Andrea Bruckner
Predict 454 Sec 55

Individual Assignment #4

Introduction

In this report I am using the wine recognition data set from the UC Irvine Machine Learning Repository in order to determine the origin of wine based on its chemical properties. After exploring the data and fitting a few naïve models, I fit three final models: a Random Forest model, a Support Vector Machine (SVM), and a neural network model. An appendix of R code relevant to the tables and graphics presented is located at the end of this report.

Data Quality Check

In order to familiarize myself with the data, I first performed a data quality check. The data used in this report is the wine recognition data set available online at the UC Irvine Machine Learning Repository. The data set contains 178 observations and 14 variables. **Table 1** provides information about each variable after data conversions.

Variable	Type	Description
Class	Factor	Cultivars in Italy (3 possible cultivars)
Alcohol	Numeric	Alcohol content
Malic_Acid	Numeric	Malic acid content
Ash	Numeric	Ash content
Ash_Alcalinity	Numeric	Alkalinity of ash content
Magnesium	Numeric	Magnesium content
Total_Phenols	Numeric	Total phenols (contribute to wine color and structure)
Flavanoids	Numeric	Flavanoid content (phenol contributing to red wines)
Nonflavanoid_Phenols	Numeric	Non-flavanoid phenol (phenol contributing to white wines)
Proanthocyanins	Numeric	Proanthocyanin content (principal polyphenols in red wine)
Color_Intensity	Numeric	Intensity of wine color
Hue	Numeric	Color of wine
OD280_OD315	Numeric	OD280_OD315 of diluted wines
Proline	Numeric	Proline content (an amino acid)

Table 1: Variables in Wine Data

The majority of variables were originally numeric, with the exception of Class, Magnesium, and Proline. I decided to convert Magnesium and Proline from integers to numeric values in order to facilitate analysis without compromising the quality of these variables. Because the Class variable assumes only three values (1, 2, or 3), I decided to convert it from an integer to a factor in order to approach this data set as a classification problem.

From running a summary function on the data, I was able to discover that this data set does not contain any missing values. I learned that the majority of observations in the data set, 71 wines, are Class 2, while 59 wines are Class 1 and 48 wines are Class 3. The summary statistics of the variables did not indicate any obvious outliers or abnormalities, with the possible exception of

Proline. As evident in **Table 2** below, the difference between the 3rd quartile and minimum value is approximately the same as the difference between the 3rd quartile and maximum value.

Proline
Min.: 278.0
1st Qu.: 500.5
Median: 673.5
Mean: 746.9
3rd Qu.: 985.0
Max.: 1680.0

Table 2: Proline Summary

After examining numeric summaries of the variables, I decided to calculate the various quantiles for each variable. **Table 3** shows the quantiles for Proline, which was the only variable that displayed a large difference between some neighboring quantiles.

Proline Quantiles	1%	5%	25%	50%	75%	90%	95%	99%
	306.94	354.55	500.5	673.5	985	1261.5	1297.25	1522.36

Table 3: Proline Quantiles

I then created plots of all the variables in order to better understand their shape and range. **Figure 1** depicts histograms of each variable.

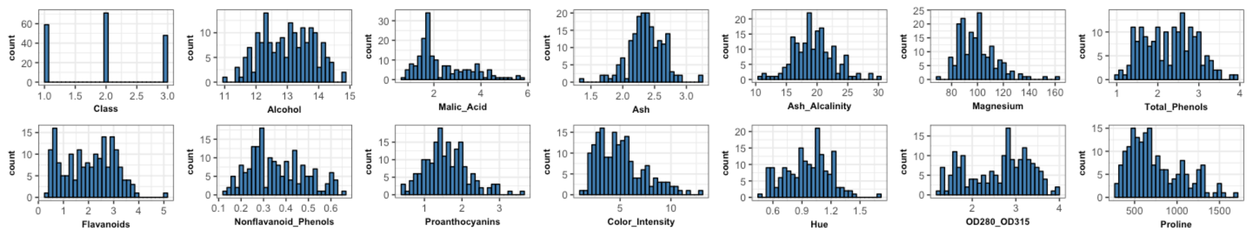


Figure 1: Visualization of Variables

Visualizing the data often makes it easier to detect outliers or abnormalities than summary tables alone. From the plots above, we see that none of these variables has a textbook normal distribution. A few variables, such as Ash, Flavanoids, and Hue, look like they might contain outliers as well. In order to better detect potential outliers, I created box plots of each variable. **Figure 2** shows the variables I originally suspected as having outliers (Proline, Ash, Flavanoids, and Hue), and **Figure 3** shows other variables with potential outliers that I did not notice from looking just at the histograms and summary statistics. The light blue circles indicate potential outliers, but a more thorough examination of the data is necessary to draw conclusions.

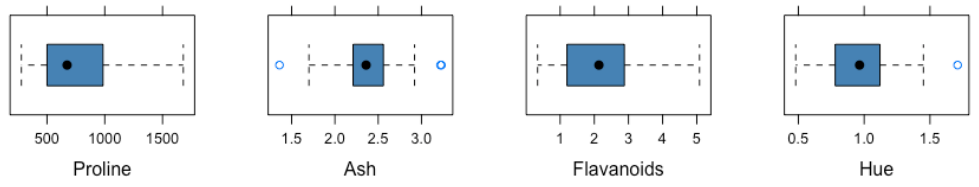


Figure 2: Box Plots of Variables Suspected of Having Outliers

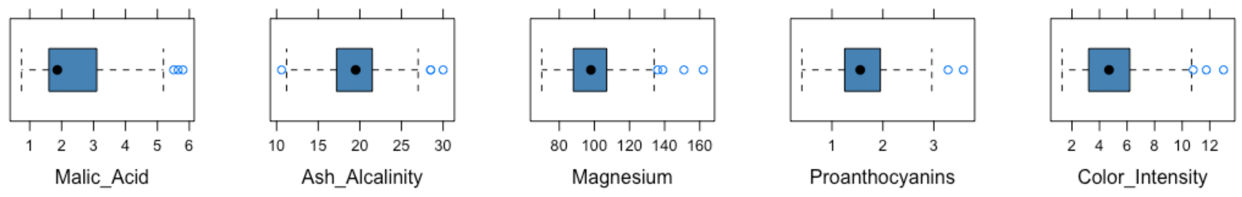


Figure 3: Box Plots of Additional Variables with Potential Outliers

Exploratory Data Analysis

After taking inventory of the data to make sure the data are correct and of good quality, I conducted a more in-depth analysis, known as EDA, to detect interesting relationships in the data. In the previous section, I mentioned that the type of statistical problem presented by the data is a classification problem. My goal is to use the variables representing the chemical and physical properties of wine in order to predict the origin, or cultivar, of wine. Both numeric and visual EDA are appropriate for classification problems; however, not all types of graphics, such as scatterplots or pie charts, are useful for this data set.

Before creating any plots, I decided to first calculate the correlations among just the numeric variables to get a sense of their relationships to each other without considering how they relate to Class. **Table 4** lists the significant correlations in the data set.

Variable 1	Variable 2	Correlation
Flavanoids	Total_Phenols	0.8645635
OD280_OD315	Flavanoids	0.7871939
OD280_OD315	Total_Phenols	0.6999494
Proanthocyanins	Flavanoids	0.6526918
Proline	Alcohol	0.6437200
Proanthocyanins	Total_Phenols	0.6124131
OD280_OD315	Hue	0.5654683
Hue	Malic_Acid	-0.5612957
Color_Intensity	Alcohol	0.5463642
Hue	Flavanoids	0.5434786
Nonflavanoid_Phenols	Flavanoids	-0.5378996
Hue	Color_Intensity	-0.5218132
OD280_OD315	Proanthocyanins	0.5190671
OD280_OD315	Nonflavanoid_Phenols	-0.5032696

Table 4: Significant Correlations

From the table above, we can see that Flavanoids and OD280_OD315 each have five significant correlations with other variables. These appear to be important variables in wine and warrant a closer look, especially with how they might relate to Class. In **Figure 4**, we can see the distribution of Flavanoids by Class using histograms and boxplots. The distributions and means of the distributions vary according to each Class. Also, the box plot suggests that some values of Flavanoids might be potential outliers in Class 2 and Class 3 but not Class 1.

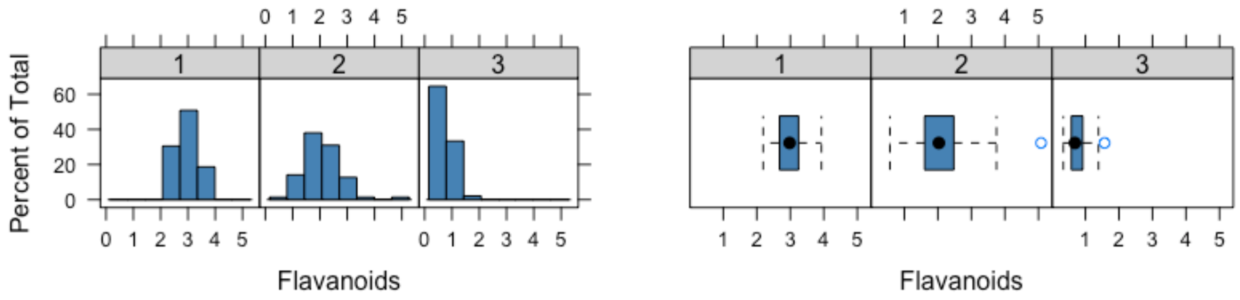


Figure 4: Flavanoids by Class

In **Figure 5**, we can see the distribution of OD280_OD315 by Class. In Class 2, the range of OD280_OD315 is much larger than in the other two classes. Class 1 contains OD280_OD315 values that appear in the mid to upper range of values, while Class 3 has mostly lower OD280_OD315 values as well as some potential outliers.

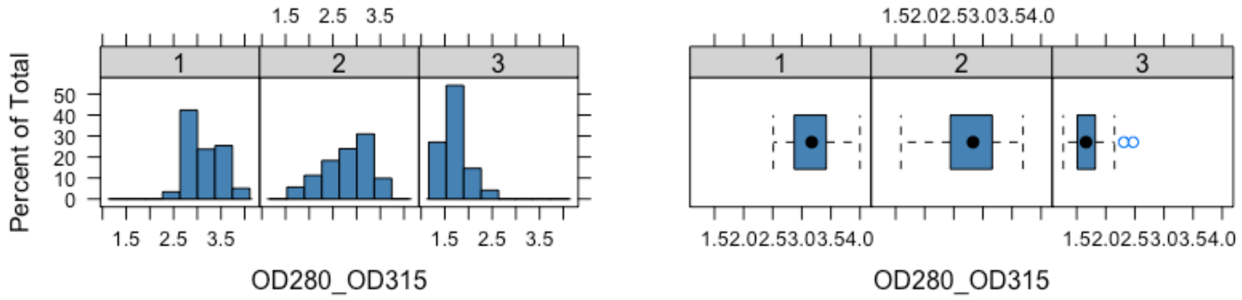


Figure 5: OD280_OD315 by Class

After examining significant correlations among numeric variables and a couple relationships between predictor variables and Class, I created correlation plots of all the variables by Class. In **Figure 6**, we can see that there are strong positive correlations between Flavanoids and Total_Phenols, Flavanoids and Color_Intensity, and Total_Phenols and Color_Intensity. There are also slightly less strong positive correlations between Color_Intensity and Proline, Flavaoids and Proanthocyanins, and Ash and Ash_Alcalinity. Only a few weak negative correlations, such as Hue and Malic_Acid, exist.

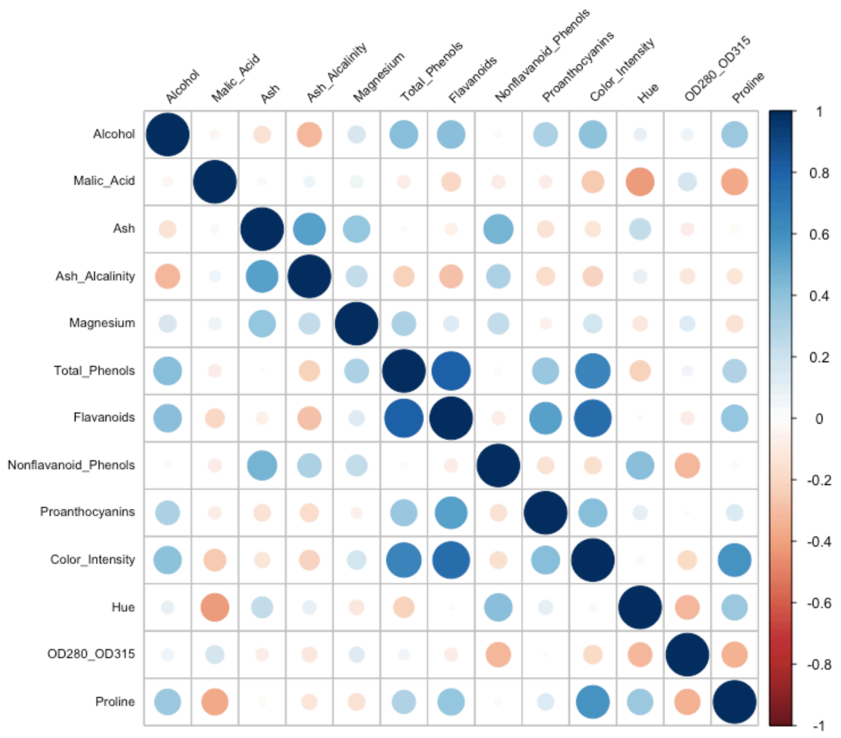


Figure 6: Class 1 Correlations

In **Figure 7**, we can see that there are strong positive correlations between Flavanoids and Total_Phenols (similar to **Figure 6**), and Ash and Ash_Alcalinity.

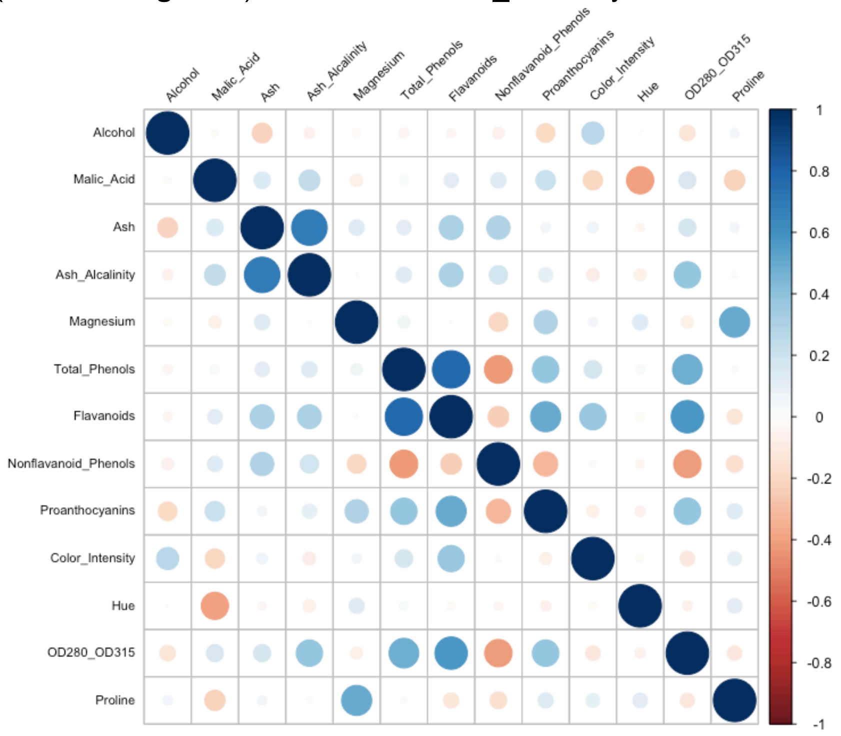


Figure 7: Class 2 Correlations

In **Figure 8**, we can see that there are strong positive correlations between Ash and Ash_Alcalinity (similar to **Figure 7**) and slightly weaker positive correlations between Color_Intensity and Proanthocyanins, and Total_Phenols and Proanthocyanins. Class 3 is the only class that shows strong negative correlations between variables, namely Flavanoids and Nonflavanoid_Phenols, and Color_Intensity and Hue, to a weaker extent.

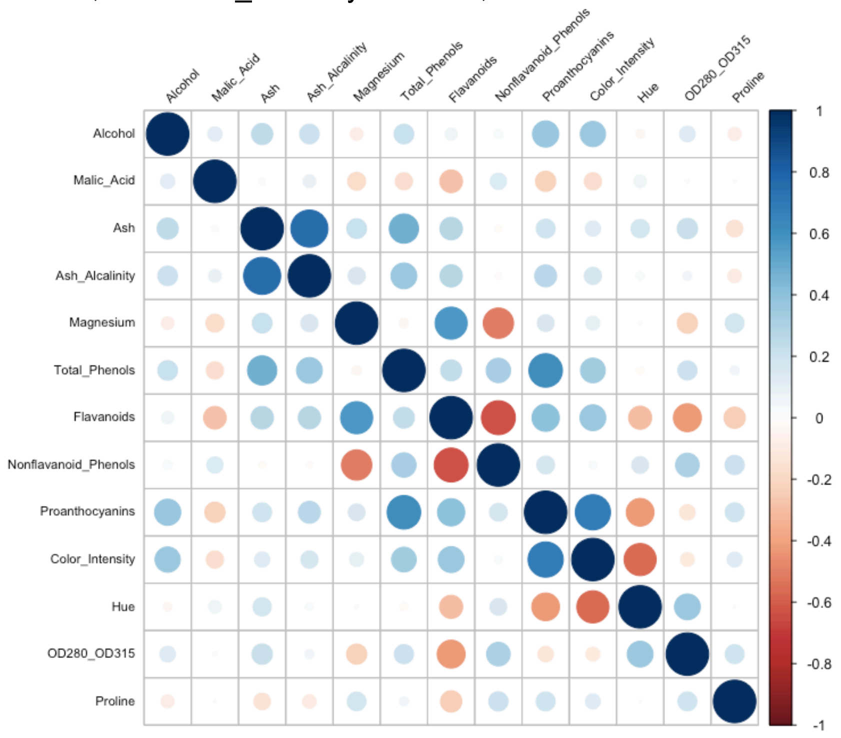


Figure 8: Class 3 Correlations

In addition to gleaning insight from the above plots and tables, I fit a tree model to get a quick overview of important relationships in the data. **Figure 9** shows a tree created from modeling all the variables as predictors of Class. Each leaf of the tree contains three rows of numbers: the first row indicates the dominant Class of that leaf; the second row indicates the percentage of observations belonging to each Class within the leaf; and the third row indicates the percentage of observations from the entire data set present in the leaf. For instance, the first leaf contains all of the data (thus 100% in the third row), and the majority of observations, 40 percent, belong to Class 2. We can calculate the percentages shown in the leaf by dividing the information we learned during the Data Quality Check (59 wines are Class 1, 71 wines are Class 2, and 48 wines are Class 3) by the total number of observations, 178, in the data set.

The first split in the tree occurs between observations that have a value of Proline greater than or equal to 755 and those that do not. The next level of leaves shows that 38 percent of observations follow that decision rule, while 62 percent of observations have a value of Proline less than 755. The majority (85 percent) of observations with a high Proline value are Class 1, while 60 percent of observations with a lower Proline value are Class 2. Reading the entire tree yields even more insights not discussed here.

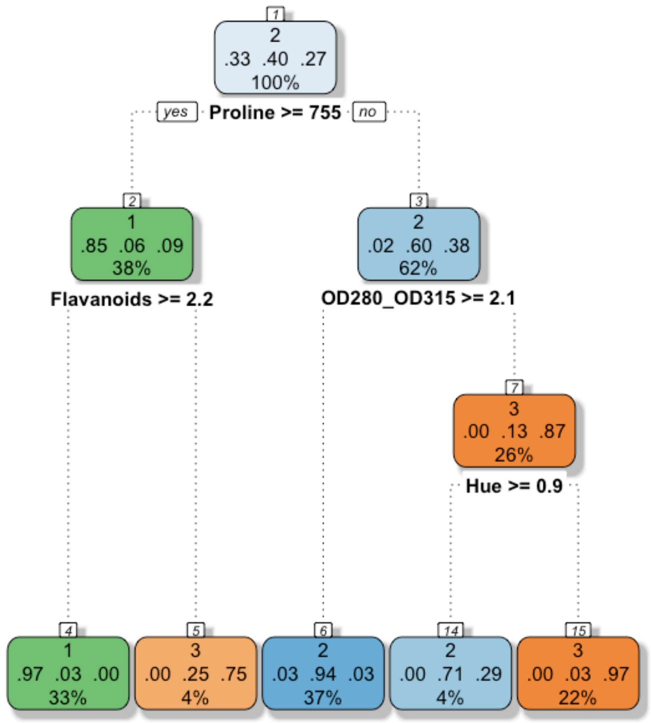


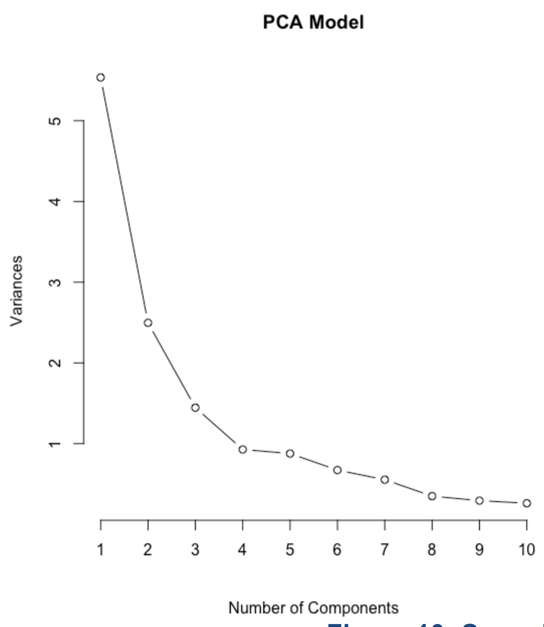
Figure 9: Tree Plot

Conducting EDA can help us identify interesting relationships that we can use when building models. In **Figures 4 and 5** we were able to see that predictor variables might have different distributions and means according to the response variable, Class. In **Table 4** we were able to identify which variables have the strongest correlations with each other. **Figures 6, 7, and 8** showed us how correlations vary by Class, and **Figure 9** gave us a glimpse of how the data splits when fitted to a tree model. Knowing all of this can help us determine which variables to include in our models, as well as help us identify a potentially bad model if its output does not align with our EDA.

Model-Based Exploratory Data Analysis

After exploring the data, I fit some naïve models. Because this is a classification problem involving three possible values for Class, not all modeling techniques are appropriate. For instance, ordinary least squares regression and logistic generalized additive models are just a couple modeling techniques that would not be appropriate for this data set. In addition to the tree model fit above, I decided to fit three types of naïve models: a principal components analysis (PCA) model, a linear discriminant analysis (LDA) model, and a Random Forest model.

In **Figure 10** below, we see two plots from the PCA model. The first plot is a scree plot, which suggests that perhaps four principal components explain most of the variability in the data. The other plot is a biplot depicting the first two principal components. The biplot reveals that the PCA model did a pretty good job separating the classes. Variables that spread out vertically from the center of the plot, such as Color_Intensity and Alcohol, have a strong effect on the second principal component, while variables that spread out horizontally from the center, such as Flavanoids and Ash_Alcalinity, have a strong effect on the first principal component.



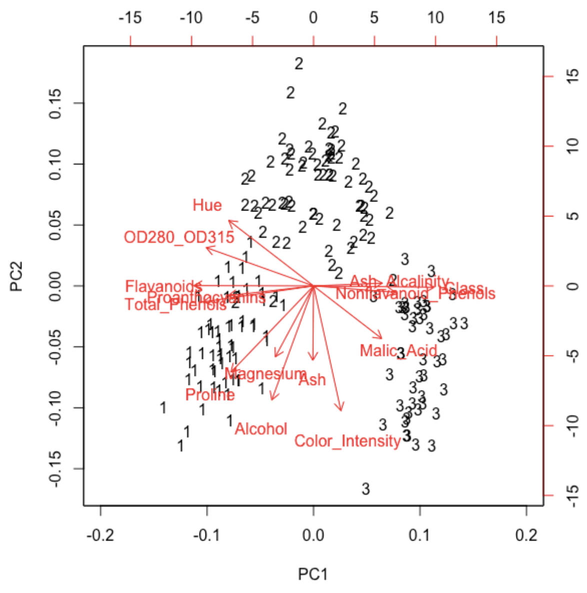


Figure 10: Scree Plot and Biplot of PCA Model

After creating a PCA model, which uses unsupervised learning, I created an LDA model, which is a supervised learning technique that uses class information. **Figure 11** shows the results of the LDA model. As the plot shows, the model effectively separated the classes. The LDA model performed better than the PCA model since there is no overlapping of classes, whereas in the PCA model, a few Class 2 points appear in the Class 1 and Class 3 clusters.

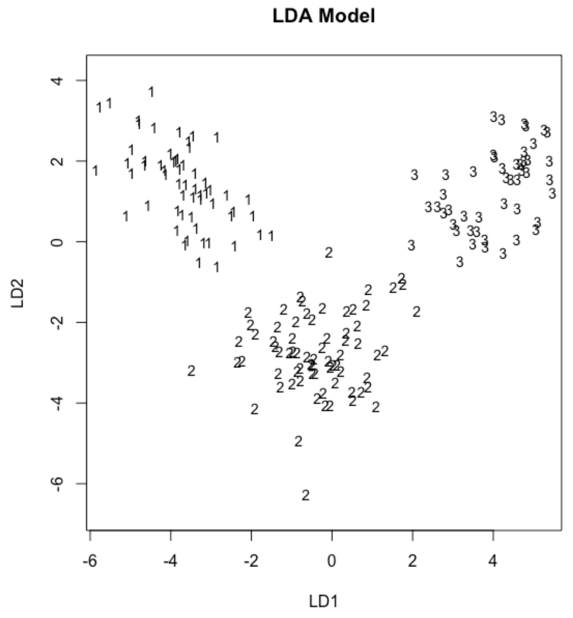


Figure 11: LDA Model Plot

The last model I created was a Random Forest model. **Figure 12** shows a plot of the variables ranked by importance. According to this model, the Proline, Flavanoids, and Color_Intensity variables are all very important in determining Class.

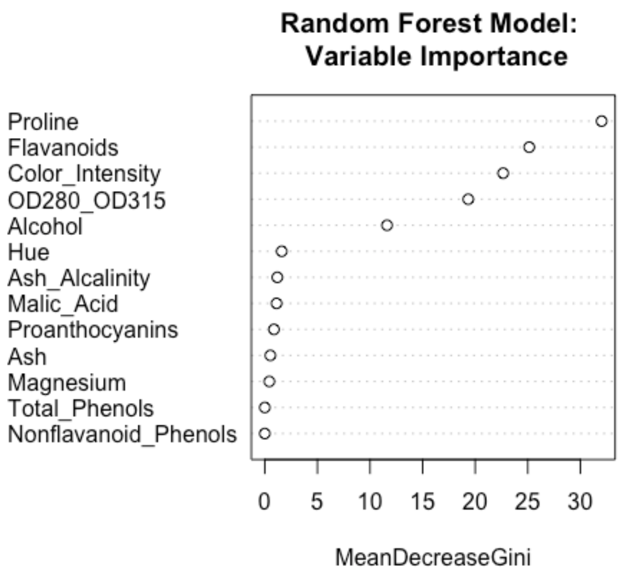


Figure 12: Random Forest Variable Importance Plot

Both PCA and LDA models are useful for this statistical problem because they reduce the dimensions of the data. However, the tree model and Random Forest model are also good to use because they call attention to specific variables important in determining Class. In the next section, I will fit my final models: a Random Forest model, an SVM, and a neural network model.

Models and Analysis

Because this is a small data set containing 178 observations, I fit and evaluated models in-sample only. I used 10-fold cross-validation repeated three times for each of the three modeling techniques. Most of these "black-box" models do not lend themselves to interpretable visualizations and are better evaluated according to statistics, such as accuracy and kappa. I will briefly discuss each model below and evaluate them in the next section.

Model 1: Random Forest Model

I created a forest of 500 trees where each split tried 2 variables. It had a 1.12 percent out-of-bag error rate estimate and a 0 percent in-sample error rate. In **Figure 13** we can see a plot showing the importance of each variable. Similar to the plot in **Figure 12**, Proline is the most important variable in predicting the response variable. The top six most important variables are the same for both the naïve and final Random Forest model, though the order slightly differs. Nonfalvaoid_Phenols is deemed the least important predictor in both the naïve and final Random Forest model.

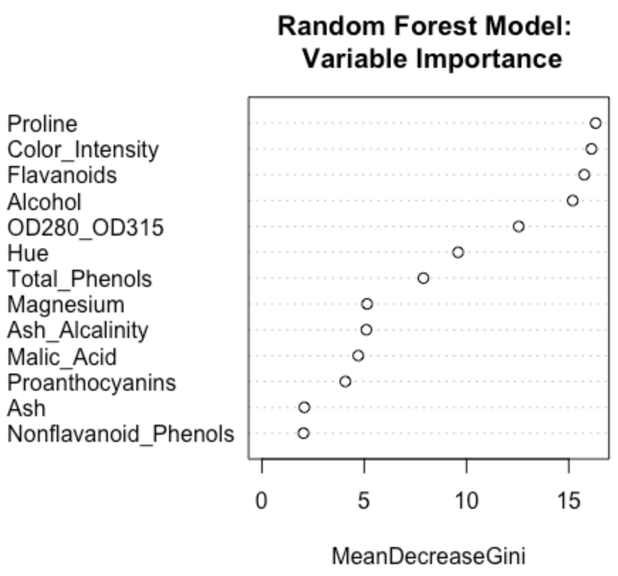


Figure 13: Random Forest Variable Importance Plot

Model 2: SVM Model

I fit an SVM model next. In addition to using 10-fold cross-validation repeated three times, I used a Gaussian Radial Basis kernel for this model. The model had a total of 69 support vectors and a training error rate of 0. **Figure 14** shows the variable importance plots for each class. Interestingly, Color_Intensity is the most important predictor rather than Proline, as was suggested in the plots for both the Random Forest naïve and final models. Proline appears to play a moderate role in predicting Class 2 and Class 3 for the SVM model, but it does not play a role in predicting Class 1 according to the figure below.

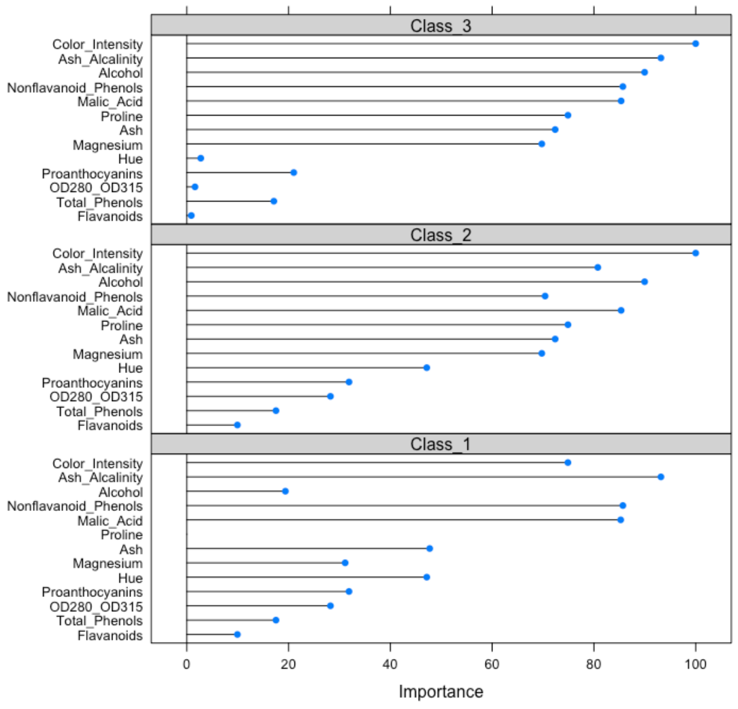


Figure 14: SVM Variable Importance Plots

Model 3: Neural Network Model

The last model I fit was a neural network model. The resulting network contained 13 input variables, 5 hidden units, and 3 output variables (Class 1, 2, and 3). It also had 88 weights and a 0.10 decay value. **Figure 15** shows how the number of hidden units influence the accuracy and weight decay. A model containing 5 hidden units performs just slightly better than a model containing 3 hidden units.

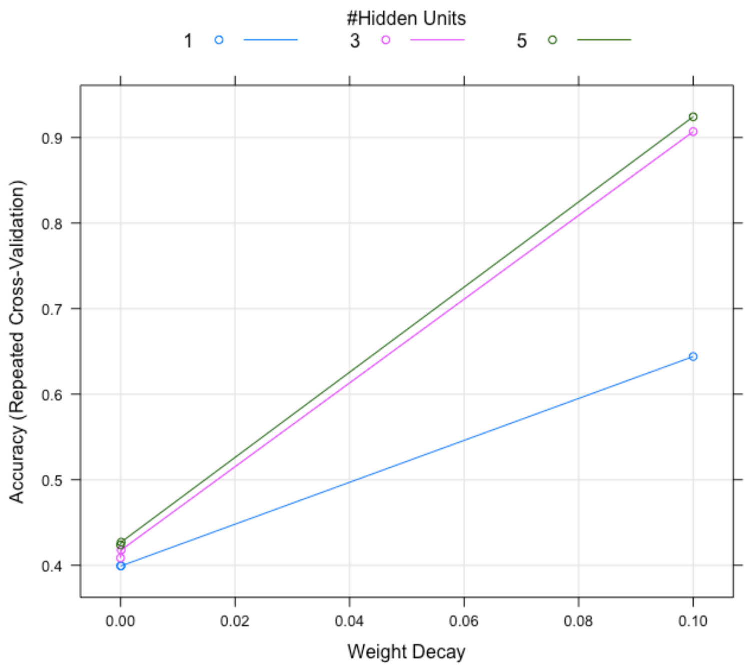


Figure 15: Neural Network Accuracy

Model Comparison

Once I fit all my models, I created confusion matrices of their predictions. All models resulted in the same in-sample predictions. **Table 5** shows the confusion matrix that each model produced.

Model	Actual/Predicted	In-Sample Performance		
		Class 1	Class 2	Class 3
Random Forest; SVM; Neural Network	Class 1	59	0	0
	Class 2	0	71	0
	Class 3	0	0	48

Table 5: Confusion Matrix

After examining the confusion matrix, I calculated the accuracy and kappa values for each model to evaluate their performance in-sample. I included the kappa statistic, which takes into account random chance, because the class distributions were unequal. **Table 6** provides a summary of accuracy and kappa for each model.

Model	In-Sample Performance	
	Accuracy	Kappa
Random Forest	1	1
SVM	1	1
Neural Network	0.9943820225	0.9914611916

Table 6: Summary of Model Performance

The Random Forest and SVM model performed equally well, while the neural network model performed slightly worse. I am not sure why the accuracy for the neural network model did not equal 1 since its confusion matrix revealed perfectly predicted classes. Nevertheless, this model slightly underperformed the other two models in terms of kappa. I am not surprised that these models performed so well since the various naïve models I built suggested the response classes were easy to distinguish. Of all the final models I created, I would prefer to use either the Random Forest or the SVM model for deployment since these had the best performance and can visually display the important predictors. The variable importance plot for the Random Forest model is simpler, but the plot for the SVM model shows more detail for each class and can reveal more nuanced relationships in the data.

Conclusion

This report demonstrated various multiclass classification modeling techniques for predicting the origin of wine. After assessing the quality of the data, I explored interesting relationships in the data by creating plots and fitting naïve model. I fit three final models using 10-fold cross-validation repeated three times: a Random Forest model, an SVM model, and a neural network model. The Random Forest and SVM models performed the best.

Appendix A: Relevant R Code

```
#############################################################
# Load Data
#############################################################

# Read data
wine <- read.csv("http://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data",
header=F)

# Create column names
colnames(wine) <- c("Class", "Alcohol", "Malic_Acid", "Ash", "Ash_Alcalinity",
                    "Magnesium", "Total_Phenols", "Flavanoids",
                    "Nonflavanoid_Phenols", "Proanthocyanins",
                    "Color_Intensity", "Hue", "OD280_OD315", "Proline")

#############################################################
# Data Quality Check
#############################################################

# Explore the data
dim(wine) # 178  14
str(wine) # all num except Class, Magnesium, and Proline are int
summary(wine) # no NA/missing values
# Perhaps Proline has some outliers--the difference between the 3rd quartile and max value is
large and is the same as the amount between the 3rd quartile and the min value (~700)

# Make Magnesium and Proline numeric instead of integers (since it will be easier to work with
all the same data type)
wine$Magnesium <- as.numeric(wine$Magnesium)
wine$Proline <-as.numeric(wine$Proline)

# Plot the variables
plot_vars <- function (data, column){
  ggplot(data = wine, aes_string(x = column)) +
    geom_histogram(color =I("black"), fill = I("steelblue"))+
    xlab(column) + theme_bw() + theme(axis.title=element_text(size=8, face="bold"))
}

plots <- lapply(colnames(wine), plot_vars, data = wine)
length(plots)
do.call("grid.arrange", c(plots, nrow=2))

# Make Class a factor since it takes only 3 values (representing 3 different cultivars/orgins of
wined)
wine$Class <- as.factor(wine$Class)

# Examine quantiles of wine variables
sapply(wine[2:14], quantile, probs = c(0.01, 0.05, 0.25, 0.50, 0.75, 0.90, 0.95, 0.99))
```

```
# Create box plot of Proline to examine potential outliers
Proline_box <- bwplot(~ Proline, data = wine, par.settings = list(
  box.umbrella=list(col= "black"),
  box.dot=list(col= "black"),
  box.rectangle = list(col= "black", fill = "steelblue")))

# Create more box plots of variables to examine potential outliers indicated in histogram
Ash_box <- bwplot(~ Ash, data = wine, par.settings = list(
  box.umbrella=list(col= "black"),
  box.dot=list(col= "black"),
  box.rectangle = list(col= "black", fill = "steelblue")))

Flavanoids_box <- bwplot(~ Flavanoids, data = wine, par.settings = list(
  box.umbrella=list(col= "black"),
  box.dot=list(col= "black"),
  box.rectangle = list(col= "black", fill = "steelblue")))

Hue_box <- bwplot(~ Hue, data = wine, par.settings = list(
  box.umbrella=list(col= "black"),
  box.dot=list(col= "black"),
  box.rectangle = list(col= "black", fill = "steelblue")))

grid.arrange(Proline_box, Ash_box, Flavanoids_box, Hue_box, ncol=4)

# Plot additional variables with potential outliers
Color_Intensity_box <- bwplot(~ Color_Intensity, data = wine, par.settings = list(
  box.umbrella=list(col= "black"),
  box.dot=list(col= "black"),
  box.rectangle = list(col= "black", fill = "steelblue")))

Proanthocyanins_box <- bwplot(~ Proanthocyanins, data = wine, par.settings = list(
  box.umbrella=list(col= "black"),
  box.dot=list(col= "black"),
  box.rectangle = list(col= "black", fill = "steelblue")))

Magnesium_box <- bwplot(~ Magnesium, data = wine, par.settings = list(
  box.umbrella=list(col= "black"),
  box.dot=list(col= "black"),
  box.rectangle = list(col= "black", fill = "steelblue")))

Ash_Alcalinity_box <- bwplot(~ Ash_Alcalinity, data = wine, par.settings = list(
  box.umbrella=list(col= "black"),
  box.dot=list(col= "black"),
  box.rectangle = list(col= "black", fill = "steelblue")))

Malic_Acid_box <- bwplot(~ Malic_Acid, data = wine, par.settings = list(
  box.umbrella=list(col= "black"),
  box.dot=list(col= "black"),
  box.rectangle = list(col= "black", fill = "steelblue")))
```

```
grid.arrange(Malic_Acid_box, Ash_Alcalinity_box, Magnesium_box, Proanthocyanins_box,
Color_Intensity_box, ncol=5)

########################################################
# EDA
########################################################

# Examine correlations among just numeric variables
c <- cor(wine[2:14], use="complete.obs")

correlations <- data.frame(c)

significant.correlations <- data.frame(
  var1 = character(),
  var2 = character(),
  corr = numeric())

for (i in 1:nrow(correlations)){
  for (j in 1:ncol(correlations)){
    tmp <- data.frame(
      var1 = as.character(colnames(correlations)[j]),
      var2 = as.character(rownames(correlations)[i]),
      corr = correlations[i,j])

    if (!is.na(correlations[i,j])) {
      if (correlations[i,j] > .5 & as.character(tmp$var1) != as.character(tmp$var2)
          | correlations[i,j] < -.5 & as.character(tmp$var1) != as.character(tmp$var2) ) {
        significant.correlations <- rbind(significant.correlations,tmp) }
    }
  }
}

significant.correlations <-
significant.correlations[order(abs(significant.correlations$corr),decreasing=TRUE),]
significant.correlations <-
significant.correlations[which(!duplicated(significant.correlations$corr)),]
significant.correlations

# Create object containing names of only numeric variables
noClass <- colnames(wine[2:14])

# Visualize correlations between numeric variables and each Class factor
corrplot(cor(wine[wine$Class == 1, noClass]),
        tl.col = "black", tl.cex = 0.7, tl.srt = 45,
        title = "Wine Class 1 Correlations",
        mar=c(1,3,1,3))

corrplot(cor(wine[wine$Class == 2, noClass]),
        tl.col = "black", tl.cex = 0.7, tl.srt = 45,
        title = "Wine Class 2 Correlations",
        mar=c(1,3,1,3))
```

```
corrplot(cor(wine[wine$Class == 3, noClass]),
        tl.col = "black", tl.cex = 0.7, tl.srt = 45,
        title = "Wine Class 3 Correlations",
        mar=c(1,3,1,3))

# Create plots of just a few significant variables in relation to Class
Flavanoids_Class_hist <- histogram(~ Flavanoids | Class, data = wine,
         layout = c(3, 1), col = "steelblue", strip = strip.custom(bg="lightgrey"))

Flavanoids_Class_box <- bwplot(~ Flavanoids | Class, data = wine,
       layout = c(3, 1),
       par.settings = list(
         box.umbrella=list(col= "black"),
         box.dot=list(col= "black"),
         box.rectangle = list(col= "black", fill = "steelblue")),
         strip = strip.custom(bg="lightgrey"))

grid.arrange(Flavanoids_Class_hist, Flavanoids_Class_box, ncol=2)

OD280_OD315_Class_hist <- histogram(~ OD280_OD315 | Class, data = wine,
         layout = c(3, 1), col = "steelblue", strip = strip.custom(bg="lightgrey"))

OD280_OD315_Class_box <- bwplot(~ OD280_OD315 | Class, data = wine,
       layout = c(3, 1),
       par.settings = list(
         box.umbrella=list(col= "black"),
         box.dot=list(col= "black"),
         box.rectangle = list(col= "black", fill = "steelblue")),
         strip = strip.custom(bg="lightgrey"))

grid.arrange(OD280_OD315_Class_hist, OD280_OD315_Class_box, ncol=2)

# Create tree model
fancyRpartPlot(rpart(Class ~ ., data = wine), sub = "")

############################################################
# Model-Based EDA
############################################################

# Create PCA model
wine$Class <- as.numeric(wine$Class) # must change Class to numeric to model
model.pca <- prcomp(wine, scale = T) # prcomp is preferred to princomp for accuracy
summary(model.pca)
par(mfrow=c(1,2))
screeplot(model.pca, type = c("lines"), main = "PCA Model", sub = "Number of Components") #
4 components explain most of variability in the data
biplot(model.pca, xlabs = wine[, "Class"], xlim=c(-0.20, 0.20))

# Reset plot display
par(mfrow=c(1,1))
```

```
# Change Class back to factor for LDA model
wine$Class = as.factor(wine$Class)

# Create LDA model
model.lda <- lda(Class ~ ., data = wine)
plot(model.lda, main = "LDA Model", cex = 0.90)

# Create Random Forest model
set.seed(123)
model.RF <- randomForest(Class~., data = wine, mtry=13, ntree =25)
importance(model.RF)
varImpPlot(model.RF, main = "Random Forest Model: \n Variable Importance")

#######################################################
# Build Models
#######################################################

# (1) Random Forest

# Create Random Forest model
levels(wine$Class) <- c("Class_1", "Class_2", "Class_3")

ptm <- proc.time() # Start the clock!
control.rf <- trainControl(method = "repeatedcv",
                           number = 10, repeats = 3,
                           classProbs = T, savePred = T, verboseIter = T)

set.seed(123)
model.rf <- train(x = wine[, -1], y = wine[, 1], data = wine, method="rf", trControl = control.rf)
proc.time() - ptm # Stop the clock

print(model.rf)

model.rf$finalModel

model.rf.pred <- predict(model.rf, newdata = wine[, -1])
model.rf.c.mat <- confusionMatrix(model.rf.pred, wine[, 1])
names(model.rf.c.mat) # "positive" "table"    "overall"  "byClass"  "mode"     "dots"
model.rf.c.mat$table

model.rf.c.mat$overall

varImpPlot(model.rf$finalModel, main = "Random Forest Model: \n Variable Importance")

# ----------------------------------------------------------------------#
# (2) a Support Vector Machine
ptm <- proc.time() # Start the clock!
svm.control <- trainControl(method = "repeatedcv",
                            number = 10, repeats = 3,
                            classProbs = T, savePred = T, verboseIter = T)
```

```
set.seed(123)
model.svm <- train(Class ~ ., data = wine, method = "svmRadial", # or svmRadialWeights?
                   trControl = svm.control)
proc.time() - ptm # Stop the clock

print(model.svm)

model.svm$finalModel

model.svm.pred <- predict(model.svm, newdata = wine[, -1])
model.svm.c.mat <- confusionMatrix(model.svm.pred, wine[, 1])
model.svm.c.mat$table

model.svm.c.mat$overall

plot(varImp(model.svm),layout = c(1, 3), strip = strip.custom(bg="lightgrey"))

# -----------------------------------------------------------------------#
# (3) a neural network model
ptm <- proc.time() # Start the clock!
nnet.control <- trainControl(method = "repeatedcv",
                        number = 10, repeats = 3,
                        classProbs = T, savePred = T, verboseIter = T)
set.seed(123)
model.nnet <- train(Class ~ ., data = wine, method = "nnet",
                trControl = nnet.control)
proc.time() - ptm # Stop the clock

print(model.nnet)

model.nnet$finalModel

model.nnet.pred <- predict(model.nnet, newdata = wine[, -1])
model.nnet.c.mat <- confusionMatrix(model.nnet.pred, wine[,1])
model.nnet.c.mat$table

model.nnet.c.mat$overall

plot(model.nnet)

#########################################################
# END
#########################################################
```