Part 1 - Exploratory Data Analysis

For this section of the takehome project, I wrangled the data using the resampling method to aggregate the data into 15 minute time intervals as instructed. However, to glean more useful information from the dense amount of data I also did hourly and daily aggregations. The below are my findings from these aggregated datasets, and all plots referenced can be found in the corresponding jupyter notebook.

- For the 15 minute aggregated time interval dataset:
    - Total count = 9788
    - Mean = 9.52 logins/15 mins
    - Standard Deviation = 8.33 logins
    - Min/Max = 0, 73 logins
    - 25%/50%/75% quartiles = 3, 7, 13 logins
- For the hourly aggregated time interval dataset:
    - Total count = 2447
    - Mean = 38.06 logins/hour
    - Standard Deviation = 30.75 logins
    - Min/Max = 0, 237 logins
    - 25%/50%/75% quartiles = 15, 29, 52 logins
- For the daily aggregated time interval dataset:
    - Total count = 9788
    - Mean = 904.29 logins/15 mins
    - Standard Deviation = 347.18 logins
    - Min/Max = 112, 1889 logins
    - 25%/50%/75% quartiles = 643, 827, 1141 logins
- Plotting the 15 minute interval aggregated data (setting dividers every 24 hours), for the most part it is far too noisy to concretely glean any information about the underlying patterns in the data. I was able to tell that there are definite peaks and valleys that occur over each day, and that there seems to be a first order difference from looking at the increasing rolling mean (just under 10 average logins/15 minutes using a windowsize of 100 to just under 20).
- Moving to the hourly aggregated data, the same graph shows a much more clear image of the login patterns. The data shows that there is on average 3.5 peaks throughout the day where the login counts have a local maxima, and the sinusoidal rolling mean shows a frequency of about 3 days.
- Finally, looking at the daily aggregated data plot, weekly trends indicate about 2 peaks a week (which match to the 3 day sinusoidal pattern mentioned above) with the daily average trending from 500 average logins/day with a window size of 5 days to 1000 average logins/day.

Part 2 - Experiment and Metrics Design

1. The key measure of success for this experiment that I choose is the rate of change in the **percentage rate of trips taken in X city** where X can be chosen as either Gotham or Metropolis (assuming only these two cities are the only cities available). I chose this because the goal is to measure if driver partners that tend to be exclusive to one city will expand their pickup area if the toll fee was not a factor. By determining the number of drivers that do change from being in the top/bottom to-be-specified percentile to having a percentage rate closer to 50%, I can determine the efficiency of the experiment. For the sake of this experiment, the null hypothesis will be **There is no significant change in the population of drivers that expanded their availability after the proposed toll reimbursement change**.

2. The implementation of the experiment is rather simple; record a month's (or more if possible) worth of data from the drivers before the toll reimbursing proposal goes into effect and record the same amount of data after.
    a. Features that need to be recorded for each trip to test my hypothesis are:
        i. City the pickup was located in (Gotham or Metropolis)
        ii. The time of the pickup (preferably with the time of the day and the day of the week in separate columns)
        iii. Driver ID
    b.
        i. Start by splitting up the datasets into weekday and weekend sets (using the value found in the day of the week). This is to account for the difference in activity levels of the cities to ensure we're comparing like distributions against each other.
        ii. For the weekend data:
            1. Since I know that activity is reasonably equal between both cities, there is no need to split up the data into day/night cycles.
            2. Calculate the percentage of pickups for each unique driver from either city (assuming adding both would give 100%) by doing a count of the number of rides from that city over the total number of rides taken.
            3. Filter out drivers that already have availability to both cities before the toll-reimbursement goes into effect. I will arbitrarily designate those who have 70% or more trips in one city as the cutoff.
            4. Do a one tailed z-test (similar to AB testing but only testing for an improvement by "B", not just "different from "A") to determine if a significant change has occurred. If it has, then the experiment was a success.
        iii. For the weekday data:
            1. Much like how the data was split into weekday/weekend data due to a change in activity in the cities (thus a change in the demand), the data needs to be split into day/night cycles. This can be done by marking trips done between the hours of 5 pm to 6 am as "Night Trips" and 6 am to 5 pm as "Day Trips"
            2. Follow the same method as the weekend data, except when conducting our z-test have the critical p value be larger to account for the imbalance in demand relative to the demand over the weekends.
    c. To interpret these results, I would start by discussing whether or not my z-tests showed any significant differences. If they did, I would recommend they keep the toll-reimbursement **if** the increase in profit from having higher availability covers the flat fee of reimbursing the drivers for the tolls based on **weekly** comparisons. If no significant differences are found **and** no weekly profit increase is reported, then on all accounts the proposed change will cause the company to lose money and should not be permanently implemented. Some caveats of this experiment are:
        i. As I am only measuring the change in percentage number of rides done exclusively to one city, the above method does not take into account the increase in availability from removing the cost barrier of driver partners living in one city who wish to work in the other but cannot afford the toll.
        ii. As the daily demand is equivalent across both cities during the weekend (and thus more incentive for the drivers to be available in both cities), I would

recommend relying on the results of that set of data compared to the results from the weekdays' data.

Part 3 - Predictive Modeling

For the EDA section of this problem, I started by taking a look at the imported data. I see 3 columns with missing values: Driver Rating, Passenger Rating and Phone Type. I also see that the Last Trip Date and Signup Date are set as objects, not datetime. I then used the describe method to quickly take a look at the statistics of the numeric columns. For visualizations, I plotted the number of people vs the date they signed up, the date their last trip was and how many trips they took in their first 30 days. Setting up new column in the dataframe with information on whether or not the user was retained, I calculated a value of 10599 users retained, about 21.2%. I also plotted a hierarchical cluster map of the data as well as a heatmap to indicate correlation between features.

Although the details can be found in the corresponding jupyter notebook, I will outline here the process I used to initially wrangle the data into a ML friendly format.

- I filled in the missing values of the driver and passenger ratings with the mean values of the respective features.
- I replaced the missing values in the Phone Type column with a 3rd unspecified type.
- I converted the boolean column of whether or not a user was an ultimate black user to numeric with 1 = True, 0 = False.
- I mapped the city the user signed up in to a numeric range of 1 to 3.
- I mapped the phone types to a numeric range of 1 to 3.
- I dropped the datetime columns of "Last Trip Date" and "Signup Date"
- For the model itself, I chose a Random Forest regressor. I chose this because not only am I most familiar with this method, it is an ensemble model that is robust against overfitting and extracting feature importance is easy.
- I start by splitting the dataset into a test, training and validation set.
- I then tune the hyperparameters: n_estimator, depth, and max_features. For each I iterated over a list of values and plotted them against the MSE using the validation set and choosing the elbow point for the lowest MSE.
- I then recombine the validation and test set and fit the training dataset to a Random Forest regressor with the tuned hyperparameters. This evaluates to an R^2 value of 0.105 and MSE of 0.148.
- Finally, I extract out feature importance by accessing the model's properties and plot them using a bar plot.

Based off of the results, the top 4 more important features are average distance of rides in the first 30 days, percentage of rides during the weekday, percentage of rides taken with a surge multiplier of more than 1 and the number of trips taken in the first 30 days. These are my suggestions to improve rider retention:

- Provide promotional deals that give discounts for riders who use Ultimate primarily for weekday rides.
- Provide an initial deal for the first month using Ultimate to discount rides at a certain %.
- Reduce the surge pricing algorithm to lower the threshold for surges and the amount surges increase the cost of the ride by.
- Provide a permanent deal where users who in the first 30 days of the program whose rides are less than 10 miles get a discount at a certain %.