Assignment 4

Predictive Modeling in Multiclass Classification

Christina Macholan

Data Overview

In this exercise, we explore whether it is possible to predict the Italian wine cultivar (plant variety) used in the production of a wine given 13 measurements of its chemical properties.

Using statsitical graphics and exploratory data analysis, we first examine the quality of the data provided and determine which variables are most likely to be useful for prediction. Then, we compare the prediction performance for three modeling algorithms – random forests, support vector machines, and neural nets.

The data set provided by Univeristy of California Irvine (http://archive.ics.uci.edu/ml/datasets/Wine) contains 13 measurements from a chemical analysis of 177 wines grown in the same region in Italy. An additional three-level classification variable is included in the dataset to distinguish which of three cultivars were used to make the wine. Table 1 describes the 14 variables. Note that a detailed data dictionary was not provided with the data set, therefore we cannot be certain what units of measurement were used for each of the numeric variables.

Table 1: Description of data

Variable Name	Data Type	Description
Cultivar	Factor with 3 levels	The class level assigned to distinguish the three different wine cultivars (1, 2, or 3). This is the response variable.
Alcohol	Numeric	The alcohol content of the wine (alcohol by volume).
Alcalinity	Numeric	The alcalinity of ash in the wine (a measurement of pH).
Ash	Numeric	The amount of ash in the wine.
ColorIntensity	Numeric	The color intensity of the wine (lightness or darkness) measured via spectrometry.
Flavanoids	Numeric	The level of flavanoids (a type of phenol) in the wine.
Hue	Numeric	The hue of the wine measured via spectrometry.
Magnesium	Numeric	The magnesium level in the wine.
MalicAcid	Numeric	The malic acid level of the wine (a measurement of pH).
Nonflavanoid	Numeric	The level of nonflavanoid phenols in the wine.
OD280/OD315	Numeric	The OD280/OD315 (protein) content of diluted wines.
Phenols	Numeric	The level of total phenols in the wine.
Proanthocyanins	Numeric	The level of proanthocyanins (a type of phenol) in the wine.
Proline	Numeric	The proline level of the wine. Proline is an amino acid.

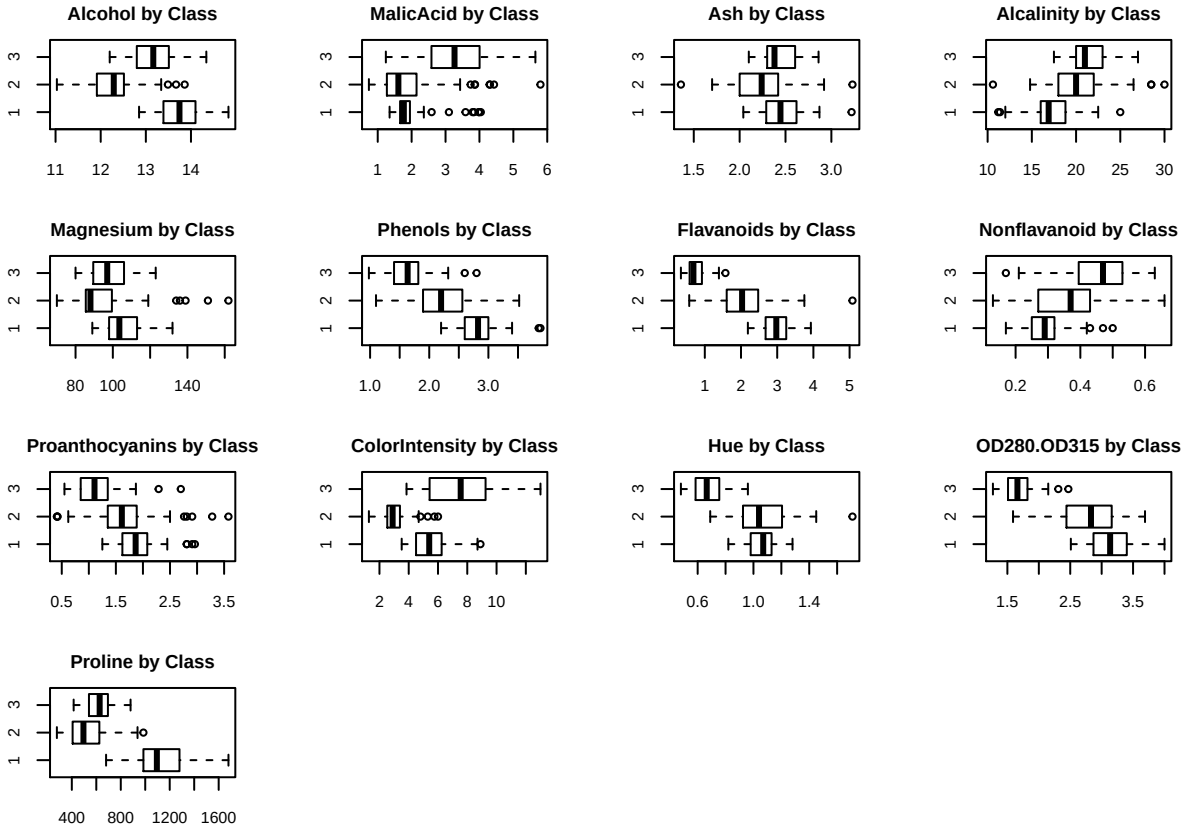
Data Quality Check

To check the data quality, we first reviewed summary statistical measurements for each variable. None of the variables have missing values, and the range of values for each metric seems reasonable (no unexpected negative or zero-value measurements).

To check for outliers, we created boxplots of each variable broken out by cultivar class as shown in Figure 1. Each of the variables has a few outliers that fall outside of 1.5 IQR of the lower and upper quartiles for one

or more of the cultivar classes (shown as points in the plots). However, all values still appear to be within a reasonable enough range to assume that none of the outliers are due to systemic measurement errors, human error, or other data quality issues.

Figure 1: Boxplots of each variable by Cultivar class



Exploratory Data Analysis

To begin the exploratory data analysis, we wanted to determine which of the 13 chemical analysis variables are most separable by class. Using the `lattice` package from R, we created side-by-side histograms for each variable broken out by cultivar class.

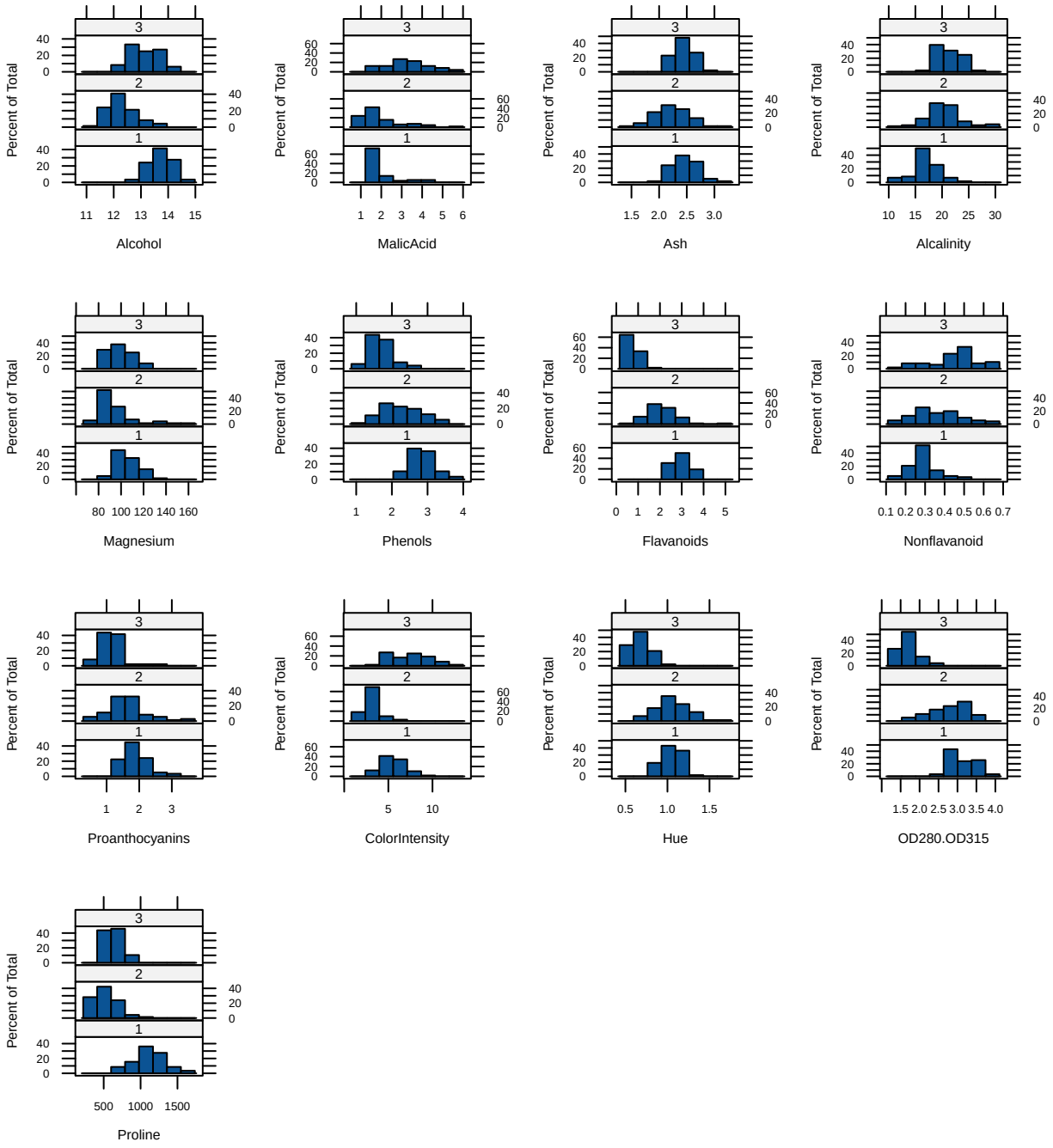
Reviewing the histograms in Figure 2, we can see that the following variables are most likely to be useful in partitioning the data by class:

- **Flavanoids:** Cultivars 1 and 3 have little overlap on Flavanoid measurements, with Cultivar 1's values skewed higher and Cultivar 3's values skewed lower.
- **OD280/OD315:** Again, Cultivars 1 and 3 have little overlap, with Cultivar 1's values skewed higher and Cultivar 3's values skewed lower.
- **Proline:** Cultivar 1 has little overlap with Cultivars 2 and 3. Cultivar 1's values skew higher.
- **Alcohol:** Cultivars 1 and 2 have little overlap on Alcohol content, with Cultivar 1's values skewed higher and Cultivar 2's values skewed lower.

- **Hue:** Cultivar 3 has little overlap with Cultivars 1 and 2. Cultivar 3's values skew lower.
- **Color Intensity:** Cultivar 3 tends to have higher values of color intensity than Cultivars 1 or 2.
- **Phenols:** Again, Cultivars 1 and 3 have little overlap, with Cultivar 1's values skewed higher and Cultivar 3's values skewed lower.

We would expect to see some of these highly separable variables represented in any predictive models that we test.

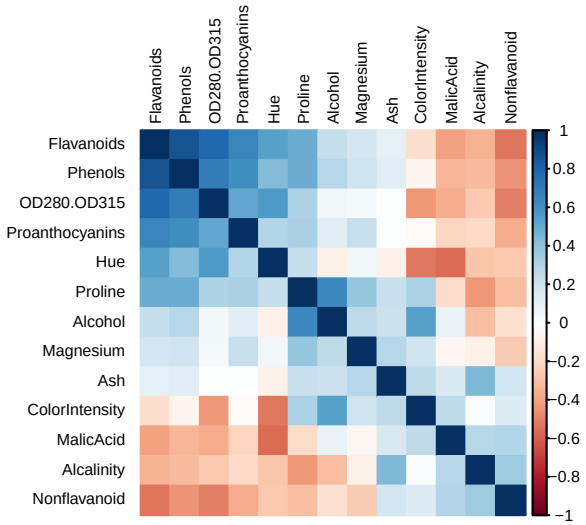
Figure 2: Histograms of each variable by Cultivar class



Because several of the chemical measurements assess similar traits of the wine, we must also consider the role of variable reduction to eliminate information redundancy when building models. A correlation plot of the 13 chemical analysis variables (Figure 3) shows especially strong positive or negative correlations between the following variables:

- **Phenol-related variables**: Phenols, Flavanoids, Nonflavanoid Phenols, Proanthocyannins
- **Color-related variables & Malic Acid**: Hue, Color Intensity, & Malic Acid
- **Proline & Alcohol**

Figure 3: Correlation plot of numeric variables



The shared characteristics of these variables may be captured through a variable reduction technique like Principal Component Analysis, which we explore in the next section.

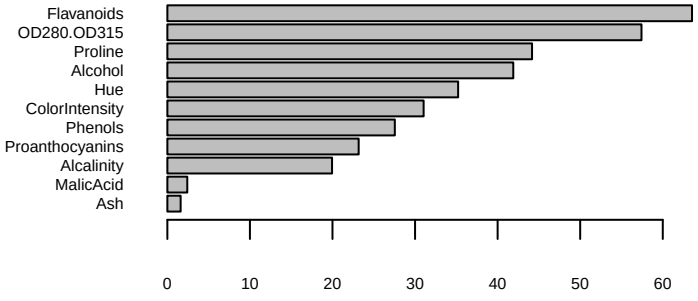
Model-based EDA

Tree-based Classification Model (all original variables)

To start the model-based exploratory data analysis, we first created a simple tree model. This model helps validate which variables are likely to be the most important for inclusion in other model algorithms.

In alignment with our earlier visual analysis of the histograms, we can see in Figure 4 a ranked list of the 11 variables that are most "important" for explaining variability in the data.

Figure 4: Variable importance plot for tree model

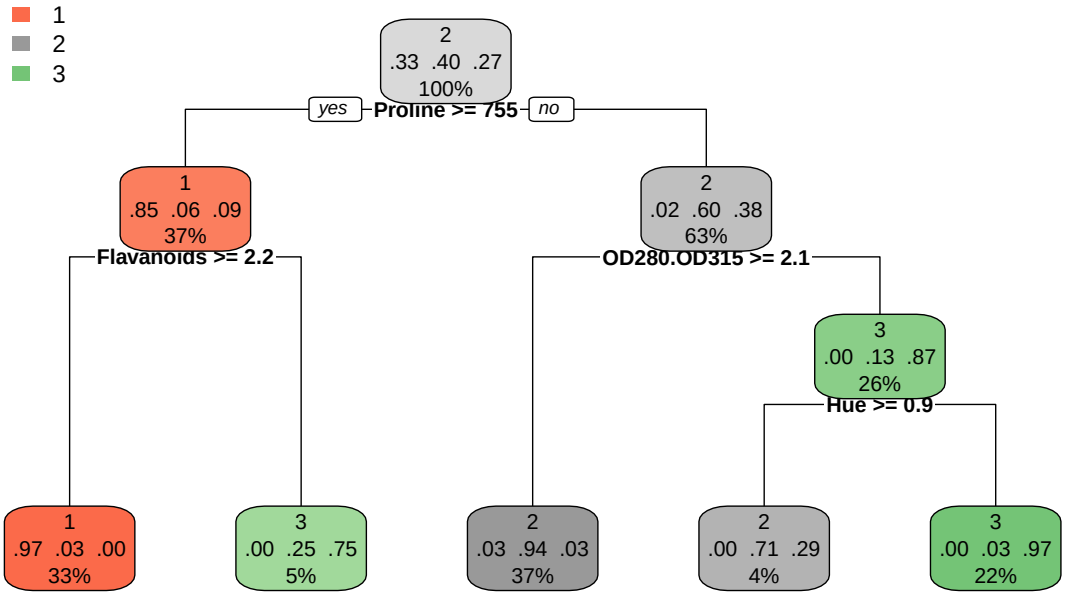


Of these 11 attributes, the model algorithm chooses only four – Flavanoids, Hue, OD280/OD315, and Proline – to define the decision rules at each branch of the tree. Figure 6 displays the details decision rules used at each branch and the purity at each node. The model overall has an accuracy of 93.79%.

From this model we can make a few generalizations about the three cultivars:

- Wines with *lower* proline and *higher* OD280/OD315 are likely to be from Cultivar 2.
- Wines with *lower* proline and *lower* OD280/OD315 are likely to be from Cultivar 1 (unless they have a darker hue, in which case they are more likely to be from Cultivar 2).
- Wines with *higher* proline are more likely to come from Cultivar 3, unless they are lower in Flavanoids, in which case they are more likely from Cultivar 1.

Figure 6: Classification tree

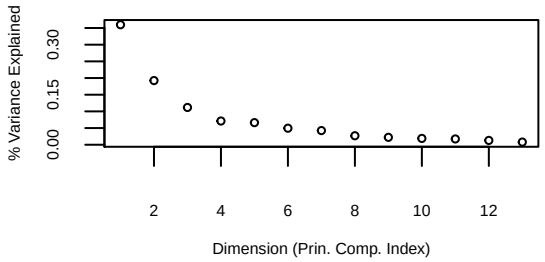


Principal Component Analysis (PCA)

Because we identified strong relationships between some variables during EDA, we continue the model-based exploration with Principal Component Analysis (PCA) for dimension reduction. Before conducting PCA we first standardized each of the numeric variables to have a mean of zero and standard deviation of one.

As displayed in Figure 7, the results of PCA show that the first six components explain approximately 85% of the variation in the data set, supporting the usefulness of variable reduction for this classification problem.

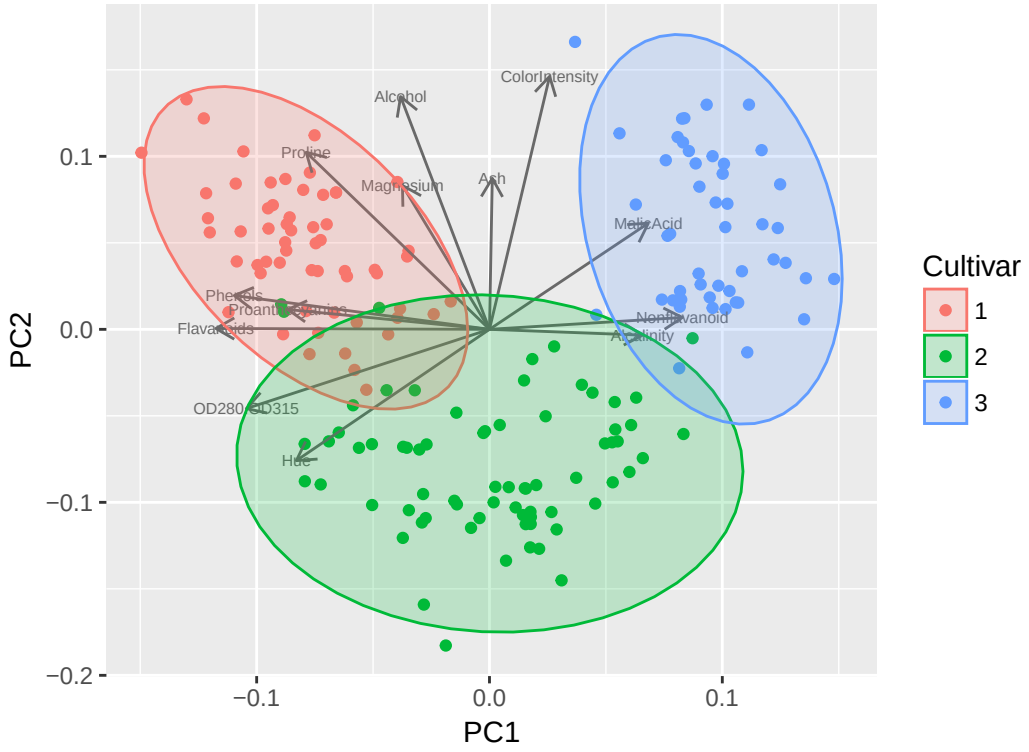
Figure 7: Percent variance explained by PCA components (scree plot)



The loadings for the first principal component are particularly high for the phenol-related attributes (Phenols, Flavanoids, Nonflavanoid Phenols, Proanthocyannins) and OD280/OD315. The loadings for the second principal component are particularly high for the Alcohol, Proline, and Color Intensity variables.

A scatter plot of the first two principal components color-coded for actual cultivar grouping (Figure 8) shows that these first two components alone are very good at separating Cultivar 3 from the other two groups, and reasonably good at separating Cultivars 1 and 2.

Figure 8: PCA Component 1 vs. 2 by Cultivar



K-means Clustering

Using k-means clustering, we can get a better sense of whether the original 13 variables or the reduced set of 6 variables from PCA is better at separating the data. This helps us decide whether to use the original variables or the reduced variables in the subsequent model building process.

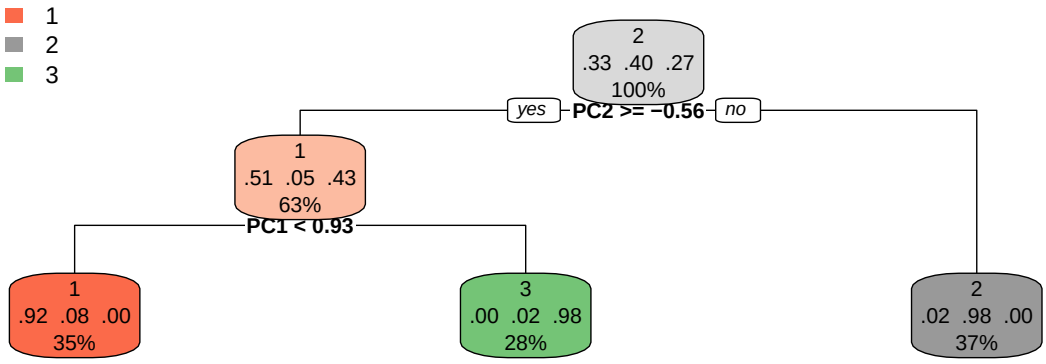
For the model with all 13 original variables, three-means clustering results in a 44.59% reduction in sums of squares, whereas the model with six variables from PCA results in a 52.31% reduction in sums of squares. The better performance of the second model suggests that the principal components, rather than the original variables, should be used when building models.

Tree-based Model (six PCA components)

Finally, building a simple tree model with the first six principal components from PCA provides additional validation for the dimensional reduction approach. The new tree-based classification model predicts with 96.05% accuracy (versus 93.79% for the model with all original variables), using only principal components 1 and 2 for the decision rules.

Figure 8 shows the decision rules used at each branch of the tree and the purity at each node. Only one decision rule is needed to distinguish most cases of Cultivar 2 from Cultivars 1 and 3. This rule is based on principal component 2, which we previously identified as having high loadings for the Proline, Alcohol, and Color Intensity variables. A second decision rule based on principal component 2, which we identified as having high loadings for phenol-related variables and OD280/OD315, is used to separate Cultivars 1 and 3.

Figure 8: Classification tree



Model Creation

Now that we have a good understanding of the relationships between the predictors and the cultivar class, we can move on to building predictive models. As stated before, for this exercise we will focus on three modeling algorithms – random forests, support vector machines, and neural networks.

In an ideal scenario, we would have a large enough dataset to create separate subsets of data for model training and model validation. However, with only 177 records provided, we will rely on in-sample model statistics to compare the performance of the algorithms and, where possible, we will use bootstrapping and k-fold cross-validation to estimate the out-of-sample classification error.

Random Forests

We start by applying a set of random forest models to the dataset using the `randomForest` R package. Random forests are an ensemble learning method for decision trees. For classification prediction, the algorithm selects the most common class assigned to each record across the n trees produced (also known as "majority vote" selection). We tested the algorithm with the parameters applied as shown in Table 2.

Table 2: Random Forest Model Parameters

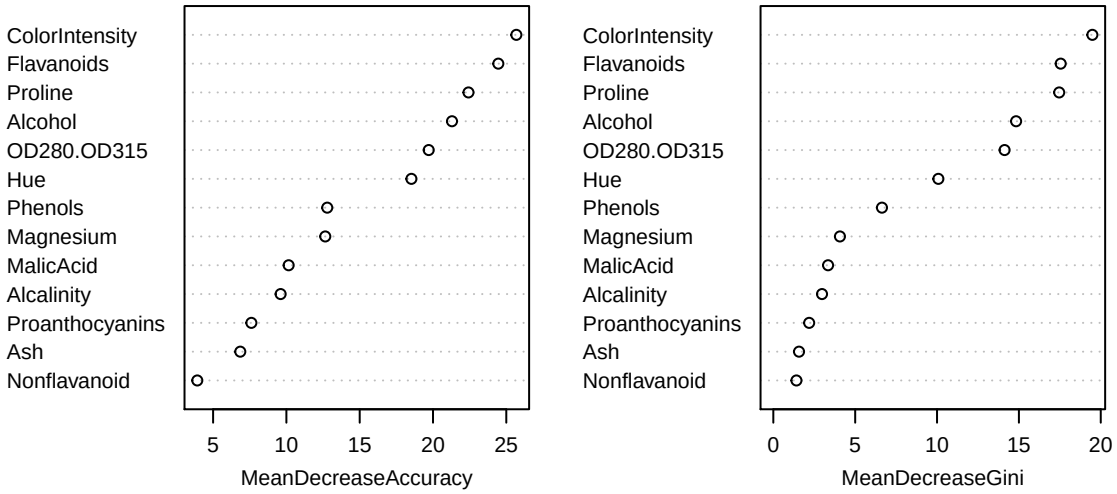
Model	# of trees	# of variables tried at splits	OOB Class. Error
Random Forest 1 (default)	500	3	2.26%
Random Forest 2	100	3	1.69%
Random Forest 3	500	6	2.26%
Random Forest 4	100	6	2.26%
Random Forest 5	500	13	2.82%
Random Forest 6	100	13	3.95%

Random Forest models 5 and 6 both represent the boostrap aggregated ("bagging") model, where B decision trees are grown using B distinct bootstrapped training sets with all m predictors tried at each split of the tree. These trees are averaged to reduce the variance of the predictions. The rest of the trees (models 1, 2, 3, and 4) sample a reduced set of variables at each split, which reduces the correlation between the trees and makes the resulting model more reliable. For all 6 models, we used the scaled version of the 13 variables since this resulted in better performance than using the reduced six-component PCA model.

We used the out-of-bag (OOB) classification error as an estimate of the out-of-sample error rate to compare the models' performance. We can see that the random forest models with reduced variables (3 or 6 using 500 or 100 decision trees) result in the lowest error rates.

Figure 9 shows the variable importance plot for model 1. It is noteworthy that the order of variable importance for the random forest model with fewer variables tried had each split has shifted compared to our previous single tree model. Here, Color Intensity, Flavanoids, Proline, Alcohol, and OD280 / OD315 are the five most important variables.

Figure 9: Variable importance plot for Random Forest with *ntree* = 500 and *mtry* = 3



For in-sample prediction, model 2 has a 98.31% accuracy rate (compared to 93.79% for the original tree model and 96.05% for the PCA tree model). The metrics in Table 3 show that classes 1 and 2 have no misclassified responses (sensitivity = 1). Class 2, however, has three misclassified responses – one from class 1 and two from class 3 (sensitivity = 95.77%).

Table 3: Confusion Matrix metrics for Random Forest Model 2

Metric	Class 1	Class 2	Class 3
Sensitivity	1.0000	0.9577	1.0000
Specificity	0.9916	1.0000	0.9845
Precision	0.9831	1.0000	0.9600
Recall	1.0000	0.9577	1.0000

Support Vector Machine

Next, we use the support vector machine (SVM) algorithm from the `e1071` R package to create a model that employs a non-linear decision boundry to separate the three cultivar classes. We tested an SVM model with default settings against an SVM model with the "best" tuning parameters selected via ten-fold cross-validation.

We also included a model using the PCA components instead of the 13 scaled variables (primarily for the purposes of visualizing the decision boundaries). For all three models, we used the "radial" kernel.

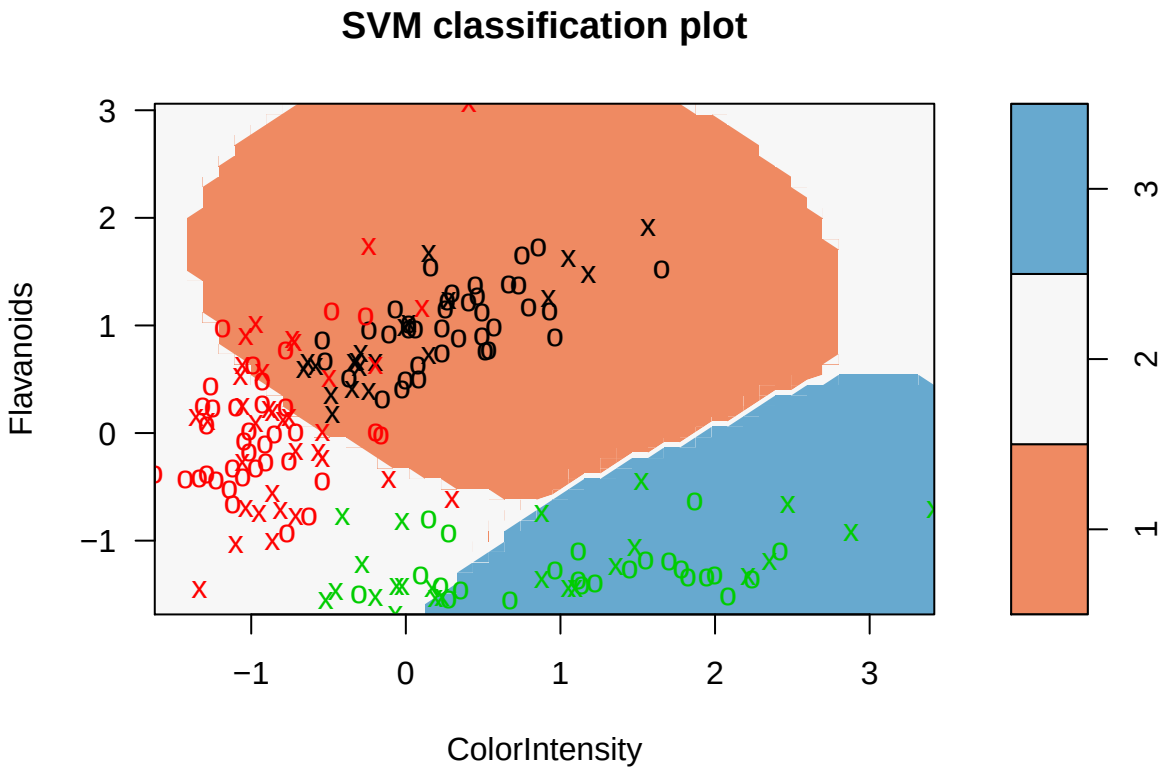
Table 4: SVM Model Parameters

Model	Kernel	Gamma	Cost	Support Vectors	CV Error
SVM 1	radial	0.0769	1	69	N/A
SVM 2	radial	0.01	1	80	1.66%
SVM 3 (using PCA variables)	radial	0.01	1	123	1.66%

All three models tested were able to predict within sample with 100% accuracy. The main concern with the SVM models is that they may have overfit the model to the data, so cross-validation is important for this approach. The cross-validation error rates for SVM models 1 and 2 are 1.66%, which means these may perform well out-of-sample.

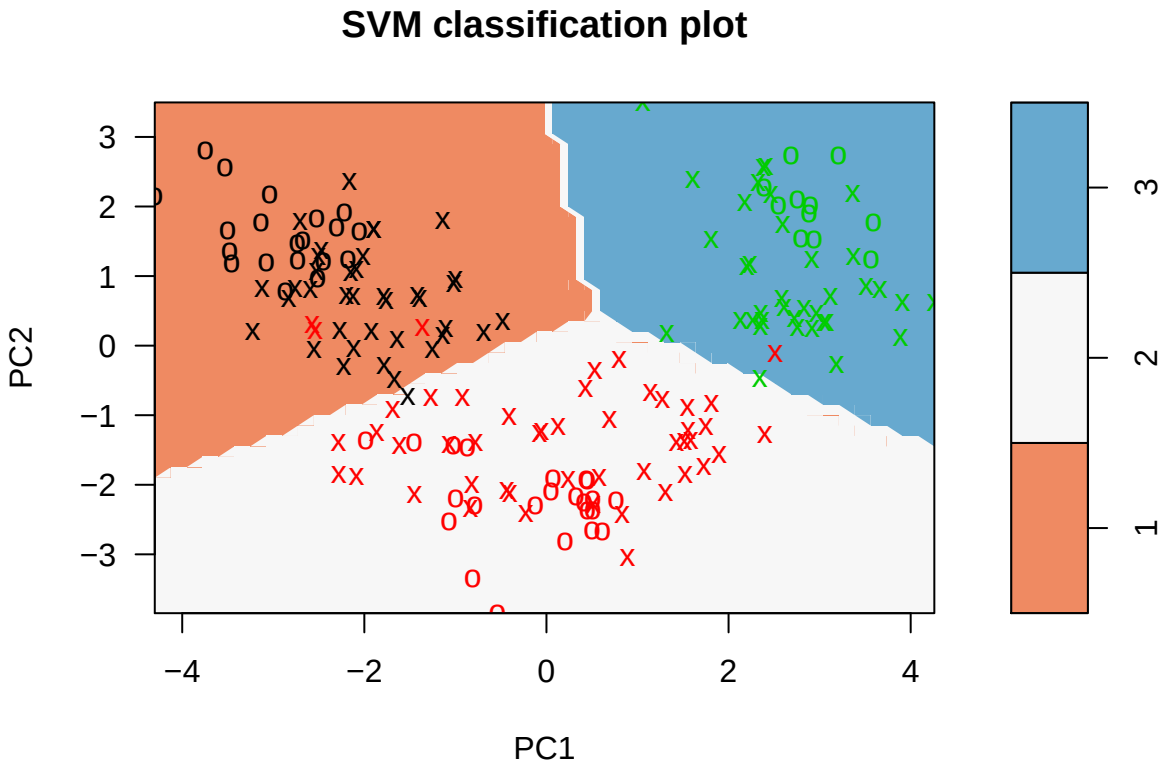
Figure 10 shows the decision boundry between the Flavanoids and Color Intensity variables SVM model 2. These two variables alone do not fully separate the classes, however the plot demonstrates the non-linear shape of the decision boundary well.

Figure 10: SVM decision boundary between the Flavanoids and Color Intensity variables



A plot of the principal components version of the SVM model (model 3) in Figure 11 clearly shows the orthogonal boundaries between components 1 and 2. This plot nicely demonstrates how PCA can be useful in reducing the dimensionality of our model to consolidate more information into our data visualizations.

Figure 11: SVM decision boundary between princiapl components 1 and 2



For in-sample prediction, all three SVM models had a 100% accuracy rate (compared to 98.31% for the best Random Forest model). The metrics in Table 5 show all three classes have no misclassified responses (sensitivity = 1).

Table 7: Confusion Matrix metrics for SVM Model 2

Metric	Class 1	Class 2	Class 3
Sensitivity	1.0000	1.0000	1.0000
Specificity	1.0000	1.0000	1.0000
Precision	1.0000	1.0000	1.0000
Recall	1.0000	1.0000	1.0000

Neural Network Model

The last model we prepared used the `neuralnet` R package. Like SVMs, this modeling approach is capable of modeling nonlinear relationships between the predictors and the response variable. For both models, we used the default "rprop+"" resilient backpropogation setting and used the original 13 variable dataset, though the reduced six-variable principal component model was tested and performed equally as well in-sample. For the first neural network we set one hidden layer and for the second we set two.

Table 8: Neural Network Model Parameters

Model	# of layers	In-Sample Error
Neural Network 1	1	131.509
Neural Network 2	2	131.509

Figure 12 shows the result of the single-layer neural network algorithm. Weights for the synapses from the variables to the first layer range from -3.17 (Proline) to 2.55 (Alcalinity).

Figure 12: Single-Layer Neural Net Model

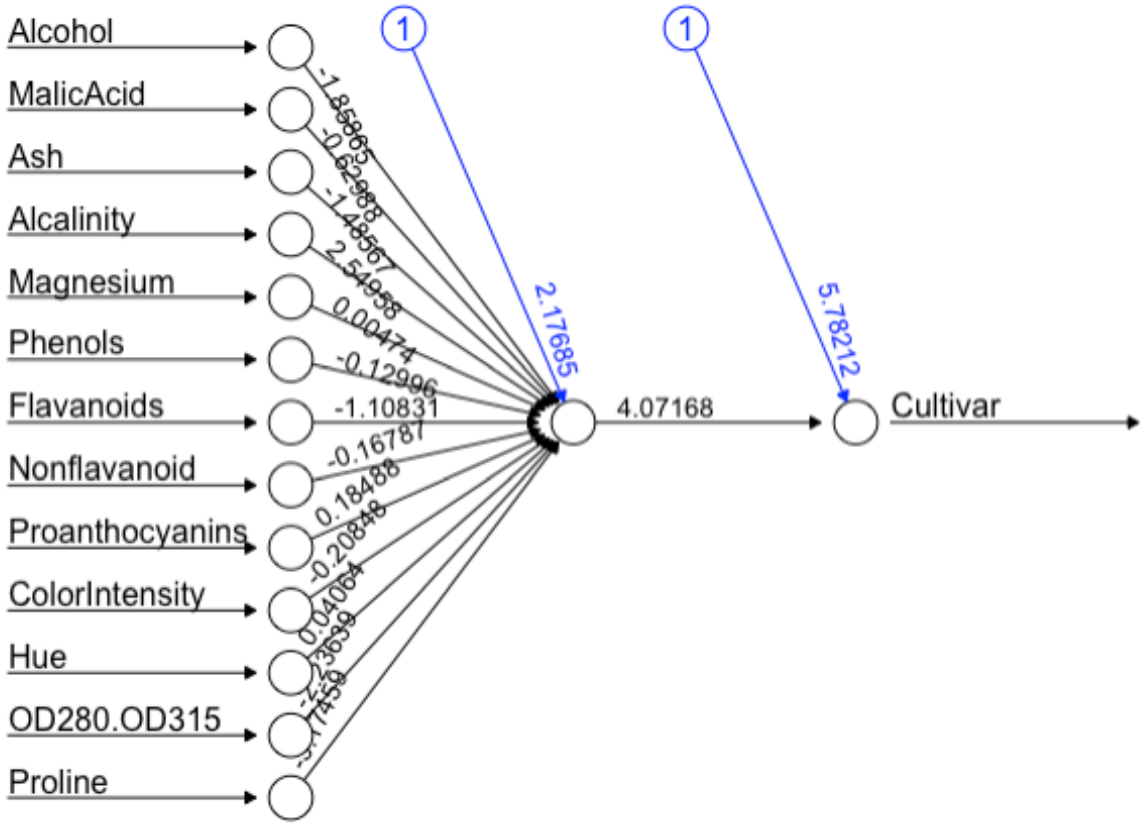


Figure 1: Single-Layer Neural Net Model

Figure 13 shows the result of the two-layer neural network algorithm. Weights for the synapses from the variables to the first layer range from -0.78 (Proline) to 0.34 (Alcanicity) and from -2.10 (Proline) to 2.50 (Alcanicity) for the second layer.

For in-sample prediction, both neural network models achieved a 100% accuracy rate (compared to 98.31% for the best Random Forest model). The metrics in Table 9 show all three classes have no misclassified responses (sensitivity = 1). For the sake of simplicity, the first model would be preferred.

Metric	Class 1	Class 2	Class 3
Sensitivity	1.0000	1.0000	1.0000
Specificity	1.0000	1.0000	1.0000
Precision	1.0000	1.0000	1.0000
Recall	1.0000	1.0000	1.0000

The major concern with these neural network models is that, unlike the previous models, they have not been validated through k-fold cross-validation or bagging. It is likely that the neural nets are overfit to the training data. Fitting the models to an out-of-sample data set or designing an algorithm to run cross-validation would help validate which of the above models (or variations with alternate tuning parameters) would actually be

best for prediction.

Figure 13: Two-Layer Neural Net Model

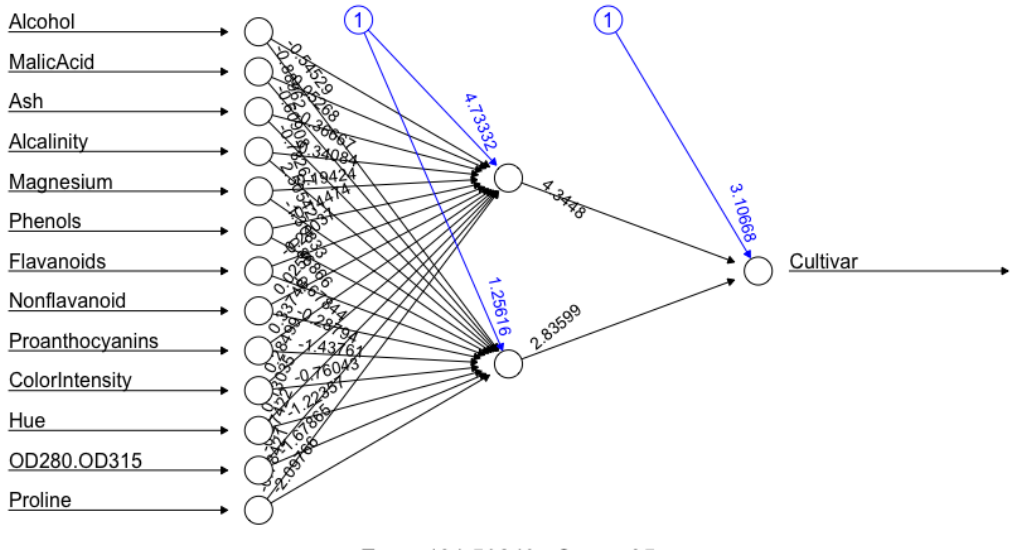


Figure 2: Two-Layer Neural Net Model

Model Comparison & Conclusions

Our initial exploratory analysis revealed that the wine data set provided is complete and reliable, with no missing values and a limited number of outliers (none of which warranted removal at this stage of the analysis process). While several variables (particularly Flavanoids, OD280/OD315, Proline, Alcohol, Hue, Color Intensity, and Phenols) showed promise for explaining the variability between and within the three cultivar groups, a Principal Component Analysis suggests that the 13 original variables can be reduced to six or fewer variables while retaining or improving model performance.

To choose viable prediction models ran three algorithms – random forests, support vector machines, and neural networks – with various tuning parameters applied in order to pick the best version of each model type. "Best" in the case of random forests and SVMs was considered the model that had the lowest k-fold cross-validation or bagging classification error rate. In the case of neural networks, we chose the simpler single-layer model over the two-layer model in lieu of having an algorithmic approach to cross-validation. Each of the three models produced excellent in-sample accuracy for the wine cultivar prediction (random forests: 98.31%; support vector machines: 100%; neural networks: 100%).

While Random Forests misclassified a few cases within the dataset, it does have the advantage of being more explainable than the other two models. The concept of combining the "majority vote" for each record's class across several hundred decision trees could be explained in a straigh-forward manner to model stakeholders. On top of that, the variable importance plot aides in explaining the patterns in the suite of classification trees (e.g. which variables end up being the "most important" in building the trees).

On the other hand, SVMs and neural networks are much more flexible, and in the in-sample case are able to capture every record's classification correctly. With proper tuning, these algorithms may produce more accurate production at the loss of explainability.

Appendix

R Code for Models

1. Classification Tree Model (all 13 variables)

```
# load libraries
library(rpart)
library(rpart.plot)

treefit <- rpart(Cultivar ~ ., method = "class", data = wine)
```

2. PCA

```
wine.norm <- wine
wine.norm[,c(2:14)] <- scale(wine.norm[,c(2:14)])

pcafit <- prcomp(wine.norm[,c(2:14)])
```

3. K-Means Models

```
comp <- data.frame(pcafit$x[,1:6])
kmeansfit1 <- kmeans(wine.norm[,2:14], 3)
kmeansfit2 <- kmeans(comp[,1:6], 3)
```

4. Classification Tree Model (principal components)

```
comp2 <- cbind(Cultivar = as.factor(wine$Cultivar), comp[,1:6])
treefit2 <- rpart(Cultivar ~ ., method = "class", data = comp2)
```

5. Random Forest & Bagging Models

```
# load library
library(randomForest)

# set seed for reproducible results
set.seed(1234)

# try various tuning parameters for the Random Forest Model
model.rf1 <- randomForest(Cultivar~., wine.norm, ntree=100, importance = TRUE)
model.rf2 <- randomForest(Cultivar~., wine.norm, importance = TRUE)
model.rf3 <- randomForest(Cultivar~., wine.norm, ntree=500, mtry = 6, importance = TRUE)
model.rf4 <- randomForest(Cultivar~., wine.norm, ntree=100, mtry = 6, importance = TRUE)
model.rf5 <- randomForest(Cultivar~., wine.norm, ntree=500, mtry = 13, importance = TRUE)
model.rf6 <- randomForest(Cultivar~., wine.norm, ntree=100, mtry = 13, importance = TRUE)

model.rf7 <- randomForest(Cultivar~., comp2, importance = TRUE)
```

```
model.rf8 <- randomForest(Cultivar~., comp2, ntree = 100, mtry = 6, importance = TRUE)
model.rf9 <- randomForest(Cultivar~., comp2, ntree = 500, mtry = 3, importance = TRUE)
model.rf10 <- randomForest(Cultivar~., comp2, ntree = 100, mtry = 3, importance = TRUE)

# review model statistics and pick the one that is best
# (default parameters perform the best)
model.rf1
model.rf2
model.rf3
model.rf4
model.rf5
model.rf6
model.rf7
model.rf8
model.rf9
model.rf10
```

6. SVM Models

```
# load library for SVM
library(e1071)

# set seed for reproducible results
set.seed(1234)

# try various tuning parameters for the SVM model
model.svm1 <- svm(Cultivar ~ ., data = wine.norm)
model.svm2 <- tune.svm(Cultivar ~ ., data = wine.norm, gamma = 10^(-6:1), cost = 10^(-2:2))
### model.svm3 <- svm(Cultivar ~ ., data = wine.norm, kernel = "linear")
### model.svm4 <- tune.svm(Cultivar ~ ., data = wine.norm, gamma = 10^(-6:1), cost = 10^(-2:2), kernel =
### model.svm5 <- svm(Cultivar ~ ., data = wine.norm, kernel = "polynomial")
### model.svm6 <- tune.svm(Cultivar ~ ., data = wine.norm, gamma = 10^(-6:1), cost = 10^(-2:2), kernel =
### model.svm7 <- svm(Cultivar ~ ., data = wine.norm, kernel = "sigmoid")
### model.svm8 <- tune.svm(Cultivar ~ ., data = wine.norm, gamma = 10^(-6:1), cost = 10^(-2:2), kernel =
model.svm9 <- tune.svm(Cultivar ~ ., data = comp2, gamma = 10^(-6:1), cost = 10^(-2:2))

# review model summary statistics
model.svm1
summary.model.svm1 <- summary(model.svm1)

model.svm2
summary.model.svm2 <- summary(model.svm2)

model.svm9
summary.model.svm9 <- summary(model.svm9)

# set the best model (gamme = 0.1; cost = 10; epsilon = 0.1)
model.svm2.best <- summary.model.svm2$best.model
model.svm2.best

model.svm9.best <- summary.model.svm9$best.model
model.svm9.best
```

7. Neural Net Models

```
# load libraries
library(neuralnet)
library(nnet)
library(caret)

# create neural net model

wine.norm.numeric <- wine.norm
wine.norm.numeric$Cultivar <- as.numeric(wine.norm.numeric$Cultivar)
model.nn1 <- neuralnet(Cultivar~Alcohol+MalicAcid+Ash+Alcalinity+Magnesium+Phenols+Flavanoids+Nonflavan
model.nn2 <- neuralnet(Cultivar~Alcohol+MalicAcid+Ash+Alcalinity+Magnesium+Phenols+Flavanoids+Nonflavan
```

R Code for Plots

0. Lattice Plot Theme Customization

```
# Set lattice plot theme
mytheme <- standard.theme("png", color=FALSE)
mytheme$fontsize$text = 6
mytheme$par.main.text$cex = 0.8
mytheme$plot.polygon$col <- "#0b5394"
mytheme$layout.heights$top.padding = 0.8
mytheme$layout.heights$main.key.padding = 0.3
```

1. Bivariate Boxplots

```
par(mfrow = c(4,4), oma = c(0,0,0,0), mar = c(2.5,2.5,2.5,2.5))
for (i in 2:ncol(wine)) {
    par(cex.main = 0.9, cex.axis = 0.8)
    boxplot(wine[,i] ~ Cultivar, data = wine, horizontal = TRUE)
    title(paste(colnames(wine)[i],"by Class"), line = 0.8)
}
```

2. Bivariate Histograms

```
# create individual histograms with lattice
alcoholplot <- histogram(~ Alcohol | factor(Cultivar), data = wine,
                         horizontal = TRUE, layout = c(1,3),
                         main = "",
                         auto.key=TRUE, par.settings=mytheme)
malicacidplot <- histogram(~ MalicAcid | factor(Cultivar), data = wine,
                         horizontal = TRUE, layout = c(1,3),
                         main = "",
                         auto.key=TRUE, par.settings=mytheme)
ashplot <- histogram(~ Ash | factor(Cultivar), data = wine,
                         horizontal = TRUE, layout = c(1,3),
                         main = "",
```

```
                        auto.key=TRUE, par.settings=mytheme)
alcalinityplot <- histogram(~ Alcalinity | factor(Cultivar), data = wine,
                        horizontal = TRUE, layout = c(1,3),
                        main = "",
                        auto.key=TRUE, par.settings=mytheme)
magnesiumplot <- histogram(~ Magnesium | factor(Cultivar), data = wine,
                        horizontal = TRUE, layout = c(1,3),
                        main = "",
                        auto.key=TRUE, par.settings=mytheme)
phenolsplot <- histogram(~ Phenols | factor(Cultivar), data = wine,
                        horizontal = TRUE, layout = c(1,3),
                        main = "",
                        auto.key=TRUE, par.settings=mytheme)
flavanoidsplot <- histogram(~ Flavanoids | factor(Cultivar), data = wine,
                        horizontal = TRUE, layout = c(1,3),
                        main = "",
                        auto.key=TRUE, par.settings=mytheme)
nonflavanoidplot <- histogram(~ Nonflavanoid | factor(Cultivar), data = wine,
                        horizontal = TRUE, layout = c(1,3),
                        main = "",
                        auto.key=TRUE, par.settings=mytheme)
proanthplot <- histogram(~ Proanthocyanins | factor(Cultivar), data = wine,
                        horizontal = TRUE, layout = c(1,3),
                        main = "",
                        auto.key=TRUE, par.settings=mytheme)
colorintensityplot <- histogram(~ ColorIntensity | factor(Cultivar), data = wine,
                        horizontal = TRUE, layout = c(1,3),
                        main = "",
                        auto.key=TRUE, par.settings=mytheme)
hueplot <- histogram(~ Hue | factor(Cultivar), data = wine,
                        horizontal = TRUE, layout = c(1,3),
                        main = "",
                        auto.key=TRUE, par.settings=mytheme)
od280plot <- histogram(~ OD280.OD315 | factor(Cultivar), data = wine,
                        horizontal = TRUE, layout = c(1,3),
                        main = "",
                        auto.key=TRUE, par.settings=mytheme)
prolineplot <- histogram(~ Proline | factor(Cultivar), data = wine,
                        horizontal = TRUE, layout = c(1,3),
                        main = "",
                        auto.key=TRUE, par.settings=mytheme)
# display lattice histograms in grid for better PDF viewing
grid.arrange(alcoholplot, malicacidplot, ashplot, alcalinityplot,
             magnesiumplot, phenolsplot, flavanoidsplot, nonflavanoidplot,
             proanthplot, colorintensityplot, hueplot, od280plot,
             prolineplot, nrow = 4, ncol = 4)
```

3. Correlation Plot

4. PCA Plot for Componenets 1 & 2

```
autoplot(pcafit, data = wine.norm, colour = 'Cultivar', loadings = TRUE,
         loadings.colour = '#696969', loadings.label = TRUE,
         loadings.label.size = 2, loadings.label.colour = '#696969',
         frame = TRUE, frame.type = 'norm')
```

5. Classification Tree Plot

```
rpart.plot(treefit, uniform=TRUE, main="", cex = 0.7)
```

6. Variable Importance Plot

```
barplot(treefit$variable.importance[order(treefit$variable.importance)],
        cex.names = 0.5, horiz = TRUE, cex.axis = 0.5, las=1)
```

7. SVM Plot

```
plot(model.svm2.best, data = wine.norm, Flavanoids~ColorIntensity, col = brewer.pal(3, "RdBu"))
```

8. Neural Network Plot

```
# plot the neural net
plot(model.nn1)
```