September 22, 2019 (F19)
Md Khaled Hassan (ID: mhassan49)
Georgia Institute of Technology

Supervised Learning
CS7641: Machine Learning (Assignment 01)

Abstract:

The objective of this project is to explore various machine learning techniques, apply them on two different datasets, and understand and analyze their behavior on these datasets. I have looked at various learning curves such as test and train scores as well as the fitting time as a function of the training data size. I have reported ranking of the explored algorithms based on the 10 fold cross validation (CV) score, test score, and the training time. We observe that depending on the nature of the datasets, an algorithms performance can vary significantly.

Introduction:

The datasets that we explored in this projects are available from UCI machine learning repository [1]. The first dataset that we considered is *EEG (electroencephalography) eye state dataset* (Link: https://archive.ics.uci.edu/ml/datasets/EEG+Eye+State). This data set determines the eye state of a person and a binary class dataset ('1' indicates the eye being closed and a '0' indicates the eye being opened). This dataset is very interesting since the neurologists have been using EEG techniques to observe brain activity of patients to identify possible abnormalities that can give rise to neurological disorder. The dataset has 14980 instances and 15 attributes. The second dataset we explored in this project is the *Letter Recognition state dataset* (Link: https://archive.ics.uci.edu/ml/datasets/Letter+Recognition). This is another interesting dataset since image recognition has gained a very wide range of applications over the last decade. This is a multi-class dataset that identifies each of the 26 alphabets based on 16 attributes. There are 20000 instances of letter recognition provided in this dataset. Both of the datasets are fairly large with at least 15 attributes and therefore, provides interesting opportunity to analyze performances of various supervised machine learning algorithms. The algorithms that we applied are as follows and their performances were analyzed in the following sections:

- Decision trees (with some form of pruning)

- Neural Network (Multi-layer Perceptron (MLP))
- Boosting (AdaBoost)
- Support Vector Machines (SVC)
- k-nearest neighbors (KNN)

Data Processing:

In this project, I have used the scikit-learn [2], a built-in machine learning library in Python. Since the Letter Recognition dataset contains categorical classes, I have converted them to numerical values using the LabelEncoder function. For parameter tuning of both datasets, we have used 75% of the dataset to train the algorithms and the remaining 25% to test the algorithms. In addition, we have applied 10 fold cross validation (10 fold CV) on the training set to determine the most optimum set of parameters for each of the algorithms. We have normalized the data for SVM and NN (MLP) since without normalization, the runtime can be significantly increased for this algorithms. To plot the learning curves, we explored training datasets in the range of between 5% and 95% of the total dataset. The random state was kept constant (=100) for all simulations. Although I have started evaluating the models using the default settings, the learning curves provided here are after parameter tuning using the GridSearchCV function, using 10 fold CV. The training score provided on the learning curves are with out CV and only for comparison purpose. The CV scores are provided in table 1 and 2 in the summary section. A comparison among the algorithm were carried out only in the discussion section where we actually determined the rank of each of the algorithms on both of the datasets.

Decision Trees (with pruning)

EEG eye state dataset:

The decision tree is a weak learner and therefore, performs poorly when applied in classification problems. In order to apply pruning, we have looked at the train and test accuracy as a function of the tree depth. Fig.1(a) shows the corresponding plot. As we can see, the algorithm starts overfitting the data when the tree size goes above 10. Therefore, pruning the tree size can improve the performance of the algorithm and avoid overfitting. I have determined the

optimum tree size to be 15 using 10 fold CV method. Fig. 1(b) and (c) show the train and test accuracy and the fitting time as a function of the Training data size. We observe that the accuracy becomes almost saturated indicating that even with additional instances in the dataset, the algorithm accuracy may not improve. Learning time increases linearly with the increase in training time

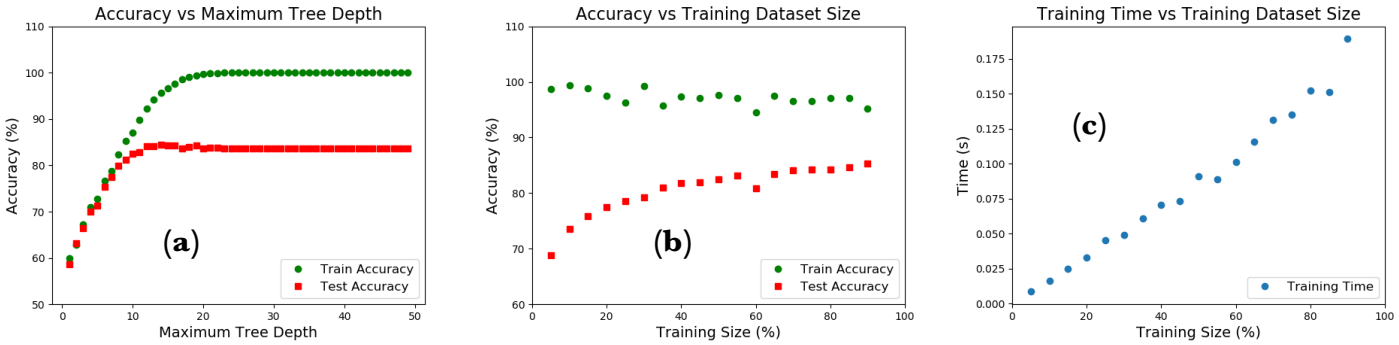


Fig.1: Performance and learning curves of Decision Tree Classifier (EEG eye state)

Letter Recognition dataset:

We observe similar trend in decision tree performance (Fig 2) for this dataset. However, both interims of training time and test accuracy, it performs slightly better on this dataset. Since, it's a weak learner, we don't expect any dramatic improvement of the accuracy score. The maximum tree depth that achieves best performance without significant overfitting is 20 in this case which also slightly higher compared to the EEG eye state dataset, which is a binary class dataset.

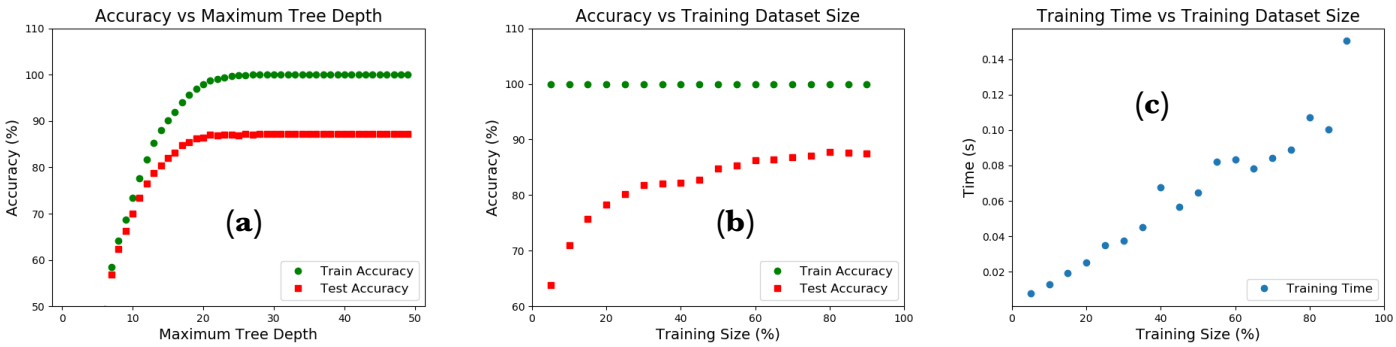


Fig.2: Performance and learning curves of Decision Tree Classifier (Letter recognition)

Neural Network (MLP Classifier)

EEG eye state dataset:

We have used Multilayer Perceptron (MLP) classifier for neural network analysis. Parameter meter tuning was carried out with 10 fold CV over four different options: 1) Penalty parameter ('alpha'), 2) The activation functions ('identity', 'tanh', 'relu'), 3) Hidden layer sizes, and 4) solvers ('lbfgs', 'sgd', 'adam'). Since MLP classifier is very time consuming, I have mostly focused on the two important parameters, activation function and hidden layer size. Based on the grid search results, the best performing activation function for this dataset was 'relu' with 'lbfgs' solver. I have also investigated the variation of accuracy as a function of the hidden layers (Fig 3(a)). We observe best accuracy for 200 hidden layers. Since MLP is a very time consuming classifier and the algorithm performs poorly on this dataset, we have limited the number to 100 in our parameter tuning process. We also observe significant increase in training time when the training dataset size is about 75% or above. The effect of the number of hidden layers seems to be random. This may have been caused by the possible noise present in the dataset. The insensitivity to training dataset size suggests this algorithm does not learn much from this dataset and therefore, performs very poorly.

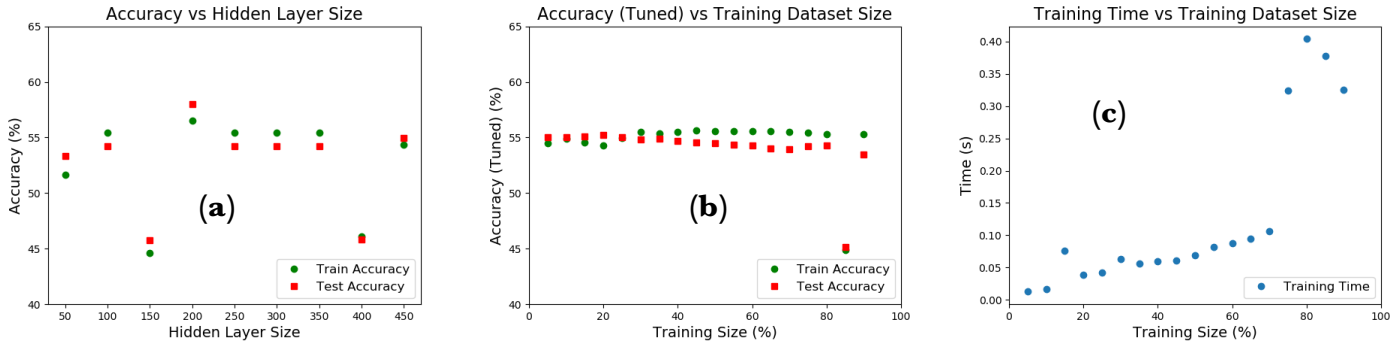


Fig.3: Performance and learning curves of Neural Network with MLP Classifier (EEG eye state)

Letter Recognition dataset:

The performance of the MLP classifier (Fig. 4) dramatically improves on this dataset and a test accuracy of about 90% is achievable after parameter tuning. The possible reason could be, this dataset is more balanced with less noise. The accuracy becomes saturated when the training size is 60% or above indicating the dataset has enough instances to train this model. However, the

fitting time is significantly high in this case and linearly increases with the increase in the size of training set. The number of hidden layers seems to have very minimal effect for the range that we considered for simulation.

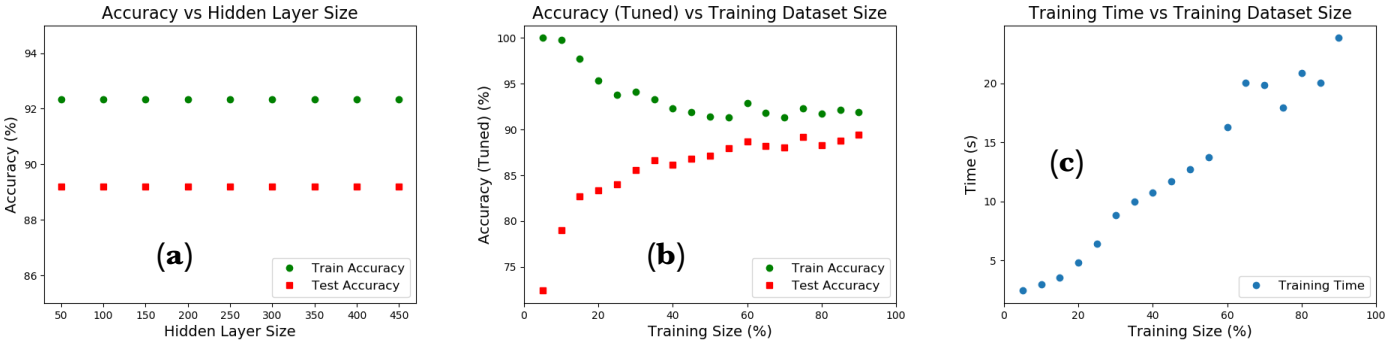


Fig.4: Performance and learning curves of Neural Network with MLP Classifier (Letter recognition)

Boosting (AdaBoost)

EEG eye state dataset:

AdaBoost algorithm was explored as the boosting algorithm. This algorithm is a strong learner with multiple weak learners as the base estimators (decision trees in this case). Grid search cross validations was performed on a number of estimators(tress), tree depth, and learning rates (between 0.75 and 1). The algorithm performs best with learning rate=0.9 and the number of estimators (n_estimator) =100, with each tree having a maximum depth of 10. Fig.5 shows various learning curves versus the number of estimators and training sizes. We observe very little change is train and test accuracy when n_estimators 100. The training score always reaches 100% since the estimators are decision trees. In addition, the test scores become almost insensitive to a training data size 60% or so. The fitting time is significantly higher and almost linearly increases with the training data size.

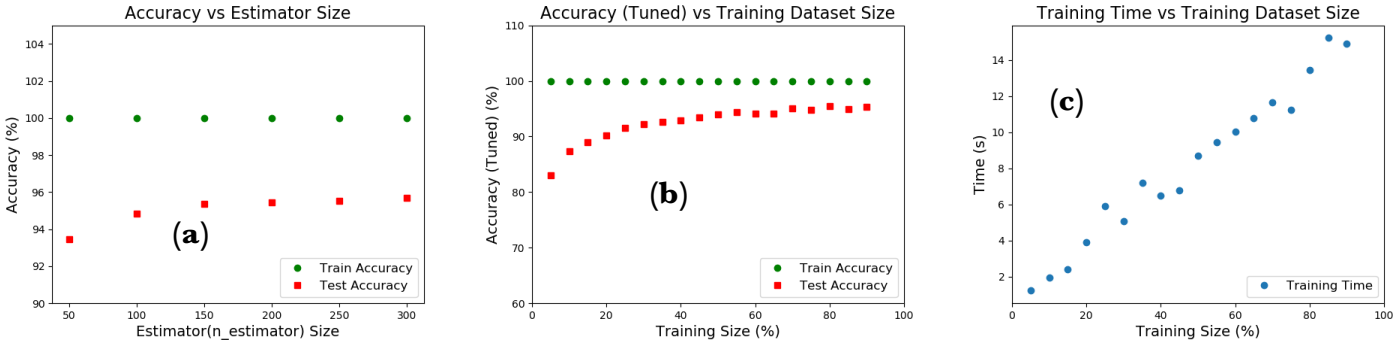


Fig.5: Performance and learning curves of AdaBoost Classifier (EEG eye state)

Letter Recognition dataset:

This algorithm performs even better on this multi-class dataset (Fig. 6). Trends are pretty similar to the EEG eye state dataset. However, the parameter tuning process revealed that for best performance, the algorithm required higher number of learners with more layers in the tree depth and slower learning rate (learning rate=0.75, number of n_estimator =300 with each tree having depth of 20). Although n_estimator =300 gives the best performance, the benefit of increase n_estimator is very marginal (Fig 6(a)). This indicates that this data set has more non linearity compared to the EEG eye state dataset. The learning time is also significantly higher and increases significantly with higher training dataset.

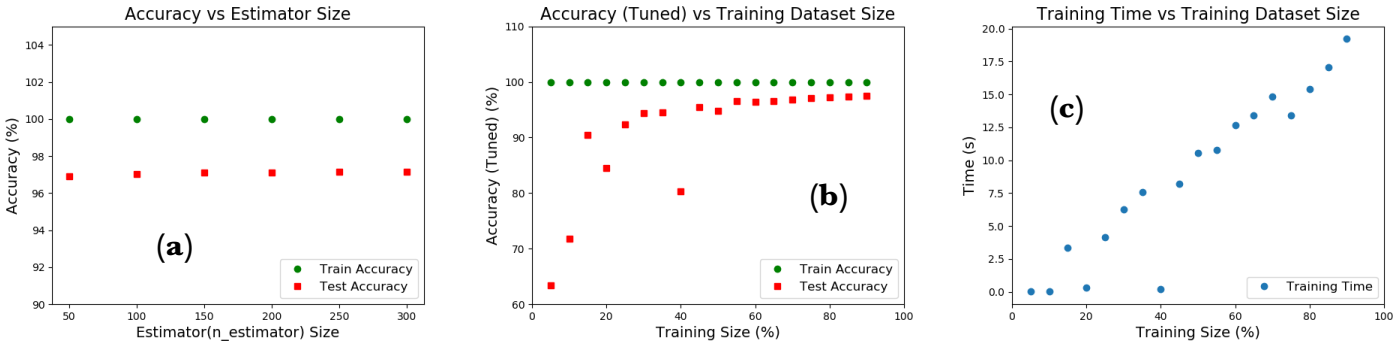


Fig.6: Performance and learning curves of AdaBoost Classifier (Letter recognition)

Support Vector Machines (SVC)

EEG eye state dataset:

I performed extensive parameter tuning with 10 fold cross validation. The algorithm performs very poorly on this dataset. For parameter tuning, a grid search with 10 fold CV was carried out. The parameters that we tuned are the penalty parameter (C) and the kernel type (linear and radial basis function (rbf)). The kernel coefficient (gamma) was set to 'scale' which passes 1 / (n_features * X.var()) as the value of gamma. The performance on this dataset is almost insensitive to kernel type indicating this algorithm is not suitable for this dataset. We observe from the figure (Fig. 7(a)) that the value of 'C' plays a role till about a value of 70 or below. The training time is very high compared to most other algorithm. The training size does not seem to play much role. However, the training time does show sensitivity to the training data size. The learning plots suggests this algorithm is not at all suitable for this dataset.

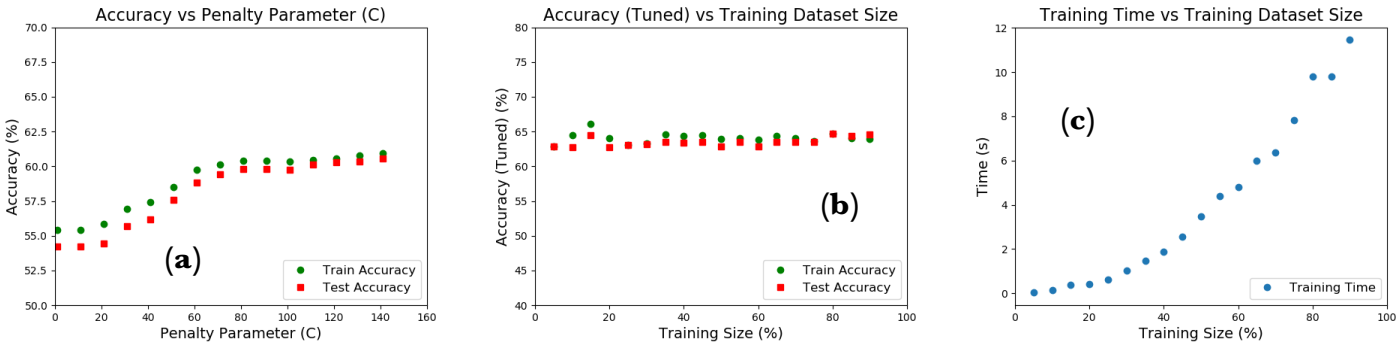


Fig.7: Performance and learning curves of SVM Classifier with 'rbf' kernal (EEG eye state)

Letter Recognition dataset:

We performed similar parameter tuning on this dataset and investigated the similar learning curves (Fig. 8). We see that the test accuracy reaches very high (close to 95%) when C~10. Plot 8(b) suggests the SVM learns very quickly on this dataset, about 50% data size is enough to train the algorithm. We observe reasonable training time that increases with the increase in training dataset size (8(c)).

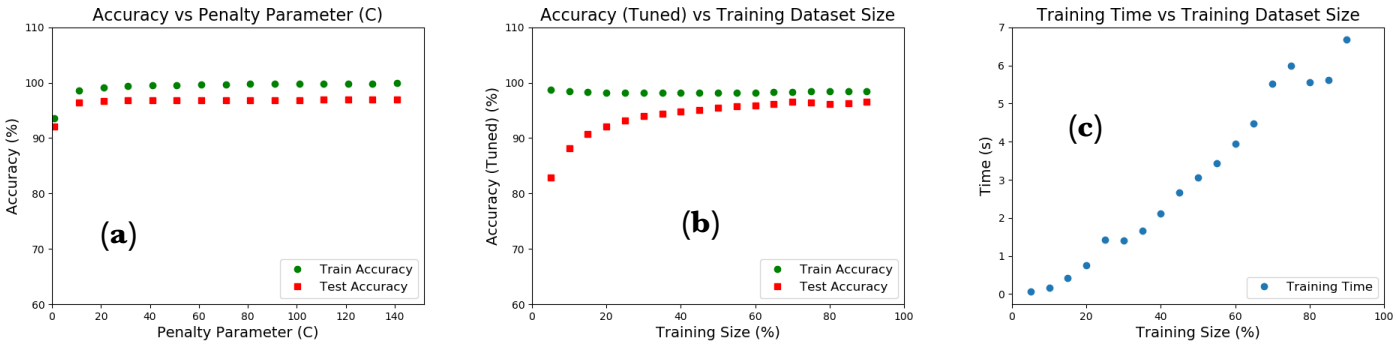


Fig.8: Performance and learning curves of SVM Classifier with 'rbf' kernal (Letter recognition)

k-nearest neighbors (KNN)

EEG eye state dataset:

For KNN algorithm, the most significant parameter is the number of neighbors (n_neighbors) and the algorithm performs best when the number is=1. This is because, the algorithm learns best from the nearest neighbor. We observe from Fig. 9(a) that the accuracy keeps increasing and the train and test accuracy merges at some point if we keep increasing the n_neighbors parameter. The parameter tuning was done for various values of n_neighbors with both Manhattan (p=1) and Euclidian distance (p=2). For n_neighbors=1, the algorithm achieves a test accuracy close to 98% with p=2 (Fig 9(a)). However, the learning curve (Fig. 9(b)) suggest that the accuracy keeps increasing even when the test size is 95%. This indicates that the dataset does not have enough instances for the complete learning of this algorithm. The training time is very fast as shown in Fig. 9(c) and increases almost linearly with the increase in the test data size.

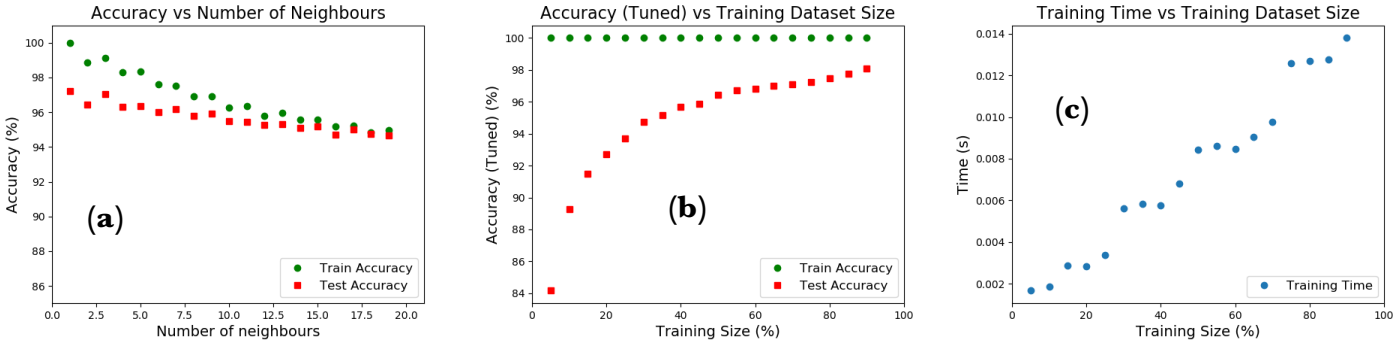


Fig.9: Performance and learning curves of KNN Classifier (EEG eye state)

Letter Recognition dataset:

The KNN algorithm performs slightly worse on this dataset compared to the EEG eye state dataset (Fig 10). This suggests that the algorithm performs better on the binary class dataset. This also indicates, a binary class dataset has higher possibility of repletion of instances. The best parameter are still same as the EEG eye state dataset (n_neighbors=1, p=2). The test accuracy becomes almost saturated when the test data set reaches about 80%. This indicates high number of samples for training is necessary for the learning of this algorithm (consistent with the other dataset). The training time suggests that it's a very fast learner and fitting time is only about 0.1 s when the training size is 95%.

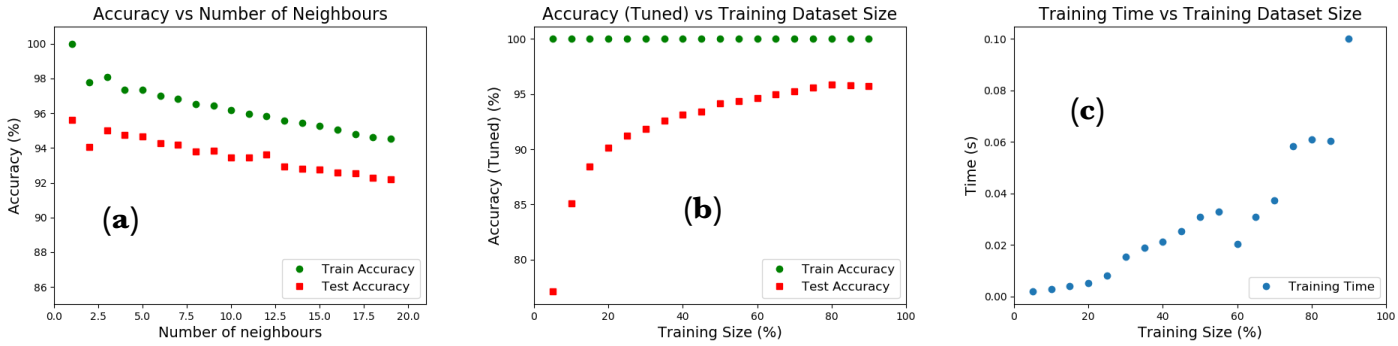


Fig.10: Performance and learning curves of KNN Classifier (Letter recognition)

Summary and Discussion

I have listed the training accuracy, CV-10 accuracy, and Test accuracy and the train or fit time for each of the datasets in Table 1 and 2. For all the cases, the 75% dataset was assigned for training and cross validation, and the remaining 25% was used to generalize the algorithms. The CV-10 score, test score, and the training times are for most optimized parameter settings. The following conclusions can be drawn based on the learning curves that we discussed so far and the parameters listed in these two tables:

- The Decision Tree, AdaBoost, and KNN algorithms performs too well on the training datasets if we don't perform cross-validation. Therefore, these algorithms have a tendency to overfit data.

- Since both datasets ate reasonably large, they algorithms have sufficient learning instances for generalization. Both SVM and MLP (NN) perform very poorly on the EEG dataset. The possible reason can be the presence of noise in the dataset that prevents this algorithms from performing all. Both of their performance considerably improves in Letter Recognition dataset. In SVM linear kernal function performs worse than the radial basis function (rbf). This implies that the data sets are not linearly separable.
- KNN and AdaBoost performs very well on both of the datasets. KNN performs better on the binary class dataset while the AdaBoost algorithm performs somewhat on the multi-class dataset. Since AdaBoost uses decision trees as weak learners and uses many of them to remove overfitting, this algorithm is expected to perform well on the large dataset. In terms of training time, decision tree and KNN are much faster than the other algorithms.
- Chart 1 and 2 provides ranking of each of the datasets (A rank of 1 being the best and 5 being the worst). The rank criteria are CV-10 score, test score, and the training time.
- KNN outperforms all other algorithms in all three categories for the first dataset (Table 1 and Chart 1). Both SVM and Neural Network perform poorly on the EEG Eye dataset. It is possible the dataset may have bias and may not be balanced.
- AdaBoost, SVM, and KNN are the top performers on the Letter Recognition dataset in terms of test score (Table 2 and Chart 2). KNN is still the fastest among all algorithms since its a lazy learner. Except decision tree, all other algorithms have at least 80% CV-10 score. These results suggest, the dataset is well balanced, has less bias and noise.

(Table.1: Accuracy with tuned parameters (EEG Eye State dataset)

Algorithms	Training Accuracy	CV-10 Accuracy	Test Accuracy	Fit Time (s)
Decision Tree	**0.966**	**0.789**	**0.842**	**0.13**
Neural Network (MLP)	**0.554**	**0.542**	**0.635**	**0.46**
AdaBoost	**1.0**	**0.905**	**0.948**	**11.28**
SVM (Kernel=rbf)	**0.604**	**0.632**	**0.598**	**12.45**
SVM (Kernel=linear)	**0.636**	**0.649**	**0.535**	**24.9**
KNN	**1.0**	**0.936**	**0.972**	**0.01**

(Table.2: Accuracy with tuned parameters (Letter Recognition dataset)

Algorithms	Training Accuracy	CV-10 Accuracy	Test Accuracy	Fit Time (s)
Decision Tree	1.0	0.781	0.871	0.11
Neural Network (MLP)	0.923	0.837	0.892	18.42
AdaBoost	1.0	0.82	0.97	17.24
SVM (Kernel=rbf)	0.984	0.928	0.964	3.31
SVM (Kernel=linear)	0.878	0.84	0.856	27.25
KNN	1.0	0.905	0.956	0.07

Chart 01. Algorithm Rank (EEG Eye State)

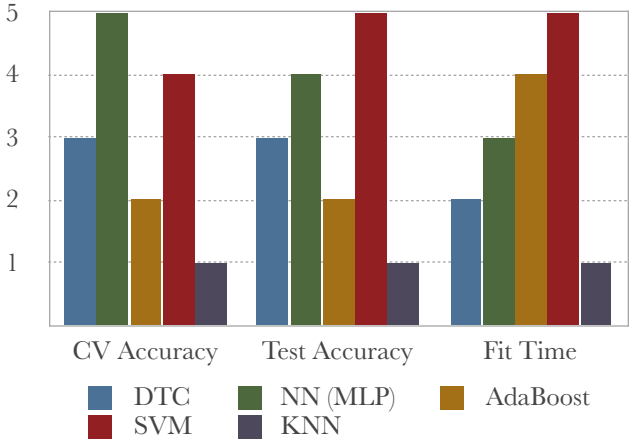
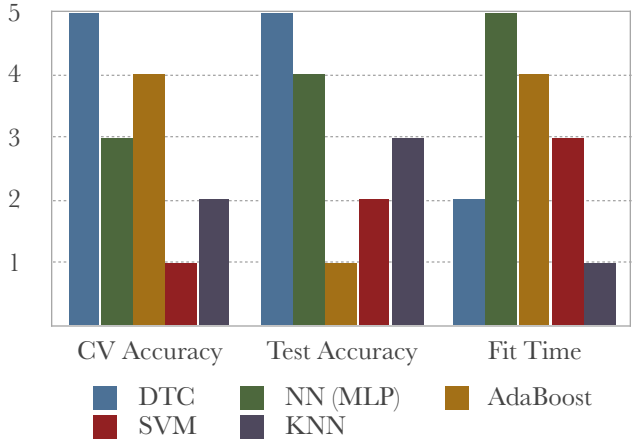


Chart 02. Algorithm Rank (Letter Recognition)



References:

1. UCI repository (https://archive.ics.uci.edu/ml/datasets.php)
2. Scikitlearn (https://scikit-learn.org/stable/index.html)
3. Empirical study of seven data mining algorithms on different characteristics of datasets for biomedical classification applications (https://www.researchgate.net/publication/320816509_Empirical_study_of_seven_data_mining_algorithms_on_different_characteristics_of_datasets_for_biomedical_classification_applications)
4. Udacity lectures of CS7641