STATS 769 - Lab 05 - bole001

Bernard O’Leary

02 September 2019

# Data format and API call details

Data is in JSON format as it comes down from the API. It is a JSON array of “trip” objects. The API allows query-sting based parameters to be passed as part of an HTTP GET request, which means that we don’t need to send an HTTP body. The paramters that we use are *limit* to tell the API to give us back a specific number of JSON rows. Without this parameter we get back 1000 by default. We also send a *year* and a *vehicle\_type* paramter to further limit our result set. Finally an *api\_token* parameter is sent to authenticate our request. I am using a combination of the the *httr* and *jsonlite* libraries to make a simple GET request and then transform the result into a dataframe.

The following code imports the data from the API and lists dataframe dimensions and first 6 rows of the data for validation purposes.

library(httr)  
library(jsonlite)

## Loading required package: methods

# Call the resource  
json\_result <- GET("https://data.austintexas.gov/resource/7d8e-dm7r.json?$limit=10000&year=2018&vehicle\_type=scooter&$$app\_token=xo9XRO6BRv2CwPMK9RfhyFzN6")  
  
# Show data frame  
trips <- fromJSON(content(json\_result, as="text"))  
head(trips)

## trip\_id  
## 1 9643609b-f548-4ccc-b96c-cc625a34a6c1  
## 2 c998b4ad-a6b0-4bda-bcf1-439f8dec79e0  
## 3 51d3b1d9-90f0-4c10-a4f7-83123a697fe4  
## 4 1b666bdb-8031-45db-9094-a0b4dcb19395  
## 5 a858eeaf-062e-40e2-b275-cd31df2575a7  
## 6 05aee845-4976-44a0-9da8-24399e7bbc3c  
## device\_id vehicle\_type trip\_duration  
## 1 a8813574-e5fe-4ce3-ac18-924ddedd1c7e scooter 46  
## 2 b14555c3-cb30-495b-92e1-18800f64d0df scooter 486  
## 3 830a59d0-78aa-4297-80ed-c40773c0a87d scooter 197  
## 4 49d245ec-f8f9-4529-a7df-276ff1d147d4 scooter 6  
## 5 830a59d0-78aa-4297-80ed-c40773c0a87d scooter 108  
## 6 ab3ed022-07cf-4077-9f29-da30a8fd793e scooter 13  
## trip\_distance start\_time end\_time  
## 1 0 2018-10-07T00:45:00.000 2018-10-07T00:45:00.000  
## 2 0 2018-11-18T00:45:00.000 2018-11-18T01:00:00.000  
## 3 0 2018-09-30T00:00:00.000 2018-09-30T00:00:00.000  
## 4 0 2018-12-02T00:00:00.000 2018-12-02T00:00:00.000  
## 5 0 2018-09-30T00:15:00.000 2018-09-30T00:15:00.000  
## 6 0 2018-11-18T00:15:00.000 2018-11-18T00:15:00.000  
## modified\_date month hour day\_of\_week council\_district\_start  
## 1 2019-04-17T01:40:51.000 10 0 0 9  
## 2 2019-04-17T01:53:51.000 11 0 0 3  
## 3 2019-04-17T04:24:37.000 9 0 0 9  
## 4 2019-04-17T04:34:17.000 12 0 0 9  
## 5 2019-04-17T02:15:42.000 9 0 0 9  
## 6 2019-04-17T01:44:22.000 11 0 0 9  
## council\_district\_end year census\_geoid\_start census\_geoid\_end  
## 1 9 2018 48453001100 48453001100  
## 2 9 2018 48453002304 48453001100  
## 3 9 2018 48453000604 48453000601  
## 4 9 2018 48453000601 48453000601  
## 5 9 2018 48453000603 48453000603  
## 6 9 2018 48453001100 48453001100

dim(trips)

## [1] 10000 16

## Cleanse dataset

Subset only trips with non-negative distances and durations, create a new *long\_trip* variable (where “long” means that the trip distance was greater than 1000m). We exclude non-positive durations and distances and make another new variable that is a log of the duration variable called *logged\_trip\_duration*. Show the first six rescords of the dataframe so that we can see these new variables and validate.

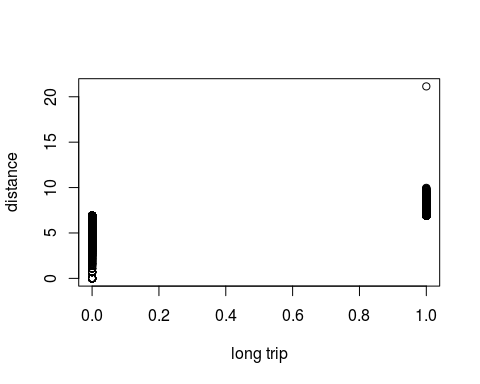
model\_trips <- subset(trips, as.integer(trip\_duration) > 0 & as.integer(trip\_distance) > 0)  
model\_trips$logged\_trip\_duration <- log(as.integer(model\_trips$trip\_duration))  
model\_trips$long\_trip <- as.integer(model\_trips$trip\_distance) > 1000  
head(model\_trips)

## trip\_id  
## 24 8df8fc9d-1272-4cc4-8646-789263296977  
## 25 ee90620e-e662-4a45-b34d-97c147b2be45  
## 26 e3332e1c-432b-4204-a9d6-04637e8dab7e  
## 27 e63ac1b4-e46f-40de-899b-afcd9e6a7f75  
## 28 ee0c177d-4d6b-4ef3-9d49-f65565989c45  
## 29 97f38c39-18ed-4620-9c4b-c6e55f6b7bd4  
## device\_id vehicle\_type trip\_duration  
## 24 8aaf96e5-6e22-4e4c-a2f8-6db00c900aca scooter 221  
## 25 c537d2ef-f112-44fa-a0ea-55a4152e53f3 scooter 1107  
## 26 771b40ad-8d45-47ff-b94d-58d4f5bd2371 scooter 451  
## 27 16740fa3-07fa-482b-940f-1902b3c4be36 scooter 437  
## 28 e3b7a489-92b6-43a6-8ceb-c837a38621d5 scooter 901  
## 29 430bd20e-666e-41d6-81fb-cdec7db29cc4 scooter 150  
## trip\_distance start\_time end\_time  
## 24 960 2018-06-21T17:45:00.000 2018-06-21T17:45:00.000  
## 25 4555 2018-08-07T18:00:00.000 2018-08-07T18:15:00.000  
## 26 1155 2018-06-21T17:45:00.000 2018-06-21T18:00:00.000  
## 27 1322 2018-06-21T17:45:00.000 2018-06-21T18:00:00.000  
## 28 1867 2018-06-21T17:45:00.000 2018-06-21T18:00:00.000  
## 29 161 2018-06-21T17:45:00.000 2018-06-21T17:45:00.000  
## modified\_date month hour day\_of\_week council\_district\_start  
## 24 2019-04-17T01:39:54.000 6 17 4 1  
## 25 2019-04-17T01:43:38.000 8 18 2 9  
## 26 2019-04-17T01:39:54.000 6 17 4 3  
## 27 2019-04-17T01:39:54.000 6 17 4 9  
## 28 2019-04-17T01:39:54.000 6 17 4 9  
## 29 2019-04-17T01:39:54.000 6 17 4 9  
## council\_district\_end year census\_geoid\_start census\_geoid\_end  
## 24 9 2018 48453000401 48453000601  
## 25 9 2018 48453001100 48453001305  
## 26 1 2018 48453000902 48453000901  
## 27 9 2018 48453001100 48453001100  
## 28 9 2018 48453001100 48453000604  
## 29 9 2018 48453000401 48453000401  
## logged\_trip\_duration long\_trip  
## 24 5.398163 FALSE  
## 25 7.009409 TRUE  
## 26 6.111467 TRUE  
## 27 6.079933 TRUE  
## 28 6.803505 TRUE  
## 29 5.010635 FALSE

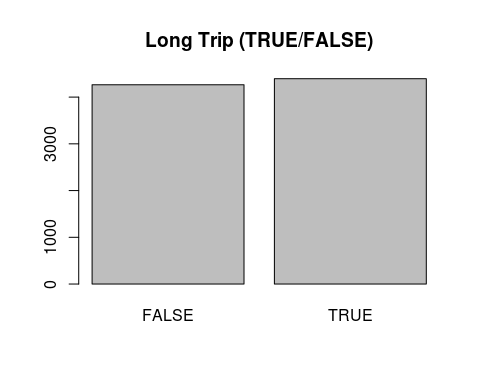
## Boxplot the data (to see what we have)

Take a look at the data that we have come back from the API call. We have at least one significant outlier. Other than that looks like we have perhaps a

plot(log(as.integer(model\_trips$trip\_distance)) ~ model\_trips$long\_trip, ylab="distance", xlab="long trip")



barplot(table(model\_trips$long\_trip), main="Long Trip (TRUE/FALSE)")



# Comparison of logistic regression and k-nearest neighbours models

We fit a logistic model that predicts the proportion of long trips as a function of trip duration and output the confusion matrix. We do that same for KNN and then try to plot a chat of the result. Unfortunately I have not been able to produce a chart successfully.

test\_index <- sample(nrow(model\_trips), nrow(model\_trips) \* 0.1)  
test\_trips <- model\_trips[test\_index, ]  
train\_trips <- model\_trips[-test\_index, ]  
  
overall\_proportion <- mean(model\_trips$long\_trip)  
glm\_fit <- glm(y ~ x, data.frame(x=train\_trips$logged\_trip\_duration, y=train\_trips$long\_trip), family="binomial")  
  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: replacing previous import by 'rlang:::=' when loading 'dplyr'

## Warning: replacing previous import by 'rlang::.data' when loading 'dplyr'

## Warning: replacing previous import by 'rlang::as\_label' when loading  
## 'dplyr'

## Warning: replacing previous import by 'rlang::as\_name' when loading 'dplyr'

## Warning: replacing previous import by 'rlang::dots\_n' when loading 'dplyr'

## Warning: replacing previous import by 'rlang::enquo' when loading 'dplyr'

## Warning: replacing previous import by 'rlang::enquos' when loading 'dplyr'

## Warning: replacing previous import by 'rlang::expr' when loading 'dplyr'

## Warning: replacing previous import by 'rlang::sym' when loading 'dplyr'

## Warning: replacing previous import by 'rlang::syms' when loading 'dplyr'

## Warning: replacing previous import by 'rlang::!!' when loading 'recipes'

## Warning: replacing previous import by 'rlang::as\_character' when loading  
## 'recipes'

## Warning: replacing previous import by 'rlang::call2' when loading 'recipes'

## Warning: replacing previous import by 'rlang::exec' when loading 'recipes'

## Warning: replacing previous import by 'rlang::expr' when loading 'recipes'

## Warning: replacing previous import by 'rlang::f\_lhs' when loading 'recipes'

## Warning: replacing previous import by 'rlang::f\_rhs' when loading 'recipes'

## Warning: replacing previous import by 'rlang::is\_empty' when loading  
## 'recipes'

## Warning: replacing previous import by 'rlang::is\_quosure' when loading  
## 'recipes'

## Warning: replacing previous import by 'rlang::na\_dbl' when loading  
## 'recipes'

## Warning: replacing previous import by 'rlang::names2' when loading  
## 'recipes'

## Warning: replacing previous import by 'rlang::quo' when loading 'recipes'

## Warning: replacing previous import by 'rlang::quo\_get\_expr' when loading  
## 'recipes'

## Warning: replacing previous import by 'rlang::quo\_squash' when loading  
## 'recipes'

## Warning: replacing previous import by 'rlang::quo\_text' when loading  
## 'recipes'

## Warning: replacing previous import by 'rlang::quos' when loading 'recipes'

## Warning: replacing previous import by 'rlang::sym' when loading 'recipes'

## Warning: replacing previous import by 'rlang::syms' when loading 'recipes'

## Warning: replacing previous import by 'tibble::tibble' when loading  
## 'recipes'

## Warning: replacing previous import by 'plyr::ddply' when loading 'caret'

## Warning: replacing previous import by 'recipes::all\_outcomes' when loading  
## 'caret'

## Warning: replacing previous import by 'recipes::all\_predictors' when  
## loading 'caret'

## Warning: replacing previous import by 'recipes::bake' when loading 'caret'

## Warning: replacing previous import by 'recipes::has\_role' when loading  
## 'caret'

## Warning: replacing previous import by 'recipes::juice' when loading 'caret'

## Warning: replacing previous import by 'recipes::prep' when loading 'caret'

##   
## Attaching package: 'caret'

## The following object is masked from 'package:httr':  
##   
## progress

glm\_prob <- predict(glm\_fit, data.frame(x=as.integer(test\_trips$trip\_distance)), type="response")  
glm\_pred <- ifelse(glm\_prob > .5, 1, 0)  
glm\_diag <- confusionMatrix(factor(glm\_pred, levels = 1:0),   
 factor(as.integer(test\_trips$long\_trip), levels = 1:0))  
glm\_diag$table

## Reference  
## Prediction 1 0  
## 1 437 420  
## 0 0 8

glm\_diag$overall["Accuracy"]

## Accuracy   
## 0.5144509

train\_x <- as.data.frame(train\_trips$logged\_trip\_duration)  
test\_x <- as.data.frame(test\_trips$logged\_trip\_duration)  
train\_y <- as.data.frame(as.integer(train\_trips$long\_trip))  
cl = train\_y[,1, drop = TRUE]  
  
library(class)  
knn\_pred <- knn(train\_x, test\_x, cl, k=31, prob=TRUE)  
knn\_prob <- ifelse(knn\_pred == 1, attr(knn\_pred, "prob"), 1 - attr(knn\_pred, "prob"))  
knn\_diag <- confusionMatrix(factor(knn\_pred, levels = 1:0),   
 factor(as.integer(test\_trips$long\_trip), levels = 1:0))  
knn\_diag$table

## Reference  
## Prediction 1 0  
## 1 386 109  
## 0 51 319

knn\_diag$overall["Accuracy"]

## Accuracy   
## 0.8150289

par(mar=c(3, 3, 2, 2))  
breaks <- seq(0, 1, .1)  
midbreaks <- breaks[-1] - diff(breaks)/2  
class(test\_x)

## [1] "data.frame"

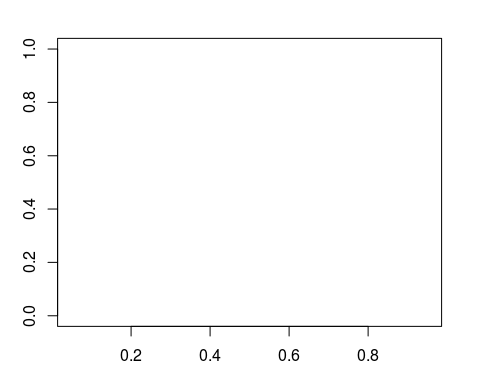
props <- tapply(as.integer(test\_trips$long\_trip), cut(test\_trips$logged\_trip\_duration, breaks), mean)  
props

## (0,0.1] (0.1,0.2] (0.2,0.3] (0.3,0.4] (0.4,0.5] (0.5,0.6] (0.6,0.7]   
## NA NA NA NA NA NA NA   
## (0.7,0.8] (0.8,0.9] (0.9,1]   
## NA NA NA

midbreaks

## [1] 0.05 0.15 0.25 0.35 0.45 0.55 0.65 0.75 0.85 0.95

plot(midbreaks, props, pch=16, ylim = c(0,1))  
lines(test\_trips$logged\_trip\_duration, glm\_prob, col="red")  
lines(test\_trips$logged\_trip\_duration, knn\_prob, col="green")



# Summary of analysis

The result of the KNN model is better than for the logistic regression, at least for the Accuracy metric. For logistic regression we have Accuracy of about 51% and for KNN we have accuracy of about 80%. For this dataset and scenario at least KNN is a better model than loogistic regression in terms of Accuracy.