**Assignment #4**

James R. Herbick

## Introduction

### Overview/ Purpose

This assignment continues with work done in assignment #1 which involved performing exploratory data analysis (EDA) on a wine dataset. Here, I complete the modeling process beyond the EDA phase and fit several models appropriate for a multi-class classification problem. The wine dataset contains characteristics of wines from a single region in Italy, and poses a multi-class classification problem. The characteristics of the wines can be used to classify a particular wine into one of three classes. The data includes wines categorized by class, therefore, this data lends itself to supervised modeling approaches.

### Modeling Problem Statement

The wine dataset lends itself to a multi-class classification problem. In this assignment, I take the modeling process to completion looking to fit multiple types of models appropriate for a multi-class classification problem. This is a pure prediction problem, however, model-based EDA most likely will provide insight into those variables with the most predictive power. This exercise will allow me to explore the nuances of performing the entire model-building process in a multi-class classification setting. Knowledge and learnings from our group project should provide guidance in this assignment as well. This assignment will also allow me to explore the nuances of the parameters for advanced modeling methods utilizing a small dataset.

## Data

### General Description

The wine dataset consists of various characteristics of individual wines from a particular region in Italy. These wine characteristics comprise the independent variables for the dataset. There are 178 observations, 13 independent variables, and 1 dependent variable. The dependent variable, class, represents a category or classification of wine. The class variable has three valid levels: 1, 2, and 3.

Figure 1 is a table of all the variables in the dataset with summary information about each.

**FIGURE 1: DATA ELEMENTS**

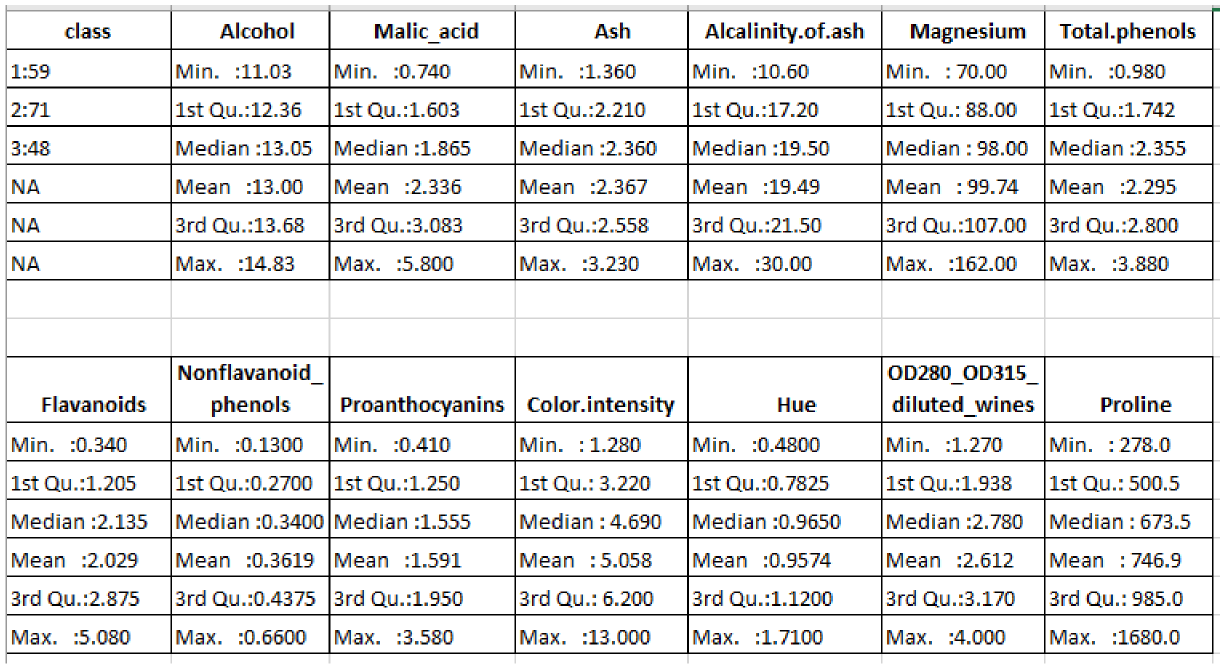
| | Variable Name | Data Type | Description |
|---|---|---|---|
| 1 | class | INTEGER | Dependent variable. Valid classes are: 1, 2, and 3. |
| 2 | Alcohol | NUMBER | Percentage of alcohol content for the wine. |
| 3 | Malic acid | NUMBER | Acid level which varies by grape varietal. |
| 4 | Ash | NUMBER | Ash content. An important indicator of quality in wines. |
| 5 | Alcalinity of ash | NUMBER | The total alkaline capacity in wine. |
| 6 | Magnesium | INTEGER | Magnesium content of the wine. Associated with red wines. |
| 7 | Total phenols | NUMBER | A large group of chemical compounds that affect the taste, color, and mouthfeel of wine. |
| 8 | Flavanoids | NUMBER | In red wine, up to 90% of the wine's phenolic content falls under the classification of flavonoids. |
| 9 | Nonflavanoid phenols | NUMBER | Compounds affecting the taste, color, and mouthfeel of wine from such items as grape skins and oak barrel ageing. |
| 10 | Proanthocyanins | NUMBER | The principal polyphenols in red wine that are under research to assess risk of coronary heart disease and lower overall mortality. |
| 11 | Color intensity | NUMBER | Deepness of color in the wine. |
| 12 | Hue | NUMBER | Hints of color. Can indicate age or spoilage in a wine. |
| 13 | OD280/OD315 of diluted wines | NUMBER | Unable to find specifics on this variable. |
| 14 | Proline | INTEGER | An amino acid. |

## Data Quality Check

The first step in examining the data is to investigate missing values and outliers in the data. Here, I look for any elements in the data that could negatively affect the modeling process. Figure 2 shows data summaries of all the variables. There are no missing values for any of the variables in the wine dataset. However, it does appear that many of the independent variables are recorded on different numerical scales. For example, the Magnesium variable has much larger numerical values than many of the other variables. Therefore, standardizing the data may prove to be necessary for certain types of modeling techniques. Please see the Data Transformations section for a brief discussion on standardizing the data.

Figure 2 also shows that there are independent variables that have the possibility of outliers. For example, the Alcalinity.of.Ash variable has a value of 21.50 for the 3rd quartile, but a maximum value of 30. In addition, the Magnesium variable has a value at the 3rd quartile of 107.00, but a maximum of 162.00. These large gaps between the 3rd quartile and the maximum values might indicate outliers to the top end of these variables. The same is apparent for the Proline variable. Finally, notice in Figure 2 that the distribution of observations for the class variable is shown.  This is because the class variable is setup as a factor.

**Figure 2: Wine Dataset – data summary**



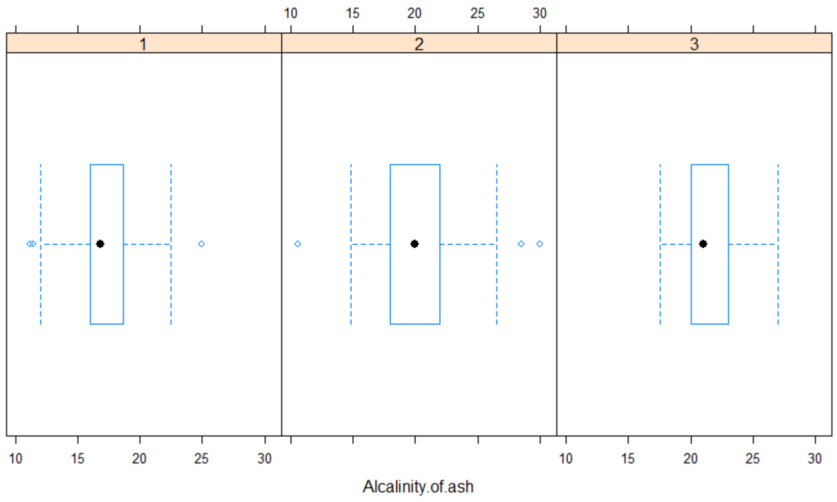
| class | Alcohol | Malic_acid | Ash | Alcalinity.of.ash | Magnesium | Total.phenols |
|---|---|---|---|---|---|---|
| 1:59 | Min.   :11.03 | Min.   :0.740 | Min.   :1.360 | Min.   :10.60 | Min.   : 70.00 | Min.   :0.980 |
| 2:71 | 1st Qu.:12.36 | 1st Qu.:1.603 | 1st Qu.:2.210 | 1st Qu.:17.20 | 1st Qu.: 88.00 | 1st Qu.:1.742 |
| 3:48 | Median :13.05 | Median :1.865 | Median :2.360 | Median :19.50 | Median : 98.00 | Median :2.355 |
| NA | Mean   :13.00 | Mean   :2.336 | Mean   :2.367 | Mean   :19.49 | Mean   : 99.74 | Mean   :2.295 |
| NA | 3rd Qu.:13.68 | 3rd Qu.:3.083 | 3rd Qu.:2.558 | 3rd Qu.:21.50 | 3rd Qu.:107.00 | 3rd Qu.:2.800 |
| NA | Max.   :14.83 | Max.   :5.800 | Max.   :3.230 | Max.   :30.00 | Max.   :162.00 | Max.   :3.880 |

| Flavanoids | Nonflavanoid_ phenols | Proanthocyanins | Color.intensity | Hue | OD280_OD315_ diluted_wines | Proline |
|---|---|---|---|---|---|---|
| Min.   :0.340 | Min.   :0.1300 | Min.   :0.410 | Min.   : 1.280 | Min.   :0.4800 | Min.   :1.270 | Min.   : 278.0 |
| 1st Qu.:1.205 | 1st Qu.:0.2700 | 1st Qu.:1.250 | 1st Qu.: 3.220 | 1st Qu.:0.7825 | 1st Qu.:1.938 | 1st Qu.: 500.5 |
| Median :2.135 | Median :0.3400 | Median :1.555 | Median : 4.690 | Median :0.9650 | Median :2.780 | Median : 673.5 |
| Mean   :2.029 | Mean   :0.3619 | Mean   :1.591 | Mean   : 5.058 | Mean   :0.9574 | Mean   :2.612 | Mean   : 746.9 |
| 3rd Qu.:2.875 | 3rd Qu.:0.4375 | 3rd Qu.:1.950 | 3rd Qu.: 6.200 | 3rd Qu.:1.1200 | 3rd Qu.:3.170 | 3rd Qu.: 985.0 |
| Max.   :5.080 | Max.   :0.6600 | Max.   :3.580 | Max.   :13.000 | Max.   :1.7100 | Max.   :4.000 | Max.   :1680.0 |

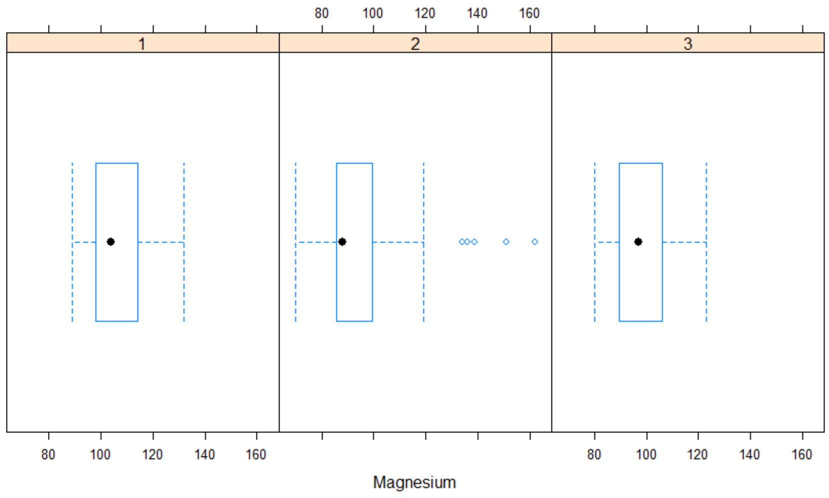
As displayed in Figure 2, there are a few variables that seem likely to have outliers.  I performed a deeper dive on these variables and produced the boxplots by class shown in Figure 3.  The circles beyond the whiskers are considered outliers and should at least be investigated.  Figure 3 shows that Alcalinity of Ash and Magnesium have outliers, predominantly at the high end of the variable range.  This is precisely what was anticipated in the initial data exploration.  The proline variable, however, looks relatively free of outliers.

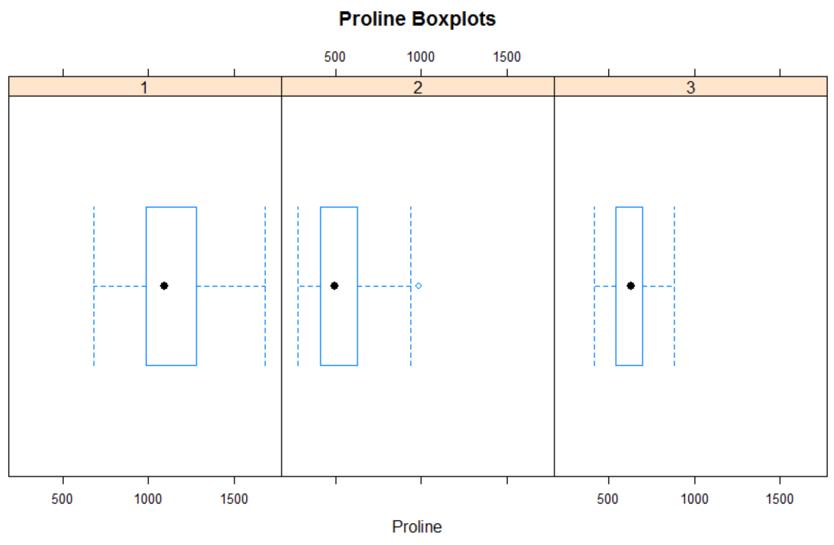
## FIGURE 3: BOXPLOTS

Alcalinity of Ash Boxplots



Magnesium Boxplots





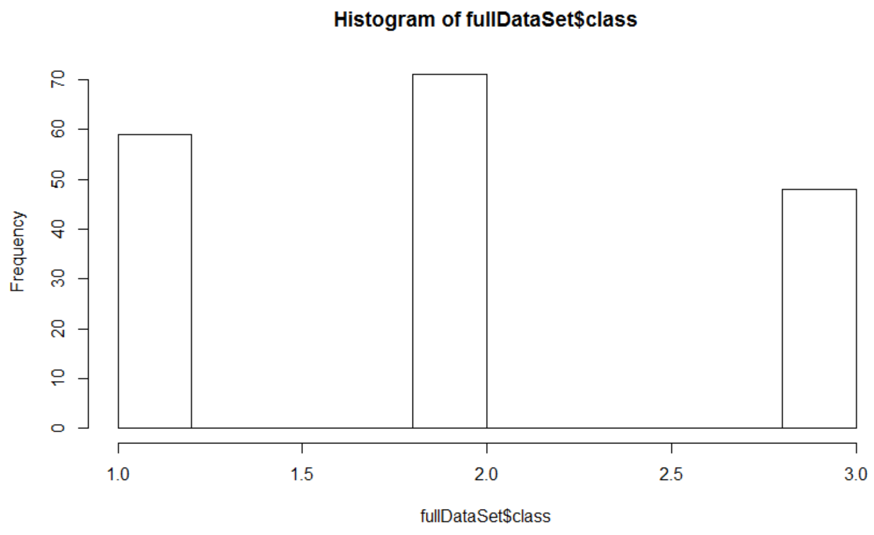
## Data Transformations

While examining the thirteen independent variables, it became clear that the variables were recorded in different scales. To have a version of the data that was standardized, I subtracted the mean and divided the result by the standard deviation for all of the independent variables. This created all variables with a mean of zero and a standard deviation of 1. If needed for certain modeling techniques, the standardized data is now available. Standardizing the data will ensure that no single variable is given more importance only due to the magnitude of its values.

## Exploratory Data Analysis (EDA)

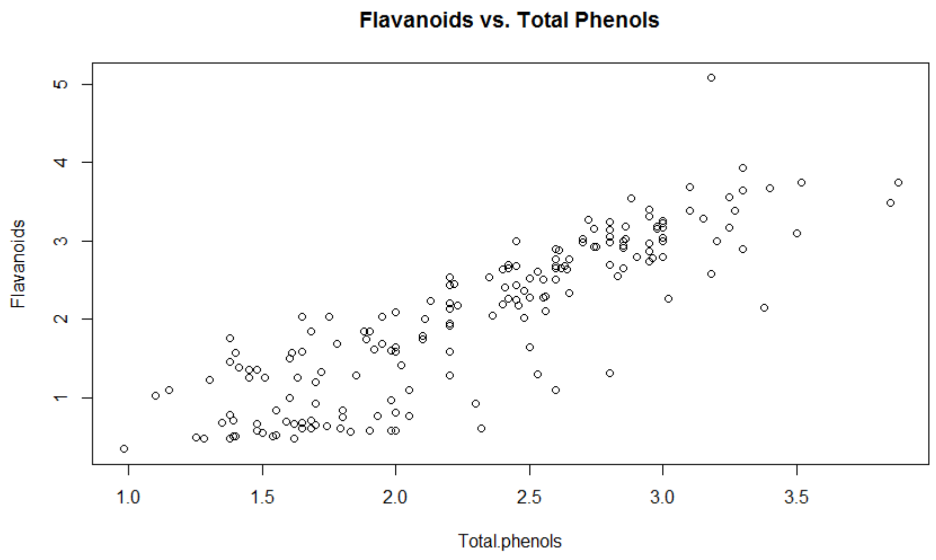
To begin the EDA, I examined the distribution of observations across the three wine classes. Figure 4 shows the results. Class 2 has a higher frequency of wines than classes 1 and 3, however, the distribution of wines across the classes is fairly stable. Therefore, these data should be good for training classification models.

**FIGURE 4: CLASS VARIABLE DISTRIBUTION**



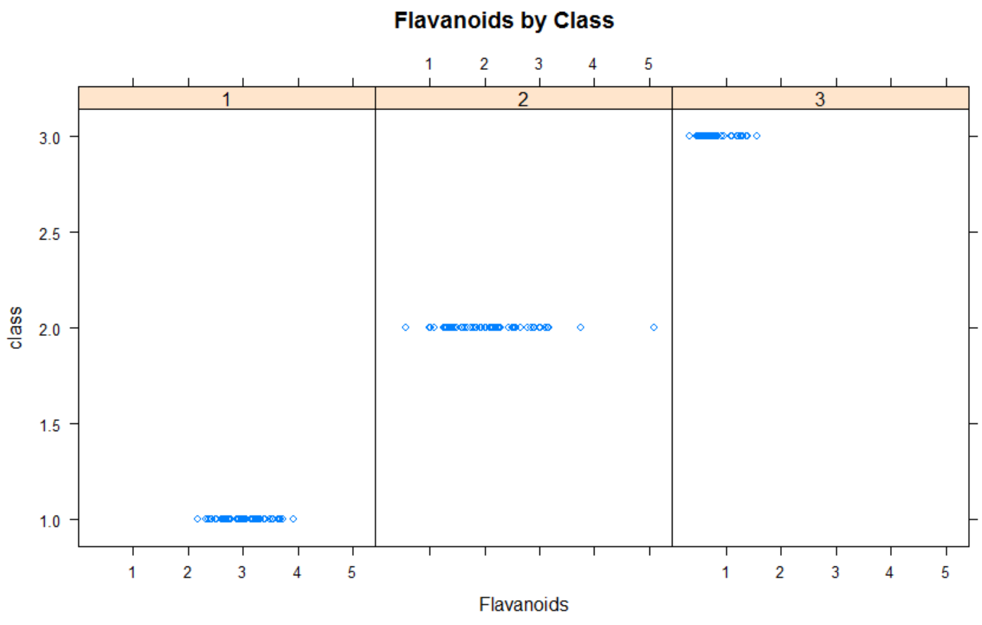
Next, I produced a correlation matrix to analyze the potential correlations among variables. From this matrix I dug deeper into some relationships between variables. For example, figure 5 shows the correlation between Flavanoids and Total Phenols. As you can see, there is a definite positive correlation. Since flavanoids are predominant in red wines, and in red wines the flavonoids make up a significant portion of the total phenols, these variables could provide good insights into when a wine should be classified as red or not. Higher levels of flavonoids could potentially suggest a red wine.

**FIGURE 5: FLAVANOIDS VS. TOTAL PHENOLS**



Taking this analysis a step further, figure 6 shows a trellis graph of the flavanoids by class. As we can see, there appear to be a higher level and concentration of flavonoids in class 1 than the other classes. Class 3 has the lowest levels of flavonoids. This suggests that potentially, class 1 represents a red wine, class 3 represents a white wine, and class 2 may be something in between, such as a rose.

**FIGURE 6: FLAVANOIDS BY CLASS**

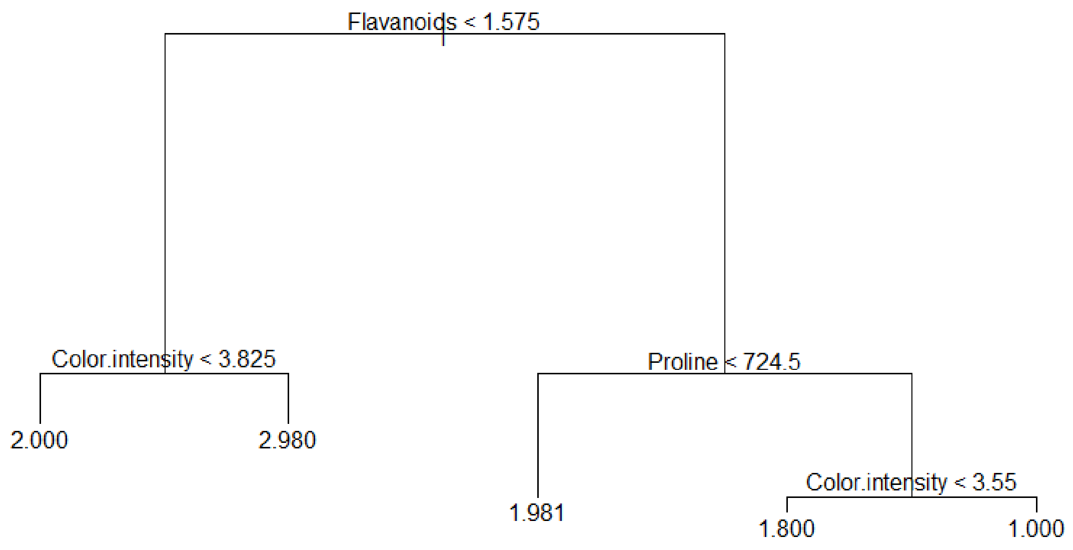


## Model Based EDA

Next, I ran a quick classification tree to gain insight into those independent variables that contribute significantly to the classification of a wine. Figure 7 shows the classification tree that resulted. As initially hypothesized, the flavonoids variable plays a key role in determining the classification of a wine. Since my research on the wine characteristics shows that a higher level of flavonoids and total phenols are predominant in red wines, it is possible that wines that fall on the right side of the classification tree are more likely to be red. Therefore, wines fall on the left side of the classification tree are more likely to be white or rose. This is consistent with the thoughts that I had earlier.

One other note, here. I did rerun the classification tree with the standardized data, but obtained the same results.

**FIGURE 7: CLASSIFICATION TREE**



## Model Suite

### Random Forest (RF)

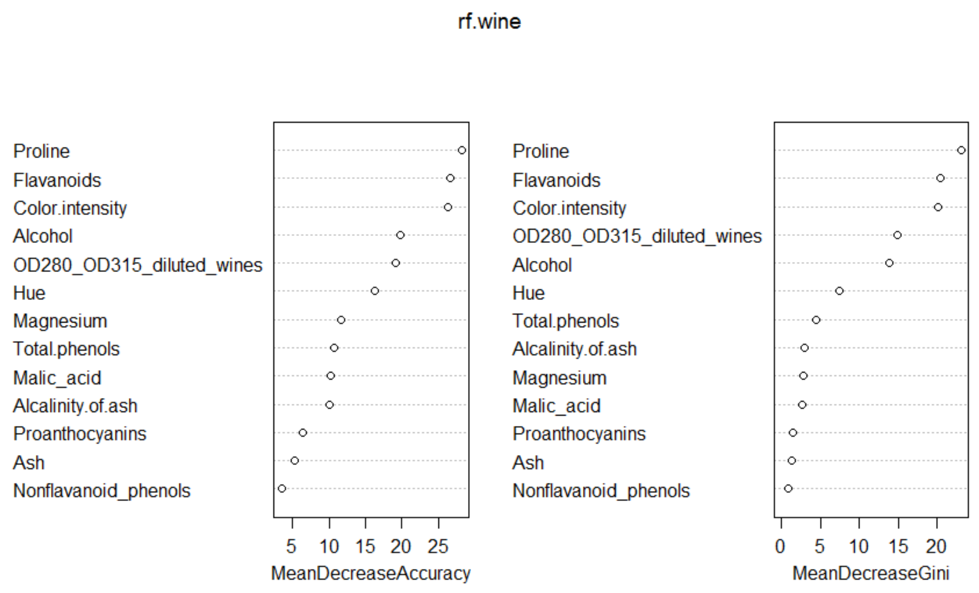
Figure 8 shows the parameters used for the random forest modeling for the wine dataset. The guideline for the number of variables chosen at each split is $\sqrt{p}$, which is ~ 4 in this case. This means that at each split, a random 4 variables will be chosen upon which to determine the split. It is also advisable to run a large number of trees. Since the dataset is small and processing should not be an issue, I ran 500 trees with the random forest model. The final parameter specifies that I want to explore the variable importance as a result of the random forest model building.

**Figure 8: Random Forest parameters**

| **Parameter** | **Value** |
|---|---|
| Number of Variables | 4 |
| Number of Trees | 500 |
| Variable Importance | TRUE |

Figure 9 shows the variable importance as a result of the random forest model building process. As discovered in the initial EDA, Proline, Flavanoids, and Color Intensity are all highly significant in determining a wine's class. These results confirm the EDA conclusions.

**Figure 9: Variable Importance**



To compare models in a multi-class classification setting, I used the confusion matrix and each model's precision (percentage correctly classified). Figure 10 shows the RF confusion matrix. The random forest's overall in-sample precision was 98% and is shown in the conclusion section along with other model performance.

**Figure 10: Random Forest Confusion Matrix**

| | **Predicted** | | |
|---|---|---|---|
| **Actual** | 1 | 2 | 3 |
| 1 | 98% | 2% | 0% |
| 2 | 1% | 96% | 3% |
| 3 | 0% | 0% | 100% |

## Support Vector Machine (SVM)

For the support vector machine (SVM), there are a few parameters that can be adjusted prior to running the model. Figure 11 shows these parameters and the values I used for this analysis. Adjusting the kernel parameter allows an increase to the feature space and apply non-linear decision boundaries between the classes. The radial kernel has very local behavior, meaning only nearby observations have an effect on the class label of another observation. The cost parameter can be thought of as a budget of how many observations can violate the margin. Therefore, in this case no more than 10 observations can violate be on the wrong side of the hyperplane. As the cost increases beyond zero, the margins are wider and more violations are accepted. Finally, the gamma parameter is used with all kernels other than the linear kernel. Larger values of gamma increase the non-linearity of the fit.

**Figure 11: SVM parameters**

| **Parameter** | **Value** |
|---|---|
| Kernel | radial |
| Cost | 10 |
| Gamma | 1 |

**Figure 12: SVM Classifications**

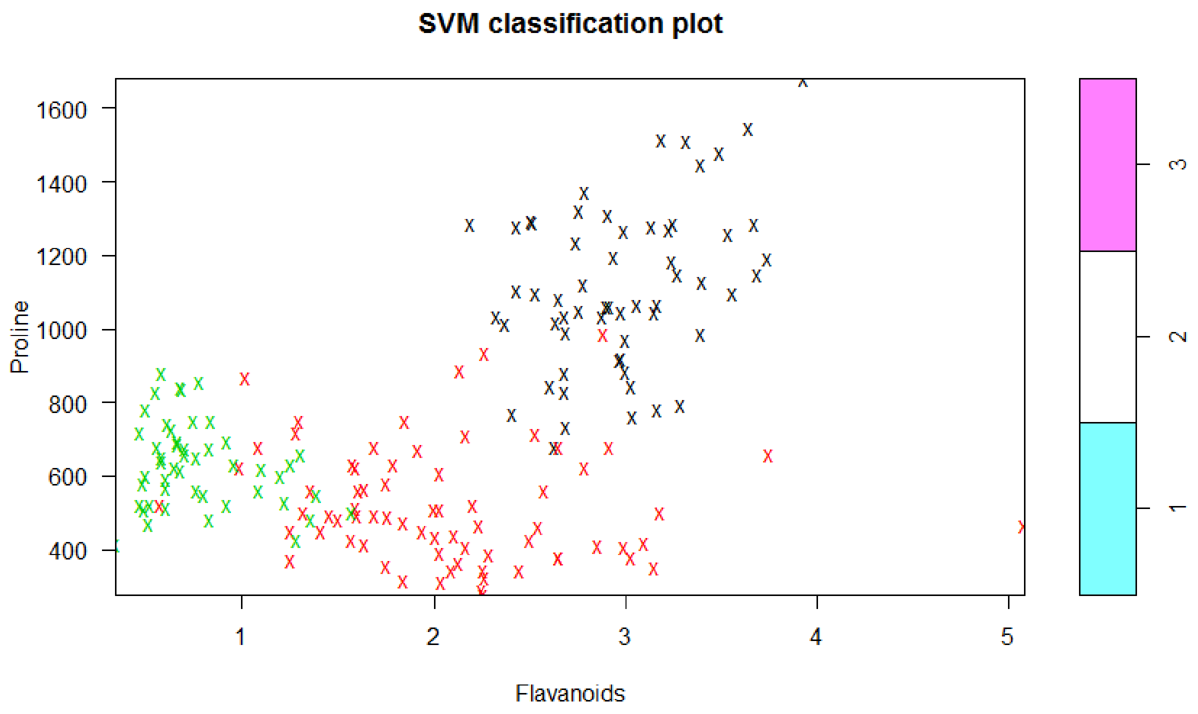


Figure 12 shows a scatterplot of the data in a 2-dimensional space utilizing the Proline and Flavanoids variables. The color coding gives an idea of the separation of the data points into each of the three wine classes. This plot suggests that there is some non-linearity in the decision boundaries between the classes.

Again, the SVM model performance was compared to other models through the use of precision in the confusion matrix. Figure 13 shows the in-sample confusion matrix for the SVM model. The support vector machine's overall in-sample performance was 100% and is shown in the conclusion section along with other model performance.

**Figure 13: Support Vector Machine Confusion Matrix**

| | **Predicted** | | |
|---|---|---|---|
| **Actual** | 1 | 2 | 3 |
| 1 | 100% | 0% | 0% |
| 2 | 0% | 100% | 0% |
| 3 | 0% | 0% | 100% |

## Artificial Neural Network (ANN)

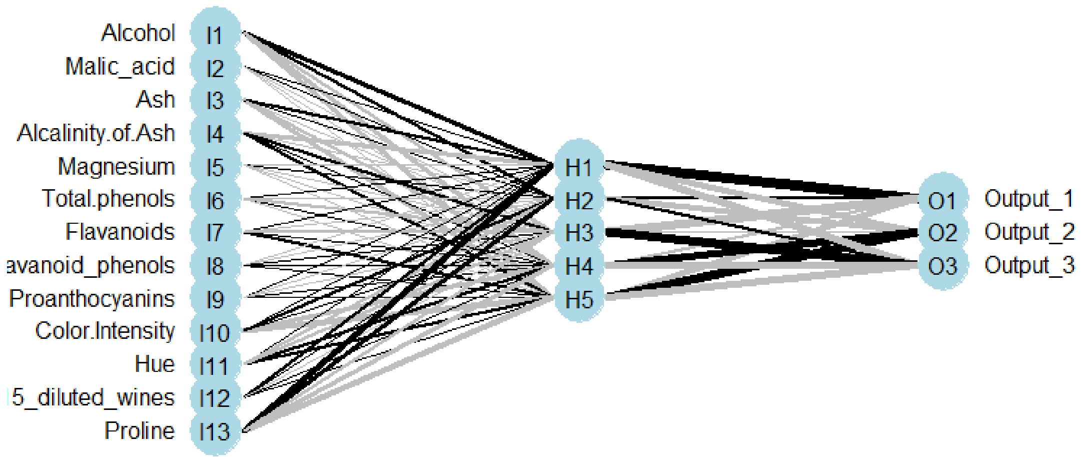
The final model I ran was an artificial neural network (ANN). Typically, it is recommended that the inputs be scaled to a mean of zero and a standard deviation of one prior to running the ANN model-building process. I discussed the data scaling in the data transformation section. Figure 14 shows the parameters that can be set as a part of the ANN model-building process.

**Figure 14: ANN parameters**

| **Parameter** | **Value** |
|---|---|
| Hidden Layers | 1 |
| Nodes | 5 |
| Weight Decay | .1 |
| Iterations | 50 |

Since this is a small dataset, and to keep things less complicated, I utilized a single hidden layer with 5 nodes. Typically, the number of nodes in each hidden layer ranges between 5-100 and increases as the number of parameters and observations increases. Again, working with a small dataset, I chose a low number of nodes within the single hidden layer. Weight decay helps prevent overfitting of the data through regularization. Weight decay shrinks some of the model's weights toward zero, as in ridge regression. Finally, 50 iterations were performed as a part of the model-building process. A visual depiction of the layout of the ANN model is shown in figure 15. The inputs begin on the left-hand side of the diagram, the hidden layer with the five nodes is in the center, and the model output classifies observations into one of three classes.

**Figure 15: ANN Model**



The ANN in-sample confusion matrix and precision is shown in figure 16. The ANN's overall in-sample performance was 100% and is shown in the conclusion section along with other model performance.

**Figure 16: Artificial Neural Network Confusion Matrix**

| | **Predicted** | | |
|---|---|---|---|
| **Actual** | 1 | 2 | 3 |
| 1 | 100% | 0% | 0% |
| 2 | 0% | 100% | 0% |
| 3 | 0% | 0% | 100% |

## Conclusions

In conclusion, the data was relatively clean in that it had no missing values. The dependent variable is pretty evenly distributed among the observations, so the data was well-suited for training the models. It appears that there are some correlations in the variables. Specifically, there are a few variables such as flavonoids and total phenols that might collectively help distinguish between red and white wines. The model-based EDA and the classification tree helped confirm those hunches.

Figure 17 shows the in-sample performance of the three models built for this assignment. Again, figure 17 shows the overall in-sample precision (percent correctly classified) for each model. All models fit the data very well in-sample. However, model performance should generally be tested against a hold-out sample to confirm performance on unseen data. Without out-of-sample testing, overfitting the model to the training data may occur. With larger datasets, model computation and performance needs to be considered. As we are seeing in our group projects, SVM and many of these more advanced modeling techniques are highly computational. Processing time and power required can grow exponentially as the data and model complexity increase.

**Figure 17: Model Results**

| Model | In-Sample Precision | Model Rank |
|---|---|---|
| Random Forest | 98% | 2 |
| Support Vector Machine (SVM) | 100% | 1 |
| Artificial Neural Network (ANN) | 100% | 1 |

Finally, based on the wine characteristics, it seems likely that class 1 might represent red wines, class 3 might represent white wines, and class 2 might represent something in the middle, such as a rose.

# 1 Appendix

## R Code

**Process for scaling the data:**

```
# standardize the data?
# standardize the independent variables only
scaled <- as.data.frame(scale(fullDataSet[,2:14], center=TRUE, scale=TRUE))

# add back in the class variable, unscaled
scaled$class <- fullDataSet$class

str(scaled)

summary(scaled)
```

**Trellis graphs using Lattice:**

```
# Try Lattice
bwplot(~ Color.intensity | factor(class), data=fullDataSet, layout= c(3,1) )

par(mfrow = c(3,3))  # this doesn't help
bwplot(~ Alcalinity.of.ash | factor(class), data=fullDataSet, main="Alcalinity of Ash Boxplots", layout=
c(3,1) )
bwplot(~ Magnesium | factor(class), data=fullDataSet, main="Magnesium Boxplots", layout= c(3,1) )
bwplot(~ Proline | factor(class), data=fullDataSet, main="Proline Boxplots", layout= c(3,1) )



xyplot(class ~ Color.intensity | factor(class), data = fullDataSet)
xyplot(class ~ Flavanoids | factor(class), data = fullDataSet, main="Flavanoids by Class")
```

**Classification Trees:**

```
# ----------------
# MODEL BASED EDA
# ----------------

# ----------------------------
# Fitting Classification Trees
# ----------------------------

library(tree)
# library(ISLR)
attach(fullDataSet)
```

```
# Create a binary response variable, attach to carseats dataframe
# High=ifelse(Sales<=8,"No","Yes")
# Carseats=data.frame(Carseats,High)

# Fit a classification tree
tree.wine=tree(class~., fullDataSet)
summary(tree.wine)

# Display the tree graphically
plot(tree.wine)
text(tree.wine,pretty=0)
tree.wine


# Run tree on scaled data
# ----------------------

library(tree)
# library(ISLR)
attach(scaled)

# Create a binary response variable, attach to carseats dataframe
# High=ifelse(Sales<=8,"No","Yes")
# Carseats=data.frame(Carseats,High)

# Fit a classification tree
tree.wine.scaled=tree(class~., scaled)
summary(tree.wine.scaled)

# Display the tree graphically
plot(tree.wine.scaled)
text(tree.wine.scaled,pretty=0)
tree.wine.scaled
```

**Model Suite:**

```
# ------------------------------
# Artificial Neural Network (ANN)
# ------------------------------

library(RSNNS)


#shuffle the vector
fullDataSet <- fullDataSet[sample(1:nrow(fullDataSet),length(1:nrow(fullDataSet))),1:ncol(fullDataSet)]
```

```
irisValues <- fullDataSet[,2:14]
irisTargets <- decodeClassLabels(fullDataSet[,1])
#irisTargets <- decodeClassLabels(iris[,5], valTrue=0.9, valFalse=0.1)

iris <- splitForTrainingAndTest(irisValues, irisTargets, ratio=0.15)
iris <- normTrainingAndTestSet(iris)


# maybe adding c(1,2,3) specifies different numbers of nodes for 3 hidden layers??
model <- mlp(iris$inputsTrain, iris$targetsTrain, size=5, learnFuncParams=c(0.1),    # is this the weight decay?
        maxit=50, inputsTest=iris$inputsTest, targetsTest=iris$targetsTest)

summary(model)
model
weightMatrix(model)
extractNetInfo(model)

par(mfrow=c(2,2))
plotIterativeError(model)

predictions <- predict(model,iris$inputsTest)

plotRegressionError(predictions[,2], iris$targetsTest[,2])

confusionMatrix(iris$targetsTrain,fitted.values(model))
confusionMatrix(iris$targetsTest,predictions)



# Visualizations....
library(NeuralNetTools)

par(mfrow=c(1,1))

vars <- garson(model, x_lab=c("Alcohol", "Malic_acid", "Ash", "Alcalinity.of.Ash", "Magnesium", "Total.phenols",
                "Flavanoids", "Nonflavanoid_phenols", "Proanthocyanins", "Color.Intensity",
                "Hue", "OD280_OD315_diluted_wines", "Proline"))

plotnet(model, x_names=c("Alcohol", "Malic_acid", "Ash", "Alcalinity.of.Ash", "Magnesium", "Total.phenols",
             "Flavanoids", "Nonflavanoid_phenols", "Proanthocyanins", "Color.Intensity",
             "Hue", "OD280_OD315_diluted_wines", "Proline"))
```

```
# --------------
# Random Forests
# --------------

library(randomForest)

# Random forest using 4 random variables
set.seed(1)
rf.wine=randomForest(class~.,data=fullDataSet,mtry=4,importance=TRUE, ntree=500)



# make predictions
# yhat.rf = predict(rf.boston,newdata=Boston[-train,])
# mean((yhat.rf-boston.test)^2)

# View the importance of each variable
importance(rf.wine)
varImpPlot(rf.wine)  # Plot the importance measures





# ---------------------------
# Support Vector Machine (SVM)
# ---------------------------

library(e1071)

# ------------------------
# SVM with Multiple Classes
# ------------------------

# if dependent variable is a factor with >2 levels,
# svm will use one vs one approach
svmfit=svm(class~., data=fullDataSet, kernel="radial", cost=10, gamma=1)

# need to provide 2 dimensions to plot when more than one ind var
plot(svmfit, data=fullDataSet, Proline~Flavanoids)


table(actual=fullDataSet$class, predicted=svmfit$fitted)
```