Enmao Diao CS4641
Project 1

Supervised Learning Report

DATASETS

Wine Quality Dataset. Two datasets are included, related to red and white wine samples, from the north of Portugal. The goal is to model wine quality based on physicochemical tests. Although there are just 6 and 7 categories in these two datasets. However, the class distribution is extremely skewed as you can see in Figure 1. The quality scores in *Wine Quality Red Dataset* range from 3 to 8 and in *Wine Quality White Dataset* from 3 to 9. Low and high Quality scores have very limited amount of data samples.

The complete *Wine Quality Red Dataset* contains 1,599 instances and *Wine Quality White Dataset* contains 4,898 instances and both of them have 12 attributes. Attributes information for these two dataset can be found in the winequality.info file (see README.txt).

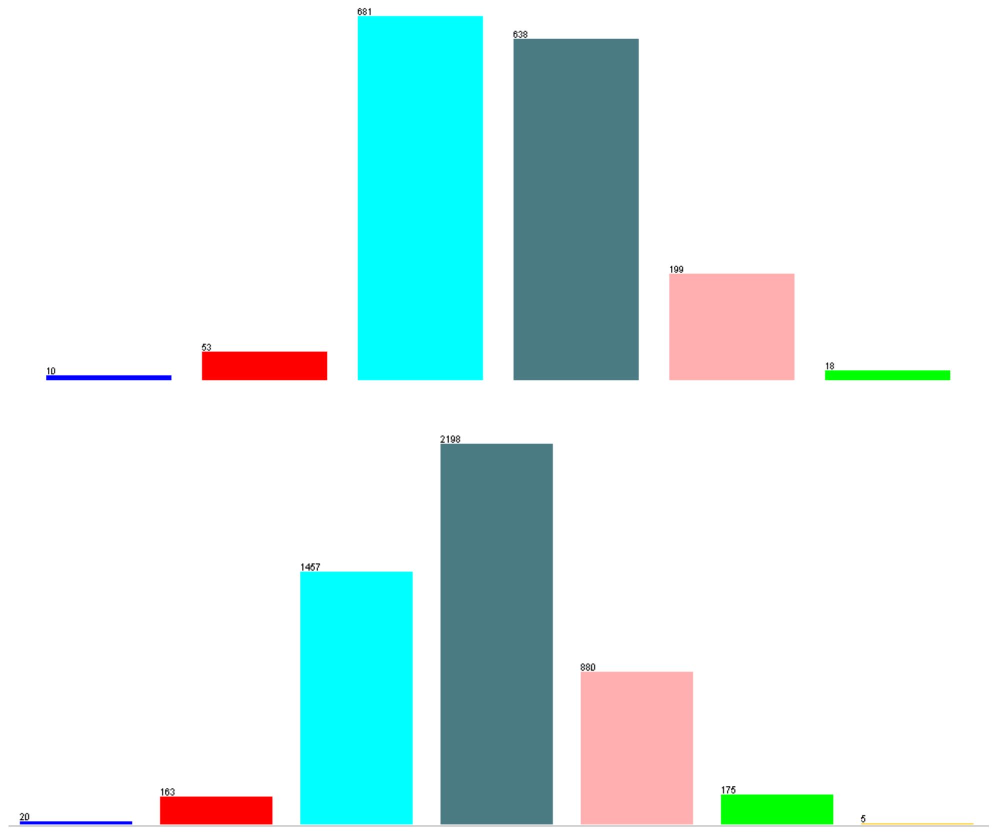


Figure 1. Class distribution of Wine Quality Dataset.

MUSK (Version 2) Dataset. This dataset describes a set of 102 molecules of which 39 are judged by human experts to be musks and the remaining 63 molecules are judged to be non-musks. The goal is to learn to predict whether new molecules will be musks or non-musks. However, the features that describe these molecules depend upon the exact shape, or conformation, of the molecule. A single molecule can have many different shapes. All the conformations of the molecules were generated to produce 6,598 conformations. The conformation name attributes are removed to create a pure numeric dataset. This dataset therefore contains 6,598 instances and 166 attributes. The class distribution is shown in Figure 2.

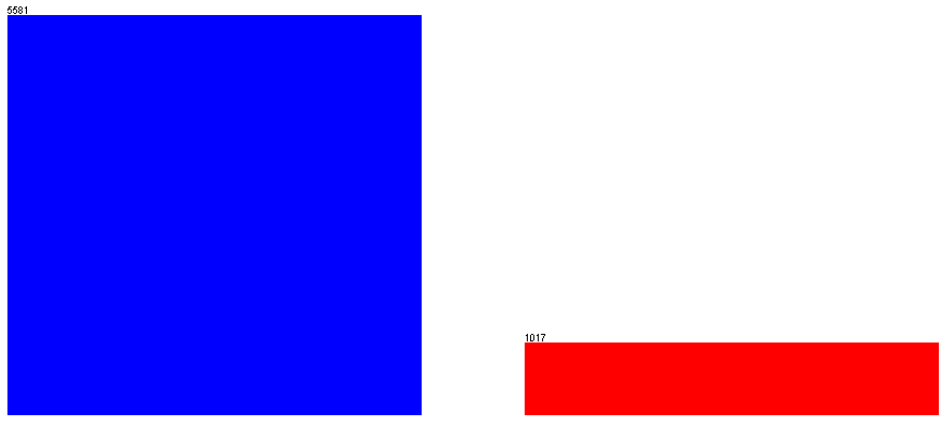


Figure 2. Data sample distribution of Musk Dataset.

WHY INTERESTING

These two datasets are interesting not only because of their class distribution but also because they share some similarities practically. The wine dataset have an interesting class distribution. As mentioned above, there are only 1~3% instances labeled as quality 3 and 4 in the datasets. In multiclass classification, it is common to encounter a situation that there is very limited amount of data for a specific class. It is interesting to dig into confusion matrix in order to see how different classification algorithms perform under this condition rather than only gather the statistical result. Musk dataset is used to compare multiclass classification with respect of binomial classification. Figure 2 shows that there are only 15% of data samples labeled as musks. In a binomial classification problem, this is rather interesting. Despite of the fact that Musk dataset also have limited amount data sample in on category, it still yield decent result thanks to binomial classification. Comparing the result of these two datasets, we can see the difficulties lie in skewed multiclass classification problem.

Moreover, these two datasets also share some interesting similarities. Both of their attributes are labeled by experts in the field. Therefore categorical data are determined subjectively. However, other attributes of these two datasets are scientific quantification like conformation of molecules and physicochemical data. All these data therefore are numeric. Through different classification algorithms, we can show how the scientific quantification can be applied for real-world classification problems. The results can help reveal how the inherent scientific model determines the extrinsic interpretation.

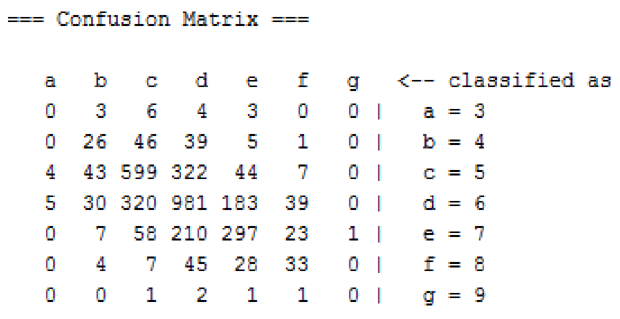
APPROACH

My analysis not only aims to gather the statistical result of classification algorithms but also attempts to improve the performance as large as possible by tuning some parameters of each algorithm. First, we split the datasets into 70% training dataset and 30% test dataset. Five classification algorithms are applied for the training dataset with 10-fold cross-validation, except neural network with 3-fold for simplification. Parameters are also tuned in these algorithms within this process. Then, the best model of each algorithm after cross-validation is select for each dataset. Next, we split the training dataset into 20%~80% subset in order to gather the learning curve information. Finally, we apply the best model for each algorithm on each subset with the 30% test set separated in the first place. In this way, not only the best model for each algorithm is found but we also have a feeling of how the algorithms perform when training data gradually increases.

DECISION TREE

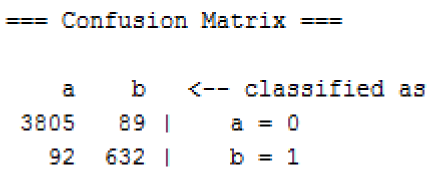
Table I
Decision Trees Results

Data	confidence	Training %	Test (cross-validation) %	leaves	trees	train time s	test time s
Wine_red	**0. 125**	88. 9187	**60. 2324**	136	271	0. 11	0. 02
	0. 25	89. 723	58. 8025	145	289	0. 12	0. 02
	0. 5	90. 6166	58. 7131	156	311	0. 12	0. 02
	Unpruned	90. 9741	58. 8025	170	339	0. 1	0. 02
Wine_white	0. 125	86. 5519	55. 8051	450	899	0. 31	0. 03
	0. 25	88. 8856	**56. 4761**	526	1051	0. 32	0. 04
	0. 5	89. 5566	56. 1552	554	1107	0. 31	0. 03
	Unpruned	89. 8191	55. 9802	578	1155	0. 24	0. 03
Musk	**0. 125**	99. 0472	**96. 1455**	74	147	1. 45	0. 15
	0. 25	99. 5886	96. 0806	92	183	1. 42	0. 15
	0. 5	99. 6752	95. 9939	97	193	1. 44	0. 14
	Unpruned	99. 6752	95. 8857	97	193	1. 36	0. 16



```
=== Confusion Matrix ===

  a   b   c   d   e   f   g   <-- classified as
  0   3   6   4   3   0   0 |   a = 3
  0  26  46  39   5   1   0 |   b = 4
  4  43 599 322  44   7   0 |   c = 5
  5  30 320 981 183  39   0 |   d = 6
  0   7  58 210 297  23   1 |   e = 7
  0   4   7  45  28  33   0 |   f = 8
  0   0   1   2   1   1   0 |   g = 9
```



```
=== Confusion Matrix ===

    a    b   <-- classified as
 3805   89 |    a = 0
   92  632 |    b = 1
```

Figure 3. (a) Wine White dataset Confusion Matrix. (b) Musk dataset Confusion Matrix.

There are several things that worth noticing from Table I. First, pruning does improve the accuracy of Wine dataset and have little impact on Musk dataset. The confidence is used to compute an upper bound on the error rate at a leaf/node. The smaller this value, the more the estimated error is and generally the heavier the pruning. Especially of Wine Red dataset, the results generally show that pruning successfully helps reverting some overfitting of decision trees. Interestingly, using aggressive pruning also yields decent result from these two datasets, since generally scientific measurements are less overlapped than empirical study. Some of the scientific measurements of wine data are alcohol and acidity. I believe that it is because the scientific measurements of wine dataset are not inherent enough to describe the features separately. Compared with wine dataset, the musk dataset which describe the features with the conformation of molecules only have slight improvement on accuracy. If the wine dataset have more detailed attributes, then pruning should have less effect on accuracy. What is interesting here is that the wine white dataset actually yields a peak accuracy at 0.25 confidence level. This means that certain attributes have different impact on two different wines. Attributes like fixed acidity and volatile acidity should have less impact on the wine white dataset as they are more closely related to each other.

As for the training time and test time, one can easily notice that training time is significantly higher than the test time which is what we expect for an eager learning technique. Generally, pruning takes more time in my implementation, since we trace back from unpruned tree based on the confidence level.

All the confusion matrix in this report is the result of the best test accuracy and you can also assume the confusion matrix of wine red dataset is similar to that of wine white dataset. Diagonal along the confusion matrix are the data that are correctly classified. From the confusion matrix, we should also notice that there are not many data samples from quality 3, 4, 8 and 9 (skewed classes) are correctly classified. Obviously, this is due to the very limited amount of data labeled as these classes. We will later compare this with other classification algorithms.

NEURAL NETWORK

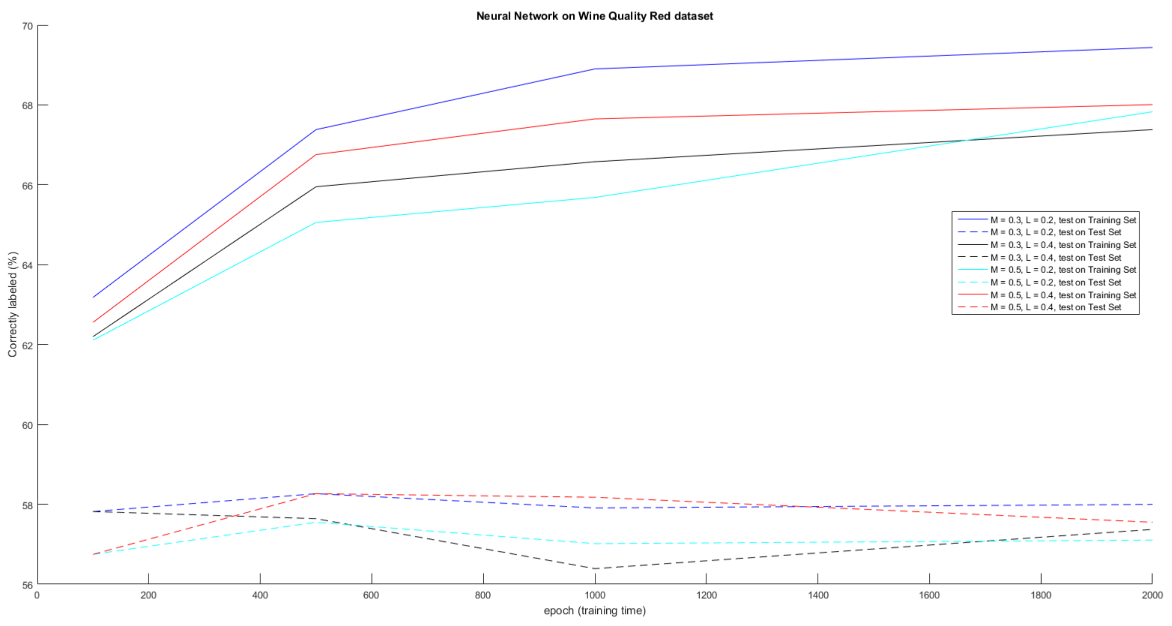


Figure 4. Neural Network on Wine Quality Red dataset.

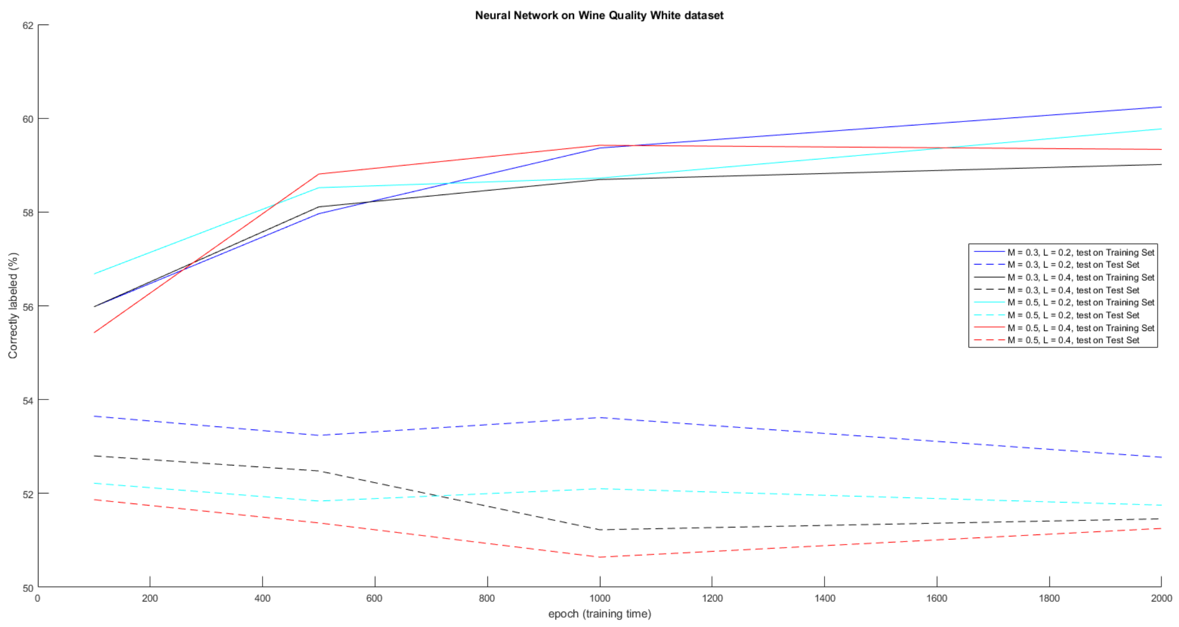


Figure 5. Neural Network on Wine Quality White dataset.

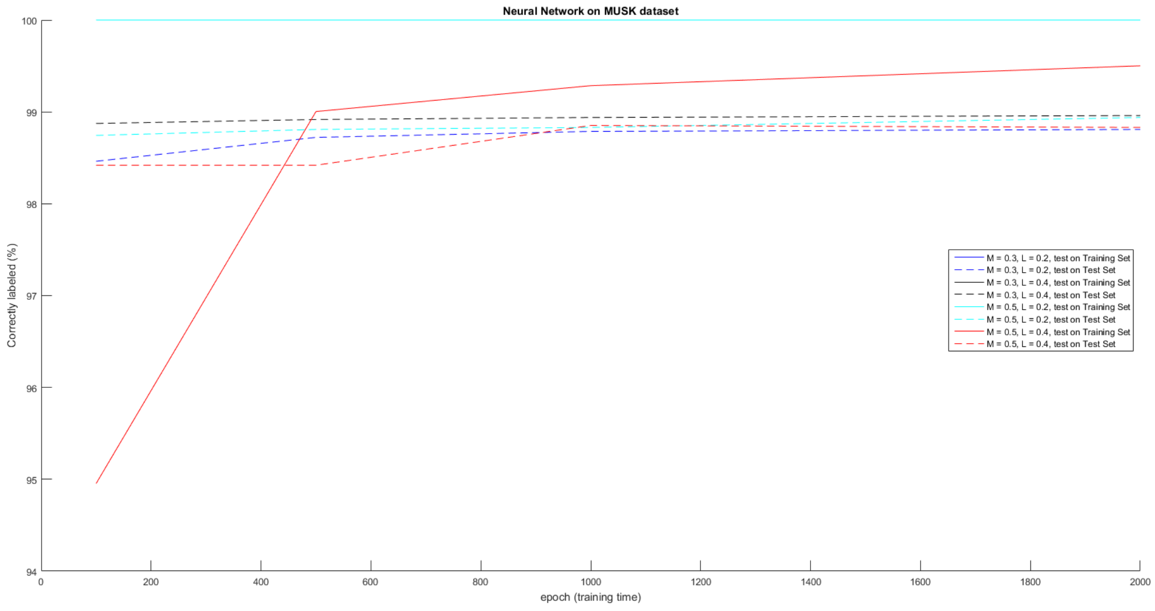
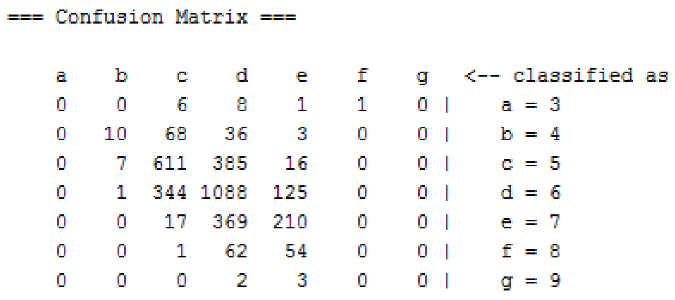
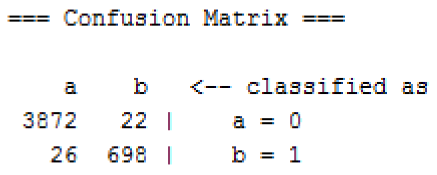


Figure 6. Neural Network on Musk dataset.



```
=== Confusion Matrix ===

   a    b    c    d    e    f    g   <-- classified as
   0    0    6    8    1    1    0 |    a = 3
   0   10   68   36    3    0    0 |    b = 4
   0    7  611  385   16    0    0 |    c = 5
   0    1  344 1088  125    0    0 |    d = 6
   0    0   17  369  210    0    0 |    e = 7
   0    0    1   62   54    0    0 |    f = 8
   0    0    0    2    3    0    0 |    g = 9
```



```
=== Confusion Matrix ===

    a    b   <-- classified as
 3872   22 |    a = 0
   26  698 |    b = 1
```

Figure 7. (a) Wine White dataset Confusion Matrix. (b) Musk dataset Confusion Matrix.

Three parameters are tuned in Neural Network including momentum, learning rate and epoch. As you can see from Figure 4-6, both datasets yield smooth curves with respect of epoch. Generally, the results soon converge after 1000 epochs. Although the lines go up and down, they do not vary beyond 5%. The detailed data including training and test time are shown in Analysis.xlsx (see README.txt). Compared with decision trees and other algorithms, Neural Network tends to yield lower accuracy for wine dataset but extremely high accuracy for musk dataset. I believe this is because the model built from wine dataset is overfitting while the musk dataset is better structured and binomial classification is less challenging. We should also notice that as shown in Figure 6 when epoch is less than 200, sometimes the test accuracy can be even higher than the training accuracy. This should be a typical case for neural network since other algorithms all yield higher training accuracy.

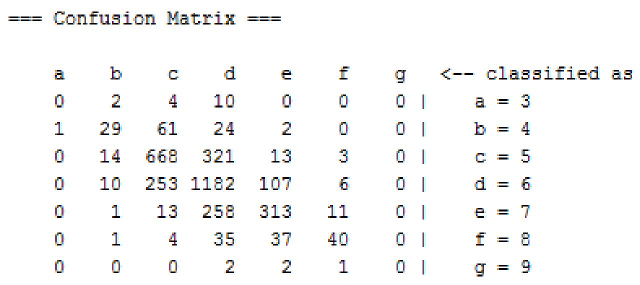
It is also interesting to note that from the confusion matrix shown in Figure 7. Compared with decision tree confusion matrix, the wine white confusion matrix from neural network have fewer instances with skewed classes (under fitting) being correctly classified. However, it does have more correct classifications from quality 5, 6 (heavy classes) which the largest number of instances are labeled as. I believed this is because neural network has a baseline quantity of instances such that those can be correctly classified. These skewed classes, especially quality 4 and 8 with less 2% instances cannot be classified at all. Of the musk dataset, although the musk class only has 15% instances, it does not degrade the performance.

The running time also epitomizes eager learner as the training time is significantly greater than the test time.

BOOSTING

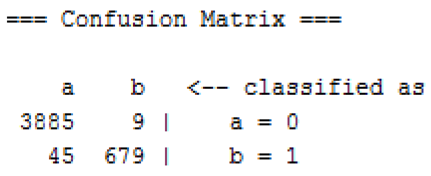
Table II
Boosting Results

Data	iteration	confidence	Training %	Test(cross-validation) %	train time s	test time s
Wine_red	10	0.125	100	63.9857	0.54	0.02
		0.25	100	64.79	0.52	0.03
		0.5	100	65.2368	0.49	0.02
		Unpruned	100	64.4325	0.43	0.03
	20	0.125	100	65.5049	0.81	0.04
		0.25	100	66.0411	0.79	0.04
		0.5	100	64.4325	0.81	0.03
		Unpruned	100	65.8624	0.82	0.04
	40	**0.125**	100	**67.7391**	1.46	0.07
		0.25	100	66.2198	1.51	0.08
		0.5	100	66.3986	1.45	0.08
		Unpruned	100	67.6497	1.26	0.08
Wine_white	10	0.125	100	62.4271	1.67	0.09
		0.25	100	63.1272	1.75	0.08
		0.5	100	62.4271	1.62	0.1
		Unpruned	100	62.9813	1.3	0.08
	20	0.125	100	63.4772	3.02	0.19
		0.25	100	64.0607	3.03	0.15
		0.5	100	63.6523	3.04	0.16
		Unpruned	100	63.098	2.43	0.2
	40	0.125	100	64.965	5.73	0.28
		0.25	100	**65.1109**	5.64	0.26
		0.5	100	64.7608	5.82	0.32
		Unpruned	100	64.3816	4.56	0.3
Musk	10	0.125	100	98.5275	17.94	0.12
		0.25	100	98.1161	17.57	0.11
		0.5	100	98.2676	16.39	0.12
		Unpruned	100	97.9212	15.46	0.13
	20	0.125	100	98.5058	41.32	0.17
		0.25	100	98.5492	37.39	0.16
		0.5	100	98.4409	38.1	0.18
		Unpruned	100	98.3543	32.76	0.17
	40	0.125	100	98.7874	77.09	0.21
		0.25	100	**98.8307**	80.5	0.22
		0.5	100	98.6791	76.99	0.23
		Unpruned	100	98.5492	74.12	0.25



```
=== Confusion Matrix ===

   a    b    c    d    e    f    g   <-- classified as
   0    2    4   10    0    0    0 |    a = 3
   1   29   61   24    2    0    0 |    b = 4
   0   14  668  321   13    3    0 |    c = 5
   0   10  253 1182  107    6    0 |    d = 6
   0    1   13  258  313   11    0 |    e = 7
   0    1    4   35   37   40    0 |    f = 8
   0    0    0    2    2    1    0 |    g = 9
```



```
=== Confusion Matrix ===

    a    b   <-- classified as
 3885    9 |    a = 0
   45  679 |    b = 1
```

Figure 8. (a) Wine White dataset Confusion Matrix. (b) Musk dataset Confusion Matrix.

Boosting is performed using the same decision tree J48 in the Decision Tree section. We tune on two parameters including iteration and confidence level of decision tree. As shown in Table II, generally more iterations improves performance but it also takes more time to train the model (eager learner). Training time increases almost linearly with respect of iterations. Interestingly, the highest accuracy of the musk dataset in Boosting has different confidence level compared with it in Decision Tree. The optimal confidence level increases from 0.125 to 0.25. Boosting primarily reduces bias (underfitting) and also variance (overfitting). Therefore, the decision tree no longer needs aggressive pruning since weak learners help reducing the variance.

Actually, the accuracy of wine white dataset increases for almost 10% and other two datasets also have decent performance improvement compared with Decision Tree. From the confusion matrix of wine white dataset, we can easily notice that more instances from both skewed classes like quality 4 and 8 and heavy classes (5, 6) are correctly classified. This is because boosting not only helps reducing variance but it also deals with bias. Compared with Neural Network, boosting also does a great job in reducing bias and variance. As you can see from the confusion matrix of musk dataset, the heavy class non-musk has only 9 samples incorrectly classified. However, class distribution does have an impact on the result. Instances from the musk class with 15 % distribution still have many incorrectly classified. Neural Network seems to have a better trade-off between bias and variance as long as it has enough data samples.

KNN

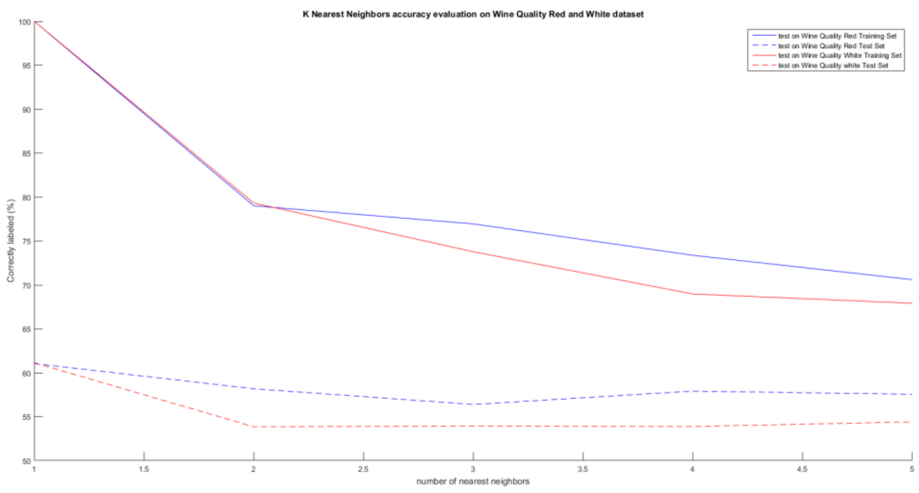


Figure 9. KNN results on Wine Quality Dataset.

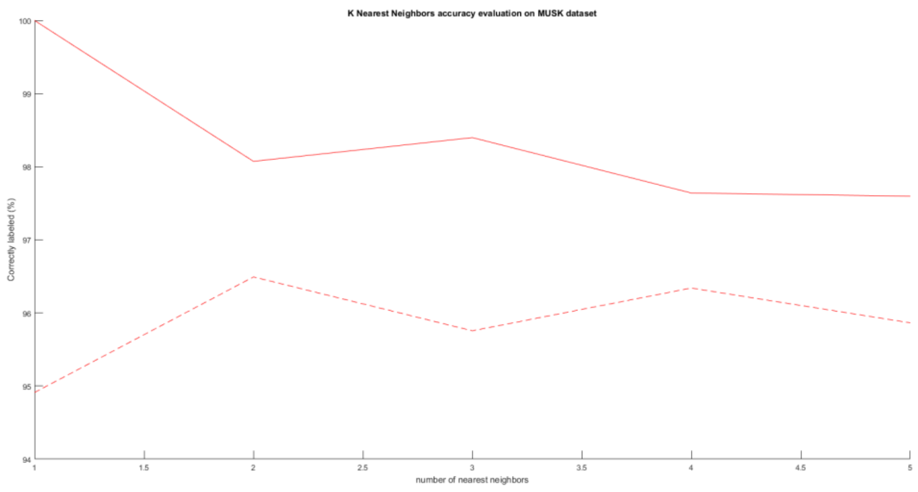
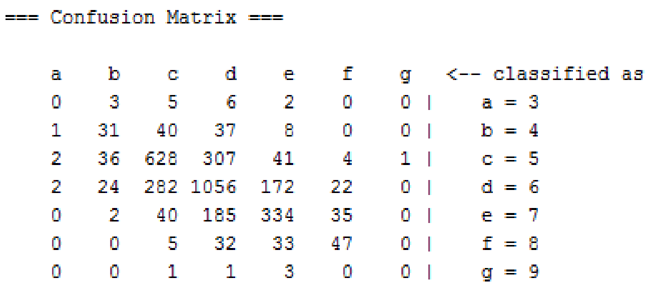
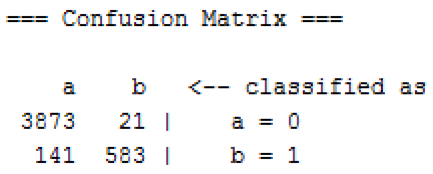


Figure 10. KNN results on Musk Dataset.



```
=== Confusion Matrix ===

   a    b    c    d    e    f    g   <-- classified as
   0    3    5    6    2    0    0 |    a = 3
   1   31   40   37    8    0    0 |    b = 4
   2   36  628  307   41    4    1 |    c = 5
   2   24  282 1056  172   22    0 |    d = 6
   0    2   40  185  334   35    0 |    e = 7
   0    0    5   32   33   47    0 |    f = 8
   0    0    1    1    3    0    0 |    g = 9
```



```
=== Confusion Matrix ===

    a    b   <-- classified as
 3873   21 |    a = 0
  141  583 |    b = 1
```

Figure 11. (a) Wine White dataset Confusion Matrix. (b) Musk dataset Confusion Matrix.

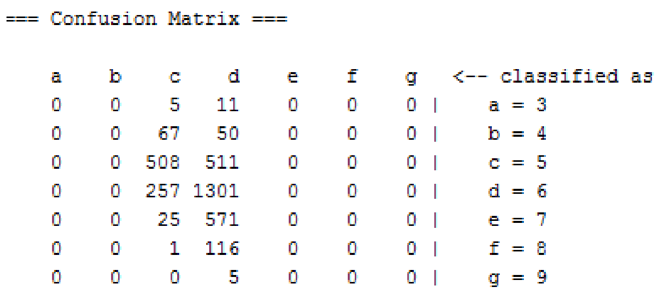
The most interesting part of KNN on these two datasets is that it only has negligible improvements in wine red and musk dataset; however, it has almost 5% improvement on wine white dataset. Since KNN is good at handling noisy data which has a great variance, KNN performs better on wine white dataset due to its relatively large amount of data. In the confusion matrix of musk dataset, many instances from the musk class are incorrectly classified. This, I believe, is because KNN suffers from skewed class distribution. Classes with fewer instances have a great possibility to be incorrectly classified. This, however, I believe can be overcome by weighing the classification, taking into account the distance from the test point to each of its k nearest neighbors. Weka does have this option, but this analysis only uses unweighted implementation. In the future, I would like to try to weight the distance and I believe it will improve the performance on the musk dataset.

The training time and test time is also interesting here. This algorithm is the only one in this analysis that has test time significantly longer than the training time. Therefore KNN is a good example of lazy learner. When increasing the number of nearest neighbors K, the test time also increases.

SVM

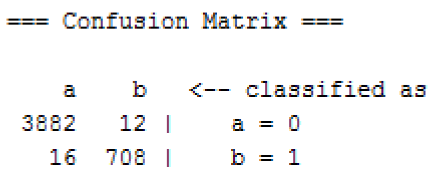
Table III
SVM Results

Data	Kernel	exponent or gamma	Training	Test (cross-validation)	train time	test time
Wine_red	**Poly**	1	60.0536	58.8025	0.16	0.03
		2	61.3941	59.9643	0.57	0.13
		3	62.6452	**60.9473**	0.71	0.22
	RBF	0.01	43.521	42.6273	1.07	0.25
		0.5	60.5004	59.6962	0.66	0.27
		1	61.7516	59.9643	0.67	0.25
Wine_white	Poly	1	52.7713	52.3629	0.79	0.04
		2	52.6838	52.1004	7.82	0.95
		3	52.4796	51.9253	17.16	2.58
	RBF	0.01	45.4492	45.4492	25.96	2.54
		0.5	52.6546	52.4212	11.02	2.89
		1	53.4131	**52.9463**	11.26	2.74
Musk	**Poly**	1	94.8463	94.7163	8.55	0.15
		2	100	**99.3937**	60.05	0.59
		3	100	99.372	38.51	0.79
	RBF	0.01	92.9623	92.5725	14.73	2.71
		0.5	98.9173	97.4015	18.22	2.84
		1	99.5019	97.3149	80.27	7.07



```
=== Confusion Matrix ===

   a    b    c    d    e    f    g   <-- classified as
   0    0    5   11    0    0    0 |    a = 3
   0    0   67   50    0    0    0 |    b = 4
   0    0  508  511    0    0    0 |    c = 5
   0    0  257 1301    0    0    0 |    d = 6
   0    0   25  571    0    0    0 |    e = 7
   0    0    1  116    0    0    0 |    f = 8
   0    0    0    5    0    0    0 |    g = 9
```



```
=== Confusion Matrix ===

    a    b   <-- classified as
 3882   12 |    a = 0
   16  708 |    b = 1
```

Figure 12. (a) Wine White dataset Confusion Matrix. (b) Musk dataset Confusion Matrix.

Two kernels are select to be analyzed in this report. One is a polynomial kernel with a varying exponent and a Radial Basis Function kernel varying gamma. There are lots of interesting things I learn from this section. The Polykernel implementation with order 2 has the highest accuracy across all algorithms and it is astoundingly close to 100%. I carefully research on this point and I find out that the Polykernel is just a way to distinguish data samples from higher orders. For example if linear data cannot be classified properly, when raised into 3-D, it might actually yield very decent results. So think about what is the data of Musk dataset? It is the conformation of molecules which is exactly in 3-D. For PolyKernel, order 1 means linear space and order 2 means 3-D. I do not believe this is a coincidence and it actually intrigues me to think about how the real-world data can be applied with corresponding algorithms which deeply related to the data in the sense of not only data structure but also the inner nature of data. As you can see from the confusion matrix of wine white dataset, the PolyKernel basically does not do anything when using only lower orders. I am curious about what would happen with an extremely high order of PolyKernel. Since I believe with higher orders the performance will converge to an optimal degree like what it shows in musk dataset, I try to use order 20 PolyKernel to have a taste of my idea. Although it yields even lower accuracy, it does classify some of the skewed classes like quality 3 and 9 correctly which has never been accomplished by all the other algorithms. Then I relates the order number to the number of classes, so I try order 7 PolyKernel. Still some data samples from quality 3 are correctly classified but the accuracy is low because most of the heavy classes are incorrectly classified. Thus, I conclude that higher order PolyKernel SVM does take skewed classes into account, but due to the run time issue it is not a proper way of using it.

When I research on RBFKernel, I find out that the gamma parameter defines how far the influence of a single record, with low values meaning 'far' and high values meaning 'close'. RBFKernel transforms data samples into Hilbert Space with Fourier Transform. This kernel can be used in multi-dimensional problem but it does not address these two datasets properly.

With higher exponent or gamma value, the training time does not monotonically increases. There are some other issues relates to the run time. When running order 2 PolyKernel on musk dataset, although it yields the higher accuracy, the run time is relative higher than that with order 1 and 3. The training time is generally larger than the test time, and therefore SVM is also an eager learning algorithm.

LEARNING CURVE

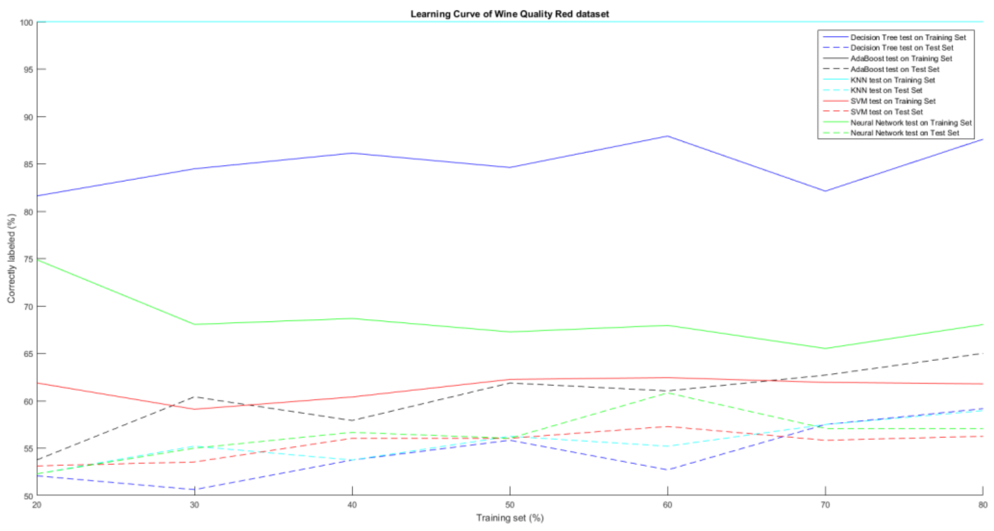


Figure 13. Learning Curve of Wine Quality Red Dataset.

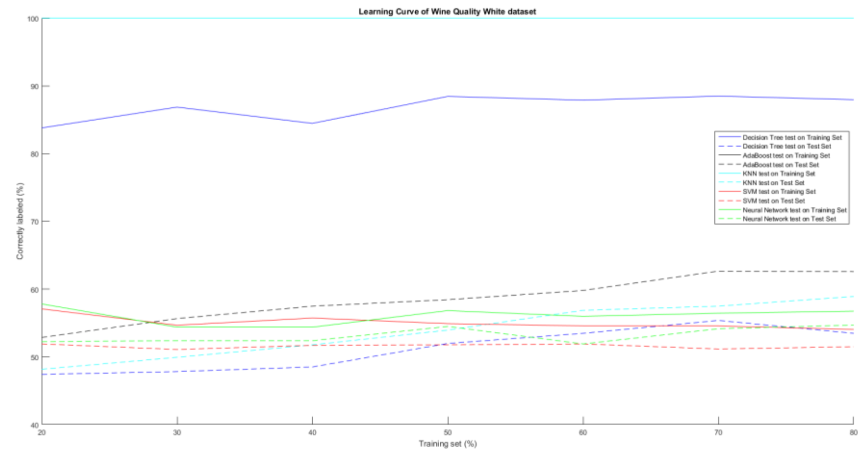


Figure 14. Learning Curve of Wine Quality White Dataset.

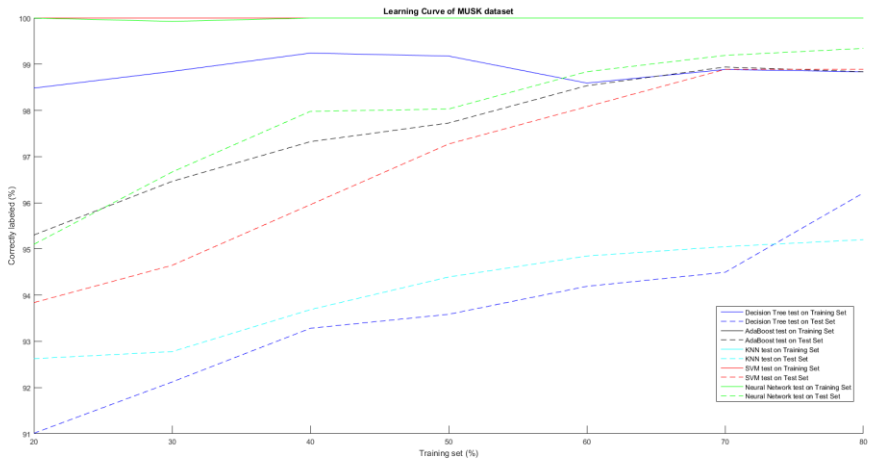


Figure 15. Learning Curve of Musk Dataset.

The most important thing I learning from drawing these learning curve is that increasing the amount of data you use to train the model does not necessarily give you better model so that when you test it, you can have better performance. As you can see from Figure 13-14, none of the curves increase monotonically. It is, I believe, partially true under several assumptions. First, the new data should not create extremely excessive variance (overfitting) since more data means more random error. Then, your test model should not be so general that even adding the new data could not describe the model completely which in other words underfitting. Finally, the new data you use should have inner structure similar to the old data in order to fit in the algorithm and its parameters you choose.