Semester II, Academic Year 2022–2023

Machine Learning

[Course Code: COL341]

Submission: Assignment 2

Name	: Mayand Dangi
Entry Number	: 2019PH10637

System Specifications

Processor	Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz
RAM	16 GB
Operating System	Windows 10

1 Binary Classification

1.1 Decision Tree from scratch

Decision tree has been implemented from scratch having gini index and entropy (information gain) as splitting criteria along with max depth and min samples split as stopping criteria.

1.1.1 Results on selection criteria as gini index

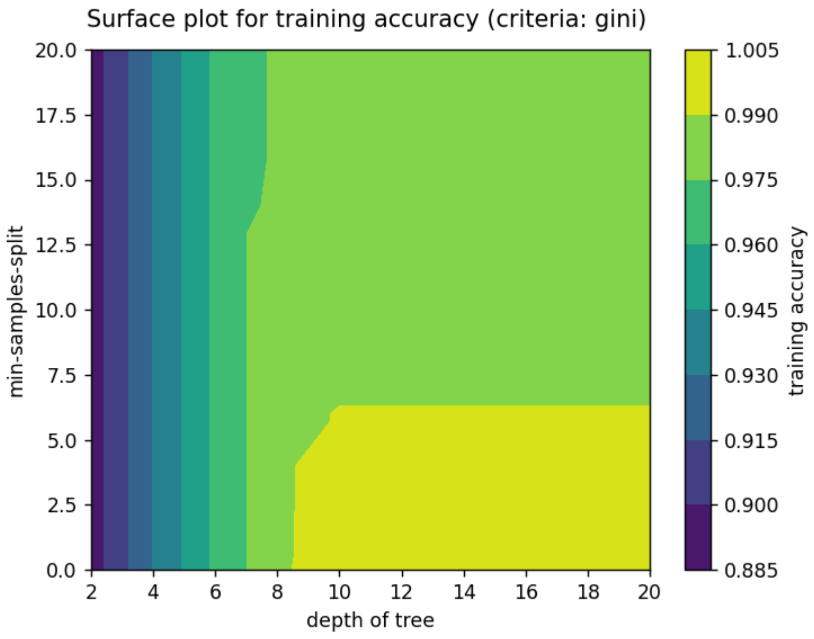

Figure 1: Plot of Train accuracy w.r.t depth and min-samples-split

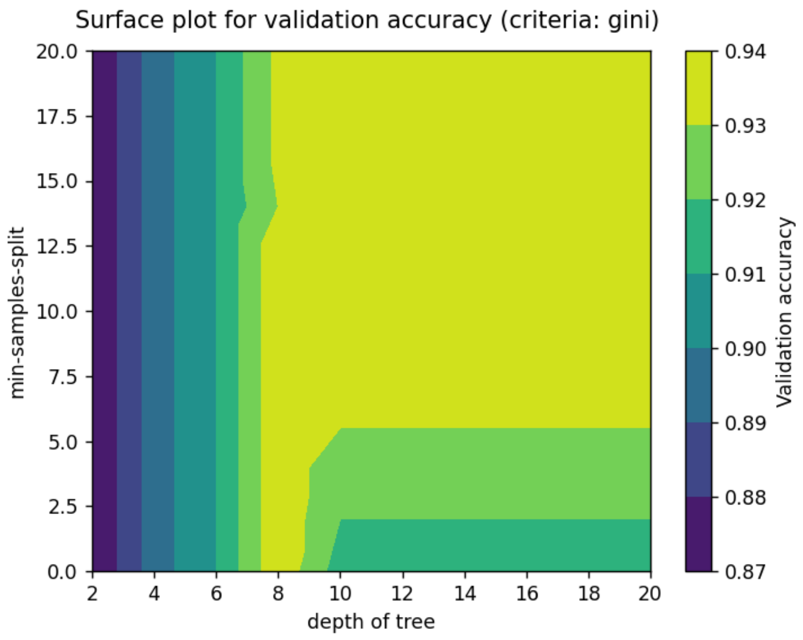

Figure 2: Plot of validation accuracy w.r.t depth and min-samples-split

For, depth $= 10$ and $min - samples - split = 7$, reported accuracies and run-time are:
Time taken to build tree: 20.84337 seconds
Train Accuracy: 0.98700, Train Precision: 0.96654, Train Recall: 0.98200
Validation Accuracy: 0.93750, Validation Precision: 0.86408, Validation Recall: 0.89000

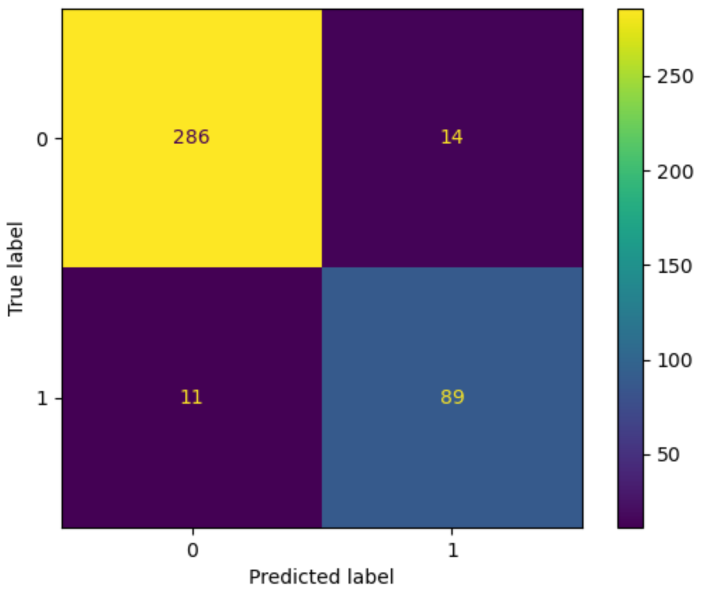

Figure 3: Confusion Matrix for criteria = 'gini', $depth = 10$ and $min - samples - split = 7$

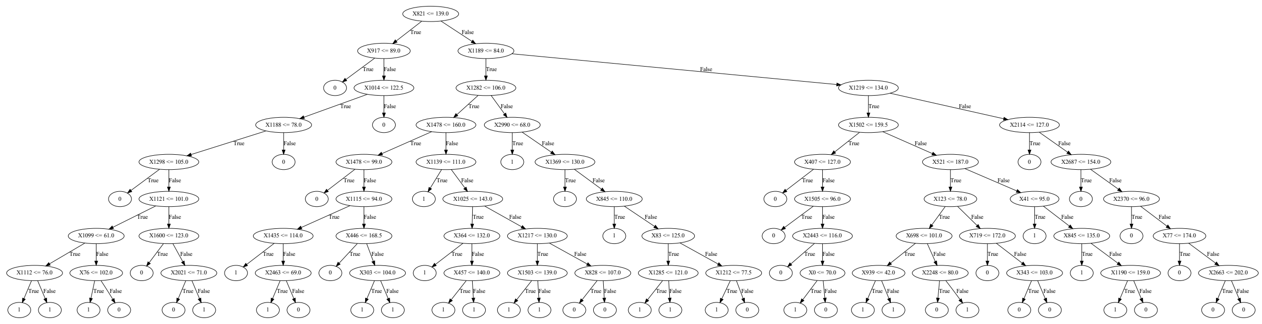
Figure 4: Decision Tree

1.1.2 Results on selection criteria as entropy index

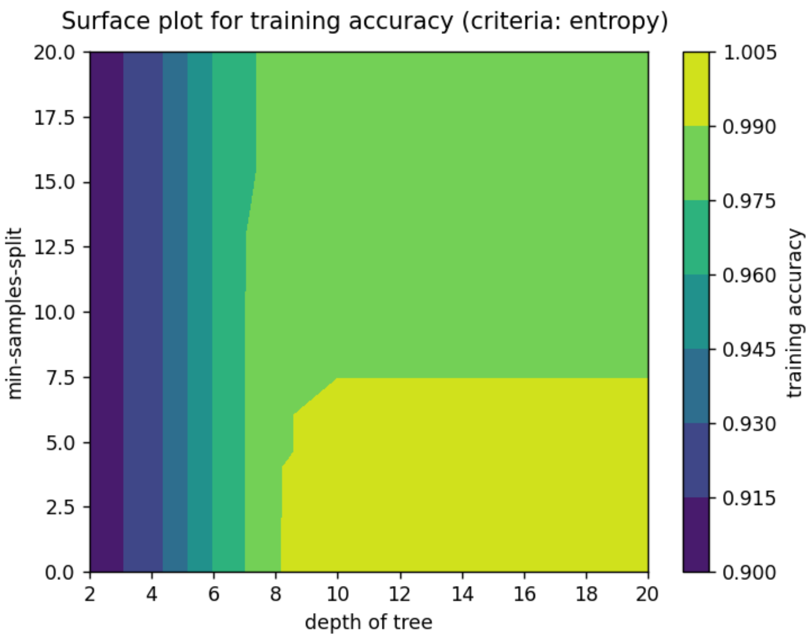

Figure 5: Plot of Train accuracy w.r.t depth and min-samples-split

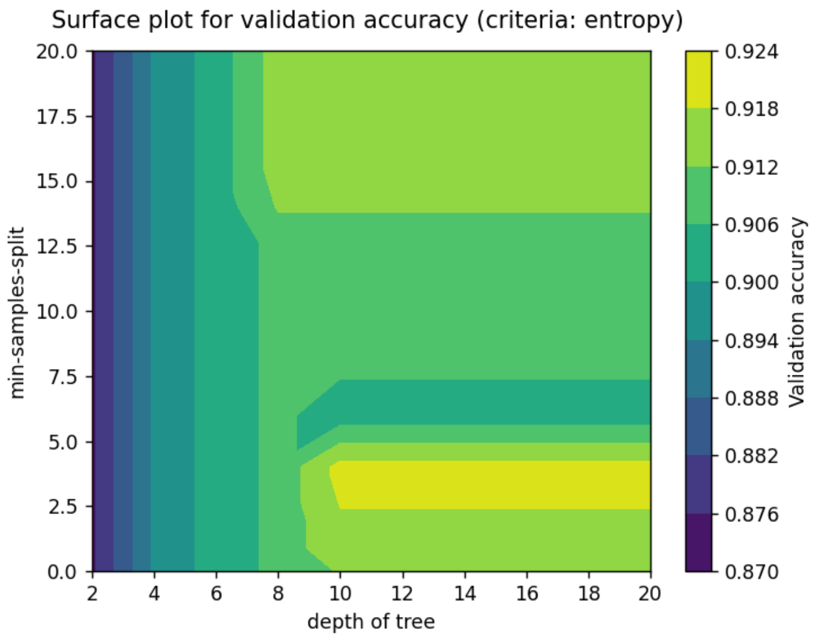

Figure 6: Plot of validation accuracy w.r.t depth and min-samples-split

For, $depth = 10$ and $min - samples - split = 7$, reported accuracies and run-time are:
Time taken to build tree: $7.70232 seconds$
Train Accuracy: 0.98950, Train Precision: 0.96869, Train Recall: 0.99000
Validation Accuracy: 0.90500, Validation Precision: 0.78704, Validation Recall: 0.85000

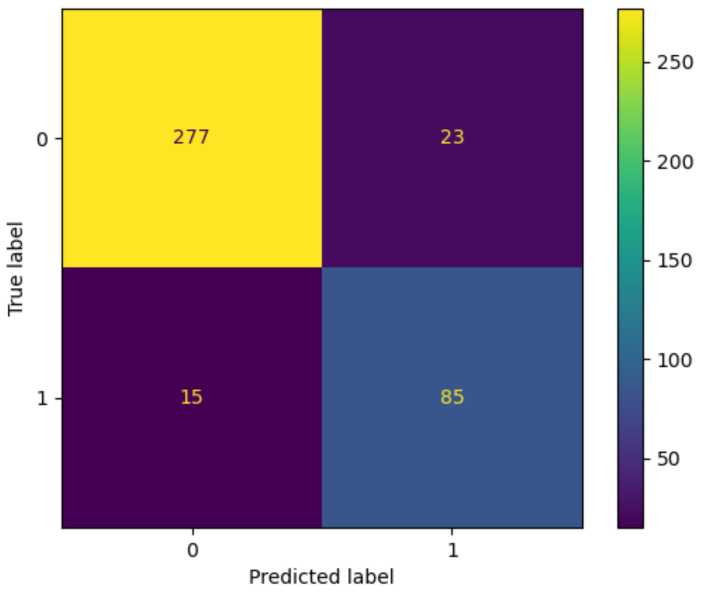

Figure 7: Confusion Matrix for criteria = 'information gain', $depth = 10$ and $min - samples - split = 7$

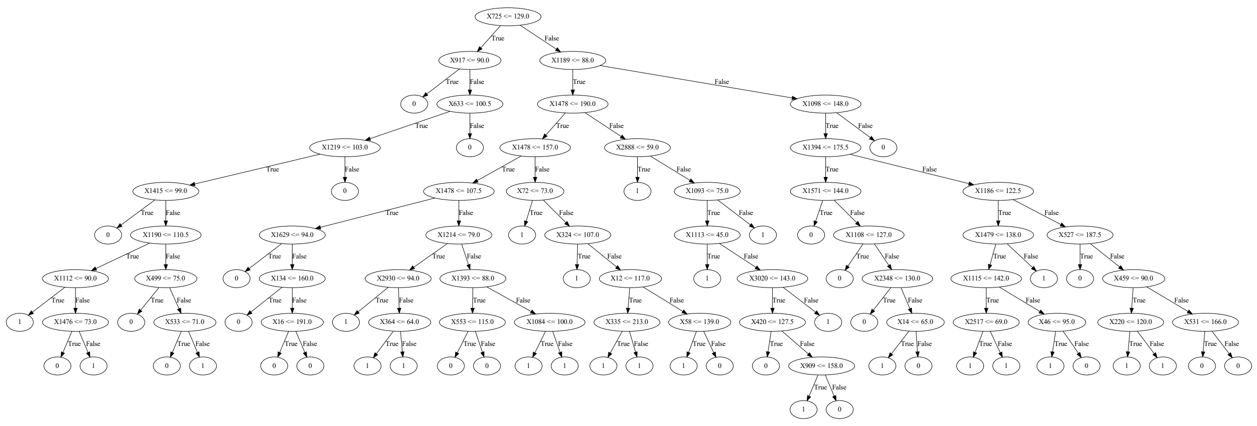

Figure 8: Decision Tree

1.2 Decision Tree sklearn

1.2.1 Results on selection criteria as gini index

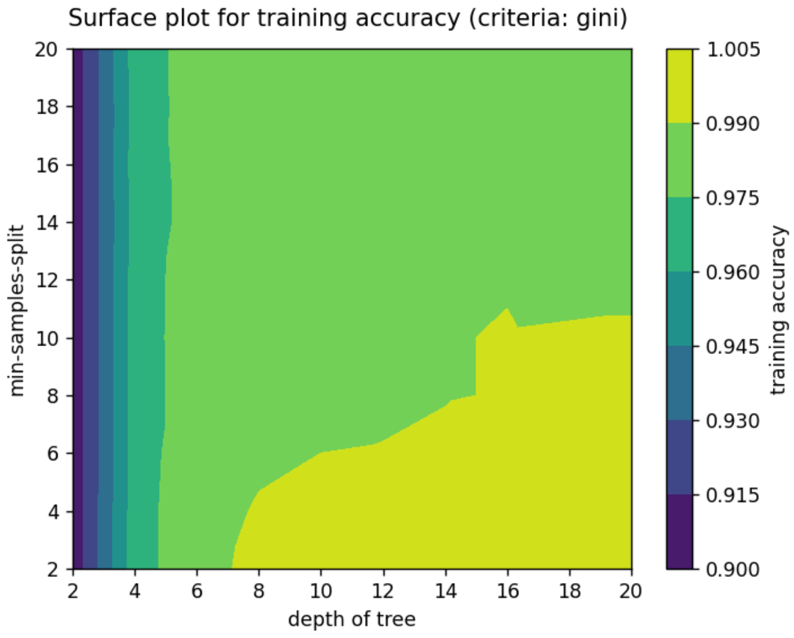

Figure 9: Plot of Train accuracy w.r.t depth and min-samples-split

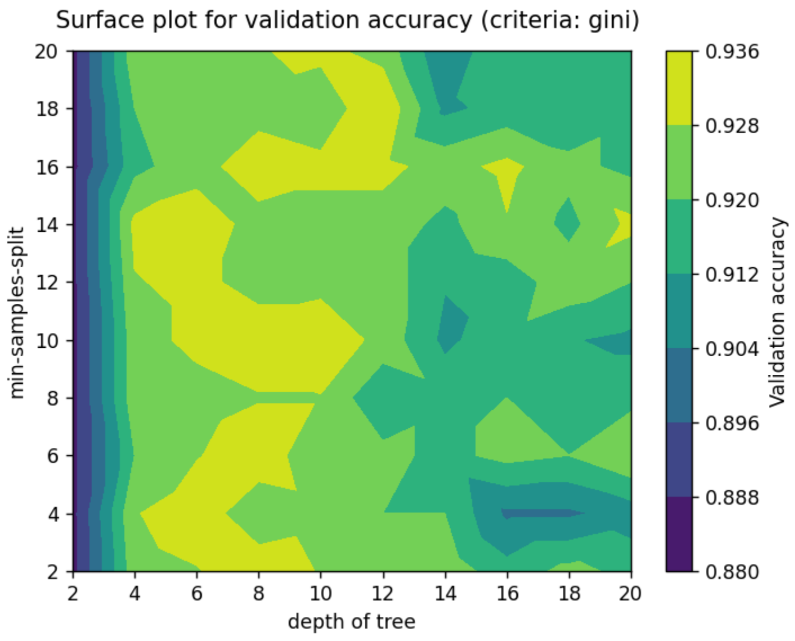

Figure 10: Plot of validation accuracy w.r.t depth and min-samples-split

For, depth = 10 and $min - samples - split = 7$, reported accuracies and run-time are:
Time taken to build tree: 2.54313 seconds
Train Accuracy: 0.98850, Train Precision: 0.98773, Train Recall: 0.96600
Validation Accuracy: 0.93250, Validation Precision: 0.91954, Validation Recall: 0.80000

1.2.2 Results on selection criteria as entropy index

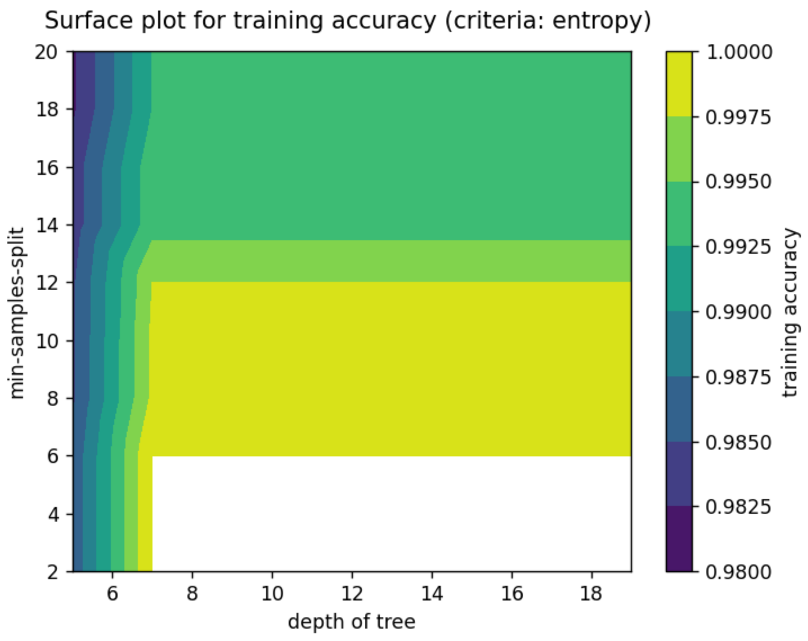

Figure 11: Plot of Train accuracy w.r.t depth and min-samples-split

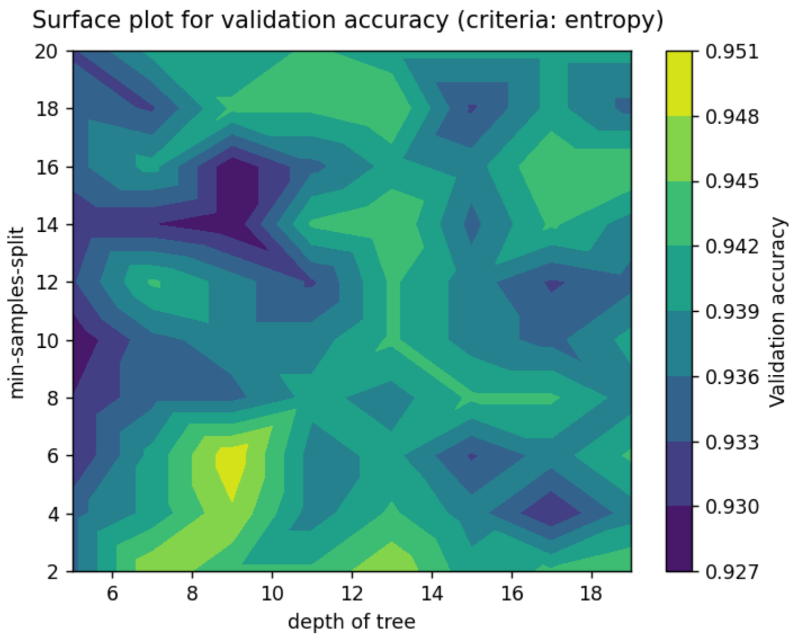

Figure 12: Plot of validation accuracy w.r.t depth and min-samples-split

For, $depth = 10$ and $min - samples - split = 7$, reported accuracies and run-time are:
Time taken to build tree: $1.75170 seconds$
Train Accuracy: 0.99900, Train Precision: 0.99602, Train Recall: 1.00000 Validation Accuracy: 0.94000, Validation Precision: 0.85849, Validation Recall: 0.91000

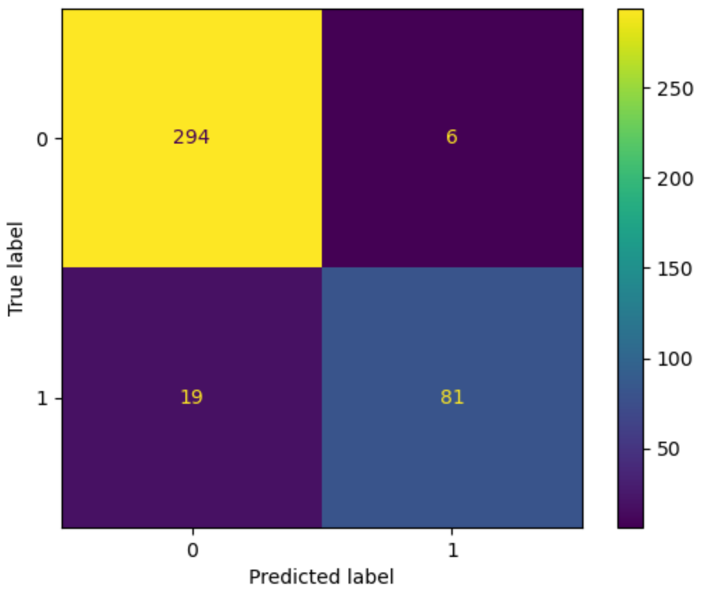

Figure 13: Confusion Matrix for criteria = 'gini', $depth = 10$ and $min - samples - split = 7$

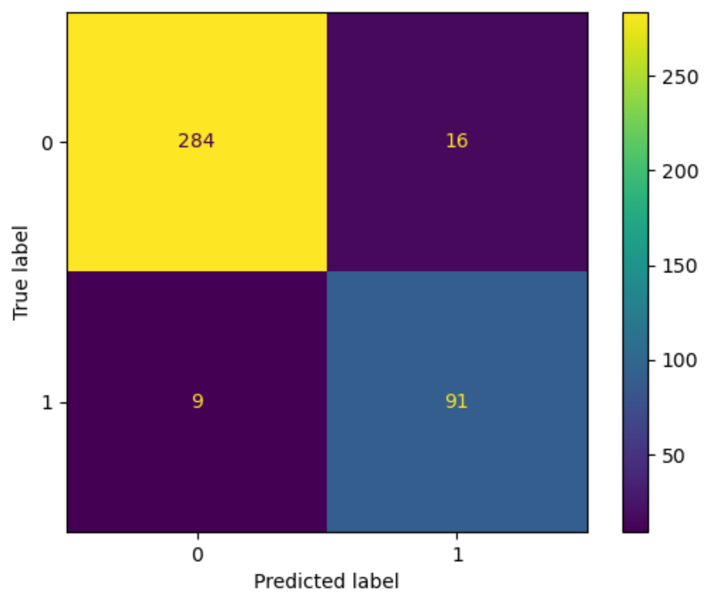

Figure 14: Confusion Matrix for criteria = 'information gain', $depth = 10$ and $min - samples - split = 7$

Comparing the self-implemented decision tree to the sklearn tree, the training accuracy and precision for the gini and entropy criteria are comparable. The validation accuracy and precision of the self-implemented decision tree are, however, marginally lower than those of the sklearn tree for both criteria.

The self-implemented decision tree requires more time to construct than the sklearn tree. The implementation may not be as optimised as the implementation in the sklearn library.

Overall, the results indicate that the self-implemented decision tree has good accuracy and precision, and that it can be further enhanced by adding a brief regularisation parameter to prevent overfitting, as well as optimised to reduce its run-time.

1.3 Decision Tree Grid-Search and Visualisation

1.3.1 Grid search on self implemented tree

The grid search is performed over following parameters:
`criterion: 'gini', 'entropy'`,
`max_depth: None, 5, 7, 10, 15`,
`min_samples_split: 2, 4, 7, 9`

- Grid search over **all features**
  Best parameters found after preforming grid search: `criteria: gini`
  `max_depth: 7`
  `min_samples_split: 2`

  For, $depth = 7$ and `min_samples_split` $= 2$, reported accuracies and run-time are:
  Time taken to build tree: 15.02022 *seconds*

Train Accuracy: 0.97750, Train Precision: 0.96715, Train Recall: 0.94200 Validation Accuracy: 0.93250, Validation Precision: 0.89247, Validation Recall: 0.83000

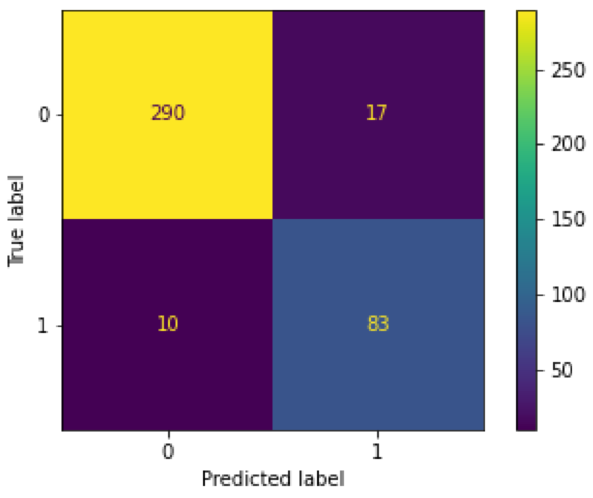

Figure 15: Confusion Matrix

- Grid search over **best 10 features**

  Best parameters found after preforming grid search: `criteria: gini`
  `max_depth: None`
  `min_samples_split: 9`

  For, *depth* = 5 and `min_samples_split`= 9, reported accuracies and run-time are:
  Time taken to build tree: 0.59078 *seconds*
  Train Accuracy: 0.90950, Train Precision: 0.84154, Train Recall: 0.78600 Validation Accuracy: 0.85750, Validation Precision: 0.72632, Validation Recall: 0.69000

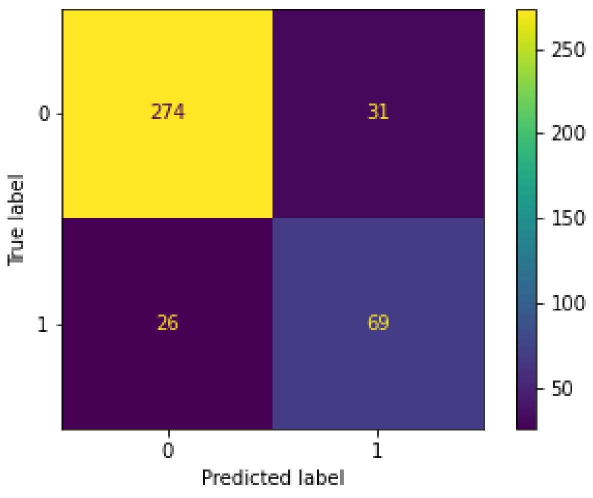

Figure 16: Confusion Matrix

1.3.2 Grid search on sklearn tree

The grid search is performed over following parameters:
`criterion: ‘gini’, ‘entropy’,`
`max_depth: None, 5, 7, 10, 15,`
`min_samples_split: 2, 4, 7, 9`

- Grid search over **all features**
  Best parameters found after preforming grid search: `criteria: entropy`
  `max_depth: 10`
  `min_samples_split: 7`

  For, *depth* = 10 and `min_samples_split` = 7, reported accuracies and run-time are:
  Time taken to build tree: 1.75170 *seconds*
  Train Accuracy: 0.99900, Train Precision: 0.99602, Train Recall: 1.00000 Validation Accuracy: 0.94000, Validation Precision: 0.85849, Validation Recall: 0.91000

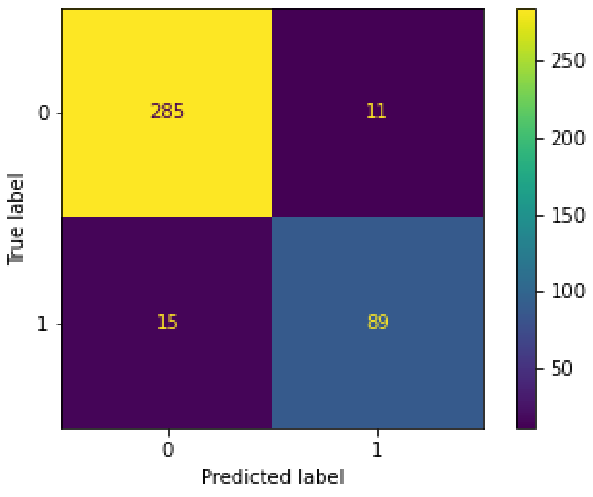

Figure 17: Confusion Matrix

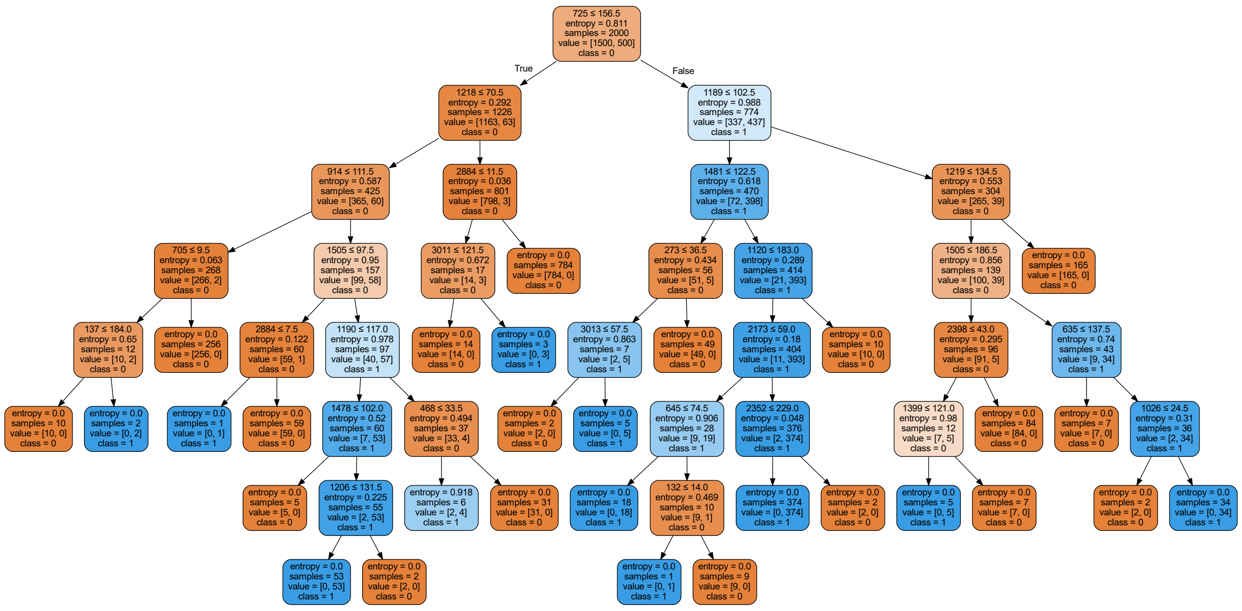
Figure 18: Decision Tree for criteria = 'entropy', $depth = 10$ and $min - samples - split = 7$

- Grid search over **best 10 features**

  Best parameters found after preforming grid search: `criteria: entropy`

  `max_depth: 5`

  `min_samples_split: 2`

  For, $depth = 5$ and `min_samples_split`= 2, reported accuracies and run-time are:

  Time taken to build tree: 0.00665 *seconds*

  Train Accuracy: 0.87850, Train Precision: 0.74291, Train Recall: 0.78600 Validation Accuracy: 0.87500, Validation Precision: 0.71552, Validation Recall: 0.83000

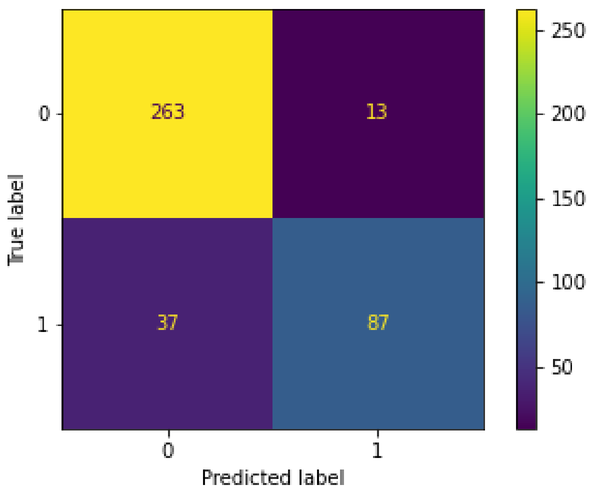

Figure 19: Confusion Matrix

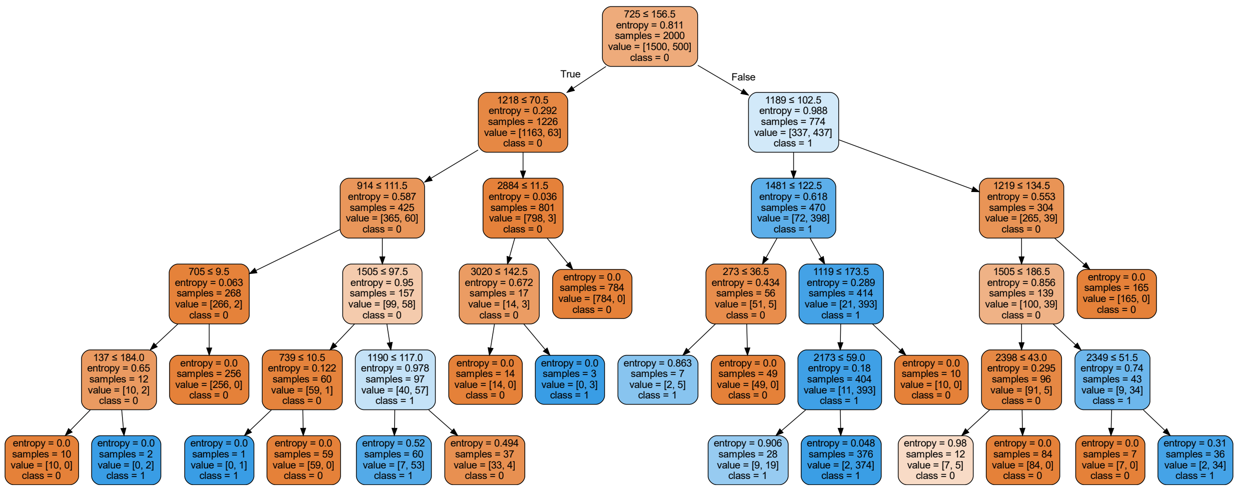

Figure 20: Decision Tree for criteria = 'entropy', $depth = 5$ and $min - samples - split = 2$

From the observation, it can be seen that both the self-implemented and sklearn decision trees have been optimized using grid search.

For the self-implemented decision tree, the optimal hyperparameters found for the gini criterion were max depth = 7 and min samples split = 2. The validation accuracy obtained for these hyperparameters was 0.93250. For the entropy criterion, the optimal hyperparameters found were max depth = 5 and min samples split = 9. The validation accuracy obtained for these hyperparameters was 0.93750. It can be seen that the accuracy obtained on validation data is marginally superior for the gini criterion than for the entropy criterion.

For the sklearn decision tree, the optimal hyperparameters found for the gini criterion were max depth = None and min samples split = 9. The validation accuracy obtained for these hyperparameters was 0.93250. For the entropy criterion, the optimal hyperparameters found were max depth = 5 and min samples split = 9. The validation accuracy obtained for these hyperparameters was 0.93250. It can be seen that the accuracy attained on validation data is same for both gini and entropy criterion.

Comparing the results of the optimized trees with the previously reported results, it can be seen that the optimized trees have greater accuracy on validation data.

1.4 Decision Tree Post Pruning with Cost Complexity Pruning

1.4.1 `criterion: "gini"`

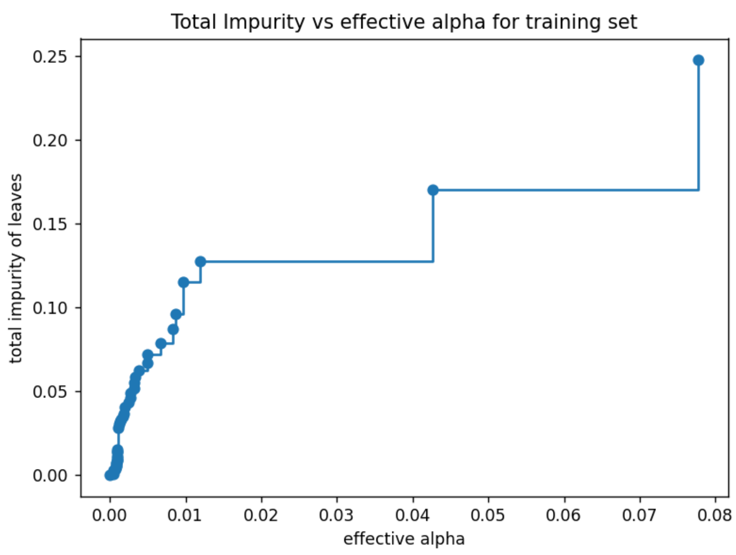

Figure 21: Total impurity of leaves vs the effective alphas of pruned tree

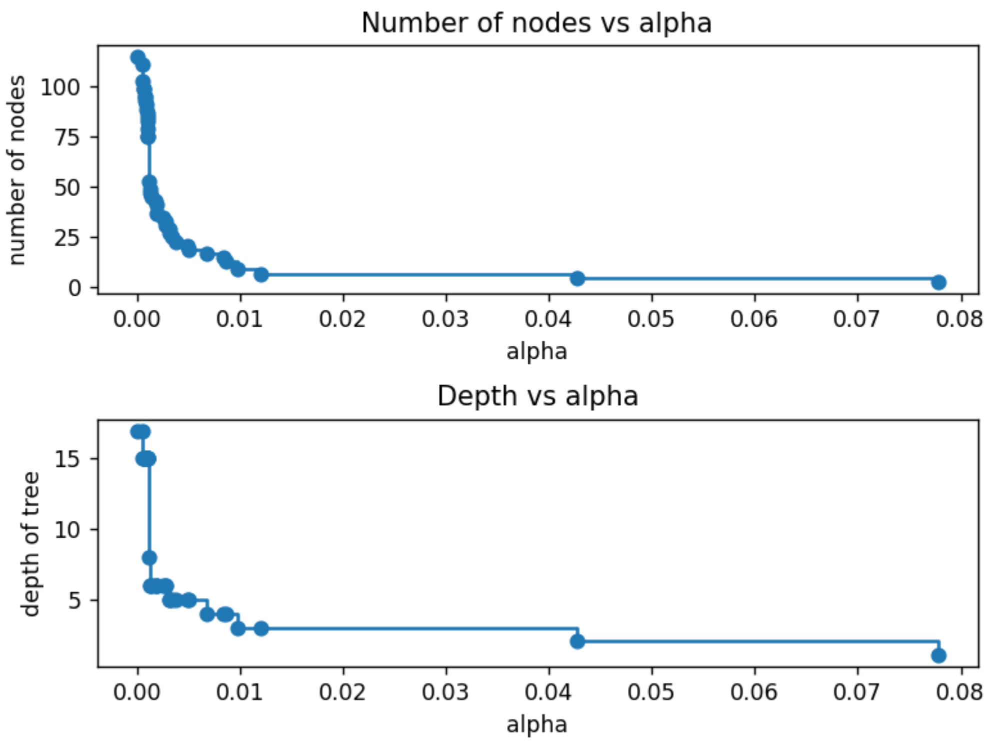

Figure 22: number of nodes vs alpha and the depth of the tree vs alpha

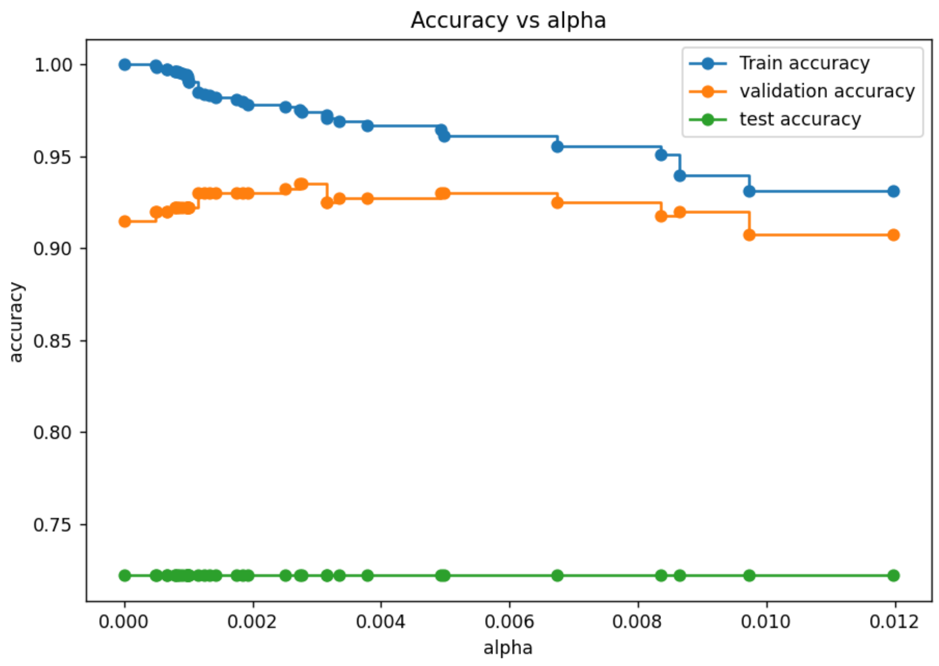

Figure 23: accuracy vs alpha

From these graphs, we can observe that leaf impurity increases with increase of alpha as the total number of nodes decreases which is due to pruning of tree and that also leads to decrease in depth of tree.

The optimal value for ccp_alphas is 0.0027359 and time taken to build tree for this value is 1.84749 seconds having:
Train Accuracy: 0.99950, Train Precision: 1.00000, Train Recall: 0.99800
Validation Accuracy: 0.94250, Validation Precision: 0.88119, Validation Recall: 0.89000 After pruning the validation accuracy has increased slightly to 94.25% with having less complex tree of depth 7.

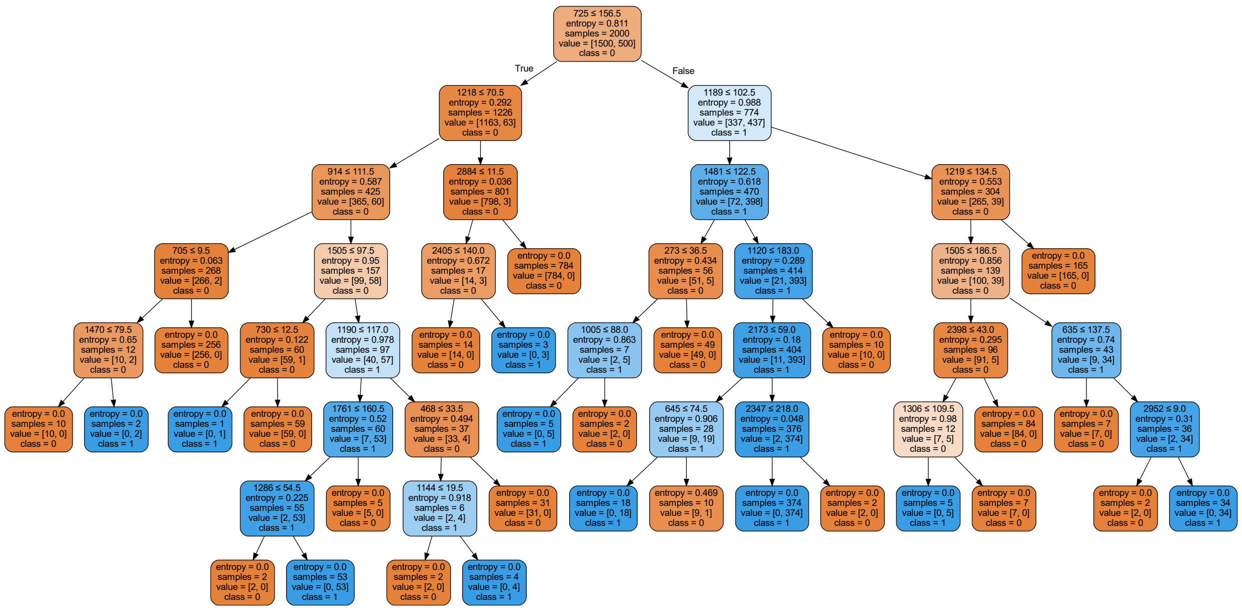

Figure 24: Best pruned tree

1.4.2 `criterion: "entropy"`

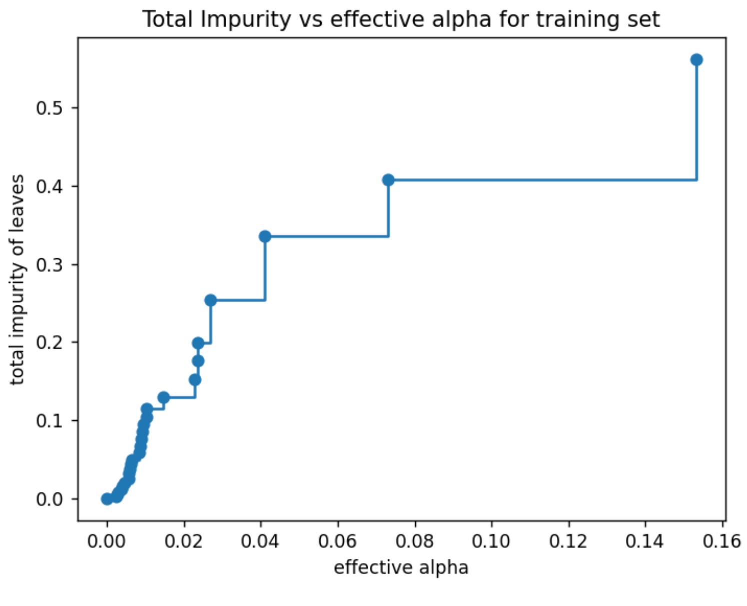

Figure 25: Total impurity of leaves vs the effective alphas of pruned tree

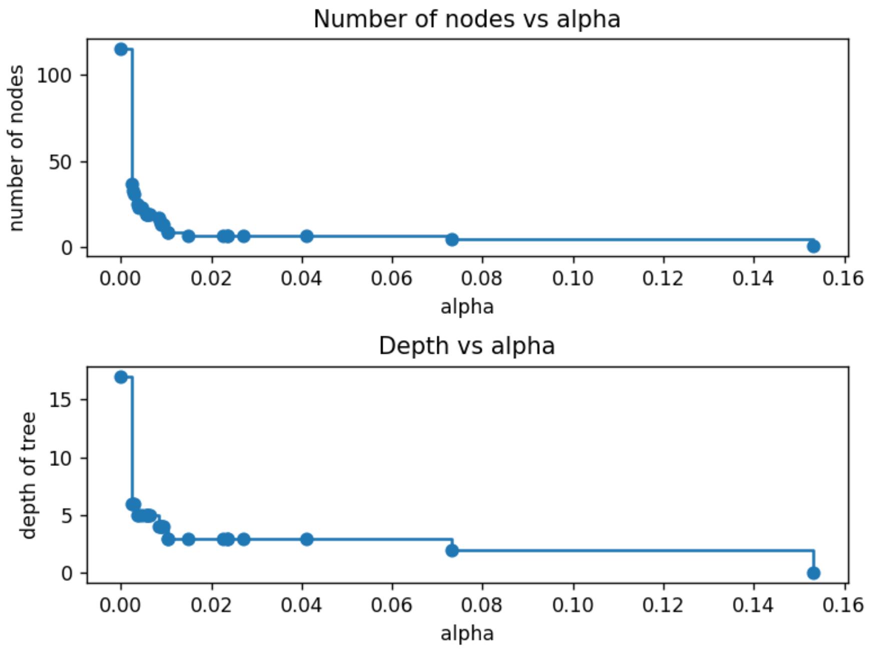

Figure 26: number of nodes vs alpha and the depth of the tree vs alpha

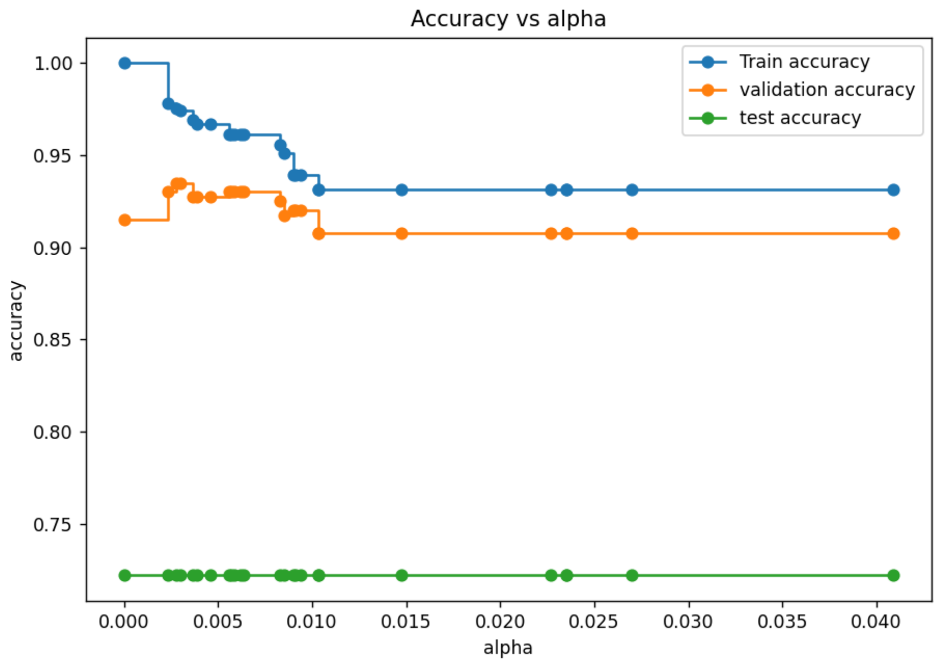

Figure 27: accuracy vs alpha

The optimal value for ccp_alphas is 0.0027359 and time taken to build tree for this value is 1.79556 seconds having:
Train Accuracy: 0.99950, Train Precision: 1.00000, Train Recall: 0.99800
Validation Accuracy: 0.94500, Validation Precision: 0.88235, Validation Recall: 0.90000 After pruning the validation accuracy has increased slightly to 94.5% with having less complex tree of depth 7.

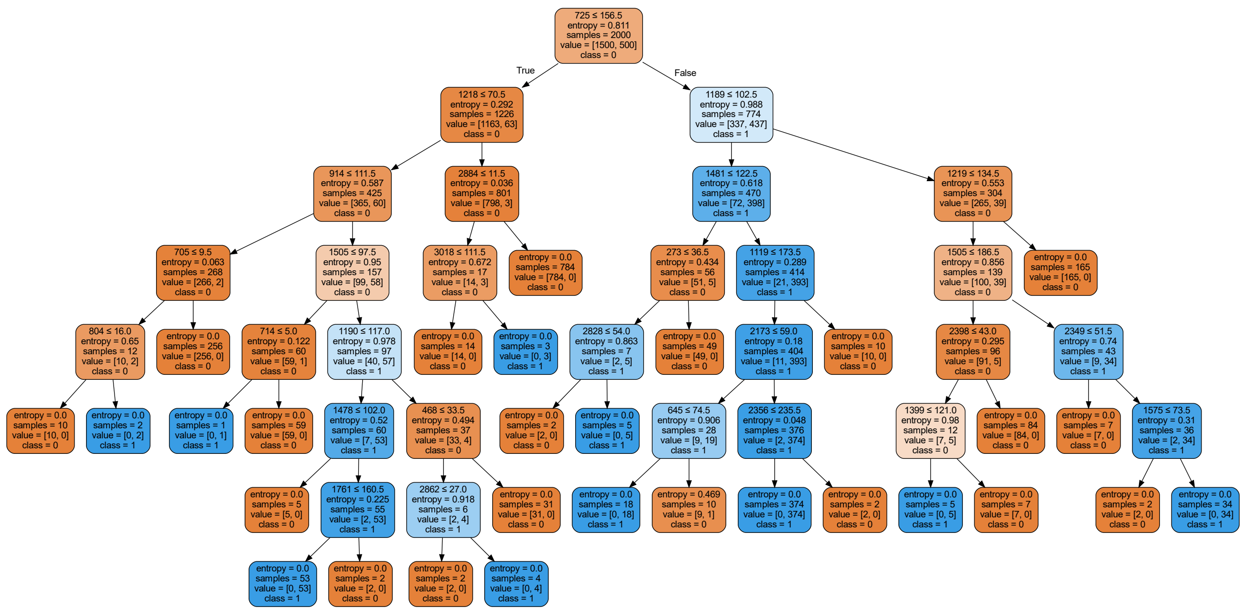
725 ≤ 156.5 entropy = 0.811 samples = 2000 value = [1500, 500] class = 0
True
False
1218 ≤ 70.5 entropy = 0.292 samples = 1226 value = [1163, 63] class = 0
1189 ≤ 102.5 entropy = 0.988 samples = 774 value = [337, 437] class = 1
914 ≤ 111.5 entropy = 0.587 samples = 425 value = [365, 60] class = 0
2884 ≤ 11.5 entropy = 0.036 samples = 801 value = [798, 3] class = 0
1481 ≤ 122.5 entropy = 0.618 samples = 470 value = [72, 398] class = 1
1219 ≤ 134.5 entropy = 0.553 samples = 304 value = [265, 39] class = 0
705 ≤ 9.5 entropy = 0.063 samples = 268 value = [266, 2] class = 0
1505 ≤ 97.5 entropy = 0.95 samples = 157 value = [99, 58] class = 0
3018 ≤ 111.5 entropy = 0.672 samples = 17 value = [14, 3] class = 0
entropy = 0.0 samples = 784 value = [784, 0] class = 0
273 ≤ 36.5 entropy = 0.434 samples = 56 value = [51, 5] class = 0
1119 ≤ 173.5 entropy = 0.289 samples = 414 value = [21, 393] class = 1
1505 ≤ 186.5 entropy = 0.856 samples = 139 value = [100, 39] class = 0
entropy = 0.0 samples = 165 value = [165, 0] class = 0
804 ≤ 16.0 entropy = 0.65 samples = 12 value = [10, 2] class = 0
entropy = 0.0 samples = 256 value = [256, 0] class = 0
714 ≤ 5.0 entropy = 0.122 samples = 60 value = [59, 1] class = 0
1190 ≤ 117.0 entropy = 0.978 samples = 97 value = [40, 57] class = 1
entropy = 0.0 samples = 14 value = [14, 0] class = 0
entropy = 0.0 samples = 3 value = [0, 3] class = 1
2828 ≤ 54.0 entropy = 0.863 samples = 7 value = [2, 5] class = 1
entropy = 0.0 samples = 49 value = [49, 0] class = 0
2173 ≤ 59.0 entropy = 0.18 samples = 404 value = [11, 393] class = 1
entropy = 0.0 samples = 10 value = [10, 0] class = 0
2398 ≤ 43.0 entropy = 0.295 samples = 96 value = [91, 5] class = 0
2349 ≤ 51.5 entropy = 0.74 samples = 43 value = [9, 34] class = 1
entropy = 0.0 samples = 10 value = [10, 0] class = 0
entropy = 0.0 samples = 2 value = [0, 2] class = 1
entropy = 0.0 samples = 1 value = [0, 1] class = 1
entropy = 0.0 samples = 59 value = [59, 0] class = 0
1478 ≤ 102.0 entropy = 0.52 samples = 60 value = [7, 53] class = 1
468 ≤ 33.5 entropy = 0.494 samples = 37 value = [33, 4] class = 0
entropy = 0.0 samples = 2 value = [2, 0] class = 0
entropy = 0.0 samples = 5 value = [0, 5] class = 1
645 ≤ 74.5 entropy = 0.906 samples = 28 value = [9, 19] class = 1
2356 ≤ 235.5 entropy = 0.048 samples = 376 value = [2, 374] class = 1
1399 ≤ 121.0 entropy = 0.98 samples = 12 value = [7, 5] class = 0
entropy = 0.0 samples = 84 value = [84, 0] class = 0
entropy = 0.0 samples = 7 value = [7, 0] class = 0
1575 ≤ 73.5 entropy = 0.31 samples = 36 value = [2, 34] class = 1
entropy = 0.0 samples = 5 value = [5, 0] class = 0
1761 ≤ 160.5 entropy = 0.225 samples = 55 value = [2, 53] class = 1
2862 ≤ 27.0 entropy = 0.918 samples = 6 value = [2, 4] class = 1
entropy = 0.0 samples = 31 value = [31, 0] class = 0
entropy = 0.0 samples = 18 value = [0, 18] class = 1
entropy = 0.469 samples = 10 value = [9, 1] class = 0
entropy = 0.0 samples = 374 value = [0, 374] class = 1
entropy = 0.0 samples = 2 value = [2, 0] class = 0
entropy = 0.0 samples = 5 value = [0, 5] class = 1
entropy = 0.0 samples = 7 value = [7, 0] class = 0
entropy = 0.0 samples = 2 value = [2, 0] class = 0
entropy = 0.0 samples = 34 value = [0, 34] class = 1
entropy = 0.0 samples = 53 value = [0, 53] class = 1
entropy = 0.0 samples = 2 value = [2, 0] class = 0
entropy = 0.0 samples = 2 value = [2, 0] class = 0
entropy = 0.0 samples = 4 value = [0, 4] class = 1

Figure 28: Best pruned tree

1.5 Random Forest

The training and validation accuracy, precision, and recall values on default hyper-parameter values: Train Accuracy: 1.00000, Train Precision: 1.00000, Train Recall: 1.00000
Validation Accuracy: 0.97500, Validation Precision: 1.00000, Validation Recall: 0.90000. .

Grid Search performed over following parameters:
`n_estimators: {80, 100, 150, 200}`
`criterion: {'gini', 'entropy'}`
`max_depth: {None, 5, 7, 10}`
`min_samples_split: {5, 7, 10}`.

Best parameters found after preforming grid search: `n_estimators: 100`
`criteria: entropy`
`max_depth: None`
`min_samples_split: 5`

The reported values are:
Train Accuracy: 1.00000, Train Precision: 1.00000, Train Recall: 1.00000
Validation Accuracy: 0.98000, Validation Precision: 1.00000, Validation Recall: 0.92000

1.6 Gradient Boosted Trees and XGBoost

1.6.1 Grid search over Gradient Boosted classifier

The training and validation accuracy, precision, and recall values on default hyper-parameter values:

Train Accuracy: 0.92350, Train Precision: 0.85481, Train Recall: 0.83600
Validation Accuracy: 0.91000, Validation Precision: 0.85481, Validation Recall: 0.83600.

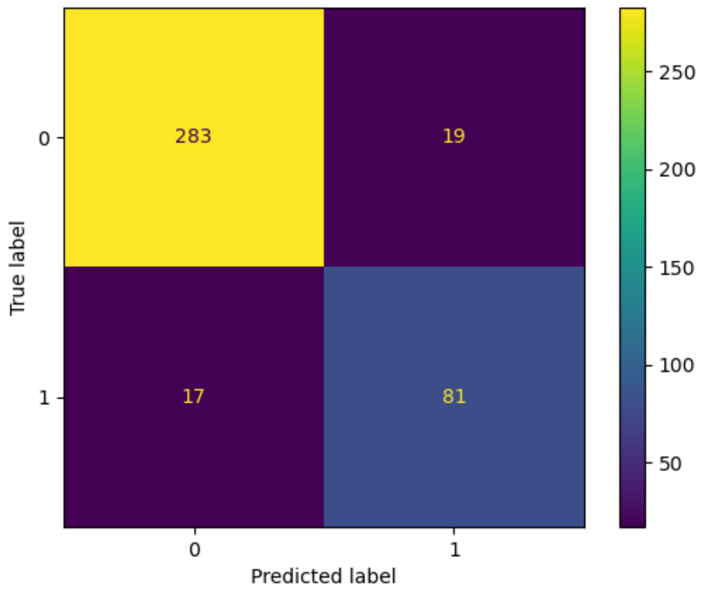

Figure 29: Confusion Matrix

Gradient Boosting is performed over 10 best features. Grid search performed over following parameters

`n_estimators: {20, 30, 40, 50}`
`sub_sample: {0.2, 0.3, 0.4, 0.5, 0.6}`
`max_depth: {5, 6, 7, 8, 9, 10}`.

Best parameters found after preforming grid search:

`n_estimators: 40`
`subsample: 0.3`
`max_depth: 7`

The reported values are:
Train Accuracy: 0.99500, Train Precision: 0.99797, Train Recall: 0.98200
Validation Accuracy: 0.96000, Validation Precision: 0.97727, Validation Recall: 0.86000

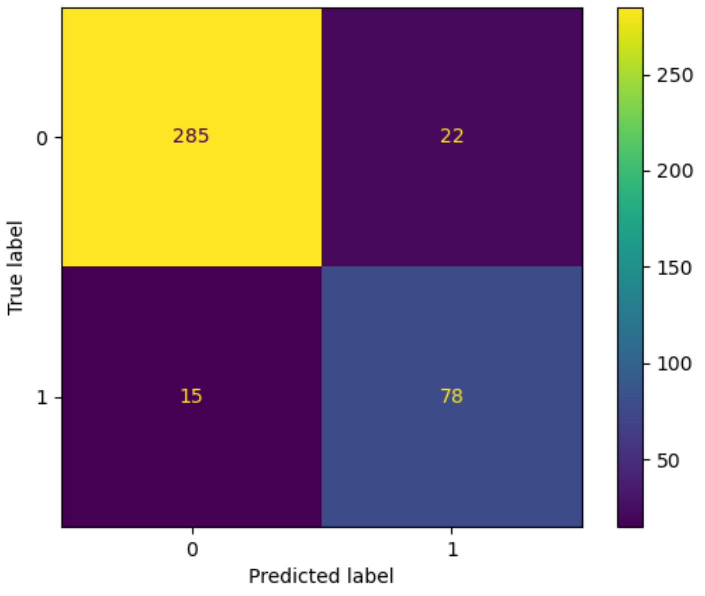

Figure 30: Confusion Matrix

1.6.2 Grid search over XGBoost classifier

The training and validation accuracy, precision, and recall values on default hyper-parameter values for 10 best features:
Train Accuracy: 1.00000, Train Precision: 1.00000, Train Recall: 1.00000
Validation Accuracy: 0.89000, Validation Precision: 0.80435, Validation Recall: 0.74000.

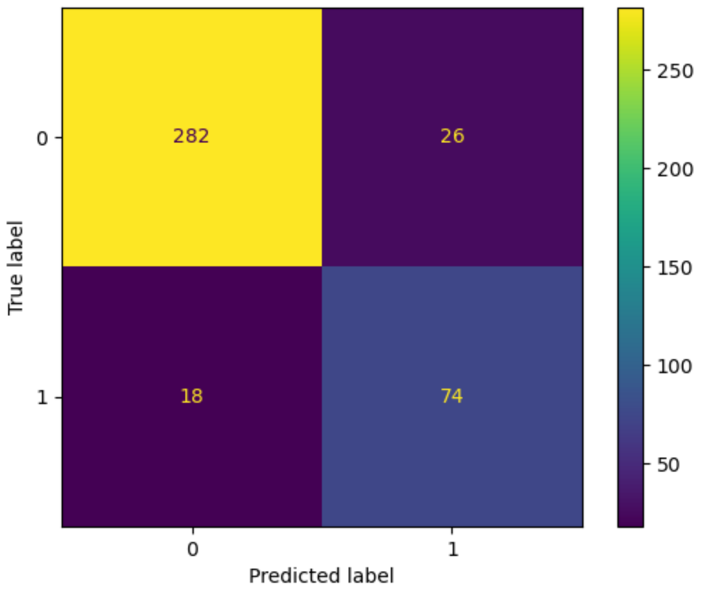

Figure 31: Confusion Matrix

Gradient Boosting is performed over 10 best features. Grid search performed over following parameters

`n_estimators: {20, 30, 40, 50}`

`sub_sample: {0.2, 0.3, 0.4, 0.5, 0.6}`

`max_depth: {5, 6, 7, 8, 9, 10}`.

Best parameters found after preforming grid search:

`n_estimators: 20`

`subsample: 0.6`

`max_depth: 5`

The reported values are:

Train Accuracy: 0.93650, Train Precision: 0.86931, Train Recall: 0.87800

Validation Accuracy: 0.90000, Validation Precision: 0.81250, Validation Recall: 0.78000

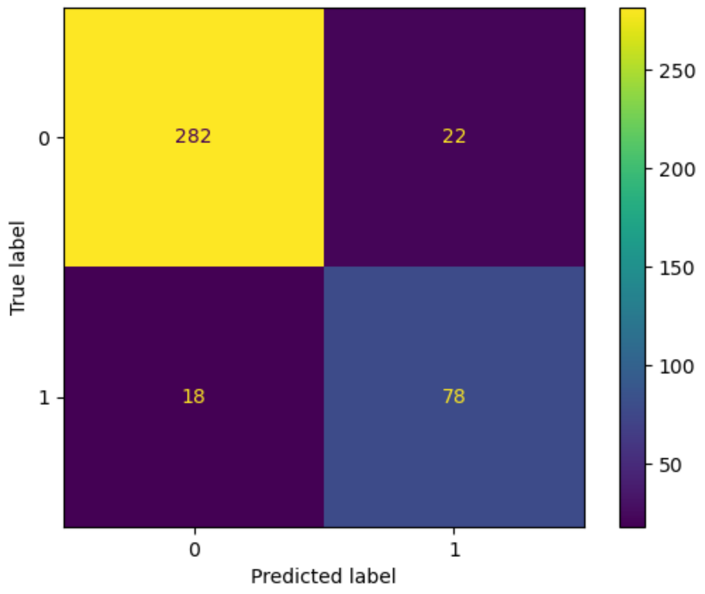

Figure 32: Confusion Matrix

The results when performed over **all features**.

The training and validation accuracy, precision, and recall values on default hyperparameter values:

Train Accuracy: 1.00000, Train Precision: 1.00000, Train Recall: 1.00000

Validation Accuracy: 0.98750, Validation Precision: 1.00000, Validation Recall: 0.95000.

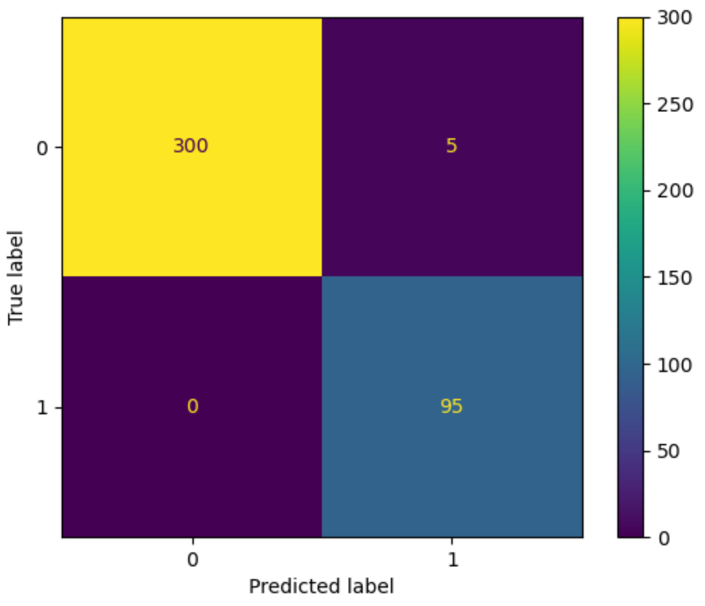

Figure 33: Confusion Matrix

Best parameters found after preforming grid search:

```
n_estimators: 50
subsample: 0.4
max_depth: 9
```

The reported values are:

Train Accuracy: 1.00000, Train Precision: 1.00000, Train Recall: 1.00000

Validation Accuracy: 0.98250, Validation Precision: 0.98947, Validation Recall: 0.94000

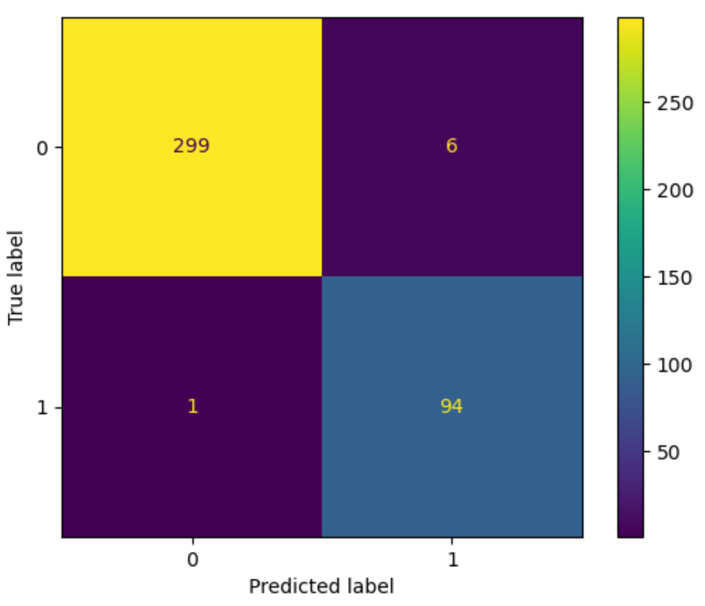

Figure 34: Confusion Matrix

2 Multiclass Classification

2.1 Decision Tree sklearn

2.1.1 Results on selection criteria as gini index

For, `depth` = 10 and `min_samples_split`= 7, reported accuracies and run-time are:

Time taken to build tree: 7.10759 seconds

Train Accuracy: 0.96900, Train Precision: 1.00000, Train Recall: 0.99383

Validation Accuracy: 0.74250, Validation Precision: 0.88608, Validation Recall: 0.93333

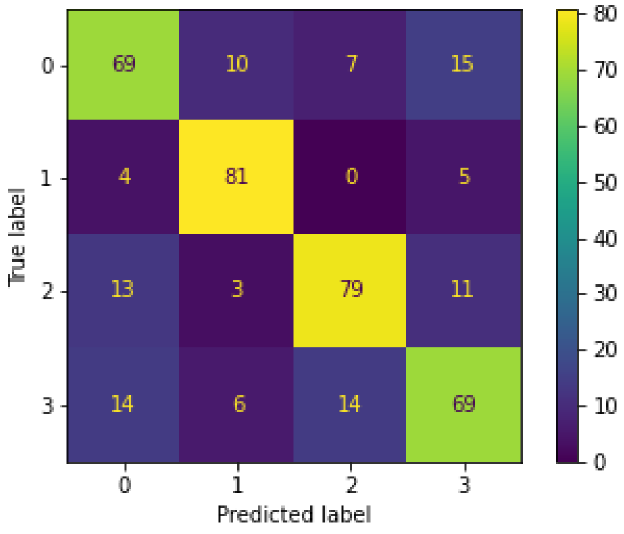

Figure 35: Confusion Matrix for `depth` = 10 and `min_samples_split` = 7

2.1.2 Results on selection criteria as entropy index

For, `depth` = 10 and `min_samples_split`= 7, reported accuracies and run-time are:

Time taken to build tree: 10.97066 seconds

Train Accuracy: 0.97100, Train Precision: 0.99795, Train Recall: 0.99591

Validation Accuracy: 0.73500, Validation Precision: 0.93506, Validation Recall: 0.94366

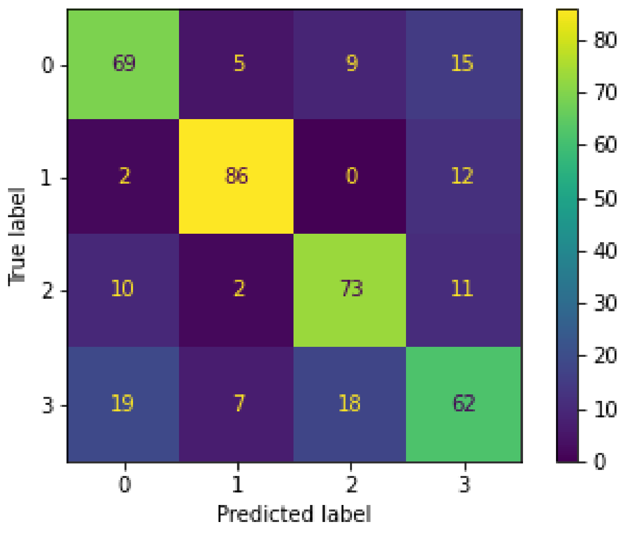

Figure 36: Confusion Matrix for `depth` = 10 and `min_samples_split` = 7

2.2 Decision Tree Grid Search and visualisation

2.2.1 Grid search on all feature

Best parameters found after preforming grid search: `criteria: entropy`
`max_depth: 18`
`min_samples_split: 7`

For, *depth* = 18 and `min_samples_split` = 7, reported accuracies and run-time are:
Time taken to build tree: 3.86105 *seconds*
Train Accuracy: 0.98200, Train Precision: 1.00000, Train Recall: 0.99389 Validation Accuracy: 0.71750, Validation Precision: 0.93056, Validation Recall: 0.94366

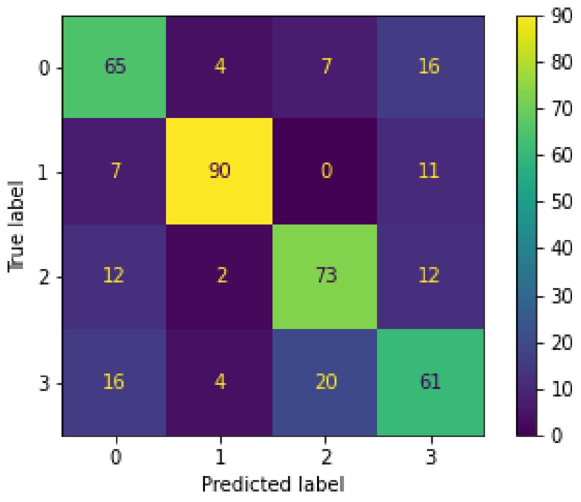

Figure 37: Confusion Matrix

2.2.2 Grid search on best 10 features

Best parameters found after preforming grid search: `criteria: entropy`
`max_depth: 7`
`min_samples_split: 4`

For, *depth* = 7 and `min_samples_split` = 4, reported accuracies and run-time are:
Time taken to build tree: 0.01590 *seconds*
Train Accuracy: 0.75600, Train Precision: 0.93231, Train Recall: 0.98614 Validation Accuracy: 0.61250, Validation Precision: 0.85417, Validation Recall: 0.96471

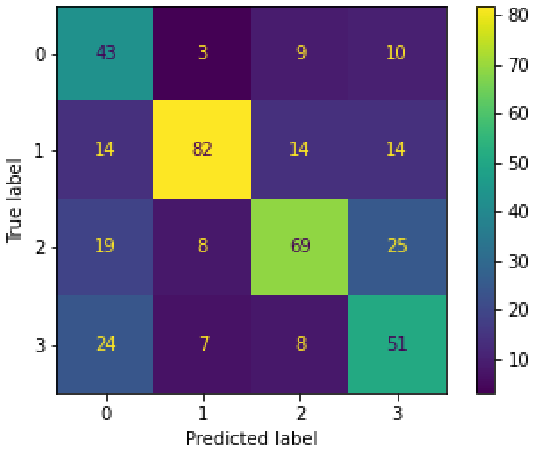

Figure 38: Confusion Matrix

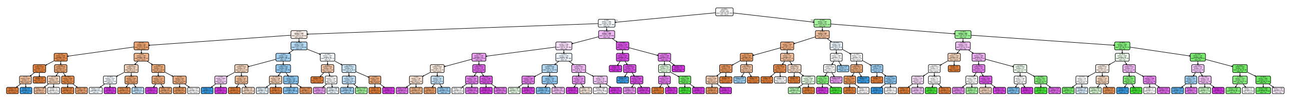

Figure 39: Decision Tree

2.3 Decision Tree Post Pruning with Cost Complexity Pruning

2.3.1 `criterion: "gini"`

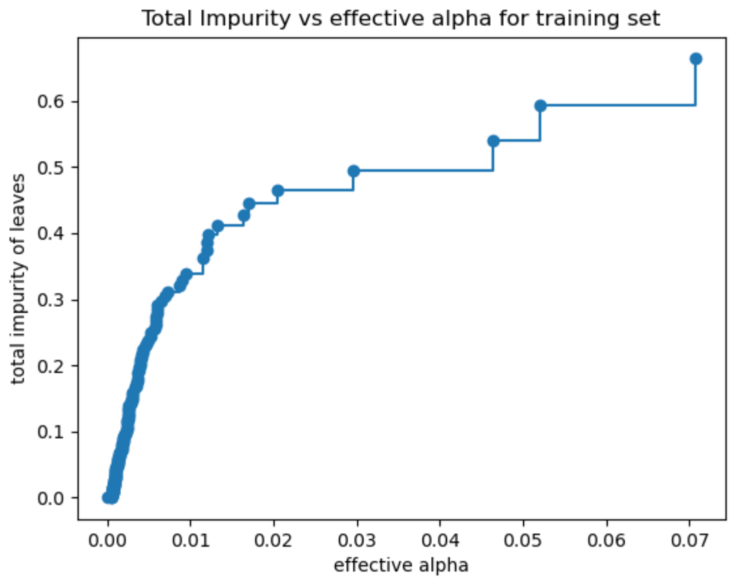

Figure 40: Total impurity of leaves vs the effective alphas of pruned tree

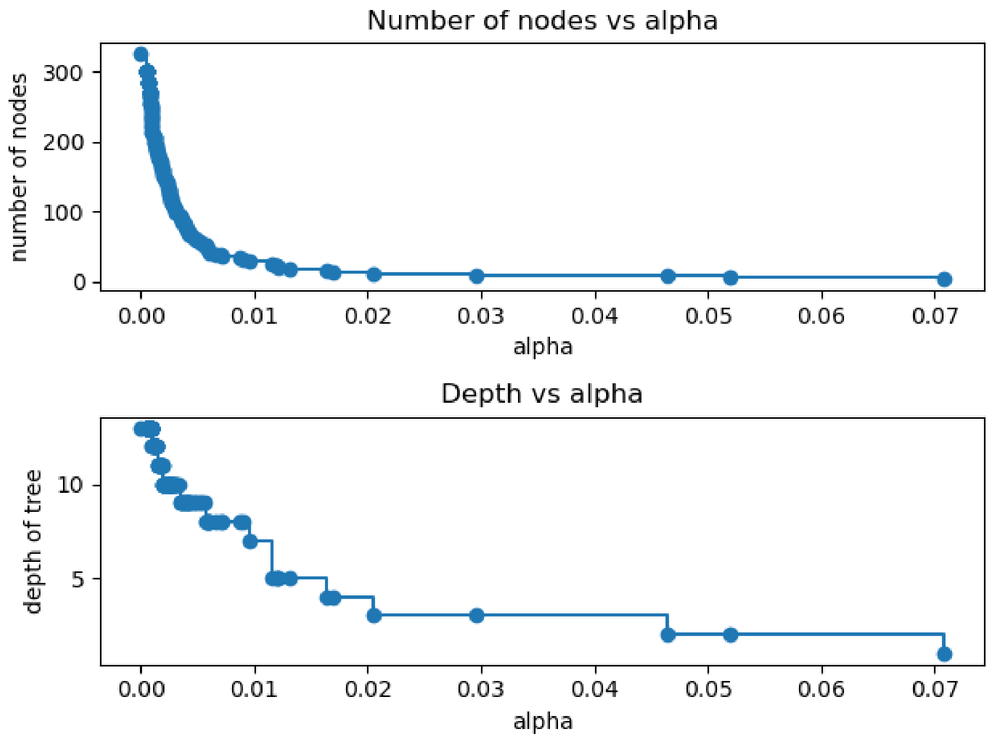

Figure 41: number of nodes vs alpha and the depth of the tree vs alpha

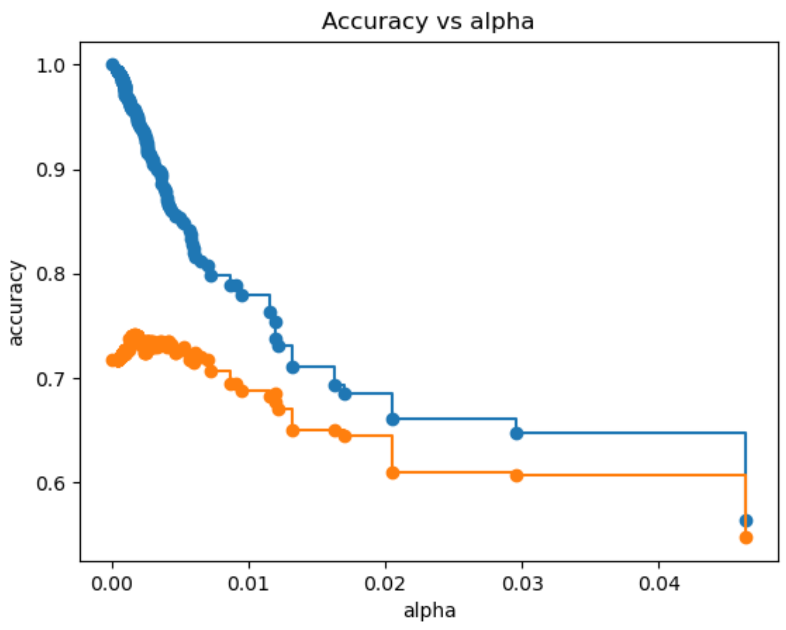

Figure 42: accuracy vs alpha

From these graphs, we can observe that leaf impurity increases with increase of alpha as the total number of nodes decreases which is due to pruning of tree and that also leads to decrease in depth of tree.

The optimal value for ccp_alphas is 0.00128571 and time taken to build tree for this value is 2.78565 seconds having:
Train Accuracy: 0.96550, Train Precision: 0.99588, Train Recall: 0.99180
Validation Accuracy: 0.73000, Validation Precision: 0.87179, Validation Recall: 0.93151

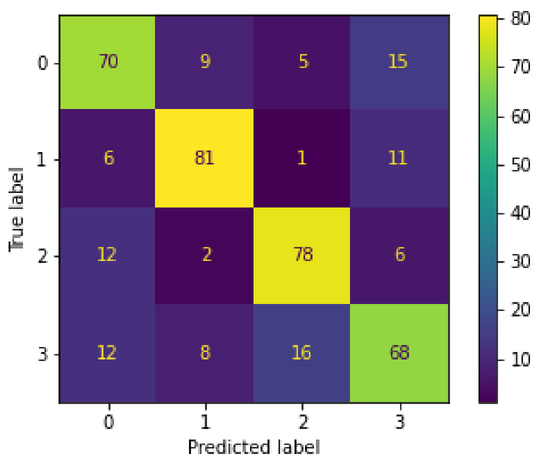

Figure 43: Confusion Matrix

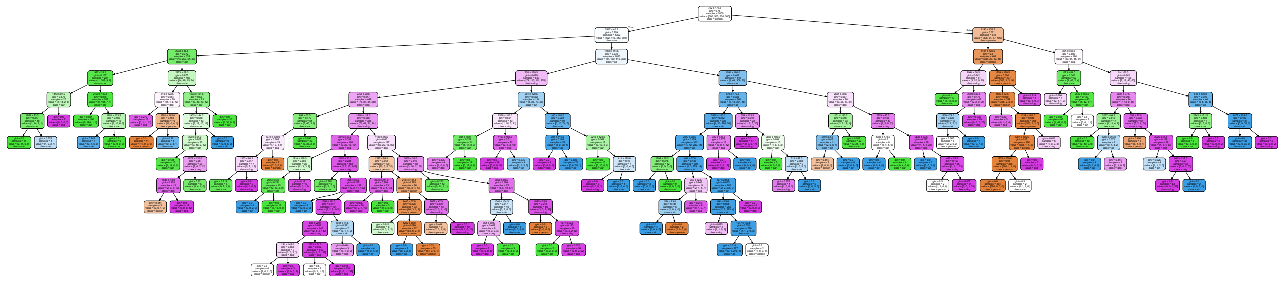

Figure 44: Best pruned tree

2.3.2 `criterion: "entropy"`

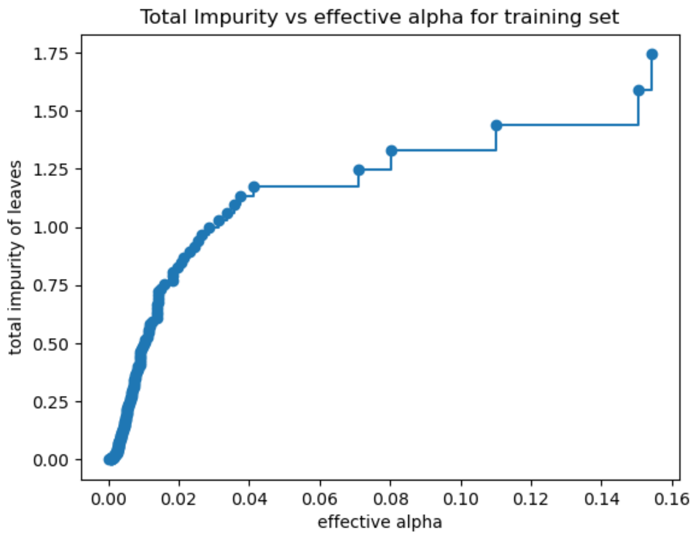

Figure 45: Total impurity of leaves vs the effective alphas of pruned tree

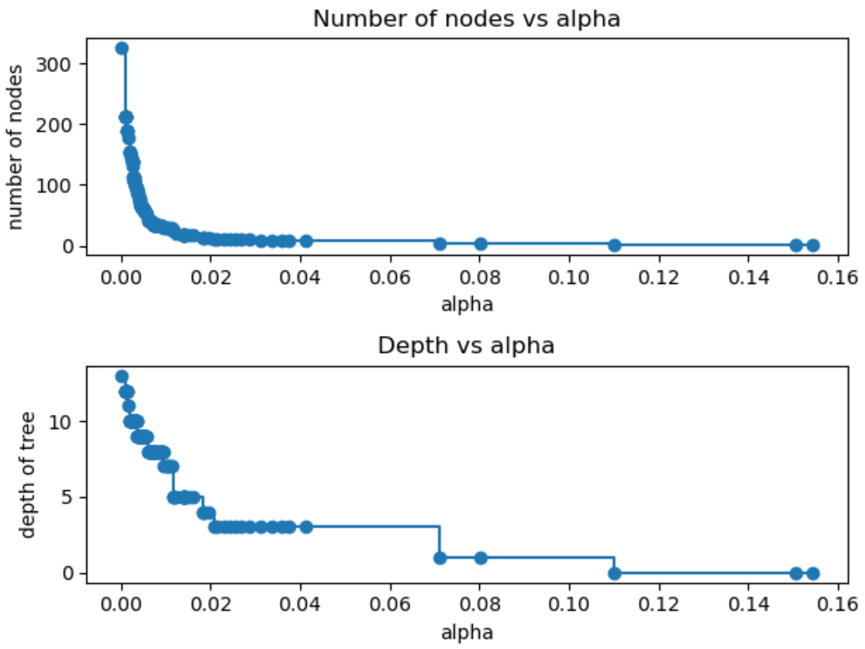

Figure 46: number of nodes vs alpha and the depth of the tree vs alpha

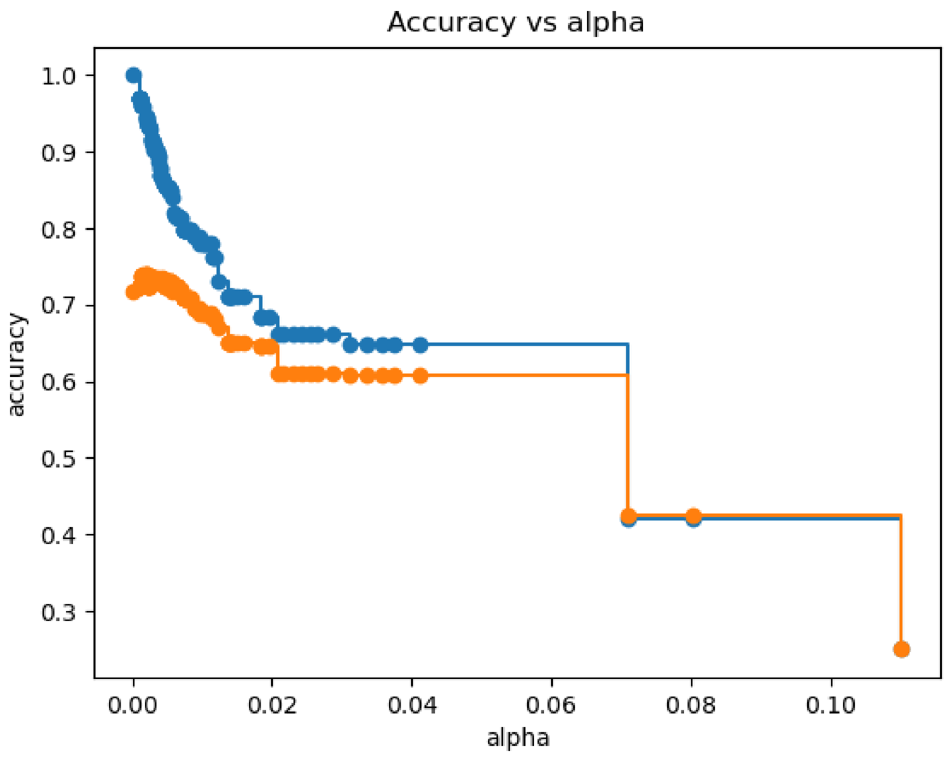

Figure 47: accuracy vs alpha

The optimal value for ccp_alphas is 0.00162255 and time taken to build tree for this value is 4.07668 seconds having:

Train Accuracy: 0.99350, Train Precision: 1.00000, Train Recall: 0.99799

Validation Accuracy: 0.73500, Validation Precision: 0.93243, Validation Recall: 0.92000

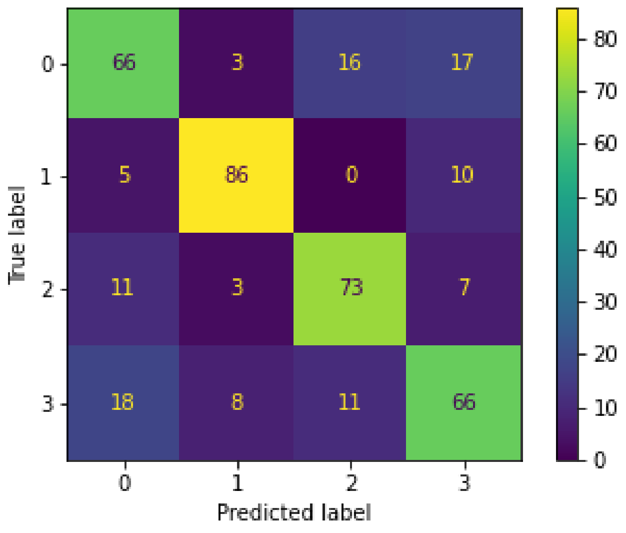

Figure 48: Confusion Matrix

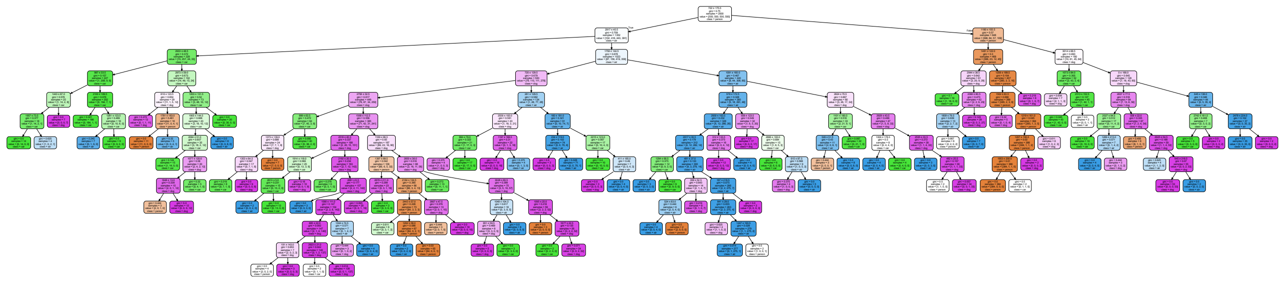

Figure 49: Best pruned tree

2.4 Random Forest

The training and validation accuracy, precision, and recall values on default hyper-parameter values:

Train Accuracy: 1.00000, Train Precision: 1.00000, Train Recall: 1.00000

Validation Accuracy: 0.87500, Validation Precision: 0.97826, Validation Recall: 0.98901.

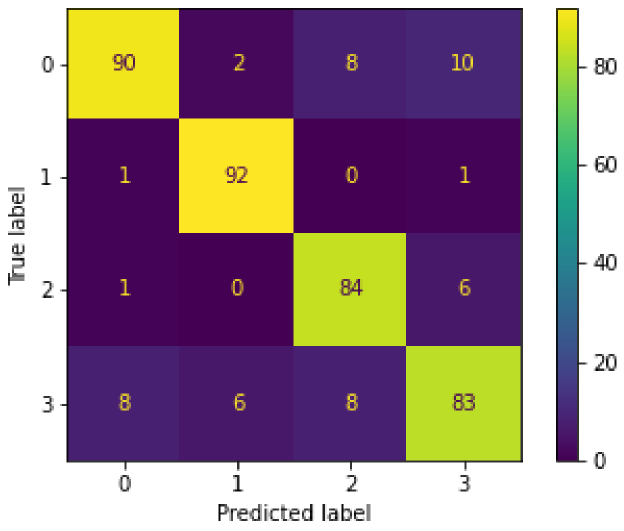

Figure 50: Confusion Matrix

. Grid Search performed over following parameters:

`n_estimators: {80, 100, 150, 200}`

`criterion: {‘gini’, ‘entropy’}`

`max_depth: {None, 5, 7, 10}`

`min_samples_split: {5, 7, 10}`.

Best parameters found after preforming grid search:

`n_estimators: 200`

`criteria: entropy`

`max_depth: None`

`min_samples_split: 5`

The reported values are:
Train Accuracy: 1.00000, Train Precision: 1.00000, Train Recall: 1.00000
Validation Accuracy: 0.88750, Validation Precision: 0.97917, Validation Recall: 0.98947

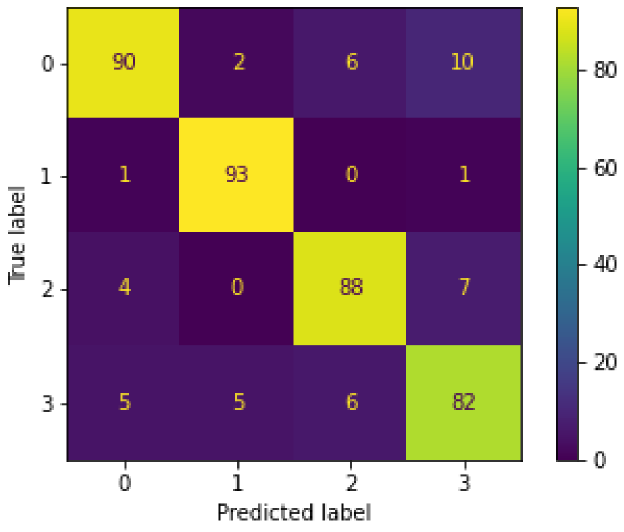

Figure 51: Confusion Matrix

2.5 Gradient Boosted Trees and XGBoost

2.5.1 Gradient Boosted Trees

The training and validation accuracy, precision, and recall values on default hyper-parameter values:
Train Accuracy: 1.00000, Train Precision: 1.00000, Train Recall: 1.00000
Validation Accuracy: 0.89000, Validation Precision: 0.98901, Validation Recall: 1.00000.

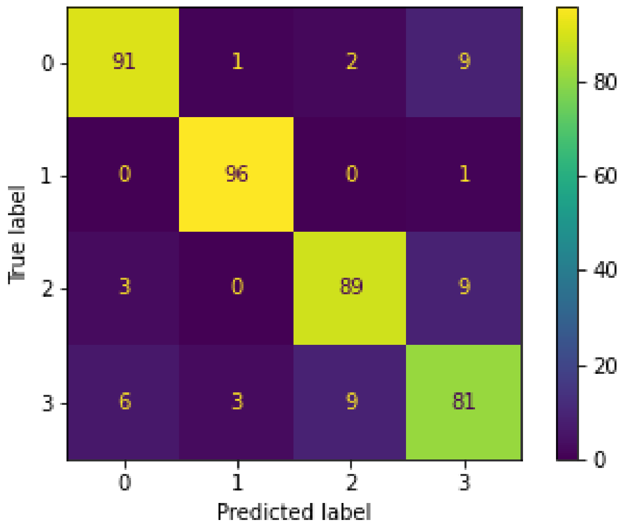

Figure 52: Confusion Matrix

Gradient Boosting is performed over all features. Grid search performed over following parameters
`n_estimators`: {20, 30, 40, 50}
`sub_sample`: {0.2, 0.3, 0.4, 0.5, 0.6}
`max_depth`: {5, 6, 7, 8, 9, 10}.

Best parameters found after preforming grid search:
`n_estimators`: 50
`subsample`: 0.6
`max_depth`: 6

The reported values are:
Train Accuracy: 1.00000, Train Precision: 1.00000, Train Recall: 1.00000
Validation Accuracy: 0.89000, Validation Precision: 0.98947, Validation Recall: 1.00000

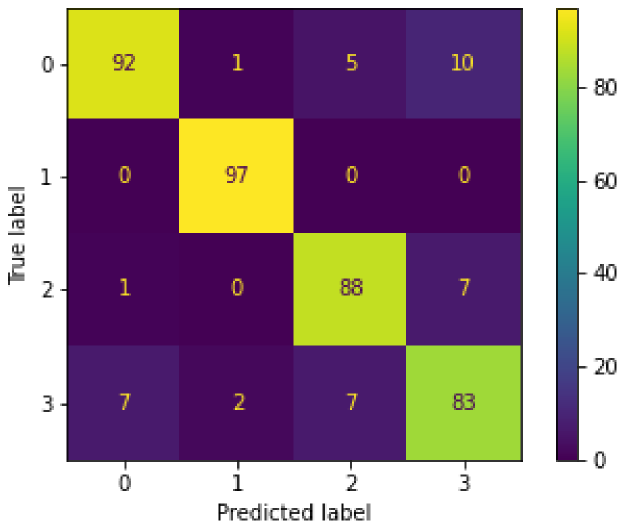

Figure 53: Confusion Matrix

2.5.2 XGBoost

The training and validation accuracy, precision, and recall values on default hyper-parameter values:
Train Accuracy: 1.00000, Train Precision: 1.00000, Train Recall: 1.00000
Validation Accuracy: 0.90250, Validation Precision: 0.98936, Validation Recall: 1.00000.

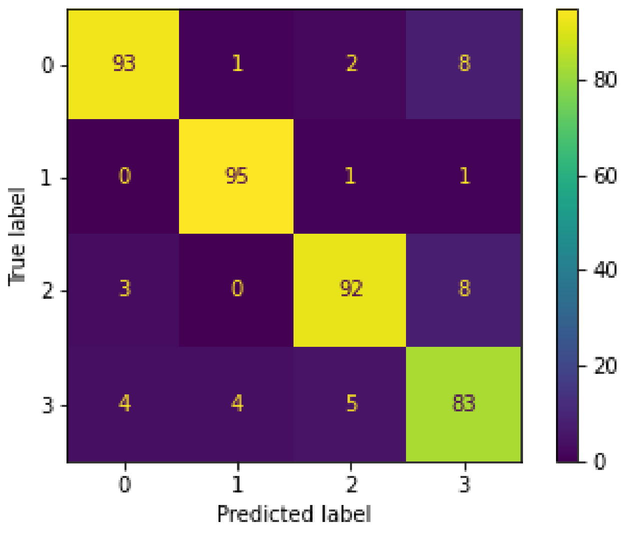

Figure 54: Confusion Matrix

Best parameters found after preforming grid search:
`n_estimators: 50`
`subsample: 0.4`
`max_depth: 8`

The reported values are:
Train Accuracy: 1.00000, Train Precision: 1.00000, Train Recall: 1.00000
Validation Accuracy: 0.90750, Validation Precision: 0.98925, Validation Recall: 0.98925

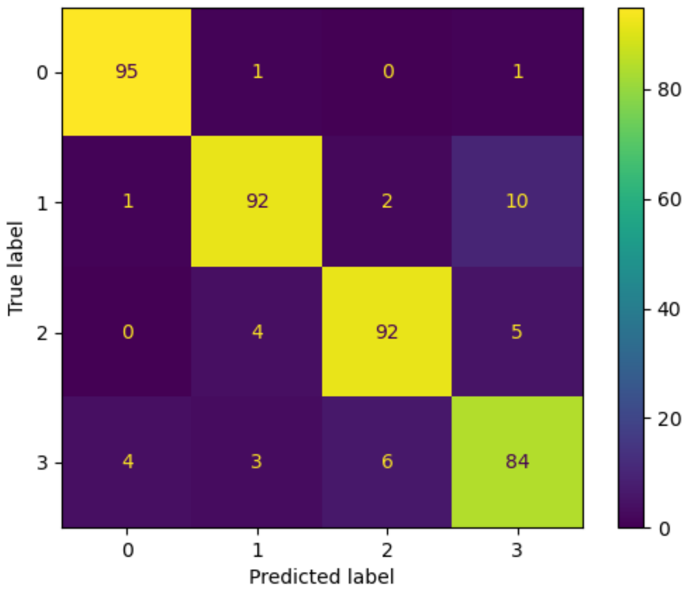

Figure 55: Confusion Matrix

2.6 Real-time Application

With the sklearn decision classifier having parameters as `depth` = 10, `min_sample_split` = 7 and criterion as entropy, the reported accuracy is 40%.

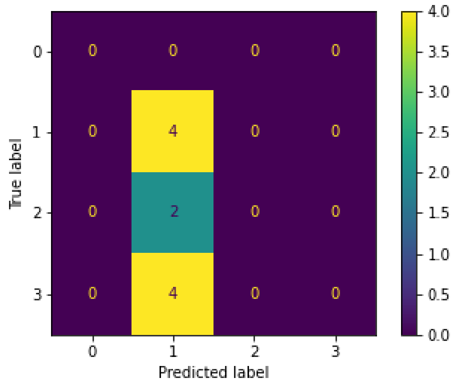

Figure 56: Confusion Matrix

Using XGBoost classifier having parameters as `max_depth` = 8, `n_estimators` = 50 and `sub_sample` = 0.4, the reported accuracy is 60%.

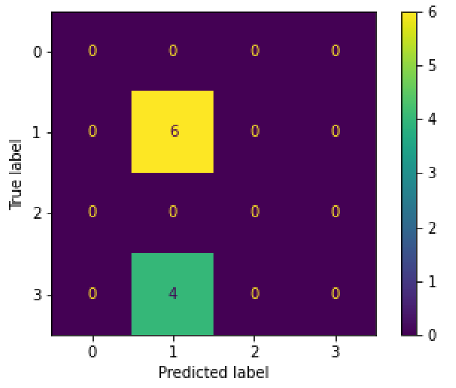

Figure 57: Confusion Matrix

The results of the image classification task on faces showed lower accuracy than expected. One possible explanation for this could be the bias in the training dataset, which has a higher proportion of white faces with short or white hair, while Indian faces have black hair. This difference in hair color and length could be a significant factor in misclassifying some Indian faces. Furthermore, the model misclassified some faces as dogs, as they have higher amount of hair on their bodies compared to human faces so training data get baised over feature of hair. So human face is misclassified if there is significant hair in image. Overall, these results highlight the importance of having a diverse and representative training dataset to avoid biases in image classification tasks.