

## Assignment 1 (Supervised Learning)

---

### Dataset A: Breast Cancer Wisconsin (Diagnostic) Dataset

This breast cancer dataset is obtained from UCI Machine Learning Repository and used as the first dataset (Dataset A) for my comparative analysis of Supervised Learning algorithms.

This dataset falls under the Classification type problem where the target attribute (i.e., class) assumes to categorical values 2 for Benign and 4 for Malignant.

The distribution of the 2 classes among the 569 instances are as follows. This shows that the distribution of the 2 class values is well balanced.

Benign: 357 (62.74%); Malignant: 212 (37.26%)

#### Problem Statement:

This purpose of this analysis is to find general trends that might help us in doing model and hyper-parameter selection. Also, the goal is to classify whether the breast cancer is benign or malignant based on a set of features which were computed from digitized images of FNA tests on a breast mass.

#### Data Preprocessing:

Remove the first column 'id'. Used Python LabelEncoder to transform the Class values "B"  $\rightarrow$  0 and "M"  $\rightarrow$  1

### Dataset B: Statlog (Vehicle Silhouettes) Dataset

This vehicle silhouettes dataset is obtained from UCI Machine Learning Repository and used as the second dataset (Dataset B) for my comparative analysis of Supervised Learning algorithms. The dataset contains 946 instances with 18 attributes.

The distribution of the 4 classes among the 946 instances are as follows. This shows that the distribution of the 4 class values is well balanced.

Opel: 240 (25.37%); Saab: 240 (25.37%); Bus: 240 (25.37%); Van: 226 (23.89%)

#### Problem Statement:

This dataset falls under the Classification type problem. The purpose is to classify a given silhouette as one of four types of vehicle (opel, saab, bus and van), using a set of features extracted from the silhouette. Also, it will help us to find general trends that might help us in doing model and hyper-parameter selection.

#### Data Preprocessing:

The data shall be preprocessed to encode the 4 Categorical Class values (i.e., opel, saab, bus and van) into integer values using Python LabelEncoder API.

Opel  $\rightarrow$  0; Saab  $\rightarrow$  1; Bus  $\rightarrow$  2; Van  $\rightarrow$  3

Also, data scaling has been done using Python API.

#### Why these two are Interesting Datasets:

Dataset A represents a "Binary Class" Classification problem, where-as Dataset B represents a "Multi Class" classification problem. Two different classification type datasets would help us to understand which Supervised algorithm better when.

Dataset A (Breast Cancer Diagnostics) helps for early detection of Breast Cancer in females with the help of a finite number of features and identify which are the most convincing features.

Dataset B (Vehicle Silhouettes) also poses a very interesting problem of identifying 3D objects using 2D pictures.

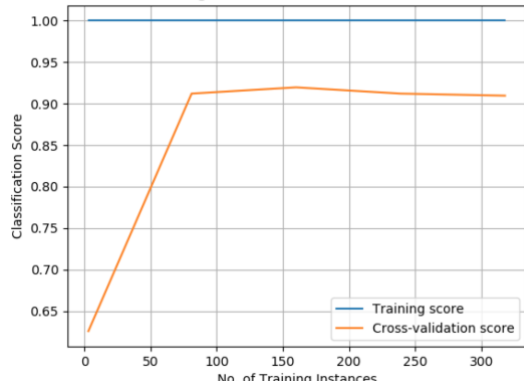
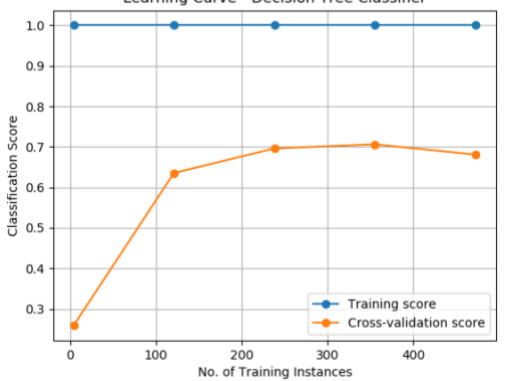
### Analysis Approach:

We shall be using 5 classifiers, i.e., Decision Tree, AdaBoost, Neural Network, SVM and kNN using scikit-learn Python APIs. Hyper-parameter tuning shall be done using Visualization through Learning curve and ModelComplexity/Validation curve using 5-fold cross-validation.

## Decision Tree:

Un-tuned Classifier (our starting baseline):

We shall start by doing a Learning Curve on the two datasets with default parameter values and NO tuning and identify what needs to be done to improve the classification score of our Decision Tree classifier .

Dataset A - Breast Cancer (Diagnostic)	Dataset B – Vehicle Silhouettes
 <p>Accuracy Score of this decision tree model is 94.15%</p> <p><b>Observation:</b> Our model overfits with a training score of 1, while the validation score reached is around 0.91 with full training data. The classification score reaches maximum around 160 Training instance and then it goes down little bit and eventually becomes flat. This shows that the model is <b>overfitting</b> and adding more training instances won't help as the validation curve flattens out.</p>	 <p>Accuracy Score of this decision tree model is 69.29%</p> <p><b>Observation:</b> Our model overfits with a max training score of 1, while the validation score is around 0.68 on entire training data. This shows that there is High Variance. The classification score starts doing down when the training instances reaches max. This shows that the model is <b>overfitting</b> (high variance) and just adding additional training instances will NOT help.</p>

Since adding more training instances will not work and also we are working with limited data. So, we need to use some generalization to improve the accuracy of our base model by doing hyper-parameter tuning.

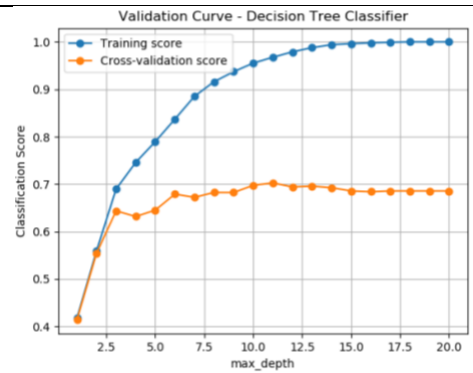
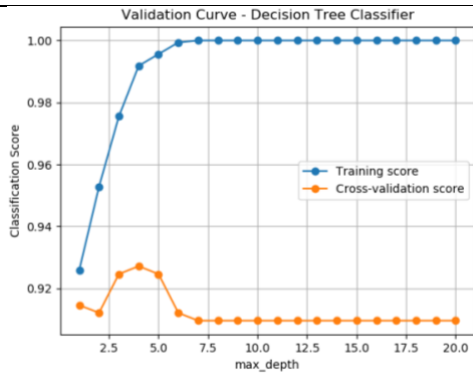
Hyper-parameter tuning process:

**Approach:** We will use Model Complexity curve of Param\_1 to identify the best performing hyper-parameter (it can be one or multiple values). Then use these possible values of Param\_1 as input to generate multiple Complexity Curve for Param\_2 and identify what would be the set of best performing values for Param\_2. We shall go up to param\_4. Using this iterative process, we will identify the best possible set of value for the 4 hyper-parameters.

[max\\_depth](#) → [min\\_samples\\_leaf](#) → [min\\_samples\\_split](#) → [max\\_features](#)

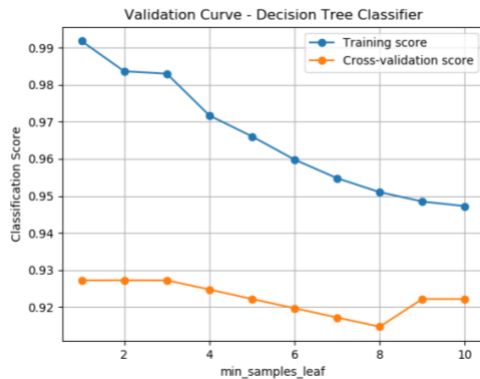
Below are the final set of Validation/Model Complexity curves.

Dataset A	Dataset B
<p><b>Hyper-Param: max_depth</b></p> <p>1) max_depth = 4 shows highest cross-validation score before the model starts to overfit. We will use this value for tuning the next parameter.</p>	<p><b>Hyper-Param: max_depth</b></p> <p>1) As the tree depth increases the tree size increases (less pruned) and the model moves towards overfitting. For max_depth = 11 and 12, the cross-validation score is pretty high with the model not too overfit.</p> <p>2) Doing few iterations shows that max_depth = 12 shows best performance.</p>



### Hyper-Param: min\_samples\_leaf

INPUT: max\_depth = 4



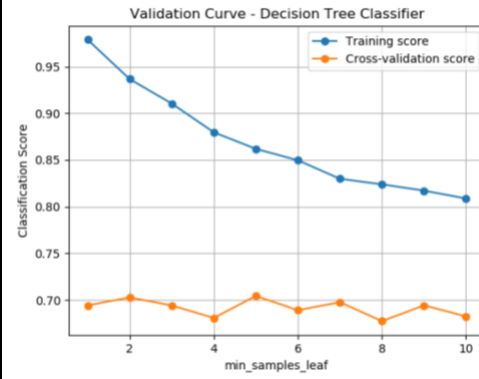
1) At min\_samples\_leaf = 2 and 3, the model starts getting more generalized on training data with highest cross-validation score.

So, we will use both min\_samples\_leaf = 2 and 3 as input and see which gives a better model complexity for the next parameter tuning.

2) min\_samples\_leaf = 3 gives the best accuracy score.

### Hyper-Param: min\_samples\_leaf

INPUT: max\_depth = 12



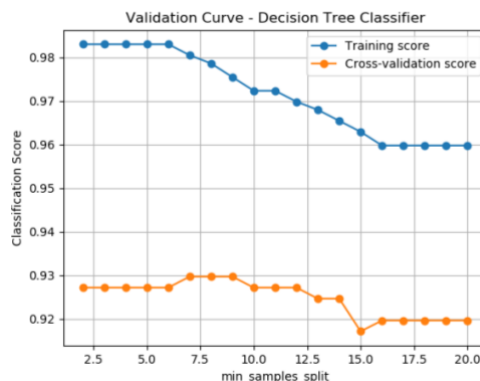
1) As the min. leaf increases the model becomes more under-fitted.

At min\_samples\_leaf = 2, 5 shows highest cross-validation score with a decent reduction of overfitting (decrease in variance or gap).

2) Doing few iterations shows that min\_samples\_leaf = 2 shows best performance.

### Hyper-Param: min\_samples\_split

INPUT: max\_depth = 4, min\_samples\_leaf = 3

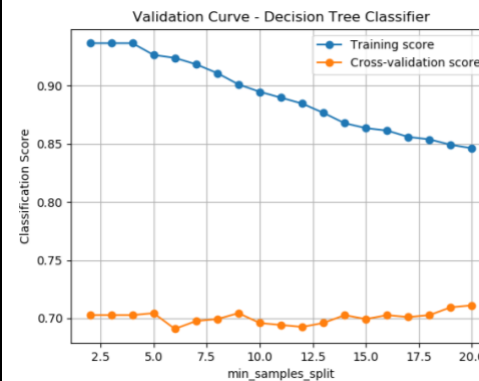


1) Given the above-mentioned inputs, the cross-validation score is best with min\_samples\_split = 7,8 and 9. The only thing changing is the generalization of the Training curve, gradually getting less overfitted.

2) min\_samples\_split = 8 gives the best accuracy score.

### Hyper-Param: min\_samples\_split

INPUT: max\_depth = 12, min\_samples\_leaf = 2



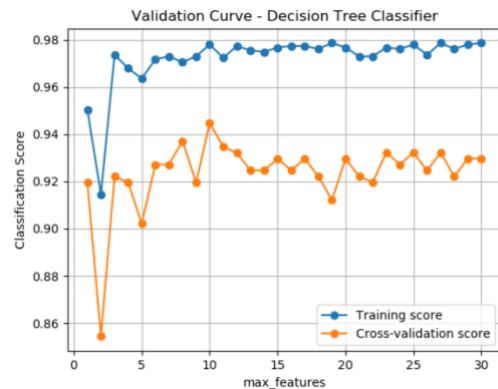
1) As the split size increases, the model becomes less overfit and the cross-validation score shows an uptrend for min\_samples\_split = 18, 19 and 20. This means the model is becoming for generalized at these points.

2) Doing few iterations shows that min\_samples\_split = 18 shows best performance.

### Hyper-Param: max\_features

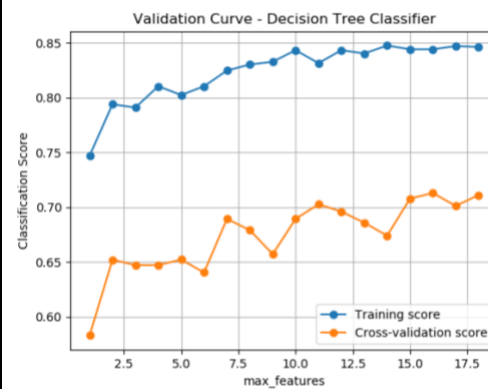
### Hyper-Param: max\_features

INPUT: max\_depth = 4, min\_samples\_leaf = 3,  
min\_samples\_split = 8



- 1) For max\_features= 10 we get highest cross-validation score but the corresponding training curve also tend to shoot up towards more overfitting.
- 2) For max\_features= 11, shows a good combination of less training overfitting and good cross-validation score.
- 3) So, running max\_features= 8, 10, 11 we get the best score and learning curve for max\_features=11.

INPUT: max\_depth = 12, min\_samples\_leaf = 2,  
min\_samples\_split = 18

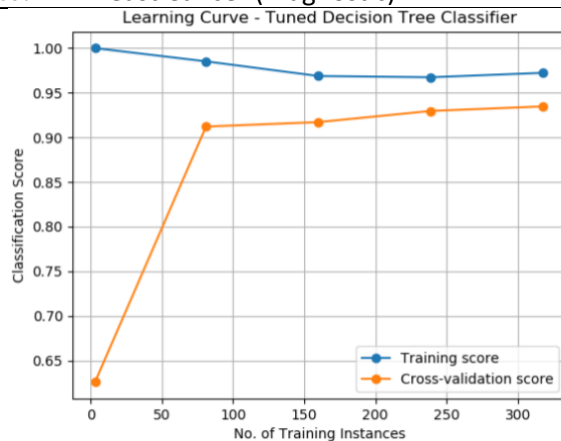


At max\_features=18, the cross-validation score reaches max. The Training score also increases but it's not overfitting as it's around 84%. Also, the gap (variance) is minimum.

For the purpose of experimentation max\_features value of 16 and 18 were used where max\_features=18 gives best accuracy score.

### Learning Curve with tuned hyper-parameters:

Dataset A - Breast Cancer (Diagnostic)

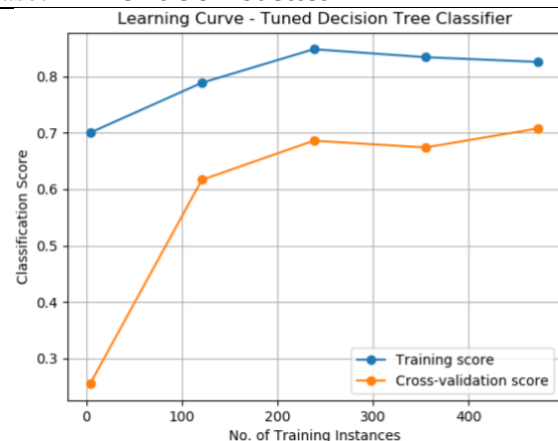


Accuracy Score (Test dataset) of this decision tree model is 97.08%

#### **Observation:**

- 1) It's very interesting to see that after the hyper-parameter tuning the Learning Curve shows an upward trend with full training sample. This means that adding more training data now would lead to a significantly better model. This observation is totally in contrast with the Learning Curve of the un-tuned model.
- 2) The gap between the Training and cross-validation curve has gone down with max. training instance, i.e, Variance has reduced.
- 3) The Optimal Accuracy score on Test dataset increased to 97.08% with max\_depth = 4, min\_samples\_leaf = 3, min\_samples\_split = 8 and max\_features = 11

Dataset B – Vehicle Silhouettes



Accuracy Score (Test dataset) of this decision tree model is 75.20%

#### **Observation:**

- 1) After tuning the hyper-parameters, the Learning Curve shows that cross-validation curve is moving up for maximum number of training instance and both the training and cross-validation curve are converging (reducing variance). This shows that adding more training data now would lead to further lowering of variance and higher generalization.
- 2) Hyper-parameter tuning and larger training data would collectively make a better predictive model.
- 3) The Optimal Accuracy score on Test dataset increased to 75.20% with max\_depth = 12, min\_samples\_leaf = 2, min\_samples\_split = 18 and max\_features = 18.

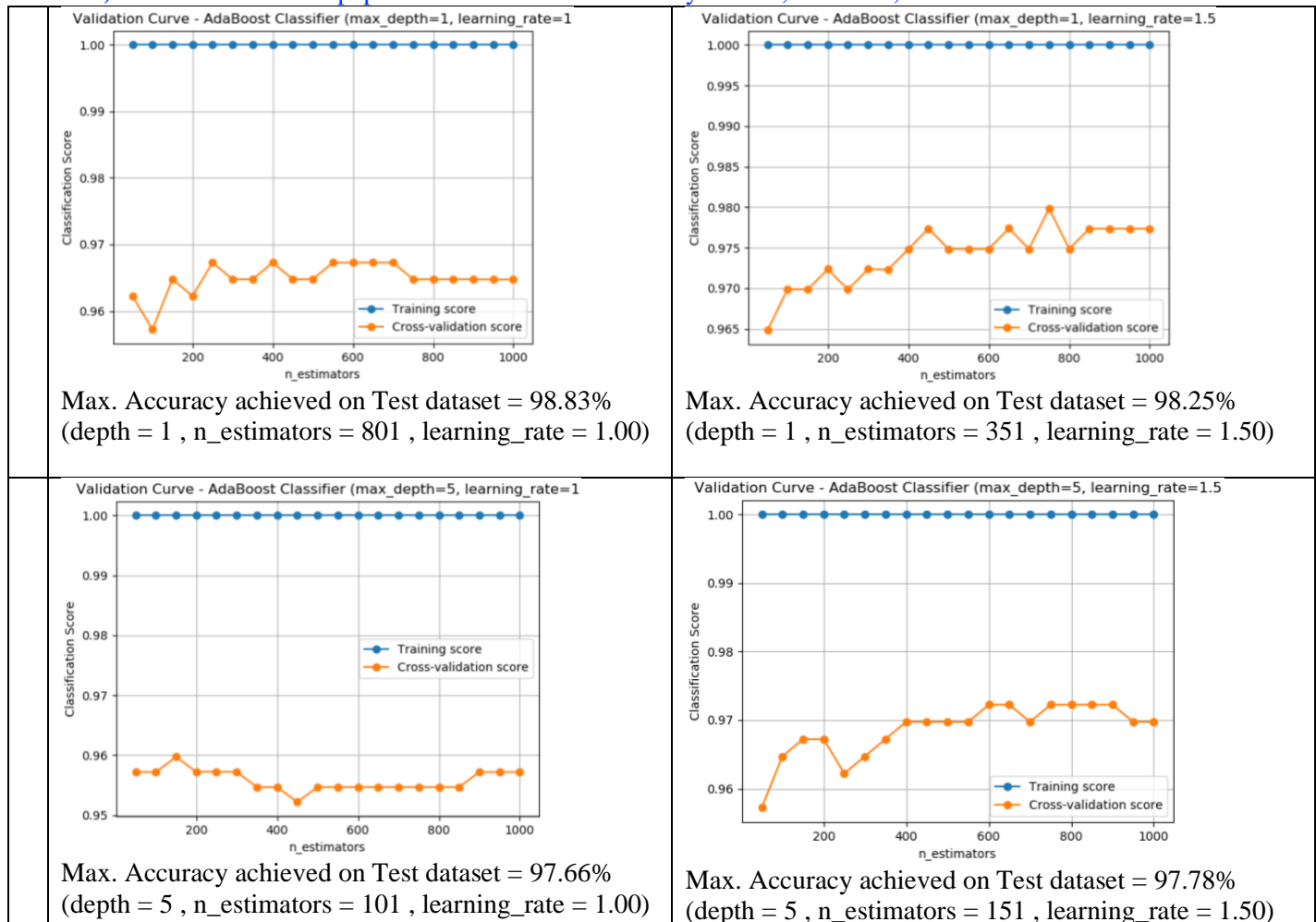
## **Boosting (AdaBoost):**

The idea of AdaBoost is to combine multiple weak predictors and to build strong predictor. For this the underlying base estimator is a Decision Tree with height 1 (i.e., a stump). Here we will see how it works on Binary and Multi-Class problems.

### **Dataset A - Breast Cancer (Represents Binary-class problem)**

Since the dataset being a Binary Class problem, we can see that the AdaBoost reached a 98% classification score (cross-validation curve) using max\_depth=1 (DT Stump) with a learning rate of 1.5 (this is more than what is achieved by Decision Tree Classifier alone). Increasing the decision tree height of the underlying height didn't help to increase the classification score.

This shows that AdaBoost with several weak classifier can be extremely successful in producing accurate classifier for binary classification problems but it may not be very effective in case of data involving multi class problem (class label > 2) as mentioned in the paper "Multi-class AdaBoost" by Ji Zhu, Hui Zou, Saharon Rosset and Trevor Hastie.



**Optimal Hyper-parameter values: depth = 1, n\_estimators = 801, learning\_rate = 1.00**

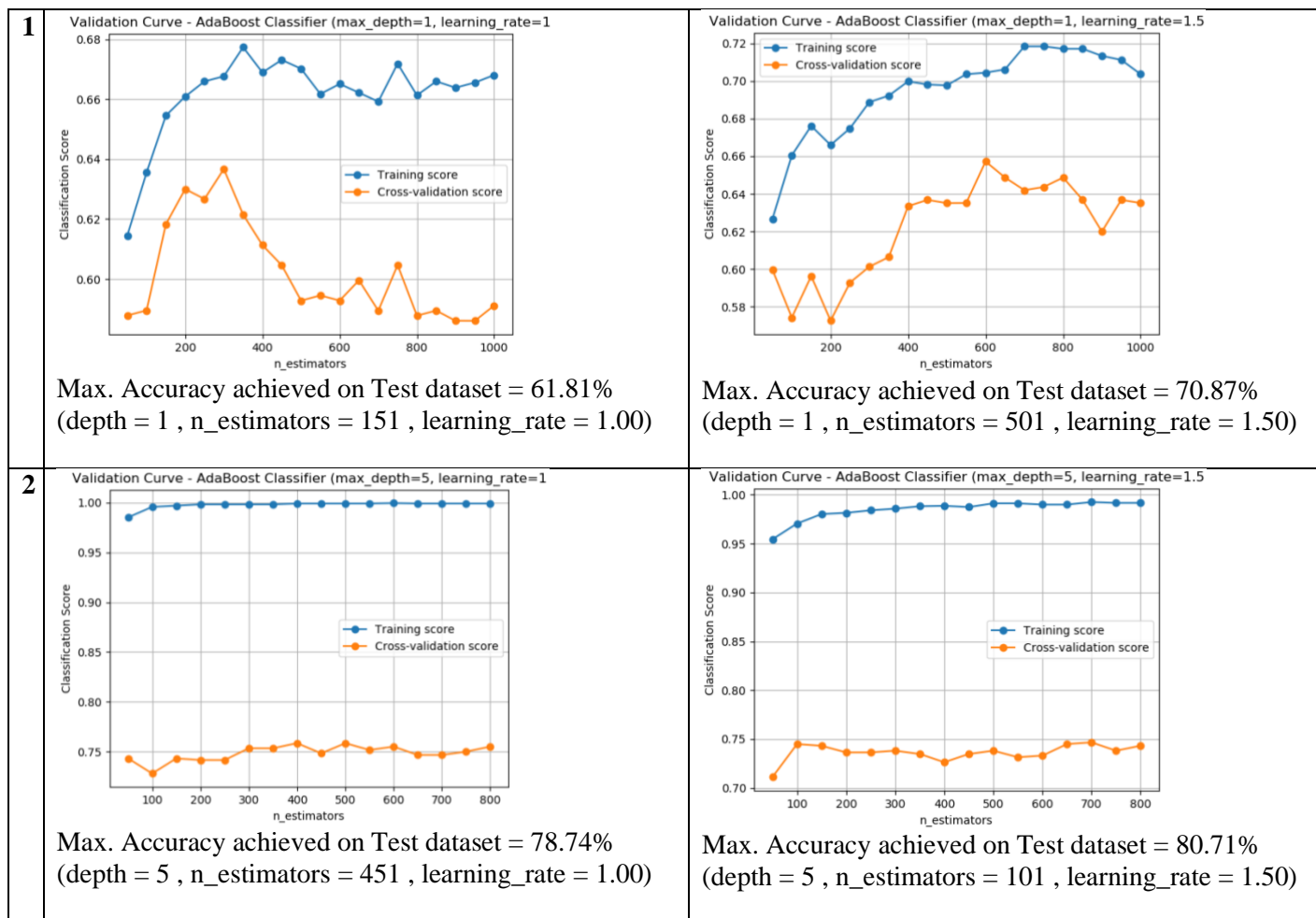
### **Dataset B - Vehicle Silhouettes (Represents Multi-class problem)**

One of the key factors is the max\_depth of the underlying decision tree classifiers.

From below Validation Curve, we can see that for a given learning\_rate, the overall cross-validation curve maintains higher Classification score and also the curve becomes more stable distribution (less fluctuations) as we increase the value of max\_depth. Also, the maximum accuracy score (test dataset) achieved for a given learning\_rate increases with the increase of max\_depth.

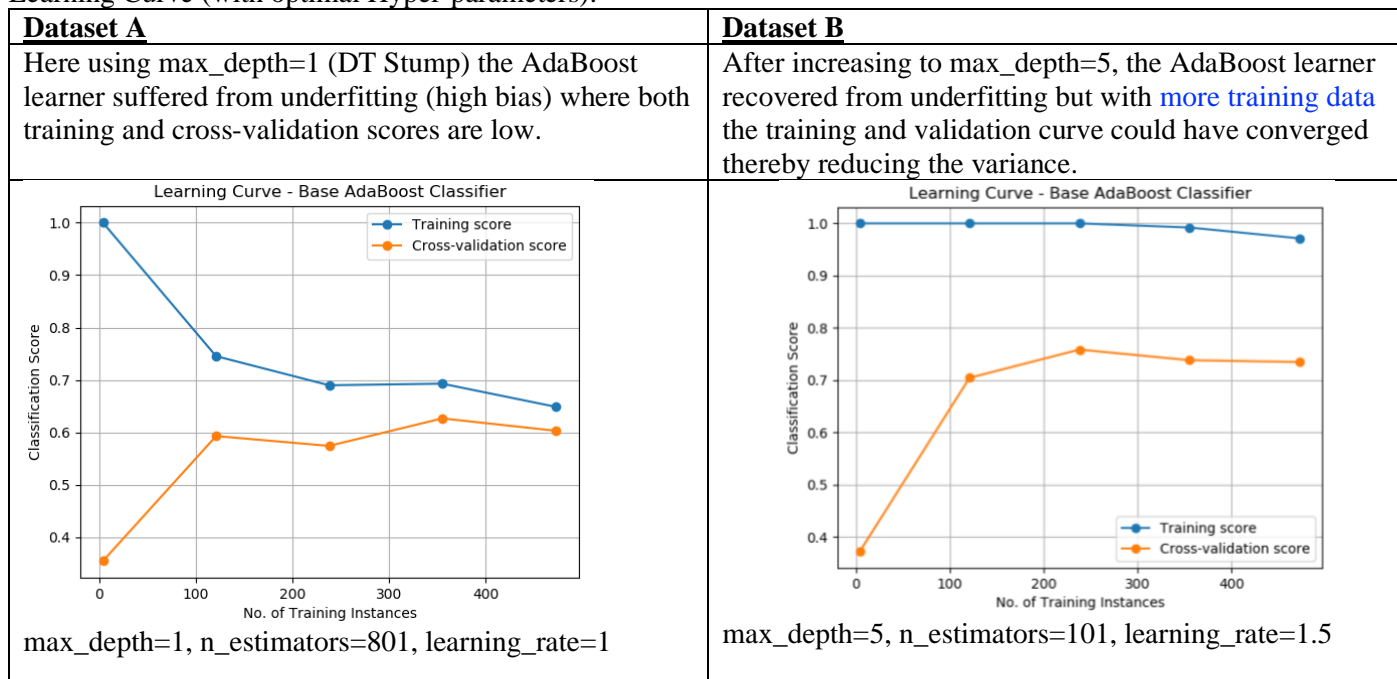
Graph Set 1 – most fluctuating and cross-validation curve lying below 66%

Graph Set 2 – most stable and cross-validation curve lying mostly above 75% and



**Optimal Hyper-parameter values: depth = 5 , n\_estimators = 101 , learning\_rate = 1.50**

Learning Curve (with optimal Hyper-parameters):



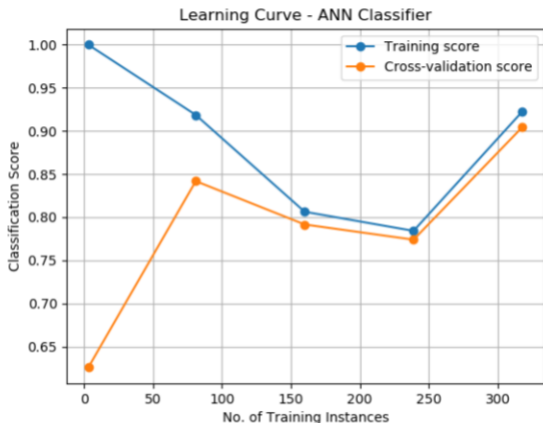
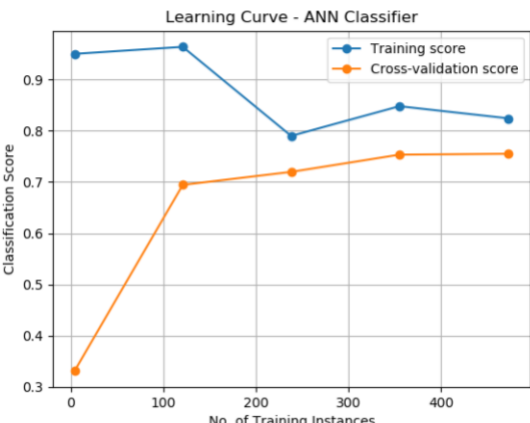
- Though AdaBoost works on multiple weak learners to build a collectively strong learner, but adding a certain amount of strength (or complexity) to the underlying learner increases the overall classification ability and accuracy score.

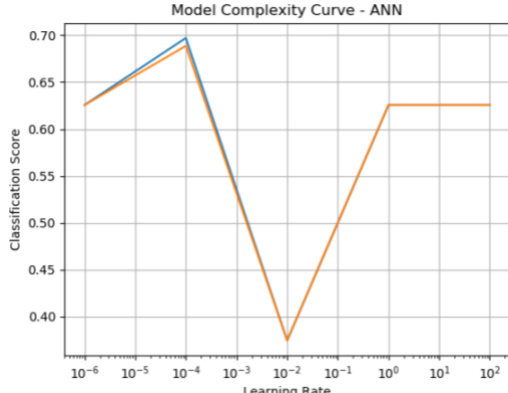
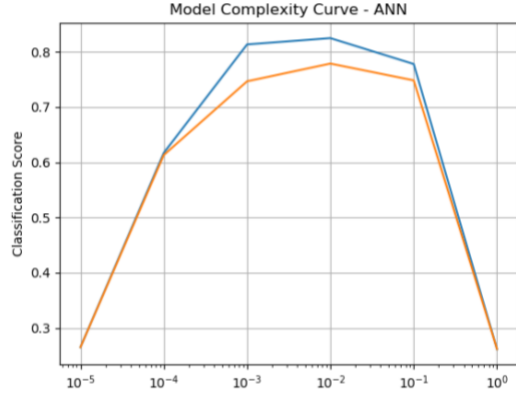
- Graph set 1 (above) shows underfitting as both training and cross-validation scores are very low. By increasing the `max_height` we added some more complexity to remove underfitting (see Graph set 2).
- Being a Multi-class problem, AdaBoost with default configuration couldn't perform well, but it improved after adding a bit of complexity by increasing `max_height` of the underlying Decision Tree classifier.

### Artificial Neural Network:

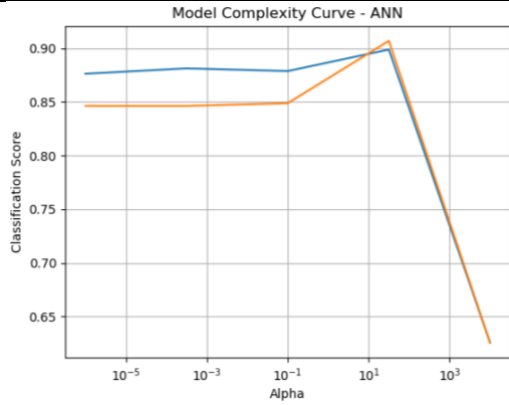
The two most crucial hyper-parameters that control the performance of an ANN classifier are its Learning Rate and Alpha. For the sake of easier analysis, we are considering a small Neural Network with 2 hidden layers of 4 and 3 nodes respectively and 4000 iterations.

We shall start with a Learning Curve with a basic ANN classifier (i.e., default hyper-parameter values).

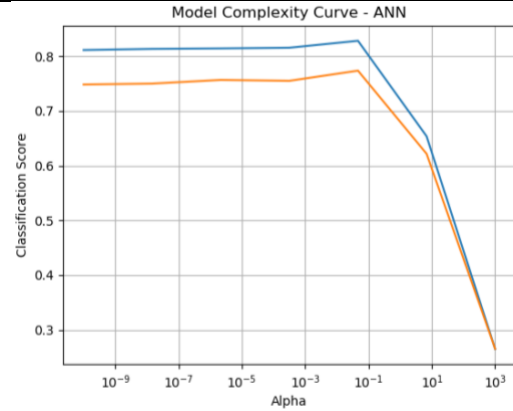
Dataset A	Dataset B
<p>Learning curve with default parameter values and <code>max_iters=2000</code></p>  <p>Accuracy score on Test dataset without any tuning is 92.98%</p> <p>The Training and cross-validation curve converges well for maximum training data. So, the classifier shows very low variance.</p> <p>Both the curves show high classification score with max training data.</p> <p>We shall try to improve the current Accuracy Score by tuning some hyper-parameters.</p>	<p>Learning Curve with default hyper-parameter values and <code>max_iters=4000</code></p>  <p>Accuracy score on Test dataset without any tuning is 80.31%.</p> <p>The model is initially over-fitting (high variance) with small number of training instances but gained better generalization with increase in training instances.</p> <p>The Cross-validation curve shows maximum performance at 350 training instance mark and then it goes flat. This means with this untuned learner we wouldn't gain any improvement with additional training data.</p>

Model Complexity Curve – Dataset A	Model Complexity – Dataset B
 <p>Possible values to tune Learning Rate are 0.0001 to 0.001 and 0.1 to 10</p>	 <p>Possible values to tune Learning Rate are 0.001 to 0.01</p>





Possible values to tune Alpha are 0.1 to 100

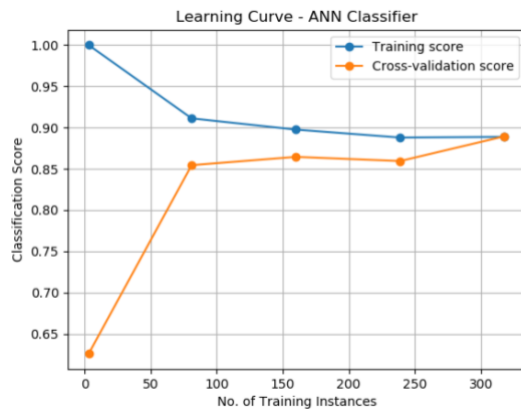


Possible values to tune Alpha are 0.001, 0.01, 0.1

After doing some hyper-parameter tuning based on the range captured from the above model-complexity curve:

### Optimal Hyper-parameters:

Alpha=10 and Learning Rate=0.001

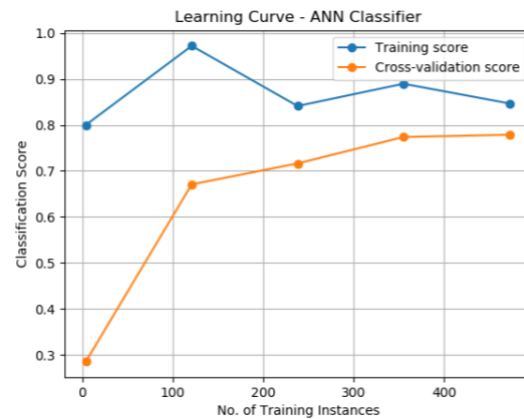


Accuracy of ANN after tuning is 97.66% which is 4.68% higher than the untuned model.

Also, the training and cross-validation curve converged at max training instance making it negligible Variance and both having high classification score making it low Bias.

### Optimal Hyper-parameters:

Alpha=0.01 and Learning Rate=0.00398



Accuracy of ANN after tuning is 84.65% which is 4.36% higher than the untuned model.

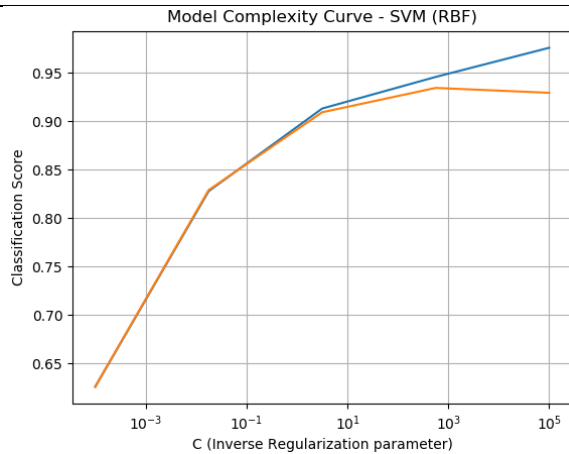
The Learning Curve shows that the cross-validation score is highest and still showing an upward trend with the max training instance. Both Training and validation curve shows a trend of convergence with additional training data.

## Support Vector Machine

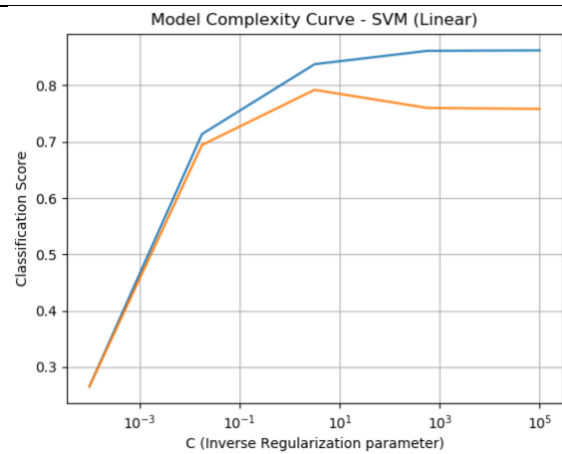
We will be applying and analysing SVM with two different kernel functions – RBF and Linear.

Dataset A	Dataset B
Accuracy of SVM (RBF) without tuning is 93.57% Accuracy of SVM (Linear) without tuning is 96.49%  We shall select the optimal hyper-parameter values based on visualization below.	Accuracy of SVM (RBF) without tuning is 79.53% Accuracy of SVM (Linear) without tuning is 81.50%  We shall select the optimal hyper-parameter values based on visualization below.

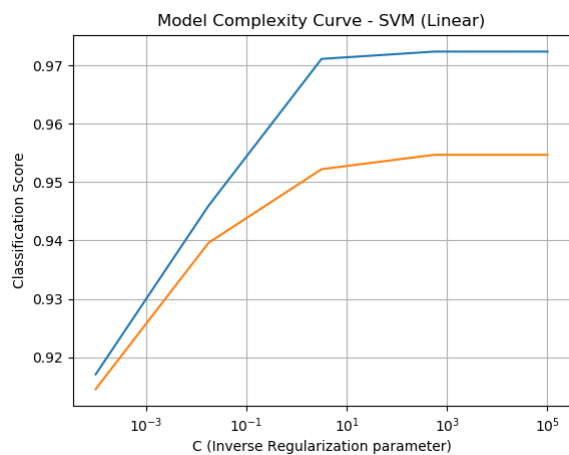




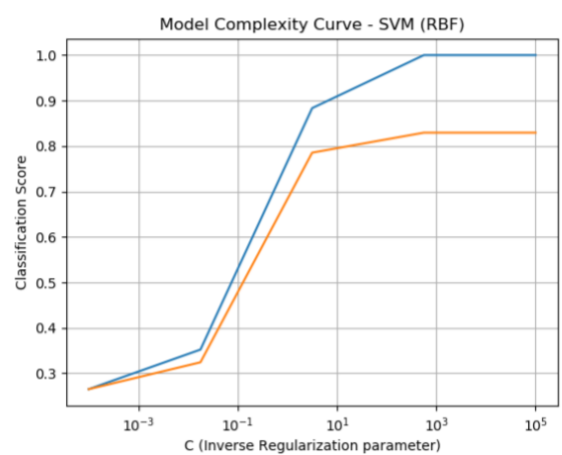
Based on the Model Complexity curve (RBF) at C=800 to 1000 the training and validation score both are high while Variance being minimum.



Based on the Model Complexity curve (RBF) at C=7 the training and validation score both are high while Variance being very low.

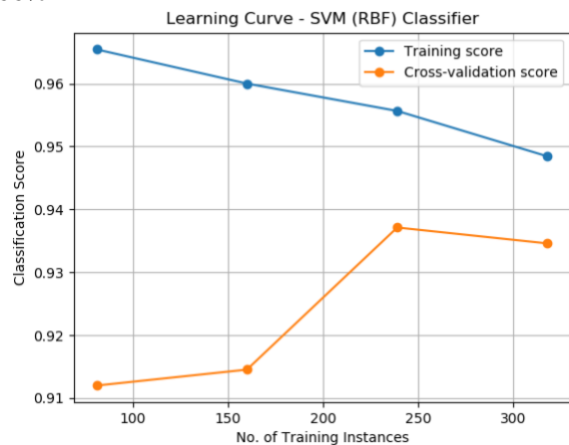


Based on the Model Complexity curve (Linear) at C=2 to 8 the training and validation score is above 95% (both are high) while Variance being nominal.



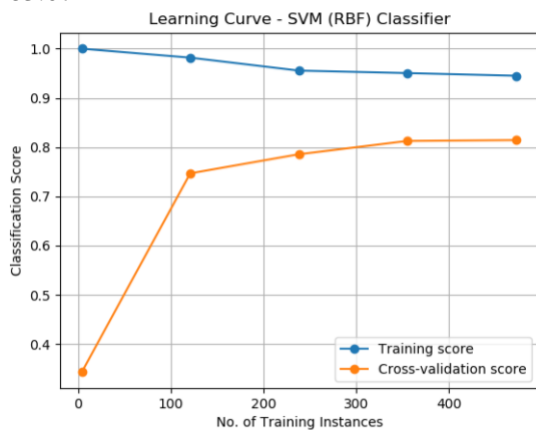
Based on the Model Complexity curve (RBF) at C=10 the training and validation curve shows low variance and Validation score is around higher range.

Accuracy (TestData) of SVM(RBF) after tuning is 97.66%

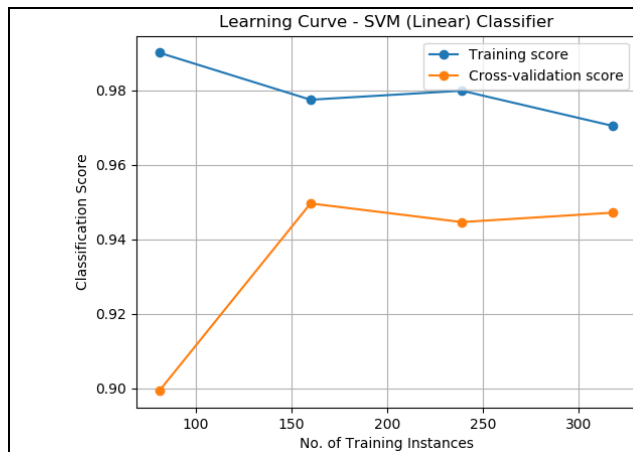


Accuracy (TestData) of SVM (Linear) after tuning is 97.66%

Accuracy (TestData) of SVM(RBF) after tuning is 83.07

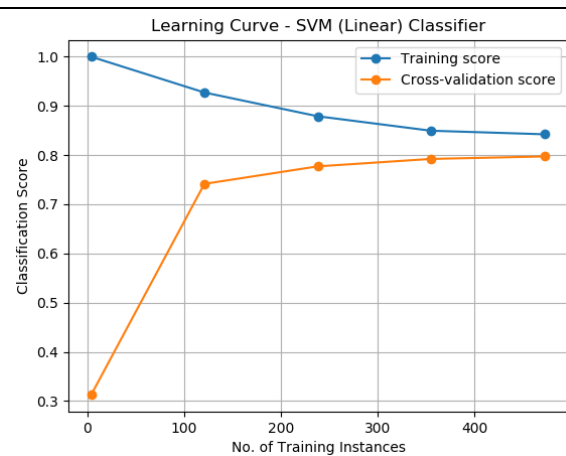


Accuracy (TestData) of SVM (Linear) after tuning is 83.07%



According to the learning curve above we can see that

- For linear SVM cross-validation curve reaches optimal score with a smaller number of training instances.
- Linear SVM reaches a higher cross-validation score (i.e., performance).



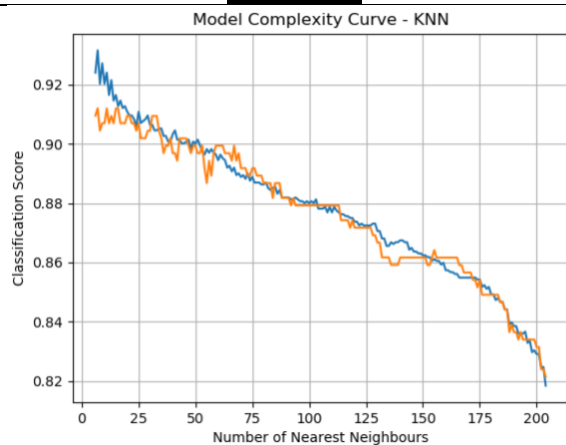
According to the learning curve above we can see that

- performance with maximum training dataset is almost same for both the kernels.
- Linear SVM shows very less Variance compared to rbf
- rbf kernel shows some overfitting.

Both kernels show same Test Score with hyper-parameter tuning since they usually gives comparable results when the number of train instances is greater than 10 times the number of features.

## K-Nearest Neighbour

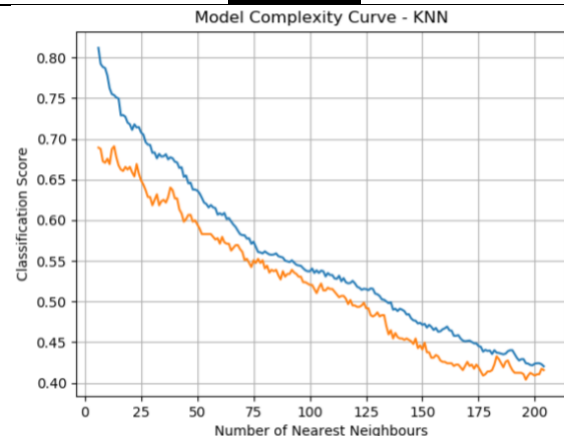
### Dataset A



At the beginning with very small value of K, the gap between the training and validation curve is max showing higher variance. At k=9, the gap is very less showing very low variance and the validation score pretty high meaning low Bias. As 'k' increases both training and validation score goes down meaning high Bias.

So, k=9 is the optimal value.

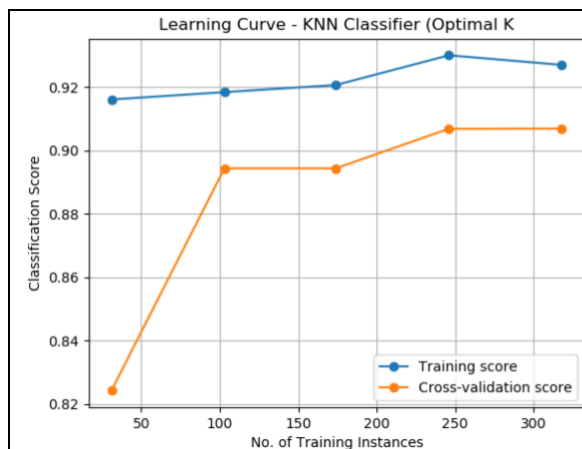
### Dataset B



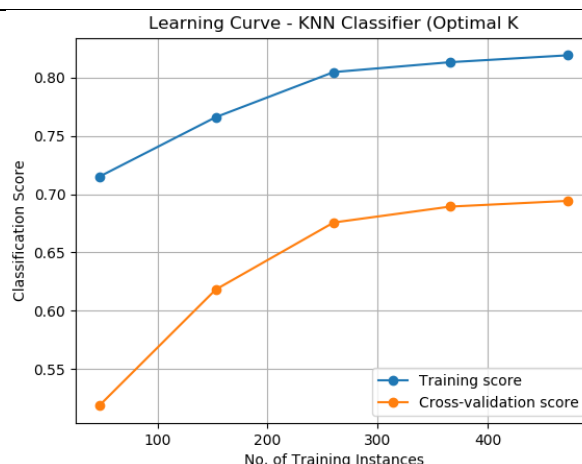
For low value of K (K=1), the gap between training and cross-validation is maximum (high variance - overfitting). After a certain point, both the scores start dropping because of high bias (underfitting).

The optimal point (k=5) where the validation score spikes up and training score comes down reducing the variance.

Max. accuracy of KNN with tuning is 76.38%



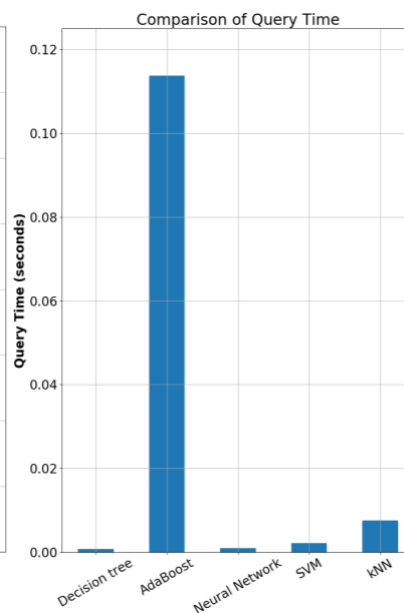
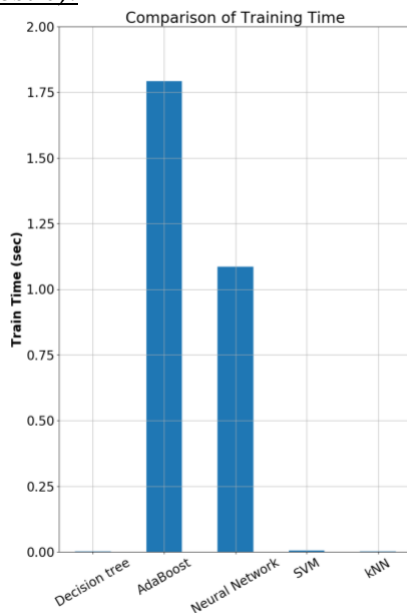
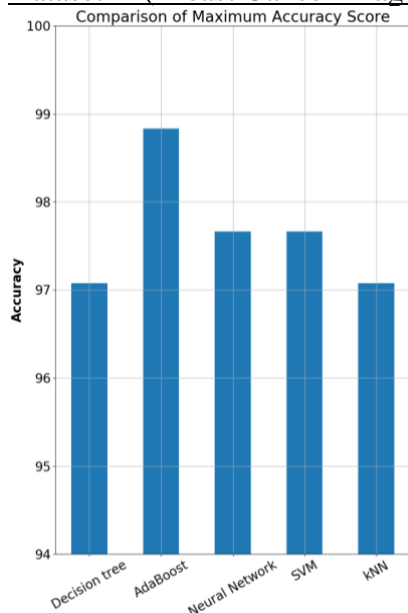
The cross-validation score is maximum with maximum training data. This shows that the model benefited with more training instances. The training curve may converge more towards the validation curve with additional training data leading to lower variance.



With less training instances shows higher variances which reduces as instances increase. The growth trend of the Learning curve shows that the model would have gained lower variance and higher cross-validation with additional training data.

## Comparison across classifiers:

### Dataset A (Breast Cancer Diagnostic):



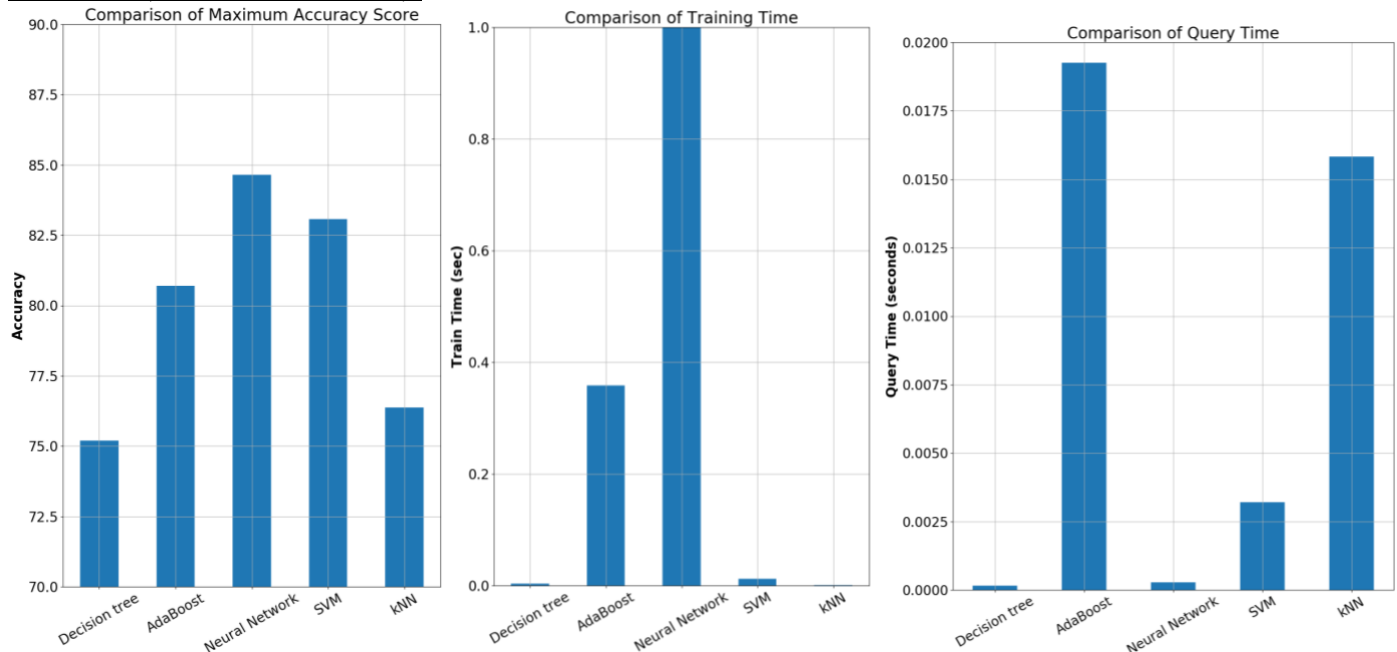
Accuracy score for Decision tree is minimum as it is the least expressive. KNN also shows least test accuracy as it is prone to overfitting. AdaBoost shows highest accuracy as it used a bunch of weak learners and possibly due to the dataset being a Binary Classification.

Training time of AdaBoost is very high as it used a huge number of base learners (i.e., 801) to come up with the high accuracy. Same with Neural Network with 2nd highest training time due to its complicated architecture and high iterations. KNN has the least which is very obvious.

Query time of AdaBoost is maximum may be because Predictions are made by calculating the weighted average of the weak classifiers and huge number of weak classifiers (i.e., 801) can increase the query time.

kNN has the second highest query time, since kNN always predicts the value on-the-fly based on distances from its neighbours which adds to its training time.

### Dataset B (Vehicle Silhouettes):



### Accuracy score

Instead of AdaBoost, this time it's Neural Network which has the highest accuracy and then SVM. This may be due to the fact that the dataset being a Multi Class (4 class values) problem needs a more complex classifier to generalize the rules. Decision Tree and kNN being the least efficient as they lack the expressive power to represent the complexities of the dataset.

### Training time

This is very obvious and intuitive that kNN has the least training time. Neural Network being the best classifier for this dataset correctly represented the complexity of the multi-classification problem, which also contributed to the high training time.

Query time of AdaBoost has been consistently high, this time it may be due to the fact that the underlying DT classifiers were used with a max\_height of 5 (instead of a Stump). kNN always showing the 2<sup>nd</sup> highest in query time as it has to calculate the weighted average of the neighbouring points (k=5) on the fly.

### Citations (Theory, data and Code referenced from the below sources):

- 1) <https://web.stanford.edu/~hastie/Papers/samme.pdf>
- 2) [https://scikit-learn.org/stable/auto\\_examples/model\\_selection/plot\\_learning\\_curve.html#sphx-glr-auto-examples-model-selection-plot-learning-curve-py](https://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html#sphx-glr-auto-examples-model-selection-plot-learning-curve-py)
- 3) <https://www.dataquest.io/blog/learning-curves-machine-learning/>
- 4) Machine Learning with scikit-learn Quick Start Guide by Kevin Jolly
- 5) <https://python-graph-gallery.com/>
- 6) <https://scikit-learn.org/stable/index.html>
- 7) [https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+\(Diagnostic\)](https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic))
- 8) [https://archive.ics.uci.edu/ml/datasets/Statlog+\(Vehicle+Silhouettes\)](https://archive.ics.uci.edu/ml/datasets/Statlog+(Vehicle+Silhouettes))