Proposal

Introduction

This is a comprehensive Exploratory Data Analysis for the New York City Taxi Trip Duration competition with tidy R and ggplot2.

The goal of this playground challenge is to predict the *duration of taxi rides in NYC* based on features like trip coordinates or pickup date and time. The data comes in the shape of 1.5 million training observations (`../input/train.csv`) and 630k test observation (`../input/test.csv`). Each row contains one taxi trip.

In this notebook, we will first study and visualise the original data, engineer new features, and examine potential outliers. Then we add two **external data sets** on the NYC weather and on the theoretically fastest routes. We visualise and analyse the new features within these data sets and their impact on the target *trip_duration* values. Finally, we will make a brief excursion into viewing this challenge as a **classification problem** and finish this notebook with a **simple XGBoost model** that provides a basic prediction.

I hope that this notebook will help you in getting started with this challenge. There is lots of room for you to develop your own ideas for new features and visualisations. In particular the classification approach and the final model are only a basic starting point for you to improve and optimise them for better performance. As always, any feedback, questions, or constructive criticism are much appreciated.

Originally, this analysis contained a few **hidden figures**, due to the memory/run-time limitations of the Kernels environment at the time. This means that I had to disable a few auxilliary visualisations as the notebook grew towards its final stage. I tried to only remove those plots that didn't affect the flow of the analysis too much. *All* of the corresponding code was still contained in this notebook and you can easily recover these hidden plots. Now, with the new and improved Kernel specs those plots are back in the "official" version. (On related matters, the movie hidden figures is a pretty awesome piece of history for maths (and space) geeks like us ;-))

Finally, I would like to **thank everyone of you** for viewing, upvoting, or commenting on this kernel! I'm still a beginner here on Kaggle and your support means a lot to me :-) . Also, thanks to NockedDown for suggesting the new updated title pun in the comments! ;-)

Enough talk. Let's get started:

Load libraries and helper functions

```r
library('ggplot2') # visualisation
library('scales') # visualisation
library('grid') # visualisation
library('RColorBrewer') # visualisation
library('corrplot') # visualisation
library('alluvial') # visualisation
library('dplyr') # data manipulation
library('readr') # input/output
library('data.table') # data manipulation
library('tibble') # data wrangling
library('tidyr') # data wrangling
library('stringr') # string manipulation
library('forcats') # factor manipulation
library('lubridate') # date and time
library('geosphere') # geospatial locations
library('leaflet') # maps
library('leaflet.extras') # maps
library('maps') # maps
library('xgboost') # modelling
library('caret') # modelling
```

We use the *multiplot* function, courtesy of R Cookbooks to create multi-panel plots.

```r
# Define multiple plot function

#

# ggplot objects can be passed in ..., or to plotlist (as a list of ggplot objects)

# - cols:   Number of columns in layout

# - layout: A matrix specifying the layout. If present, 'cols' is ignored.

#

# If the layout is something like matrix(c(1,2,3,3), nrow=2, byrow=TRUE),

# then plot 1 will go in the upper left, 2 will go in the upper right, and

# 3 will go all the way across the bottom.

#

multiplot <- function(..., plotlist=NULL, file, cols=1, layout=NULL) {



  # Make a list from the ... arguments and plotlist

  plots <- c(list(...), plotlist)



  numPlots = length(plots)



  # If layout is NULL, then use 'cols' to determine layout

  if (is.null(layout)) {

    # Make the panel

    # ncol: Number of columns of plots

    # nrow: Number of rows needed, calculated from # of cols

    layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),
                     ncol = cols, nrow = ceiling(numPlots/cols))
  }


 if (numPlots==1) {

    print(plots[[1]])


  } else {

    # Set up the page

    grid.newpage()

    pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout))))

    # Make each plot, in the correct location

    for (i in 1:numPlots) {

      # Get the i,j matrix positions of the regions that contain this subplot

      matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))

      print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row,

                                      layout.pos.col = matchidx$col))

    }

  }

}
```