Abstract:

In this report, 5 learning algorithms (Decision Tree, Neural Network, Boosting, Support Vector Machines and k-nearest neighbors) are implemented for two classification datasets. SKlearn is used for implementation and gridSearchCV is used for parameter tuning. Results and comparison are given and discussed.

Datasets:

1. First data set is the ***Letter Recognition Data Set***[1] from UCI. It consists 20000 instances of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. Each instance has 16 primitive numerical attributes such as *width of box*, *total pixel,* etc. The dataset is featured as a balance set.
2. Second set is the ***Car Evaluation Data Set***[2] from UCI. It has 1728 sample of car evaluation from unacceptable, acceptable, good to very good. The 6 attributes deciding the value consist buying cost, number of doors, safety level and so on. The dataset is featured as an un-balance set. Detail information shown in reference.

Objectives:

1. ***Letter Recognition Data Set***: Letter recognition is a fundamental but crucial task in sophisticated computer vison framework. The application of the corresponding machine learning model could be applied to recognize the road sign for autopilot, recognize the ancient letters for archaeology or instruments for disabilities. It is also used to test the performance of these algorithms on balance data set.
2. ***Car Evaluation Data Set***: Evaluation problem could be widely adopted in financial market. Current car evaluation model could be used in car trading. The results could also suggest the best model for all kinds of evaluation system used in stock, real estate or other properties. The performance of 5 algorithms are also showed for un-balance data set.

Environment:

Python version: 3.7

Package used: sklearn, Numpy, SciPy, Pandas, matplotlib and time.

Results for Letter Recognition:

Decision Tree (DT)

The Gini impurity is used for tree split. Except the max_depth of the tree, all parameters are set as default value. Whereas max_depth is first tuned to optimize the learning process and prevent overfitting. It also works as the pruning function to avoid the extreme growth of the tree since sklearn does not have a built-in pruning method.

gridSearchCV (10 folds for all CV in this report) is used to check the accuracy score for max_depth in range [1, 16], the mean test scores and standard deviation of test scores of cross validation with different max_depth is shown in Fig.1 and Fig.2. Though growth of the tree may bring overfitting, in this particular case it does not. The average accuracy of cross validation is higher with deeper DT whereas the std growing slowly. That means basically all 16 features have some influence on the decision and the database is internally consistent. The noise is low which results in higher accuracy with bigger tree.

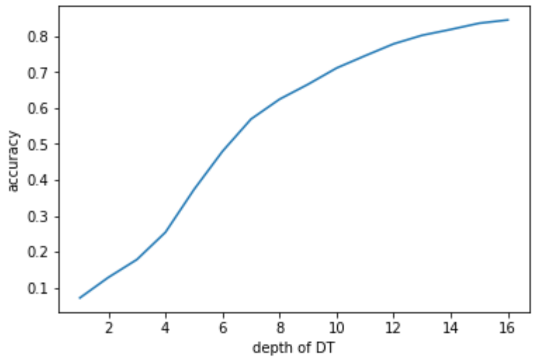


Fig. 1 Mean accuracy of cross validation

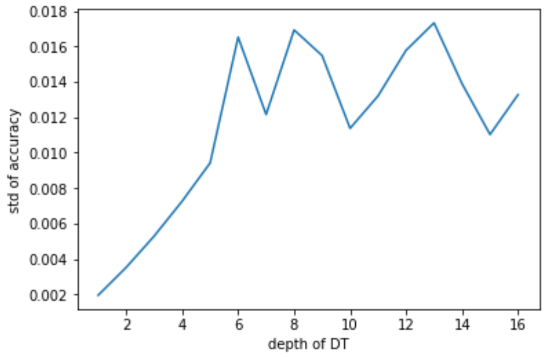


Fig. 2 Std of accuracy of cross validation

As a result, basically no pruning needs to be conducted in this case and use the DT with depth of 15 to train the data. Because the accuracy is at a same level of DT with depth of 16 but the std is smaller thus more stable. Fig.3 shows the training scores of DT with depth of 15. The accuracy is pretty high and the cross validation score keeps growing with more training examples. Fig.4 shows the learning curve with fractional training data size. With more training, the test scores grows constantly.

In all, DT shows pretty good performance in recognizing the letters. The dataset shows low noise and all 16 attributes show some influence on the recognition.

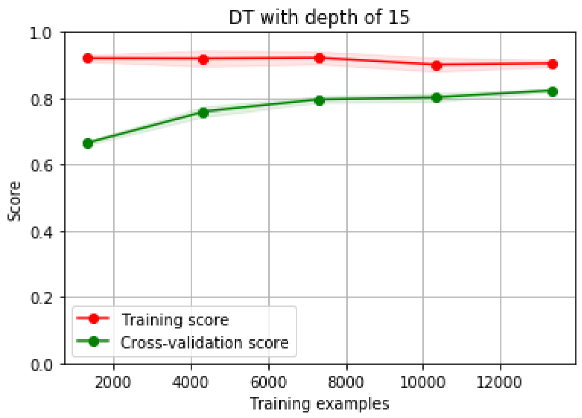


Fig. 3 Learning curve of DT with depth of 15

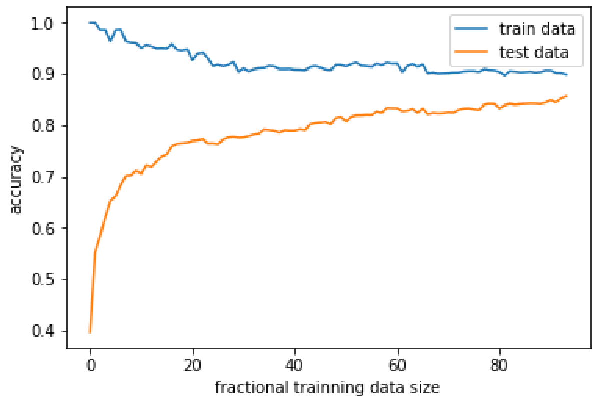


Fig. 4 Learning curve with fractional data size

Boosting with Decision Tree

For boosting, we use the DT as weak learner. Number of estimators in boosting is optimized by gridSearchCV for DT with different depth. Results in Fig.5 shows that more estimators in boosting yield higher accuracy, however the improvement is limited, especially when take the computing time into consideration. Take DT with depth of 5 as example, shown as Tab.1, basically the computing time doubled with twice number of estimators, however the accuracy increases slightly. For the DT with depth of 10 and more, the accuracy score increases even lesser.

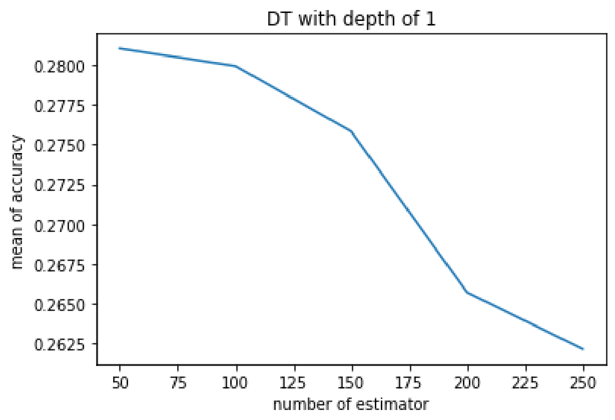
Tab.1 Computing time for DT with depth of 5

n_estimator	50	100	150	200	250
Nomalized time	1.00	2.00	2.97	3.95	4.90

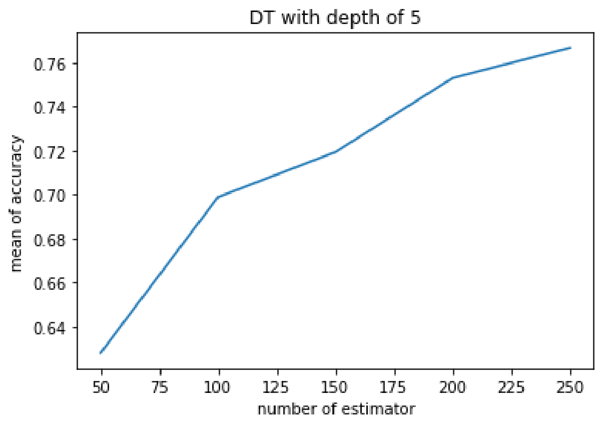
Notice that with boosting, low depth DT also gives relatively accurate results. For same deep DT, the improvement of using boosting is remarkable, especially when the tree is not deep, shown as Tab.2.

Tab. 2 Accuracy of different cases

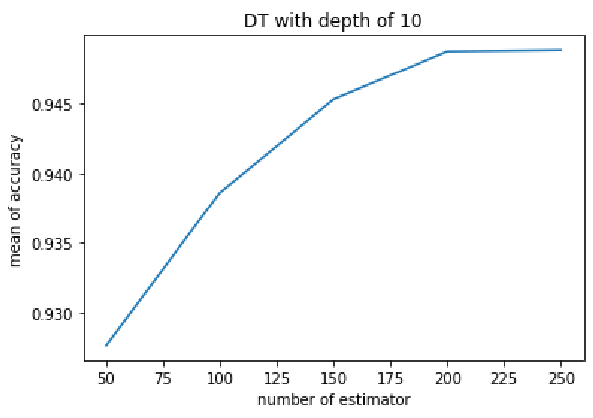
Tree depth	DT	Boosting with 50 estimator	Improvement
1	0.08	0.28	250%
5	0.36	0.63	75%
10	0.7	0.93	33%
15	0.85	0.96	13%



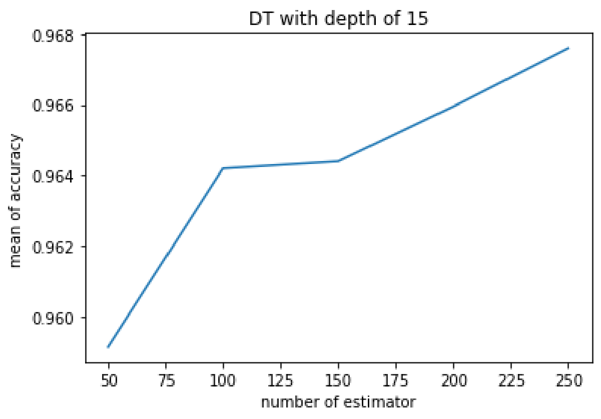
(a) DT with depth of 1



(b) DT with depth of 5



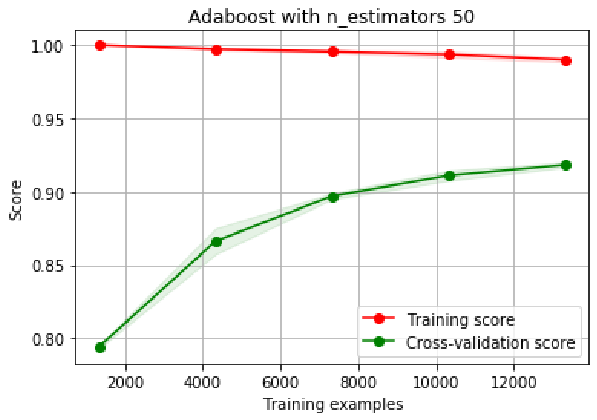
(a) DT with depth of 10



(b) DT with depth of 15

Fig.5 Mean accuracy of cross validation for different boosting cases

Balance the accuracy and time, boosting with 50 estimators and DT with depth of 50 is used for data prediction. As shown in Fig.6 and Fig.7, predicting accuracy is pretty good, even better than the DT, though the tree depth is lower.



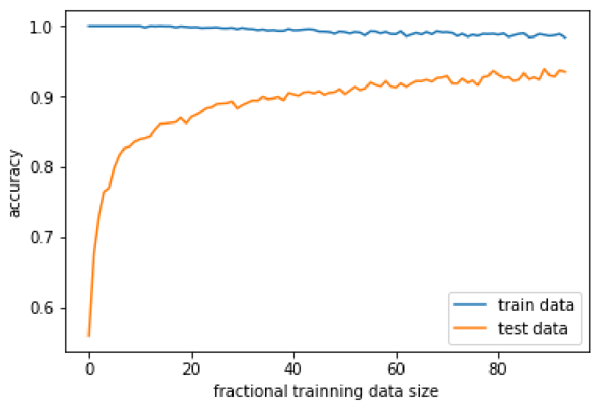


Fig. 6 Learning curve of DT with depth of 15 **Fig. 7** Learning curve with fractional data size

Neural Network (NN)

For the Neural Network, the key parameter, number of hidden layer and perceptron could be selected from a quite large range. Use gridSearchCV may be extremely time consuming. Thus we decide to use two different schemes and take a quick look at the influence of the choosen parameters. Thus we tried a 1 layer (26) and a 2 layer (16, 8) since there are 26 letters. Results in Fig.8 and Fig.9 shows that more perceptron at last layer may lead higher accuracy, even though the total perceptron for the 2 layer case is more. Overall the accuracy is less than DT and Boosting.

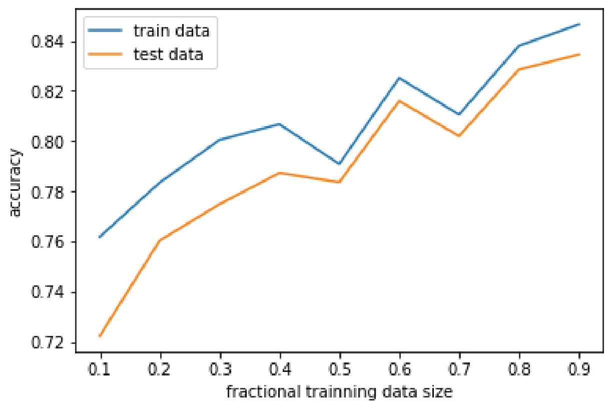


Fig.8 Learning curve of NN(26)

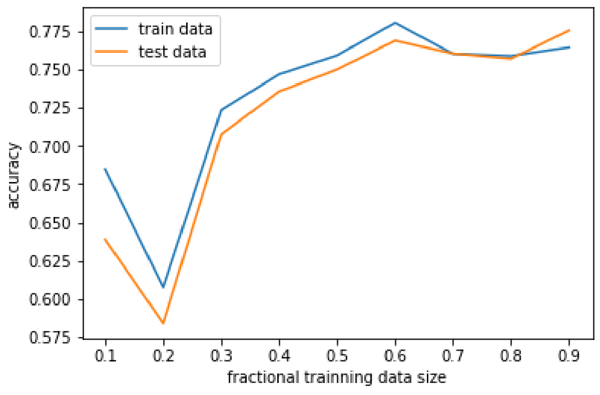


Fig.9 Learning curve of NN(16, 8)

Support Vector Machines (SVM)

For the SVM, RBF, linear and sigmoid kernel are adopted for comparison. Fig.10, Fig.11 and Fig.12show that RBF and linear kernel can produce acceptable results. However the sigmoid kernel is not available to this problem.

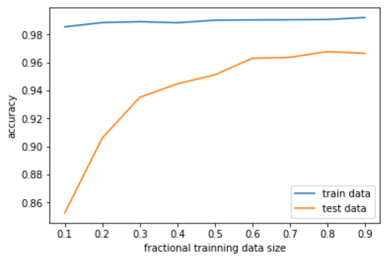


Fig.10 RBF kernel

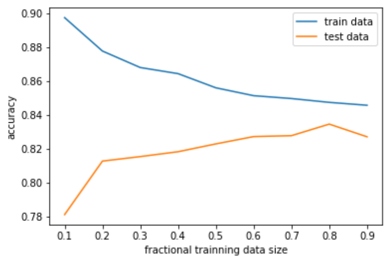


Fig.11 Linear Kernel

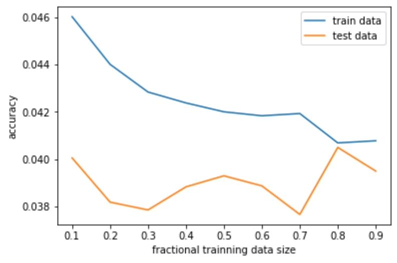
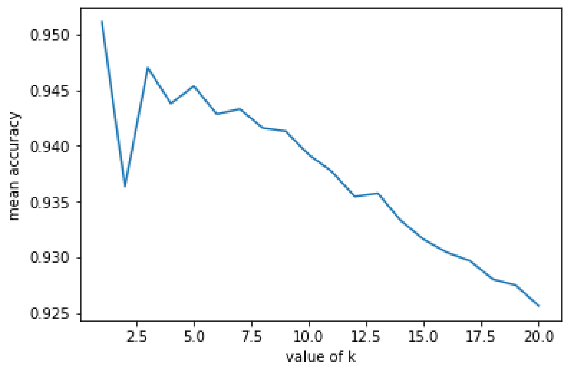


Fig.12 Sigmoid Kernel

k-Nearest Neighbors (KNN)

For KNN, the optimal k value is searched by gridSearchCV in range of [1, 20], shown as Fig.13. Fig.14. Results show that k=1 gives highest accuracy and lowest std. Thus k=1 is used for prediction.



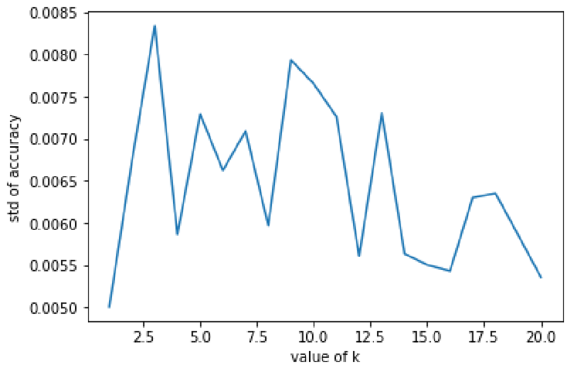
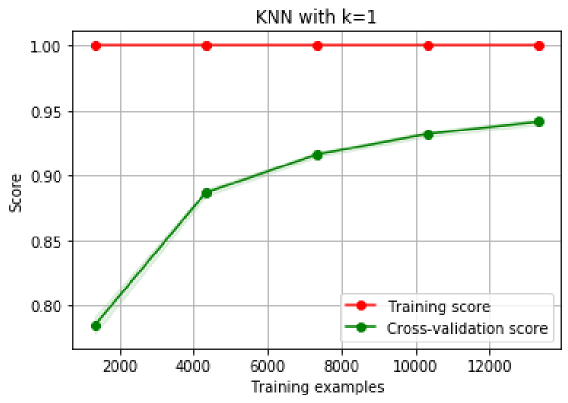


Fig.13 Mean accuracy of KNN at different k **Fig.14** Std of accuracy of KNN at different k

Fig.15 and Fig.16 show the learning carves of KNN with k=1. KNN shows great performance in this problem. It is reasonable since each letter is relatively independent to each other, thus the instance could be classified by nearest 1 instance.



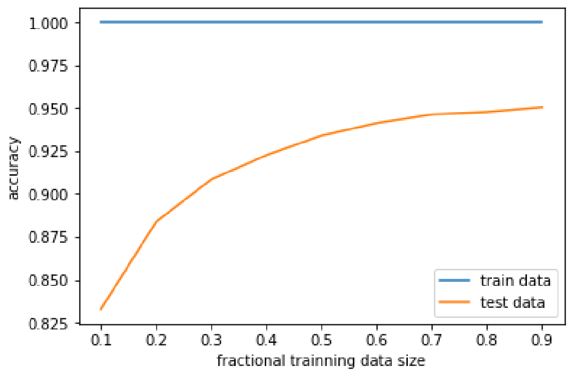


Fig. 15 Learning curve of KNN(k=1) **Fig. 16** Learning curve with fractional data size

Results for Car Evaluation:

Decision Tree (DT)

Firstly, the depth of DT is check by gridSearchCV for pruning purpose. Shown as Fig.17 and Fig.18, when the depth is 4 it reaches the highest accuracy. Thus in this case when the depth higher than 4 it may cause overfitting. The standard deviation of DT with different depth however maintain a low level. Thus for current case we choose max_depth=4 for data prediction. Learning curves are shown as Fig.19 and Fig.20. DT shows good performance with accuracy around 0.8 for train data, cross validation date and test data. The computing time is quite impressive which is too short to be plot by python.

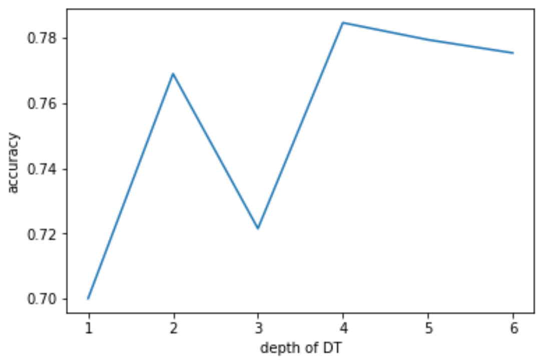


Fig. 17 Mean accuracy of cross validation

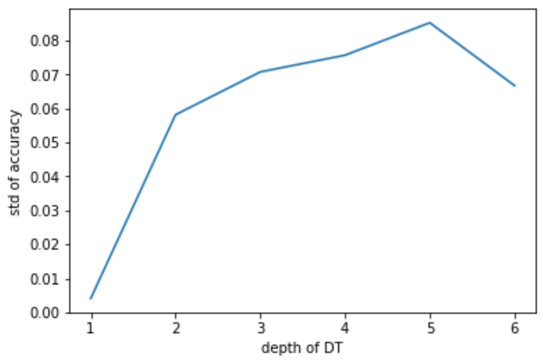


Fig. 18 Std of accuracy of cross validation

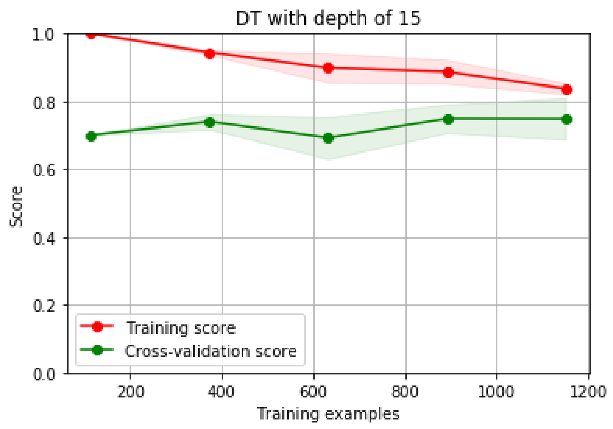


Fig. 19 Learning curve of DT with depth of 15

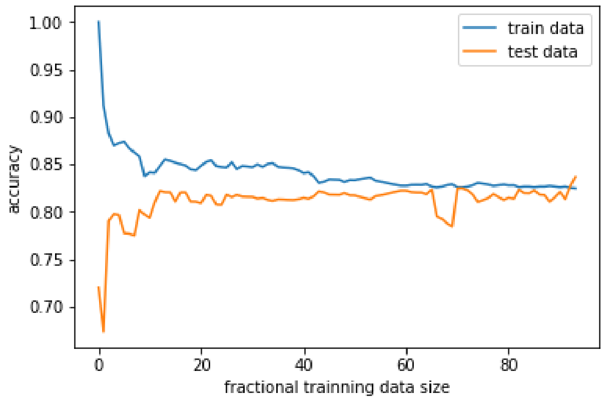


Fig. 20 Learning curve with fractional data size

Boosting with Decision Tree

For boosting, we use the DT as weak learner. According to the result from DT, we use the depth=4 directly. The estimator number is checked for parameter tuning. As shown in Fig.21, again, more estimator can just improve the performance limitedly. Thus the n_estimator=50 is used for further modelling.

Learning curve is shown in Fig.22 and Fig.23. Boosting show better results comparing with DT again. The computing time is around 0.7s.

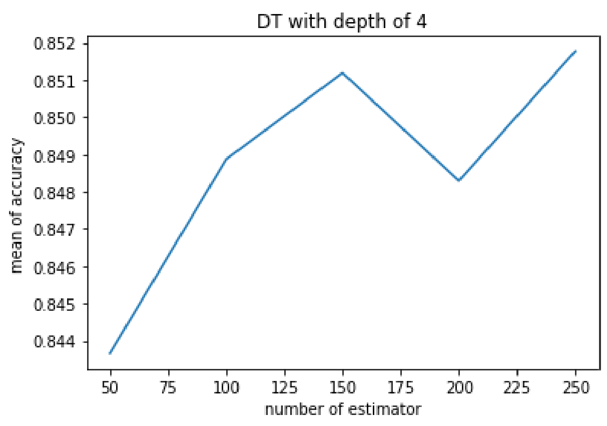
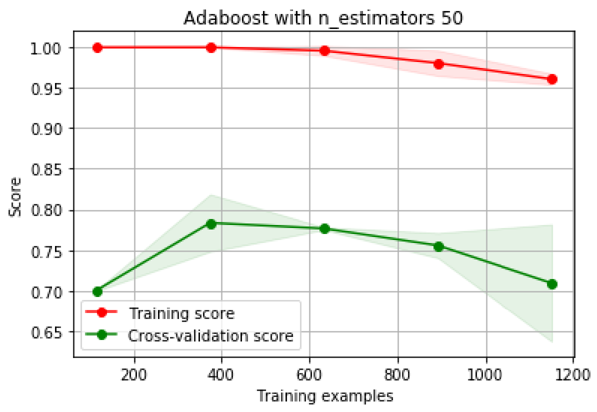


Fig.21 Mean accuracy of cross validation for boosting with different estimator



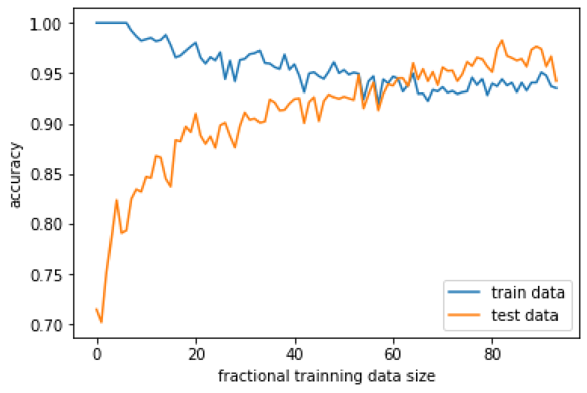
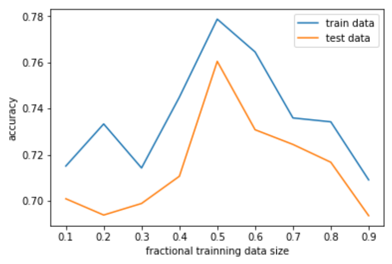
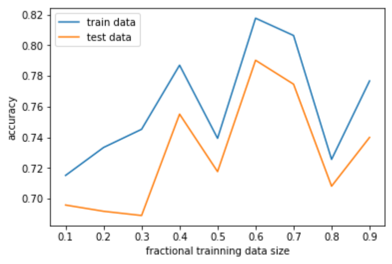


Fig. 22 Learning curve of boosting **Fig. 23** Learning curve with fractional data size

Neural Network (NN)

For the Neural Network, the key parameter, number of hidden layer and perceptron are selected with three strategies: 1 layer(10), 2 layer(10, 6) and 3 lay(25, 15, 10). Different from the letter recognition problem, the car evaluation shows higher complexity that needs more layers and perceptron for modelling. Fig.24-26 show that reasonable increase of perceptron does improve the performance. Fig.27 shows the cross validation scores of NN(25, 15, 10). Result shows that current choice of layer and perceptron is quite ideal. The computing time cost around 6.6s.





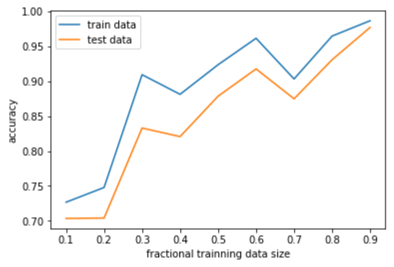


Fig.24 Learning curve of NN(10) **Fig.25** NN(10, 6) **Fig.26** NN(25, 15, 10)

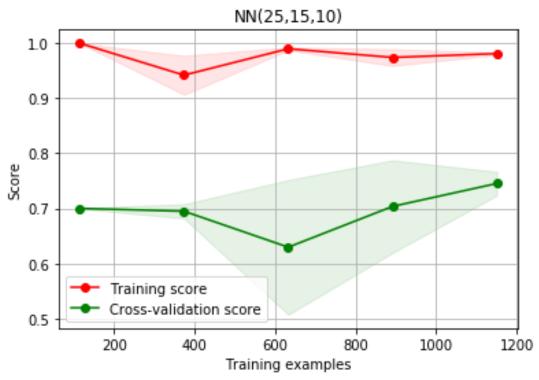
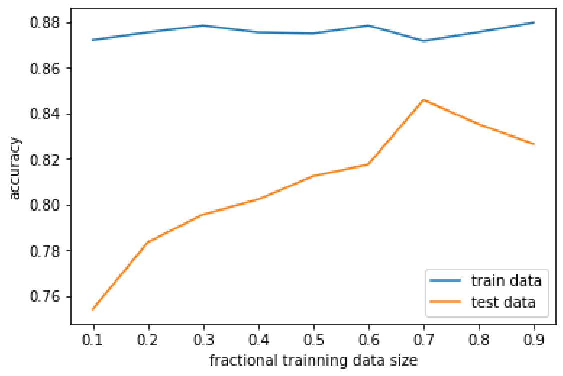


Fig.27 Mean accuracy of cross validation for NN(25, 15, 10)

Support Vector Machines (SVM)

For the SVM, RBF and poly kernel are used for test. Fig.28 and Fig.29 show that both kernels can produce accurate results. Recalling the results from Letter Recognition, the rbf kernel seems to be an ideal choice for various problems. Consider the accuracy, the time consumed is quite short, around 0.1s.



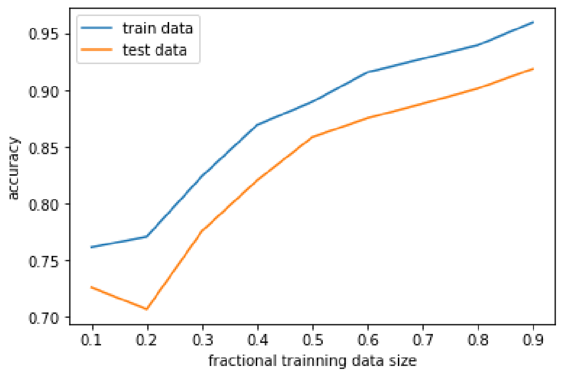
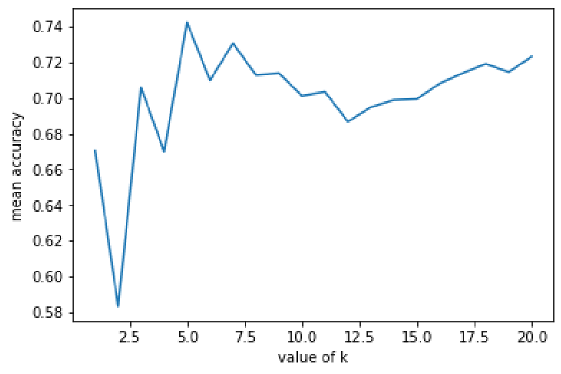


Fig. 28 Learning curve of poly kernel **Fig. 29** Learning curve of rbf kernel

k-Nearest Neighbors (KNN)

Firstly, the optimal k value is searched by gridSearchCV in range of [1, 20], shown as Fig.30. Fig.31. Results show that k=5 gives highest accuracy. The standard deviation of accuracy for k=5 is at the middle range, thus we use the k=5 for our KNN learning. Fig.32 and Fig.33 show the learning curve of KNN(k=5). It yields a relatively good result with a short time, around 0.1s.



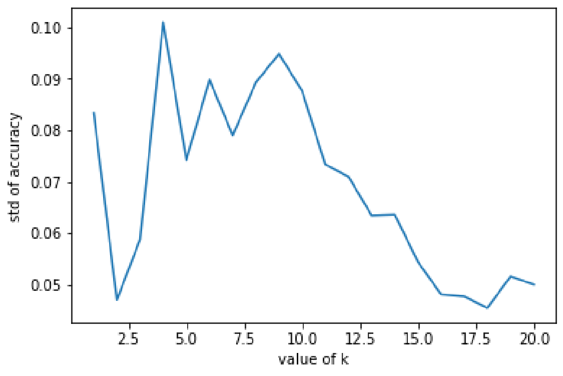
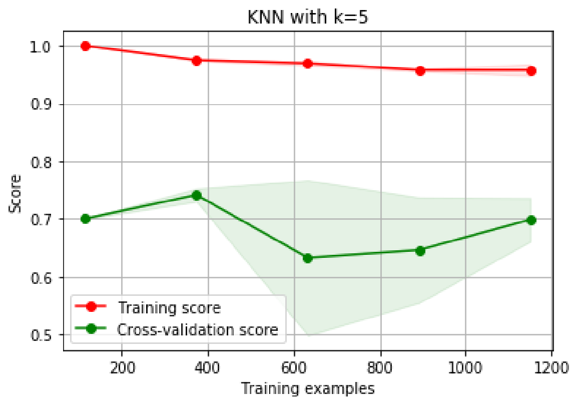


Fig.30 Mean accuracy of KNN at different k **Fig.31** Std of accuracy of KNN at different k



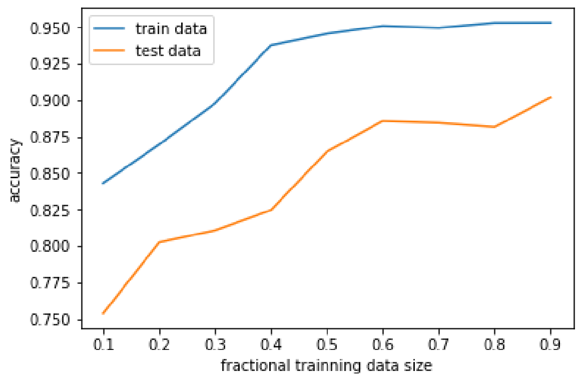


Fig. 32 Learning curve of KNN(k=5) **Fig. 33** Learning curve with fractional data size

To give a deeper look of these 5 algorithms, we compared the results from dataset2, shown as Tab.3. After reasonable parameter tuning, all 5 algorithms yield good prediction results. The accuracy of decision tree is relatively lower, however the computing time is extremely fast. Neural network yields highest accuracy, both train accuracy and test accuracy, but the computing time is much more than the other. Notice that neural network is quite good for the car evaluation problem, however for the letter recognition, the performance is not so good as the car evaluation. Boosting, SVM(rbf) and KNN actually show pretty good results for both problems. The improvement of DT using boosting is really impressive. The accuracy increases even with aggressive pruning.

Tab. 3 Comparison between algorithms for dataset2

	Train accuracy	Test accuracy	Time/s
Decision Tree	0.83	0.82	~0
Boosting	0.95	0.94	0.7
NN(25, 15, 10)	0.97	0.95	6.6
SVM(rbf)	0.95	0.9	0.1
KNN	0.95	0.9	0.1

Summary:

In this report we applied 5 machine learning algorithms to 2 different datasets. With reasonable parameter tuning, all of them yield acceptable or good prediction to the data. Detail performance could be conclude as:

Decision Tree: For both datasets, the decision tree shows show acceptable results. The needs for pruning heavily dependent on the quality and feature of data. For the dataset1 (Letter Recognition), even the depth expand to 16, the cross validation accuracy and test accuracy is high, so we did not apply pruning. However for the second dataset (car evaluation), though the depth is 5 or 6, the cross validation scores decrease. So we choose to prune the tree to depth of 4. Comparing with other algorithms, decision tree is always the fastest.

Boosting with Decision Tree: It is really impressive that boosting actually "boost" the decision tree quite well. We could use less deep tree with boosting to yield a result better than Decision Tree. So we could be pretty aggressive on pruning when busing boosting. Besides, more estimator does gives a higher accuracy, however the improvement is limited. The computing time is relative higher than others, except Neural Network.

Neural Network: The performance depends on the feature of the dataset and parameter adjustment. For the Letter Recognition less perceptron and layer gives higher accuracy. But even after tuning, the results is just acceptable. However, for car evaluation problem the result of NN is the best. Time consuming might be the biggest drawback of NN. It costs much more than the other.

k-Nearest Neighbors: After adjust the value of k, KNN shows good performance for both case. The k value has big influence on the prediction. But it is quite different for different dataset. So we need adjust it for each problem. The computing time is pretty short.

Support Vector Machines: For both datasets, it shows good performance. However the choice of kernel is very important. Basically the default kernel "rbf" could be adopted to both, but the "Sigmoid" is relatively worse. It is efficient too. The time used is of same magnitude as KNN.

Reference:

1. https://archive.ics.uci.edu/ml/machine-learning-databases/letter-recognition/

2. https://archive.ics.uci.edu/ml/machine-learning-databases/car/