Predictive Modeling in Multiclass Classification

Eric Gero
Northwestern University

Predict 454

Introduction

For this assignment, the "Wine" data set as obtained from the UC Irvine Machine Learning Repository will be examined. The data set represents the chemical analysis of 178 observations of wine grown in the same region in Italy, but having been processed by three different cultivars (Lichman, 2013). A data quality check to learn about the data and an exploratory data analysis [EDA] to identify the relationships that exist within the data will be performed. Finally, three models will be fit to explore the capabilities of random forests, support vector machines, and neural networks.

Data Quality Check

The "Wine" data set contains 14 variables in total, with a categorical response variable suggesting that this is a classification problem. The response variable *Cultivar*, which indicates where the wine was processed, has possible values of 1, 2, or 3.

Table 1 lists all variables with their types, and provides a brief description what each variable is measuring and why it is important in wine.

Table 1 – Variable definitions

Variable	Type	Description
Cultivar	Categorical	Designates which cultivar the wine originated from. (Lichman, 2013).
Alcohol	Continuous	The alcohol content in the wine.
Malic_acid	Continuous	If there is not enough, the wine will taste "flat," and will be more susceptible to spoilage. If there is too much, the wine will taste "green," or "sour." (Calwineries, 2017)
Ash	Continuous	Ash being defined as the inorganic matter that remains after evaporation and incineration. (WineEducation, n.d.)
Alcalinity_of_ash	Continuous	Alkalinity of ash measures the basicity (alkalinity) of the ash obtained from a sample. This is accomplished by adding acid to the ash until the solution is neutralized. (ETS, 2017)
Magnesium	Continuous	Does not appear to contribute to the quality of wine, but is an important vitamin for the body and alcohol consumption depletes it. (Robertson, 2017)
Total_phenols	Continuous	The phenolic content in wine refers to the phenolic compounds—natural phenol and polyphenols, which include a large group of several hundred chemical compounds that affect the taste, color and mouthfeel of wine. (Wikipedia, 2017)
Flavanoids	Continuous	Flavonoids include the anthocyanins and tannins which contribute to the color and mouthfeel of the wine. (Wikipedia, 2017)
Nonflavanoid_phenols	Continuous	The non-flavonoids include the stilbenoids such as resveratrol and phenolic acids such as benzoic, caffeic and cinnamicacids. (Wikipedia, 2017)
Proanthocyanins	Continuous	Proanthocyanidins play an important role in wine; with the capability to bind salivary proteins, these condensed tannins strongly influence the perceived astringency of the wine. (UC Davis, 2017)
Color_intensity	Continuous	The intensity of color can be observed with the wine's opacity. Deeply opaque red wines have been noted for having more pigment and phenolics than more translucent red wines. Darker wines tend to be bolder. (Puckette, 2016)
Hue	Continuous	The color of wine is one of the most easily recognizable characteristics of wines. Color is also an element in wine tasting since heavy wines generally have a deeper color. (Wikipedia, 2017)
OD280/OD315	Continuous	A measurement of protein concentration in the wine. (Maujean, n.d.)
Proline	Continuous	An amino acid, its effect in wine is unclear.

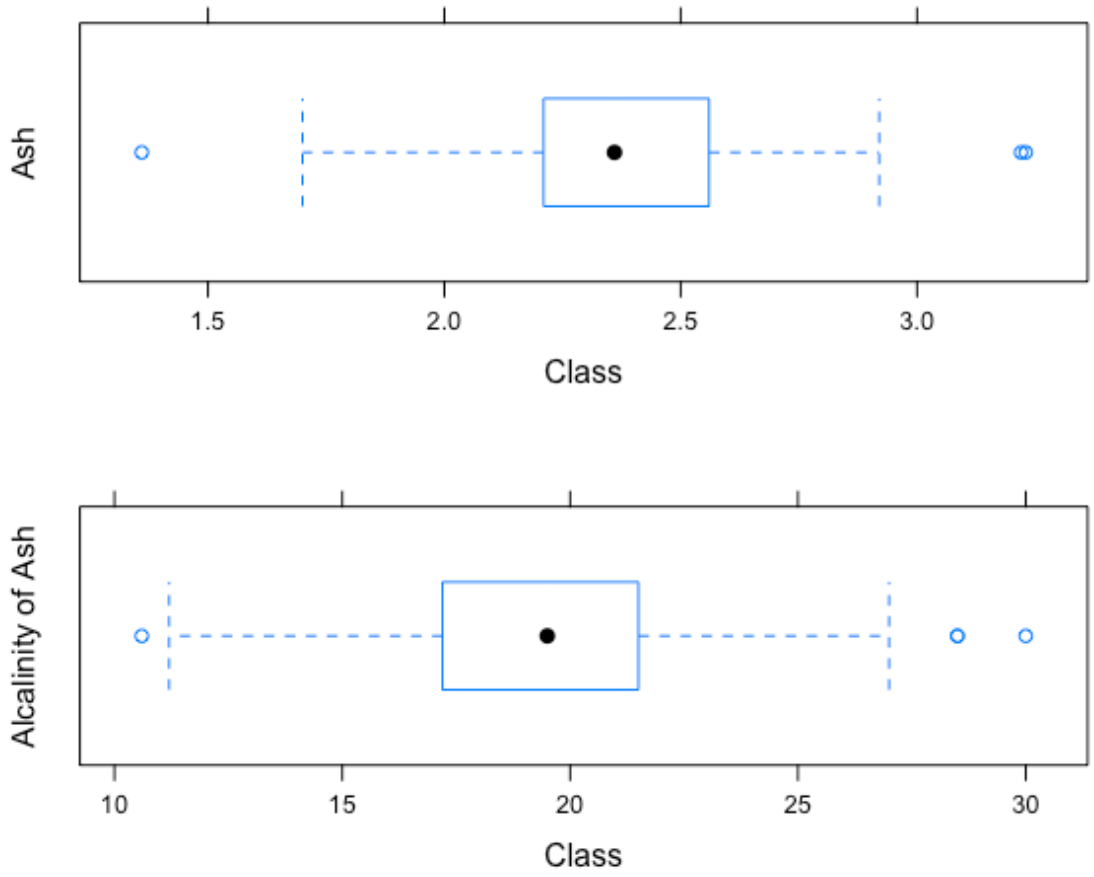
Table 2 provides summary statistics for the 13 predictors. The upper and lower limits, based on 1.5 * interquartile range [IQR], have also be provided to identify variables that may contain outliers. These variables are highlighted in red. While no method exists to definitively identify outliers, there are multiple tools that can provide some insight.

Table 2 – Variable summary information

Variable	Minimum	1st Quartile	Median	Mean	3rd Quartile	Maximum	Lower Limit	Upper Limit
Alcohol	11.030	12.360	13.050	13.000	13.680	14.830	10.380	15.660
Malic_acid	0.740	1.603	1.865	2.336	3.083	5.800	-0.617	5.303
Ash	1.360	2.210	2.360	2.367	2.558	3.230	1.688	3.080
Alcalinity_of_ash	10.600	17.200	19.500	19.490	21.500	30.000	10.750	27.950
Magnesium	70.000	88.000	98.000	99.740	107.000	162.000	59.500	135.500
Total_phenols	0.980	1.742	2.355	2.295	2.800	3.880	0.155	4.387
Flavanoids	0.340	1.205	2.135	2.029	2.875	5.080	-1.300	5.380
Nonflavanoid_phenols	0.130	0.270	0.340	0.362	0.438	0.660	0.019	0.689
Proanthocyanins	0.410	1.250	1.555	1.591	1.950	3.580	0.200	3.000
Color intensity	1.280	3.220	4.690	5.058	6.200	13.000	-1.250	10.670
Hue	0.480	0.783	0.965	0.957	1.120	1.710	0.276	1.626
OD280/OD315	1.270	1.938	2.780	2.612	3.170	4.000	0.090	5.018
Proline	278.000	500.500	673.500	746.900	985.000	1680.000	-226.250	1711.750

Figure 1 contains box plots for the *Ash* and *Alcalinity of Ash* variables. Note the data points on the outside of each of the whiskers. As identified in **Table 2**, these points identify possible outliers and warrant further investigation.

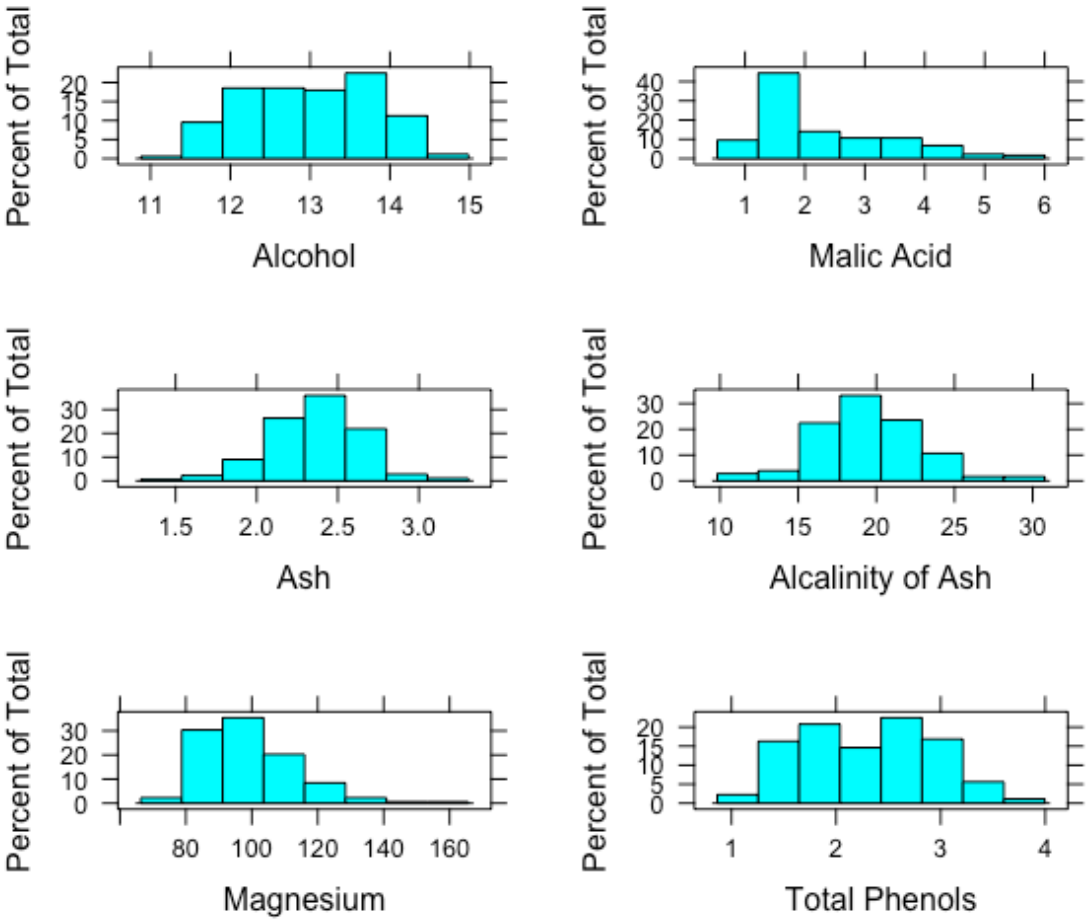
Figure 1 – Box plots for Ash and Alcalinity of Ash



During the data quality check, it is important to identify any observations that have missing values, values that fall outside of the valid range as determined by the business, or values that do not conform to the variable type. This may occur when a character is inserted for a numeric field to indicate that the value is missing or NULL. There are no missing values or anomalies present in the "Wine" data set. It is assumed that all values are within acceptable ranges since no information was provided in the data dictionary.

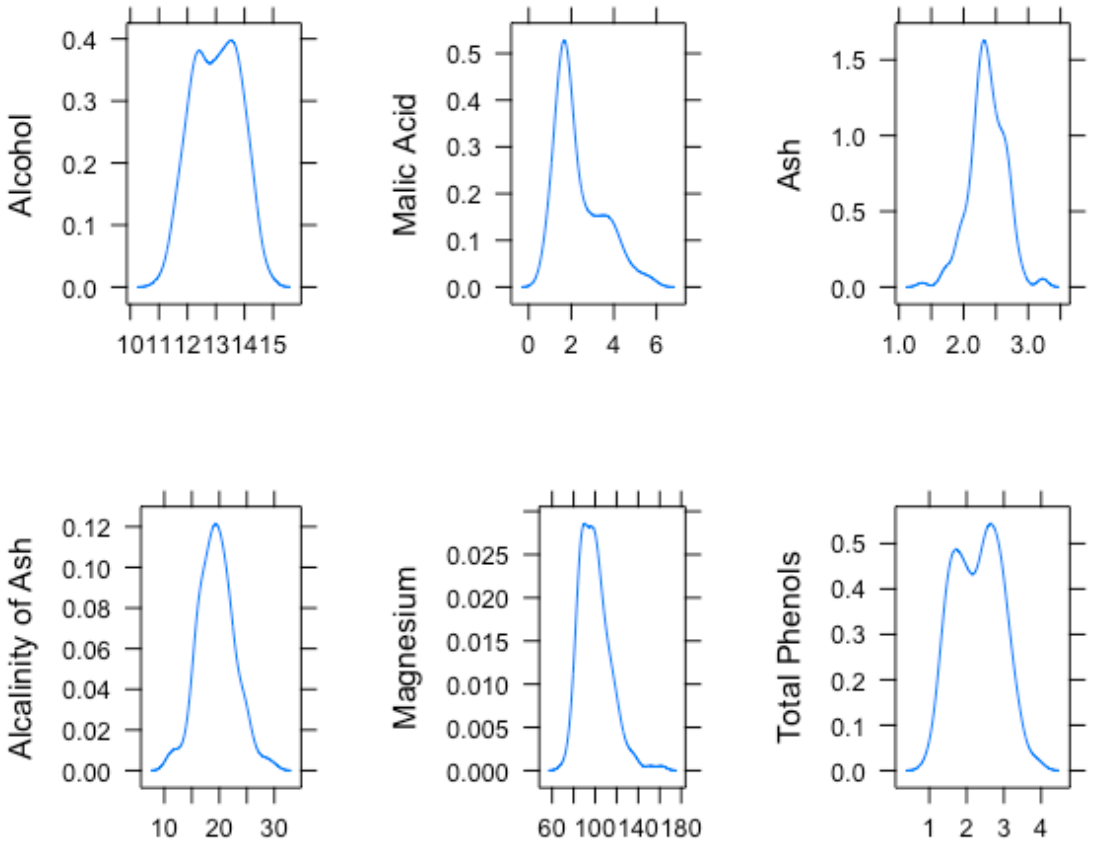
It is also important to review the shape of the distributions for each variable. **Figure 2** provides histograms for the first six variables. Several variables do not display a normal distribution. Specifically, *Alcohol* and *Malic Acid*, while *Total Phenols* appears to be bimodal.

Figure 2 – Histograms of the first six variables



Density plots provide another means of viewing a variable's distribution. As observed in **Figure 3,** the density plots tell a different story than the histograms. This is not uncommon since the histograms can vary based on the number of bins selected. Note how the variables *Alcohol*, *Malic Acid*, and *Total Phenols* now appear to have bimodal distributions. Similar patterns are observed for many of the remaining variables.

Figure 3 – Density plots of first six variables



Exploratory Data Analysis

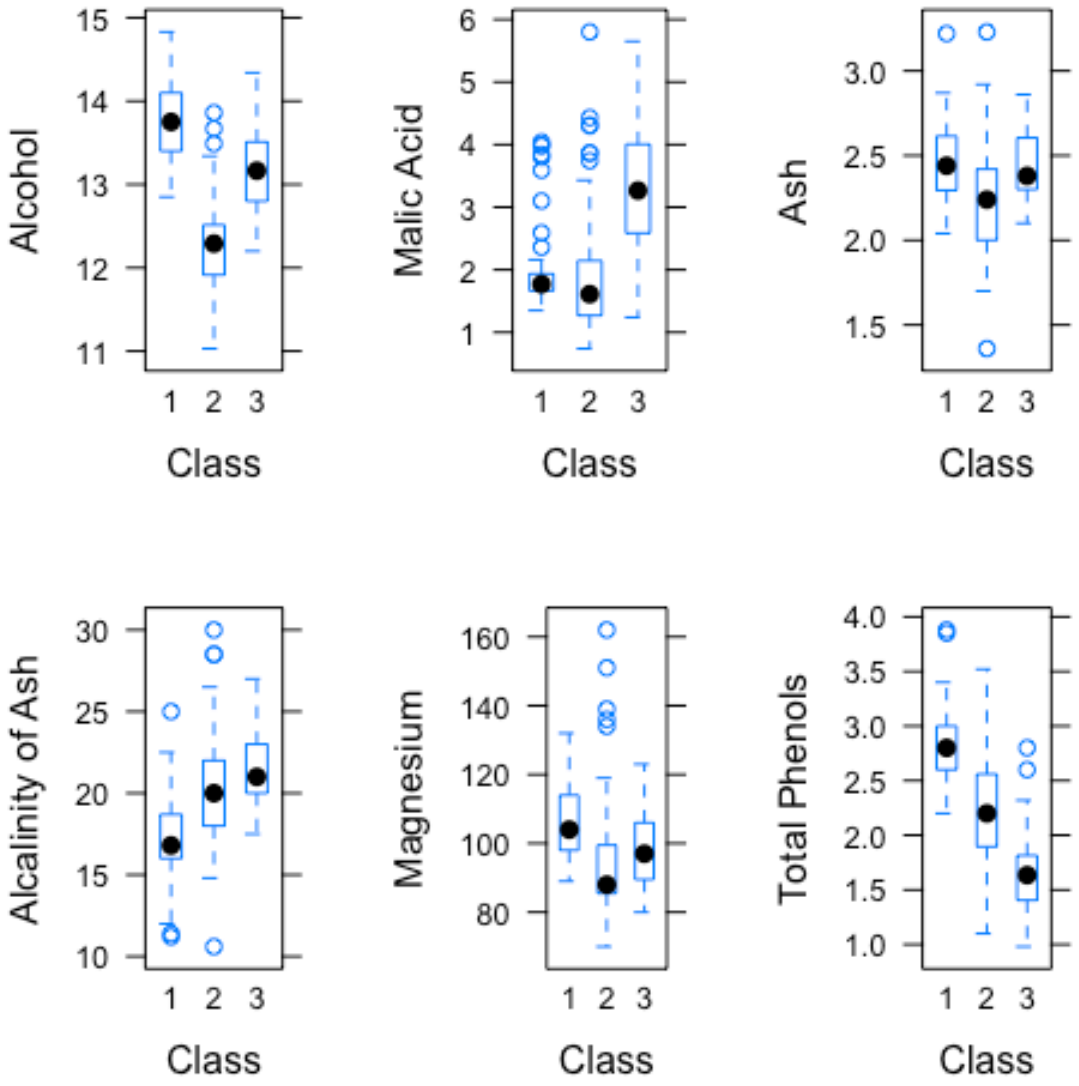
The purpose of EDA is to further examine the data and identify the relationships that are present. Earlier, the categorical response variable was identified and determined to have three levels, which suggests that this is a multiclass classification problem. Understanding the type of problem is important because it helps determine which EDA methods will be most effective.

To better understand the relationships that are present in the data, we need to take a deeper dive and explore each variable's interaction with its corresponding category in the response variable. Many of the tools that were used in the data quality check, such as box plots, density plots and histograms will be used in the EDA process. Only this time, the variables will be against the value in the *Cultivar* variable.

In **Figure 1**, the box plots for the variables *Ash* and *Alcalinity of Ash* are available. While the plots were useful to understand the mean and identify potential outliers, it did not tell us much about the relationships that the variables have to the response variable. **Figure 4** provides another box plot of the first six variables. Here, it is observed that the means for the variables *Alcohol* and *Total Phenols* are significantly different when viewed by the

values of *Cultivar*. Such information may be useful in determining which variables to use in our model, as well as which modeling type may be preferred. Additionally, we see that there are considerably more points outside the whiskers in several of the box plots. This may indicate that outliers need to be considered at the class level, and not for the variable's range as a whole.

Figure 4 – Box plots by Cultivar



In **Figure 2** and **Figure 3**, the distribution of several variables is examined. In **Figure 5** and **Figure 6**, the distributions are examined again, but grouped by the *Cultivar* variable. Such grouping reveals that the distribution by class varies for several of the variables. For example, both plots show that the individual groups for the *Alcohol* variable are approximately normal. In **Figure 6**, it is easy to see that the mean for *Alcohol* differs by

the value of *Cultivar*, and that there appears to be less variance for the first and second classes than that of the third.

Some time was spent reviewing scatter plots as they are typically helpful in identifying patterns in the data. Due to the number of combinations and the lack of insight that they provided, scatter plots were not reported.

Figure 5 – Histograms by Cultivar

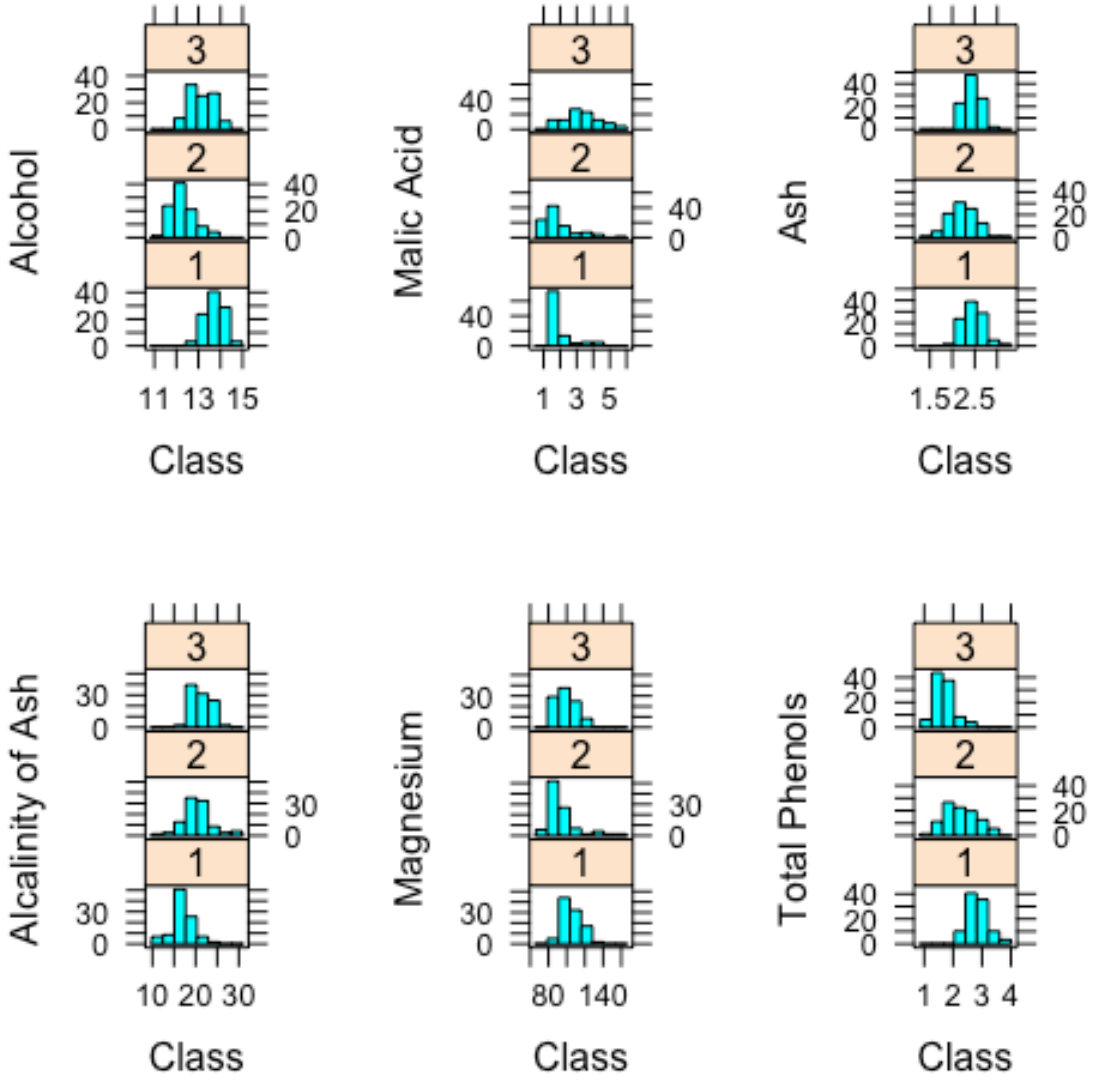
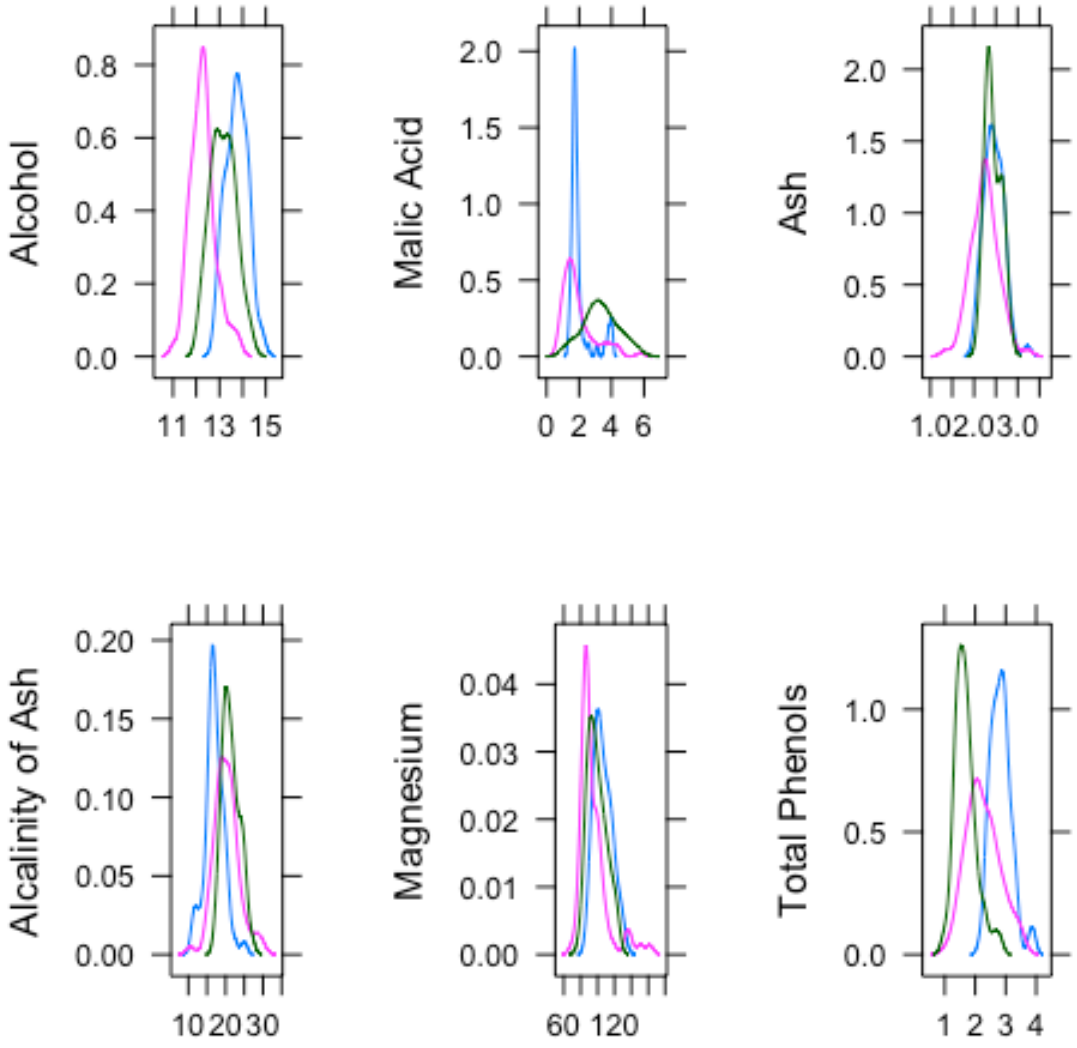
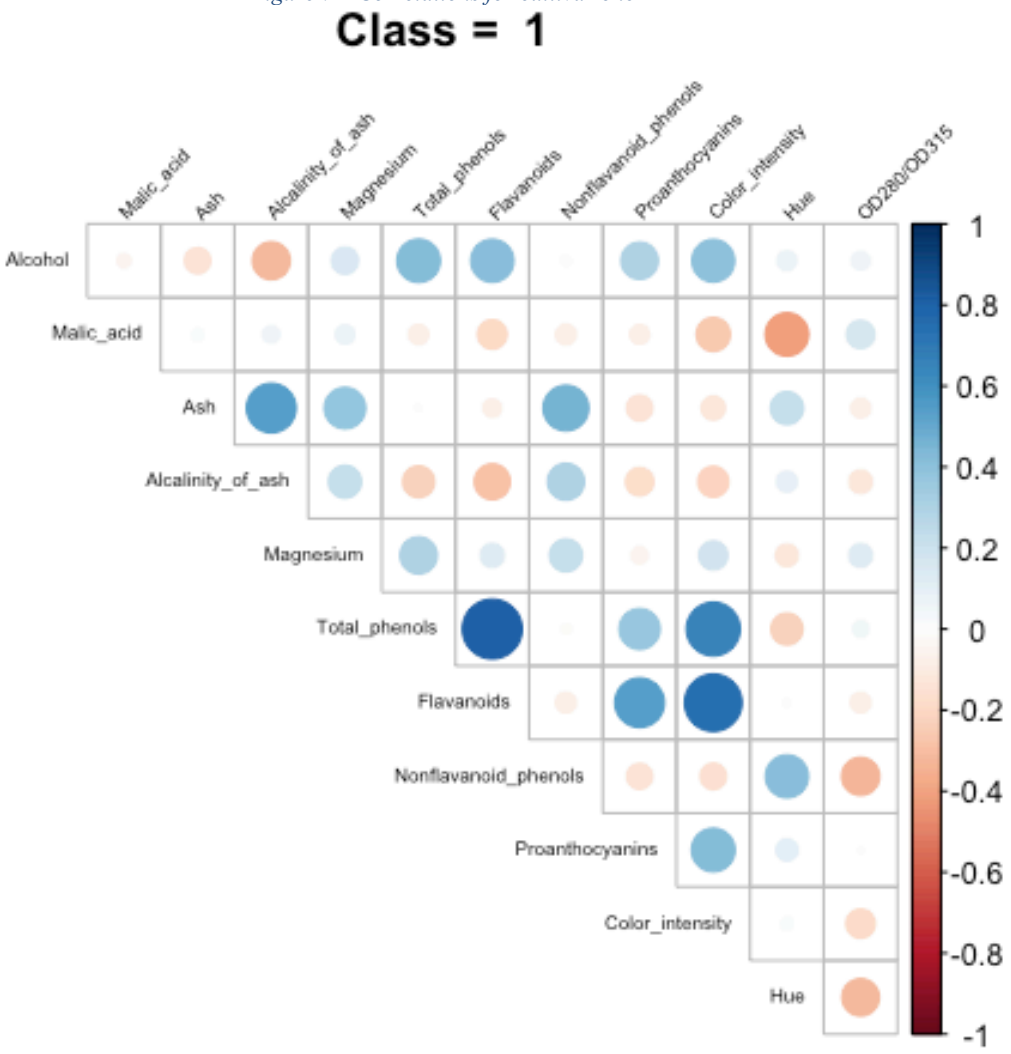


Figure 6 – Density plots by Cultivar



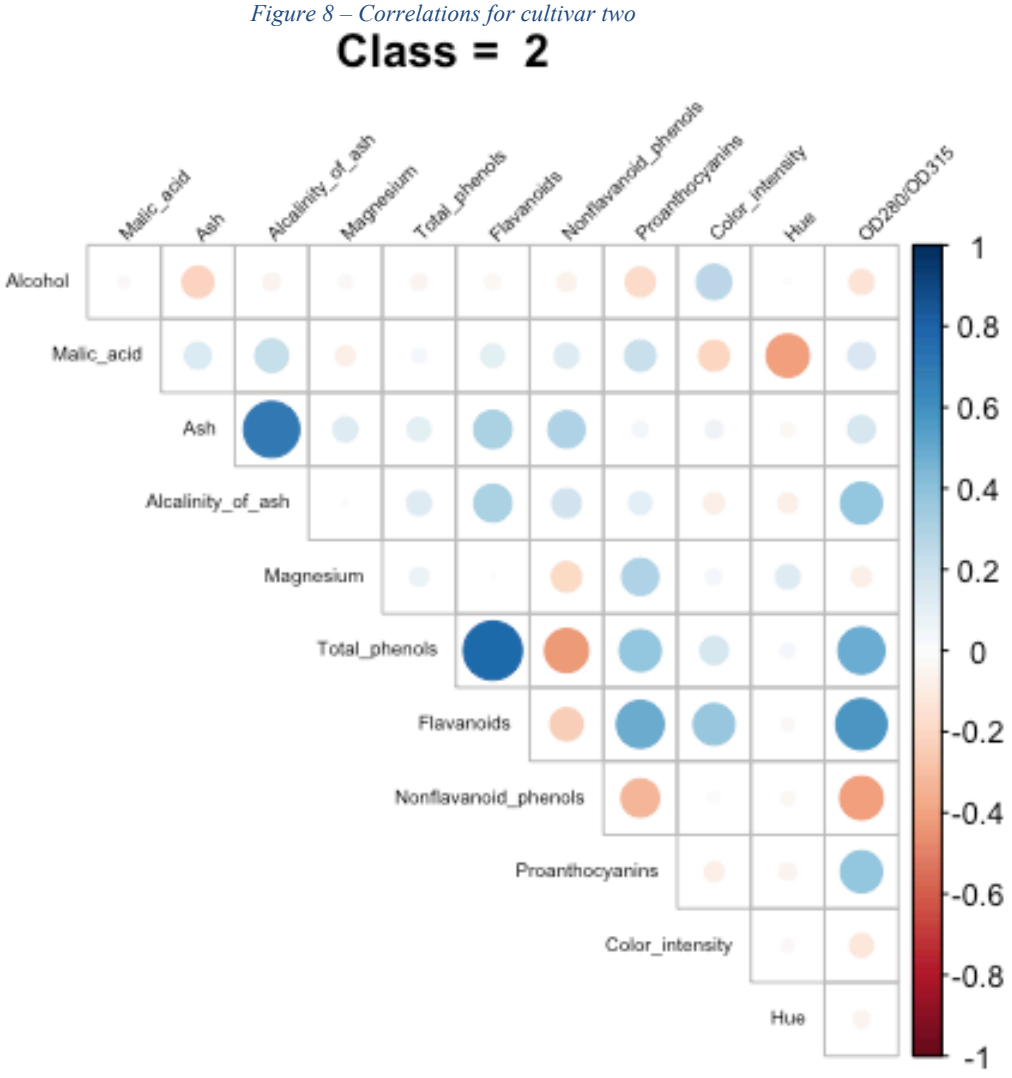
In **Figure 7** the correlations between variables by class where *Cultivar* = 1 are reviewed. There are strong positive correlations between *Color Intensity*, *Total Phenols*, and *Flavanoids*, and a strong negative correlation between *Hue* and *Malic Acid*.

Figure 7 – Correlations for cultivar one



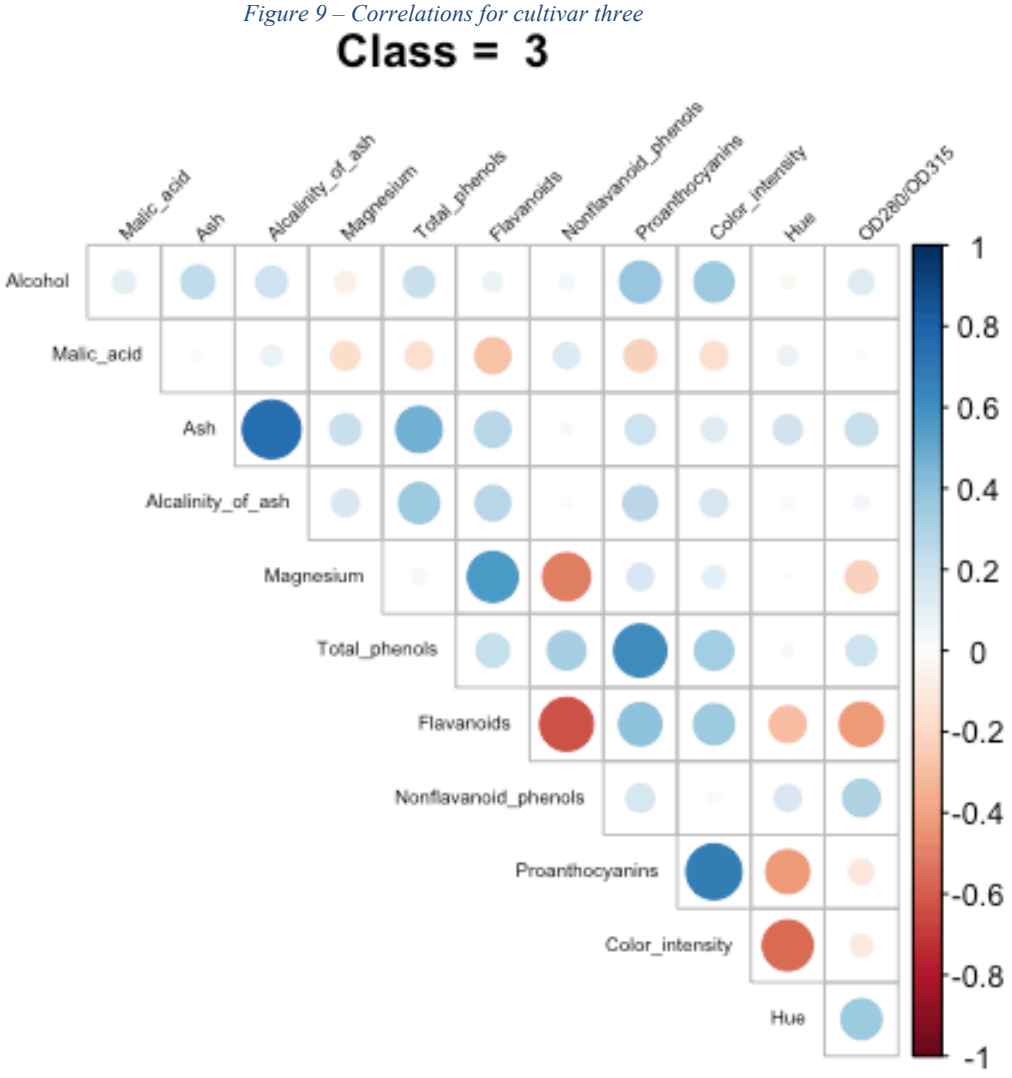
In **Figure 8**, the correlations between the variables when *Cultivar* = 2 is reviewed. There is still a strong positive correlation between *Flavanoids* and *Total Phenols*, but now a strong correlation between *Ash* and *Alcalinity of Ash* is present.

Figure 8 – Correlations for cultivar two



Finally, in **Figure 9** the correlations of the variables when *Cultivar* = 3 are reviewed. It is now observed that the correlation between *Flavanoids* and *Total Phenols* is much less that it was in the previous classes. However, there is a very strong negative correlation between *Flavanoids* and *Nonflavanoid Phenols*. A strong negative correlation between *Color Intensity* and *Hue* is now present as well.

Figure 9 – Correlations for cultivar three

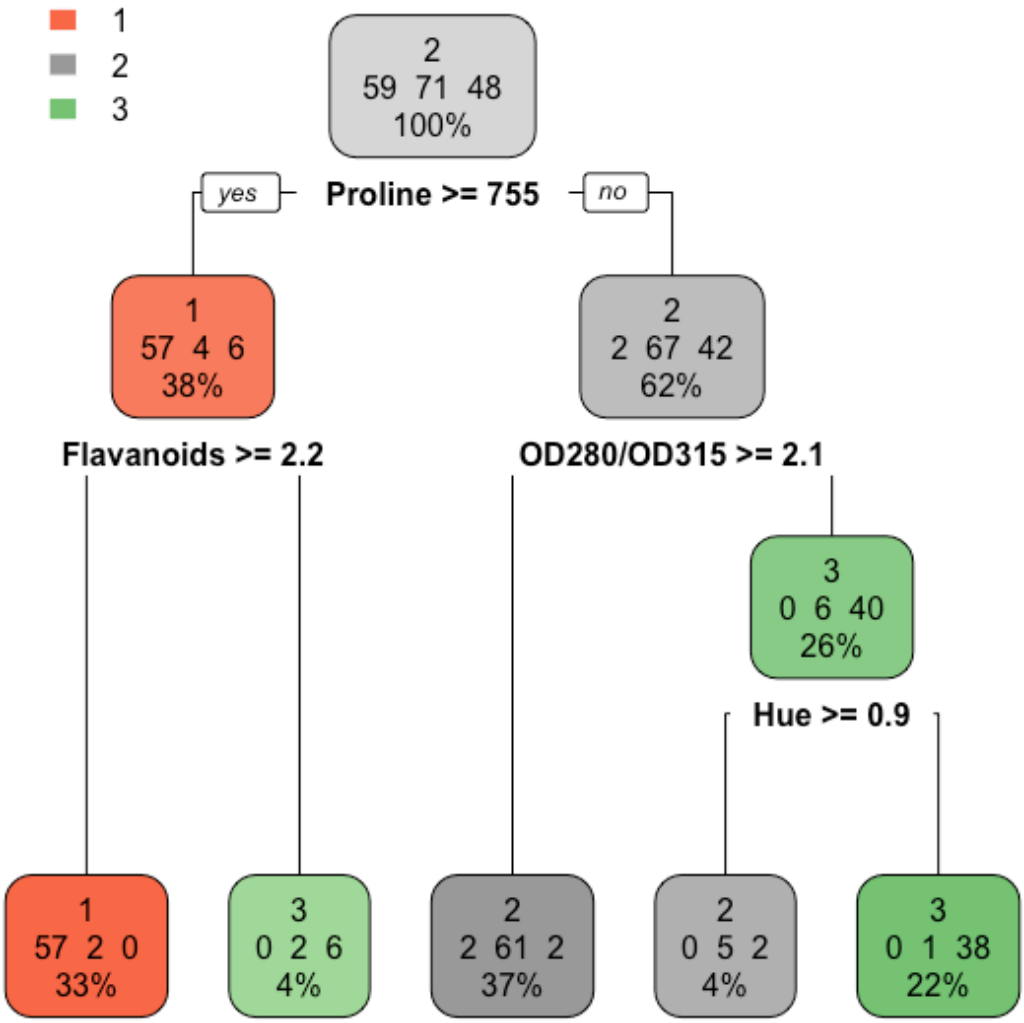


Through the EDA process, some interesting relationships in the data have been identified. There is evidence that wine produced within the same region of Italy does differ by cultivar, and these differences may be exploited to build a predictive model.

Finally, in **Figure 10** a naïve tree model is presented. At this stage, the purpose is not to build advanced models, and the model's actual performance is not critical. The model is simply used to gain insight into the data. During the initial phase of our EDA, it was thought that *Alcohol* might be a good predictor. From the tree model, it is seen that *Proline*, *Flavanoids* and *OD280_OD315* were chosen by the model as significant variables. In this model, the first split occurs for the values of *Proline.* If they are greater than or equal to 755, they branch left where the values of *Flavanoids* are considered, otherwise they branch to the right where the values of *OD280/OD315* are evaluated. If

the values of *OD280_OD315* are less than 2.1, then the *Hue* variable is evaluated. When combined with the correlation and distribution insights from our EDA process, the information from this tree may be leveraged to build a well performing model.

Figure 10 – Naïve tree model



Model Build

In this section, three model types will be fit and reviewed: random forests, support vector machines, and neural networks. While these are known to be popular and powerful modeling techniques for regression and binary classification, they may also be used for multiclass classification problems. However, they do represent some challenges as they are considered 'black-box' techniques, meaning it is not easy or possible to fully understand what is happening within the model.

Model 1 – Random Forest

Model 1 is a random forest built using the 'rf' method from the 'train' function of the 'caret' package. The model was trained against the entire data set and consists of 500 trees with two variables tried at each split. The out-of-bag error estimate [OOB] is 1.12%, as reported by the model summary. This estimate is made during the model build. As each tree is constructed, some data is left out. The left-out cases are then run through the finished trees to obtain a classification. The OOB error estimate is then determined by averaging the number of times the predicted class does not equal the actual class (Breiman & Cutler, n.d.).

The confusion matrix for Model 1 is available in **Table 7**. The results were obtained by using the 'predict' method on the final model. The true positive rate for each cultivar is 100%, with no classification errors present. The results are not too surprising since the predictions were made with in-sample data.

Table 3 – Model 1 confusion matrix

	Reference		
Prediction	One	Two	Three
One	59 : 100%	0	0
Two	0	71 : 100%	0
Three	0	0	48 : 100%

Additional model evaluation metrics are available in **Table 8**. The results are not terribly interesting because of the model's perfect prediction capabilities against the in-sample data, but are displayed because the review of these evaluation metrics is a common means of evaluating a multiclass classification model. The sensitivity and specificity of each class, along with the other evaluation metrics, provides insights into the model's strengths and weaknesses.

Table 4 – Model 1 evaluation metrics

	Class: One	Class: Two	Class: Three
Sensitivity	1	1	1
Specificity	1	1	1
Pos Pred Value	1	1	1
Neg Pred Value	1	1	1
Prevalence	0.3315	0.3989	0.2697
Detection Rate	0.3315	0.3989	0.2697
Detection Prevalence	0.3315	0.3989	0.2697
Balanced Accuracy	1	1	1

It is common to review the ROC curve of a binary classification model. However, since this is a multiclass classification problem, the ROC curves are not valid. Research for this assignment suggested that there are alternative approaches to a ROC curve for multiclass classification problems, such as an averaged ROC curve for all levels of the response variable, or a plot that superimposes the ROC curve from each level of the response

variable onto a single plot. However, opinions on the usefulness of such approaches are conflicting. ROC curves were created and reviewed for each cultivar, but were useful as they provided no additional information not already available in **Table 8**. Additionally, since there is not an agreement on the validity of such an approach the plots are not reported here.

The Variable Importance plot for Model 1 is available in **Figure 11**. The importance of each variable is determined by recording the prediction accuracy of each OOB portion of the data, then permuting each predictor variable. The difference between the two accuracies are averaged over all trees, and normalized by the standard error (Kuhn, 2017). It was a bit of a surprise to see that each cultivar had its own variable importance plot. This suggests that the classification of each level of the response variable is independent and driven by different variables. For cultivar three, the important variables are *Flavanoids* and *Color_intensity*; but for cultivar one the variables *Proline* and *Alcohol* are the most important.

Figure 11 – Model 1 variable importance

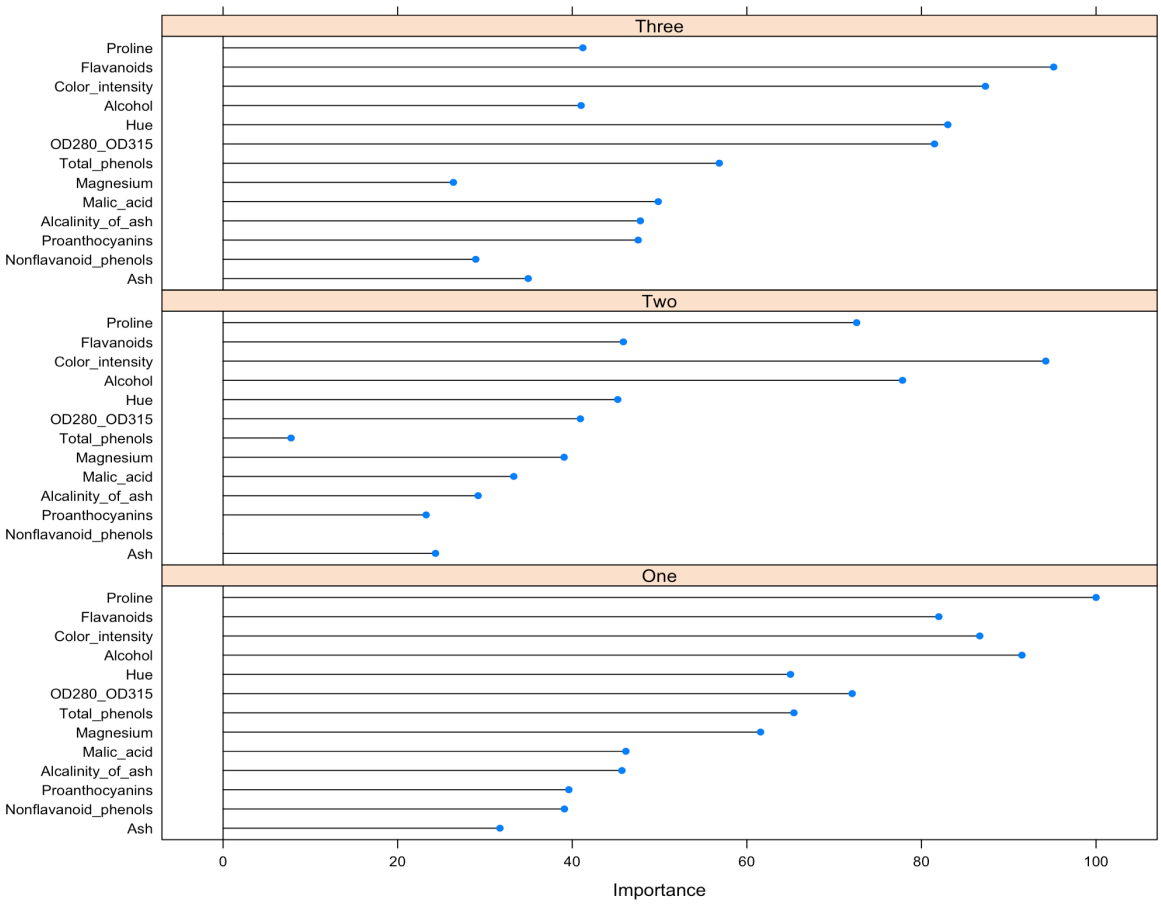
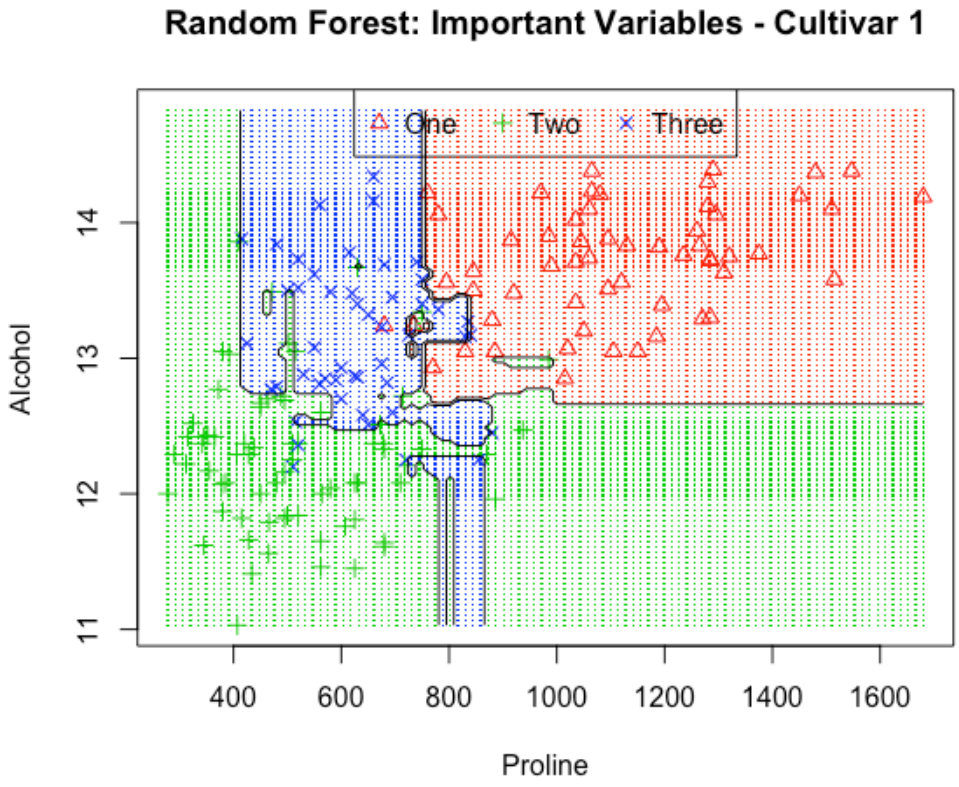


Figure 12 provides a contour plot with decision boundaries for the important variables. This plot was obtained by running a random forest model containing only the variables *Proline*, *Alcohol,* and *Cultivar*, which were identified as important variables for cultivar

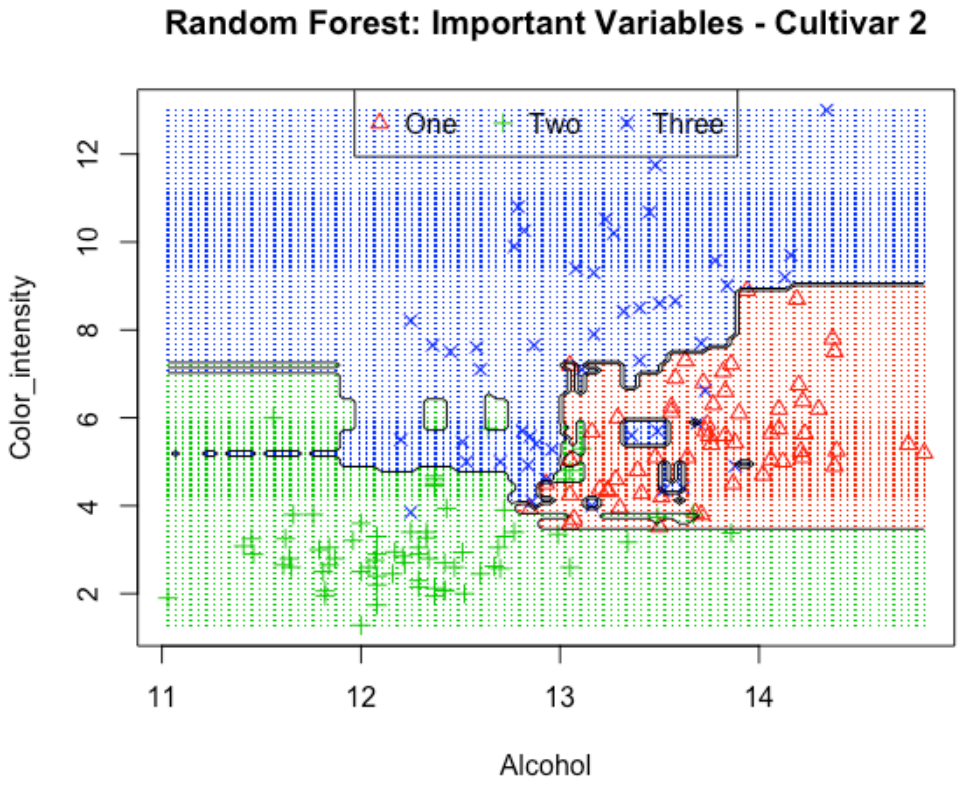
one in **Figure 11**. The background of the plot represents the predicted area for each cultivar. This was obtained by plotting the predicted values on a grid. The actual values are represented by the symbols "Δ", "+", and "X". Ideally, each symbol should fall in the colored area that matches its color, which would represent perfect separation. In **Figure 12**, it is observed that some instances of cultivar one were predicted as cultivar three since they fall in the blue shaded area, which suggests that other variables are needed to completely classify cultivar one.

Figure 12 – Decision boundaries



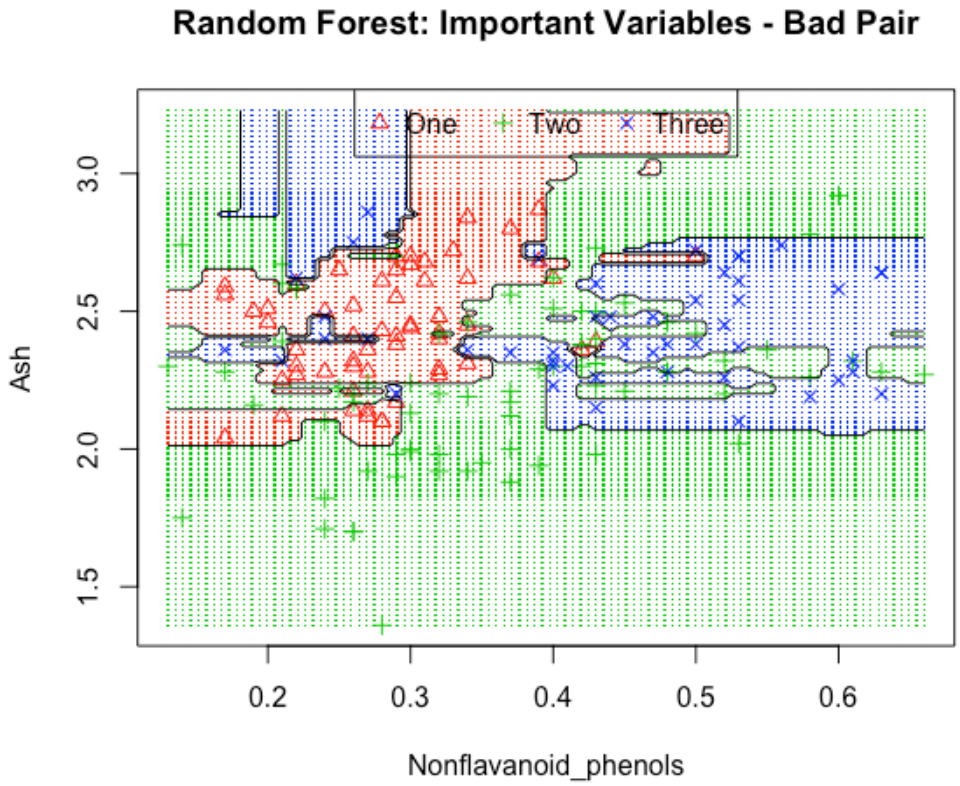
A similar contour plot, **Figure 13,** was created for the important variables for cultivar two. These variables seem to do a good job of identifying cultivar two and there are only a few misclassifications present on the plot. However, note that there are pockets of green shaded areas inside the blue shaded area. This could be due to the way in which the decision boundary function is defined. Perhaps with more observations or additional variables in the model, the boundaries of the shaded areas would adjust so there would be no interruptions. A contour plot was also generated for the important variables of cultivar three, but was excluded from the report to conserve space.

Figure 13 – Decision boundaries



To verify and understand the significance of the variable importance, two lower ranking variables for cultivar one were selected and plotted. This plot is available in **Figure 14**. While this plot still shows acceptable classification between the actual and predicted values, there is no clear grouping of the cultivars. The blue area representing cultivar three is split into two sections, which are separated by both red and green shaded areas.

Figure 14 – Decision boundaries



The 'rf' method of the 'train' function does allow for some tuning of the model. By adjusting the 'mtry' parameter, the number of randomly selected variables to try at each split can be set, which may result in better model performance. Since the results of the model were quite adequate, and no test set was available, no tuning was performed and the final model, as provided by the defaults was accepted.

Model 2 – Support Vector Machine

Model 2 is a Support Vector Machine that was built using the 'svmRadialWeights' method of the 'train' function from the 'caret' package. The model was fit against the full data set using 10-fold cross validation, repeated 3 times. The resulting model is an object of the class 'kvsm' with 178 support vectors, using the Gaussian Radial Basis kernel function, which is used to classify data that is not linearly separable.

The confusion matrix for Model 2 is provided in **Table 9**. As seen, the model has perfect sensitivity and specificity for all classes. Model 2's performance is similar to that of Model 1 against the in-sample data.

Table 5 – Model 2 confusion matrix

	Reference		
Prediction	One	Two	Three
One	59 : 100%	0	0
Two	0	71 : 100%	0
Three	0	0	48 : 100%

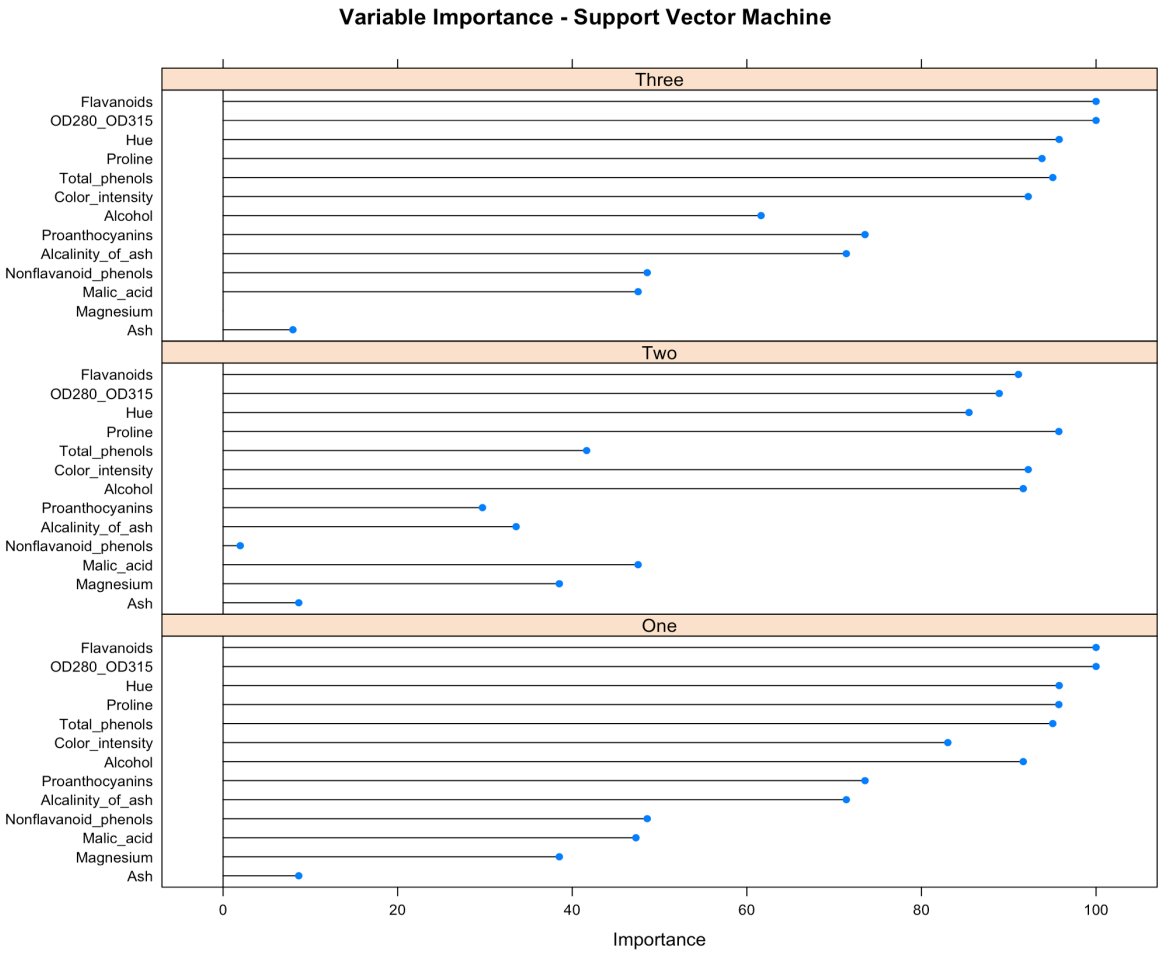
Table 10 lists standard model evaluation metrics that are made available through the 'confusionMatrix' function of the 'caret' package. As with Model 1, there is not much to review here since the model makes no classification errors. However, the table is included here because it is felt that the information it contains may be the best way to understand the performance of a multiclass classification model. By reviewing the metrics for each class, it can be determined how well the model is identifying each class, since it is possible for a model to identify several classes quite accurately, but not perform as well on others.

Table 6 – Model 2 evaluation metrics

	Class: One	Class: Two	Class: Three
Sensitivity	1	1	1
Specificity	1	1	1
Pos Pred Value	1	1	1
Neg Pred Value	1	1	1
Prevalence	0.3315	0.3989	0.2697
Detection Rate	0.3315	0.3989	0.2697
Detection Prevalence	0.3315	0.3989	0.2697
Balanced Accuracy	1	1	1

The Variable Importance plot is available in **Figure 15**. What is most interesting here is that the significant variables identified for cultivars one and three are quite similar, with *Flavanoids* and *OD280_OD315* as having the highest impact. *Proline* and Color_intensity are the most significant variables for cultivar two.

Figure 15 – Model 2 variable importance



Numerous tools are available to display the decision and hyperplane margins in linear and binary classification models, sadly though; many of these tools simply do not work for multiclass classification. Attempts were made to create decision boundary plots that included all predictors, but were unsuccessful. It is believed that the dimensionality of the data set is the culprit, since each predictor adds another dimension and significantly complicates the visualization. At best, only slices may be viewed.

Figure 16 provides such a slice. Since Model 2 identified *Flavanoids* and *OD280_OD315* as important variables for cultivars one and three, a contour plot of their decision boundaries was created. To do this, another SVM model was fitted using only the selected variables, and then the results were plotted. It is observed that these two variables do provide adequate separation, and create three distinct groups. However, there are several misclassifications for cultivars one and two.

Figure 16 – Decision boundaries

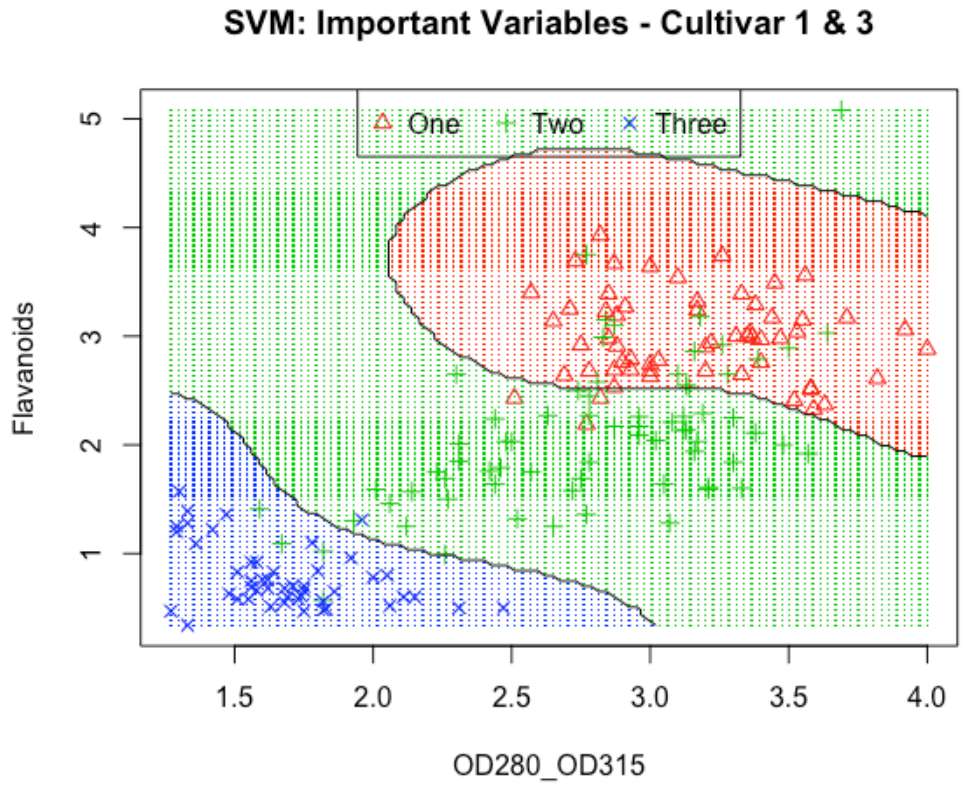


Table 11 and Table 12 provide more details on how well the partial model identifies the cultivars using the two predictors. In total, four cultivar one observations were misclassified, while 19 cultivar two observations were misclassified. The true positive and true negative rates (sensitivity and specificity) are listed in **Table 12**. Here, another complication with multiclass classification is observed. The true positive rate is simple to understand, as there is only one true positive rate for each class. It is different for the true negative, false positive and false negative rates since there are multiple values of each for each class. In **Table 11**, the true negative counts for cultivar two are 55 and 47, but the true negative rate is a single value of 0.9533 so it is not possible to know from just the specificity which class is having a larger impact.

Table 7 – Partial model confusion matrix

		Reference	
Prediction	One	Two	Three
One	55	15	0
Two	4	52	1
Three	0	4	47

Table 8 – Partial model evaluation metrics

	Class: One	Class: Two	Class: Three
Sensitivity	0.9322	0.7324	0.9792
Specificity	0.8739	0.9533	0.9692

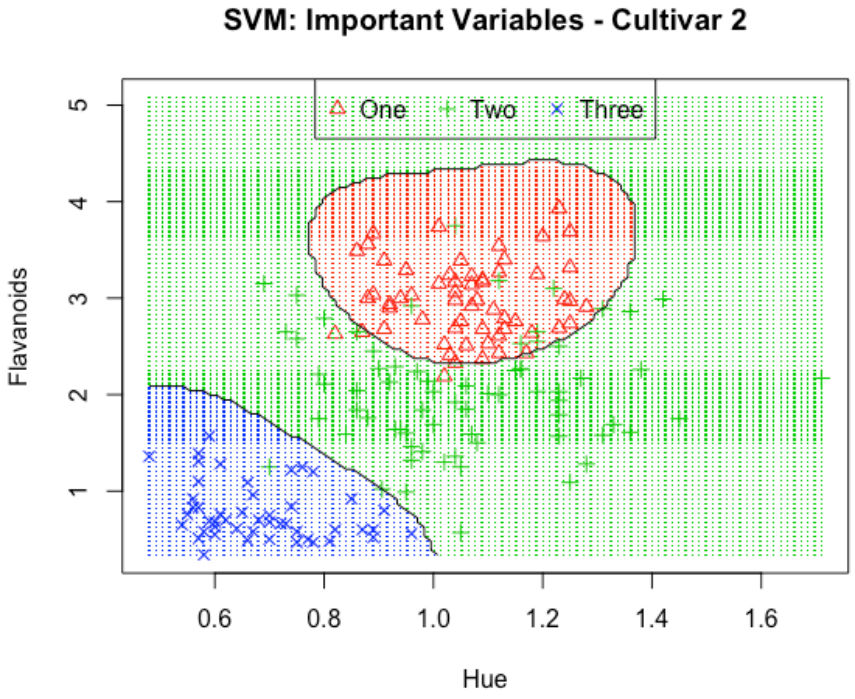
Calculating the rates is a bit more complicating for a multiclass classification problem. **Table 13** provides the manually calculated rates for cultivar one from **Table 11**. Basically, the values for cultivar two and three are merged, then the rates are calculated as class one versus the combination of class two and class three. This would need to be done for the remaining two classes to get the true positive [TPR], true negative [TNR], false positive [FPR], and false negative [FNR] rates. Ideally, a custom function in R would be constructed to produce this information going forward.

Table 9 – Partial model confusion matrix with rates

	Reference	
Prediction	One	Two and Three
One	55 : 93%	15 : 15%
Two and Three	4 : 6.7%	99 : 87%

Since the important variables for cultivar two differ from those of cultivars one and three, a separate plot was created, which is available in **Figure 17**. Again, the separation between classes is not perfect, but it is acceptable. Cultivars one and three are tightly grouped, while cultivar two is more widely spread.

Figure 17 – Decision boundaries



The confusion matrix and evaluation metrics for the model are available in **Table 14** and **Table 15**. The largest misclassification is between cultivars one and two.

Table 10 – Partial model confusion matrix

		Reference	
Prediction	One	Two	Three
One	56	7	0
Two	3	63	0
Three	0	1	48

Table 11 – Partial model evaluation metrics

	Class: One	Class: Two	Class: Three
Sensitivity	0.9492	0.8873	1
Specificity	0.9412	0.972	0.9923

The TPR, TNR, FPR, and FNR rates have been calculated for this model as well. The results are in **Table 16**. The 3x3 confusion matrix is useful, but breaking the matrix down by class and calculating the rates makes it much easier to understand the model's performance for a specific class.

Table 12 – Partial model confusion metrics with rates

	Reference	
Prediction	One	Two and Three
One	56 : 95%	7 : 7%
Two and Three	3 : 5%	111 : 94%

Tuning is an important part of building SVM models. With the 'svmRadialWeights' of 'train' function, there are three parameters that may be tuned to improve model performance. The cost parameter determines how much the model should avoid misclassification. A large value of C will result in a hyperplane with a smaller margin. The sigma parameter essentially determines if the classifier is local or more general, and influences the decision boundary. A large value of sigma makes the classifier more general the decision boundary will be much smoother (Wang, 2014). Lastly, the gamma parameter defines how far the influence of a single training example reaches. Low values of gamma means the reach is far, while smaller values of gamma shorten the reach. No tuning was performed on these models since no test set was available.

Model 3 – Neural Network

Model 3 is an example of a Neural Network. It was trained against the entire data set, and was built using the 'nnet' method of the 'train' function in the 'caret' package. Model 3 is a 13-5-3 network with 88 weights with a decay value of 0.1. That means there were 13 predictors, 5 hidden units, and 3 output layers.

Surprisingly, this model did not perform quite as well as Model 1 and Model 2 with the default parameters. As seen in the confusion matrix in **Table 17**, there was one misclassification of a cultivar one observation.

Table 13 – Model 3 confusion matrix

		Reference	
Prediction	One	Two	Three
One	58 : 98%	0	0
Two	1 : 1.7%	71 : 100%	0
Three	0	0	48 : 100%

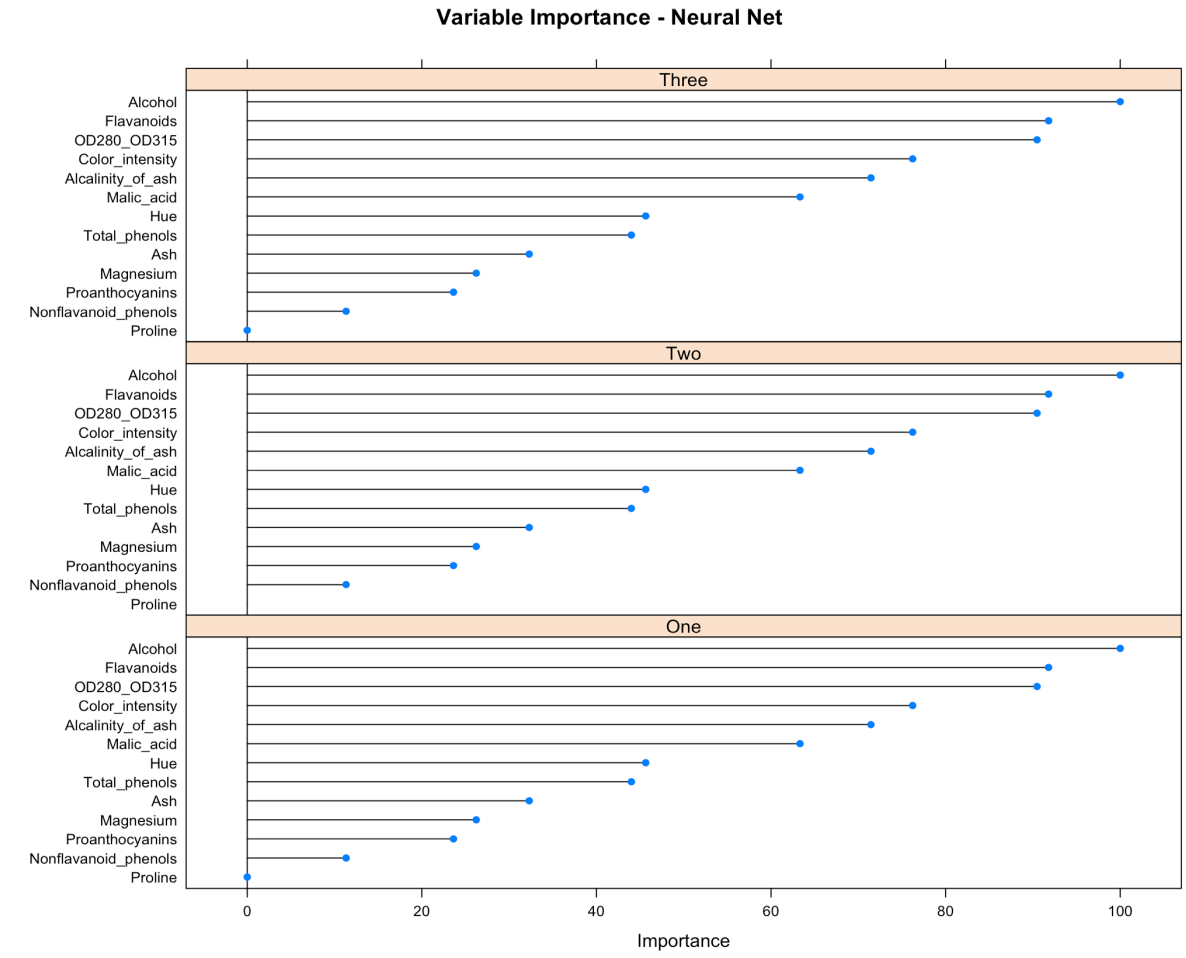
The evaluation metrics for Model 3 are available in **Table 18**. Note how the single misclassification had an impact on the sensitivity of class one and the specificity, or true negative rate of class two. It also impacted the overall model and brought the overall accuracy to 0.9944. Also note that the misclassification had no impact on class three.

Table 14 – Model 3 evaluation metrics

	Class: One	Class: Two	Class: Three
Sensitivity	0.9831	1	1
Specificity	1	0.9907	1
Pos Pred Value	1	0.9861	1
Neg Pred Value	0.9917	1	1
Prevalence	0.3315	0.3989	0.2697
Detection Rate	0.3258	0.3989	0.2697
Detection Prevalence	0.3258	0.4045	0.2697
Balanced Accuracy	0.9915	0.9953	1

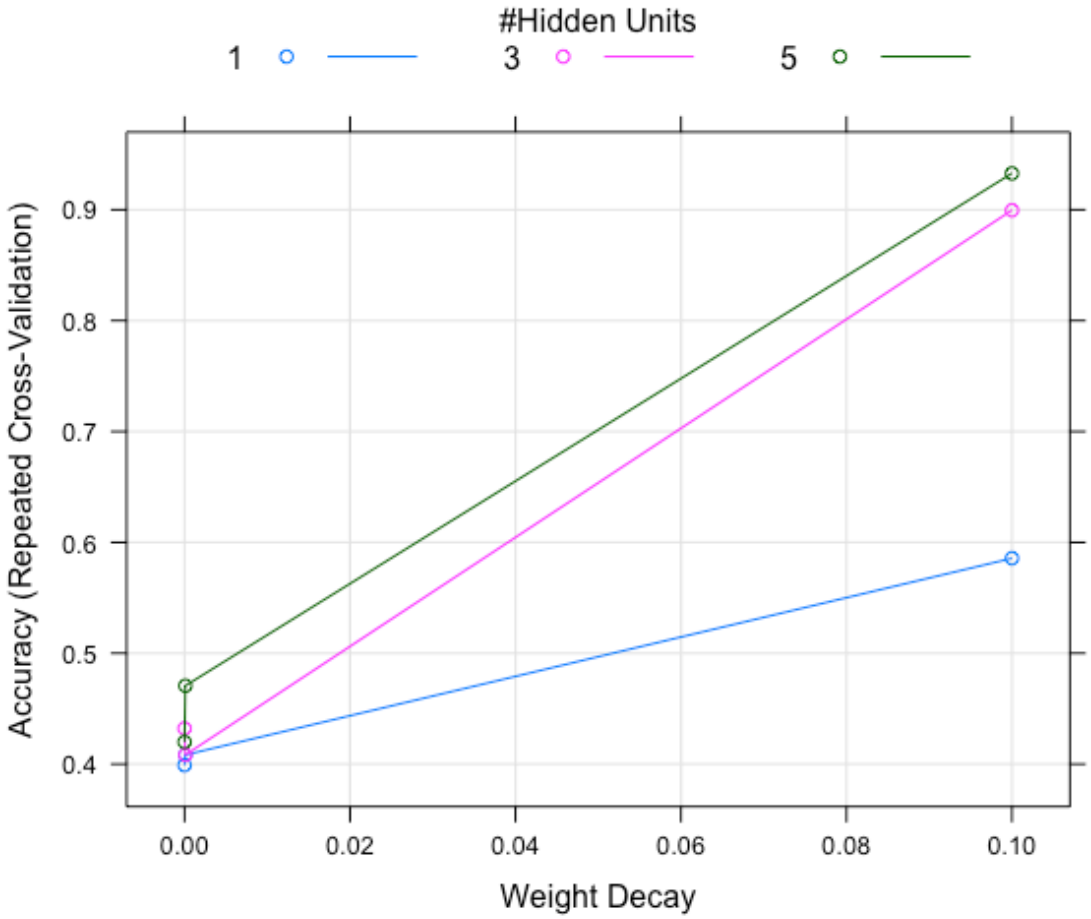
Another interesting aspect of this model is that there appear to be no differences of the variable importance between the classes as observed in Model 1 and Model 2. The Variable Importance table for Model 3 is available in **Figure 18**. *Alcohol*, *Flavanoids*, and *OD280_OD315* play significant roles in the model. That is not a huge surprise as Model 1 and Model 2 also selected these variables as being significant.

Figure 18 – Model 3 variable importance



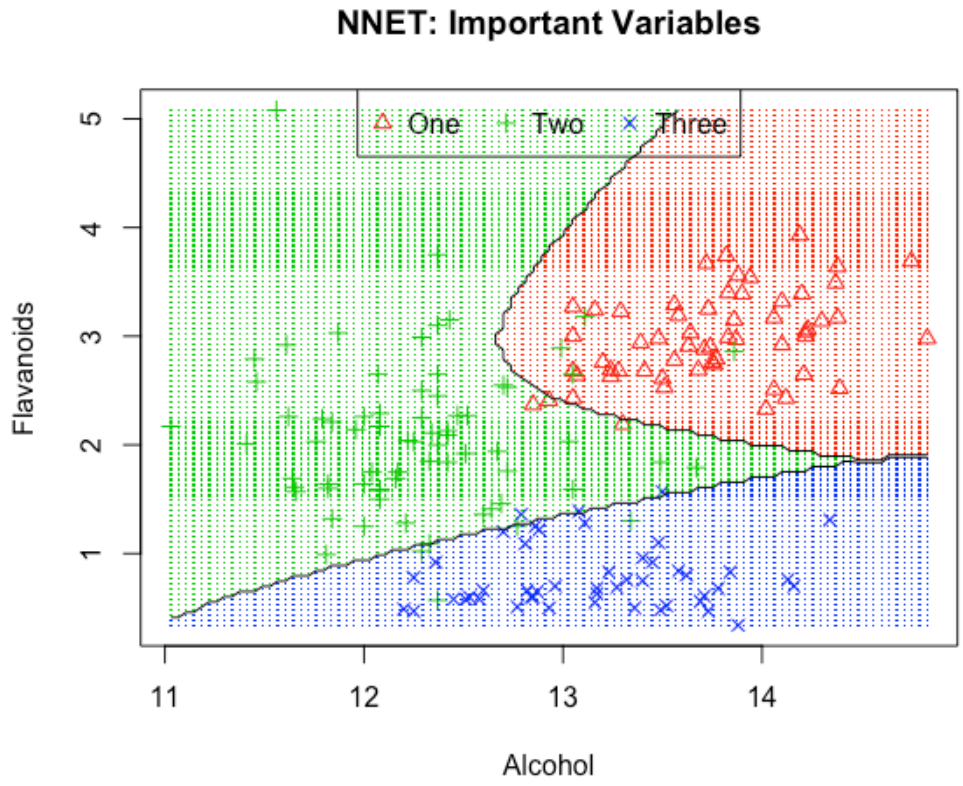
The Accuracy vs Decay plot for Model 3 is available in **Figure 19**. This plot shows the relationship of accuracy and decay for the number of hidden units. Accuracy increases as decay increases for all counts of hidden units, but the optimum accuracy occurs at 0.1 decay with 5 hidden units.

Figure 19 – Model 3 Accuracy vs Decay



A contour plot of the decision boundaries for Model 3 is available in **Figure 20**. The variables *Alcohol* and *Flavanoids* alone are able to adequately separate the classes, but not without some misclassifications. Cultivars one and three are grouped, while cultivar two is more widely spread.

Figure 20 – Decision boundaries



As with Model 1 and Model 2, **Figure 20** was produced by running the model with only the important variables included. The confusion matrix and a sampling of the evaluation metrics for the model are available in **Table 19** and **Table 20**. This information, combined with the contour plot, appear to be the best way to understand how well the model is actually performing.

Table 15 – Partial model confusion matrix

		Reference	
Prediction	One	Two	Three
One	56	4	0
Two	3	63	3
Three	0	4	45

Table 16 – Partial model evaluation metrics

	Class: One	Class: Two	Class: Three
Sensitivity	0.9492	0.8873	0.9375
Specificity	0.9664	0.9439	0.9692

The TPR, TNR, FPR, and FPR rates for cultivar one have been calculated and are available in **Table 21**. Breaking the confusion matrix down in this manner is the best way

to understand how the model is actually performing by class. If this were an actual project, the rates would be calculated for the other classes as well.

Table 17 – Partial model confusion matrix with rates

	Reference	
Prediction	One	Two and Three
One	56 : 95%	4 : 4%
Two and Three	3 : 5%	108 : 96%

The 'nnet' method of the 'train' function allows for some tuning through the 'size' and 'decay' hyper-parameters. The size parameter allows for the number of hidden layers to be set, and decay is the regularization parameter to prevent over-fitting the model. No tuning was performed on Model 3 and the model from the default parameters was accepted.

Model Comparison

While many of the modeling functions that are available in R for binary classification do work for multiclass classification, they are less well developed; which adds to the challenge of interpreting and communicating the results of multiclass classification models. For example, the 'kernlab' package has an implementation of plot for the SVM that produces a nice visualization of the model's hyperplanes and decision boundaries. However, this function only works for binary classification. Such is the case for many of the functions available in R.

The confusion matrix and evaluation metrics that are available for binary classification models are also available for multiclass classification models. They are essentially the same, however each new level of the response variable will add an additional column and an additional row to the confusion matrix, and an additional column to the evaluation metrics table as seen in **Table 17** and **Table 18**. Additionally, the meaning of the confusion matrix and evaluation metrics is essentially the same for binary classification and multiclass classification, but the interpretation is more challenging. The best way to interpret the confusion matrix is to select one class as the base, and interpret the table based on that class. Then move to the next class and do the same, until all classes have been reviewed. Personally, I found it helpful to write down my observations to help keep them straight. It is also helpful to pull out each class and create a "one versus the rest " style confusion matrix to calculate the rates for TPR, TNR, FPR, and FNR, since it is difficult to fully digest the misclassifications in the multiclass confusion matrix.

Visualizing a multiclass classification model is also a challenge. For random forests, it is possible to plot individual trees, and even plot small forests. However, neither is recommended as the individual trees do not tell the whole story, and plotting a forest is impractical and would be difficult to digest. For Support Vector Machines, it is possible to plot the hyperplanes and their margins in a binary classification model, but the added dimensionality of a multiclass classification makes this method impossible. Lastly, for Neural Networks, there were many interesting examples of plots online, but none provided much value. The best alternative for visualizing these models is to view slices

or fragments of the model by plotting the predictions against actual values in a contour plot. While this method does not provide a complete picture, it does provide some great insights on how well the model is able to separate the classes based on variable pairs.

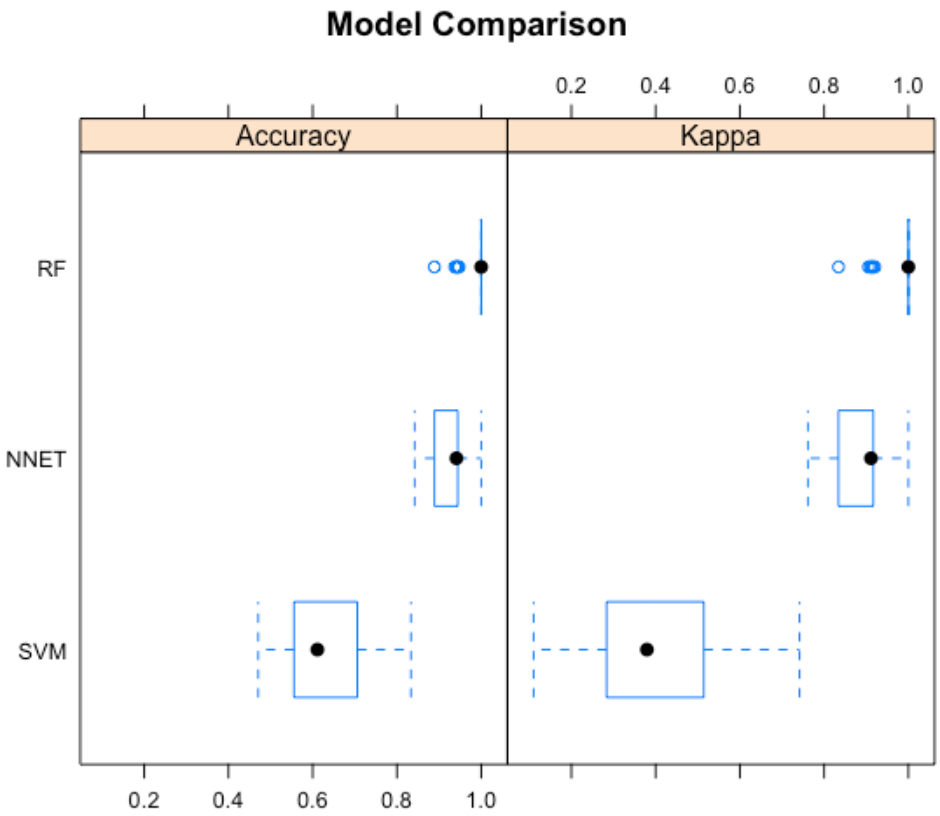
The options for comparing and communicating the results of a multiclass classification model are somewhat limited. The best method for a quick summarization is to simply review the confusion matrix. The evaluation metrics as output by the 'confusionMatrix' function in the 'caret' package and in this report are useful, but they need to be reviewed as a whole so that the information for each class is available. Simply reviewing one class is not sufficient to understand how the model is actually performing.

For future multiclass classification projects a custom function to calculate the TPR, TNR, FPR, and FNR for each class should be investigated, as a single confusion matrix with all rates would be very valuable. Such a confusion matrix would be very valuable for communicating the model as well, especially to a non-technical audience as the rates for TPR, TNR, FPR, and FNR are intuitive and easy to explain, while many of the evaluation metrics are not.

Lastly, as modelers it may be desirable to communicate the number of trees in a random forest, or the number of support vectors in an SVM model, or even the values of the hyper-parameters. Such information may be useful to other modelers to help explain the complexity of the model or defend decisions that were made, but such information will rarely be interesting or useful to business leaders. Also, aside from general information about the model, it really does not provide much value when comparing or selecting a preferred model.

Figure 21 provides box plots for the accuracy and kappa values for Model 1, Model 2, and Model 3. The plot was produced by collecting the resample results for each model. It provides a visualization for selecting the preferred model based on accuracy and the kappa value. It is clear that Model 1 – Random Forest is the clear winner. Model 2 – SVM, came in third even though the model results against the in-sample data out performed the neural net model. According to the resamples that were collected during the model fit, Model 3 – Neural Network consistently outperformed Model 2- SVM, which is something that would not be obvious based on the final models.

Figure 21 – Model comparison



Conclusion

Having completed the data quality check and exploratory data analysis, several models have been fitted. Example random forest, support vector machine, and neural network models have been built and reviewed and the challenges of comparing and communicating the results of such 'black-box' models and multiclass classification models has been examined and discussed. While R is quite robust and offers many options for fitting multiclass classification models, it does not offer many tools to help interpret such models.

'Black-box' models can be difficult to interpret and communicate because it is not possible to see what is happening behind the scenes. But, their performance – which is often superior to other modeling methods can be reviewed and compared, making them a valuable tool in the data scientist's toolbox.

References

Breiman, L., & Cutler, A. (n.d.). Random forests - classification description. Retrieved November 5, 2017, from https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

Calwineries. (2017). Malic Acid in Wine | Calwineries. Retrieved October 1, 2017, from http://www.calwineries.com/learn/wine-chemistry/wine-acids/malic-acid

ETS. (2017). ETS Laboratories - Wine Analysis. Retrieved October 1, 2017, from https://www.etslabs.com/analyses/ALK

Kuhn, M. (2017, September 4). The caret Package. Retrieved November 5, 2017, from https://topepo.github.io/caret/variable-importance.html

Lichman, M. (2013). UCI Machine Learning Repository: Wine Data Set. Retrieved October 1, 2017, from http://archive.ics.uci.edu/ml/datasets/Wine

Maujean, A. (n.d.). Handbook of Enology, The Chemistry of Wine: Stabilization and Treatments - Google Books. Retrieved October 1, 2017, from https://books.google.com/books?id=a03C-aFy2jsC&pg=PA190&lpg=PA190&dq=what+is+OD280+in+wine&source=bl&ots=pBjGwufsUp&sig=CATgukdAeGis1OX_GWDdZqkvPGs&hl=en&sa=X&ved=0ahUKEwjuvZaDh7_WAhVIzFQKHQxWDfQ4FBDoAQgnMAA#v=onepage&q=what%20is%20OD280%20in%20wine&f=false

Puckette, M. (2016, September 16). What Color Tells You About a Wine | Wine Folly. Retrieved October 1, 2017, from http://winefolly.com/review/know-a-wine-just-by-looking-at-the-color/

Robertson, D. (2007, September 17). The skinny on red wine and magnesium - Houston Chronicle. Retrieved October 1, 2017, from http://www.chron.com/news/health/article/The-skinny-on-red-wine-and-magnesium-1806764.php

UC Davis. (2015, October 31). Proanthocyanidins — Waterhouse Lab. Retrieved October 1, 2017, from http://waterhouse.ucdavis.edu/whats-in-wine/proanthocyanidins

Wang, H. (2014, March 16). [ML] How sigma matters in SVM RBF kernel. Retrieved November 5, 2017, from http://haohanw.blogspot.com/2014/03/ml-how-sigma-matters-in-svm-rbf-kernel.html

Wikipedia. (2017, September 11). Phenolic content in wine - Wikipedia. Retrieved October 1, 2017, from https://en.wikipedia.org/wiki/Phenolic_content_in_wine

Wikipedia. (2017, July 13). Wine color - Wikipedia. Retrieved October 1, 2017, from https://en.wikipedia.org/wiki/Wine_color

WineEducation. (n.d.). What wine is made of. Retrieved October 1, 2017, from http://www.wineeducation.com/wineismadeof.html

Appendix

```
# Libraries
library('lattice')
library('hexbin')
library('klaR')
library('caret')
library('corrplot')
library('rpart')
library('rpart.plot')

# helper function to print decision boundaries for variables
# adapted for this assignment
# original author: http://michael.hahsler.net/SMU/EMIS7332/R/viz_classifier.html
decisionplot <- function(model, data, class = NULL, predict_type = "class",
        resolution = 100, showgrid = TRUE, ...) {

        if(!is.null(class)) cl <- data[,class] else cl <- 1
        data <- data[,1:2]
        k <- length(unique(cl))

        plot(data, col = as.integer(cl)+1L, pch = as.integer(cl)+1L, ...)
        legend("top", legend = unique(cl), horiz = TRUE,
              col = as.integer(unique(cl))+1L, pch = as.integer(unique(cl))+1L)

        # make grid
        r <- sapply(data, range, na.rm = TRUE)
        xs <- seq(r[1,1], r[2,1], length.out = resolution)
        ys <- seq(r[1,2], r[2,2], length.out = resolution)
        g <- cbind(rep(xs, each=resolution), rep(ys, time = resolution))
        colnames(g) <- colnames(r)
        g <- as.data.frame(g)

        ### guess how to get class labels from predict
        ### (unfortunately not very consistent between models)
        p <- predict(model, g, type = predict_type)
        if(is.list(p)) p <- p$class
        p <- as.factor(p)

        if(showgrid) points(g, col = as.integer(p)+1L, pch = ".")

        z <- matrix(as.integer(p), nrow = resolution, byrow = TRUE)
        contour(xs, ys, z, add = TRUE, drawlabels = FALSE,
              lwd = 1, levels = (1:(k-1))+.5)

        invisible(z)
}
```

```
# Helper function to print multiple plots on a lattice panel
print_plot_panel <- function(plots) {
	print(plots[[2]], split = c(1, 1, 3, 2), more = TRUE)
	print(plots[[3]], split = c(2, 1, 3, 2), more = TRUE)
	print(plots[[4]], split = c(3, 1, 3, 2), more = TRUE)

	print(plots[[5]], split = c(1, 2, 3, 2), more = TRUE)
	print(plots[[6]], split = c(2, 2, 3, 2), more = TRUE)
	print(plots[[7]], split = c(3, 2, 3, 2), more = FALSE)

	print(plots[[8]], split = c(1, 1, 3, 2), more = TRUE)
	print(plots[[9]], split = c(2, 1, 3, 2), more = TRUE)
	print(plots[[10]], split = c(3, 1, 3, 2), more = TRUE)

	print(plots[[11]], split = c(1, 2, 3, 2), more = TRUE)
	print(plots[[12]], split = c(2, 2, 3, 2), more = TRUE)
	print(plots[[13]], split = c(3, 2, 3, 2), more = FALSE)

	print(plots[[14]], split = c(1, 1, 3, 2), more = FALSE)
}

print_plot_panel_6 <- function(plots) {
	print(plots[[2]], split = c(1, 1, 3, 2), more = TRUE)
	print(plots[[3]], split = c(2, 1, 3, 2), more = TRUE)
	print(plots[[4]], split = c(3, 1, 3, 2), more = TRUE)

	print(plots[[5]], split = c(1, 2, 3, 2), more = TRUE)
	print(plots[[6]], split = c(2, 2, 3, 2), more = TRUE)
	print(plots[[7]], split = c(3, 2, 3, 2), more = FALSE)
}

# Data import and preparation
wine_data <- read.csv("Data/wine.data.txt", header = FALSE)
var_names <- c("Cultivar", "Alcohol", "Malic_acid","Ash", "Alcalinity_of_ash"
	, "Magnesium", "Total_phenols", "Flavanoids", "Nonflavanoid_phenols"
	, "Proanthocyanins", "Color_intensity", "Hue", "OD280_OD315"
	, "Proline")

names(wine_data) <- var_names

dim(wine_data)

summary(wine_data)

# set Cultivar as factor
```

```
wine_data$Cultivar <- as.factor(wine_data$Cultivar)

# Set factor variable levels
levels(wine_data$Cultivar) = c("One", "Two", "Three")var_cnt <- dim(wine_data)[2]

# Correlation
for (level in unique(wine_data$Cultivar)) {
      corrplot(cor(wine_data[wine_data$Cultivar == level, 2:13]),
            tl.col = "black", tl.cex = 0.5, title = paste("Cultivar = ", level),
            tl.srt = 45, mar=c(0,0,1,0), diag = FALSE, type = "upper")
}

# Decision tree - create a simple tree to see what we can learn
tree <- rpart(Cultivar ~ ., data = wine_data)
rpart.plot(tree, extra = 101, type = 2)


# Box plots - groups = Cultivar
myplots <- list()
for (var in 2:var_cnt) {
      myplots[[var]] <- bwplot(wine_data[,var] ~ wine_data$Cultivar,
ylab=paste(var_names[var]), xlab = "Cultivar")
}

print_plot_panel(myplots)

# Histogram - groups = Cultivar
myplots <- list()
for (var in 2:var_cnt) {
      myplots[[var]] <- histogram(~wine_data[,var] | wine_data$Cultivar,
ylab=paste(var_names[var]), xlab = "Cultivar", layout = c(1,3))
}

print_plot_panel(myplots)

# Density plot; groups = Cultivar
myplots <- list()
for (var in 2:var_cnt) {
      myplots[[var]] <- densityplot(~wine_data[,var], groups = wine_data$Cultivar,
ylab=paste(var_names[var]), xlab = "", plot.points = FALSE)
}

print_plot_panel(myplots)
```

**Note – Model 2 and Model 3 are fitted using the same technique as Model 1. For that reason, only the ‘train’ function declaration will be included. To duplicate any plot,

simply change the variables in the ‘vars’ declaration then run the partial model and decisionplot function.

```
## Model Build
# Model 1 - Random Forest
# Set run parameters
fitControl = trainControl(method="repeatedcv", number=10, repeats=3, classProbs = TRUE)

# Fit the model
set.seed(123)
wine_rf <- train(Cultivar ~ .,
                 data = wine_data,
                 method = "rf",
                 importance = TRUE,
                 trControl = fitControl)

# Summary
wine_rf$finalModel

# Plot important variables identified by the random forest model
plot(varImp(wine_rf), main = "Variable Importance - Random Forest", layout = c(1,3))

# Training set predictions for roc
wine_rf_pred_probs <- predict(wine_rf, wine_data, type = "prob")

## Create ROC curve
wine_rf_roc <- roc(response = wine_data$Cultivar, predictor = wine_rf_pred_probs[,2], levels = c("One", "Two"))

## Plot ROC curve
plot(wine_rf_roc, col = "blue", main = paste0("ROC Curve for Random Forest - 1: AUC ", round(wine_rf_roc$auc[1], 4)))
# not falid for or useful for two reasonse... one this is multiclass.. 2 with a sensitivy of 1, all plots are the same and no information can be gained
# Create the confusion matrix
wine_rf_pred <- predict(wine_rf, wine_data)
confusionMatrix(data = wine_rf_pred, wine_data$Cultivar, positive = "One")

# Decision boundaries - important variables: Cultivar One
vars <- wine_data[,c("Proline", "Alcohol", "Cultivar")]
fitControl = trainControl(classProbs = TRUE)

set.seed(123)
impVars <- train(Cultivar ~ .,
                 data = vars,
```

```
            method = "rf",
            importance = TRUE,
            trControl = fitControl)

decisionplot(model = impVars, data = vars, class = "Cultivar",
        main = "Random Forest: Important Variables - Cultivar 1", predict_type = "raw")

# Decision boundaries - important variables: Cultivar Two
vars <- wine_data[,c("Alcohol", "Color_intensity", "Cultivar")]
fitControl = trainControl(classProbs = TRUE)

set.seed(123)
impVars <- train(Cultivar ~ .,
             data = vars,
             method = "rf",
             importance = TRUE,
             trControl = fitControl)

decisionplot(model = impVars, data = vars, class = "Cultivar",
        main = "Random Forest: Important Variables - Cultivar 2", predict_type = "raw")

# Decision boundaries - important variables: bad pair
vars <- wine_data[,c("Nonflavanoid_phenols", "Ash", "Cultivar")]
fitControl = trainControl(classProbs = TRUE)

set.seed(123)
impVars <- train(Cultivar ~ .,
            data = vars,
            method = "rf",
            importance = TRUE,
            trControl = fitControl)

decisionplot(model = impVars, data = vars, class = "Cultivar",
        main = "Random Forest: Important Variables - Bad Pair", predict_type = "raw")

# Model 2
set.seed(123)
wine_svm <- train(Cultivar ~ .,
             data = wine_data,
             method = "svmRadialWeights",
             importance = TRUE,
             trControl = fitControl)

# Model 3
set.seed(123)
wine_nnet <- train(Cultivar ~ .,
```

```
            data = wine_data,
            method = "nnet",
            trace = FALSE,
            importance = TRUE,
            trControl = fitControl)

# Compare models
# Collect resamples
results <- resamples(list(RF=wine_rf, SVM=wine_svm, NNET=wine_nnet))

# Summarize the distributions
summary(results)

# Boxplots of results
bwplot(results, main = "Model Comparison")
```