CS7641 Assignment1 - Supervised Learning

Ben Parli

February 7, 2016

Introduction

In assignment 1 we are to identify two interesting datasets and explore a variety of Supervised Learning techniques. Since I've chosen R as my language of choice for this assignment, all of these algorithms were implemented using standard R packages and libraries. That is not to say each dataset fit neatly into each algorithm, however. Data preparation was neccessary in some cases which will be explained in more detail in the following sections. The rest of this assignment is laid out as follows; I begin with a discussion of the chosen classification problems and datasets and the motivation for each. For each classification problem the dataset is outlined and each classifier implementation is discussed. A final summary and takeaway section will finish the assignment.

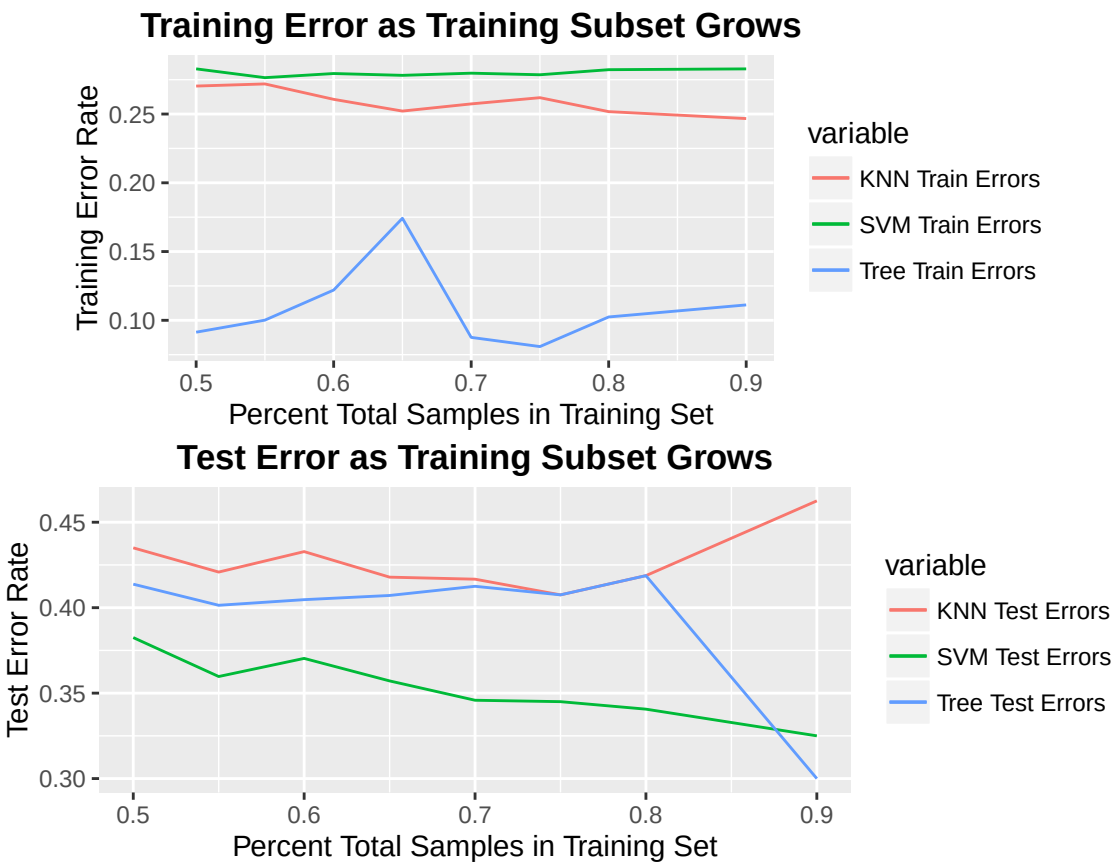
Classification Problems

The classification problems chosen for this assignment are the Wine Quality dataset and Car Evaluation dataset. Both were downloaded from UC Irvine's public repository and both present examples of interesting and potentially far-reaching applications of machine learning and pattern recognition.

Wine quality has been shown time and again to be extremently subjective and even based on visual cues. As a business then, the wine-maker may seek some additional data-driven guarrantee to ensure quality. Further, if patterns can be objectively derived to determine quality based on physicochemical properties in as fickle an industry as wine, there is no reason a similar approach couldn't be applied to industries elsewhere. On the other side of the counter and in a similarl way consumers can leverage such Machine Learning output to determine product quality independent of a subjective human "expert."

The Car Evaluation dataset is another example of the potential of data-driven business decisions. By identifying consumer purchasing patterns, the car manufacturer can better optimize their resources. Though this particular dataset is not particularly wide relative to all the variables of a car, it does capture an essence. That is, by classifying optimal outputs based on a car design's featrure sets, plans can be routed through an objective data-driven vetting process by the business. In this way the business can potentially get ahead of poor design decsions before its too late in the manufacturing schedule. In addition, the approach could easily be extended and scaled, or for that matter applied to other manufacturing industries all together.

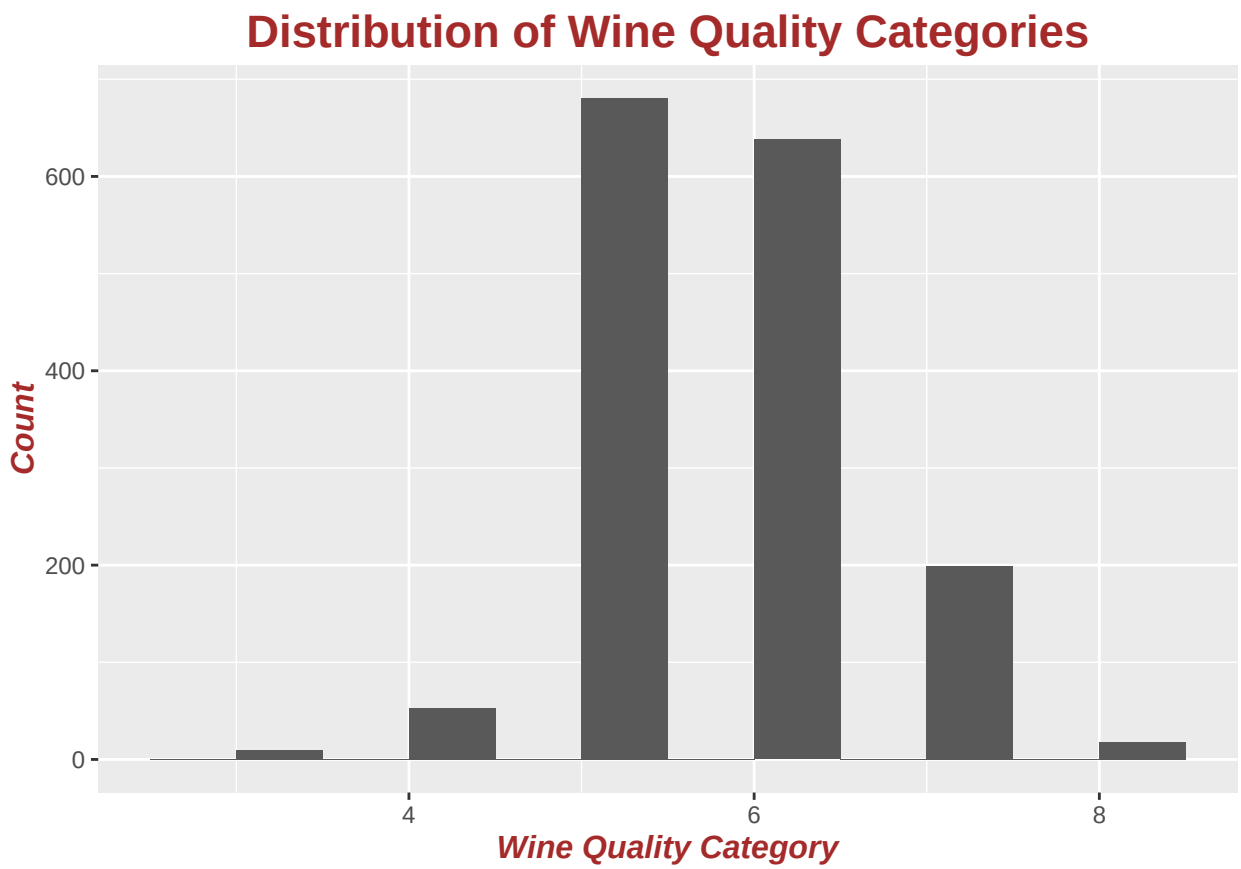
It should be noted in all algorithms the dataset is randomly sampled such that 75% is used for training the model while the remaining 25% is held out for test validation. An alternative is to use cross-validation, an approach in which smaller test samples are taken from the dataset accross separate iterations of training. Since in both cases the sampling is random (using R's sample function) I opted for the one upfront subsetting into Train and Test sets. This approach was used for both datasets. To validate the approach, however, the training an test errors were plotted for the wine dataset algorithms.



The practice of using ~65-75% of the dataset out for use as a test validation set seems to hold here. Although the algorithms behave differently, they seem to trend toward that range. Intuitively, we can expect an underfitted, too general model with too little data to train on. Of course, different algorithms may learn better than others under those sub-optimal scenarios where 50% of the data was held out for testing. Holding out only 10% of the data for testing resulted in fairly unpredcictable behavior. This may have been the result of too small a sample set to test against. Another result of not holding out enough test validation data is we won't be able to properly validate our model against an independent dataset. We cannot have any confidence in the model Without this proper, simulated real world scenario and for that we need an adequate amount of data.

Wine Quality Dataset and Classifications

The Wine Quality dataset is comprised of 11 variables, all the output of physicochemical tests and fairly technical in nature. The classifier is a score between 1 and 10 based on sensory data. The physicochemical test inputs are measurements and therefore can be considered continuous instead of discrete categories. Immediately, we can hypthesize an algorithm such as Neural Networks will perform better than Decsion Trees as a result of this inital observation. In all algorithms some data preparation occured, primarily to convert the inpute from character strings to their numeric values. As one might expect, the data displays a normal distribution; that is, most wines fall in the average range and only a little are very poor or very good.



Wine Quality Decision Trees

The Decision Tree implementation method leveraged the C50 algorithm and R library. C50 has the added convenience of incorporating pruning into its implementation. As a result of this convenience, no additional pruning was conduxted. Interestingly, however, setting the global pruning parameter (nglobalpruning) in this case did not yield any difference in the error rate. Neither did adjusting the mincases parameter yield any positive results. Unfortunately, this method did not prove itself viable for this dataset as the test error rate was approximately 40% higher than the train error rate. This could be an indication of overfitting, but also an indication the decision tree approach does not lend itself to this dataset with so many quantitative, continuous variables. As has been previously written on in Machine Learning textbooks such as Mitchell's, Decision Trees more naturally represent represent disjunctive expressions. The perofrmance of this implemntation would seem to support that theory.

It should be noted, however, the algorithm determines alcohol content to be the primary factor as it is the root of the tree. Splits in this algorithm are determined by usage (metric="usage" is the default parameter input to the function). That is, the algorithm determines the optimal split of all the candidates by recognizing the feature which best splits the samples into subsets of classs. As in the C45 and other tree algorithms, the splitting criterion is the information gain. That is, the C50 algorithm believes the feature "alcohol" contains the most information of any feature and will learn the most from it. Althoguh the algorithm does not perform well overall, these levels of importance are still something that can be learned from it, or at least compared and contrasted with other algorithms.

Attribute usage:

```
100.00% alcohol
 94.16% volatile.acidity
```

```
83.57% sulphates
78.57% total.sulfur.dioxide
65.39% citric.acid
57.30% fixed.acidity
50.13% free.sulfur.dioxide
47.29% chlorides
44.45% density
41.45% residual.sugar
25.77% pH
```

Wine Quality Neural Nets

The Neural Network implementation yielded slightly better results than Decision Trees as one would expect given the nature of the dataset. Since the Neural Network backpropagation operates on the basis of gradient decent I would have intuitively expected it to do much better than Decision Trees for this data set. Indeed it did perform better at 75% correct on the train set and 64% correct on the test set. Similar data pareparation was performed, except min-max scaling was also performed on all the input variables. Scaling is neccessary in Neural Networks, and this dataset, in order to put all the features on a similar playing field. Without scaling some of the features which naturally are measured in a larger numerical magnitude could skew the algorithm in their favor.

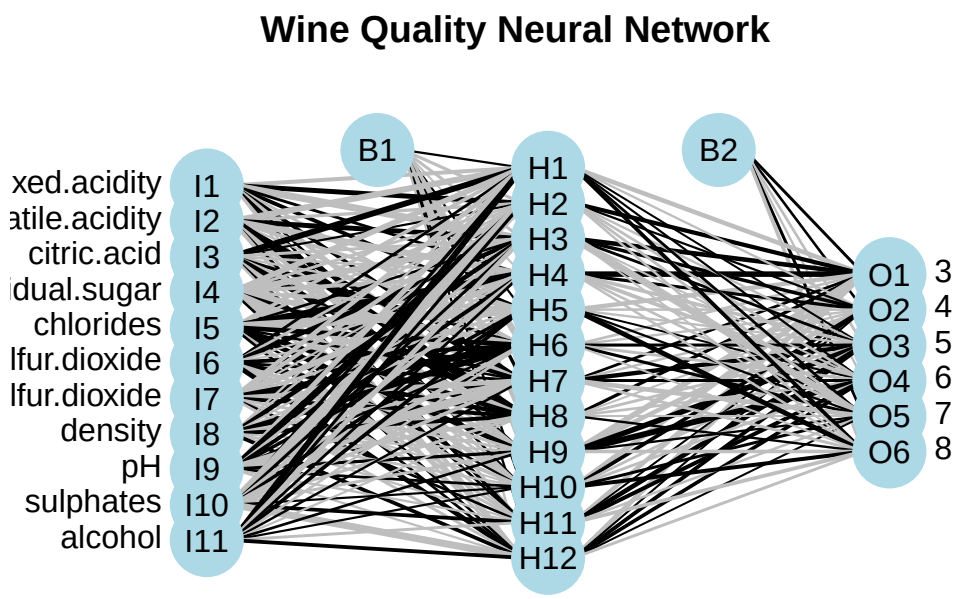
I was able to arrive at this model only after adjusting some of the neural net levers available in this library (the R package nnet). Some findings are summarised here:

- The max iterations were set high enough such that the model could complete. A max iter value of 1000 was sufficient to allow the model to complete. This many iterations was not time intensive so no analysis of the iterations/clock time is provided here as it was for the Boosting algorithm.
- While the max iterations of 1000 allowed the model to find its minima, it also was found to overfit some. By scaling the maximum iterations back to 500 the overall performance improved and the difference betwwen the Training and Test sets narrowed, albeit only slightly. For reference, the algorithm convereged at ~800-850 iterations typically

Neural Net Convergence Performance Table

Max Iterations	Train Error	Test Error	Difference
300	29.4%	40.7%	16.5%
500	26%	40%	14%
750	29%	43.75%	14.75%
1000	28%	44.5%	16%

- After trying a few values of weight decay I opted to leave it out of the model. The weight decay is meant to prevent to the weights from growing too large but this adversely affected performance so the default of 0 was left.
- Identifying the right number of hidden layers took some trial and error. The model stabilised at 12 nodes in the single hidden layer.
- The activation function for this library is the logistic function (a function called sigmoid(double sum) in the c source code). An alternative is to use a linear function as the activation, however, that did not prove to be as performant as the default logistic activation function. This would indicate the dataset has a higher degree of nonlinearity, a finding to keep in mind for the SVM algorithm when comparing SVM kernels.

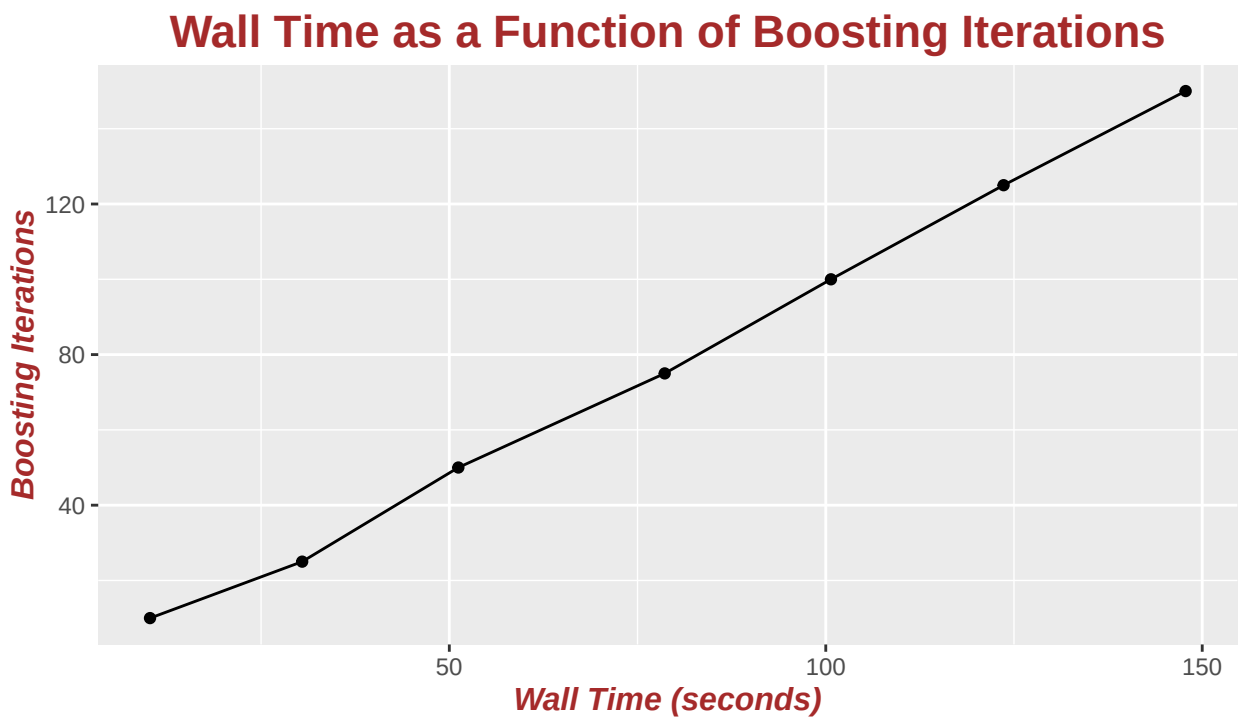


Wine Quality Boosting

The adaboost implementation from the adabag R package was used for Boosting. As before, the classification variable, wine quality, is labeled as a factor with the remaining variables being considered continuous (numeric variables in this case). Two levers were adjusted to gauge how effective they would be against this dataset:

- The boos argument (TRUE or FALSE) determines whether weights are used for each indivdual observation as oppsed to the overall sample. As expected this increased the computational time, however, the performance of the model did improve somewhat.

- The other argument, mfinal, determines how many iterations to run. Again, this becomes a question of computational time vs model performance. Adjusting the metric down did have a slightly adverse effect on the model, but adjusting up only helped a little. As such, this was left at mfinal=100. The computational time growth is visualized in the plot below and again indicates a strong linear growth pattern with the number of iterations.



Since the adaboost implementation of boosting leverages the rpart algorithm, its interesting to compare the performance of boosting to the C50 Decision Tree implementation discussed above. Indeed, the Boosting algorithm test error improves on the C50. This is expected to an extent since, as described previously, the data does not naturally lend itself to decision trees, and boosting is designed to incrementally learn through each iteration (as opposed to one hard iteration with the C50 or even rpart Decision Tree algorithm). This is referred to as "learning slowly" in the boosting reference guides and shows itself to be more performant on this dataset than the "hard learning" of the Decision Tree algorithms.

Also comparing the choices of the Boosting algorithm to the C50 Decision Tree; both algorithms interestingly agree on the most important feature in this dataset. Both identified "alcohol" as the root node to first decide on.

Wine Quality SVMs

The e1071 R package was used for the SVM implementation. This package proved to be very convenient in identifying an optimal solution:

- The svm function includes the option to scale which proved to be a more performant parameter. Comparing the prediction with and without scaling validated that scaling is indeed needed for this dataset. Intuitively, this would be expected since we wouldn't want some columns to carry more weight solely based solely on their magnitude (as also discussed above in the Neural Networks section).
- The package also allows for testing a range of cost parameters. This came in handy in identifying an optimal cost for each kernel.
- A range of costs was tested for the linear, sigmoid and polynomial kernels. Comparing the kernels with their optimal costs then yielded the better kernel to be polynomial. Although the linear kernel performed

admirably, indicating the solution can be classified with at least some linearity, the polynomial kernel was superior. Thus, the optimal decision boundary is quadratic.

SVM Tuning Table

Kernel	Optimal Cost	Test Error Rate
Sigmoid	0.1	43.7%
Linear	0.1	41.2%
Poly	5	34.5%

Wine Quality KNN

Wrapping up the Wine Quality dataset ation problems is the KNN classifier. The KNN implementation came from the R "class" package. The KNN function takes as two of its inputs a training dataset and a test dataset. The classifier was run a series of times in order to tune to k value.

- Interestingly, it turned our to be a low k value of only 3 for which the classifier performed the best.
- Tne training dataset still exhibited a 31% error rate though, and the test dataset exhibited a 41% error rate.
- Another interesting finding of the KNN classifier is it actually performed worse when the classifier variable (wine quality) was prepared as a factor (as it was for the other algorithms).
- Initially I did not scale the dataset features, however, after performing a scaling the algorithm performed ~8% better. This can be attributed to the way in which KNN determines the distance for clustering. If some features have a larger magnitude it must have thrown off the distances during the clustering. For reference, this implementation uses Euclidean distance.
- Leaving the classifier, the wine quality feature, in its initial cast as an int also yields a slightly better performance.

All in all though, KNN turns out to be a poor fit for this dataset.

Wine Quality Algorithms Comparison

The Wine Quality dataset with its continuous variables was a difficult dataset to classify. More data preparation was neccessary and there was a lot of variance in performance of the classification methods (as can be seen below). If we were to judge solely on the Training data performance the Decision Tree would be clearly superior, however, we know the Test dataset performance is critical since it simulates the final real-world performance of our model. The best balance of the five algorithms is actually the SVM implementation. It combines a decent Training error rate with a decent Test error rate. While not an ideal measure of performance in either case, the SVM implementation is the best for this dataset.

Final Wine Quality Comparison Table

Algorithm	Train Error Rate	Test Error Rate
Decision Tree	8%	40.7%

Algorithm	Train Error Rate	Test Error Rate
Neural Net	26%	38.75%
Boosting	30.7%	38%
SVM	27.8%	34.5%
KNN	26%	40.5

Cars Quality Dataset and Classifications

The Car evaluation dataset seeks to classify a car as a desirable purchase or not based on several basic inputs. This dataset differs from the Wine Quality dataset in a couple interesting ways: 1) The Car Evaluation dataset variables are much less technical and more intuitive and 2) unlike the Wine measurements, this dataset is comprised of all categorical variables. Less data preparation was neccessary in this case since the data features were imported into the R environment already in a good categorical state.

The Cars Evaluation dataset contains 6 attributes and a classifier to determine whether a car is desirable or not. 1. Buying: A factor variable. What is the initial cost of the car 2. Maint: A factor variable. What is the ongoing cost of maintaining the car 3. Doors: A factor variable. How many doors are on the car 4. Persons: A factor variable. How many people can ride in the car 5. Lug_boot: A factor variable indicating the size of the luggage area 6. Safety: A factor variable indicating overall safety of the car 7. Class: the classifer to determine desirability of the car. Factors of this variable range from unacc, acc, good to vgood.

As a reminder, in all algorithms the dataset is randomly sampled (using R's builtin sampling function) such that 75% is used for training the model while the remaining 25% is held out for test validation.

Cars Quality Decision Trees

As noted above, the Car Evaluation dataset is fairly intuitive and already highly categorical. One of the benfits of Decsion Trees is the approach is also very intuitive to humans. Such a categorical dataset naturally lends itself to a Decision Tree algorithm, so it was not surprising to see a strong performance. The C50 package in R was again used to implement the C50 Decision Tree algorthim. Splits in this algorithm are determined by usage (metric="usage" is the default parameter input to the function). That is, the algorithm determines the optimal split of all the candidates by recognizing the feature which best splits the samples into subsets of classs. As in the C45 and other tree algorithms, the splitting criterion is the information gain. The algorithm performed exceptionally well; over 98% against the train set and 96% against the test set. The C50 algorithm is a recursive Decision Tree algorithm but still built a tree of only 42 leaves. Interestingly, we can see the primary dividing factor (i.e. the root of the tree) is the persons variable, indicating a buyer's first preference is how many can fit inside the car.

Attribute usage:

```
100.00% persons
 65.97% safety
 44.44% buying
 44.44% maint
 30.94% lug_boot
  8.64% doors
```

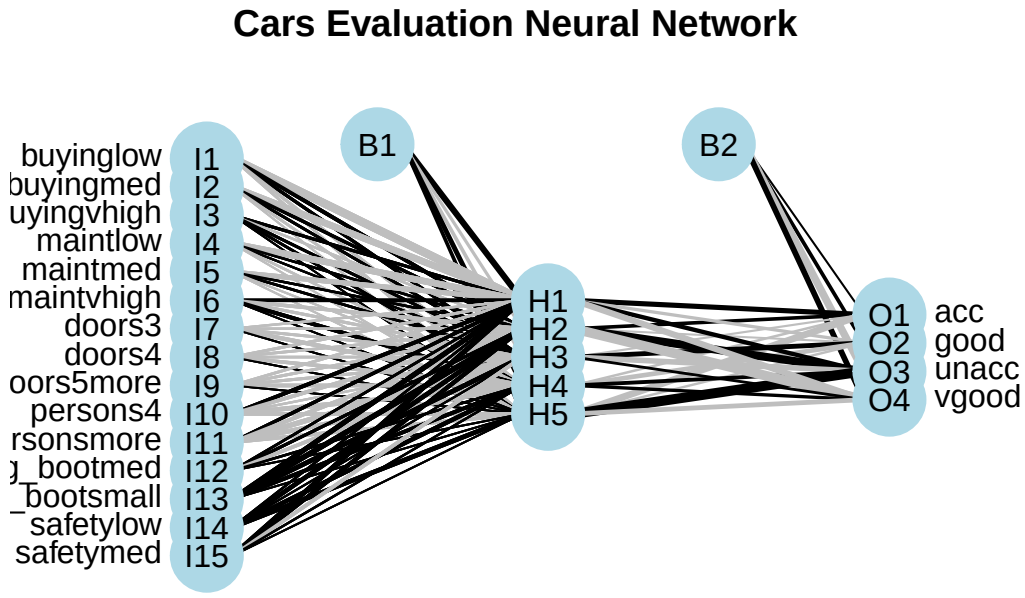
Cars Quality Neural Nets

Neural Networks are a sexy algorithm, but the performance of the C50 Decision Tree algorithm will be very tough to beat. Additionally, the categorical features of the dataset do not lend themselves to the Neural

Network (at least intuitively). Again, the nnet R package and function was used in this implementation. It was assumed that in order to get the algorithm to work some additional data preparation was needed. Namely, the factor variables needed to be converted to numeric values. Scaling would not be needed though since all the scores were already set similarly in magnitude (i.e. scores between 1-4). After assigning numeric scores to each factor the Neural Net can iterate through its backpropogations and decide on a set of weights. A max iterations of only 200 was enough to allow the algorithm to converge to a minima in this case. This many iterations was not time intensive so no analysis of the iterations/clock time is provided here as it was for the Boosting algorithm.

As with the Wine Quality implementation, the activation function for this library is the logistic function (a function called sigmoid(double sum) in the c source code). An alternative is to use a linear function as the activation, however, (also as before) that did not prove to be as performant as the default logistic activation function.

Interestingly enough, the optimal setup for this dataset and the Neural Net algorithm was closer to its original form. The performance actually improved by reloadig the data and running against the Neural Net without setting any of the factor variables to numeric scores. By running the categorical dataset through the Neural Net an impressive performance of 1% training error rate and 4% testing error rate was achieved. This was achieved with 5 nodes in the hidden layer.



Cars Evaluation Boosting

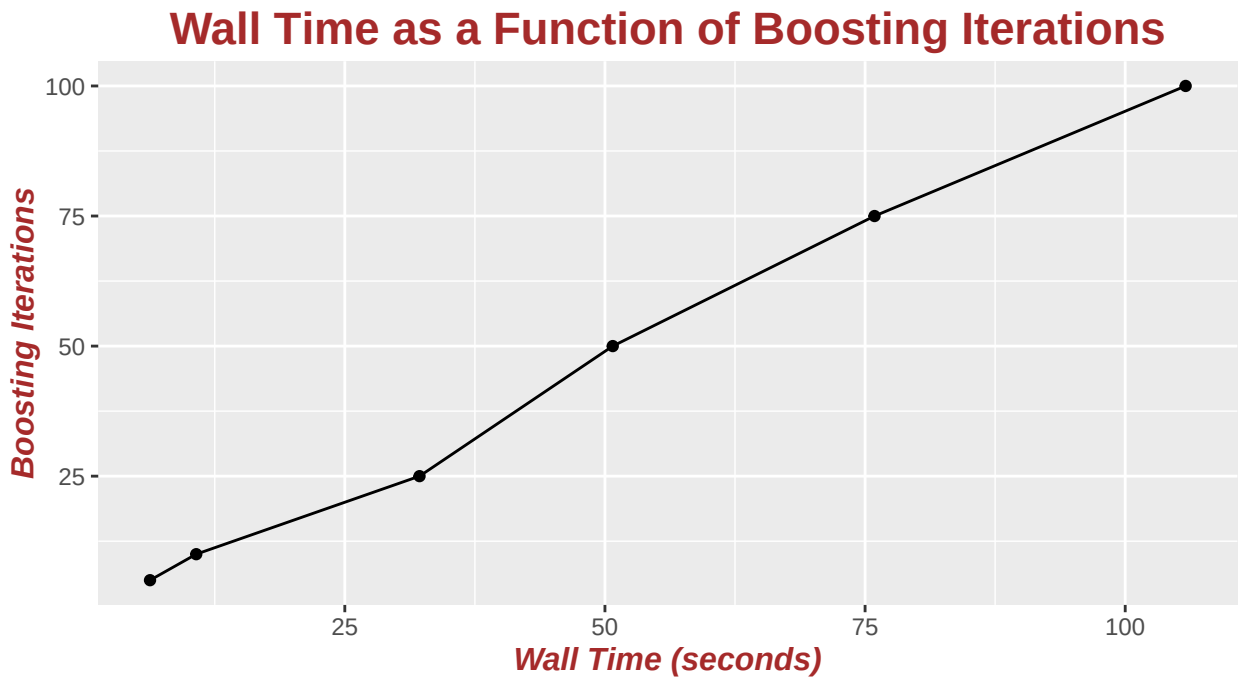
The adabag R package and library were again used in the Boosting algorithm. Since the adabag boosting algorithm uses the Rpart Decision Trees algorithm, and we already saw the performance of the C50 Decision Tree on this dataset, we can expect a strong performance of the Cars Evaluation Boosting implementation. Additioanlly, it was observed in the Wine Quality dataset that the Boosting algorithm improved on the

Decision Trees algorithm. It turns out the case is not different here as the Boosting implementation again proves more performant than Decision Trees, albeit narrowly. The Boosting algorithm scored perfectly against the Training set. It could reasonably be expected that overfitting had occured, however, in this case the implementation performed almost as well against the Test set at 2% error rate. There wasn't much tuning of parameters and unlike the Wine Quality Boosting, the mfinal could be left low at a value of mfinal=10. In this way the Boosting algorithm is impressively both performant on a time basis as well as a prediction basis.

Another interesting comparison of the C50 Decision Tree and Adabag Boosting implementations is which feature they settle on being the most important. In the C50 this is the root of the tree. Similarly the Boosting algorithm must decide on the most important over its iterations. The C50 algorithm identified persons as the primary feature, while the Boosting algorithm identified safety as the most important with persons being second.

```
##    buying     doors  lug_boot     maint   persons    safety
## 15.980933  8.470896 14.716517 19.139903 17.915961 23.775790
```

As with the Wine Quality Boosting implementation the algorithm was run with a series of iterations as the as an input parameter ("mfinal"). The output can be seen in the plot below and again indicates an approximately linear growth pattern with the number of iterations.



Cars Quality SVMs

The e1071 R package and library were again leveraged as an implementation for the Support Vector Machine algorithm. A tuning approach similar to that taken with the Wine Quality dataset was taken, but this time a much wider range of optimal parameters were identified. Interestingly the cost parameter varied from 10 all

the way to 450 across the three kernels. Since the larger cost is optimized to a smaller-margin hyperplane we can expect that to be the most likely to overfit. Another way to think of this parameter is the larger cost will drive the algorithm to be less general. However, this concern did not seem to be the case as the Polynomial kernel with the highest cost performed significantly better on the Trains and Test sets.

SVM Tuning Table

Kernel	Optimal Cost	Test Error Rate
Sigmoid	50	11.9%
Linear	10	8.7%
Poly	450	1%

Cars Quality KNN

The class R package was again used in the KNN algorithm implementation. Since KNN decides on similarity to neighbors through a distance vector (namely Euclidean distance), this algorithm was expected to perform well against the Car Evaulation dataset once the feature set was prepared as a set of numeric scores, similar to Neural Networks. The Cars categories naturally lend themselves to scores since each is increasing in value (i.e. unacc, acc, good, vgood can intuitively be mapped to 1-4 without losing any meaning of the categories).

After mapping the features from categories to numerical scores, the KNN algorithm was found to perform well. There was only a 5% error rate on the test set and an 8% error rate on the test set. Similar to the Wine Quality KNN implementation, a small k parameter (k=4) was found to be optimal.

Car Evaluation Algorithms Comparison

The Car Evaluation dataset with its basic categorical features proved an ideal Machine Learning dataset. As can be seen below all the algorithms performed admirably, but interestingly the SVM and Boosting did the best. We might expect more difficulty if we added to the feature space, and particularly if additional features are more continuous and quantitative than this set. Of course, when comparing these implementations we cannot with certainty claim any of these superior to the others with the exception of possibly the KNN algorithm. The differences could be due to noise, error, or just chance.

Final Car Evaluation Comparison Table

Algorithm	Train Error Rate	Test Error Rate
Decision Tree	02%	04%
Neural Net	01%	04%
Boosting	0%	02%
SVM	0%	01%
KNN	05%	08%

Final Analysis

In the end the Wine Quality dataset proved to be the more challenging (and thus interesting) of the two. The Car Evaluation implementations didn't need much data manipulation at all, even in the case of the Neural Networks, although some was performed for the KNN implementation. Also, the Car Evaluation algorithms

all performed extremely well in all the algorithms where the Wine Quality metrics were much more difficult to classify. There was also more variance in performance among the Wine Quality implementations.

Citations

UCI Machine Learning Repository, Wine Dataset [https://archive.ics.uci.edu/ml/datasets/Wine]. Irvine, CA: University of California, School of Information and Computer Science.

UCI Machine Learning Repository, Car Evaluation Dataset https://archive.ics.uci.edu/ml/datasets/Car+Evaluation]. Irvine, CA: University of California, School of Information and Computer Science.

Machine Learning, 1997, Tom Mitchell