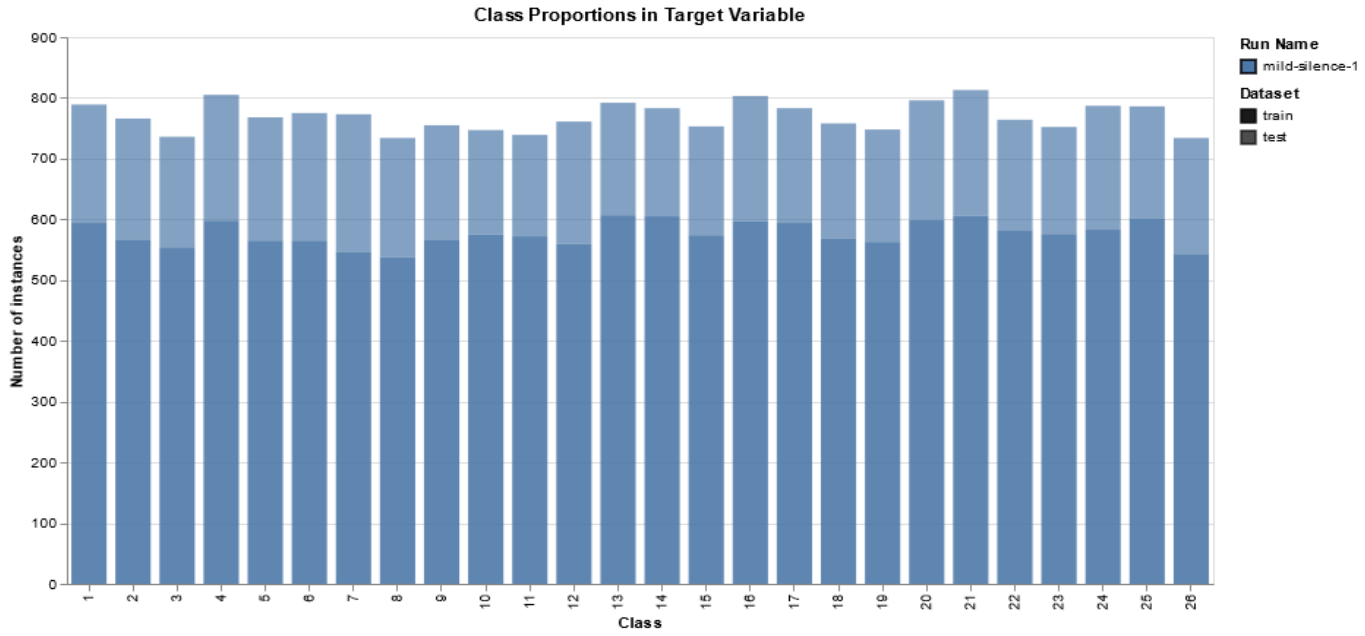
Comp 7118- Assignment C3
Prachi Jadhav

TASK 2: Comparison of Classifiers on multiclass classification.
I used wandb platform to visualize plots.
Disclaimer: I did not use seed so the results are variational.

The data statistics:

data	size	# of features
Training	15,000	16
testing	5,000	16

Class_proportion_plot:
Given plot shows the distribution of target classes in training and test sets. I believe modeling target won't be an issue, apparently, classes are equally distributed.



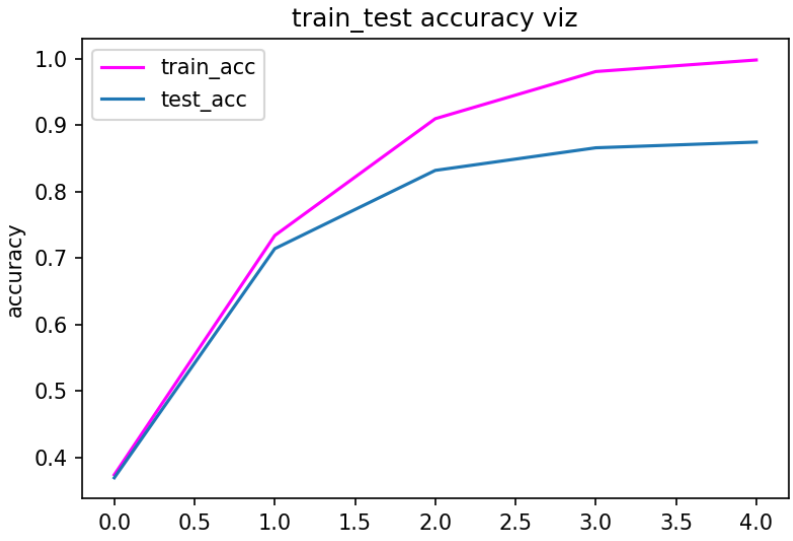
Preprocessing:

- Decision trees don't require scaling the data.
- No null-values were found, so we can go ahead
- with the split is 0.66/0.33

Decision Tree:

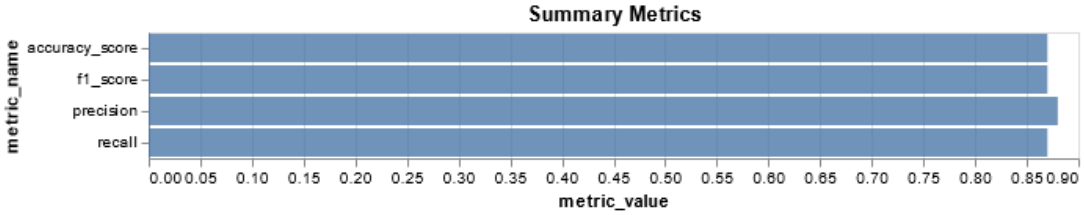
Classifier parameters: for different max_depth (height of the tree) the accuracies are shown below.

1. Visualizing model performance for different max_depth:

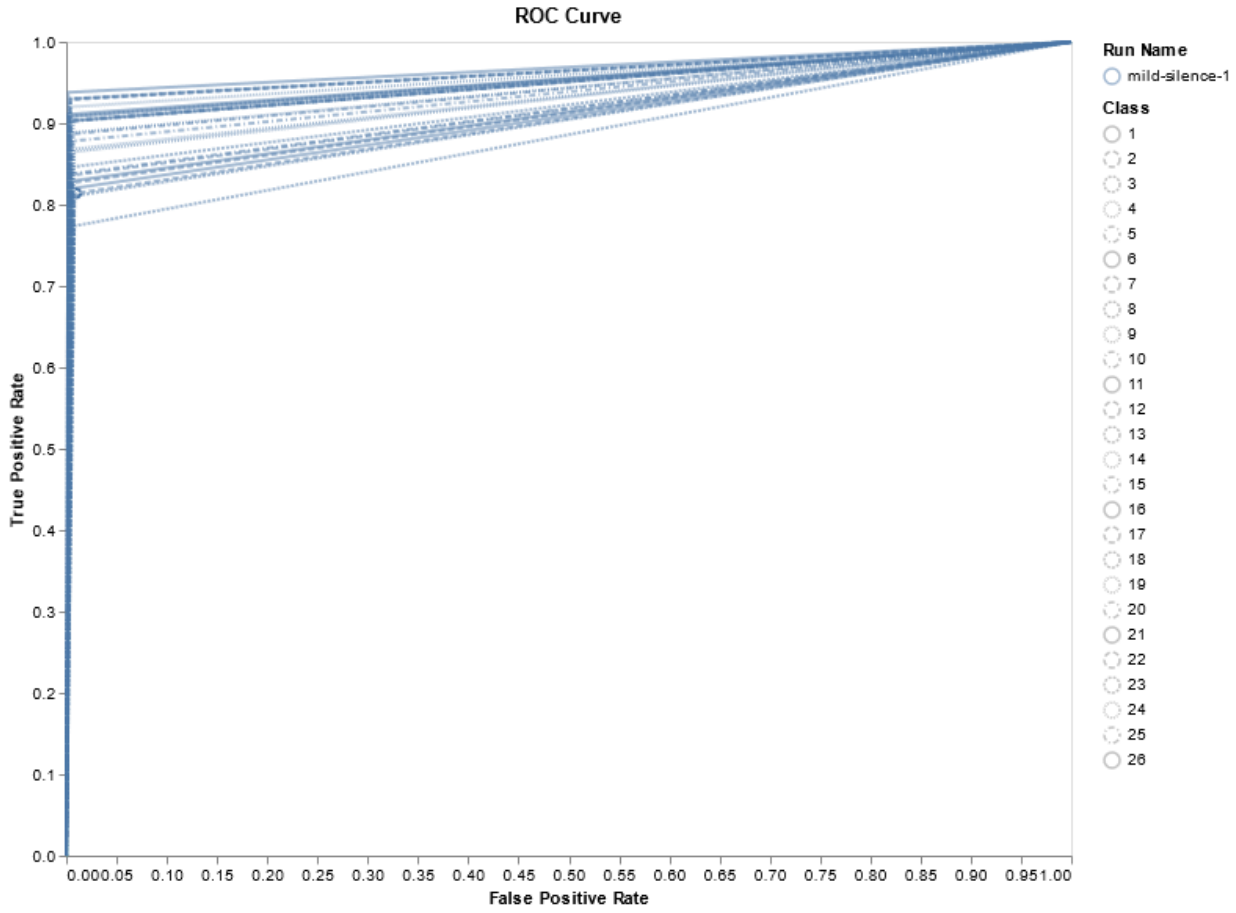


Best model(max_depth = 25, high accuracy):

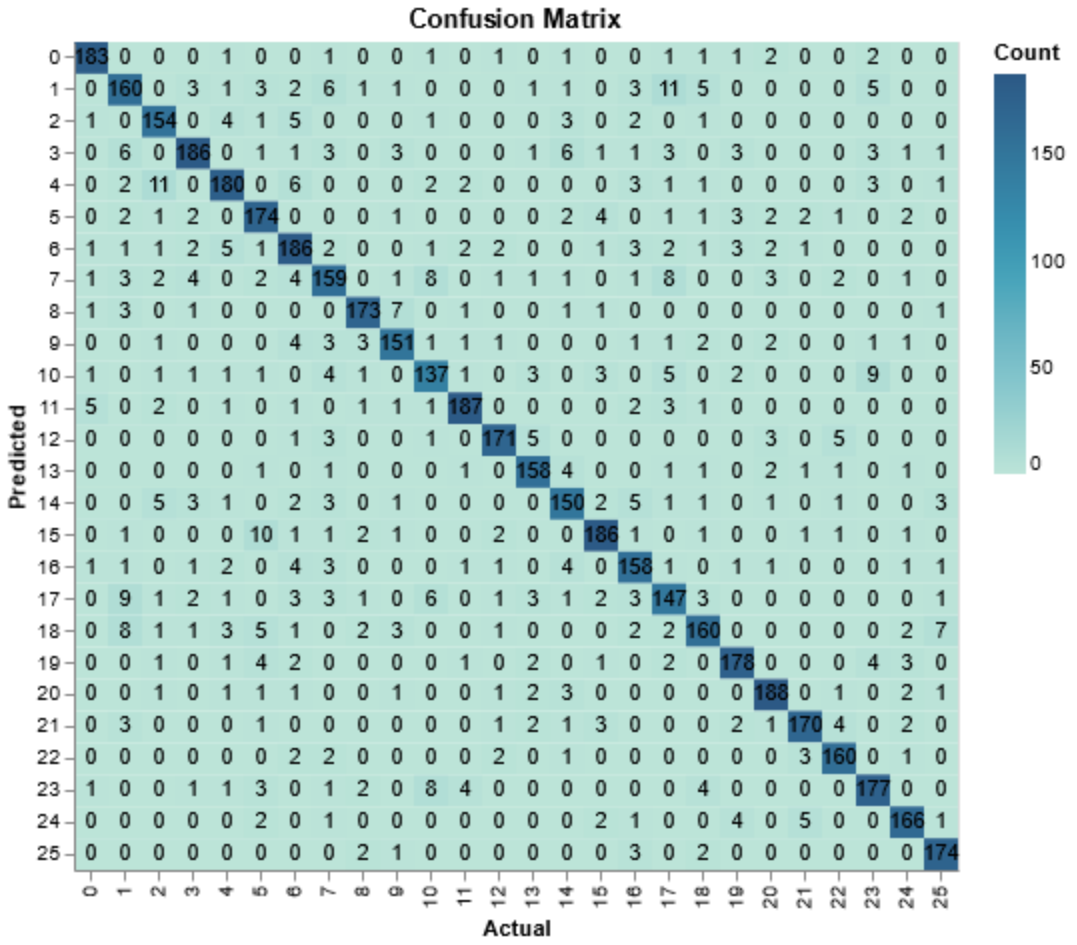
Summary metrics: here we can see accuracy, f1, precision, recall for the classification task. Precision is the highest.



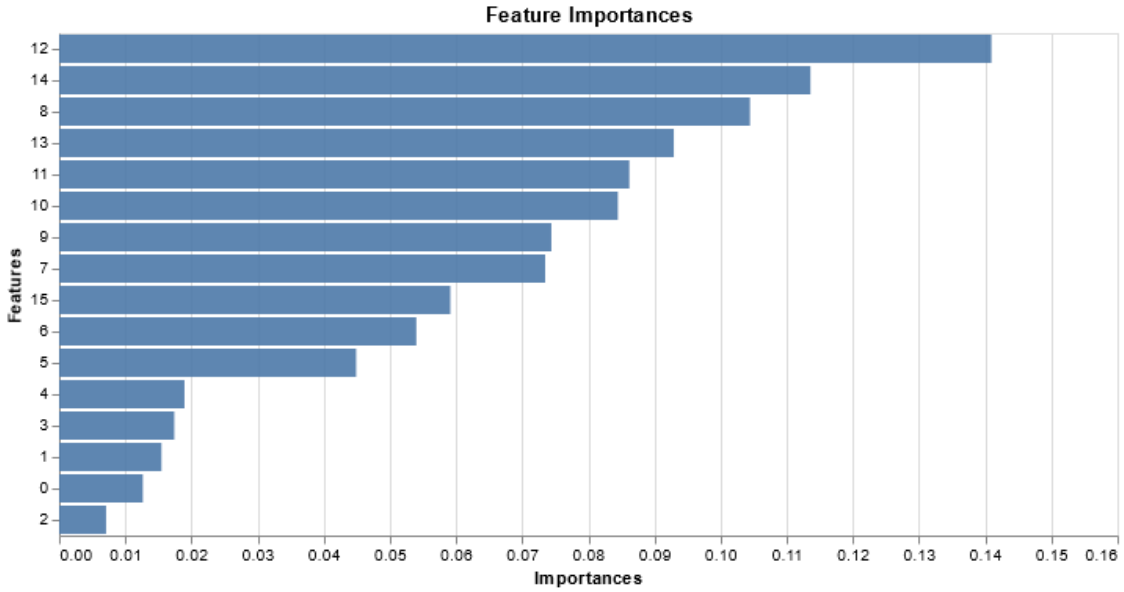
ROC curves plot true positive rate(y-axis) vs false positive rate(x-axis) for each class. The ideal score is a TPR=1 and FPR=0, which is the point on the top left. The greater the AUC-ROC the better.



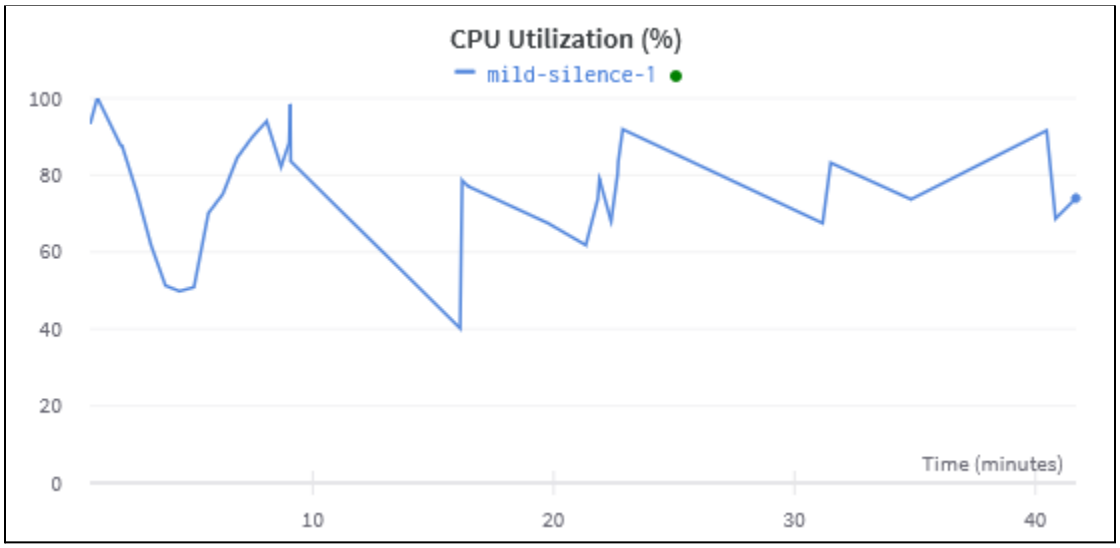
The following plot is the confusion matrix to evaluate the accuracy of classification. Intuitively, we can able to check the quality of model predictions and find patterns in the predictions the model gets wrong. The darker cell —> accurately classified.



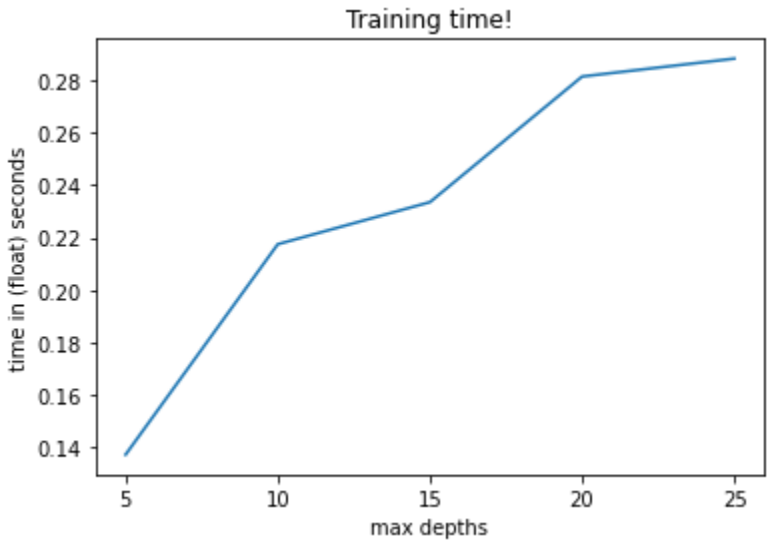
Given plot, plots the importance of each feature for the classification task. Available for trees only. Seems target is influenced mostly by feature_12, feature_14,...



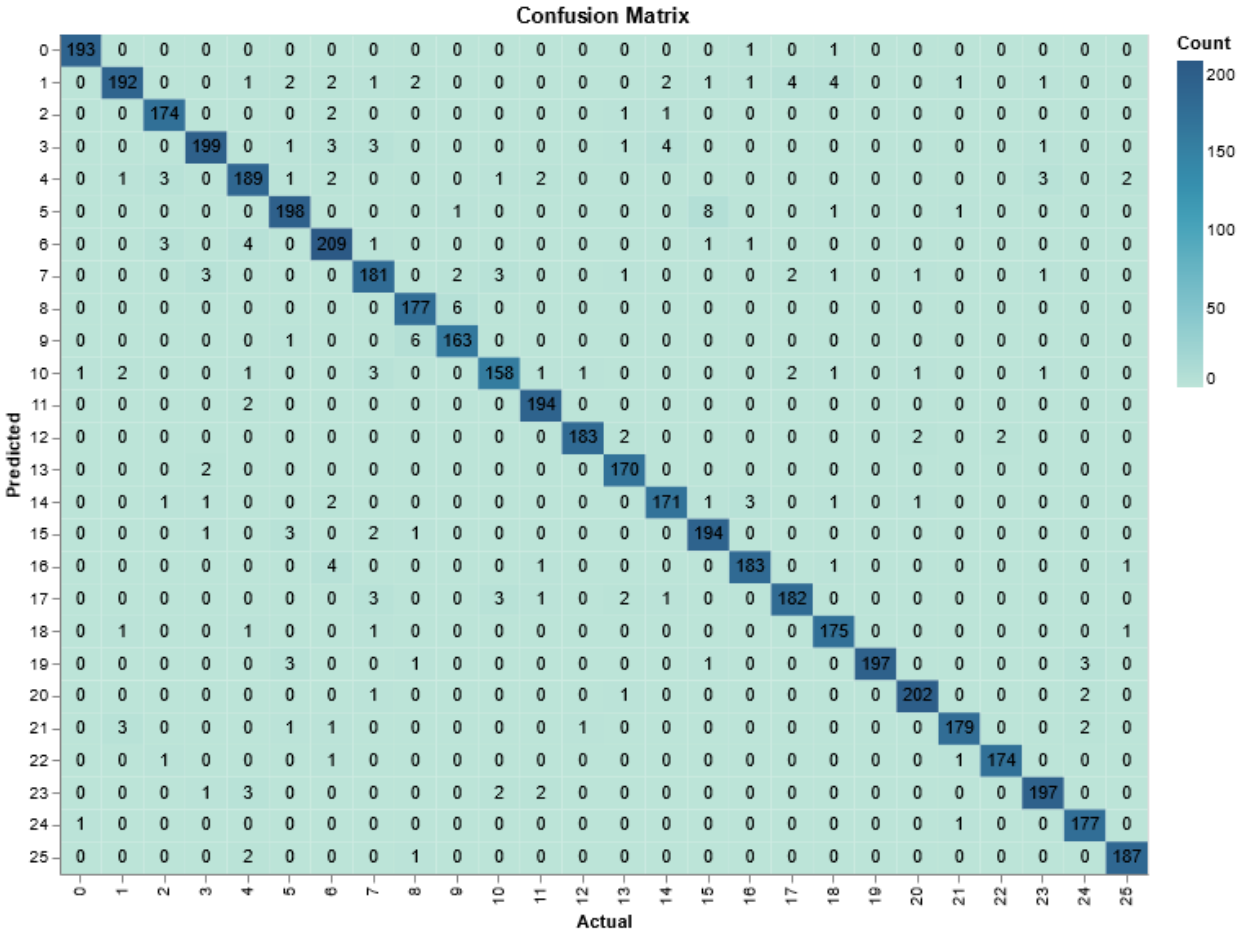
For the overall decision tree experiment, the CPU utilization is (including some running gaps):



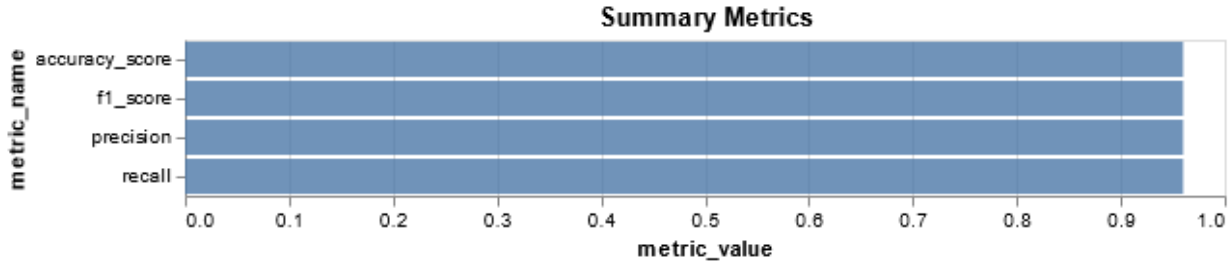
Training_time VS max_Depth:
It's obvious that the greater the max_depth the greater the training time because that many decision boundary gets added.

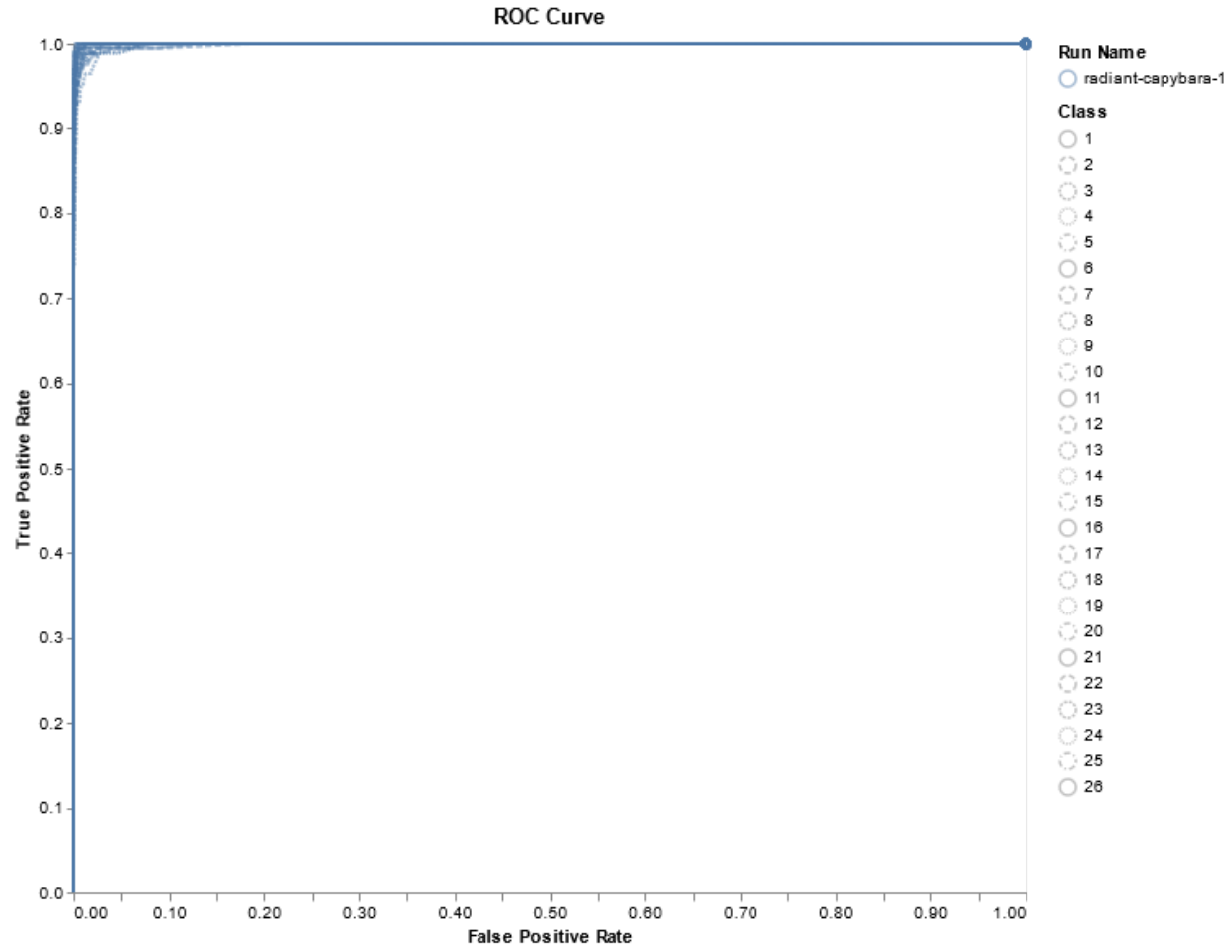


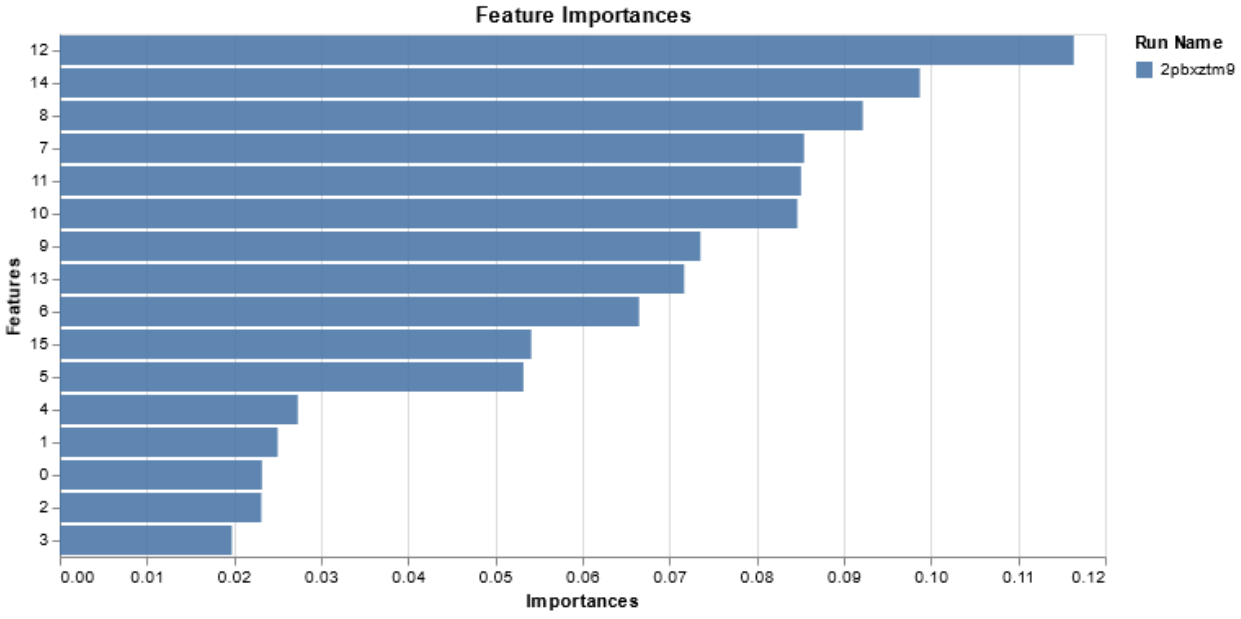
A decision tree with criterion entropy
Confusion matrix, we can see the good quality of predictions:



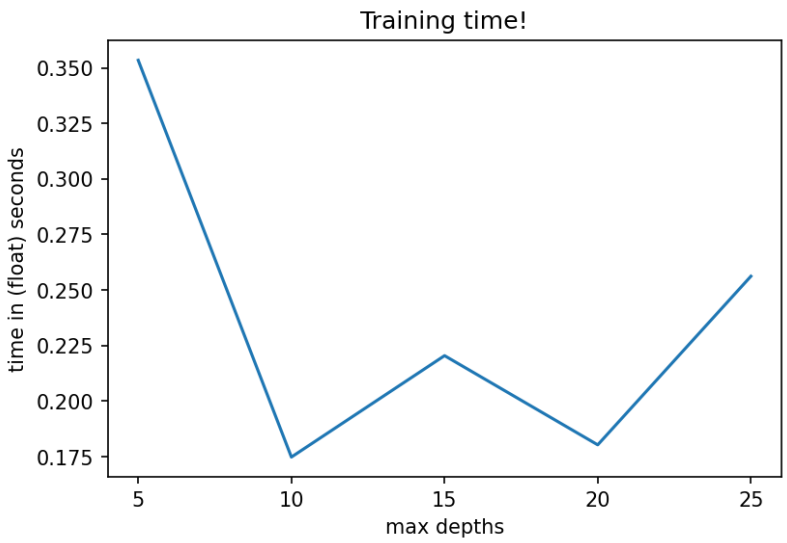
All metrics are better than gini criterion model:



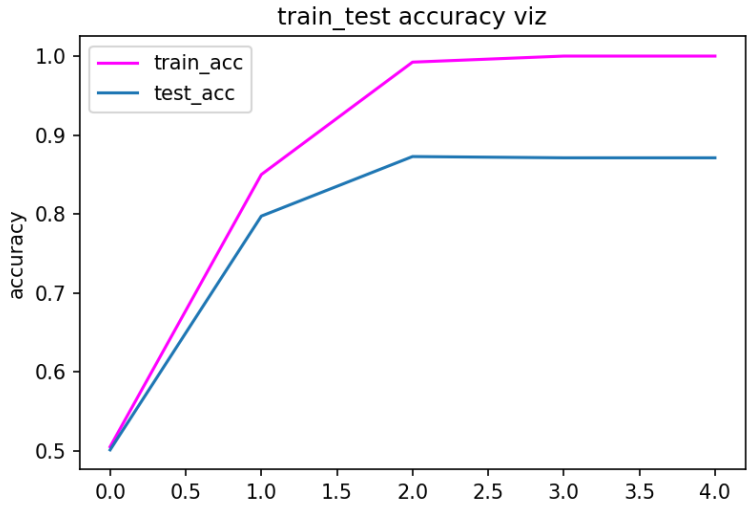




The training time is quietly fluctuating with max_depth, apparently, it's low for high max_depth.

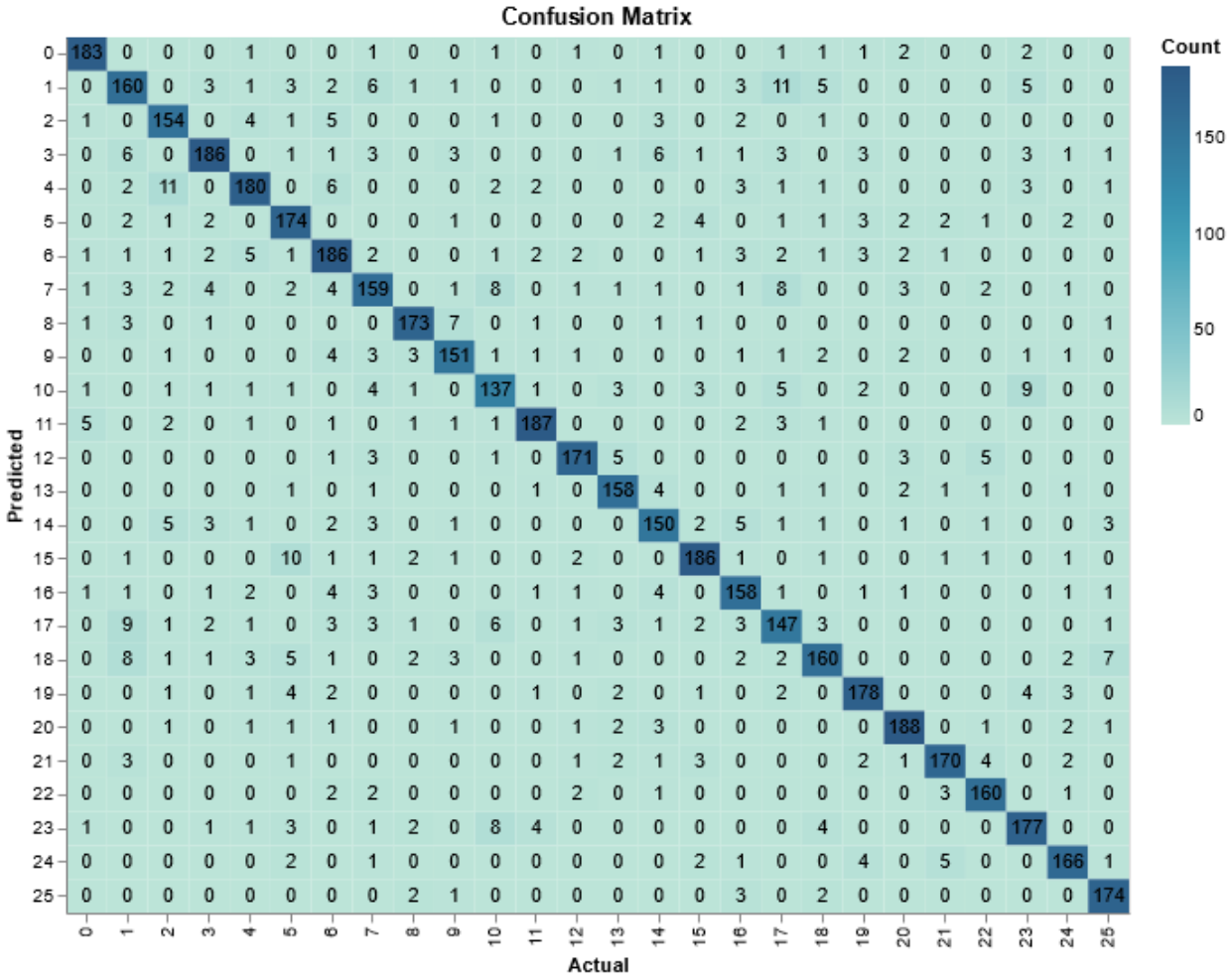


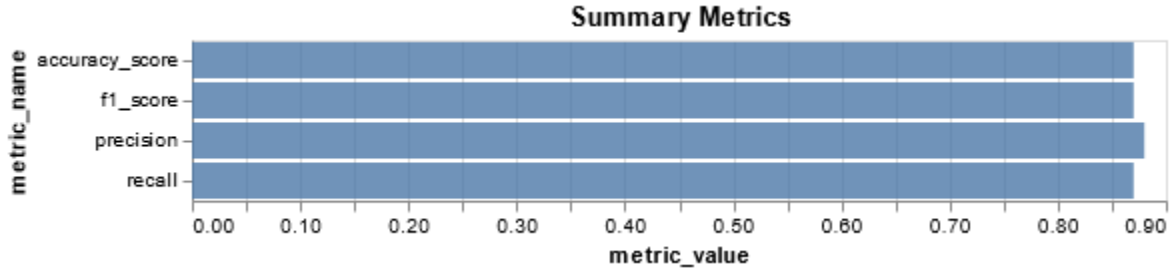
The training and testing accuracy are both stabilizing over the period.

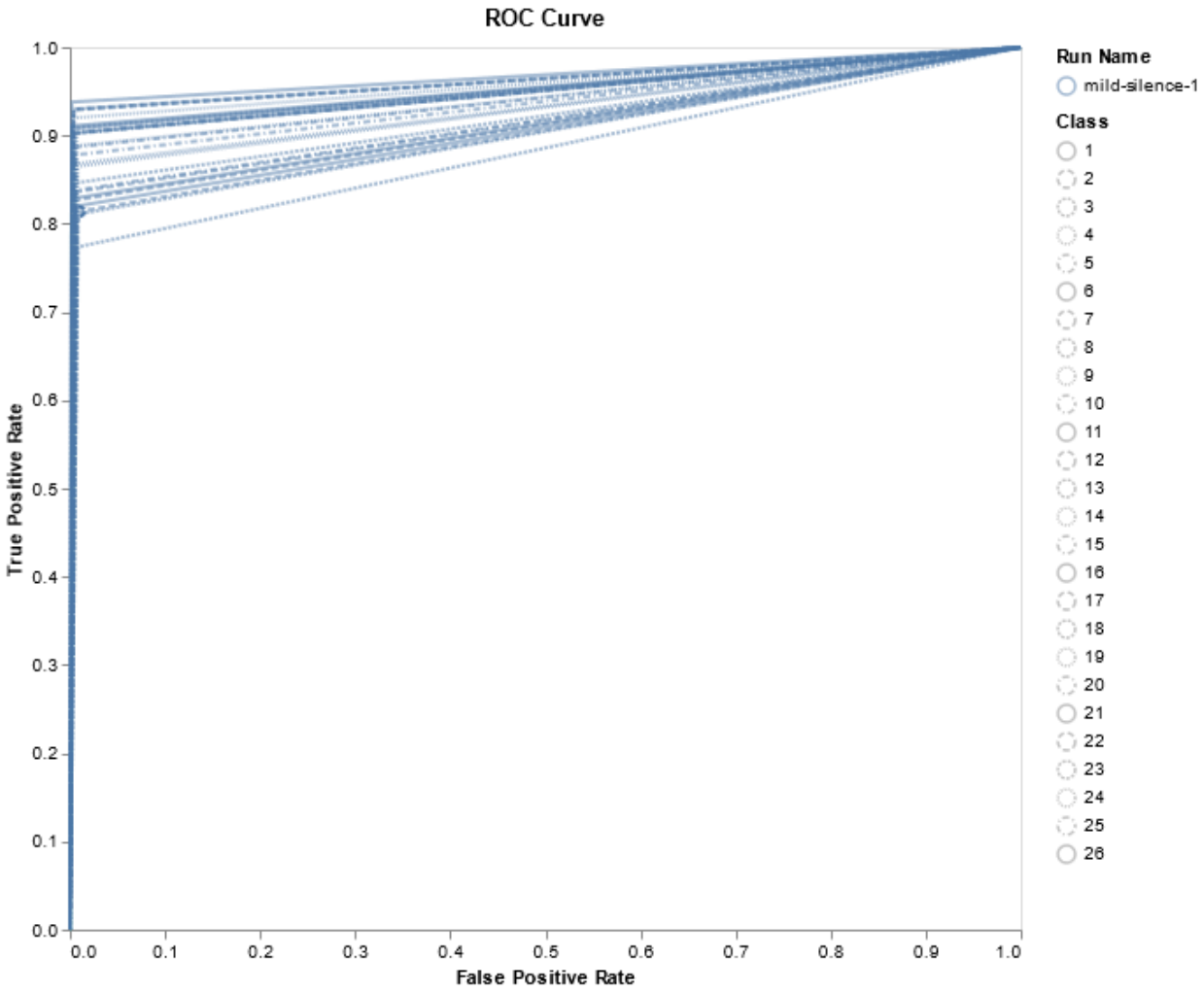


KNN Classifier performance

Here are some plots for KNN: n_neighbors = 5 (explained in a code):


Confusion Matrix
Predicted
Actual
Count
150
100
50
0

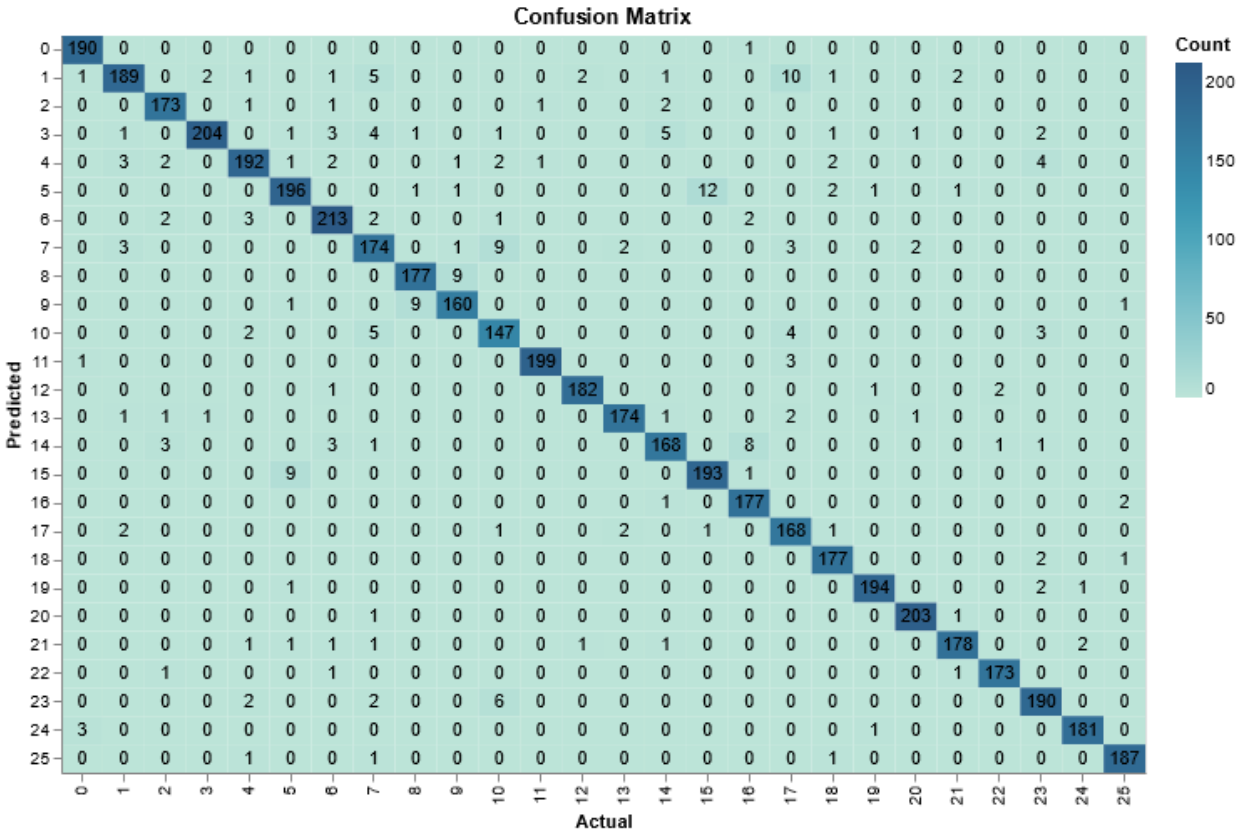

Summary Metrics
metric_name
accuracy_score
f1_score
precision
recall
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
metric_value

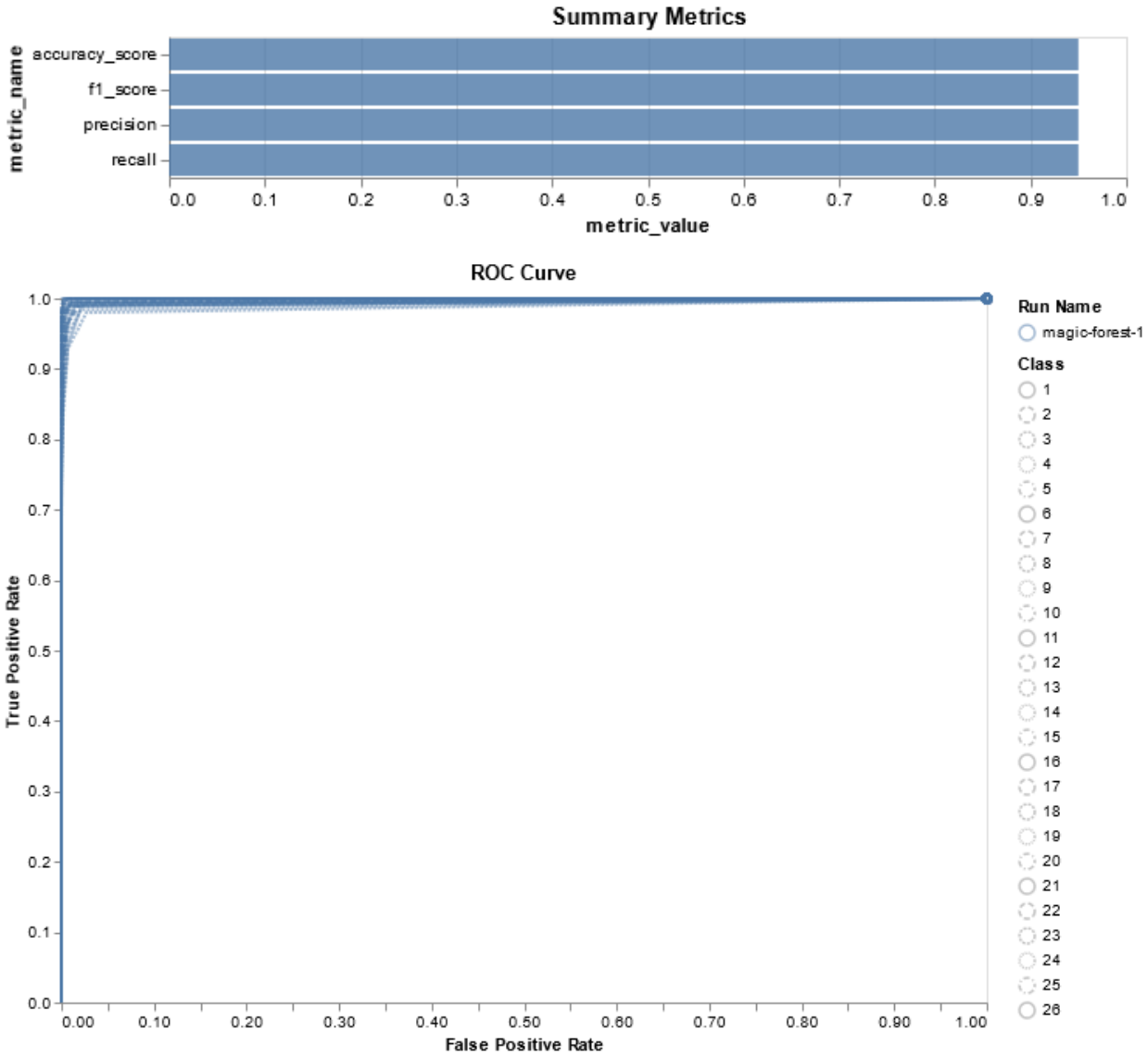


Hyperparameter tuning: best parameters are shown of knn, looks like n_neighbors=5 is the best choice

```
training_time(sec) for grid search  39.60739779472351
best accuracy:{} 0.9415333333333334
best estimator:{} KNeighborsClassifier(metric='manhattan', weights='distance')
best params:{} {'metric': 'manhattan', 'n_neighbors': 5, 'weights': 'distance'}
```

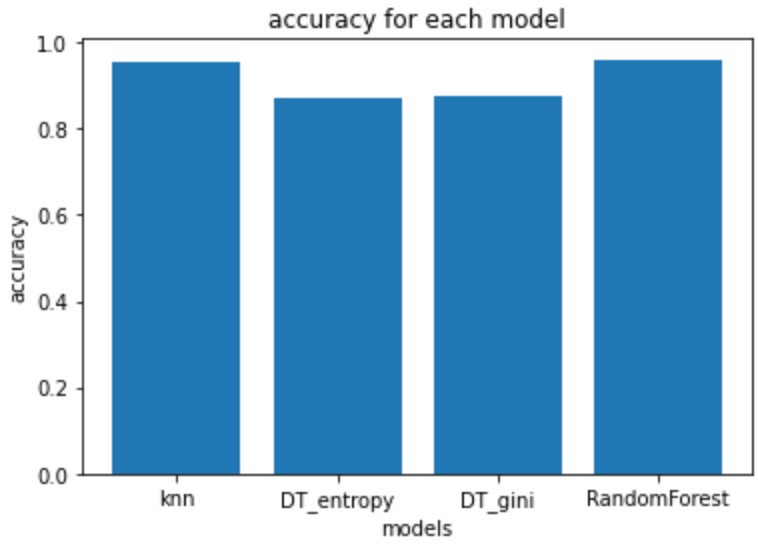
Random Forest Classifier performance
Following are the plots for 100 trees in a random forest:


Confusion Matrix
Predicted
Actual
Count
200
150
100
50
0
0 190 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
1 1 189 0 2 1 0 1 5 0 0 0 0 2 0 1 0 0 10 1 0 0 2 0 0 0 0
2 0 0 173 0 1 0 1 0 0 0 0 1 0 0 2 0 0 0 0 0 0 0 0 0 0 0
3 0 1 0 204 0 1 3 4 1 0 1 0 0 0 5 0 0 0 1 0 1 0 0 2 0 0
4 0 3 2 0 192 1 2 0 0 1 2 1 0 0 0 0 0 0 2 0 0 0 0 4 0 0
5 0 0 0 0 0 196 0 0 1 1 0 0 0 0 0 12 0 0 2 1 0 1 0 0 0 0
6 0 0 2 0 3 0 213 2 0 0 1 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0
7 0 3 0 0 0 0 0 174 0 1 9 0 0 2 0 0 0 3 0 0 2 0 0 0 0 0
8 0 0 0 0 0 0 0 0 177 9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
9 0 0 0 0 0 1 0 0 9 160 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
10 0 0 0 0 2 0 0 5 0 0 147 0 0 0 0 0 0 4 0 0 0 0 0 3 0 0
11 1 0 0 0 0 0 0 0 0 0 0 199 0 0 0 0 0 3 0 0 0 0 0 0 0 0
12 0 0 0 0 0 0 1 0 0 0 0 0 182 0 0 0 0 0 0 1 0 0 2 0 0 0
13 0 1 1 1 0 0 0 0 0 0 0 0 0 174 1 0 0 2 0 0 1 0 0 0 0 0
14 0 0 3 0 0 0 3 1 0 0 0 0 0 0 168 0 8 0 0 0 0 0 1 1 0 0
15 0 0 0 0 0 9 0 0 0 0 0 0 0 0 0 193 1 0 0 0 0 0 0 0 0 0
16 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 177 0 0 0 0 0 0 0 0 2
17 0 2 0 0 0 0 0 0 0 0 1 0 0 2 0 1 0 168 1 0 0 0 0 0 0 0
18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 177 0 0 0 0 2 0 1
19 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 194 0 0 0 2 1 0
20 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 203 1 0 0 0 0
21 0 0 0 0 1 1 1 1 0 0 0 0 1 0 1 0 0 0 0 0 0 178 0 0 2 0
22 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 173 0 0 0
23 0 0 0 0 2 0 0 2 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0 190 0 0
24 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 181 0
25 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 187
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

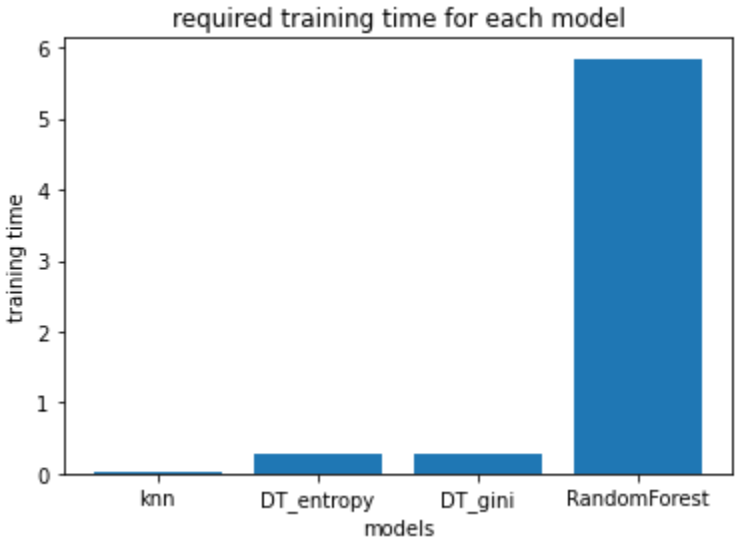


Global Observations:

1. Accuracy for each model:
    a. Random forest has a greater accuracy but it takes time to train it.
    b. Decision trees with both criteria tas reasonable time and accuracy.
    c. For this data, KNN will be a good choice because its performance is good training time and accuracy wise.



2. Training time for each model



Further notes:
I did not do hyperparameter tuning for RF because the given model gives reasonable accuracy for test data.

This model-centric approach can be further investigated:

1. Further hyperparameters tuning for each algorithm, and investigate how accuracy is varying with them.
2. By understanding data features in detailed (feature engineering)