Assignment 1 - Supervised Learning

Xujian Liang GTID:903421366

Abstract

This report analyzes the result of two different datasets trained by five learning algorithms including Decision Tree, Neural Network, Decision Tree of Boosting, Support Vector Machines and k-Nearest Neighbors. In this report, effect of different parameters of learning algorithms is also analyzed.

Dataset Description and Preprocessing

Letter Recognition (Dataset 1)

This dataset is obtained from the UCI Machine Learning Repository and donated by David J. Slate. The objective is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. The character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20,000 unique stimuli. Each stimulus was converted into 16 primitive numerical attributes (statistical moments and edge counts) which were then scaled to fit into a range of integer values from 0 through 15. The dependent variable "capital letter" has 26 categories from A-Z. It can be seen from Figure 1 that the distribution of dependent variable is about evenly distributed, i.e. this dataset is not very skewed. Therefore, we choose to use accuracy as scoring for this dataset when training this dataset in the future.

Census Income (Dataset 2)

This dataset is also obtained from the UCI Machine Learning Repository. The objective is to predict whether income exceeds $50K/yr based on census data. There are 48842 instances and 14 attributes including 6 continuous variables (age, fnlwgt, education-num, capital-gain, capitalloss and hours-per-week) and 8 categorical variables (workclass, education, marital-status, relationship, race, sex and native-country).

I think these two datasets are interesting because both of them have a large number of instances and attributes. Additionally, I chose one of datasets as binary-class and the other as multiclass.

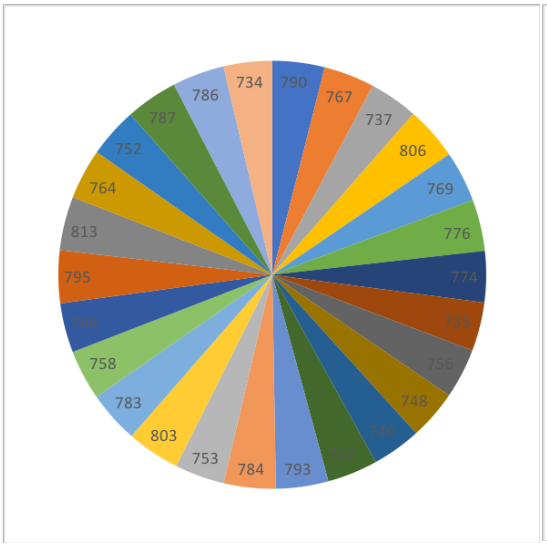


Fig. 1 Distribution of dependent variable (dataset 1)

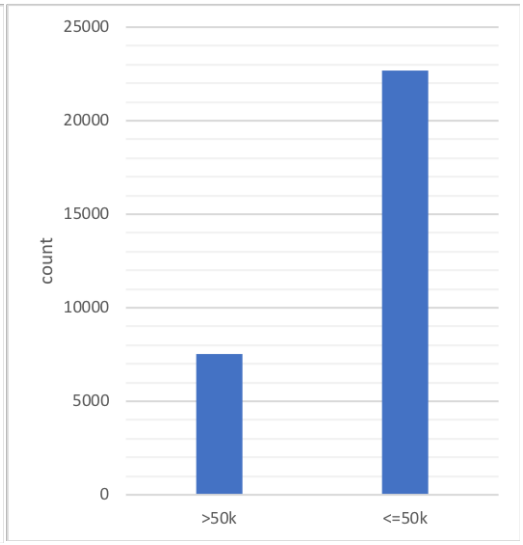


Fig. 2 Distribution of dependent variable (dataset 2)

Implementation and Analysis

All dataset is 30-70 split into two datasets by resample method in Weka Preprocess. 70% is training and remaining is for test. Various supervised learning algorithms are applied on the training set and then on test set. Hypeparameters are tuned by 10-fold cross validation on the training set in the model selection. The model will be selected based on the best cross validation score, which is the classification accuracy on the training set. Test set will not be touched during the model selection process and will only be used to test the models. A detailed analysis and the learning curve will be given in each of the learning algorithm. The highlight parameters will be the best model for this dataset.

Decision Trees

For decision tree, J48 is used. I will tune the following parameters:

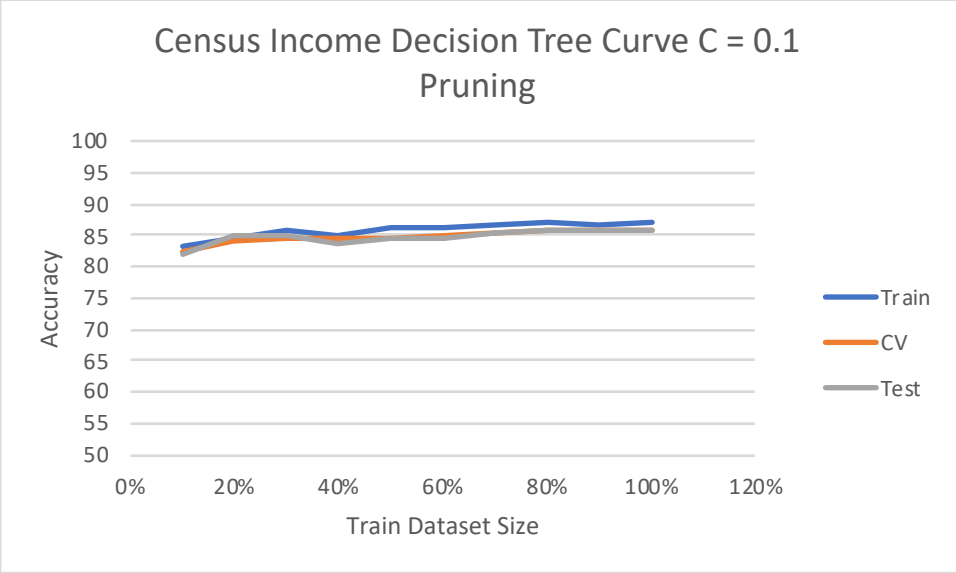
1. Confidence factor, which set the confidence threshold for pruning. Default is 0.25, larger confidence factor will lead to more pruning

2. Unpruned or pruned.

Pruning is often used to reduce the size of tree. However, pruning would decrease the accuracy of the training set, but in general it will increase the accuracy for test data. It is used to mitigate overfitting, in case the model achieves too good accuracy for training data, but it will perform poorly on test data. The training is usually fast. The minimal instance at leaf node is set 2 at default.

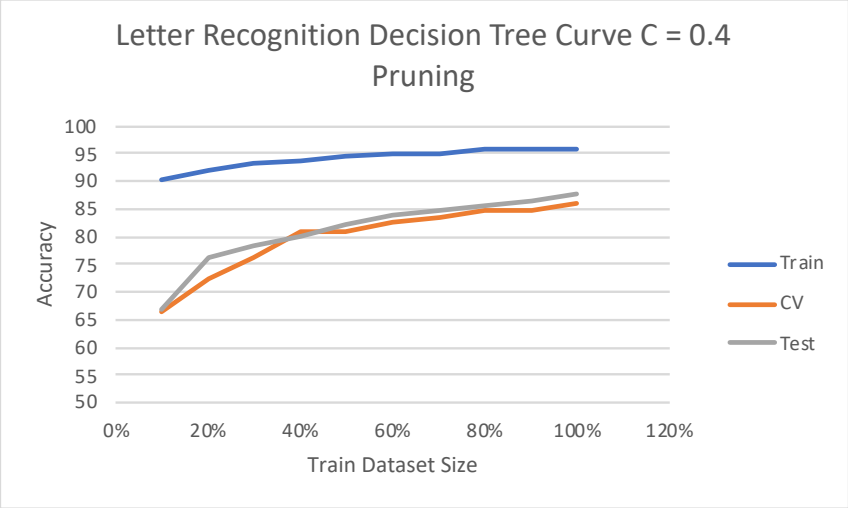
Census Income

Confidence Factor	Unpruned	Minimum Instance	Training Accuracy	CV-10 Accuracy	Testing Accuracy
NA	Yes	2	91.3865	83.8671	83.4457
0.1	No	2	86.9604	85.6674	85.2691
0.25	No	2	87.7495	85.6304	85.2028
0.4	No	2	89.3575	85.2596	84.8271
0.6	No	2	91.3965	83.8273	83.4015



Letter Recognition

Confidence Factor	Unpruned	Minimum Instance	Training Accuracy	CV-10 Accuracy	Testing Accuracy
NA	Yes	2	96.64	88.04	87.25
0.1	No	2	95.875	87.765	87.3667
0.25	No	2	96.345	87.915	87.5167
0.4	No	2	96.49	87.97	87.5167
0.6	No	2	96.645	87.11	87.3617



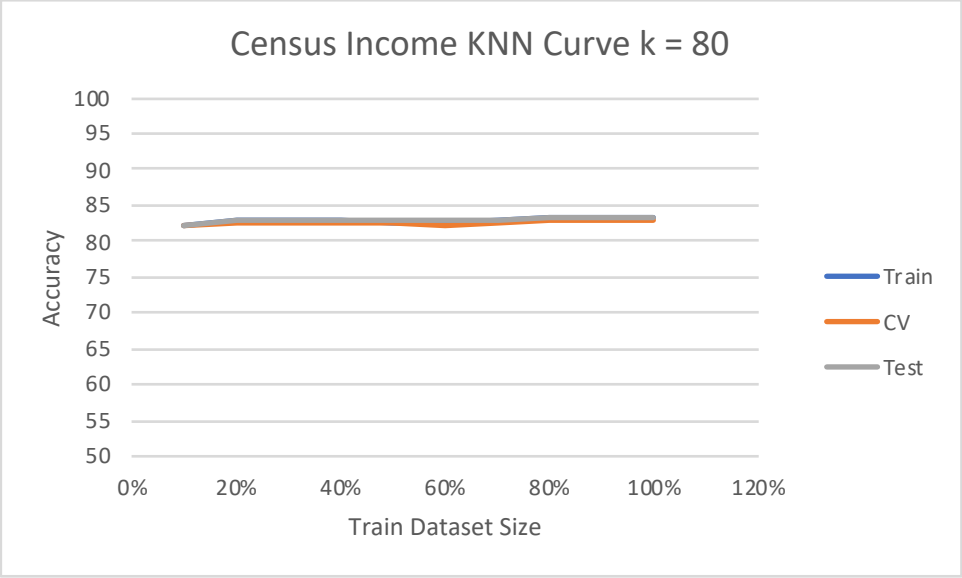
As we can see, for income data table, pruning decreases the accuracy, but the overall CV score is close. When confidence factor is 0.1, it achieves the best CV score. The testing results also indicate that C = 0.1 is the best model. As we can see from the letter reconginition, confidence factor 0.4 best fits the model. But the little difference regrading pruning oberserved in letter reconginition set implies that the dataset is not easily separable. In terms of running time, the training takes a lot time than testing, indicating it is an eager learner. The CV and Training convergers to high value indicating more training data helps achieves better performance.

K-Nearest Neighbors

For KNN, IBK learner is used. KNN, the number of neighbor is tuned.

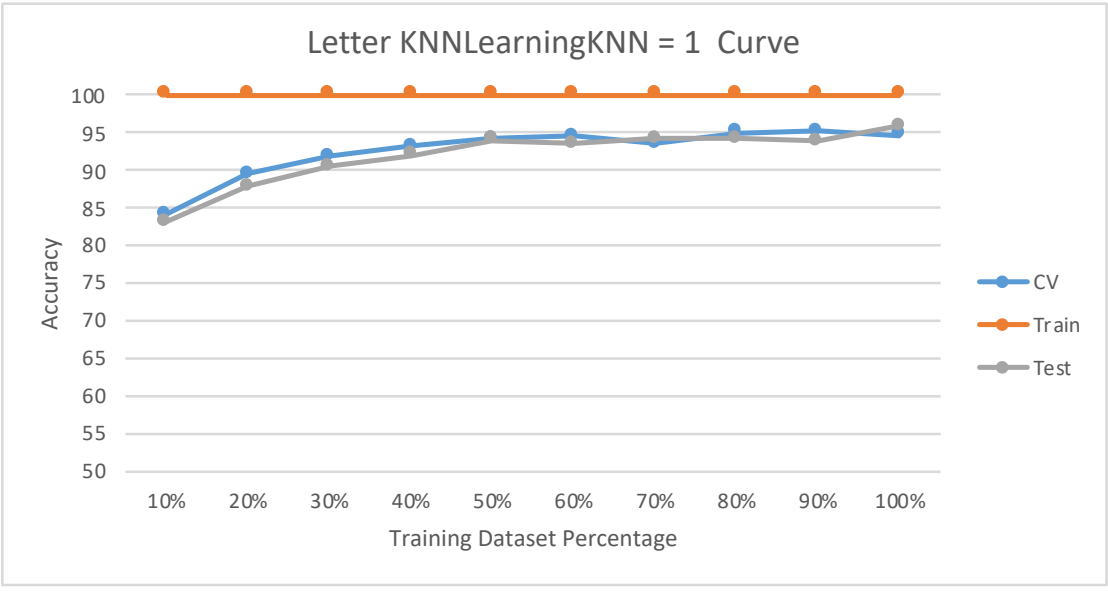
Census Income

KNN	Training Accuracy	CV-10 Accuracy	Testing Accuracy
1	99.9953	78.4304	78.4065
5	87.2259	81.8832	82.4953
10	85.2366	82.3474	82.7495
30	83.8535	82.7358	83.2026
80	83.4225	82.8684	83.2357



Letter Recognition

KNN	Training Accuracy	CV-10 Accuracy	Testing Accuracy
1	100	96.005	95.6167
5	97.77	95.49	94.8333
10	96.805	94.85	93.8667
30	94.105	91.97	90.55
80	88.39	86.075	84



As we can see from the parameter table for Census income, varying k does not make a big difference in CV when k is larger than 10. And when k = 80 works best for this dataset. The above observation indicates that for census income, the dataset has low repetition of

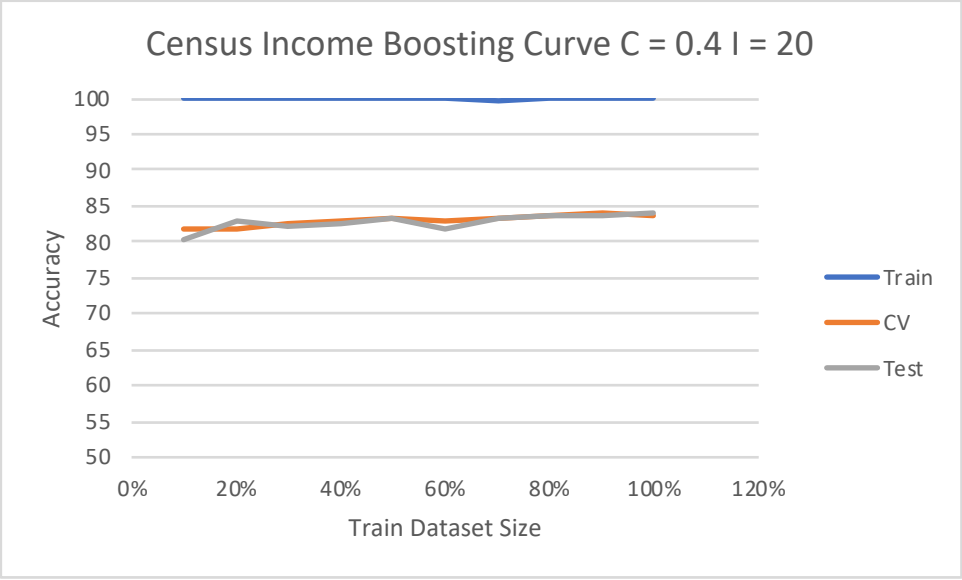
instances that have similar features. For letter recognition, poorer performance at higher K indicates that the dataset feature is more equally weighted and increasing K would significantly increases bias and decrease performance. KNN is a lazy learner and the test time is corresponding to the number of instances. The learning curves show that CV and training score converges to a high value, indicating this is a good model and more training dataset helps.

Boosting

Adaboost algorithm is used in boosting with decision tree as underlying learner, since decision tree is a weak learner and adaboost can significantly boost the decision tree performance through iterations. Decision tree confidence factor and boosting iteration number are tuned to compare model.

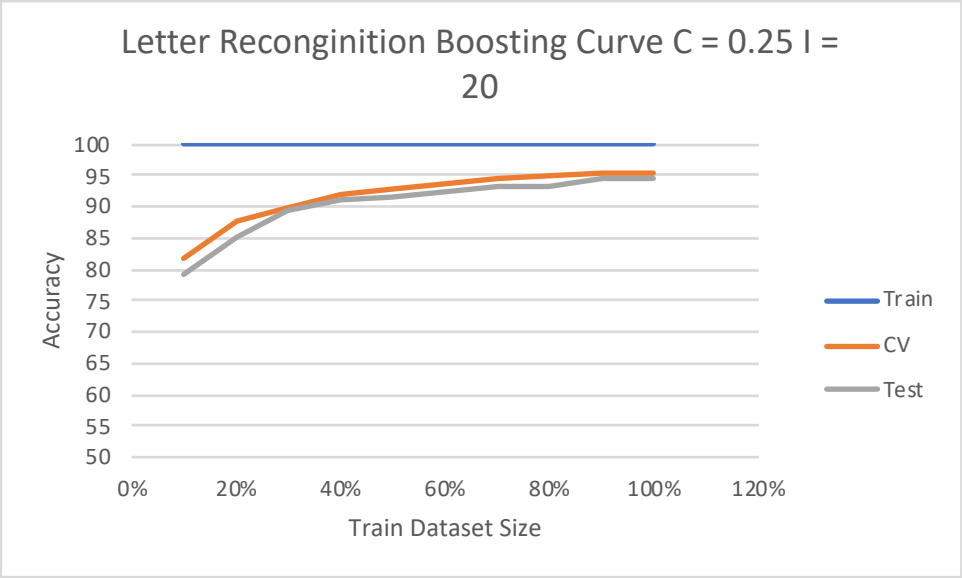
Census income

Leaner	Iteration	Training Accuracy	CV-10 Accuracy	Testing Accuracy
Decision Tree-C 0.25	10	99.9967	83.1211	83.07
Decision Tree-C 0.4	10	99.9901	83.2107	83.2026
Decision Tree-C 0.25	20	99.9967	83.7411	83.7109
Decision Tree-C 0.4	20	99.9967	83.8008	83.954



Letter Recognition

Leaner	Iteration	Training Accuracy	CV-10 Accuracy	Testing Accuracy
Decision Tree-C 0.25	10	100	95.73	94.8667
Decision Tree-C 0.4	10	100	95.35	95.0833
Decision Tree-C 0.25	20	100	96.545	95.9833
Decision Tree-C 0.4	20	100	96.35	95.8176



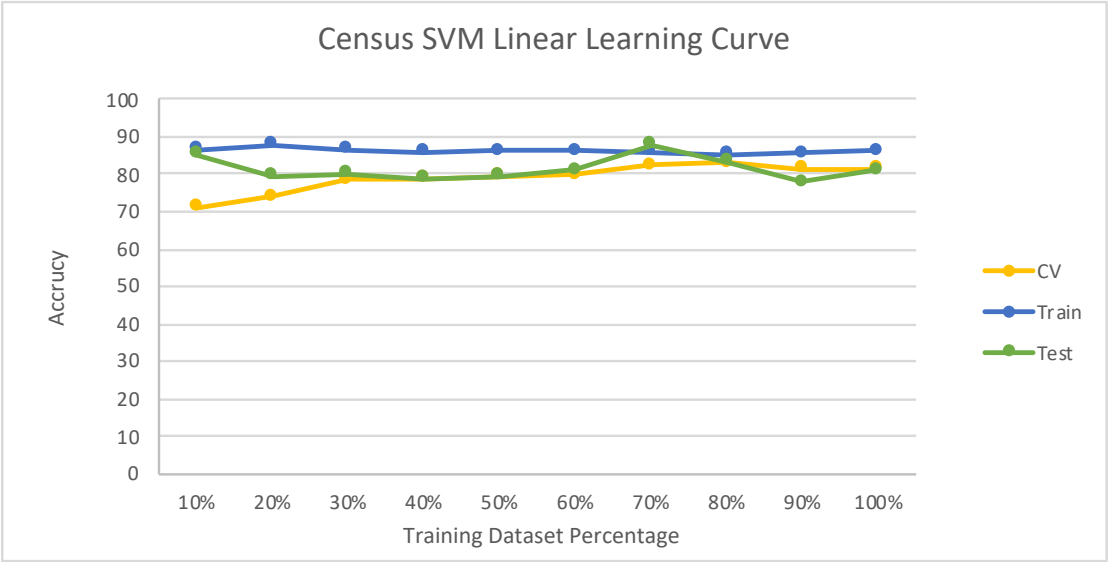
As we can see from the table, Adaboost with J48 decision tree improves the performance compared with only J48 decision tree in previous example. For the census income, different confidence factor and iteration only make little improvement. This could be due to possibility that dataset has a lot of noise instance since adaboost is prone to noise. Since the previous decision tree model already fits the data very well, using adaboost will not significantly improve performance on validation/test data and could risk overfitting the data. That is why cross validation is needed. Then training time with large number of iteration is very large. What's more, I found out that for letter recognition, when iteration grows very large, the accuracy will go down since overfitting.

Support Vector Machine

Support Vector Machine is an eager learner. I use SMO classifier in Weka. It tries to optimize the distance between two distances and hyperplane, which is decided by kernel. I tried different kernels: linear kernel(PolyKernel –E=1), polynomial with degree 3(-E=3) and radial basis function with gamma.

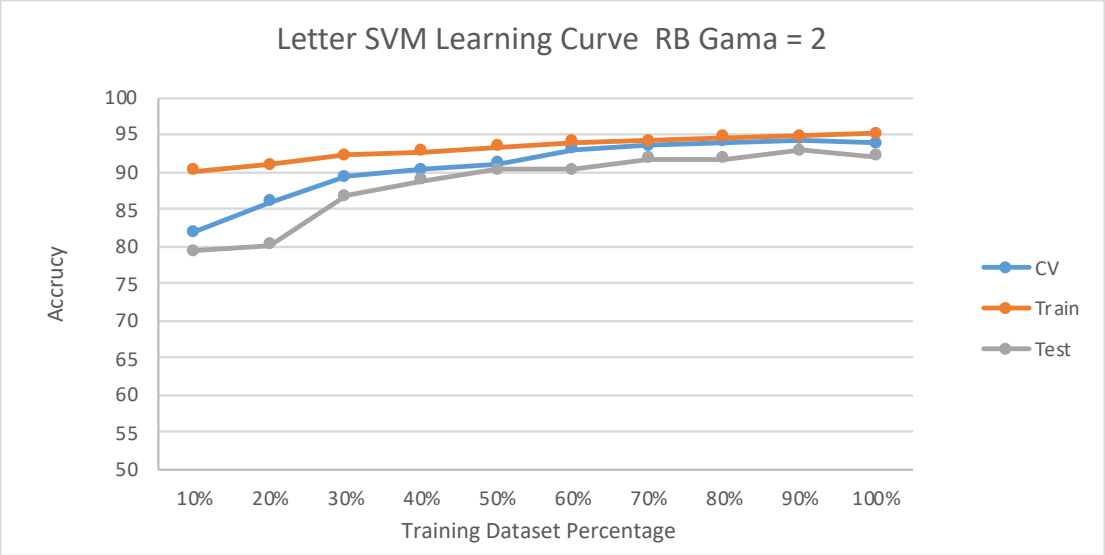
Census income

Kernel	Training Accuracy	CV-10 Accuracy	Testing Accuracy
PolyKernel –E = 1	85.6195	82.9646	83.0258
PolyKernel –E = 3	97.787	80.8118	69.5028
RBK garma = 1	93.031	81.3053	80.4428



Letter Recognition

Kernel	Training Accuracy	CV-10 Accuracy	Testing Accuracy
PolyKernel –E = 1	83.11	82.34	80.8833
PolyKernel –E = 3	95.155	93.295	92.1333
RBK garma = 2	95.155	93.83	92.1333



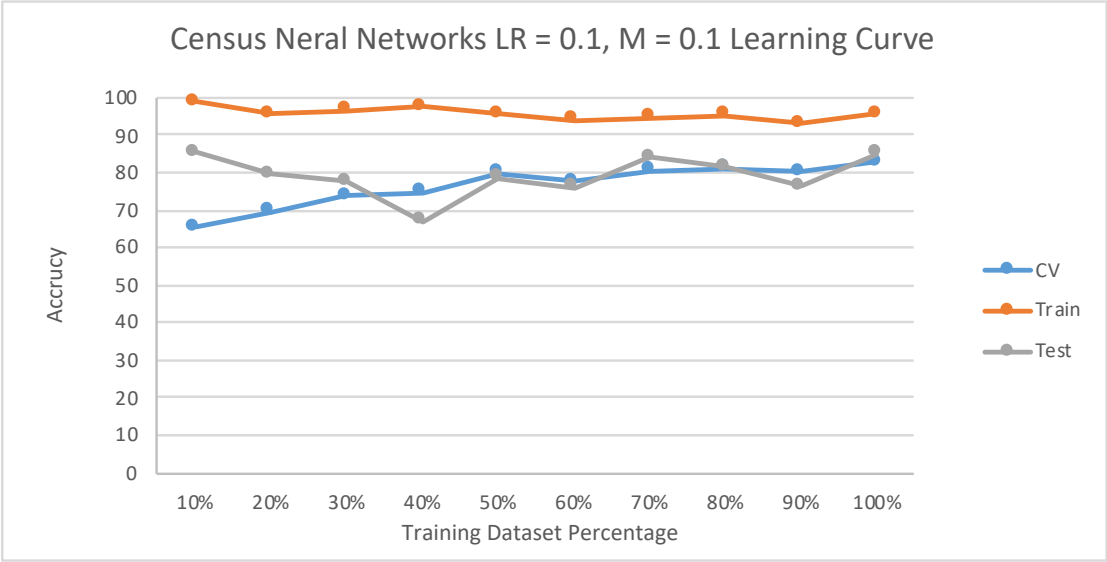
For Census income, Linear Kernel Function performs best. But there is no much different between 3 kernel functions. It's hard to say which kernel is best for census. But for Letters, RB perfroms best. And we can see for letters, polynomial kernel performs better than linear one. Both learning curves suggest that more training data helps achieving better performance. It is also noted that higher degree and bigger gamma for RB will increase training time and accuracy.

Neural Network

Multilayer Perception classifier in Weka is used to evaluate the datasets. The hidden layer is using the default setup. We tuned the learning rate and momentum values in experiment. Neural Networks is an eager learner, it uses back propagation to set the weight of different perception.

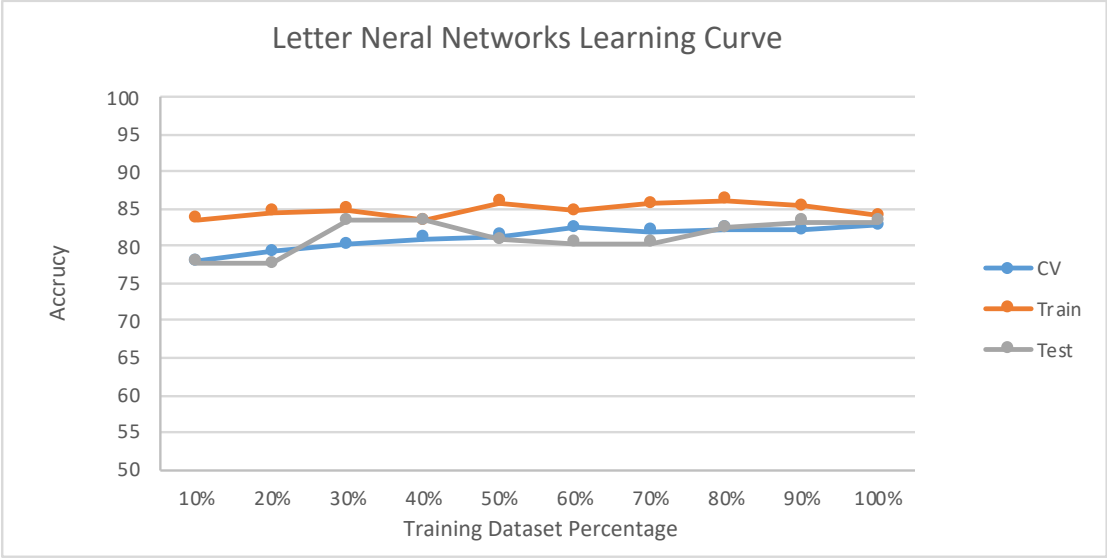
Census income

Layer	Training Accuracy	CV-10 Accuracy	Testing Accuracy
LR = 0.1, M = 0.1	95.5752	82.854	84.8708
LR = 0.3, M = 0.2	88.6062	81.5256	83.954
LR = 0.5, M = 0.5	83.6283	77.323	82.2878



Letter Recognition

Layer	Training Accuracy	CV-10 Accuracy	Testing Accuracy
20, LR = 0.1, M = 0.1	84.0946	82.76	82.12
20, LR = 0.3, M = 0.2	82.455	82.085	81.9
20, LR = 0.5, M = 0.5	81.85	80.368	78.631



As we can see from the table, changing learning rate and momentum value does not significantly improve performance. LR = 0.1 and momentum = 0.1 have best performance for both. I also tried to add some layers but it made no difference. It also risks overfitting. This is probably because the default model is well-fit the data.

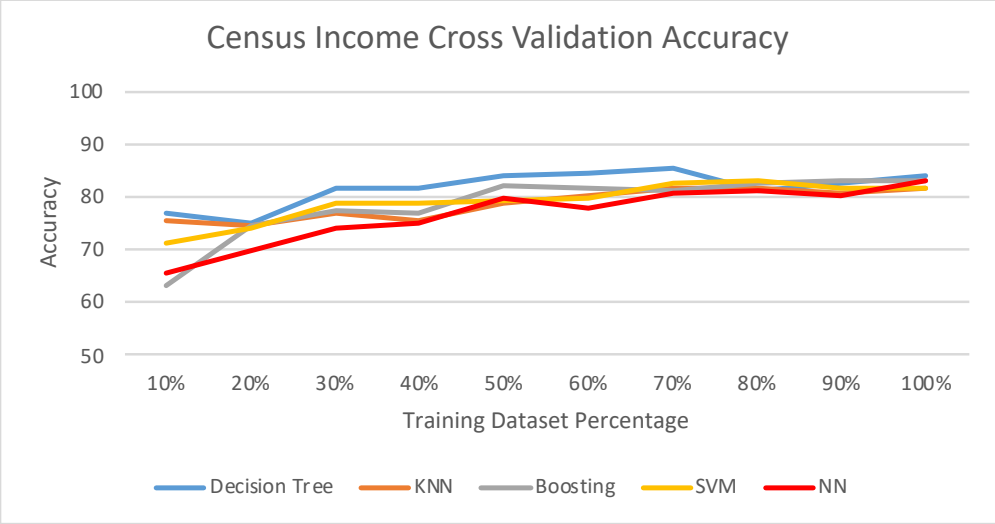
However, increasing layers will increase the running time and become very cost. From the learning curves, we can find out more data will help us get better performance.

Summary

Comparison of different classifiers:

Census Income

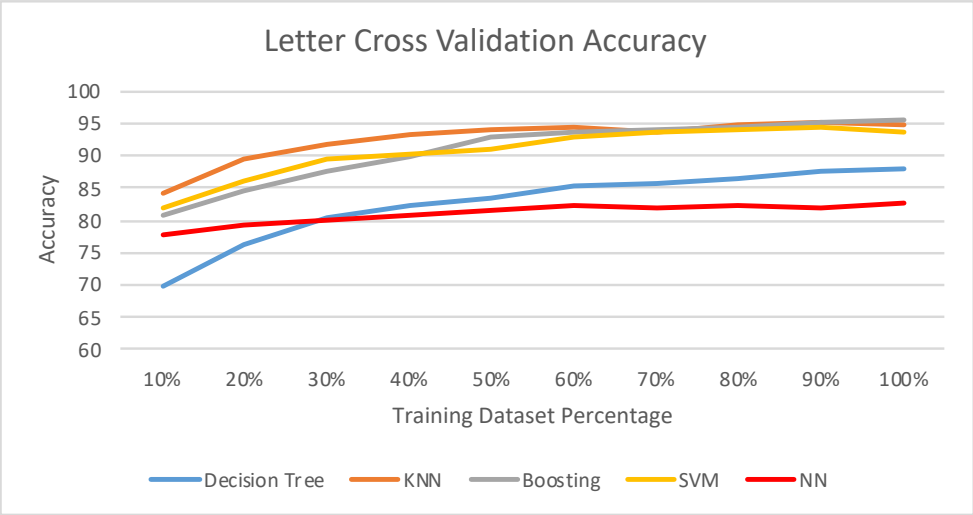
	Training Accuracy	CV-10 Accuracy	Testing Accuracy
Decision Tree	86.9604	83.8496	85.2691
KNN	81.4159	81.4159	80.8118
Boosting	99.134	82.9646	83.3948
SVM	86.3938	81.4159	81.1808
Neuron Network	95.5752	82.854	84.8708



Letter Recognition

	Training Accuracy	CV-10 Accuracy	Testing Accuracy
Decision Tree	96.49	87.97	87.5167
KNN	100	96.005	95.6167
Boosting	100	96.545	95.9833
SVM	95.155	93.83	92.1333
Neuron Network	84.0946	82.76	82.12

The Census incoming dataset has achieved best performance under Decision Tree and for letters, Boosting and KNN did a good job. However, NN worked worst suggesting that the model overfits the dataset.



From the cross validation curve, we know that SVM, Boosting and NN can learn very fast to converges to a high accuracy for census incoming. However, NN in Letters did not perform well. Actually, it is flatten showing that it did not learn more with increasing training dataset.

Based on Markam's table, the following table compares the 5 classifiers in this experiment. [3]

Algo	Results Interpretable	Average Accuracy	Training Speed	Prediction speed	Amount of parameter tuning
KNN	Yes	Lower	Fast(less than 1s)	Depends on K	Minimal
Decision Tree	Somework	Lower	Fast(less than 1s)	Fast	Some
SVM	A little	Higher	Slow(around 200s on average	Fast	Some
AdaBoost	A little	Higher	Slow(around 60s)	Fast	Some
Neural network	Hard	Higher	Slow (more than 400s on average)	Fast	Lots

Reference

1. Census Income dataset. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/Census+Income

2. Letter Recognition dataset. UCI Machine Learning Repository
   https://archive.ics.uci.edu/ml/datasets/Letter+Recognition
3. Kevin Markam. Comparing supervised learning algorithms
   https://www.dataschool.io/comparing-supervised-learning-algorithms/