Exploring Various Techniques in Surprised Learning

Kam Wing Chung (Kenneth)

Department of Computer Science

Georgia Tech University

Kchung42@gatech.edu

# Purpose

The purpose of this assignment is to explore supervised learning techniques using decision trees, neural networks, boosting, support vector machines and k-nearest neighbor algorithms. All of above supervising learning techniques will be applied on Wine Quality dataset found on (<http://archive.ics.uci.edu/ml/datasets/Wine+Quality>). All of above supervising learning techniques will be using R and R supervising learning packages. R code, dataset and README.txt can be found in the same package as this document.

# Wine Quality Dataset and Goal

Original dataset contains white and red wine dataset in separate files. WineColor column is added to each dataset and Red is assigned to red wine and White is assigned to white wine new WineColor column.

The two individual datasets are combined together to form one dataset.

Input variables (based on physiochemical tests and in real values)

1 - Fixed acidity (Numeric)

2 - Volatile Acidity (Numeric)

3 – Citric Acid (Numeric)

4 – Residual sugar (Numeric)

5 – Chlorides (Numeric)

6 – Free sulfur Dioxide (Numeric)

7 – Density (Numeric)

9 – pH (Numeric)

10 – Sulphates (Numeric)

11 – Alcohol (Numeric)

12 – WineColor(Numeric: 0=red,1=white)

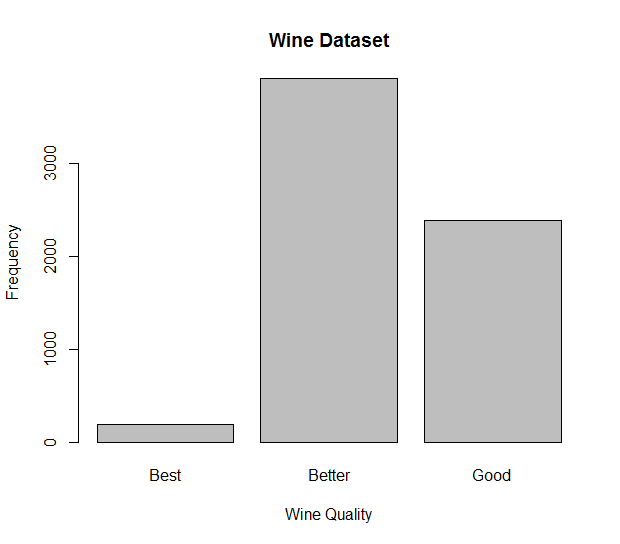
Output variables (based on sensory data)

13 – Quality (1 to 9)

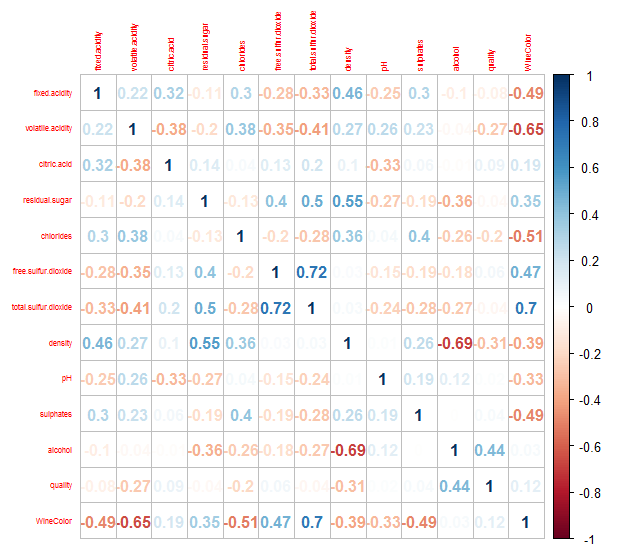
14– Quality3 (Good, Better and Best)

The combined wine quality dataset (WineDataset) will be used in the following supervised learning algorithm to predict wine quality. The original quality real numbers are transformed to Good (<6), Better (6 and 7) and Best (8 and 9).

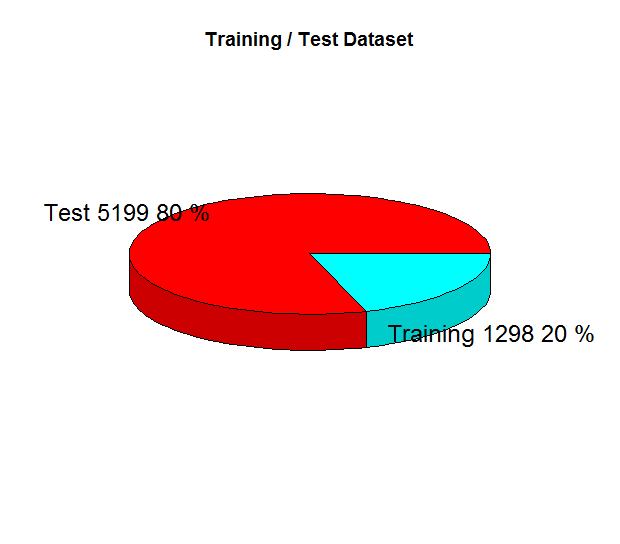
Here is the quality distribution based on three-class classification.



Linear regression suggested that density and WineColor variables which are highly to be a dependent variables predicted by the other variables. Density and WineColor variable will not be considered. The following correlation matrix has given overview about the relationship between independent variables.



The WineDataset is divided into 80% (size=5199) for training and 20% (1298) for testing.



There is a pattern to determine quality purely based on the following the variables or subjective taste preference from a group of wine experts.

There are no missing attributes values in both datasets.

Figure 1

Figure 2

The goal on this dataset is to classify the wine quality based on the attributes.

# Decision Tree

Representation of Decision Tree

Top down and greedy algorithm inducting decision tree ID3

Expressive of Decision Tree

Bias of ID3

Best Attributes (Gain(S,A)

Deal of with Over fitting

## 3.1 No pruning

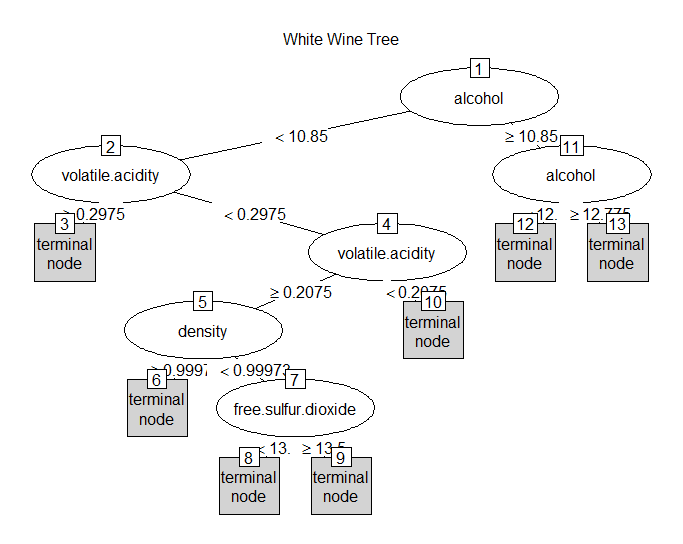


Figure 3

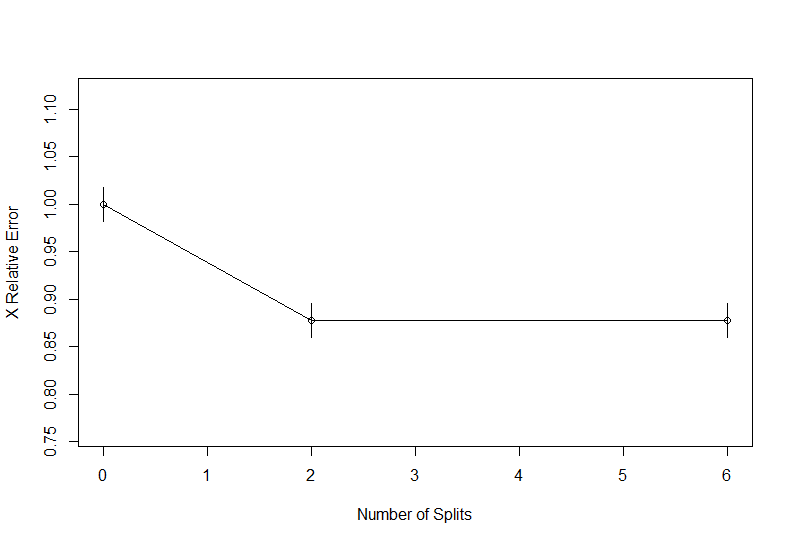


Figure 4

Figure 3 shows that there are six splits on decision tree using the white wine test data. Figure 4 indicates that the starting from split 2, the cross errors remains at just above 0.85.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Predicted Class | | | | | | |
|  |  | 3 | 4 | 5 | *6* | 7 | 8 | 9 |
| Actual Class | 3 | 0 | 0 | 3 | 7 | 0 | 0 | 0 |
| 4 | 0 | 0 | 28 | 38 | 0 | 0 | 0 |
| 5 | 0 | 0 | 377 | 343 | 0 | 0 | 0 |
| 6 | 0 | 0 | 220 | 894 | 0 | 0 | 0 |
| 7 | 0 | 0 | 14 | 436 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 85 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 4 | 0 | 0 | 0 |

Table 2

The probability of predicting correct classes on the white wine test dataset is 0.518. The calculation is based on 1/n where n is the number of test dataset sample and k is the index of prediction and actual classes. If prediction value equals to actual class, returns 1 otherwise 0.

## 3.2 Pruned Tree

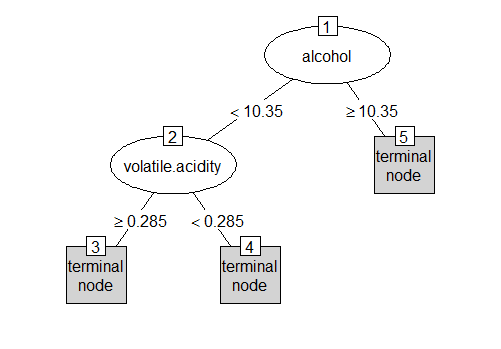


Figure 5

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Predicted Class | | | | | | |
|  |  | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Actual Class | 3 | 0 | 0 | 2 | 8 | 0 | 0 | 0 |
| 4 | 0 | 0 | 21 | 45 | 0 | 0 | 0 |
| 5 | 0 | 0 | 268 | 452 | 0 | 0 | 0 |
| 6 | 0 | 0 | 142 | 972 | 0 | 0 | 0 |
| 7 | 0 | 0 | 10 | 440 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 85 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 4 | 0 | 0 | 0 |

Table 3

Figure 5 has pruned tree based on 2 splits and cross error indicated on Figure 4. The probability of predicting correct class on white wine test dataset using pruned is 0.506 which is slightly lower than non-pruned tree.

## 3.3 Decision Tree Conclusion

|  |  |  |
| --- | --- | --- |
|  | No Prune | Pruned |
| Probability correct prediction | 0.518 | 0.506 |
| Classes capable of recognizing | 5 and 6 | 5 and 6 |
| Classes incapable of recognizing | 3,4,7 and 8 | 3,4,7 and 8 |
| Number of Splits | 6 | 2 |
| Variables actually used in tree construction | Alcohol, volatile. Acidity, Density, Free Sulfur dioxide | Alcohol and volatile.acidity |
| Stop at Cross Error | ~0.85 | Slightly lower than ~0.85 |

Table 4

The probability of correct prediction has slight dropped in the pruned tree, essentially it has no significant impact in in both training and test dataset just under 2500 samples.

Both trees are only capable of predicting class 5 and 6; and incapable of predicting class 3, 4, 7, 8 and 9.

In the pruned tree, there are two out of eleven variables that are actually used to construct the decision tree.

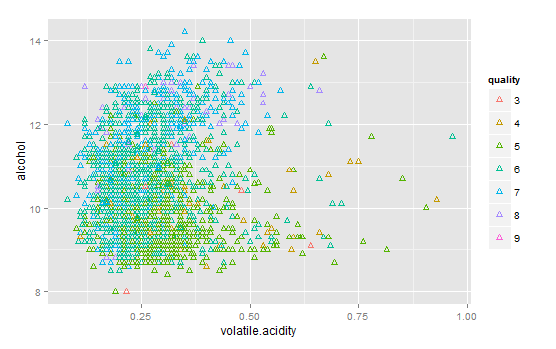


Figure 6

In the figure 6, class 5 wine is populated around low volatile acidity and low alcohol. Class 6 wine is populated around also low volatile and high alcohol. Very small number of class 3 and 8 scattered across. There might be slight greater number of class 4, 7 and 8 that are populated inside the clusters of class 5 and 6.

# Neural Network

Cross Valuation

Linear and polynomial regression

Perceptron – threshold unit

Back propagation/Gradient Descent

Preference/Restriction bias of neural networks

## Configuration and Cross Valuation

Two major parameters for neural network for this implementation is the maximum iteration and k-fold valuation. Iteration is set to 1000 to archive some global minimum and k-fold is set to 10.

Here is the model and cross valuation summary.

|  |  |
| --- | --- |
| Sample Size | 5199 |
| Predictors | 10 |
| Classes | Best, Better, Good |
| Feature Scaling | To [0, 1] |
| Cross-validated | 10-fold |
| Summary of Sample | 4680, 4679, 4679, 4679, 4680, 4679 |

Feature scaling helps to converge faster when the larger range of real values in different variables.

|  |  |  |
| --- | --- | --- |
| size | decay | Accuracy |
| 1 | 0 | 0.659 |
| 1 | 0.0001 | 0.712 |
| 1 | 0.1 | 0.712 |
| 3 | 0 | 0.679 |
| 3 | 0.0001 | 0.729 |
| 3 | 0.1 | 0.723 |
| 5 | 0 | 0.673 |
| 5 | 0.0001 | 0.738 |
| 5 | 0.1 | 0.731 |

The size of hidden activation nodes is 5, and the decay is zero (close to zero) is the final model to be used in the prediction.

## 2.2 Prediction Results and Analysis

The confusion matrix us shown as fellow:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Prediction Class | | |
|  |  | Best | Better | Good |
| Actual Class | Best | 0 | 38 | 0 |
| Better | 0 | 746 | 37 |
| Good | 0 | 364 | 112 |

The overall statistics is shown as follow:

|  |  |
| --- | --- |
| Accuracy | 0..661 |
| 95% CI | (0.6345, 0.6868) |
| No Information Rate | 0.8852 |
| P-Value [Acc > NIR] | 1 |
| Kappa | 0.2004 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Best | Better | Good |
| Sensitivity | NA | 0.6493 | 0.75168 |
| Specificity | 0.96995 | 0.7517 | 0.6832 |
| Pos Pred Value | NA | 0.9527 | 0.23529 |
| Neg Pred Val | NA | 0.2175 | 0.95499 |
| Prevalence | 0 | 0.8852 | 0.11479 |
| Detection Rate | 0 | 0.5747 | 0.08629 |
| Detection Prevalence | 0.03005 | 0.6032 | 0.36672 |
| Balanced Accuracy | NA | 0.7005 | 0.71744 |

Compare to k-nearest neighbors, there is slightly increase in accuracy but this requires high cost of computation spending on training neural network.

# Support Vector Machine

Margins

Kernel Tricks

Optimization problem for finding max margins

Support Vectors

## Configuration and Cross Valuation

Here is the model and cross valuation summary.

|  |  |
| --- | --- |
| Sample Size | 5199 |
| Predictors | 10 |
| Classes | Best, Better, Good |
| Feature Scaling | To [0, 1] |
| Cross-validated | 10-fold |
| Summary of Sample | 4680, 4679, 4679, 4679, 4680, 4679 |

The configuration and cross validation remains the same as Neural Network. SVM Linear and SVM Radial are examined.

|  |  |  |
| --- | --- | --- |
| C | Accuracy (Radial) | Accuracy (Linear) |
| 0.25 | 0.733 |  |
| 0.5 | 0.739 |  |
| 1 | 0.739 | 0.716 |

For SVM Radial, tuning parameter 'sigma' was held constant at a value of 0.09516869274. Accuracy was used to select the optimal model using the largest value. The final values used for the model were sigma = 0.09516869274 and C = 0.5.

In Linear, there is one value of tuning parameter 'C' which was held constant at a value of 1. The accuracy is 0.716 which is a bit lower than Radial. This reflects the dataset has fairly linear relationship quality and the input variables.

## Prediction Results and Analysis

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | SVM Radial Prediction Class | | |
|  |  | Best | Better | Good |
| Actual Class | Best | 0 | 38 | 1 |
| Better | 0 | 667 | 116 |
| Good | 0 | 159 | 317 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | SVM Linear Prediction Class | | |
|  |  | Best | Better | Good |
| Actual Class | Best | 0 | 38 | 1 |
| Better | 0 | 664 | 119 |
| Good | 0 | 199 | 277 |

The overall statistics is shown as follow:

|  |  |  |
| --- | --- | --- |
|  | SVM Radial | SVM Linear |
| Accuracy | 0.7580077 | 0.7249615 |
| 95% CI | 0.7306636, 0.7781912) | (0.6997931, 0.7491068) |
| No Information Rate | 0.6656394 | 0.6941448 |
| P-Value [Acc > NIR] | 2.527E-13 | 0.008222765 |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| SVM Radial | Best | Better | Good |
| Sensitivity | NA | 0.7679 | 0.7245 |
| Specificity | 0.96995 | 0.7269 | 0.8118 |
| Pos Pred Value | NA | 0.8493 | 0.6576 |
| Neg Pred Val | NA | 0.6097 | 0.8552 |
| Prevalence | 0 | 0.6672 | 0.3328 |
| Detection Rate | 0 | 0.5123 | 0.2411 |
| Detection Prevalence | 0.03005 | 0.6032 | 0.3667 |
| Balanced Accuracy | NA | 0.7474 | 0.7682 |

|  |  |  |  |
| --- | --- | --- | --- |
| SVM Linear | Best | Better | Good |
| Sensitivity | NA | 0.7366 | 0.6915 |
| Specificity | 0.96995 | 0.694 | 0.779 |
| Pos Pred Value | NA | 0.8429 | 0.584 |
| Neg Pred Val | NA | 0.5417 | 0.8491 |
| Prevalence | 0 | 0.6903 | 0.3097 |
| Detection Rate | 0 | 0.5085 | 0.2142 |
| Detection Prevalence | 0.03005 | 0.6032 | 0.3667 |
| Balanced Accuracy | NA | 0.7153 | 0.7353 |

# *k*-nearest neighbors

Instance based learning

Lazy and eager learnings

Similarity function (distance function)

Averaging

Locally weighted regression

## 4.1 Configuration and Cross Validation

**More features you include, more data you need** (The curses functionality)

|  |  |
| --- | --- |
| Sample Size | 5199 |
| Predictors | 10 |
| Classes | Best, Better, Good |
| Feature Scaling | To [0, 1] |
| Cross-validated | 10-fold |
| k-fold selection | 1,2,3,5,10,20,30,50 |

K=1 is selected because of having highest accuracy.

|  |  |
| --- | --- |
| k | Accuracy |
| 1 | 0.754 |
| 2 | 0.701 |
| 3 | 0.715 |
| 5 | 0.722 |
| 10 | 0.718 |
| 20 | 0.713 |
| 30 | 0.711 |
| 50 | 0.72 |

As the k increases, the accuracy remains pretty flat until k reaches 50.

## 4.2 Prediction Results and Analysis

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Prediction Class (k=1) | | |
|  |  | Best | Better | Good |
| Actual Class | Best | 21 | 17 | 1 |
| Better | 24 | 643 | 116 |
| Good | 1 | 152 | 323 |

|  |  |
| --- | --- |
| Accuracy | 0.7604 |
| 95% CI | (0.7362, 0.7834) |
| No Information Rate | 0.6256 |
| P-Value [Acc > NIR] | <2e-16 |
| Kappa | 0.5182 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Best | Better | Good |
| Sensitivity | 0.45652 | 0.7919 | 0.7341 |
| Specificity | 0.98562 | 0.7119 | 0.8217 |
| Pos Pred Value | 0.53846 | 0.8212 | 0.6786 |
| Neg Pred Val | 0.98014 | 0.6718 | 0.8577 |
| Prevalence | 0.03544 | 0.6256 | 0.339 |
| Detection Rate | 0.01618 | 0.4954 | 0.2488 |
| Detection Prevalence | 0.03005 | 0.6032 | 0.3667 |
| Balanced Accuracy | 0.72107 | 0.7519 | 0.7779 |

# Boosting

Ensembles are good

Bagging are good

Combine simple classifiers -> Complex

Boosting is really good

Weaker Learners

Boosting avoid over fitting