schuler_5779_ece_527_report_06

October 21, 2024

1 GMU ECE 527 - Computer Exercise #06 - Report

Stewart Schuler - G01395779
20241021

1.1 Exercise 6.1

The dataset being used for these experiments consists of 70,000 28x28 grayscale images. The data is partitioned into 60,000 images for training and the remaining 10,000 are held out as a test set. With each pixel as a feature the feature vector has 784 dimentions.

Questions

i. **With the default values, what is the structure of the tree? For example, are there any limits on the number of leaves, is the depth of the tree constrained, and so on.**

There are no limits imposed by the tree strucutre. It will grow until each leaf is entirely pure. That is, all the training data located within the region defined by the leaf will be of the same class.

ii. **How many leaves are in the tree that is learned, and what is the purity of the leaves? What is the maximum depth of the tree?**

Once trained the tree had a depth of 48, and in total 4996 leaves. The purity of each leaf was 1.0.

iii. **What is the training accuracy of the model on the training data? Discuss your results and any conclusions that you may have. Were you surprised by the results?**

The training accuracy was 100%. This isn't suprising because as we stated in answer *i*, with no limits on tree growth it will grow until each leaf if 100% pure, which corresponds to a perfect training accuracy.

iv. **What is the accuracy of your decision tree on the test set? What does this tell you?**

Having a perfect training accuracy tells us that we are heavily overfitting the training dataset, and that this specific classifier would almost certainly not perform well on the held-out test set.

When applying this classifier to the held-out test set we see an overall accuracy of 0.7947, which as expected for an overfit model is drastically differant than the training score of 1.0. Interestingly we can see from the confusion matrix that when it gets labels wrong it tends to get them wrong within the same category of clothing. That is, *shirts*, *t-shirts*, *pullovers* and *coat* are all commonly confused since they all belong to the category of a top. Likewise the classifier strugles distringusihing between

different types of footware, *boots*, *sneakers*, and *sandles* are often incorrectly identified. There is less significant overlap between categories.

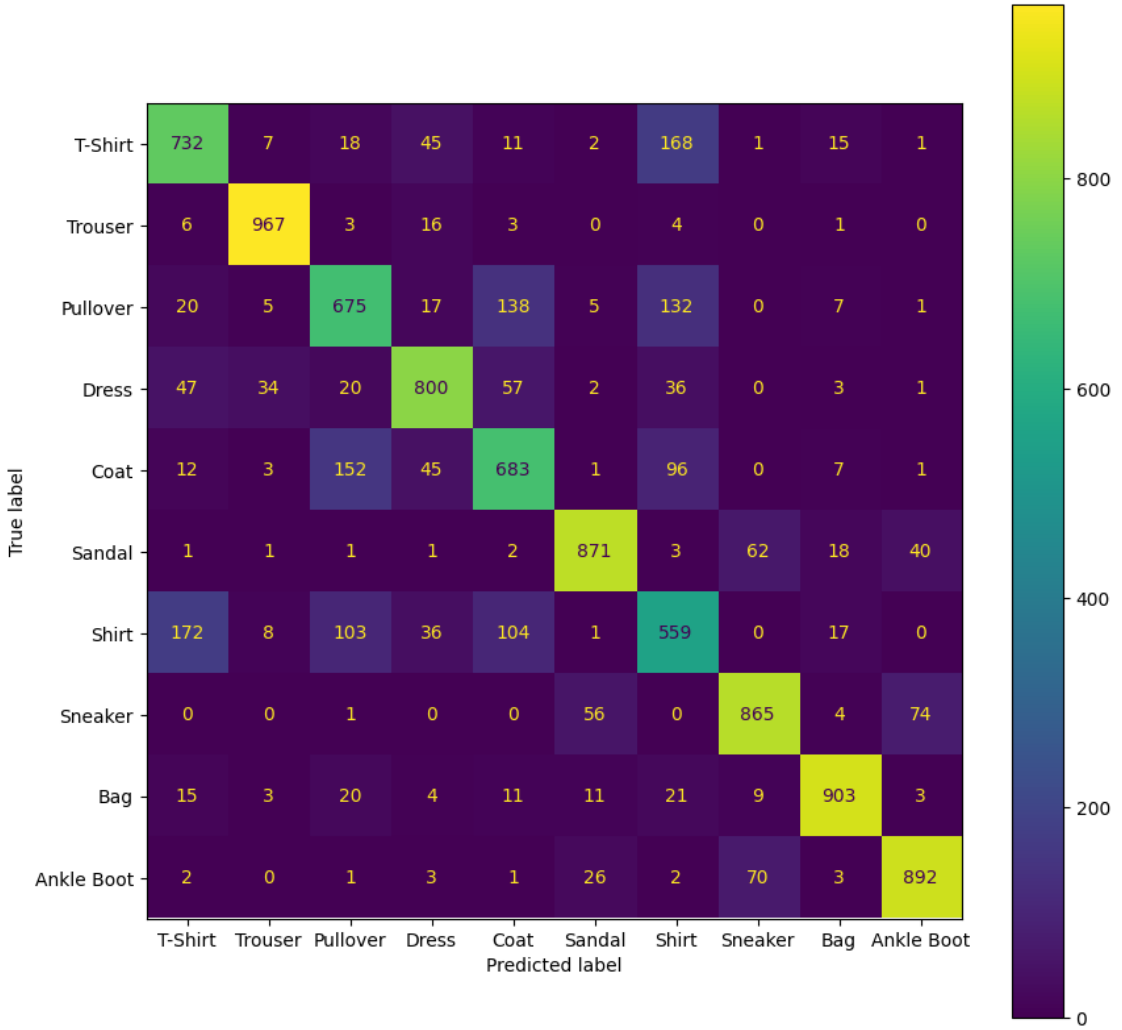

Figure 1. Test Set Confusion Matrix for Unrestricted Tree

Next we apply cross validation to the same model to see if we would have been able to predict the test set accuracy we previously found. Cross validation can be done with any number of folds, typically more folds will give an average accuracy closer to the expected accuracy of the test set. The cost of apply more fold is that cross validation is slow and timing consuming because you are training as many models as there are folds.

Using 5-fold cross validation I found an average accuracy on the validation set was 0.793, which is very clsoe the accuracy we previously measured on the held out test set. That demonstrates the cross validation approach did a good job of estiamting the expected accuracy. This also demonstrates that we can detect overfitting on training data even before applying the model to the test set because of the large descrepency between the training accuracy (1.0) and the average validation set accuracy.

Next we impose some contrains on the size of the tree. We define a tree with, $min\_samples\_leaf = 5$ and $max\_depth = 12$. The resulting classifier had a accuracy of 0.876 on the training set and a 5-fold cross validation average expected accuracy of 0.814, we can see with the classifier because

the difference between the two numbers is much closer there is less overfitting going on.

When the model is applied to the test set we see the confusion matix shown in *Figure 2*, and can compute an overall accuracy of 0.819. Of note this model generalized better to the test set than the unrestricted model, and it's training error is very close to that predicted by the cross validation estiamte.

Dispite the slight overall accuracy increase this model still has trouble correctly classifying the test set as described for the unrestricted tree, that is withing an overall clothing category (i.e. tops, shoes, ect.). That tells us that within the 728 dimentional feature space those classes which is has difficult differentiating must have some overlap.

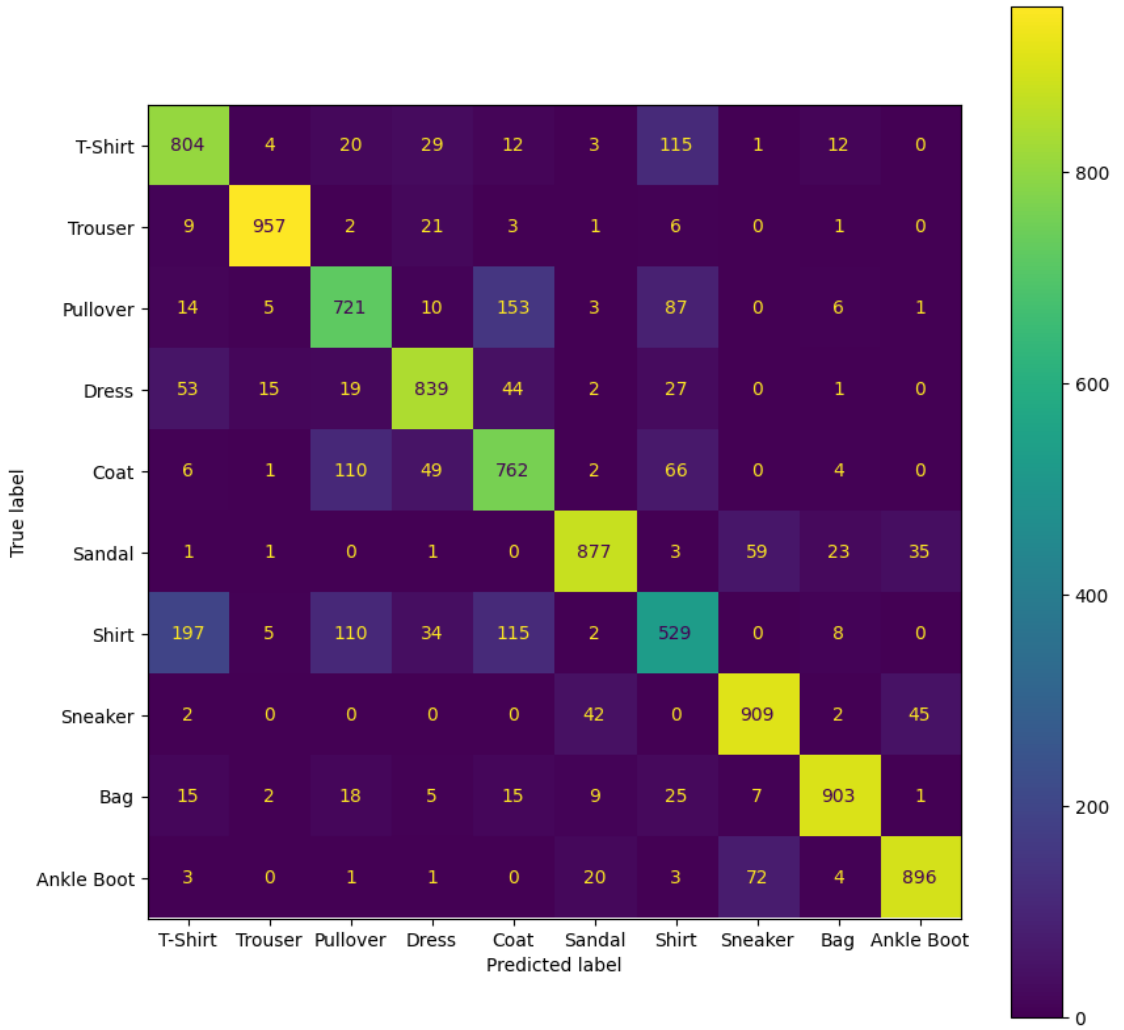

Figure 2. Test Set Confusion Matrix for Restricted Tree

Lastly we apply a grid search technique to find the best hyperparameters for the tree. For this search space I considered the possible values of $min\_samples\_leaf = [3, 5, 10, 20]$ and $max\_depth = [10, 12, 16, 18]$ with a 5-fold cross validation. This search looks at 16 different models and each is trained 5 times. The results found the best parameters to be $min\_samples\_leaf = 3$ and $max\_depth = 12$ with an expected accuracy of 0.815. Which was interesting because that was a worse expected accuracy than the 0.819 found in the previous model (whose parameters were

included in the search space). However that is a result of the randomness of the validation set draw, if I had the computational resources to do a »5 model we would expect these validation set varriances to average away.

1.2 Experiment 6.2

For this experiment we will be using a *Random Forest* classifier. For this classifier I initial chose the parameters shown in *Table 1*.

Parameter	Value
n_estimators	200
max_depth	18
min_samples_split	10
max_features	28

Table 1. Model Parameters

For this model I found a training accuracy of 0.975, and an OOB validation accuracy of 0.878. That tells me that even though I tried to added some restrictions to the trees to prevent overfitting, I didn't add enough and the model is still overfitting. Interestingly even with a large difference between training/OOB accuracy (i.e. overfitting) the expected testing accuracy is stil higher than any of the single tree models tested in the previous experiment.

Next we consider the effect of increating the number of features randomly selected at each split point in the tree. For the previous model I used the entire dataset. For this test, the model will hold all the other parameters constant (as defined by *Table 2*) and vary the *max_features* parameter linearly between 1 and 728 with 20 different tested values.

Parameter	Value
n_estimators	50
max_depth	5
min_samples_split	10

Table 2. Model Parameters

The result of OOB accuracy vs number of features is shown in *Figure 3*. As can be clearly seen (unsprisingly) 1 feature is too little, then considering 39 features produces the best result and after that point including any more features hurts the OOB accuracy. This isn't all that suprising considering the *sklearn* documentation uses the square root of the total number of features as a rule of thumb for choosing *max_features*.

Also this model's raw accuracy value performs significantly worse than the previously tested model. That is beccause, as can be seen between *Table 1* and *Table 2*, the model was significnatly more restricted in growth depth. This was done because producing *Figure 3* took multiple hours to generate, increasing the depth further would've been unrealistic to generate. However since we know bagging methods such as *Random Forests* preform better with strong learners we can assume increasing the depth would've improved the accuracy of these ensemble as a whole for each *max_features* value tested.

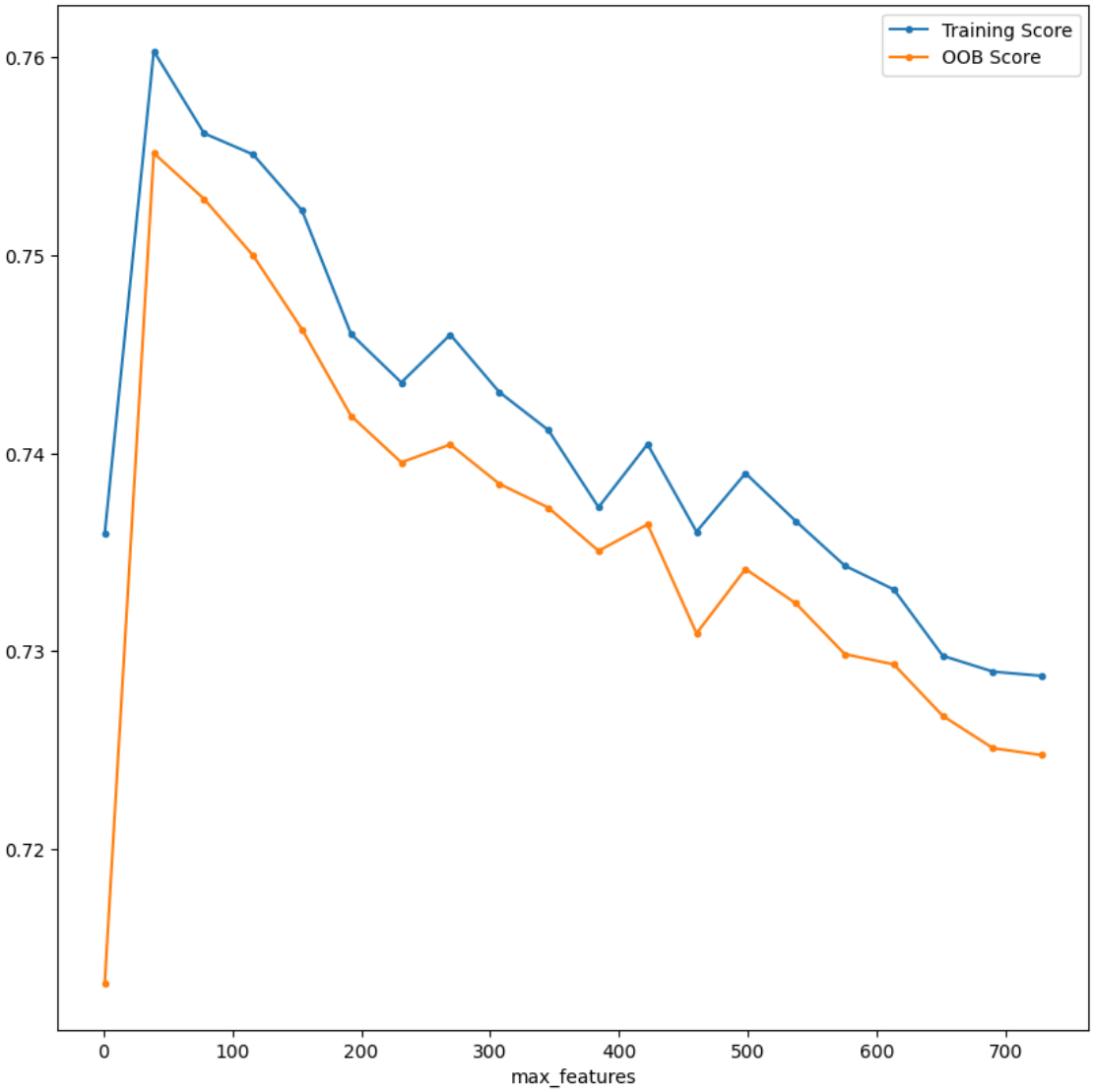

Figure 3. OOB Accuracy vs N Features

Next the experiment was run to vary over the the number of estimators in the model. For this test the *Random Forest* models used again used the same parameters as *Table 2*. With *max_features* being set to 28. *Figure 4* shows the effect of OOB accuracy on that model. What can be seen is that the number of estimators beyond a certain point have little change on the OOB accuracy. I suspect that if stronger (deeper) moduls were used as the base classifiers the number of estimators needed to before flatting would increase. However as with the previous experiment this took an extremely long time to run because of the nature of doing cross validation on ensamble methods means we are going a very large number of trees. Likewise during the prediction stage to come up with a result a specific feature must be passed through *n_ estiamtor* number of trees and the results averaged. Which was *n_ estimators* gets very large can be a non-trival ammount of time.

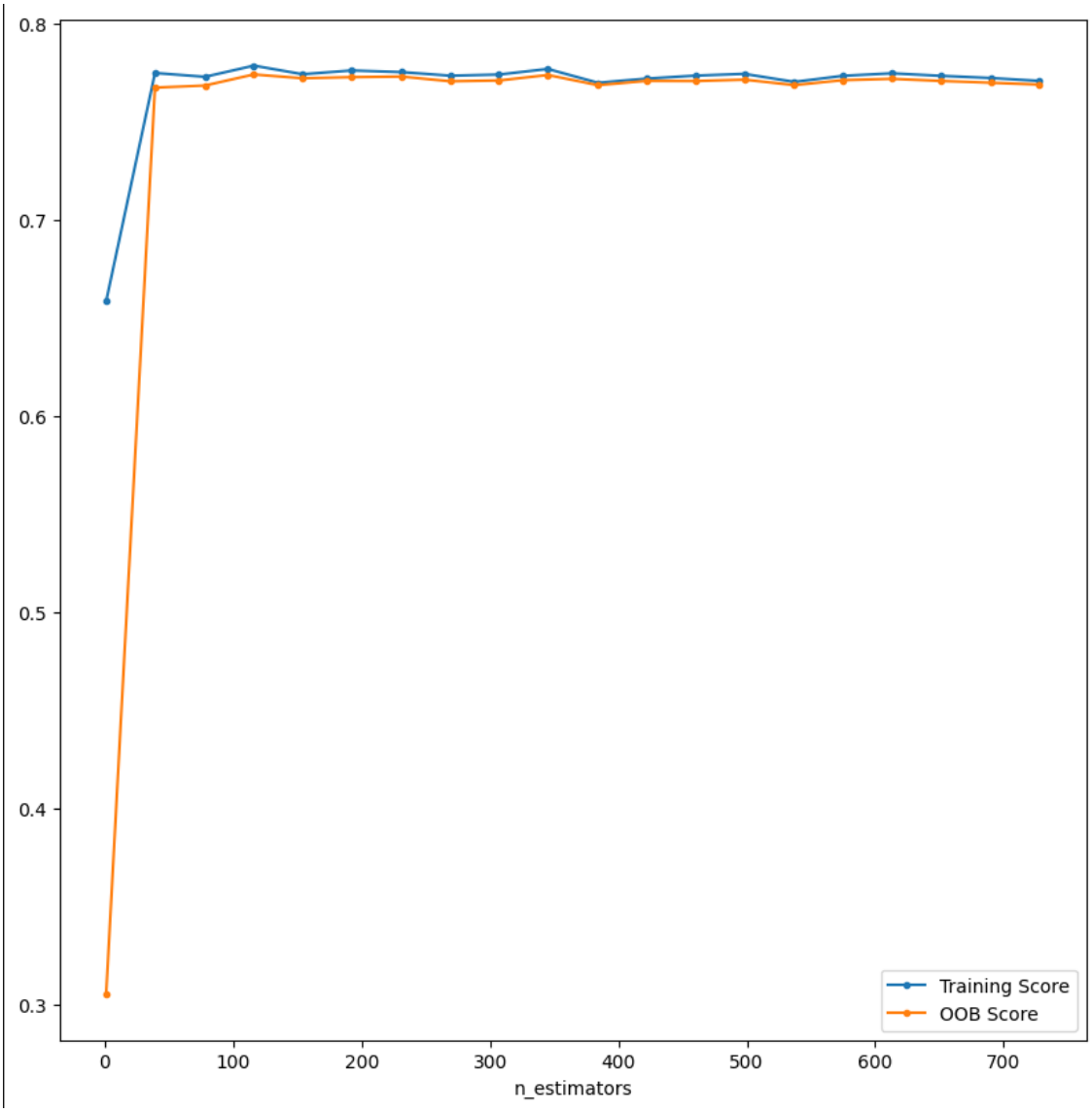

Figure 4. OOB Accuracy vs N Features

The above exeriment was than repeated using *Extra Random Trees*. With the same parameters used in their associated *Random Forest* experiments the resulting training accuracy was 0.958 and an OOB accuracy of 0.869. Which shows that the equilvalant tree performs slightly worse but it has the advantage of training quicker since thresholds are randomly chosen rather than being optimally computed for each featuere for each split in each tree. And since the OOB accuracy only very slightly decreased this model could be a good choise for a system with limited training resources, but still wanting to use an ensemble method.

The experiments of chaning *max_features* and *n_estimators* was repeated for this model type. The results are shown in *Figures 5* and *6* respectivly. While increasing *max_features* degrades performance like it did previously the rate at which is decreases is much slower than in the *Random Forest* approach. This is a result of the randomly chosen thresholds during training, having more gives you a better chance of getting a good threshold on a relevant feature. Which as before it was only needed to include a relevant feature and the optimal threshold would always be found.

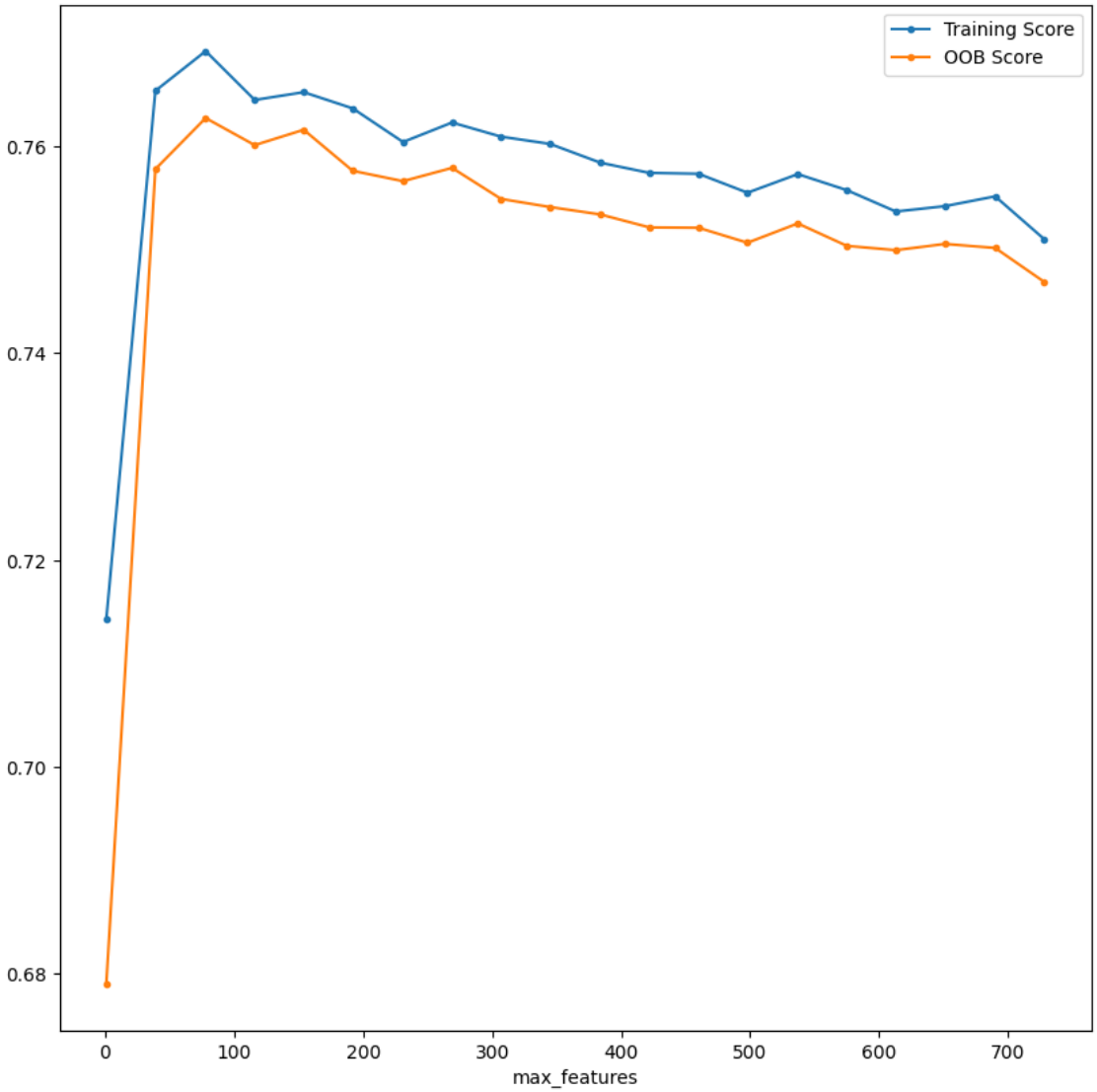

Figure 5. OOB Accuracy vs N Features

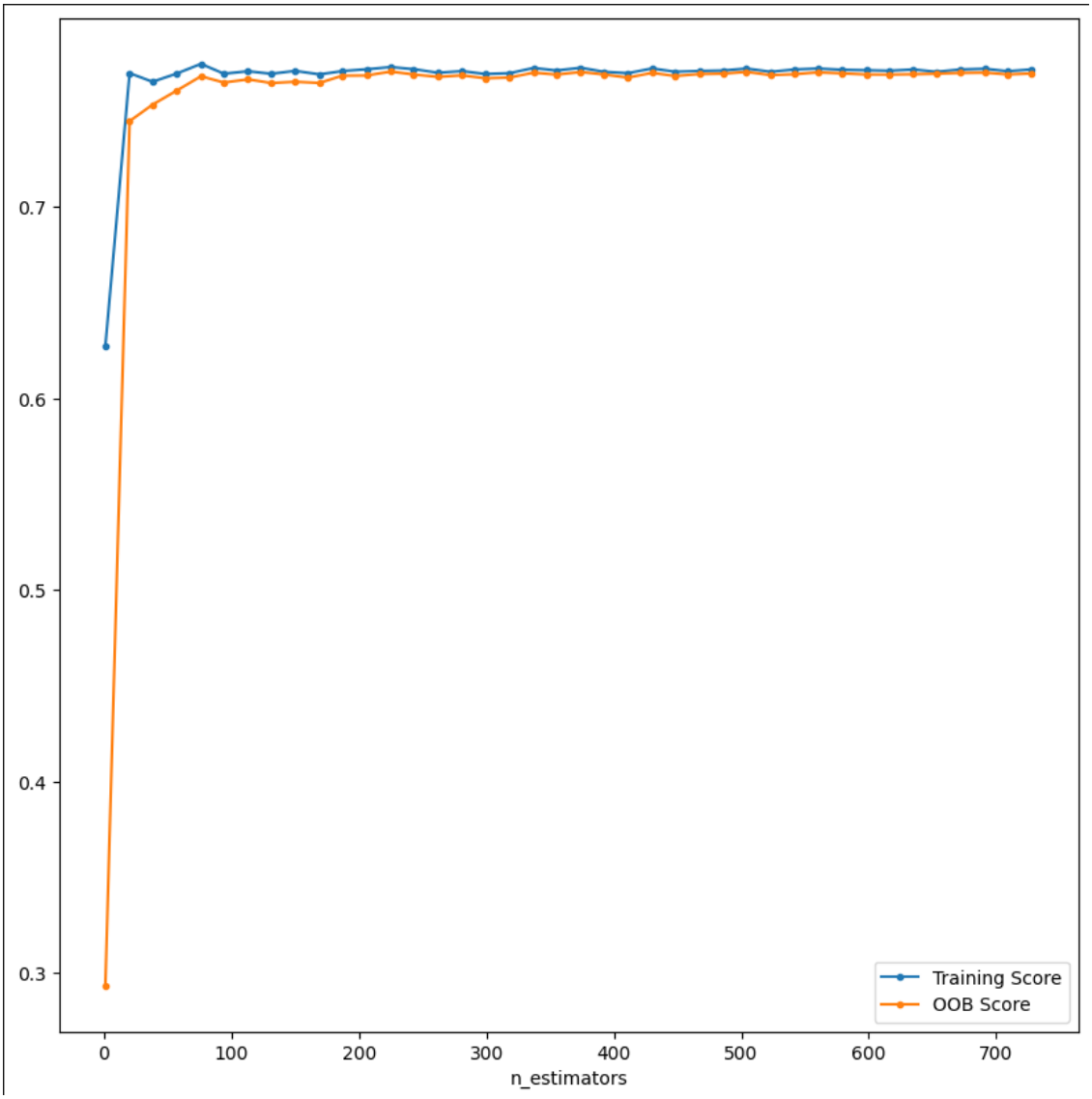

Figure 6. OOB Accuracy vs N Features

1.3 Experiment 6.3

AdaBoost Next we consider the *boosting* approach of the *AdaBoost* classifier. This was tested using 3000 estimators and using a decesion stump as the base model, then expanding it to a shallow decision tree of depths 2 and 4. The results OOB accuracies of those model are shown in *Table 3*. Suprisingly the accuracy increased as the tree grew which I found interesting since Boosting approaches leverage weak learners, and by increasing the base model tree depth we are strengthing the base model. Since we are only increasing the depth to a maximum for 4, the models are still considered weak. I would expect at some point the accuracy to start falling the base trees get deeper. However using 3000 esimators means that the training time to test that hypothesis would be unrealistic.

Tree Depth	OOB Acc
1 (stump)	0.593

Tree Depth	OOB Acc
2	0.668
4	0.818

Table 3. Adaboost OOB Acc

In a boosting model increasing the number of classifiers in the ensemble will lower the bias from each weak classifier as the models serially learn. With that being the case increasing the number of models should generally improve the models with deminishing returns. The weaker the base classifiers used in the ensamble are the more estimators will be needed to achieve good results. That can be demonstrated by *Table 3*, where *n_ estimators* is held constant.

I suspect there is a point however where there are to many estimators and the model becomes to complex for the ammount of data. Attemping to find that point was unphesible given how long the model took train with 3000 estimators.

Gradient Boost The final ensemble method to be considered during this experiment was *Gradient Boosting*. When trained on the dataset a model with 100 esimators and a 0.1 learning rate had a training accuracy of 0.901 and an 3-fold cross validation OOB accuracy of 0.874.

The OOB accuracy for the *Gradient Boosting* approach is on par with the best of the models tested thus far, however, this approach also had the lowest different between the training accuracy and OOB accuracy. That means this clasifier is performing just as well as the others while showing less signs of overfitting.

1.4 Final Evaluation 6.4

Of all the ensemble methods evaluated during this lab I choose the *Gradient Boosting* model with the hyperparameters used in *Experiment 6.3* as the best model tested. This is because the estimated accuracy from the OOB measurement was insignificantly differenet with the other top models, while as the same time having the least eveidence of overfitting. Which makes me trust the OOB accuracy more.

Applying that classifier to the held out test set resulted in a test set accuracy of 0.878, which as desired is near the expected accuracy from the OOB set. The test set confusion matrix is shown in *Figure 7*. It can be see that while it still makes some of the same mistakes discussed in *Exercise 6.1* it makes them less often, and has a high correct percentage for every single class compared to the confusion matrices in *Figures 1* and *2*.

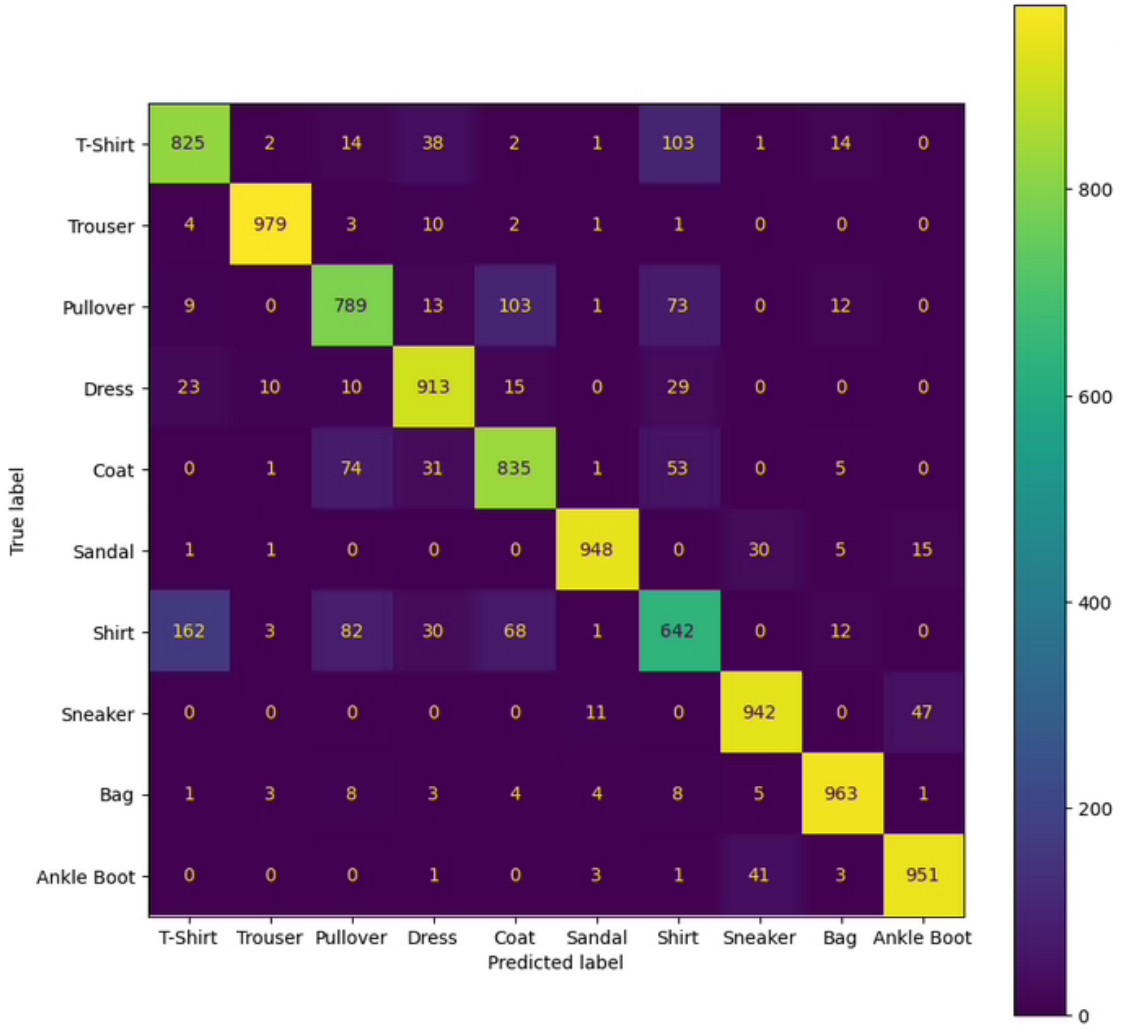

Figure 7. Gradient Boost on Test Set