Decision Tree and SVM

Will Edwards

3/1/2020

Classification

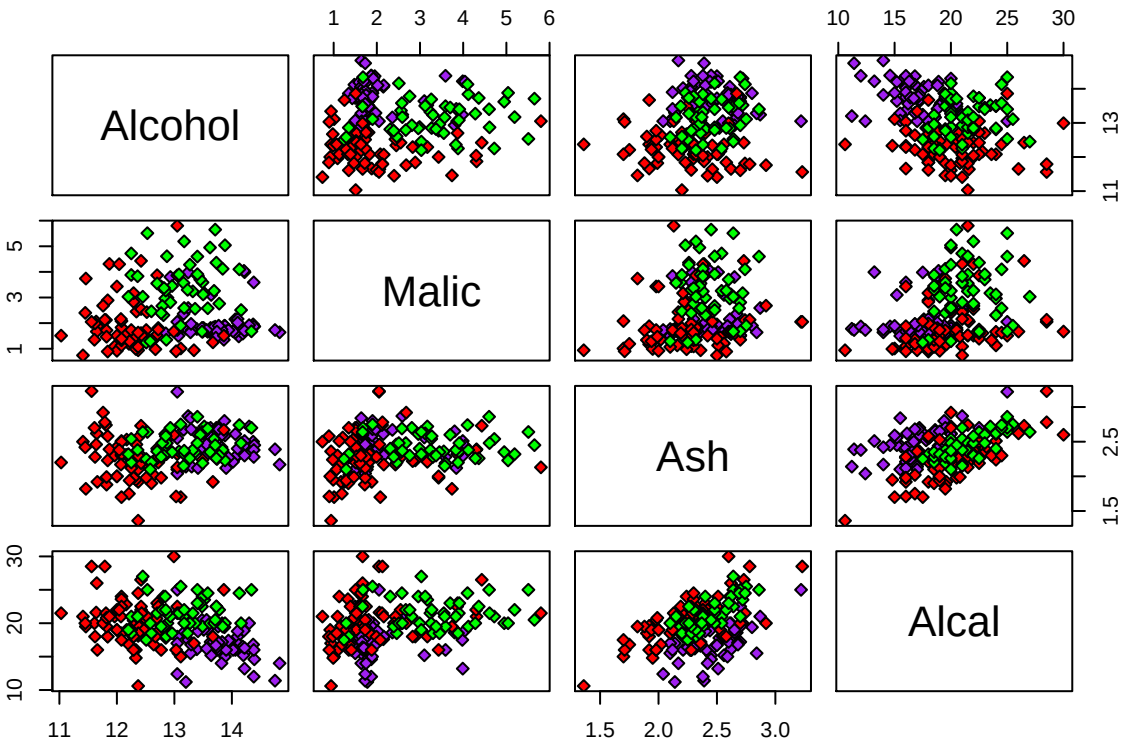
A common problem in the real world is classification. Our objective is to take observations and based on their attributes, assign, or "classify" them as being from one group or another.

This dataset is the result of a chemical analysis on wine that was made in the same region in Italy. However, the grapes were derived from three different cultivars.

```
# Read the data
wines <-read.table("/Users/bulldogwill/Documents/ST 537/Datasets/Wines.txt", header = TRUE)
# classes of wine
table(wines$Class)
```

```
##
##  1  2  3
## 59 71 48
```

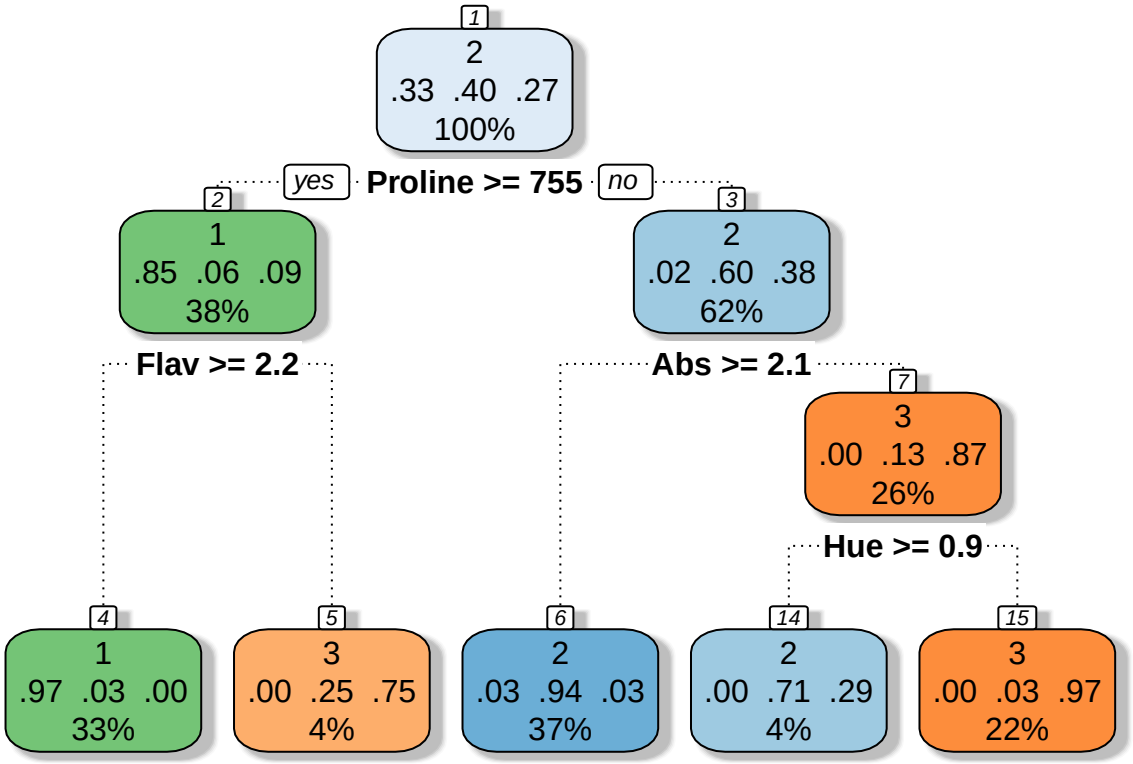
```
colors <- c('purple', 'red', "green")[unclass(wines$Class)]
pairs(wines[ , 2:5], bg = colors , pch = 23)
```



```
library(rpart)
# Classification of the full wine data
tree <-rpart(factor(Class) ~ ., data = wines,method = "class", control =rpart.control(cp =0.01))
# Draw the tree
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
fancyRpartPlot(tree, sub = "", main = "")
```



Support Vector Machines

Support vector machine applied upon the `wine` dataset from before, using Alcohol and Flavine. A plot with the points colored by predicted class has been produced.

```
library(kernlab)
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'
```

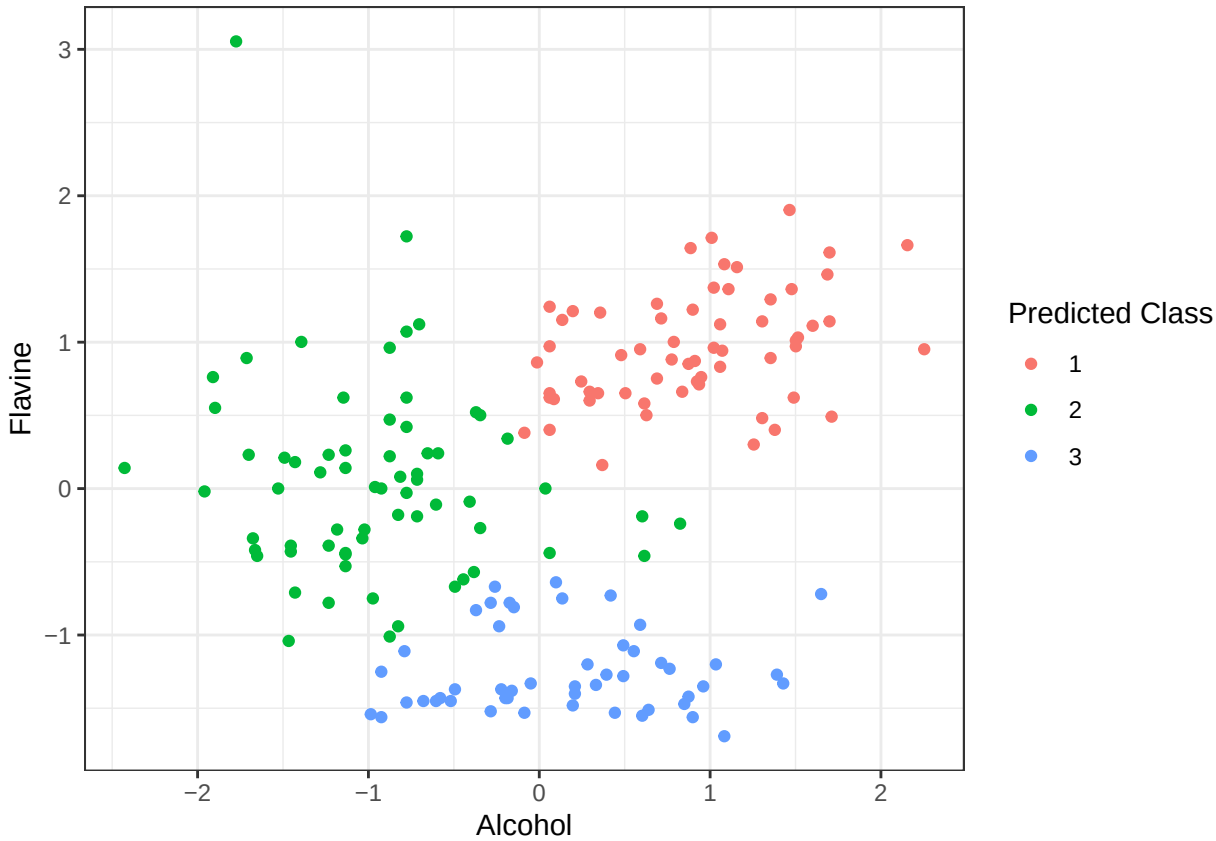
```
## The following object is masked from 'package:kernlab':
##
##     alpha
```

```
# Extract alcohol, proline and classes
alc <- wines$Alcohol
flav <- wines$Flav
group <- as.factor(wines$Class)
```

```r
# create the data frame
df <- data.frame(flavine = flav, alcohol = alc,
group = group)
# SVM with radial kernel
sv.wine <- ksvm(group ~ ., data = df, type = "C-svc",
kernel = "rbfdot", C = 1)

pred.class <- predict(object = sv.wine, newdata = df, type = "response")
err <- errorMatrix(actual = df$group, predicted = pred.class)
new <- data.frame(alc,flav,sv.wine@ymatrix)

ggplot(new, aes(x=new$alc,y=new$flav, color=as.factor(pred.class)))+geom_point()+theme_bw() +
  labs(
    x = "Alcohol",
    y = "Flavine",
    color = "Predicted Class"
  ) + theme_bw()
```



Confusion matrix:

```r
err
```

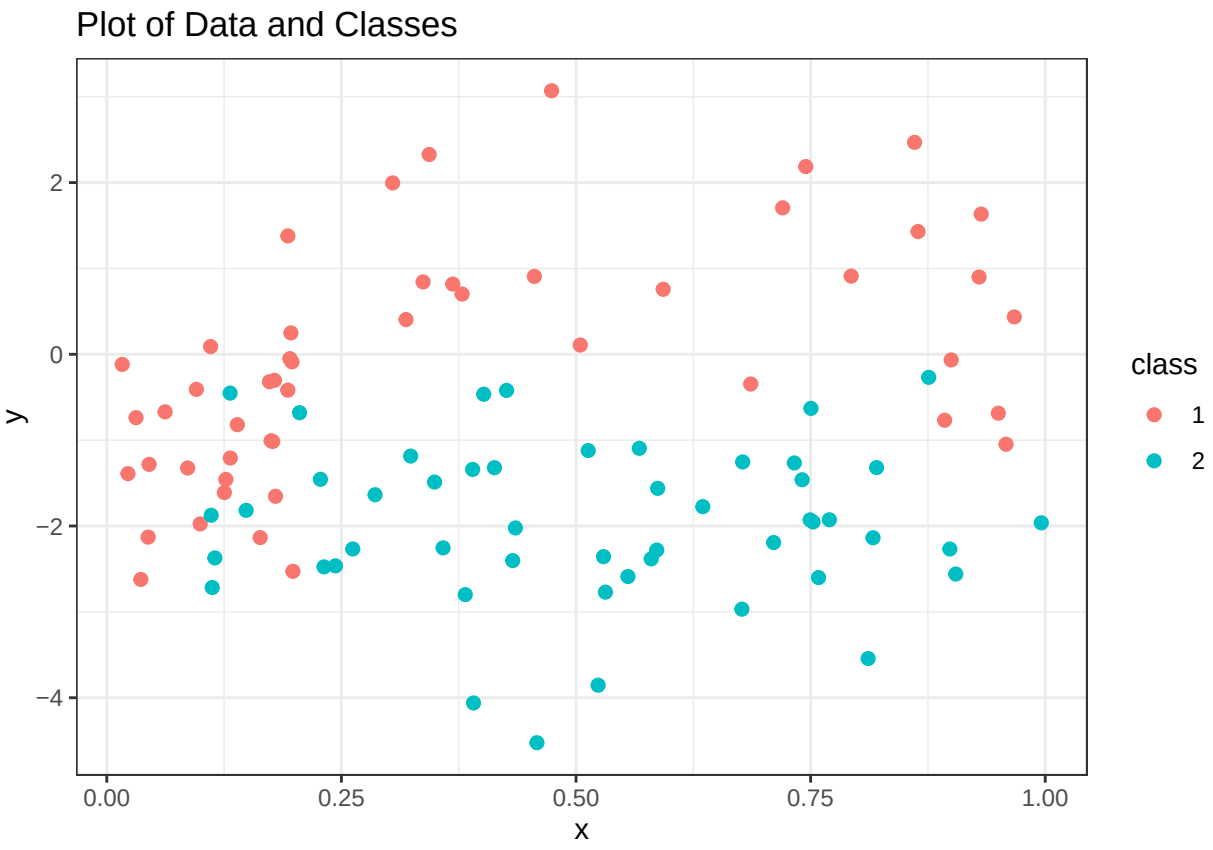
```
##       Predicted
## Actual    1    2    3 Error
##      1 32.6  0.6  0.0   1.7
##      2  2.2 36.0  1.7   9.9
##      3  0.0  0.6 26.4   2.1
```

Below is randomly generated bivariate data. I used a few different random number generators to give the random points a sort of "shape." We can see that most of class 1 has larger vales of y, but there is a cluster of red points on the left side that extend down near y = -2. An advantage of th SVM is that it can create non-linear classification boundaries and also performs well with high-dimensional data.

```
set.seed(342)
x_1 <- runif(25,0,.2)
x <- runif(75,.1,1)
y_1 <- rnorm(25,-1,1)
y_2 <- rnorm(25,1,1)
y_3 <-rnorm(50,-2,1)

classes <- as.factor(c(rep(1,50),rep(2,50)))
data <- data.frame(c(x_1,x),c(y_1,y_2,y_3),classes)
colnames(data) <- c("x","y","class")

ggplot(data, aes(x=x,y=y, color=class))+geom_point(size=2)+theme_bw()+ggtitle("Plot of Data and Classes
```

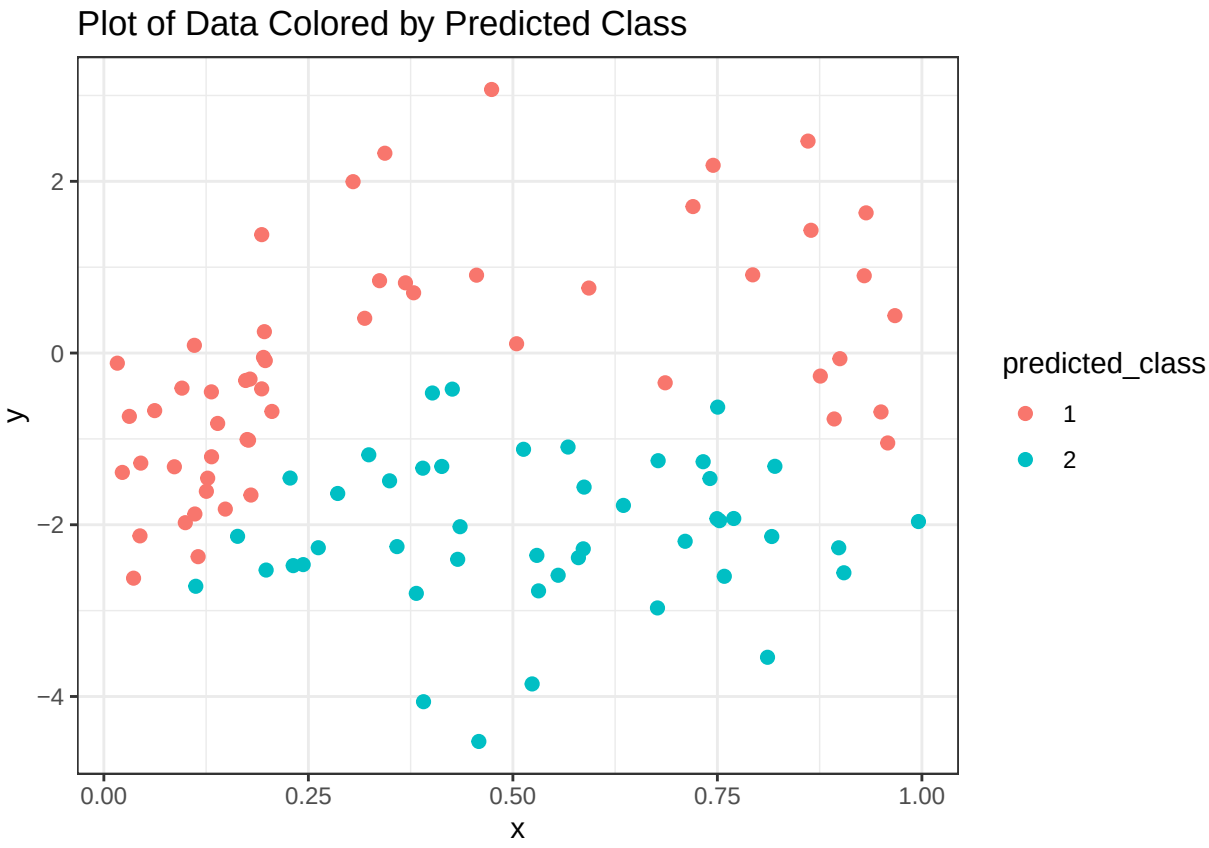


Using the radial kernel, specified by the `kernel` argument, allows the classification boundary to be non-linear. The plot below is the same data from above, but colored by class as predicted from the SVM. We can see that the SVM has attempted to capture this clustering of class 1 down near y = -2 while still classifying the the other points correctly.

```
set.seed(342)
sv.simulated <- ksvm(class ~ ., data = data, type = "C-svc",
kernel = "rbfdot", C = 1, cross=20)
pred.class <- predict(object = sv.simulated, newdata = data, type = "response")
```

```
data_predicted <- data.frame(data$x,data$y,pred.class)
colnames(data_predicted) <- c("x","y","predicted_class")

#plot the data colored by its predicted class
ggplot(data_predicted, aes(x=x,y=y, color=predicted_class))+geom_point(size=2)+theme_bw()+ggtitle("Plot
```




```
#confusion matrix
err <- errorMatrix(actual = data$class, predicted = data_predicted$predicted_class)
round(err, 3)
```

```
##       Predicted
## Actual  1  2 Error
##      1 48  2     4
##      2  6 44    12
```

We can also check the training error and cross validation error:

```
sv.simulated@error
```

```
## [1] 0.08
```

```
sv.simulated@cross
```

```
## [1] 0.13
```