Practical 1 Solutions

Jumping Rivers

Course R package

Installing the course R package is straightforward. First install `drat`, a package that makes it easy to host and distribute packages.

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

Then

```
drat::addRepo("jr-packages")
install.packages("jrPred")
```

This R package contains copies of the practicals, solutions and data sets that we require. It will also automatically install any packages, that we use during the course. For example, we will need the `caret`, `mlbench`, `pROC` and `splines` to name a few. To load the course package, use

```
library("jrPred")
```

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

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

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

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

```
data(FuelEconomy, 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

An Initial Model

- 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.
- Create a simple linear model fit of `FE` against `EngDispl` using the `train` function. Hint: use the `train` function with the `lm` method.

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

- Using your model, what level of FE would you expect a car with an `EngDispl` of 7 to have?

```
predict(m1, newdata = data.frame(EngDispl = 7))
```

```
##        1
## 18.91672
```

- What is the training error rate (RMSE) for this model?

```
sqrt(mean(resid(m1)^2))
```

```
## [1] 4.620076
```

```
# or
RMSE(fitted.values(m1), cars2010$FE)
```

```
## [1] 4.620076
```

Extending 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")
```

- How do the two models compare in training error rate? Have we improved the model?

```
sqrt(mean(resid(m2)^2)) - sqrt(mean(resid(m1)^2))
```

```
## [1] -0.3852504
```

```
# Yes
```

- 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")
```

- What is the RMSE for `m3`?

```
sqrt(mean(resid(m3)^2))
```

```
## [1] 4.59588
```

- Does the model improve with the addition of an extra variable?

Visualising the models

- Plot `EngDispl` against `Fe`

```
plot(cars2010$EngDispl, cars2010$FE)
```

- We can use the `abline()` function to overfit a model with one predictor that is linear in the x and y axes. Try running

```
abline(m1$finalModel, col = 2)
```

- For `m2`, the model is a quadratic fit. The `abline()` function only draws straight lines and so is no longer suitable. We'll switch to the `lines()` function. By overplotting the original points against the a sequence of predicted values we'll get a representation of the fit

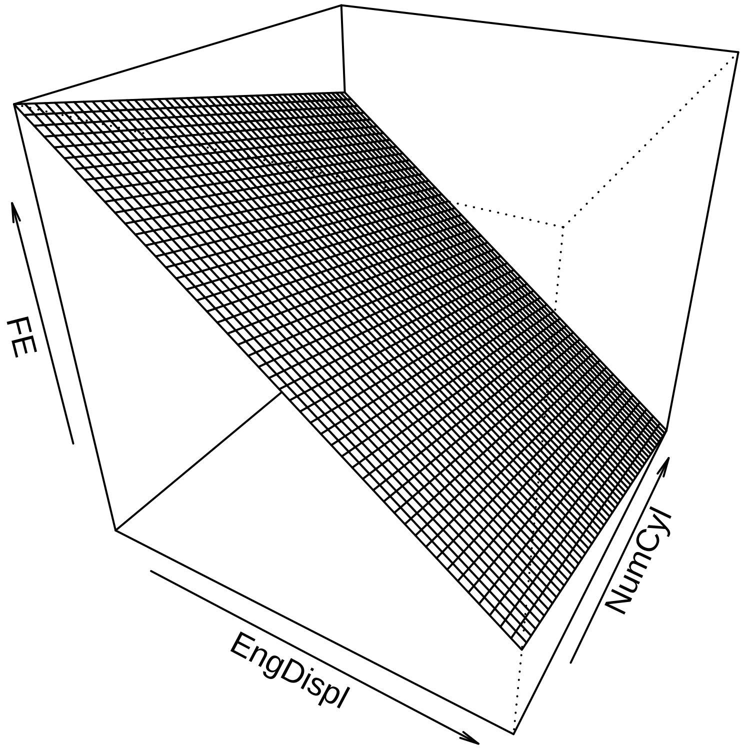
```
x_values = seq(1,8.4,0.1)
new_pred_values = predict(m2, newdata = data.frame(EngDispl = x_values)
lines(x = x_values, y = new_pred_values, col = 3)
```

- Does this back up the RMSE results from the previous question?

```
# Yes, line looks to curve with the data now we have added a quadratic term
```

- It's harder to compare the visually compare `m3` with `m2` and `m1` as we now have more than 1 predictor. The `jrPred` package contains a `plot3d` function to help with viewing these surfaces in 3D.

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



We can also examine just the data interactively, via

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