Assignment 4: Wine Revisited

Nikhil Agarwal
PREDICT 454 | Sec.55
Northwestern University

Table of Contents

Introduction

The purpose of this analysis is to perform a data quality check, conduct an exploratory data analysis, and build three specific model types: random forest, support vector machine (SVM), and neural network. Due to the limited number of observations, all models were built and tested against in-sample data. Therefore, the models constructed are not necessarily models that would be implemented in a production environment. Both the dataset and the corresponding data dictionary were provided. The dataset consisted of 177 observations across 13 predictors. The goal of the models is to determine the type of wine (multi-classification) based on the different attributes. This report is divided into five key sections: Data Quality Check, Exploratory Data Analysis, Model Development, Model Comparison, and Next Steps.

Data Quality Check

Data Overview

The dataset consisted of 177 observations spanning 13 predictors and the response was a categorical variable with three different classifications: W1 (denoting wine type 1), W2 (denoting wine type 2), and W3 (denoting wine type 3). Upon importing the data, the variables *magnesium* and *proline* were converted to numerical values. Furthermore, the response variable was renamed to *WineType* and coded as a categorical (i.e., factor) variable with three levels (W1, W2, and W3). Table 1 describes the different variables in the dataset.

Table 1

Attribute	Type	Category
WineType	categorical	response
Alcohol	numerical	predictor
MalicAcid	numerical	predictor
Ash	numerical	predictor
AlcalinityOfAsh	numerical	predictor
Magnesium	numerical	predictor
TotalPhenols	numerical	predictor
Flavanoids	numerical	predictor
NonflavanoidPhenols	numerical	predictor
Proanthocyanins	numerical	predictor
ColorIntensity	numerical	predictor
Hue	numerical	predictor
OD280_OD315_of_DilutedWines	numerical	predictor
Proline	numerical	predictor

Table 2 provides a breakdown of the observation count by the different factors in the variable *WineType*.

Table 2

Response Type	Observation Count	Observation Percentage
W1	58	32.8%
W2	71	40.1%
W3	48	27.1%

Table 3 provides the descriptive statistics on the predictor variables.

Table 3

Variable	Q 0.01	Q 0.05	Q 0.25	Q 0.50	Q 0.75	Q 0.95	Q 0.99	Mean	Variance	Min	Max
Alcohol	11.4404	11.658	12.36	13.05	13.67	14.22	14.4764	12.994	0.654	11.03	14.83
MalicAcid	0.8976	1.058	1.6	1.87	3.1	4.464	5.5436	2.34	1.253	0.74	5.8
Ash	1.7	1.92	2.21	2.36	2.56	2.742	2.992	2.366	0.076	1.36	3.23
AlcalinityOfAsh	11.352	14.76	17.2	19.5	21.5	25	28.5	19.517	11.129	10.6	30
Magnesium	78	80.8	88	98	107	123.2	141.88	99.588	200.903	70	162
TotalPhenols	1.138	1.38	1.74	2.35	2.8	3.276	3.5992	2.292	0.392	0.98	3.88
Flavanoids	0.47	0.544	1.2	2.13	2.86	3.5	3.7932	2.023	0.997	0.34	5.08
NonflavanoidPhenols	0.14	0.19	0.27	0.34	0.44	0.6	0.63	0.362	0.016	0.13	0.66
Proanthocyanins	0.42	0.73	1.25	1.55	1.95	2.712	3.0368	1.587	0.327	0.41	3.58
ColorIntensity	1.8616	2.112	3.21	4.68	6.2	9.604	11.028	5.055	5.403	1.28	13
Hue	0.5476	0.57	0.78	0.96	1.12	1.286	1.4272	0.957	0.053	0.48	1.71
OD280_OD315_of_DilutedWines	1.29	1.46	1.93	2.78	3.17	3.572	3.7364	2.604	0.497	1.27	4
Proline	306.72	354.4	500	672	985	1298	1522.68	745.096	99151.962	278	1680

Note the abnormally high variances for the variables *Magnesium* and *Proline*. Furthermore, if the mean and median (see column Q0.50) values are equal, that would suggest a normal distribution. In other words, the larger the difference between the mean and median, the stronger the evidence for skewness. The variable *MalicAcid*, for instance, has a difference of 0.47 units between the mean and median. Although this difference is not very large, there is some slight skewness. Figure 1 shows the boxplot and histogram for this variable.

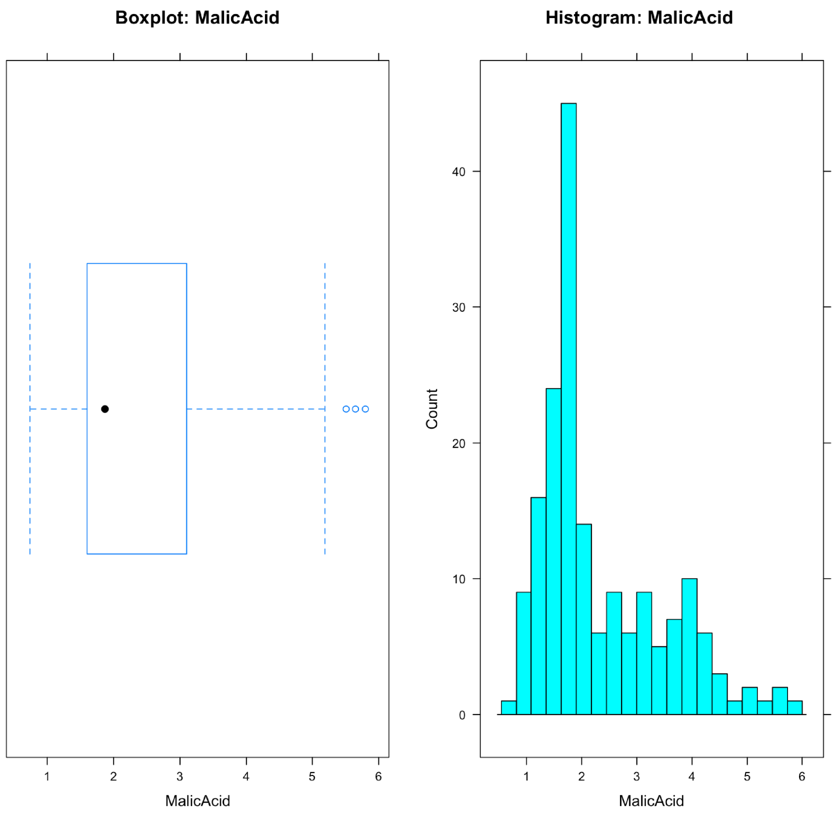


Figure 1

Note how the variable, *MalicAcid*, is slightly skewed to the right. In contrast, Figure 2 illustrates the density plot (which is similar to a histogram) of the variable *Ash*.

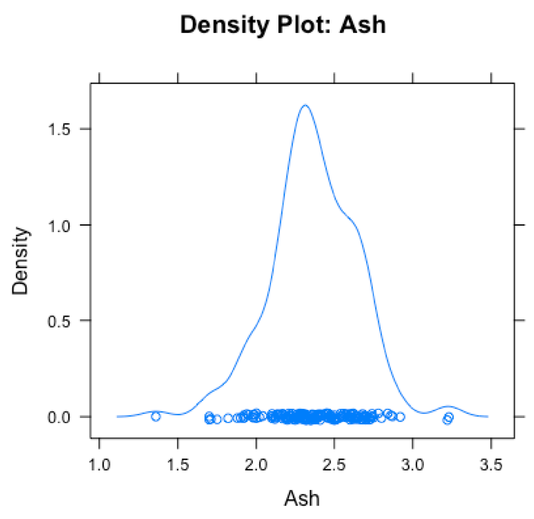


Figure 2

This plot shows that the variable is close to normal distribution. However, looking closely at the row of points, it is clear to see that a few points are the extreme ends. This could suggest the presence of outliers. Visual checks for outliers (such as boxplots and histograms) are discussed in the Exploratory Data Analysis section. Multiple plots were constructed, but not all plots are documented in this report for relevancy and brevity.

Outlier Detection

Detecting outliers can be a subjective task and can also be done numerically or visually.

Table 4

Variable	Q 0.01	Q 0.99	Min	Max	Range
Alcohol	11.4404	14.4764	11.03	14.83	3.8
MalicAcid	0.8976	5.5436	0.74	5.8	5.06
Ash	1.7	2.992	1.36	3.23	1.87
AlcalinityOfAsh	11.352	28.5	10.6	30	19.4
Magnesium	78	141.88	70	162	92
TotalPhenols	1.138	3.5992	0.98	3.88	2.9
Flavanoids	0.47	3.7932	0.34	5.08	4.74
NonflavanoidPhenols	0.14	0.63	0.13	0.66	0.53
Proanthocyanins	0.42	3.0368	0.41	3.58	3.17
ColorIntensity	1.8616	11.028	1.28	13	11.72
Hue	0.5476	1.4272	0.48	1.71	1.23
OD280_OD315_of_DilutedWines	1.29	3.7364	1.27	4	2.73
Proline	306.72	1522.68	278	1680	1402

Table 4 is very similar to Table 3, but it also has the range for each variable. As mentioned earlier, the variable *Proline* has a very large range. The large range of values also contributes to the large variance which in turn can be used to help understand the skewness. A large difference, for example, between Q0.99 (the 99th percentile) and the maximum value could be indicative of an outlier. For example, the difference between the 99th percentile and the maximum value for the variable, *Magnesium*, is about 20 units. Figure 3 illustrates the boxplot for this variable.

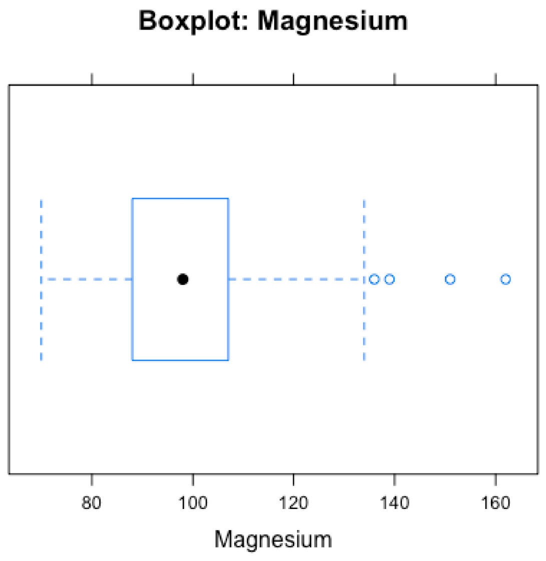


Figure 3

Note the four observations that are beyond the last whisker to the right. This is not necessarily a clear indicator of outliers, but suggestive that these points could be influential or even outliers.

Missing Data

No missing data were found in this dataset.

Exploratory Data Analysis

Quantitative Exploration

A correlation matrix with 13 predictors is not easy to present visually in this report. However, Table 5 lists the combination of predictors that were found to have an absolute value correlation of 0.6 or higher between themselves. In this instance, all of these combinations meeting the threshold of 0.6 or higher were positive correlations.

Table 5

Predictor 1	Predictor 2	Correlation	Absolute Value of Correlation
Flavanoids	TotalPhenols	0.864	0.864
OD280_OD315_of_DilutedWines	Flavanoids	0.786	0.786
OD280_OD315_of_DilutedWines	TotalPhenols	0.7	0.7
Proanthocyanins	Flavanoids	0.65	0.65
Proline	Alcohol	0.641	0.641
Proanthocyanins	TotalPhenols	0.611	0.611

Generally, the stronger a correlation between the two variables, the higher the likelihood that collinearity may be an issue. Note how the variables *Flavanoids* and *TotalPhenols* have a very strong positive correlation of 0.864. Figure 4 illustrates this relationship in a scatterplot form.

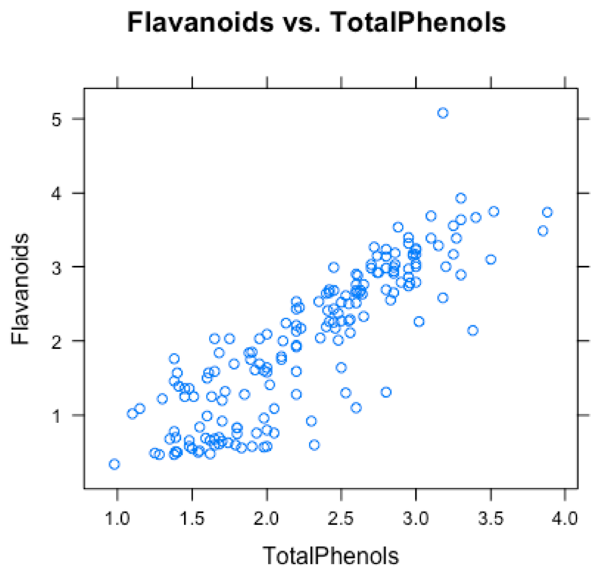
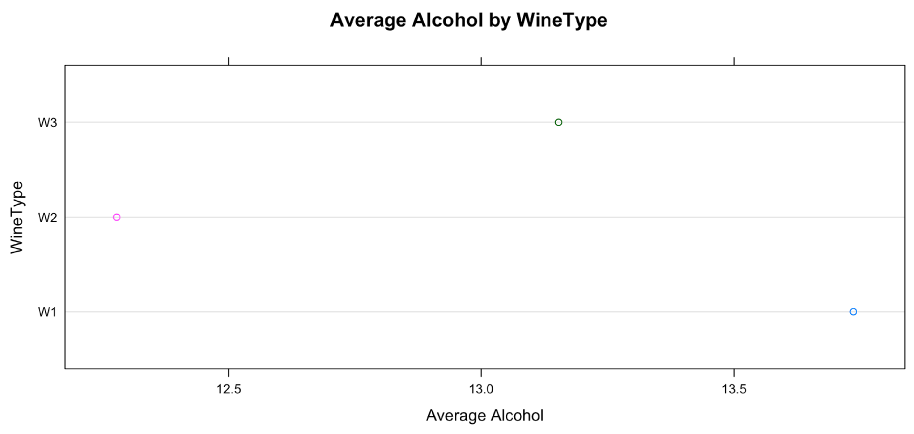


Figure 4

Note how there is distinctive linear relationship between these two variables. As the value of *TotalPhenols* increases, it can be expected that the *Flavanoids* value would also increase.

Qualitative Exploration

The average values for each predictor was calculated and then appropriate dotplots (illustrating the average values by the type of wine) were created. One interesting conclusion drawn was that wine type 2 (W2) has a low average alcohol content as well as a low average color intensity. Figure 5 illustrates the average alcohol content by type of wine versus the average color intensity by the type of wine.



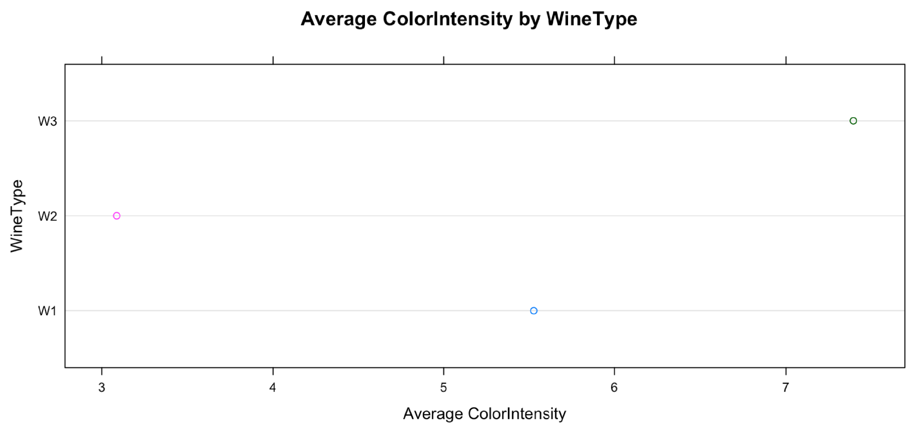


Figure 5

From the background documentation provided, it is not known if what type of wine is wine type 2 (e.g, rosé, white, red, etc.). However, generally, white wines have reduced color intensity and lower alcohol content than red wines. However, this cannot be a definitive conclusion. Regardless, this does provide some subjective insight into the relationship between the types of wine.

Outlier Detection – Visual

Since the type of wine is a categorical variable, boxplots and histograms were constructed for each predictor variable conditional to the three different types of wine. Figure 6 is an example of the predictor Magnesium. Note the presence of outliers[1] of

1 The individual circles in a boxplot are strong indicators of the presence of outliers

wine types 1 and 2 (note that *Magnesium* – as a whole – has outliers as seen in Figure 3). Furthermore, the histogram shows that there is some skewness for this predictor by wine type.

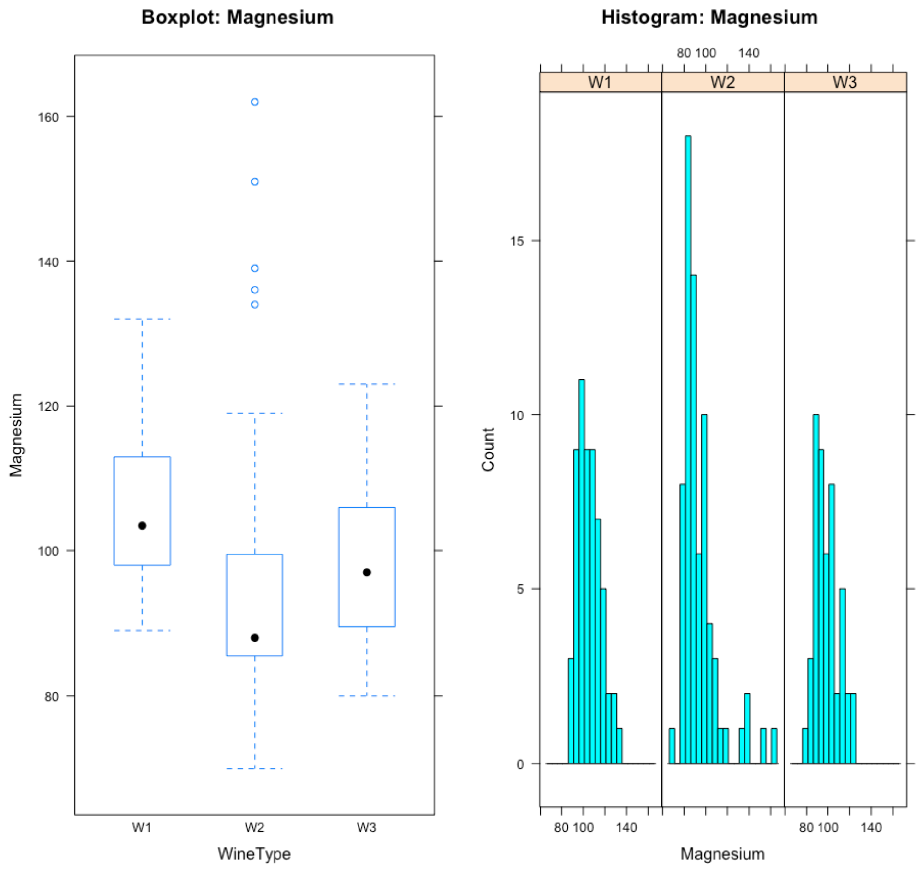


Figure 6

Typically, transformations would be undertaken to stabilize the variances and imputations would be made to manage the outliers. However, for this assignment, neither variable transformations or imputations are in scope.

Naïve Tree Model

A naïve tree model[2] was created in which the response variable, price, was fitted against the other predictors (see Figure 7). The first node is at the top and is

2 The rendering system on a Mac was not able to clearly draw the lines in this figure properly

considered the root. The value listed within each rectangle (on the first line) is the class (response). The second line (in the rectangle) lists the percentage of observations by class within that node. The third line is the percentage of observations at that node. Note how the first split is on the predictor *Proline* and if the observed value is greater than or equal to 755, then it will go to the left and proceed to the next node *Flavanoids*. The color (green for W1, orange for W3, blue for W2) denotes the prevailing class at that node.

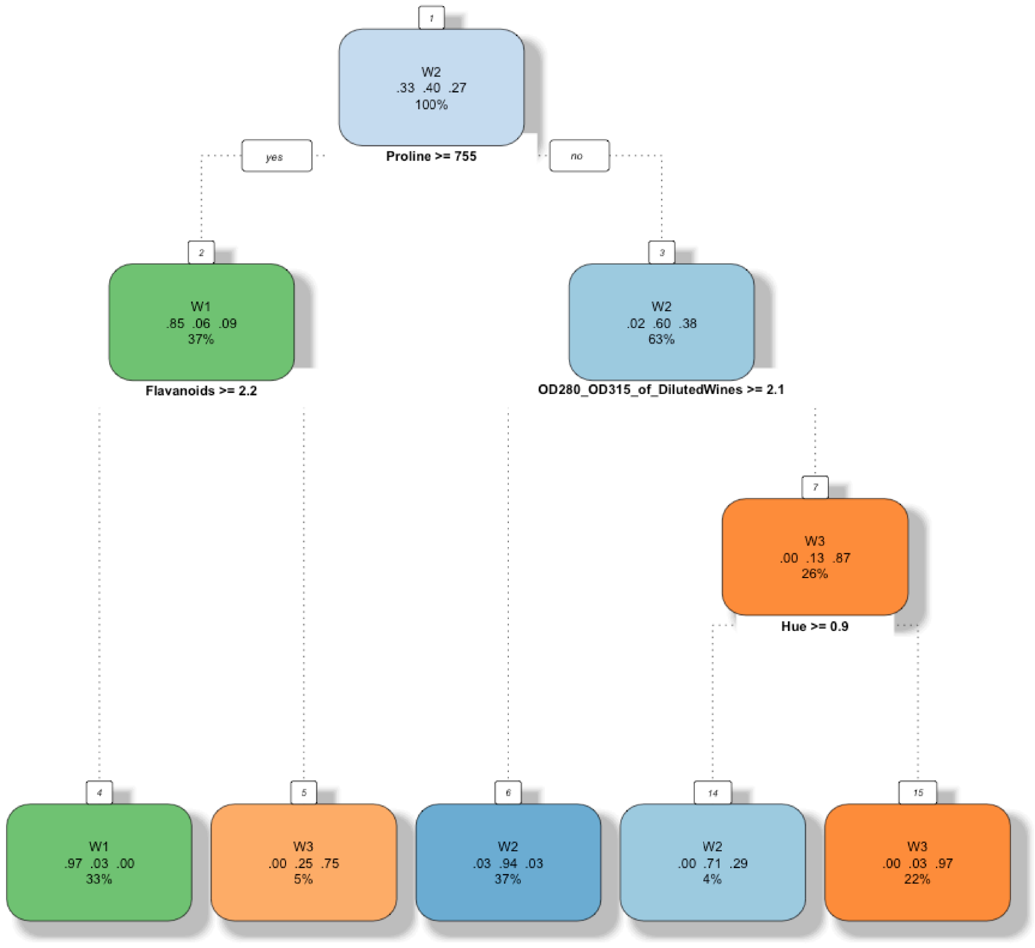


Figure 7

Note how this naïve tree model leverages only four distinct predictors: *Proline*, *Flavanoids*, *OD280_OD315_of_DilutedWines*, and *Hue*. This naïve model provides a quick glance at the important variables that were considered for this model. This tree model clearly suggests that not all 13 predictor variables may be necessary for the overall model.

Modeling Suite

Three different models were constructed: random forest, support vector machines (SVM), and neural network. Typically, the dataset would be divided into a training and test sets, however, the assignment instructions specifically state that all models should use the original dataset for training and for in-sample testing due to the limited number of observations.

Random Forest

For the random forest model, cross validation was completed using out of bag samples. Implicitly, the random forest model conducts a repeated cross validation. , random forest is a complex tree-based model that does not yield a simple equation. Hypertuning of the ‘mtry’ parameter (see Random Forest Model code in Appendix) was executed to determine the optimal number of random variables to sample at each split (see Table 11). In this instance, the parameter value of four was chosen as it yielded the highest kappa value.

Table 6 summarizes the variable importance. These values represent the mean decrease in the Gini Index. This table can be leveraged to identify key variables that are impactful to the model. Note how the importance is separated by each potential categorical value for the response variable *WineType*.

Table 6

	WineType		
Predictor	**W1**	**W2**	**W3**
Alcohol	100.000	61.279	26.010
Flavanoids	86.290	41.488	95.350
ColorIntensity	67.890	67.109	68.980
Hue	48.950	31.494	54.310
Magnesium	52.020	30.096	19.550
TotalPhenols	50.380	9.024	37.170
MalicAcid	37.600	26.790	26.130
AlcalinityOfAsh	28.530	15.027	29.050
Proanthocyanins	24.520	7.576	27.850
Ash	19.030	18.577	24.340
NonflavanoidPhenols	22.690	0.000	16.310

Note how the predictor *Alcohol* is extremely important for the class W1 (wine type 1), but is not very important for class W3. However, the variable *Flavanoids* is quite important for both classes W1 and W3. This implies that although a particular variable may not be as essential for a particular response, it may be essential for another.

500 trees were constructed for this model. Figure 8 illustrates the error based on the number of trees. Note that after about 75 trees, the error stays fairly constant. The three lines correspond to the different repeats that were done during the model development.

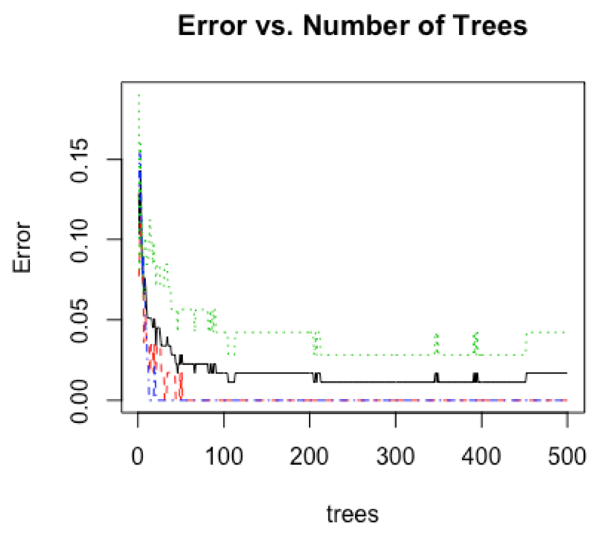


Figure 8

In Sample Performance

Table 7 summarizes the proportion of observations (using in-sample data) that are predicted to be W1, W2, or W2 (the three different types of wine). Unsurprisingly, the model has perfect fit. This perfect fit means perfect accuracy and kappa values.

Table 7

		Observed Values		
		W1	**W2**	**W3**
Predicted Values	**W1**	0.328	0	0
	W2	0	0.401	0
	W3	0	0	0.271

Conclusion

Evaluating predictive accuracy in a multi-classification problem with random forest is fairly simple with a confusion matrix. Unfortunately, random forest models do not have a simple tree diagram that can be leveraged to visually understand the decisions the multitude of trees are making.

Support Vector Machine

In general, SVMs are "a kernel-based classification approach where the kernels are represented in terms of a…subset of the training examples" (Zumel & Mount, 2014). SVMs have a tendency to have better performance than logistic regression when the classes are well separated as compared to logistic regression where it may perform well with not so well separated classes (James, Witten, Hastie, & Tibshirani, 2015). For this model, the cost parameter (which is helpful in reducing training errors) was hypertuned. Table 12 documents the results of the hypertuning. In this iteration, a cost of 0.01 was used. Furthermore, this iteration also used a linear kernel.

The challenge with SVMs are that they are designed to perform well with binary classifiers. In order to leverage SVMs for multi-classification problems, SVMs can be modified to use several approaches such as One-vs-All (OVA). Inherently, this presents several challenges as the interpretability is reduced and becomes more complex.

In Sample Performance

Table 8 summarizes the proportion of observations (using in-sample data) that are predicted to be W1, W2, or W2 (the three different types of wine). Somewhat surprisingly, the model is close to perfect, but not as perfect as was seen in the confusion matrix for the random forest model. The accuracy and kappa for this model are 0.9944 and 0.9914 respectively.

Table 8

		Observed Values		
		W1	W2	W3
Predicted Values	W1	0.328	0	0
	W2	0	0.395	0
	W3	0	0.006	0.271

Conclusion

SVMs could be used for multi-classification problems, however, their reduced interpretability and increased complexity does not necessarily make this approach a desirable one. Implicitly, the SVM approach is creating a binary model for each type of wine and then combining the results. This increases the likelihood of potential errors and is reflected in the model's inability to be perfect – as compared to the random forest model.

Neural Network

Neural network models are perhaps the most convoluted models to describe. At a fundamental level, neural networks are "two-stage...classification model[s]" (Hastie, Tibshirani, & Friedman, 2009). For this iteration, the package nnet was used in the package caret. There are two parameters for this approach: size and decay. Size is essentially the number of hidden layers whereas decay is the regularization parameter to reduce overfitting. In this instance, the size was kept constant at 5 and decay was set to 0.1. Furthermore, the predictors were preprocessed (prior to training the model) to be centered and scaled. This essentially made each predictor have a mean of 0 and a standard deviation of 1. Furthermore, the model was cross-validated using a 5-fold repeat three times methodology (repeated cross-validation).

In Sample Performance

Table 9 summarizes the proportion of observations (using in-sample data) that are predicted to be W1, W2, or W2 (the three different types of wine). Unsurprisingly, the model has perfect fit. This perfect fit means perfect accuracy and kappa values.

Table 9

		Observed Values		
		W1	**W2**	**W3**
Predicted Values	**W1**	0.328	0	0
	W2	0	0.401	0
	W3	0	0	0.271

A personal comment on Neural Network

The methodology that I used in this iteration does not have the capability of producing a visual plot. However, the visual plot created by the package ‘neuralnet’ is quite cumbersome to read and interpret. For that reason, the plot was excluded from this report.

Conclusion

The neural network model is by far extremely complex to interpret and explain. This is a clear example of a “black-box” model that works well (as can be seen from the confusion matrix) but is very difficult to explain. One of the key challenges with this model is the inability to clearly articulate the model in an easy to understand and verifiable method.

Model Performance Summary

TABLE XX summarizes the three models’ performance. Recall that this performance is based on in-sample testing (i.e., using the same data that were used to train the model). Therefore, there is a strong likelihood that overfitting may have occurred.

Table 10

Model	Accuracy	Kappa
Random forest	1.0000	1.0000
Support Vector Machine (SVM)	0.9944	0.9914
Neural Network	1.0000	1.0000

It is not clear which model should be chosen. All three models have significant complexity and reduced interpretability. Furthermore, all three models are not suitable for operational use since they were never tested against out-of-sample data (data that were not used in any way for training the models).

Conclusion & Next Steps

First and foremost, the models constructed are nowhere near ready for operational use. Since they were 'tested' using in-sample data, there is no way to really know how well the models may perform on unseen data. In addition, both the random forest and neural network models performed perfectly, but explaining the exact 'decisions' or the math behind their algorithms are not simple to explain. For instance, 500 trees were constructed for the random forest model, yet there is no clear way to visualize each tree.

In a multi-classification problem, metrics such as accuracy or kappa could be used to assess a model's performance. ROC curves are typically for binary classification problems. Therefore, if each type wine was converted into a binary classification, three different ROC curves could be constructed. However, summarizing all three together would make for challenging interpretations. Therefore, it is not appropriate to leverage ROC curves in a multi-classification model. In this assignment, all three models in the modeling suite can be classified (no pun intended) as "black-box models". Technically, neural network models can have visualizations, but they are convoluted and extremely difficult to explain. At a fundamental level, confusion matrices can easily summarize the results of a fitted model and provide the end-user with a clear perspective. Furthermore, standard metrics such as accuracy and kappa can be used as KPI's. Perhaps the most challenging aspect of these models is the communication. In a typical business setting, the inability to clearly and easily explain models could be construed as a liability. Another challenge is that there is a very limited way to replicate the math behind each of these models. This reduces the ability to audit the algorithms.

Next steps include further examination of neural network models. The author of this assignment spent quite some time learning about neural network models, but was unable to fully comprehend their abilities. Furthermore, hypertuning the parameters will also enable greater understanding of their impact on the final model.

Works Cited

Hastie, T., Tibshirani, R., Friedman, J. (2009). The Elements of Statistical Learning. New York, New York: Springer Science & Business Media.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2015). An Introduction to Statistical Learning with Applications in R. New York, New York: Springer Science & Business Media.

Zumel, N., & Mount, J. (2014). Practical Data Science with R. Shelter Island, New York: Manning Publications Co.

Appendix A – R Code

Table Code

Table 2

```
wine %>%
  group_by(WineType) %>%
  summarize(
    TotalObservations = n(),
    ObservationPCT = n()/nrow(wine)
  ) %>%
  ungroup()
```

Table 3, Table 4, Table 6

```
custom.summary <- function(df) {
  # create the quantile values
  q01 <- df %>%
    summarise_all(funs(quantile), probs = 0.01)
  q05 <- df %>%
    summarise_all(funs(quantile), probs = 0.05)
  q25 <- df %>%
    summarise_all(funs(quantile), probs = 0.25)
  q50 <- df %>%
    summarise_all(funs(quantile), probs = 0.50)
  q75 <- df %>%
    summarise_all(funs(quantile), probs = 0.75)
  q95 <- df %>%
    summarise_all(funs(quantile), probs = 0.95)
  q99 <- df %>%
    summarise_all(funs(quantile), probs = 0.99)
  missingPCT <- df %>%
    summarize_all(funs(sum(is.na(.)) / length(.)))

  # create the mean, variance, min, max
  avg <- round(df %>%
                 dplyr::summarize_all(mean),3)
  variance <- round(df %>%
                      dplyr::summarize_all(var),3)
  minimum <- df %>%
    dplyr::summarize_all(min)
  maximum <- df %>%
    dplyr::summarize_all(max)


  # flatten & widen the dataframe
  q01 %<>%
    reshape2::melt() %>%
    dplyr::rename('Q 0.01' = value)
  q05 %<>%
    reshape2::melt() %>%
    dplyr::rename('Q 0.05' = value)
  q25 %<>%
    reshape2::melt() %>%
    dplyr::rename('Q 0.25' = value)
  q50 %<>%
    reshape2::melt() %>%
    dplyr::rename('Q 0.50' = value)
  q75 %<>%
    reshape2::melt() %>%
    dplyr::rename('Q 0.75' = value)
  q95 %<>%
    reshape2::melt() %>%
    dplyr::rename('Q 0.95' = value)
  q99 %<>%
    reshape2::melt() %>%
    dplyr::rename('Q 0.99' = value)
  avg %<>%
    reshape2::melt() %>%
    dplyr::rename('Mean' = value)
  variance %<>%
    reshape2::melt() %>%
    dplyr::rename('Variance' = value)
  minimum %<>%
    reshape2::melt() %>%
    dplyr::rename('Min' = value)
  maximum %<>%
    reshape2::melt() %>%
    dplyr::rename('Max' = value)
  missingPCT %<>%
    reshape2::melt() %>%
    dplyr::rename('Missing PCT' = value)

  # combine the flat files
  tblGrande <- inner_join(q01, q05, by='variable')
  tblGrande <- inner_join(tblGrande, q25, by = 'variable')
  tblGrande <- inner_join(tblGrande, q50, by = 'variable')
  tblGrande <- inner_join(tblGrande, q75, by = 'variable')
  tblGrande <- inner_join(tblGrande, q95, by = 'variable')
  tblGrande <- inner_join(tblGrande, q99, by = 'variable')
  tblGrande <- inner_join(tblGrande, avg, by = 'variable')
  tblGrande <- inner_join(tblGrande, variance, by = 'variable')
  tblGrande <- inner_join(tblGrande, minimum, by = 'variable')
  tblGrande <- inner_join(tblGrande, maximum, by = 'variable')
  tblGrande <- inner_join(tblGrande, missingPCT, by = 'variable')

  # return the final table
  return(tblGrande)
}

CustomSummary <- custom.summary(wine[2:14])
CustomSummary %<>%
   mutate(
      range = Max - Min
)
```

Table 5

```
library(corrr)
x <- correlate(wine[,2:14])
x.m <- melt(x)
x.m %<>%
  mutate(
    corr = round(value, 3),
    absCorr = abs(corr)
  ) %>%
  # filter(absCorr >= 0.7) %>%
  arrange(desc(absCorr))
```

Figure Code

Figure 1 Code

```
duo2 <- function(y) {
  a <- bwplot(~ wine[[y]], data=wine, xlab=y, main=paste('Boxplot:',y))
  b <- histogram(~wine[[y]], data=wine, xlab=y, type='count', nint=20, layout
= c(1,1), main=paste('Histogram:',y))
  grid.arrange(a,b,ncol=2)

}
duo2('MalicAcid')
```

Figure 2 Code

```
dp <- function(y) {
  densityplot(~wine[[y]] | WineType, data = wine, as.table=T, xlab = y,
main=paste('Density Plot:',y))
}
for(i in colnames(wine)) {
  print(dp(i))
}
```

Figure 3 Code

```
bwp <- function(x) {
  bwplot(~wine[[x]], data = wine, xlab = x, main = paste('Boxplot:',x,sep='
'))
}
bwp('Magneisum')
```

Figure 4 Code

```
xyp <- function(x,y) {
  xyplot(wine[[y]] ~ wine[[x]] | WineType, data=wine, as.table = T,
type=c('g','p'), xlab=x, ylab=y, main=paste(y,'vs',x,'by WineType'),
layout=c(1,3))
}
xyp('TotalPhenols','Flavanoids')
```

Figure 5 Code

```
a1 <- dotplot(WineType ~ Alcohol, data = AvgVars, group = WineType, xlab =
'Average Alcohol', ylab = 'WineType', main = 'Average Alcohol by WineType')
a2 <- dotplot(WineType ~ ColorIntensity, data = AvgVars, group = WineType,
xlab = 'Average ColorIntensity', ylab = 'WineType', main = 'Average Hue by
WineType')

grid.arrange(a1, a2)
```

Figure 6 Code

```
duo <- function(x,y) {
  a <- bwplot(wine[[y]] ~ wine[[x]], data=wine, xlab=x, ylab=y,
main=paste('Boxplot:',y))
  b <- histogram(~wine[[y]] | wine[[x]], data=wine, xlab=y, type='count',
nint=20, layout = c(3,1), main=paste('Histogram:',y))
  grid.arrange(a,b,ncol=2)

}
duo('WineType', 'Magnesium')
```

Model Code

The code in this section was used to generate the appropriate tables and figures for each model.

Libraries Used in Script

```
library(tidyverse)      # easier data frame manipulation
library(lattice)        # for the graphs/charts
library(moments)        # for skewness & kurtosis
library(gridExtra)      # for multiple types of plots on one 'page'
library(magrittr)       # for easier storing of data frame changes
library(rpart)          # for tree making
library(rpart.plot)
library(rattle)
library(reshape2)
library(caret)
library(gridExtra)

library(doMC)
registerDoMC(4)
```

Naïve Tree Model

```
tree <- rpart(WineType~., data = wine)
fancyRpartPlot(tree, sub = '', cex = 0.7)
```

Random Forest Model

```
set.seed(33)
rforest_grid <- expand.grid(
  mtry=c(1:12)
)

fitControl_RF <- trainControl(
  method = 'oob',
  classProbs = F,
  allowParallel = T
)
set.seed(2017)
rFit1 <- train(x = wine[,2:12], y=wine[,1], data = wine, method = "rf",
trControl = fitControl_RF, verbose = F, tuneGrid = rforest_grid, num.trees =
500, metric='Kappa', importance = T)

rFit1
plot(rFit1$finalModel, main = 'Error vs. Number of Trees')
plot(rFit1$results$mtry, rFit1$results$Kappa, lty = 2, type = 'o', xlab =
'mtry value', ylab = 'Kappa',main='Kappa Results for mtry Parameter')
abline(col='red',v=4, h = max(rFit1$results$Kappa))
varImp(rFit1)
```

```
wineRF.fitted <- predict(rFit1, newdata = wine)
confusionMatrix(wineRF.fitted, wine$WineType)
```

Support Vector Machines (SVM) Model

```
svmControl <- trainControl(
  method = "repeatedcv",
  number = 5,
  repeats = 3,
  allowParallel = T
)

grid <- expand.grid(cost = c(0.001, 0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.25,
1.5, 1.75, 2,5))
set.seed(2017)

svm_Lin2 <- train(WineType ~., data = wine, method = "svmLinear2",
trControl=svmControl,tuneLength = 10, tuneGrid = grid, metric = "Kappa")

svm_Lin2

wineSVM.fitted <- predict(svm_Lin2, newdata = wine)
confusionMatrix(wineSVM.fitted, wine$WineType)
```

Neural Network Model

```
nnControl <- trainControl(
  method = "repeatedcv",
  number = 5,
  repeats = 3,
  allowParallel = T
)
set.seed(2018)
nn1 <- train(WineType ~ ., data = wine, method ='nnet', trControl = nnControl,
tuneGrid=expand.grid(size=c(5), decay=c(0.1)), preProcess = c('center',
'scale'))
nn1
nn1$finalModel
nn1$finalModel$wts
nn1.fitted <- predict(nn1, newdata = wine)
confusionMatrix(nn1.fitted, wine$WineType)
nn1$finalModel$wts
```

Appendix B – Figures and Tables

Random Forest Hypertuning

Table 11

mtry	Kappa
1	0.966
2	0.974
3	0.974
4	0.983
5	0.974
6	0.974
7	0.966
8	0.966
9	0.949
10	0.940
11	0.940
12	0.914

Support Vector Machine Hypertuning

Table 12

cost	Kappa
0.001	0.0173
0.01	0.9803
0.05	0.9715
0.1	0.9657
0.25	0.9485
0.5	0.9569
0.75	0.9599
1	0.9627
1.25	0.9600
1.5	0.9600
1.75	0.9600
2	0.9600
5	0.9600