Sagar Laud CS 4641
Homework 1

Supervised Learning

Datasets:

Voting:
The voting dataset contains votes for each of the US House of Representatives on 16 key issues as identified by the CQA (Congressional Quarterly Almanac). This dataset was provided with my download of weka and has been included in the submission archive. There are 17 attributes in total with 16 being issues that were voted on (with possible values of each attribute being yes or no) and the 17th being the class attribute (democrat or republican). There were 435 instances in this dataset: one for each representative in the House. For my purposes, I used a 66/33 split to train and test the algorithms with 66% of the original dataset going towards training and 33% of the original data going towards testing.

Wine Quality:
The wine quality dataset was created by using physiochemical data from Portuguese wine samples. This data was acquired from the UCI database and has been included in the submission archive. The entire premise for the data was to use these attributes to determine the quality of the wine without needing to necessarily sample it. For our purposes we will be working only with the red wine portion (the original dataset had both red and white wines present but the white wine portion was very small). Overall, there are 1599 instances each with 11 attributes. The first 10 attributes are numeric values representing different chemical properties of the wine. The final attribute is the class attribute which ranges from 3 to 8 (with 3 being bad wine and 8 being good wine). For my purposes, I used a 66/33 split to train and test the algorithms with 66% of the original dataset going towards training and 33% of the original data going towards testing.

Interest:

Voting: The voting dataset has significant ramifications in terms of political knowledge. In Congress votes are power and drive all major decisions. The ability to classify a group of people into liberals and conservatives based simply upon their votes on certain issues is powerful. This allows us to detect dishonesty in the system as we can now predict what party the representative is in based upon his votes. If he is said to be in the wrong party for example, we can see that he is not voting along party lines for everything.

Wine Quality: The wine quality dataset has significant ramifications in terms of commercial usage. By being able to determine the quality of the wine based on these chemical parameters, vineyards would have the ability to produce higher quality wine by focusing on certain aspects of their product.

Machine Learning:
From a machine learning standpoint, both of these datasets are interesting but for different reasons. The wine dataset has varying performance on all of the algorithms but has a tough time cracking 60% accuracy on any of them. On the other hand, the voting dataset has a very easy time on all of the algorithms. The voting dataset did not really overfit for any algorithms which was impressive given the small size of the dataset. The datasets are also interesting because the number of attributes is actually greater in the voting dataset over the wine quality dataset. However, because the attributes in the voting dataset have binary values, it does not seem to suffer from the curse of dimensionality as the wine quality dataset does when run through the algorithms.

Decision Trees:

Voting						
Pruned	Confidence %	Training %	Testing %	Leaves	Size	Training Times(seconds)
Yes	0.1	97.5945	93.0556	4	7	0
Yes	0.2	97.5945	93.0556	4	7	0
Yes	0.3	97.5945	93.0556	4	7	0
Yes	0.4	97.5945	93.0556	4	7	0

Yes	0.5	97.5945	93.0556	4	7	0
No	N/A	97.5945	93.75	8	15	0
Wine Quality						
Pruned	Confidence %	Training %	Testing %	Leaves	Size	Training Times(seconds)
Yes	0.1	86.5403	48.8971	105	209	0.03
Yes	0.2	89.1943	45.7721	129	257	0.03
Yes	0.3	91.6588	45.4044	154	307	0.03
Yes	0.4	91.8483	44.8529	156	311	0.03
Yes	0.5	91.8483	44.8529	156	311	0.03
No	N/A	92.3223	44.6691	168	335	0.03

In the voting dataset, the results that we received were somewhat expected. This dataset almost seems perfect for something like a decision tree and it shows in the results. Each of the attributes in the voting dataset has one of two values: whether or not the representative voted yes or no on an issue. It stands to reason that the votes that are made by representatives typically lie among party lines. As a results, on almost all issues, republicans will vote one way while democrats will vote the other way. Because decision trees rely on information gain per attribute, there is quite a lot of information to be gained with every attribute as the decision is almost always divided equally between republicans and democrats. It is no surprise that with pruning, the size of the decision tree was only 7 with only 4 leaves, this is a testament to the amount of information gain per attribute. Additionally, with representatives voting predictably and our class attribute only having two possibilities (republican and democrat), the J48 algorithm seemingly had a very easy time classifying these instances without overfitting.

Looking at the wine dataset, we see that the opposite of the voting dataset has happened here. We can clearly see signs of overfitting as the training % always increases while the testing % always decreases when we reduce our amounts of pruning. This is probably as a result of increased pruning working to remove some of the overfitting that has occurred in our algorithm by removing branches that have simply “memorized the data.” The wine dataset also performed far worse than the voting dataset. We see from looking at the confusion matrices which are in the result folders that potential reasons for this include that the classifier guesses the more common classes more often, simply because there are more instances of certain instances of the class attributes than other. There is also quite a bit of overlap in the attributes of multiple classes making a classifier's job even tougher. In addition, the fact that the classifier has to classify using a class attribute with 6 possible values may be an issue as well. This is a far more difficult task than classifying into simply republican or democrat.

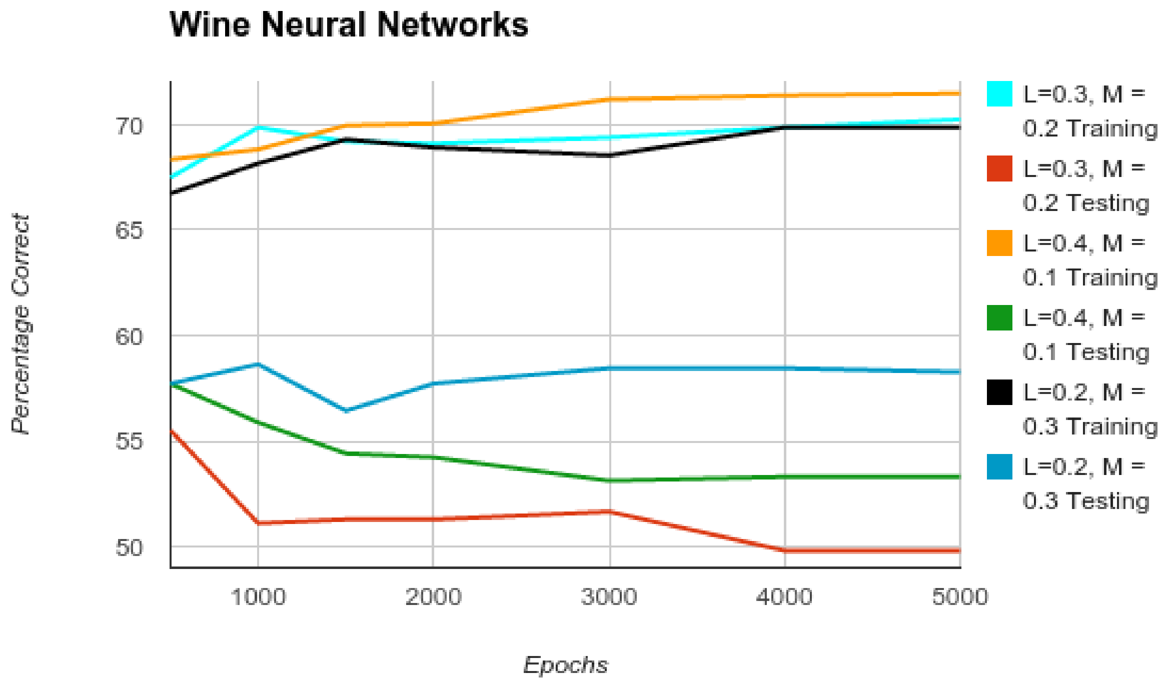
Boosting:

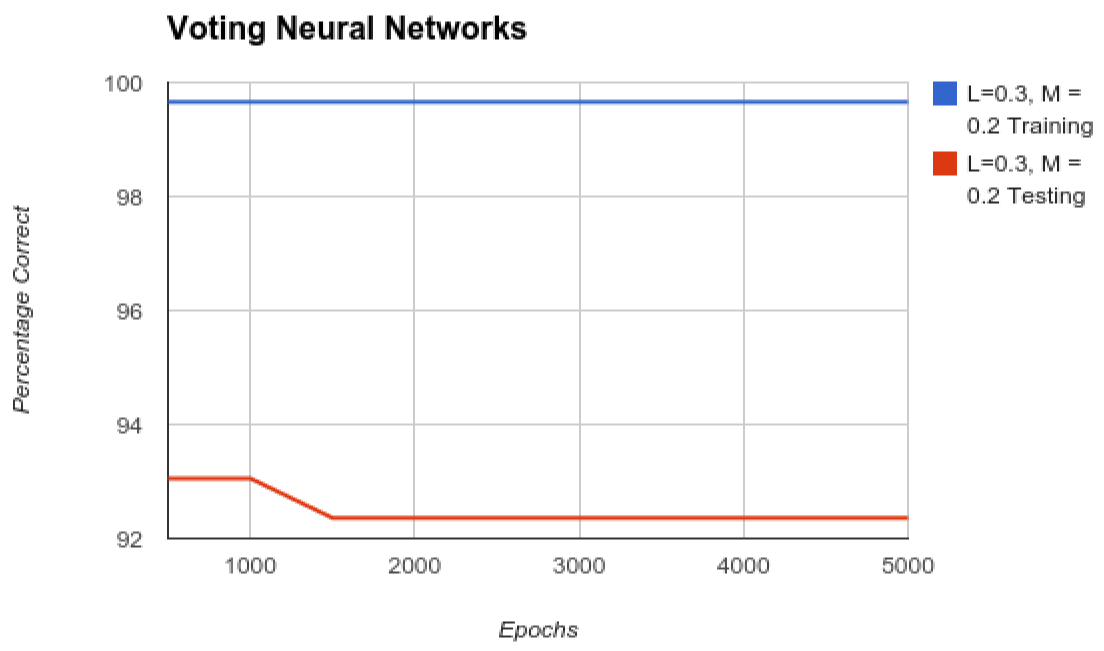
Voting							
Pruned	Confidence %	Training %	Testing %	Leaves	Size	Iterations	Training Times (seconds)
Yes	0.1	100	95.8333	10	19	10	0.01
Yes	0.2	100	95.8333	1	1	10	0.01
Yes	0.3	100	95.8333	5	9	10	0.02
Yes	0.4	100	95.1389	1	1	10	0.2
Yes	0.5	100	95.8333	10	19	10	0.01
No	N/A	98.2818	95.8333	8	15	10	0.01
Wine Quality							
Pruned	Confidence %	Training %	Testing %	Leaves	Size	Iterations	Training Times(seconds)
Yes	0.1	100	51.4706	124	247	10	0.33
Yes	0.2	100	48.5294	117	233	10	0.33
Yes	0.3	100	52.0221	121	241	10	0.32
Yes	0.4	100	52.2059	114	227	10	0.32
Yes	0.5	100	50.9191	127	253	10	0.34
No	N/A	100	51.8382	122	243	10	0.27

Boosting was done by augmenting the J48 algorithm with AdaBoost. The results from this are interesting. The results for both datasets improved. In fact, looking at the results shows that the evidence for overfitting has decreased in the wine dataset. The results for the wine dataset also improved quite a bit. It seems that boosting worked reasonably well because the number of attributes allowed the algorithm to separate the data points in an adequate manner. However, it was not able to complete conquer the dataset. I believe this to be simply because there is a lot of overlap in the dataset itself. There are attributes that are very similar that map to two completely different wine qualities. It seems that this overlap causes some issues with the boosting algorithm.

This section provided the best results for the voting dataset. I believe this to be simply because of how boosting works. As shown in the decision tree algorithm, the results were already quite good. However, there were some nuances which may not have been picked up by a typical classifier. But due to the process involved in boosting, the classifier was able to pick up on these nuances. As a result, the boosting algorithm was able to remove some of the error in this regard (simply because the attributes had binary values with seemingly little overlap).

Neural Networks:



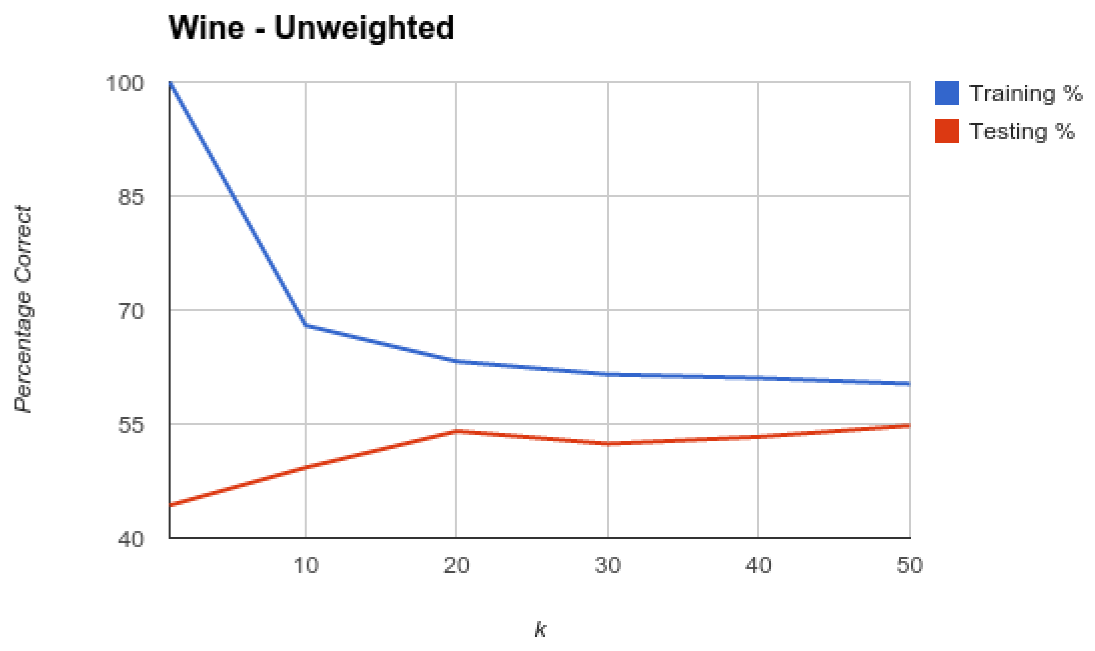


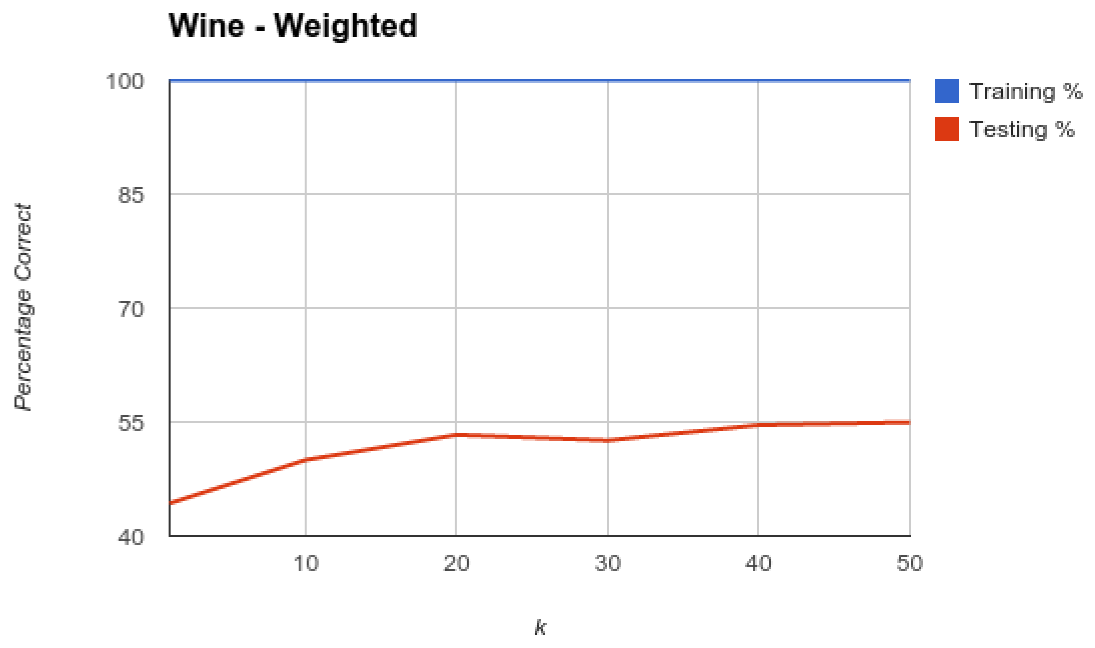
The results for Neural Networks were very interesting. Training times naturally varied depending on the dataset. For the wine dataset, the shortest training time was around 2 seconds while the longest training time was around 23 seconds. For voting the training time was roughly between 0.7 and 7 seconds. However on both occasions, the time to test the trained model was much less than training time. For the wine dataset it was less than a second and for the voting dataset, it was less than 0.4 seconds. This property serves to show us that we are dealing with an eager learner.

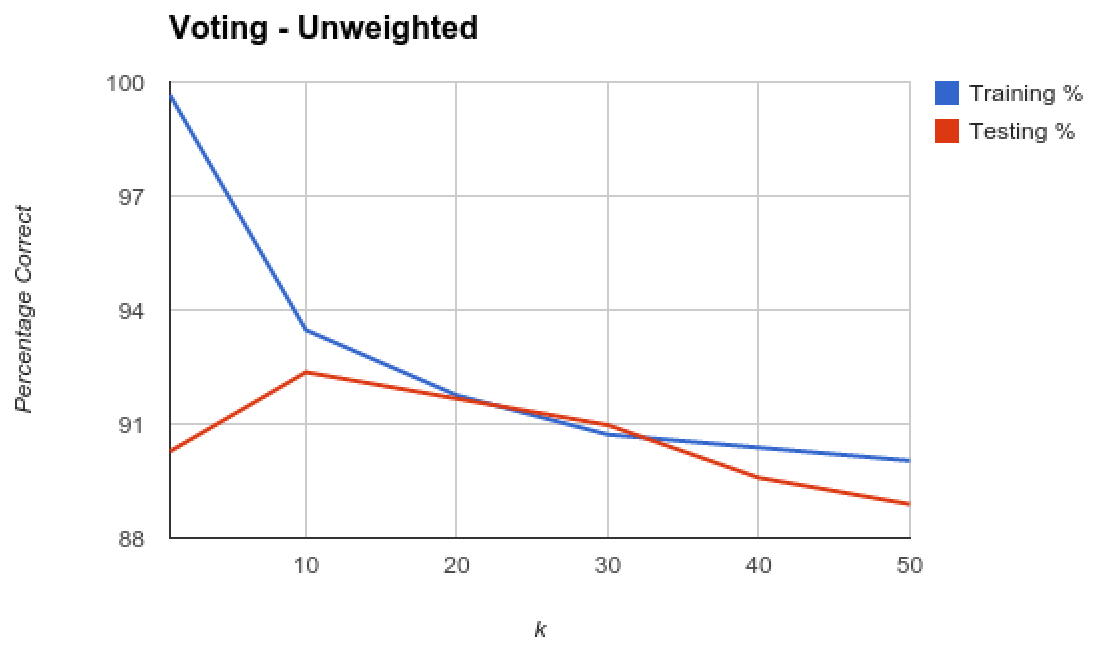
Looking specifically at the results from the wine dataset, we see that the variations in momentum and learning actually do make a difference in some cases. We see that if we use the default options, L=0.3 and M=0.2, overfitting begins to occur as the number of iterations increases as the lines for training and testing times begin to diverge. We see the same for L=0.4, M=0.1. In fact, this variation causes it to overfit far sooner which seems to be a result of increasing the learning constant. The interesting result is what happens we set L=0.2 and M=0.3. In this case, it actually does not seem to truly overfit by the time we hit 5000 epochs. This is wonderful as it seems with a lower learning coefficient, even with a decent amount of momentum, it overfits less because it learns less quickly. In fact, not only does it not seem to overfit but out of all of the runs with different parameters, this set of values produced the best result for Neural Networks, again most likely a result of it not overfitting. It is worth noting that overfitting did not occur immediately, at the very least due in some part to the continuous nature of the attributes.

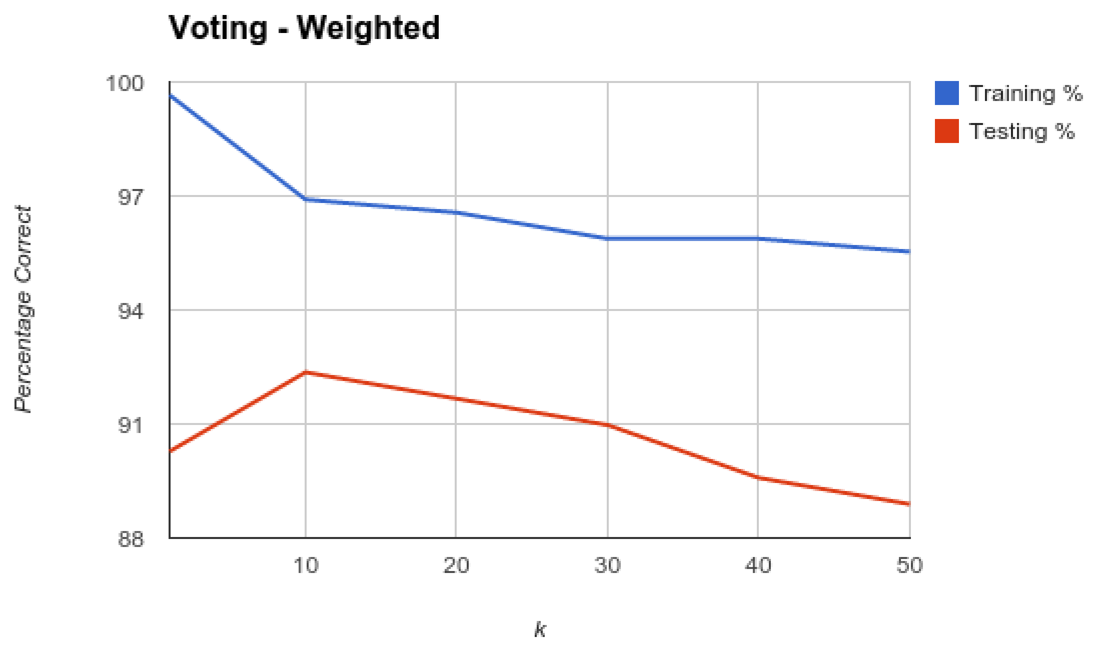
The voting dataset provided some very different results. The results more or less remained constant (which is why the results from other varying parameters have been omitted, they are still available in the Neural Networks folder however for viewing). The training accuracy did not change, there was always one element which was misclassified. Upon inspection, this actually seems to be because one datapoint, a republican, did not vote on any issues. As a result, the algorithm had nothing to classify it with and as a result always made an incorrect decision in that classification. The high level of consistency can also be attributed to the nature of the attributes. The votes themselves are actually indicative of the class attribute, in that politicians are predictable. In addition, the attributes for this dataset have discrete values which makes learning a little more simple. Finally, there are a small number of data points with which to classify so increasing the number of epochs did not have a drastic effect (the testing % was constant, changed once, and then was constant again) on the result seemingly because after training on a small number of points, there really wasn't much more that it could learn in the later iterations.

kNN:


Wine - Unweighted
Training %
Testing %
Percentage Correct
100
85
70
55
40
10
20
30
40
50
k


Wine - Weighted
Training %
Testing %
Percentage Correct
100
85
70
55
40
10
20
30
40
50
k


Voting - Unweighted
Training %
Testing %
Percentage Correct
100
97
94
91
88
10
20
30
40
50
k


Voting - Weighted
Training %
Testing %
Percentage Correct
100
97
94
91
88
10
20
30
40
50
k

For the wine dataset, the kNN algorithm produced very good results. It does not seem that there is any major evidence of overfitting here. In addition, as k increased, so too did the results for the dataset. Naturally the progression in the graphs for the test set looks very similar, but looking at the data (available in the data.xlsx file), we see that in the end, the weighted kNN algorithm yielded slightly better results for the wine dataset. This makes sense if analyzed simply because it makes sense for the nearest neighbors to be more likely to contribute to the result than some arbitrary data point which is a neighbor simply because it is at the edge of a boundary. It is also possible that not all of the features are relevant to the classification problem.

The IBk algorithm produced identical results on the voting dataset whether the kNN algorithm was weighted or not. What is worth noting in this case is that the accuracy increased steadily up until a certain threshold for k. Once it passed this threshold, the accuracy steadily dropped and by the time it reached the largest value of k it produced the worst accuracy for the voting dataset across all algorithm. This makes sense. I believe this to be simply because the algorithm started clustering multiple classes together as k increased to the point that instances from both classes were in the cluster. This would result in misclassification which would explain the results.

SVM:

Wine Dataset			
PolyKernel	Exponent	Training %	Testing %
	1	57.8199	56.25
	2	58.8626	56.0662
	3	63.0332	58.4559
	4	63.7915	57.7206
	5	65.4976	58.8235
RBF	Gamma	Training %	Testing %
	0.01	42.2749	37.6838
	0.25	58.1991	56.0662
	0.5	58.5782	56.25

	0.75	59.2417	56.8015
	1	59.6209	56.9853
Voting Dataset			
PolyKernel	Exponent	Training %	Testing %
	1	98.2818	93.75
	2	99.6564	93.0556
	3	99.6564	93.75
	4	99.6564	93.75
	5	99.6564	93.75
RBF	Gamma	Training %	Testing %
	0.01	95.189	93.0556
	0.25	98.2818	94.4444
	0.5	98.9691	95.8333
	0.75	99.3127	93.75
	1	99.3127	91.6667

SVMs seem to be the overall best method for the wine dataset. There is one outlier in terms of data, and that is for the RBF Kernel where gamma is 0.01. However, we also see that the training percentage dropped here too so a gamma value this low just was not proper for the dataset. Looking at the rest of the data, we see that we had a very consistent set of results. The highest percentage correct for the wine dataset in the testing set appeared in this section in the PolyKernel section. It is worth noting that the results do appear to start showing signs of overfitting in the wine dataset with a PolyKernel, as the accuracy in the training set is always increasing while the value in the testing set is fluctuating. In both the PolyKernel and RBF Kernel, the final run of the algorithm resulted in the best results overall for the dataset in this particular section. It is possible that the increase in accuracy was the result of the SVM algorithm splitting the attributes into more clearly separated hyper-planes by maximizing the gap between different classifications. The results from the RBF Kernel were as expected on the wine dataset. As gamma increases, the SVM algorithm limits the influence of a given data point more and more. In our dataset, the data itself is highly clustered. So, in this case it makes sense to influence only datapoints which are close to a given data point as opposed to datapoints which fall on the border between two classifications.

For the voting dataset, the results were relatively consistent regardless of the kernel or exponent/gamma used. However, using an RBF Kernel with a gamma of 0.5 resulted in the percentage correct tying the previous maximum (Boosting tied this mark). The accuracy quickly declined after but I believe this to be the result of the same issue that we ran into in the kNN algorithm. Adjusting the value for gamma determines how much weight each training example carries. It stands to reason that just as in kNN where our results declined after a certain threshold, we have found the threshold to be 0.5 for an RBF Kernel in SVM most likely due to different classes ending up influencing each other.