Predictive Analytics: practical 2

The `OJ` data set

The `OJ` dataset of the `ISLR` package contains information on which of two brands of orange juice customers purchased and can be loaded in using

```
data(OJ, package = "ISLR")
```

After loading the `caret` package

```
library("caret")
```

Make an initial examination of the relationships between each of the predictors and the response[1]

[1] Use the `plot` function with a model formula or the `pairs` function.

```
par(mfrow = c(4,5), mar= c(4,.5,.5,.5))
plot(Purchase~., data = OJ)
```

Initial model building

- To begin, create a logistic regression model that takes into consideration the prices of the two brands of orange juice, `PriceCH` and `PriceMM`.[2]
- What proportion of purchases does this model get right?
- How does this compare to if we used no model?

[2] Hint: The train function does model fitting, the method argument specifies the type of model. `method = "glm"` is used for logistic regression.

The following code produces a plot of the decision boundary as seen in figure 1.

```
## Set up a grid for prediction
chrange = range(OJ$PriceCH)
mmrange = range(OJ$PriceMM)
chseq = seq(chrange[1],chrange[2],length.out = 100)
mmseq = seq(mmrange[1],mmrange[2],length.out = 100)
grid = expand.grid("PriceCH" = chseq, "PriceMM" = mmseq)

# make the predictions
predictions = predict(m1,grid,type = "prob")
# turn the predictions into a matrix for a contour plot
predmat = matrix(predictions[,2], nrow=100)
contour(chseq, mmseq, predmat, levels = 0.5,
        xlab = "Price CH", ylab = "Price MM",
        lwd = 2, main = "Blue = MM")

# the background points indicating prediction
points(grid,col = c("red","blue")[predict(m1,grid)],
       cex = 0.2)
```

```
# there are few unique combinations of prices,
# jitter can help see the points
# points of prices coloured by purchased brand
points(jitter(OJ$PriceCH,factor = 2),
       jitter(OJ$PriceMM, factor = 2),
       col = c("red","blue")[OJ$Purchase],
       pch = 19, cex  =0.6)

# add dashed line of equal price
abline(0,1,lwd = 2, lty = 2)
```

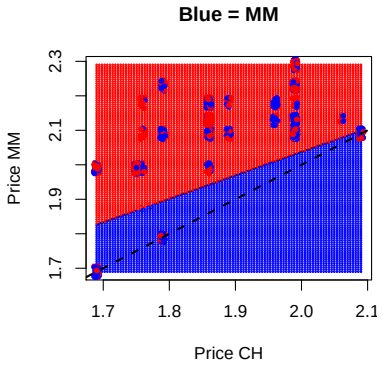

Figure 1: Examining the decision boundary for orange juice brand purchases by price.

- What happens if we add an interaction term? How does the boundary change?

Using all of the predictors

- Fit a logistic regression model using all of the predictors
- Is there a problem?

  We can view the most recent warning messages by using the `warnings` function

```
warnings()

## NULL
```

  This suggests some rank–deficient fit problems,

- Look at the final model, you should notice that a number of parameters have not been estimated

```
?ISLR::OJ
```

  gives further insight, the `PriceDiff` variable is a linear combination of `SalePriceMM` and `SalePriceCH` so we should remove this. In addition we have a `StoreID` variable and a `STORE` variable are different encodings of the same information so we should remove one of these too. We also have `DiscCH` and `DiscMM` which are the differences between `PriceCH` and `SalePriceCH` and `PriceMM` and `SalePriceMM` respectively and `ListPriceDiff` is a linear combination of these prices. Removing all of these vairables allows the model to be fit and all parameters to be estimated.[3]

[3] This is to highlight that we need to understand what we have in our data.

```
OJsub = OJ[!(colnames(OJ) %in% c("STORE", "SalePriceCH",
           "SalePriceMM","PriceDiff", "ListPriceDiff"))]
OJsub$Store7 = as.double(OJsub$Store7) - 1
m.log = train(Purchase ~ ., data = OJsub, method = "glm")
```

The problem of linear combinations of predictors can be shown with this simple theoretical example. Suppose we have a response

y and three predictors x_1, x_2 and the linear combination $(x_1 + x_2)$. On fitting a linear model we try to find estimates of the parameters in the equation

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 (x_1 + x_2).$$

However we could just as easily rewrite this as

$$\begin{aligned} y &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 (x_1 + x_2) \\ &= \beta_0 + (\beta_1 + \beta_3) x_1 + (\beta_2 + \beta_3) x_2 \\ &= \beta_0 + \beta_1^* x_1 + \beta_2^* x_2. \end{aligned}$$

This leads to a rank deficient model matrix, essentially we can never find the value of the β_3 due to the fact we have the linear combination of predictors.

We could acheive the same using the caret package function `findLinearCombos`. The function takes a model matrix as an argument. We can create such a matrix using the `model.matrix` function with our formula object

```
remove = findLinearCombos(model.matrix(Purchase~., data = OJ))
```

The output list has a component called `remove` suggesting which variables should be removed to get rid of linear combinations

```
(badvar = colnames(OJ)[remove$remove])

## [1] "SalePriceMM"   "SalePriceCH"   "PriceDiff"     "ListPriceDiff"
## [5] "STORE"

OJsub = OJ[, -(remove$remove)]
```

- How accurate is this new model using more predictors?]
- What are the values of sensitivity and specificity?
- What does this mean?

ROC curves

If we were interested in the area under the ROC curve, we could retrain the model using the `twoClassSummary` function as an argument to a train control object. Alternatively we can use the `roc` function in the `pROC` package. This also allows us to view the ROC curve, see figure 2.

```
library("pROC")

## Type 'citation("pROC")' for a citation.
##
## Attaching package:  'pROC'
```

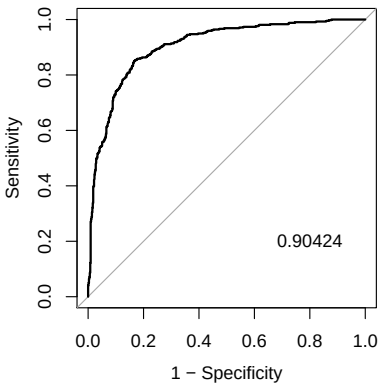

Figure 2: An example of a ROC curve for the logistic regression classifier. We can overlay ROC curves by adding the `add = TRUE` argument.

```
##
## The following objects are masked from 'package:stats':
##
##    cov, smooth, var
```

```
curve = roc(response = OJsub$Purchase,
  predictor = predict(m.log, type = "prob")[,"CH"])
## this makes CH the event of interest
plot(curve, legacy.axes = TRUE)
auc(curve)
```

Other classification models

- Try fitting models using the other classification algorithms we have seen so far.

We have seen LDA, QDA, KNN and logistic regression.

```
## Loading required package:  MASS
```

- How do they compare?
- How does varying the number of nearest neighbours in a KNN affect the model fit?

The KNN algorithm described in the notes can also be used for regression problems. In this case the predicted response is the mean of the k nearest neighbours.

- Try fitting the KNN model for the regression problem in practical 1.
- How does this compare to the linear regression models?

An example with more than two classes

The `Glass` data set in the `mlbench` package is a data frame containing examples of the chemical analysis of 7 different types of glass. The goal is to be able to predict which category glass falls into based on the values of the 9 predictors.

```
data(Glass, package = "mlbench")
```

A logistic regression model is typically not suitable for more than 2 classes, so try fitting the other models using a training set that consists of 90% of the available data.

The function `createDataPartition` can be used here, see notes for a reminder.