19Spring CS7641 **Homework 1**
GTID: 903383499 Supervised Learning
Yanzhe Cheng **Date: 02/03/2019**

1 Assignment Overview

In this assignment, some machine learning techniques in supervised learning are explored and implemented to analyze the influence of various hyperparameters as well as characters of them on different data set. The program language is Python and the main package used to implement several algorithms is Scikit-learn, a machine learning library, which guarantee that the experiment on both data set could be reproduced on a CoC Linux box.

2 Data Sets

2.1 Predicting a Pulsar Star

A Pulsar Star is highly magnetized rotating neutron star that can produce electromagnetic radiation. The periodical radio emission can be detected when the pulsars rotating and for each pulsar its radiation pattern would be different. However, because of the existence if noises and radio frequency interference(RFI), the detection of radiation would not enough to decide whether a candidate is a Pulsar Star.

Thus, machine learning algorithms can be applied to this classification problem to label the pulsar candidates. The data set contains a total of 17898 data which has been labeled and confirmed by human annotators. Each candidate has 8 attributes to describe its characters of radiation, composed of 4 variables from pulse profile and 4 variables form DM-SNR curve. The class label is 0 and 1, which means negative or positive classification.

This data is selected in this assignment for several reasons. First is that I am personally interested in this area and would like to combine this hobby with what I have learned. Second is that it makes sense to solve this problem with machine learning tools, since the radiation emission of Pulsar stars conform to certain pattern. Besides, this data set is a binary-class classification problem, which is a common problem category in machine learning area.

2.2 Mobile Price Classification

Mobile phones is one of the most popular and frequently used electronic devices in our daily life. It's price could vary according to the performance, brand, battery and so on. This data set is used to train the model to predict the price level of mobile phones according to its characters. 2000 entries are included in this data set with label. Totally 20 attributes are used to describe the sample, while the price of mobile phones are divided into 4 ranges.

This problem is tightly related to our daily life and it could be used to predict a device's potential price range whose official one has not published. Moreover, I am curious to inform from this data what is the most crucial characters to determine the price. This problem is a multi-class classification problem, which is different from the former data set.

2.3 Summary of Data set

There are two data set that are selected to support the experiments. Both of them are collected from Kaggle. These two data sets belong to different category of classification problems, which is binary-class and multi-class. Besides, the distinct quantity of sample and the number of attributes

make the result comparison between data sets reasonable. I expect to analyze the performance difference between several supervised learning algorithms on these data sets.

3 Q1 Pulsar Star

3.1 Decision Tree

The decision tree classifier is integrated in Scikit-learn library. Gini Index is used to measure the split of data, which is not changed in this assignment. The minimum samples leaf(min_samples_leaf) and max depth(max_depth) are set manually and we want to pruning both of them.

In order to achieve the pruning of hyperparameters, the function of GridSearchCV is used to implement the choosing of the best hyperparameters. This method will search over the specified parameter grid by implementing the fit and score procedure of the classifier. Cross validation will be used to evaluate the chosen parameters and all the scores of different parameters will be returned back, from where the optimal ones will be chosen for the further step.

For this problems, two hyperparameters are contained in th the parameter grid we set for the pruning. The minimum samples leaf number is chosen from [2,5,10,20,50,100], and the max depth number is in [1,3,5,10,15,20].

Table 1: Average Accuracy Score during Pruning (Pulsar Star)

Max Depth \ Min Leaves	2	5	10	20	50	100
1	0.9777993	0.9777993	0.9777993	0.9777993	0.9777993	0.9777993
3	0.97578783	0.97586233	0.97601132	0.97571333	0.97511734	0.97645832
5	**0.97899128**	0.97817179	0.97846979	0.97839529	0.97683081	0.97645832
10	0.97437235	0.97407435	0.9774268	0.9777248	0.97683081	0.97645832
15	0.97265887	0.97429785	0.9775013	0.9774268	0.97683081	0.97645832
20	0.97169038	0.97414885	0.9775758	0.9775758	0.97683081	0.97645832

We use the average accuracy score among 10 fold cross validations as the evaluation of the pruning procedure, the result of which is listed in Table 1. As it can be illustrated, the testing score will reach its maximum when minimum samples leaf number is 2 and max depth is 5.

Figure 1: DT Curve in Pulsar Star Problem

Thus, we set these values as the hyperparameters in the decision tree model. The scores of Accuracy, Recall, F1 and Precision is given as below. Although the performance of this set of hy-

Table 2: Adaboosting of Decision Tree Score (Pulsar Star)

	Accuracy	Recall	F1 Score	Precision
Scores	0.9787709497206704	0.9787709497206704	0.9782049230827293	0.9782436853859365

perparameters is not much obvious over others, all the score of which is higher than that of default one, which means the pruning works in this problem.

A sectional structure of the decision tree with optimal hyperparameters is shown below, the full version of which is in the attachment and could be generated by running the code.

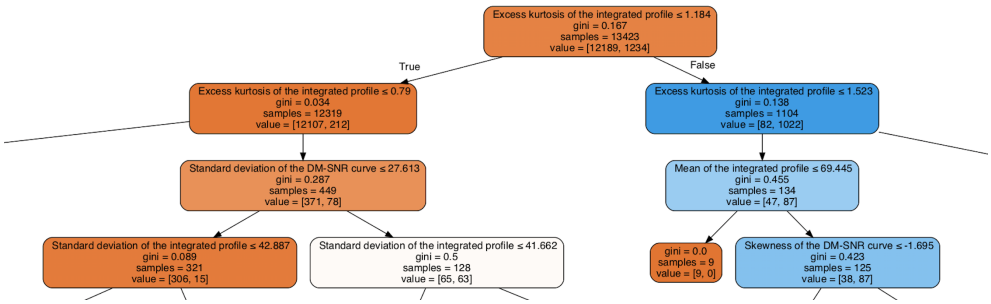
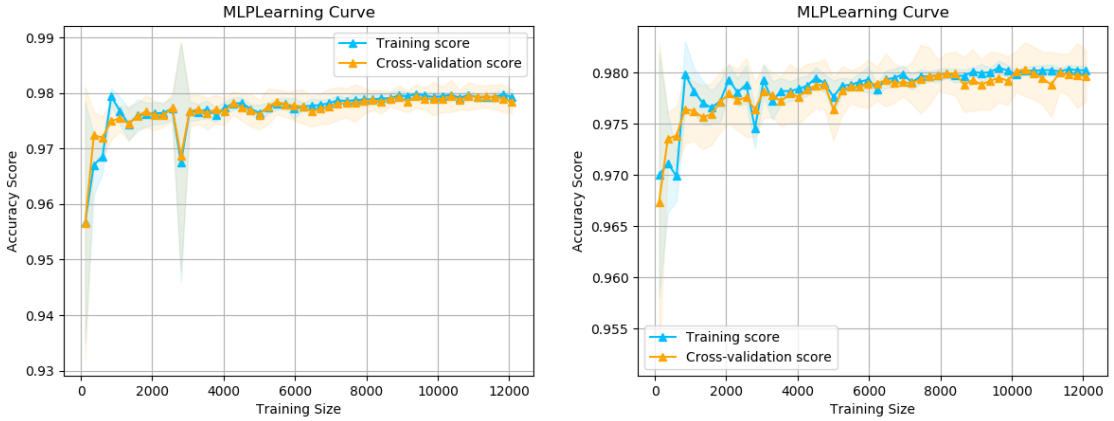


Figure 2: Decision Tree Structure in Pulsar Star Problem

3.2 Neural Networks

The Neural Network classifier is integrated in Scikit-learn library, which is the Multi-Level Perceptron classifier.

Rather than useing GridSearchCV, several structures of neural network are implemented and the performance is compared. The hidden layer and the number of neural in each layer are various and the performance of each structure is shown below. Other hyperparameters remains default.

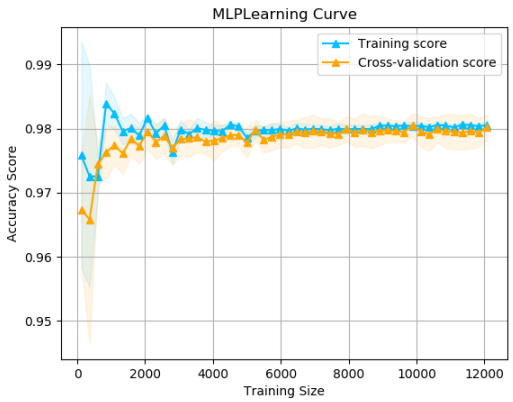


(a) Hidden Layer (5,5)

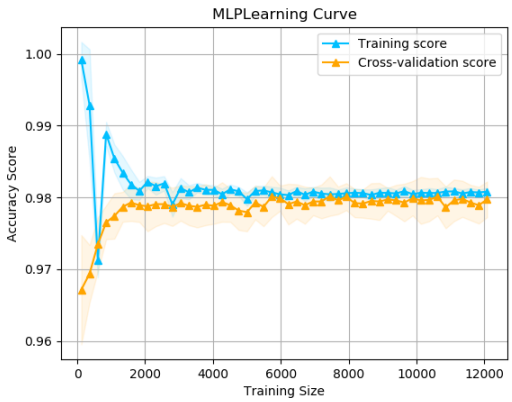
(b) Hidden Layer (10,10)

Figure 3: Curve of Neural Network with Different Structure (Pulsar Star)

The overall performance of different neural network structure are close to each other on this data set, with a high accuracy score and a relatively low bias.



(a) Hidden Layer (10,10,10)



(b) Hidden Layer (50,50)

Figure 4: Curve of Neural Network with Different Structure (Pulsar Star)

Table 3: Neural Network Accuracy Score(10,10,10) (Pulsar Star)

	Accuracy	Recall	F1 Score	Precision
Scores	0.9787709497206704	0.9787709497206704	0.9782846415521995	0.9782463272829767

3.3 Boosting

In order to implement boosting, the AdaBoostClassifier function in Scikit-learn library is applied. GridSearchCV is used to find the optimal parameter n_estimators, which is the number of estimators to terminate the boosting. This value is changed among [1, 5, 10, 50, 100, 200, 500].

Table 4: Adaboosting of Decision Tree with Different Number of Estimators (Pulsar Star)

n_estimators	1	5	10	50	100	200	500
Accuracy	0.97429785	0.97504284	0.9778738	0.97891678	**0.97951278**	0.97943828	0.97936378

Similar to the result in the former part, the performance with different parameter is not much far from each other. The overall trend of performance will slightly increase with the number of estimators increase and reach a peak at around 100.

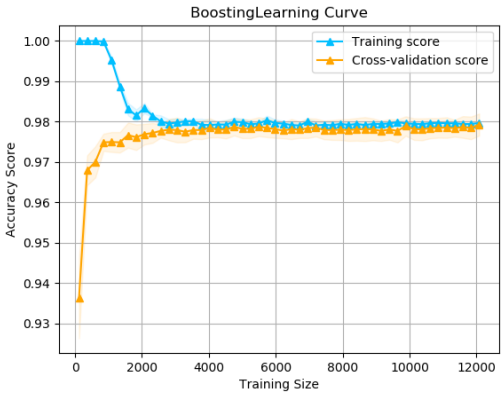


Figure 5: Boosting Curve in Pulsar Star Problem

This value is used to train the adaboosting model of decision tree, the curve result of which is shown in the following figure 3. We can see from the result that the boosting could barely brings

improvement on the testing performance, but the difference between training error and testing error is significantly narrower than that of decision tree model, which means the bias of model is reduced.

Table 5: Adaboosting of Decision Tree Score (Pulsar Star)

	Accuracy	Recall	F1 Score	Precision
Scores	0.9774301675977654	0.9774301675977654	0.9767421563888399	0.9768354979877822

3.4 Support Vector Machine

The classifier used in this assignment is also included in Scikit-learn library. GridSearchCV is used to implement the choosing of the best hyperparameters, including the kernel functions and some special parameters according to different kernel. Besides, the tol, which is the error threshold is set to 0.001. Max iter is set to -1, which means no limit on iteration. Other parameters are as default.

SVM with three different kernel functions are implemented and compared, including Linear kernel, Radial Basis Function(rbf) kernel, Polynomial(poly) kernel. For each kernel various parameters are set accordingly, like the gamma value for rbf kernel and poly kernel, the degree value for poly kernel.

Table 6: Score of SVM with Different Kernel and Parameters (Pulsar Star)

Kernel	Score	Kernel	Score
Linear	0.9778738	Poly, Degree = 5, Gamma = 0.001	0.90806824
Poly, Degree = 2, Gamma = 0.001	0.90806824	Poly, Degree = 5, Gamma = 0.0001	0.90806824
Poly, Degree = 2, Gamma = 0.0001	0.90806824	RBF, Gamma = 0.001	0.95336363
Poly, Degree = 3, Gamma = 0.001	0.90806824	RBF, Gamma = 0.0001	0.90806824
Poly, Degree = 3, Gamma = 0.0001	0.90806824		

The advantage of linear kernel over others are prominent, thus we chose it as the kernel of SVM model.

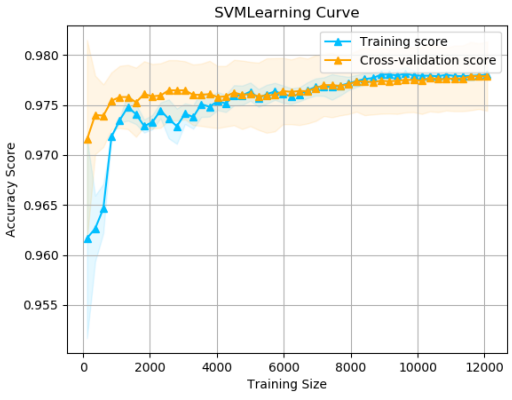


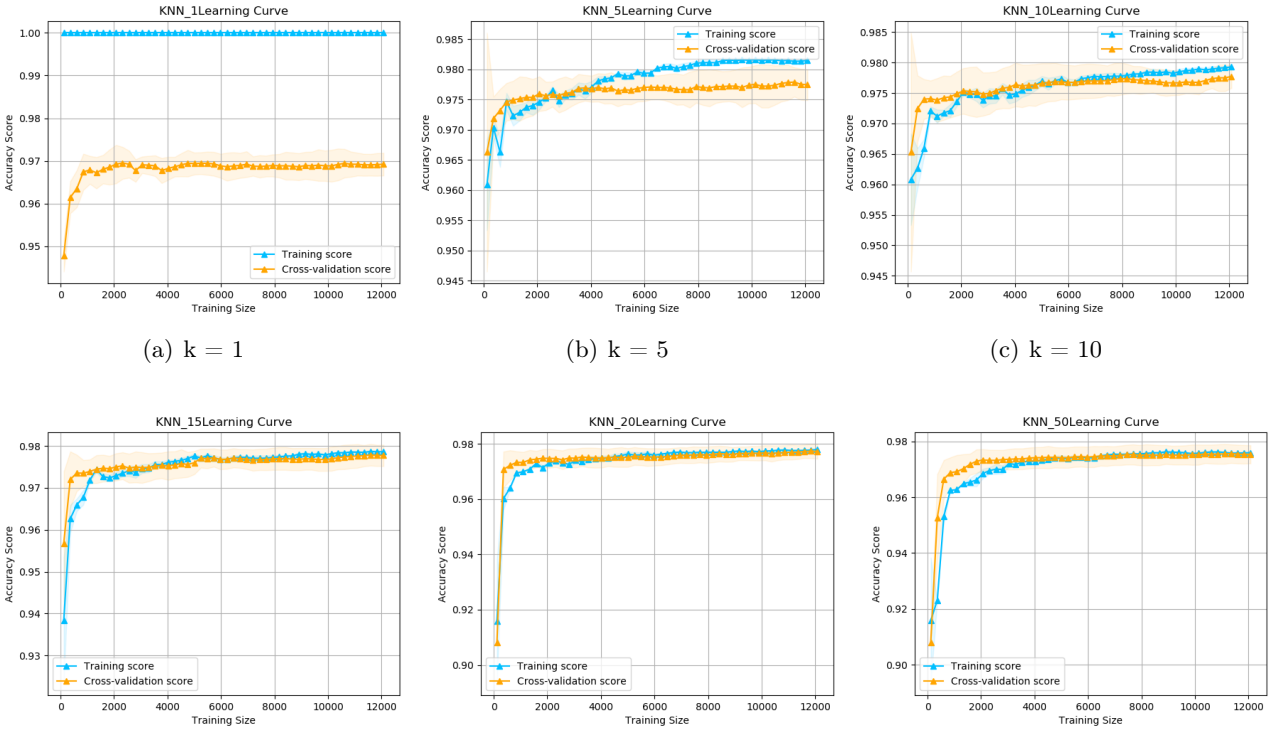
Figure 6: SVM Curve in Pulsar Star Problem

Table 7: SVM with Linear Kernel Score (Pulsar Star)

	Accuracy	Recall	F1 Score	Precision
Scores	0.9774301675977654	0.9774301675977654	0.9765046780363343	0.9769811360013773

3.5 K-Nearest Neighbors

Different K values are set to implement KNN algorithm. GridSearchCV is also used to compare each hyperparameters by its performance score. The k value varies from [1, 5, 10, 15, 20, 50].



(a) k = 1 (b) k = 5 (c) k = 10

(d) k = 15 (e) k = 20 (f) k = 50

Figure 7: Performance of K-NN Model with Different N (Pulsar Star)

It is when k equals to 15 that the accuracy reaches the highest. However, its score is really close to the models where k is 10 and 20.

Table 8: Optimal K-NN Score(k = 15) (Pulsar Star)

	Accuracy	Recall	F1 Score	Precision
Scores	0.9796648044692737	0.9796648044692737	0.9789391845215926	0.9792946065610372

4 Q2 Mobile Price

4.1 Decision Tree

Same as the solution in Q1, the Scikit-learn library is used to implement the decision tree algorithm and GridSearchCV is the base of hyperparameter pruning. Gini index is used as the measurement of the data split.

For this problems, two hyperparameters are contained in the parameter grid we set for the pruning. The minimum samples leaf number is chosen from [2,5,10,20,50,100], and the max depth number is in [1,3,5,10,15,20].

We use the average accuracy score among 10 fold cross validations as the evaluation of the pruning procedure, the result of which is listed in Table 1. As it can be illustrated, the testing score will reach its maximum when minimum samples leaf number is 5 and max depth is 20.

Table 9: Average Accuracy Score during Pruning (Mobile Price)

Max Depth \ Min Leaves	2	5	10	20	50	100
1	0.49133333	0.49133333	0.49133333	0.49133333	0.49133333	0.49133333
3	0.758	0.758	0.758	0.75733333	0.75066667	0.752
5	0.82266667	0.82466667	0.82133333	0.81266667	0.77466667	0.74933333
10	0.838	0.84933333	0.844	0.82666667	0.77466667	0.74933333
15	0.832	0.846	0.84666667	0.82666667	0.77466667	0.74933333
20	0.83066667	**0.85066667**	0.84466667	0.82666667	0.77466667	0.74933333

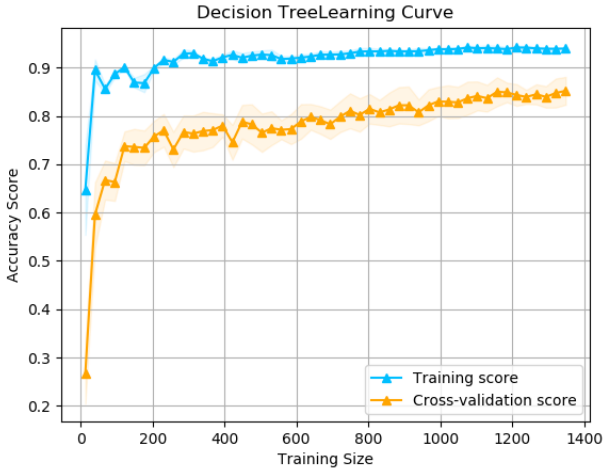


Figure 8: DT Curve in Mobile Price Problem

Thus, we set these values as the hyperparameters in the decision tree model. The scores of Accuracy, Recall, F1 and Precision is given as listed. All the score of this set of hyperparameters is higher than that of other set or a default one, which means the pruning works in this problem. However, different from the question 1, the performance of decision tree model will significantly vary according to the variation of hyperparameters, especially when the max depth number is small around 3. This will be discussed at the end of this report.

A sectional structure of the decision tree with optimal hyperparameters is shown below, the full version of which is in the attachment and could be generated by running the code.

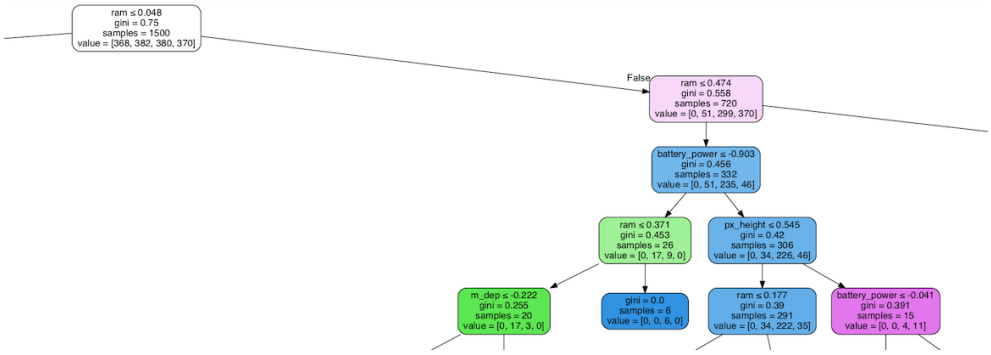


Figure 9: Decision Tree Structure in Mobile Price Problem

Table 10: Decision Tree Score (Mobile Price)

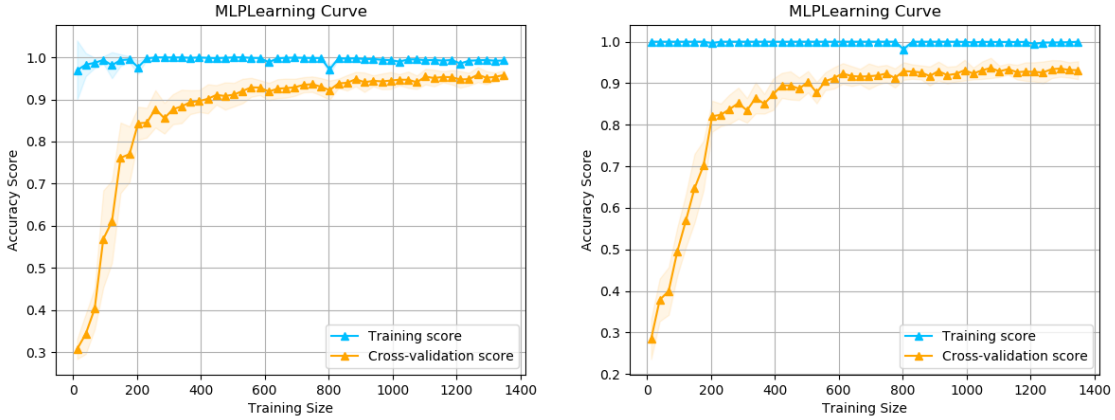
	Accuracy	Recall	F1 Score	Precision
Scores	0.826	0.826	0.8269102397818351	0.8299532818130649

4.2 Neural Networks

The Neural Network classifier is integrated in Scikit-learn library, which is the Multi-Level Perceptron classifier.

Same as problem 1, several structures of neural network are implemented and the performance is compared. The hidden layer and the number of neuron in each layer are various and the performance of each structure is shown below. The activation function of neural network is set as relu, which is the rectified linear unit function. The rest are all set to default.

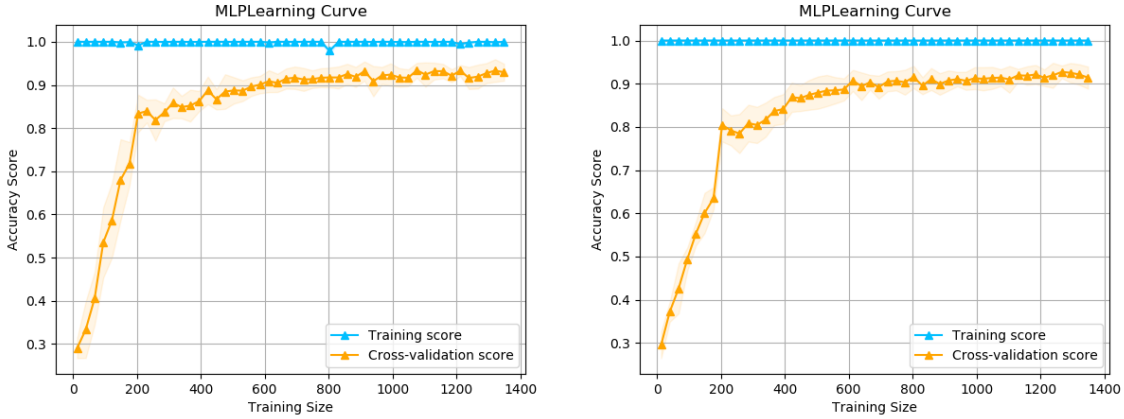
Four structure of hidden layer is examined. Three of them has two hidden layers, with 5, 10 or 50 neurons on each, while the other has three 10-neuron hidden layers.



(a) Hidden Layer (5,5)

(b) Hidden Layer (10,10)

Figure 10: Curve of Neural Network with Different Structure (Mobile Price)



(a) Hidden Layer (10,10,10)

(b) Hidden Layer (50,50)

Figure 11: Curve of Neural Network with Different Structure (Mobile Price)

Although the overall performance of different neural network structure are close on this data set, neural network with (5,5) hidden layer performs better with a higher accuracy score. Moreover, the bias of this neural network is smaller than the others.

Table 11: Optimal K-NN Score(k = 50) (Mobile Price)

	Accuracy	Recall	F1 Score	Precision
Scores	0.972	0.972	0.9720244843327674	0.9724154377217533

4.3 Boosting

In order to implement boosting, the AdaBoostClassifier function in Scikit-learn library is applied. GridSearchCV is used to find the optimal parameter n_estimators, which is the number of estimators to terminate the boosting. This value is changed from 1 to 500, to specify that is in [1, 5, 10, 50, 100, 200, 500].

Table 12: Adaboosting of Decision Tree with Different Number of Estimators (Mobile Price)

n_estimators	1	10	100	200	500	700	1000
Accuracy	0.84533333	0.876	0.88866667	0.908	0.90266667	**0.912**	0.90733333

Different from Q1, the performance with different parameter is more obviously far from each other. The overall trend of performance will increase with the number of estimators and reach a peak at around 700. This value is used to train the adaboosting model of decision tree, the curve result of which is shown in the following figure.

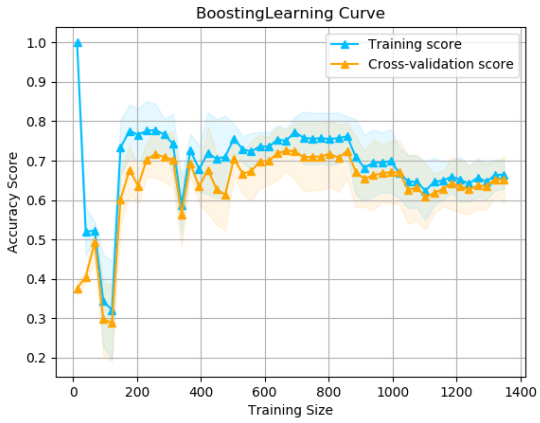


Figure 12: Boosting Curve in Mobile Price Problem

We can see from the result that the performance of boosting is even worse than its base model, but the difference between training error and testing error is significantly narrower than that of decision tree model, which means the bias of model is greatly reduced.

Table 13: Adaboosting of Decision Tree Score (Mobile Price)

	Accuracy	Recall	F1 Score	Precision
Scores	0.9	0.9	0.9003463801834186	0.9022854225021504

4.4 Support Vector Machine

Same as Q1, SVM with three different kernel functions are implemented and compared, including Linear kernel, Radial Basis Function(rbf) kernel, Polynomial(poly) kernel. For each kernel various parameters are set accordingly, like the gamma value for rbf kernel and poly kernel, the degree value for poly kernel. The tol is set to 0.001 and max iter is set to -1. Other parameters are as default.

Table 14: Score of SVM with Different Kernel and Parameters (Mobile Price)

Kernel	Cross Validation Score
Linear	**0.94866667**
Poly, Degree = 2, Gamma = 0.001	0.25466667
Poly, Degree = 2, Gamma = 0.0001	0.25466667
Poly, Degree = 3, Gamma = 0.001	0.25466667
Poly, Degree = 3, Gamma = 0.0001	0.25466667
Poly, Degree = 5, Gamma = 0.001	0.25466667
Poly, Degree = 5, Gamma = 0.0001	0.25466667
RBF, Gamma = 0.001	0.25466667
RBF, Gamma = 0.0001	0.25466667

Same as Q1, the advantage of linear kernel over others are prominent. The scores of SVM are listed.

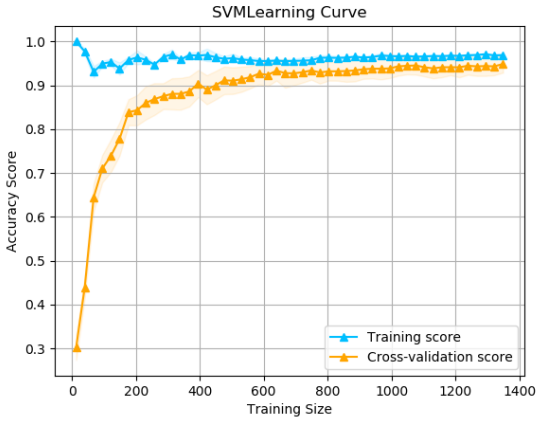


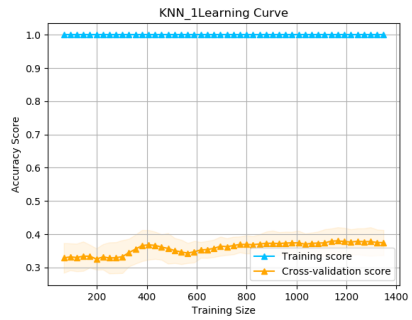
Figure 13: SVM Curve in Mobile Price Problem

Table 15: SVM with Linear Kernel Score (Mobile Price)

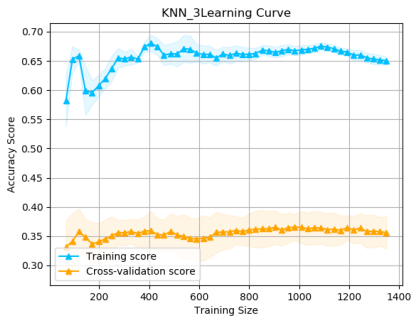
	Accuracy	Recall	F1 Score	Precision
Scores	0.964	0.964	0.9640140311112771	0.9658044254884806

4.5 K-Nearest Neighbors

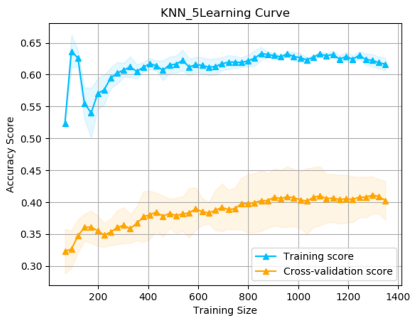
Different K values are set to implement KNN algorithm. GridSearchCV is also used to compare each hyperparameters by its performance score. The k value varies from [1, 3, 5, 10, 20, 50].



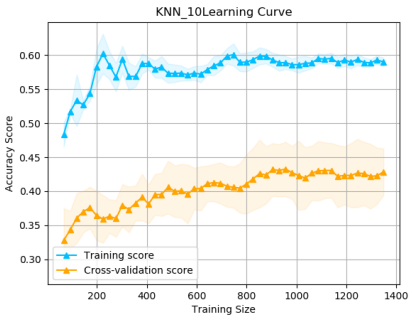
(a) n = 1



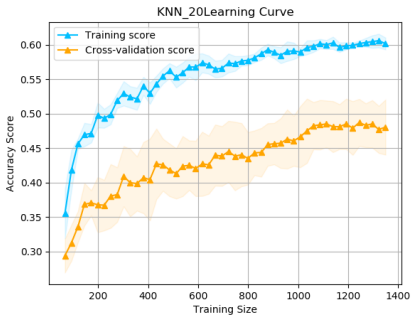
(b) n = 3



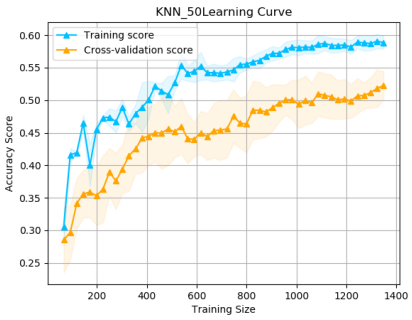
(c) n = 5



(d) n = 10



(e) n = 20



(f) n = 50

Figure 14: Performance of K-NN Model with Different N (Mobile Price)

It is when k equals to 10 that the accuracy reaches the highest. However, even under that circumstances, the accuracy as well as other scores are really low. This will be discussed later.

Table 16: Optimal K-NN Score(k = 50) (Mobile Price)

	Accuracy	Recall	F1 Score	Precision
Scores	0.528	0.528	0.5336349331765647	0.5449906746415685

5 Analysis and Summary

In order to reveal the characters of each classifiers, the experiment of each algorithm will be adjusted to the special problem. Comparison can be made over data set(size, problem, features, etc.), algorithms and hyperparameters. Basically the result of assignment meet my expectation.

What is the best algorithm to tackle each problems?
The problem we picked in this assignment belongs to different classification problem, which has been proved to have great impact on model performance.

How to define best? I think it should be evaluated from several aspect. Accuracy is the evaluation of correct classification, while the time consumption and the memory consumption can somehow represent the efficiency in time and space. For example, it will not be a good choice to use k-NN algorithm when the data set is gigantic(which means the great pressure on memory and testing time). There is no best, but the most suitable algorithm for every single problem.

For the problem I pick, in the pulsar problem, the k-NN algorithms with k = 15 performs a little better than that of other algorithm. It has the highest accuracy and the other scores, as well

as the lowest bias. Even so, the distinct between other models are not large, which is smaller than 0.001 for SVM and Neural Network. I think it is because the positive samples in data set is not much according to negative ones(around 20% of positive samples), which accords with the problem itself. The aggregation of data also make it in favor of k-NN model. While, however, in the other problem, the result is totally different. SVM with linear kernel performs the best both in accuracy and bias. k-NN, on the other hand, is the worst. I think it is because the data is not much as the first problem. Also, this is a multi-class problem and the distribution of data could not be easily divided linearly, which makes the advantage of SVM larger.

Will boosting(adaboosting) improve the performance?
The answer is uncertain. It can be seen from the result in problem 1 that the accuracy score of adaboosting is even lower than that of its base model, which is the decision tree. Since boosting is the superposition of estimators, it is not much possible to beats the optimal one. However, the bias is much lower than that of the other, especially when compared to decision tree model. And that is what boosting for.

Differences and characters of algorithms
Since k-NN is a lazy-learning algorithm, it will not spend much time to train the model. However, the space consumption of it will be very large and it takes time for searching result. Also, it is a great example of overfitting when k equals to 1.

SVM is a powerful classifier to deal with high dimension data, but the selecting of kernel function can have great impact on its performance.

Neural Network is powerful and that is proved in both problems. However, the structure of layers should be transformed according to the special problem and data. Also, the time consuming is considerable.

Performance of Decision Tree is influenced by the max depth and the leaf number. Although it may not be powerful to use a single tree, it is a good idea to build a forest.

Would cross validation help?
Yes, it is helpful to understand the real performance of the model.

What to improve in the performance?
This is the problem I faced when I implemented SVM classifier. It takes really long time for me to handle the 10000-entry data, which is not large. This would exist more frequently especially when using linear kernel when the model calculating the support vector. Data scaling is the solution to this problem and the efficiency of SVM with linear kernel improved greatly.

Besides, it will help a lot if chosen the hyperparameters properly. This is proved by several comparison in the classification result, like selecting kernel function for SVM, constructing neural network layer etc.