CS 7641 Machine Learning Assignment 1

Siyuan Li
sli761@gatech.edu

***Abstract*—**In this report, I used 2 datasets to apply the 5 supervised learning algorithms, which are Decision Tree, Neural Network, Boosting, SVM and k-NN. I will first describe the datasets and corresponding problem to discuss. Secondly, I will individually train and test each algorithm and analyze the pros and cons. Finally, I will compare the test result and discuss the methods I used to improve the model performance.

1 PROBLEM INTRODUCTION

In this paper, I will use 2 datasets as my research base. The first one is White Wine quality dataset. The second one is Italian Mortgage Loan Default data prior year 2004. The reason I chose 2 datasets is because the first dataset contains multiple classes while the second one is binary. I think this will be a better way for my study.

1) White Wine Quality Problem

The first classification problem is problem about classifying the quality of white wine (score from 0-10). The dataset contains the following features:

- Explanatory Variable: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates and alcohol
- Response Variable: Quality Score (0-10)

quality	*Number of observations*
3	20
4	163
5	1457
6	2198
7	880
8	175
9	5

I first got in touch with this dataset on CS 7646 Machine Learning for Trading Class. This is an introduction to how to do the classification. The dataset contains 4898 records with 11 explanatory variables. The response variable is expert's scoring, which is very common in many classification problems.

From my own perspective, it is an interesting problem because the it is very difficult to quantify the wine's quality. There are many different features, however, none of these features can alone explain the quality. With this research, I can know how to tell the difference between each kind of wine.

From the machine learning perspective, the dataset covers different kind of features, which is a good indicator as for research purpose.

2) Italian Mortgage Loan Default Problem

The second classification problem is a real case I encountered in my job. The whole dataset contains over 300,000 records from year 1990 to year 2020 of Italian mortgage loans. In this project, for simplicity, I only truncated the data before 2004, which contains 21,235 records. The dataset contains the following features:

- Explanatory Variables: OLTV, DTI, Employment Status, Loan Purpose, Payment Frequency, Interest Rate Type, the property's occupancy status, the loan's originator channel, payment type and whether the borrower is foreigner.
- Response Variable: Default (0: not default, 1: default)

Default	*Number of observations*
0	18518
1	2452

This is an interesting topic because its practical meaning in financial industry. When the lender/bank decide whether they will approve the loan application from a borrower, the probability of default is always the first thing to consider. When I did the project in my company, I used logistic regression. But after I took the courses, I think this should be a brilliant sample to apply classification algorithms. To avoid too much running time, I only take the data prior year 2004 as my sample. Please note, the data I presented are real-world data and may contain private information. I hided some features that may reveals borrower's privacy and this research is only for academic use. Access to this dataset is not permitted without notice.

2 DATA PROCESSING

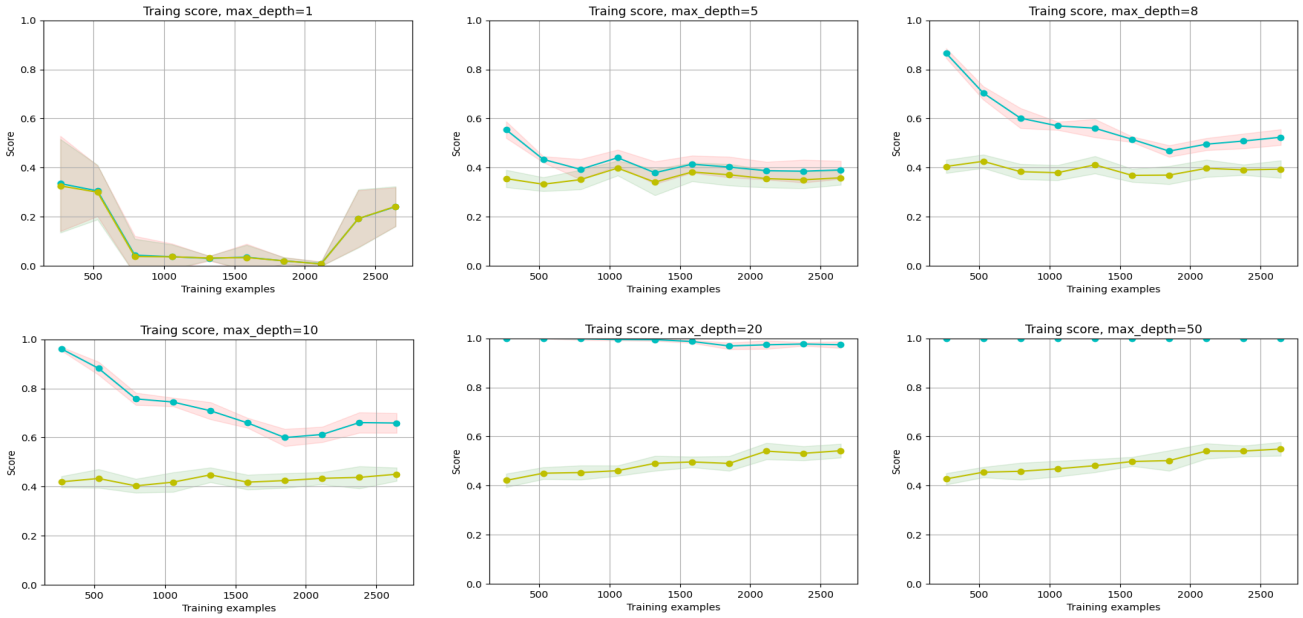
Both datasets are preprocessed and do not contain any missing values. The only process I did during this project is on Italian Mortgage Default dataset, most of the explanatory variables are categorical. I labeled them with python ScikitLearn LabelEncoder package.

3 ALGORITHMS DISCUSS

3.1 Decision Trees

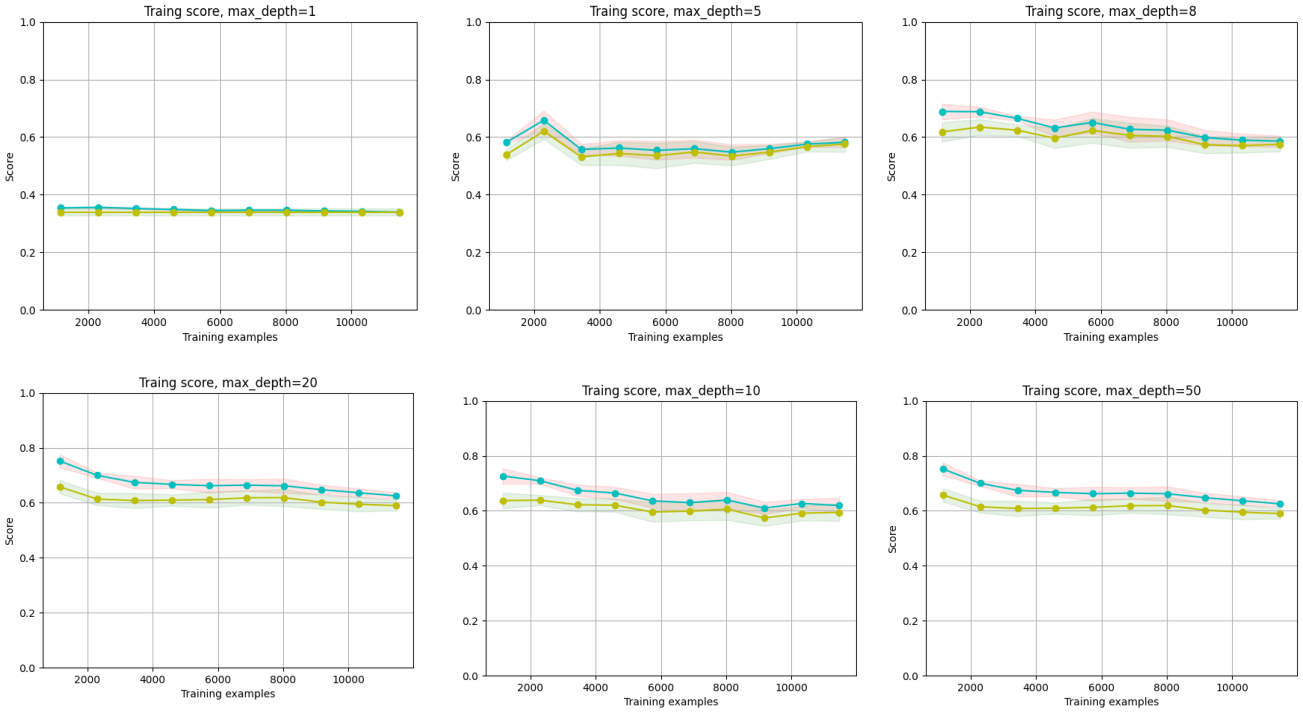
1) Performance Presentation

Wine Quality Problem:



Max Depth	Best Split Method	Tree Size	Training Score	Test Score	Running Time
1	entropy	3	0.2675	0.2602	0.1622
5	entropy	61	0.4129	0.3694	0.2425
8	entropy	329	0.5071	0.3796	0.3807
10	entropy	661	0.6430	0.4148	0.3770
20	gini	1753	0.9677	0.5531	0.5078
50	gini	1941	1.0000	0.5699	0.5024

Italian Default Problem:



Max Depth	Best Split Method	Tree Size	Training Score	Test Score	Running Time
1	gini	3	0.3393	0.3490	0.1865
5	gini	61	0.5857	0.5842	0.2558
8	gini	321	0.5903	0.5812	0.2944
10	gini	677	0.6131	0.6012	0.3024
20	entropy	1477	0.6146	0.5871	0.3384
50	entropy	1467	0.6146	0.5874	0.2910

2) Discuss

Overall, the decision tree algorithm works well for both datasets, especially for Italian Mortgage Default problem. There are some important observations I would like to discuss:

Overfitting:

Both datasets have the problem of overfitting, especially for wine quality problem. When the tree max depth goes from 10 to 20, the in-sample accuracy increases a lot, while the testing set is merely of no change, if not getting any worse. For a proper max_depth, I recommend 10 after a loop through 1 to 50

Effect of training size in cross validation:

It is to my surprise when I found that in the training score graph, the score does not change significantly with the training sample size increase. After I did some research, I think it is explainable because the smallest training set is 10% of dataset and as I shuffled data before using, 10% of data can be a good sample reflecting the true distribution of the whole dataset.

Choosing Parameter:

The main parameter I need to tune during the decision tree training is the split criterion. Both "gini" and "entropy" were tested using the GridSearch method. Comparing the two methods, "gini" is better in wine quality problem when the max depth is small and in Italian Mortgage Default problem when the max depth is large. "Entropy" is dominant in the opposite way. Thus, it is hard to say whether gini or entropy is better than the other one in my story.

Running time:

In term of running time, there is no significant difference, despite the default dataset is about 5 times larger than the wine quality dataset. My guess is although the default dataset is larger, it has only 2 classes while the wine quality dataset has 8.

Accuracy:

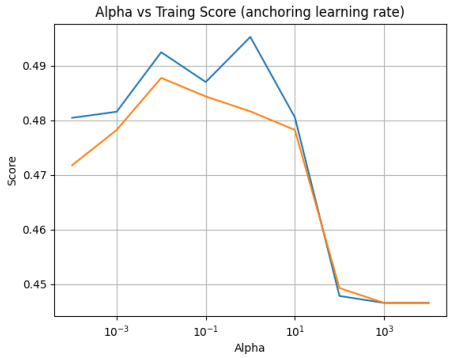
Taken overfitting into consideration, the best max depth should be set to 10 and the corresponding testing score for wine quality and default problems is around 0.5 and 0.6. The decision tree algorithm has a better performance on default dataset.

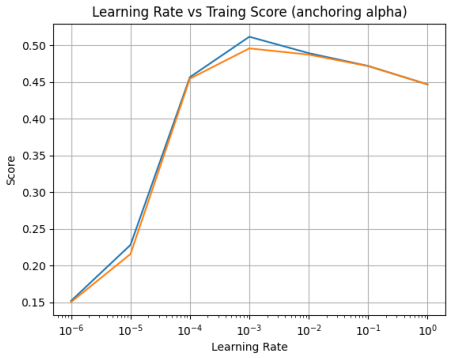
3.2 Neural Network

Before I proceed to neural network parameter tuning, I would first want to try the predict accuracy without any tuning. The predicting accuracy for wine quality dataset and mortgage default dataset is 40% and 85% respectively when I experiment with a small neural network with 2 hidden layers containing 5 and 5 nodes. I will use this as my benchmark.

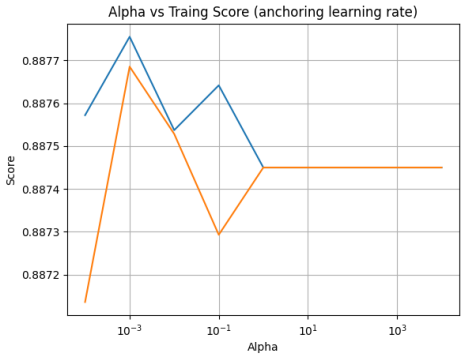
For neural network algorithm, the core parameters are learning rate and alpha. To get an idea about the proper range of to tune these two parameters, I first generated the predicting scores anchoring one of the parameters while changing the other.

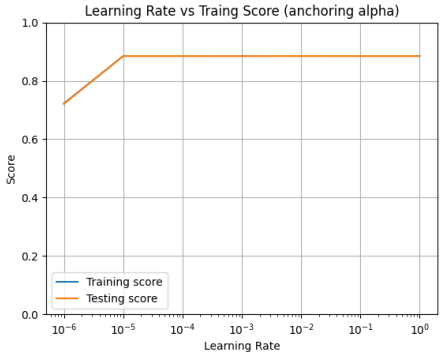
For wine quality dataset:





For default dataset:



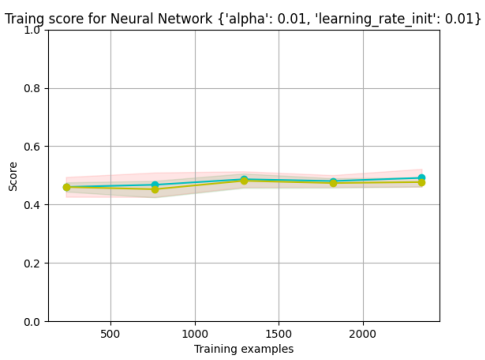


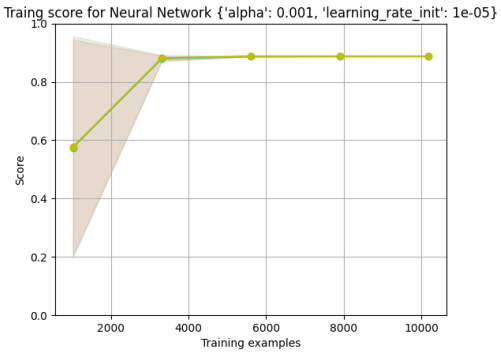
Thus, in next step of parameter tuning, the parameter for wine quality dataset will be {"alpha": [0.01,0.1,1,2,5,10],"learning_rate":[0.0001,0.001,0.01,0.1,1]}. For default dataset, the parameter range will be {"alpha":

[0.001,0.005,0.01,0.1], ”learning_rate”: [0.000001,0.00001,0.0001]}. After using the GridSearchCV function to loop through every parameter, the best parameter combos are as below

Best parameter	Wine Quality Dataset	Mortgage Default Dataset
Alpha	0.01	0.001
Learning Rate	0.01	0.00001
Training Time	306.06s	390.71s

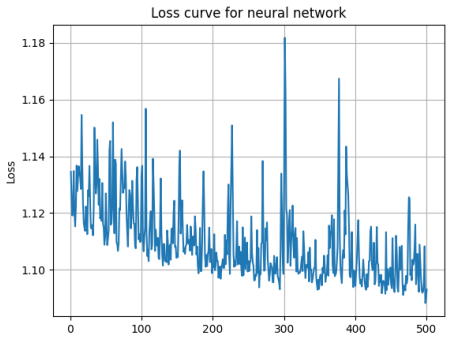
The learning rate for each best parameter sets are as below:

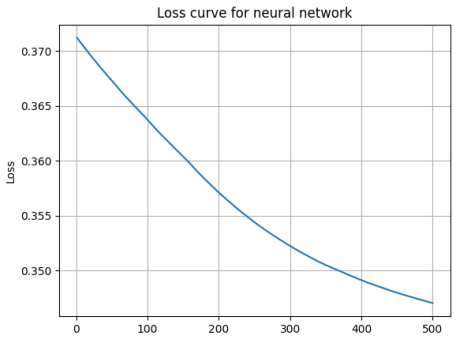


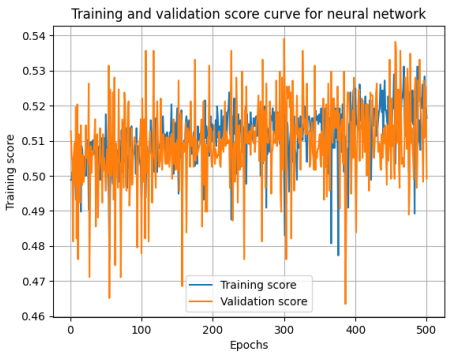


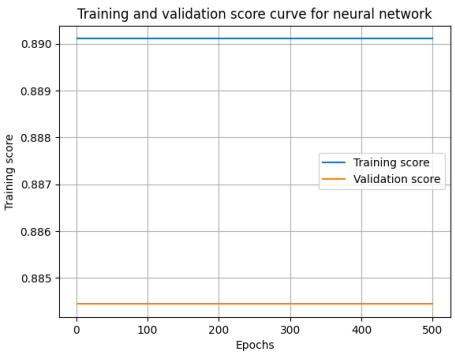
After tuning the parameters, the final model predict accuracy for wine quality dataset is 50.87%, while for Italian mortgage default dataset, the number goes to 89.97%. Both numbers see an obvious increase comparing to the benchmark.

The loss curve and prediction accuracy curve for each dataset is as below for 500 iterations:









It's kind of weird to see a loss curve like this. I did not think of the possible explanation for the time being. Had I got some time afterwards, I will try to revisit this part and try to find the possible solution for this.

Summary:

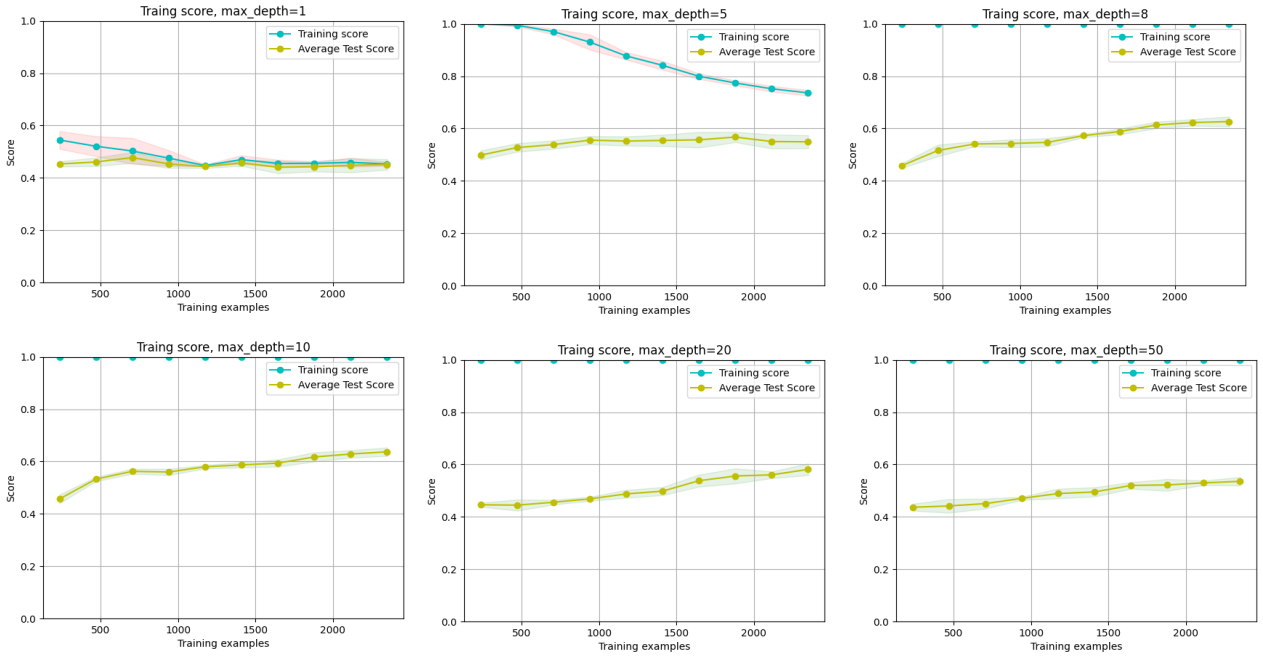
Overall, the parameter tuning increased the performance of each neural network by at least 5% of accuracy, this is a good indicator for my tuning. Compared to the previous decision tree algorithm, neural network can better handle the Italian mortgage default problem. As to wine quality problem, both algorithms have around 50% of prediction accuracy.

3.3 Boosting

As the boosting algorithm is closely related to decision tree, most of the analysis will be similar in this part. For comparison purpose, both max depth possible range were set the same

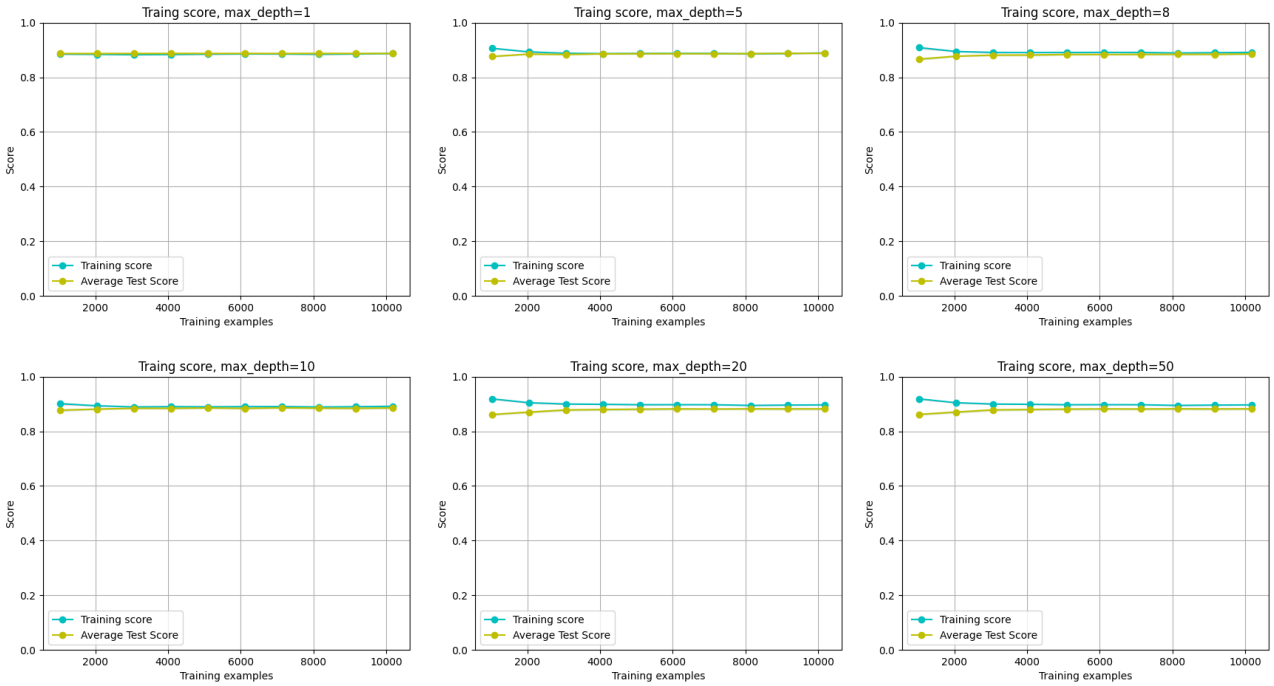
Performance Presentation:

For wine quality dataset:



Max Depth	Number of estimators	Split Method	Learning rate	Tree Size	Training Score	Test Score	Running Time
1	100	entropy	0.1	100	0.4585	0.4735	14.5583
5	100	entropy	0.1	100	0.6961	0.5526	33.5935
8	100	entropy	0.1	100	0.9993	0.6607	54.2231
10	100	entropy	1	100	1.0000	0.6480	71.7003
20	20	gini	1	2	1.0000	0.5765	22.8861
50	50	gini	0.1	1	1.0000	0.5561	2.8081

For mortgage default dataset:



Max Depth	Number of estimators	Split Method	Learning rate	Tree Size	Training Score	Test Score	Running Time
1	10	gini	1	10	0.8871	0.8811	38.7330
5	50	gini	0.01	50	0.8890	0.8812	64.1341
8	1	gini	0.001	1	0.8905	0.8794	63.3581
10	5	gini	10	5	0.8907	0.8798	32.2187
20	100	entropy	0.1	100	0.8955	0.8784	40.8676
50	100	entropy	0.1	100	0.8955	0.8784	38.5140

Discuss:

Performance Enhancement:

When comparing the performance between boosting and decision tree, we can easily find that the boosting method can significantly increase the predict accuracy. For both datasets, the predict accuracy increased by around 20%. This is a great performance enhancement, especially for wine quality problem.

Parameter tuning:

The main parameter I tuned in boosting method is number of estimators and learning rate. For the split method, I inherit the result from decision tree part. For number of estimator and learning rate, I let the GridSearchCV function loop from a list. It turns out that for wine quality dataset, the most effective parameter combo is max_depth=8, num_of_estimator=100, learning_rate=0.1 and split_method=entropy, while for mortgage default problem the best parameter combo is max_depth=5, num_of_estimator=50, learning_rate=0.01 and split_method=gini.

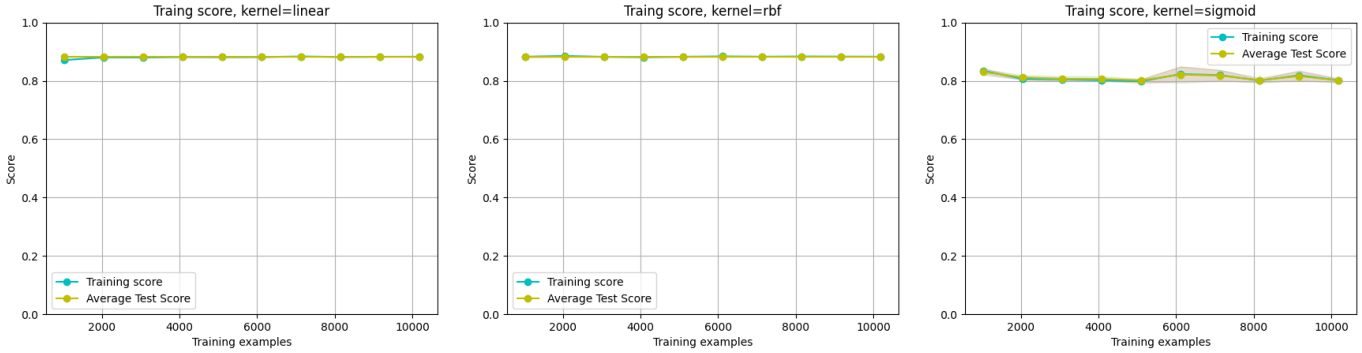
Overfitting:

Compared to decision tree without boosting, the boosting algorithm experienced a more severe overfitting problem. From the graphs above, we can clearly see that for wine quality dataset, the overfitting appears when the max_depth is larger than 8, while for mortgage default dataset, overfitting appears earlier when the max_depth is larger than 5.

3.4 Support Vector Machines

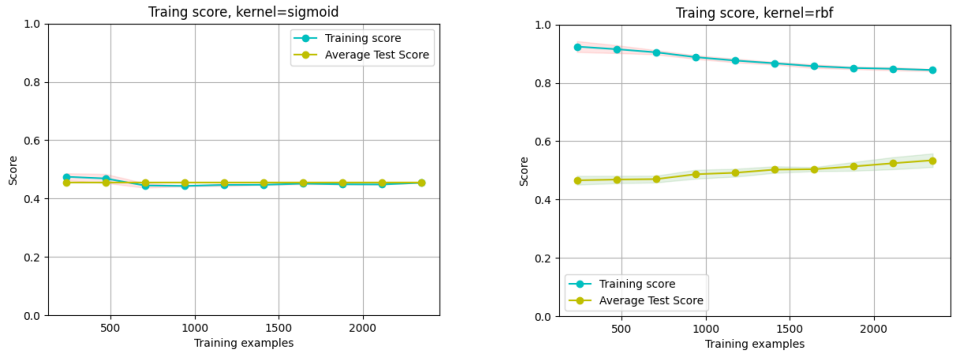
For SVM algorithm, the main parameter I tune is kernel. I set 3 possible kernels as my range, which are linear, rbf and sigmoind, and loop through it to see the performance. The performance as presented as below:

For mortgage default problem:



Kernel	Training Score	Test Score	Running Time
linear	0.8829	0.8870	7.2812
rbf	0.8829	0.8870	22.8009
sigmoid	0.7948	0.8020	17.4262

For wine quality problem:



Kernel	Training Score	Test Score	Running Time
rbf	0.8359428182437030	0.5239795918367350	5.998882
sigmoid	0.4543907420013620	0.4403061224489800	1.474352

Discuss:

There are many parameters to tune in SVM algorithm, including kernels, gammas, etc. However, in my analysis, I mainly tune the kernel and used the default gamma, which is 1/num_of_features at first step. I did this because the running time of SVM is significantly longer than previous algorithms. From the chart above, we can easily find that rbf kernel is the best one. I will then tune the gamma anchoring the kernel as rbf. The performances are as below:

Wine Quality				**Mortgage Default**			
Gamma	**Training Score**	**Test Score**	**Running Time**	**Gamma**	**Training Score**	**Test Score**	**Running Time**
0.01	**0.5616**	**0.4944**	**4.6925**	**0.01**	**0.8852**	**0.8836**	**6.7038**
0.05	**0.7308**	**0.5199**	**7.7906**	**0.05**	**0.8852**	**0.8836**	**10.9541**
0.1	**0.8346**	**0.5577**	**9.6849**	**0.1**	**0.8852**	**0.8836**	**11.7586**
0.5	**0.98093941**	**0.58469388**	**11.064649**	**0.5**	**0.8880**	**0.8830**	**8.6971**
1	**0.99387338**	**0.58571429**	**9.749358**	**1**	**0.8905**	**0.8817**	**8.6694**
2	**0.99795779**	**0.58418367**	**2.948743**	**2**	**0.8935**	**0.8792**	**8.4340**

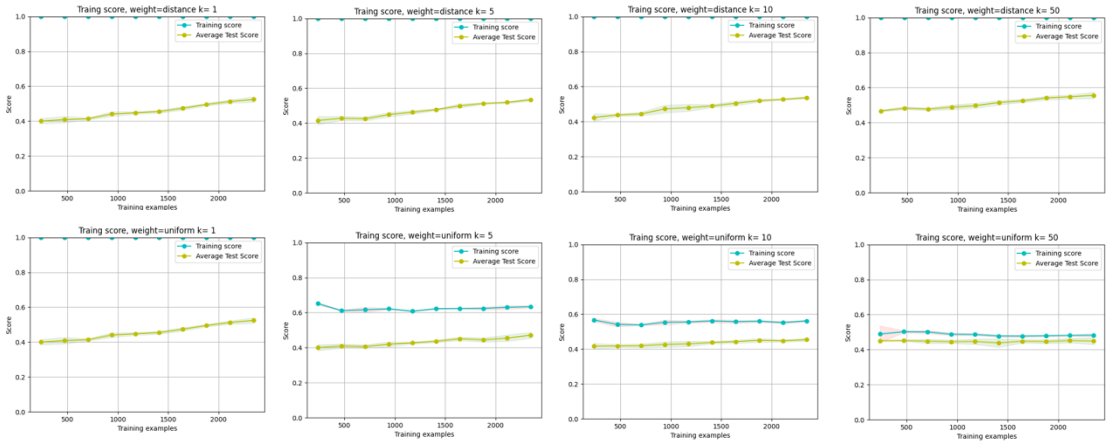
From the chart above, we can clearly see that for wine quality problem, the best parameter combo is "rbf"+"gamma=1" while for mortgage default problem, the best combo is "rbf"+"gamma=0.01".

As for running time, the rbf is longest, and it increase greatly while the dataset size increase. It may cause a problem when the dataset is large enough as currently my datasets are both relatively small.

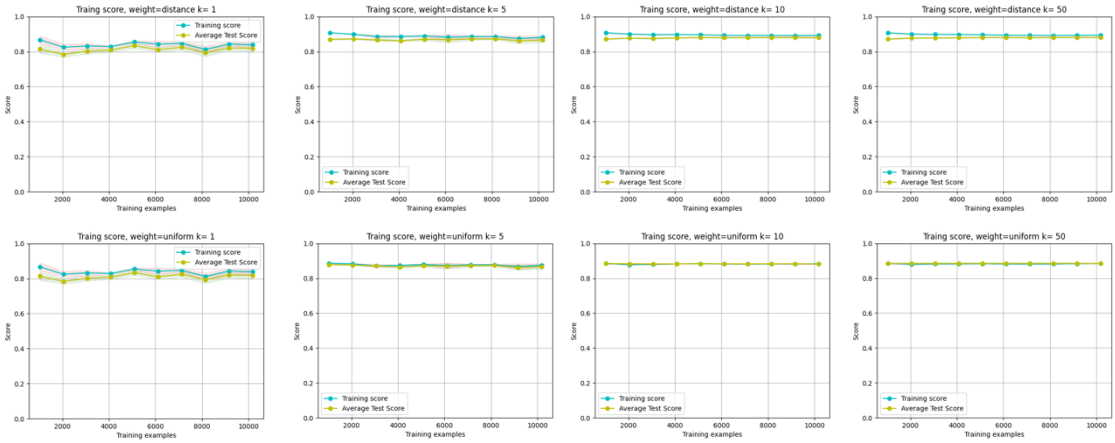
3.5 k-nearest Neighbors

For k-NN algorithm, the main parameter I tuned are the weight methods and k. For weight method, I loop thought "uniform" and "distance". For k, I loop through 1 to 50. Due to the limitation of length, here I just present some certain graphs.

For win quality dataset:



For mortgage default dataset:



To show more clearly, I generated a chart with number:

Wine Quality					Mortgage Default				
Weight	**K**	**Training Score**	**Test Score**	**Running Time**	**Weight**	**K**	**Training Score**	**Test Score**	**Running Time**
uniform	1	1.0000	0.5469	0.3395	uniform	1	0.8120	0.7926	2.1156
uniform	5	0.6419	0.4418	0.3499	uniform	5	0.8689	0.8605	2.2278
uniform	10	0.5725	0.4587	0.4544	uniform	10	0.8849	0.8834	2.3304
uniform	20	0.5259	0.4704	0.3653	uniform	20	0.8849	0.8842	2.5753
uniform	50	0.4847	0.4551	0.4227	uniform	50	0.8846	0.8844	3.2643
distance	1	1.0000	0.5469	0.1066	distance	1	0.8120	0.7926	1.1666
distance	5	1.0000	0.5515	0.1580	distance	5	0.8743	0.8594	1.3192
distance	10	1.0000	0.5653	0.1286	distance	10	0.8924	0.8809	1.3869
distance	20	1.0000	0.5923	0.1713	distance	20	0.8924	0.8807	1.6727
distance	50	1.0000	0.5898	0.2056	distance	50	0.8924	0.8807	2.4905

From the chart above, we can clearly see the best parameter combo for wine quality problem is “distance” + ”k=20”, while for the mortgage default problem, the best combo is “uniform” + “k=50”. We can see that for both dataset, the k-NN algorithm can perform a good classification.

As for running time, the k-NN algorithm is a relatively faster algorithm comparing to the other 4. However, the running time also increase significantly with the increase of the dataset size.

The overfitting problem also exist in k-NN algorithm, especially for “uniform” weight. When k is small, the training score is high while the test score is low, indicating the overfitting exists.

4 PERFORMANCE COMPARISON

In this part, I will discuss the performance of each algorithm. For a clearer comparison, I generate a table below.

For win quality dataset:

	Highest test score	Best Parameter combo	Running time	Overfitting?
Decision Tree	57%	{max_depth=50,split_method=”gini”}	0.5s	Exist
Neural Network	51%	{alpha=0.01,learning_rate=0.01}	306.06s	N/A
Boosting	67%	{max_depth=8, num_of_estimator=100, learning_rate=0.1, split_method=entropy}	54s	Exist
SVM	59%	{kernel-“rbf”,gamma=1}	9.75	N/A
k-NN	59%	{weight=”distance”,k=20}	0.17	Exist

For Italian Mortgage Default problem:

	Highest test score	Best Parameter combo	Running time	Overfitting?
Decision Tree	60%	{max_depth=10,split_method=”gini”}	0.7s	Exist
Neural Network	90%	{alpha=0.001,learning_rate=0.00001}	370s	N/A
Boosting	88%	{max_depth=5, num_of_estimator=50, learning_rate=0.01 and split_method=gini}	64s	Exist
SVM	88%	{kernel-“rbf”,gamma=0.01}	6.73s	N/A
k-NN	88%	{weight=”uniform”,k=50}	3.26s	Exist

We can easily find the as for all method I tried, for wine quality problem, the best method is boosting while for Italian Default Problem, the best method is Neural Network.

In terms of running time, neural network takes longest time to run while k-NN and Decision Tree both take short time to run.

5 FURTHER DISCUSSION

For the time being, I feel strange about the performance curve of neural network. If I have some more time, I may need to revisit this part to see if there is anything I can do to reduce the running time.

After running all method, I think boosting algorithm is the one that can apply most conveniently. The running time is acceptable, the accuracy is high, there are multiple parameter we can tune. All of these conditions make it a good algorithm in classification problem.