Predictive Modeling in Multiclass Classification

Steven Futter

2/13/2017

In this paper I fit a model suite of the following models: Random Forest, a Support Vector Machine, and a Neural Network model. The data is the 'Wine' data set that can be downloaded from the Irvine Machine Learning Repository. Since this is a small data set I only focus on fitting models in-sample.

(1) Data Quality Check

The wine data set consists of 178 observations and 14 variables. Of the 14 variables there is one class variable and 13 continuous variables that describe the profile of the wine. In this assignment we are dealing with a classification problem since the goal of the assignment is to predict which class a wine falls into based upon the continuous variables, such as alcohol level, magnesium content, and color intensity.

Missing Data, Data Ranges and Distributions

Of the 178 observations the larger portion of wines belong to class 2 (40%), followed by class 1 (33%), and then class 3 (27%). The my.summary() function below confirms that there are no missing observations in the wine data. Based upon the mean and median values the proline variable appears to have a non-normal distribution sine mean and median values differ relative to other variables in the data set. The mean and median values of the proline variable are noticeable different. The mean value of proline is 746.89 whereas the median value is 673.50. Other variables in the data set appear to have extreme observations appearing as minimum and maximum values. There are no negative minimum values which makes sense and although there are outliers the values appear to be valid at first glance.

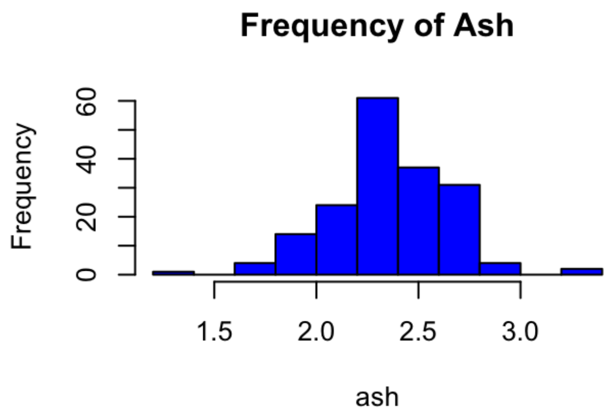
```
my.summary(wineData[,c(2:14)])
```

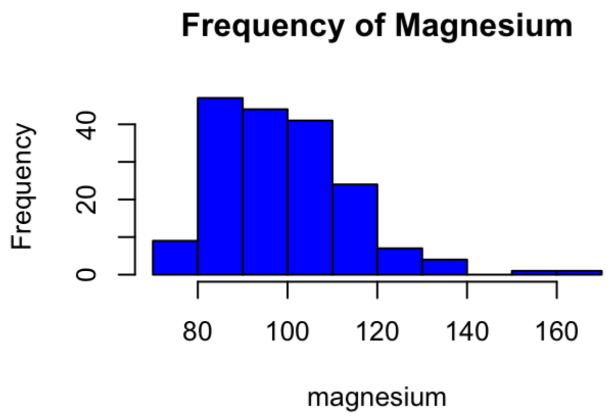
```
##            alcohol malicAcid  ash alcalinityOfAsh magnesium totalPhenols
## max          14.83      5.80 3.23           30.00    162.00         3.88
## min          11.03      0.74 1.36           10.60     70.00         0.98
## 0.01         11.44      0.90 1.70           11.35     78.00         1.14
## 0.05         11.66      1.06 1.92           14.77     80.85         1.38
## 0.25         12.36      1.60 2.21           17.20     88.00         1.74
## 0.5          13.05      1.87 2.36           19.50     98.00         2.36
## 0.75         13.68      3.08 2.56           21.50    107.00         2.80
## 0.95         14.22      4.46 2.74           25.00    124.30         3.27
## 0.99         14.47      5.54 2.99           28.50    141.76         3.60
## mean         13.00      2.34 2.37           19.49     99.74         2.30
## variance      0.66      1.25 0.08           11.15    203.99         0.39
## % missing     0.00      0.00 0.00            0.00      0.00         0.00
##            flavanoids nonflavanoidPhenols proantocyanins colorIntensity
## max              5.08                0.66            3.58          13.00
## min              0.34                0.13            0.41           1.28
## 0.01             0.47                0.14            0.42           1.86
## 0.05             0.55                0.19            0.73           2.11
## 0.25             1.20                0.27            1.25           3.22
## 0.5              2.13                0.34            1.56           4.69
## 0.75             2.88                0.44            1.95           6.20
## 0.95             3.50                0.60            2.71           9.60
## 0.99             3.79                0.63            3.03          11.02
## mean             2.03                0.36            1.59           5.06
## variance         1.00                0.02            0.33           5.37
## % missing        0.00                0.00            0.00           0.00
##             hue OD280_OD315OfDilutedWines  proline
## max        1.71                      4.00  1680.00
## min        0.48                      1.27   278.00
## 0.01       0.55                      1.29   306.94
## 0.05       0.57                      1.46   354.55
## 0.25       0.78                      1.94   500.50
## 0.5        0.96                      2.78   673.50
## 0.75       1.12                      3.17   985.00
## 0.95       1.28                      3.58  1297.25
## 0.99       1.43                      3.84  1522.36
## mean       0.96                      2.61   746.89
## variance   0.05                      0.50 99166.72
## % missing  0.00                      0.00     0.00
```

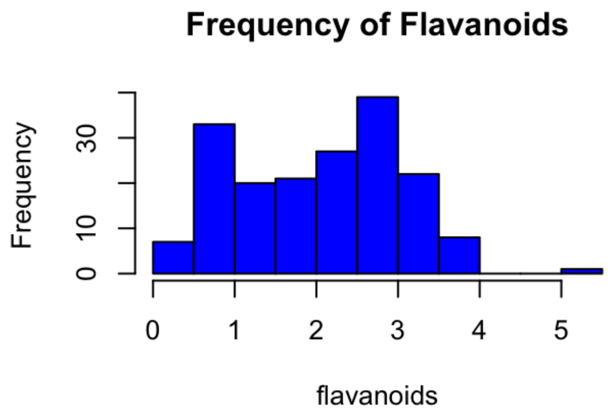
By plotting a matrix of histograms we can visually identify clear outliers in the data set that are described above by the my.summary() function. Outliers can be seen in the following variables: ash, magnesium, flavanoids, hue.

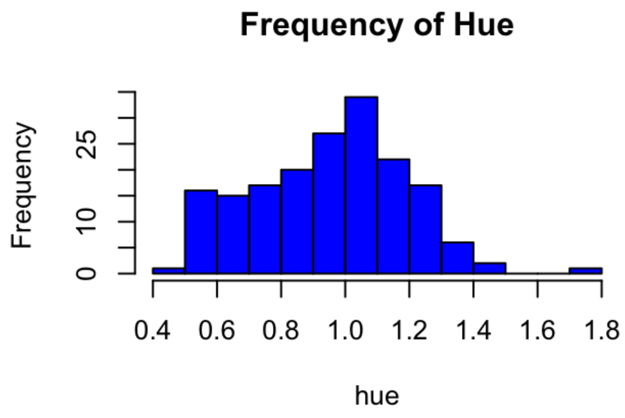
Outliers: i. left tail of ash ii. right tail of ash iii. right tail of magnesium iv. right tail of flavanoids v. right tail of hue

Outliers





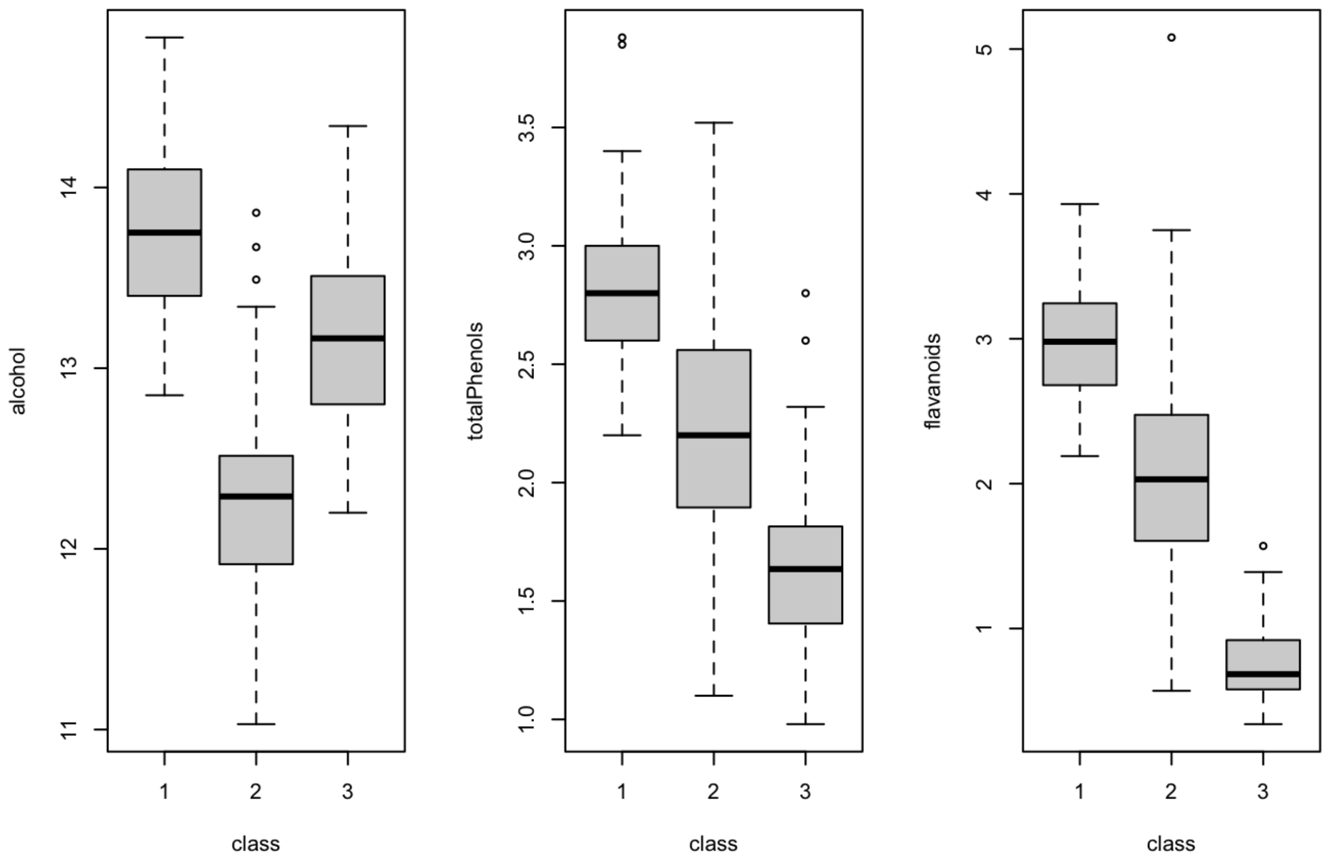




(2) Exploratory Data Analysis

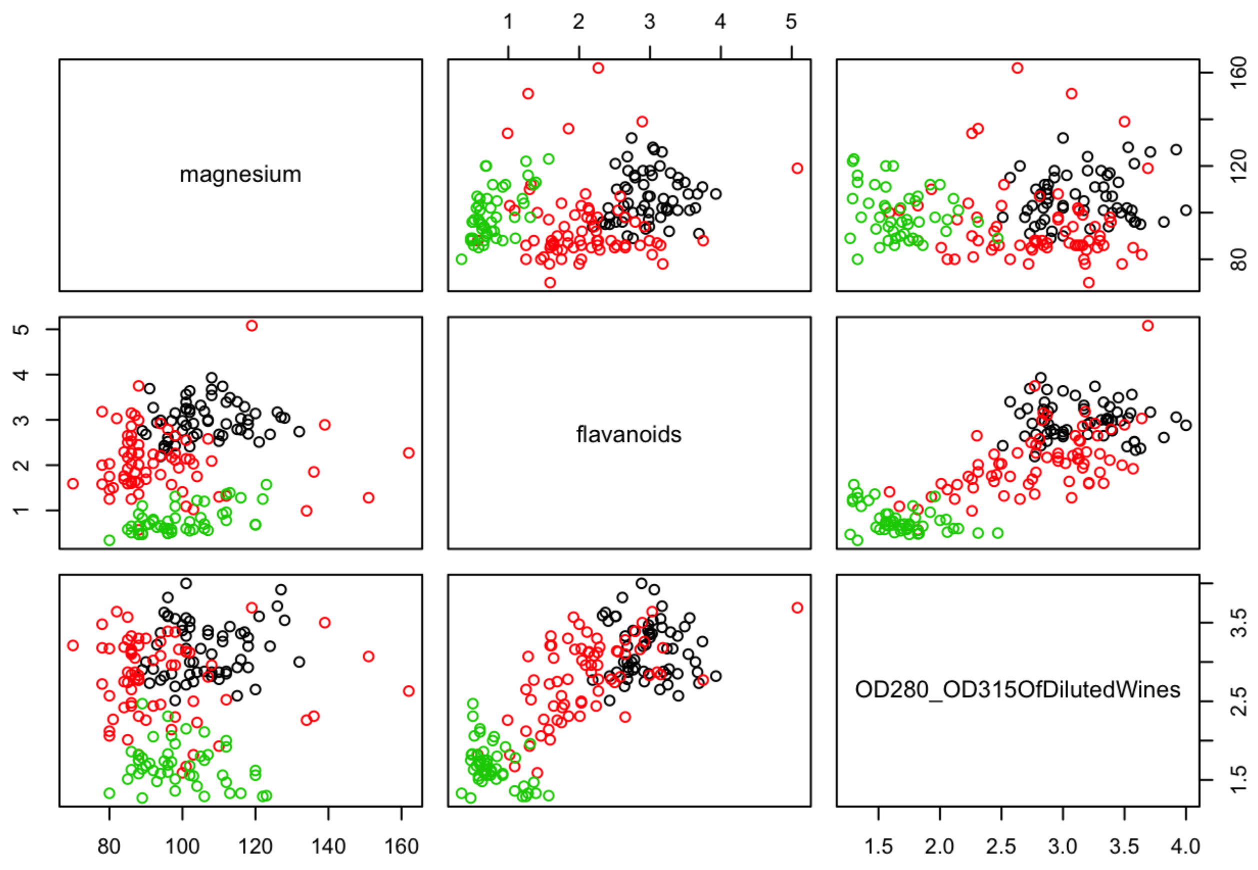
With the wine data set we have a classification problem which can benefit from viewing the data distributions with boxplots. From the boxplots below we can see that alcohol, totalPhenols, and flavanoids appear to be particularly predictive with the middle 75% (grey shaded bar) of values separating the three wine classes with minimal overlap. This looks promising since the wine classes are differentiated quite nicely by these three variables alone.

Boxplots: Alcohol, totalPhenols, and flavanoids differentiated across wine classes

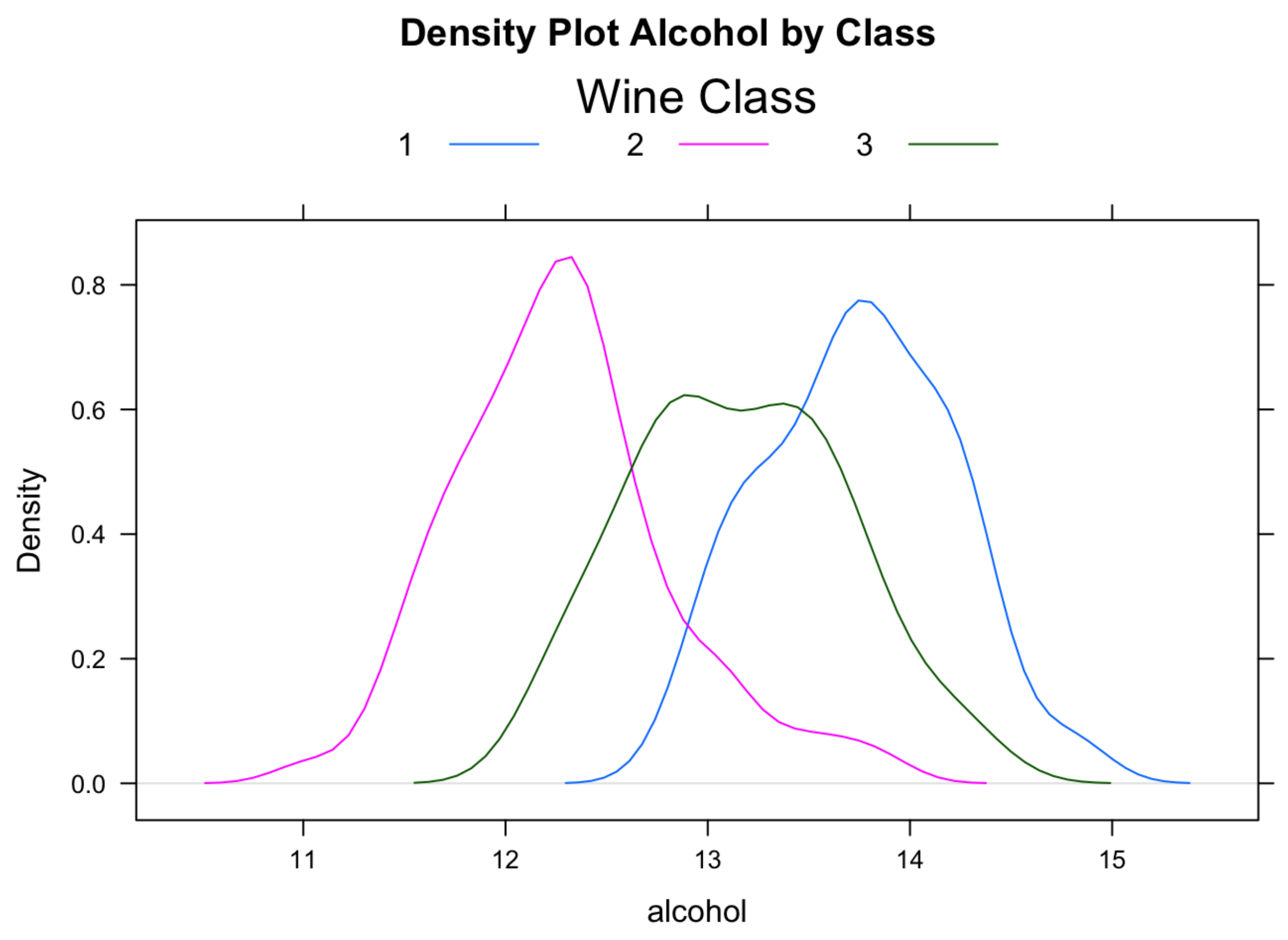


Several of the variables appear to be good candidates for a descriminative classifier such as a Support Vector Machine (SVM). An SVM uses an algorithm to split the data classes into optimal hyperplanes. Below I show that various pairs of variables cluster the wine classes into more or less separate wine classes. From initial EDA it appears that a clustered algorithm approach to modeling may also have a low misclassification error percentage. Below we can see how magnesium, flavanoids, and OD280_OD315OfDilutedWines pairs segregate the three wine classes into separate clusters.

Scatterplots: magnesium, flavanoids, and OD280_OD315OfDilutedWines pairs segregate the wine classes



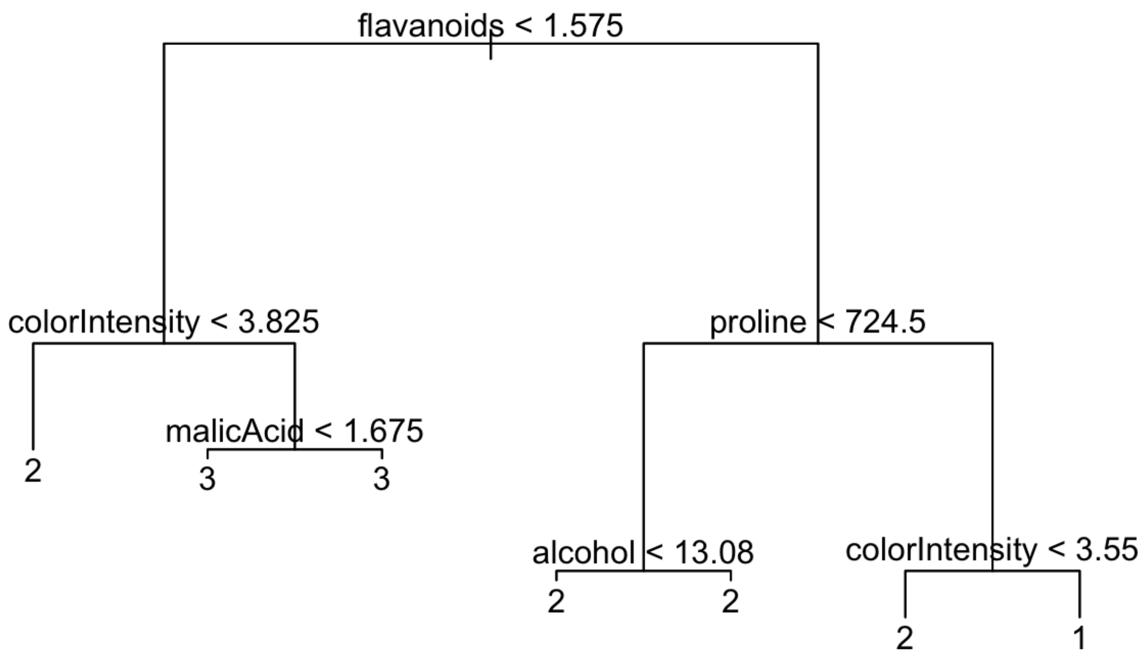
Using the Lattice library we can see the density function of each wine class against alcohol. We can see that the peak densities of each wine class vary as alcohol concentration increases. Generally speaking we can see that class 2 wines have the least alcohol content and class 1 wines have the most.



I now use a tree model to explore the data further and evaluate missclassification error.

Tree Model

```
##
## Classification tree:
## tree(formula = class ~ ., data = wineData)
## Variables actually used in tree construction:
## [1] "flavanoids"     "colorIntensity" "malicAcid"      "proline"
## [5] "alcohol"
## Number of terminal nodes:  7
## Residual mean deviance:  0.08779 = 15.01 / 171
## Misclassification error rate: 0.01685 = 3 / 178
```



The tree model above has a misclassification of 1.7%. flavanoids, proline, colorIntensity, malicAcid, and alcohol are important variables in determining the wine's class. Wine class '3' is segregated along the left branch with flavanoid values less than 1.575 and color intensity greater than 3.825. Class 1 and 2 are split either side of the first tree root.

Fitted Model Suite:

1. Random Forest

Firstly, let's create a random forest model taking in all variables within the wine data set. I write them out in the model to make it explicit which variables are being included into the random forest model.

```

Call:
 randomForest(formula = class ~ alcohol + malicAcid + ash + alcalinityOfAsh + 
magnesium + totalPhenols + flavanoids + nonflavanoidPhenols +      proanthocyanins 
+ colorIntensity + hue + OD280_OD315OfDilutedWines +      proline, data = wineData, 
mtry = 2, importance = TRUE) 
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 1.12%
Confusion matrix:
   1  2  3 class.error
1 59  0  0  0.00000000
2  1 69  1  0.02816901
3  0  0 48  0.00000000
```

0% missclassification can be seen in the confusion matrix table.

```

randomForest.pred  1  2  3
                1 59  0  0
                2  0 71  0
                3  0  0 48
```

2. Support Vector Machine

Let’s now run a SVM model and evaluate the confusion matrix below. Similarly to the Random Forest model the SVM produces a 0% misclassification error. See the confusion matrix table below.

```


Call:
svm.default(x = x, y = y, data = wineData)


Parameters:
   SVM-Type:  C-classification 
 SVM-Kernel:  radial 
       cost:  1 
      gamma:  0.07692308 

Number of Support Vectors:  69

 ( 19 31 19 )


Number of Classes:  3 

Levels: 
 1 2 3
```

```
        y
svm.pred  1  2  3
       1 59  0  0
       2  0 71  0
       3  0  0 48
```

3. Neural Network Model

Finally, I run a neural network model using the nnet library. Again, I produce a confusion matrix to evaluate the model’s performance.

```
# weights:  428
initial  value 213.223719
iter  10 value 136.377871
iter  20 value 95.624574
iter  30 value 26.037956
iter  40 value 21.125682
iter  50 value 20.389365
iter  60 value 19.173270
iter  70 value 15.030089
iter  80 value 12.598054
iter  90 value 11.187454
iter 100 value 11.138673
final  value 11.138673
stopped after 100 iterations
```

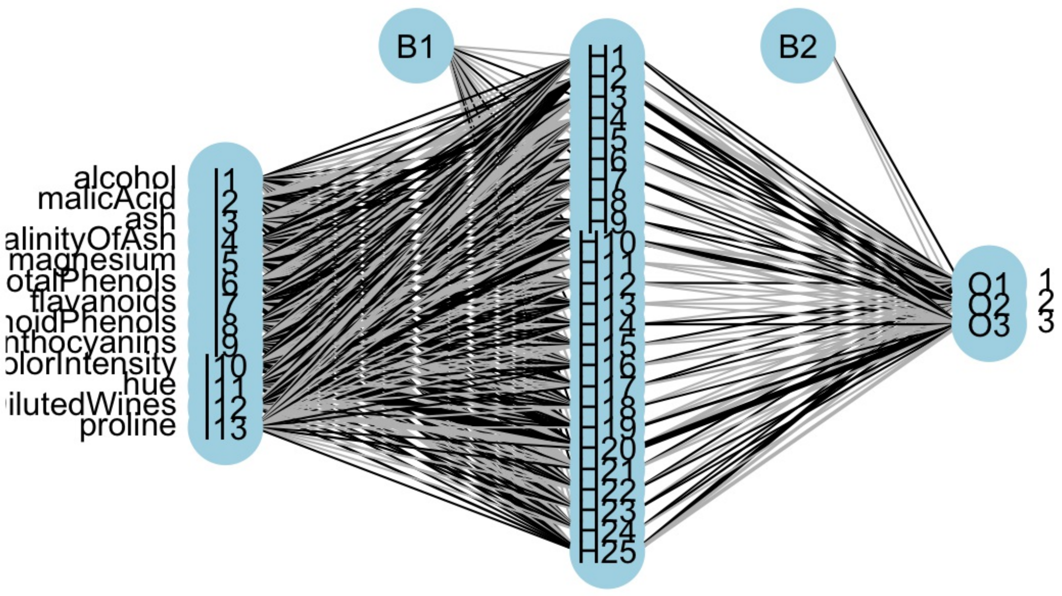
```
   pred
     1  2  3
  1 57  2  0
  2  0 70  1
  3  0  1 47
```

```
[1] 0.9775281
```

A plot of a neural network can be produced using the plot.nnet function, however, it's output is best to demonstrate that 25 nodes were created in the process. These nodes can be seen in the middle of the plot below.

Neural Network Plot: 1 layer separating variable inputs and class outputs

The plot below highlights that different weights are added to each model input to produce the neural network model's class output prediction.



Results

Since each model is a ‘black box’ when it comes to determining how the inputs are treated, it is important that each model is compared for predictive accuracy. To compare each model I present the output table below which provides the misclassification error rate for each model. As the assignment asked to evaluate results in-sample I provide the misclassification error rate for each model using the entire set of wine data observations.

```
                   model model.errors
1          Random Forest   0.00000000
2 Support Vector Machine   0.00000000
3         Neural Network   0.02247191
```

As can be seen from the results table the Random Forest and SVM models have excellent results — both have a 0% misclassification error. The Neural Network model did not perform as well, but its misclassification error was still very low with. Misclassification error for the Neural Network model is 2.25%.

Appendix

```
# 1. Random Forest
set.seed(1)
randomForest.fit = randomForest(class~
    alcohol + malicAcid + ash + alcalinityOfAsh +
    magnesium + totalPhenols + flavanoids + nonflavanoidPhenols + proanthocyanins
+
    colorIntensity + hue + OD280_OD315OfDilutedWines + proline,
    data=wineData, mtry=2, importance=TRUE)
randomForest.fit


# 2. SVM
x = subset(wineData, select=-class)   # Divide wineData to x (the variables) and y
the classes
y = wineData$class

svm.fit = svm(x,y,data=wineData)    # Create SVM model
summary(svm.fit)

svm.pred=predict(svm.fit, x)
table(svm.pred, y)
svm.train.error = 1 - mean(svm.pred==y)
svm.train.error


# 3. Neural Net
#The neural network requires that the species be normalized using one-of-n normaliz
ation. We will normalize between 0 and 1. This can be done with the following comma
nd.
ideal <- class.ind(wineData$class)

# Train the neural network
set.seed(123)
wineANN = nnet(wineData[,-1], ideal, size=25, softmax=TRUE)

# test the output from the neural network on the validation data
pred = predict(wineANN, wineData[,-1], type="class")
table(wineData[,1], pred)

# Plot tutorial found here: https://beckmw.wordpress.com/2013/11/14/visualizing-neu
ral-networks-in-r-update/
# Plot the neural net output
install.packages('devtools')
install.packages('reshape')
library(devtools)
require(reshape)
source_url('https://gist.githubusercontent.com/fawda123/7471137/raw/466c1474d0a505f
f044412703516c34f1a4684a5/nnet_plot_update.r')

#plot the model
plot.nnet(wineANN)
```