Practical 3

Jumping Rivers

Predictive Analytics: practical 3

The `OJ` data set

The `OJ` data set from the `ISLR` package contains information on which of two brands of orange juice customers purchased[1] and can be loaded using

After loading the `caret` and `jrPred` package

make an initial examination of the relationships between each of the predictors and the response[2]

Initial model building using logistic regression

- To begin, create a logistic regression model that takes into consideration the prices of the two brands of orange juice, `PriceCH` and `PriceMM`. Hint: Use the `train` function, with `method = 'glm'`. Look at the help page for the data set to understand what these variables represent.
  - What proportion of purchases does this model get right?
  - How does this compare to if we used no model?
- Use your model to predict if a customer will buy CH or MM if the price of CH and MM is 2.3 and 2.4 respectively

Visualising the boundary

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

```
boundary_plot(m1,OJ$PriceCH, OJ$PriceMM, OJ$Purchase,
              xlab="Price CH", ylab="Price MM")
```

Run the boundary code above, and make sure you get a similar plot.

- What happens if we add an interaction term? How does the boundary change?
- Try adding polynomial terms.

Using all of the predictors

- Instead of just using 2 predictors we want to use all of them. However, we have a few problems to tackle first. A few of our predictors are linear combinations of the others. This leads to what is called rank-deficiency problems. For instance, if you run the following model you'll realise there are a few NAs.

```
mLM = train(Purchase ~ ., data = OJ, method = "glm")
```

[1] The response variable is `Purchase`.
[2] Use the `plot` function with a model formula or the `pairs` function.

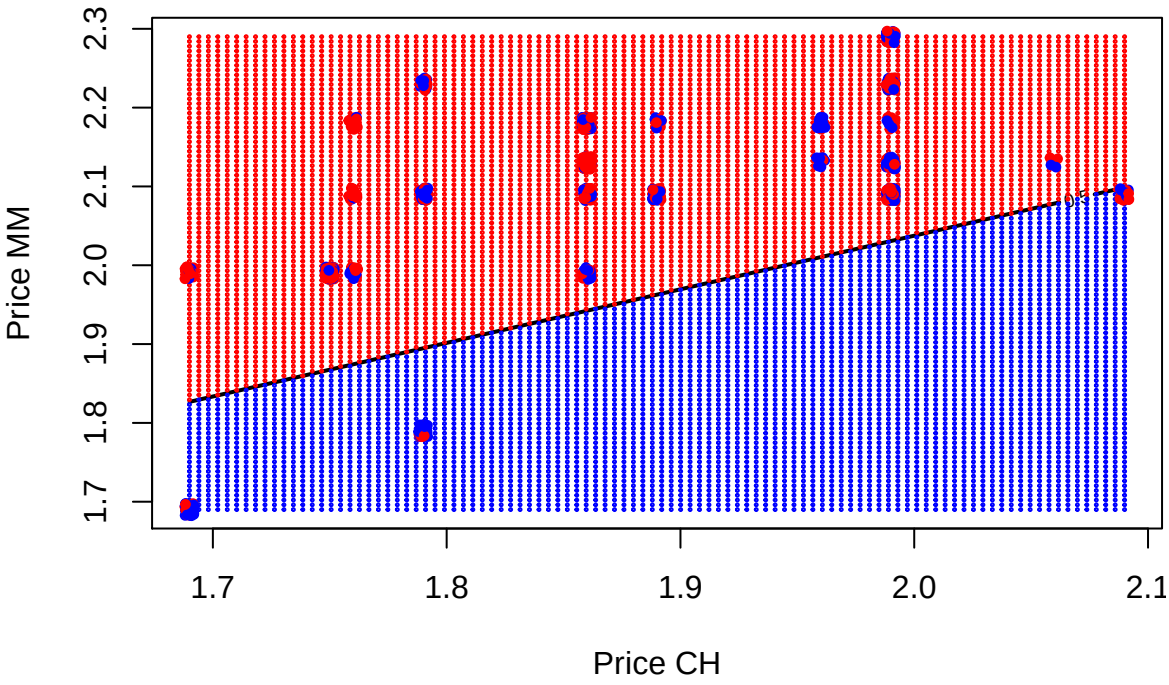

Figure 1: Examining the decision boundary for orange juice brand purchases by price.

Take the predictor PriceDiff. It is impossible to estimate it's coefficient as it is a linear combination of PriceCH and PriceMM i.e. `PriceDiff = PriceCH - PriceMM`. In this particularly data set, there are quite a few linear combinations. We can find them using the `findLinearCombos()` and `model.matrix()` functions

```
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])
```

We can then remove these variable from the data

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

- Use the new `OJsub` data set to model `Purchase` using all of the predictors. How accurate is the model?
- What are the values of sensitivity and specificity?
- What does this mean?

K nearest neigbours

- Try fitting models using the K nearest neighbours algorithm. To begin with, just have two covariates and use the `boundary_plot` function to visualise the results.
- How do they comparein accuracy, sensitivity and specificity?
- 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?

Resampling methods

- Fit a KNN regression model to the `cars2010` data set with `FE` as the response.

```
data(FuelEconomy, package = "AppliedPredictiveModeling")
```

- Estimate test error using 10-fold cross validation
- Again using 10 fold CV, estimate the performance of the k nearest neighbours algorithm for different values of k.
- Which model is chosen as the best?
- Create new `trainControl` objects to specify the use of 5 fold and 15 fold cross validation to estimate test RMSE.
  - Go through the same training procedure attempting to find the best KNN model.

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 a k nearest neighbour model. Use k-fold cross validation is you want to. What proportion of predictions does your model get correct?