Lab 05 STATS 769 Due 4:00pm Tuesday 3 September

The main point of this lab is to **scrape data from the Web**. We will again generate a literate report that describes a simple analysis, but we will go back to predicting long trips (a binary variable) based on trip duration. In addition to fitting a logistic regression as before, we will also fit a k-nearest neighbours model (see the section at the end of this document).

The Data

The data are again electronic bicycle or scooter trips, but we will access the data from the original source: `data.austintexas.gov`, the official City of Austin data portal.

The "Dockless Vehicle Trips" data set that we have been using has its main page here:
`https://data.austintexas.gov/Transportation-and-Mobility/Dockless-Vehicle-Trips/7d8e-dm7r`
We will use the API that is described here:
`https://dev.socrata.com/foundry/data.austintexas.gov/7d8e-dm7r`
NOTE: you will need to register for an app token in order to use the API.

The Task

1. Scrape data from the City of Austin data portal and create a data frame with two columns: `trip_distance` and `trip_duration`. You should extract 10,000 scooter trips that occurred in 2018 (your API request must specify records with `year` equal to 2018 and `vehicle_type` equal to "scooter").
2. Subset only trips with non-negative distances and durations, log the durations, and create a new "long trip" variable (where "long" means that the trip distance was greater than 1000m).
3. Fit a logistic regression model to the training data and calculate the accuracy of the model on the test data.
4. Fit a k-nearest neighbours model and calculate its accuracy. You might need to try a few different values of k.
5. Produce a plot based on the test data that shows the predictions from the logistic regression model and the predictions from the k-nearest neighbours model.

The Report

Your submission should consist of a *tar ball* (as generated by the `tar` shell tool) that contains an R Markdown document *and* a Makefile *and* a processed version of your R Markdown document, submitted via Canvas.

You should write your document and your Makefile so that the tar ball can be extracted into a directory anywhere on one of the virtual machines provided for this course (`sc-cer00014-04.its.auckland.ac.nz` or `sc-cer00014-05.its.auckland.ac.nz`) and the markdown document can be processed just by typing `make`.

Your report should include:

- A description of the data format.
- A description of the method used to scrape the data into R, inclduing an explanation of your API request.
- A comparison of logistic regression and k-nearest neighbours.
- A conclusion summarising the analysis.

Your report should NOT be longer than **10 pages**.

Marks will be lost for:

- Submission is not a tar ball.
- More than 10 pages in the report.
- R Markdown file does not run.
- Section of the report is missing.
- R Markdown file is missing.
- Processed file (pdf or docx or html) is missing.
- Makefile is missing.
- Significantly poor R (or other) code.

k-nearest neighbours

A k-nearest neighbours (classification) model predicts y for a set of test x based on the majority vote from a set of training y, only using the nearest k neighbours, where distance between neighbours is (by default) euclidean distance between the test x and the training x values.

The k-nearest neighbours model allows for much greater flexibility than a simple logistic regression; simple logistic regression only allows a single (smooth) transition between two categories, but k-nearest neighbours allows both sharp transitions and more than one transition between categories.

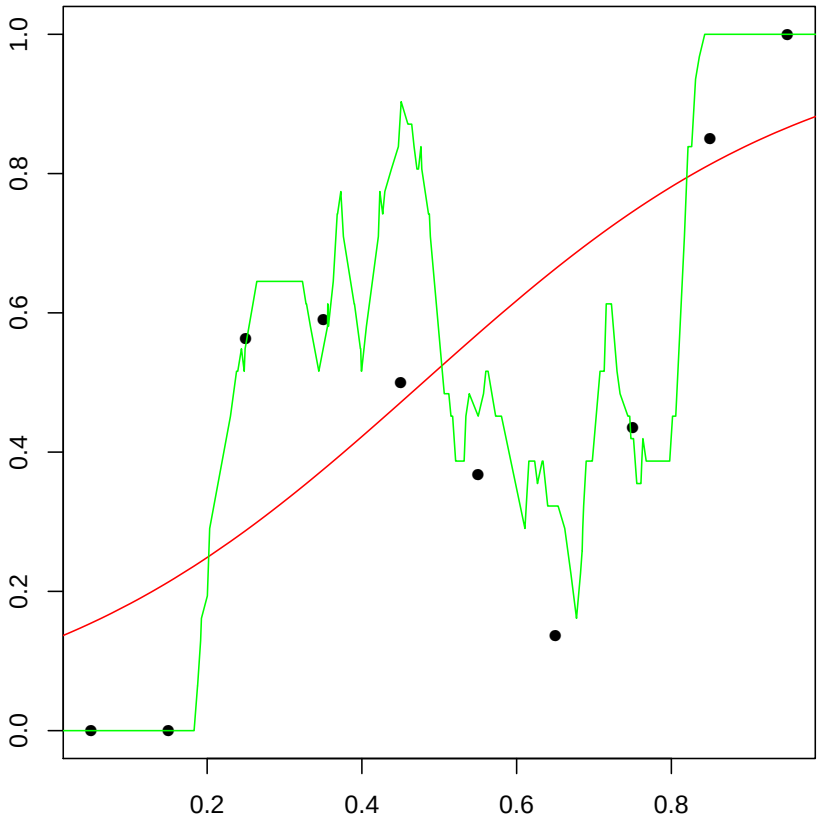
The danger with this greater flexibility is a higher risk of over-fitting; a trained model may pay too much attention to details in the training set, which can then lead to more errors in a test set.

The choice of k is critical; ideally, we get a balance between flexibility and generality, so our model captures more complex trends, but only genuine trends. We typically choose an odd number for k (to reduce chance of ties).

We can use the `knn()` function from the **class** package to generate predictions for a k-nearest neighbours.

```
> ## KNN example on randomly generated data
> library(class)
> library(caret)
> set.seed(10)
> x <- sort(runif(1000))
> y <- c(rep(0, 200), sample(0:1, 300, replace=TRUE, prob=c(.4, .6)),
+        sample(0:1, 300, replace=TRUE, prob=c(.6, .4)), rep(1, 200))
> testset <- sort(sample(1:1000, 200))
> trainx <- x[-testset]
> trainy <- y[-testset]
> testx <- x[testset]
> testy <- y[testset]
```

```
> glmFit <- glm(y ~ x, data.frame(y=trainy, x=trainx), family="binomial")
> glmProb <- predict(glmFit, data.frame(x=testx), type="response")
> glmPred <- ifelse(glmProb > .5, 1, 0)
> glmDiag <- confusionMatrix(factor(glmPred), factor(testy))
> glmDiag$table

          Reference
Prediction  0  1
         0 62 30
         1 48 60

> glmDiag$overall["Accuracy"]

Accuracy
    0.61

> knnPred <- knn(matrix(trainx, ncol=1),
+                matrix(testx, ncol=1),
+                trainy,
+                k=31, prob=TRUE)
> knnProb <- ifelse(knnPred == 1,
+                   attr(knnPred, "prob"),
+                   1 - attr(knnPred, "prob"))
> knnDiag <- confusionMatrix(knnPred, factor(testy))
> knnDiag$table

          Reference
Prediction  0  1
         0 79 19
         1 31 71

> knnDiag$overall["Accuracy"]

Accuracy
    0.75

> par(mar=c(3, 3, 2, 2))
> breaks <- seq(0, 1, .1)
> midbreaks <- breaks[-1] - diff(breaks)/2
> props <- tapply(testy, cut(testx, breaks), mean)
> plot(midbreaks, props, pch=16)
> lines(testx, glmProb, col="red")
> lines(testx, knnProb, col="green")
```


0.0
0.2
0.4
0.6
0.8
1.0
0.2
0.4
0.6
0.8