Mini Project 3: Predicting Ridership and Hottest Pickup Zones in January Based on Weather Data

ENGR-UH 4560 Machine Learning, Fall 2019

Barkin Simsek, *bs3528@nyu.edu*; Nishant Aswani, *nsa325@nyu.edu*

Abstract

With the rapid development of modern cities, everyday more and more people use taxicabs. Coordinating taxicabs in the city and meeting the demand of the passengers is real world a problem that needs to be addressed in the New York City. Due to its complex structure, it is a problem that is suitable for using Machine Learning. In this project we aimed to create a model for the yellow taxicabs in New York City that the taxi service providers can predict the number of pickups in a location in New York City based on the weather conditions, pick-up location and the day of the week. After the experimentation process, it is found that XGBoost algorithm performed the best with a mean absolute percentage error of 79.92% and Random Forest performed worst with a mean absolute percentage error of 82.58%.

I. Introduction

A. Research Motivation

A taxicab (or taxi as commonly known) is a private type of vehicle, where passengers decide on the pick-up and drop-off locations, as opposed to the public transportation vehicles. As a result, meeting the demand of the passengers by carefully distributing taxi vehicles around a crowded city like New York is an important issue. If there is an high demand in a certain part of the city, taxi service providers need to be able provide more vehicles to meet the demand. This would both increase the utilization of the available taxi resources and decrease the wait times of the passengers in return.

These being said, there are many different variables that play a role in people's decision to take a taxi ride instead of walking in the streets or using the subway. Some of these variables can be the price of the transportation vehicle, time of the day, day of the week, proximity of the stations or weather conditions. Therefore, this is a great problem to utilize Machine Learning techniques to figure out the complex relationships between these variables and the human nature. By modelling this complex relationship, taxi service providers can start making predictions for the future and provide a better service for their customers while making more profit. Thus, we want to solve the real world problem about strategically distributing taxicabs around the New York City by predicting number of pick-ups based on the *location* and *weather conditions*. By this way, taxi drivers can focus more on the locations with higher pick-up counts to decrease passenger waiting times while utilizing most of the taxi resource available and making more profit.

B. Research Objective

In this project, the New York City taxi data provided by New York City government was used in conjunction with the historical daily New York City weather forecast provided by Weather Underground company. As mentioned before, it is important for a city to cleverly distribute the taxi services and this projects intends to ease the taxi distribution. The project aims to create a Machine Learning model

that takes information regarding day of the week, weather conditions, and the location for the pickup, while predicting the pickup frequency for the given location, as an output. As a result, both taxi service providers can locate the crowded spots in the city while+++ passengers can locate the nearest least crowded neighborhood in the city. We aim to maximize the taxicab resourced utilized while minimizing the passenger wait times.

C. Scoping the Project

Having plotted the pickup location heatmap, we decided to limit our exploration to Manhattan. Figure 1, plotted with a random subset of data, shows Manhattan to have the hottest pickup locations. Although not visible below, another popular pickup area is the airports. However, given our focus on weather's effects, we deemed it was appropriate to leave airports out, as taxi pickups are most likely independent of weather conditions.

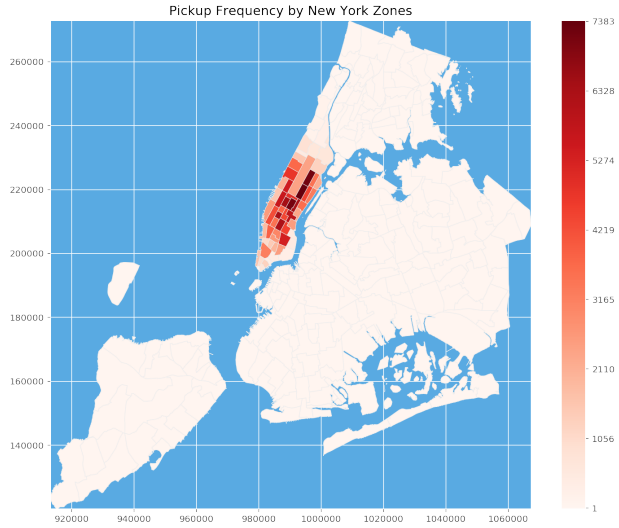
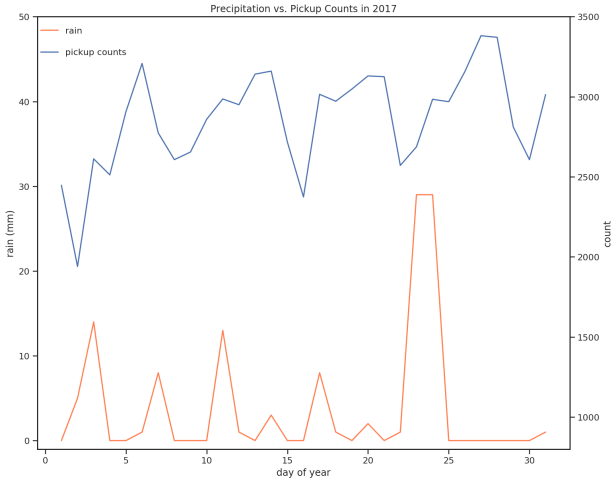
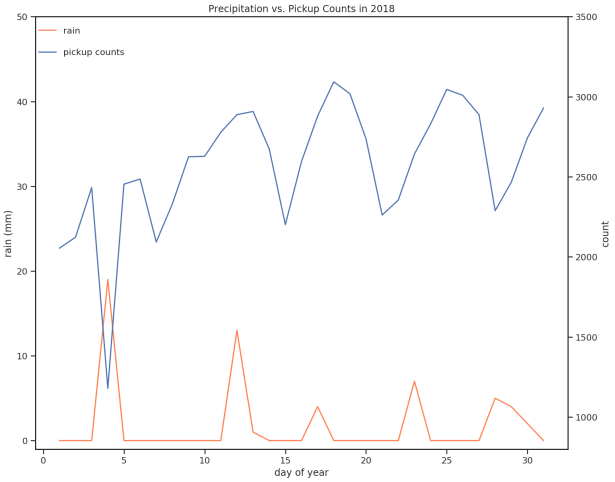


Fig. 1: Pickup frequency by zones

We chose to pursue this idea after visualizing the timeseries pattern of precipitation against the pickup counts. Figures below show promising correlations between ridership and precipitation on days of extreme weather conditions.





II. Data Procurement, Processing, and Filtering

Data was loaded into the workspace using the dask dataframe allowing for quicker access. As part of the `read_csv` function, the `tpep_pickup_datetime` and `tpep_dropoff_datetime` columns were parsed as datetime formatted to allow for easier indexing and future data cleaning. Upon reviewing the dataframe, there were $1.846966e+07$ data points for 2017 and 2018 January. The challenge was then to identify methods for removing worthless datapoints and selecting a subset for training and testing. We decided against using the entire dataset because of limited computational resources.

A. Checking for Anomalies in TLC Data

Once it was confirmed that there were no null values present in the 2017 and 2018 January dataset, we selected a 80% fraction of the dataset to build our code with. We then sorted and reindexed the data to obtain a dataframe of purely January datapoints, removing stray datapoints from different months or years.

To aid in finding unreasonable data, we turned to the Adopted Rules of the Taxi and Limousine Comission (TLC). The maximum trip duration is 12 hours. Hence, we iterated through all columns to determine the trip duration by subtracting pickup time from dropoff time. Any duration greater than 12 hours (or in negative time) was deemed to be garbage and was removed from the dataset due to the regulation that was set by New York City government. In each iteration we also calculated the speed and removed all results with values less than 0 or greater than 90 miles per hours

For the remaining columns, such as `total_amount` and `extra`, all rows with values less than zero were removed.

B. Shapefile Data

Having little experience with shapefiles, we relied on functions from an existing machine learning project to manipualate the shapefile and produce maps with a pick up heatmap.

C. Weather Data

Despite being relatively simple data, we ran into some trouble obtaining weather data for New York. As we had decided to focus on Manhattan, we found it fit to select a weather station there and select data between 2017 and 2018.

The data was obtained from the Weather Underground company website using their custom history viewing tool. Later, the data from the website was copied into a spreadsheet for preparing the data set. After cleaning the the copied data, an CSV version of the data set was created and imported.

Examining our dataset, we found null values for some days. Instead of removing them, we made the decision to linearly interpolate the data so as not to render corresponding values in our taxi dataframe unusable. Additionally, some of the values in the precipitation column were also ”T”, instead of numerical values. We deleted the rows with invalid precipitation data but it was not enough to use the weather dataset with the project. Pandas was reading the index column as an object rather than an float value. Objects can be anything and float values are needed to to work with Machine Learning algorithms. Finally, casting the precipitation column to string and later to float gave us the results we needed to run

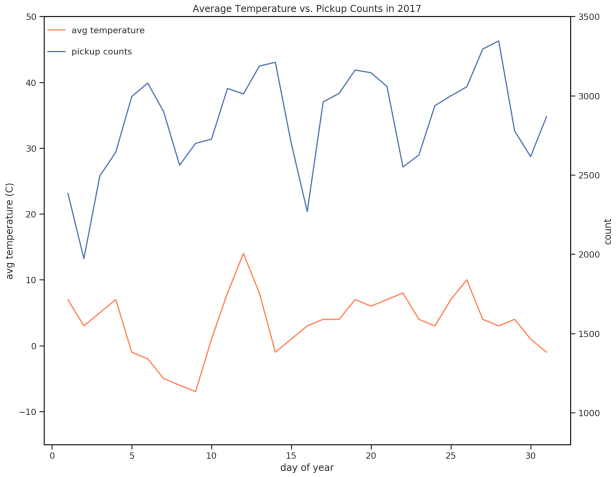
the algorithms.

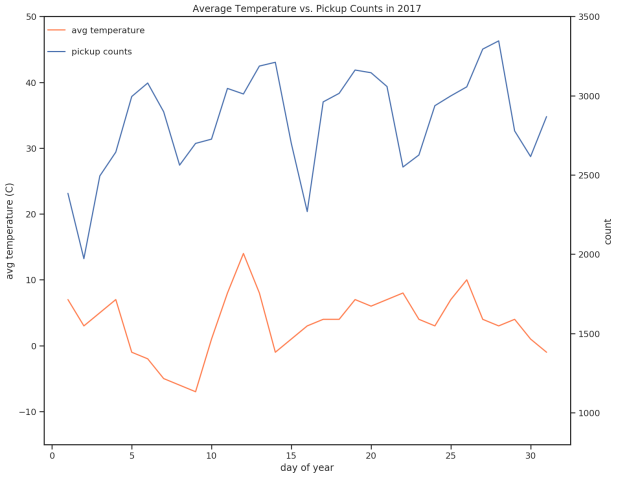
Next, we combined the date, month, and year columns to produce a datetime column. In order to correspond the weather data with the TLC data, we used the `pd.to_datetime` method to obtain an indexable column. Using this column, we then merged the relevant rain, average temperature, and average humidity data into the main dataframe.

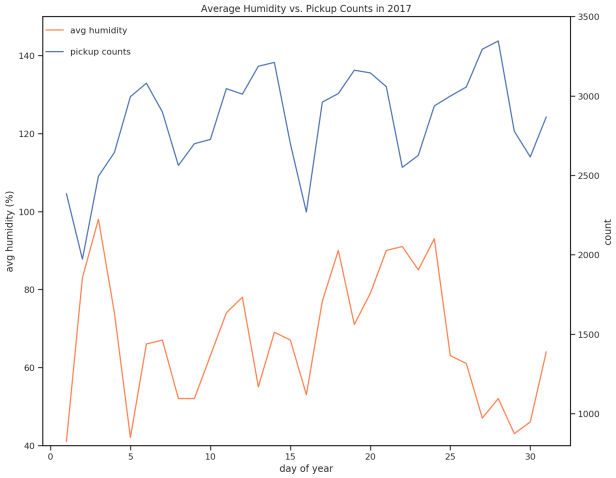
III. Methodology

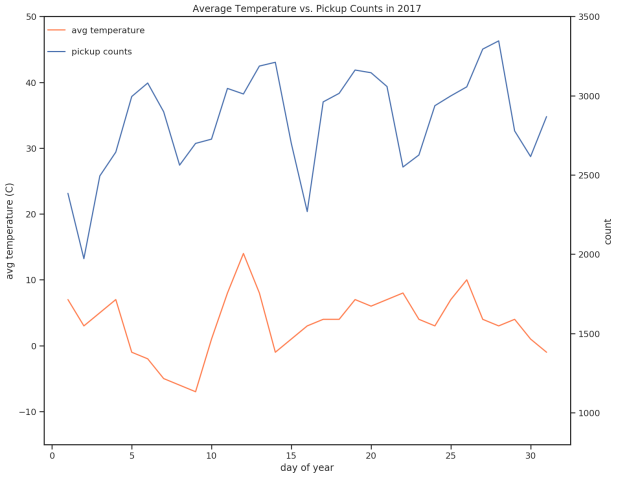
A. Data Visualization

After meticulous cleaning and data vetting, we produced a dataframe combining taxi and weather data named `nyc_df`. We used this data to visualize the relation between ridership and the three weather features. The following figures allow for a rough visualization of any time series correlations.









B. Training Preparation

After obtaining these timeseries visualizations we were adamant to move forward with the project, as the data seemed to suggest that extreme weather events affected ridership. For example, days with high humidity and precipitation tend to reduce ridership. This insight, combined with the inference that extreme weather reduces taxi mobility, implies that it becomes more imperative for.

We used `nyc_df` to produce `df_17` and `df_18`, both of which contain the day of the month, locationID, rainfall, average temperature, average humidity, and pickup count, for the years 2017 and 2018, respectively. Figure 4 shows how the dataframes look.

	day	PULocationID	rain	temperature_avg	humidity_avg	counts
0	1	158	0	7	41	12
1	1	170	0	7	41	1
2	1	48	0	7	41	29
3	1	211	0	7	41	13
4	1	234	0	7	41	16

```
[347] df_18.head()
```

	day	PULocationID	rain	temperature_avg	humidity_avg	counts
0	1	48	0	-11	50	9
1	1	152	0	-11	50	1
2	1	238	0	-11	50	9
3	1	164	0	-11	50	7
4	1	231	0	-11	50	20

Fig. 2: Snippets of df_2017 and df_2018

Instead of the traditional `train_test_split` method, we opted to make 2017 the training data and 2018 the testing data. This allows for us to test how well our model performs given how an entire month looks like. If this were to be used as an application or software, this implementation would make the most sense, as the user would simply input the aggregate data of the current year.

C. Linear Regression

Linear regression makes the assumption that there is a linear correlation between the independent and dependent variable. While this may seem easily dismissable, it is interesting to dissect this model. Looking at the data visualizations in the sections above, we noticed that on heavy weather event days, there was lower ridership. Hence, it may not be completely absurd to apply a linear fit to the model. However, because we also include day of the month and location ID as features, it becomes less reasonable to use a linear regression model.

For this implementation we simply used the `squared_loss` as the loss function and applied a grid search over a range of alpha values, optimizing for negative mean absolute error with 3-cross-fold validation. We then calculated the `mean_absolute_errors` and `mean_errors` for the training and test fits.

D. Random Forest Regression

Random forest regression operates on the idea of multiple decision trees. Each of these trees is built by randomly selecting data (with replacement) to create a bootstrapped dataset. Randomly selected features from the bootstrapped dataset are then used to decide what the root node for the decision tree is. This process is repeated multiple times to obtain a ”forest”. The data is then used to run down each of the trees and obtain the final answer.

For our experiment we searched through `[10, 40, 80, 160, 320]` to find the optimal number of trees in the forest, seeking to minimize the negative mean absolute error and using 3 folds cross validation. We then trained a model with this and obtained the `mean_absolute_errors` and `mean_errors` for the training and test fits along with the accuracy. We also sought the out of bag error, which simply tests datapoints on trees that did not use those datapoints to build their bootstrap dataset for a score. Moreover,

we also extracted the feature importances to view how important weather data was for our experiment when compared with day and location ID.

E. XGBoost Regression

XGBoost Regression is built upon the idea of decision trees, but it incorporates the errors of previous trees to build better ones in the future. Our XGBoost model takes the average of all pick up counts. The next tree built simply takes the errors into account by subtracting the calculated pick up average by the actual amount for each datapoint. However, the depth of each tree is specified. In each tree, each residual is added to the sum to obtain a guess. However, the XGBoost Regressor model avoids the high variance problem (overfitting problem) by assigning a learning rate to each of the trees so only small steps are taken in (hopefully) the right direction.

For our model, we searched through [4, 8, 10, 15, 20, 25, 32] as options for depth and [40, 80, 150, 600] as options for number of trees, minimizing for squared error and using 3 folds cross validation. We then obtained the the `mean_absolute_errors` and `mean_errors`, as well as the accuracy scores to test for performance.

F. KNeighbors Regression

Eventually, we also used the K Nearest Neighbors Regressor to predict the pickup count. The idea of KNR significantly deviates from decision trees, because KNR makes its decisions based on how close another datapoint is numerically. At first thought, this may seem to work because the algorithm would likely correlate location ID and give counts based on how close it is to another location. However, looking at the location file, we see that the IDs are not assigned based geographically. In other words, location 1 is not necessarily geographically close to location 2.

G. Expectations

From the analyses above, we did not expect promising results from the linear and KN regression models because of their assumptions. We did expect reasonable results from the decision-tree-based models because of their approach.

IV. Initial Results

Below, MAPE refers to mean absolute percentage error and MSE refers to the mean squared error.

A. Linear Regression

GridSearchCV selected alpha = 130 and produced the following results

TrainMAPE	TrainMSE	TestMAPE	TestMSE	R^2
70.58	591878	75.31	528067	-0.014

TABLE I: Linear regression results

B. Random Forest Regression

GridSearchCV initially selected for 320 trees. The range of tree selection was then updated to values = [320, 350, 400, 500, 600, 700, 800, 900], from which it selected 400 trees. Although, this value did not remain consisten after multiple runs.

TrainMAPE	TrainMSE	TestMAPE	TestMSE	Train Score	Test Score	OOB Score
27.94	95001	77.89	585321	0.84	-0.12	-0.1929

TABLE II: Random Forest regression results

We also plotted the feature importances for the random forest to see how the regression model weighted each feature. We learned that the model weighed LocationID as the most important feature, followed by day, humidity, temperature.

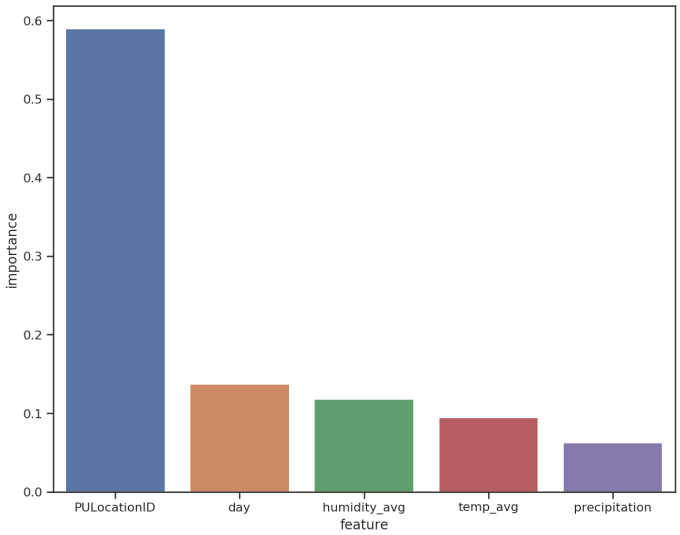


Fig. 3: Feature importance in Random Forest Model

C. XGBoost Regression

XGBoost selected for 40 estimators with a tree depth of 1.

TrainMAPE	TrainMSE	TestMAPE	TestMAPE	R^2
70.55	592156	75.23	527607	-0.013

TABLE III

Looking back, we realized that our results were wholly unsatisfactory. Each of the models seemed to perform surprisingly poorly with high error and practically no accuracy. Hence, we returned to re-evaluate the issues we may have caused or any extra analyses we may have been missing.

After some discussion we decided that we had to make the following changes to our approach:

- We should train on 2017 and 2018, then test with 2019. Again, the model has too little experience to make any meaningful predictions.
- Perhaps, we can use more than just January for training data. The models have too little experience with the days of the months.
- We can use PCA to find the best weather features, instead of picking certain ones.
- Our data is organized by day and cross validation is not shuffling the data. Hence, there may be a biased and misleading set of parameters being used for the model, leading to suboptimal performance

V. Revised Approach

After evaluating our options and our project timeframe, we opted to make three changes to our model. We included 2019 data and used a combination of 2017 and 2018 for training, while using 2019 for testing. We added a feature called day of the week, under the assumption that taxi data may be correlated with what day of the week is because of a cyclic pattern. Finally, we updated our 3 cross fold validation to shuffle the data. We also included results from the KNR approach to see if it fared better than the other models.

A. Linear Regression

GridSearchCV selected alpha = 1e-14 and produced the following results

TrainMAPE	TrainMSE	TestMAPE	TestMSE	R^2
71.80	1410132	81.74	1046655	-0.075

TABLE IV: Linear regression results after revisions

B. Random Forest Regression

GridSearchCV selected for 400 trees.

TrainMAPE	TrainMSE	TestMAPE	TestMSE	Train Score	Test Score	OOB Score
27.92	217035	82.58	1112606	0.84	-0.14	-0.12

TABLE V: Random Forest regression results after revisions

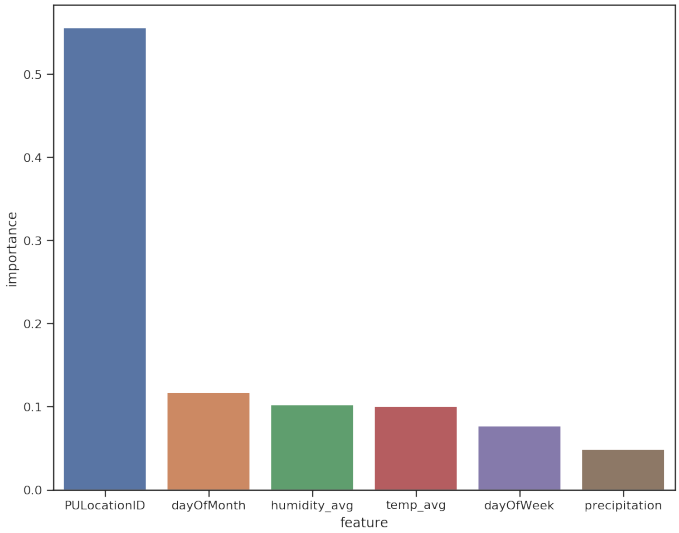


Fig. 4: Feature importance in Random Forest Model after revisions

C. XGBoost Regression

XGBoost selected for 40 estimators with a tree depth of 3.

TrainMAPE	TrainMSE	TestMAPE	TestMAPE	R^2
68.98	1319462	79.92	1008289	-0.035

TABLE VI

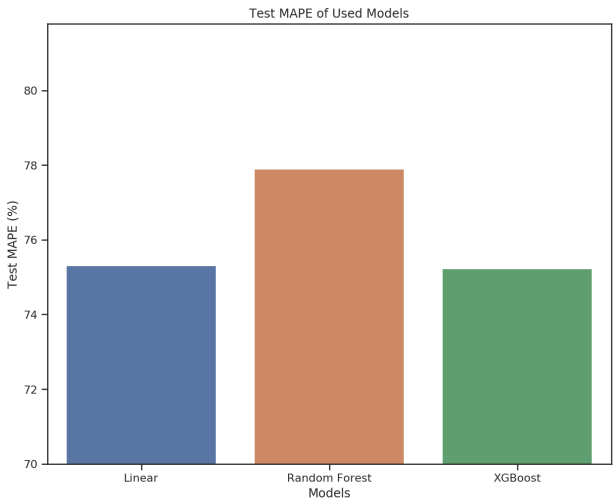
D. KNeighbors Regression

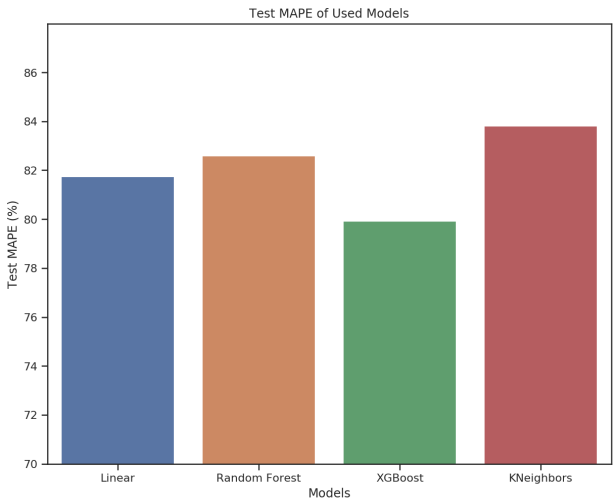
GridSearch selected for 40 neighbors

TrainMAPE	TrainMSE	TestMAPE	TestMAPE	Train Accuracy	Test Accuracy
70.06	1347247	83.80	1103577	0.059	-0.13

TABLE VII

E. Comparing Mean Absolute Percentage Error (MAPE)





VI. Discussion and Conclusion

To our surprise, the revisions we made resulted in increasing the Test MAPE and Train MAPE values. From this we can draw that perhaps in the initial attempt there was overfitting, resulting in a much lower Train MAPE. The increase in Test MAPE may also be a sign implying that there was overfitting in the original attempt, which falsely led to a lower MAPE. We decided to focus on the MAPE metric over accuracy because we were carrying out regression rather than classification. In the case of the latter, data points are placed into discrete bins, where it makes sense to use accuracy. In our case, the model is predicting an output from a range of values, where it is highly likely to never output the exact output. Hence, we use the percentage error as a metric to gauge how close it is to the ideal solution.

Overall, we believe our lackluster results were a combination of too little data and attempting to find a correlation where one does not exist. Other projects have shown that there is some but insignificant correlation between ridership and weather. Despite this, we sought to find a method based on weather data in curiosity to see if it would be possible. Although our results do not verify our claim that it is not possible, they do imply that it is not necessarily a simple task.