Predict 454: Advanced Machine Learning

Assignment 4

Predictive Modeling in Multiclass classification

Jack Doig

Introduction

This paper outlines the results from an exercise in exploratory data analysis (EDA) and statistical graphics, and predictive modeling, involving the “Wine Data Set” from the University of California, Irvine’s (UCI) Machine Learning Repository.

The Data originate from a chemical analysis of wines grown from the same region in Italy but from three different cultivars (groupings of plants propagated through cultivation). Each observation represents data collected on a particular wine. The Data were donated to the UCI in 1991by Stefan Aeberhard.

This paper firstly does a simple data quality check, followed by some basic EDA, some more advanced model-driven EDA, and finally some predictive modeling.

The modelling exercise in question is a multi-class classification problem, where the continuous predictors in the data set are used to predict the cultivar, or wine grouping. The EDA focuses on interesting findings in the data which may inform future classification modeling, rather than describing every aspect and relationship, and then finally models and selects the best model.

Part 1: Data Quality Check

1.1 Basic Data understanding

The purpose of this section is to gain a basic statistical understanding of the data set. The table below describes the structure of the data set.

Metric	Value
Number of observations	177
Number of Categorical Variables	1
Number of Continuous Variables	13
Total Number of Variables	14
Size of Data File	3.0 kb

Figure 1.1: basic data set metrics

The table above shows that this data set is made up exclusively of continuous variables (with the exception of the target variable). It is also a relatively small data set, with only 177 observations, so there should be no computational challenges in this project, however this may have a downstream impact on the decision of validation and training set size during modeling.

The table below describes each of the 14 variables. It is important to invest time in gaining a real-world understanding of the data so as to understand the real-world impact of outliers, patterns and distributions, and possibly get clues so as to optimize the EDA and modeling process.

The target variable is the classID, which is a categorical variable, representing the type of wine, or specifically which cultivar the wine comes from.

Variable	Type	Description
Class ID	Categorical	Represents the three types of wine, or which cultivar the wine comes from, represented as a number (1-3)
Alcohol	Continuous	Represents the alcohol (mainly ethonol) content of the wine
Malic acid	Continuous	Main contributor of acidity in the grape, used as fuel by the vine in respiration. Carefully controlled by winemaker so as to prevent flat (low) or sour (high) taste.
Ash	Continuous	Defined as the inorganic matter that remains after evaporation and incineration, expect 2.5g/litre
Alcalinity of ash	Continuous	Alcalinity of the inorganic matter
Magnesium	Continuous	Wine tends to have an average content of Magnesium of 114mg/l. Naturally occuring in grapes
Total phenols	Continuous	A large group of chemical compounds that affect taste. Color and mouthfeel of the wine, separated into Flavanoids and non-flavanoids
Flavanoids	Continuous	Phenols responsible for much of the flavor and body of wine, a major component of the wine. Can make up 90% of the phenol content in red wine.
Nonflavanoid phenols	Continuous	Tthe majority of nonflavonoid phenols are sourced naturally from grape pulp. Most nonflavonoids are present below their sensory threshold, though collectively they can have an impact on bitterness and astringency.
Proanthocyanins	Continuous	Proanthocyanidins are the principal polyphenols in red wine, influence the aroma, flavor, mouth-feel and astringency of red wines
Color intensity	Continuous	A scientific measure of the wine color
Hue	Continuous	The degree to which the wine color is consistent or different from the primary hues
OD280/OD315 of diluted wines	Continuous	unknown
Proline	Continuous	The most abundant amino acid present in wine.

Figure 1.2: basic description of variables

1.2 Basic Descriptive Statistics

The table below outlines the basic statistics of the data.

	classID	Alcohol	MalicAcid	Ash	AlcalinityOfAs	Magnesium	TotalPhenols	Flavanoids	NonflavanoidP	Proanthocyan	ColorIntens	Hue	OD280OD31	Proline
nobs	177	177	177	177	177	177	177	177	177	177	177	177	177	177
NAs	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Minimum	1	11.03	0.74	1.36	10.6	70	0.98	0.34	0.13	0.41	1.28	0.48	1.27	278
Maximum	3	14.83	5.8	3.23	30	162	3.88	5.08	0.66	3.58	13	1.71	4	1680
1. Quartile	1	12.36	1.6	2.21	17.2	88	1.74	1.2	0.27	1.25	3.21	0.78	1.93	500
3. Quartile	3	13.67	3.1	2.56	21.5	107	2.8	2.86	0.44	1.95	6.2	1.12	3.17	985
Mean	1.944	12.994	2.340	2.366	19.517	99.588	2.292	2.023	0.362	1.587	5.055	0.957	2.604	745.096
Median	2.000	13.050	1.870	2.360	19.500	98.000	2.350	2.130	0.340	1.550	4.680	0.960	2.780	672.000
Sum	344.000	2299.880	414.160	418.810	3454.500	17627.000	405.730	358.150	64.130	280.890	894.700	169.386	460.960	131882.000
SE Mean	0.058	0.061	0.084	0.021	0.251	1.065	0.047	0.075	0.009	0.043	0.175	0.017	0.053	23.668
LCL Mean	1.829	12.874	2.174	2.325	19.022	97.485	2.199	1.875	0.344	1.502	4.710	0.923	2.500	698.386
UCL Mean	2.058	13.114	2.506	2.407	20.012	101.690	2.385	2.172	0.381	1.672	5.400	0.991	2.709	791.806
Variance	0.599	0.654	1.253	0.076	11.129	200.903	0.392	0.997	0.016	0.327	5.403	0.053	0.497	99151.962
Stdev	0.774	0.809	1.119	0.275	3.336	14.174	0.626	0.999	0.125	0.572	2.324	0.229	0.705	314.884
Skewness	0.096	-0.046	1.014	-0.169	0.201	1.103	0.096	0.036	0.433	0.524	0.856	0.027	-0.315	0.771
Kurtosis	-1.335	-0.875	0.201	1.011	0.420	2.109	-0.867	-0.905	-0.689	0.507	0.288	-0.408	-1.128	-0.278
0.01Percentile	1	11.41	0.89	1.7	11.2	78	1.1	0.47	0.14	0.42	1.74	0.54	1.29	290
0.05Percentile	1	11.65	1.01	1.92	14.6	80	1.38	0.52	0.19	0.73	2.08	0.57	1.42	352
0.95Percentile	3	14.22	4.6	2.75	25	124	3.3	3.54	0.6	2.76	9.7	1.31	3.58	1310
0.99Percentile	3	14.75	5.65	3.22	28.5	151	3.85	3.93	0.63	3.28	11.75	1.45	3.82	1547

Figure 1.3: data set descriptive statistics

The following observations can be made about the basic statistics outlined in the above table:

1. There appear to be no missing variables, which is an important observation as it means no data will need to be inputed.
2. There is a strong skewness in 2 of the variables, with Malic Acid and Magnesium with and absolute value of greater than 1. These distributions may need transformation or special treatment during modeling to maximize predictive power

3. There is a reasonably strong Kurtosis for the Magnesium variable, again this may need treatment as per 2.

1.3 Unconditional Distributions of variables

The distributions of the continuous data are shown in the image below.

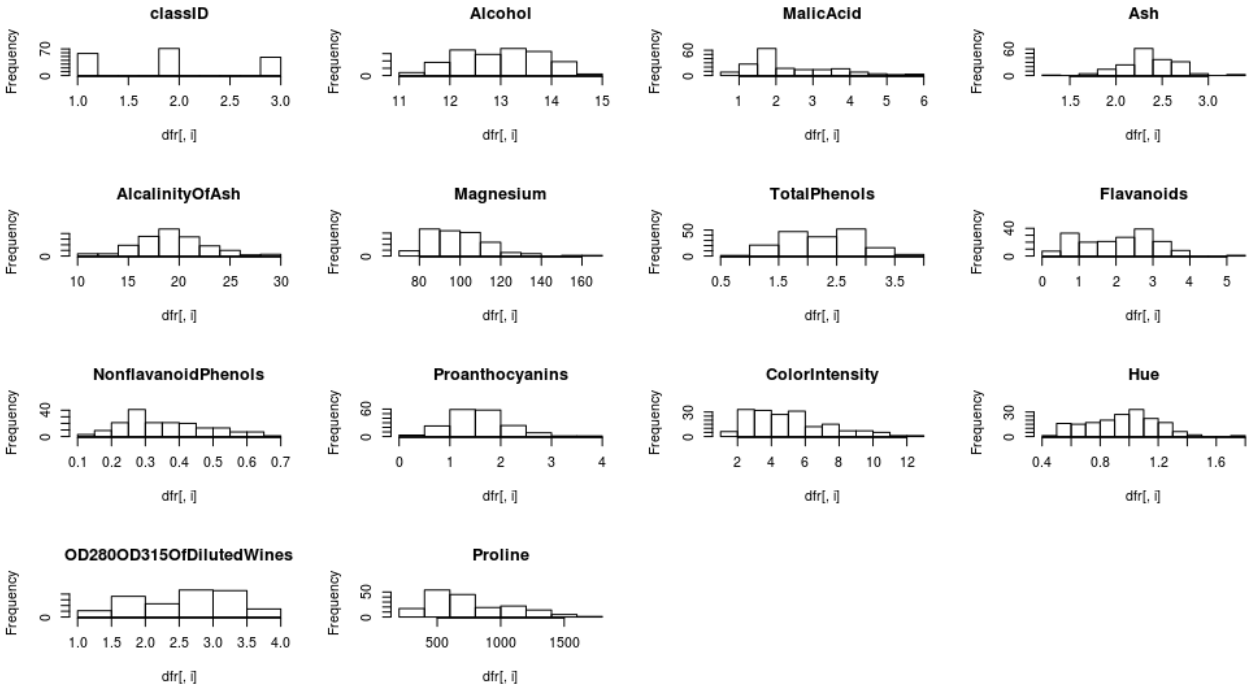


Figure 1.4: All variable distributions

The observations of these distributions are:

1. Some of the independent variables appear to have a normal distribution (Alcalinity of Ash, Ash, Hue). This could suggest these variables are independent of the wine cultivar.
2. Several appear to have multiple peaks (Alcohol, Total Phenols, OD280/OD315 of diluted wines, Flavanoids), indicating there is possibly some underlying pattern which we may discover later.
3. Others have either a possible square distribution (Magnesium), or a skewed distribution with a very narrow peak and long tail(s) (Malic Acid, NonFlavanoid Phenols)
4. There appear to be some possible outliers (Magnesium, Color Intensity)

1.5 Outlier detection

As the scaled boxplot shows below, there are a number of possible outliers in a number of variables, but in particular Ash and Magnezum.

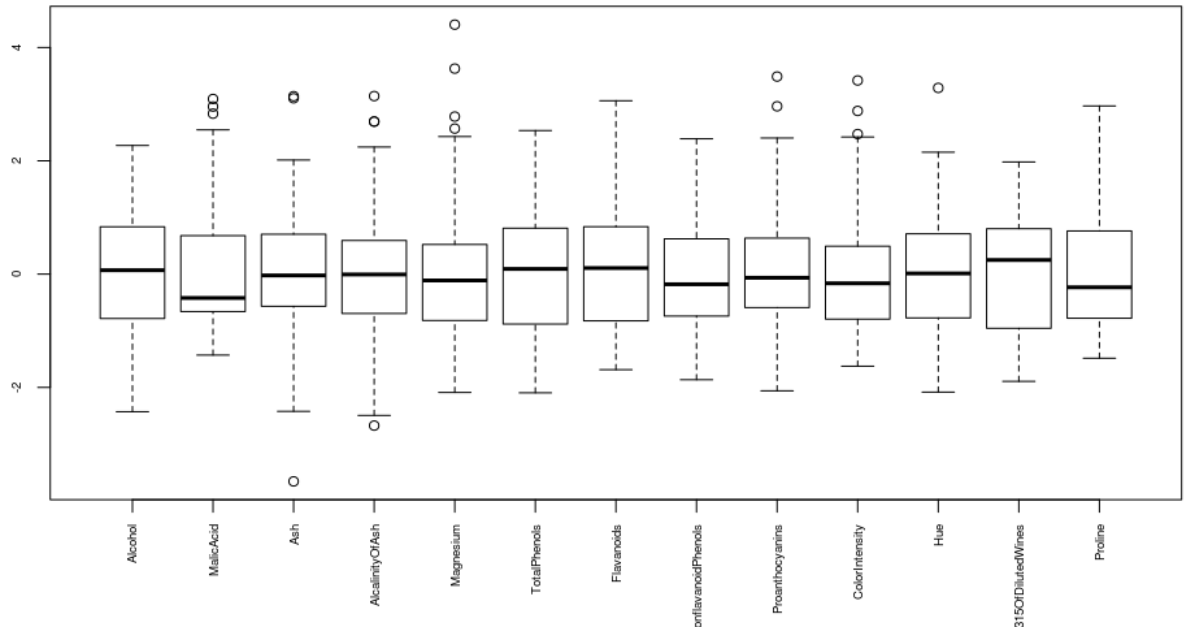


Figure 1.5: Box plot of all continuous variables

Studying the outliers for Ash and Magnesium and using 3 standard deviations as an indicator of an outlier, I found the following outliers which would warrant further investigation so as to possibly improve predictive accuracy in a modeling process:

1. Magnesium, has values of 162 and 151, has values 4.4 and 3.62 standard deviations from the mean
2. Ash has a minimum value of 1.36, which is -3.66 times from the mean. The maximum of 3.23 is 3.14 standard deviations from the mean

Using the Grubbs test for outliers, to validate the above results, I found that these outliers are confirmed by the Grubbs test at the 95% level.

1.6 Understanding the distribution of the target variable

The target variable is the only categorical variable in the dataset, although it is represented as a number (1..3). The table below shows a reasonably even distribution of classID across the 177 observations.

Class	Count	Percentage
1	58	32.77%
2	41	23.16%
3	78	44.07%
	177	100.00%

Figure 1.6: Distribution of Target Variable (ClassID)

Part 2: Exploratory Data Analysis

The next section is a natural continuation of the data quality check, however it extends the data analysis in that is begins to apply exploratory techniques to the relationships between the predictors, rather than considering them in isolation as we did in the data quality check.

2.1 Correlations

Highly correlated data within a multi-variate data set can impact the performance of some models, so it is useful to understand the correlations within the predictor data set. The diagram below shows the strength of correlation between the predictors.

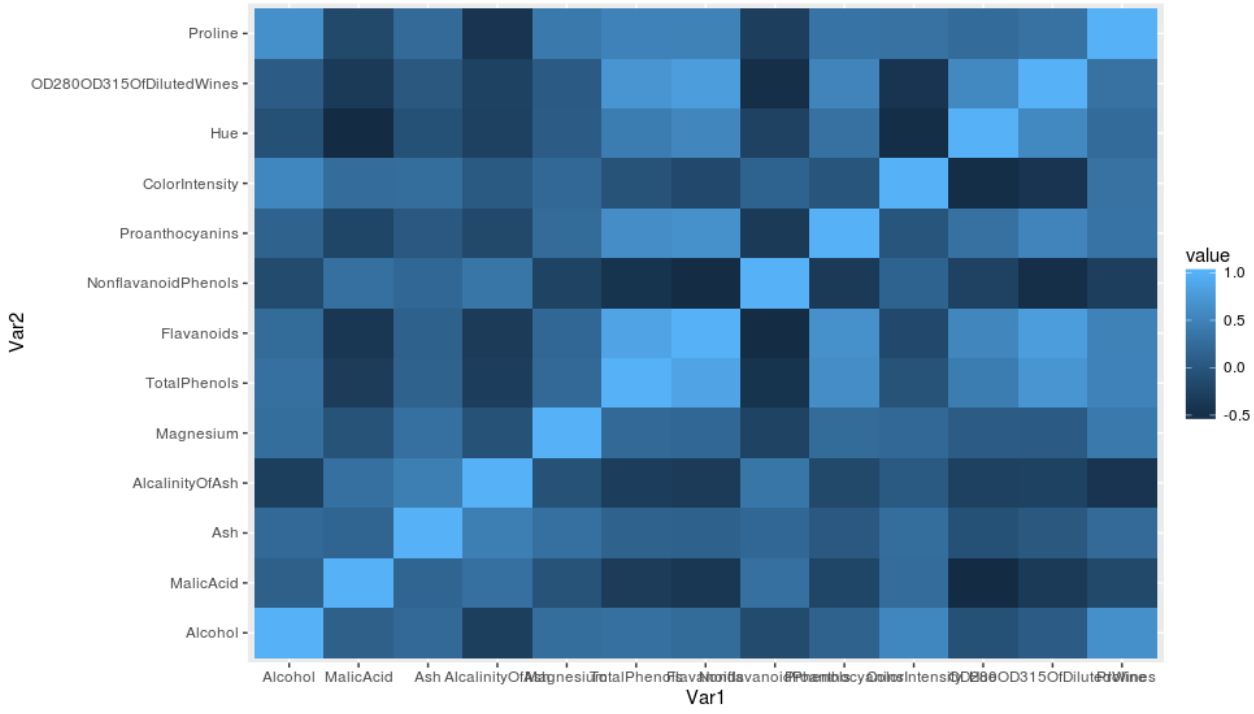


Figure 2.1: Correlation Heat map

Observations from this table of correlations are as follows:

1. There is a very strong correlation (0.86) between Total Phenols and Flavanoids. This is unsurprising given it was discovered that Flavanoids can make up 90% of the Total Phenols in wine.
2. Total Phenols and Flavanoids are in turn both very strongly correlated (0.70, 0.79 respectively) with OD280/OD315 Of Diluted Wines, as well as both reasonably strongly correlated (065, 0.61 respectively) with Proanthocyanins.
3. Alcohol is reasonably strongly correlated with Proline (0.64).
4. No other correlations are greater than 0.6 in absolute value.

Given these correlations, it may make sense to remove predictors by combining them, or reduce dimensionality in some other way such as PCA or Factor analysis.

2.1 Studying the strong correlations in more depth

Given the strong Pearson correlations found in the data, it is useful to graph these relationships to validate the strength of the correlation (this is particularly important given the low number of observations), while simultaneously looking for separability.

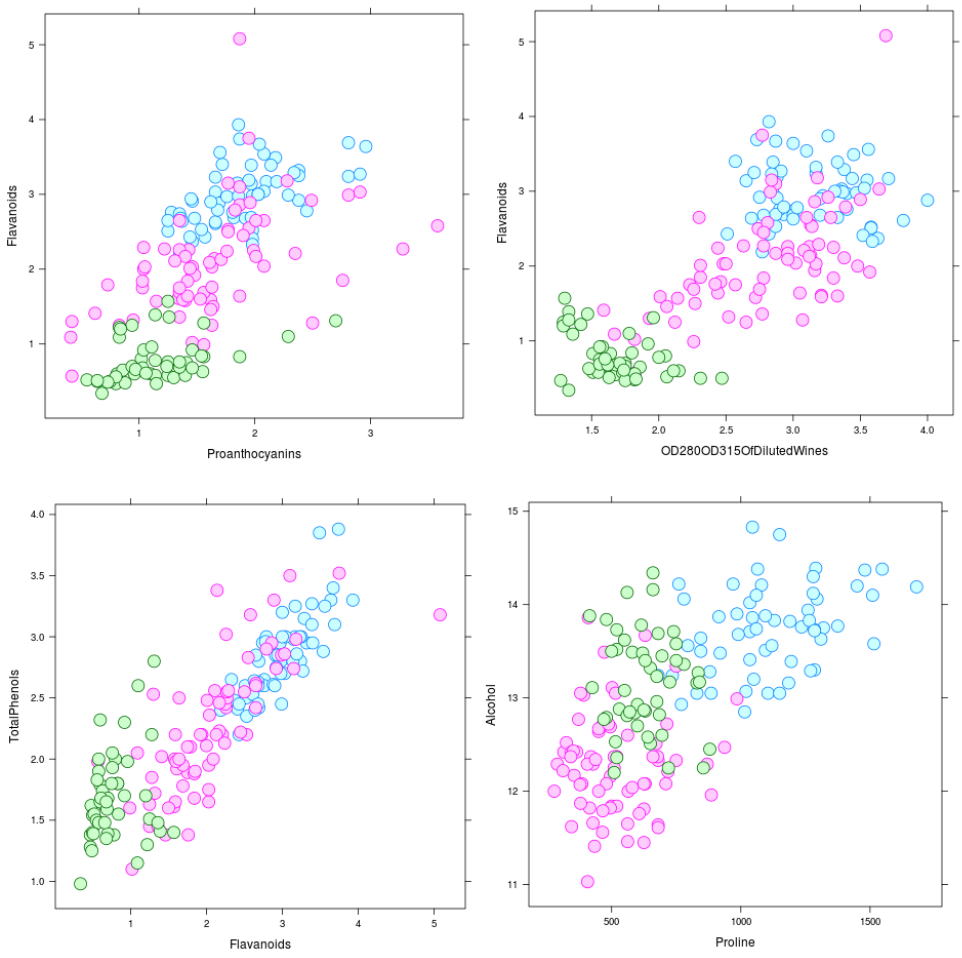


Figure 2.2: Scatter plot of strongly correlated variables, grouped by Class ID.

The scatter plots above show that the correlation values reasonably reflect the data, but moreover by grouping the points on the ClassID (represented by the coloration), we see a strong relationship, or strong separability, between the ClassID and several of the predictors.

2. 2 Conditional Distributions against the target variable

The evidence from the basic distributions and the grouped scatter plots suggests there is strong separability of the classID in the predictor variables. By studying the conditional histograms, we can see where this effect is strongest and weakest.

The graphs below show the most conditional density plots where the ClassID is most separable:

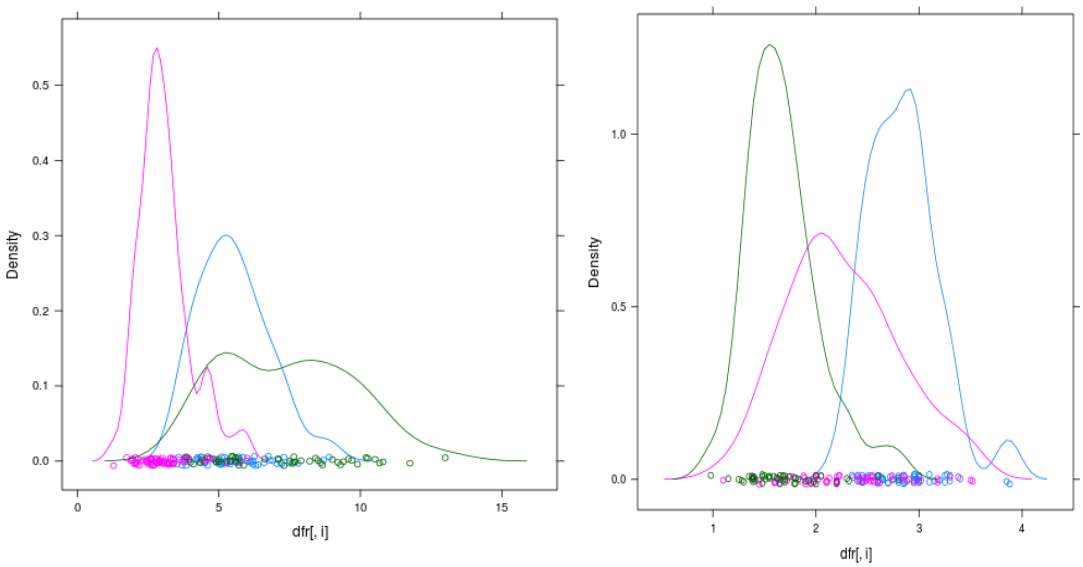


Figure 2.3: Sample of Conditional Density Plots: Color Intensity (Left), and Total Phenols (Right) grouped by Class ID.

Most of the conditional density plots which show a significant separation between the classIDs, if not as significant as the two examples above. These plots serve to underline the predictive power of these variables. Having plotted all the conditional density plots, the only variables where this technique did not show a significant difference were Ash, and Magnesium.

2.3 Fitting a tree for EDA insight

The diagram below shows the resulting of fitting a tree using a minimum split of 5:

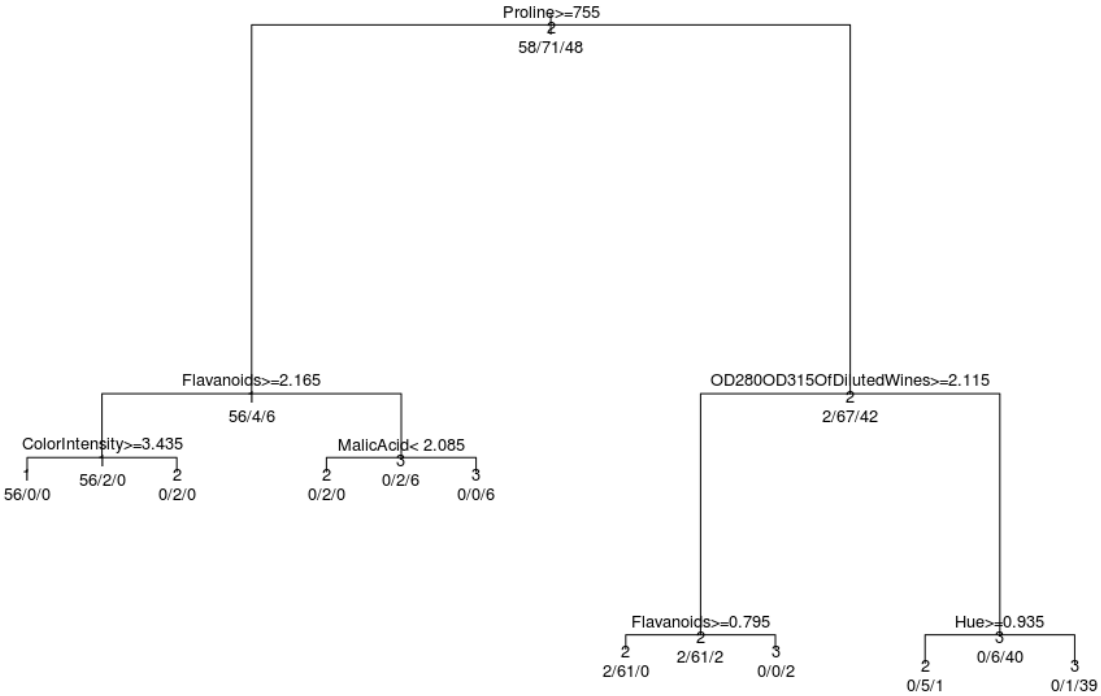


Figure 2.5: Vizualization of a simple tree fit to the Wine Data

The observations from this tree are as follows:

1. A simple tree can separate the data very quickly, this is consistent with the findings in the conditional density plots
2. Using the variable importance measures in the R package Caret, the 6 most important variable (in order of importance decending) are ColorIntensity, Flavanoids, OD280OD315OfDilutedWines, Proline, Alcohol and Hue. This is outlined in the table below

Variable	Importance Measure
ColorIntensity	101.60
Flavanoids	92.08
OD280OD315OfDilutedWines	85.99
Alcohol	67.14
Hue	48.29
Proline	44.17
TotalPhenols	10.29
Proanthocyanins	9.12
MalicAcid	7.85
AlcalinityOfAsh	7.52
Magnesium	3.86
NonflavanoidPhenols	1.36
Ash	0.00

Figure 2.5: Variable importance from Rpart tree using Caret package varImp()

Part 3: Model Based Exploratory Data Analysis

3.1 Principal Components Analysis

Doing a Principal Components analysis in order to reduce the data into its underlying structure was very instructive for the wine data. Below is a chart showing the first two principal components, with the grouping of the ClassID highlighted. The PCA has served to separate the data.

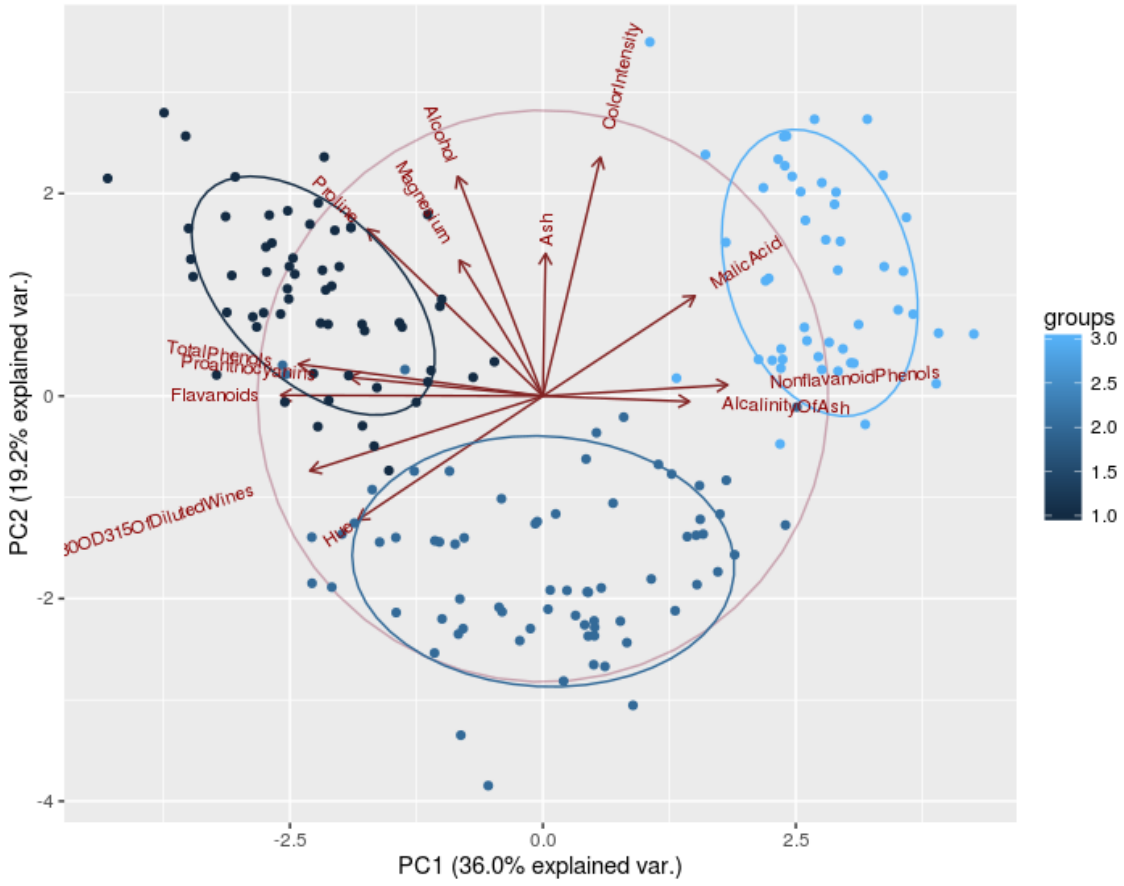


Figure 3.1: Results of PCA applied to the wine data

This indicates that dimensionality reduction might be helpful in grouping the data into their underlying classes, and this would serve to solve any unwanted multi-colinearity issues as discovered earlier in this paper.

3.2 Using Random Forests to understand variable importance

Random Forests can be used to also asses variable importance and possible interaction terms. Fitting a Random Forest with 500 trees, with a mtry of 7, yielded the following variable importance measures:

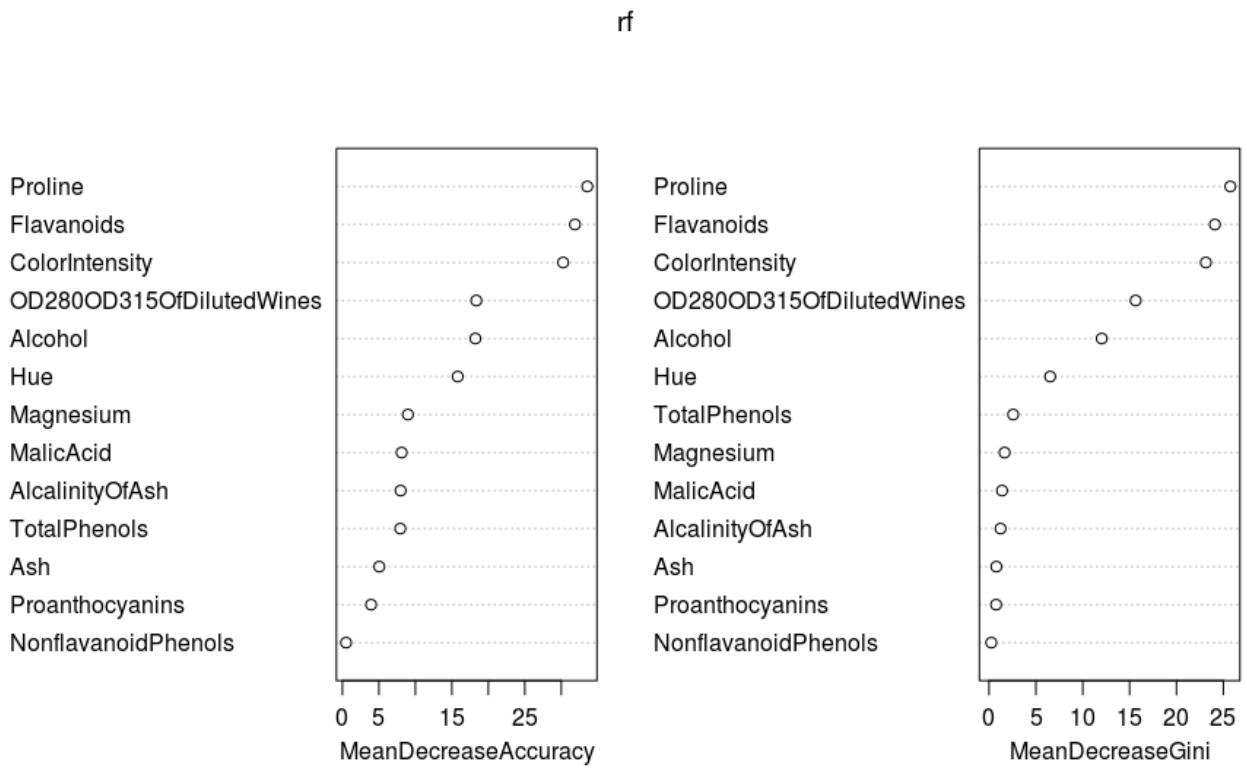


Figure 2.6: Variable importance from Random Forest

This is only roughly consistent with the findings from the tree, with the top 6 variables being the same, however the order is quite different.

Part 4: Variable Transforms

There are no missing variables in the data set, and generally the outliers cannot be discounted as being unreasonable, so therefore there is no data imputation required.

Unlike a linear regression, there are no assumptions about the linearity of the relationship between the predictors and the target variable, so transforms for linearity will not be required.

Part 5: Choosing the performance metric

There are a number of metrics which could be used to determine the best model. In a mutliclass binary classification problem, some of those available are:

1. Accuracy, or how accurate the model across positive and negative predictions.
2. Precision, the proportion of true positives over all positive predictions
3. Sensitivity or Recall, which measures the positive outcomes correctly identified as such
4. Specificity, the proportion of true negatives over all negative predictions

For this business problem, I think the best metric to use here is simply accuracy, there is not necessarily a significant cost difference to each of the measures described above.

Therefore the best measure to determine the best model will be accuracy, or performance across positive and negative predictions.

Part 6: Modeling

For this section, I split the data (70/30) where the model did not offer an unbiased way of testing the model performance such as the OOB performance for Random Forests.

6.1 Baseline model: mutliclass logistic model with all variables

The initial baseline model provides a point of reference for the performance of all the future models. I chose the multi-nomial logistic regression for this purpose.

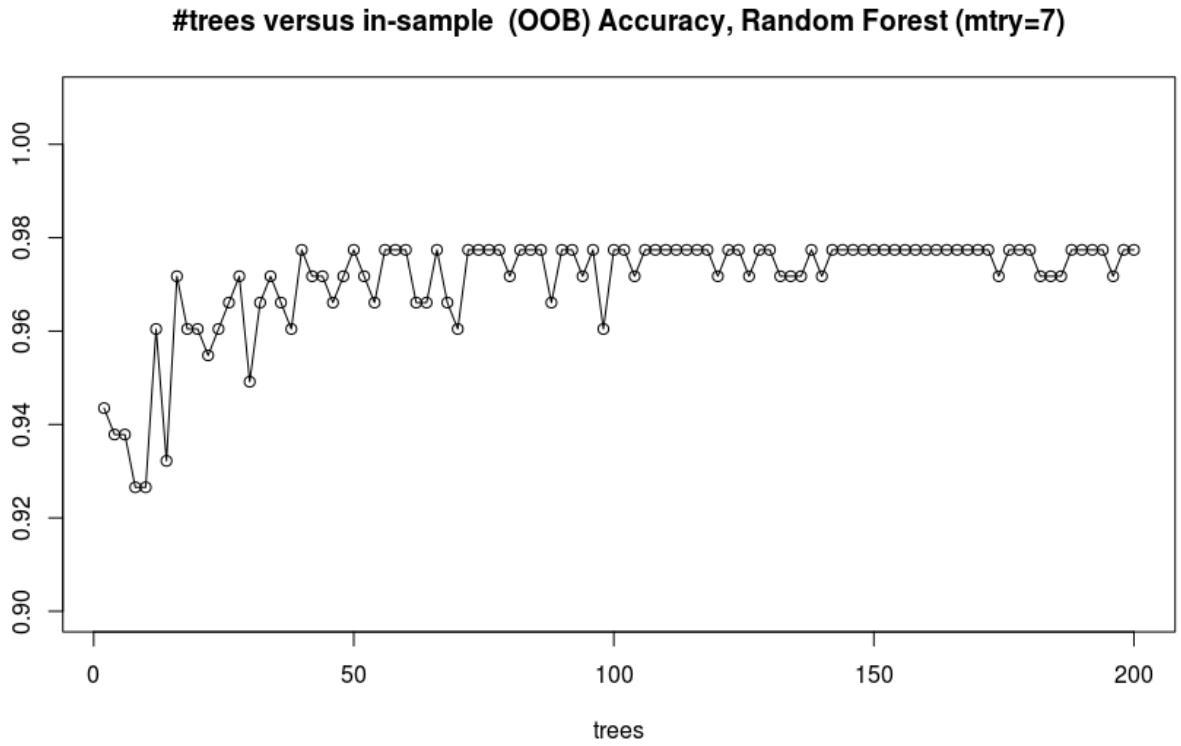
The model coefficients for the model are as follows:

Class	**Coeffcients:**				
	(Intercept)	Alcohol	MalicAcid	Ash	NonflavanoidPhenols
2	347.23042	-11.773425	-31.48552	-93.88314	165.317
3	-19.39829	-8.397173	-10.93497	14.54516	-142.9178
	ColorIntensity	Hue	OD280OD315OfDiluted	Proline	Proanthocyanins
2	-2.85984	143.7951	-27.15479	-0.5886634	75.424499
3	28.78517	-108.6522	-42.65685	-0.3408921	-4.389097
	AlcalinityOfAsh	Magnesium	TotalPhenols	Flavanoids	
2	12.58563	-0.10716004	65.59139	-42.27581	
3	16.70743	-0.06889364	159.86811	-102.9413	

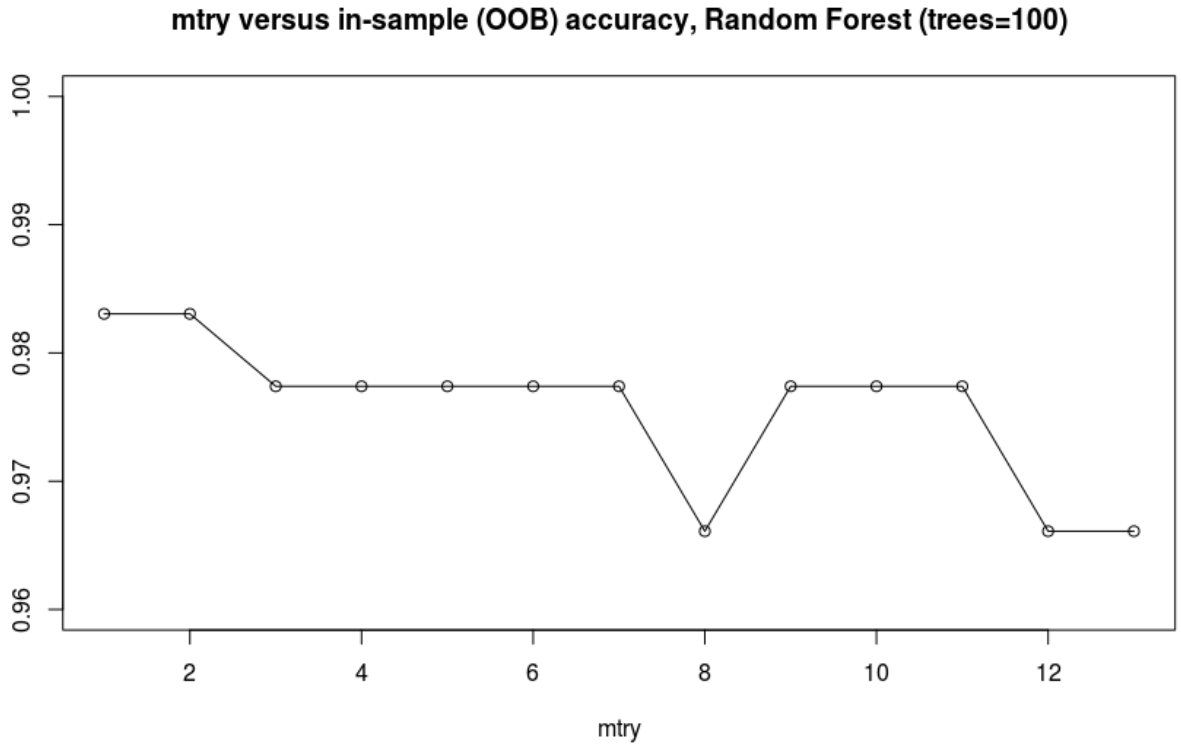
Note that only 2 classes are shown as the first class (1) is the baseline against the other two models for classes (2,3). The out-of-sample performance was 0.9434 and the AIC was 56.000.

6.2 Fitting a Random Forest

Below are the results of fitting Random Forests, varying both mtry (the number of predictors in each tree) and the number of trees.



The accuracy stabilizes at approximately 100 trees, so this is the setting I used going forward.



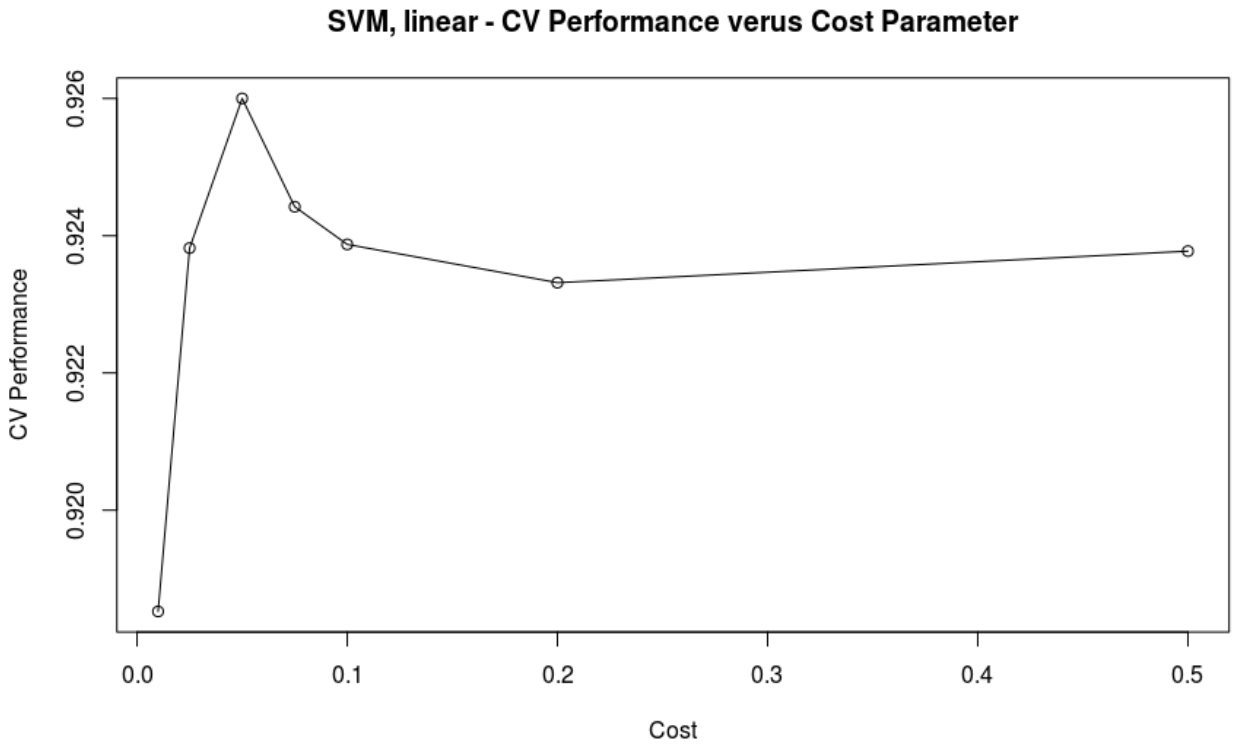
The best mtry value, surprisingly, was when the number of variables was equal to 1 or 2. I used the value 2 for the “best” Random Forest candidate model.

The best Random Forest model performed, using the out-of-bag measure, at an accuracy of 98.3%.

6.3 Fitting a support vector machine

I fitted a number of SVM types (linear, radial, polynomial, sigmoid). The scaled linear model appeared to perform equally with the Radial and Sigmoid kernels, so in the interest of simplicity I pursued the linear kernel type of SVM,

Below is the cross-validation performance of the SVM, varying the cost parameter.

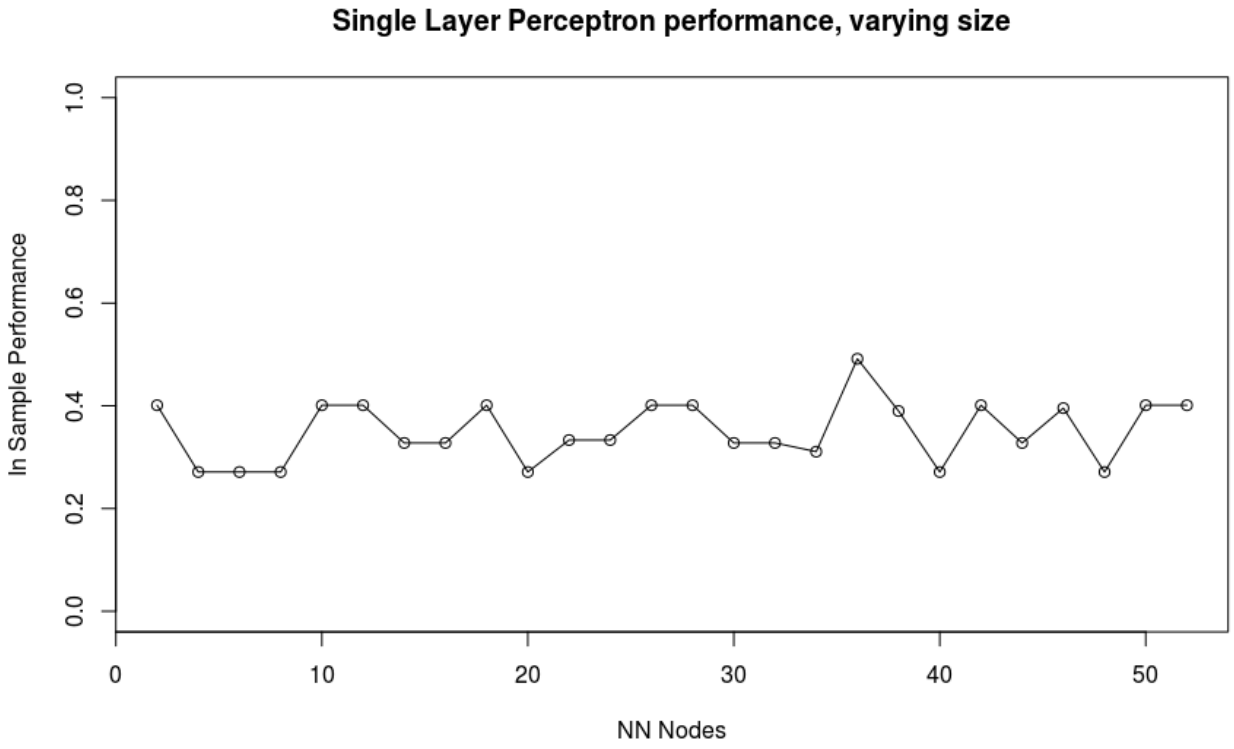


So the final SVM was a linear kernel, scaled, with a cost of 0.05, which minimized the CV error. This model performed with an accuracy of 0.9811321 out of sample.

6.4 Training a Neural Network

a) Single Layer Perceptron

The first task here is to try a single layer perceptron model, the simplest type of Neural Net. The performance of this model was poor. The in-sample performance, varying the size of the single layer, is below:



This type of model performs well below the baseline, with an average performance of 0.4, so is therefore inappropriate for this problem.

a) Multi-Layer Perceptron

The next type of Neural Network to apply is a multi-layer perceptron. I varied the number of hidden nodes in the first and second hidden layers, while keeping the number of ensemble members to fit constant at 15.

I tried the following combinations of Network construction:

- Varying Hidden1, hidden2 = 0
- Varying Hidden1 and hidden2 together,
- Hidden2 high, hidden 1 low
- Hidden1 high, hidden 2 low-import

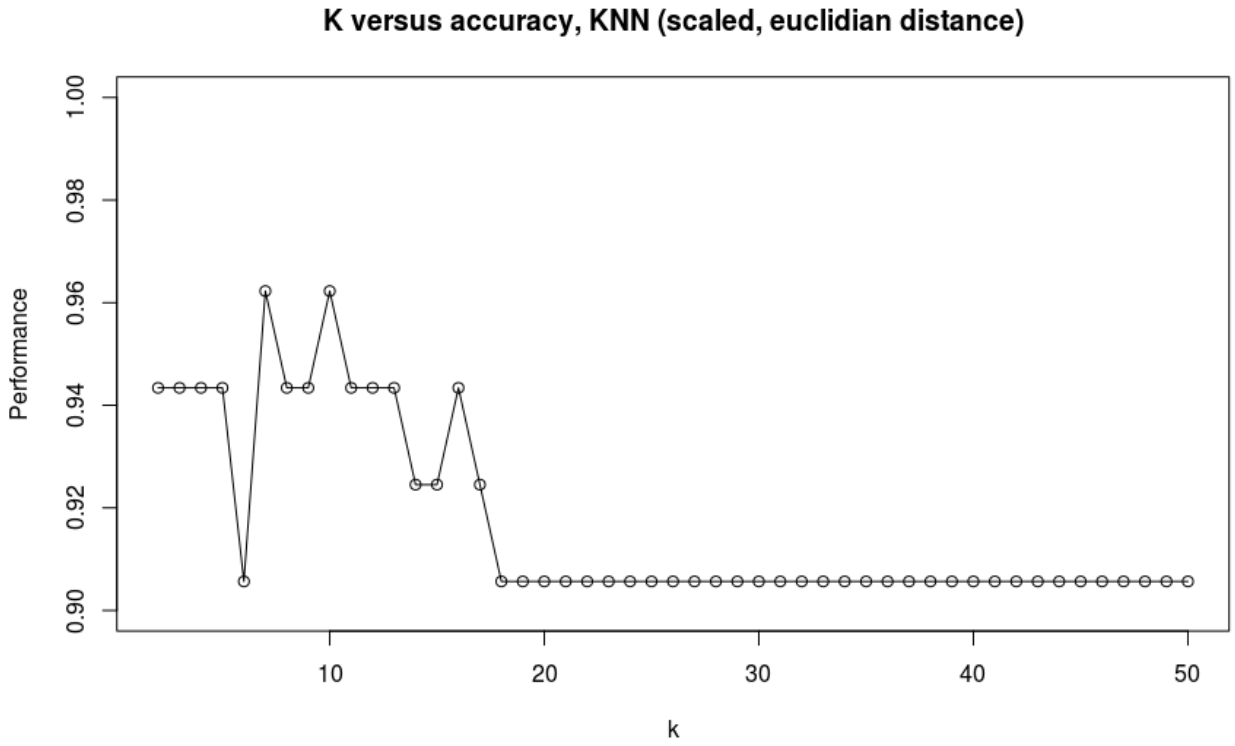
The performance results from this exercise are below:

hidden2	hidden1	ensemble	out of sample performance
0	3	15	0.9245
0	5	15	0.9433
0	10	15	0.962264
0	20	15	0.962264
0	40	15	0.981132
0	50	15	0.981132
3	3	15	0.981132
5	5	15	0.962264
10	10	15	0.439623
3	5	15	0.962264
3	8	15	0.962264
8	3	15	0.9433

As can be seen from the table above, the highest performing and simplest multi-layer perceptron model is when there are tow layers of 3 nodes.

6.5 Fitting a KNN model

The graph below shows the results of fitting a KNN model on the scaled predictors.



The best performance is when k=7 or 10, and the out-of-sample performance was 0.9622642

Part 7: Model Comparison and Conclusion

7.1 Model Comparison

The final model comparison for all models developed is shown below, ordered in descending order of performance.

Model Type	Number of Variables	Accuracy	Position
Best Random Forest, mtry=2, ntree=100	14	0.983	1
SVM, Scaled, Sigmoid Kernel, cost=10	14	0.981	2
SVM, Scaled, Radial Kernel, cost=10, gamma = 0.01		0.981	2
SVM, Scaled, Radial Kernel, cost=10	14	0.981	2
SVM, Scaled, linear Kernel, cost=10	14	0.981	2
Neural Net, Multilayer, [3,3]	14	0.981	2
Best KNN, k=7, scaled variables, Euclidian distance	14	0.962	7
SVM, Scaled, Polynomial Kernel, cost=10	14	0.962	7
Baseline multi-nomial logistic regression	14	0.943	9
SVM, Unscaled, linear Kernel, cost=10	14	0.925	10
Neural Net, Single Layer Perceptron	14	0.401	11

The Random Forest was the best performing, followed equally by the Support Vector Machine models and the best performing Neural Net.

7.2 Conclusion

The best performing model was a tuned Random Forest model, although the Support Vector Machine and Neural Net gave near comparable performance. KNN and the baseline multi-nomial regression also performed well.

The particularly interesting aspects of this project are the separability of the predictor variables, and that the dataset lends itself well to non-linear modeling such as Neural Networks.

There was little opportunity or value in variable transformation, outside of scaling the variables where appropriate, as the models did not depend on a linear relationship between the independent and the target variable, and there was no missing variables or imputation required.

However model tuning was a big part of this project, with the Random Forest and SVM performance able to be improved significantly with tuning.

Sources

“Rapid method for proline determination in grape juice and wine.” https://www.ncbi.nlm.nih.gov/pubmed/22480274

“Phehnolic Content in Wine” Wikipedia : https://en.wikipedia.org/wiki/Phenolic_content_in_wine

“Wine is Made of” - Wine Education : http://www.wineeducation.com/wineismadeof.html

“Malic Acid” - Cal Wineries : http://www.calwineries.com/learn/wine-chemistry/wine-acids/malic-acid

Appendix: R-Code

Some notable R-code developed in the course of this assigment, note this is the modeling code as the EDA was included in assignment 1.

```
#MODELING

library("randomForest")
str(df)
df$classID <- as.factor(df$classID)
rf = randomForest(classID ~ ., data = df, mtry = 7, ntree = 500, importance =
T)
importance(rf,type=1)
varImpPlot(rf)
partialPlot(rf, df, "Alcohol" )
partialPlot(rf, df, "MalicAcid" )
partialPlot(rf, df, "OD280OD315OfDilutedWines" )

#trying to find interaction
library(plotmo)
plotmo(rf)

#actual models 1, RF
library("randomForest")
str(df)
df$classID <- as.factor(df$classID)
df$scored.class <- NULL
df$class <- NULL
df2<-df

#model: RF (tuning tree number)
results = matrix(nrow=100,ncol=2)
n<-1
for(tr in seq(from=2, to=200, by=2))
{
  rf = randomForest(classID ~ ., data = df, mtry = 7, ntree = tr, importance =
T)

  df2$scored.class <- as.character(rf$predicted)
  df2$class <- as.character(df$classID)
  results[n,1] <- tr
  results[n,2] <- FAccuracy_multiclass(df2)
  n<-n+1
}

plot(results, xlab="trees", ylab="", main="#trees versus in-sample  (OOB)
Accuracy, Random Forest (mtry=7)", ylim=c(0.9,1.01))
lines(results)
```

```
#model: RF (tuning mtry number)
results = matrix(nrow=13,ncol=3)
n<-1
df$scored.class <- NULL
df$class <- NULL
df2<-df

for(mt in c(1,2,3,4,5,6,7,8,9,10,11,12,13))
{
  rf = randomForest(classID ~ ., data = df, mtry = mt, ntree = 100, importance
= T)
  df2$scored.class <- as.character(rf$predicted)
  df2$class <- as.character(df$classID)
  results[n,1] <- mt
  results[n,2] <- FAccuracy_multiclass(df2)
  n<-n+1
}

plot(results, xlab="mtry", ylab="", main="mtry versus in-sample (OOB) accuracy,
Random Forest (trees=100)", ylim=c(0.96,1))
lines(results)

#best RF here
rf = randomForest(classID ~ ., data = df, mtry = 1, ntree = 100, importance =
T)
1-rf$err.rate[50]

#2 model: knn, note for report: scaling improves perofrmance significantly,
presumably because it distorts distance less
library(class)
results = matrix(nrow=50,ncol=2)
for(kn in 2:50)
{
  predknn <- knn(scale(df[training_index,2:14]), scale(df[-
training_index,2:14]), df[training_index,1], k = kn)
  df3$scored.class <- predknn
  results[kn,1] <- kn
  results[kn,2] <- FAccuracy_multiclass(df3)
}

plot(results, xlab="k", ylab="Performance", main="K versus accuracy, KNN
(scaled, euclidian distance)", ylim=c(0.90,1))
lines(results)

#3 MODEL -------- SVM

#set up training/test
training_index <- sample(nrow(df), round(nrow(df)*0.7))
df3 <- df[-training_index,]
df3$class <- df3$classID

#model: support vector machine
library(e1071)
c <- 10
svmmodel <- svm(classID ~ ., data = df[training_index,], kernel="linear", cost
= c, scale=FALSE)
svmpred <- predict(svmmodel, df[-training_index,])
df3$scored.class <- svmpred
q <- FAccuracy_multiclass(df3)
q

svmmodel2 <- svm(classID ~ ., data = df[training_index,], kernel="linear", cost
= c, scale=TRUE)
```

```
svmpred2 <- predict(svmmodel2,  df[-training_index,])
df3$scored.class <- svmpred2
q <- FAccuracy_multiclass(df3)
q

svmmodel3 <- svm(classID ~ ., data = df[training_index,], kernel="polynomial",
cost = c, scale=TRUE)
svmpred3 <- predict(svmmodel3,  df[-training_index,])
df3$scored.class <- svmpred3
q <- FAccuracy_multiclass(df3)
q

svmmodel4 <- svm(classID ~ ., data = df[training_index,], kernel="radial", cost
= c, scale=TRUE)
svmpred4 <- predict(svmmodel4,  df[-training_index,])
df3$scored.class <- svmpred4
q <- FAccuracy_multiclass(df3)
q

svmmodel5 <- svm(classID ~ ., data = df[training_index,], kernel="sigmoid",
cost = c, scale=TRUE)
svmpred5 <- predict(svmmodel5,  df[-training_index,])
df3$scored.class <- svmpred5
q <- FAccuracy_multiclass(df3)
q

#using the internal cross validation of SVM
svmtune <- tune(svm,as.numeric(classID) ~ ., data = df, kernel="linear", ranges
= list(cost=c(0.01, 0.025, 0.05, 0.075, 0.1,0.2, 0.5)))

graphSVM <- summary(svmtune)$performances
plot(graphSVM[,1],1-graphSVM[,2] , xlab="Cost", ylab="CV Performance",
main="SVM, linear - CV Performance verus Cost Parameter")
lines(graphSVM[,1],1-graphSVM[,2])
graphSVM

#best model
results = matrix(nrow=15,ncol=2)
n<-1
for(co in c(0.01, 0.025, 0.05, 0.075, 0.1,0.2, 0.5,1))
{
  df$classID <- as.factor(df$classID)
  svmmodel6 <- svm(classID ~ ., data = df[training_index,], kernel="linear",
cost = co, scale=TRUE)
  svmpred6 <- predict(svmmodel6,  df[-training_index,])
  df3$scored.class <- svmpred6
  results[n,1] <- co
  results[n,2] <- FAccuracy_multiclass(df3)
  n<-n+1
}

plot(results[,1],results[,2] , xlab="Cost", ylab="CV Performance", main="SVM,
linear - CV Performance verus Cost Parameter")
lines(results[,1],results[,2])
results

df$classID <- as.factor(df$classID)
svmmodel6 <- svm(classID ~ ., data = df[training_index,], kernel="linear", cost
= 10, scale=TRUE)
svmpred6 <- predict(svmmodel6,  df[-training_index,])
df3$scored.class <- svmpred6
q <- FAccuracy_multiclass(df3)
q
```

```
#baseline model
baseline <- multinom(classID ~ ., data = df[training_index,])
write.csv(baseline$C)
pred10 <- predict(baseline,df[-training_index,])
df3$scored.class <- pred10
q <- FAccuracy_multiclass(df3)
q
baseline

#finally, Neural Network
library(neuralnet)
library(nnet)
library(RSNNS)

target <- class.ind(df$classID)
#single layer perceptron
#train the neural net

results <- matrix(nrow=27, ncol=2)
n<-1
for(s in seq(from=2, to=54, by=2))
{
NN <- nnet(df[,-1], target, size=s, softmax=TRUE)
#predict
pred <- predict(NN, df[,-55], type="class")
df2$scored.class <- pred
results[n,1]<-s
results[n,2]<- FAccuracy_multiclass(df2)
n<-n+1
}

plot(results[,1],results[,2], ylim=c(0,1), ylab = "In Sample Performance",
xlab="NN Nodes", main="Single Layer Perceptron performance, varying size")
lines(results[,1],results[,2])

#now using a multi-layer perceptron in order to see if we get better results
#MPL

library(monmlp)
df4<-df
#out of sample
df5 <- df4[training_index,]
df3 <- df4[-training_index,]

matrix4 <- data.matrix(df4[training_index,-1], rownames.force = NA)
matrix5 <- data.matrix(df4[training_index,1])
matrix6 <- data.matrix(df4[-training_index,-1])
class(matrix5)<-"numeric"

r <- monmlp.fit(matrix4,matrix5, hidden1=3,hidden2 = 8, n.ensemble=15,
monotone=1, bag=TRUE)
z <- monmlp.predict(x = matrix6, weights = r)
z2 <- round(z)

df3$class <- df3$classID
df3$scored.class <- z2
q <- FAccuracy_multiclass(df3)
q
```