**HW2 Randomized Optimization**

Student name:  
Kamran Mohtadi Haghighi

khaghighi7@gatech.edu

# Introduction

In this assignment I have selected two datasets namely dataset1 which represents the occurrence of a stroke in relation to bunch of factors such as age and some health history and the second dataset, dataset2, which represents the quality of wine driven from bunch of features such as acidity and alcohol content. I chose these datasets because the first dataset is a small one and has only 2 outcome classes namely 0 or 1 (No stroke or stroke) and the second dataset has more instances and it’s to be predicted value has a range of values which are more than 2. They are interesting for me because the combination of both gave me great use cases to practice ML as I am new in this field. I started working with more complex and unstructured datasets but soon I realized they are not practical and not useful for my learning purpose. (At least at this stage)

# Decision Trees

Using scikit learn I have “implemented” a decision tree model and carried out couple of analysis with my model. The first analysis I did was to check the relationship of the accuracy of the model to the maximum length of the tree. I carried out this experiment with both criteria of gini and entropy. For dataset 1 I found out that the optimum max depth is 4 or 5 depending on which criterion I have used (Entropy or Gini).

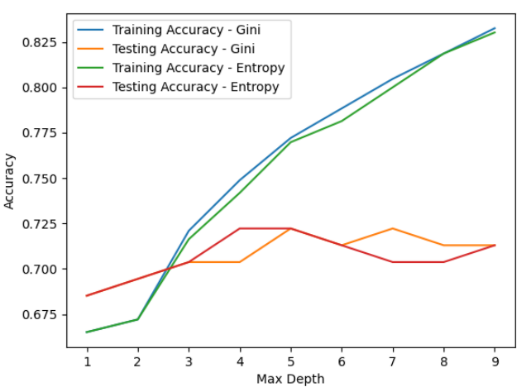
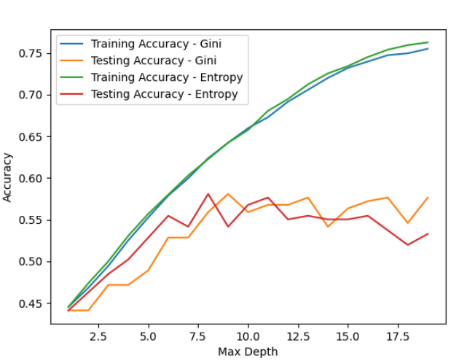


Figure - Dataset2 – Accuracy/ Max depth dependency

Figure - Dataset 1 - Accuracy/Max depth dependency

For dataset 2 however I observed that accuracy keeps increasing and doesn’t really reach a stable state so I have increased the range of max length to 20 and I could identify max length of 8 as optimum depth of the tree for dataset 2.

I did a similar experiment with minimum number of leaf samples (number of leaves in each classification) and observed that the number of samples doesn’t have an impact in the accuracy of my models.

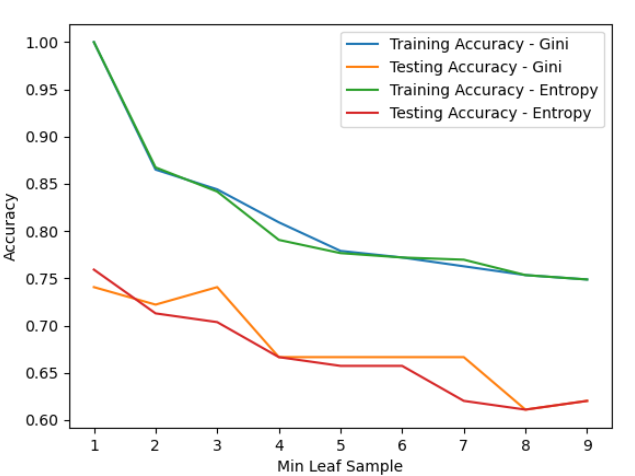
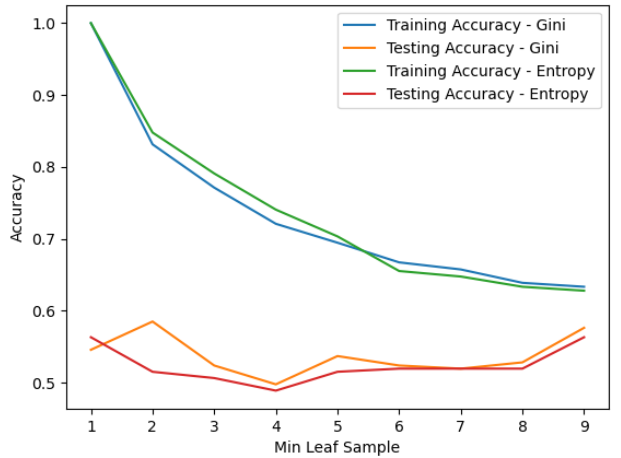


Figure 4 - Dataset 2

Figure 3 – Dataset 2

I had a doubt that maybe for dataset 2 with more leaf samples I maybe able to see a trend but the experiment also confirmed the independency even with a bigger range.

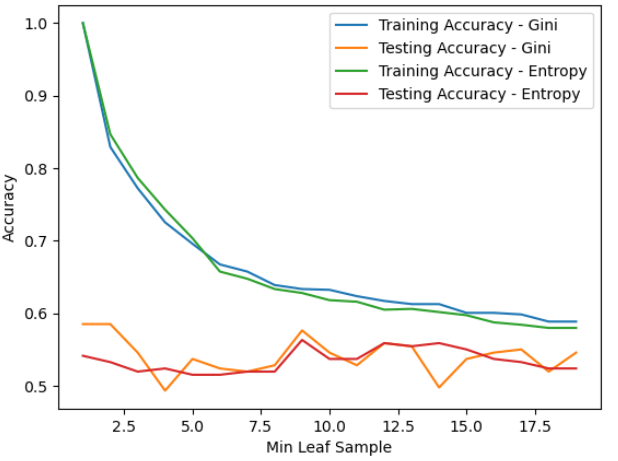


Figure 5 - Dataset 2 with a bigger range

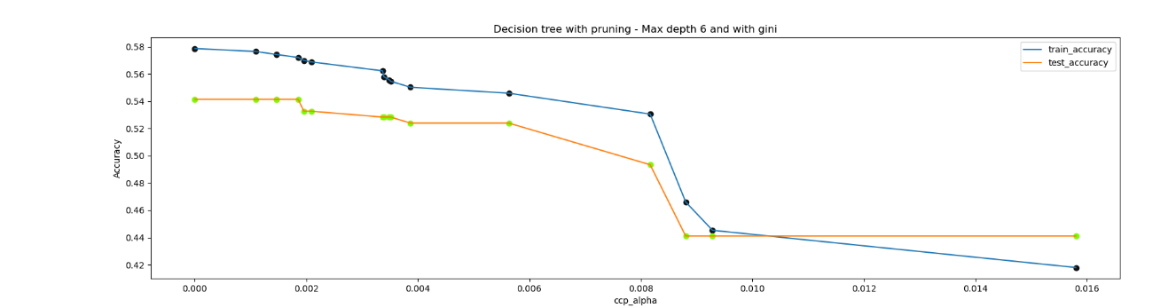
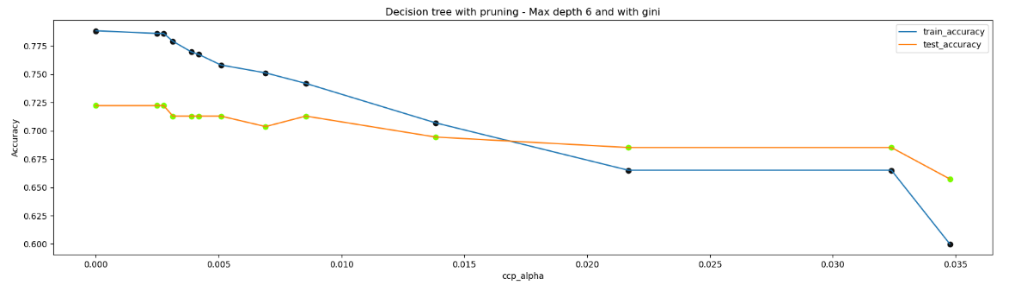
I also did some post pruning and identified the optimum cost complexity parameter – alpha - for each of the datasets. I have identified that alpha’s bigger than 0.0325 and 0.006 will drastically impact the accuracy of my models for data set 1 and dataset 2 respectively.

Figure 7 - Dataset 2

Figure 6 - Dataset 1

Lastly, I have illustrated the learning curves and performances of my models without and with cross validation as below.

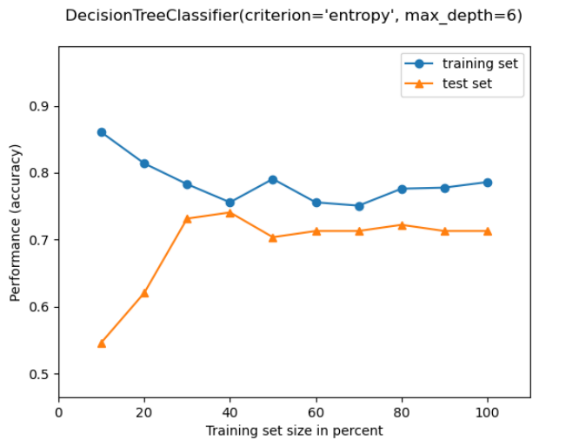
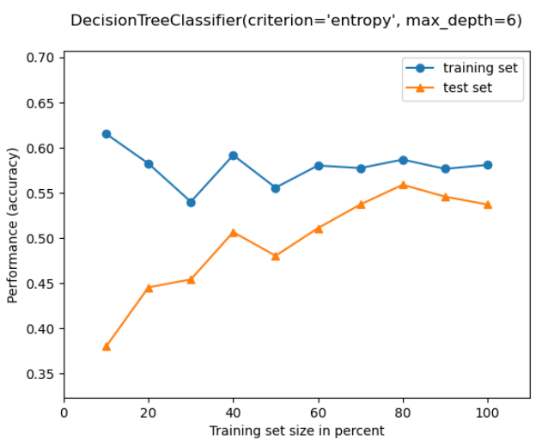
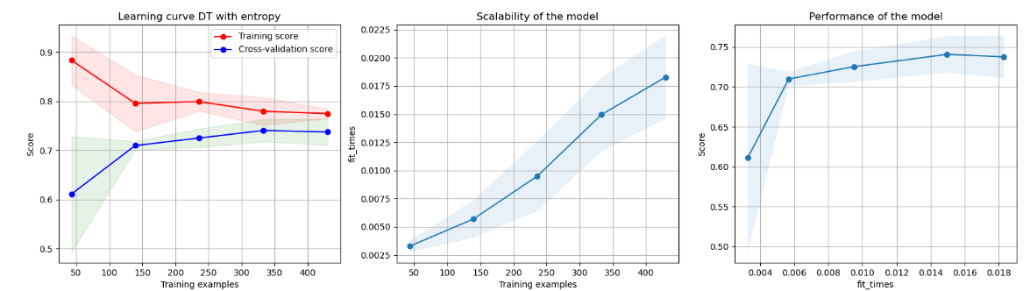


Figure 9 - Dataset 2 - Learning curve without cross validation

Figure 8 - Dataset 1 - Learning curve without cross validation

Although dataset 1 seems to be a good fit after already 40% of training, dataset 2 seems to reach it’s best performance by 80% of training size and after that is seems that it tends to overfit as the accuracy starts to decrease but the trend isn’t clear so I can’t say for sure. All in all, model 2 doesn’t look to be a good model as the accuracies of training and test data reach each other at slightly higher than 50%.

Now I have performed cross validation with KFold validation and 3 splits.

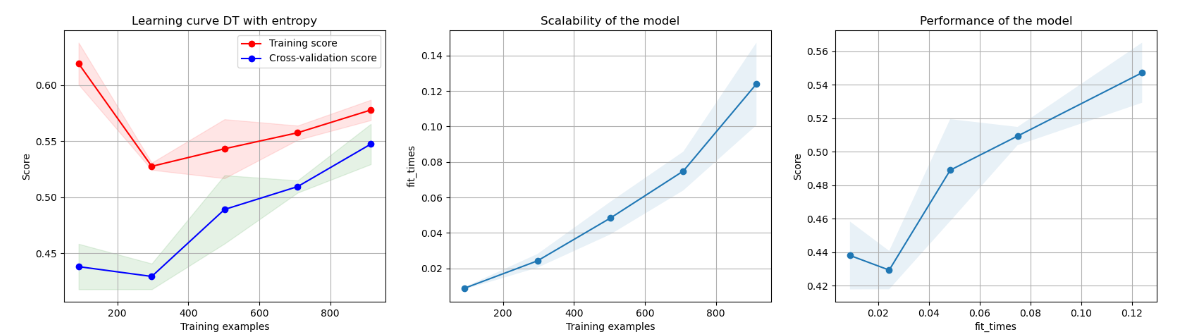


Figure 9 - Dataset 1 - Learning curve with cross validation

Figure 10 - Dataset 2 - Learning curve with cross validation

With introduction of cross validation, the models became more meaningful and without noise in the curves. The learning curves have a clear direction in both models now. In dataset 1 the performance doesn’t seem to have changed dramatically but in dataset 2 one can clearly observe the upward learning trend. And now it’s clearer that in data set 2 we have underfitting as the model still enhances itself with every iteration.

# Neural Networks

My next model is a multi-layer perception (MLP). For this experiment I have trained a MLP model with both relu and identity functions. I also tried to change the hidden layer and see how it impacts the accuracy of the model.

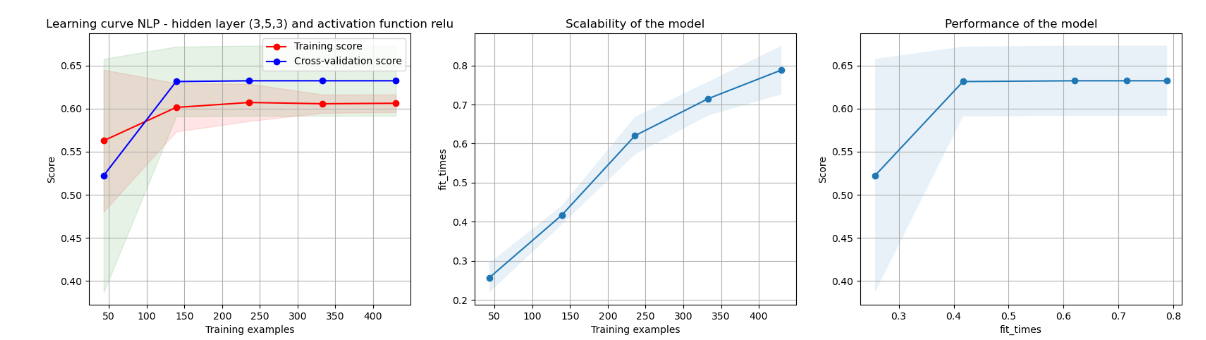
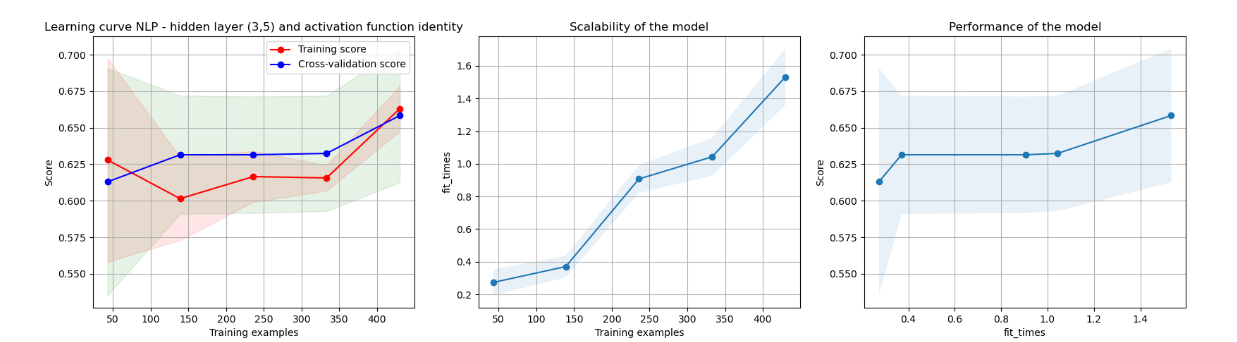
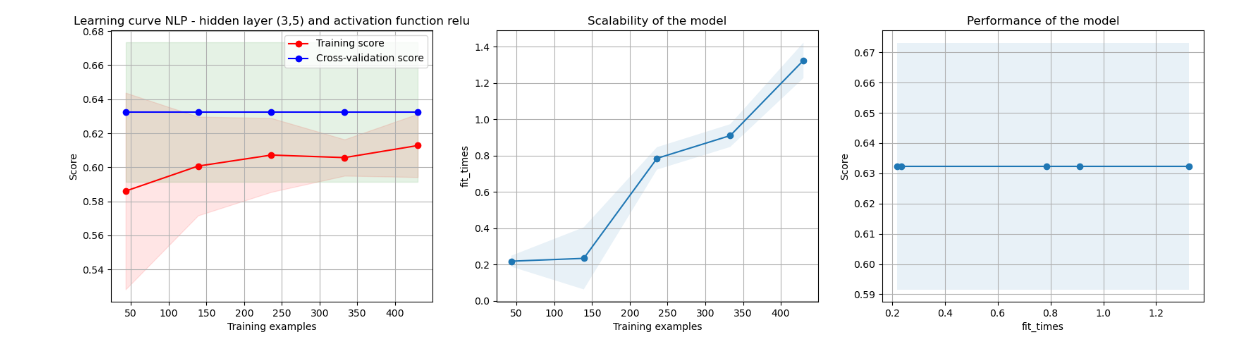
 

Figure 11 - Dataset 1 MLP experiments

Although I see changes by running the experiment with different parameters for hidden layer and the activation function like a clearer trend with activation function relu or a better learning with a wider hidden layer of (3,5,3) but since my validation dataset shows a better result than my training dataset I can see that my testing set is not representative and learns better the model than the training dataset. I have increased the testing set from 20% to 30% and 50% of total dataset but still I have observed unrepresentative learning curves. For dataset 2 I have observed also similar behavior.

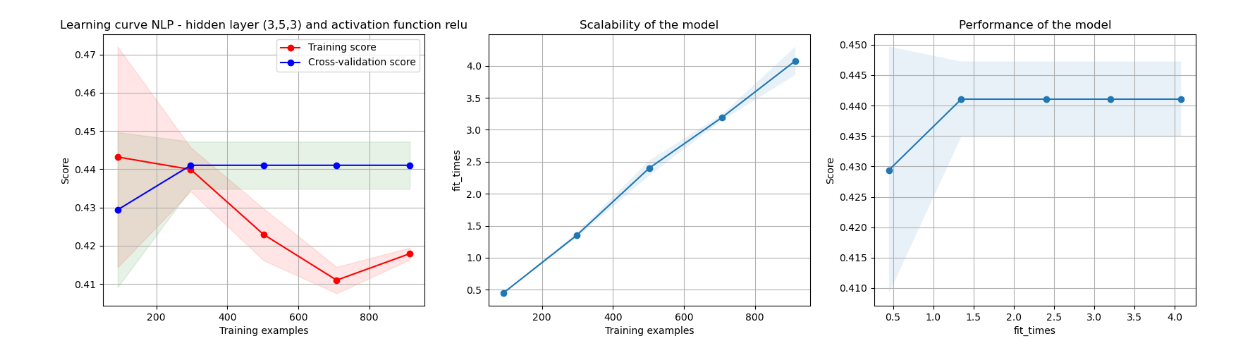
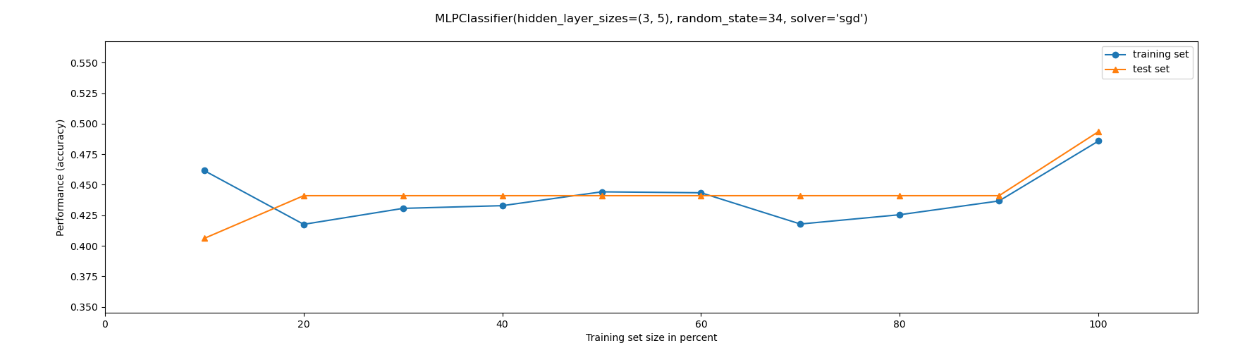
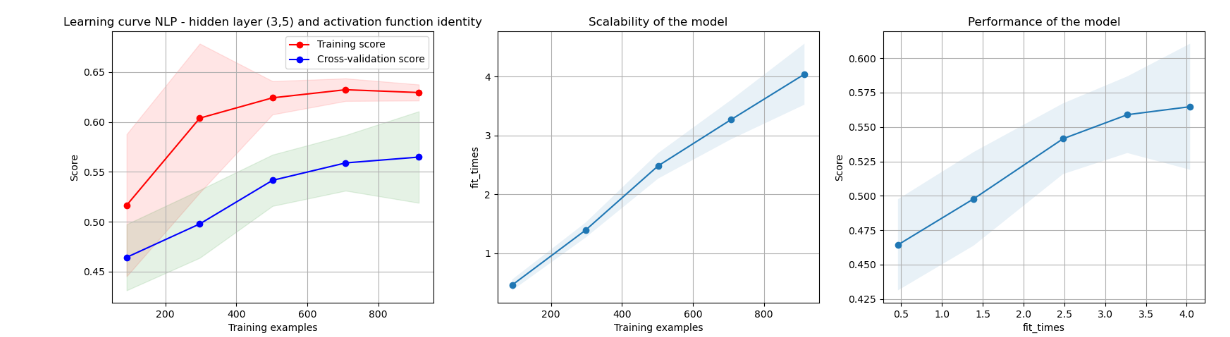
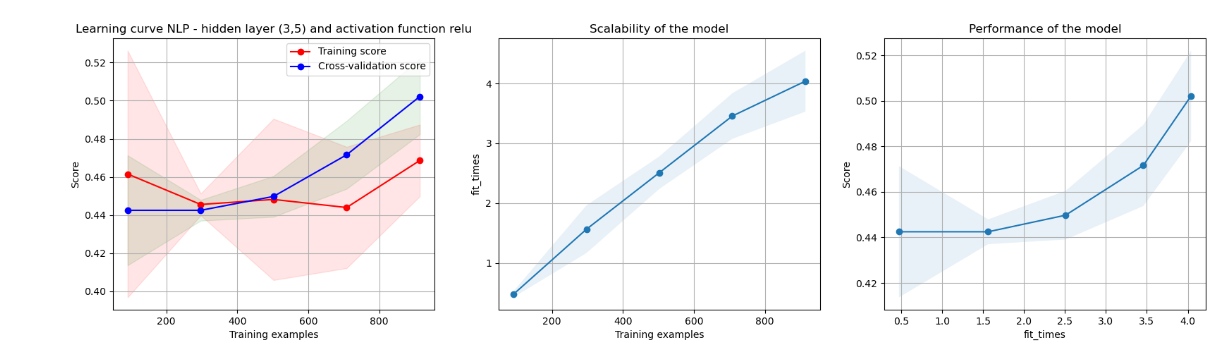
 With hidden layer of (3,5) and activation function identity I observed a kind of steady state but since the gap between training and test curve is big it is rather an unrepresentative training data set and with relu function I observed completely useless curves which are persumabley due to unrepresentative data . I ran this experiment also without cross validation and there I couldn’t see any learning model either. Again I argument it that datasets were not representative for NLP learner.

Figure 1- Dataset 2 MLP without CV

Figure 12 - Dataset 2 MLP experiments

It looks that MLP is not a good model for my datasets.

# Boosting

My next model is gradient boosting which I have “implemented” through Sci-kit library. First, I checked its performance in relation to maximum depth and minimum sample leaf. For my dataset 1 I haven’t observed any increase of accuracy with the length of the tree. The accuracy of the testing set seems to be steady irrespective of the maximum depth of the tree. For my dataset 2 I observed a slight increase in the accuracy until max depth of 4 where the level of accuracy stabilizes.

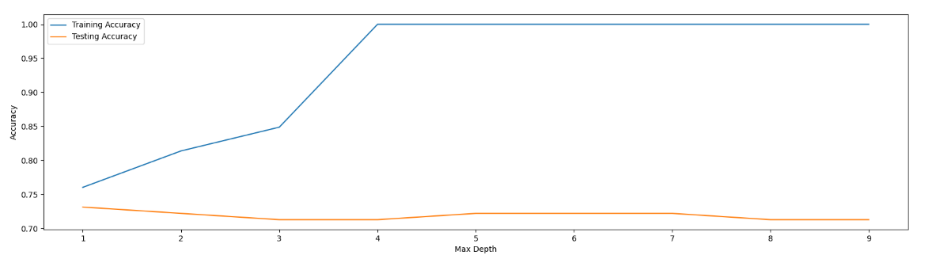


Figure 1 – Dataset 1 - Boosting/Max length dependency

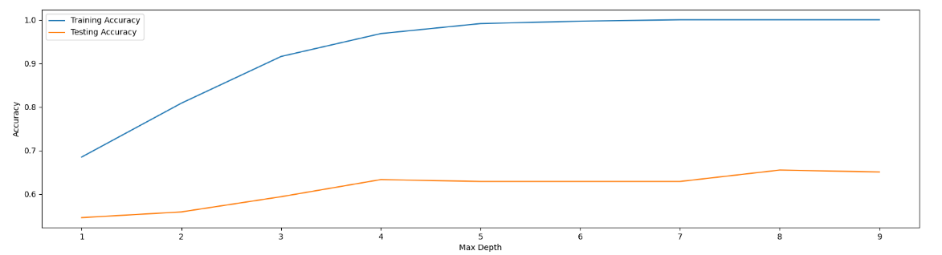


Figure 15 – Dataset 2 - Boosting/Max length dependency

Furthermore I also performed pruning on my booster model. And found out break even alpha values of .08 and .007 for dataset 1 and dataset 2 respectively.

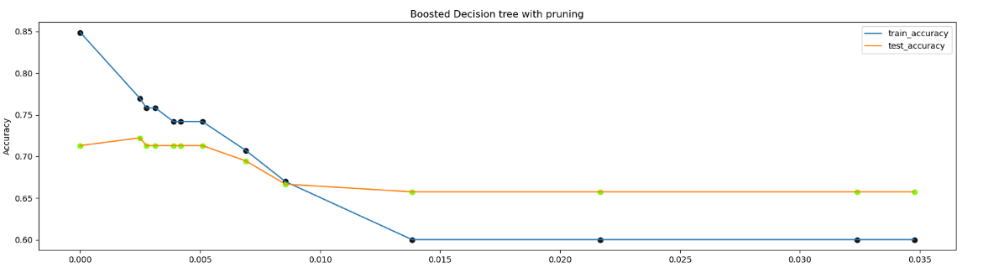


Figure 16 – Dataset 1 - Boosting/ Pruning

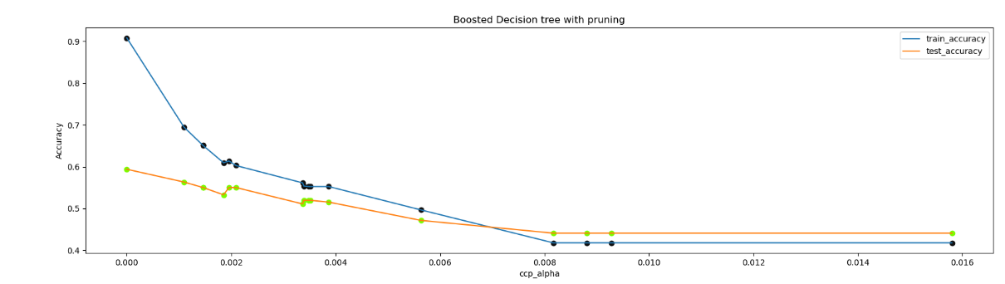


Figure 16 – Dataset 2 - Boosting/ Pruning

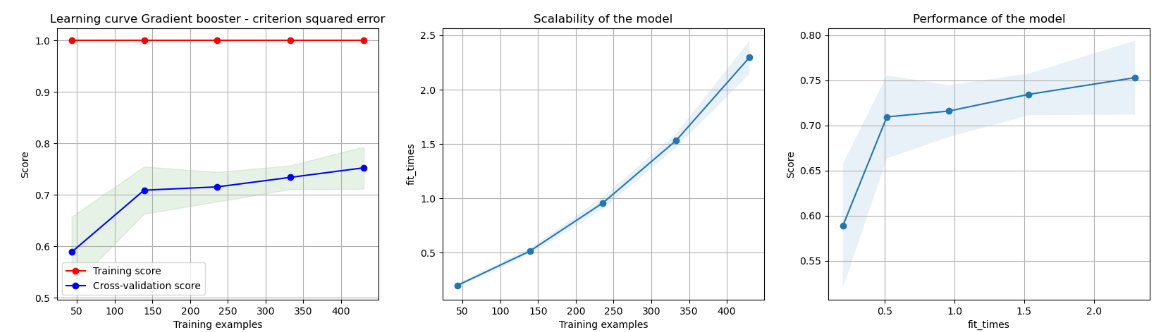
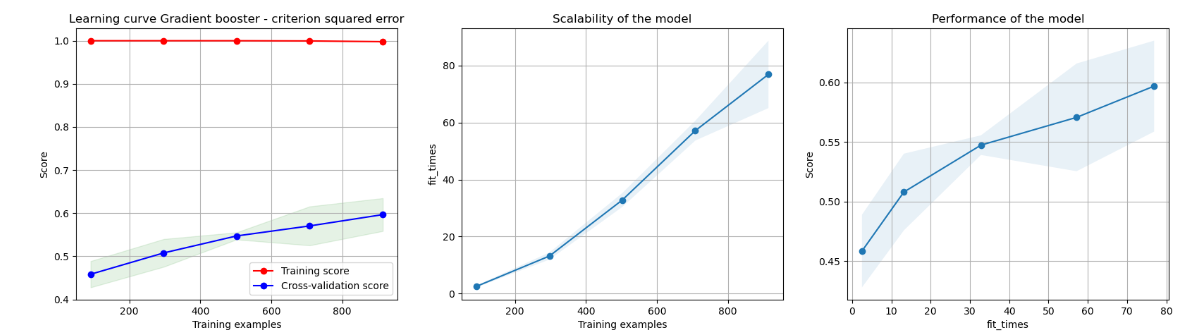
And the learning curves of my boosting model for dataset 1 and dataset 2 look like below:

Figure 17 – Dataset 1 and 2 – Boosting Learning Curve

For dataset 1 the booster seems to be a relatively good model as the validation learning curve increases drastically and then slightly stabilizes. There is still a gap to converge with training dataset, but training dataset is at accuracy of 1 so the gap can be accepted. Maybe with more data it would have become more mature. But Boosting certainly has improved the performance and it seems that with boosting the model learns better. For dataset 2 it doesn’t seem to be an underfitting case since the learning curve still increases.

# Support Vector MAchines

Next, I carried out the experiments with SVM model. I experimented with a support vector classifier (SVC) with both kernel functions of Poly and RBF. This method stands out because it performed extremely faster in comparison to Boosting and MLP models.

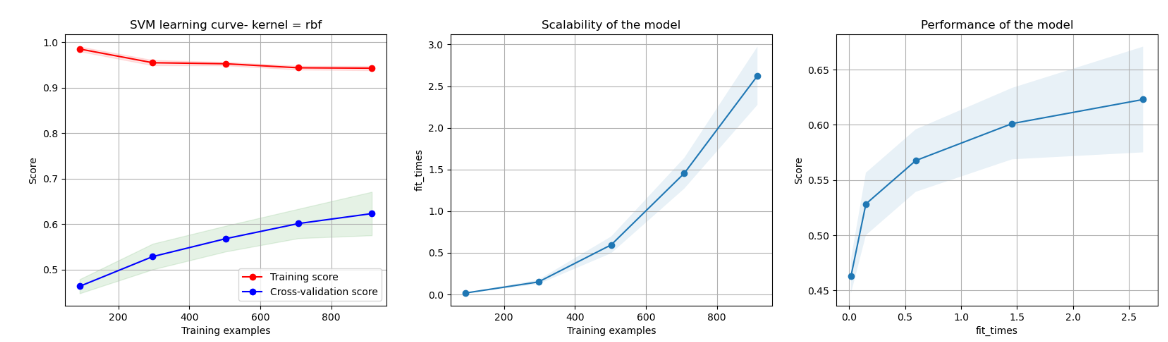
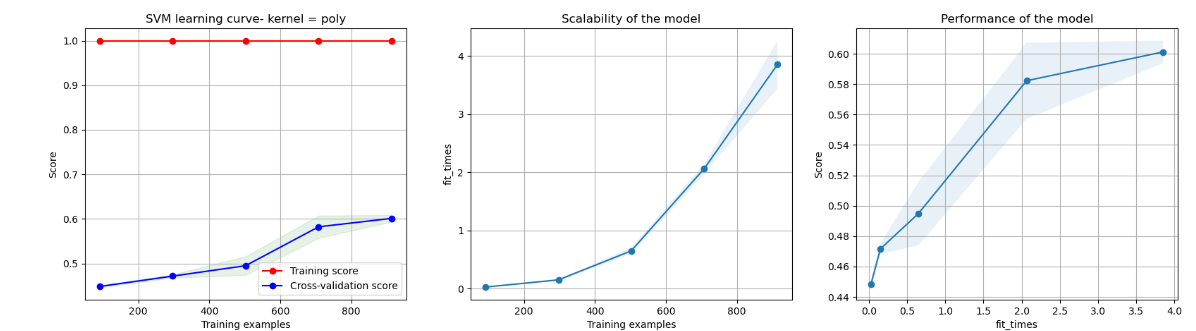
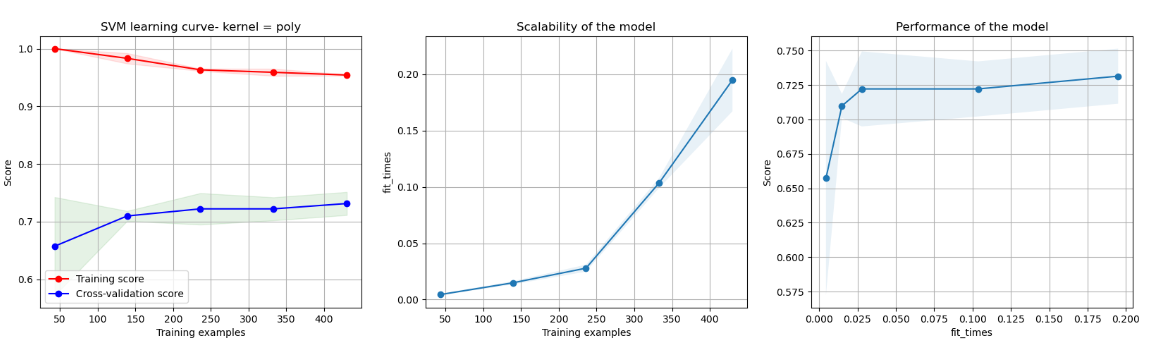
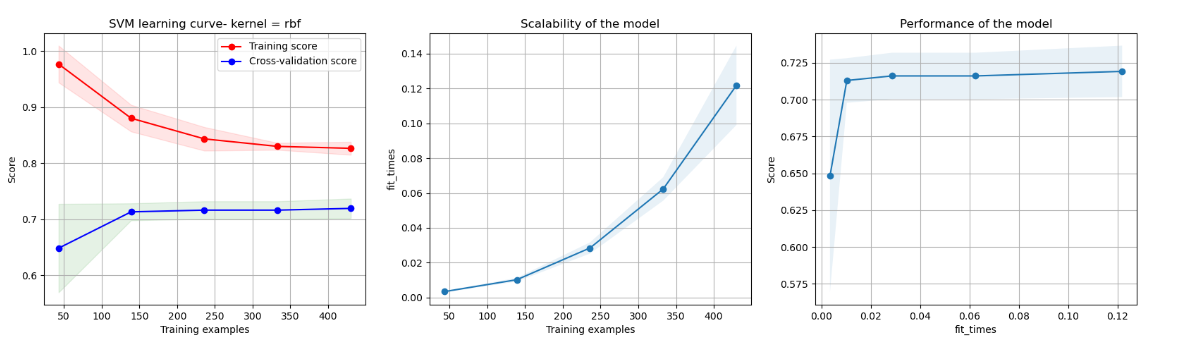


Figure 19 – Dataset 2 – SVN learning curve with poly and rbf kernels

Figure 18 – Dataset 1 – SVN learning curve with poly and rbf kernels

For dataset 1 the model seems to be “relatively” a good fit specially with rbf kernel as the learning increases in the beginning and then it reaches a steady state of course still far from convergence though. I probably need to add more data to make this model more mature. For dataset 2 the performance look worse compared to dataset 1. Presumably with a bit more data input cross validation line also reaches a steady state but I can not confirm with my current data set.

# K-Nearest Neighbors

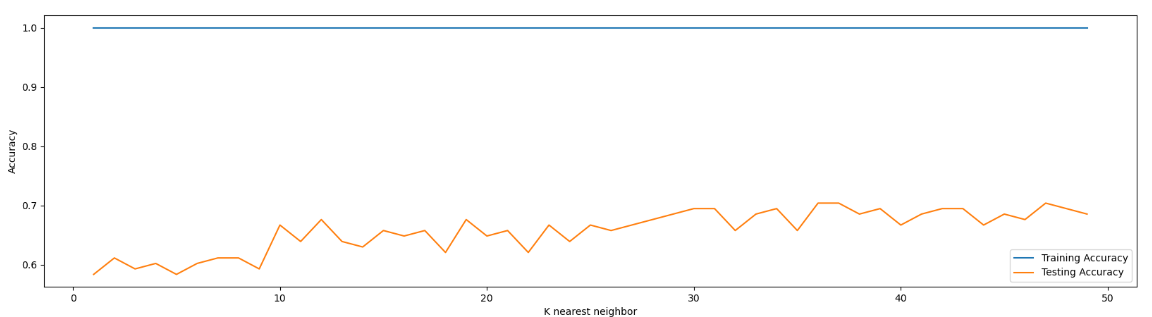
Lastly, I did the experiment with K nearest neighbor. I did this experiment with both of my datasets and identified the optimum number for K to reach the best accuracy. This model was the second fastest among all after the decision tree.

Figure 20 - Dataset1 - Optimum K

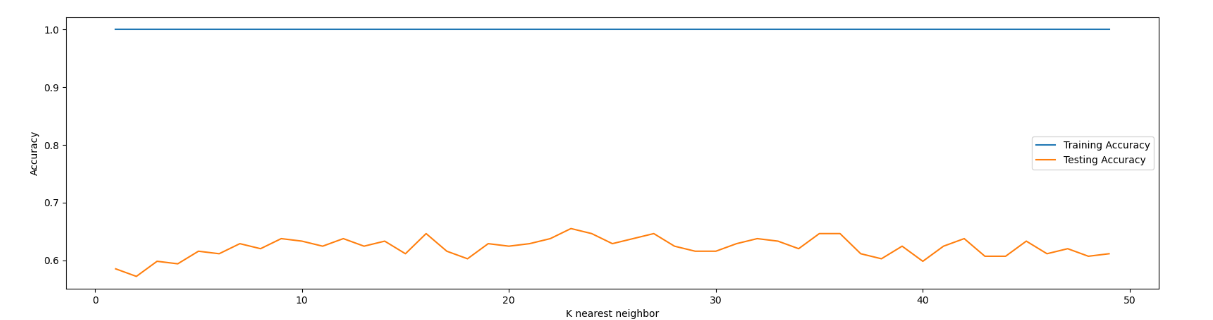
For dataset 1 there is a big jump with K=10 neighbors but after that the accuracy increases only slightly with the number of K.

Figure 21 – Dataset2 - Optimum K

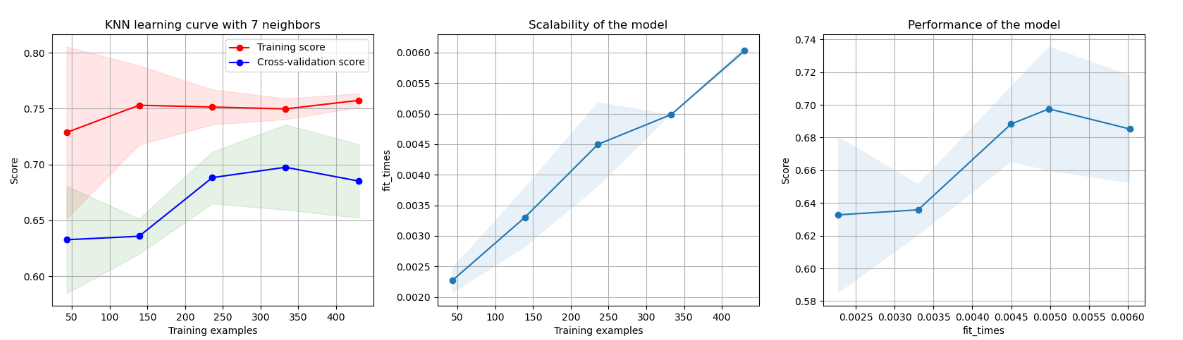
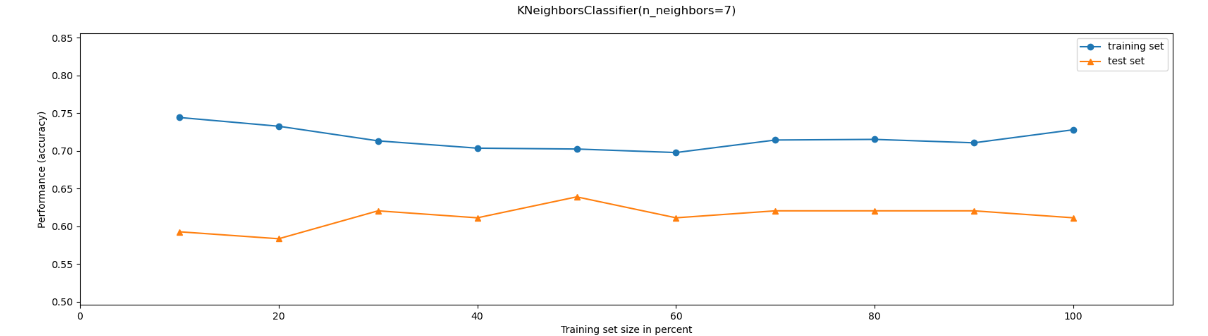
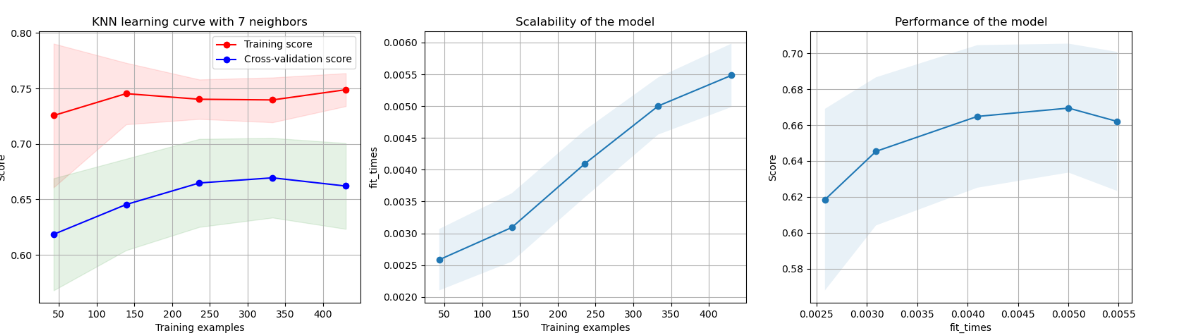
For dataset 2 I also observed an increase up to k=9 but after that the improvements in Accuracy are negligible and not worth the complexity of training time. Furthermore, I examined the learning curve without cross validation and with cross validation (Shuffle split) with 3 and 10 splits and observed the below results:

Figure 23 - Dataset 1 Learning curve with cv and split 3

Figure 22 - Dataset 1 Learning curve without cv

Looking at the results of the learning curves I can see that cross validation with split of 3 has improved the performance but cross validation with split of 10 doesn’t make it really any better in terms of performance but both cross validations reduce the noise in the learning curve. Aside from that I also see an overfitting after already 90% of the training examples.

Figure 24 - Dataset 1 Learning curve with cv and split 10

# Conclusions

For dataset 1, looking at the learning curves I see that Boosting has the best performance and accuracy after 40% of the training examples. In terms of speed, I also measured the computation time of each and observed that Decision tree followed by KNN and SVM were considerably faster than Boosing and MLP.

|  |  |  |
| --- | --- | --- |
| Dataset 1 | Time - cv split 3 | Time - cv split 10 |
| DT max depth =6 & gini | 0.2525596618652344 | 0.4152083396911621 |
| MLP HL (3,5) & relu | 2.686795473098755 | 9.475661039352417 |
| Gradient (Boosting) | 4.789404392242432 | 16.3657169342041 |
| SVM (rbf) | 0.9609847068786621 | 2.714372158050537 |
| KNN (K =7) | 0.8891174793243408 | 2.230470895767212 |

For dataset 2 SVM models reaches the highest accuracy for predictions in the shortest time of validation dataset. Boosting also performed almost as well as SVM (still low accuracy of 0.6) but took much more time. MLP didn’t quite learn well for dataset 2 and the data wasn’t representative. DT was underfit and required more data and presumably with more data it would have performed better.

|  |  |  |
| --- | --- | --- |
| Dataset 2 | Time - cv split 3 | Time - cv split 10 |
| DT max depth =6 & gini | 0.689910888671875 | 1.372737169265747 |
| MLP HL (3,5) & relu | 9.53848934173584 | 30.98004150390625 |
| Gradient (Boosting) | 154.82149124145508 | 599.5086171627045 |
| SVM (rbf) | 10.072510957717896 | 36.93666362762451 |
| KNN (K =7) | 1.3213317394256592 | 3.4904897212982178 |