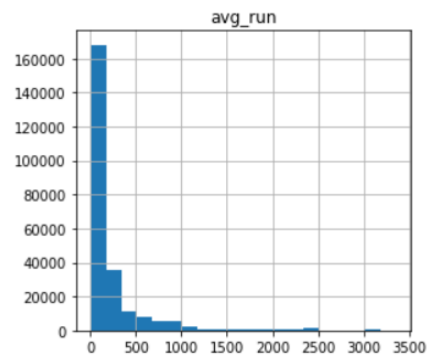
SVM, Decision Trees and Boosting Techniques

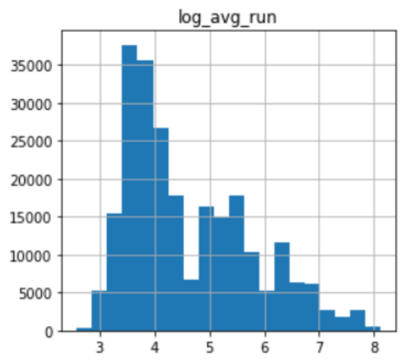
Data Description & Preparation

Dataset 1 - SGEMM GPU kernel performance Data

This data set measures the running time of a matrix-matrix product A*B = C, where all matrices have size 2048 x 2048, using a parameterizable SGEMM GPU kernel with 241600 possible parameter combinations. For each tested combination, 4 runs were performed.

- The dataset has 241600 observations with 18 features.
- None of the column contains any missing value.
- The target variable will be the average of the four run times.
- After computing the average, the data in the target variable is right skewed. Taking the log of the values reduces the skewness of the target variable.





- For this report, we will use all the independent variables to train our models.
- The data is divided into train test datasets in 70:30 ratio. The data is then scaled appropriately.

Dataset 2 - Letter Recognition Data

The objective is to identify each of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. The character images are based on 20 different fonts and each letter within these 20 fonts is randomly distorted to produce a file of 20,000 unique stimuli. Each stimulus is converted into 16 primitive numerical attributes (statistical moments and edge counts) which were then scaled to fit into a range of integer values from 0 through 15.

- The dataset has 20000 observations with 17 features.
- None of the column contains any missing value.
- The target variable will be column "Letter". The rest of the 16 variables, of datatype Integer, will be used as predictors.
- For this report, we will use all the independent variables to train our models.
- The data is divided into train test datasets in 80:20 ratio. The data is then scaled appropriately.

Support Vector Machines

Kernels

We will run the SVM algorithm on the datasets with three different kernels.

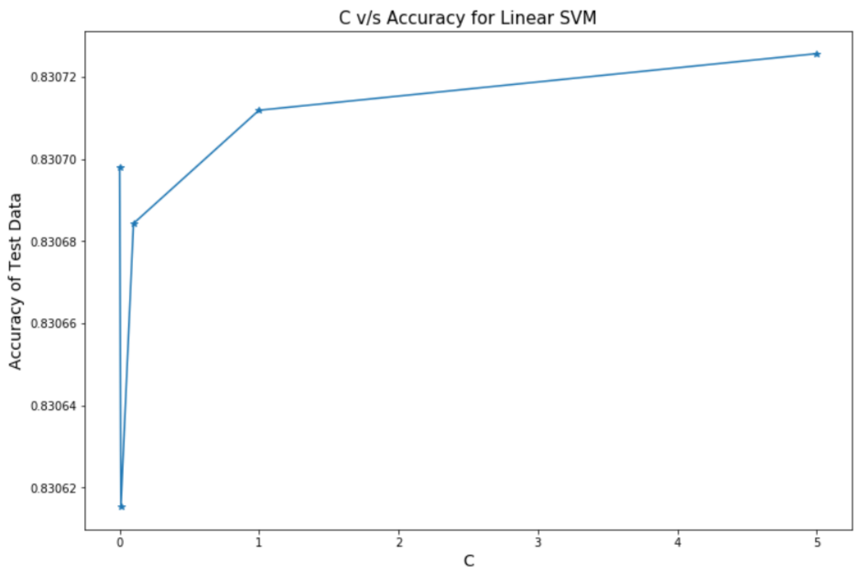
- **Linear** – The Linear kernel tries to separate the different classes by creating linear boundaries.
- **Rbf** – The Radial Basis Function kernel can choose non-linear decision boundaries.
- **Poly** – The Polynomial kernel can also choose non-linear decision boundaries depending on the degree of the polynomial function.

Hyper-parameters

- **C** - The C parameter trades off correct classification of training examples against maximization of the decision function's margin. For larger values of C, a smaller margin will be accepted if the decision function is better at correct classification, leading to overfitting. A lower C will encourage a larger margin, therefore a simpler decision function, at the cost of accuracy.
- **Gamma (Rbf kernel)** - The gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. If gamma is too large, it will lead to overfitting. When gamma is very small, the model is too constrained and cannot capture the complexity or "shape" of the data, leading to underfitting.
- **Degree (Poly kernel)** - The polynomial kernel of degree 1 leads to a linear separation. Higher-degree polynomial kernels allow more flexible decision boundaries.

Dataset 1

Linear Kernel



The maximum test data accuracy we achieve is **83.072%** at value of C as 5.

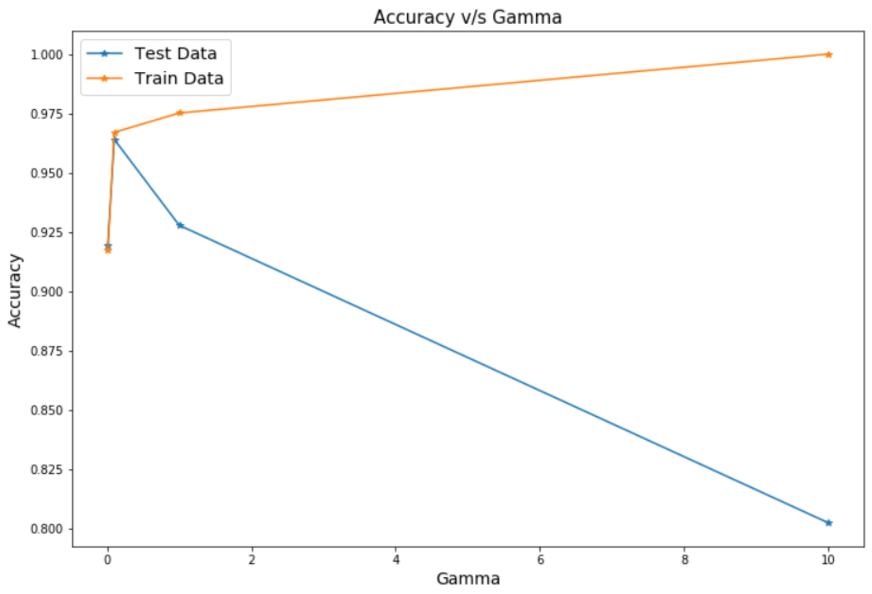
As expected, with increase in values of C, the cost of misclassification increases, and we get better accuracies in the test dataset.

However, the increase in accuracy when C increases from 1 to 5 is not very significant and is prone to overfitting.

Hence, the optimal value of C for this dataset is 1 where the accuracy is 83.071%.

Rbf Kernel

Here we have experimented with the different values of gamma and have kept the C constant as 1. The values of C as 1 is neither too high nor too low to increase the variance or bias and is optimal for our experiment.



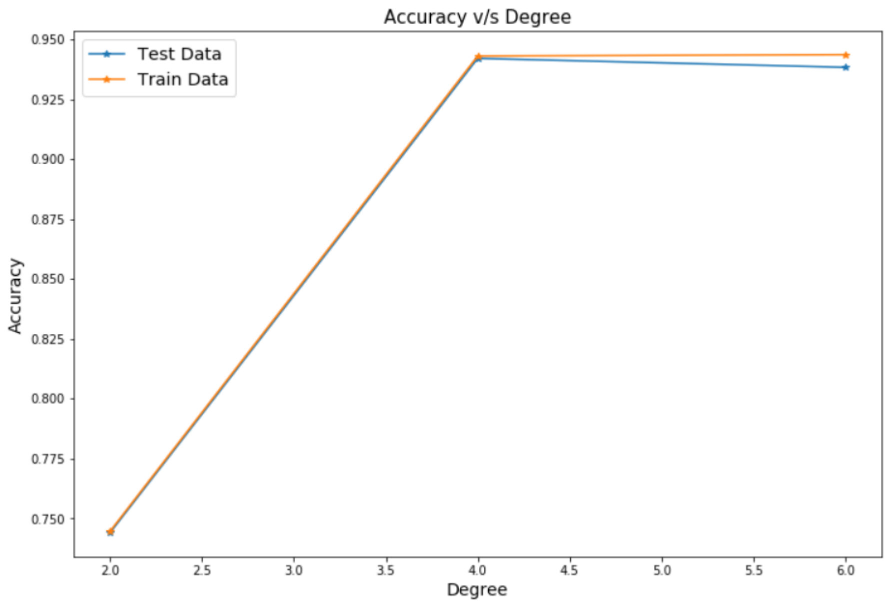
The maximum test data accuracy is **96.4%** at value of gamma as 0.1.

As seen in the graph, the accuracy for test data set starts decreasing with increase in gamma value after 0.1. The decrease is considerable for gamma 10. This is because large gamma is prone to overfitting. The train data set accuracy at gamma 10 is 100%.

Therefore, the optimal gamma for this dataset is 0.1, which is high enough to gauge the shape of the data and low enough to not overfit the training dataset.

Poly Kernel

Here we have experimented with the different values of degree and have kept the C constant as 1.



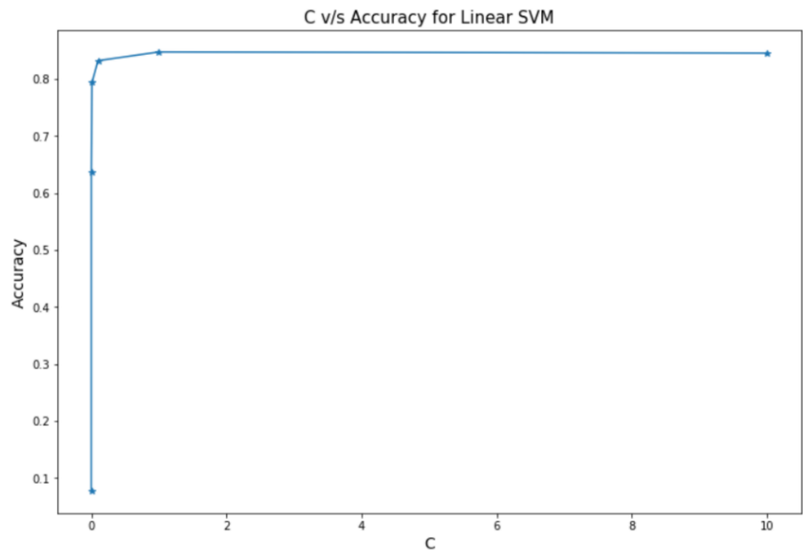
The maximum test data accuracy is **94.2%** at value of degree 4.

We see the higher degree (6) polynomial function achieves slightly lower test accuracy because it allows more flexible decision boundaries and therefore is prone to overfitting.

In similar manner, lower degree (2) also has lower accuracy in comparison because it cannot fully accommodate the shape of training data. The highest accuracy is achieved at degree of 4, which gives the right fit.

Dataset 2

Linear Kernel



The maximum test data accuracy we achieve is **84.77%** at value of C as 1.

As expected and explained before, the accuracy in the test data set increases with increase in C.

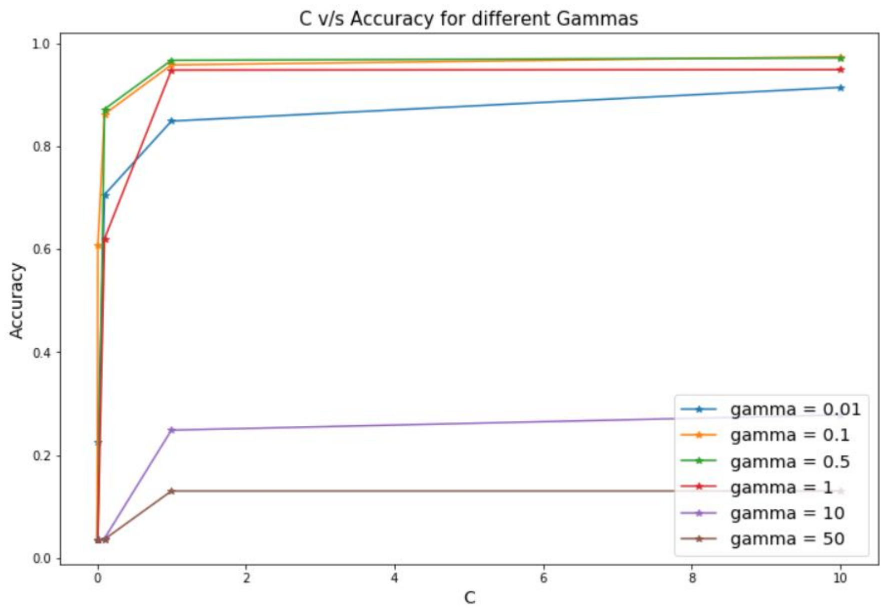
There is no significant increase in accuracy after C = 1 and large values of C are prone to overfitting.

Hence, the optimal value of C for this dataset is 1.

RBF Kernel

Here we are experimenting with varying the values of C and gamma to find the optimal combination.

The maximum test data accuracy we achieve is 97.4% at C as 10 and Gamma as 0.1.

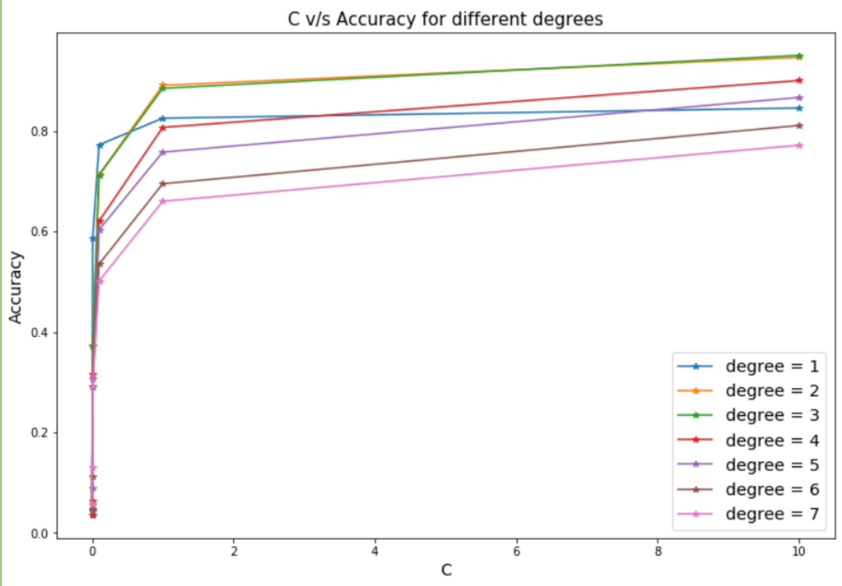


From the graph, we notice that the accuracy increases with increase in C as expected. However, for all the values of gamma, there is no significant increase in accuracy after C as 1.

For large values of gamma (10 and 50), we achieve very low accuracies. This is because the radius of the area of influence of the support vectors only includes the support vector itself and no amount of regularization with C was able to prevent overfitting. On the other hand, very low value of gamma (0.01) also gave less accuracy as the model is underfit. We get highest accuracies at the optimal values of gamma in the range of 0.1 to 1.

Poly Kernel

Here we are experimenting with varying the values of C and degree to find the optimal combination.



The maximum accuracy achieved is **95.07%** at C as 10 and Degree as 3.

As expected, the accuracy increases with increase in C.

We see that higher degree (5, 6 and 7) and lower degree (1) polynomial functions achieve lower accuracies because of overfitting and underfitting respectively. The highest accuracies are achieved at degrees of 2 and 3, which give the right fit.

Decision Tree

Information Gain Function

Decision trees recursively split features based on the target variable's "purity". Each split is optimized to gain maximizing purity. Gini index measures how often a randomly chosen element from the set would be incorrectly labeled. Entropy is also another way of measuring purity, but it is more computationally heavy due to the log in the equation. Therefore, we will use Gini Index to measure information gain in our Decision Tress, Bagging and Boosting algorithms.

Dataset 1

We achieve test data accuracy of **99.1%** for a fully grown decision tree.

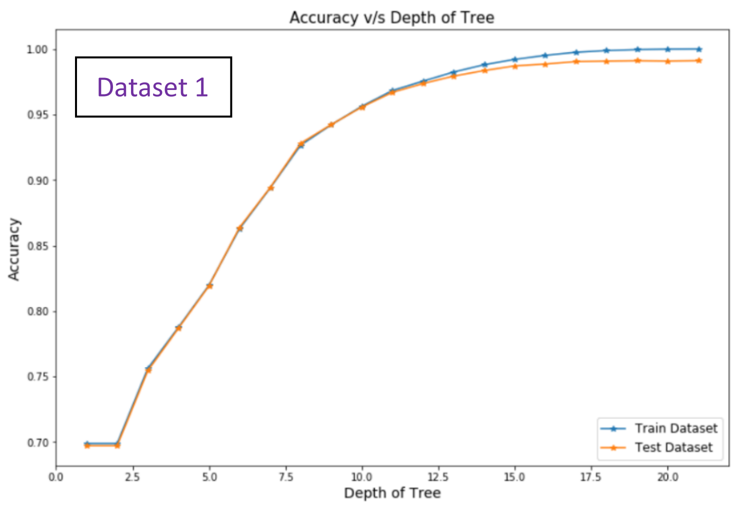
Dataset 2

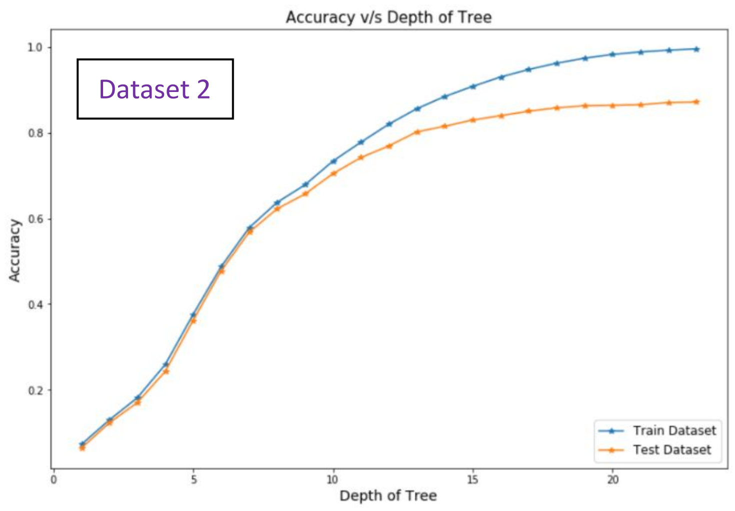
We ran Grid Search with the below parameters to find the best ones:

Maximum Leaf Nodes: [1200, 1000, 800, 600, 400, 200]
Minimum Samples on a Leaf: [1,2,3,4,5]
Criterion: ['entropy', 'gini'],
Maximum Depth of the Tree: 1 to 30

The best parameters we get are Maximum Depth 24, Maximum Leaf Nodes 1200, Minimum Samples on Leaf 1 and Criterion Gini. These parameters are found by comparing the accuracy scores on the training data set. The accuracy of **86.47%** is achieved on test dataset using these parameters.

Limiting Depth of the Tree



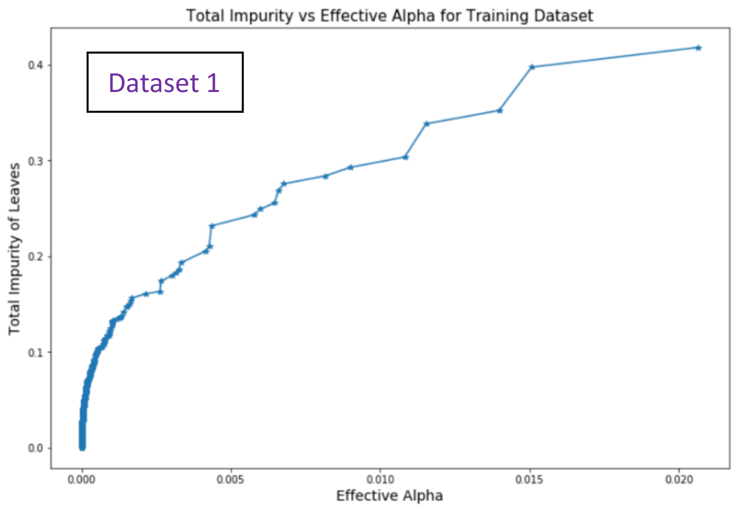


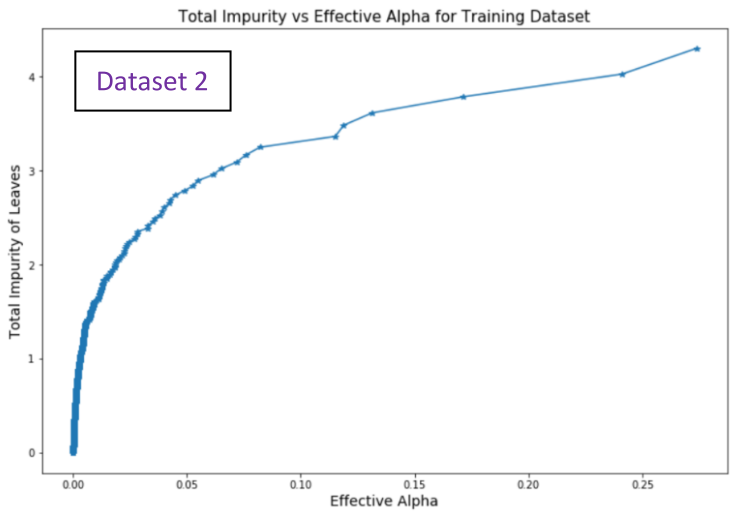
For dataset 1, we achieve maximum test data accuracy at depth of 21. For dataset 2, we achieve maximum test data accuracy at depth of 23.

We notice that both training and test data accuracies increase with increase in the Maximum Depth allowed for both the datasets. It is expected for training data, because with increase in depth, the purity of the leaf nodes also increases, leading to higher accuracies. However, the test data accuracies are getting constant after a certain maximum depth. In dataset 1, there is no significant increase in test data accuracy after the Maximum Depth of 16. In dataset 2, this is happening at Maximum Depth of 18. Therefore, we can stop growing the trees at these depths and get good accuracies.

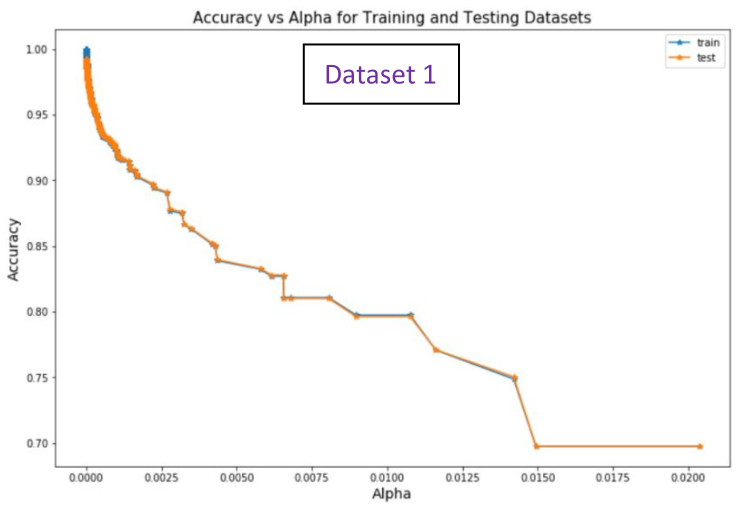
Pruning

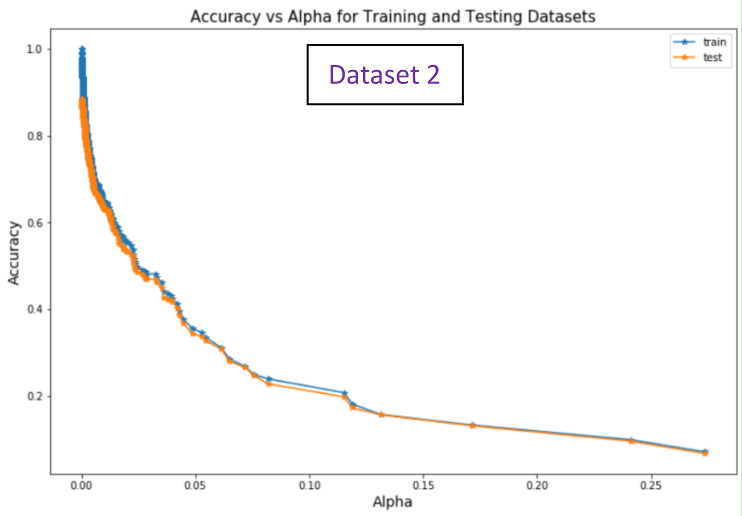
Cost complexity pruning can control the size of a tree. In DecisionTreeClassifier, this pruning technique is parameterized by the cost complexity parameter, ccp_alpha. Greater values of ccp_alpha increase the number of nodes pruned, thus creating a decision tree that generalizes better.





In the above two graphs, as alpha increases, more of the tree is pruned, which increases the total impurity of its leaves.



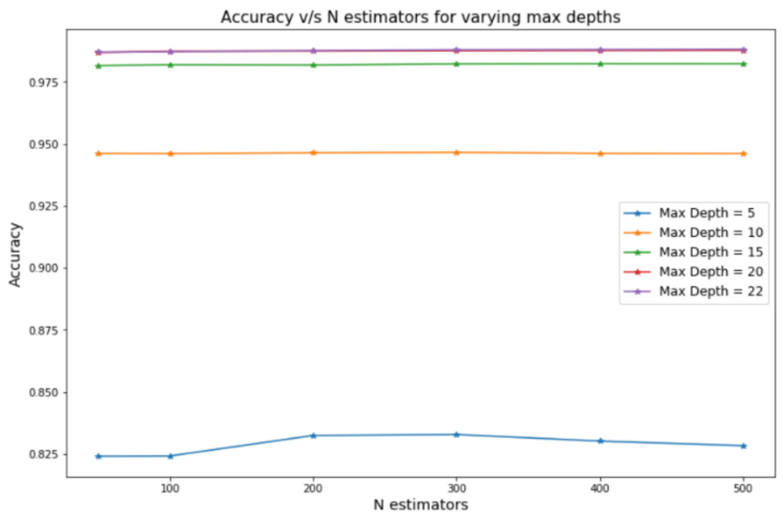


In the graphs above, as expected we see a decrease in train data accuracy because of increase in the impurity of the leaves. However, pruning the tree is also leading to decrease in the test data accuracy. Hence, pruning is not helpful for these datasets.

Random Forest (Bagging)

In a Random Forest several decision trees, are grown on bootstrapped samples of the training data. The decision trees are not pruned because we are less concerned about individual trees overfitting the training data. Moreover, the learning algorithm is limited to a random sample of features for splitting at each split in individual decision trees. The results of the estimators are averaged together to give a low-variance algorithm.

In the below sections we have experimented on bagging by varying the number of estimators and Maximum Depth of the individual trees. We will see that fully grown trees perform best.



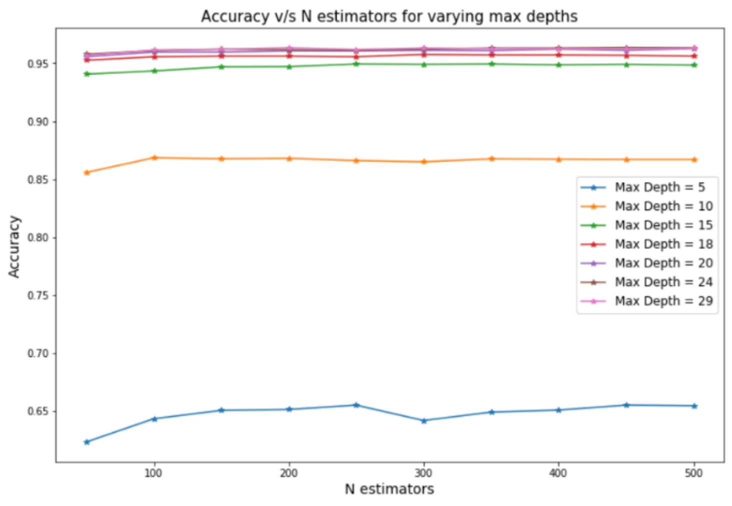
Dataset 1

We achieve maximum test data accuracy of 98.8% at the Maximum depths of 20 and 22.

For this dataset, bagging is giving comparable result to using a single decision tree. The accuracy we got from single tree was 99%.

Also, increasing the number of estimators is not significantly increasing the test data accuracy at any given depth.

Dataset 2



We achieve maximum test data accuracy of **96.3%** at the Maximum depths of 24 and 29.

We see that, using bagging the accuracy is considerably increased than using just one decision tree.

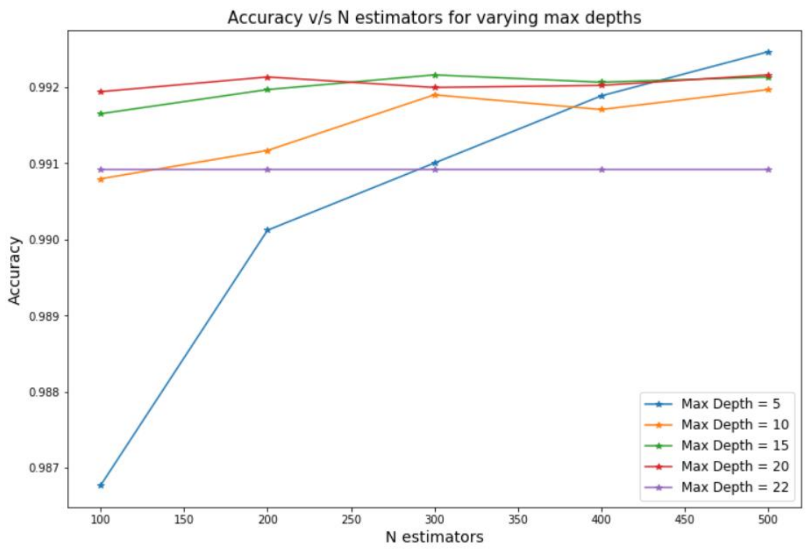
Here also, the accuracy does not considerably increase with increase in the number of estimators. Also, the models with more depth allowed perform better than those with depths of 5 and 10. This is because models with 5 and 10 depth are underfitting the model.

Boosting

The objective of the boosting algorithms is to iteratively learn from weak classifiers and add them together to form a strong classifier. In each step the misclassified data points gain weight and correctly classified data points lose weight. Therefore, in each iteration the learning algorithm chooses a subset of data which is hard to classify based on weight and tries to improve on them.

For Boosting we have used AdaBoostClassifier of the sklearn package in python. In the below sections we have experimented on boosting by varying the number of estimators and Maximum Depth of the individual trees.

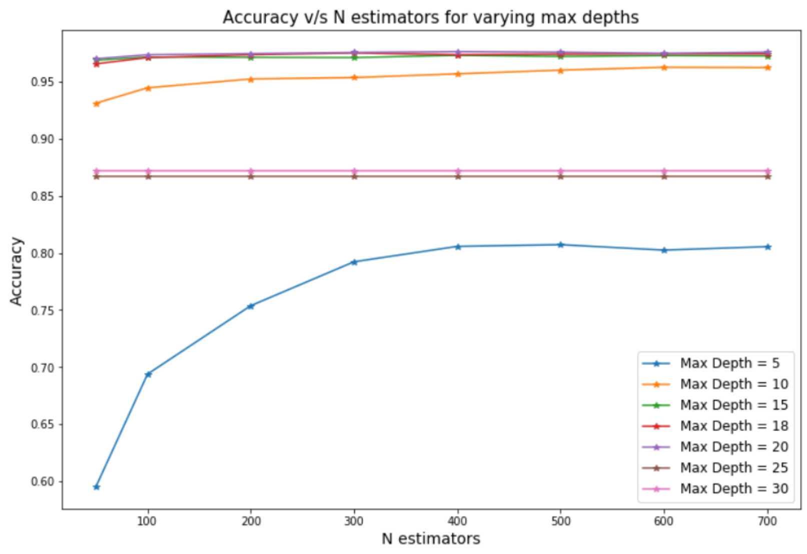
Dataset 1



We achieve maximum test data accuracy of **99.24%** at Max Depth 5 and 500 estimators. At the same time, we achieve an accuracy of 99.1% for full grown trees with just 100 estimators.

Therefore, it is a tradeoff between depth and number of estimators. It takes more number of estimators to achieve comparable accuracy, if we restrict the depth of the individual trees. On the other hand, if we let the trees grow, it takes less number of estimators and is computationally cheap.

Dataset 2



We achieve maximum and optimal test data accuracy of **97.52%** at Maximum Depth of 18 with 300 Estimators.

We observe that depth does impact the performance of the model. Too low depths (5 and 10) and too high depths (25 and 30) are doing worse than optimal depth (15 to 20) models.

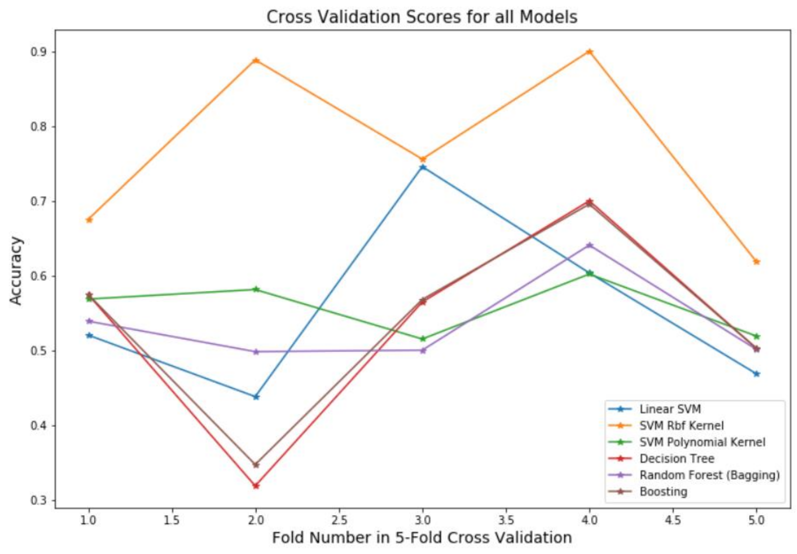
One interesting thing to note is, boosting is doing better than bagging and giving an accuracy of 80% even if we restricted the depth at 5. In bagging the maximum accuracy we got at 5 depth was 65.5%. This shows that by trying to work more on harder to classify datapoints, boosting is doing better than the averaging approach of bagging.

Cross Validation

Cross validation is a model evaluation method. We randomly split our entire dataset into k folds. For each k-fold in the dataset, we will build our model on k-1 folds of the dataset. Then, test the model to check the effectiveness for kth fold.

In the experiments below, we have checked the effectiveness for all the algorithms above by doing 5-fold cross validation. We have reported the test accuracies for each of the fold. The models are not overfitting and performing good if the accuracy scores don't vary by a huge amount for a single algorithm.

Dataset 1



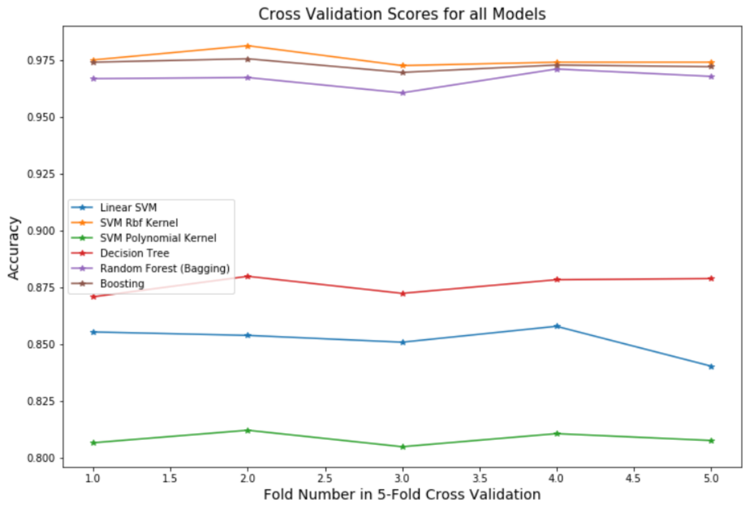
Here the models do not have consistent scores, which means that these models are prone to overfitting.

Anyhow, the accuracy given by the SVM model with Rbf kernel is the best followed by Boosting.

The graphs of Random forest and the Polynomial kernel SVM are the most consistent.

More consideration should be given to these models to help them generalize better.

Dataset 2



All the models are giving consistent scores, which is good.

The best performing models are SVM with Rbf kernel and Boosting, followed by Random forest.

How powerful is a decision tree can be seen by even a single decision tree is performing better than Linear SVM and SVM with Polynomial kernel.

Conclusion

Dataset 1

Algorithm	Test Data Accuracy %age	Parameters
SVM (Linear)	83.07	C = 1
SVM (Rbf)	96.4	C =1, Gamma = 0.1
SVM (Polynomial)	94.2	C = 1, Degree = 4
Decision Tree	99.1	Fully Grown
Random Forest	98.8	Estimators = 500
Boosting	99.1	Estimators = 100

Dataset 2

Algorithm	Test Data Accuracy %age	Parameters
SVM (Linear)	84.7	C = 1
SVM (Rbf)	97.4	C = 10, Gamma = 0.1
SVM (Polynomial)	95.1	C = 10, Degree = 3
Decision Tree	86.47	Max Depth = 24
Random Forest	96.3	Max Depth = 24, Estimators = 300
Boosting	97.52	Max Depth = 18, Estimators = 300

We see that for both datasets, the Rbf and Polynomial kernels perform significantly better than the linear kernel in SVM. This is because they can accommodate the shape of the data better by forming non-linear decision boundaries.

In the same manner, the boosted and the bagged version perform better than single decision trees because they are less prone to overfitting and have the power of several decision trees behind to give the prediction.