Report for Final Project Task 2

Gabriel Afriyie and Ranveer Kaur

14/03/2020

Abstract

Classification can be considered as an estimation of sets, where the risk is defined by means of a specific distance between sets associated with misclassification error. This project develops different classification techniques for the wine data. It presents Linear and Quadratic Discriminant Analysis, K-Nearest Neighbourhood, Classification trees and Support Vector Machine algorithms to predict the type of wine based on their chemical analysis. The best method after prediction is selected based on misclassification error. In particular, this project addresses the following questions: Does the selected method perform better than others? Do you think the classifier is safe enough to guarantee high accuracy in prediction? Are all inputs relevant? Ultimately, the Quadratic Discriminant Analysis classifier is selected as the best among the others.

Introduction

Classification is the process of predicting the class of given data points. This process belongs to the category of supervised learning where the targets are also provided with the input data. In this project, we will use a number of different supervised algorithms to predict the type of wine using data containing the results of chemical analysis of wines grown in an area in Italy.

The best candidate algorithm will be chosen based on their misclassification errors. Our main goal is to build a classifier that accurately predicts the type of wine based on their chemical components. This sort of task can arise in wine manufacturing companies, where manufacturers try to find the attributes of a particular type of wine to use that in creating new varieties.

Data

The dataset for this project originates from the UCI Machine Learning Repository. Three types of wine are represented in the 178 samples, with the results of 13 chemical analyses recorded for each sample. The Type variable has been transformed into a categoric variable.

The data contains no missing values and consits of only numeric data, with a three class target variable (Type) for classification.

Predictor Variables

- *Alcohol*: Alcohol
- *Malic*: Malic acid
- *Ash*: Ash
- *Alcalinity*: Alcalinity of ash

- *Magnesium*: Magnesium
- *Phenols*: Total phenols
- *Flavanoids*: Flavanoids
- *Nonflavanoids*: Nonflavanoid phenols
- *Proanthocyanins*: Proanthocyanins
- *Colour*: Colour intensity
- *Hue*: Hue
- *Dilution*: D280/OD315 of diluted wines
- *Proline*: Proline

Target Variable

Type: The type of wine, into one of three classes, 1 (59 observations), 2(71 observations), and 3 (48 observations).

Import Libraries and Load Data

We will first load the libraries that we are going to use, as well as the wine data. The first column will be our target variable, 'Type', and the rest will be the predictor variables.

```
library(rattle)
library(ggplot2)
library(caret)
library(MASS)
library(gridExtra)
library(grid)
library(e1071)
attach(wine)
library(creditmodel)
library(ROCR)
library(tree)
library(rpart.plot)
```

Splitting the Dataset

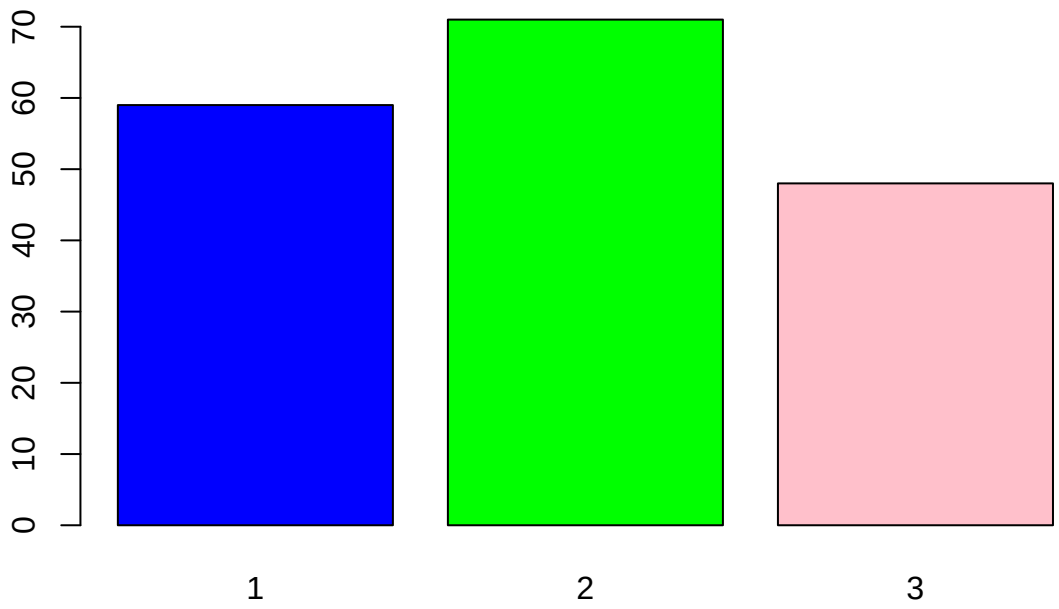
We split the dataset into training data (80%) and test data (20%). We fit the models on the training data and test our models' prediction on the test data.

```
set.seed(1)
datasplit = train_test_split(wine, prop = 0.8, split_type = "Random")
Wine.training = datasplit$train
Wine.test = datasplit$test
```

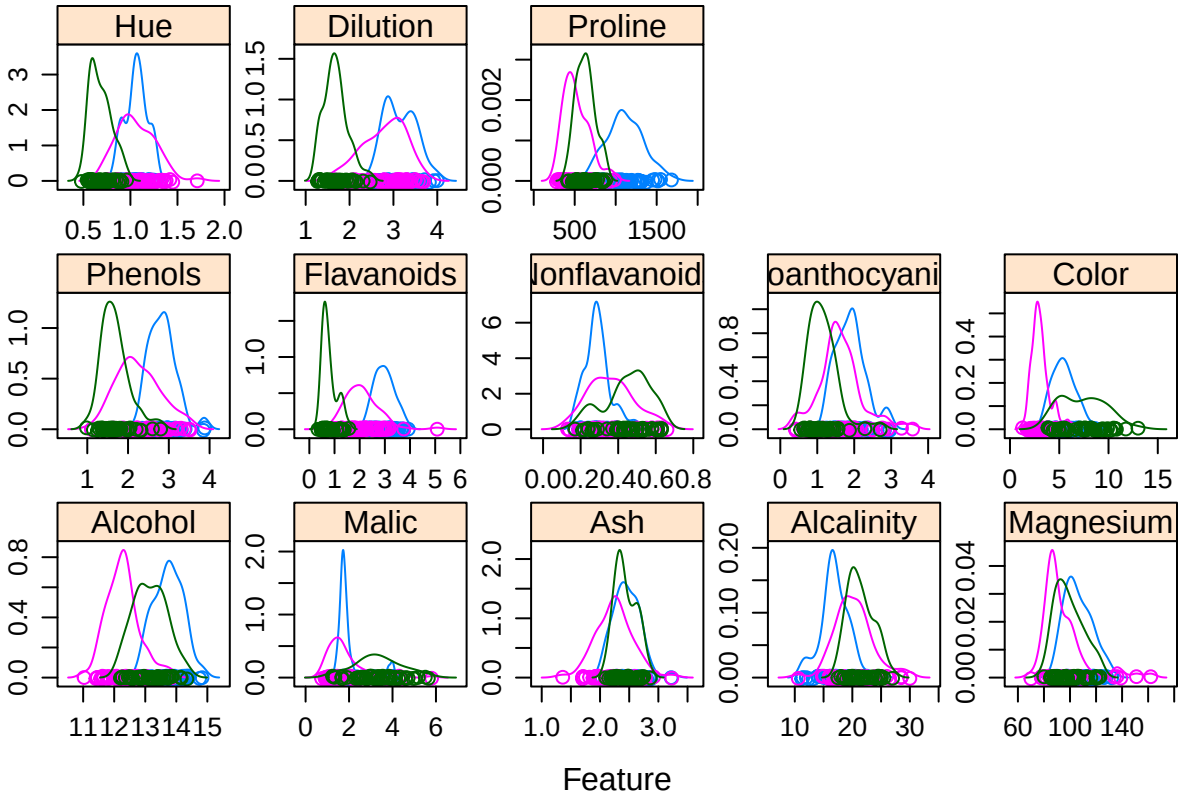
Exploratory Data Analysis

An initial exploration of the dataset will show us how many observations fit in each group and the distributions of the target variable with respect to each predictor variable. We can review the density distribution of each variable broken down by class value. Like the scatterplot matrix, the density plot by class can help see the separation of classes. It can also help to understand the overlap in class values for a variable.

```
plot(Type, col = c("Blue","green","pink"))
```



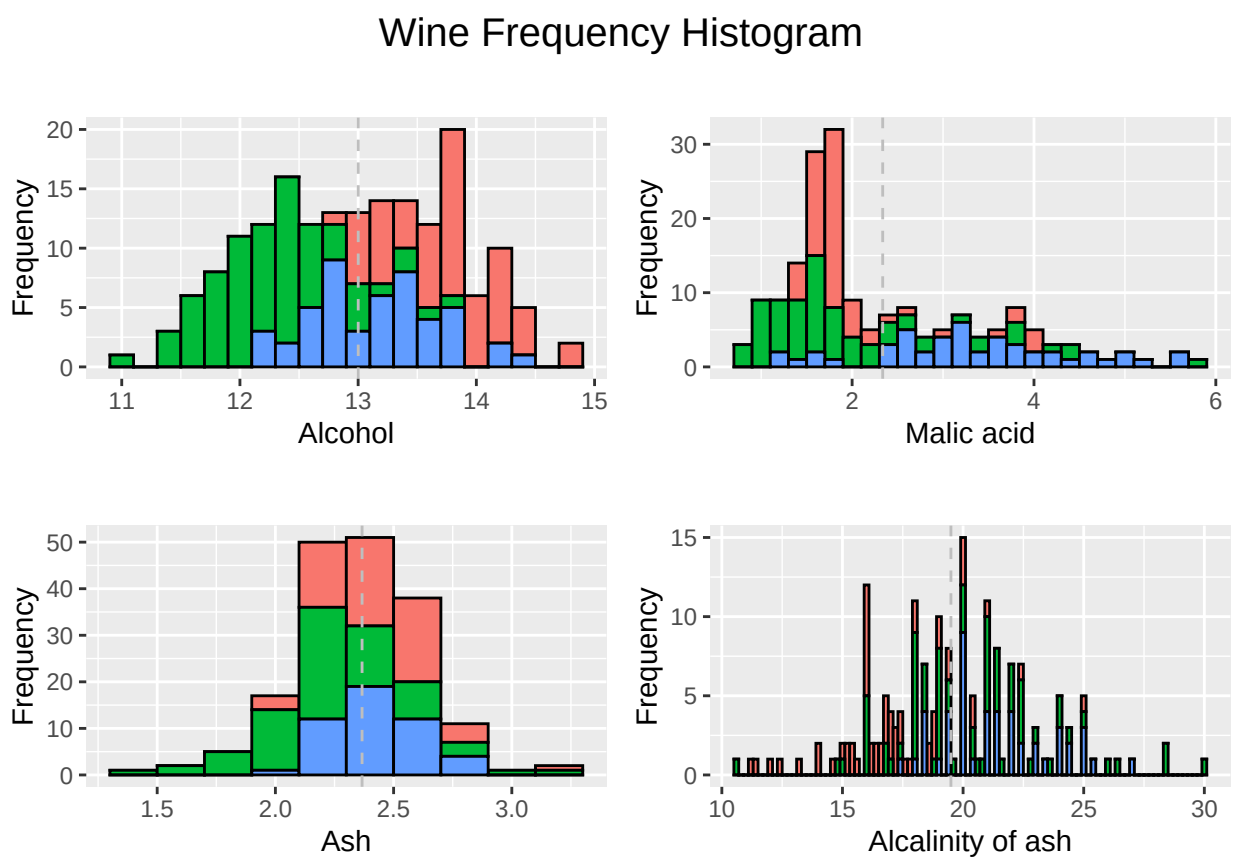

```
x = wine[,2:14]
y = wine[,1]
scales = list(x=list(relation="free"), y=list(relation="free"))
featurePlot(x=x, y=y, "density", scales = scales)
```



Notice the shape of the data, most variables exhibit a normal distribution.

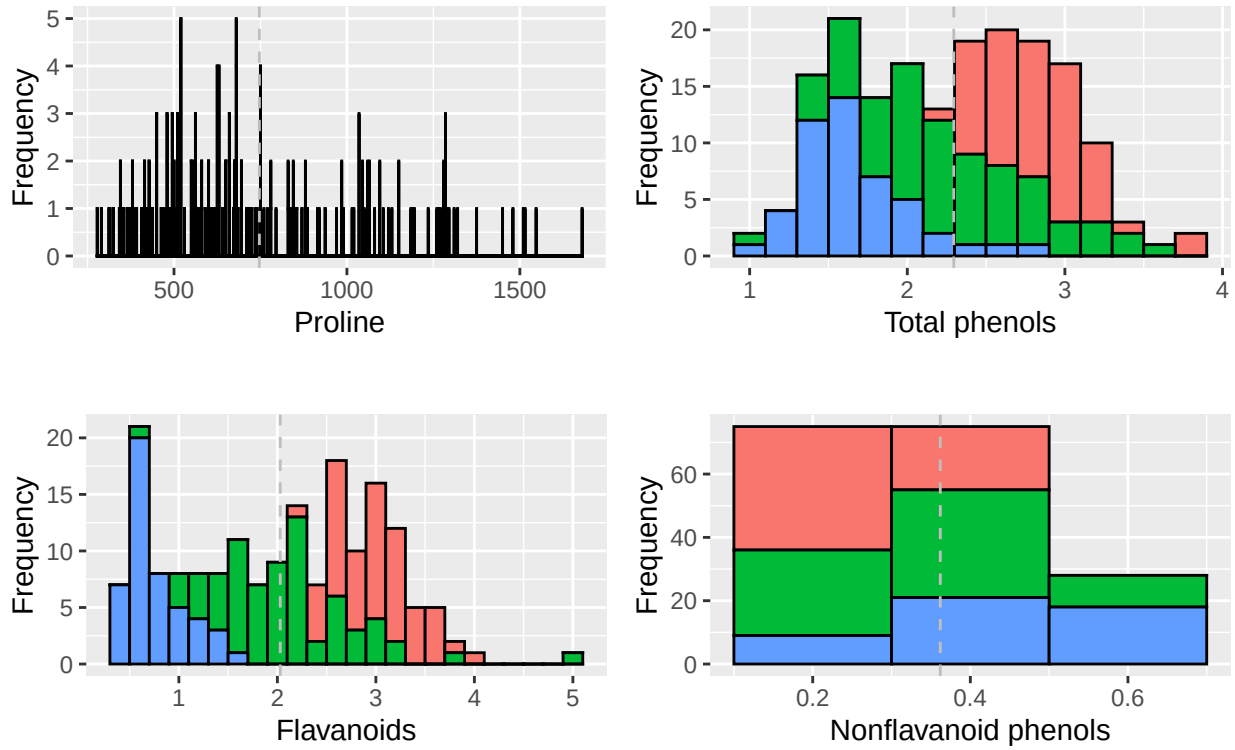
```
HisAl <- ggplot(data=wine, aes(x=Alcohol))+
  geom_histogram(binwidth=0.2, color="black", aes(fill=Type)) +
  xlab("Alcohol") +
  ylab("Frequency") +
  theme(legend.position="none")+
  ggtitle("Histogram of Alcohol")+
  geom_vline(data=wine, aes(xintercept = mean(Alcohol)),linetype="dashed",color="grey")

HisMa <- ggplot(data=wine, aes(x=Malic))+
  geom_histogram(binwidth=0.2, color="black", aes(fill=Type)) +
  xlab("Malic acid") +
  ylab("Frequency") +
  theme(legend.position="none")+
  ggtitle("Histogram of Malic acid")+
  geom_vline(data=wine, aes(xintercept = mean(Malic)),linetype="dashed",color="grey")

HisAsh <- ggplot(data=wine, aes(x=Ash))+
  geom_histogram(binwidth=0.2, color="black", aes(fill=Type)) +
  xlab("Ash") +
  ylab("Frequency") +
  theme(legend.position="none")+
  ggtitle("Histogram of Ash")+
  geom_vline(data=wine, aes(xintercept = mean(Ash)),linetype="dashed",color="grey")

HisAlca <- ggplot(data=wine, aes(x=Alcalinity))+
```

```
  geom_histogram(binwidth=0.2, color="black", aes(fill=Type)) +
  xlab("Alcalinity of ash") +
  ylab("Frequency") +
  theme(legend.position="none")+
  ggtitle("Histogram of Alcalinity of ash")+
  geom_vline(data=wine, aes(xintercept = mean(Alcalinity)),linetype="dashed",color="grey")


# Plot all visualizations
grid.arrange(HisAl + ggtitle(""),
             HisMa + ggtitle(""),
             HisAsh + ggtitle(""),
             HisAlca  + ggtitle(""),
             nrow = 2,
             top = textGrob("Wine Frequency Histogram",
                            gp=gpar(fontsize=15))
)
```



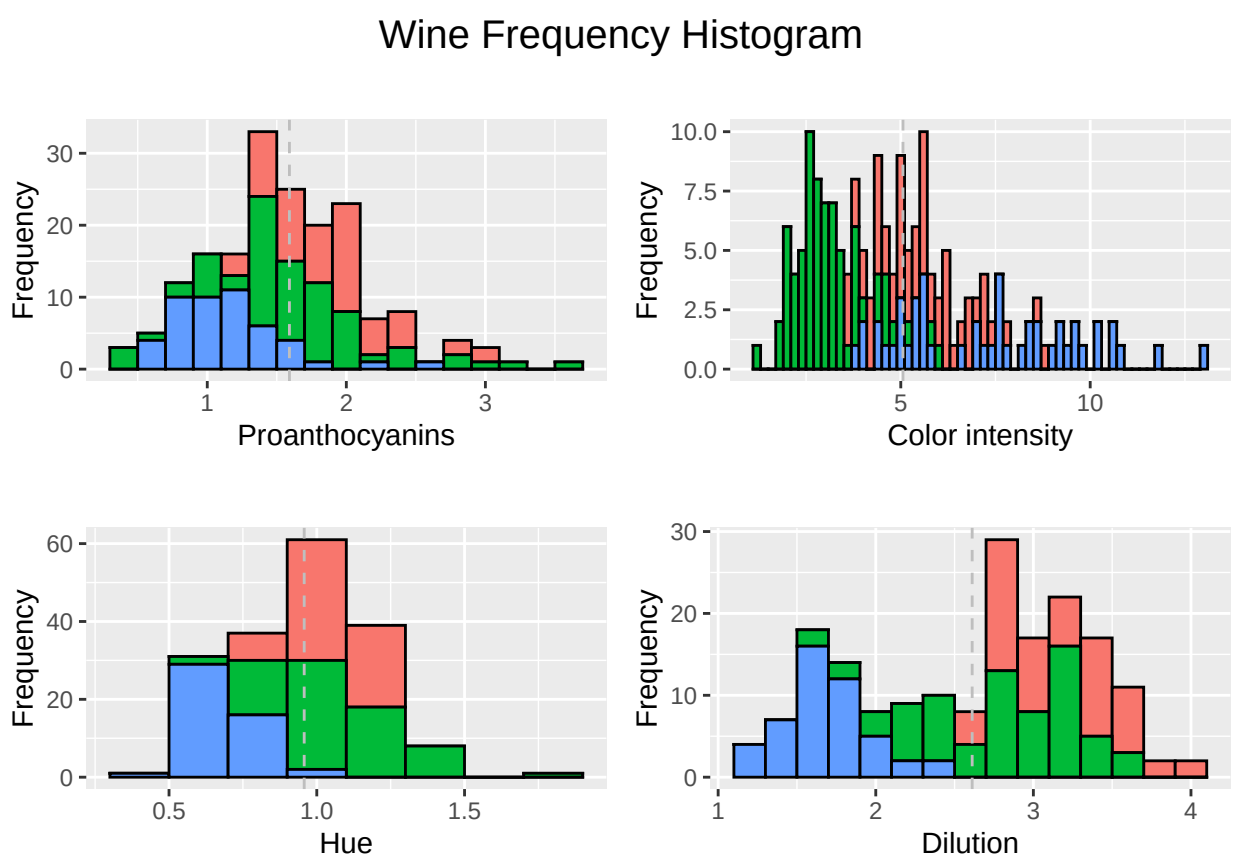

```
HisPl <- ggplot(data=wine, aes(x=Proline))+
  geom_histogram(binwidth=0.2, color="black", aes(fill=Type)) +
  xlab("Proline") +
  ylab("Frequency") +
  theme(legend.position="none")+
  ggtitle("Histogram of Proline")+
  geom_vline(data=wine, aes(xintercept = mean(Proline)),linetype="dashed",color="grey")
```

```r
HisPh <- ggplot(data=wine, aes(x=Phenols))+
  geom_histogram(binwidth=0.2, color="black", aes(fill=Type)) +
  xlab("Total phenols") +
  ylab("Frequency") +
  theme(legend.position="none")+
  ggtitle("Histogram of Total phenols")+
  geom_vline(data=wine, aes(xintercept = mean(Phenols)),linetype="dashed",color="grey")

HisFl <- ggplot(data=wine, aes(x=Flavanoids))+
  geom_histogram(binwidth=0.2, color="black", aes(fill=Type)) +
  xlab("Flavanoids") +
  ylab("Frequency") +
  theme(legend.position="none")+
  ggtitle("Histogram of Flavanoids")+
  geom_vline(data=wine, aes(xintercept = mean(Flavanoids)),linetype="dashed",color="grey")

HisNonFl <- ggplot(data=wine, aes(x=Nonflavanoids))+
  geom_histogram(binwidth=0.2, color="black", aes(fill=Type)) +
  xlab("Nonflavanoid phenols") +
  ylab("Frequency") +
  theme(legend.position="none")+
  ggtitle("Histogram of Nonflavanoid phenols")+
  geom_vline(data=wine, aes(xintercept = mean(Nonflavanoids)),linetype="dashed",color="grey")

# Plot all visualizations
grid.arrange(HisPl + ggtitle(""),
             HisPh + ggtitle(""),
             HisFl + ggtitle(""),
             HisNonFl  + ggtitle(""),
             nrow = 2,
             top = textGrob("Wine Frequency Histogram",
                            gp=gpar(fontsize=15))
)
```




```
HisPr <- ggplot(data=wine, aes(x=Proanthocyanins))+
  geom_histogram(binwidth=0.2, color="black", aes(fill=Type)) +
  xlab("Proanthocyanins") +
  ylab("Frequency") +
  theme(legend.position="none")+
  ggtitle("Histogram of Proanthocyanins")+
  geom_vline(data=wine, aes(xintercept = mean(Proanthocyanins)),linetype="dashed",color="grey")

HisCol <- ggplot(data=wine, aes(x=Color))+
  geom_histogram(binwidth=0.2, color="black", aes(fill=Type)) +
  xlab("Color intensity") +
  ylab("Frequency") +
  theme(legend.position="none")+
  ggtitle("Histogram of Color intensity")+
  geom_vline(data=wine, aes(xintercept = mean(Color)),linetype="dashed",color="grey")

HisHue <- ggplot(data=wine, aes(x=Hue))+
  geom_histogram(binwidth=0.2, color="black", aes(fill=Type)) +
  xlab("Hue") +
  ylab("Frequency") +
  theme(legend.position="none")+
  ggtitle("Histogram of Hue")+
  geom_vline(data=wine, aes(xintercept = mean(Hue)),linetype="dashed",color="grey")

HisDi <- ggplot(data=wine, aes(x=Dilution))+
  geom_histogram(binwidth=0.2, color="black", aes(fill=Type)) +
```

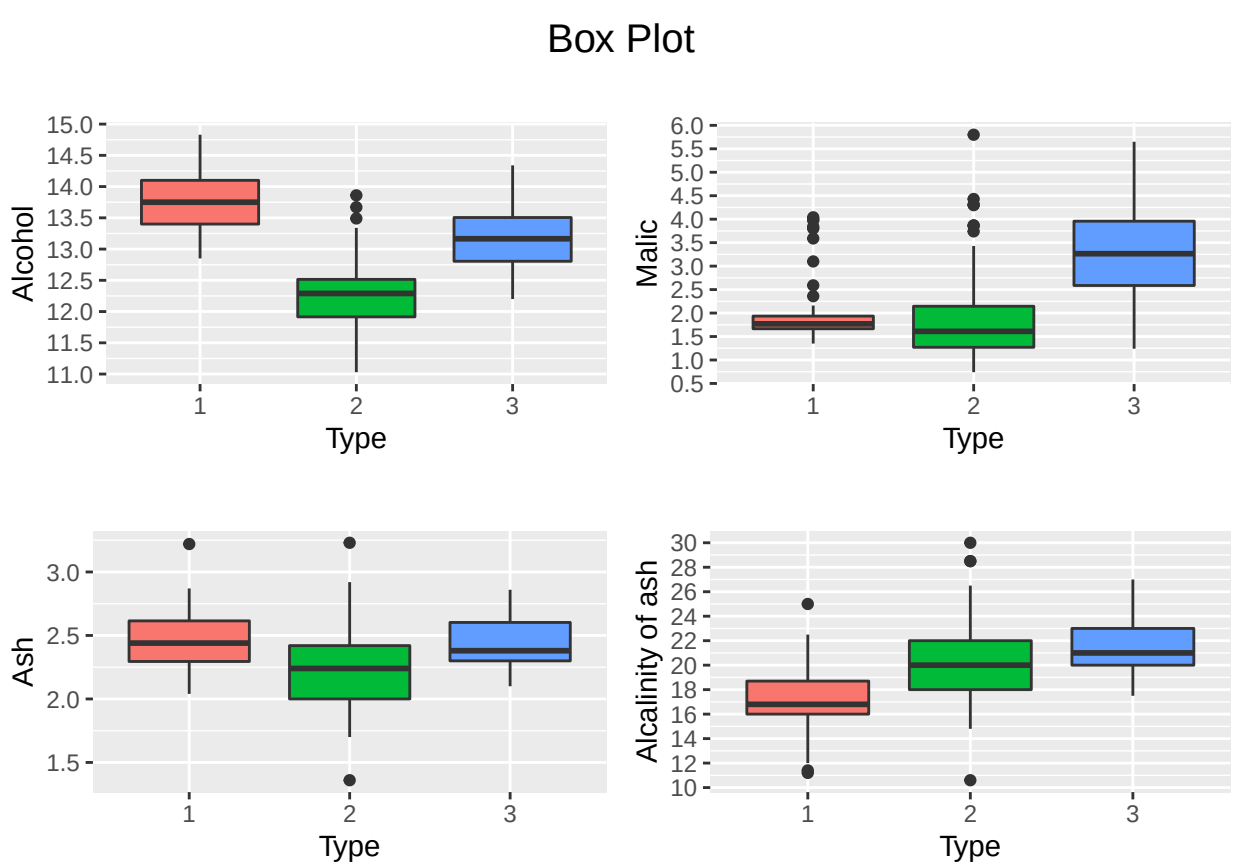
```
  xlab("Dilution") +
  ylab("Frequency") +
  theme(legend.position="none")+
  ggtitle("Histogram of Dilution")+
  geom_vline(data=wine, aes(xintercept = mean(Dilution)),linetype="dashed",color="grey")

# Plot all visualizations
grid.arrange(HisPr + ggtitle(""),
             HisCol + ggtitle(""),
             HisHue + ggtitle(""),
             HisDi  + ggtitle(""),
             nrow = 2,
             top = textGrob("Wine Frequency Histogram",
                            gp=gpar(fontsize=15))
)
```



Next with the boxplots, we will identify some outliers. As we see below, some classes do not overlap at all (e.g. Alcohol), where as with other variables, they are hard to tear apart (Alcalinity of Ash).

```
# Let's plot all the variables in a single visualization that will contain all the boxplots

BpAl <- ggplot(wine, aes(Type, Alcohol, fill=Type)) +
        geom_boxplot()+
        scale_y_continuous("Alcohol", breaks= seq(0,30, by=.5))+
        theme(legend.position="none")

BpMa <- ggplot(wine, aes(Type, Malic, fill=Type)) +
```

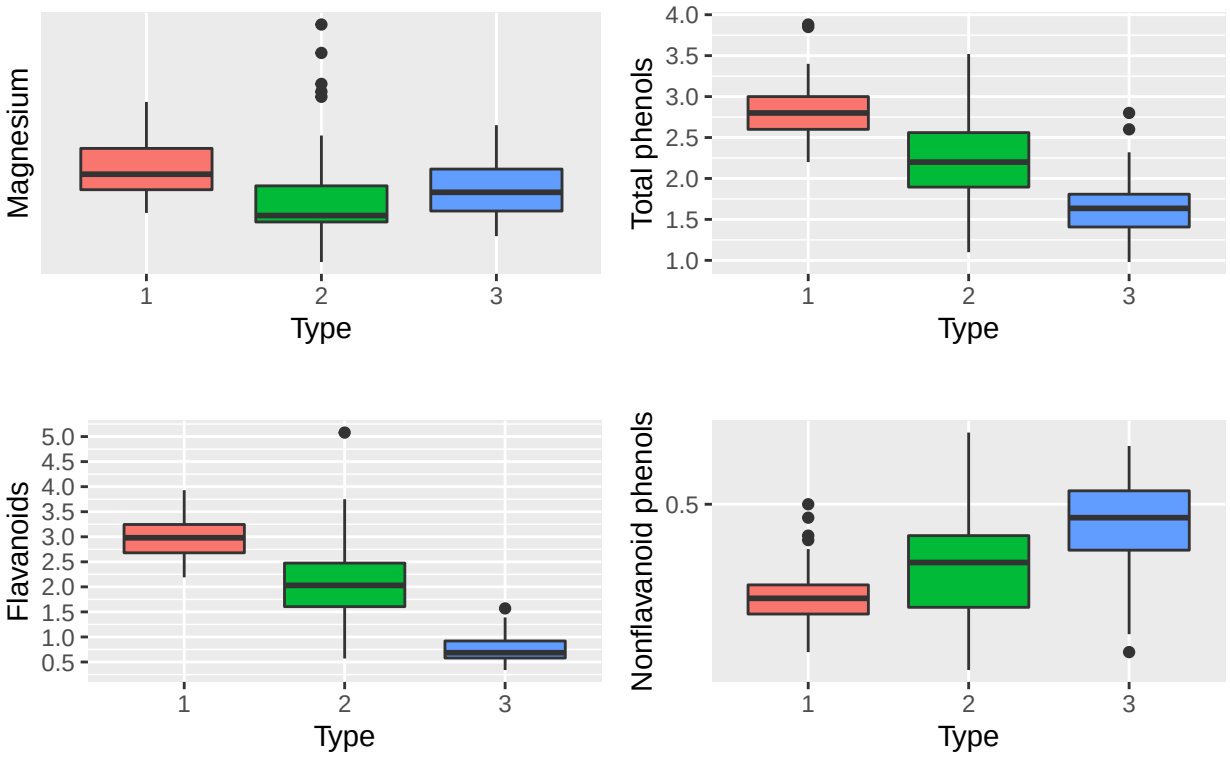
```
        geom_boxplot()+
        scale_y_continuous("Malic", breaks= seq(0,30, by=.5))+
        theme(legend.position="none")

BpAsh <- ggplot(wine, aes(Type, Ash, fill=Type)) +
        geom_boxplot()+
        scale_y_continuous("Ash", breaks= seq(0,30, by=.5))+
        theme(legend.position="none")

BpAlc <- ggplot(wine, aes(Type, Alcalinity, fill=Type)) +
        geom_boxplot()+
        scale_y_continuous("Alcalinity of ash", breaks= seq(0,30, by=2))+
        theme(legend.position="none")

# Plot all visualizations
grid.arrange(BpAl  + ggtitle(""),
             BpMa  + ggtitle(""),
             BpAsh + ggtitle(""),
             BpAlc + ggtitle(""),
             nrow = 2,
             top = textGrob("Box Plot",
                            gp=gpar(fontsize=15))
)
```




```
BpMag <- ggplot(wine, aes(Type, Magnesium, fill=Type)) +
        geom_boxplot()+
```

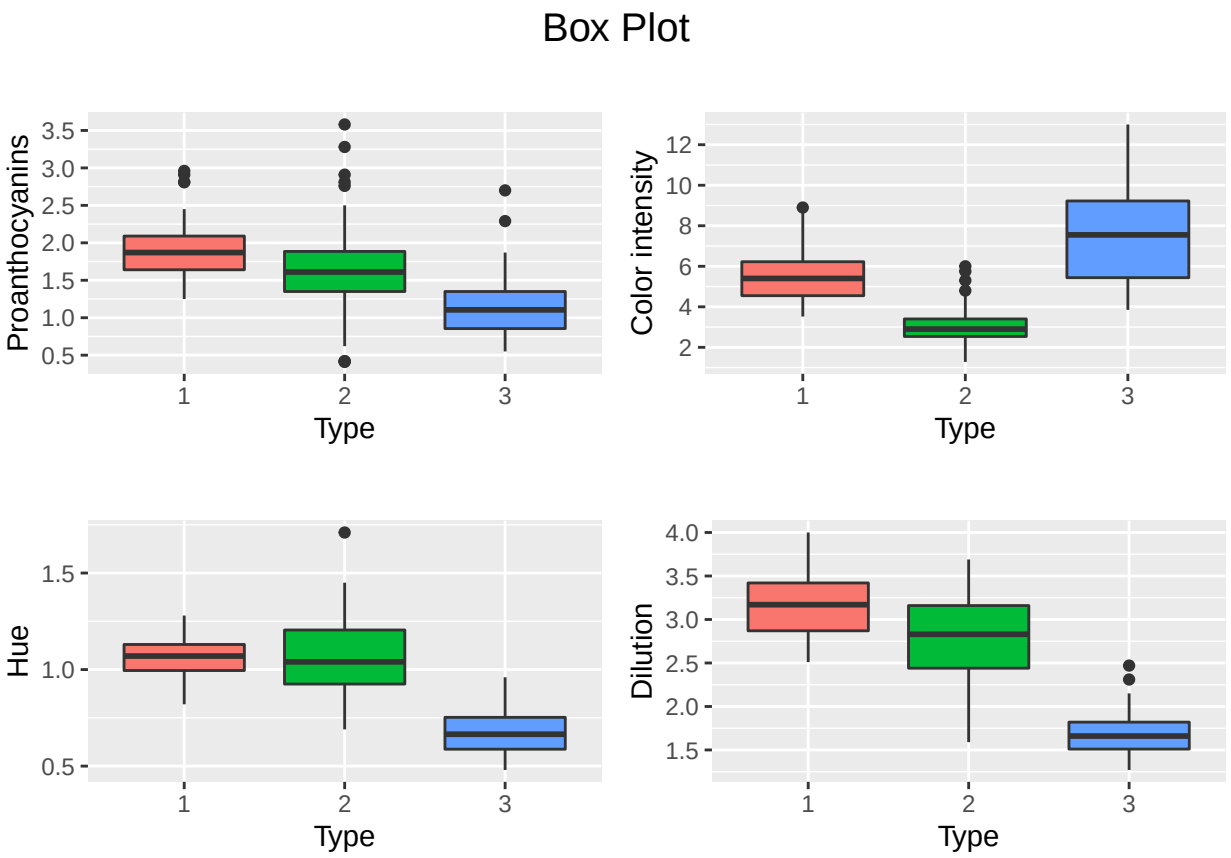
```
        scale_y_continuous("Magnesium", breaks= seq(0,30, by=.5))+
        theme(legend.position="none")

BpPh <- ggplot(wine, aes(Type, Phenols, fill=Type)) +
        geom_boxplot()+
        scale_y_continuous("Total phenols", breaks= seq(0,30, by=.5))+
        theme(legend.position="none")

BpFl <- ggplot(wine, aes(Type, Flavanoids, fill=Type)) +
        geom_boxplot()+
        scale_y_continuous("Flavanoids", breaks= seq(0,30, by=.5))+
        theme(legend.position="none")

BpNonFl <- ggplot(wine, aes(Type, Nonflavanoids, fill=Type)) +
        geom_boxplot()+
        scale_y_continuous("Nonflavanoid phenols", breaks= seq(0,30, by=.5))+
        theme(legend.position="none")

# Plot all visualizations
grid.arrange(BpMag  + ggtitle(""),
             BpPh  + ggtitle(""),
             BpFl + ggtitle(""),
             BpNonFl + ggtitle(""),
             nrow = 2,
             top = textGrob("Box Plot",
                            gp=gpar(fontsize=15))
)
```

Box Plot

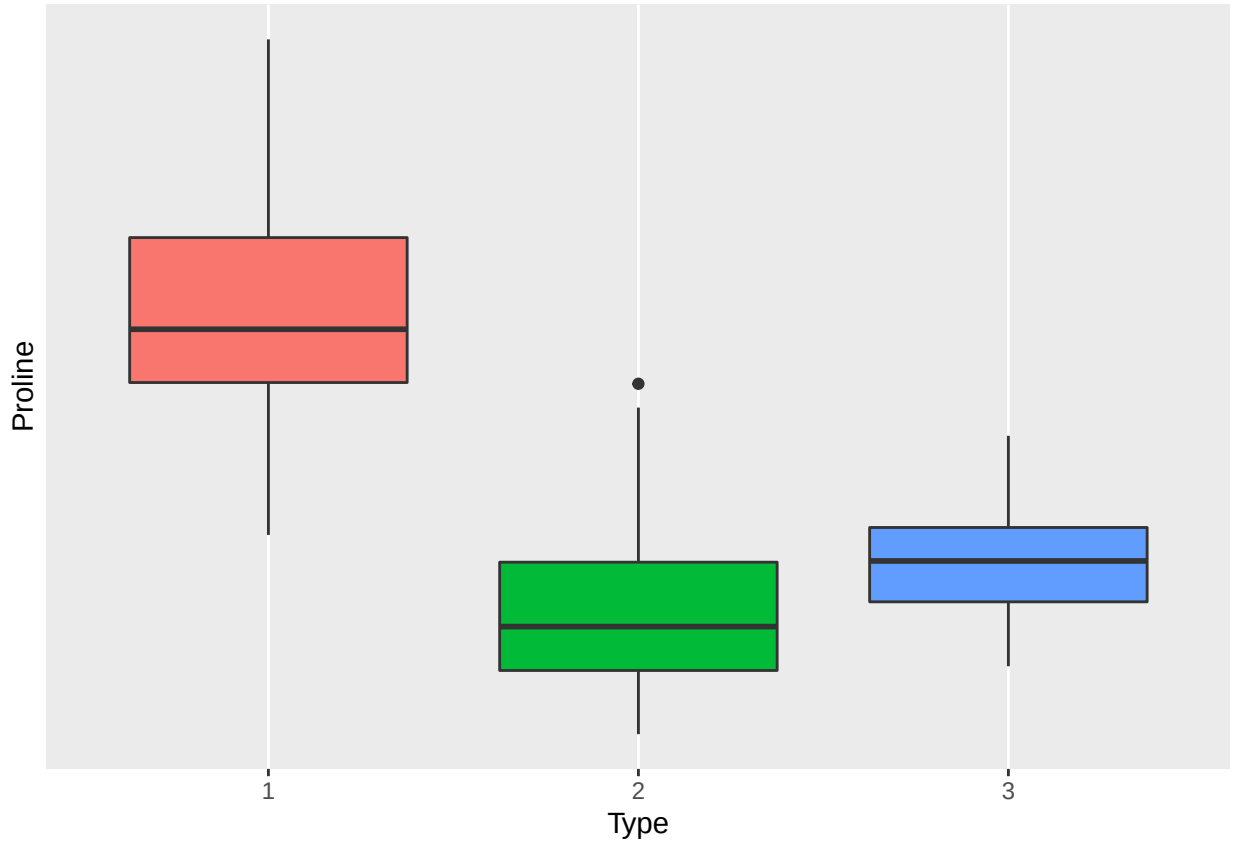



```
BpPr <- ggplot(wine, aes(Type, Proanthocyanins, fill=Type)) +
        geom_boxplot()+
        scale_y_continuous("Proanthocyanins", breaks= seq(0,30, by=.5))+
        theme(legend.position="none")

BpCol <- ggplot(wine, aes(Type, Color, fill=Type)) +
        geom_boxplot()+
        scale_y_continuous("Color intensity", breaks= seq(0,30, by=2))+
        theme(legend.position="none")

BpHue <- ggplot(wine, aes(Type, Hue, fill=Type)) +
        geom_boxplot()+
        scale_y_continuous("Hue", breaks= seq(0,30, by=.5))+
        theme(legend.position="none")

BpDi <- ggplot(wine, aes(Type, Dilution, fill=Type)) +
        geom_boxplot()+
        scale_y_continuous("Dilution", breaks= seq(0,30, by=.5))+
        theme(legend.position="none")

# Plot all visualizations

grid.arrange(BpPr  + ggtitle(""),
             BpCol  + ggtitle(""),
             BpHue + ggtitle(""),
             BpDi + ggtitle(""),
```

```
            nrow = 2,
            top = textGrob("Box Plot",
                           gp=gpar(fontsize=15))
)
```




```
BpPro <- ggplot(wine, aes(Type, Proline, fill=Type)) +
        geom_boxplot()+
        scale_y_continuous("Proline", breaks= seq(0,30, by=.5))+
        theme(legend.position="none")
BpPro
```



Model's Performance Evaluation

In this section, we will study 5 different algorithms and determine the best in terms of predicting our data. We will consider:

- Linear Discriminant Analysis (LDA)
- Quadratic Discriminant Analysis (QDA)
- K-Nearest Neighbourhood (KNN)
- Classification Trees
- Support Vector Machines (SVM)

LDA

```
#building the LDA model of wine using Training Set
ldafit = lda(Type~., data = Wine.training)
ldafit
```

```
## Call:
## lda(Type ~ ., data = Wine.training)
##
## Prior probabilities of groups:
##         1         2         3
```

```
## 0.3239437 0.4014085 0.2746479
##
## Group means:
##    Alcohol    Malic      Ash Alcalinity Magnesium  Phenols Flavanoids
## 1 13.77196 2.052174 2.437391   16.78696 106.19565 2.794130  2.9573913
## 2 12.28614 1.968772 2.256667   20.40526  93.85965 2.277719  2.0796491
## 3 13.17179 3.332051 2.433846   21.67949  99.64103 1.686410  0.7838462
##   Nonflavanoids Proanthocyanins    Color       Hue Dilution   Proline
## 1     0.2917391        1.897826 5.510000 1.0723913 3.151522 1119.7174
## 2     0.3615789        1.681404 3.057018 1.0525614 2.805439  530.8947
## 3     0.4525641        1.168205 7.603077 0.6751282 1.640256  638.3333
##
## Coefficients of linear discriminants:
##                            LD1          LD2
## Alcohol         -0.130043801  0.762464139
## Malic            0.052229480  0.309616220
## Ash             -0.731810418  2.031440831
## Alcalinity       0.162865260 -0.162216334
## Magnesium       -0.008906907  0.001120386
## Phenols          1.149166480  0.041621030
## Flavanoids      -2.225968260 -0.309492354
## Nonflavanoids   -2.212227457 -1.163922742
## Proanthocyanins  0.338844899 -0.397847459
## Color            0.335151216  0.274771139
## Hue             -0.791650325 -1.266599198
## Dilution        -1.403440052 -0.057178717
## Proline         -0.002552020  0.002924111
##
## Proportion of trace:
##    LD1    LD2
## 0.7238 0.2762
```

```
#predict the test set using the LDA model
preds = predict(ldafit, Wine.test)

#evaluating the misclassification error rate
lda.class=preds$class
classtype = Wine.test$Type
table(lda.class, classtype)
```

```
##          classtype
## lda.class  1  2  3
##         1 13  1  0
##         2  0 13  0
##         3  0  0  9
```

```
mean(lda.class!=classtype)
```

```
## [1] 0.02777778
```

From the result, we know that LDA predict test set well with a misclassification rate of 2.78%.

QDA

```
#building the QDA model of wine using Training Set
qda.fit=qda(Type~.,data=Wine.training)
qda.fit
```

```
## Call:
## qda(Type ~ ., data = Wine.training)
##
## Prior probabilities of groups:
##         1         2         3
## 0.3239437 0.4014085 0.2746479
##
## Group means:
##    Alcohol    Malic      Ash Alcalinity Magnesium  Phenols Flavanoids
## 1 13.77196 2.052174 2.437391   16.78696 106.19565 2.794130  2.9573913
## 2 12.28614 1.968772 2.256667   20.40526  93.85965 2.277719  2.0796491
## 3 13.17179 3.332051 2.433846   21.67949  99.64103 1.686410  0.7838462
##   Nonflavanoids Proanthocyanins    Color       Hue Dilution   Proline
## 1     0.2917391        1.897826 5.510000 1.0723913 3.151522 1119.7174
## 2     0.3615789        1.681404 3.057018 1.0525614 2.805439  530.8947
## 3     0.4525641        1.168205 7.603077 0.6751282 1.640256  638.3333
```

```
#predict the test set using the QDA model
qdapreds=predict(qda.fit,Wine.test)

#evaluating the misclassification error rate
qda.class = qdapreds$class
testclass = Wine.test$Type
table(qda.class,testclass)
```

```
##          testclass
## qda.class  1  2  3
##         1 13  0  0
##         2  0 14  0
##         3  0  0  9
```

```
mean(qda.class!=testclass)
```

```
## [1] 0
```

QDA perfectly predicts the test set with no misclassification.

KNN

We fit different KNN models with different values of $k = 1, \ldots, 10$. For the sake of simplicity, we present the case with the least misclassification rate after prediction.

```
#building the KNN model (with k=1) of wine using Training Set and predicting the test set.
library(class)
train.X = data.frame(Wine.training[,2:14])
test.X = data.frame(Wine.test[,2:14])
knnpred = knn(train.X, test.X, Wine.training$Type, k=1)
```

```r
#evaluating the misclassification error rate
table(knnpred, Wine.test$Type )
```

```
##
## knnpred  1  2  3
##       1 10  1  0
##       2  0 10  1
##       3  3  3  8
```

```r
mean(knnpred!=Wine.test$Type)
```

```
## [1] 0.2222222
```

The misclassication rate for KNN with $k = 1$ is 22.2%. KNN poorly predicts the test set. This is evidence that KNN is effective when the training data is large.

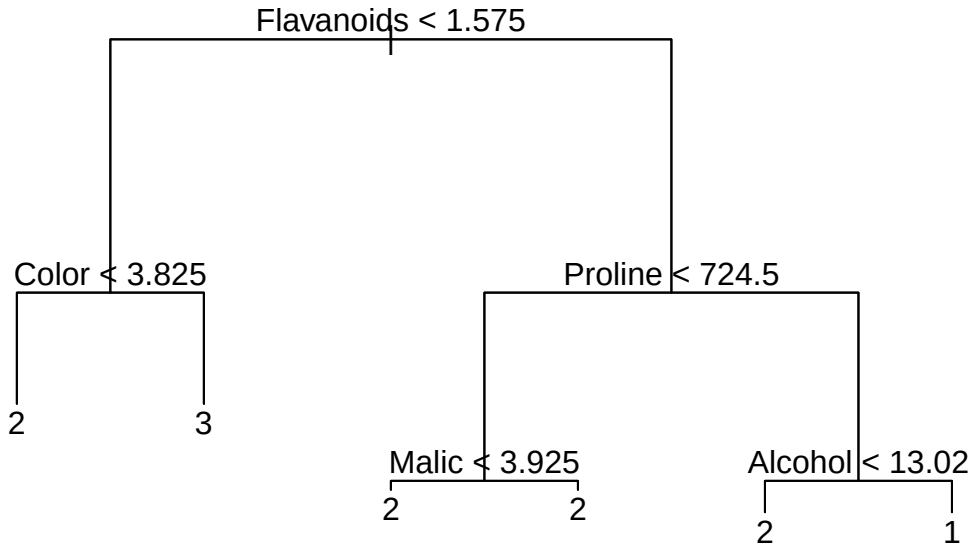
Classification Trees

Decision trees are intuitively very easy to explain. We predict that each observation belongs to the most commonly class of training observations in the region to which it belongs. We now fit the classification tree model.

```r
#building the classification tree model of wine using Training Set
tree.wine =tree(Type~., Wine.training )
summary(tree.wine)
```

```
##
## Classification tree:
## tree(formula = Type ~ ., data = Wine.training)
## Variables actually used in tree construction:
## [1] "Flavanoids" "Color"      "Proline"    "Malic"      "Alcohol"
## Number of terminal nodes:  6
## Residual mean deviance:  0.08628 = 11.73 / 136
## Misclassification error rate: 0.02113 = 3 / 142
```

```r
#plot the tree
plot(tree.wine)
text(tree.wine,pretty =0)
```




```
#predict on test set
tree.pred=predict (tree.wine, Wine.test ,type ="class")
table(tree.pred ,Wine.test$Type)
```

```
##
## tree.pred  1  2  3
##         1 13  0  0
##         2  0 13  0
##         3  0  1  9
```

```
mean(tree.pred!=Wine.test$Type)
```

```
## [1] 0.02777778
```

The following variables are involved: Flavanoids, Color, Proline, Alcohol. There are 6 terminal nodes. The residual mean deviance is 0.0863 and the misclassification error rate on the training data is 2.11%. After predicting on the test set, we obtain a misclassification error rate of 2.78%. This is, as expected higher than that for the training set. Plotting helps us to visualize well the tree.

Support Vector Machines

For our wine dataset, we will build a linear SVM classifier to fit a model to predit types of wine. The classifier is built by performing ten-fold cross-validation to compare SVMs with linear kernel, using a range of values of the cost parameter (c).

```
#SVM with linear kernel with different values of c.
set.seed(111)
```

```
tune_out3 = tune(svm,
Type~.,
data = Wine.training,
kernel = "linear",
ranges = list(cost = c(0.001, 0.01, 0.1, 1,5,10,100)))

summary(tune_out3)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##  0.01
##
## - best performance: 0.02095238
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-03 0.59761905 0.10898765
## 2 1e-02 0.02095238 0.03376142
## 3 1e-01 0.02095238 0.03376142
## 4 1e+00 0.03476190 0.04830459
## 5 5e+00 0.03476190 0.04830459
## 6 1e+01 0.03476190 0.04830459
## 7 1e+02 0.03476190 0.04830459
```

For the SVM with a linear kernel, the cost paramter, $c = 0.01$ produces the SVM with the smallest cross validation error (0.02095238). We can then select the best model and use it for predictions on the test set.

```
#selecting the best model
best_model3 = tune_out3$best.model
best_model3
```

```
##
## Call:
## best.tune(method = svm, train.x = Type ~ ., data = Wine.training,
##     ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01
##
## Number of Support Vectors:  90
```

```
#predicting on the test data
svm.pred1 = predict(best_model3, newdata = Wine.test, decision.values = TRUE)

conf.mat.svm1 = table(Wine.test$Type,svm.pred1)
conf.mat.svm1
```

```
##    svm.pred1
```

```
##      1  2  3
##   1 13  0  0
##   2  0 14  0
##   3  0  0  9
```

```
mean(svm.pred1!=Wine.test$Type)
```

```
## [1] 0
```

There are 90 support vectors and no misclassification. Other kernels (polynomial and radial) were considered but since the linear kernels perform better, we decide to omit those cases.

Model Selection

Based on the misclassification error rates, the QDA and Linear SVM classifiers are the best candidates. To select the best between the best, we perform cross validation for QDA and compare the error to the already calculated cross validation error for linear SVM.

```
K = 10
folds2 = cut(seq(1,nrow(Wine.training)), breaks = K, labels = FALSE)
set.seed(1)
cv.qda2 = sapply(1:K, FUN = function(i){
  testid2 = which(folds2 == i, arr.ind = TRUE)
  Test2 = Wine.training[testid2,]
  Train2 = Wine.training[-testid2,]
  qda.fit=qda(Type~.,data=Wine.training)
  qdapreds=predict(qda.fit,Wine.test)
  cv.est.qda2 = mean(qdapreds$class!=Wine.test$Type)
  return(cv.est.qda2)
})
mean(cv.qda2)
```

```
## [1] 0
```

Comparing the errors, we find that QDA is the best model and the QDA classifier is best to predict the type of wine.

Conclusion

Throughout this project, we developed different classification models by using the training set of the wine data. The target variable in question is the type of wine, with 3 classes. We considered a set of classifiers that perform well in multi-class predictions. After predicting the test set, the QDA and linear SVM performed best, with no misclassifications. We select QDA because it produced the lowest cv error. There are other ways to check the accuracy of a classifier, such calculating the specificity and sensitivity, AUc and classification accuracy of the classification model.