Predictive Analytics: practical 2

Cross validation and the bootstrap

- Fit a linear regression model to the `cars2010` data set with `FE` as the response, using `EngDispl`, `NumCyl` and `NumGears` as predictors.
- What is the training error rate (RMSE) for this model?
- Re–train your model using the validation set approach to estimate a test RMSE, make your validation set equivalent to a half of the full available data.

The data set can be loaded `data("FuelEconomy", package = "AppliedPredictiveModeling")`.

Hint: The training error can be found by taking the square root of the average square residuals. The `sqrt` and `resid` functions may be useful.

- How does this compare to the training error that we estimated above?
- Go through the same process using the different methods for estimating test error. That is leave one out and k–fold crossvalidation as well as bootstrapping.

10–fold cross validation can be shown to be a good choice for almost any situation.

- How do these estimates compare with the validation set approach?
- The object returned by `train` also contains timing information that can be accessed via the `times` component of the list. Which of the methods is fastest?

The $ notation can be used pick a single list component.

- Using k–fold cross validation to estimate test error investigate how the number of folds effects the resultant estimates and computation time.
- Experiment with adding terms to the model, transformations of the predictors and interactions say and use cross validation to estimate test error for each. What is the best model you can find? You can still use the `validate` and `mark` functions to look at how your models fair on the unseen data.

Penalised regression

The `diabetes` data set in the `lars` package contains measurements of a number of predictors to model a response y, a measure of disease progression. There are other columns in the data set which contain interactions so we will extract just the predictors and the response. The data has already been normalized.

```r
## load the data in
data(diabetes, package = "lars")
diabetesdata = cbind(diabetes$x, y = diabetes$y)
```

- Try fitting a lasso, ridge and elastic net model using all of the main effects, pairwise interactions and square terms from each of the predictors.[1]
- Try to narrow in on the region of lowest RMSE for each model, don't forget about the `tuneGrid` argument to the train function.
  We can view the coefficients via

[1] Hint: see notes for shortcut on creating model formula. Also be aware that if the predictor is a factor a polynomial term doesn't make sense
`fraction = 0` is the same as the null model.
$y \sim (.) \wedge 2$ is short hand for a model that includes pairwise interactions for each predictor, so if we use this we should only need to add the square terms

```r
coef = predict(mLASSO$finalModel,
  mode = "fraction",
  s = mLASSO$bestTune$fraction,# which ever fraction was chosen as best
  type = "coefficients"
)
```

- How many features have been chosen by the `lasso` and `enet` models?
- How do these models compare to principal components and partial least squares regression?