Predictive Analytics: practical 1 solutions

Course R package

Installing the course R package[1] is straightforward. First install `drat`[2]

```
install.packages("drat")
```

Then

```
drat::addRepo("rcourses")
install.packages("jrPredictive")
```

This R package contains copies of the practicals, solutions and data sets that we require. It will also automatically install any packages[3] that we use during the course. To load the course package, use

```
library("jrPredictive")
```

During this practical we will mainly use thehe `caret` package, we should load that package as well

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

[1] A package is an *add-on* or a *module*. It provides additional functions and data.
[2] `drat` is a package that makes it easy to host and distribute packages.
[3] For example, we will need the `caret`, `mlbench`, `pROC` and `splines` to name a few.

The `cars2010` data set

The `cars2010` data set contains information about car models in 2010. The aim is to model the `FE` variable which is a fuel economy measure based on 13 predictors.[4]

The data is part of the `AppliedPredictiveModeling` package and can be loaded by

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

[4] Further information can be found in the help page, `help("cars2010", package = "AppliedPredictiveModeling")`.

There are a lot of questions below marked out by bullet points. Don't worry if you can't finish them all, the intention is that there is material for different backgrounds and levels

Exploring the data

- Prior to any analysis we should get an idea of the relationships between variables in the data. Use the `pairs` function to explore the data. The first few are shown in figure 1.

  An alternative to using `pairs` is to specify a plot device that has enough space for the number of plots required to plot the response against each predictor

  ```
  op = par(mfrow = c(3, 5), mar = c(4, 2, 1, 1.5))
  plot(FE ~ ., data = cars2010)
  par(op)
  ```

  We don't get all the pairwise information amongst predictors but it saves a lot of space on the plot and makes it easier to see what's going on. It is also a good idea to make smaller margins.

The `FE ~ .` notation is shorthand for `FE` against all variables in the data frame specified by the `data` argument.

```
## Error in setnicepar(mfrow = c(1, 2)):  could not find func-
tion "setnicepar"
## Error in mypalette(1):  could not find function "mypalette"
```

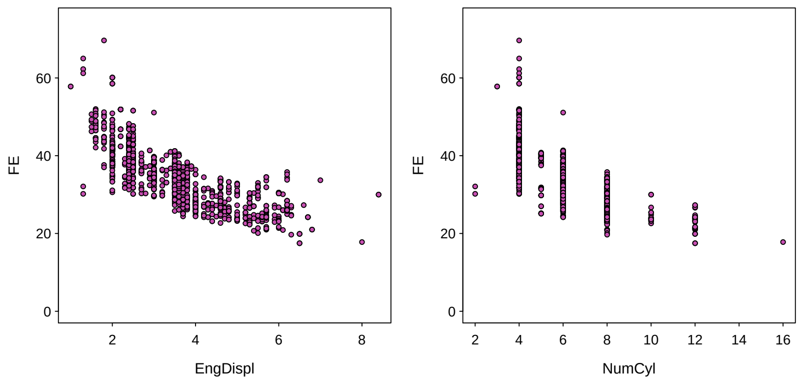

Figure 1: Plotting the response against some of the predictor variables in the cars2010 data set.

- Create a simple linear model fit of FE against EngDispl using the train function.[5]

```
m1 = train(FE ~ EngDispl, method = "lm", data = cars2010)
```

[5] Hint: use the train function with the lm method.

- Examine the residuals of this fitted model, plotting residuals against fitted values

```
rstd = rstandard(m1$finalModel)
plot(fitted.values(m1$finalModel), rstd)
```

We can add the lines showing where we expect the residuals to fall to aid graphical inspection

```
abline(h = c(-2, 0, 2), col = 2:3, lty = 2:1)
```

```
## Error in setnicepar():
could not find function
"setnicepar"
## Error in mypalette(1):
could not find function
"mypalette"
```

- What do the residuals tell us about the model fit using this plot?

```
# There definitely appears to be some trend in the
# residuals.  The curved shape indicates that we
# potentially require some transformation of variables.
# A squared term might help.
```

- Plot the fitted values vs the observed values

```
## Error in setnicepar():  could not find function "setnicepar"
## Error in mypalette(1):  could not find function "mypalette"

plot(cars2010$FE, fitted.values(m1$finalModel), xlab = "FE",
    ylab = "Fitted values", xlim = c(10, 75), ylim = c(10,
        75))
abline(0, 1, col = 3, lty = 2)
```

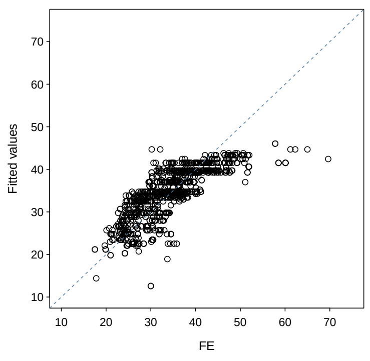

Figure 2: Plot of fitted against observed values. It's always important to pay attention to the scales.

  - What does this plot tell us about the predictive performance of this model across the range of the response?

```
# We seem to slightly over estimate more often than not
# in the 25-35 range. For the upper end of the range we
# seem to always under estimate the true values.
```

- Produce other diagnostic plots of this fitted model, e.g. a q-q plot

```
qqnorm(rstd)
qqline(rstd)
plot(cars2010$EngDispl, rstd)
abline(h = c(-2, 0, 2), col = 2:3, lty = 1:2)
```

- Are the modelling assumptions justified?

```
# We are struggling to justify the assumption of
# normality in the residuals here, all of the diagnostics
# indicate patterns remain in the residuals that are
# currently unexplained by the model.
```

Extending the model

- Do you think adding a quadratic term will improve the model fit?

```
# We are struggling to justify the assumption of
# normality in the residuals here, all of the diagnostics
# indicate patterns remain in the residuals that are
# currently unexplained by the model
```

- Fit a model with the linear and quadratic terms for `EngDispl` and call it `m2`

```
m2 = train(FE ~ poly(EngDispl, 2, raw = TRUE), data = cars2010,
    method = "lm")
```

  - Assess the modelling assumptions for this new model.
  - How do the two models compare?

```
# The residual diagnostics indicate a better fit now that
# the quadratic term has been included.
```

- How does transforming the response variable affect the fit?

Common transformations may be a log or square root function.

```
# Perhaps the residuals more closely match the assumption
# of normality under this transformation. However we need
# to be careful about interpretation now as the response
# is on the log scale. Likewise for prediction we need to
# remember to undo the transformation.
```

- Add `NumCyl` as a predictor to the simple linear regression model `m1` and call it `m3`

```
m3 = train(FE ~ EngDispl + NumCyl, data = cars2010, method = "lm")
```

- Examine model fit and compare to the original.
- Does the model improve with the addition of an extra variable?

Visualising the model

The `jrPredictive` package contains a `plot3d` function to help with viewing these surfaces in 3D as in figure 3.[6]

```r
## points = TRUE to also show the points
plot3d(m3, cars2010$EngDispl, cars2010$NumCyl, cars2010$FE,
    points = FALSE)
```

[6] We can also add the observed points to the plot using the `points` argument to this function, see the help page for further information.

We can also examine just the data interactively, via

```r
threejs::scatterplot3js(cars2010$EngDispl, cars2010$NumCyl,
    cars2010$FE, size = 0.5)
```

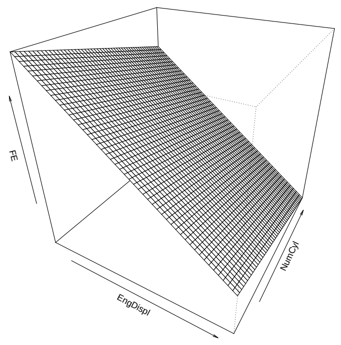

Figure 3: A surface plot from a linear model of fuel economy against the number of cylinders and engine displacement including the interaction term.

- Try fitting other variations of this model using these two predictors. For example, try adding polynomial and interaction terms

  ```r
  m4 = train(FE ~ EngDispl * NumCyl + I(NumCyl^5), data = cars2010,
      method = "lm")
  ```

  How is prediction affected in each case? Don't forget to examine residuals, R squared values and the predictive surface.
- If you want to add an interaction term you can do so with the `:` operator, how does the interaction affect the surface?

Other data sets

A couple of other data sets that can be used to try fitting linear regression models.

Data set	Package	Response
diamonds	`ggplot2`	price
Wage	`ISLR`	wage
BostonHousing	`mlbench`	medv