Quality of Wine Capstone Project

Meghan Patterson

Abstract

A variety of machine learning techniques are incorporated into this project in order to determine the best model for predicting the quality of wine. The models created incorporate logistic regression, linear discriminant analysis, k-nearest neighbors (KNN), support vector machine (SVM), decision trees and random forests. The best model was chosen based off of the highest accuracy. In this project, the highest accuracy was associated with the random forest model and that accuracy was a value of 0.8529.

Contents

1 Introduction

Machine learning techniques can be implemented into many industries, including those such as entertainment, food, and healthcare. What some consider as a subset of the food industry is the wine industry. Over the past few years, machine learning and artificial intelligence has revolutionized the wine industry by being able to craft a better product and produce better sales forecasts in the highly competitive market. In this project, a wine dataset is used in order to predict the quality of the wine based off of a series of predictors.

The dataset used in this project is the wine quality dataset originating from the UCI Machine Learning Repository. This dataset has 6463 rows, 13 columns, 2 types of wine (red and white), a

quality dependent variable and 11 additional variables that will be considered as predictors. In the dataset, there are 1593 cases of red wine and 4870 cases of white wine. Because this variable is only an identification of the wine used in each case and has no effect on the quality, this variable will not be included when creating the model. Additionally, the quality variable has been reconstructed into a factor with two levels: 0 when the quality is less than or equal to 5 and 1 when the quality is greater than 5. This was done in order to improve the efficiency and prediction of the models given that there is now only 2 levels (2 categories) instead of the original 9. When creating and testing the models, the dataset was split into a total of three subsets: a training set, a test set and a validation set. The training and test sets will be used to construct and tune the models, while the validation set will be tested on the final validation set in the end.

The machine learning techniques that are included in this project include logistic regression, linear discriminant analysis, k-nearest neighbors (KNN), support vector machine (SVM), decision trees and random forests. These supervised machine learning algorithms are all considered classification algorithms. These methods were chosen to be implemented into the project due to the main objective of predicting whether a dataset of wine is high quality or low quality.

2 Methods

2.1 Exploratory Data Analysis

The structure of the wine dataset shown below illustrates that there are 11 numeric variables, a factor variable named "type" that holds whether the wine was red or white, and a integer variable that holds values denoting the quality of each wine.

```
## 'data.frame':    6497 obs. of  13 variables:
##  $ type                : chr  "white" "white" "white" "white" ...
##  $ fixed.acidity       : num  7 6.3 8.1 7.2 7.2 8.1 6.2 7 6.3 8.1 ...
##  $ volatile.acidity    : num  0.27 0.3 0.28 0.23 0.23 0.28 0.32 0.27 0.3 0.22 ...
##  $ citric.acid         : num  0.36 0.34 0.4 0.32 0.32 0.4 0.16 0.36 0.34 0.43 ...
##  $ residual.sugar      : num  20.7 1.6 6.9 8.5 8.5 6.9 7 20.7 1.6 1.5 ...
##  $ chlorides           : num  0.045 0.049 0.05 0.058 0.058 0.05 0.045 0.045 0.049 0.044 ...
##  $ free.sulfur.dioxide : num  45 14 30 47 47 30 30 45 14 28 ...
##  $ total.sulfur.dioxide: num  170 132 97 186 186 97 136 170 132 129 ...
##  $ density             : num  1.001 0.994 0.995 0.996 0.996 ...
##  $ pH                  : num  3 3.3 3.26 3.19 3.19 3.26 3.18 3 3.3 3.22 ...
##  $ sulphates           : num  0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.45 ...
##  $ alcohol             : num  8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
##  $ quality             : int  6 6 6 6 6 6 6 6 6 6 ...
```

Because the "type" variable is an identifier for what type of wine was used, it will be removed from the dataset. Additionally, the "quality" variable will be recreated into a factor variable with two levels: a 0 will take the place of any value that is less than or equal to 5, and a 1 will take the place of any value that is above 5. With the removal of the 38 NAs that were found in the dataset and tne adjustments to be made noted above, the wine dataset now has the structure below.

```
## 'data.frame':    6463 obs. of  12 variables:
##  $ fixed.acidity       : num  7 6.3 8.1 7.2 7.2 8.1 6.2 7 6.3 8.1 ...
##  $ volatile.acidity    : num  0.27 0.3 0.28 0.23 0.23 0.28 0.32 0.27 0.3 0.22 ...
```

```
##  $ citric.acid         : num  0.36 0.34 0.4 0.32 0.32 0.4 0.16 0.36 0.34 0.43 ...
##  $ residual.sugar      : num  20.7 1.6 6.9 8.5 8.5 6.9 7 20.7 1.6 1.5 ...
##  $ chlorides           : num  0.045 0.049 0.05 0.058 0.058 0.05 0.045 0.045 0.049 0.044 ...
##  $ free.sulfur.dioxide : num  45 14 30 47 47 30 30 45 14 28 ...
##  $ total.sulfur.dioxide: num  170 132 97 186 186 97 136 170 132 129 ...
##  $ density             : num  1.001 0.994 0.995 0.996 0.996 ...
##  $ pH                  : num  3 3.3 3.26 3.19 3.19 3.26 3.18 3 3.3 3.22 ...
##  $ sulphates           : num  0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.45 ...
##  $ alcohol             : num  8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
##  $ quality_new         : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  - attr(*, "na.action")= 'omit' Named int [1:34] 18 34 55 87 99 140 175 225 250 268 ...
##   ..- attr(*, "names")= chr [1:34] "18" "34" "55" "87" ...
```

The "quality_new" variable is the dependent variable for the project. The proportion of values associated with this variable can be seen below.

In order to visualize the differences between the two qualities of wine and the predictor variables, boxplots can be made to show some of these distinctions. Figure 1 below shows a boxplot for the two qualities of wine and wine acidity. This boxplot demonstrates that the fixed acidity for the high quality of wine is slighter lower than that for the low quality of wine.

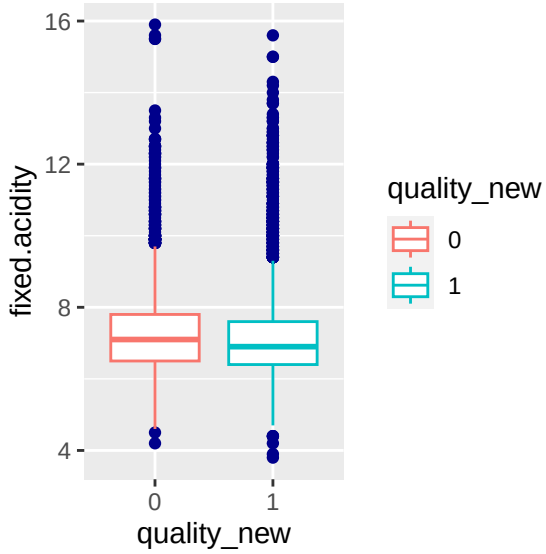

Figure 1: Boxplot of the Two Qualities of Wine and the Fixed Acidity

Figure 2 below shows a boxplot for the two qualities of wine and volatile acidity. This boxplot illustrates that the volatile acidity for the high quality wine is slightly lower than the volatile acidity for the low quality wine.

Figure 3 below shows a boxplot for the two qualities of wine and the amount of total sulfur dioxide. In this boxplot, the total sulfur dioxide amount in the low quality wine is slightly higher than that for the high quality wine. Additionally, the outliers for the lower quality wine are much higher than those for the higher quality wine.

Figure 4 below showcases a boxplot for the two qualities of wine and the pH. This boxplot illustrates that the higher quality wine has a slighly higher pH than that for the lower quality wine.

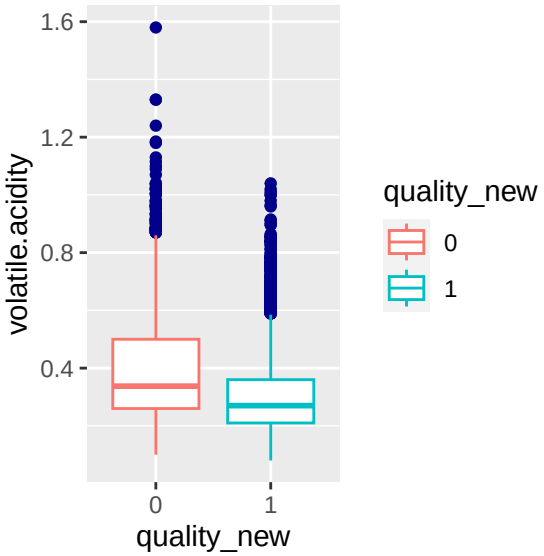

Figure 2: Boxplot of the Two Qualities of Wine and the Volatile Acidity

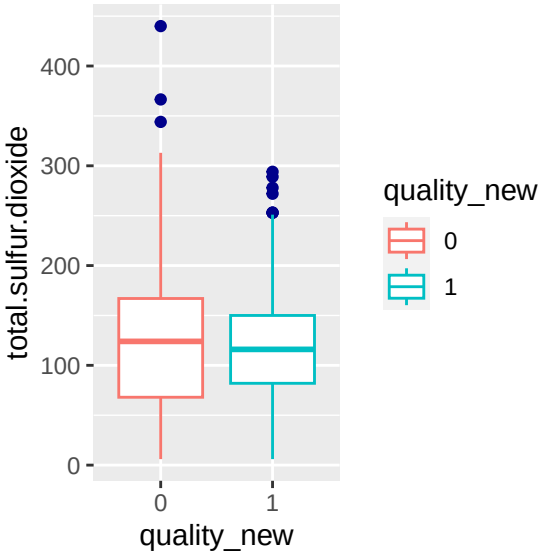

Figure 3: Boxplot of the Two Qualities of Wine and the Total Sulfur Amount

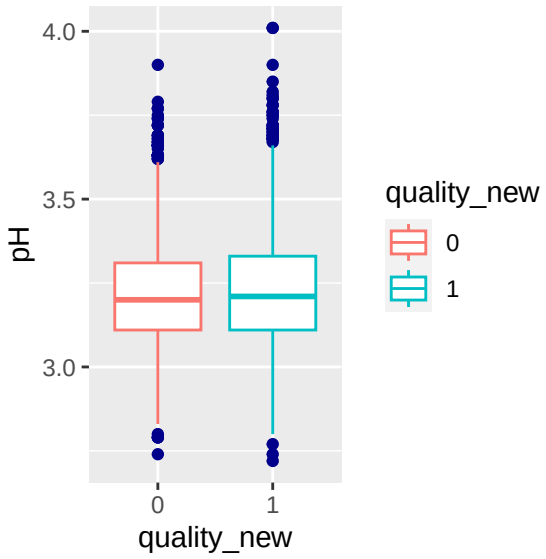

Figure 4: Boxplot of the Two Qualities of Wine and the pH

Figure 5 below shows a boxplot for the two qualities of wine and the citric acid amount. This boxplot displays a higher median value for the higher quality wine, but a lower range of values for the higher quality wine compared to the lower quality wine

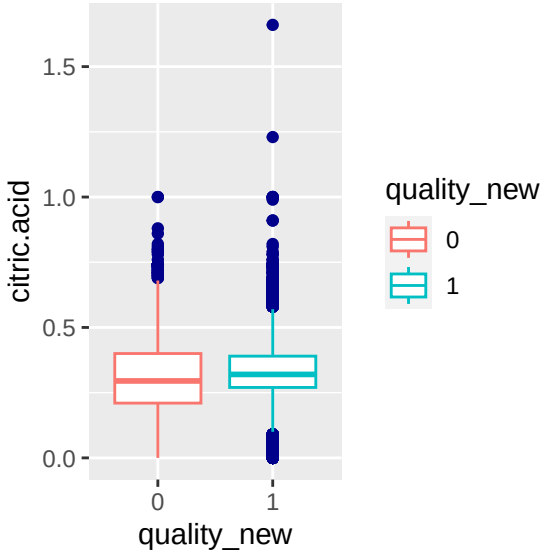

Figure 5: Boxplot of the Two Qualities of Wine and the Citric Acid Amount

A correlation plot can be constructed with the predictors in order to visualize which predictors have a higher correlation than others. Figure 6 below displays this correlation plot. From this plot, it can be seen that the free sulfur dioxide and total sulfur dioxide have the highest correlation with a value of 0.7. The density and residual sugar have a correlation of 0.6, the density and fixed acidity have a correlation of 0.5, and the total sulfur dioxide and residual sugar have a correlation of 0.5. The remaining combinations have correlations ranging from -0.7 to 0.4. These correlations will be taken into consideration when constructing the models.

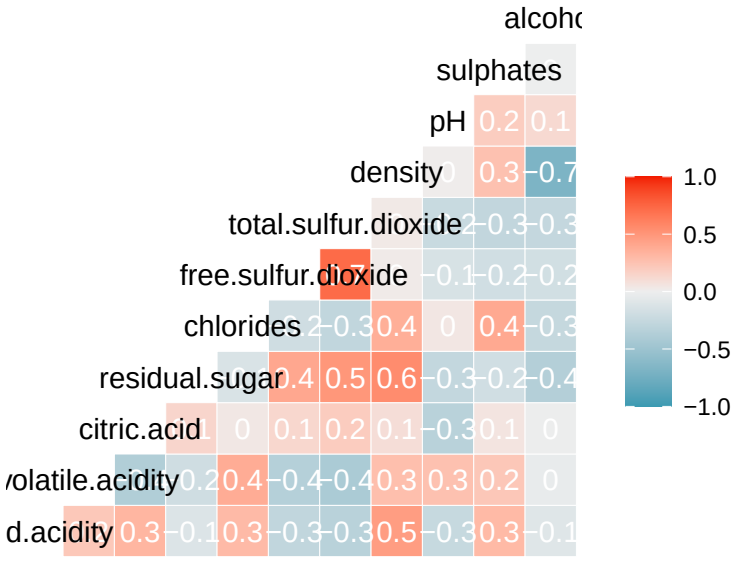

Figure 6: Correlation Plot of the Predictors

2.2 Model Preparation

The wine dataset will be divided into three total subsets: a training set, a test set, and the validation set. The training and test sets will be used to construct and tune the models. The test set will be used in the prediction for these models, and the model with the highest accuracy will be deemed the best model. This best model will then be applied to the validation set and the final accuracy will be obtained and recorded. The training set will receive a total of 80% of the data, the test set will receive a total of 10% of the data, and the validation set will receive a total of 10% of the data. The partitioning of the dataset(s) can be seen below.

```
split_index <- createDataPartition(wine_data$quality_new, p = 0.9, times = 1, list = FALSE)
model_data <- wine_data[split_index, ]
validation_set <- wine_data[-split_index, ]

train_index <- createDataPartition(model_data$quality_new, p = 0.9, times = 1, list = FALSE)
training_set <- model_data[train_index, ]
test_set <- model_data[-train_index, ]
```

In order to ensure that the partitions were successful, proportion tables can be constructed on the training and test sets. The proportion tables below confirm that the partitions were successful because of the relatively same amounts of low quality and high quality wine in each.

```
prop.table(table(training_set$quality_new))
```

```
##
##         0         1
## 0.3670741 0.6329259
```

```
prop.table(table(test_set$quality_new))
```

```
##
##         0         1
```

```
## 0.3666093 0.6333907
```

2.3 Models

2.3.1 Logistic Regression Model

The first model to be constructed is a logistic regression model. The logistic regression model is appropriate for this dataset given that the dependent variable is binary. The data used to create the model can be seen below.

```
glm_model <- train(quality_new ~ ., method = "glm", data = training_set)
```

This model uses the train() function in the caret package with the method set to "glm." The "glm" method specifies a generalized linear model. The prediction can be calculated with the code below. The test set is now being used in order to make the prediction.

```
y_pred_glm <- predict(glm_model, test_set, type = "raw")
```

With the prediction, a confusion matrix can now be constructed and can be seen below.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 118  66
##          1  95 302
##
##                Accuracy : 0.7229
##                  95% CI : (0.6846, 0.7589)
##     No Information Rate : 0.6334
##     P-Value [Acc > NIR] : 3.062e-06
##
##                   Kappa : 0.3857
##
##  Mcnemar's Test P-Value : 0.02733
##
##             Sensitivity : 0.5540
##             Specificity : 0.8207
##          Pos Pred Value : 0.6413
##          Neg Pred Value : 0.7607
##              Prevalence : 0.3666
##          Detection Rate : 0.2031
##    Detection Prevalence : 0.3167
##       Balanced Accuracy : 0.6873
##
##        'Positive' Class : 0
##
```

The confusion matrix indicates that the accuracy for the model is 0.7228. Additionally, the sensitivity is 0.5540 and the precision is 0.6413. Additional models will be constructed in order to attempt to improve (increase) these values.

Table 1 below shows the accuracy results for the model.

Table 1: Logistic Regression Model and Corresponding Accuracy Results

	Model	Accuracy
Accuracy	Logistic Regression	0.7228916

2.3.2 Linear Discriminant Analysis (LDA) Model

The second model to be constructed is the linear discriminant analysis (LDA) model. This method is also suitable for the binary classification problem at hand. The data used to create the model can be seen below.

```
lda_model <- train(quality_new~., method = "lda", data = training_set,
                   trControl = trainControl(method = "cv"))
```

This model uses the train() function in the caret package with the method set to "lda." The "lda" method specifies linear discriminant analysis. The prediction can be calculated with the code below. The test set is now being used in order to make the prediction.

```
y_pred_lda <- predict(lda_model, test_set)
```

With the prediction, a confusion matrix can now be constructed and can be seen below.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 116  64
##          1  97 304
##
##                Accuracy : 0.7229
##                  95% CI : (0.6846, 0.7589)
##     No Information Rate : 0.6334
##     P-Value [Acc > NIR] : 3.062e-06
##
##                   Kappa : 0.3832
##
##  Mcnemar's Test P-Value : 0.01167
##
##             Sensitivity : 0.5446
##             Specificity : 0.8261
##          Pos Pred Value : 0.6444
##          Neg Pred Value : 0.7581
##              Prevalence : 0.3666
##          Detection Rate : 0.1997
##    Detection Prevalence : 0.3098
##       Balanced Accuracy : 0.6853
```

```
##
##        'Positive' Class : 0
##
```

The confusion matrix indicates that the accuracy for the model is 0.7228. Additionally, the sensitivity is 0.5546 and the precision is 0.6444. These values are similar to those found with the logistic regression model, but this is understandable due to the similarity between the two approaches in the models. Additional models will be constructed in order to attempt to improve (increase) these values.

Table 2 below shows the accuracy results for the model added to the existing table.

Table 2: Linear Discriminant Analysis Model and Corresponding Accuracy Results

	Model	Accuracy
Accuracy	Logistic Regression	0.7228916
...2	Linear Discriminant Analysis (LDA)	0.7228916

2.3.3 K-Nearest Neighbors Model (KNN) Model

The third model to be constructed is the k-nearest neighbors (KNN) model. The data used to create the model can be seen below.

```
ctrl <- trainControl(method = "repeatedcv", repeats = 5)
knn_model <- train(quality_new~., method = "knn", data = training_set,
                   trControl = ctrl)
knn_model
```

```
## k-Nearest Neighbors
##
## 5236 samples
##   11 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 4712, 4712, 4712, 4712, 4712, 4713, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.6704747  0.2705191
##   7  0.6697486  0.2618538
##   9  0.6713931  0.2587150
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```

This model uses the train() function from the caret package with the method set to "knn." The "knn" method specifies k-nearest neighbors.

A plot can be constructed with the model in order to showcase the number of neighbors versus the accuracy. Figure 7 below demonstrates this concept.

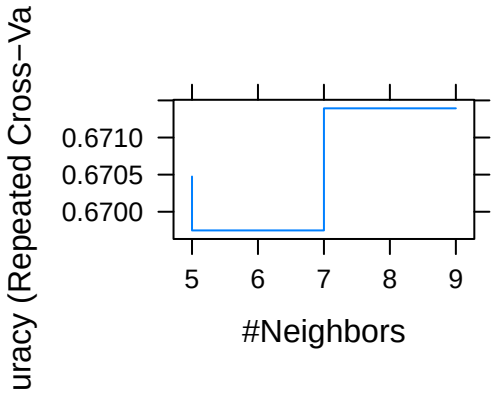

Figure 7: Number of Neighbors versus Accuracy for the KNN Model

Figure 7 illustrates that the accuracy is increased when the number of neighbors is between 5 and 7-9. Thus, the ideal value of k is within those range of values. The data above tells that the optimal value of k is 5.

The prediction can be calculated with the code below. The test set is now being used in order to make the prediction. The test set is now being used in order to make the prediction.

```
y_pred_knn <- predict(knn_model, test_set)
```

With the prediction, a confusion matrix can now be constructed and can be seen below.

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction   0   1
##          0 103  84
##          1 110 284
##
##                Accuracy : 0.6661
##                  95% CI : (0.6261, 0.7044)
##     No Information Rate : 0.6334
##     P-Value [Acc > NIR] : 0.05490
##
##                   Kappa : 0.262
##
##  Mcnemar's Test P-Value : 0.07267
##
##             Sensitivity : 0.4836
##             Specificity : 0.7717
##          Pos Pred Value : 0.5508
##          Neg Pred Value : 0.7208
##              Prevalence : 0.3666
##          Detection Rate : 0.1773
```

```
##    Detection Prevalence : 0.3219
##       Balanced Accuracy : 0.6277
##
##        'Positive' Class : 0
##
```

The confusion matrix indicates that the accuracy for the model is 0.6660. Additionally, the sensitivity is 0.4836 and the precision is 0.5508. The results for this model are the lowest seen thus far. Because of this, it can already be suspected that the KNN model will not be the ideal model for the dataset. Additional models will be constructed in order to attempt to improve (increase) these values.

Table 3 below shows the accuracy results for the model added to the existing table.

Table 3: K-Nearest Neighbors Model and Corresponding Accuracy Results

	Model	Accuracy
Accuracy	Logistic Regression	0.7228916
...2	Linear Discriminant Analysis (LDA)	0.7228916
...3	K-Nearest Neighbors (KNN)	0.6660929

2.3.4 Support Vector Machine (SVM) Model

The fourth model to be constructed is the support vector machine (SVM) model. In the SVM model, there is a cost of constraints violation term. The ideal model would have the optimal value for the cost term. In order to find this optimal value, a for-loop is created with different values of "k" in order to find the cost term that gives the highest accuracy. The for-loop can be seen below.

```
Y <- NULL
k <- 30
for (i in 1:k) {
  svm.fit <- svm(quality_new~., training_set, scale = FALSE, kernel = "linear", cost = i)
  pred_svm <- predict(svm.fit, test_set)
  conf <- confusionMatrix(pred_svm, test_set$quality_new)
  Y[i] <- conf$overall[1]
}
```

The data from this for-loop can be transformed into a plot in order to visually see the trend of k values and corresponding accuracy values. Figure 8 below shows this plot and the vertical line on the plot indicates the cost term that holds the highest accuracy.

```
plot(1:k, Y, pch = 20, xlab = "Cost", ylab = "Accuracy")
abline(v = which.max(Y))
```

The plot indicates that the cost term equaled to 4 has the highest accuracy. Therefore, this value will be used in construction of the model.

The data for the model can be seen below.

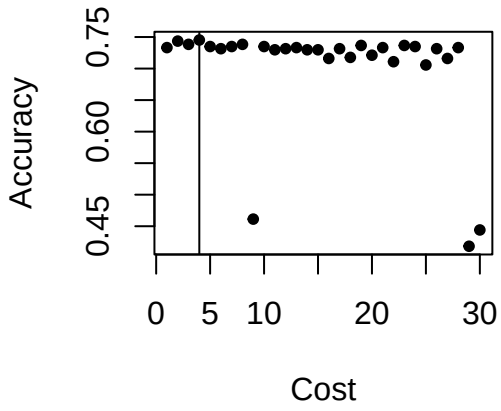

Figure 8: Cost Terms and Corresponding Accuracies

```
svm_fit <- svm(quality_new~., training_set, scale = FALSE, kernel = "linear", cost = 4)
```

With this ideal model, the prediction can be calculated with the test set.

```
svm_pred <- predict(svm_fit, test_set)
```

With the prediction, a confusion matrix can now be constructed and can be seen below.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 110  45
##          1 103 323
##
##                Accuracy : 0.7453
##                  95% CI : (0.7078, 0.7802)
##     No Information Rate : 0.6334
##     P-Value [Acc > NIR] : 5.922e-09
##
##                   Kappa : 0.4181
##
##  Mcnemar's Test P-Value : 2.795e-06
##
##             Sensitivity : 0.5164
##             Specificity : 0.8777
##          Pos Pred Value : 0.7097
##          Neg Pred Value : 0.7582
##              Prevalence : 0.3666
##          Detection Rate : 0.1893
##    Detection Prevalence : 0.2668
##       Balanced Accuracy : 0.6971
##
##        'Positive' Class : 0
```

##

This confusion matrix indicates that the accuracy for the model is 0.7452. Additionally, the sensitivity is 0.5164 and the precision is 0.7097. The accuracy with the SVM model is the highest accuracy thus far. This is likely due to the fine-tuning (finding the optimal cost term) that was performed. This high-accuracy model will be taken into account when choosing the best model. Additional models will be constructed in order to attempt to improve (increase) these values.

Table 4 below shows the accuracy results for the model added to the existing table.

Table 4: Support Vector Machine Model and Corresponding Accuracy Results

	Model	Accuracy
Accuracy	Logistic Regression	0.7228916
...2	Linear Discriminant Analysis (LDA)	0.7228916
...3	K-Nearest Neighbors (KNN)	0.6660929
...4	Support Vector Machine (SVM)	0.7452668

2.3.5 Decision Tree Model

The fifth model to be constructed is the decision tree model. The data used to create the model can be seen below.

```
ctrl <- rpart.control(minsplit = 5L, maxdepth = 5L, minbucket = 5, cp = .002, maxsurrogate = 4)
dec_tree.model <- rpart(quality_new~., training_set, method = "class", control = ctrl)
```

This model uses the rpart() function that splits the data recursively and creates decision trees.

A decision tree plot can be constructed with the results from the model. This model can be seen in Figure 10 below.

```
prp(dec_tree.model)
```

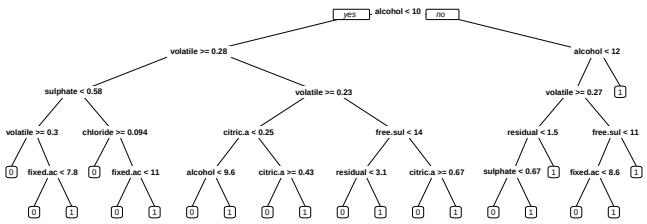

Figure 9: Decision Tree Plot

The results in Figure 9 indicate which predictors have a higher influence than others. This includes "alcohol", "volatile", and "sulphate."

The prediction can be calculated with the code below. The test set is now being used in order to make the prediction.

```r
y_pred_dec <- predict(dec_tree.model, test_set, type = "class")
```

With the prediction, a confusion matrix can now be constructed and can be seen below.

```
## Confusion Matrix and Statistics
##
##
## y_pred_dec   0   1
##          0 125  74
##          1  88 294
##
##                Accuracy : 0.7212
##                  95% CI : (0.6828, 0.7573)
##     No Information Rate : 0.6334
##     P-Value [Acc > NIR] : 4.667e-06
##
##                   Kappa : 0.3912
##
##  Mcnemar's Test P-Value : 0.3071
##
##             Sensitivity : 0.5869
##             Specificity : 0.7989
##          Pos Pred Value : 0.6281
##          Neg Pred Value : 0.7696
##              Prevalence : 0.3666
##          Detection Rate : 0.2151
##    Detection Prevalence : 0.3425
##       Balanced Accuracy : 0.6929
##
##        'Positive' Class : 0
##
```

The confusion matrix indicates that the accuracy for the model is 0.7211. Additionally, the sensitivity is 0.6432 and the precision is 0.6313. Additional models will be constructed in order to attempt to improve (increase) these values.

Table 5 below shows the accuracy results for the model added to the existing table.

Table 5: Decision Tree Model and Corresponding Accuracy Results

	Model	Accuracy
Accuracy	Logistic Regression	0.7228916
...2	Linear Discriminant Analysis (LDA)	0.7228916
...3	K-Nearest Neighbors (KNN)	0.6660929
...4	Support Vector Machine (SVM)	0.7452668
...5	Decision Tree	0.7211704

2.3.6 Random Forest Model

The sixth model to be constructed is the random forest model. The data used to create this model can be seen below.

```
control <- trainControl(method = "cv", repeats = 5)
rf_model <- train(quality_new~., training_set, method = "rf", trControl = control)
```

This model uses the train() function from the caret package with the method set to "rf." The "rf" method specifies random forest.

The prediction can be calculated with the code below. The test set is now being used in order to make the prediction.

```
y_pred_rf <- predict(rf_model, test_set)
```

With the prediction, a confusion matrix can now be constructed and can be seen below.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 149  46
##          1  64 322
##
##                Accuracy : 0.8107
##                  95% CI : (0.7764, 0.8417)
##     No Information Rate : 0.6334
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.5849
##
##  Mcnemar's Test P-Value : 0.105
##
##             Sensitivity : 0.6995
##             Specificity : 0.8750
##          Pos Pred Value : 0.7641
##          Neg Pred Value : 0.8342
##              Prevalence : 0.3666
##          Detection Rate : 0.2565
##    Detection Prevalence : 0.3356
##       Balanced Accuracy : 0.7873
##
##        'Positive' Class : 0
##
```

The confusion matrix indicates that the accuracy for the model is 0.8106. Additionally, the sensitivity is 0.7136 and the precision is 0.7755. The accuracy of 0.8106 makes the random forest model have the highest accuracy out of all the models constructed and thus, the best model for the dataset.

Table 6 below shows the accuracy results for the model added to the existing table.

Table 6: Random Forest Model and Corresponding Accuracy Results

	Model	Accuracy
Accuracy	Logistic Regression	0.7228916
...2	Linear Discriminant Analysis (LDA)	0.7228916
...3	K-Nearest Neighbors (KNN)	0.6660929
...4	Support Vector Machine (SVM)	0.7452668
...5	Decision Tree	0.7211704
...6	Random Forest	0.8106713

2.3.7 Random Forest Model - Validation Dataset

The random forest model constructed on the training set and evaluated on the test set had an accuracy of 0.8193. This accuracy of 0.8193 was the highest accuracy out of all of the models constructed and therefore will be chosen as the best model. The random forest model will be evaluated on the validation dataset that has not previously been used. In order to evaluate with the most data possible, the model_data (data subset before being split into training_set and test_set) will be used. The data used to create this model can be seen below.

```
control <- trainControl(method = "cv", repeats = 5)
rf_valid_model <- train(quality_new~., model_data, method = "rf", trControl = control)
```

The prediction can be calculated with the code below. The test set is now being used in order to make the prediction.

```
y_pred_rf_valid <- predict(rf_valid_model, validation_set)
```

With the prediction, a confusion matrix can now be constructed and can be seen below.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 166  24
##          1  71 385
##
##                Accuracy : 0.8529
##                  95% CI : (0.8232, 0.8794)
##     No Information Rate : 0.6331
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6697
##
##  Mcnemar's Test P-Value : 2.364e-06
##
##             Sensitivity : 0.7004
##             Specificity : 0.9413
##          Pos Pred Value : 0.8737
```

```
##          Neg Pred Value : 0.8443
##              Prevalence : 0.3669
##          Detection Rate : 0.2570
##    Detection Prevalence : 0.2941
##       Balanced Accuracy : 0.8209
##
##        'Positive' Class : 0
##
```

The confusion matrix indicates that the accuracy for the model is 0.8529. Additionally, the sensitivity is 0.7046 and the precision is 0.8743. Therefore, the final and best accuracy obtained for the dataset is 0.8529 from the random forest model.

Table 7 below shows the accuracy results for the model added to the existing table.

Table 7: Random Forest Model (Validation Set) and Corresponding Accuracy Results

	Model	Accuracy
Accuracy	Logistic Regression	0.7228916
...2	Linear Discriminant Analysis (LDA)	0.7228916
...3	K-Nearest Neighbors (KNN)	0.6660929
...4	Support Vector Machine (SVM)	0.7452668
...5	Decision Tree	0.7211704
...6	Random Forest	0.8106713
...7	Random Forest - Validation	0.8529412

3 Results

Table 8 below shows the complete table of models and their corresponding accuracies. In this table, the KNN model has the worst accuracy with a value of 0.6764, while the random forest model has the best accuracy with a value of 0.8192. The logistic regression, LDA, SVM and decision tree models have similar accuracies in the range of 0.7228-0.7452. With the random forest model having the highest accuracy, this model was deemed the best model and was applied to the validation dataset. The accuracy of that model was 0.8529 and thus, 0.8529 is the final accuracy for the project.

Table 8: All Models and Corresponding Accuracies

	Model	Accuracy
Accuracy	Logistic Regression	0.7228916
...2	Linear Discriminant Analysis (LDA)	0.7228916
...3	K-Nearest Neighbors (KNN)	0.6660929
...4	Support Vector Machine (SVM)	0.7452668
...5	Decision Tree	0.7211704
...6	Random Forest	0.8106713
...7	Random Forest - Validation	0.8529412

4 Conclusion

The wine quality dataset was partitioned, trained, tested, and validated with an assortment of machine learning techniques. With this approach, it was found that the optimal model for the dataset was the random forest model and this provided a final accuracy of 0.8192. The wine type was removed from this dataset due to it being an identifier rather than a useful predictor for the dependent variable, the quality of the wine. Future projects can split the dataset into the two different types of wine and perform the machine learning methods separately on those two new datasets. Additionally, instead of predicting the quality of wine, the analysis can be focused on predicting the type of wine.