Citi Bike Station Shortage Prediction

Team CHARM

Hong Gao (hg1196), Andrea Wang (ayw255), Ramya Dhatri (rdv253), Manal Zorigtbaatar (mz1804)

Business Understanding

Introduction

Bike share system is growing rapidly across the U.S with over 88 million trips made on bike share bikes since 2010. Many cities such as Washington DC, Miami, Chicago has growing number of people using bike share system. But so far the Citi bike in NYC outrun the other bike share programs, generating around 40% of all bike share trips in U.S. (NACTO. 2017)

Since its initiation in 2013, Citi bike has rapidly developed and by now has 750 active stations with 12’000 bikes. With over 133’944 annual members with membership fee of 163$ a year and average 6 times more casual members using the bike each month than that of annual members. (CitiBike, 2017) Citi bike has its own interactive mobile map app indicate bike station locations, available number of bikes and docks. Affordable pricing, built out protected bike lane networks and expanding bike share system provides great opportunity for everyone.

Demand-Supply Mismatch

With the shear volume of users, it is important for both users and Citi Bike to obtain an accurate sense of demand/supply mismatch. The utilization of different stations varies significantly. In residential areas such as Lower East side and Uptown, stations emptying out in morning rush hours and tend to fill up in the night. Reverse patterns occur in stations located in office and commercial areas, such as Midtown and Downtown having low availability of bikes in the evening (Todd W. Schneider, 2016). Due to this reason, Citi bike staffs need to balance bicycles from full station to empty ones. On January 2016,

31’616 bicycles were rebalanced, averaging 1020 bicycles moved per day. During the busiest month, August, number of rebalanced bicycles peek to 134’028, averaging 4323 bikes a day (CitiBike, 2017). Addition to that, inaccuracy of the app during morning and evening rush hours is troubling. Because bike fleet is largely depleted during these hours. App seems to underreport the number of bikes in the rack of anytime by the factor of one to five bikes.

If Citi Bike could predict the shortage of the bikes prior 24 hours, they can act quickly and rebalance the bikes more effectively. One benefit of predicting shortage in advance is for Citi Bike to allocate bikes across stations so that customers are able to find bikes upon needs and have better experiences. This is very important for Citi Bike to secure long-time subscribers if the company is able to provide a better and reliable commute option. It’s also crucial for increasing short-time memberships when it’s more convenient for visitors to bike around popular places.

Data Understanding

Bike Station Data

Citi bike allows access to real-time station statuses via API calls. We identified historical Citi Bike station status data scraped by a developer (Abe Stanway, 2016) for roughly every 15 minutes from 03/2015 to 09/2017 with 9.498e+06 records for 943 docks. Features in the dataset include station ID, station GPS coordinates, timestamp, the number of available bikes, and the total number of docks. Descriptive statistics of raw data is in Table 1. This data allows us to explore temporal changes in available bikes in a fine scale. As stated above, we are interested in knowing when a given Citi Bike station has a shortage of bikes (defined as having less than 5 bikes available) at a specific time in the near future (ideally 24-48 hours). Thus, our target variable is shortage, binary with 1 indicating a shortage of bikes at a given station and 0 indicating no shortage. Each data instance is the indicator variable for shortage at a given station at a given time with other features. For example, station A at

8:00 is one instance, and station B at 8:00 is another instance. This structures our research question into a classification problem where given an exact time, i.e. 08:00 or 17:30, and a station ID, the model will classify target variable as 1 or 0.

Weather

We assume there are certain trends of bike usage in seasonality, for example, February being the most unpopular month and August being the popular month for using Citi bike. This can be represented by temperature, precipitation, wind speed, and visibility under the assumption that the lower the temperature and visibility, the higher precipitation and wind speed, the lower the bike usage there would be. Raw weather data is obtained from Kaggle through Wundergrounds API, which contains the above mentioned features by hour (Wundergrounds API, 2017).

Subway Stations

Literature has shown that bikeshare stations located near busy subway stations and bicycle infrastructure see greater utilization(Noland et al. 2016). Adding subway stations as features would likely capture the variations in bike shortage. Subway station location dataset is downloaded from NYC Open Data portal (NYC open data, 2017), which contains name, ID, latitude, and longitude of all 473 subway stations in New York City.

Data Preparation

Data Cleaning and Splitting

Due to considerations of the data volume and seasonality, we decided to keep data from 01/2016 to 10/2016 to save computation and modeling time. We removed all the 'Coming Soon' stations that has no bikes and all the stations with total number of docks as zero. Some of the bike stations have wrong geolocations, so we only kept those that are located in New York City metropolitan area and Jersey City. After cleaning, there are 653 stations with 8,677,508 records (Table 1). Since we are interested in

predicting shortage in the future, we split training and test set based on time, i.e., 01/2016-08/2016 as training set and 09/2016-10/2016 as test set.

Feature Extraction and Engineering

1. Target Variable: **'Shortage' (**Binary)
    a. 1 when the available bikes in a given station is less than or equal to 5. 0 otherwise.
    b. Base rate for Training dataset: 0.40
    c. Base rate for Testing dataset: 0.36
2. Basic Features included Baseline model:
    a. Bike Dock related features:
        i. **Latitude** and **Longitude** of each bike dock
        ii. **Total docks** of each bike dock

Notes: Instead of using bike dock id as feature, we use longitude and latitude of bike dock because latitude and longitude are numerical values and unique to each dock. We can thus treat latitude and longitude as unique identifiers for each dock and at the same time avoid creating numerous more columns of dummy variables for each dock id. Also, the model will be able to capture the information of the docks nearby by latitude and longitude. We believe this will further enable our model to learn to predict for new bike docks in the future if retrain.

    b. Time related features:
        i. Dummy variables for Day of Week (**Monday**, **Tuesday**, …, **Sunday**).
        ii. Binary variable for **Weekday** or not (1 if weekday)
        iii. Binary variable for **Holiday** or not (1 if national holiday)
        iv. In order to capture the cyclical effect of time variables, we create 2 variables, time_x and time_y by taking sin and cosine of the original hour (24) of the day and minute. The transformation formulas are as followed:

1. **time_x =** $cos\,(hour + \frac{minute}{60}) \times 2\pi \div 24)$
2. **time_y =** $sin\,(hour + \frac{minute}{60}) \times 2\pi \div 24)$

   This is to capture that midnight (hour 24) is actually very close to 1 a.m. (hour 1) even though numerically they seem far away.

3. Features added to train the Final model:
   a. Bike Dock related features: **Subway Stations**
      i. **Longitude** and **Latitude** of the nearest subway station.
      ii. **Euclidian Distance** from each bike dock to its nearest subway station.
   b. Time related features: **Weather**
      i. **Temperature**, **Wind Speed**, **Visibility** as continuous variables.
      ii. **Rain** and **Snow** as binary variables. (1 if rain/snow; 0 if not)
      iii. The weather data is roughly an hourly data from Wunderground API. If there are several data instances in an hour, for the continuous variables we averaged them, and for the binary variables, we treat them as 1 if the averaged value is greater than 0. The missing values were filled with the data from previous hour. We then merge weather data with original dataset on hour as key.

Modeling & Evaluation

Evaluation Metric

We used AUC because it is a metric to evaluate ranking mechanism. Essentially we want our model to rank the probability of bike shortage across every bike docks at a given time as accurately as possible so that Citi Bike knows which bike docks are in more ‘emergency’ states than others in order to allocate its limited resources (for example, personnel to move the bikes) or design programs to encourage users helping move bikes accordingly.

Baseline Model

1. Data: instead of running on whole dataset, we randomly picked 10 docks (listed as below) to train our baseline models due to computational limitation.

Dock name (Dock id): 'St James Pl & Pearl St' (82), 'Warren St & Church St' (152), 'Clinton St & Tillary St' (322), 'E 19 St & 3 Ave' (325), 'Carmine St & 6 Ave' (368), 'W 31 St & 7 Ave' (379), 'W 4 St & 7 Ave S' (380), 'Duffield St & Willoughby St' (390), '8 Ave & W 33 St' (490), 'E 20 St & Park Ave'(503)

2. Models: We picked scikit-learn default settings of Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, and Polynomial Regression with up to 3-degree interaction terms. The resulting ROC curves and AUC scores were shown in **Figure 2**.

In general, Logistic Regression is the simplest model that takes the least amount of time to train. However, given its linearity, it doesn't perform very well on our dataset. To illustrate this, take the location features of bike docks in our dataset as example. Logistic Regression only learn bike shortage North versus South (by Latitude) or East versus West (by Longitude). It doesn't have the capability to learn that in residential area such as Uptown and Lower East Side in the morning rush hours have high probability of bike shortage, opposite to office areas like Midtown. Tree structures, however, can better capture this spatial variation by bisecting the space into smaller and smaller regions, leading to superior classification performance. Therefore, it is no surprise that *Decision Tree, Random Forest, and Gradient Boosting* performed much better on our dataset. It is also expected that the ensemble methods (Random Forest and Gradient Boosting) gave better results than single algorithm (Decision Tree). Similarly, compared to Logistic Regression, Polynomial Regression is able to capture spatial variations as well as associations between different features with interaction terms.

Looking closely to the ROC curve of Random Forest, Gradient Boosting, and Polynomial model, we can see that they performed very similarly at middle to lower left corner (Regions in red square in **Figure 2**).

Since the goal is to tackle the docks with higher probability of shortage accordingly with limited resources, we care more about the accuracy of ranking when the probability of shortage is mid to high than the ranking of bike docks with lower probability of shortage. We can see that Random Forest performs better in the red-squared region. In addition, it takes less time to train Random Forest than Gradient boosting and Polynomial (in default setting), hence, we decided to pick **Random Forest** as our model to further develop on.

Polynomial Regression has higher AUC and outperform other models on the 10 stations, however, when fitting the same 3-degree interaction Polynomial on the complete dataset, the performance dropped significantly (AUC = 0.619). We believed it is because the Polynomial is able to fit on less data but is not complex enough to fit every station therefore it has experienced severe underfitting. Hence, we are more convinced that Random Forest is a better algorithm for our dataset.

Improving Upon Baseline Random Forest Algorithm

1. Before feature selection and hyperparameters tuning, we ran default Random Forest Algorithm on entire dataset (all stations) with subway and weather features (22 features in total) to get a comparison baseline. The resulting AUC is 0.75.
2. Hyperparameters tuning:
    a. **Splitting Criterion**: Gini v.s. Entropy:

       Keeping all other settings at sklearn default, we have observed that the model with Entropy criterion gives a little better AUC (0.752) than Gini (0.749). Hence we set Entropy as our splitting criterion. Please refer to **Figure 3** for the ROC curve.

    b. **Forest-level hyperparameters**/ Number of Trees and Number of features in each tree:

       To test forest-level hyperparameters, we used entropy as splitting criterion and no maximum depth of each tree in order to overfit every tree. We tested 50, 100, 200, 300 trees each with 3,4,5,6 features. (number around default sqrt(22) features)

From **Figure 4** we can see that 300 trees with 5 features in each tree gives the best AUC score of 0.771. This is about 3% improvement from default model.

With 22 features in our dataset, we expect the model will perform better with even more trees. However due to the computational and time constraint, we only tested to 300 trees. Also the model did not have signs of overfitting.

3. Feature Selection: We have run hyperparameters tuning based on the first 11 important features (by mutual information) identified by basic random forest model. However, even with 500 trees, our model would only give an AUC of 0.74, even worse than Random Forest with default setting on all the 22 features. Therefore we decided not to exclude any feature in our dataset.
4. Define the best threshold based on false positive and false negative to minimize expected cost: With the best performing model, we implemented the metric proposed by Provost and Fawcett to find a best threshold: $m = \frac{c(FP)p(N)}{c(FN)p(P)}$

   , where c(FP) is the cost of false positive, c(FN) is the cost of false negative, p(N) is the prevalence of negative cases, and p(P) is the prevalence of positive cases. In our dataset, the numbers of Negative cases is 1.85 times of Positive cases. Assuming the cost of false positive is 5 times the false positive, we arrive at the best slope $m = 0.37$. The corresponding threshold is 0.08. Please refer to **Figure 5** for the ROC curve of the models we pick with corresponding threshold based on our assumption.

Time Series

Besides the models we ran above, we also explored Time Series models and compared it against the baseline models on the same dataset of the 10 docks with the same train and test split. The seasonality has been dealt with in each of the Time Series models. One point to be noted here is, the peak winter

months might show a drastic change in the usage of bikes and data distribution could be very different from the data that the model has been trained on. One dock in the 10 docks subset is located at West 4th Street (next to NYU). We suspect that if a dock is next to a School the usage might go down during the holiday months. Thus, we have trained a whole year's data and tried predicting on the Winter/Holiday months on this particular case. And the model has been differentiated to the first order to capture these changes and it gave an MSE of 31.2 on the December month's data.

Data Preparation for Time Series Models

The dock data was available generally for every 15 minutes. As we are mainly interested in predicting the shortage for every hour, the data has been condensed from 15 minute interval to 1 hour interval by choosing the least number of bikes available at the dock during any hour. All missing values have been replaced by the average of the immediate previous and next values available. Instead of omitting the outliers we've fed them into the model so that the stationarity/non-stationarity of the model could be observed. The data for the Time Series has been entirely processed on ***PySpark***.

Methodology

We've used the **Box-Jenkins method** to plot Time Series while keeping every model a univariate one.

Step 1: Checking for stationarity

As a first step, the pattern for the number of bikes available at each dock was checked for stationarity. To do this, a sequence plot was run on the 8 months of the training data. If there were any visible outliers and if the dock has shown any increasing/decreasing trends then it has been differentiated to the first order. The order of the differentiation was used to determine the parameter 'd' of the ARIMA model. In rare cases, we had to differentiate the model to the second order and it has worked on fine on few but, on the others, it would cause a problem either with the convergence or the singular value decomposition. In such situations, the rest of the parameters were tuned in a way they would work fine even if there has been a slight deviation from the actual Box-Jenkins methodology.

Step 2: Determining the type of model

In order to determine if we have to use an Auto-Regressive (AR) or a Moving Average (MA) or an Auto-Regressive Integrated Moving Average (ARIMA) model, we've studied the auto-correlation (ACF) and the partial auto-correlation (PACF) plots of the data. Again, in most of the cases, the ACF plot has shown a strong positive correlation at lag 1. Also, it had an alternating behavior before decaying to a zero. This hinted at an Auto-Regressive model. Here too, there were some exceptions. Few of the docks have exhibited seasonality (ex: The dock next to NYU). For such docks, there has been a Moving Average term included in addition to the differentiation to make it stationary. The **Table 2** in Appendix described the Box-Jenkins methodology that has guided us in choosing the appropriate model.

Step 3: Determining the order of the AR/MA/ARIMA model:

After determining the model from the previous step, the order of the terms to be used have been determined from the ACF and PACF plots. If it's an AR model, the number of terms to be included were the number of significant lags (p) observed in the PACF plus one, i.e, p+1. If it's an MA model, the number of terms to be included were the number of significant lags (q) observed in the ACF plus one, i.e, q+1. And if it's an ARIMA model, the order of differentiation (d) was included. For all these models, the significance of lags were determined by a 95% Confidence Interval. In straightforward cases like that in **Figure 6**, the number of lags was determined by a simple count. While in more complicated cases like that in **Figure 7**, since it would *not* make sense to include the number of terms as 30 (implying a seasonality factor of 30 units) trial and error has been carried out to determine the appropriate number of lags.

<u>Miscellaneous Things:</u>

a) Though Box-Jenkins method helped us in predicting the actual number of bikes available at every instance with an MSE as low as 4.44, we were more interested in being accurate in the

determining the whether or not the number is less than or equal to 5. Hence, **we could afford slight deviation from the expected value above a certain number of bikes and re-engineered the models to work more efficiently on the marginal values**. We therefore tuned the models to predict the numbers in the range 0-10 more accurately than those above. This resulted in some models with higher MSE (37.9 or 33.5) but lower number of false predictions.

b) The AR models took about 40 mins while the mixed models and ARIMA models took a duration spanning from 4 to 7 hours.

c) Almost 27 models have been run for all the 10 docks (roughly 2-3 for each dock) to arrive at the best possible model.

Result

Appendix **Table 3** for summary of model prediction and **Table 4** for comparison to other baseline models. In conclusion, Time Series Models give a much better AUC than other baseline models.

Deployment

Our predictive model can be trained on the newest station status data Citi Bike has collected recently, i.e., 01/2017-12/2017, and predict which stations are at risk of shortage in the future. Our proposal is to integrate the model into Citi Bike rebalancing management information system. One advantage of our model is that we are more certain when the probability of shortage is mid to high so Citi Bike will be able to allocate its resources by priority. Another advantage is that docks don't have to be constantly monitored. Since proximal weather forecast is more accurate, Citi bike will have more days to plan for rebalancing and thus to effectively redistribute the bikes several hours prior to those stations.

There will be new patterns appear as Citi Bike start to rebalance the bikes according to our predictive model. In order to capture new patterns and avoid lower performance of the model, we recommend

retraining on the new data and using newest data to replace the oldest data from training set. In that way, we are enabling the model to catch up with the latest patterns, including patterns from new docks. We also recommend Citi Bike to record information related to bike reallocation, such as time, the number of bikes, origin and destination docks. In this way, if there is a problem with model performance, we will be able to retrain the models by adding more features to reflect the bike reallocation.

Discussion and Future Improvements

In this paper, we developed a full random forest model that makes use of **temporal** and **geographical** data to predict bike shortage in citibike docks. We observed an underfitting on our model due to the lack of temporal information. Specifically, because of the way we split our training and test set, we did not include date as our features, so our random forest model was not able to learn from a timeframe exceeding 24 hours. Judging from the performance of Time Series models, the ability to learn from previous days is very important for our target predictions. However, random forest model still carried the advantage of making use of the geographical data, essentially capturing the status of nearest bike docks to predict shortage of a specific bike dock.

The result of time series exploration suggests that it is highly accurate when compared to the other models like Logistic Regression or Random Forest or Gradient Boosting, and it doesn't require dealing with multiple features as the interactions are, indirectly, included as seasonality or non-stationarity. It also gives a better visualization of the target variable with time and makes it easier for us to focus on what is essential. However, time series also carries its disadvantage. First, potential insightful patterns or interactions of other features could be missed out (This might be valid only in our case as we kept the model to be univariate). Second, it's sometimes

extremely time consuming and hard to determine the parameters for mixed models and might require several trials to arrive at the best model.

Hence, for our future improvements, we are hoping to select a whole year as our training dataset and additional random two months in the other year as test set, adding **cyclic day of year** as features to run random forest and improve upon it. We'd also want to try out Polynomial Regression with higher degree interaction terms with the new dataset and more features to test if our assumption of model underfitting is correct and to potentially improve its performance. Expanding time series models to every other bike docks and comparing their performance to the other models is also a valuable direction.

Moreover, we'd also like to explore more tailored evaluation metric. For example, if we get to know more about the cost of moving bikes by Citi Bike personnel as well as customer unhappiness of not getting a bike, we can set an evaluating metric with a goal of maximizing customer happiness with limited cost. To illustrate, ideally we would also take into account the distance and route between docks with shortage, the number of people Citi Bike needs to send, and the time to reallocate bikes in order to lower the cost of operation.

Our models were built in the perspective of Citi Bike and focusing on solving bike shortage problem as a business solution. In the future, if we extend our models for citibike users, we can also design an evaluation metric with the goal of minimizing expected walking time to get a bike.

Acknowledgement

All team members contributed to the final project equally. In particular, Manal contributed to identifying our research idea. She reviewed many related literature and compiled the business understanding section. Ramya was responsible for all parts related to the Time Series sections of our papers, including data preparation (on PySpark), binning the data, building time series models and time series

model discussion. Andrea and Hong collaboratively completed data preparation, feature engineering, baseline models, and improvements upon baseline models as well as discussion section and the write-up of the corresponding parts. The deployment section is a result of countless discussions and collaborations between all members.

We thank Brian for the great support, including but not limited to providing us valuable suggestions in shaping our business understanding, model selection, and finding suitable evaluation metrics.

Appendix

Figures

Figure 1. Citi Bike Docks after Cleaning and New York City Subway Stations

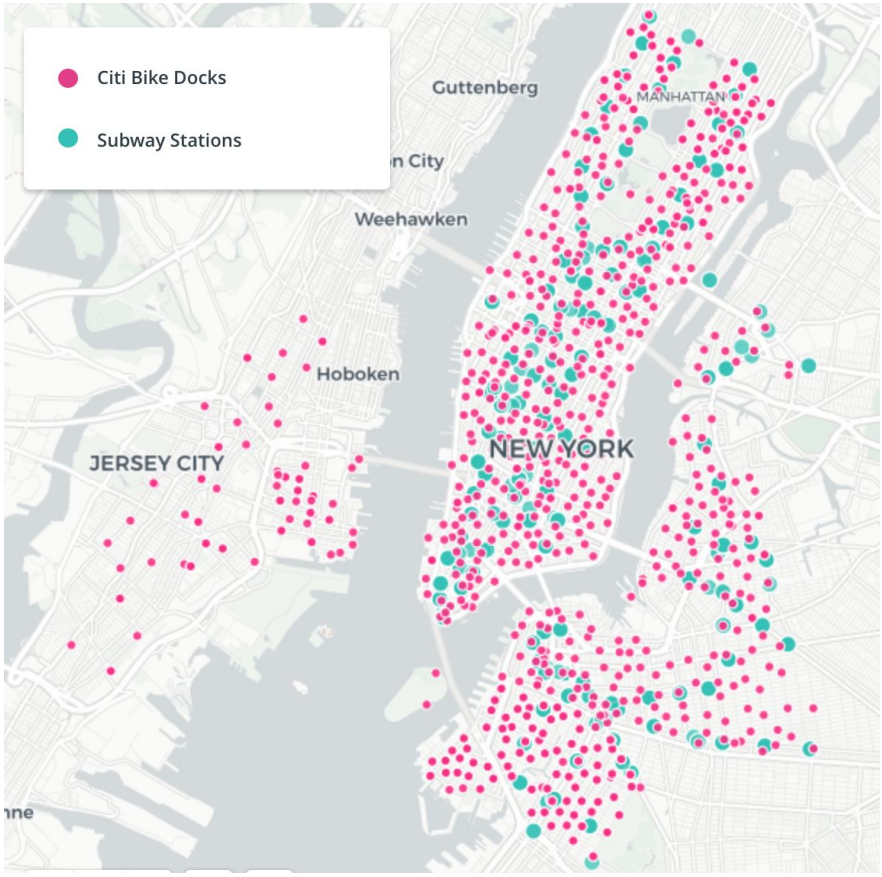


Figure 2. ROC curves and AUCs for Baseline Model

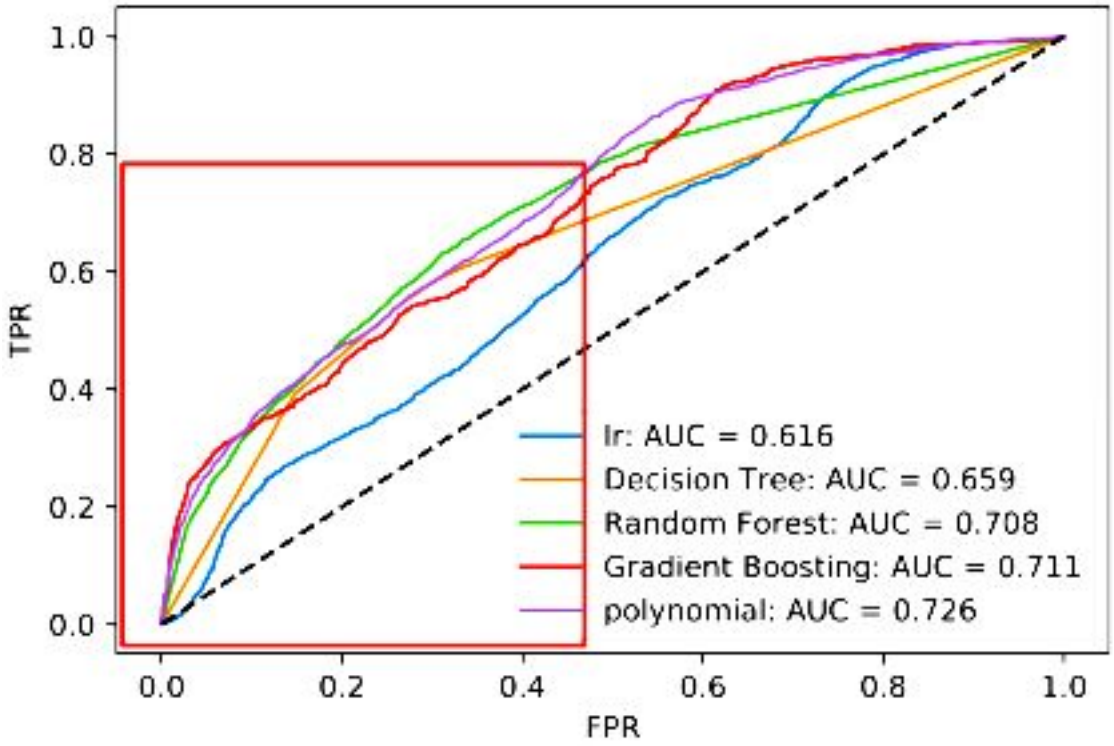


Figure 3. Comparing Splitting Criterion of Entropy and Gini

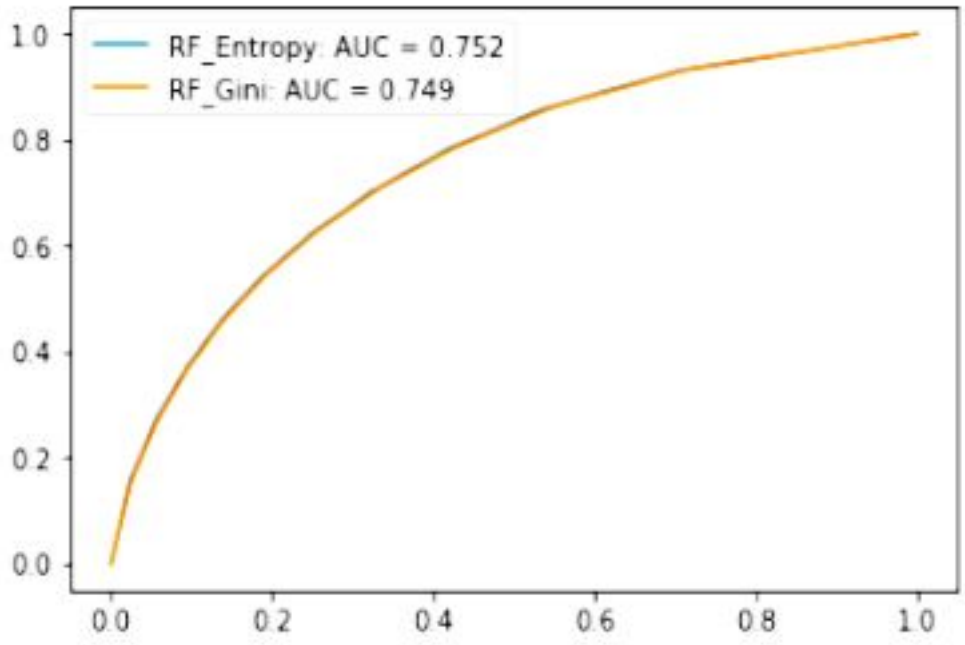


Figure 4. Test AUCs for tree-level hyperparameters

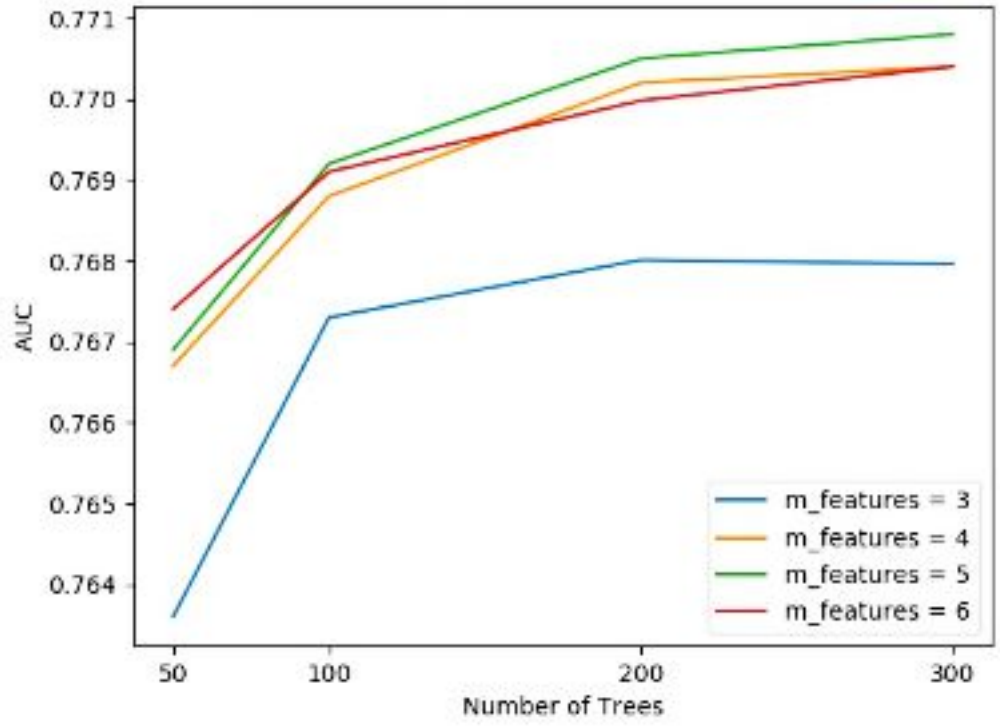


Figure 5.

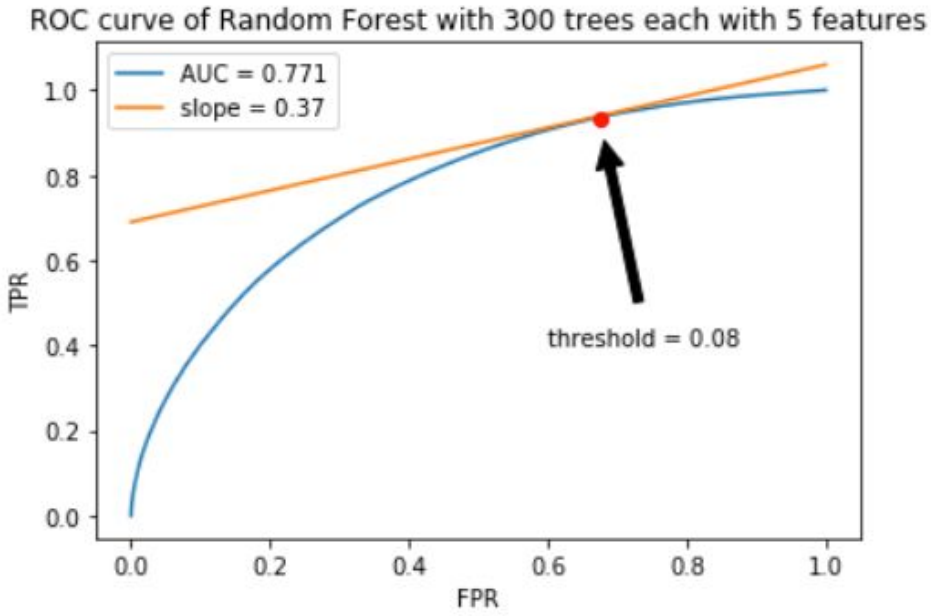
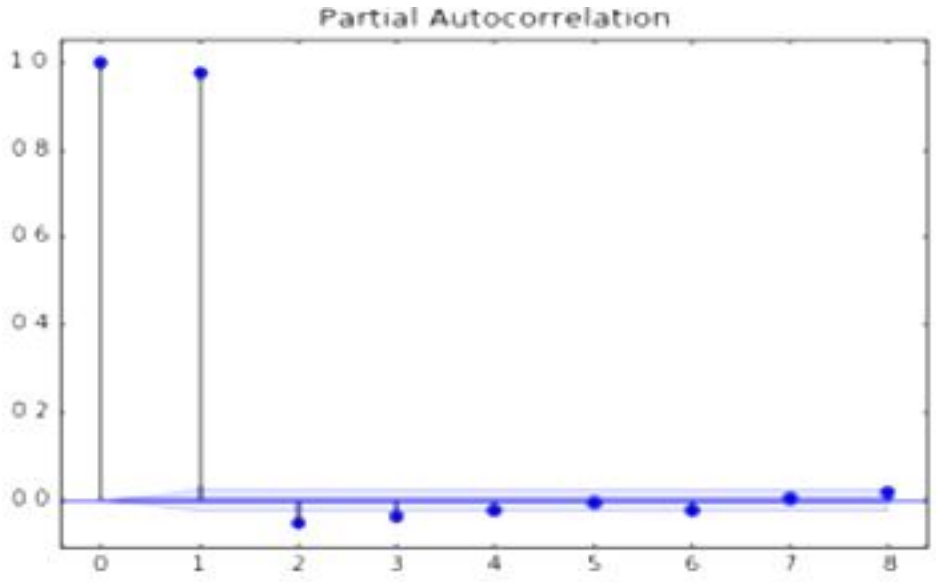
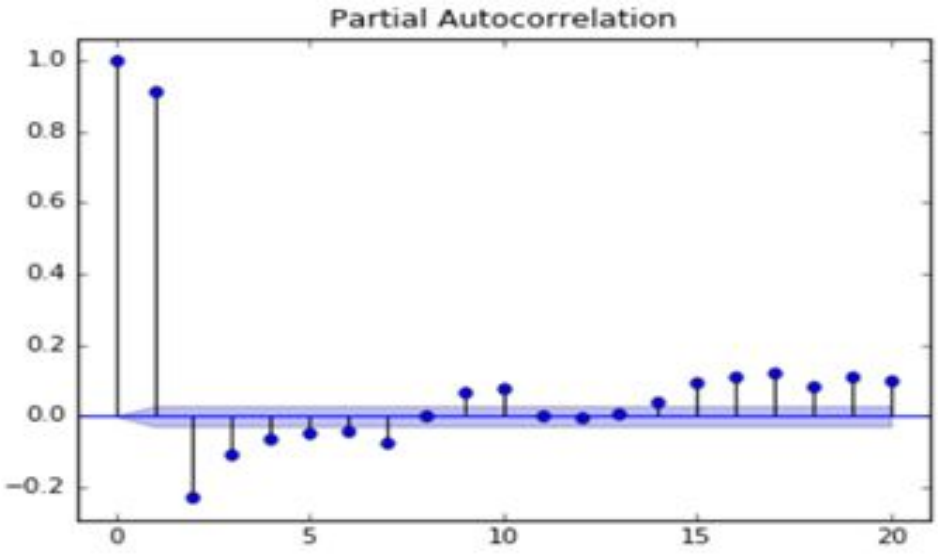


Figure 6. ACF plot for dock 380 which has significant lags of just 3 in number



In this case, the order of AR model that has been found to work best was 4.

Figure 7. PACF plot for dock 503 which has significant lags of 30 in number



In this case, the order of AR model that has been found to work best was 12.

(Showing only the first 20 lags for better visualization)

Figure 8: Time Series With Least MSE (Dock 82, MSE = 4.44)

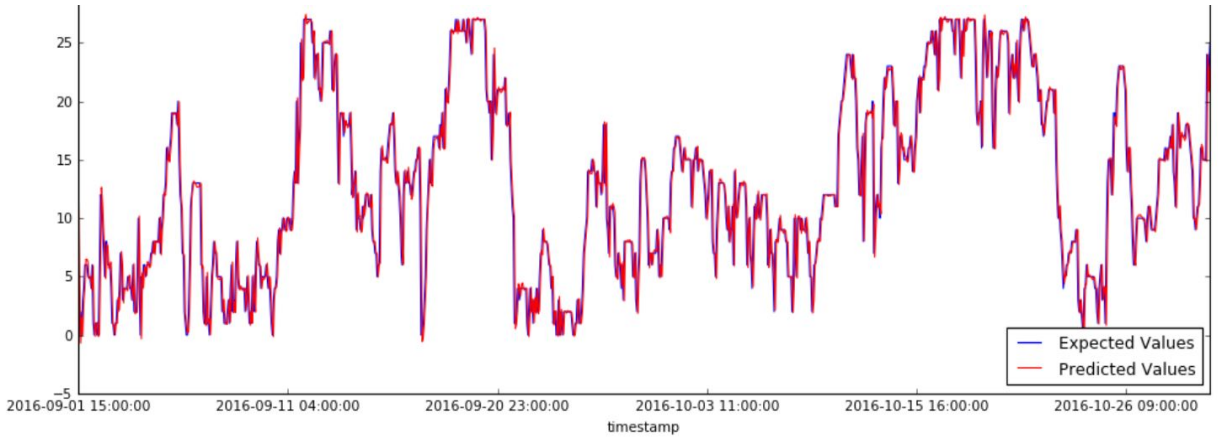
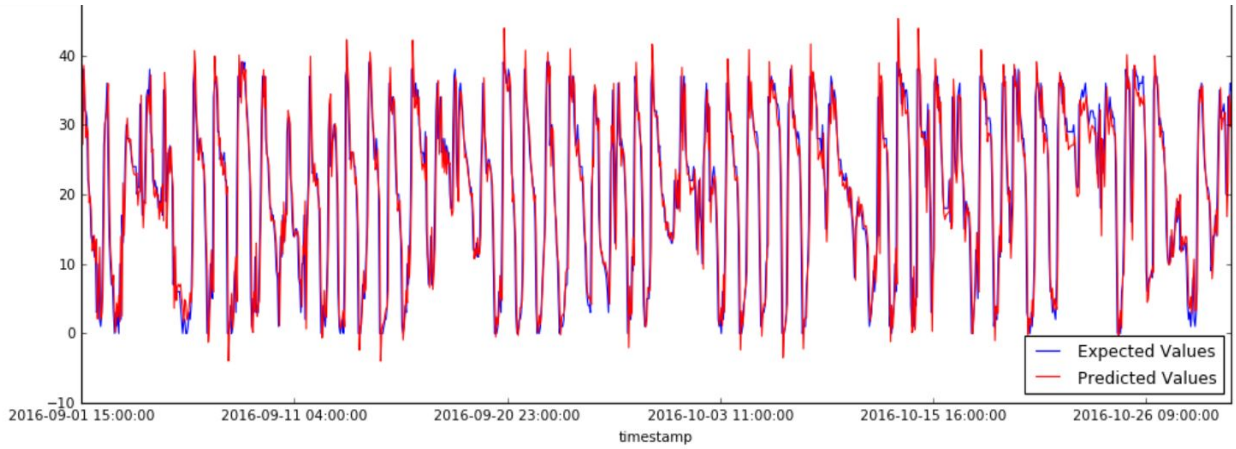


Figure 9: Time Series With Worst MSE (Dock 380, MSE = 37.9)



Tables

Table 1. Selected Summary Statistic on Raw Citi Bike Dock Data

	Count	Mean	Std	Min	Max
Available bikes	9.498e+06	10.92	10.69	0	67
Available docks	9.498e+06	20.12	13.45	0	67
Total docks	9.498e+06	32.06	11.88	0	2727

Table 2. Table describing shape of ACF plot to choose appropriate model (Source: Box-Jenkins)

Shape Of The ACF Plot	Indicated Model
Exponential, decaying to zero	Auto-Regressive model
Alternating positive and negative, decaying to zero	Auto-Regressive model
One or more spikes, rest are essentially zero	Moving Average model
Decay, starting after a few lags	Mixed Auto-Regressive and Moving Average model
All zero or close to zero	Data are essentially random
High values at fixed intervals	Include seasonal Auto-regressive term
No decay to zero	Series is not stationary

Table 3. Summary of the number of true predictions and false predictions for each dock in the Time Series models.

Dock ID	True Predictions	False Predictions	Total Predictions
82	1006	75	1081
152	1007	74	1081
322	946	135	1081
325	944	137	1081
368	1010	71	1081
379	983	98	1081
380	986	95	1081
390	975	106	1081
490	959	122	1081
503	932	149	1081

Table 4. Comparison of various metrics for different models.

Model	**AUC**
Logistic Regression	0.616
Decision Tree	0.659
Random Forest	0.708
Gradient Boosting	0.711
Polynomial	0.726
Time Series	0.888

References

1. Bike Share in the US: 2010-2016, NACTO
   https://nacto.org/bike-share-statistics-2016/
2. A Tale of Twenty-Two Million Citi Bike Rides: Analyzing the NYC Bike Share System, Todd W. Schneider,2016
   http://toddwschneider.com/posts/a-tale-of-twenty-two-million-citi-bikes-analyzing-the-nyc-bike-share-system/
3. Citi Bike Monthly operating reports, 2016-2017
   https://www.citibikenyc.com/system-data/operating-reports
4. Historical Bike Station Status (roughly every 15 minutes), Abe Stanway, 01/2016 - 12/2016,
   https://github.com/astanway/citibike-data
5. New York City Taxi Trip -Hourly Weather Data, Kaggle, 2015/12/31 - 2016/12/31,
   https://www.kaggle.com/meinertsen/new-york-city-taxi-trip-hourly-weather-data
6. Noland R. B., Michael J.Smart M. J., and Guo Z., Bikeshare trip generation in New York City, *Transportation Research Part A: Policy and Practice*, 2016,
   https://www.sciencedirect.com/science/article/pii/S0965856416307716
7. Subway Stations, NYC Open Data,
   https://data.cityofnewyork.us/Transportation/Subway-Stations/arq3-7z49/data
8. Provost F., Fawcett T., 2000, Robust Classification for Imprecise Environments
   http://pages.stern.nyu.edu/~fprovost/Papers/rocch-mlj.pdf