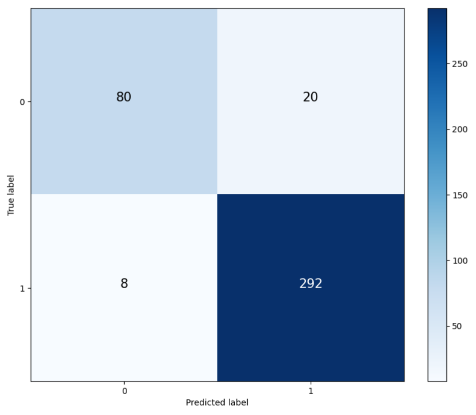
The corresponding training and validation accuracies are as follows :-
Training Accuracy: 0.981
Validation Accuracy: 0.93

The alpha corresponding to these accuracies is - 0.0025041388472594184

Confusion Matrices for the best tree are as follows :



(a) Training Data



(b) Validation Data

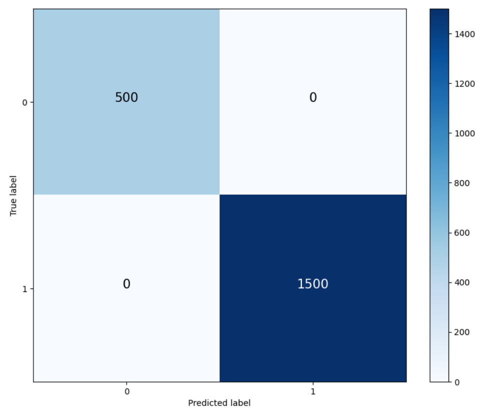
1.5 Random Forest

With the default parameters, following are the observations obtained.

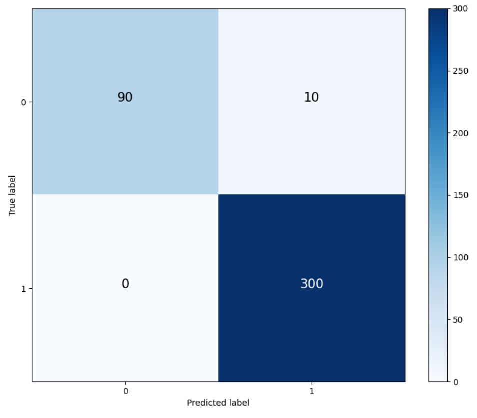
Default hyperparameters:

Training accuracy: 1.0
Training precision: 1.0
Training recall: 1.0

Validation accuracy: 0.9725
Validation precision: 1.0
Validation recall: 0.89
The confusion matrices for the default hyperparameters are as follows:-



(c) Training Data



(d) Validation Data

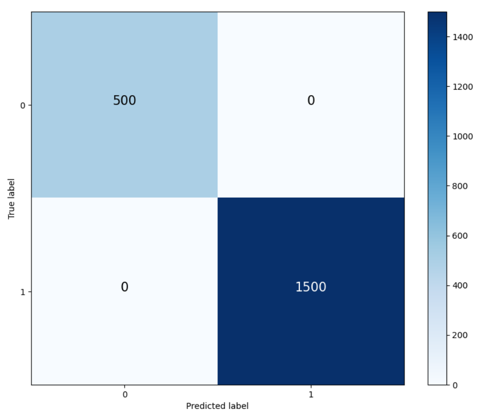
For Best hyperparameters:

Training accuracy: 1.0
Training precision: 1.0
Training recall: 1.0

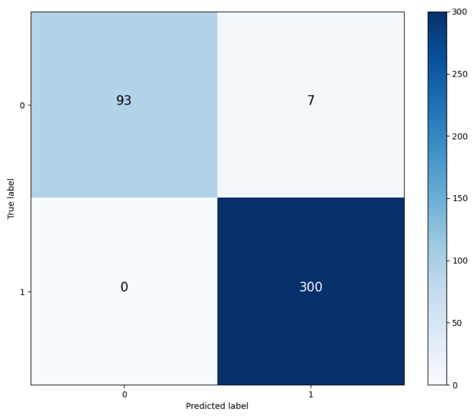
Validation accuracy: 0.9825
Validation precision: 1.0
Validation recall: 0.93

The best hyperparameters are as follows:-
1. criterion: 'entropy'
2. max_depth: 10
3. min_samples_split: 5
4. n_estimators : 100

The confusion matrices for the default hyperparameters are as follows:-



(e) Training Data



(f) Validation Data

1.6 Gradient Boosted Trees and XGBoost

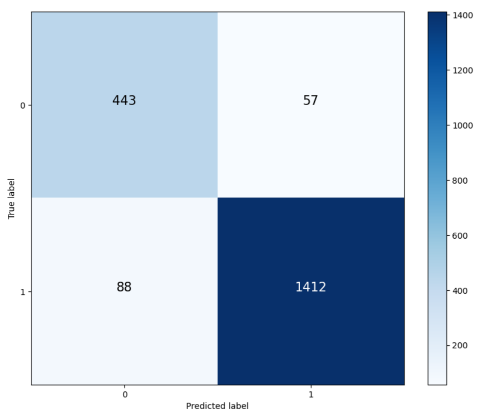
1.6.1 For Gradient Boosted Trees:-

Gradient boosting was taking a lot of time to run when trained with all features. I tried on google-colab as well as my own laptop, hence I have selected 10 best features for purpose of analysis
The best hyperparameters are as follows:-
1. max_depth: 6
2. subsample: 0.6
3. n_estimators : 20

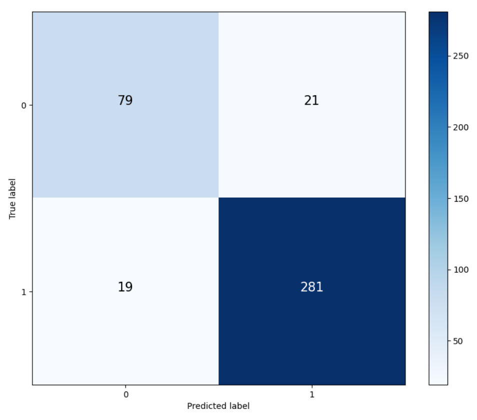
Training Time: 241.1902494430542 seconds

Training accuracy: 0.9275
Training precision: 0.8342749529190208
Training recall: 0.886

Validation accuracy: 0.9
Validation precision: 0.8061224489795918
Validation recall: 0.79
The confusion matrices are as follows:-



(g) Training Data



(h) Validation Data

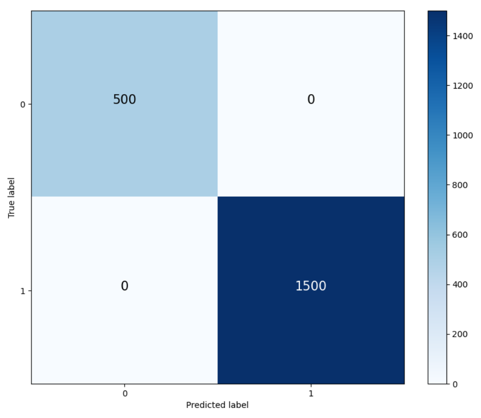
1.6.2 Using XGBoost:-

XGBoost did ran in about 30 minutes on my system when trained with all 3072 features and hence the below analysis is based on that.
The best hyperparameters are as follows:-
1. max_depth: 6
2. subsample: 0.5
3. n_estimators : 40

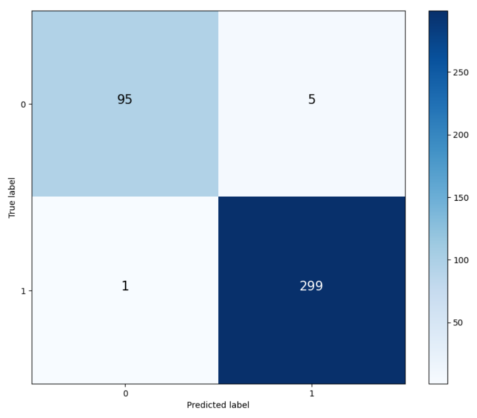
Training Time: 2036.6859481334686 seconds

Training accuracy: 1.0
Training precision: 1.0
Training recall: 1.0

Validation accuracy: 0.985
Validation precision: 0.9895833333333334
Validation recall: 0.95
The confusion matrices are as follows:-



(i) Training Data



(j) Validation Data

2 Multi-Class Classification

2.1 Decision Tree sklearn

The analysis is based on the stopping criterion specified in the pdf, i.e. Max Depth (set as 10) and Minimum samples split (set as 7).

1. For Gini Index:-
   Training Accuracy: 0.969
   Validation Accuracy: 0.75
   Training Time: 3.33 seconds
   Following are the confusion matrices for training and validation data.

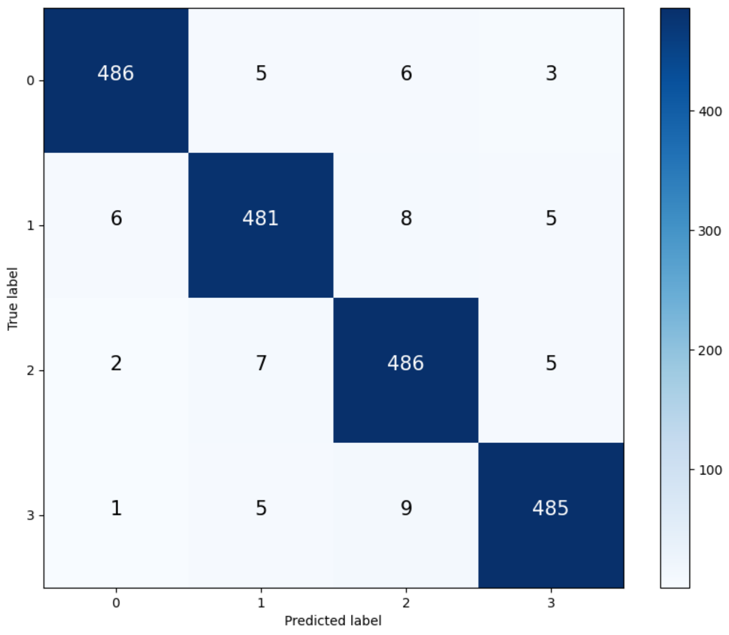


Figure 11: Training Data

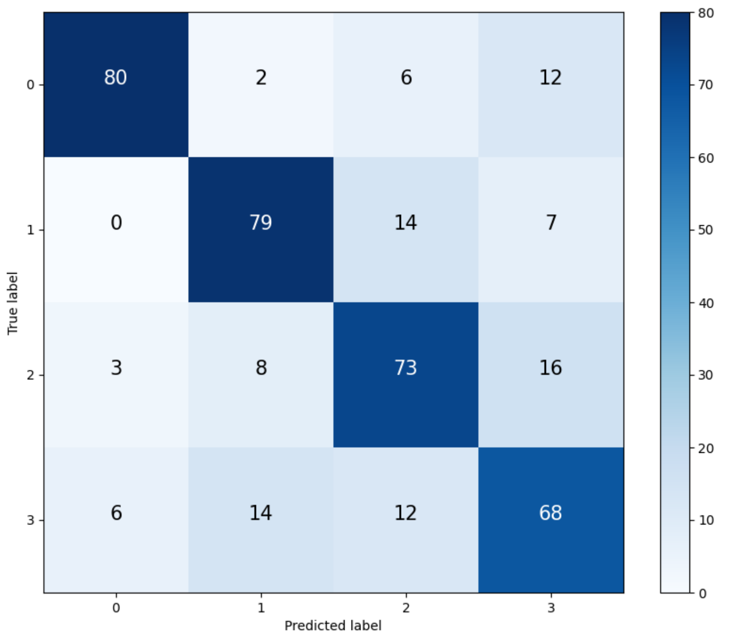
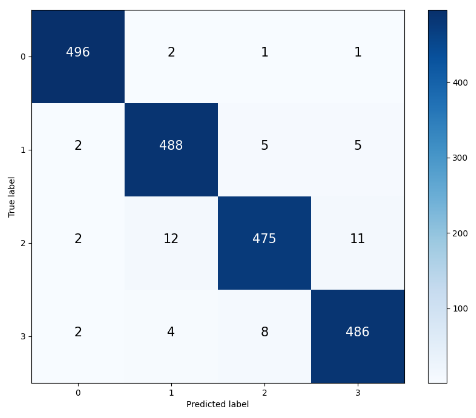


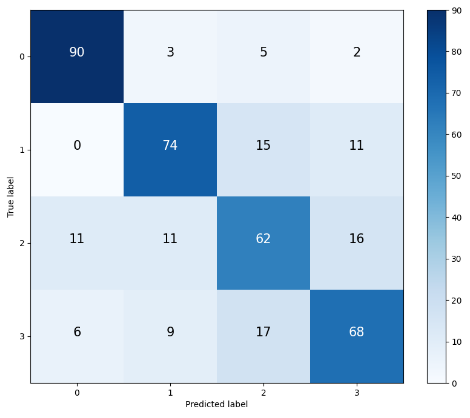
Figure 12: Validation Data

2. For Entropy:-
   Training Accuracy: 0.9725
   Validation Accuracy: 0.735

Total Time Taken: 5.08 seconds
Following are the confusion matrices for training and validation data.



(a) Training Data



(b) Validation Data

2.2 Decision Tree Grid Search and visualization

Following is the decision tree obtained on feature selection through sklearn. This has the following parameters, max-depth = 10 and minimum samples split - 7. I have made use of matplotlib and plot tree function of sklearn to create the tree.

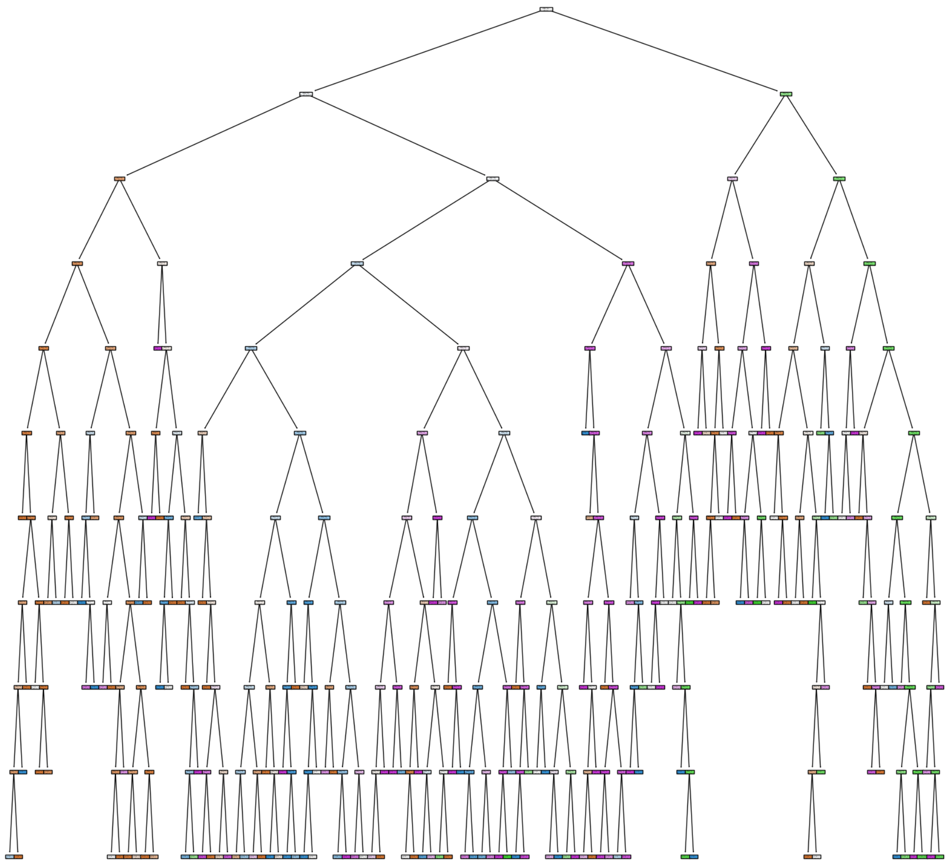
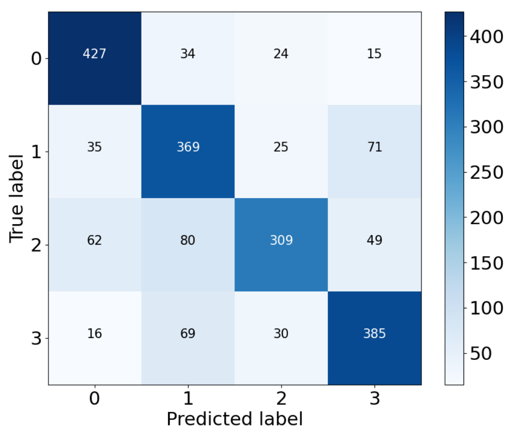


Figure 13: Decision Tree

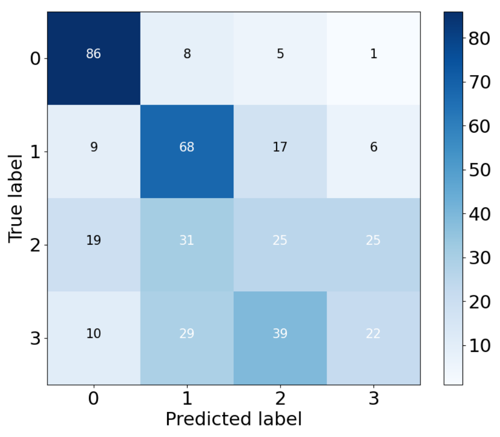
Now on running the grid search, the following were the best parameters.
1. criterion: 'entropy'
2. max depth: 7
3. min samples split: 4

Training Accuracy: 0.745
Validation Accuracy: 0.5025

The training for these hyperparameters was - 2.58 seconds
Confusion Matrices for the decision tree trained on the best set of parameters are as follows:-



(a) Training Data



(b) Validation Data

Well, comparing this with part 1.1 we can clearly see the higher accuracy(both training and validation) in part 1.1. Part 1.2 performs poorly due to a lack of enough features. We have only selected the 10 best features that may not capture the pattern in the data and hence the best we could get even after grid search was pretty low as compared to the model trained on all the features in part 1.1. The time taken to fit the model does become half in part 1.2.

2.3 Decision Tree Post Pruning with Cost Complexity Pruning

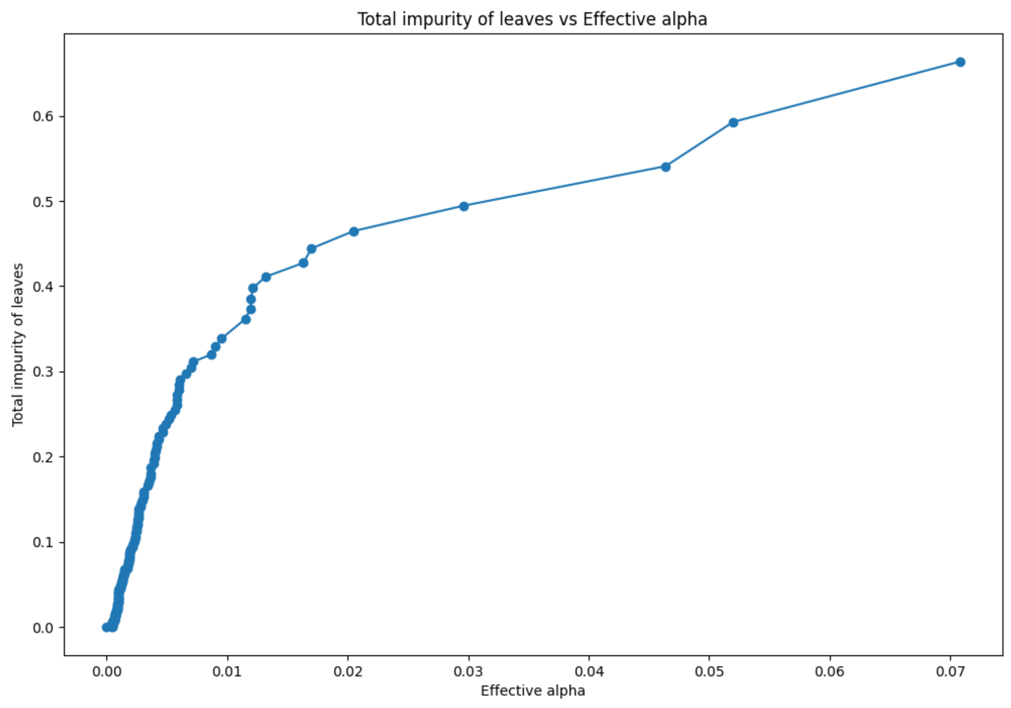


Figure 14: Total Impurity vs Effective alpha

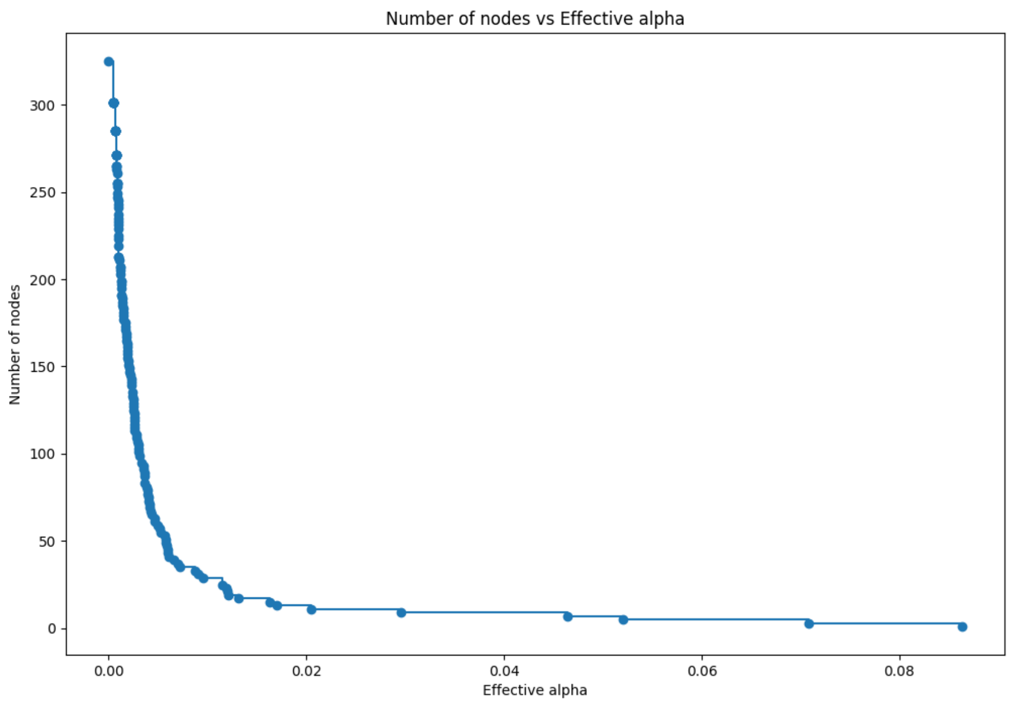


Figure 15: Number of nodes vs effective alpha

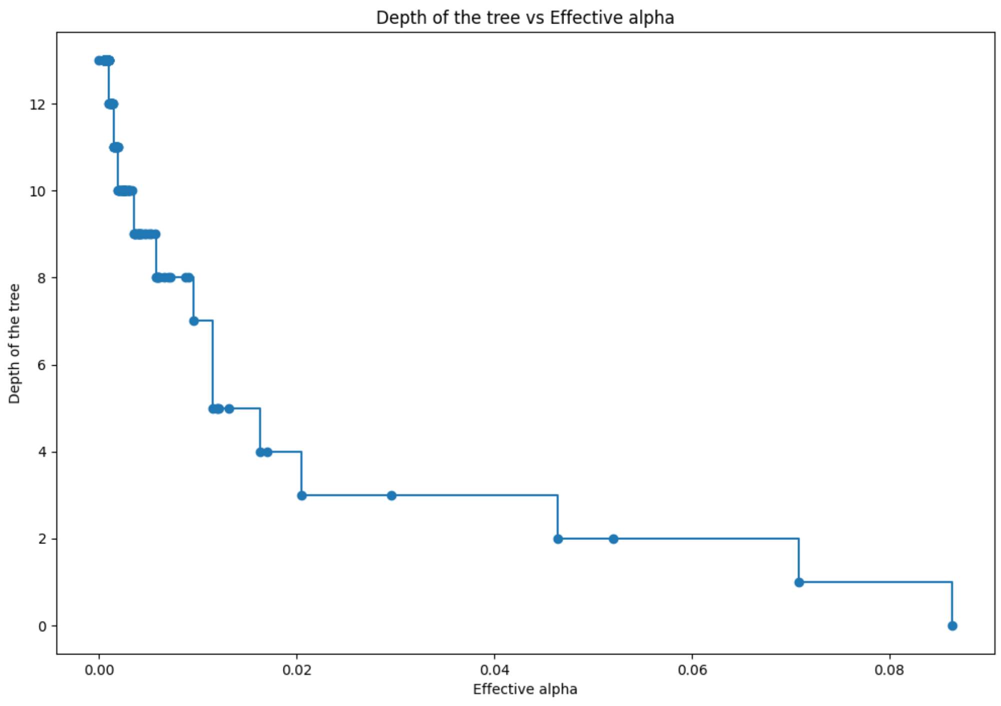


Figure 16: Depth vs effective alpha

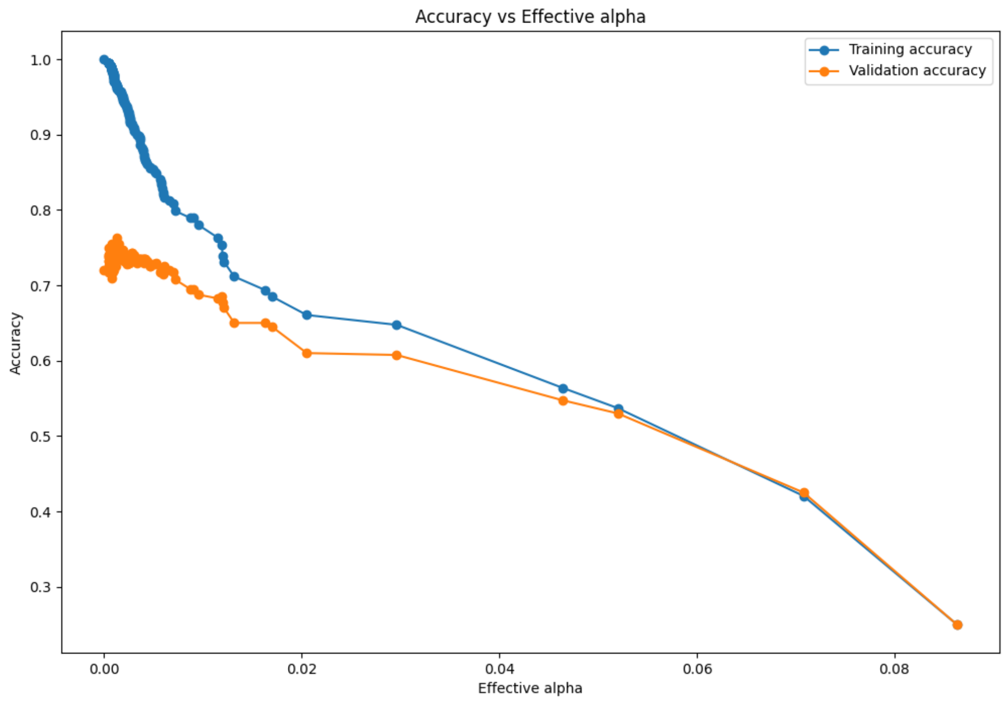


Figure 17: Training and Validation accuracies vs effective alpha

Observations -

As the effective alpha increases the value of total impurity in the leaves increases. This means that increasing alpha leads to non-homogeneity increasing at the leaf nodes. This is coupled with the decrease in the number of nodes as alpha increases, thus leading to more pruning of the tree. Both of those effects are again related to the depth of the tree. Higher alpha prunes the tree more thus leading to a decrease in the depth of the tree. For all three plots described, the values change rather slowly when alpha is small but the change is drastic when alpha reaches a larger value.
Coming to the Accuracy plot, we can see that the training accuracy decreases as we increase the alpha. This is due to an increase in the pruning of the tree. The validation accuracy, however, initially increases with alpha, reaches a maximum value which is the best-performing tree for us and then it decreases with alpha due to unreasonable pruning of the tree.

We use the validation data for getting the best tree. Following is the tree

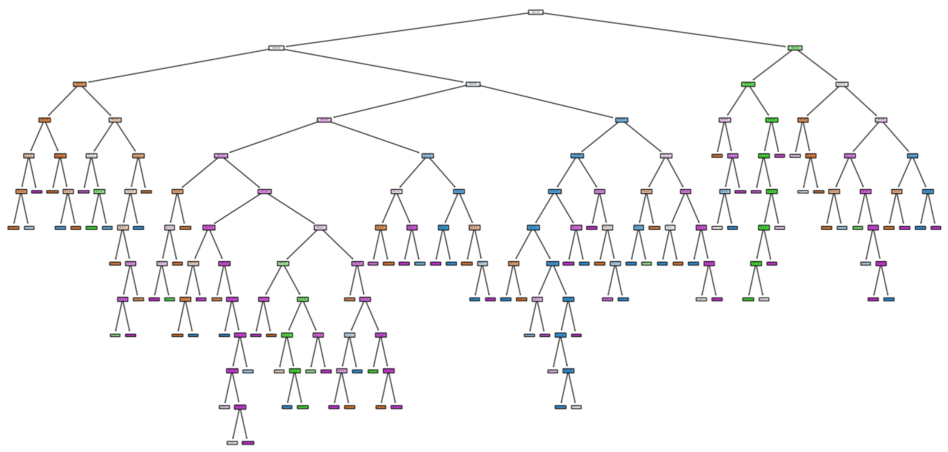
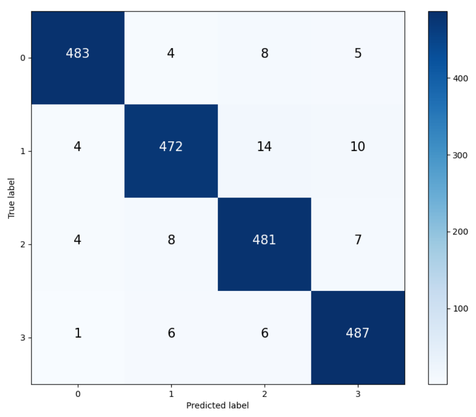


Figure 18: Best-performing tree

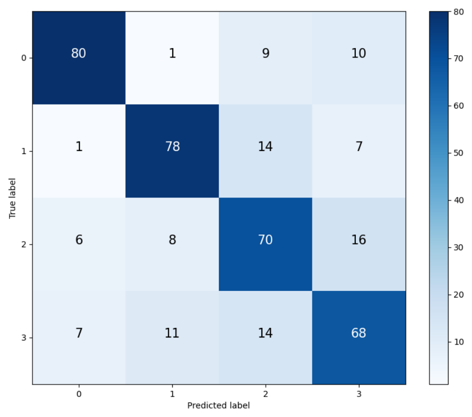
The corresponding training and validation accuracies are as follows :-
Training Accuracy: 0.9615
Validation Accuracy: 0.74

The alpha corresponding to these accuracies is - 0.0011666666666666667

Confusion Matrices for the best tree are as follows :



(a) Training Data



(b) Validation Data

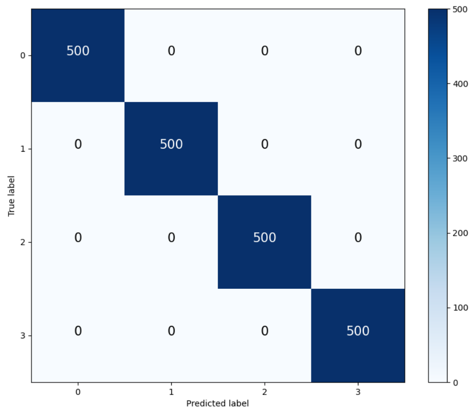
2.4 Random Forest

With the default parameters, following are the observations obtained.

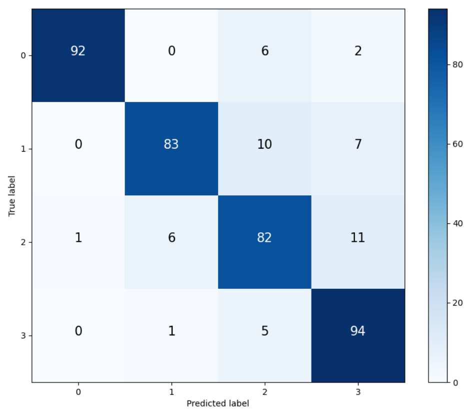
Default hyperparameters:

Training accuracy: 1.0
Validation accuracy: 0.8775

The confusion matrices for the default hyperparameters are as follows:-



(c) Training Data



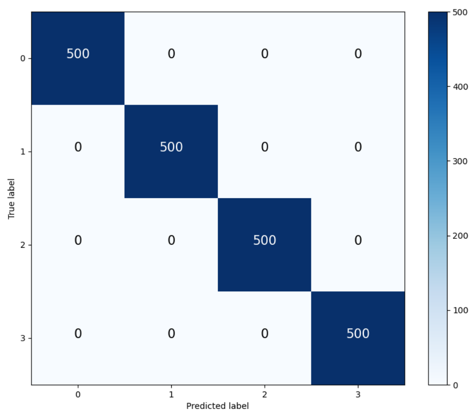
(d) Validation Data

For Best hyperparameters:

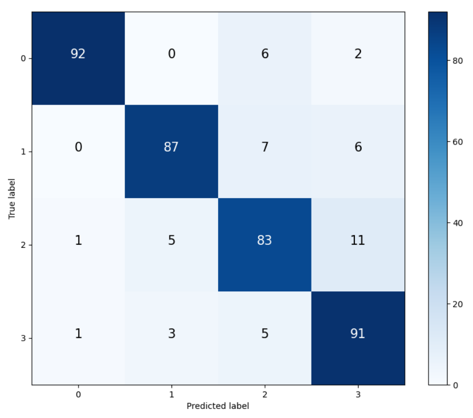
Best hyperparameters: Training accuracy: 1.0
Validation accuracy: 0.8825

The best hyperparameters are as follows:-
1. criterion: 'entropy'
2. max_depth: None
3. min_samples_split: 10
4. n_estimators : 100

The confusion matrices for the default hyperparameters are as follows:-



(e) Training Data



(f) Validation Data

2.5 Gradient Boosted Trees and XGBoost

The training with all features was taking a large amount of time and hence I have selected the 10 best features with the help of sklearn and trained my model on them. The analysis is based on the same.

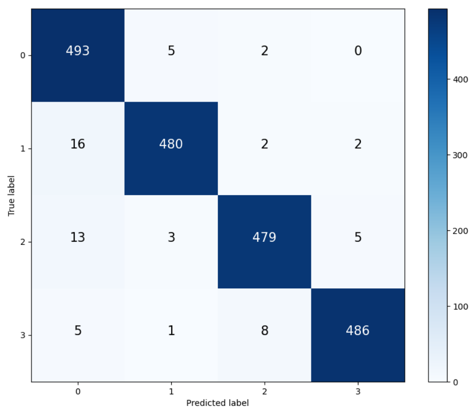
2.5.1 For Gradient Boosted Trees:-

The best hyperparameters are as follows:-
1. max_depth: 7
2. subsample: 0.4
3. n_estimators : 50

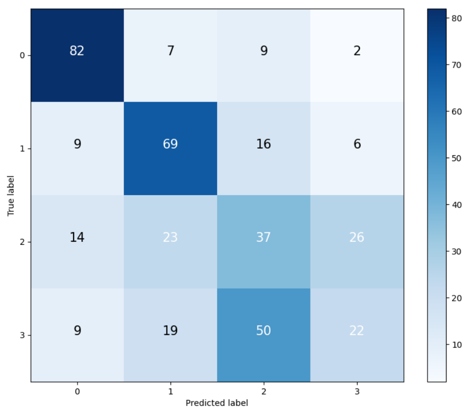
Training Time: 521.3968815803528 seconds

Training accuracy: 0.969
Validation accuracy: 0.525

The confusion matrices are as follows:-



(g) Training Data



(h) Validation Data

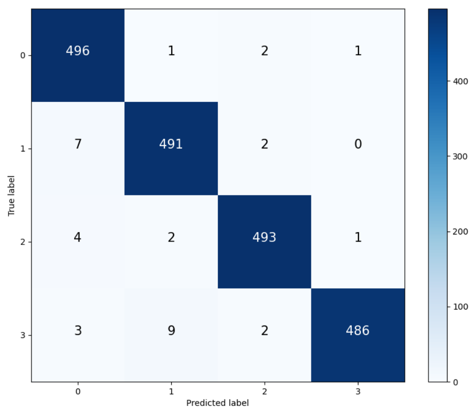
2.5.2 Using XGBoost:-

The best hyperparameters are as follows:-
1. max_depth: 10
2. subsample: 0.6
3. n_estimators : 20

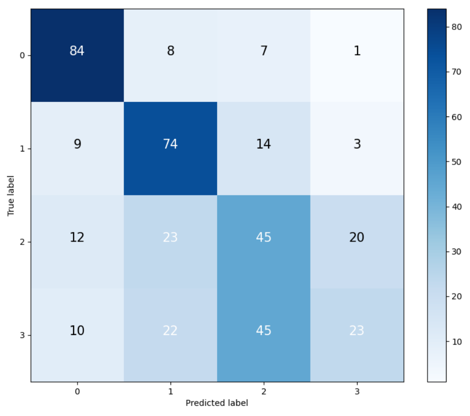
Training Time: 152.4618935585022 seconds

Training accuracy: 0.983
Validation accuracy: 0.565

The confusion matrices are as follows:-



(i) Training Data



(j) Validation Data

2.6 Real-Time Application

For doing this, I took about 10 photos of myself with me looking in different directions. I also removed my spectacles in some of the photos. I then converted these images into 32*32 pixel images and then took prediction on them. I used my best model i.e the Random Forest for training and then predicting on my images. The accuracy obtained was not very high as it could correctly classify 4 out of the 10 images, therefore, having an accuracy of 40%. In two of these images, I am facing almost straight towards the camera. In the other two, 50% part of my face is visible with me looking towards the side. Following are the 4 32*32 images which were correctly classified by my model:-

3 Some Additional things that I tried doing

As the number of features is pretty high, my own decision tree implementation for part 1.1 takes a good amount of training time. Hence, I tried using a popular technique majorly used in neural networks, pooling. I tried doing average pooling with a kernel of size 2*2 and stride as 1 effectively reducing the number of features by a factor of 4. Even this model produced a comparable accuracy to the model trained with all features. Using information gain as the criterion, the avg. pooling model ran in about 7 minutes and produced a validation accuracy of 93.25%. I would have made use of this model in my submission but as it was mentioned in Piazza to not use any technique to reduce the features and to work with all of them, hence I did not submit the avg pooling model.