Chinmay Mittal (2020CS10336)	Assignment 3 | COL341 Machine Learning	April 20, 2023

Binary Classification

Decision Trees from Scratch

Gini Index

Metric	Value
Training Accuracy	98.85 %
Validation Accuracy	92.75 %
Training Precision	0.9877
Validation Precision	0.8817
Training Recall	0.9660
Validation Recall	0.8200
Training Time	480 s

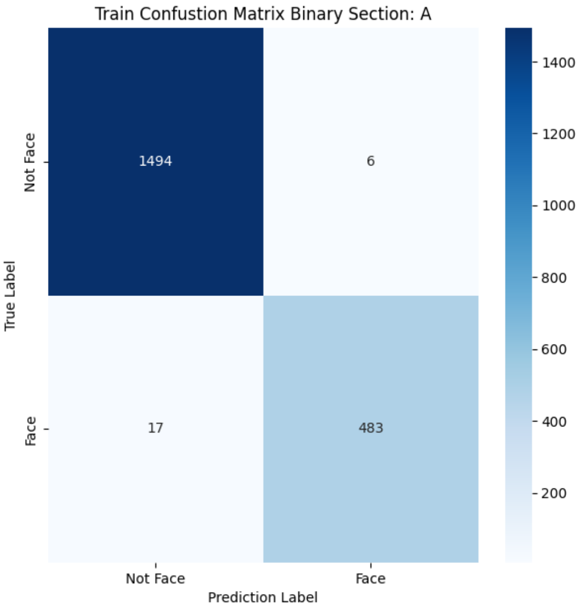

Figure 1: Training set Confusion Matrix

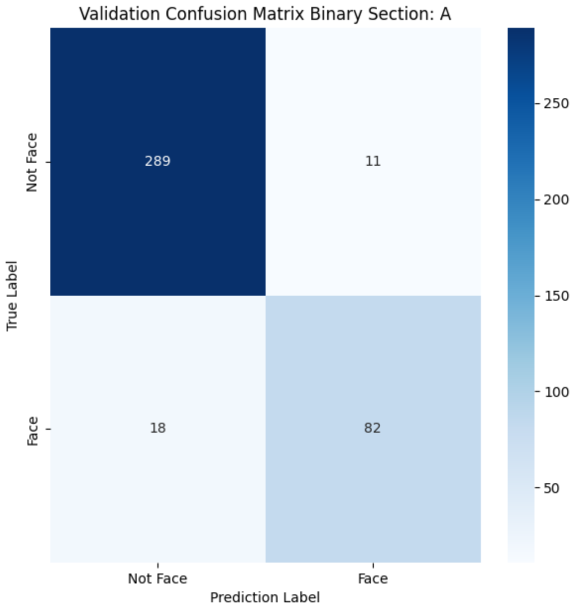

Figure 2: Validation set Confusion Matrix

Information Gain

Metric	Value
Training Accuracy	99.90 %
Validation Accuracy	93.50 %
Training Precision	0.9960
Validation Precision	0.8426
Training Recall	1
Validation Recall	0.9100
Training Time	558s

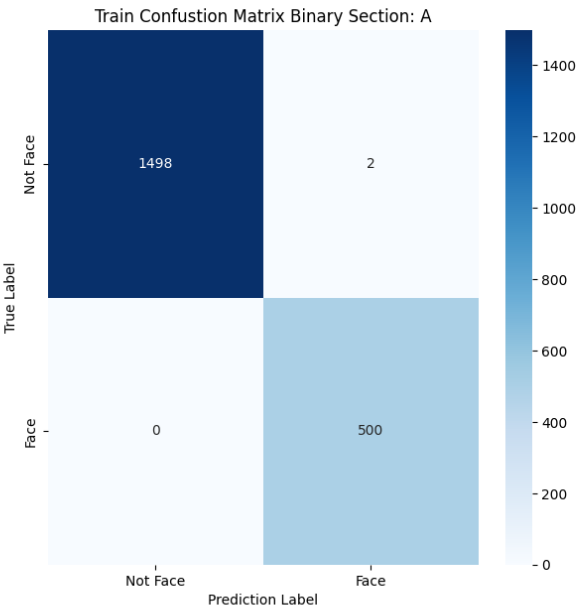

Figure 3: Training set Confusion Matrix

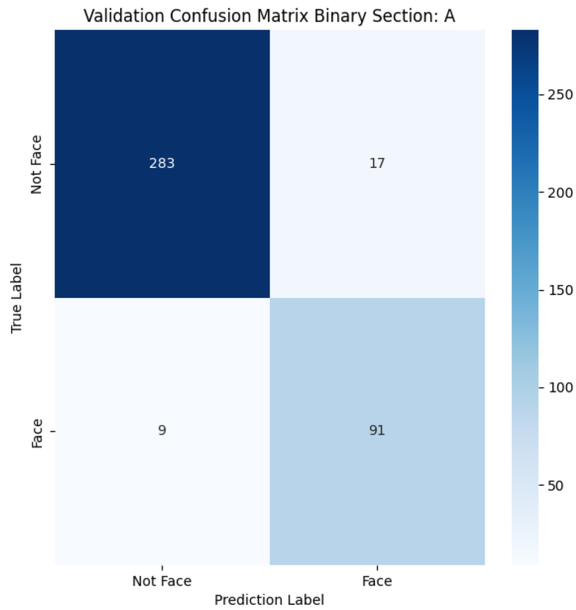

Figure 4: Validation set Confusion Matrix

Sklearn Decision Trees

Decision Tree implementation from Sklearn with default parameters except *max_depth* which is set to 10 and *min_samples_split* which is set to 7.

Metric	**Value**
Training Accuracy	98.85 %
Validation Accuracy	93.5 %
Training Precision	0.9877
Validation Precision	0.9205
Training Recall	0.9660
Validation Recall	0.8100
Training Time	2389.4045 ms

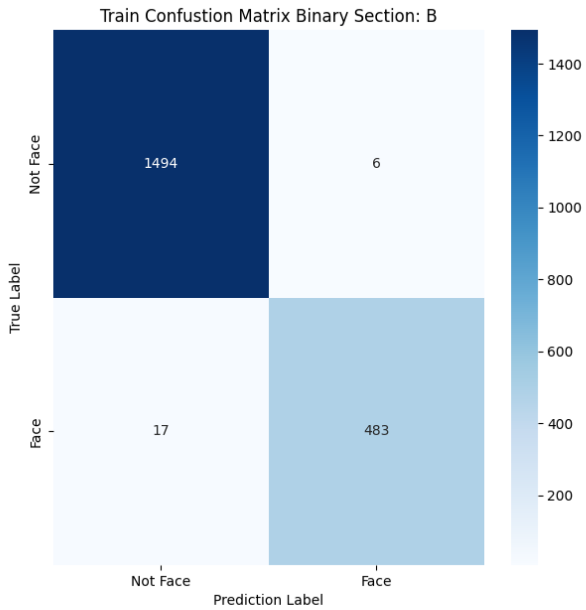

Figure 5: Training set Confusion Matrix

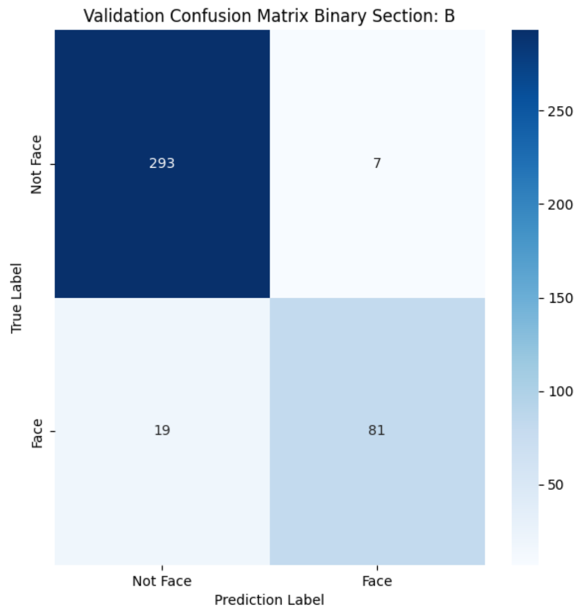

Figure 6: Validation set Confusion Matrix

Comparison with part (a) (Sklearn vs Self-Implementation)

The accuracy of the model for the same set of hyper-parameters remains almost the same. There can exact slight differences since there are sources of randomness in the decision tree learning algorithm. For eg., in case multiple features give same information gain, then which feature to choose is a source of randomization and different features can give different eventual results. There is no way to control this randomness in my implementation each decision is deterministic (for eg. choosing the first feature in case of ties mentioned above). In case of the sklearn implementation the *random_state* variable controls this source of randomness and appropriately choosing this variable we can get exactly same results for both the implementaitons.
The sklearn implementation is much faster on the other hand because it has been algorithmic-ally optimized and different routines are used depending on nature of the training data. Also python APIs internally use Cython and C++ which are much faster than native python implementations.

Decision Tree Grid Search and Visualization

First I used feature selection to select 10 features then trained the decision tree on top of these features. The results for the tree learnt and its visualization is as follows:

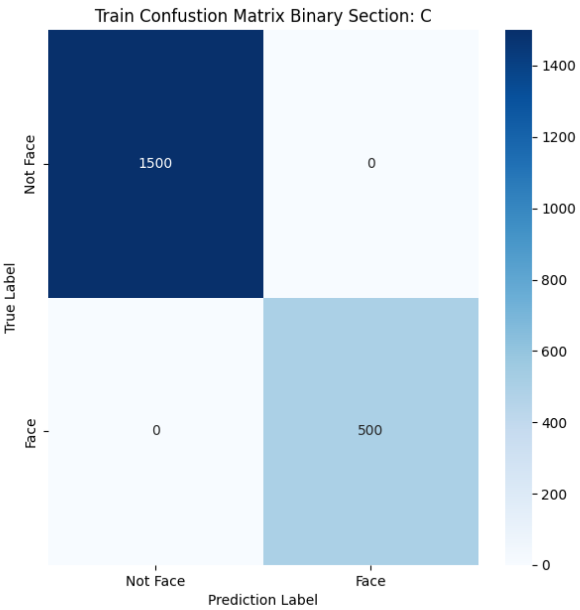

Figure 7: Training set Confusion Matrix

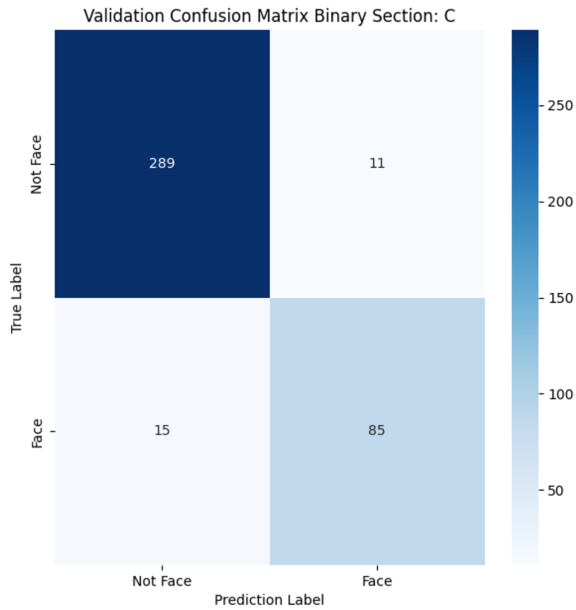

Figure 8: Validation set Confusion Matrix

Metric	**Value**
Training Accuracy	100 %
Validation Accuracy	93.50 %

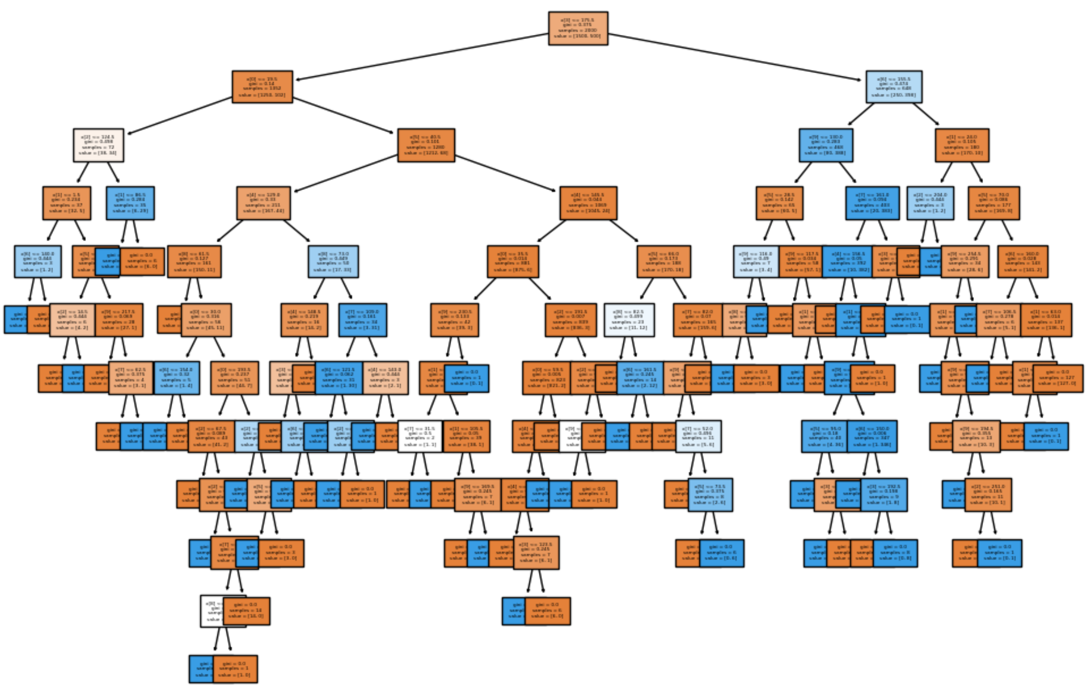

Figure 9: Visualization of Tree

Then I performed grid search the results for which are as follows:

Metric	**Value**
Training Accuracy	99.85 %
Validation Accuracy	95.00 %
Best Criterion	Entropy
Best Max Depth	None
Best Min Samples Split	4

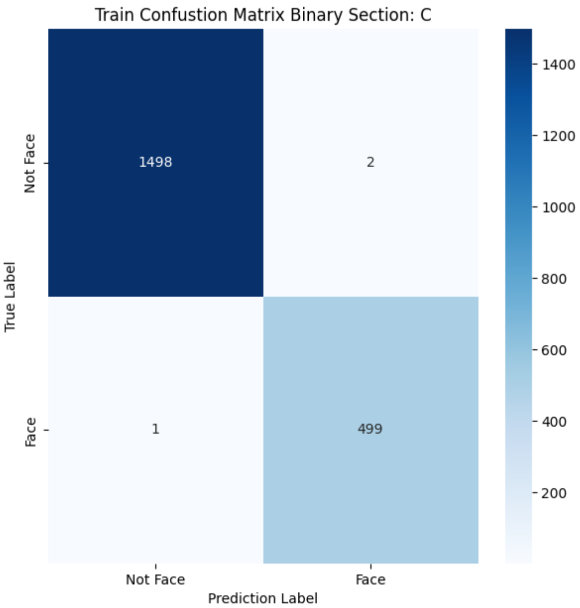

Figure 10: Training set Confusion Matrix

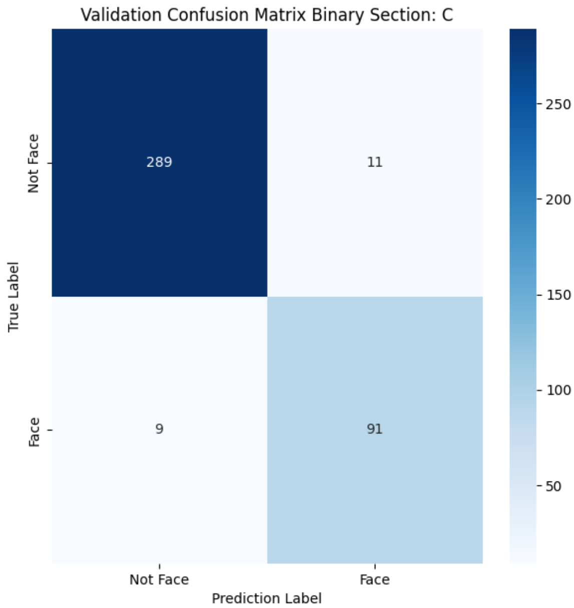

Figure 11: Validation set Confusion Matrix

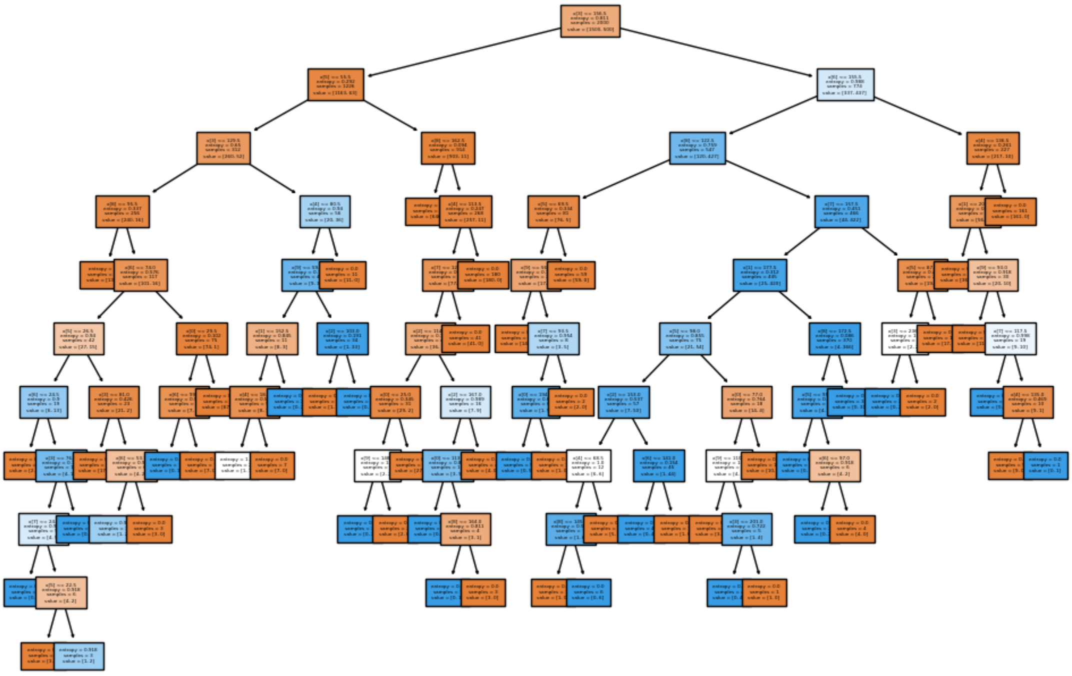

Figure 12: Visualization of Tree

Observations

Performing grid search over large hyperparameter space is pretty computationally expensive.
It is surprising that using just 10 features is enough to get good accuracies on both the training and the validation data. This corresponds to using just 10 pixel values to determine the class of an image. This happens because of biases in the dataset. For eg. almost all images of faces are of white coloured people. Hence checking for a few pixels in the image for whiteness is enough to determine whether the image if of face or not. Thus our machine learning model does pretty good till the data follows this distribution but performs pretty poorly on out-of-distribution data, the model is not learning the actual features present in face images (such as eyes and lips), but is finding simpler patterns to fit the data (intensity of central part of image). This happens because of the biases in the training data and the inductive biases in our learning model (decision tree) which learn peculiar kind of functions which might not be suitable for all tasks.
Entropy as a criterion performs much better than information gain since it is robust to slight noises in the dataset or outliers (due to the presence of the *log* function compared to *polynomials*).
Grid Search uses cross-validation to find the best set of parameters and hence optimizes for out of sample error rather than the training error. As seen the training accuracy decreases after grid search while the validation accuracy increases.

Comparison with Part (a) and Part (b)

Using very few features we can train a simpler tree with almost similar accuracies on both the training data and validation data. Hence very few features are actually being used by the models in part (a) and part (b) to actually make the prediction. Using few features also improves training times a lot and allows to perform much more exhaustive grid search in time constraints to find better hyperparemters. Using simpler trees also guarantees better generalization.

Decision Tree Post Pruning with Cost Complexity Pruning

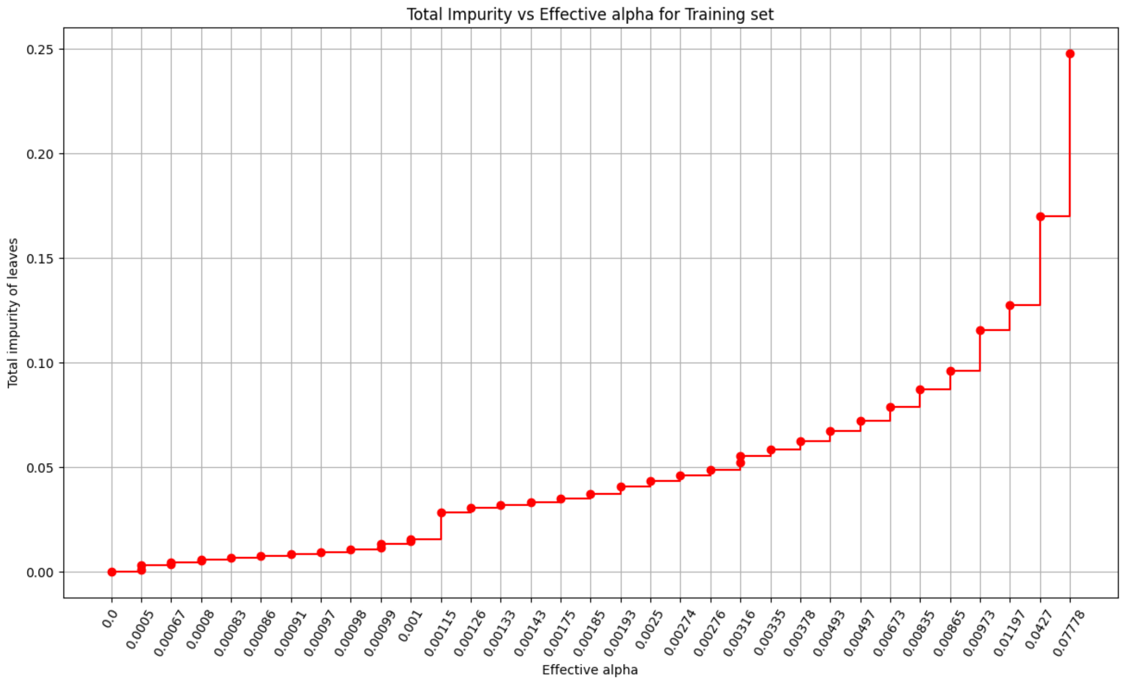

Figure 13: Impurity vs Alpha

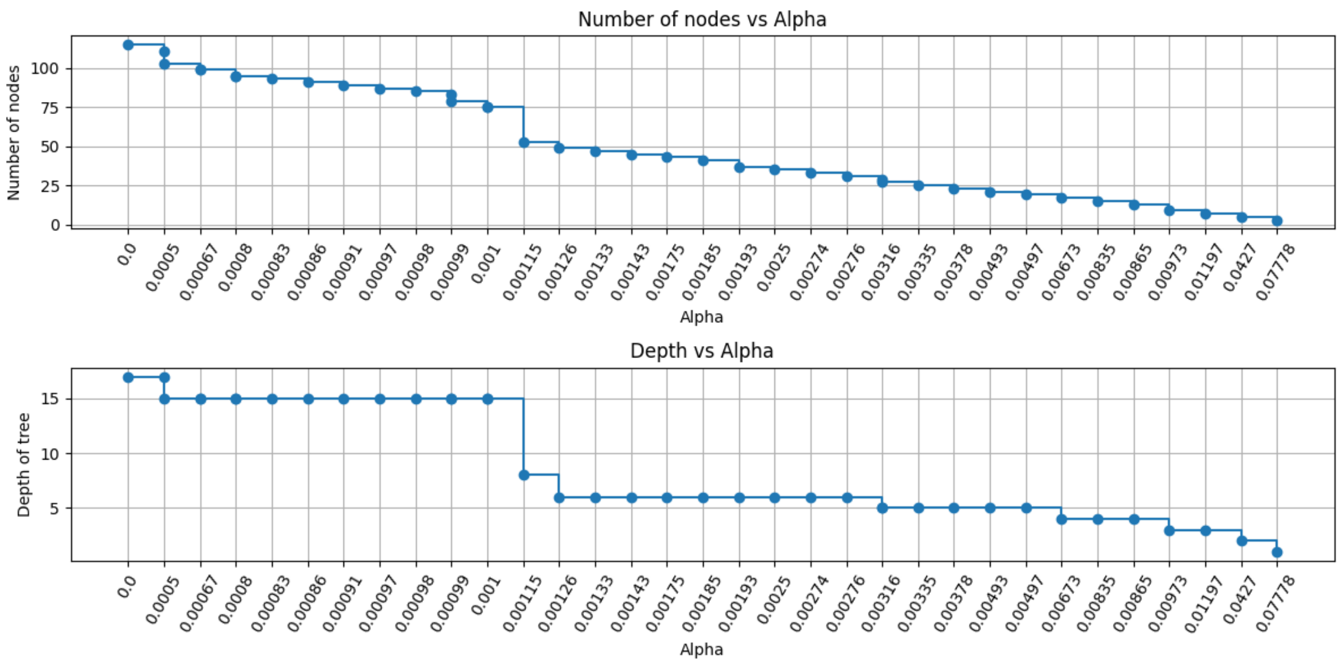

Figure 14: Nodes and Depth vs Alpha

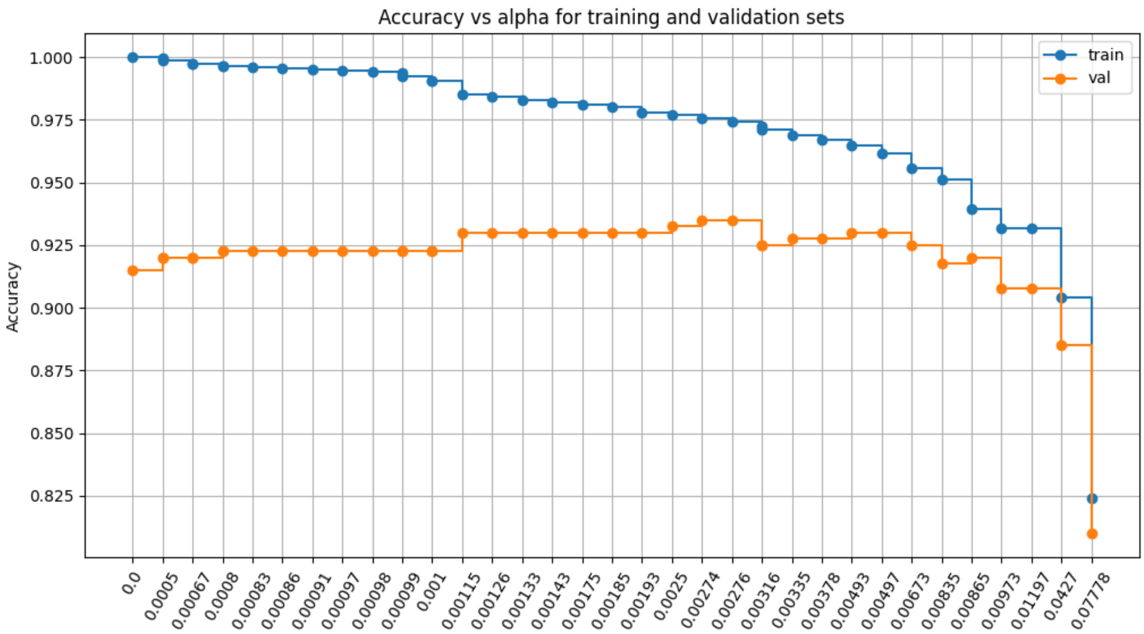

Figure 15: Training and Validation vs Alpha

Metrics for Best True according to pruning validation accuracy

Metric	Value
Training Accuracy	97.55 %
Validation Accuracy	93.50 %

Confusion Matrix for best decision tree according to validation accuracy

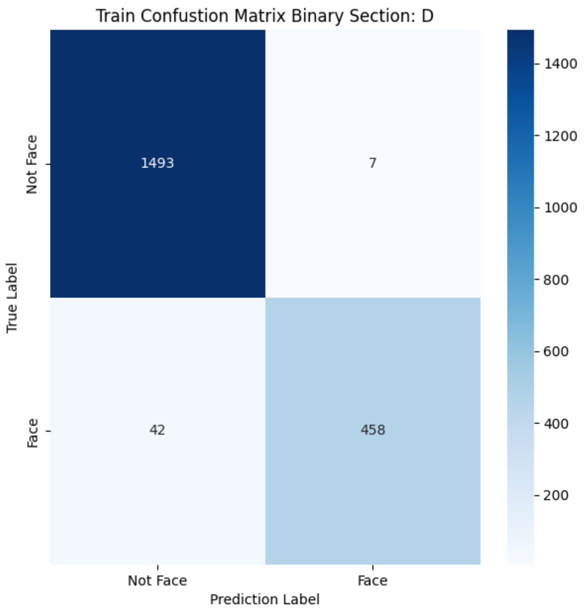

Figure 16: Training set Confusion Matrix

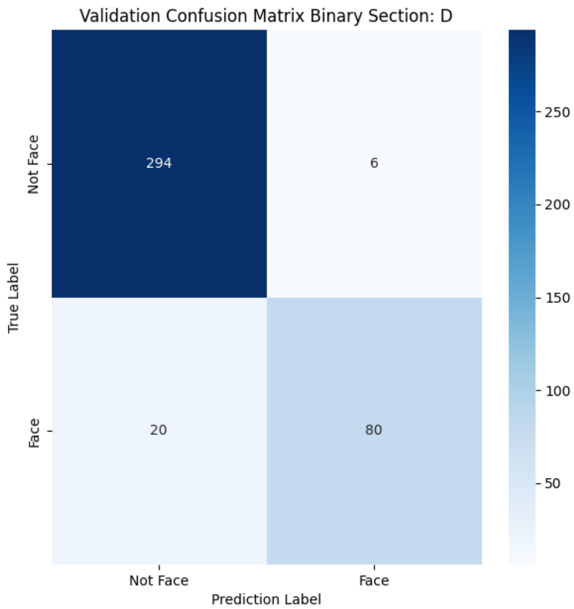

Figure 17: Validation set Confusion Matrix

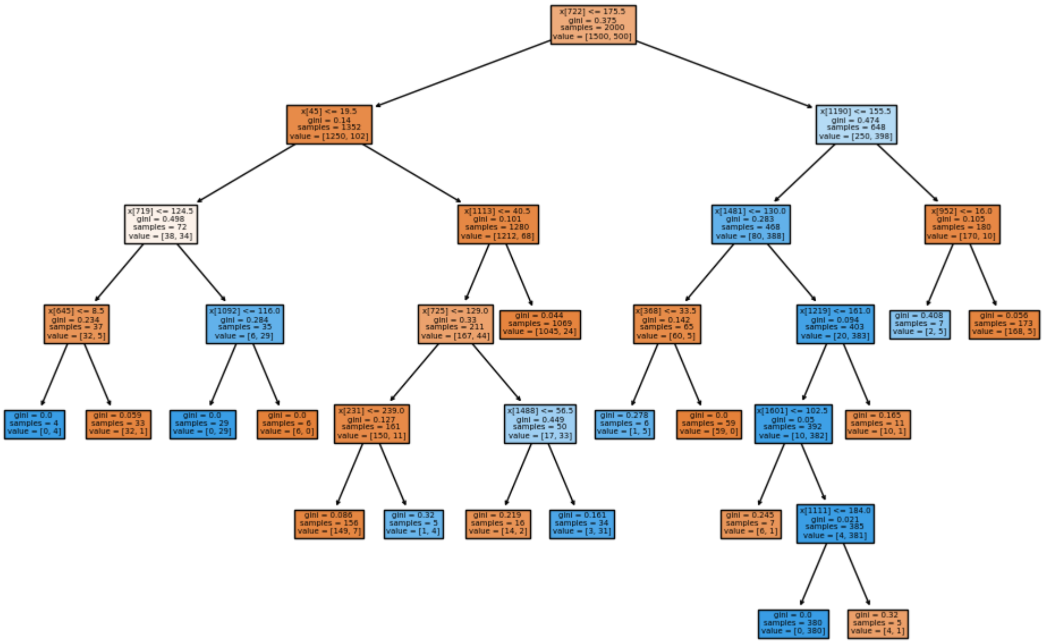

Figure 18: Visualization of Pruned Tree

Observations

As we increase the value of α, the pruning of the trees increase which corresponds to simpler trees. As we decrease the complexity of the trees learnt fitting the training data becomes difficult. Training accuracies decrease and Impurity at leaves increases. As the trees becomes simpler the number of nodes in the tree and the depth of the tree decreases. Since we prevent over-fitting the training dataset by restricting the tree in this fashion we are likely to get better generalization on the validation dataset which is seen by the initial increase in the validation accuracies. But if we make the tree very simple then due to under-fitting the validation accuracy starts decreasing steeply. Thus we can use cost complexity pruning to pick the best tree under the bias-variance trade-off to guarantee best out of sample errors.

Random Forests

The results for the default set of parameters are as follows:

Metric	**Value**
Training Accuracy	100 %
Validation Accuracy	97.5 %
Training Precision	1
Validation Precision	1
Training Recall	1
Validation Recall	0.9
Training Time	2998 ms

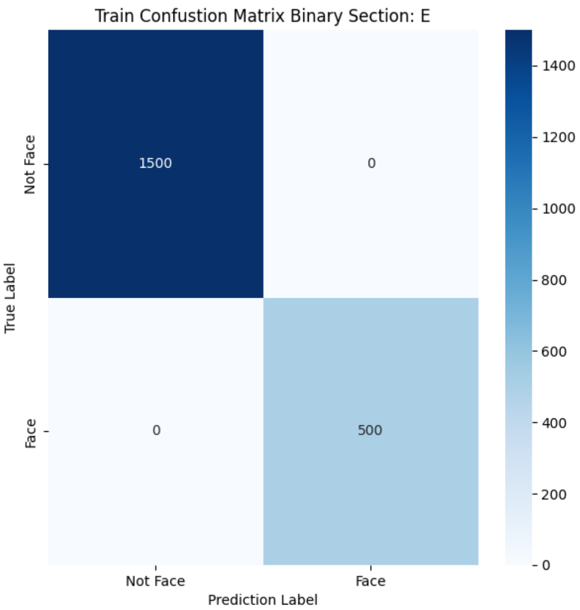

Figure 19: Training set Confusion Matrix

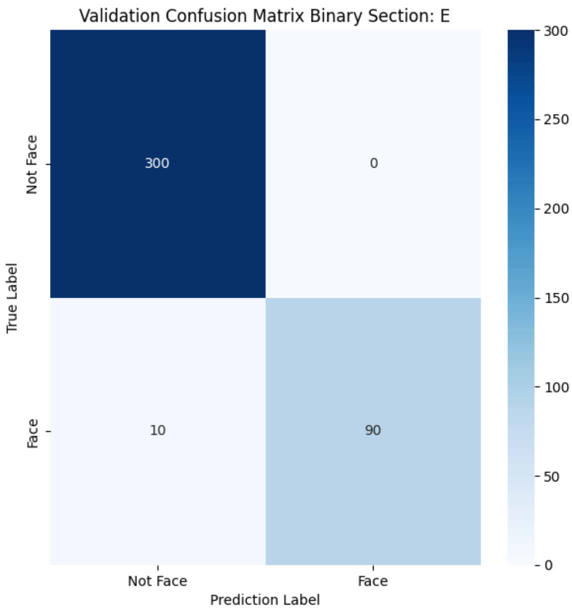

Figure 20: Validation set Confusion Matrix

Then I performed grid search over the hyperparameter space mentioned in the assignment the results are as follows:

Metric	**Value**
Training Accuracy	100.00 %
Validation Accuracy	98.25 %
Training Precision	1
Validation Precision	1
Training Recall	1
Validation Recall	0.9300
Training Time for Best Parameters	2415 ms
Grid Search Time	1248 s
Best Criterion	Entropy
Best Max Depth	None
Best Min Samples Split	5
Best Number of Estimators	100

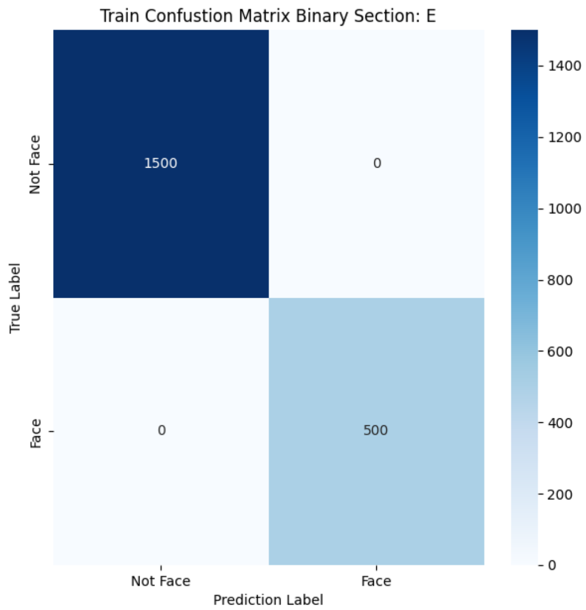

Figure 21: Training set Confusion Matrix

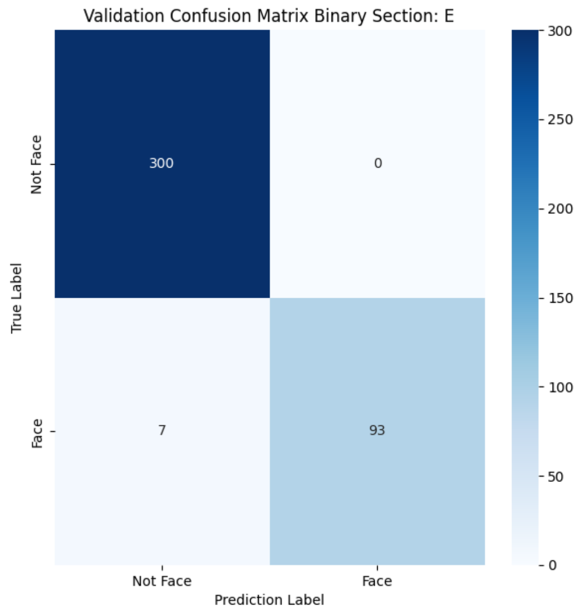

Figure 22: Validation set Confusion Matrix

Gradient Boosted Trees and XGBoost

I did a grid search on the hyperparameter space mentioned in the assignment both for Gradient Boosted trees and Extreme Gradient boosting

Gradient Boosting

Metric	**Value**
Training Accuracy	100 %
Validation Accuracy	97.50 %
Training Precision	1
Validation Precision	0.9891
Training Recall	1
Validation Recall	0.9100
Training Time for best parameters	43263 ms
Grid Search Time	3535 s

Parameter	**Best Value**
Number of Estimators	50
Max Depth	5
Subsample	0.6

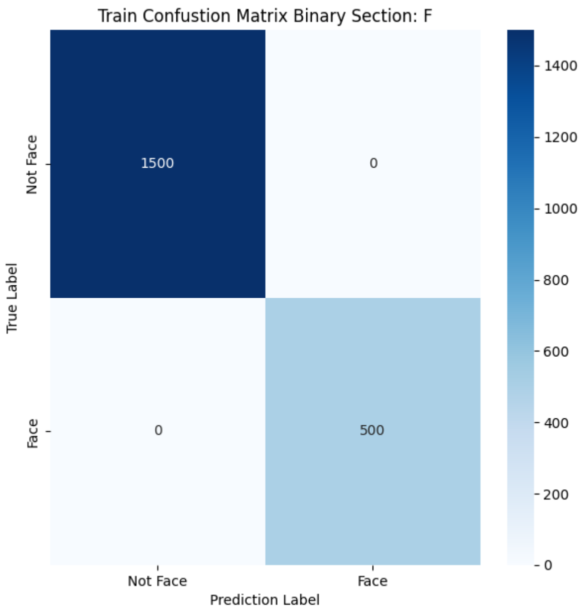

Figure 23: Training set Confusion Matrix

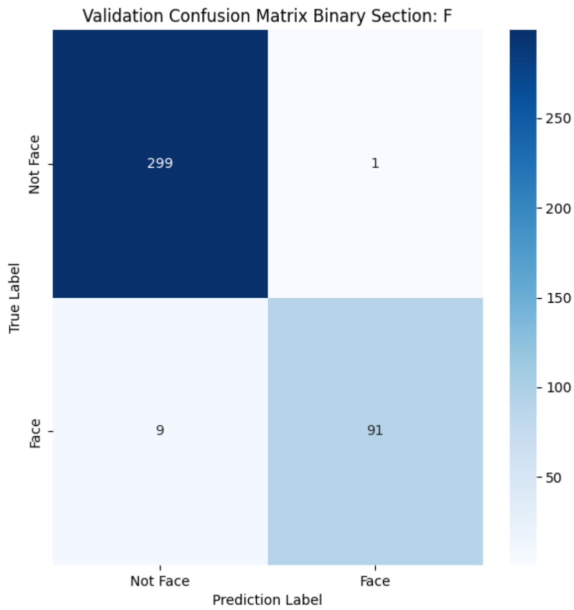

Figure 24: Validation set Confusion Matrix

XGBoost

Metric	**Value**
Training Accuracy	100 %
Validation Accuracy	98.75 %
Training Precision	1
Validation Precision	0.9897
Training Recall	1
Validation Recall	0.9600
Training Time for best parameters	3766 ms
Grid Search Time	1316 s

Parameter	**Best Value**
Number of Estimators	50
Max Depth	5
Subsample	0.5

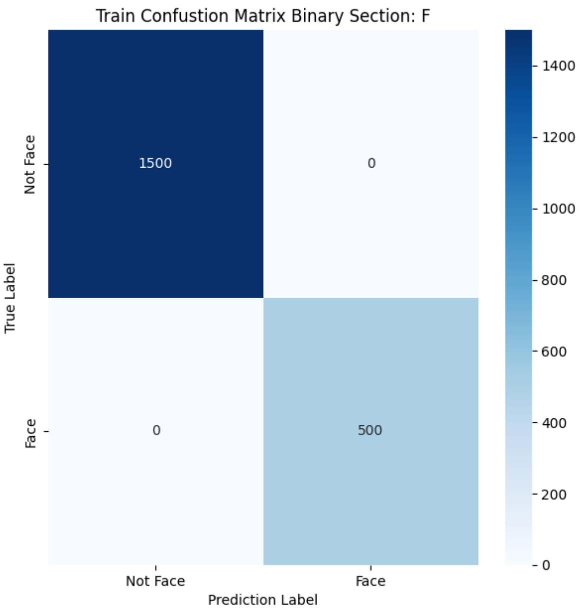

Figure 25: Training set Confusion Matrix

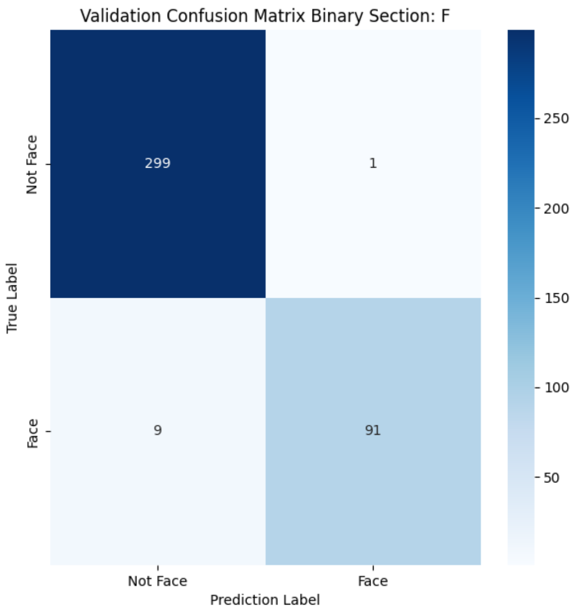

Figure 26: Validation set Confusion Matrix

Competitive Part

The best model I obtained from all sets of experiments along with its metrics and hyperparameters is as follows:

Name	**Value**
Model	XGBoost
Training Accuracy	100 %
Validation Accuracy	98.75 %
Training Precision	1
Validation Precision	0.9897
Training Recall	1
Validation Recall	0.9600
Number of Estimators	50
Max Depth	5
Subsample	0.5

Multiclass Classification

Decision Trees Sklearn

Decision Tree implementation from Sklearn with default parameters except *max_depth* which is set to 10 and *min_samples_split* which is set to 7.

Metric	Value
Training Accuracy	96.90 %
Validation Accuracy	73.5 %
Training Macro Precision	0.9691
Validation Macro Precision	0.7395
Training Macro Recall	0.9690
Validation Macro Recall	0.7350
Training Time	2593.4442 ms

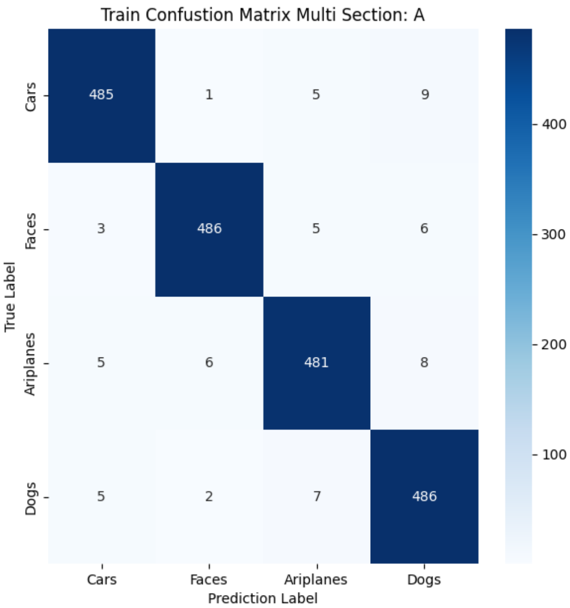

Figure 27: Training set Confusion Matrix

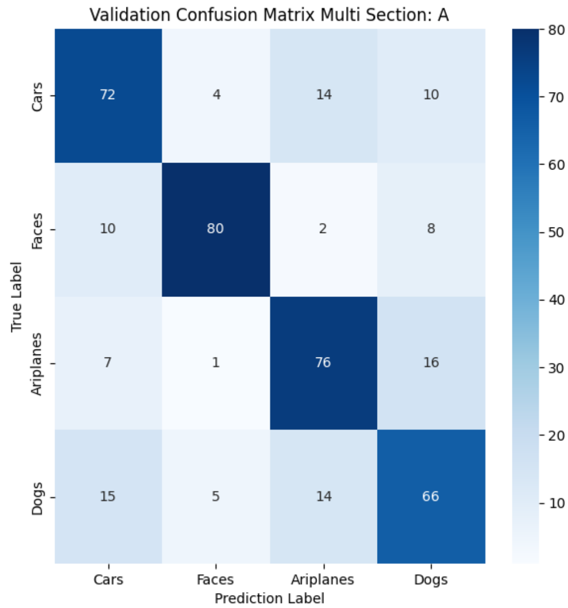

Figure 28: Validation set Confusion Matrix

Decision Tree Grid Search and Visualization

I performed a grid search over the hyperparameter space mentioned in the assignment after selecting the top 10 features the results are as follows:

Metric	Value
Training Accuracy	83.15 %
Validation Accuracy	72.25 %
Time for Grid Search	1689 ms
Time for Training Best Model	2087 ms
Best Criterion	gini
Best Max Depth	7
Best Min Samples Split	9

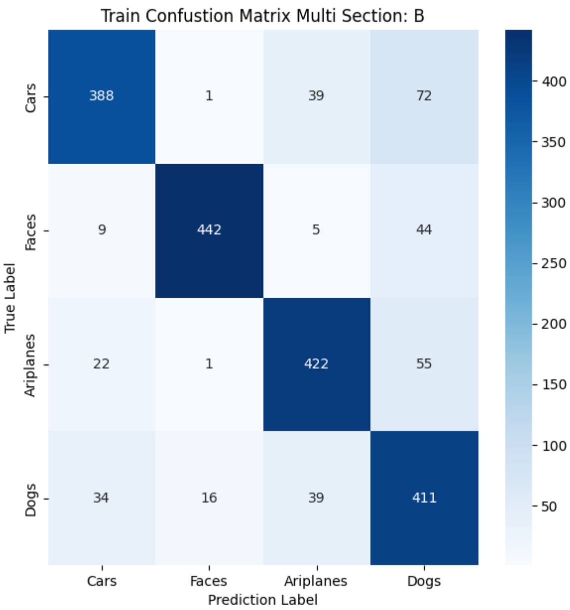

Figure 29: Training set Confusion Matrix

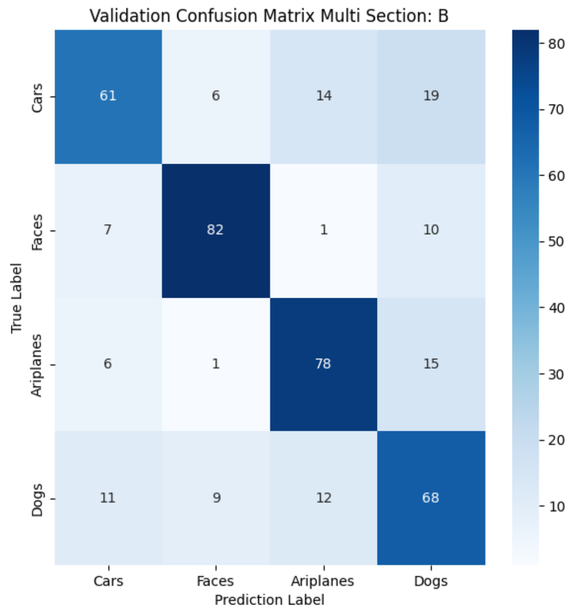

Figure 30: Validation set Confusion Matrix

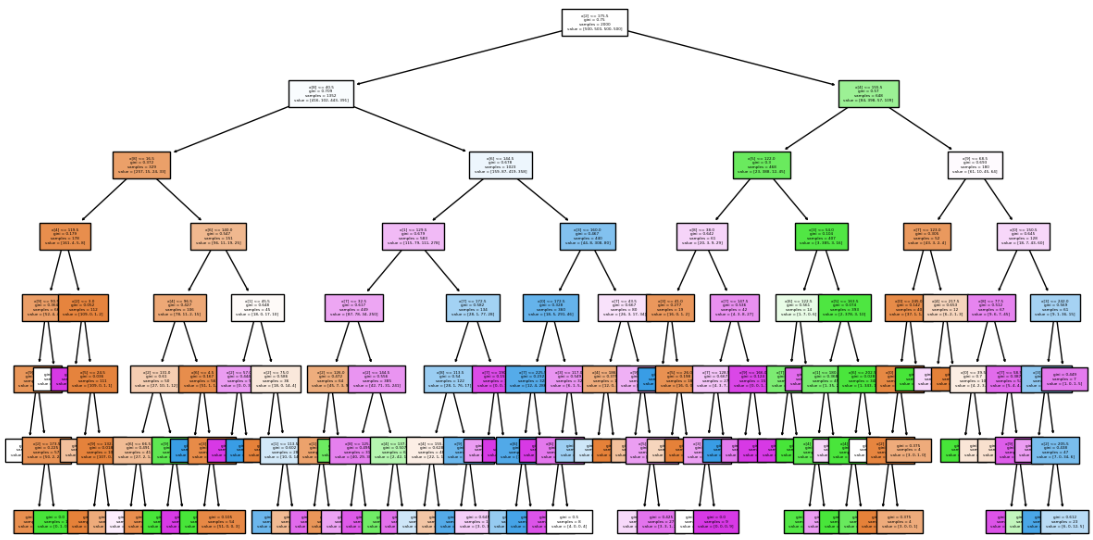

Figure 31: Visualization of Tree

Comparison with tree learnt over all features (Part (a))

Multiclass classification is an inherently difficult problem. Using just 10 features which corresponds to using 10 pixel values to classify an image might not suffice which can be seen with a corresponding drop in the training and validation accuracies. Despite the fact that we are exploring a much larger hyperparameter space the model in any case is underfitting the training data.

Decision Tree Post Pruning with Cost Complexity Pruning

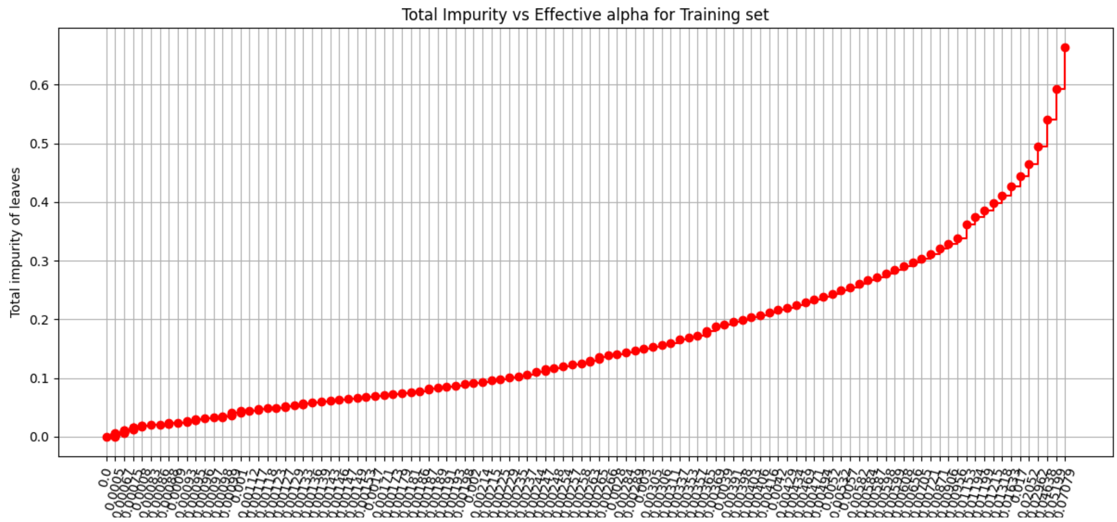

Figure 32: Impurity vs Alpha

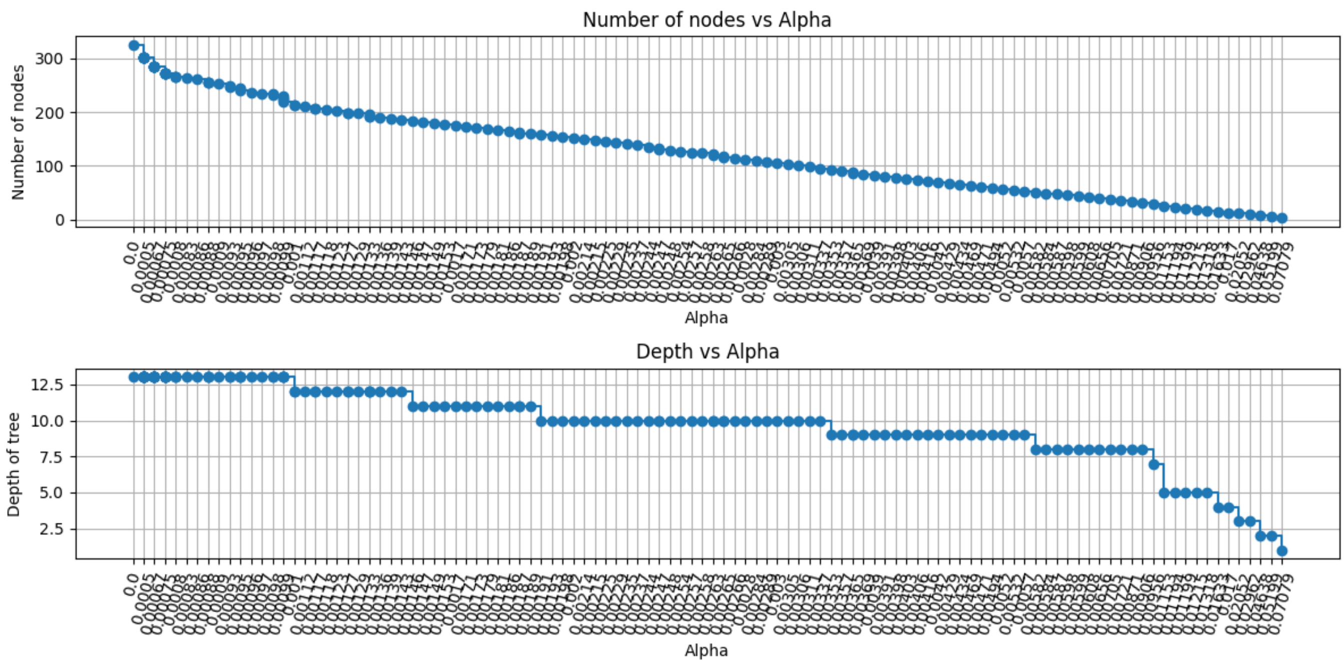

Figure 33: Nodes and Depth vs Alpha

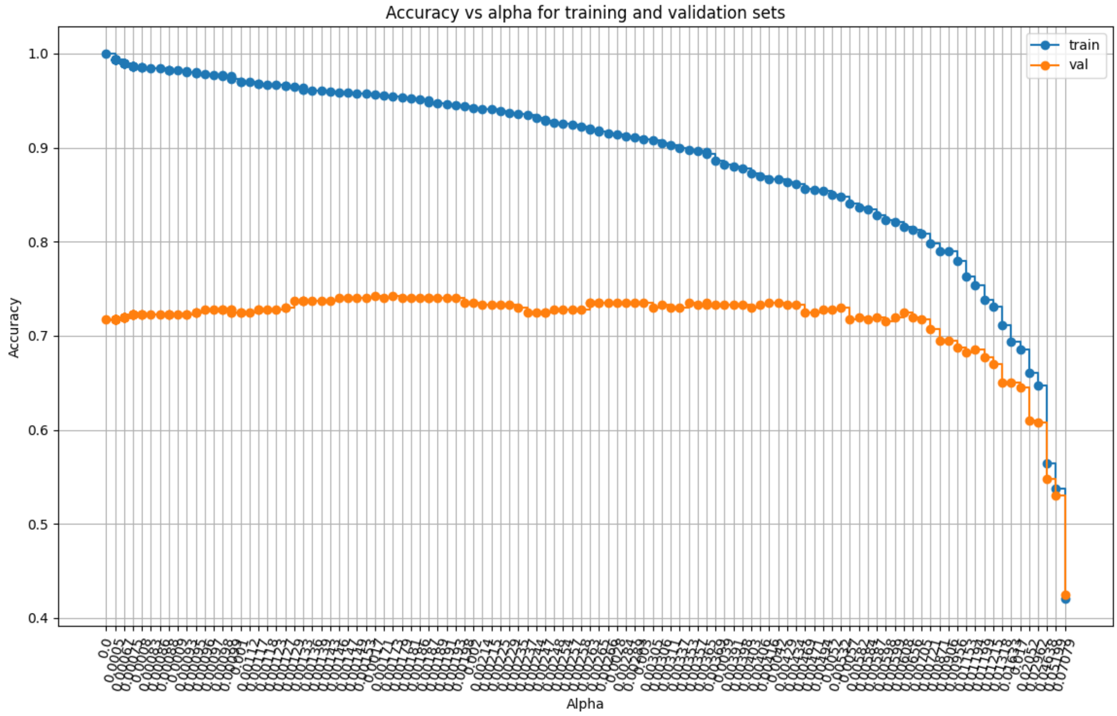

Figure 34: Training and Validation vs Alpha

Metric	**Value**
Training Accuracy	95.70 %
Validation Accuracy	74.25 %

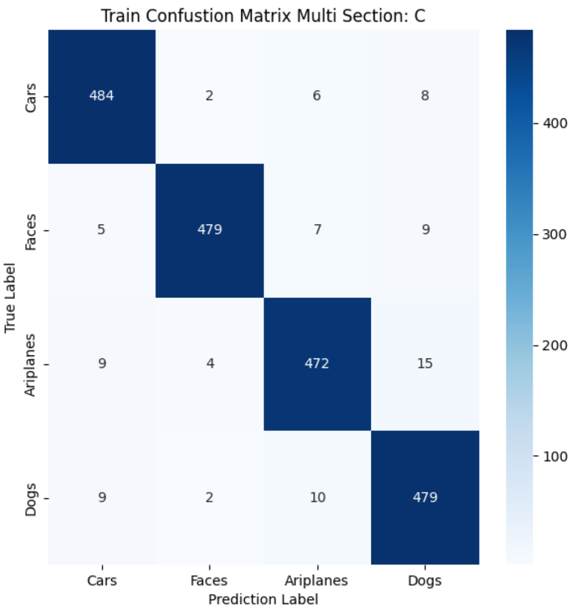

Figure 35: Training set Confusion Matrix

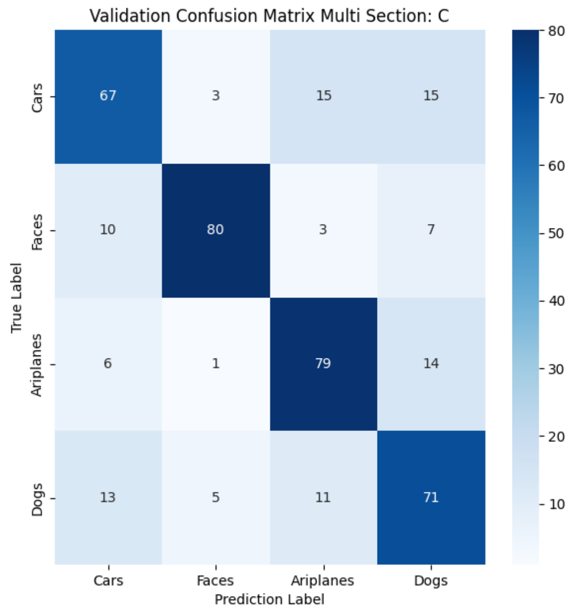

Figure 36: Validation set Confusion Matrix

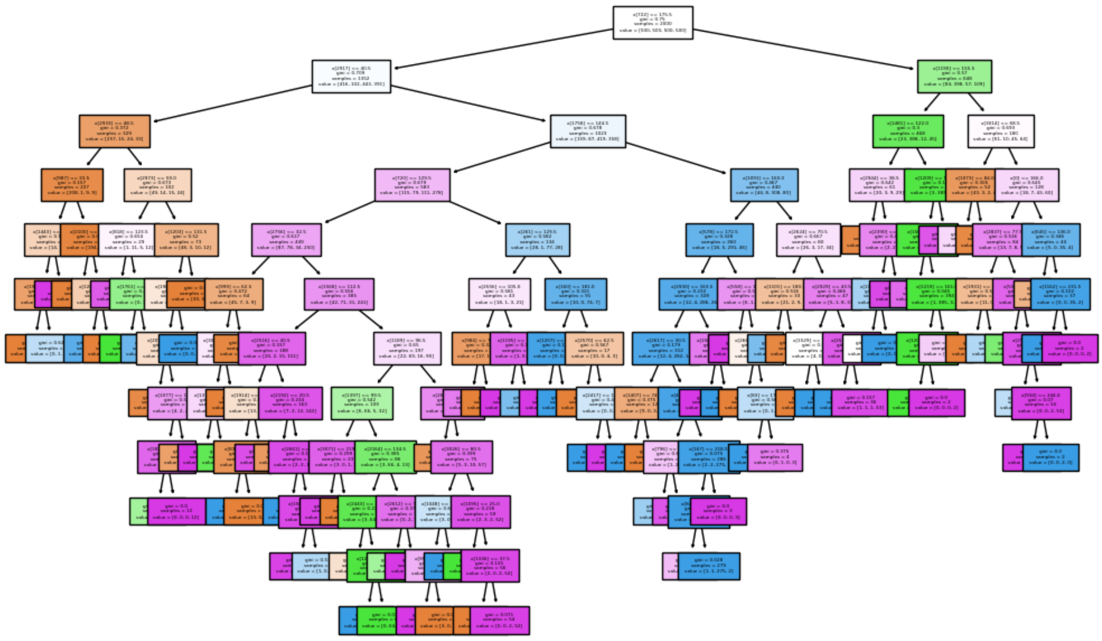

Figure 37: Visualization of Pruned Tree

Random Forests

The results for the default set of parameters are as follows:

Metric	**Value**
Training Accuracy	100 %
Validation Accuracy	87.75 %
Training Marco Precision	1
Validation Macro Precision	0.8808
Training Macro Recall	1.00
Validation Macro Recall	0.8775
Training Time	3807 ms

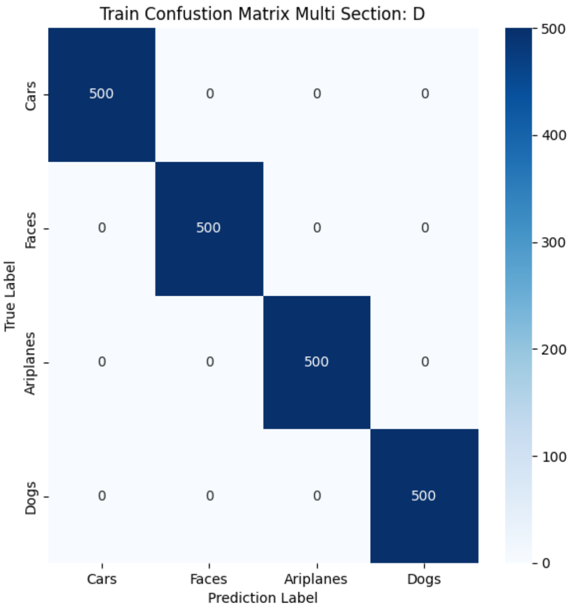

Figure 38: Training set Confusion Matrix

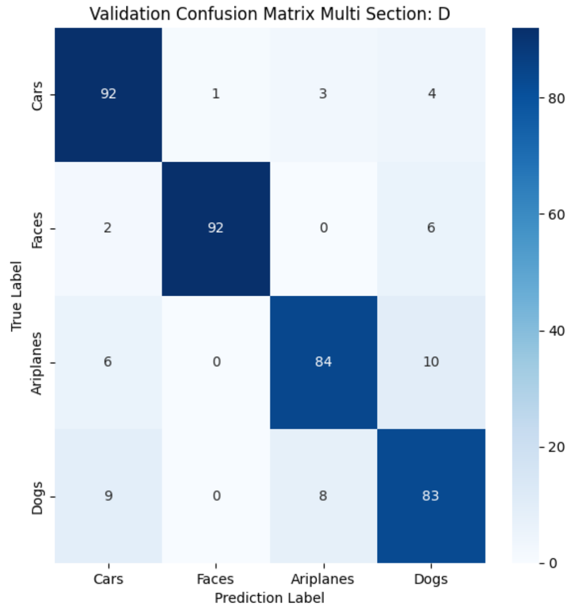

Figure 39: Validation set Confusion Matrix

Then I performed grid seach over the hyperparameter space mentioned in the assignment the results are as follows:

Metric	**Value**
Training Accuracy	100 %
Validation Accuracy	87.25 %
Training Macro Precision	1
Validation Macro Precision	0.8737
Training Macro Recall	1.00
Validation Macro Recall	0.8725
Training Time with Best Parameters	4640 ms
Grid Search Time	1814s
Best Criterion	entropy
Best Max Depth	10
Best Min Samples Split	5
Best Number of Estimators	100

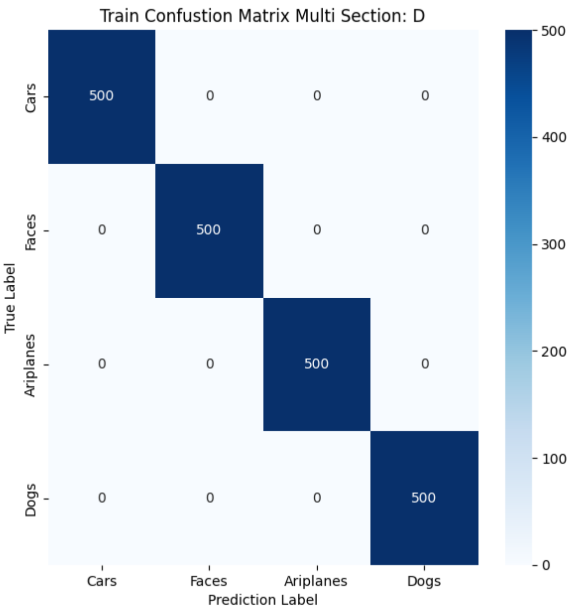

Figure 40: Training set Confusion Matrix

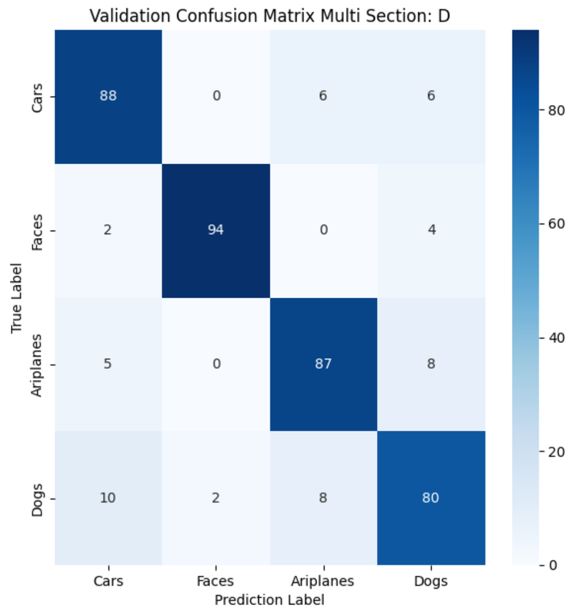

Figure 41: Validation set Confusion Matrix

Gradient Boosted Trees and XGBoost

I did grid search on the hyperparameter space mentioned in the assignment for both gradient boosting and extreme gradient boosting

Gradient Boosting

Metric	**Value**
Training Accuracy	100 %
Validation Accuracy	88.5 %
Training Marco Precision	1
Validation Macro Precision	0.8855
Training Macro Recall	1
Validation Macro Recall	0.8850
Training Time for best parameters	200850 ms
Grid Search Time	13178 s

Parameter	**Best Value**
Number of Estimators	50
Max Depth	6
Subsample	0.6

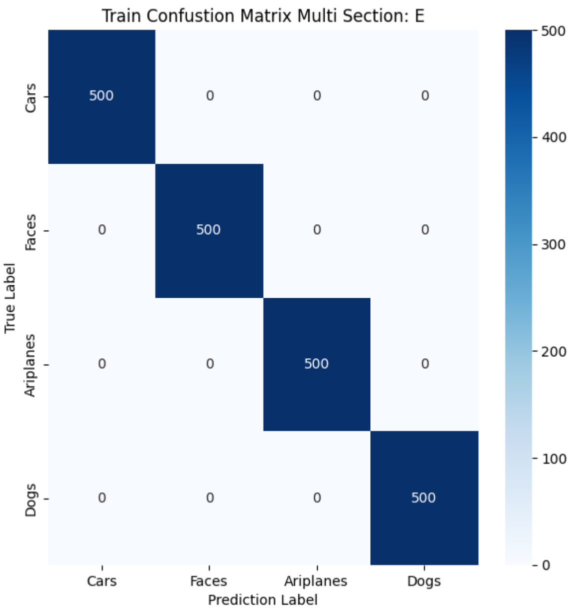

Figure 42: Training set Confusion Matrix

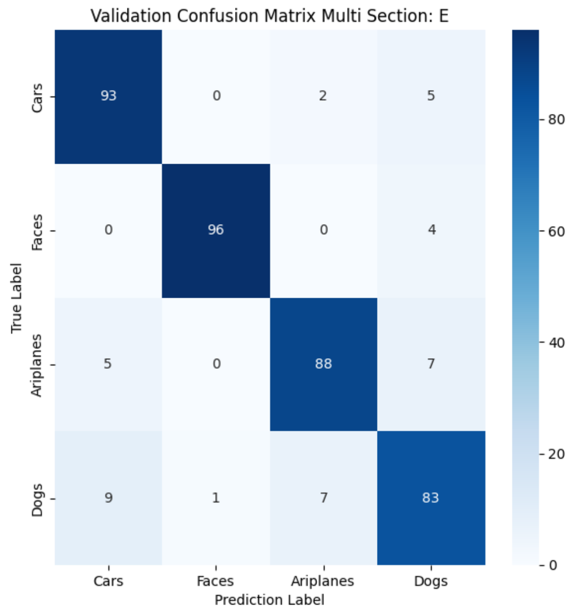

Figure 43: Validation set Confusion Matrix

XGBoost

Metric	**Value**
Training Accuracy	100 %
Validation Accuracy	90.00 %
Training Marco Precision	1
Validation Macro Precision	0.9011
Training Macro Recall	1
Validation Macro Recall	0.9000
Training Time for best parameters	15778 ms
Grid Search Time	5517 s

Parameter	**Best Value**
Number of Estimators	50
Max Depth	5
Subsample	0.6

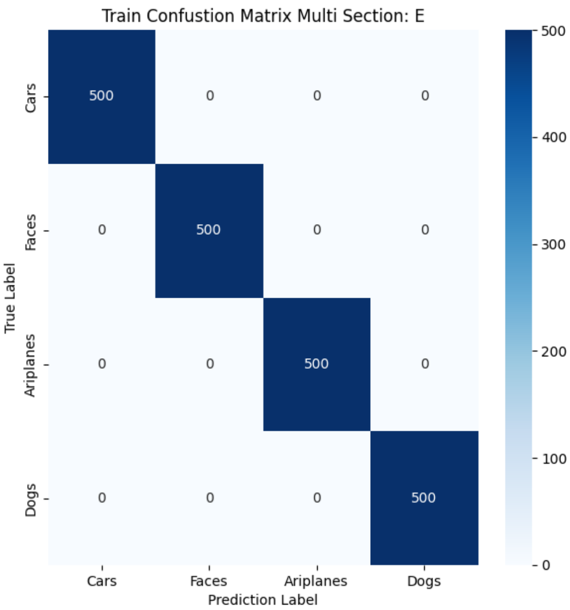

Figure 44: Training set Confusion Matrix

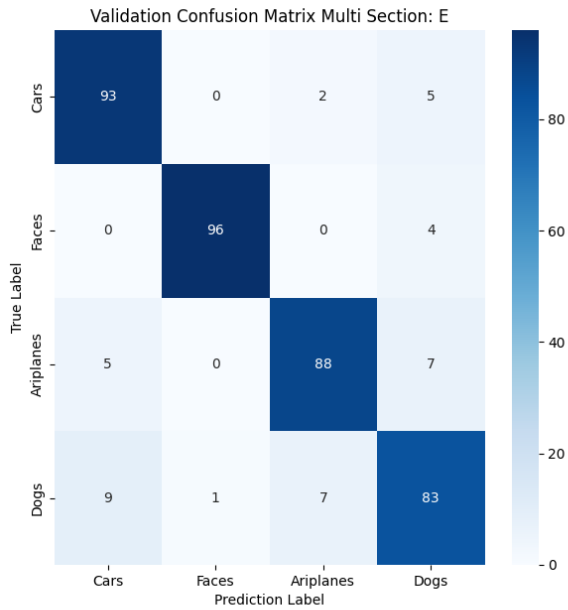

Figure 45: Validation set Confusion Matrix

Real Time Application

Metric	Value
Real Time Dataset (10 images) Accuracy	10 %

The model performs extremely poorly on real time data. This can be attributed to the following:

1. Many images in the real time dataset are not front facing and hence are out of distribution in that sense on which the model does not work.
2. Biases present in the training dataset which leads to poor out of domain generalization. The dataset has mostly faces one particular kind i.e. of white men or women without mustaches, front facing and smiling. Hence it leads the model into wrongly believing what the definition of a person is, this leads to the model poorly performing on faces which don't follow the above mention characteristics
3. Decision trees have the wrong inductive bias. The kind of functions a decision tree learns on raw pixel values do not model what we are looking for in face detectors. For eg. simply checking the intensity/whiteness of central pixels can give good results on our dataset but this function learnt by the trees does not model what we are looking for in face detectors. With relatively small training datasets it is easy for models with even the wrong inductive bias to work well which is the case while performing poorly out of sample.

Competitive Part

The best model I obtained from all sets of experiments along with its metrics and hyperparameters is as follows:

Name	Value
Model	XGBoost
Training Accuracy	100 %
Validation Accuracy	90 %
Training Marco Precision	1
Validation Macro Precision	0.9011
Training Macro Recall	1
Validation Macro Recall	0.90
Number of Estimators	50
Max Depth	5
Subsample	0.6