Analysis Report

In this assignment, I am analyzing two different classification problems. One is to classify the Titanic sinking survivors. And the other is multi-classification of letter recognition.

1. Titanic survival Classification

RMS Titanic was a passenger liner that sank in the North Atlantic Ocean in the early morning of April 15, 1912, on its maiden voyage. This sinking killed 1502 out of 2224 passengers and crew.

This problem is to classify either the passengers were survivors or not given the features provided in the titanic3.csv dataset. This analysis problem is interesting firstly because of the famous story of Titanic. I am very interested to know what sort of people would survive in this tragedy and what would happen to me if I was in Titanic. In addition, this analysis problem is also a subset of ongoing Kaggle competition, leading to many cool and interesting results. From the machine learning perspective, the dataset is not perfect, but with missing data in some features, and numerical and categorical data, requiring much data mining work. These make the problem very practical for machine learning application.

In this dataset, the features of 1309 passengers are listed in Table 1, the description are listed in https://www.kaggle.com/c/titanic/data.

```
RangeIndex: 1310 entries, 0 to 1309
Data columns (total 14 columns):
pclass       1309 non-null float64
survived     1309 non-null float64
name         1309 non-null object
sex          1309 non-null object
age          1046 non-null float64
sibsp        1309 non-null float64
parch        1309 non-null float64
ticket       1309 non-null object
fare         1308 non-null float64
cabin        295 non-null object
embarked     1307 non-null object
boat         486 non-null object
body         121 non-null float64
home.dest    745 non-null object
dtypes: float64(7), object(7)
```

Table 1, Titanic Data Info

```
Int64Index: 1043 entries, 0 to 1308
Data columns (total 8 columns):
pclass       1043 non-null float64
survived     1043 non-null float64
sex          1043 non-null object
age          1043 non-null float64
sibsp        1043 non-null float64
parch        1043 non-null float64
fare         1043 non-null float64
embarked     1043 non-null object
dtypes: float64(6), object(2)
```

Table 2, Cleaned Titanic Data Info

I dropped the “name”, “ticket”, “cabin”, “body”, “boat” and “home.dest” data because they are either missing too much data or irrelevant to “survived” data. The instances of missing “age”, “embarked” and “fare” are also dropped, just focusing on more on complete data (Table 2)

1.1 Visualization and analysis of Data

The rest features include class level, sex, age, siblings on board, parents on board, passenger fare and embarked port. Each feature is explored here to understand their correlations.

From the correlation table (Table 3), we can find that, “pclass” and “fare” are the relevant numeric features, which should be in the model selection of supervised learning. The correlation coefficient of “age” is very small, requiring being studied by group. Another interesting note here is that “pclass”, “age” and “fare” also have a large correlation coefficients between each other. I will deal with these later in different algorithms when requiring independent features. The scaling of features is also taken care later in algorithm part.

	pclass	survived	age	sibsp	parch	fare
pclass	1.000000	-0.317737	-0.409082	0.046333	0.016342	-0.564558
survived	-0.317737	1.000000	-0.057416	-0.011403	0.115436	0.247858
age	-0.409082	-0.057416	1.000000	-0.242345	-0.149311	0.177205
sibsp	0.046333	-0.011403	-0.242345	1.000000	0.373960	0.142131
parch	0.016342	0.115436	-0.149311	0.373960	1.000000	0.217650
fare	-0.564558	0.247858	0.177205	0.142131	0.217650	1.000000

Table 3, Correlation Table

Figure 1 shows the distribution of "age" feature. This feature is separated into 8 groups (Table 4) and the survival rate in each group is listed. Because of the "women and children first" code in life-threatening situations, more attention was on the children age groups. People who were at the age from 10 to 60 do not have any particular difference in survival rate. People who were below 10 years old had a higher survival rate. People, who were above 60 years old, were less likely to survive. The interesting groups of this feature are below 10, 10-60, and above 60.

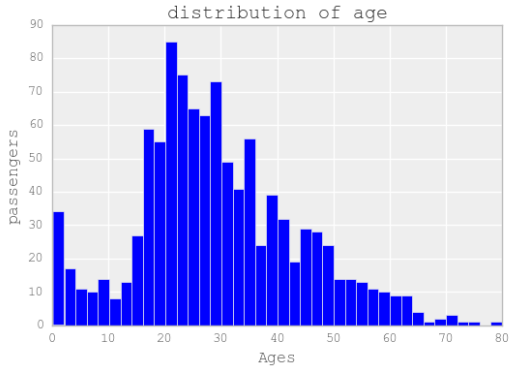

Figure 1, Distribution of Ages

Group	Survival rate
Children(0-5 years)	0.65
Children2 (5-10 years)	0.55
Teenagers (10-20 years)	0.39
Adults1 (20-30 years)	0.37
Adults2(30-40 years)	0.42
Adults3(40-50 years)	0.39
Adults4(50-60 years)	0.46
Adults5(> 60 years)	0.29

Table 4, Survival rate by different age groups

Another two important features are the "sex" and "embarked". As in Figure 2, females were much more likely to be survivors than males. In figure 2 and 3, it is clear that people who embarked from Q port were less likely to survive than other two ports, though the amount of people from Q port is the least. But if the passengers, who embarked from Q port, were in the 1st class, they also have a high likelyhood to survive.

Based on the above analysis, "pclass", "sex", "embarked", "fare", and "age" are the important features in this classification. The categorical data is converted to numerical data in the algorithms analysis part.

Fig 2, Distribution of Survived class according to embarked port and age

1.2 Machine Learning Algorithm Implementation

In the cleaned dataset, there are 1043 instances. So the training data (834 instances) and test data (209 instances) are split with 0.8 and 0.2 ratios. In the training set, the cross validation set is 167 instances while the training set has 667 instances.

1.2.1 Decision Tree Model

The Scikit learn DecisionTree classifier python package is used in this implementation. A decision tree without pruning is developed first to have general look of the classification. The average training accuracy is 0.94 while the cross validation accuracy is only 0.77. The model has an overfitting problem as in Fig 3.

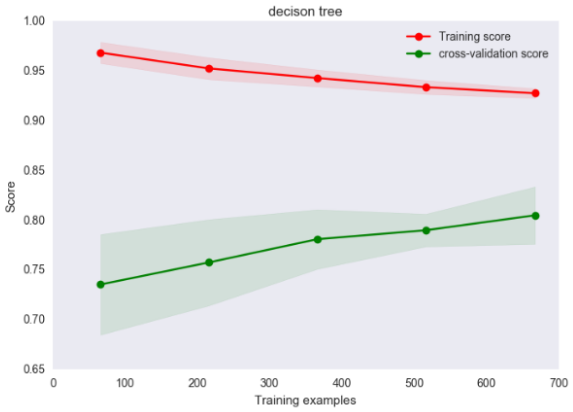

Figure 3(a), overfitting of decision tree

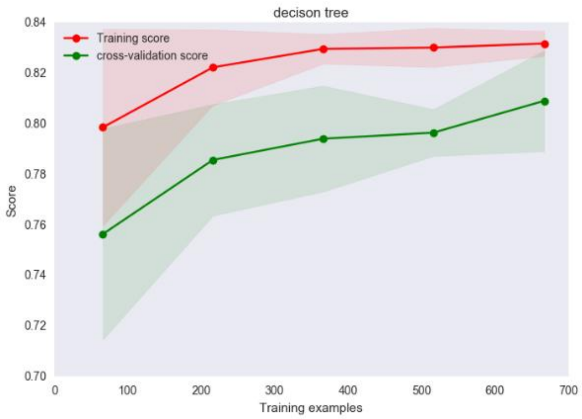

Figure 3(b), learning curve after pruning

The decision tree complexity is $O(n_{features} n_{samples} \log(n_{samples}))$ in the sklearn library.

- Pre-pruning of the decision tree

Scikit learn decision tree classifier provides several parameters for pre-pruning, which is to stop the growing of the tree. In the classifier, the parameters I tuned to prune the decision tree is min_sample_leaf and max_depth, which are the minimum leaves a node at least needs to have if this node is to be split, and the max depth of the tree. The criterion I used is "entropy", which is computed by the information gain used in the ID3 algorithm.

I used the minimum samples per leaf as 10, and the maximum depth as 4. So the training accuracy is 0.83, the cross validation error is 0.81 (Figure 3(b)). The pre-pruning improved 4% of accuracy in cross validation set. After applying this to the test set, the accuracy is 0.79, the evaluation matrix for the test set is:

	precision	recall	f1-score	support
0.0	0.77	0.92	0.84	125
1.0	0.83	0.60	0.69	84
avg / total	0.80	0.79	0.78	209

Testing evaluation matrix

	precision	recall	f1-score	support
0.0	0.82	0.92	0.87	98
1.0	0.86	0.71	0.78	69
avg / total	0.84	0.83	0.83	167

Cross validation evaluation matrix

1.2.2 Neural Network Model

Before implementing neural network algorithm, it is important to scale the “fare” data and group the “age” data.

The sklearn MLP classifier is used in this analysis. The activation method for the hidden layer is logistic, which is the sigmoid function. The optimization method is selected to be stochastic gradient descent. The hidden layer has 100 nodes. After several tuning, the learning rate is set constantly to be 0.02 and the alpha for regularization is 0.0001 at first (Figure 4).

From Figure 4, one can conclude that at the beginning of this training, because of the small number of iterations, the average training accuracy (red line) is much higher than the average cross validation accuracy. As more and more iterations in training, both the training accuracy and cross validation accuracy are at the same level. The test accuracy in this method is 0.77. Since this algorithm is an online training approach, the saw tooth shape learning curve is ok.

In Figure 5, the ‘lbfgs’ optimization, which is a quasi-Newton method, is chosen. In the figure 5 (a), the nodes in the hidden layer is selected as 500, which leading to overfitting problem. As the training accuracy is very high, the cross validation accuracy is much lower. And the cross validation accuracy drops down with the increase of the training accuracy.

The overfitting issue is solved by decreasing the number of nodes in the hidden layer and increasing the regularization parameter, alpha, from 0.0001 to 0.001. The average training accuracy is 0.83, the average cross validation accuracy is 0.81 and the test accuracy is 0.79.

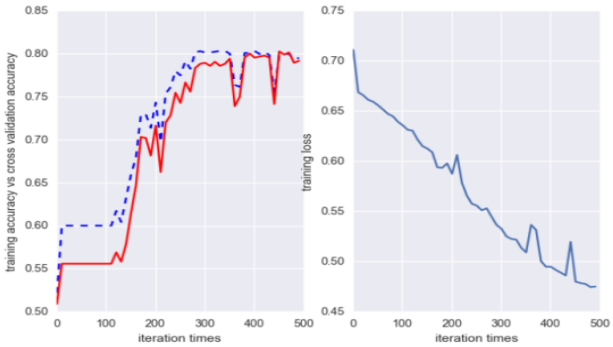

Figure 4, stochastic gradient descent optimization, and training loss

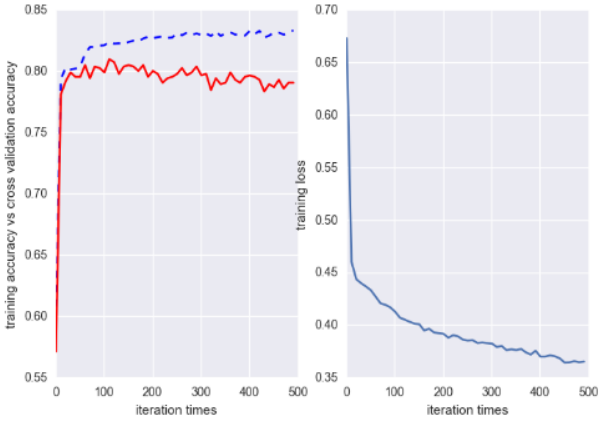

Figure 5 (a), “lbfgs” optimization: overfitting

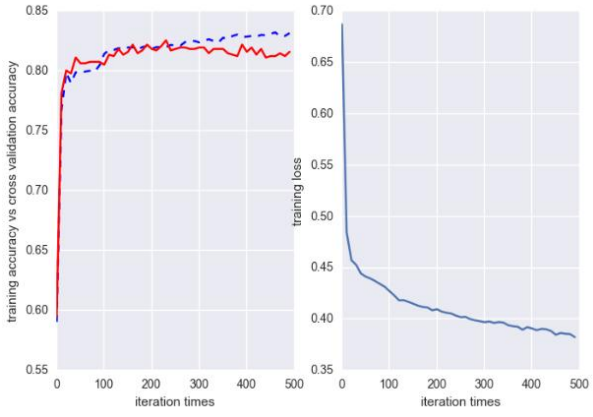

Figure 5 (b). “lbfgs” optimization

The backpropagation update of weights is used in the algorithm. The complexity of the application is $(n_{samples} \cdot m_{features} \cdot h_{neurons}^{k} \cdot o_{outneurons} \cdot i_{iteration})$, where h is number of neurons in hidden layers, k is the number of layers (1 in this case. This algorithm would take much time if the dataset is large and the amount of neurons in hidden layer is large.

1.2.3 Support Vector Machine Model

The libSVM library is used in this SVM model. The two kernels are linear kernel and Gaussian kernel. Before implementing the algorithm, the features in x have to be set into same scale because the SVM algorithms are not scale invariant.

The complexity of SVM algorithm based on the libSVM implementation scales between $O\left(n_{features} \times (n_{samples})^2\right)$ to $O\left(n_{features} \times (n_{samples})^3\right)$. So in this case, when $n_{features}$ and $n_{samples}$ are small, this algorithm should fit very well.

1.2.3.1 RBF-Gaussian Kernel

In the Gaussian Kernel method, the key parameters are C and gamma. C is the penalty parameter of the error term. C is like the inverse of the regularization parameter in the Neural network model. If C is very large, the model would lead to high variance; if C is very small, the model might make high bias. Gamma defines how far the influence of a single training example reaches. Gamma is proportional to the inverse of radius of the area of support vector influence. So if gamma is small, the model might be constrained, resulting in high bias. If gamma is large, the model would only includes the support vector itself, resulting in overfitting.

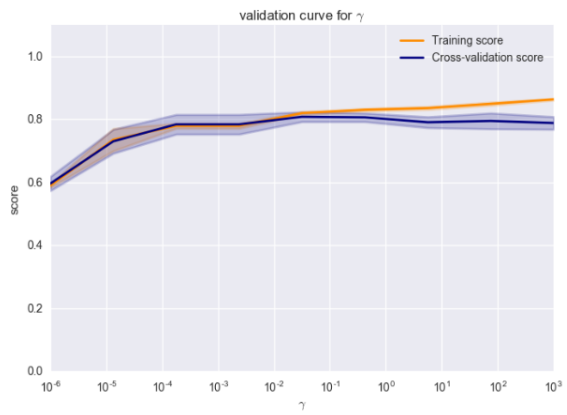

Figure 6 (a), Validation curve for gamma

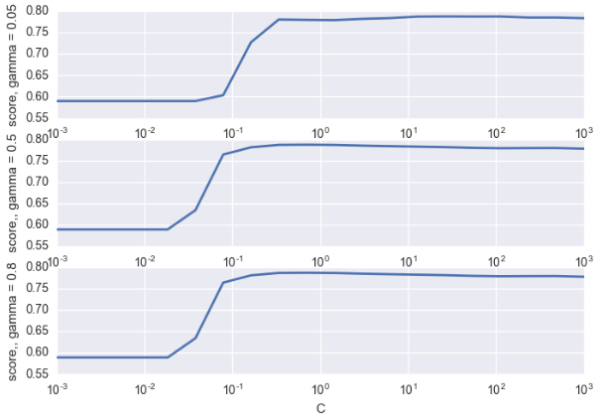

Figure 6 (b), Scaling of C

From the above figure, C can be chosen as 0.5, at which the cross validation data has the highest score. The gamma can be selected as 0.6. The learning curve is shown in Fig 7. The training accuracy is 0.83 and the cross validation accuracy is 0.81. The test accuracy is 0.79.

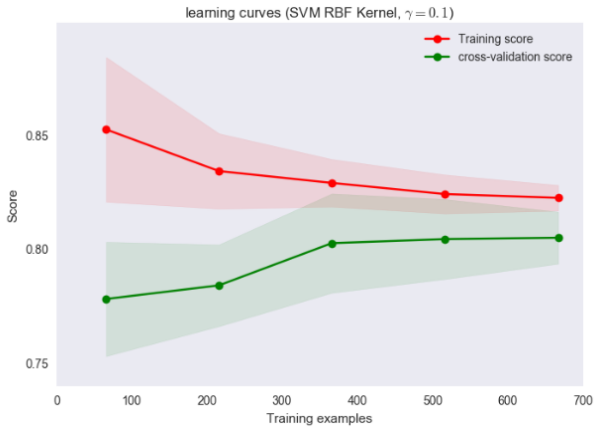

Figure 7, learning curve

In the kernel parameter, the linear kernel function is also tested in the codes. The testing accuracy is 0.77.

1.2.4 Boosting Decision Tree

For this model, the max depth of the decision tree for the Adaboost algorithm is the key parameter to tune. When the maximum depth is set 3 as in the early decision tree part, this model has overfitting issue. The training score is much larger than the cross validation score. Therefore, maximum depth is set as 2 for more aggressive pruning (Figure 8).

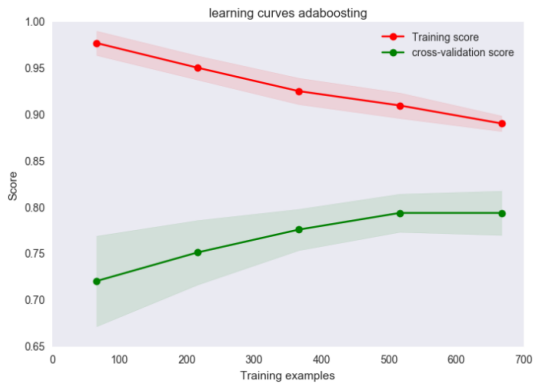

Figure 8 (a), overfitting issue in Adaboost

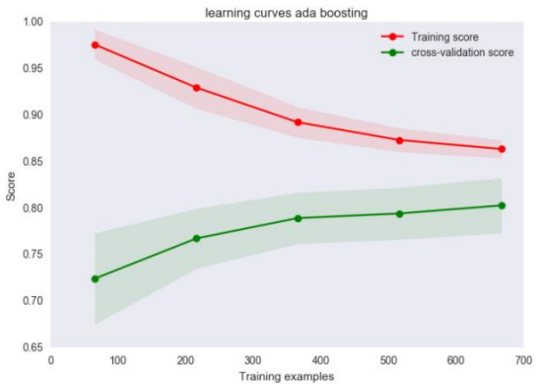

Figure 8 (b), Correct Adaboost

Based on the model developed by the above parameters, I applied it to the test set (Figure 9). The Training error is 0.15 and the test error is 0.17. This is a good classification for this dataset.

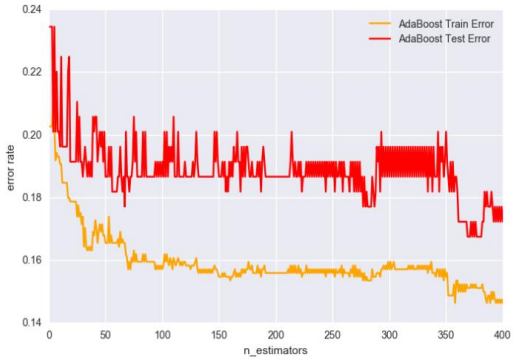

Figure 9, Test error and Train error

1.2.5 K Nearest Neighbors Model

Because this dataset is small, the brute force algorithm is implemented in this model. The tree based algorithms would be used in next problem. The complexity of brute force algorithm is $O(D_{dimensions} \cdot n_{samples})$.

Because each feature is equally weighted in the KNN algorithm, the scales and correlation coefficients of the each feature have to be taken more care of. In the encoded x-features, the correlation coefficients with y are as follows:

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
0.54	-0.54	0.21	0.07	-0.16	-0.33	-0.11	0.24

X1 and X2 represent female and male respectively. X3, X4 and X5 represent different embarked ports. X6 represents the class level of passengers. X7 is the age feature. X8 is the fare feature. The age feature might be important in the decision tree algorithm, but not very useful here. Therefore, I drop out the X4 and X7 features. Since the gender feature is more important than any other features, I would still keep them both in the KNN model, which is just like to double the weights of this age feature.

The validation of K (Fig 10) shows that when K is around 180, the cross validation could reach 0.805 accuracy. After applying this parameter and model to the test set, the accuracy is 0.78.

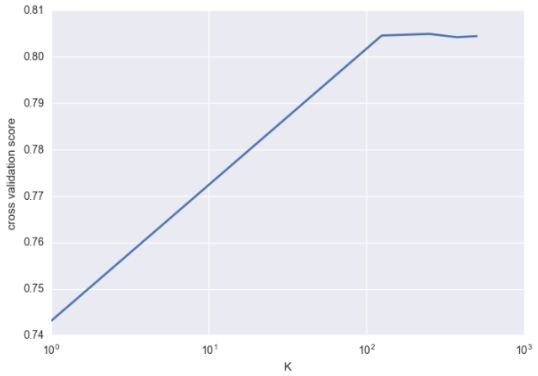

Figure 10, Validation curve for K

1.2.6 Summary

Model	Test Accuracy	Key Parameters
Decision Tree	0.79	Max_depth=4, min_samples_leaf=10
Neural Network	0.79	Hidden layers=(100,), solver='lbfgs', alpha=0.001
SVM	0.79	C = 0.5, gamma = 0.5
AdaBoost Decision Tree	0.79	Max_depth=2, algorithm="SAMME"
K nearest neighbors	0.78	K = 180

From the above results, all the algorithms have very close results. The result can be improved if given more features and having more data. Neural network and SVM are more complex than others based on the parameters. Because this dataset is kind of noisy, small with just a few parameters, neural network and SVM might be better than others.

2. A-Z Letter Recognition

This is dataset is from the UCI dataset. The objective is to identify the black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. From the

application perspective, letter recognition is widely applicable every day. It is interesting to develop something with practical usage. From the machine learning point of view, this is a multi-classification problem. I am very interested in seeing the differences with the previous binary classification.

This dataset has 16 features. All of the data are in very close scalers (Table 5).

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16
Mean	4.02	7.05	5.12	5.47	3.5	6.89	7.5	4.63	5.17	8.28	6.45	7.92	3.05	8.34	3.69	7.8
Std	1.91	3.31	2.01	2.26	2.19	2.02	2.32	2.70	2.38	2.49	2.63	2.08	2.33	1.55	2.57	1.62
(min, max)	(0,15)															

Table 5, General statistics of features

2.1 Decision Tree Model

A decision tree model without any pruning causes overfitting issue (Figure 11). The cross validation set has a large lower accuracy than training set, which is almost perfect.

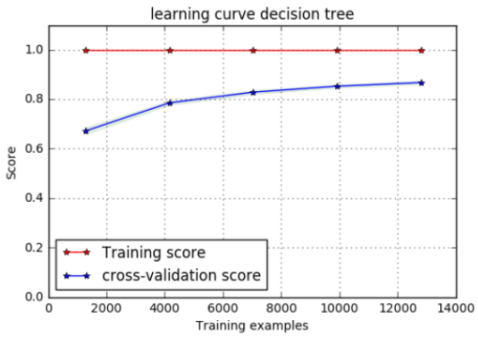

Figure 11, Learning curve for decision tree

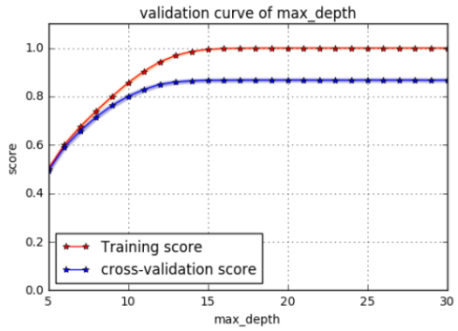

Figure 12, Max Depth, validation curve

To prune the tree, max_depth, which is the total depth of the tree structure, is tuned to prevent from overfitting. When the max_depth equals 13, the cross validation score reaches its maximum (Fig 12). The min_smaples_split is set 10. The pruned decision tree accuracy (Fig. 13) reaches cross validation accuracy of 0.88 (Table 5). “A” letter has the highest prediction accuracy while “K” letter has the lowest. The amount of nodes decreases from more than 3600 level to 2000 level.

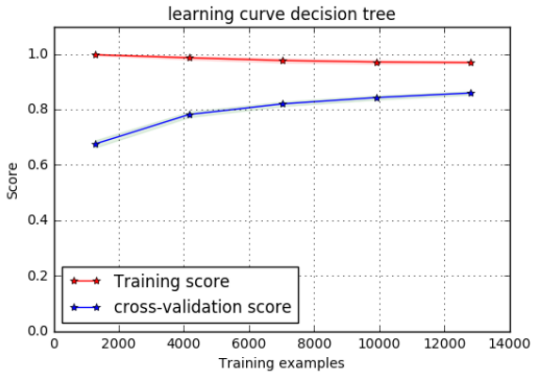

Figure 13, Learning curve after pruning

	precision	recall	f1-score	support
0	0.98	0.97	0.97	149
1	0.79	0.86	0.82	153
2	0.88	0.86	0.87	137
3	0.82	0.90	0.86	156
4	0.82	0.90	0.86	141
5	0.77	0.84	0.81	140
6	0.84	0.89	0.86	160
7	0.83	0.81	0.82	144
8	0.92	0.88	0.90	146
9	0.92	0.90	0.91	149
10	0.86	0.75	0.80	130
11	0.92	0.92	0.92	155
12	0.96	0.95	0.96	168
13	0.92	0.89	0.91	151
14	0.89	0.86	0.87	145
15	0.92	0.87	0.89	173
16	0.86	0.88	0.87	166
17	0.79	0.85	0.82	160
18	0.89	0.87	0.88	171
19	0.89	0.84	0.86	163
20	0.95	0.91	0.93	183
21	0.89	0.90	0.90	158
22	0.93	0.95	0.94	148
23	0.86	0.87	0.87	154
24	0.92	0.92	0.92	168
25	0.91	0.88	0.89	132
avg / total	0.88	0.88	0.88	4000

Table 5, classification report

2.2 Neural Network Model

In this model, a multiple layer neural network structure is used. A structure with a hidden layer of 50 nodes is implemented. This system looks like to be underfitting because both the training set and cross validation set has a large error, which is probably caused by bias (Figure 14).

To prevent the underfitting problem, a two-hidden-layer model, with 100 nodes in the first layer and 50 nodes in the second layer, is implemented. This model improves the accuracy (CV: 0.81 to 0.84, training: 0.86 to 0.9). But this causes some overfitting, which shows that the training accuracy seems to be much higher than the cross validation error (Figure 15).

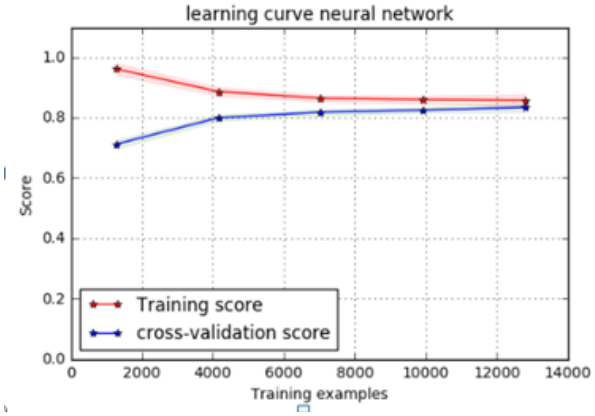

Figure 14, Learning curve (underfitting)

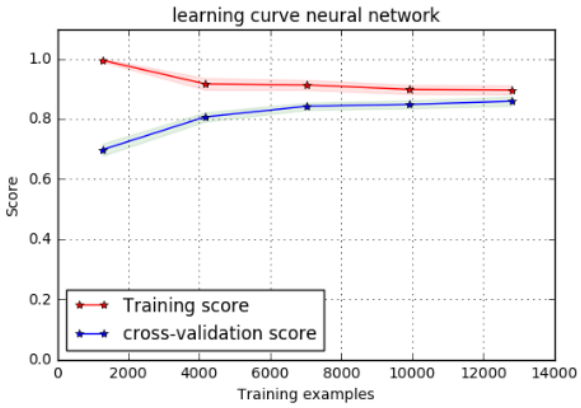

Figure 15, Learning curve (overfitting)

For the overfitting issue, the validation curve of the regularization parameter is studied. When alpha is about 1 (Figure 16), the model has the highest accuracy. If the alpha is too small, the model has a large variance; the model has a bias error when alpha is too large.

The hidden layer and alpha parameters are set as 1, the learning rate is set to be 0.02 and maximum iteration is 500. Here is the performance of this implementation in this dataset. The accuracy of cross validation is 0.9 and the accuracy of test set is 0.89.

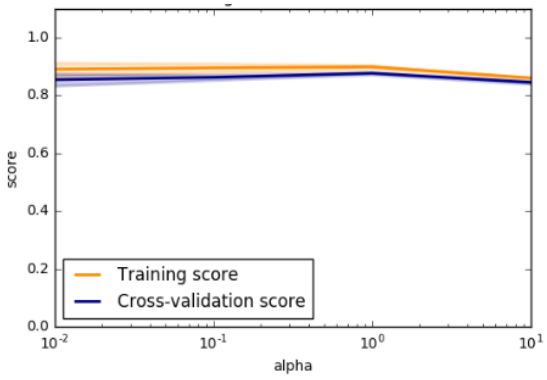

Figure 16, validation curve of alpha

	precision	recall	f1-score	support
0	0.91	0.93	0.92	149
1	0.79	0.87	0.83	153
2	0.93	0.84	0.88	137
3	0.82	0.89	0.86	156
4	0.83	0.82	0.82	141
5	0.86	0.87	0.87	140
6	0.82	0.86	0.84	160
7	0.87	0.72	0.79	144
8	0.95	0.86	0.91	146
9	0.91	0.90	0.91	149
10	0.81	0.85	0.83	130
11	0.91	0.90	0.91	155
12	0.93	0.93	0.93	168
13	0.93	0.88	0.90	151
14	0.83	0.87	0.85	145
15	0.93	0.87	0.90	173
16	0.91	0.89	0.90	166
17	0.80	0.84	0.82	160
18	0.88	0.92	0.90	171
19	0.91	0.90	0.90	163
20	0.91	0.95	0.93	183
21	0.98	0.91	0.94	158
22	0.93	0.95	0.94	148
23	0.85	0.94	0.89	154
24	0.93	0.92	0.93	168
25	0.94	0.94	0.94	132
avg / total	0.89	0.89	0.89	4000

Table 6. Testing data accuracy

2.3 AdaBoost Decision Tree

The adaboost here is still used for decision tree algorithms. If the maximum depth is still the same as in the early decision tree part, the model makes some overfitting problem (Figure 17 a). After a validation studies, the max_depth is set to be 8. More than one third of the previous decision tree depth is cut. The average accuracy is 0.95 when applying this to the test set.

.

Figure 17 (a). Decision tree with adaboost

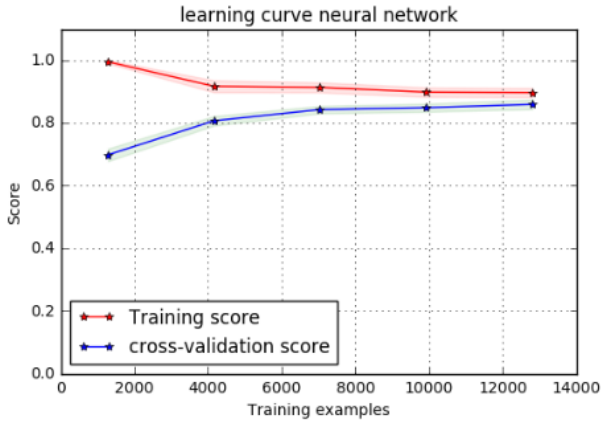

Figure 17 (b). Adaboost, further pruning

	precision	recall	f1-score	support
0	1.00	0.95	0.98	149
1	0.73	0.92	0.81	153
2	0.91	0.86	0.88	137
3	0.78	0.85	0.81	156
4	0.81	0.89	0.85	141
5	0.76	0.93	0.83	140
6	0.83	0.91	0.87	160
7	0.80	0.74	0.77	144
8	0.92	0.88	0.90	146
9	0.95	0.89	0.92	149
10	0.85	0.80	0.82	130
11	0.95	0.92	0.93	155
12	0.97	0.92	0.95	168
13	0.90	0.81	0.86	151
14	0.78	0.91	0.84	145
15	0.94	0.87	0.91	173
16	0.94	0.89	0.91	166
17	0.83	0.88	0.86	160
18	0.91	0.95	0.93	171
19	0.92	0.83	0.87	163
20	0.96	0.93	0.94	183
21	0.94	0.93	0.93	158
22	0.99	0.96	0.97	148
23	0.94	0.92	0.93	154
24	0.93	0.86	0.90	168
25	0.99	0.87	0.93	132
avg / total	0.89	0.89	0.89	4000

Table 7. Testing data accuracy

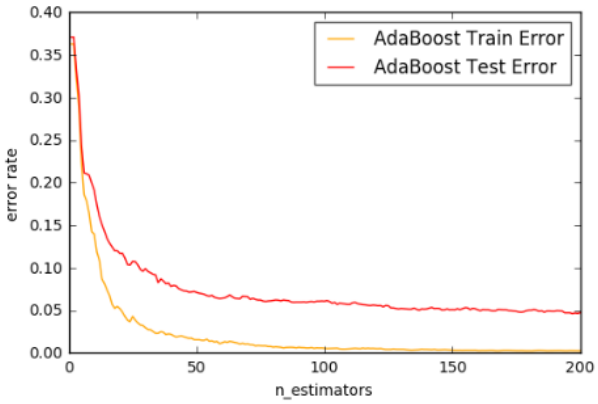

Figure 18. Errors with number of iteration

2.4 KNN Model

Validation curve for k (Figure 19) shows that when k is about 6, the cross validation can be most accurate. The average F1 score and precision is 0.96. (Table 8)

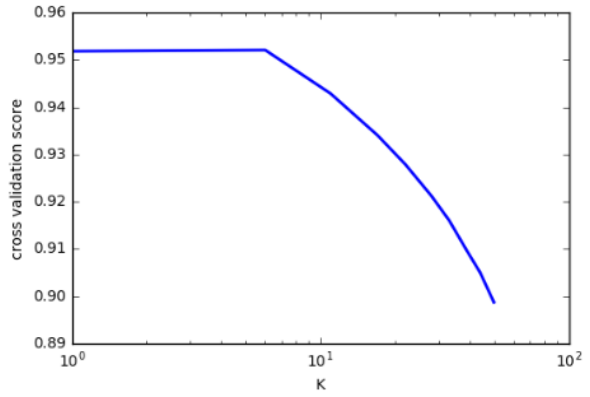

Figure 19. Validation curve of K

	precision	recall	f1-score	support
0	0.97	1.00	0.99	149
1	0.89	0.95	0.92	153
2	0.98	0.95	0.97	137
3	0.88	0.97	0.92	156
4	0.96	0.97	0.96	141
5	0.90	0.94	0.92	140
6	0.96	0.96	0.96	160
7	0.94	0.83	0.88	144
8	0.97	0.93	0.95	146
9	0.94	0.96	0.95	149
10	0.93	0.92	0.92	130
11	0.99	0.97	0.98	155
12	1.00	0.98	0.99	168
13	0.99	0.93	0.96	151
14	0.93	0.97	0.95	145
15	0.97	0.90	0.93	173
16	0.98	0.96	0.97	166
17	0.87	0.94	0.91	160
18	0.98	0.98	0.98	171
19	0.95	0.98	0.96	163
20	0.98	0.98	0.98	183
21	0.95	0.97	0.96	158
22	0.99	1.00	1.00	148
23	0.97	0.95	0.96	154
24	0.97	0.96	0.97	168
25	0.98	0.98	0.98	132
avg / total	0.96	0.95	0.96	4000

Table 7. Testing data accuracy

2.6 Summary

There are more data and more features in this multi-classification dataset. Thus more complex models are constructed for the learning algorithms. The overall accuracy is very good. The accuracy of the decision tree and neural network both reach 90%. However, the neural network structure with two hidden layers is very good at training, but also causing overfitting problem. Methods to prevent overfitting include increasing the regularization parameter, such as alpha in neural network or inverse of C in SVM, reducing features, and adding more data.