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 use the following supervised learning techniques; decision trees, neural networks, boosted tree, support vector machines and k-nearest neighbor algorithms on a certain dataset to learn and predict.

Wine Quality dataset is chosen and found on (<http://archive.ics.uci.edu/ml/datasets/Wine+Quality>). Wine are sold at different prices because of quality. The quality is usually subjective to a group of wine experts who would use their experience and knowledge to taste the wine; and grade each type of wine.

Grading the wine quality require many years of experience and knowledge, in this machine learning exercise, we will examine how much of different supervised learning algorithm to learn and how accurate these algorithm can predict.

# Wine Quality Dataset Examination

There are white and red wine datasets in separate files.

Each file has the following variables

Input variables

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)

Output variables (based on sensory data)

13 – Quality (1 to 9)

There are no missing attributes values found in both datasets.

All the input variables are based on physiochemical tests and in real values. (You are right, year is not in this set of variable)

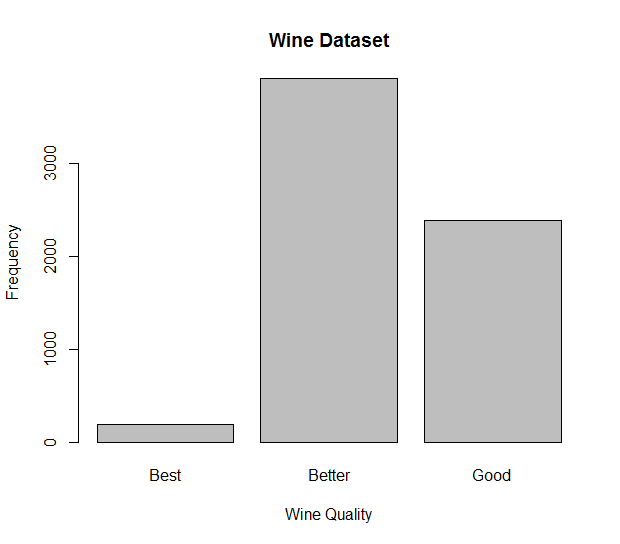
WineColor column is added to each dataset and 0 (Red) is assigned to red wine and 1 (White) is assigned to white wine new WineColor columns in each file.

These two files are combined together to form a single dataset.

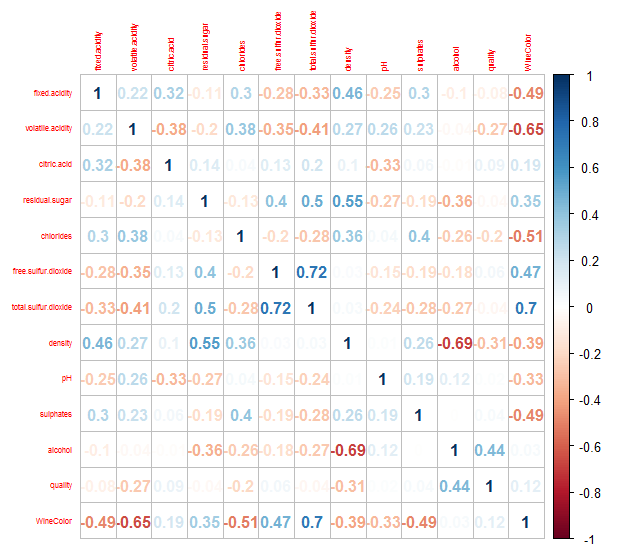
The combined wine quality dataset (WineDataset) will be used in the following supervised learning algorithms to predict wine quality.

The original quality real numbers are transformed to Good (quality <6), Better (6 and 7) and Best (8 and 9) to a new column “myclass”

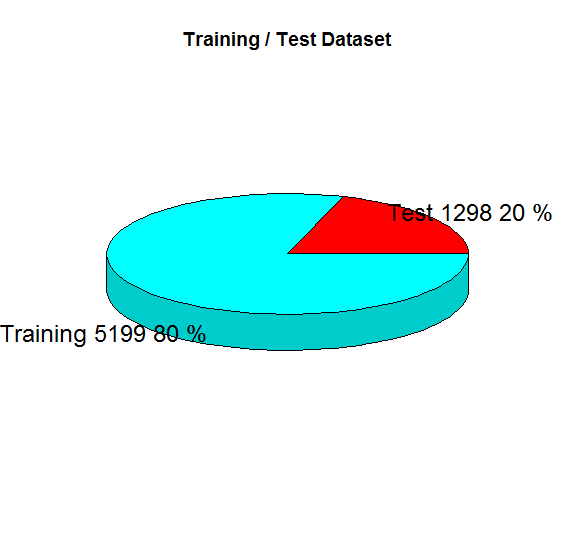
Here is the quality distribution based on three-class classification.



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 us an overview about the relationship between independent variables.



To avoid using the same dataset for training and testing, 80% (size=5199) of the combined WineDataset is will be used for training and 20% (1298) for testing.



# Training Set up, Prediction and Evaluation Strategies

Training data will be used to create a model for prediction. The model with highest accuracy will be chosen for prediction. Accuracies between training and prediction will be compared.

Since real values between attributes have a fair large range, it is always a good idea to apply feature scaling on the data. The training data will be re-scaled to [0, 1]. The feature scaling will not affect the accuracy and training but it will reduce the time on cost optimization problem to find global minimum.

Over-fitting (high variance) and high bias are the major problems in machine learning. High bias could be caused by irrelevant or insufficient features. Over-fitting can be caused by trying to fit very well on single dataset but cannot be generalize on the test dataset.

K (10)-fold Cross-validation and resampling are the strategies to avoid over fitting. A random proportion of data (80% in this experiment) are used to train a model. The rest will be used for testing.

The training set will be resampled k times and trained with different set of tuning parameters. After k times, the model with highest accuracy will be chosen for testing.

All of the tests will have training configuration similar to the following table.

|  |  |
| --- | --- |
| Sample Size | 5199 |
| Predictors | 10 |
| Classes | Best, Better, Good |
| Pre-processing: re-scaling | To [0, 1] |
| Cross-validated | 10-fold |
| Summary of Sample | 4680, 4679, 4679, 4679, 4680, 4679 |

After the model with highest accuracy is chosen and this model will be applied on test dataset for prediction.

The confusion matrix is the primary evaluation strategy to review each supervised learner strength and weakness. The following is one of resulting confusion matrices for three-class wine quality classification.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Predict Best | Predicted Better | Predicted Good |
| Actual Best | 0 | 39 | - |
| Actual Better | 0 | 662 | 121 |
| Actual Good | 0 | 259 | 217 |

The predictor predict 39 “Better” but they are actually “Best”. So for and so on.

Sensitivity, specificity, Positive Prediction Value, Negative Prediction Value and Prevalence will be examined as needed to investigate the efficiency of different learners.

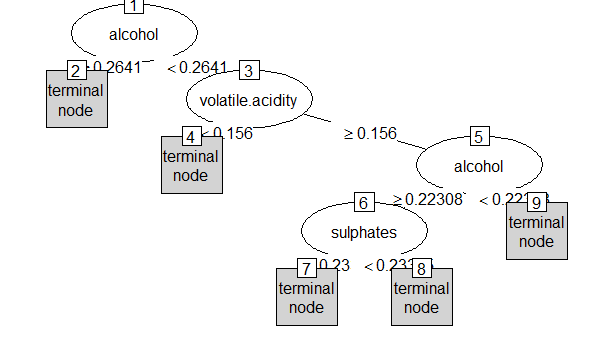
# Decision Tree

A classification tree searches through each predictor to find a value of a single variable that best splits the data into groups. The process is repeated until a hierarchical structure is created. In this three-class wine classification problem, all input variables are in real values, the same variable could be asked multiple times in order to split data. There are many trees or prediction models that can possibly split the data and make a classification decision. Splitting would stop when the minimum number of observation is reach or simply nothing is found in the split.

In this experiment, a full tree is built first, the pruning and over pruning will be examined.

In rpart package, Cp Cost Complexity parameter can be configured and to specify the split or the cross relative errors.

## 3.1 Decision Tree



### 3.1.1 Training

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CP | Accuracy | Kappa | Accuracy SD | Kappa SD |
| 0 | 0.712 | 0.41 | 0.0162 | 0.0335 |
| 0.00552 | 0.712 | 0.386 | 0.0138 | 0.0288 |
| 0.01104 | 0.709 | 0.368 | 0.0128 | 0.0284 |
| 0.01657 | 0.709 | 0.368 | 0.0128 | 0.0284 |
| 0.02209 | 0.709 | 0.368 | 0.0128 | 0.0284 |
| 0.02761 | 0.709 | 0.368 | 0.0128 | 0.0284 |
| 0.03313 | 0.709 | 0.368 | 0.0128 | 0.0284 |
| 0.03865 | 0.709 | 0.368 | 0.0128 | 0.0284 |
| 0.04418 | 0.709 | 0.368 | 0.0128 | 0.0284 |
| 0.0497 | 0.709 | 0.368 | 0.0128 | 0.0284 |
| 0.05522 | 0.709 | 0.368 | 0.0128 | 0.0284 |
| 0.06074 | 0.709 | 0.368 | 0.0128 | 0.0284 |
| 0.06626 | 0.709 | 0.368 | 0.0128 | 0.0284 |
| 0.07178 | 0.709 | 0.368 | 0.0128 | 0.0284 |
| 0.07731 | 0.709 | 0.368 | 0.0128 | 0.0284 |
| 0.08283 | 0.709 | 0.368 | 0.0128 | 0.0284 |
| 0.08835 | 0.709 | 0.368 | 0.0128 | 0.0284 |
| 0.09387 | 0.709 | 0.368 | 0.0128 | 0.0284 |
| 0.09939 | 0.709 | 0.368 | 0.0128 | 0.0284 |
| 0.10492 | 0.706 | 0.366 | 0.0162 | 0.0298 |
| 0.11044 | 0.695 | 0.356 | 0.0299 | 0.0405 |
| 0.11596 | 0.68 | 0.343 | 0.0291 | 0.0372 |
| 0.12148 | 0.67 | 0.338 | 0.0218 | 0.0338 |
| 0.127 | 0.668 | 0.339 | 0.0203 | 0.0349 |
| 0.13253 | 0.668 | 0.339 | 0.0203 | 0.0349 |
| 0.13805 | 0.665 | 0.327 | 0.0226 | 0.0707 |
| 0.14357 | 0.66 | 0.314 | 0.0228 | 0.0911 |
| 0.14909 | 0.658 | 0.303 | 0.025 | 0.1076 |
| 0.15461 | 0.652 | 0.277 | 0.0272 | 0.1292 |
| 0.16014 | 0.625 | 0.145 | 0.0261 | 0.1585 |

In the above long list of models during training, highest accuracy ends up in the second one with CP = 0.00552.

### 3.1.2 Prediction and Analysis

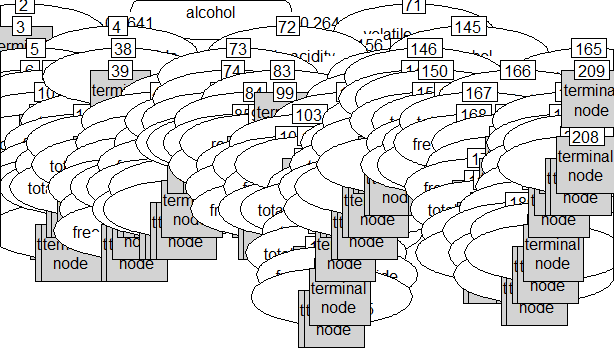
The prediction confusion matrix is shown below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Full Tree |  | Prediction Class | | |
|  |  | Best | Better | Good |
| Actual Class | Best | 0 | 39 | 0 |
| Better | 0 | 642 | 121 |
| Good | 0 | 259 | 217 |

|  |  |
| --- | --- |
|  | Full Tree |
| Accuracy | 0. 6772 |
| 95% CI | (0.651, 0.7026) |
| No Information Rate | 0. 7396 |
| P-Value [Acc > NIR] | 1 |
| Kappa | 0.2957 |

The prediction is not only lower the training and cross-validation and the lowest among the other supervised learner.

### 3.1.2 Under-Pruned Tree (CP 0.01104)



The above tree reflects the under-pruned scenario when CP is 0.01104.

The two tables below show the result of prediction.

Definitely, in the above decision tree, the more questions we asked, the higher chances to find the right answer. This is exactly right, this tree can identify Best class correctly but only one.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Under-Pruned Tree |  | Prediction Class | | |
|  |  | Best | Better | Good |
| Actual Class | Best | 1 | 37 | 1 |
| Better | 0 | 639 | 144 |
| Good | 0 | 168 | 308 |

|  |  |
| --- | --- |
|  | Under-Pruned |
| Accuracy | 0. 7304 |
| 95% CI | (0.7053, 0.7543) |
| No Information Rate | 0. .6502 |
| P-Value [Acc > NIR] | 3.945e-10 |
| Kappa | 0.4379 |

### 3.1.3 Over-Pruned Tree (CP 0.127)

## 

The above reflects the over-pruned scenario when CP is 0.0127.

The below shows the result of prediction.

It is not surprising that this single alcohol variable can determine Better and Good classes with accuracy of 0.6348, that is more than half. Even not a wine expert, regular wine drinkers usually consider the higher the alcohol content, the higher class of the wine.

This decision tree also tells us that at the 0.2641 level, it can determine Better and Good classes.

Well, Best class still requires other variables to determine.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Over-Pruned Tree |  | Prediction Class | | |
|  |  | Best | Better | Good |
| Actual Class | Best | 0 | 37 | 0 |
| Better | 0 | 508 | 275 |
| Good | 0 | 160 | 316 |

|  |  |
| --- | --- |
|  | Over-Pruned |
| Accuracy | 0. 6348 |
| 95% CI | (0.608, 0.6611) |
| No Information Rate | 0. 5431 |
| P-Value [Acc > NIR] | 1.403e-11 |
| Kappa | 0.2766 |

# Neural Network

Neural network is power and mature machine learning algorithm. It can work quite well in nonlinear regression and time series problems. Weights can be created and applied to some functions such as sigmoid function for classification problem like our three-class wine quality classification.

Number of neurons and layers will be examined. The default threshold will be taken from the package and maximum iteration is set to 1000 to find the global minimum.

In the caret avNNet, we can observe to weight decay which is related to regularization of cost function. Whereas, the multiple layer perceptron uses mlp, learning is not required and so as the decay weight.

## Training and Cross Valuation

Both mlp and avNNet have selected 5 neurons, single layer as the best candidate in the training process. avNNet has ran through three different weight decay and 0.0001 has been selected. This indicated that slightly weight penalization has helped with accuracy. The size of neurons are limited due to computer hardware performance. It is logical to guess that as the number of neurons in both mlp and avNNet could increase the accuracy.

|  |  |  |  |
| --- | --- | --- | --- |
| size | decay | Accuracy | Accuracy (mlp) |
| 1 | 0 | 0.659 | 0.712 |
| 1 | 0.0001 | 0.712 |  |
| 1 | 0.1 | 0.712 |  |
| 3 | 0 | 0.679 | 0.721 |
| 3 | 0.0001 | 0.729 |  |
| 3 | 0.1 | 0.723 |  |
| 5 | 0 | 0.673 | 0.726 |
| 5 | 0.0001 | 0.738 |  |
| 5 | 0.1 | 0.731 |  |

## 2.2 Prediction Results and Analysis

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| avNNet |  | Prediction Class | | |
|  |  | Best | Better | Good |
| Actual Class | Best | 0 | 39 | 0 |
| Better | 0 | 644 | 139 |
| Good | 0 | 180 | 296 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| mlp |  | Prediction Class | | |
|  |  | Best | Better | Good |
| Actual Class | Best | 0 | 38 | 1 |
| Better | 0 | 625 | 158 |
| Good | 0 | 131 | 345 |

In the above confusion matrix, avNNet has poor prediction on Good class. It is unclear to me that mlp has or has not created a training model to have multiple layer (*some technical challenges to find the right packages and parameters to clearly specify the layer*) that has done better prediction on Good class. Other packages such as qrnn and neuralnet can be used to specify layers. Unfortunately, both neural network can predict Best class.

|  |  |  |
| --- | --- | --- |
|  | mlp | avNNet |
| Accuracy | 0. 7473 | 0.7242 |
| 95% CI | (0.7227, 0.7707) | (0.699, 0.7484) |
| No Information Rate | 0. 6117 | 0.6649 |
| P-Value [Acc > NIR] | < 2.2e-16 | 2.45e-06 |
| Kappa | 0. 4828 | 0.4206 |

The mlp accuracy on the test dataset has outperformed the training cross-validation a bit vs the avNNet accuracy has underperformed the training cross- validation. The underperformance could be caused by over-fitting using weight decay for regularization vs mlp outperform might have better generalization.

|  |  |  |  |
| --- | --- | --- | --- |
|  | 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 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Best | Better | Good |
| Sensitivity | NA | 0. 7872 | 0. 6845 |
| Specificity | 0.96995 | 0. 6865 | 0. 8350 |
| Pos Pred Value | NA | 0. 7982 | 0. 8066 |
| Neg Pred Val | NA | 0. 6718 | 0. 3883 |
| Prevalence | 0 | 0. 6117 | 0.11479 |
| Detection Rate | 0 | 0. 4815 | 0. 2658 |
| Detection Prevalence | 0.03005 | 0.6032 | 0.36672 |
| Balanced Accuracy | NA | 0. 7368 | 0. 7598 |

The accuracy of using neural network came at high cost of training effort and computers. Instance based learner, no training, K-nearest neighbor has big contracts to neural network.

# Support Vector Machine

Basic idea of SVM is to maximize the margins of hyper-plane to separate two or multiple classes.

Cost and two different kernels were applied to this experiment to find the model with highest accuracy.

## Training and Cross Valuation

The following costs (0.25, 0.5, 1.0, and 1.5) were used in both Linear and Radial. 0.02 has been chosen as Sigma for Radial Kernel.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | C | Accuracy | Kappa | Accuracy SD | Kappa SD |
| Linear | 0.25 | 0.718 | 0.4 | 0.0172 | 0.0379 |
| 0.5 | 0.718 | 0.4 | 0.0174 | 0.0382 |
| 1 | 0.718 | 0.4 | 0.0174 | 0.0381 |
| 1.5 | 0.718 | 0.4 | 0.0174 | 0.0383 |
| Basic Radial(Sigma=0.02) | 0.25 | 0.72 | 0.396 | 0.0168 | 0.0374 |
| 0.5 | 0.723 | 0.406 | 0.018 | 0.0386 |
| 1 | 0.728 | 0.418 | 0.0187 | 0.0397 |
| 1.5 | 0.73 | 0.422 | 0.0176 | 0.0376 |

Radial with sigma = 0.02 has achieved slightly higher accuracy when C reaches 1.5. In fact the accuracy of Linear Kernel remains the same for four Cs versus Radial kernel has achieved higher as the C increases. The parameter C relates to the cost function which weighs on the input variables and sigma to kernel that maps the input data to a feature space. Gaussian kernel is one of common non-linear kernels. As C increase and various sigma testing, the accuracy could also be improved using Radial kernel. This strongly suggested that the three-class wine quality classification can be easily separated by linear function.

## Prediction Results and Analysis

The following three tables show the results of the prediction on the test set using C = 1 with Linear Kernel; and C=2 and sigma = 0.02 for Radial Kernel.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | 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 |

In the above confusion matrix, SVM has significant poor prediction on Good class. This suggested that non-linear SVM has harder time to predict on Good in this dataset. Further examination of data and selection of input variables and of course using non-linear kernel with various sigma would improve the accuracy.

It is obvious that both SVMs are not capable of predicting the Best class. It could be Best class wine is very subjective the wine experts and patterns cannot not found from the input variables.

|  |  |  |
| --- | --- | --- |
|  | 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 |
|  |  |  |
|  |  |  |

The prediction accuracy remains very close to training and in fact there is a bit improvement in accuracy. It concludes that test dataset migh have fewer extreme cases; and k-fold cross validation, resample has reduced high variance and improve the generalization.

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |

The above two tables reinforces the previous discussion that SVM Linea kernel has significant lower sensitivity to Good class than SVM Radial kernel.

# *k*-nearest neighbors

K-nearest neighbors uses similarities to group or classify them into the same class. Let takes this three-class wine quality classification problem, and k to be number of nearest neighbors in terms of Euclidean distance (R knn package) based on our 10 predicators.

In the training, different Ks will be used in the training and the k with highest accuracy will be chosen to be used in the prediction.

## 4.1 Training and Cross Validation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| k | Accuracy | Kappa | Accuracy SD | Kappa SD |
| 1 | 0.745 | 0.49 | 0.0189 | 0.0371 |
| 2 | 0.698 | 0.396 | 0.0148 | 0.0298 |
| 3 | 0.714 | 0.417 | 0.0208 | 0.0411 |
| 5 | 0.718 | 0.418 | 0.016 | 0.0332 |
| 10 | 0.718 | 0.41 | 0.0165 | 0.0347 |
| 20 | 0.721 | 0.408 | 0.0163 | 0.0354 |
| 30 | 0.72 | 0.407 | 0.017 | 0.0372 |
| 50 | 0.717 | 0.398 | 0.0147 | 0.0333 |

Highest accuracy is achieved when k is 1. There is an accuracy drop when k increases to 2. From k =3 to k=30, the accuracy has increased slightly then the k has dropped again. The highest accuracy is achieved when k=1, it means the object is assigned to the class of that single nearest neighbor. A small k usually means that noise will have higher influence on the result. Further analysis on the predictors or assigning more weight on critical predictor might improve the accuracy and increase k. When k=1, Kappa is also the highest which can be seen good agreement between predictions and actuals.

## 4.2 Prediction Results and Analysis

The following three tables show the results of the prediction on the test set using k=1.

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

Compare to other learners, only knn is capable of predicting Best correctly. Two ones could be caused by some special input which have similar distance but they actually fall on the other ends.

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

The prediction accuracy remains very close to training. It concludes that k-fold cross validation and resample has reduced high variance and improve the generalization.

|  |  |  |  |
| --- | --- | --- | --- |
|  | 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 Tree

Boosting tree is an implementation of ensemble method where many trees or weaker learners are combined and prediction are aggregated across tree. Unlike the previous single tree where some predicator are used in the selected tree and usually has lower accuracy.

## Training and Cross Validation

Number of trials is the tuning parameter to determine number of boosting iterations.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| model | winnow | Trials | Accuracy | Kappa | Accuracy SD | Kappa SD |
| rules | FALSE | 1 | 0.739 | 0.459 | 0.0118 | 0.0242 |
| rules | FALSE | 10 | 0.753 | 0.494 | 0.0149 | 0.0292 |
| rules | FALSE | 20 | 0.754 | 0.495 | 0.0148 | 0.0289 |
| rules | FALSE | 30 | 0.756 | 0.499 | 0.0162 | 0.0328 |
| rules | FALSE | 40 | 0.755 | 0.498 | 0.0172 | 0.035 |
| rules | FALSE | 50 | 0.755 | 0.498 | 0.0174 | 0.0354 |
| rules | FALSE | 60 | 0.755 | 0.499 | 0.0171 | 0.0346 |
| rules | FALSE | 70 | 0.755 | 0.499 | 0.0171 | 0.0346 |
| rules | FALSE | 80 | 0.755 | 0.499 | 0.0171 | 0.0346 |
| rules | FALSE | 90 | 0.755 | 0.499 | 0.0171 | 0.0346 |
| : | : | : | : | : | : | : |
| tree | TRUE | 1 | 0.732 | 0.447 | 0.0145 | 0.0313 |
| tree | TRUE | 10 | 0.755 | 0.489 | 0.0153 | 0.0326 |
| tree | TRUE | 20 | 0.753 | 0.486 | 0.0159 | 0.0342 |
| tree | TRUE | 30 | 0.753 | 0.486 | 0.0162 | 0.0351 |
| tree | TRUE | 40 | 0.753 | 0.485 | 0.0174 | 0.0371 |
| tree | TRUE | 50 | 0.755 | 0.49 | 0.0188 | 0.0401 |
| tree | TRUE | 60 | 0.755 | 0.489 | 0.0194 | 0.0413 |
| tree | TRUE | 70 | 0.754 | 0.487 | 0.0177 | 0.0381 |
| tree | TRUE | 80 | 0.754 | 0.488 | 0.0186 | 0.04 |
| tree | TRUE | 90 | 0.754 | 0.488 | 0.0187 | 0.0403 |
| tree | TRUE | 100 | 0.756 | 0.49 | 0.0196 | 0.0419 |