CLASSIFICATION PROBLEM ONE: CENSUS INCOME

The first classification problem I chose to solve is the classification problem of predicting if **"a person makes over 50K a year**" using the 1994 census data. The dataset is from UCI Machine Learning Repository. It includes both continuous and categorical attributes which include age, workclass (Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked), fnlwgt, education (Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool), education-num (continuous), marital-status (Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse), occupation (Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces), etc. Details of all the attributes and the dataset can be obtained from UCI Machine Learning Repository. Details are provided in the README.

While doing the analysis I felt that this dataset is interesting classification problem as it clearly highlights the difference between the algorithms mentioned in the assignment. But before dwelling deep into the analysis, preprocessing and cleaning is necessary to turn this dataset into good shape for the prediction using algorithms.

CLEANING AND PREPROCESSING

Following steps are performed on the dataset sequentially to make it suitable for classification by different algorithms:

1. **Cleaning:** Cleaning all the columns (stripping with extra spaces), changing the datatype and replacing the missing value with NaN.
2. **Imputation:** Replacing missing values with median (for continuous features) and most common values (categorical features).
3. **Encoding:** Encoding categorical data by creating binary attribute corresponding to each category of categorical feature.
4. **Preprocessing:** Preprocessing variables by removing mean and scaling to unit variance.

SPLITTING THE TRAINING AND TESTING DATA

Divided the dataset randomly into training and testing data with 20% data as testing data and rest as training data. Testing dataset is kept aside and not used until the final model hyperparameters are tuned.

HYPERPARAMETER TUNING OF THE ALGORITHMS:

1. DECISION TREE:

For tuning any algorithm I have used three step iterative approach to find the optimal hyperparameters. The approach I have taken is generalized approach and with minor

modifications can be used for any algorithm. Main aim of this approach is to find a tradeoff between hyperparameters such that they have low **cross validation** error (a subset data from the training data to check the correctness of the hyperparameters), and free from any **bias** (when model is underfit) and **variance** (when model is overfit). The three step approach is as follows:

1. Use brute force GridSearch on hyperparameters to find the optimal hyperparameters.
2. Draw validation curve and check the correctness of the hyperparameters for bias, variance and cross validation error.
3. Draw learning curve to again check for the bias, variance and if more/less data/attributes is needed.

Iterate to first step

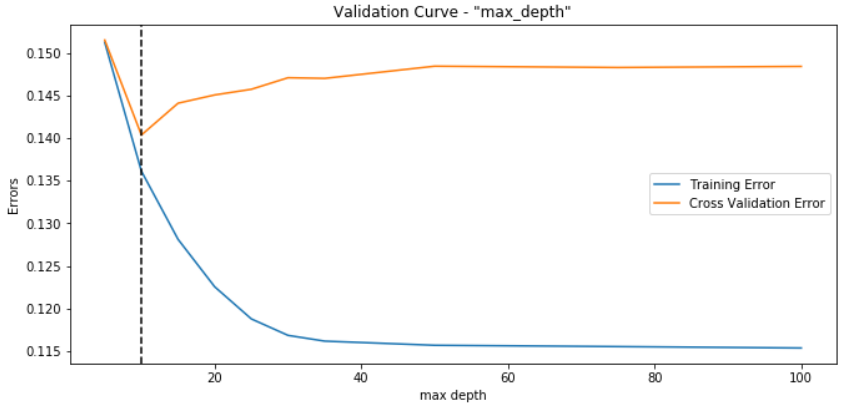
Grid Search, validation curves and learning curves are drawn on a small random subset of data and later tested on the full dataset.

For the decision tree I have chosen following three hyperparameters to tune and their optimal value obtained using GridSearchCV are:

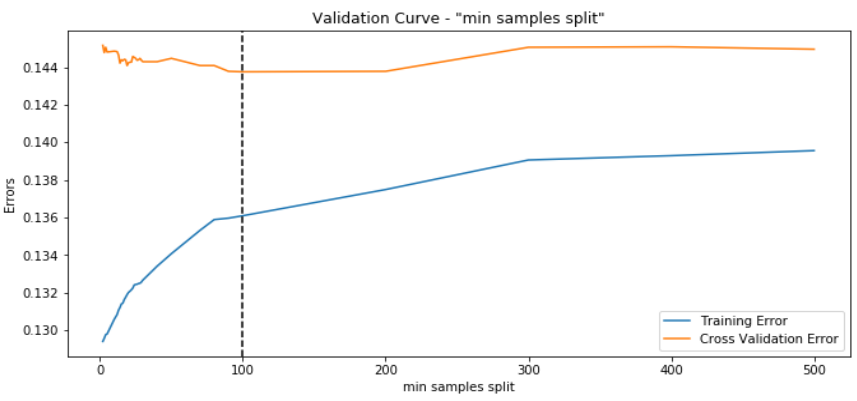
1. Splitting Criteria: entropy (measure of impurity among the data)
2. Max depth: 10 (maximum depth of the tree)
3. Minimum samples split: 100 (minimum number of samples in a node for splitting)

Cross Validation criteria: 3-fold cross validation (training dataset is divided into three equal folds and two different combinations are trained thrice and cross validation is done on the third fold. Later, average cross validation error is taken as the final cross validation error).

Validation curves:



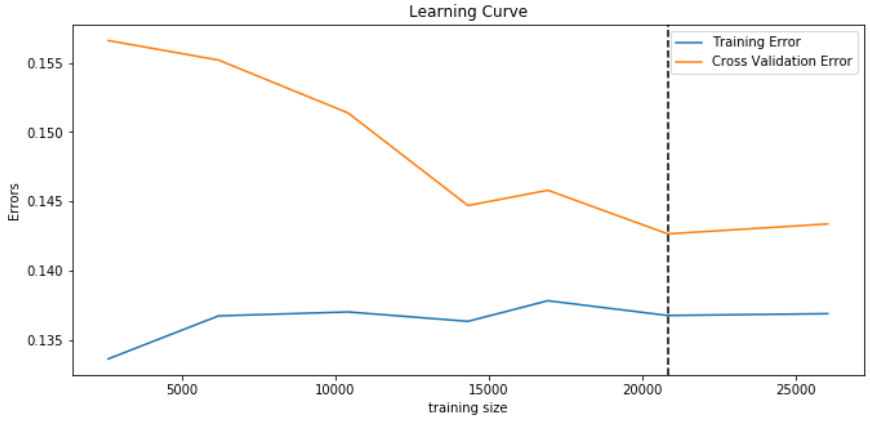
1. Validation curve for maximum depth: It shows cross validation error is minimum for maximum depth of 10.



2. Validation curve for minimum samples split: It shows cross validation error is minimum for minimum samples split of 100.

Learning Curve:

Learning curve plots training and testing accuracy as a function of no. of training samples. It differentiates between if the model has high bias or variance (can be corrected by increasing more samples or reducing the features). High variance is when there is huge gap between training and testing accuracy in the learning curve. High bias is when training error is high which approximately matches with testing error.



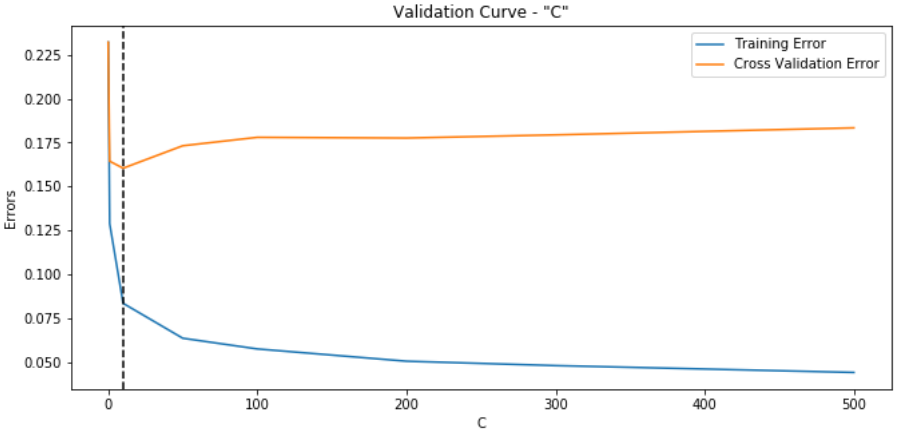
Accuracies of Decision tree algorithm obtained and their comparison with other algorithms are presented in the analysis part.

2. SUPPORT VECTOR MACHINES:

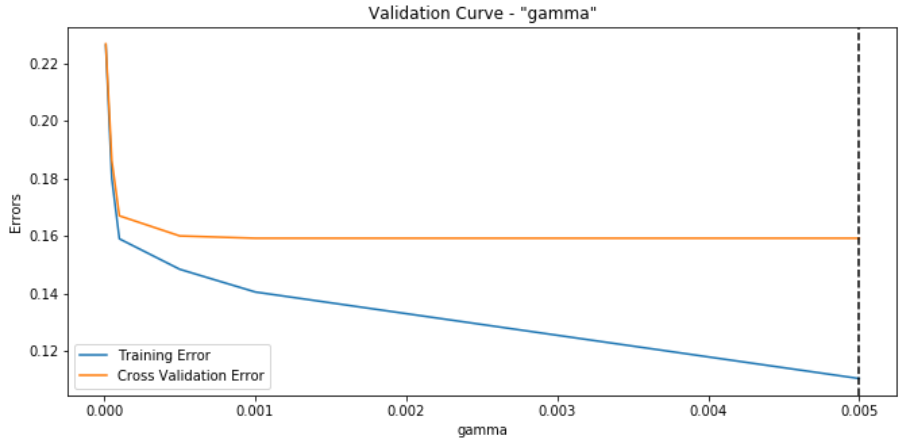
Optimal tuned hyperparameters obtained using GridSearchCV are:

1. C (inverse of regularization parameter): 10
2. Gamma (inverse of variance of the kernel used): 0.01
3. Kernel: rbf (gaussian)

Validation Curve:



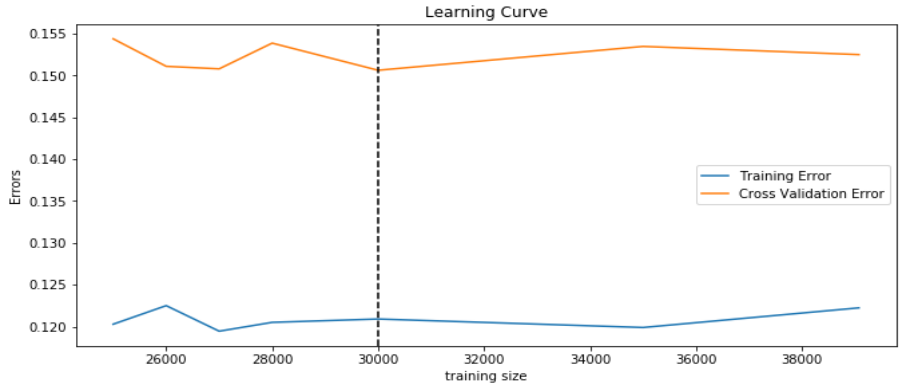
Validation curve for Parameter C: It shows minimum cross validation error is when C = 10.



Validation curve for parameter gamma: optimal gamma is 0.005.

Learning Curve:

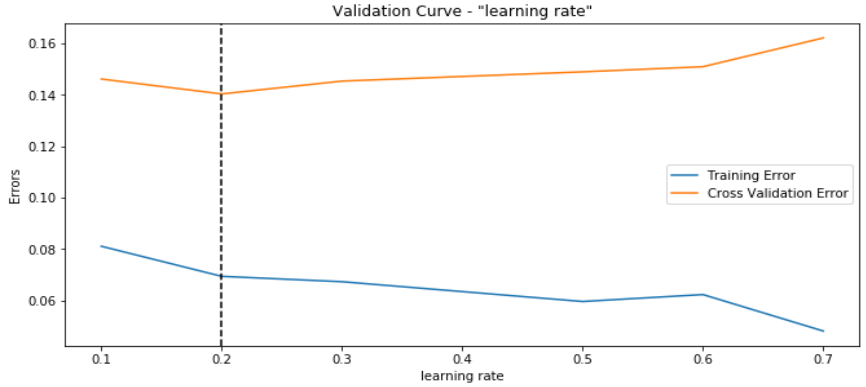
This learning curve is drawn only between training sizes from 25000 to 39000.



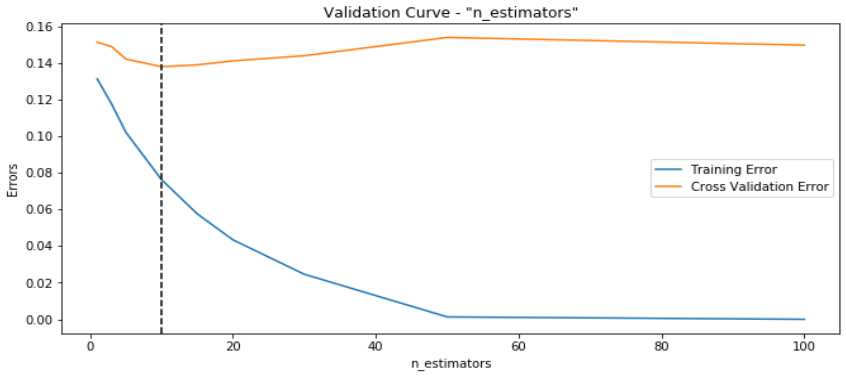
3. BOOSTING (ADABOOST CLASSIFIER):

Best parameters obtained using Grid Search CV: learning rate (0.1), number of estimators (10).

Validation curve for learning rate: optimal learning rate (0.2) which is different but close to grid search.



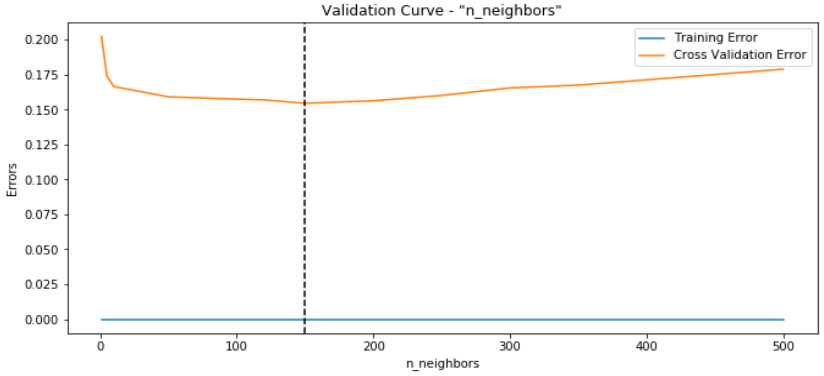
Validation curve for number of estimators: Optimal number of estimators obtained is 10.



4. K NEAREST NEIGHBOURS:

Optimal parameters obtained using Grid Search are: no. of neighbours (K): 120, weights: distance.

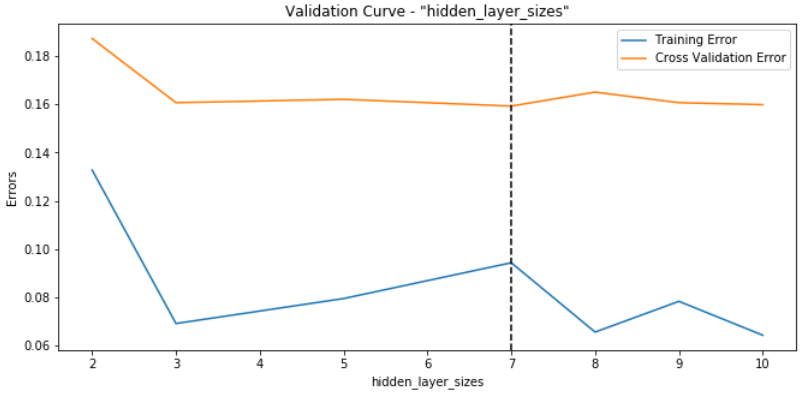
Validation curve: Slightly different optimal no of neighbours are obtained: 150



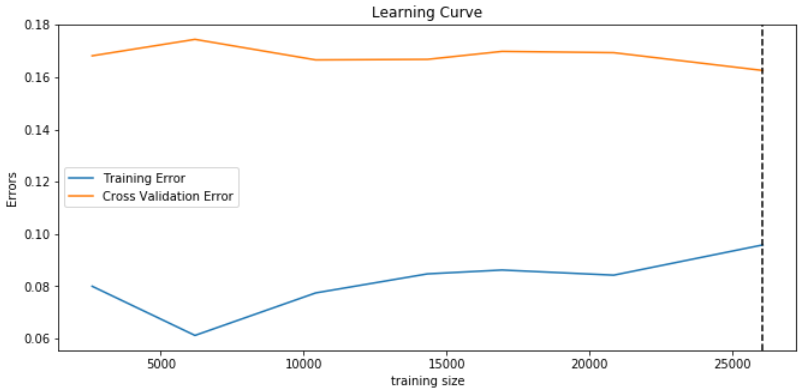
5. NEURAL NETWORK:

Optimal hyper parameters of neural networks:
Input layer size: 105 (number of attributes)
Hidden layer: 8
Output layer: 2
Number of iterations: 700

Validation curve for hidden layer size: Optimal hidden layer size for minimum cross validation error: 7



Learning Curve:



ANALYSIS OF ALGORITHMS:

The following table shows the testing accuracy of different algorithms when different samples of dataset were used for training. It can be clearly seen that some algorithms perform better even when trained with extremely less amount of data as compared to others. Decision tree achieves higher accuracy when trained with only 20,000 data instances as compared to SVM and AdaBoost which achieves similar accuracy when trained with full dataset.

	10,000	**20,000**	**25,000**	**27,000**	**30,000**	**Full**
Decision Tree	85.25	85.587	85.556	84.859	85.576	85.249
SVM	84.4098	84.2870	84.706	84.716	84.7271	85.1059
AdaBoost	85.47	85.648	86.078	86.242	85.8736	86.682
KNN	82.64	82.669	83.27	83.0177	82.73	83.10983
Neural Network	82.976	82.853	83.1819	82.3318	82.7925	82.925

Following table shows the training time (in seconds) of all the algorithms when trained with different sized samples of the dataset. It can be clearly seen that decision tree is a winner by a huge margin. Almost for all sizes decision tree is trained in fraction of seconds as compared to other algorithms. SVM is the slowest of all, while KNN takes less time than SVM.

	Training time (s)						**Testing Time (s)**
	10,000	**20,000**	**25,000**	**27,000**	**30,000**	**Full**	**For optimal size**
Decision Tree	0.0866	0.16407	0.2214	0.2519	0.2934	0.4174	0.00268
SVM	12.580	38.26	59.329	86.7285	97.9908	174.034	0.00366
AdaBoost	0.6584	1.6149	2.307	2.487	2.9406	4.419	0.0011

KNN	14.5851	28.934	35.558	37.935	41.9145	62.8517	0.00205
Neural network	3.347	7.435	11.993	10.555	14.915	11.973	0.00181

Following observations can be drawn from these too tables about the different algorithms:

1. Decision tree is the best algorithm to classify if a person makes over 50K a year or not when considered a trade-off between size of dataset used, testing accuracy and training time.
2. Decision tree takes only 20,000 instances to produce an accuracy which is only worse than AdaBoost probably because the dataset had more categorical features.
3. Decision tree takes the least time to train the classifier (in fraction of seconds) as compared to others whereas SVM is the slowest.
4. AdaBoost have the highest testing accuracy (86.682%) better than decision tree because it trains iteratively many weak learner (decision tree in this case).
5. Neural network and KNN have low accuracy because the data has more features which are categorical. Neural network and KNN requires more data instances as more the higher dimensions are due to curse of dimensionality.
6. SVM is slow and it takes full dataset instances to produce comparable accuracy.

CLASSIFICATION PROBLEM TWO: PREDICTING FOREST COVER TYPE

The second dataset for the classification problem is also taken from UCI Machine learning repository. The classification problem is predicting forest cover type from cartographic variables. The dataset includes four wilderness areas located in Roosevelt National Forest of northern Colorado. Details about the dataset are provided in the README.

The dataset was already cleaned and it didn't had any missing values so no preprocessing was needed. The major constraint was the dataset had more than half a million instances so it was difficult to run the experiments fast. So I randomly sampled around 5K instances and performed my experiments on that.

The dataset had 54 features and on doing experiments (plotting learning and validation curves) I decided to use Principal Component Analysis (PCA) method to check if there is correlation between the features. Principal Component Analysis method converts set of correlated features into orthogonal features that have highest possible variance. This is an orthogonal transformation that uses eigenvalue decomposition.

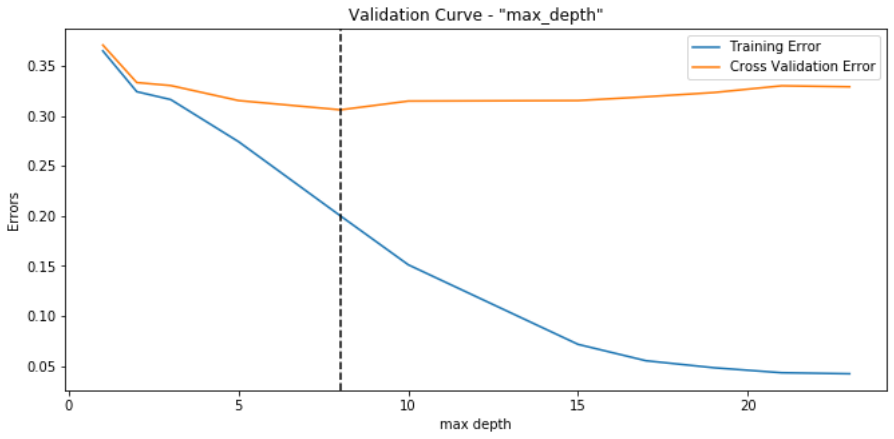
So, after using PCA and retaining 99% of the variance to reduce the dimensions of the training vector, the set of 54 feature vector were reduced to only 3 feature vector. Now, optimal hyperparameters were searched using GridSearchCV on both the original training set and reduced training set. The one which had better cross validation score is used for further analysis. The following table shows the optimal hyperparameters obtained using GridSearchCV for both the full and dimensionally reduced dataset for all the algorithms. The highlighted box shows best score for each algorithm.

Algorithm	Without PCA	With PCA
Decision Trees	Max depth: 15 Min sample split: 5 Criteria: gini	Max depth: 10 Min sample split: 80 Criteria: gini
	Best Score: 0.69225	**Best Score:** 0.61575
Support Vector Machines	C: 1 Gamma: 0.05 Kernel: rbf	C: 1 Gamma: 10 Kernel: rbf
	Best Score: 0.4915	**Best Score:** 0.65825
AdaBoost	Learning Rate: 1 No. of estimators: 400	Learning Rate: 0.7 No. of estimators: 200
	Best Score: 0.76025	**Best Score:** 0.6585
KNN	No. of neighbors: 9 Weights: distance	No. of neighbors: 30 Weights: distance
	Best Score: 0.69775	**Best Score:** 0.66625
Neural Networks	Hidden layer sizes: (54,5,2) Max iters: 500	Hidden layer sizes: (3,2,2) Max iters: 2000
	Best Score: 0.55775	**Best Score:** 0.55425

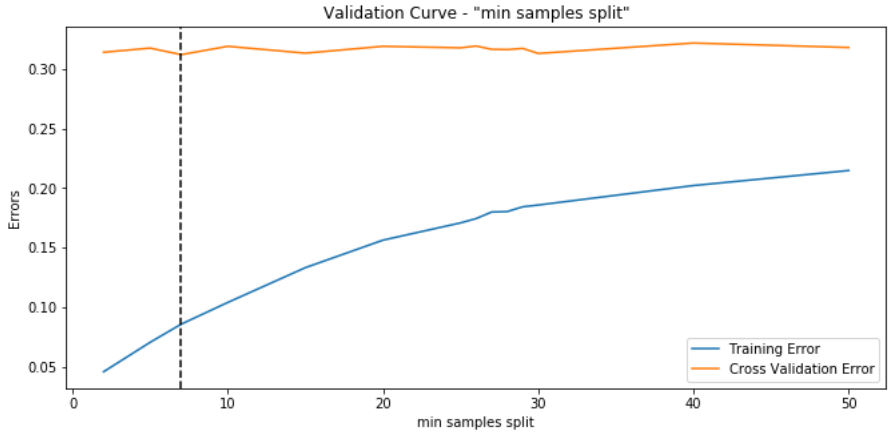
Validation and learning curves:

1. Decision Trees:

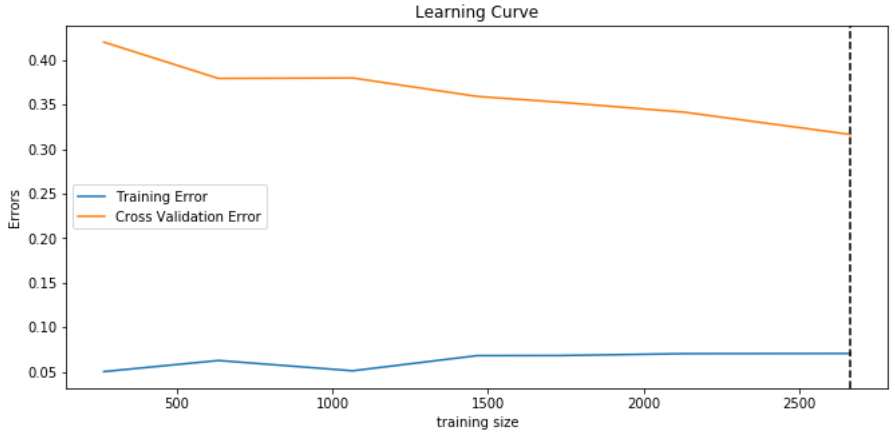
Maximum depth due to minimum cross validation is 8 which is different from what was obtained in GridSearch.



min_sample_split for minimum Cross validation error: 7

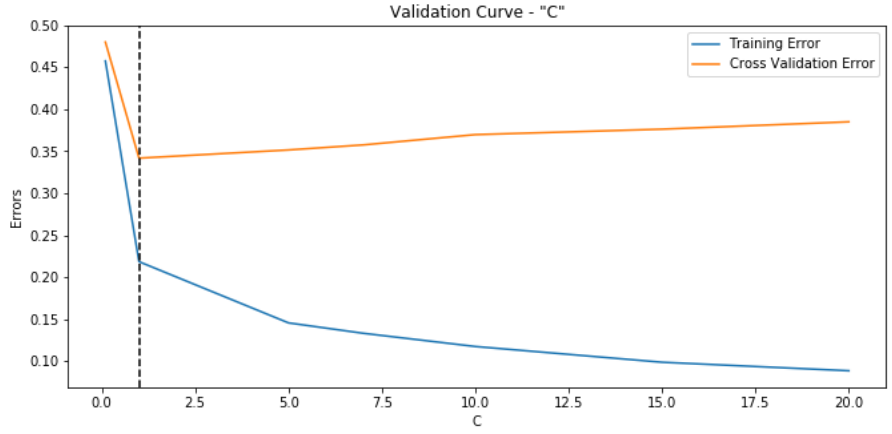


The learning curve is plotted below. It shows that it is still a high variance problem if we increase the data sample size then Cross validation accuracy might increase further.

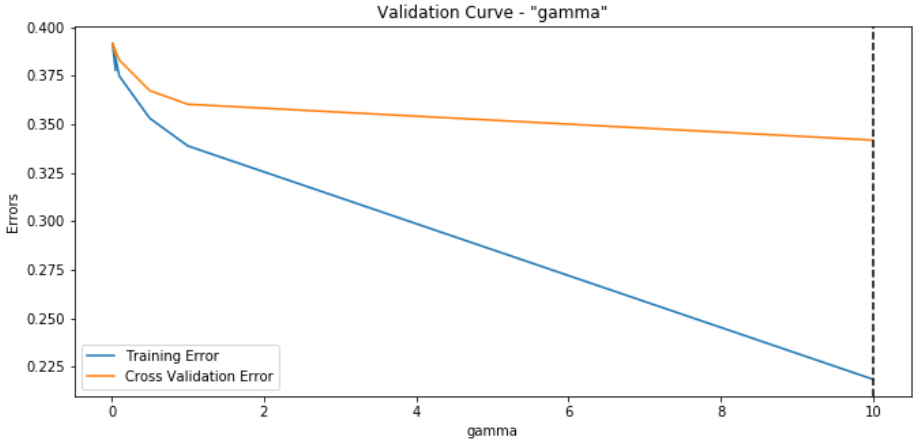


2. Support Vector Machines:

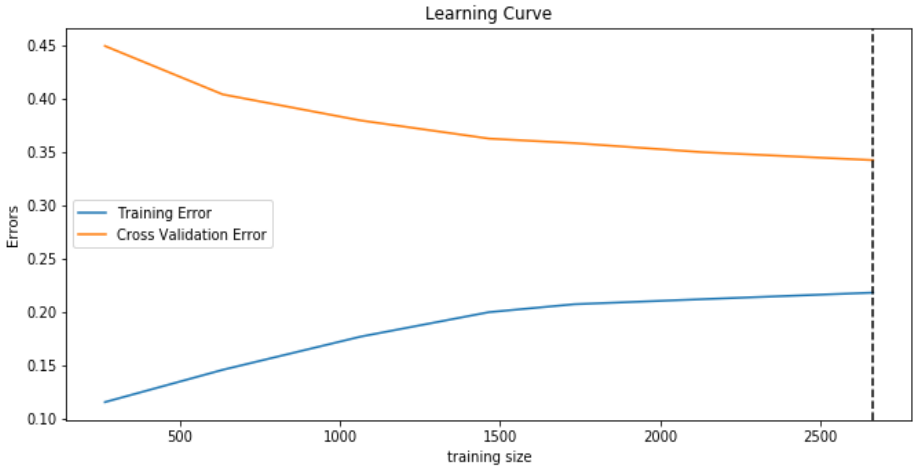
C for min cross validation error is 1 same as obtained using GridSearch with reduced dataset PCA. As can seen in the table only Support Vector Machines is producing better cross validation accuracy when it is fed with dimensionally reduced dataset. That is it is more prone to correlated features and thus overfit when kernel is applied on correlated features.



gamma for min cross validation error: 10

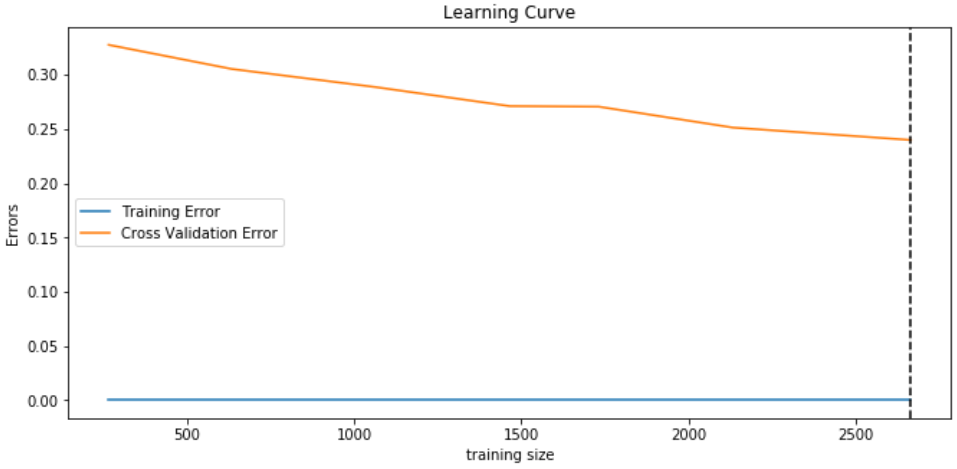


If we compare learning curve of decision tree and learning curve curve of SVM, it can be clearly seen the for support vector machines both training and testing accuracy are still closer than that for decision tree. Thus, support vector machine classifier is less overfit than decision tree classifier.



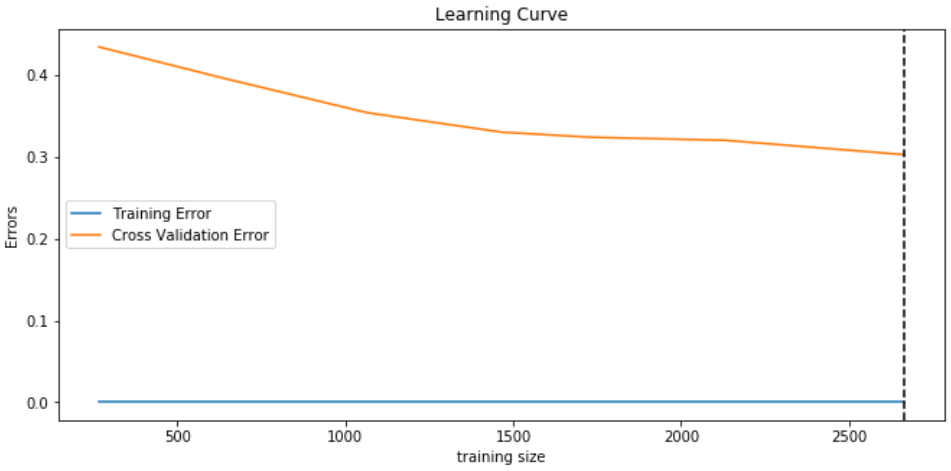
3. Boosting:

Learning curve for AdaBoost is shown below. If more data is added Cross validation error would further decrease.



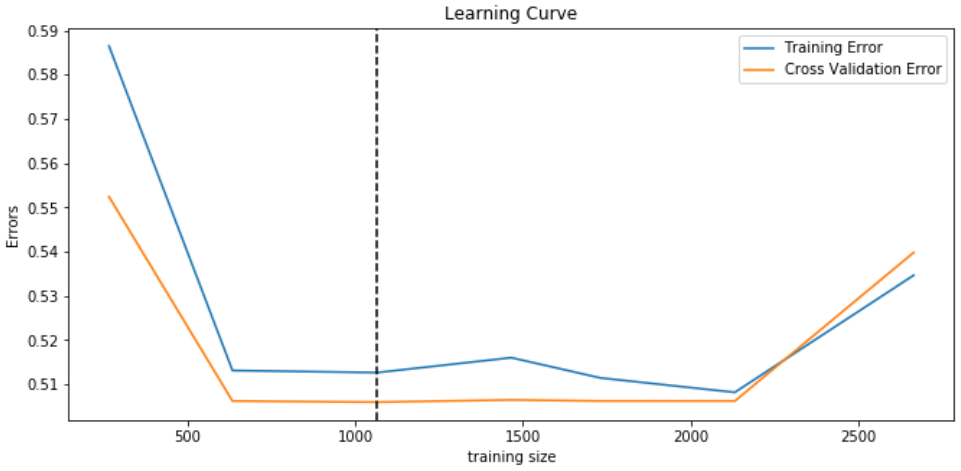
4. K Nearest Neighbors:

Figure below is the learning curve for K Nearest Neighbors algorithm. It can be seen that as sample size is increasing cross validation accuracy is decreasing. This is because KNN suffers from curse of dimensionality. The more the dimension the greater samples it needs to cover the full span so that cross validation error is small.



5. Neural Networks:

Learning curve for neural networks is shown below. It can be seen that after fixed number of samples cross validation error rate is saturated.



ANALYSIS OF ALGORITHMS:

Following table shows testing accuracy for different algorithms when trained with different sizes of dataset. Optimal accuracy for each algorithm is marked in a highlighted box.

	500	1000	2000	3000	Full
Decision Tree	61.3	61.4	65.6	66.1	65.9
SVM	57.8	61.6	60.2	61.3	64.2
AdaBoost	67	71.5	71.9	73.4	76.2
KNN	61.2	61.4	62.5	66.6	69.5
Neural Network	34.4	49.5	49.5	49.5	49.5

This table shows training and testing time for each algorithm when trained with different sizes of dataset. Conclusions from both these tables are mentioned below.

	Training time (s)					Testing Time (s)
	500	1000	2000	3000	Full	For optimal size
Decision Tree	0.00843	0.01182	0.02446	0.03363	0.04492	0.0032
SVM	0.05203	0.10719	0.3072	0.5579	1.04266	0.00182
AdaBoost	1.6991	3.0965	5.6525	8.2076	12.02875	0.000769
KNN	0.02341	0.0333	0.034816	0.04315	0.058180	0.001829
Neural network	0.86944	1.4626	2.047	2.3479	2.34420	0.0007548

Observations:

1. AdaBoost is the best performing algorithm in predicting forest cover area. It's testing accuracy of 76.2% is way higher than rest of the algorithms. Closest one is KNN with 69.5% accuracy.
2. Decision tree achieves best accuracy in least number of data samples (3000). Although best decision tree accuracy is around 66.1% which is smaller than AdaBoost (76.2%) which achieves this accuracy in 4000 samples (full dataset).
3. Testing accuracy increases as data size for training increases for Adaboost, KNN and Neural Networks. Increase in this accuracy is drastic for KNN as it suffers from curse of dimensionality whereas for AdaBoost the more the data samples are the better the weights are for weak learner.
4. Neural Network accuracy saturates after 1000 samples and as shown in the learning curve decreases after 2100 samples. The more the samples are the more accurate are its network weights trained.
5. In terms of training time AdaBoost is the slowest which is natural since more the number of estimators more time it will take train the classifier. It iteratively trains weak learners (Decision Tree in this case) and combines them later.
6. Although SVM is supposed to be slowest algorithm it is faster than Neural networks and AdaBoost in this case since we have used dimensionally reduced dataset by applying PCA on the larger dataset.