Nick Robinson
Supervised Learning Report

Data Overview

Wine Rating Dataset:

The first dataset chosen for this assignment was from UC Irvine's Machine Learning datasets available here. The dataset chosen relates to the Portuguese "Vinho Verde" wine and the ratings that a panel of wine experts gave to a white version of this wine. The following physical properties are provided for each different wine sample:

fixed acidity	volatile acidity	residual sugar	chlorides	free sulfur dioxide
total sulfur dioxide	density	pH	sulphates	alcohol

Each wine was given a rating by a panel of three wine experts and the median of their score was recorded for the data. This dataset provides a great opportunity for using some of the algorithms that we have discussed in this class in order to attempt to predict the correct rating given physical properties of the wine. Since most of the data in this dataset are in the 5-7 range ranking wise it is valuable to be able to recognize wines that would be viewed as particularly favorable by wine experts as these wines are typically worth more money.

Number Recognition from Images

The second dataset used in the assignment came from the website kaggle.com. Each entry in the dataset is made up of 784 (28x28) 8-bit pixel values and the number to which these pixels correspond. The dataset includes over 40,000 hand drawn number pixel values. This dataset provides an opportunity to determine which of the supervised learning algorithms seen so far perform best with a real world problem of identifying text in images. This type of machine learning has a large application in trying to defeat CAPTCHA style authenticators used on many websites in order to prove that a user signing up is actually a human.

More information on the particular dataset can be obtained here.

Decision Trees

For the Decision Tree portion of the assignment I decided to use the Random Forest algorithm from the sklearn package. Random Forests seek to fix the problem of high variance in a particular decision tree by averaging an ensemble of trees. This averaging helps to reduce the variance of the overall estimator which helps prevent the overfitting scenario that we would see in a typical decision tree.

In the interest of time I pruned my dataset down to 5000 randomly chosen samples.

Random Forests are currently one of the most popular black box methods around. The downside to this being a black box method is that the output does not provide clear insight into the decisions made by the algorithm

Number Classification Data Set:

For the number classification problem I varied the number of estimators which amount to the number of trees in the forest from 1-100. Since the point of a Random Forest is to reduce the overall variance of the computed decision tree we would expect that as the number of estimators increases that the overall correctness (Calculated as the percentage of correct predictions) would increase as well.

The figure below shows the number of estimators on the x-axis with the correct prediction percentage on the y-axis. Overall the Random Forest algorithm performed very well on this dataset. Thinking about the problem objectively for a moment sheds some light on why the results turned out so well (i.e ~95% correct percentage). We know that there are certain regions of the pixel map that when combined with others can do a good job of ruling out certain numbers. Since the algorithm has all of the available data about the pixelmap available to it it is able to do a good job of classifying the data.

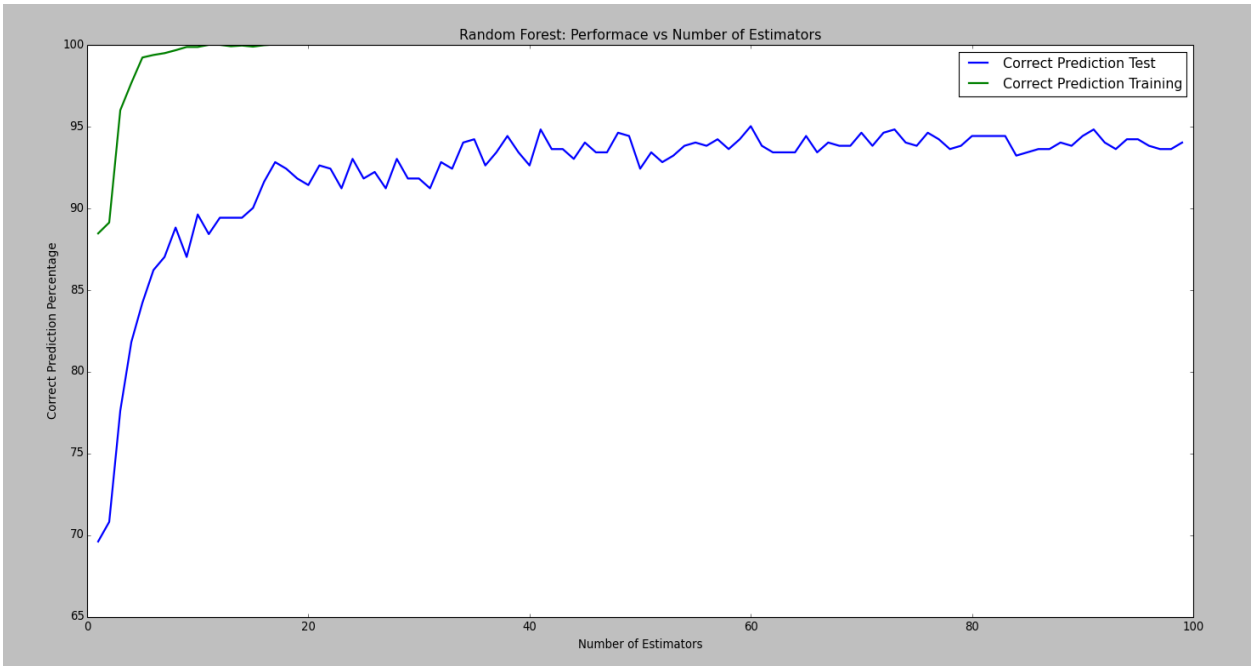


Figure: Random Forest with Number of Estimators vs Correct percentage for Number Image Dataset

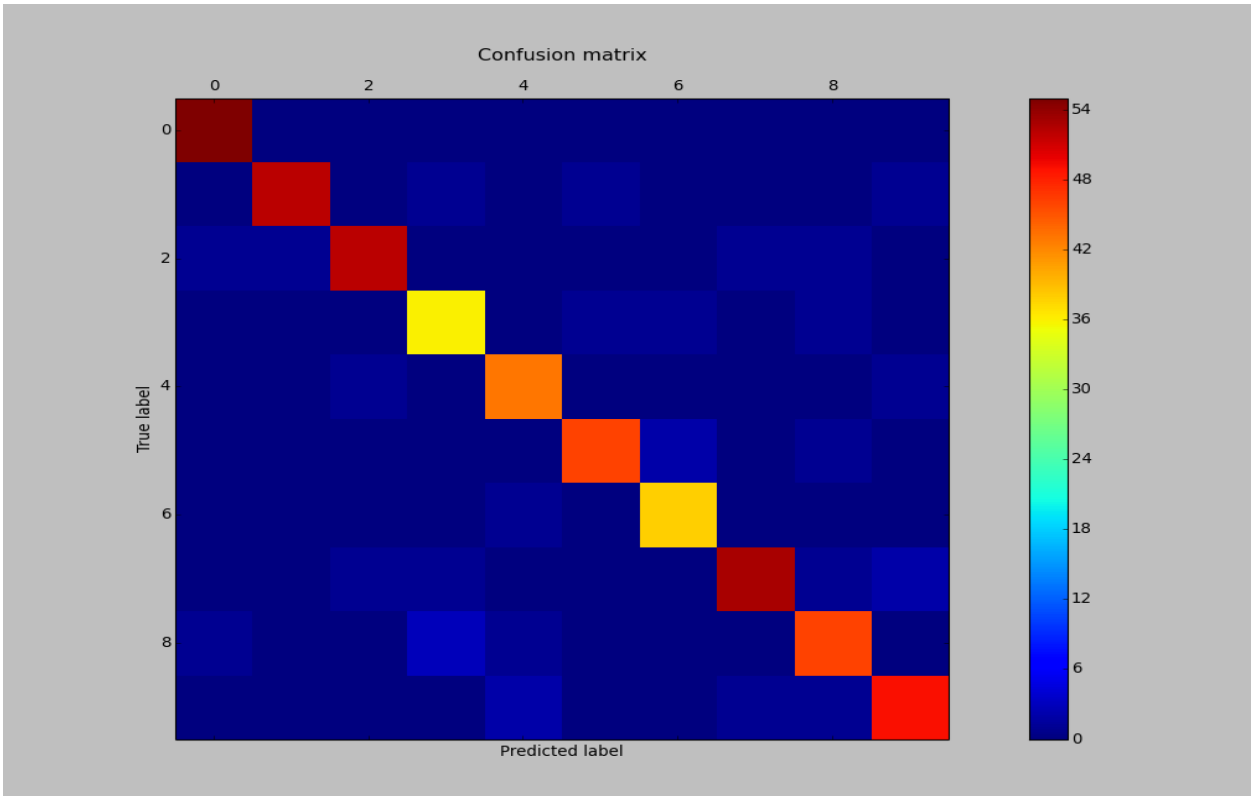


Figure: Confusion Matrix for Random Forests run against Number Image Dataset

Wine Quality Dataset:

I used the same settings as the image data for the Wine Quality dataset and the results were less promising. Overall the performance of Random Forests against this dataset hovered around 70%. This is actually a relatively good performance since we can assume there are physical properties of the dataset that we are missing in our data.

The confusion matrix shown below indicates that we seem to do well with the wines that are in the middle of the pack but tend to struggle to identify wines at the extremes quality wise.

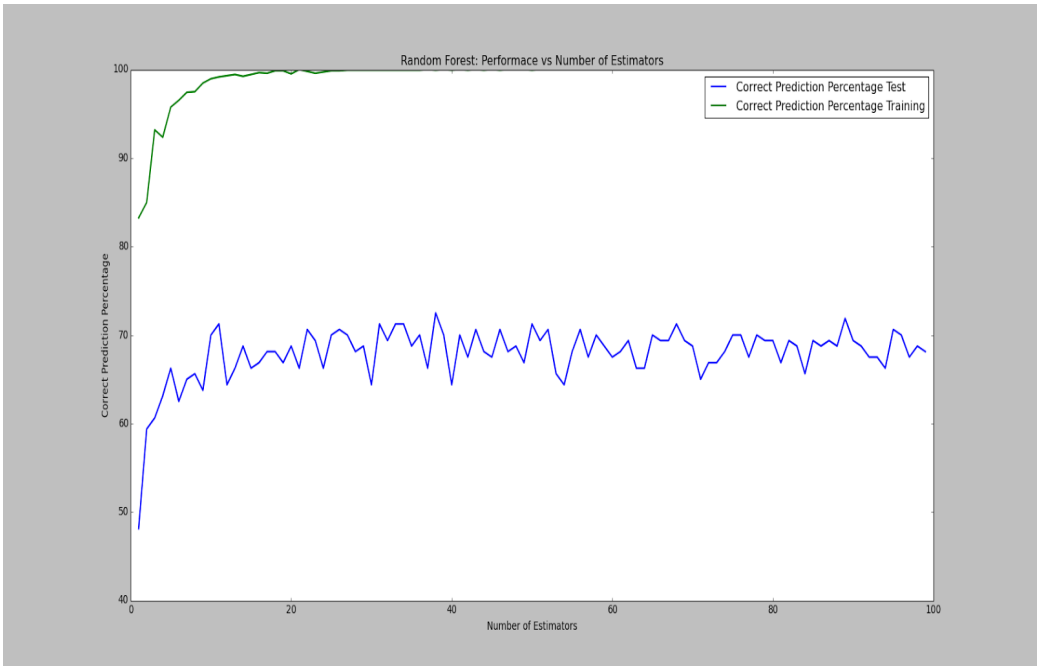


Figure: Random Forest Number of Estimators vs Correct Percentage for Wine Quality dataset

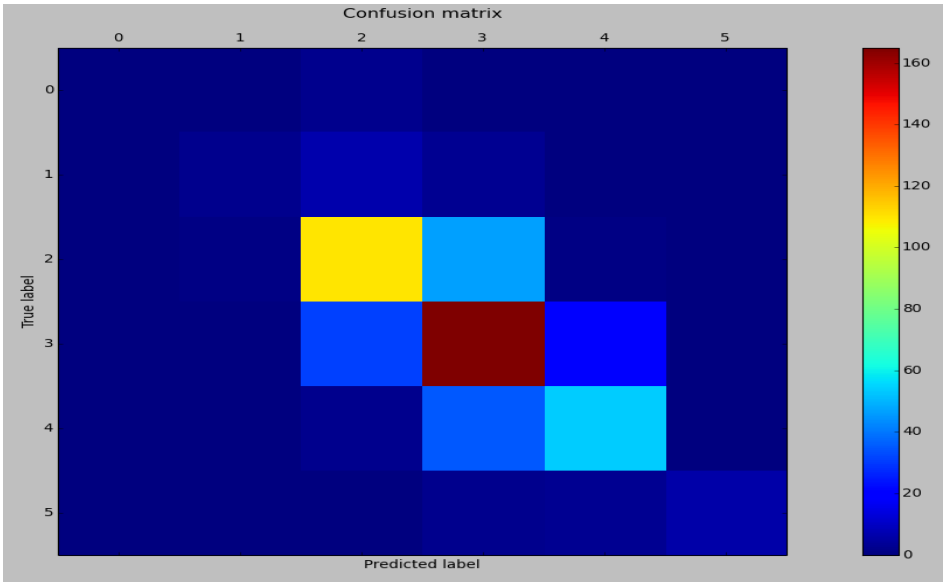


Figure: Random Forest Confusion Matrix for Wine Dataset

Neural Networks:

For the Neural Network part of the assignment I chose to use the Pybrain Library. The trainer I chose to use for testing my data was a Back Propagation trainer. This type of trainer will perform an error calculation at the end of each cycle and then adjust the weights at each layer of our network then run the data through again.

Overall the numbers from the Neural Network algorithm fell in line with the other algorithms. The digit dataset ended up getting around 70% correctness while the wine dataset got a correct percentage of around 55%.

I am not sure why the digit dataset performed worse with neural networks but I believe part of the problem was with the complexity of the library I chose in order to perform the testing.

Boosting

For the Boosting part of the assignment I used the sklearn Gradient Boosting algorithm. I varied the number of learners which equates to the number of boosting cycles that the algorithm will run. Since Gradient Boosting is relatively strong against overfitting I chose to go all the way up to 400 boosting cycles.

Wine Data:

For the wine dataset the algorithm performed relatively well with the maximum percentage correct falling around 65%. This was a good result and boosting performed as well overall as any of the algorithms used. My takeaway on why the percentage is not higher relates back to the dataset itself and its failure to capture additional physical properties that a wine taster would take into account.

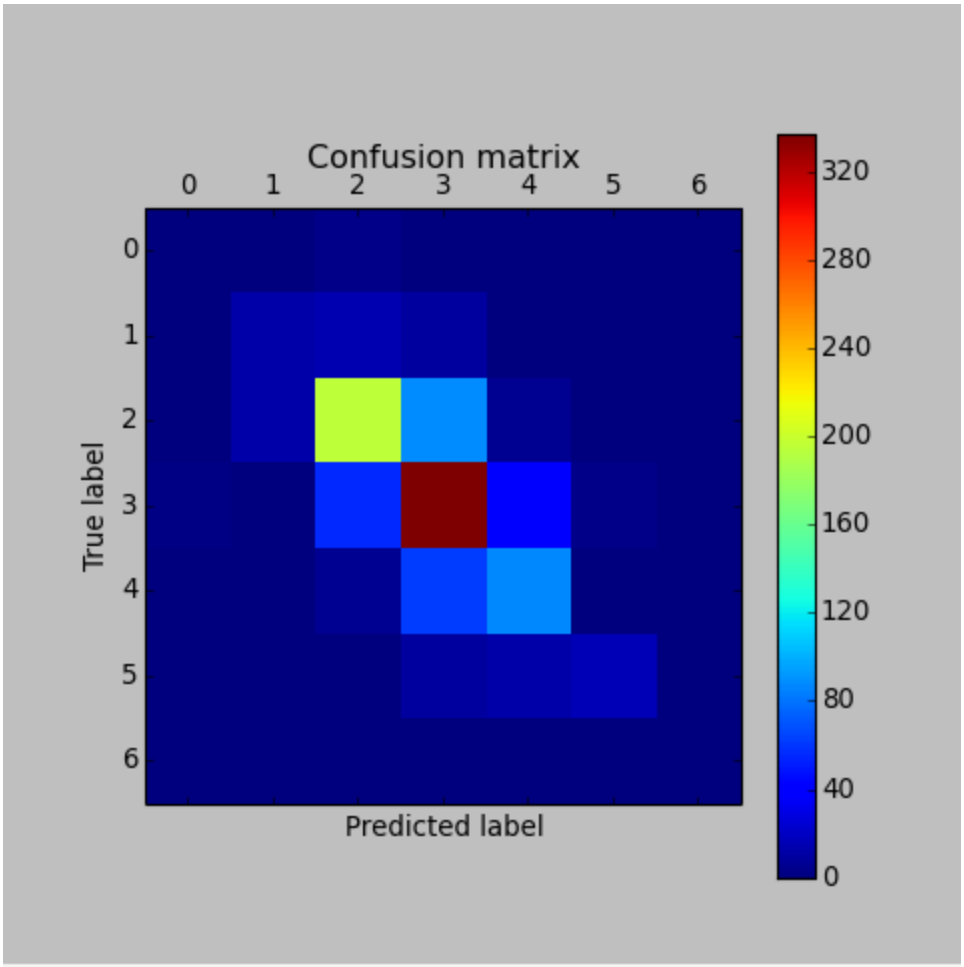


Figure: Confusion matrix for wine data with 400 learners

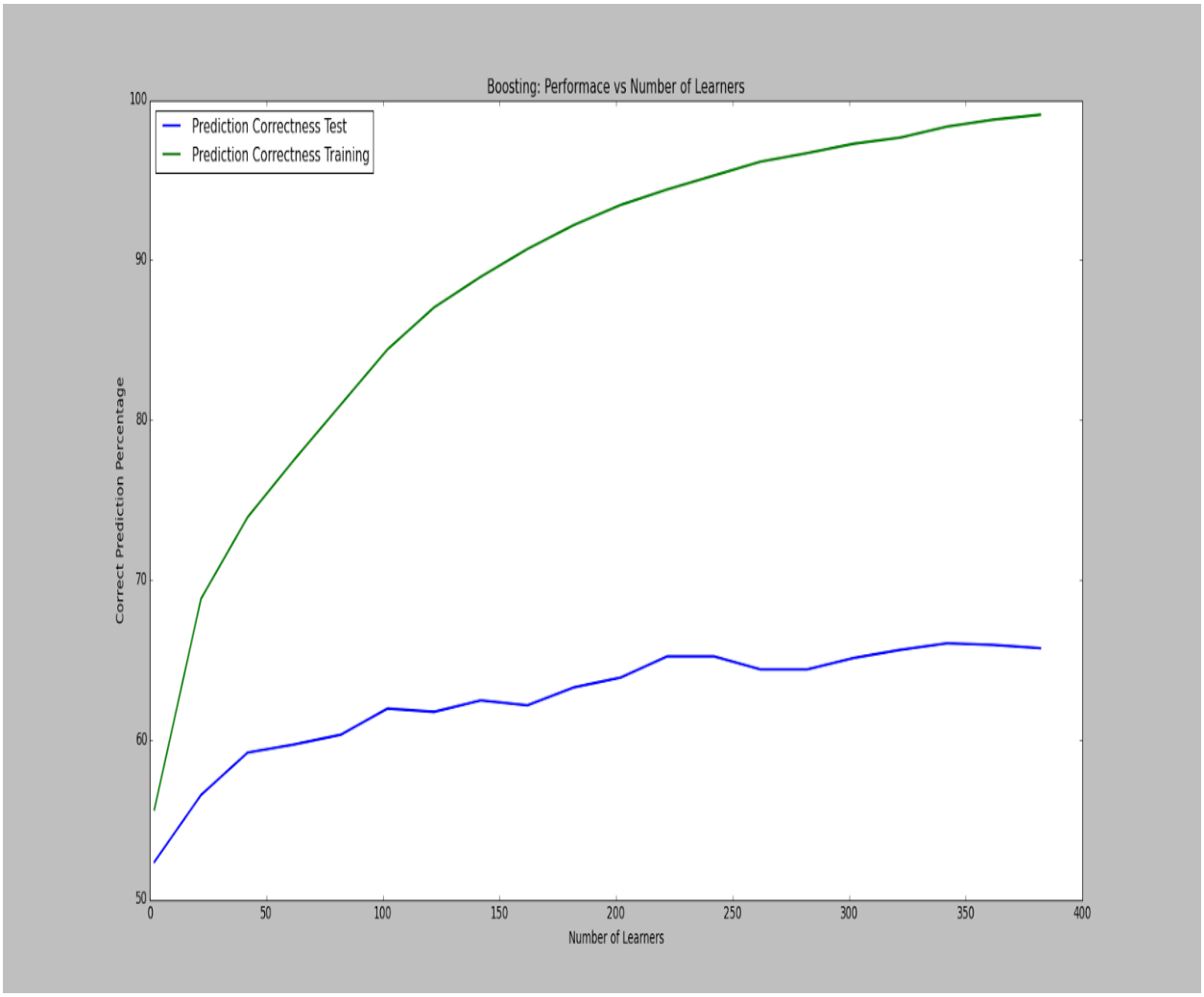
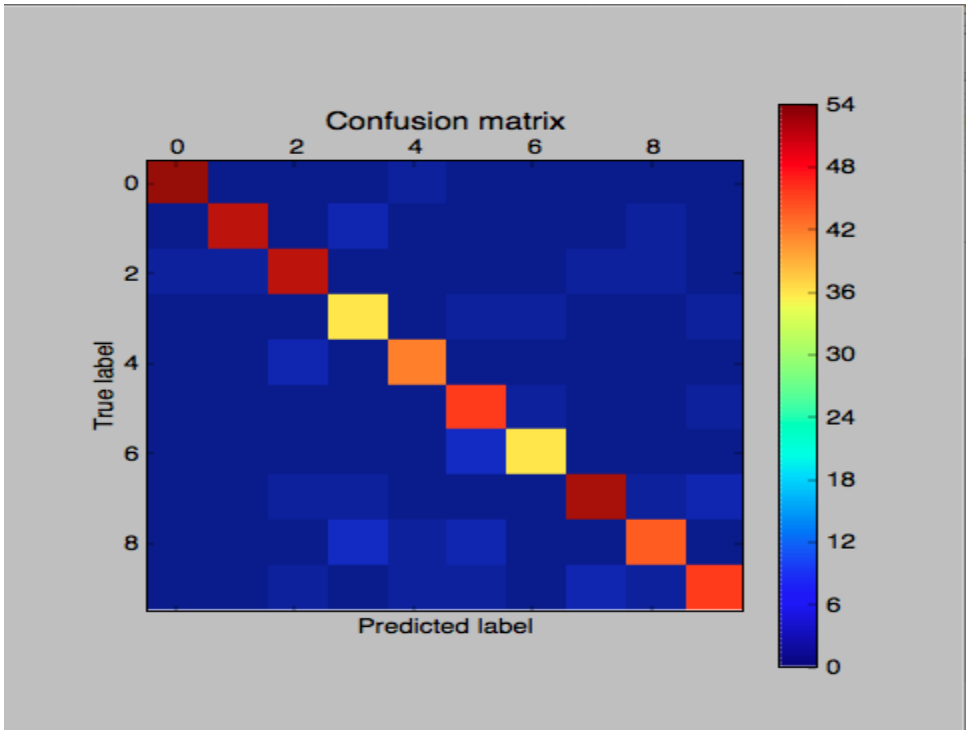
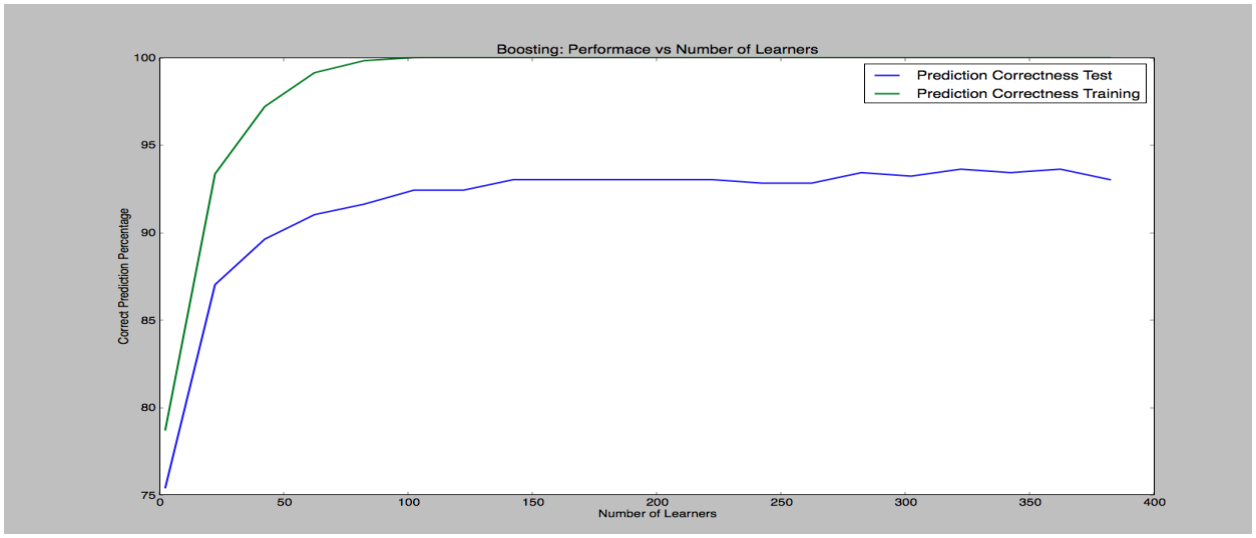


Figure: Number of Learners vs Correctness Percentage for Wine Data

Digit Data:


Confusion matrix
0
2
4
6
8
True label
Predicted label
54
48
42
36
30
24
18
12
6
0


Boosting: Performace vs Number of Learners
Prediction Correctness Test
Prediction Correctness Training
Correct Prediction Percentage
Number of Learners
100
95
90
85
80
75
0
50
100
150
200
250
300
350
400

KNN:

In order to test my data sets with a KNN algorithm I used python based scikit. The questions with KNN is what is the appropriate number of neighbors to use in order to get the best prediction results. Another interesting setting that one can apply is the weight measurement for what truly classifies a close neighbor. For my testing scenario I varied the number of nearest neighbors and used pyplot to graph the results.

Image Data:

The numerical image dataset is the first dataset I used my KNN program against and overall it performed astonishingly well. I took 5000 samples at random from the initial 40,000 set sample in order to be able to run the algorithm in a reasonable amount of time. I ran my KNN program with 20 different values for number of neighbors and used two different weighting algorithms.

The takeaway from this program was that KNN performs very well against this type of data. This makes sense if one thinks about the algorithm itself for a bit. KNN is designed to assign some label or classification depending on other samples that look similar to yourself. Since most of the numbers drawn in the images would have groups of pixels located very close to similar instances we would expect good performance.

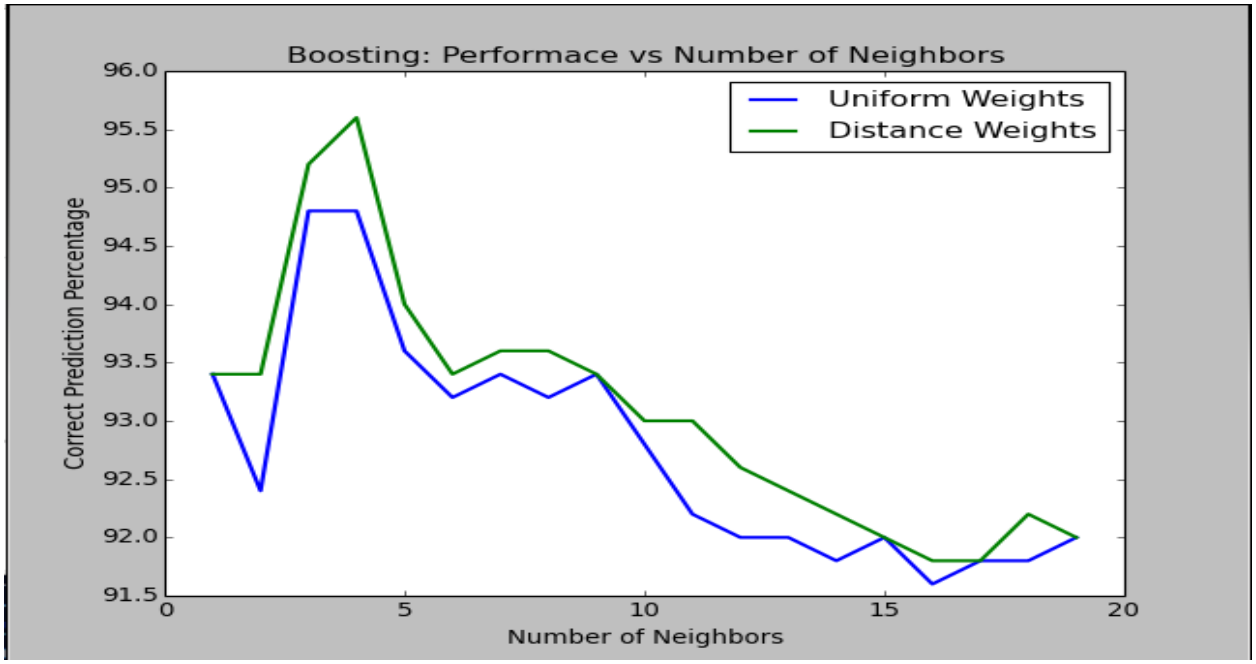


Figure: KNN vs Correct Percentage with Uniform and Distance based weights for Numerical Image Classification

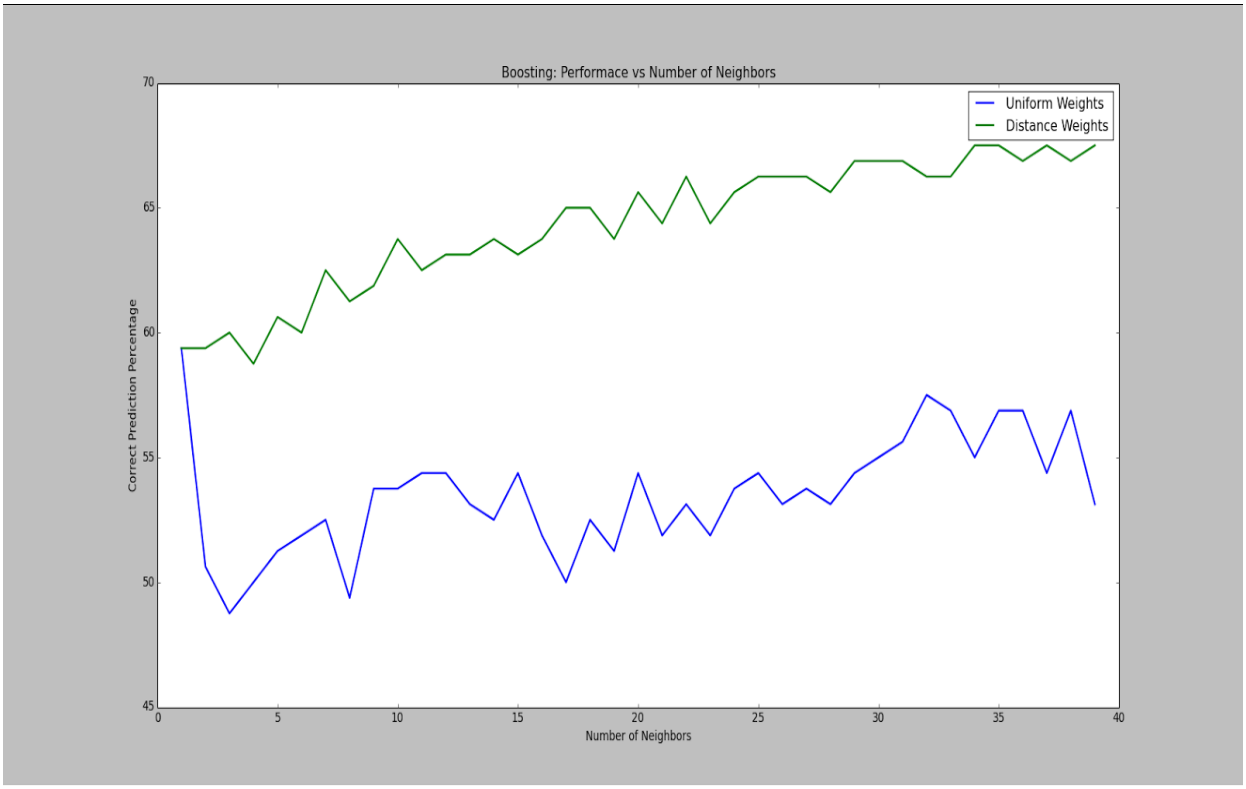


Figure: Number of Neighbors vs Correct Prediction percentage Wine Quality Classification

SVM

For the support vector machine part of the assignment I used SciKit's implementation of Support Vector Machines with a few different kernel options. For the image dataset the default parameters provided excellent while results of the SVM algorithm were less helpful for the wine dataset.

Thinking about how the SVM algorithm typically works sheds a bit more light on the situation. Since at we can map each image bitmap to a higher level plane where linear separation is more possible it is quite likely that we can get decent results given our data.

One surprising result from the Image dataset is how bad the rbf kernel performed. We would typically expect this kernel to outperform the Linear kernel as its function space is much larger but for this particular problem it does not.

Number Images Dataset

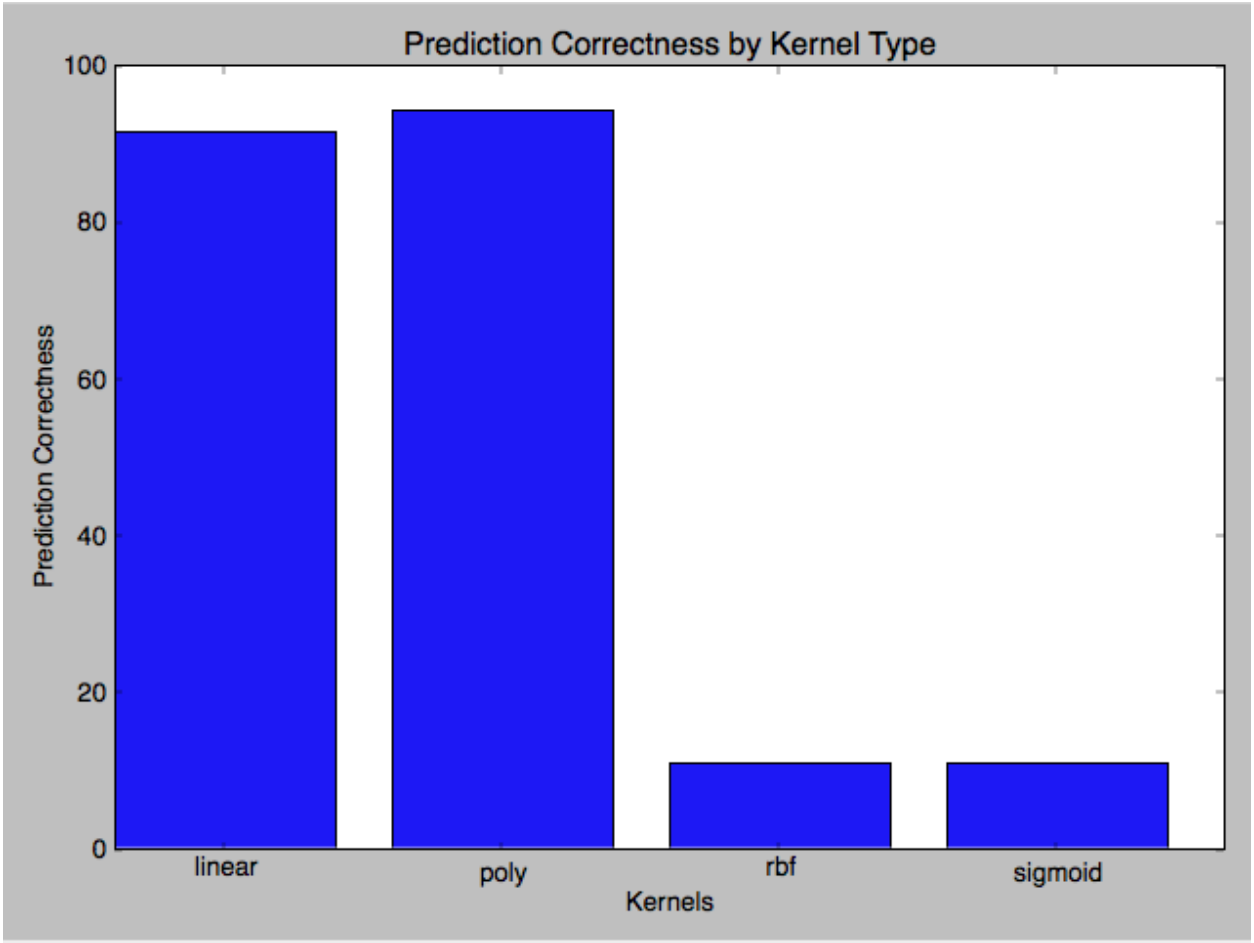


Figure : SVM Kernel type vs correctness for Image data

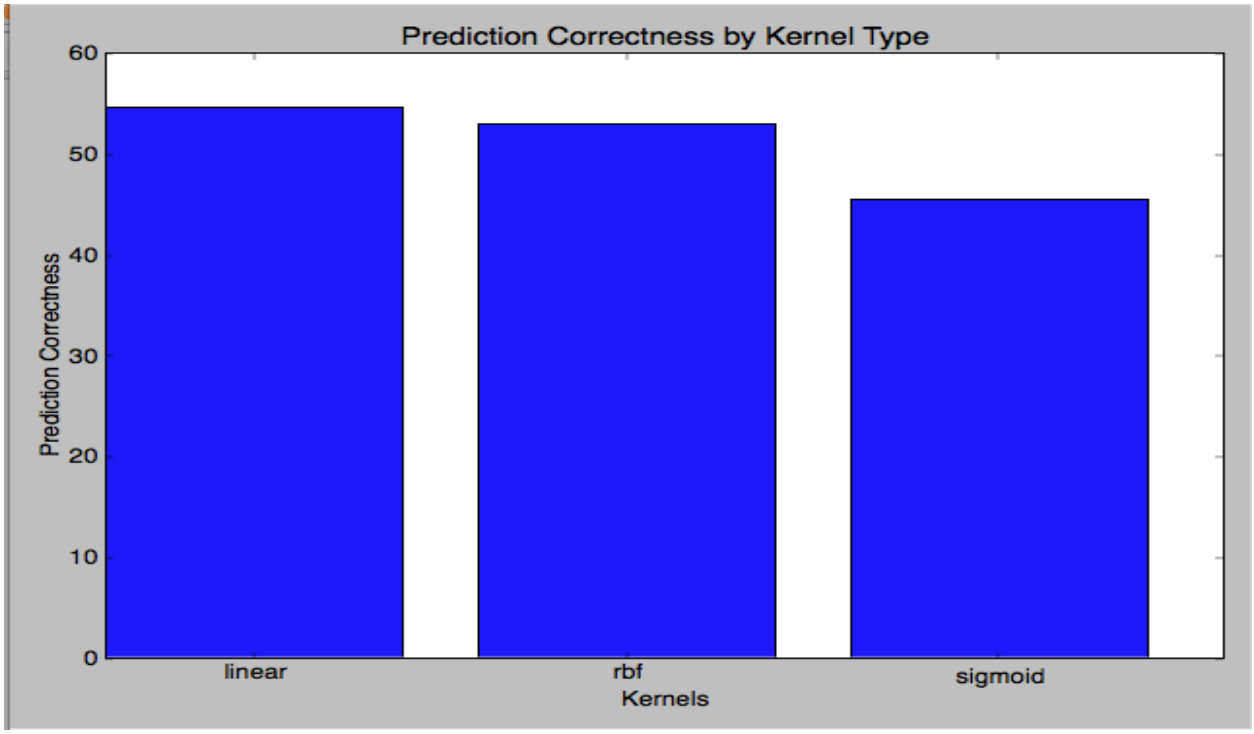


Figure : SVM Kernel type vs correctness for Wine data

Conclusions:

There were a few takeaways from the assignment that should be noted. Overall it was clear that most of the algorithms seen in this assignment can be tweaked to provide pretty good results for most classification problems run across. From a computational complexity vs correct percentage it would appear from the data in this assignment that KNN provides the most “bang for your buck” in terms of information gain. This has real world implications in that if you have the storage space available to you may be better off to use a simple algorithm like KNN vs dealing with the computational overhead of the other algorithms.

Another inference from the data seen in this assignment is the importance of having all necessary data available to the algorithm. In the case of the wine quality ratings the algorithm performance was consistently less than the same algorithms performance on the number dataset. My conclusion is that while many of the physical properties of the wine were available in my dataset there are many more physical properties of the wine that appear to be influencing tasters decision on a wines overall quality. With more physical properties available in the dataset I feel as if the performance of each of the algorithms could improve significantly.

Overall this assignment provided a great opportunity for trying out an array of supervised learning algorithms. It provided a nice opportunity to weigh the pros and cons of many algorithms we have seen in the class so far.