Predicting Trip Duration and Forecasting Ride Demand for the NYC Taxi Cab

Justin Kim

July 10, 2019

Abstract

Using the a dataset from Kaggle.com, this paper will explore the various insights regarding the New York City taxi cabs using various information such as pickup location, drop off location, time of day, etc. in order to predict total ride duration of a taxi ride, and also forecast taxi ride demand. Techniques such as linear regression, gradient boosted trees, seasonal ARIMA models, and neural networks (LSTM) will be used as predictive models.

1 Introduction

The New York City yellow taxi cabs are one of the most distinguishing symbols of the city that never sleeps. While there are numerous ways to get around the city, perhaps the most convenient way is via the taxi cabs. It gives people a chance to take a breather, sit back, relax, and use their phone for recreational or professional purposes. In a city that is always cluttered with millions of pedestrians, cars, public transit, and others, it would be beneficial to understand just how long you can expect your trip from point A to point B to take.

2 Problem Statement

Given various data about geolocation, trip duration, and the weather, it would be beneficial for both the taxi driver and the passengers to know when and where the taxi demand would be high, and also to know how long a trip can be expected to take.
By clustering based on geolocation, and by the time of day (week, month, year, etc), we can expect to see when and where there are surges of taxi demand. We can also expect insight into average ride duration, along with whether or not ride duration has correlative properties with regards to geolocation and the weather. By predicting the appropriate amount of time a ride can be expected to take, both the customers and the taxi drivers can optimally plan the use of their time spent in transit.

3 Data

Data Description :

- The primary dataset was acquired from Kaggle (https://www.kaggle.com/c/nyc-taxi-trip-duration/data).
  The dataset contains more than 1.4 million entries recording taxi trips detailing columns such as the following: unique taxi vendor id, pick-up and drop-off geolocation coordinates, trip duration, passenger count, and more.
- A second, complementary data set was found at (https://www.kaggle.com/mathijs/weather-data-in-new-york-city-2016), and this contained the entries regarding data gathered from a weather station in Central Park for the first six months of 2016. For each day, there are relevant columns pertaining to the minimum temperature, maximum temperature, average temperature, precipitation, new snow-fall, and current snow depth. Obviously, hazardous weather conditions adversely affect traffic conditions, so by merging the weather data of central park (an arbitrarily chosen point to represent the weather condition of all of NYC), we can take this into account when predicting ride duration.

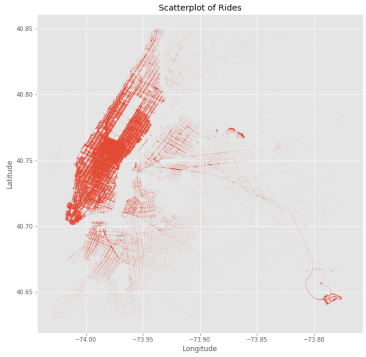
Data Cleaning and Wrangling :

- First, the columns of "pickup_datetime", "dropoff_datetine" and "pickup_date" were all converted to datetime modules using pandas. The first two columns contained up to the second increment, where the "pickup_date" column retained to the day of the trip.
- There were some obviously erroneous trips, where the recorded trip duration was more than 979 hours long. In order to account for a more "normal" data, all trips deviating further than 2 stand deviations away from the mean of the trip duration were disregarded. More than 1.43 million data observations remained in total after this filter.
- Other datetime related variables were made from the pick up time such as month, day of week, and hour.
- The weather data was cleaned such that precipitation, snow-fall, and snow depth was obtained. There was a value 'T' for trace amounts, which was converted to 0. After these values were converted to float types, the weather data was merged by date to the NYC taxi data along with the minimum temperature of each day.
- The longitude and latitude borders of New York City was found through a simple internet search (an array of -74.03, 40.63, -73.77, 40.85) and data points were further filtered to be within these city limits. 1.43 million observations remained in total.
- Finally, distance metrics such as the haversine distance, Manhattan distance, and distance bearing were found by applying the appropriate equations to the given pickup/dropoff longitudes and latitudes.

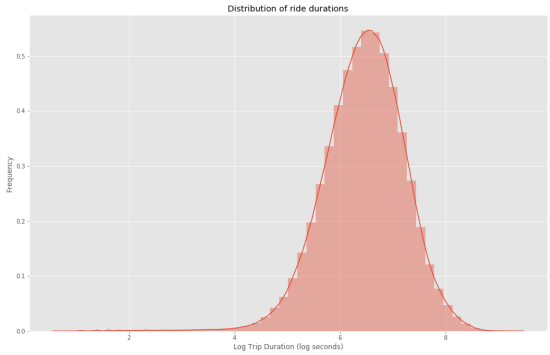
4 Exploratory Data Analysis

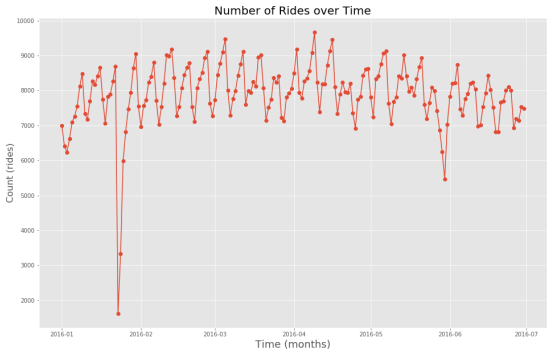
4.1 Insights

The remaining data points from the cleaned data set depicts the greater New York City area when displayed on a scatter plot. We can see that a high den-

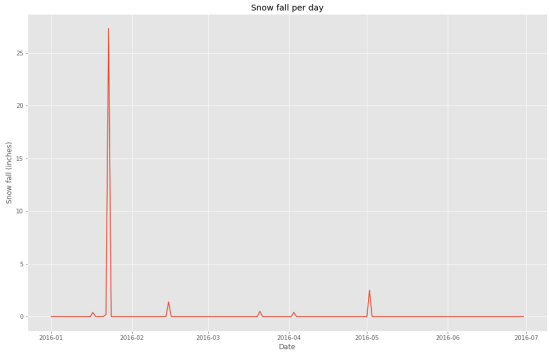


sity of rides originating within the island of Manhattan as expected. Looking at our main predictor variable of log trip duration, we see that it is normally distributed with a mean of around 6.5 log-seconds.





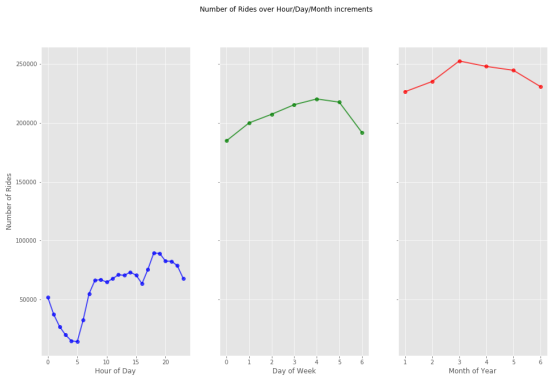
As for the dip in the number of rides in late January, we must think of outside factors that may have caused this significant dip in the number of rides. Since January is in the middle of Winter, we may expect there to be extremely inclement snow related weather conditions that may have resulted in a city wide driving ban. Looking at the average snowfall per day plot, we can confirm that this is indeed the case, with there being 27 inches of snow fall on that same day.



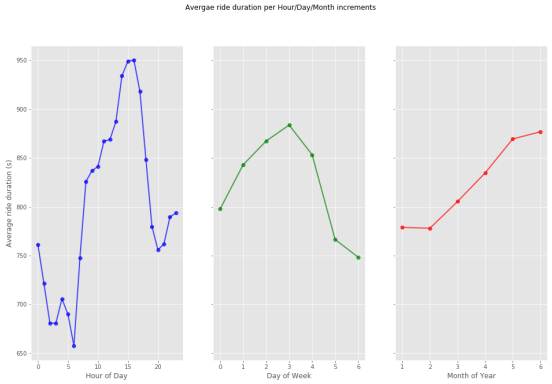
4.2 Average Trip Duration, Distance, Speed

The trends of the number of rides over different increments of time may be important in determining when a customer or taxi driver should plan to be

active. The number of rides taken over the day hits a minimum at 5am and a

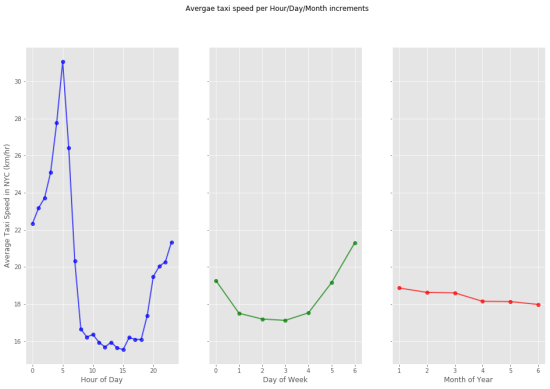


maximum at around 6pm. The number of rides taken over the days of the week is at a minimum on Monday (0 on the x axis) and gradually rises to peak on Friday (4) before dropping sharply on Sunday. Over the months of the first half of the year, January has the least number of rides, while March has the most. However, this January count of rides was significantly affected by the lack of rides on the 23rd, and the following days also serviced quite a bit below the average number of rides. During the day, the shortest trips also occur around



5am, when there are the fewest number of trips. This may be due to the lack of traffic at the hour, and the lack of need for people to go "far" in a cab at that hour. The average ride duration seems to peak at around 3pm each day, and continues to decrease in duration until around 8pm. Surprisingly enough, the longest average ride durations did not occur at the customary "rush hour" times of 5-6pm.

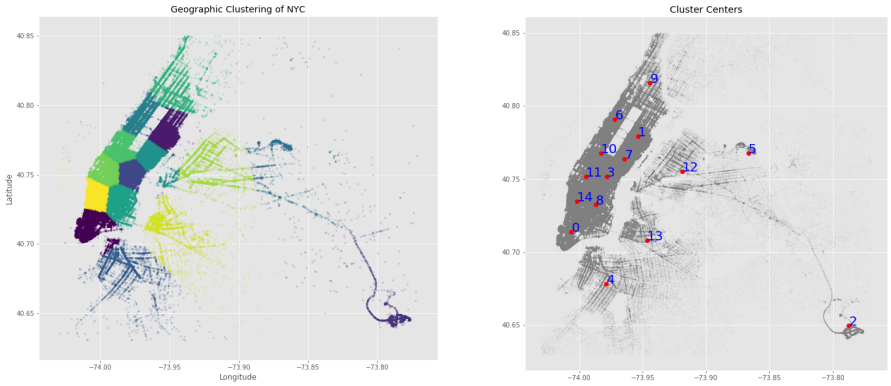
During the week, Sunday has the lowest average ride duration followed by Saturday. Somewhat intuitively, more people have to work and have a pressing need for cabs during the weekdays, and that may be attributed to the higher taxi ride durations that gradually peak on Thursdays. On average, people are in taxis for less amount of times on weekends. Finally, over the course of the first 6 months of the year, the average ride duration is the shortest during January-February, and grows almost linearly until June (when the average ride duration is at highest).



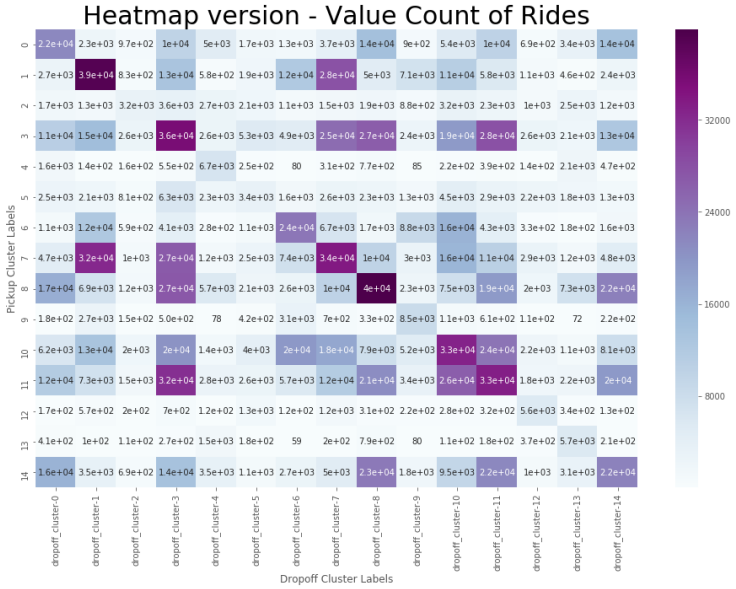
Here we can see that during the day, the taxis are cruising at their fastest average speeds of around 31 km/hr at 5 am, which we know to be a time of day where there are the fewest number of cars around. Otherwise during the day, starting at around 8am, the cars crawl around 16 km/hr until 6pm, where the speed gradually climbs back up until 5 am the next day. During the week, Sundays have the fastest taxi speeds on average. This is again the day of the week that we know to have the fewest number of cars around. As for the month, the highest taxi speeds are in January, and it gradually declines until the average taxi speeds hit their minimum in June. This seems somewhat counter-intuitive, since one would think that taxi speeds would be lower during the colder months where there are snow and other adverse weather conditions on the road.

4.3 Clustering

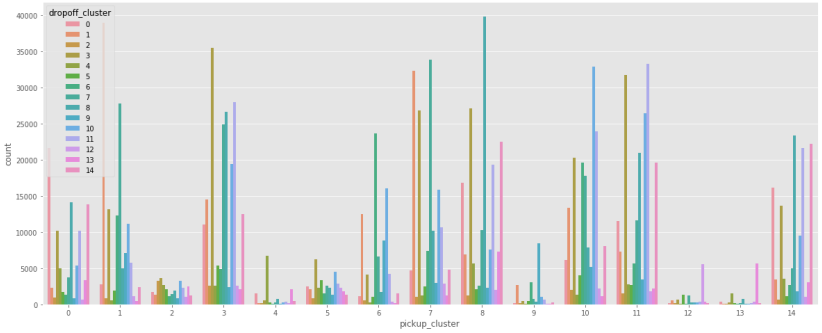
Using the same cleaned data set, KMeans Clustering algorithm was used to cluster the taxi rides taken within NYC based on geolocation coordinates into 15 clusters.



The heatmap can be interpreted as following: the row labels indicate the pickup_cluster labels. the column labels indicate the corresponding dropoff_cluster label. The values are the value counts of the taxi ride instances that the pickup_cluster label resulted in the corresponding dropoff_cluster.
For example, 21673 rides were picked up at cluster 0 and dropped off at cluster



0 (top left corner of the table). Then 2734 rides were picked up at cluster 1 and dropped off at cluster 0 (the cell immediately below the top right corner of the table). The table follows such format etc.
The heatmap version of the table is a more visually intuitive way of seeing which clusters are frequently traversed between via taxis in NYC. Obviously, there is a lot of taxi traffic between solely within cluster 8, amounting to more than 50000 rides. The more "purple" the cell, the more rides there are between the two corresponding pickup and drop-off clusters.



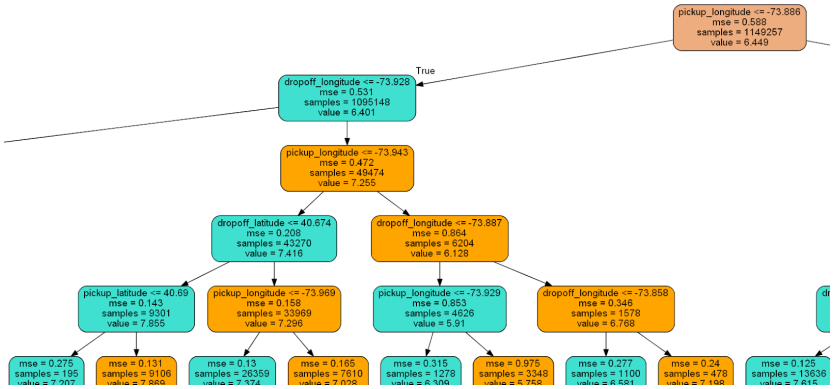
In general, not many people seem to be getting picked up by taxis in clusters 2,4,5,9,12, and 13. These clusters all correspond to the geographic locations outside of the island of Manhattan. It information also matches up with the density of the scatter plots shown on the cluster labeled plots, as there are significantly less densities of scatter points outside of Manhattan.

4.4 Decision Tree Boundaries - Classification

Using the same cleaned data set, it is possible to use the Decision Tree Classifier to identify how long a taxi trip duration will take. First, the log trip duration was converted into 6 bins representing the following about their original log trip duration values.

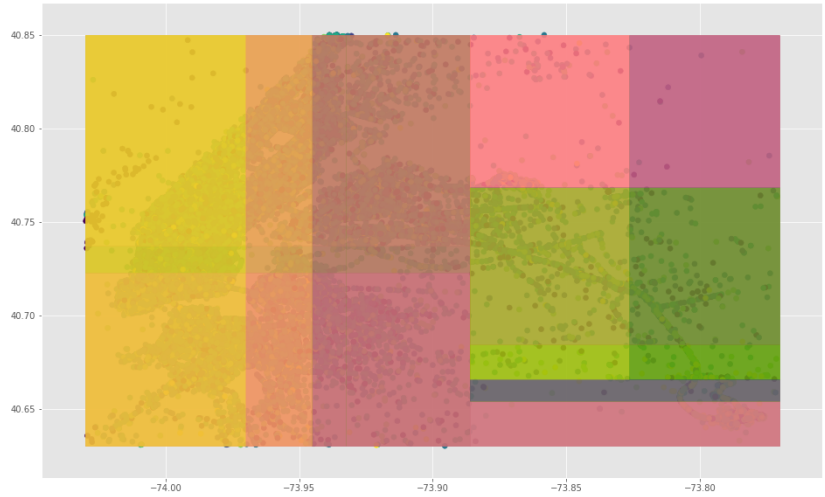
- 0 : less than mean - 2 std
- 1 : between mean - 2 std and mean - 1 std
- 2 : between mean -1 std and mean
- 3 : between mean and mean + 1 std
- 4 : between mean + 1 std and mean + 2 std
- 5 : more than mean + 2 std

Only the geolocation coordinate data columns were used as the predictor variables and the log trip duration bins were the target variables. Setting the max_depth to 5 such that our tree will be visually interpretable, we get the following results (cropped to see more specific splits of the leaves).



These results can then be overlayed onto our original scatter plot of data, and be visualized as a 2D decision boundary map.

Upon viewing the 2D plots, we visually confirm that the Decision Trees as a classifier to using geolocation data as predictor variables for the log_trip_duration bins doesn't work too well. This is probably because as we've seen in the clustering models, there are a variety of trips taken throughout each cluster. In many of the clusters, taxi trips tend to stay within the same cluster, which means that the trip distance is short. However, given that this is Manhattan, that does not exactly translate to shorter trip duration either, because of traffic and public events. When we observe the plot above, we see that the majority of the map is either yellow or magenta/pink. This would suggest that many of our geolocational data point towards the log_trip_duration taking around the mean amount of time. We know that yellow is slightly above the mean duration and



we can see that it intuitively covers the area known as downtown/midtown of Manhattan. However, this plot and model must be viewed with a grain of salt, since its RMLSE error (0.64) metric was relatively high compared to even just simple linear regression.

5 Models

5.1 Predicting Log Trip Duration

Linear Regression

Since we have a variety of data types within the dataset, we must wrangle them accordingly to get them ready for modelling.

- Get rid of irrelevant columns such as "id"
- Only use log_trip_duration instead of both the former and trip_duration
- Use one hot encoding (pandas get_dummies) for categorical variables such as "vendor_id", "passenger_count", various date columns (day, week, month), and cluster labels
- Randomly split the data, assign predictor and target variables and train!

```
model = LinearRegression()
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
```

```
print("R^2: {}".format(model.score(X_test, y_test)))
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("Root Mean Log Squared Error: {}".format(rmse))
```

```
R^2: 0.5415465972593874
Root Mean Log Squared Error: 0.5185110017503476
```

For the simple linear regression model, the Root Mean Log Squared Error is still quite high at approximately 0.5185. The highest positive valued coefficients of the linear regression model is the vendor ids, and several of the dropoff cluster

labels. Similarly, on the other end of the spectrum, the most negative valued coefficients of the linear regression model were some of the passenger count dummy variables, followed by the month dummy variables.

XGBoost: Gradient Boosted Trees for Regression

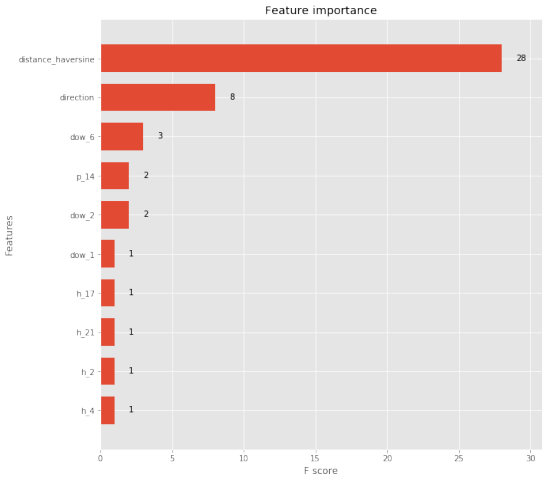
Using the same predictor variables along with the same target variable, we get the following results when using XGBoost with the following hyperparameters.

```
xgb_pars = {'min_child_weight': 1, 'eta': 0.3, 'colsample_bytree': 0.9,
            'max_depth': 7, 'subsample': 0.9, 'lambda': 1., 'nthread': -1,
            'booster' : 'gbtree', 'silent': 1, 'eval_metric': 'rmse', 'objective': 'reg:linear'}
model = xgb.train(xgb_pars, dtrain, 20, watchlist, early_stopping_rounds=2, maximize=False, verbose_eval=1)
print('Modeling RMSLE %.5f' % model.best_score)
```

```
[0]     train-rmse:4.21124      valid-rmse:4.21162
Multiple eval metrics have been passed: 'valid-rmse' will be used for early stopping.

Will train until valid-rmse hasn't improved in 2 rounds.
[1]     train-rmse:2.96461      valid-rmse:2.96482
[2]     train-rmse:2.09832      valid-rmse:2.09849
[3]     train-rmse:1.50041      valid-rmse:1.50043
[4]     train-rmse:1.09363      valid-rmse:1.09362
[5]     train-rmse:0.822409     valid-rmse:0.82236
[6]     train-rmse:0.648375     valid-rmse:0.648299
[7]     train-rmse:0.541825     valid-rmse:0.541714
[8]     train-rmse:0.478316     valid-rmse:0.478233
[9]     train-rmse:0.442395     valid-rmse:0.44239
[10]    train-rmse:0.422669     valid-rmse:0.422674
[11]    train-rmse:0.411192     valid-rmse:0.411129
[12]    train-rmse:0.402609     valid-rmse:0.402614
[13]    train-rmse:0.397606     valid-rmse:0.397821
[14]    train-rmse:0.39415      valid-rmse:0.394486
[15]    train-rmse:0.39092      valid-rmse:0.391286
[16]    train-rmse:0.388608     valid-rmse:0.389039
[17]    train-rmse:0.386758     valid-rmse:0.387339
[18]    train-rmse:0.385137     valid-rmse:0.385821
[19]    train-rmse:0.383545     valid-rmse:0.384294
Modeling RMSLE 0.38429
```

The XGBoost model was able to get the Root Mean Log Squared Error of the test set down to 0.384, a considerable improvement from the simple linear regression model. We can see that the model performs as well as it did on the training set as it did on the test set (randomly split set, same as above for the lin-reg model).



The feature importance plot shows which features have the greatest effect on trip duration by plotting the number of times each feature is split on across all boosting rounds in the model, illustrated as a horizontal bar plot. Since there were 88 features, it would be difficult to see all of the feature importance,

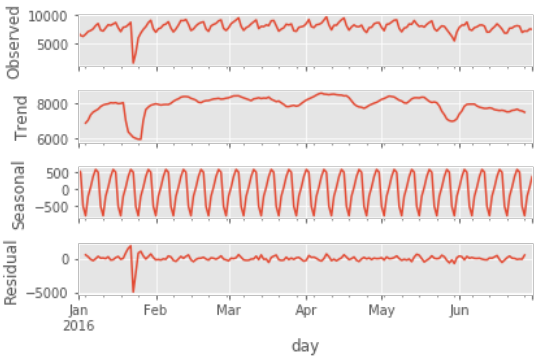
so the above plot shows the top 10 features (arbitrarily determined number). Furthermore, we see that direction, distance, day of week variables, and pickup cluster variables are the biggest factors in determining trip duration for taxis in NYC.

5.2 Time Series Forecasting

ARIMA: AR, I, MA

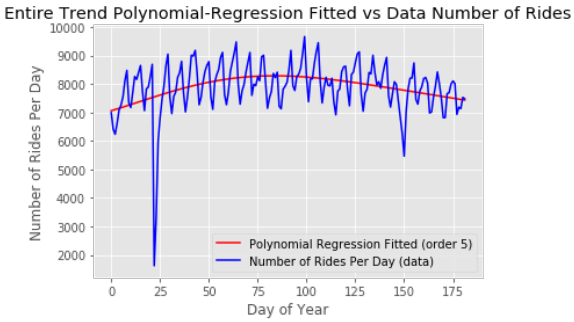
- AR or Autoregression (p): A regression model that uses the dependent relationship between the current observation and observations over a previous period
- I or Integrated (d): Differencing of observations (subtracting an observation from an observation at the previous time step) to make the time series stationary
- MA or Moving Average (q): Using past errors to predict the current value. To put it simply if the series is stationary then it will have a constant mean, however at any point in time it may not actually be at the mean. The distance from the mean is called the error

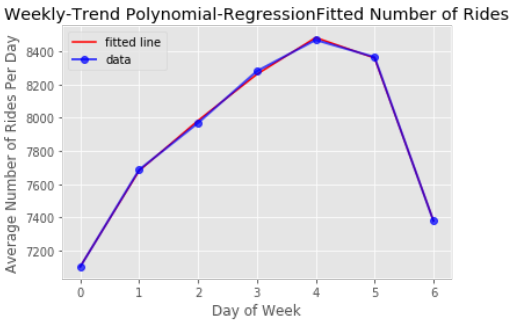
Other than the anomalous dip in the number of rides on January 23rd (which we know from EDA to be caused by inclement weather conditions) we can see a strong weekly seasonal component existing in the time series. We can use the statsmodel.tsa.seasonal modules seasonal_decompose to look at the error-trend-seasonality graph to further validate this claim.

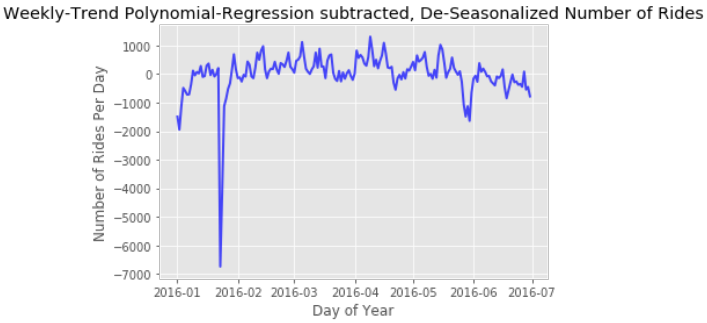


Since the weekly seasonality of our time series has been identified, it can be modeled. The model of seasonality can be removed from the time series. This process is called seasonal adjustment. A time series were the seasonal component has been removed is called seasonal stationary. One of the assumptions of doing ARIMA modelling is that the time series is stationary - that it has constant mean, variance, and autocorrelation over time. This is why doing some sort transformation is usually the first step, because most time series data are inherently not stationary.
We can model the seasonal component directly as a sine wave over a generally fixed period and amplitude. We can choose a single week/month of data, or all of the data. We could also smooth the observations using a moving average

centered on each value. When using np.polyfit(), a consistent sine wave can usually be modeled by using order 4 or 5.







When we fit an order 5 polynomial fit over the entire dataset, we can see that it fails to accurately capture the weekly seasonal component of the data. Therefore we need to restructure the polynomial fit in terms of weekly time increments to better fit the seasonal component. We take the data set and group by the day of the week, and take the average value for each day of the week over the course of the 26 weeks represented by the data set. We then fit an order 5 polynomial on this single 1-week curve for the average number of rides for each day of the week.

We can take these fitted couts of the number of rides predicted by the polynomial fit, and create a dataframe that consists of the days of the week and the polynomial fit predicted values for the number of rides. We can then merge it into the original dataframe, obtaining what is a weekly trend subtracted, seasonal stationary time series.

The ACF and PACF are good diagnostic tools to obtain some more general in-

formation about the de-trended time series as well. We can use the statsmodel's SARIMAX function to model a seasonal ARIMA model on our dataset by prescribing a set of p,d,q variables. Calling the summary method on the fitted model then gives information about the fitted model including information criteria such as AIC and BIC. We will also take the first 90% of the dataset as our prediction values, and leave out the most recent 10% of the dataset as the prediction (test) values. We don't want to take a random split of training and test dataset because in this time series, we want to see how well we can predict our out of sample "future" values for ride demand. The SARIMAX model summary returns statistics back regarding the defined model. We can also see which p, d, q values would result in the lowest BIC information criterion score.

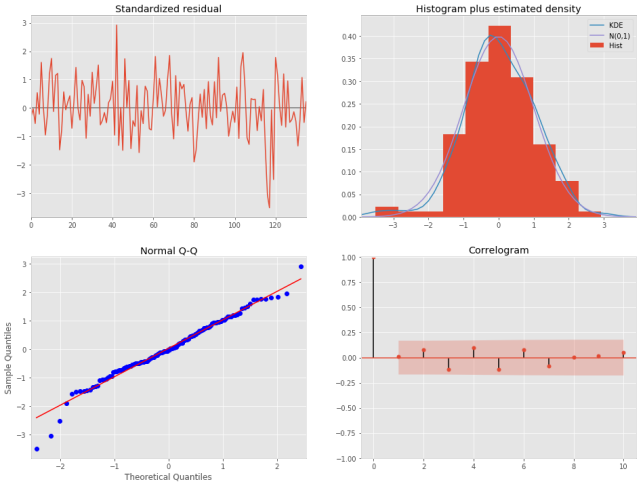
Statespace Model Results

Dep. Variable:	final_count	**No. Observations:**	136
Model:	SARIMAX(1, 0, 1)x(1, 0, 1, 7)	**Log Likelihood**	-969.518
Date:	Sun, 14 Jul 2019	**AIC**	1949.035
Time:	15:27:56	**BIC**	1963.599
Sample:	0	**HQIC**	1954.953
	- 136		
Covariance Type:	opg		

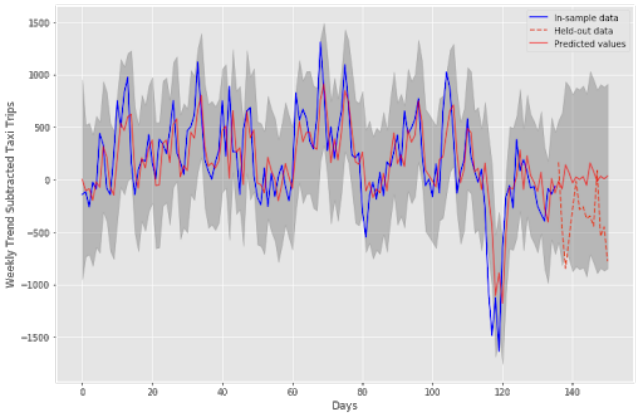
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.7255	0.061	11.803	0.000	0.605	0.846
ma.L1	0.0261	0.099	0.264	0.792	-0.167	0.219
ar.S.L7	0.9659	0.048	20.285	0.000	0.873	1.059
ma.S.L7	-0.8614	0.119	-7.220	0.000	-1.095	-0.628
sigma2	8.831e+04	1.03e+04	8.614	0.000	6.82e+04	1.08e+05

Ljung-Box (Q):	31.08	**Jarque-Bera (JB):**	7.36
Prob(Q):	0.84	**Prob(JB):**	0.03
Heteroskedasticity (H):	1.56	**Skew:**	-0.30
Prob(H) (two-sided):	0.14	**Kurtosis:**	3.97

Finally, using the plot_diagnostic method on the model defined with AR 1 and MA 1 order components, we can see features such as residuals, histogram, normality, and correlogram. Thee plot distributions of the residuals from the SARIMAX model shows us that the residuals look scattered around zero, and are roughly normally distributed when looking at the histogram and the linear fit. The correlation values mostly stay within the 95% confidence range as well. By taking out the first 25 days of values, we have effectively made our dataset more normally pleasant.

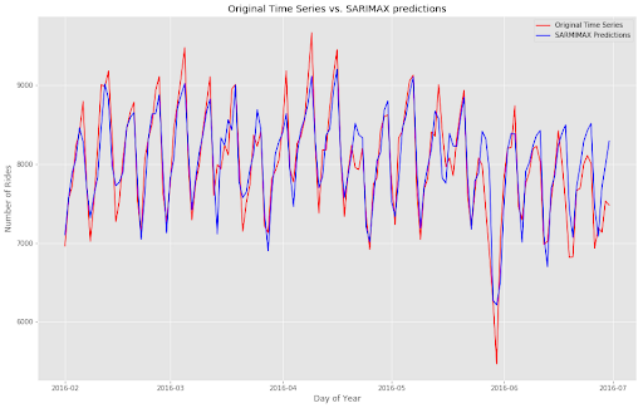


We can also plot the predicted results of the SARIMAX model along with the in-sample data and the held-out data. With our SARIMAX model, we achieve an in sample RMSE error value of 300, while we achieve an out of sample forecast RMSE of 490. While our forecast model was only 15 data samples, we get a promising error metric, when considering that there are more than 7 thousand trips on average given any day in NYC.



Now, by merging back into the original dataframe, we can reapply the weekly trends, and get the final transformed predictions on top of the original time series data. The re-transformed SARIMAX predictions show how the RMSE

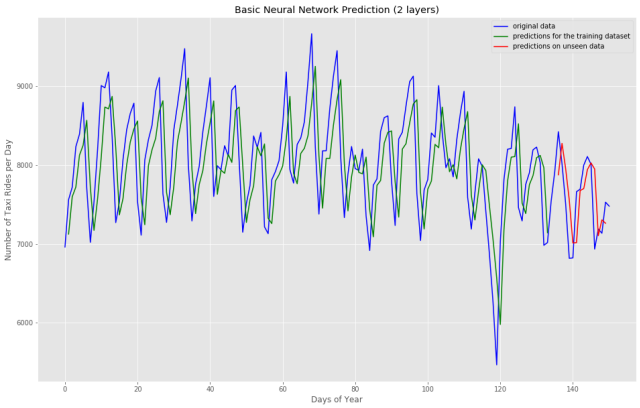
error of around 300 cars per day was not a bad measure at all. Our SARMIX model was able to fairly accurately predict the model and its weekly seasonality.



5.3 Neural Network Predictions

Basic 2 Layer Neural Network

In order to use neural networks as a regression predictor for time series datasets, we first had to preprocess the data such that there were two columns, first with the data, and second with the shifted "lag" data. The neural networks would be using the past values as a predictor for the future values. Furthermore, since the activation functions are usually sigmoid or hyperbolic tangent functions, the scale of the input data was also handed with MinMaxScaler to be in the range between 0 to 1. We again split the data into a 90/10 train-test split, such that the last 10% of the data set is the "future" set that we are forecasting for.
Without doing any further detrending or removing any sort of seasonality, we will use the neural networks to see how they perform in forecasting the time series. The following neural network had 2 layers with 50 neurons each, with the activation function of ReLu.



We get an in-sample train RMSE of 585, and out of sample test RMSE of 448 makes this simple two layer neural network with ReLu activation function the best predictive model above both the LSTM and the SARIMAX models in terms of forecasting for the out of sample data.

Long Short Term Memory Network

Long Short Term Memory networks are a type of recurrent neural network that is trained through back propagation and overcomes the vanishing gradient problem. The recurrent neural network makes use of memory cells and the output of the previous time period values to make predictions of the future.

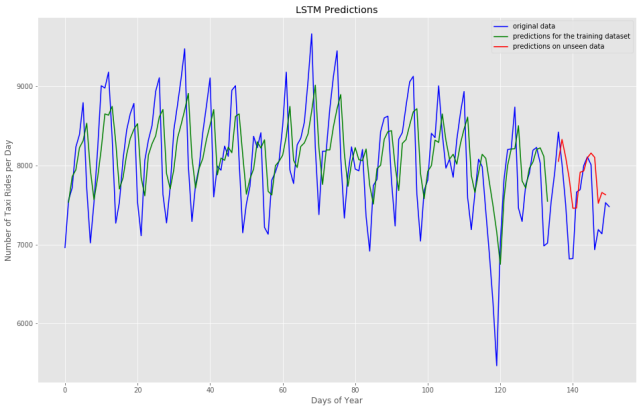
Instead of neurons LSTM networks have memory blocks that are connected through layers. A block contains gates that manage the block's state and output. A block operates on an input sequence and each gate within a block uses the sigmoid activation units to control whether they are triggered or not, making the change of state and addition of information flowing through the block conditional.
There are three types of gates in each unit:

- Forget Gate: decides what information to throw away from the block
- Input Gate: decides which values from the input to update the memory state
- Output Gate: decides what to output based on input and the memory of the block

Each unit is like a mini state machine where the gates of the units have weights that are learned during the training procedure. Phrasing the data frame as a regression problem, we can phrase the question as given the number of rides these past couple months, what will be the number of taxi rides in the next couple of days?

Again, we convert the single column of number of taxi rides into two columns, one for the value of count of taxi rides of the day, and the second column containing the number of taxi rides for the day afterwards (to be predicted)
The network has a visible layer with 1 input, a hidden layer with 50 LSTM blocks or neurons, and an output layer that makes a single value prediction. The default sigmoid activation function is used for the LSTM blocks. The network is trained for 40 epochs and a batch size of 1 is used.

This model achieved in-sample RMSE of 565 and out of sample (test) RMSE of 527, becoming the worst performing forecasting model out of the SARIMAX, basic NN, and LSTM. This may be because intuitively, the number of taxi rides do not inherently depend on the number of taxi rides from the day(s) before. It could also be that the dataset lacks more observations, since we are only testing on 15 observations based off of training on about 140 observations. For future work, more time series data should be gathered to see if the model performances for the three forecasting models discussed here would change significantly.



6 Conclusion

When predicting ride duration, we obtained the initial Root Mean Log Squared Error of the linear regression model of 0.518. This was further reduced to a RMLSE of 0.384 using XGBoost gradient boosted trees for regression method. We saw that the most important features in determining how long a taxi trip is going to take are the distance, direction, day of week variables, and pick up location variables.

In terms of ride demand, we saw that there was a constant weekly trend of taxi ride time series of New York City. Achieving seasonal stationarity was done by applying a regression fit to the average weekly trend values, and subtracting the corresponding days of week values from the original time series. This resulted in a good detrending of the time series. After making the time series stationary, the SARIMAX model was applied, and we were able to forecast up to 15 days worth of the next time series data, with an out of sample test RMSE of 490. Since there are upwards of 7 to 8 thousand rides each day, that error value seems reasonable.

Furthermore, we saw the unreasonable effectiveness of neural networks in predicting forecasting time series values, in that we achieved a better out of sample test RMSE result without even detrending the time series. A basic 2 layer neural network achieved an out of sample RMSE of 448, which was the best quantitative RMSE result we got out of the three models we tested out in this project.

Future work can be done in developing more variables to reduce the RMLSE of the log trip duration predictions. We can also collect more historical data for the time series so that we have more than 150 data observations. More work can be done with tuning the hyperparameters of the neural networks to increase performance as well.

References

[1] Jeffery Yau - https://www.youtube.com/watch?v=tJ-O3hk1vRw

[2] Aileen Nielsen - https://www.youtube.com/embed/zmfe2RaX-14?showinfo=0rel=0controls=1autoplay=1

[3] Beluga - https://www.kaggle.com/gaborfodor/from-eda-to-the-top-lb-0-367

[4] Jason Brownlee - https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/