Assignment 1 – Supervised Learning

Qianzhen Li

Abstract

This report analyzes the result of two different datasets trained by five learning algorithms including Decision Trees, Neural Networks, Decision Tree of boosted version, Support Vector Machines and k-Nearest Neighbors. In this report, effect of different parameters of learning algorithms is also analyzed. Finally, we compare and contrast the different algorithms to find the best learning algorithm for the two problems.

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, capital-loss and hours-per-week) and 8 categorical variables (workclass, education, marital-status, relationship, race, sex and native-country). As shown in Figure 2, more than 75% instances have dependent variable of >50k. Even if we predict the result to be >50k for all test cases, we could still have 75% accuracy. Therefore, this dataset is somewhat skewed and we will use both accuracy and f1 as scoring. Figure 3 shows number of instances with different labels in a specific categorical variable. Since this dataset has some missing values, we delete those instances having missing values. Moreover, since continuous variables are in different orders, we performed feature scaling.

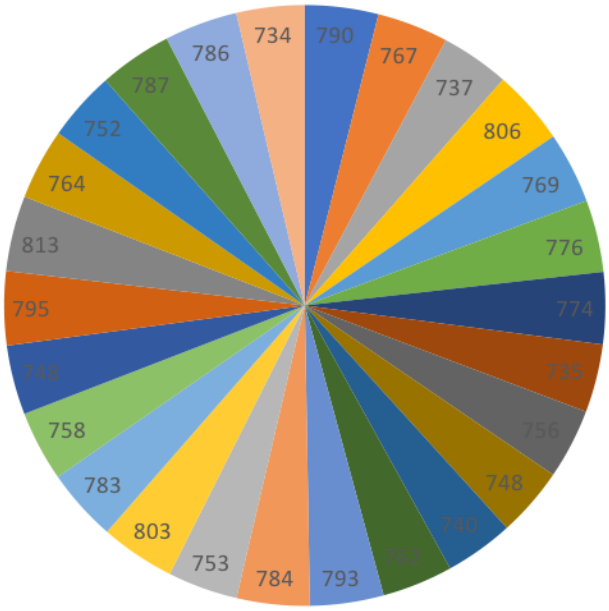


Fig. 1 Distribution of dependent variable (dataset 1)

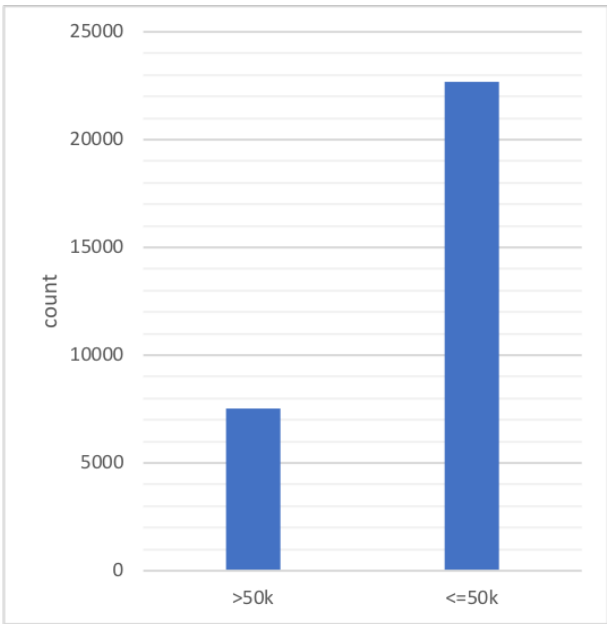
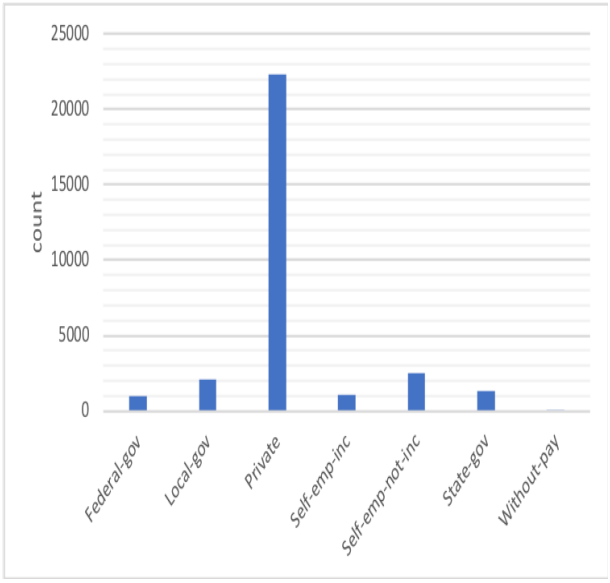


Fig. 2 Distribution of dependent variable (dataset 2)



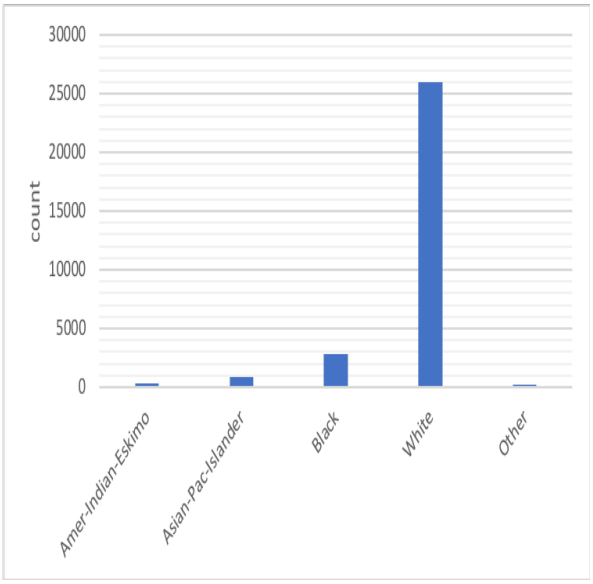


Fig. 3 Number of instances with different labels in work-class and race (dataset 2)

I think these two datasets are interesting because both of them have a large number of instances and lots of attributes. Therefore, these two datasets are not trivial and result of different learning algorithms will not be same. Additionally, I deliberately chose one of datasets to be binary-class and the other to be multi-class, and one to be skewed and not skewed.

Implementation and Analysis

All algorithms were imported from sklearn library and were run on Spider environments. 5-fold cross validation was utilized on training and test data to plot validation and learning curve, and data was split as 80% for training set and 20% for test set. For each learning algorithm, we will try to tune every parameter in this algorithm as possible as we can, in order to obtain the optimal parameter of this algorithm for both datasets. After we got the optimal parameter, we used the optimal parameters to the learning curve as the iteration (for iterative algorithm) and number of instance increase.

This analysis report is arranged as follow: First, we will discuss the implementation and result of two datasets for every algorithm. Then, we will compare and contrast the results of different algorithms in the end.

Decision Trees

We imported decision tree classifier from sklearn library and use the 'Gini' criteria to measure the quality of a split. In terms of pruning, we use pre-pruning to limit 'over-growth' of the decision tree, i.e., to avoid the overfitting of the decision tree. There are many parameters which can be adopted to perform pre-pruning, such max depth, min samples split and min samples leaf, etc. For these two datasets, we chose to use max depth and min sample leaf as parameters to perform pre-pruning. Max depth means the maximum depth the decision tree can grow and min sample leaf means this node cannot be split if the number of instance is less or equal than this value.

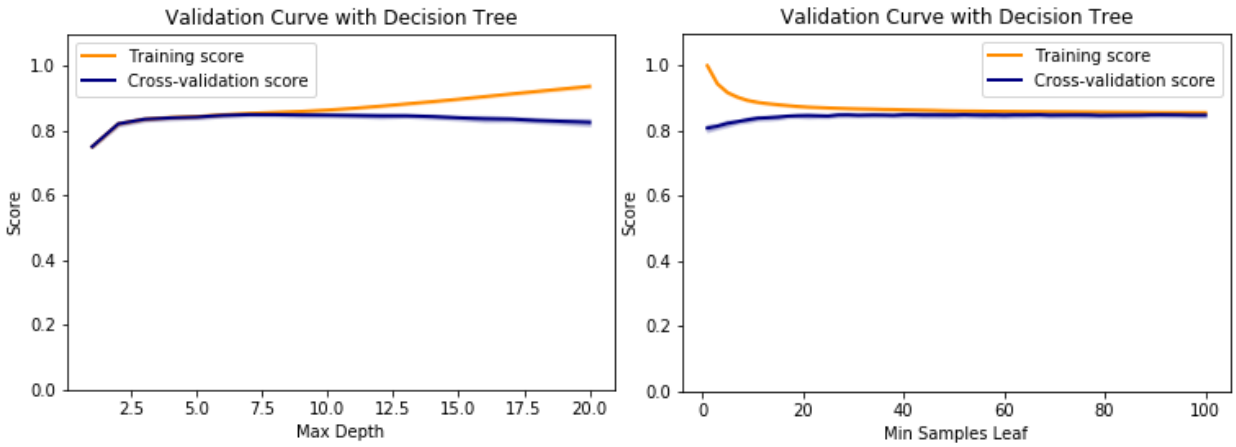


Fig. 4 Parameter tuning for dataset 1

As shown in Figure 4, for dataset 1 decision tree appear to be overfitting the training examples when max depth is larger than 8 or min samples leaf is less than 50. In order to obtain better test accuracy, parameters we chose for decision tree should be just slightly overfitting to avoid underfitting. What's more, since we use two parameters to perform pre-pruning together, we

chose the max depth to be slightly larger than and min sample leaf to be slightly smaller than the thresholds mentioned above. As a result, the max depth and min sample leaf for dataset 1 are both 10.

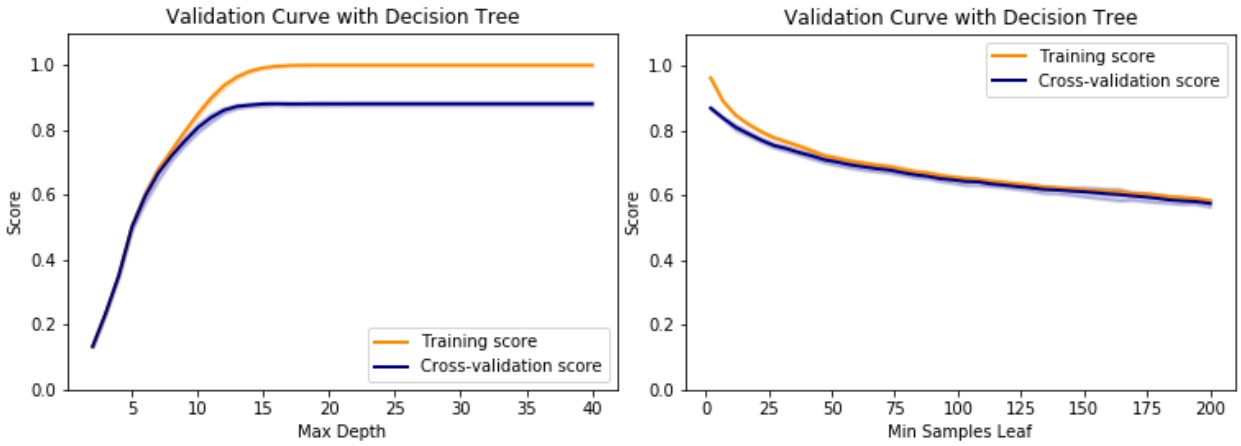


Fig. 5 Parameter tuning for dataset 2

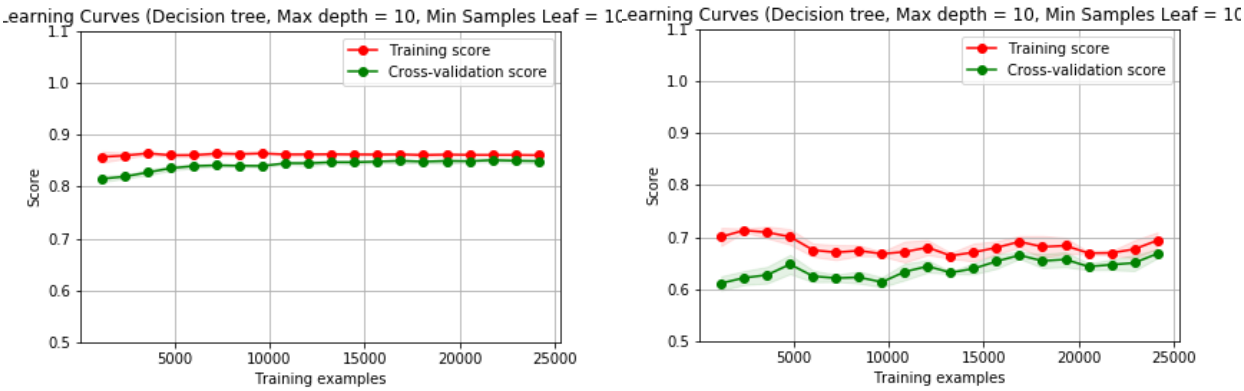


Fig. 6 Learning curve for dataset 1 (left: accuracy, right: f1)

While as shown in Figure 5, for dataset 2 decision tree appear to be overfitting when max depth is larger than 8. However, for min samples leaf, even though increase min sample can avoid overfitting, the accuracy of test set also drops. Since our ultimate goal is to obtain better performance for test set instead of avoiding overfitting, we chose to use max depth only to perform overfitting and set max depth to be 13.

As shown in Figure 6, as the number of training examples increase, the training score and cross-validation score converge together as we expected. It means that our algorithm works well for this dataset. As shown in Figure 7, even though cross-validation score come close to the training score as training examples increase, there is still gap between cross-validation and training score which indicates overfitting and adding more data can help us to improve the accuracy.

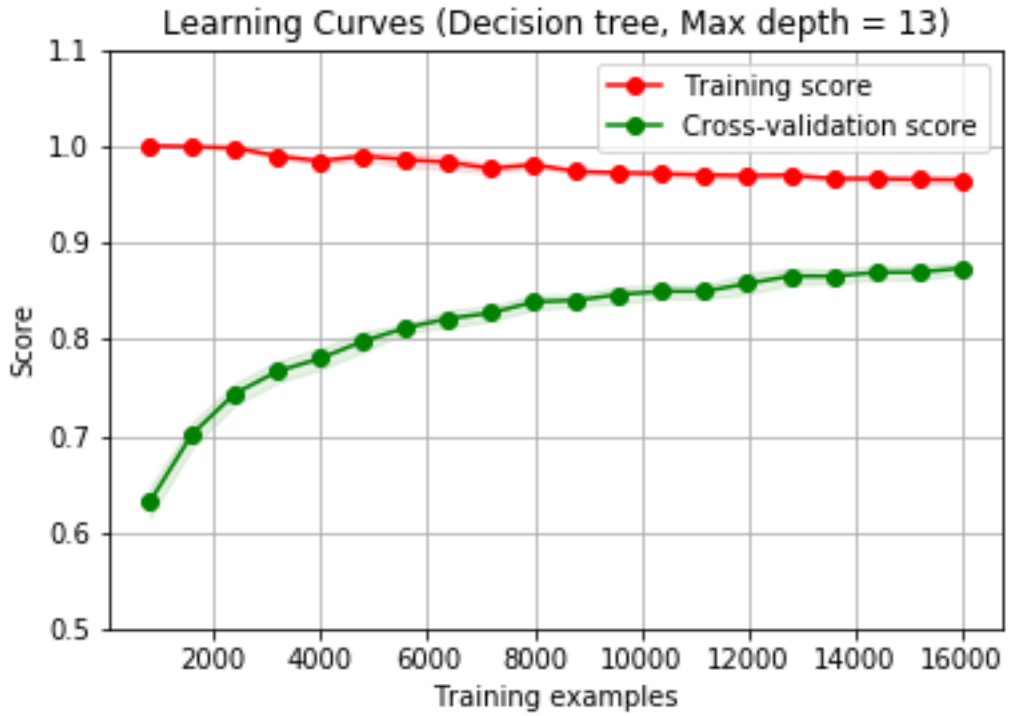


Fig. 7 Learning curve for dataset 2

Neural Networks

We imported Multi-layer Perceptron classifier from sklearn library and used Relu as activation function. Moreover, we used batch update when performing back-propagation to train our neural network. In order to improve our accuracy for test set, we chose to tune the hidden layer size of neural networks.

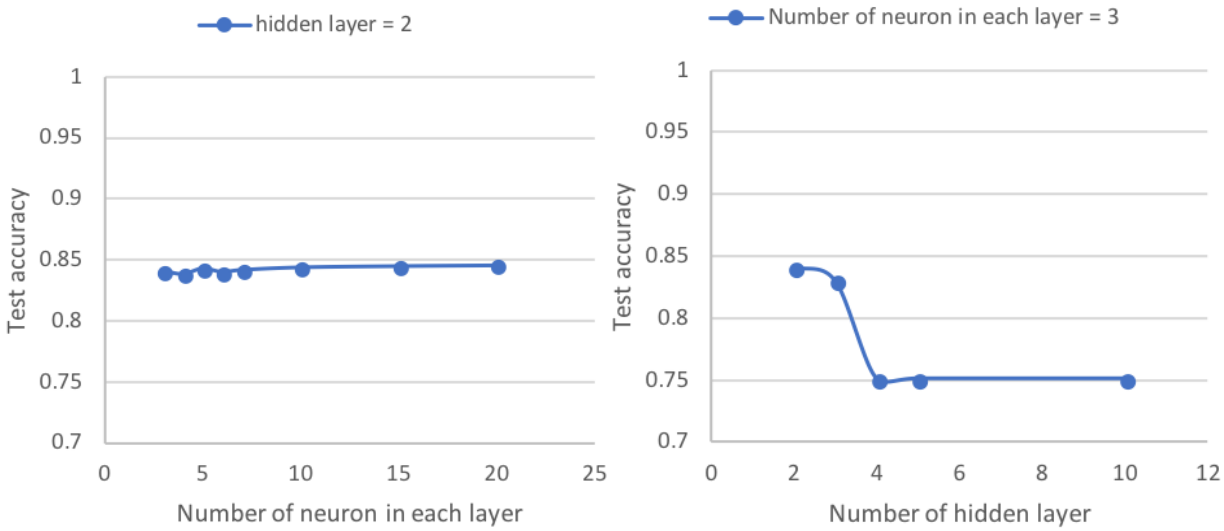


Figure 8 Effect of hidden layer and number of neuron in each layer (dataset 1)

As shown in Figure 8, for dataset 1, the increase of hidden layer and number of neuron in each layer does not help to increase test accuracy. Therefore, we set hidden layer size to be (3, 3) for

dataset 1. As shown in in Figure 9, for dataset 2, increase of number of neurons in each layer helps to significantly increase the test accuracy while increase of hidden layers does not. Therefore, we set hidden layer size to be (50, 50).

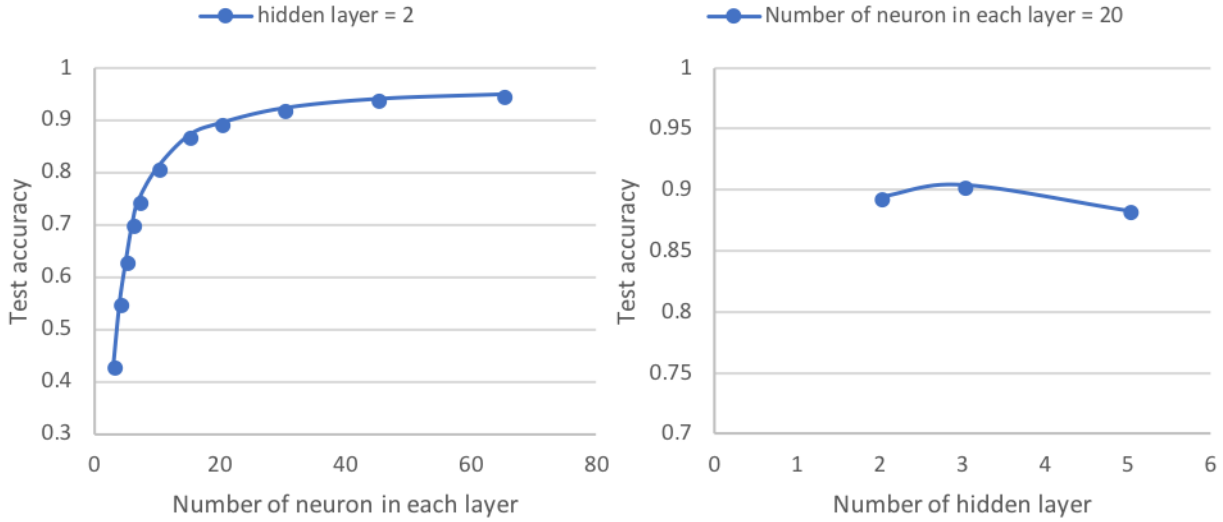


Fig. 9 Effect of hidden layer and number of neuron in each layer (dataset 2)

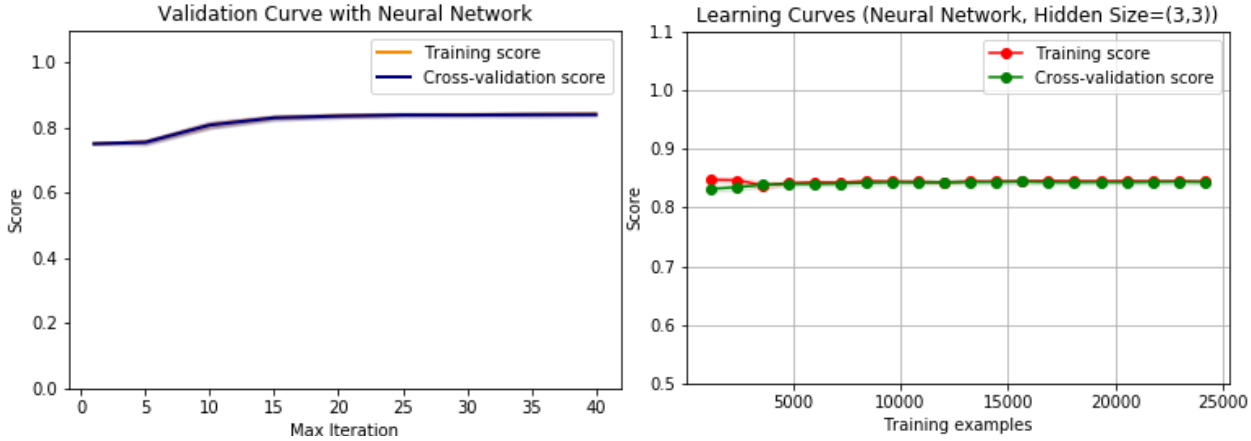


Fig. 10 Learning curve for dataset 1 (Hidden layer size = (3, 3))

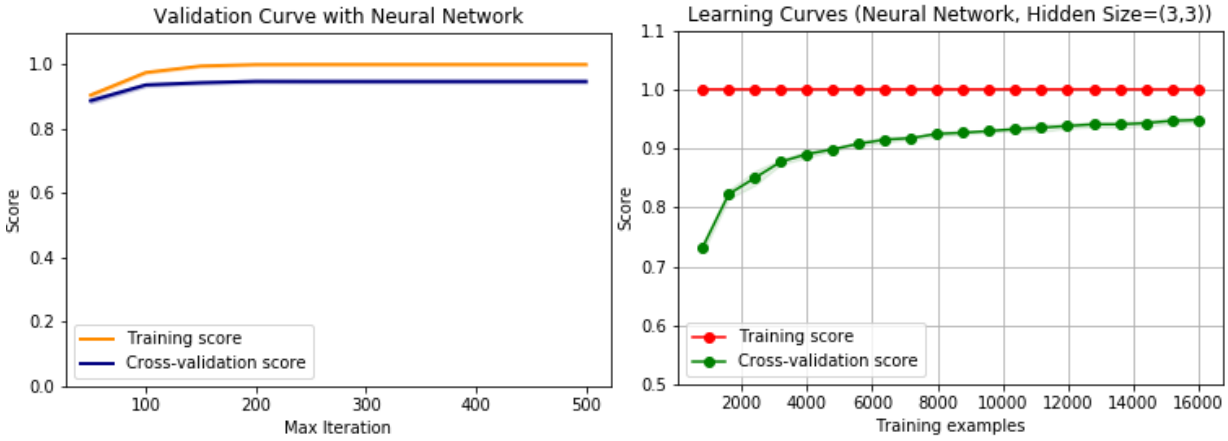


Fig. 11 Learning curve for dataset 1 (Hidden layer size = (50, 50))

For dataset 1, it is shown in Figure 10 that neural network converges when iteration is larger than 20 and we can obtain high test accuracy even though the training examples is small.

For dataset 2, it is shown in Figure 11 that our neural network is completely trained when iteration reaches to about 170. Moreover, the learning curve indicates high variance(overfitting) and adding more training examples moderates overfitting and increases test accuracy. It means that our algorithm generalizes well to new data and collecting more training examples can help our algorithm find better weight coefficients during the training process of our neural networks.

Boosting

Since we are required to implement a boosted version of your decision trees, we imported AdaBoostClassifier from sklearn library and use decision trees with some form of pruning as the base estimator. In order to keep the base estimator (decision tree) simple, we set the max depth of decision tree to be 3. What's more, in order to obtain better test accuracy, there is a parameter (number of estimator) to tuning.

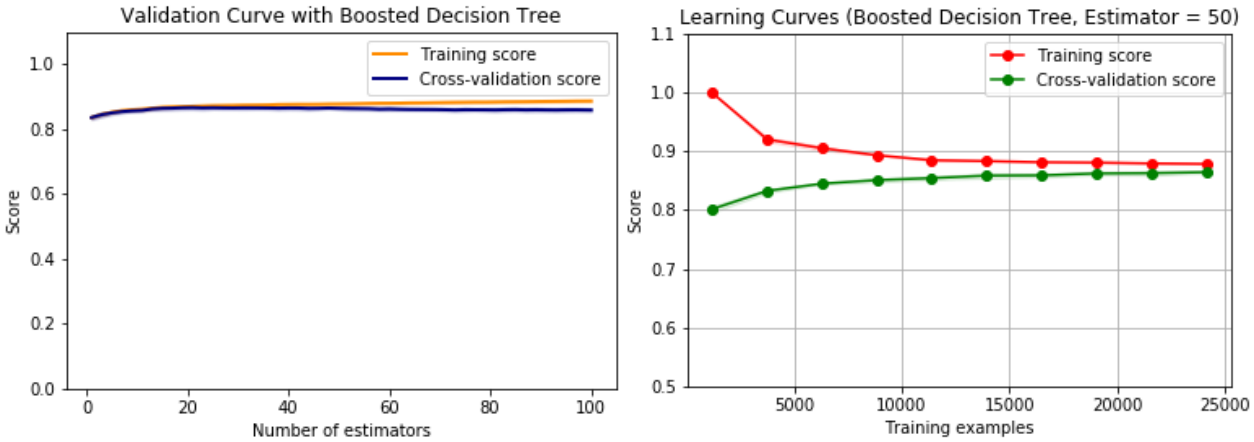


Fig. 11 Parameter tuning and learning curve for dataset 1

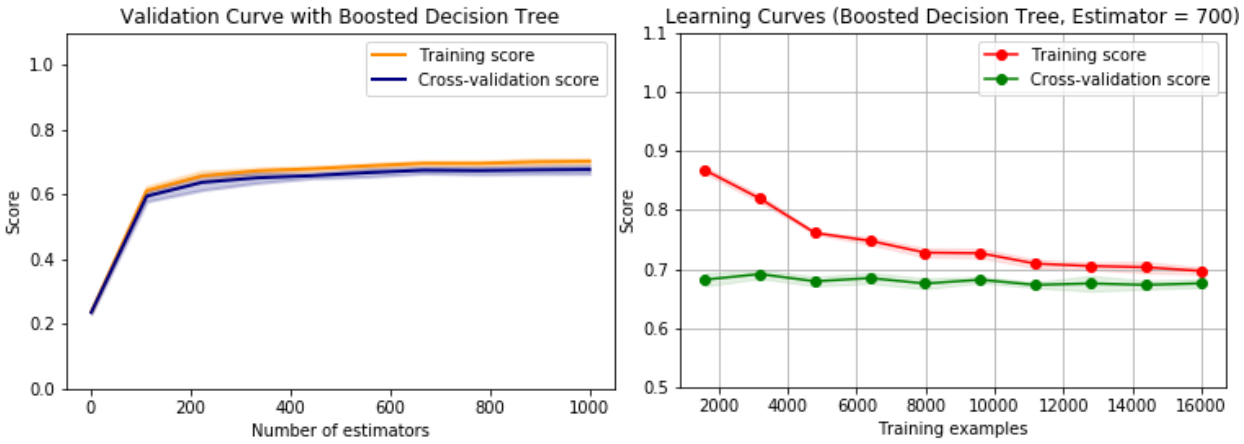


Fig. 12 Parameter tuning and learning curve for dataset 2

As shown in Figure 11, when the number of estimators reach to 40, overfitting start to occur. Therefore, we chose the number of estimator to be 50 which is slightly larger than 40. Moreover, the learning curve indicate that our algorithm generalizes well to new data.

As shown in Figure 12, when the number of estimators reach to 700, the test accuracy stops increasing. Therefore, we chose the number of estimator to be 700. However, as the number of training examples increase, the test accuracy remains unchanged while training score decreases, which means that we fail to capture some characteristics of the datasets. In this case, adding more training examples could not improve our algorithm.

Support Vector Machines

Since we are only required to implement linear SVM for this assignment, we imported LinearSVC from sklearn library. In order to improve test accuracy of both datasets and avoid overfitting, we need to tune penalty parameter C of the error term to regulate the linear SVM. The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example.

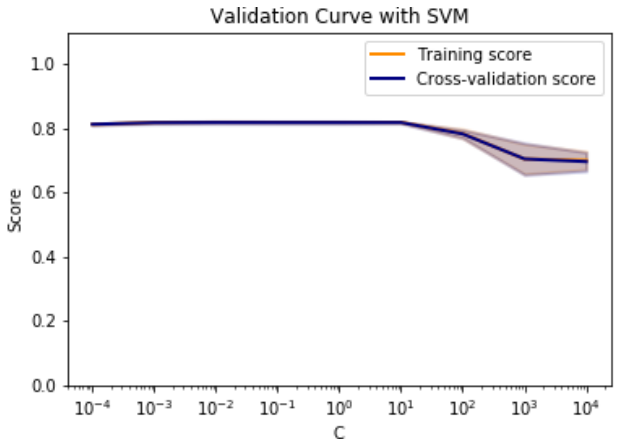


Fig. 13 Parameter tuning for dataset 1

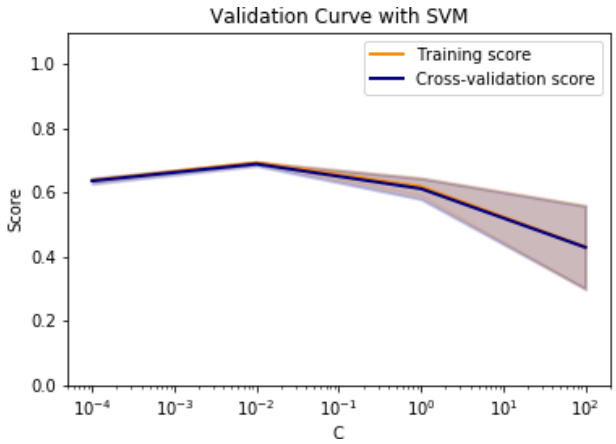


Fig. 14 Parameter tuning for dataset 2

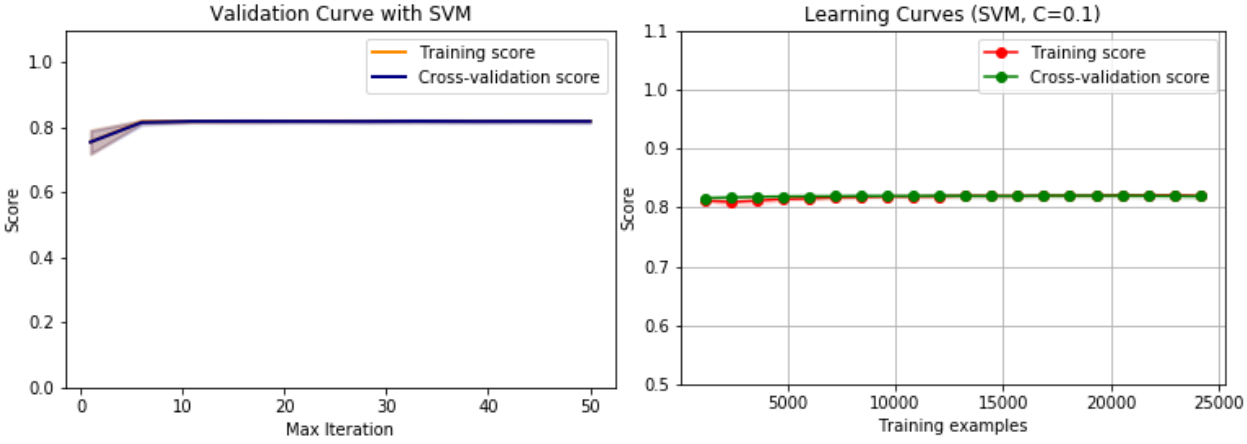


Fig. 15 Learning curve for dataset 1 (C = 0.1)

If the penalty parameter C is too large, the support vector machine will try to classify all training point correctly, even if leading to a smaller-margin hyperplane, in other words, overfitting. According to the result from Figure 13 and Figure 14, we set C to 0.1 and 0.01 for dataset 1 and dataset 2 separately.

In Figure 15, SVM converge quickly in several iteration. Moreover, the increase of training example does not affect the test score and training score, because we cannot capture some characteristics of datasets by only adopting linear kernel.

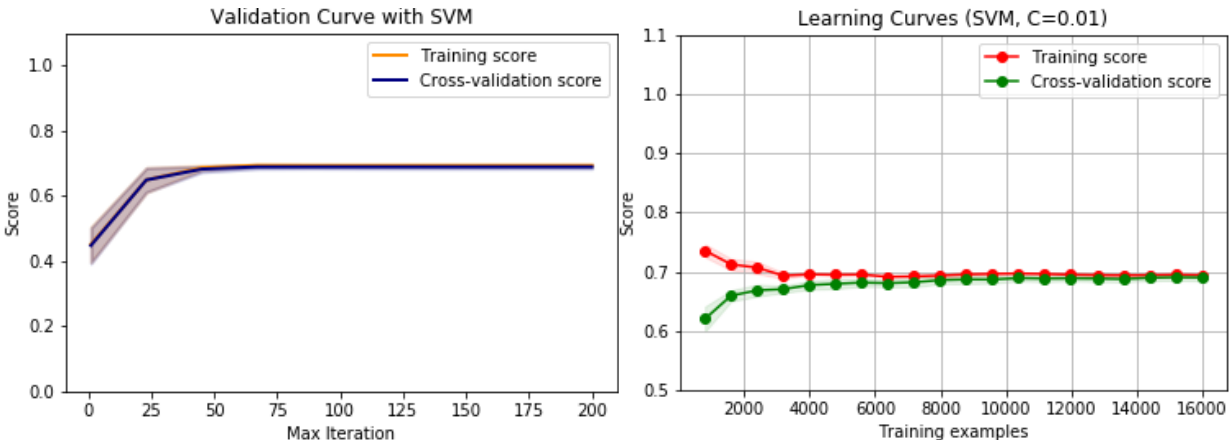


Fig. 16 Learning curve for dataset 2 (C = 0.01)

As shown in Figure 16, SVM converges within 60 iterations. Similarly, due to the linear kernel, the SVM does not perform well on the dataset 2.

k-Nearest Neighbors

We imported K-Neighbors Classifier from sklearn library. In order to improve test accuracy of both datasets and avoid overfitting, we chose to tune the parameter of number of neighbors.

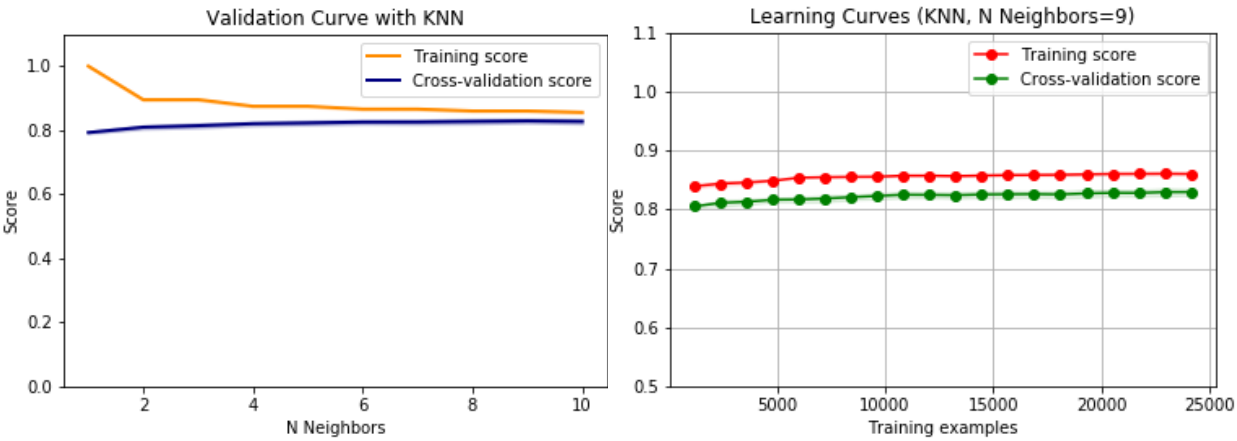


Fig. 17 Parameter tuning and learning curve for dataset 1

As shown in Figure 17, when number of neighbors is small, our algorithm appears to be overfitting for dataset 1. In order to avoid overfitting, we chose the number of neighbors to be 9. Moreover, adding more training examples does not improve the accuracy of our algorithm for this dataset.

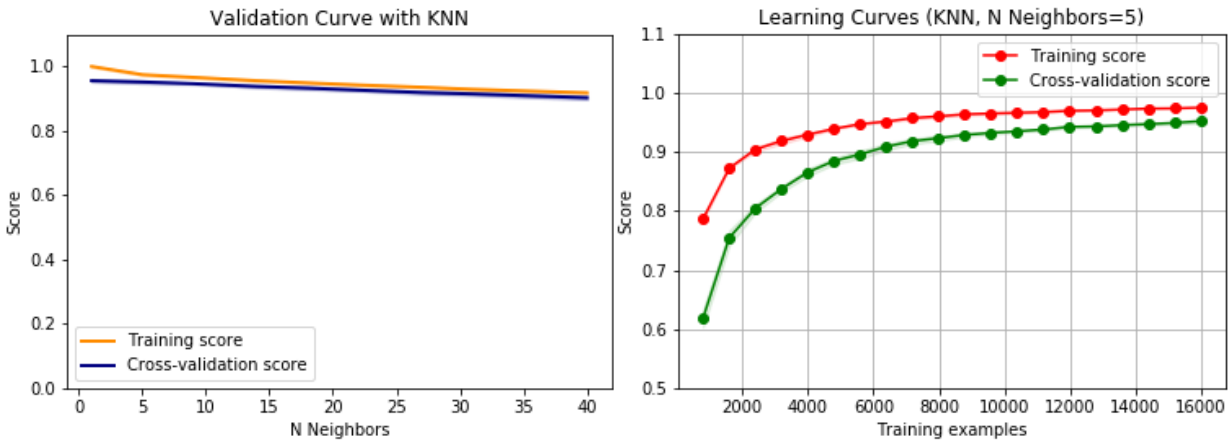


Fig. 18 Parameter tuning and learning curve for dataset 2

According to the Figure 18 (left), we chose the number of neighbors to be 5 for dataset 2. What's more, adding more training examples significantly increase the test accuracy of the KNN algorithm for dataset 2.

Comparison between different algorithms

For all five classification algorithms, we applied k-fold validation to both datasets and we set k to be 5. It can be seen from learning curves above that k-fold validation does not help. It is might because our dataset is large enough and we can directly split the instances into training set and test set in the beginning. K-fold validation will help a lot if the dataset is small. Moreover, the runtime of training of different learning algorithms can be seen from the Table 1.

Table 1 Runtime of different learning algorithms

Method	Dataset 1	Dataset 2
	Runtime (s)	Runtime (s)
Decision Trees	0.49	0.53
Neural Networks	5.45	35.15
Boosting	9.68	127.02
Support Vector Machines	4.11	19.92
k-Nearest Neighbors	65.09	14.62

In terms of the best learning algorithm for both datasets, we need to define what is the best. In instead of training score, we need to determine which algorithm is the best based on test accuracy as listed in Table 2. Moreover, since dataset 1 is somewhat skewed, we also compare

the f1 score for different algorithm for dataset 1. It can be seen from table 2 that boosted decision tree classifier perform best for dataset 1 in terms of both test accuracy and f1 score. For dataset 2, k-Nearest Neighbors perform best. It can be concluded that for dataset 1, all algorithms can predict well for new data even if it is a simple learning algorithm. While for dataset 2, some simple algorithms such as linear support vector machines fail to give a trustworthy predict.

Table 2 Performance of different learning algorithms

Method	Dataset 1			Dataset 2	
	Parameters	Accuracy(%)	F1 score	Parameters	Accuracy(%)
Decision Trees	Max_depth=10, Min_samples_leaf =10	0.85	0.67	Max_depth = 13	0.87
Neural Networks	Hidden layer = (3,3)	0.84	0.65	Hidden layer = (50, 50)	0.94
Boosting	N_estimator = 50	0.86	0.71	N_estimator = 700	0.68
Support Vector Machines	C = 0.1	0.82	0.54	C = 0.01	0.69
k-Nearest Neighbors	N_neighbors = 9	0.83	0.63	N_neighbors = 5	0.95