# Assignment 1: CS 7641

Matthew Kirk

February 14, 2014

## Introduction

As an avid food lover I have grown a special affinity towards mushrooms and wine. I love Morrels, Chantrelles, Boletes, and many more. Since I live in the Northwest where we have a plethora of mushrooms to be had in the forest, I've always wanted to go mushroom picking. But unfortunately I don't know much about which ones to pick and know that if I pick the wrong one I'll probably end up dead or with ill effects from psychedelic mushrooms.

Wine isn't as dangerous to consume but the one thing that always bothers me is that when we're at a store buying wine there is a 50/50 chance you'll get bad quality wine. Sometimes you can end up with a 10 dollar bottle of wine that tastes better than the 100. But unless you drink all of the bottles in between how would you know the difference? Surely there is a better way.

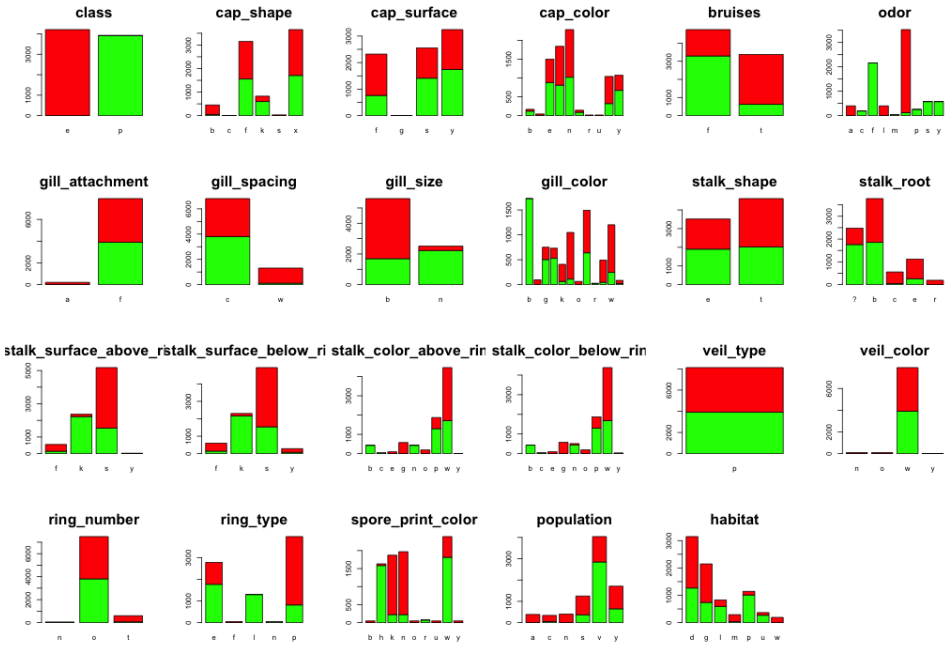
That is why I have decided to apply supervised learning algorithms to mushroom picking in the forest and wine selection. More specifically I want to classify whether mushrooms are poisonous and whether a wine is above average or not based on characteristics. The different methods I will use to analyze these problems are K Nearest Neighbors, Support Vector Machines, a Neural Network, Boosting, and a pruned decision tree (in this case we'll use J48, or the java version of C4.5). But before we delve into the analysis portion we need to first explain what the data looks like and where it came from.

## Data acquisition

Both data sets come from the UCI Machine Learning repository.

## Mushroom Data Set

The first data set comes from the Audubon Society Field Guide to North American Mushrooms (1981). In it there are 8124 instances of mushrooms being either poisonous or edible. There are 22 different attributes that are captured in this data set. Instead of listing them out let's look at what the distribution of each one is. What is intriguing is that the distribution of attributes if mapped in histogram form with 'red' being poisonous and 'green' being edible.

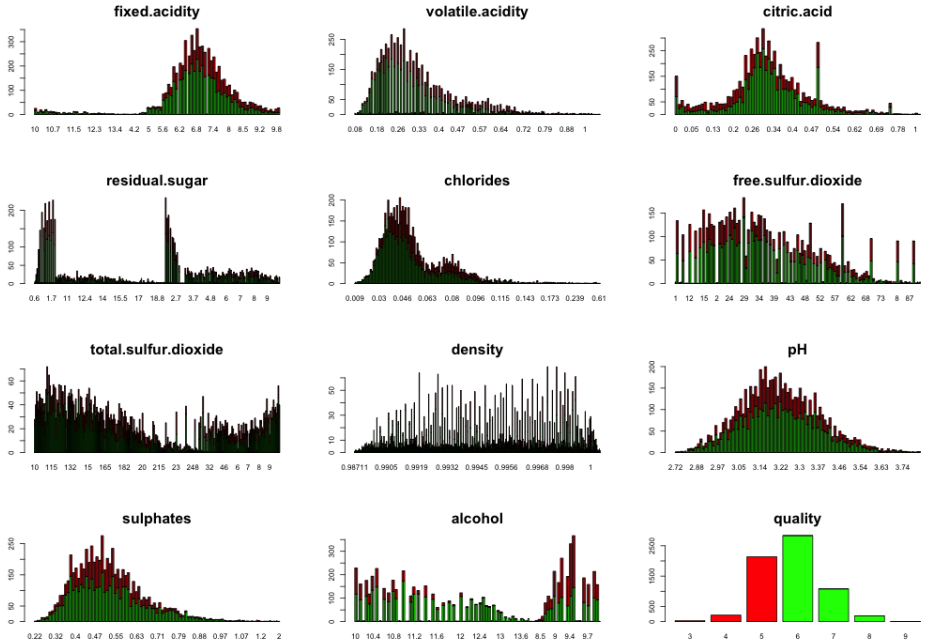



You can see that there is an obvious attribution of 'odor' to poisonousness. This makes the dataset intriguing and maybe able to actually grep some useful analysis out of.

## Wine Data Set

The wine data set comes from a study done by Paulo Cortez at the University of Minho. This includes two data sets of the red and white variants of 'Vinho Verde' wine. These data sets came with 12 original attributes but we need to pre-process the data to better fit our classification problem. If you remember the problem is about classifying the above average wine and we have two data

sets (red and white). To achieve this we added a new attribute 'red' which is simply a binary variable that is either 't' or 'f'. On top of that we took out quality which was in the original data set and used above_average to denote any wine that received strictly greater than a 5 out of 10. In summary our data looks like this.




The only thing that I can see is that it seems that wines in the middle of distirbutions on alcohol tend to be higher quality wines. This would make since because high alcoholic wines tend to be cheap, and so do wines that are low in alcohol or 'watered down'. Now that we know a bit more about the data let's analyze it using some supervized learning methods!

# Tools Used and Methodology

To accomplish all of these algorithms I used Weka 3.6 and JRuby 1.7. Weka because it simplifies building classification problems and JRuby because it simplifies the development time. There is more about this in the README. Implementation wise the tooling is built into algorithms, some helper classes as well as the data directory.

For each algorithm (KNN, SVM, etc), I take a test centric approach by

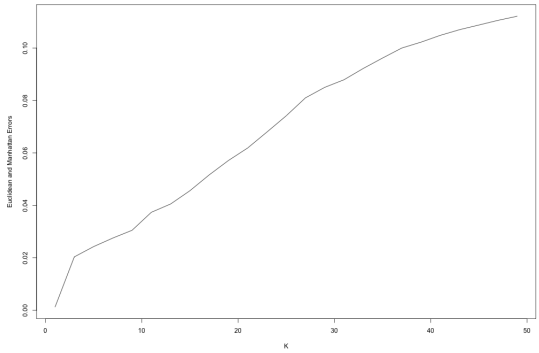
running cross validation using 10 folds. There is one exception to this rule but we'll get to that in one of the next sections. The purpose of running cross validation first is to build models that will withstand future information as it gets added over time. We want to actually test how the model works with regards to data outside of the model opposed to just memorizing inputs.

# K-Nearest Neighbors

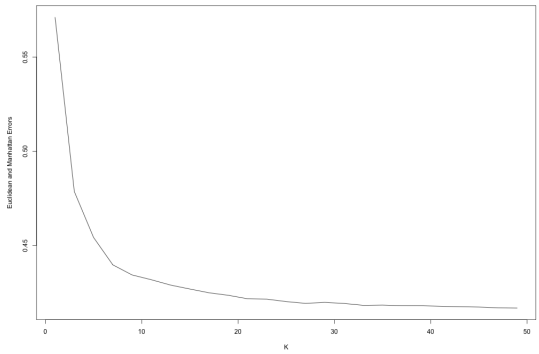
With K-Nearest Neighbors classification you have a data set where given a query point and given the closest $k$ points to that point $q$ you vote on what is the most common classification. The domain knowledge of K-Nearest Neighbor is what distance function you use, and what K.

Given that picking $k$ and a distance function is domain knowledge, I chose to test the following models. For $k$ I would test each using a 10-fold cross validation for the odd numbers from 1 to 49. As well I wanted to test the difference between manhattan and euclidean distances.

What resulted was a reasonable classification on mushrooms and a mediocre one with wine. What happened for the mushroom classification problem is what you'd expect from a KNN problem. As K gets bigger so does the error. This has to do with the curse of dimensionality as well as comparing too many mushrooms to make a classification. While $k = 1$ has the lowest root mean squared error I think that $k = 3$ is the best choice. Because if given a classification problem where you are trying to separate out two classes it's best to at least have a tie breaker. The data which came from cross validation trying all of those K combinations and euclidean and manhattan looks like this. Notice that euclidean and manhattan distances result in the same error rate decay.



What happened with the wine dataset was different. As $k$ went up the error went down, which signals to me that KNN is not a good solution for classifying wine in this manner. The reason for that is if the error is going down and $k$ approaches the instance count then we are not derriving any induction out of it and looking at the entire data set which doesn't solve the problem! Instead if $k = n$ where $n$ is the entire data set (in this case around 8000) then we'd just be averaging the entire data set. It seems that KNN does not model this very well.



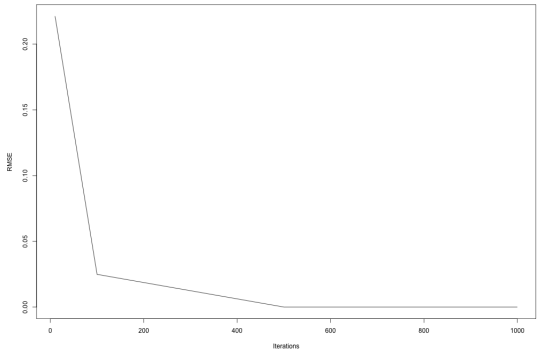

While KNN is a simplistic algorithm it did well with mushroom classification. This is due to the fact that mushrooms that are poisonous are similar in nature. For instance odor seems to be a big determiner we saw earlier in the distribution of attributes. The wine data set did not fare as well because wine classification isn't as similarity based. While a wine might share similar characteristics it may or may not be an above average wine. Chemically wine is pretty similar even though one might be a Bota Box while the other is a Grand Cru.

# Boosting

The AdaBoost adaptively determines the best fit through iterations. The downside to AdaBoost is using too many iterations and overfitting the data, but a lot of work has been done with this and shows great performance. AdaBoost doesn't have a lot of tweaks to be had like KNN does, and most of the work can be done by setting how many iterations you want to go through.

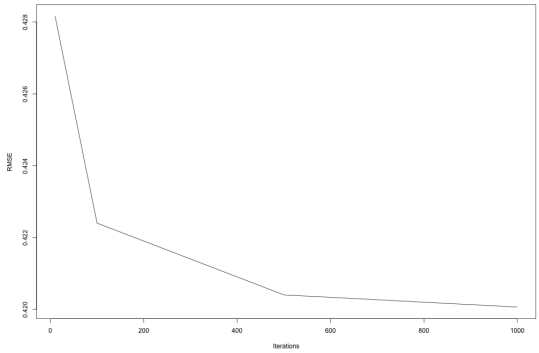
So I decided to try [10, 100, 500, 1000] which gives a decent hypothesis set of AdaBoost algorithms. Running this against both data sets yields good answers in a quick period of time. Again the mushroom data set worked well

against this algorithm. It went from a 0.22 RMSE to 0.02 between 10 and 100 iterations and then dropped off after that.




This is indicative of the underlining distribution that we observed earlier where there is a definite pattern to the attributes of the mushrooms.

On the other hand the wine data set didn't improve much over iterations. Between 10 and 100 iterations it dropped 0.006 RMSE from 0.428 to 0.422. At 500 it seems to have found it's asymptope at 0.420. This is very similar to the fact that the distribution we observed on the wine data set is noisy and doesn't have as simple as a pattern as the mushroom dataset does.




Overall though the AdaBoost algorithm is exceptional because unlike the KNN algorithm it runs fast. This is the quickest one to run on my Mac Book Air and yielded good results.

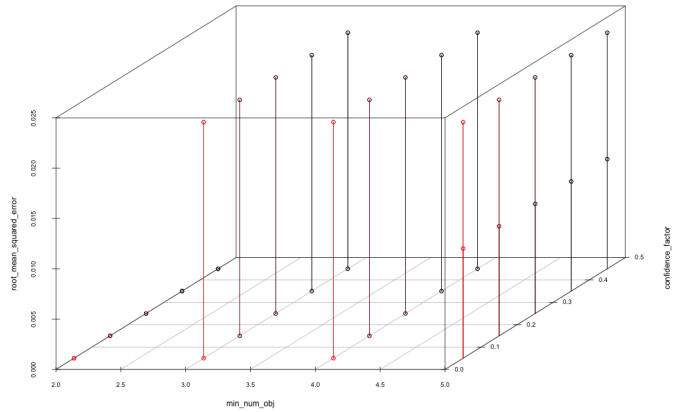
# 1 Decision trees with pruning

Decision trees are one of the most beautiful algorithm because it takes the form of 20 questions. On top of that it is the most intuitive representation of

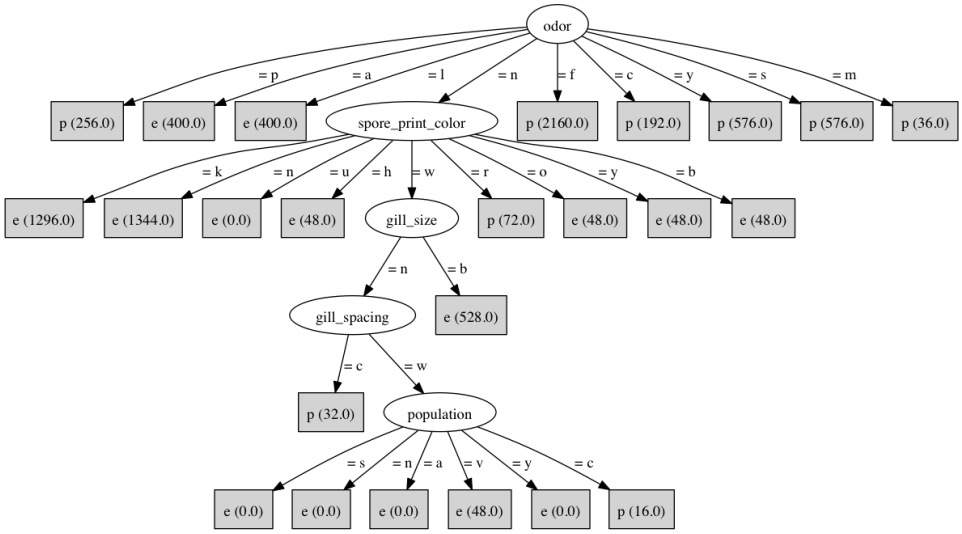
problems. The added benefit of these trees is that you can determine what is the most 'important' variable out of your data set by looking at the first split in the tree and so forth. Decision trees have the unfortunate problem though of sometimes overfitting so that's why we use pruning.

To accomplish analysing our two data sets using decision trees I used the Weka J48 tree which is a C4.5 implementation in Java. The hypothesis set in this case is more complex than KNN or AdaBoost and takes the form of two questions what are the minimum number of objects in each leaf and what is the confidence factor. For this I've decided to use $confidence \in 0.05, 0.15, 0.25, 0.35, 0.45$ and $min\_num\_obj \in [2, 5]$.

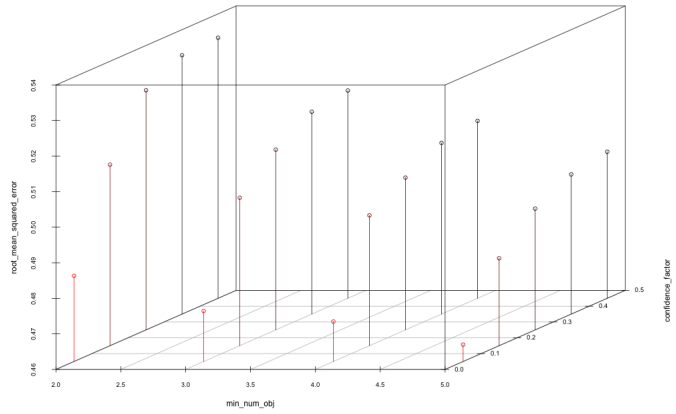
The mushroom data does well yet again but what's interesting is that for the most part whether the confidence level is moved, or min_num_obj is moved around the RMSE stays pretty much the same. That means that the mushrooms should be classified this way! It is a stable solution and no matter what you throw at it does well.




What is almost more interesting is the resulting tree that shows that odor is by far the most important attribute etc. This tree is useful for someone like myself wanting to go out and find mushrooms and when I found it I got very excited cause it means I can classify mushrooms easier!

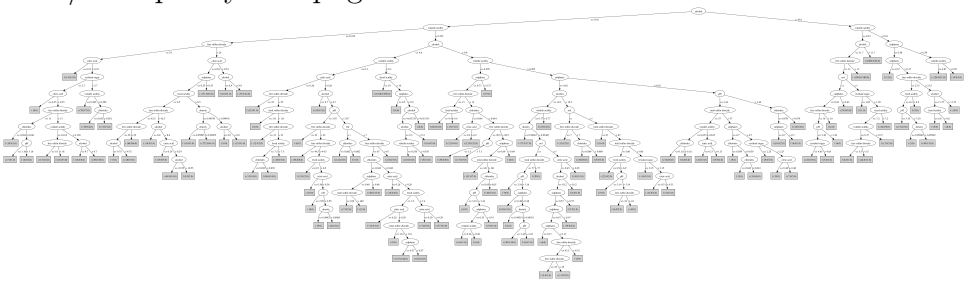



Unfortunately our wine dataset is still not performing. The data set revolves around a RMSE of 0.5 which is not great. At the lowest there is an RMSE of 0.46 when the confidence is 0.5 and min_num_obj is 5. What this signals to me is that the data is noisy because it's relying on the factor of limiting numbers of objects in each leaf.




The resulting tree graphic doesn't help much either. The thing that I take out of it is that alcoholic content is a big determiner of whether it's a good or bad wine. I suppose that makes sense given the fact that high alcohol wines can be considered less tasty. After that volatile acidity and free sulphur dioxide seem to have some determining on whether the wine is good or not but then things become chaotic at the bottom. The graph is so

large that it'll be hard to view this and I recommend you look at it under './assets/winequality_tree.png'.



Decision trees seem to be very well suited for some problems and not for others. In the case of Wine quality there is a lot of continuous variables that seem to perform poorly. The benefit though of using the J48 / C4.5 tree is that it yields good results fast. I think that the best classification of the Mushrooms is the decision tree because it's easy to see what determines what makes a poisonous mushroom which is odor, spore_print_color, gill_size, gill_spacing, and population! Wine didn't fare as well but it's useful to know that some attributes like alcoholic content are a big determinant.

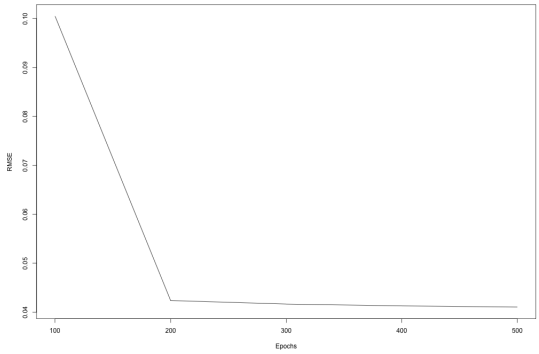
# Neural Networks

Neural Networks are a very deep subject ranging from perceptrons to cyclic RBF networks. In a lot of cases though a feed foward network or multilayer perceptron can achieve great results. This is where I had the toughest time in terms of analysis due to the implementation of Weka's multilayer perceptron package. Since we are dealing with nominal variables being expanded out into binary variables means that we had around 120 inputs for mushrooms and only two outputs. Using the back propagation algorithm this is very slow to compute, so slow in fact that I had to dial back analysis to only concern epochs. Even then there wasn't a lot of opportunity to analyze different methods due to the massive complexity of the network.

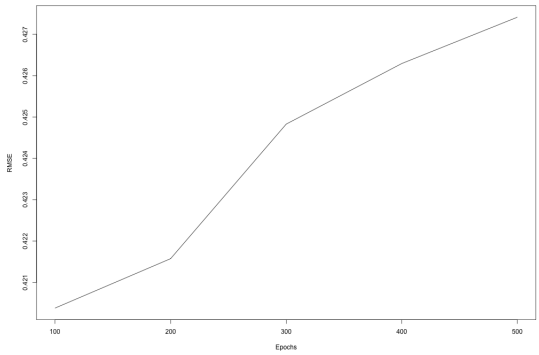
Using an algorithm like RProp would have yielded faster results.

The neural network didn't perform well in either data set. I think the reason why is because on the input layer you have around 120 inputs due to nominal variables being exploded into binary attributes and then only 2 output nodes. Neural networks are good at outputting lots of nodes and I think that it would have been better suited for a problem like classifying across many different categories like frequency of letters to languages.

The mushroom data set started at a RMSE of 0.1 and went down to 0.04 which was not much of a change over epochs of training.




The complexity of the neural network doesn't add anything to the wine classification problem either since it hovers around 0.42 RMSE. This is a good RMSE considering the other classifications but still not exceptional. One way we could improve this would be to split the quality into buckets like it was before (1-10) and assign each quality to an output node. That would improve the RMSE by better classifying wines into qualities.




# Support Vector Machines

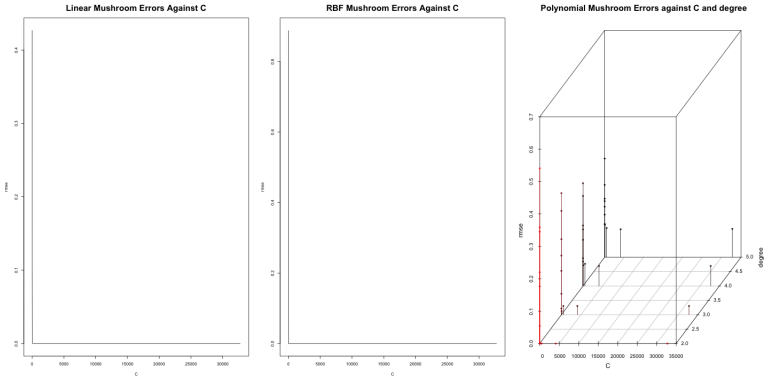
Support Vector Machines are a great way of building classifications. It has the benefit of being a simple quadratic program that can be calculated using the Karush Kuhn Tucker theorem. They become very useful when introducing two things: different kernels and different Cost parameters. The cost parameter relates to introducing slack into the original optimization problem so it can have the added benefit of allowing more to happen. Kernels

also have the added benefit of taking what looks like a noisy dataset and mapping it to a new dimensional set that might be linear in that space. The kernels that a package like LibSVM has is polynomial, sigmoid and radial basis functions.

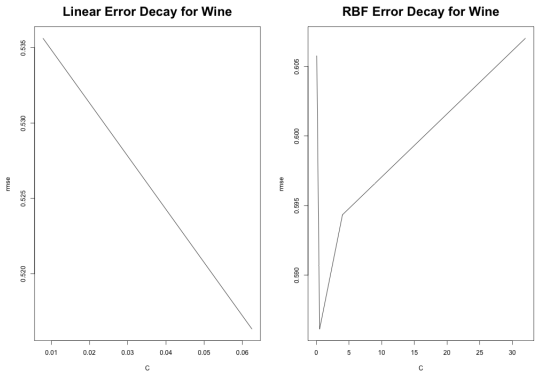
Trying to find the best classificaiton for our datasets I figured that the best way would be to do a grid search by cross validating $2^{-15} < C < 2^{15}$ incrementing the exponent each time by two. As well I figured that looking at linear, polynomial, and rbf kernels would be good enough to search.

While the classification ran fine with our behaved data set, the mushrooms, things didn't operate so well with the wine data set. I'll return to the wine dataset in one of the next paragraphs.

In the mushroom case there was no decay there was more of a cliff from massive error to zero error. Looking at the graph below you'll notice that varying C across the various kernels yielded not much of a different. In this case picking a linear kernel with C at or above 64 would yield good results.




Things weren't so easy in the wine's case. Matter of fact I was not able to finish all of the hypotheses for SVM. Running the cross validation grid search for over a day still yielded no results and it didn't finish. An algorithm that doesn't train fast is the wrong algorithm to use. According to Ockham's Razor it's important to pick the simplest solution. The reason of course for this is because if you build a complex model it will overfit and never work. That being said I was able to run a smaller sample set of hypotheses. In the case of wine I ran $2^{-7} < C < 2^{7}$ and the kernel being either linear or rbf. The rbf kernel yielded faster results and would be my choice to use if we were using this algorithm to classify wine. Looking at the error function things are not very great, there is no real change or decay.




The wine data set is just about impossible to classify using the supervised learning methods outlined in this assignment.

# Conclusion

Overall the question of whether I can go out into the woods and classify mushrooms is answered! Yes! Using the tree that was built using J48 I can determine whether a mushroom is poisonous or not given a few simple questions. And if I'm not sure I can always look it up just to double check. The real benefit though is that now I know what makes a poisonous mushroom.

Unfortunatly I can't say the same thing for determining the above average wine. But then again the question is maybe one that requires more data and more research to answer. Wines are numerous but not as diverse as mushrooms are. A good quality wine doesn't have a different chemical composition than a bad wine. Though I did learn that alcohol volume of a wine does make a difference and that sulphur also impacts wine quality. Whether or not I'll become a sommolier is up to question but I will enjoy revisiting this problem in the future to better classify wines.

Supervised learning methods work well for data sets that have an underlining distribution that we can infer from the past. It seems that the cursory data analysis we did at the beginning made a big difference as to whether the problem itself was solveable or not. Things did not work as well for the wine data set which seems to have a lot of noise in it.