Team: You Come Up with Something (Eric Wang, Alex Levine, and Xueru Xie)

Section 1: Data Cleaning/Preprocessing

Our first impression of the dataset, based on the prompt, was that we might want to create a loss function that minimized the time of each route. This would allow us to see which bart station is most ideal to get off at to minimize time spent travelling. Thus, we began by trying to plot a map of the San Francisco area with the calculated distances that each route travels. This involved attempting to acquire a Google Maps API that we would use to map the routes directly.

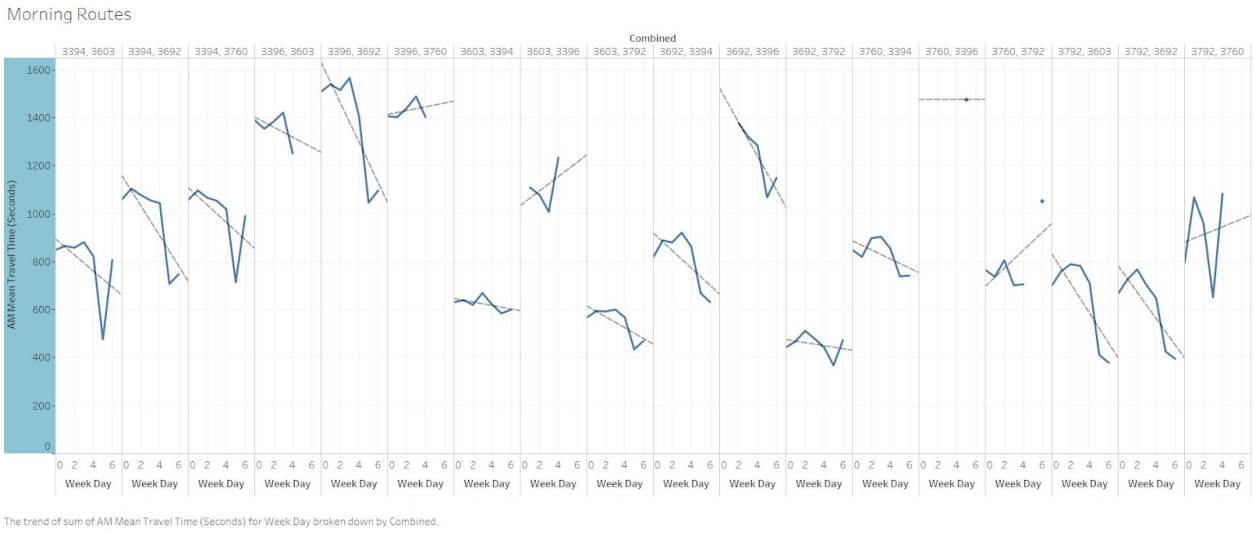
However, closer inspection of the data revealed that the distances wouldn't be the primary focus of the majority of the dataset, so we shifted gears to relating the routes individually with the average times spent travelling. We were most interested to see how the data would fit in a time series, revealing weekly trends, because identifying this kind of data enables potentially business-practice altering analysis. In order to get a more accurate read on the average travel times, we took advantage of the dataset's separation of average travel times by time of day (morning, midday, and evening). In order to do so, we specifically split the dataset by these three parameters and plotted average travel time trends for every single route based on the day of the week using Tableau.

The process of plotting the mean travel times for each of the 18 routes for each day of the week was extremely difficult. Our first few scans of the data revealed an alarmingly large quantity of "NaN" values that appeared for rides that only ran during certain times of the day (the other time(s) of the day were filled with "NaN" values). We were quite conflicted on how to approach these, as some mentors suggested replacing these null values with the average value of the time that the Uber ride actually ran during other times of the day, but we worried that this would add noise, hiding factors that affect travel times depending on the time of day.

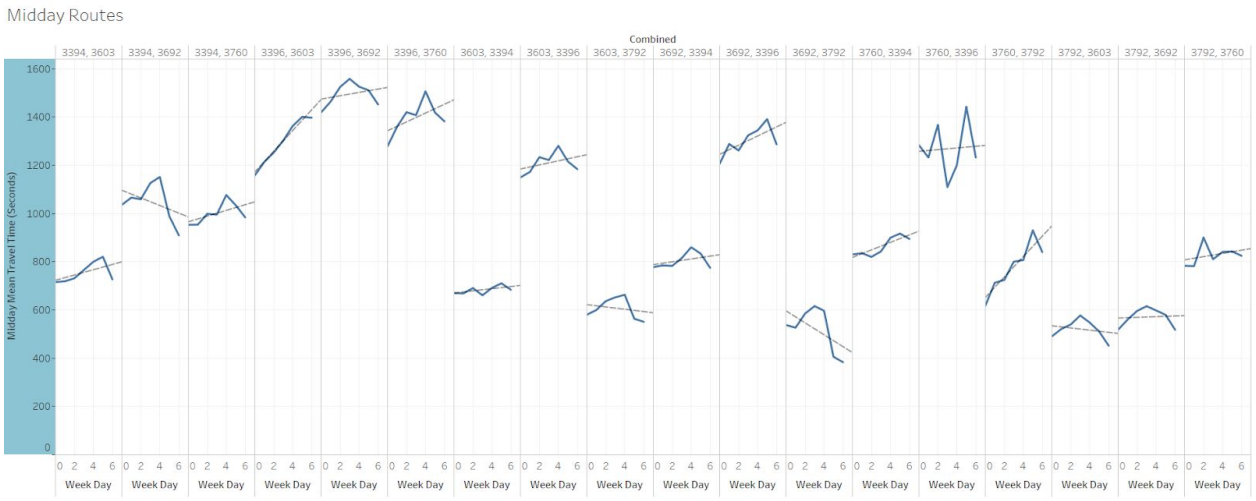
Essentially, we felt the columns weren't correlated in a way that would allow us to simply assume a static correlation across different times of the day. This line of thinking later led us to simply separate our analysis for each time of day, thus allowing us to drop rows containing "NaN" values when necessary (so we decided to have three different general plots, morning, midday, and evening). This made sense because we are only accounting for travel data recorded at certain times of the day, which lessens the chance for human error in our analysis.

Section 2: Data Visualizations

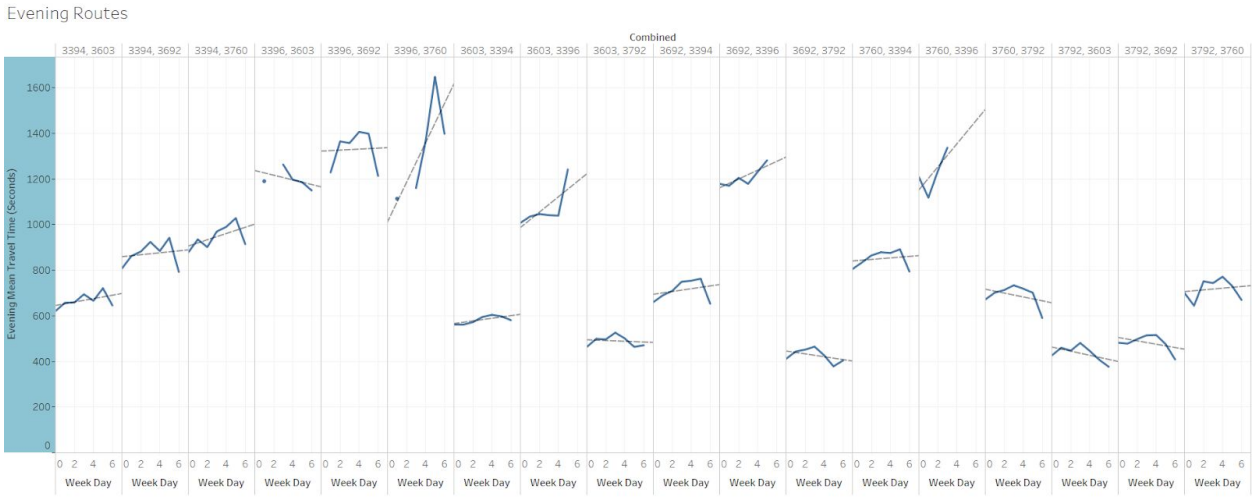
Weekly Time Series Data of Each Route for Morning Rides



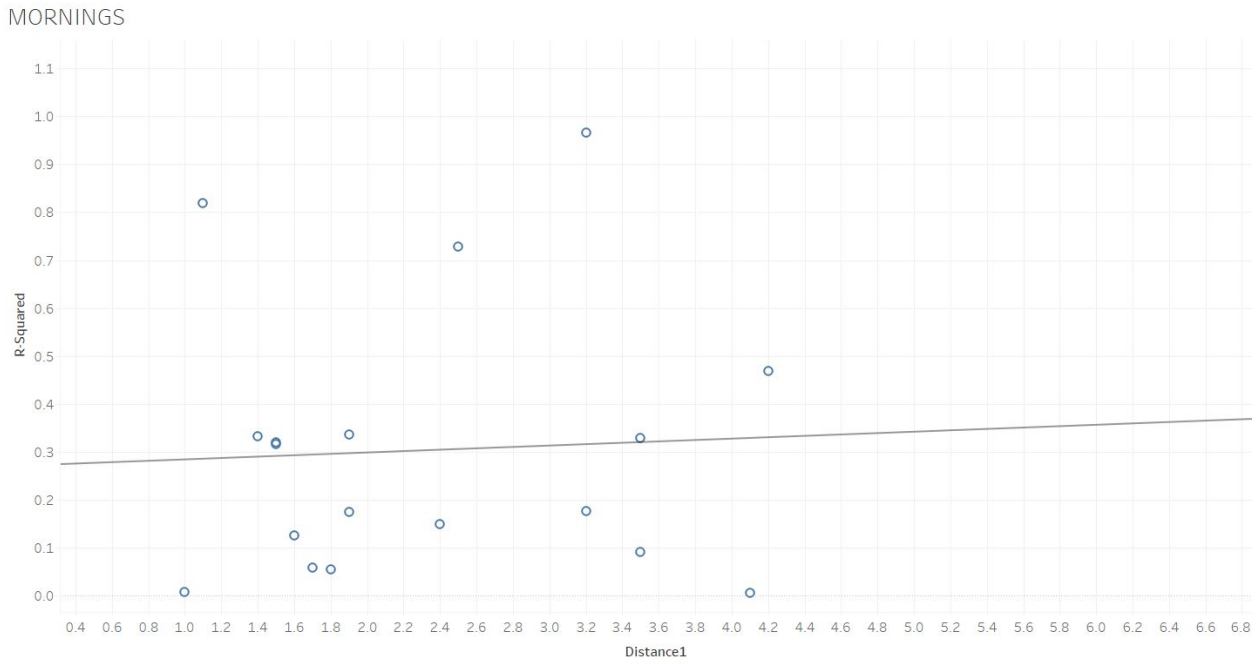
Weekly Time Series Data of Each Route for Midday Rides



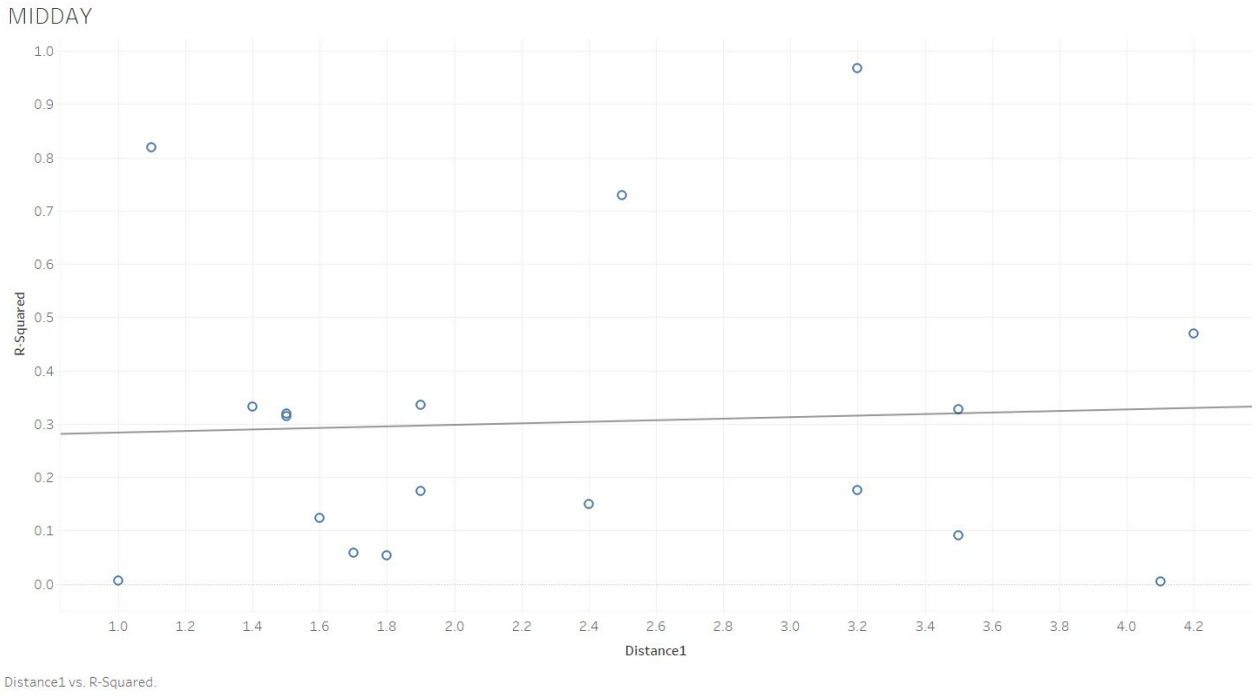
Weekly Time Series Data of Each Route for Evening Rides



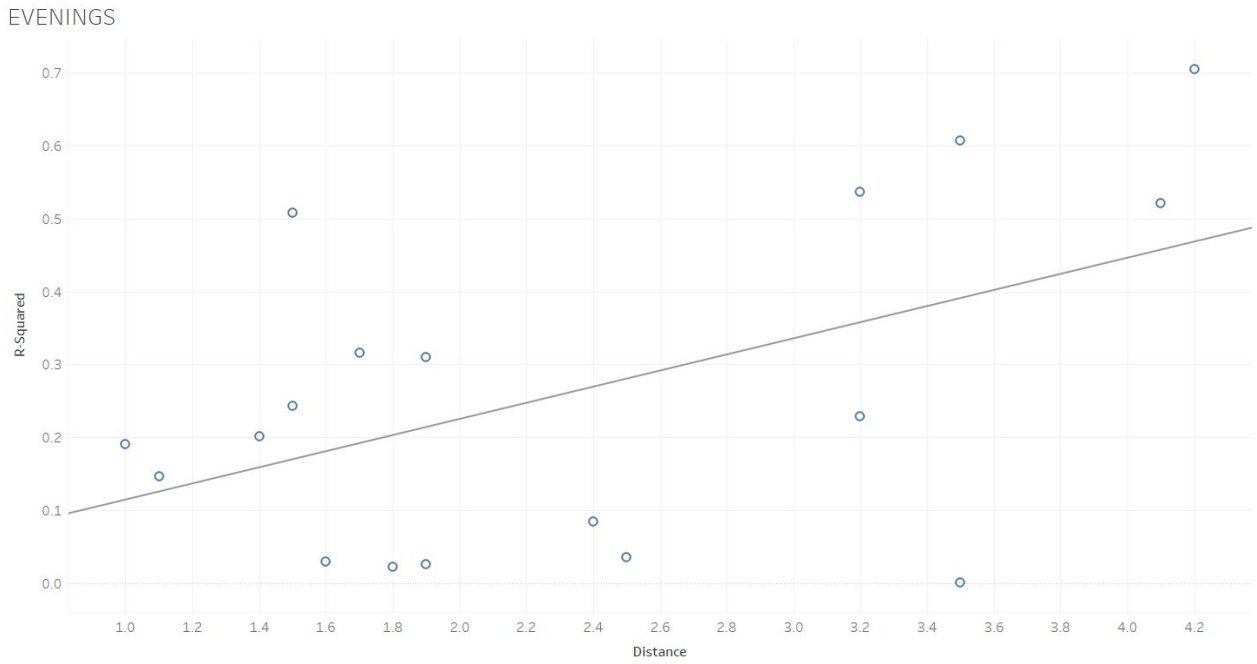
Linear Correlation Analysis of Morning Rides for Each Day of the Week



Linear Correlation Analysis of Midday Rides for Each Day of the Week



Linear Correlation Analysis of Evening Rides for Each Day of the Week



Distance vs. R-Squared.

(more visualizations down below for the analysis questions)

Section 3: Forecasting/Machine Learning

So ideally, each route would be represented by its own line and all would be juxtaposed on one plot. However, accomplishing this proved to be an arduous task. We initially created a multi-index dataframe with 126 entries and three indexes: the day of the week, the origin ID, and the destination ID. While we were able to represent our data in an orderly manner, this proved to be ineffective when we attempted to plot the data using pyplot because we consistently got ambiguous error messages that we suspect was due to the fact that our DataFrame had multiple indexes. One of the mentors hypothesized that the error was due to the fact that nine data points were being plotted for a single “x” value. It appeared that there wasn’t an easy way to plot all 18 lines alongside each other using Python. After hours of attempting to find a workaround, we finally resorted to learning Tableau which enabled us to create effective visualizations after a few minutes of playing around with the variables.

After this crucial step, we decided to tackle the next steps by determining potential predictors, which actually brought us right back to using the Google Maps Distance Matrix API to calculate distance travelled for each route. While we saw a strong correlation between distance and average travel time, we wanted to see if we could create a linear predictor based on the day of the week. However, the weekly time travelled trends were slightly too erratic to utilize a linear predictor for. We figured this out by solving for the r-squared values of each route's provided data and setting that against the route's distances. This seemed like the optimal method to determine usability as a predictor because we would be able to see how linear the weekly average travel time is.

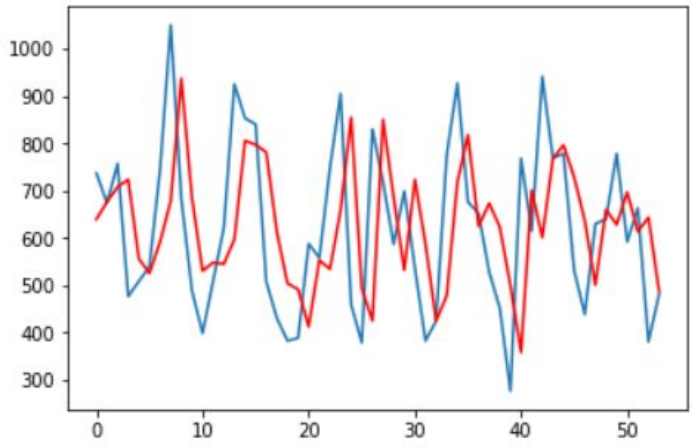
We initially also noticed that for some of the routes, the day of the week is correlated with the mean travel time. To further investigate this observation, we utilized Tableau to create a regression line for each route and calculated the R-squared and p-values associated with each line. Surprisingly, we found extremely low R-squared values, suggesting that there was weak to no correlation between the distances and average travel times.

The process of finding a linear predictor was relatively tedious, because we decided to calculate the r-squared values of each route based on the day of the week for each time of day. Morning rides had an average r-squared value of 0.003 and a p-value of 0.835, midday rides had an average r-squared value of 0.003 and a p-value of 0.835, and evening rides had the largest r-squared value of 0.255 and the lowest p-value of 0.033. This means that there is a 3% chance that the evening correlation occurs due to chance, and it can most likely be attributed to less traffic flow. Traffic flow is a seasonal variable that increases average travel times by a varying amount primarily during the morning and midday. Because it is non-stationary, it increases variance as well, which decreases linear correlation for the morning and midday average travel time distributions.

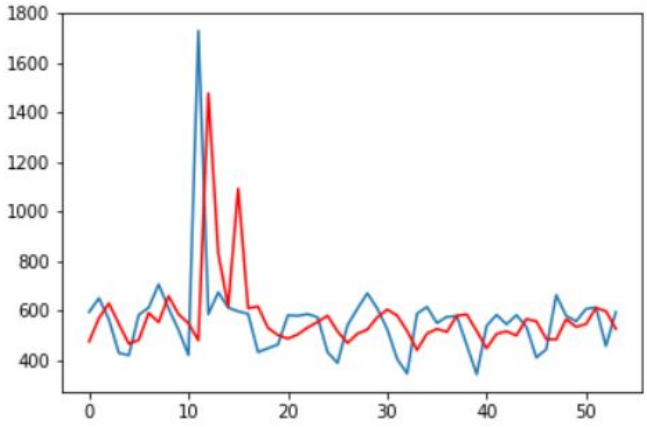
The next step was to implement a machine learning algorithm with the goal of predicting future travel times. In particular, we used the ARIMA model. The preliminary step we decided to take was to calculate the ratio of NaN values for each route for each time of day. In order for the machine learning algorithm to be relatively accurate, we needed to have the lowest ratio of NaN values possible, thus we chose the three routes for each time of day that had the lowest Nan value ratios. We picked the data for these routes for Time Series Forecasting using Long Short Term Memory Networks, which is a predictive model designed to learn from a set of training data. We then split the data into training and testing data using a ratio of 70:30 and ran it through the model. In order to implement the ARIMA, we relied on online tutorials.

To train the predictive model, we decided to take averages of the weekly distributions that we currently had, because we didn't really have any other data to spare on training. Essentially, we used the average of each week's distribution's average times spent travelling to train our ARIMA model who's predictions had a margin of error around 50%. While the predictive model we created was quite inaccurate in terms of predicting the absolute travel times, it was able to predict within the general shape of the correct values. It is especially notable that it was able to predict the peaks in the time series data.

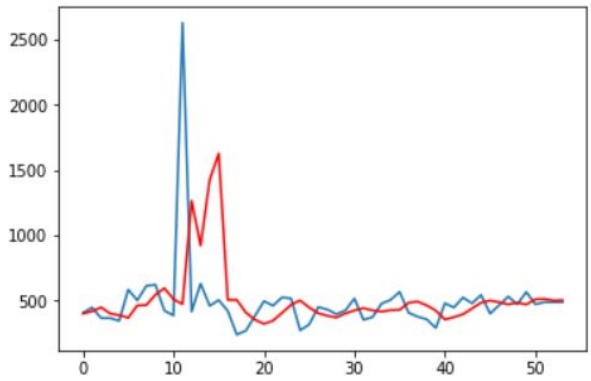
(Oracle Park to 2nd Street and Stevenson Street in Morning w/ Red Prediction)



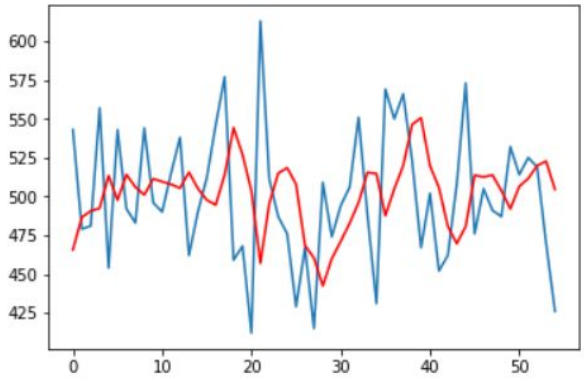
(Embarcadero to Oracle Park in Morning w/ Red Prediction)



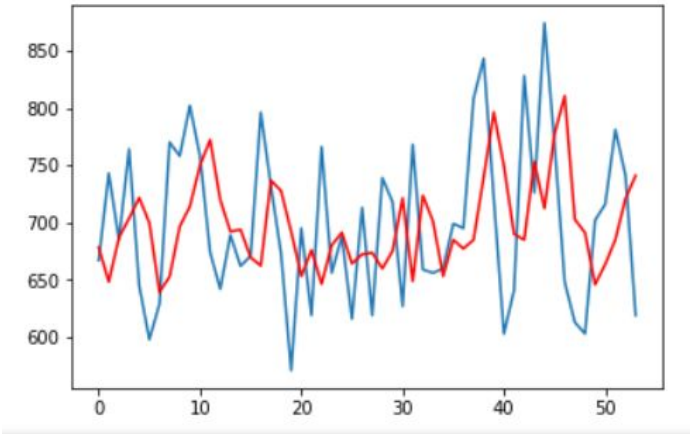
(2nd Street and Stevenson Street to Oracle Park in Morning w/ Red Prediction)



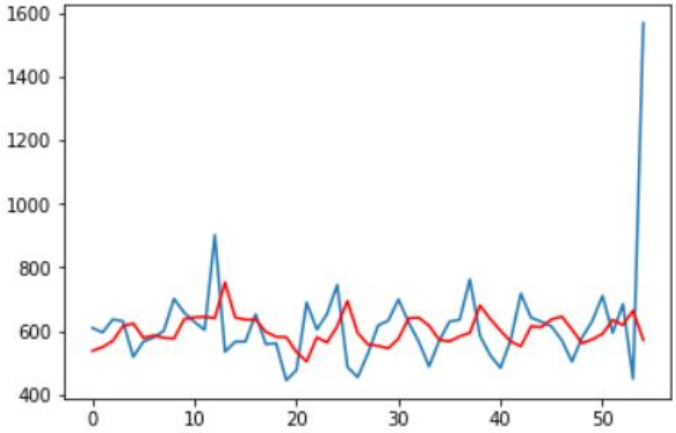
(Embarcadero to Oracle Park in Evening w/ Red Prediction)



(2nd Street and Stevenson Street to Fisherman's Wharf in Evening w/ Red Prediction)



(Embarcadero to Oracle Park in Midday w/ Red Prediction)



(Embarcadero to Oracle Park in Midday Prediction vs. Actual)

(these are only 12/55 tested values for space's sake)

1 Actual=609.000000, Predicted=537.023012

2 Actual=595.000000, Predicted=548.724964

3 Actual=636.000000, Predicted=568.211737

4 Actual=630.000000, Predicted=614.687239

5 Actual=518.000000, Predicted=623.506398

6 Actual=566.000000, Predicted=577.841465

7 Actual=579.000000, Predicted=585.170664

8 Actual=600.000000, Predicted=577.845134

9 Actual=701.000000, Predicted=575.534435

10 Actual=657.000000, Predicted=636.383683

11 Actual=628.000000, Predicted=642.416113

12 Actual=603.000000, Predicted=644.198596

The performance of our ARIMA predictive model is well within reason considering we handpicked routes that had the most usable non-NaN values. However, because we distributed the dataset weekly and by time of day, the number of data values we had was relatively low, which definitely reduces the accuracy of our predictive model, but at least we didn't have to worry about overfitting!

Overall, I think the predictions could be improved in two major ways. First, we need a larger training dataset for the model to learn from. Pretty self-explanatory. Second, I think we could add more parameters for the predictive model to learn from. For example, if we had more time I think it would greatly improve accuracy if we aggregated the data differently and had the predictor account for a multi-dimensional approach rather than just the single data aggregation that we decided to use. Of course there are many more improvements we could make to the model, and I'm still curious as to how accurate a predictive model could actually get based on just this dataset. I hypothesize that random sampling of some kind can be used to increase the number of data values we have, but we didn't have time to attempt this.

Section 4: Business Analysis and Proposals

According to the time series visualizations and forecasts, rides in the morning generally take less time, which I would assume increases revenue that Uber earns, as the drivers are able to fit more rides in a smaller amount of time. Thus, I would propose that Uber incentivizes tourists to take more rides in the mornings. Based on additional visualizations (found down in part 2 of the analysis questions), it seems that Uber driver strikes don't have the greatest effect on average travel times. As counter-intuitive as it seems (and we need annual data to confirm), there was another unknown factor that had a greater impact than the Uber drivers' strike on March 25th. Thus, solely based on the data and not intuition, there is a possibility that profit could be increased further by cutting more pay to the Uber drivers. In addition, based on the predictive models Uber can actually predict spikes in average travel time to an extent. Even when causes are not necessarily known, the model can make adjustments that shouldn't be dismissed easily. We believe a predictive model would help Uber avoid large spikes in average travel times and thus improve their customer satisfaction as well as profit.

Section 5: Conclusion

Overall, it would be extremely interesting to continue analysis on factors affecting Uber's business model using the travel time data. The scope of our analysis was greatly limited, and we definitely didn't use as much of the dataset as I would have liked. However, from the parts of the dataset that we did use to construct our predictive model, it is clear that Uber can implement promotions or alter their business model to improve profit.

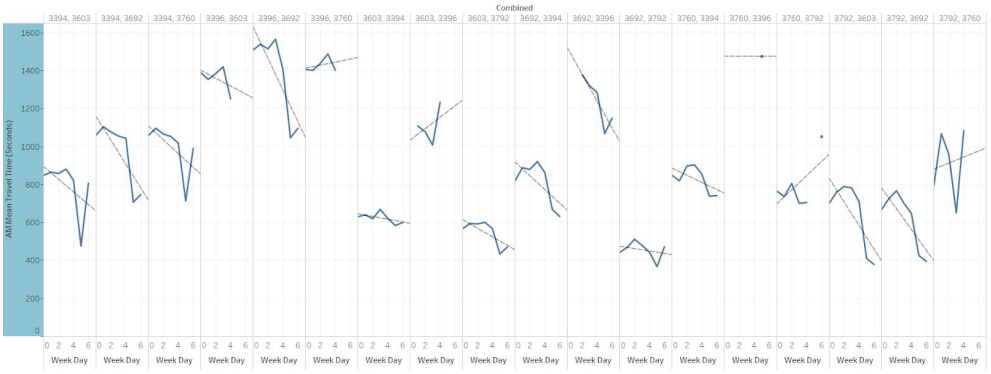
Analysis questions:

1. We first cross-referenced the different csv files to determine the BART station ID's and hotspot ID's, which we then used to filter the dataset so we only have rides with Origin ID's matching hotspot ID's and Destination ID's matching BART ID's. Next, we utilized the PANDAS dataframe built-in groupby() function to get the minimum average travel time for every one of the 9 unique routes. We then picked the three BART stations that minimized average travel times.
    a. Oracle Park, Fisherman's Wharf, and The Palace of Fine Arts all have minimal travel time when getting off at Embarcadero on average.
    b. For AM:
        i. Fisherman's Wharf has minimal travel time getting off at 2nd Street and Stevenson Street.
        ii. The Palace of Fine Arts has minimal travel time getting off at Embarcadero.
        iii. Oracle Park has minimal travel time getting off at 2nd Street and Stevenson Street.
    c. For PM:
        i. Fisherman's Wharf has minimal travel time getting off at Embarcadero.
        ii. The Palace of Fine Arts has minimal travel time getting off at the Powell BART station.
        iii. Oracle Park has minimal travel time getting off at Embarcadero.

Origin Display Name	Destination Display Name	AM Mean Travel Time (Seconds)	PM Mean Travel Time (Seconds)	Midday Mean Travel Time (Seconds)
Fisherman's Wharf, San Francisco, CA	Embarcadero, San Francisco, CA	904.0	1081.0	732.0
Fisherman's Wharf, San Francisco, CA	2nd Street and Stevenson Street (Montgomery BA...	752.0	1096.0	974.0
Fisherman's Wharf, San Francisco, CA	Powell BART Station, Market St and Powell St, ...	998.0	1120.0	848.0
The Palace Of Fine Arts, 3601 Lyon St, San Fra...	Embarcadero, San Francisco, CA	1327.0	1527.0	1370.0
The Palace Of Fine Arts, 3601 Lyon St, San Fra...	2nd Street and Stevenson Street (Montgomery BA...	1506.0	1595.0	1370.0

2. As shown above in the process section:

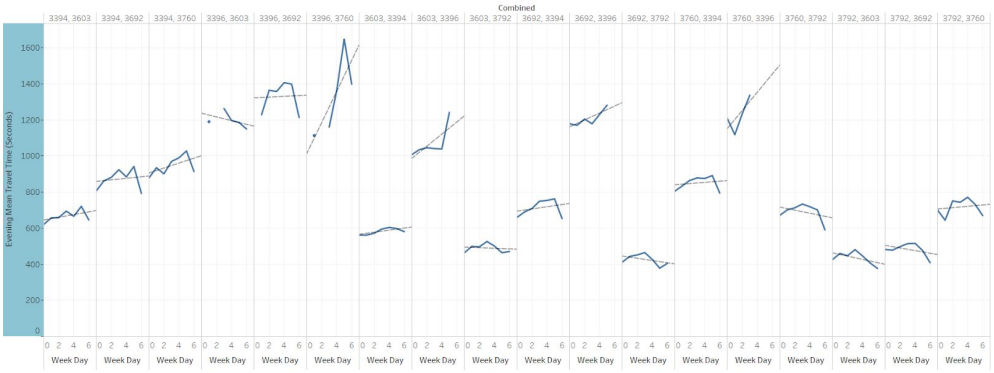
Morning Routes



The trend of sum of AM Mean Travel Time (Seconds) for Week Day broken down by Combined.

Midday Routes

Evening Routes



The trend of sum of Evening Mean Travel Time (Seconds) for Week Day broken down by Combined.

3. Recommendations:

Origin Display Name	Destination Display Name	AM Mean Travel Time (Seconds)	PM Mean Travel Time (Seconds)	Midday Mean Travel Time (Seconds)
Embarcadero, San Francisco, CA	Fisherman's Wharf, San Francisco, CA	601.0	664.0	654.0
Embarcadero, San Francisco, CA	The Palace Of Fine Arts, 3601 Lyon St, San Fra...	845.0	1387.0	1063.0
Embarcadero, San Francisco, CA	Oracle Park, 24 Willie Mays Plaza, San Francis...	650.0	753.0	614.0
2nd Street and Stevenson Street (Montgomery BA...	Fisherman's Wharf, San Francisco, CA	869.0	968.0	746.0
2nd Street and Stevenson Street (Montgomery BA...	The Palace Of Fine Arts, 3601 Lyon St, San Fra...	1255.0	1284.0	1053.0
2nd Street and Stevenson Street (Montgomery BA...	Oracle Park, 24 Willie Mays Plaza, San Francis...	364.0	496.0	558.0
Powell BART Station, Market St and Powell St, ...	Fisherman's Wharf, San Francisco, CA	781.0	927.0	816.0
Powell BART Station, Market St and Powell St, ...	The Palace Of Fine Arts, 3601 Lyon St, San Fra...	1476.0	1556.0	1192.0
Powell BART Station, Market St and Powell St, ...	Oracle Park, 24 Willie Mays Plaza, San Francis...	528.0	1091.0	588.0

a. Based on this chart (for tourists travelling from BARTs to hotspots) that is grouped by Origin Display Name and Destination Display Name, we can see that travelling in the mornings consistently yields the fastest average travel times.

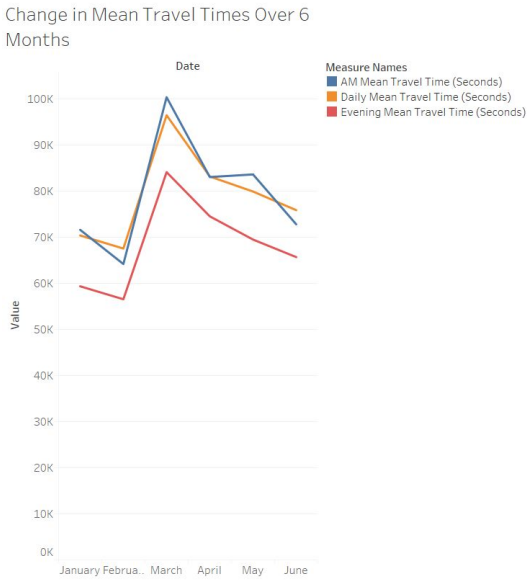
Origin Display Name	Destination Display Name	AM Mean Travel Time (Seconds)	PM Mean Travel Time (Seconds)	Midday Mean Travel Time (Seconds)
Fisherman's Wharf, San Francisco, CA	Embarcadero, San Francisco, CA	904.0	1081.0	732.0
Fisherman's Wharf, San Francisco, CA	2nd Street and Stevenson Street (Montgomery BA...	752.0	1096.0	974.0
Fisherman's Wharf, San Francisco, CA	Powell BART Station, Market St and Powell St, ...	998.0	1120.0	848.0
The Palace Of Fine Arts, 3601 Lyon St, San Fra...	Embarcadero, San Francisco, CA	1327.0	1527.0	1370.0
The Palace Of Fine Arts, 3601 Lyon St, San Fra...	2nd Street and Stevenson Street (Montgomery BA...	1506.0	1595.0	1370.0
The Palace Of Fine Arts, 3601 Lyon St, San Fra...	Powell BART Station, Market St and Powell St, ...	1582.0	1370.0	1156.0
Oracle Park, 24 Willie Mays Plaza, San Francis...	Embarcadero, San Francisco, CA	648.0	529.0	528.0
Oracle Park, 24 Willie Mays Plaza, San Francis...	2nd Street and Stevenson Street (Montgomery BA...	588.0	662.0	507.0
Oracle Park, 24 Willie Mays Plaza, San Francis...	Powell BART Station, Market St and Powell St, ...	652.0	1180.0	1036.0

b. Based on this chart (for tourists travelling from hotspots to BARTs) grouped by Origin Display Name and Destination Display Name, we can see that travelling

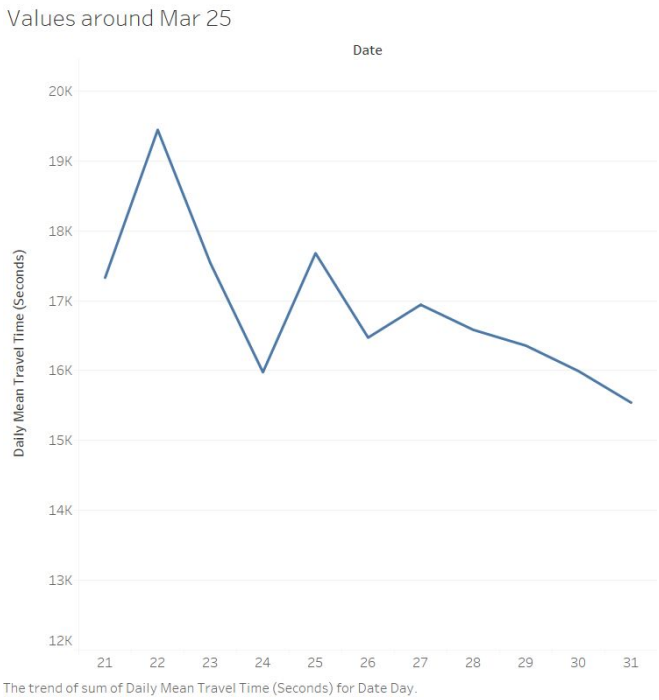
during midday typically yields the fastest average travel times, but occasionally travelling during the morning may yield the fastest average travel times depending on which hotspot to BART station route you take.

Part 2 Analysis Questions:

It definitely seems that there was a spike in average travel times around the month of March and April before going back down to around the starting average travel times. Based on the data, this spike is possibly an irregularity because there were a couple events happening around March including the Mardi Gras, but it could also be seasonal because of the annual St. Patrick's Parade. However, I would go out on a limb and hypothesize that this is a cyclic circumstance caused by an Uber pay cut which caused Uber drivers to go on strike, thus drastically increasing average travel times. (Source: https://www.forbes.com/sites/janetwburns/2019/03/25/uber-and-lyft-drivers-strike-in-la-after-yet-another-pay-cut/#6ff4c96e26e4)



I would like to test my hypothesize by cross referencing more exact values. The exact date referenced in the Forbes article is March 25, so I filtered the DataFrame to just the days from March 20th to March 31st and created a graph visualization using Tableau. The visualization rejects my hypothesis because while there is a spike in average travel times on March 25th, there is an even larger spike on March 22nd. I'm uncertain as to what caused this, but it definitely suggests that the March 25th Uber driver strike was not the biggest cause of higher average travel times.



ARIMA Tutorial:

https://towardsdatascience.com/forecasting-exchange-rates-using-arima-in-python-f032f313fc56