ES7023 - Assignment 4 : Cross-Validation (CV)

K-fold Cross-validation (CV)

In this assignment we'll be exploring cross-validation (CV). Specifically, K-fold cross-validation is a way to estimate model prediction error. It utilizes the standard technique of splitting your dataset into a training set (to fit your model) and a test set (to get test errors on new data that your model has not seen before). However, it does this many (K) times, each time using a different test set. The idea is then that you can actually use all of your data to train the model, and use K-fold CV to estimate the prediction error.

Data

You are provided with a csv file `Air Quality Index Delhi.csv` that contains several variables:

- `pm25` - particulate matter smaller than 2.5 microns (micrograms per cubic meter).
- `T` - average temperature (°C)
- `TM` - maximum temperature (°C)
- `H` - average relative humidity (%)

Problem 1 - Data exploration

- Read the file `Air Quality Index Delhi.csv` into a data-frame
- Provide summary descriptive statistics of the data (typically mean, standard deviation etc…)
  - using base R functions
  - using dplyr
- Provide descriptive plots for each variable (e.g. histogram, density-plot, box-plot choose what is best)
- Try to visualize the relationships between the different parameters
  - using ggplot2
  - and other tools - example: corrplot (see reference below)

Notes

- Try to use R functions from the tidyverse packages (`dplyr, ggplot2, tidyr, readr`)
- Other packages that you can explore:
  - summarytools: https://cran.r-project.org/web/packages/summarytools/vignettes/Introduction.html
  - corrplot : https://cran.r-project.org/web/packages/corrplot/vignettes/corrplot-intro.html
- Label all your figures

Problem 2 - Prediction model

Now let's say we want to predict PM2.5 levels based on the other predictor variables in our dataset. We'll build a simple linear model.

```
mod<-lm(formula = pm25~ T + TM + H, data = data)
summary(mod)
```

The summary description of the our fitted model (output of `summary(mod)`) gives us some information about the model errors. This however tells us only about the goodness of fit to the training data, but little about the prediction errors (expected errors on new data). For this, we will perform K-fold cross-validation.

We need to decide on a loss function to define our metric for errors. The loss function we will use is the 'root mean squared error.' For simplicity, this function is provided (you do not need to modify it). The function assumes that we wish to use a linear regression model, and that all variables in the data.frame will be used to predict y. The data.frame `train` will include the data used to train the model, and the data.frame `test` includes the data to test the model.

```
rmse = function(train,test) {
   fit = lm(y~., data=train)  #PERFORMS A LINEAR REGRESSION
   train.err = sqrt(mean((fit$resid)^2))  #RMS OF TRAINING RESIDUALS
   test.err = sqrt(mean((test$y - predict(fit,test))^2)) #RMS CV RESIDUALS
   c(train.err,test.err)        #RETURN BOTH
}
```

The `lm` function performs a linear regression of y on other variables in data.frame train (the dot in `lm(y~.,data=train)` tells R to use all remaining variables in the data frame `train` as predictor variables, otherwise we could specify the name of specific variables using `train$var_name`).

Now let's develop and conduct cross-validation on our data:

- Since the `rmse()` function we created expects a response variable y in the training and test set, rename the `pm25` variable in your data frame to `y` using dplyr (so that you can feed the data.frame to the `rmse()` function).
- Write a function called `cv.err` that takes as input a `data.frame` and a variable K. K corresponds to the number of splits used in cross validation. *Hint*: the first steps in this function should be to determine the total number of samples, and then identify K roughly equally sized groups of data. You can use the function `cut()` to do this: the command `groups = cut(1:n,K,label=F)`creates a vector of length n where each component specifies the group number of the corresponding component in the vector $1 : n$. Once you have K different groups, run the rmse function K times, each time using a different subset as the test data. *Note*: the cv.err function should be fewer than 15 lines, and should return a [K x 2] matrix of training and test errors.
- Conduct K-fold cross validation on your data. For a K-fold CV, you will have K estimates of both training and test error, which means you should compute both the mean and standard deviation of the CV error. You might want to try out several different values of K (e.g. K=5, K=10).
- Briefly discuss the results.

Problem 3 - Model Evaluation

- Add three more columns to the dataset with random numbers $N(0, 1)$:

```
data <- data %>% mutate( x4=rnorm(nrow(data)),
                         x5=rnorm(nrow(data)),
                         x6=rnorm(nrow(data)))
```

- Compute the training and cv error (using a value of K of your choice, e.g. $K = 5$) for a model that uses just the first predictor variable, then first two predictor variables, the first three, and so on until it uses all 6 variables.
- Provide a plot of the training and test cv error as a function of the number of predictor variables.
- Show both the mean and standard deviation of the errors on the same plot using `ggplot2` package.
- Answer the following questions:
  - Does this show what you'd expect?
  - Is the test error consistently above/below the training error?
- You may wish to try this for a few different values of K and/or different levels of standard deviation in the noise (e.g., $N(0, 2)$).