**Predicting Where I need to be as an Uber Driver**

*Project Report for Springboard’s Data Science Intensive course*

Mathew Thomas

## **Introduction**

Uber is a worldwide transportation network company which allows users to request a ride using their smart phone. Not only is it convenient for consumers but also serves as a very flexible source of additional income for most drivers. Uber uses an algorithm to increase prices to “surge” price levels, responding rapidly to changes of supply and demand in the market, and to attract more drivers during times of increased rider demand. As a driver, it would be extremely beneficial to know what time of the day and the area in which I am probable to make the most amount of money. Since this service provides a lot of autonomy to the driver in terms of when he/she is willing to work, it is particular interest to find out which area they should be in to maximize their revenue based on time of day.

There can be multiple factors which decide where a Uber driver needs to be

1. Time of day
2. Special event/ Occasion
3. Weekends / Weekday
4. Points of Interest ( eg: Airport)
5. Other drivers availability (competition)

The objective of this project is to utilize these data points to determine where a Uber driver should be to maximize his earnings. A real time detection of traffic can be found out by analyzing twitter stream. The system fetches tweets according to certain criteria and will be used to model traffic for our forecast. The model would take into account the other factors mentioned above too. Another application of this model would be to predict traffic patterns in cities by analyzing social media content like twitter.

## 

## **Background:**

## Ridesharing services are quickly transforming the market for taxicabs and based on expense reports from business travelers in the first quarter of 2015, Certify says that all total paid car rides were through Uber in major markets across the US. That’s a steep rise from a mere 15% in the first quarter of 2014. In 2015 Uber doubled the number of active drivers on its platform as its grown. By end of 2015 the company had 327000 active drivers on the road in the US more than doubling the 160,000 that gave rides in 2014. This number has only increased in 2016 and a large number of drivers compete for a customer in the busy areas of a city. Since a major portion of the drivers in Uber drive the cab as a part time job to earn a little extra cash it would be highly beneficial to be able to predict areas in which there may be a high demand in the near future. With the rapid growth of online social media, more and more people are using twitter, Facebook etc. to communicate their mood, activities etc. which creates a huge repository containing information not accessible from conventional data sources. This information can be analyzed and a forecasting model can be created for the immediate future based on social media content for the last one hour. The main focus of this project is investigating how twitter data can be used as an external data source for improving near term prediction of demand for cabs. This model would have to be tweaked based on the region because of language semantics but it can be altered to serve all riders to better utilize their time throughout the world.

## **Data Sources:**

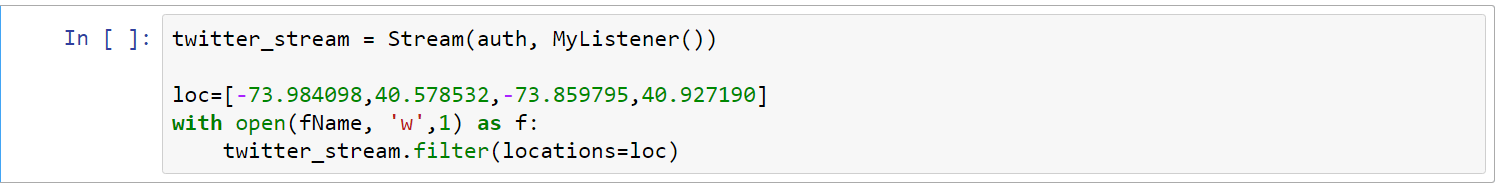
The two main data sources utilized her twitter and Uber. Both of these applications have public API’s which can be used to get data.

* **Twitter Data Stream**

Tweepy is an easy to use python library for accessing the twitter API. The API class provides access to the entire twitter RESTful API methods including the twitter streaming API which is used to download twitter messages in real time. The stream method was used as compared to just getting historic tweet data as I also needed to connect the data with estimate of uber price at that particular moment. So in the same method there is call to the Uber API to get the estimated price at that particular geo location. A total of 11178 tweets which had geo coordinates information were downloaded using the tweepy API from 2/14/2017 to 2/21/2017. Out of this some of them were removed as the uber API gave an error and did not provide an estimate for those coordinates which resulted in a total of 9981 tweets. The data had the following elements

|  |  |
| --- | --- |
| Data Column | Description |
| tweet\_id | Unique Twitter ID |
| tweet\_text | Tweet Text |
| lat\_origin | Latitude Coordinates |
| long\_origin | Longitude Coordinates |
| Tweet\_Time | Time of tweet |
| user\_id | Unique Id of User |

The only filter in the search query would be location which is specified using the binding box geo coordinates of the city which you want to analyze.



Tweeting with location is a geo-tagging feature in the twitter API and it has to be enabled by the user for the data to have geo coordinates in them when it is collected. So location is preferentially taken from the Geotagging API, but will fall back to their Twitter profile. So it will give all tweets that have geo coordinates in the area mentioned above or if it blank then it will also retrieve all tweets from users who are in the given location. This can be misleading for our analysis as a user may be belong to a particular city but may be tweeting from another location altogether. Hence I have considered only tweets which had geo coordinates in them to restrict our data set to tweets which are genuinely from our desired geographical location. An important caveat to be considered here is the density of useful data for developing our model here currently depends on the city that is chosen. Highly populated cities with a large proportion of twitter users yield better results as only a small percentage of twitter users have geo tagging feature enabled. So smaller cities would have a tweet with geo tagging enabled only every 2-3 minutes whereas bigger cities would have at least 10 tweets every second.

* **Uber API**

The get\_price\_estimate method returns an estimated price for each product given the coordinates of the starting and ending location. When surge is active for a particular product, its surge\_multiplier will be greater than 1, but the price estimate already factors in this multiplier.

|  |  |
| --- | --- |
| Data Column | Description |
| Uber\_product | Type of Uber Product |
| High Estimate | High Estimate of Price |
| Low Estimate | Low Estimate of Price |
| Surge Multiplier | Surge Multiplier |
| Request\_Time | Time of Request |

In order to get the coordinates of the destination, an algorithm is used which when given a latitude and longitude pair and a vector translation in meters, it returns a new latitude-longitude pair which is essentially an offset in meters from the original coordinates.

//Earth’s radius, sphere

R=6378137

//offsets in meters

dn = 100

de = 100

//Coordinate offsets in radians

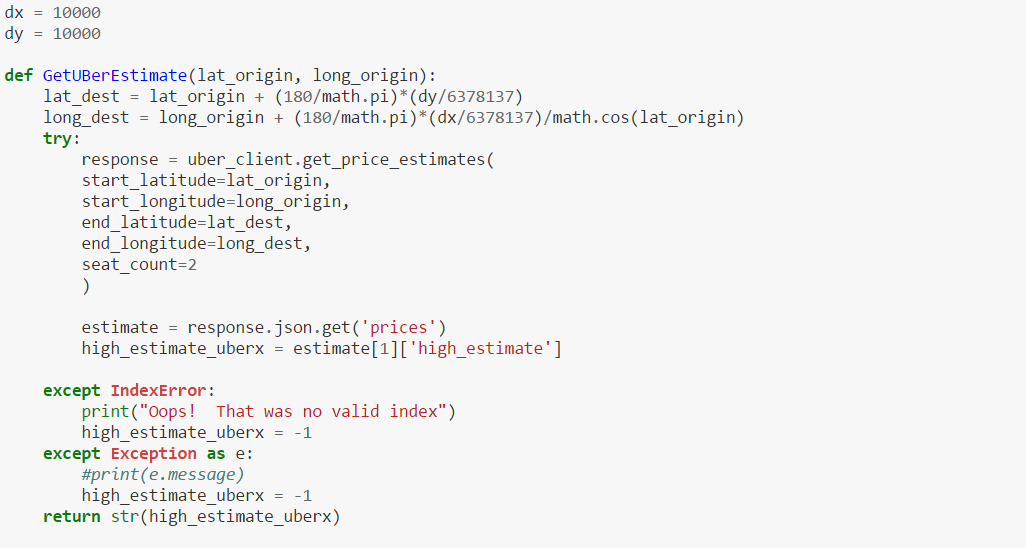
dLat = dn/R

dLon = de/(R\*Cos(Pi\*lat/180))

//OffsetPosition, decimal degrees

latO = lat + dLat \* 180/Pi

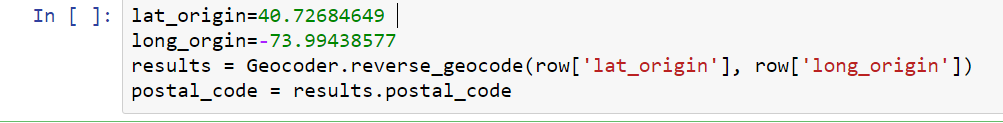
lonO = lon + dLon \* 180/Pi



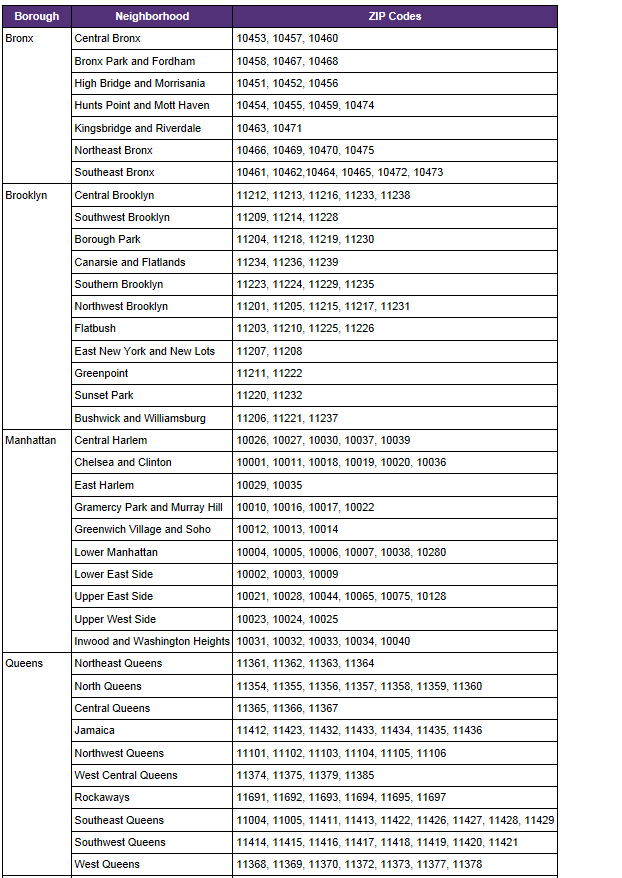
While using the API to retrieve information it is important to observe the rate at which it is being polled as there is a limit on the rate at which this information can be retrieved. A wait has to be introduced in the data retrieval mechanism to avoid crossing the threshold. Also there are certain coordinates for which the method returns an invalid coordinates error and they need to be handled accordingly.

* **Additional Data Sources**

From the data collected from twitter, we can get the geo coordinates for each location. Since these geo coordinates are too granular, it is difficult to do any kind of relevant analysis on them. The first step is to use Google maps API to get the postal code from each location



The zip code is then mapped with publicly available information ([NY\_Postal\_Code\_Info](https://www.health.ny.gov/statistics/cancer/registry/appendix/neighborhoods.htm)) to assign a neighborhood to the element. Since neighborhoods are broader they can be considered to provide some sort of correlation between the neighborhood and demand.

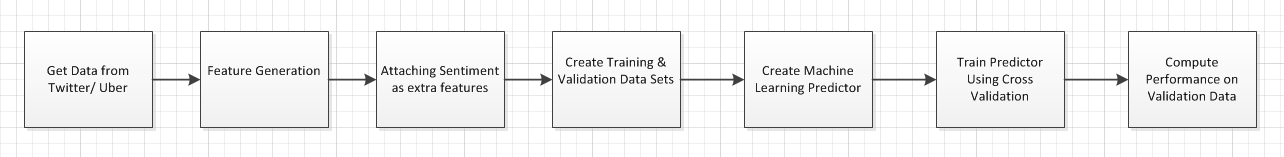


* **Removing Outliers:**

In order to get a better estimate, it is important to remove outliers from our data and make sure that certain data points do not skew our prediction. On average our uber estimate price is between $30 to $90. However in the data captured there are estimates with values as high as $300. However since the number of such cases is extremely low , we have removed all data points which had estimate above $150.

**Analysis:**

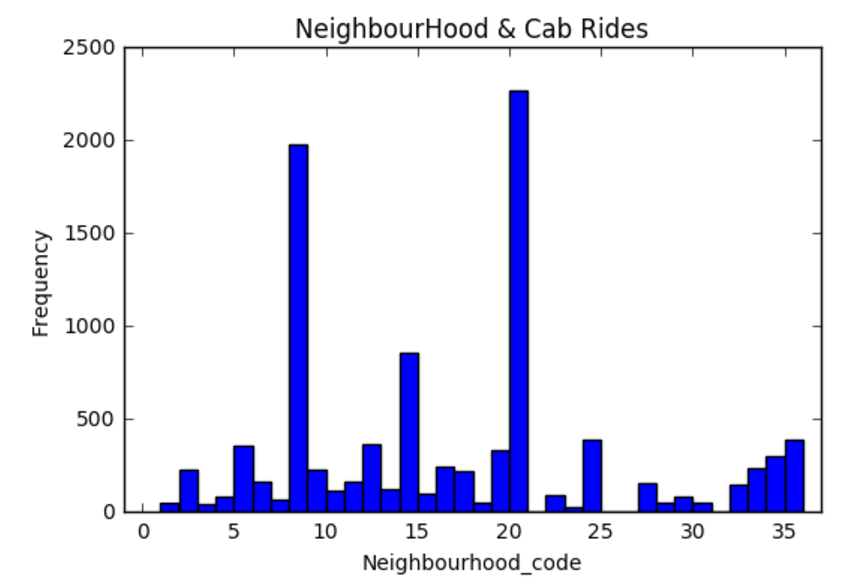
In order to perform our analysis we can break down our process into the following basic steps



The first step of this model is to get the data and prepare it for our analysis and it was explained in the section above. The next step is to define and generate features which we think would have an effect on the estimated price of a cab ride. Intuitively one would assume that the most important factors affecting the price of an Uber ride would be the demand. Various factors can drive the demand of a cab at any given time.

* **Neighborhood**

The location of an area has significant impact on the demand for a cab. There are certain neighborhoods which busier and will have a higher proportion of people tweeting which can indicate that there are more users requesting a cab in that location. The histogram below in which the city is divided into 36 neighborhoods and each neighborhood is assigned a code shows that there are certain neighborhoods where the number of people tweeting is significantly higher than others.

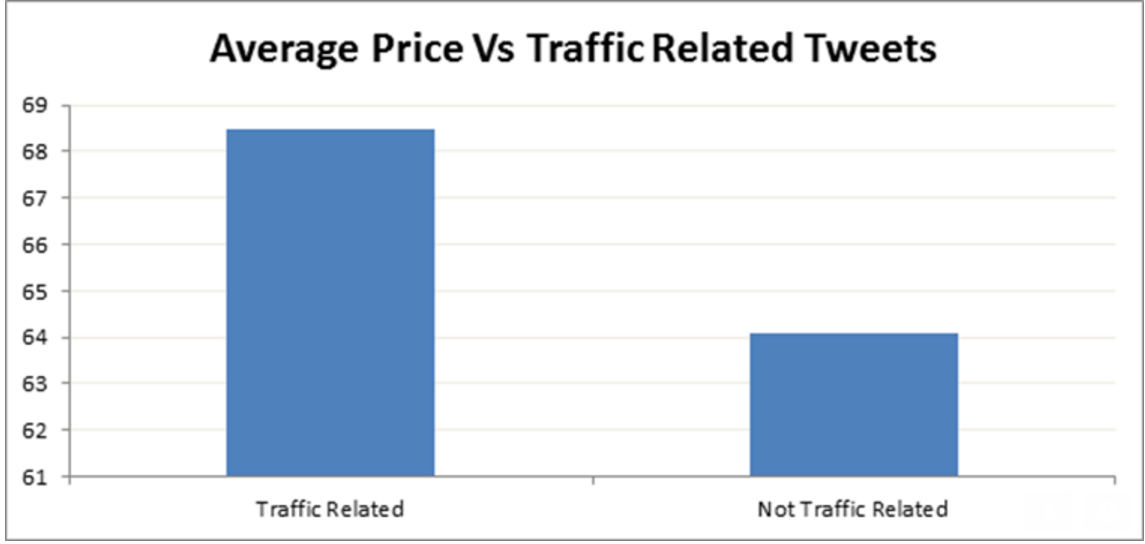


* **Bag of words**

The main challenge in this project is to analyze each individual tweet and to predict which of them can be identified as having a correlation with the demand for a cab at that particular point in time. The first part is to identify the most frequently occurring hashtags, words and bag of words. While doing this it is important to note that there has to be an array of items which need to be ignored for our analysis. This array is called the stop term and it includes punctuations, emoji’s, prepositions and city specific words. These can be excluded using regular expressions.

The initial analysis shows that these words have a higher correlation with high estimated prices and is chosen as our bag of words.

* Construction
* Road
* Incident
* Accident
* Traffic
* Event
* Concert
* Friday
* Saturday
* Sunday
* Ramp
* Terminal
* Show
* Game



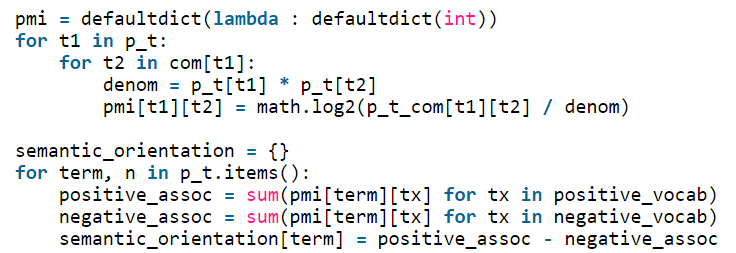
Apart from the occurrence of a word, it is important to analyze the sentiment of the tweet and it may have a correlation with demand at that particular point. For example the phrase “stuck in bad traffic as signal” vs “happy to get all green lights at traffic signal” both have the word traffic which is in our initial list of words to watch out for , but both have totally opposite meanings. The first phrase has a negative sentiment and can potentially mean more demand for cabs due to the high traffic in that area whereas the second phrase is a positive observation on the traffic situation at that time and may have the inverse effect on our prediction.

Based on the data we got two sets of vocabularies are defined:

|  |  |
| --- | --- |
| Positive Vocabulary | Negative Vocabulary |
| Good | Bad |
| Nice | Terrible |
| Great | Hate |
| Terrific | Crap |
| Super | Pissed |
| Awesome | Unhappy |
| Happy | Angry |
| Love | Useless |
| Like | Annoyed |

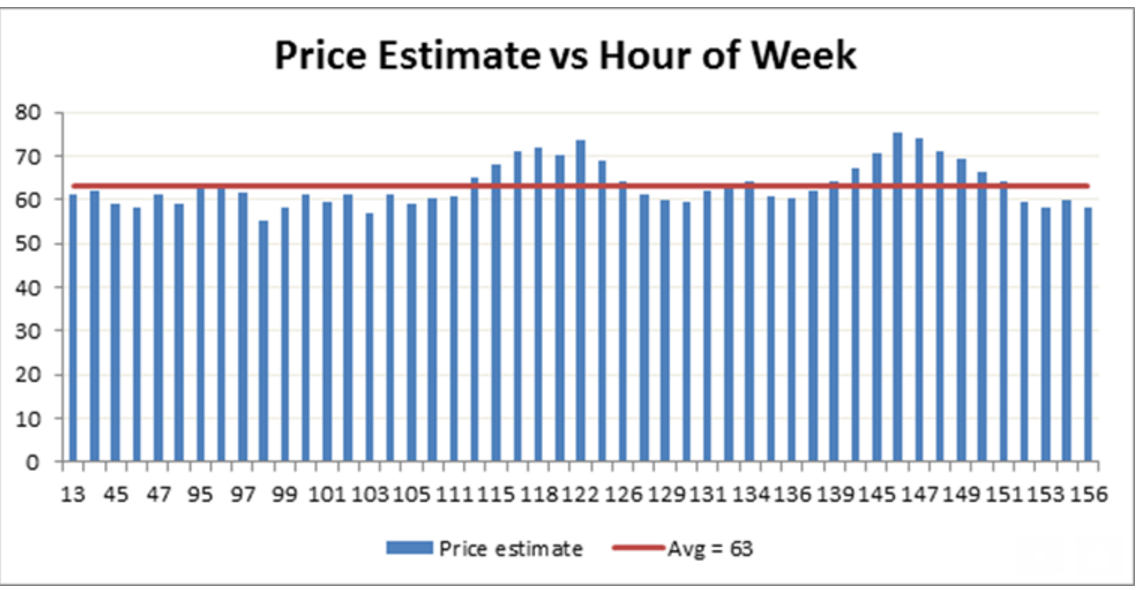
We define Semantic Orientation as the difference between its associations with positive and negative words and a measure of closeness called “Pointwise Mutual Information” is calculated as follows:

\mbox{PMI}(t_1, t_2) = \log\Bigl(\frac{P(t_1 \wedge t_2)}{P(t_1) \cdot P(t_2)}\Bigr)



* **Time**

The price estimate varies not only on the time but also on the day of week. For example 9pm on a friday night intuitively will have a higher demand than 9pm on Tuesday night whereas 7 am Saturday will have less demand than 7 am Monday. So to avoid collinearity a single independent variable must be used which is hour of week. Hour of week is calculated as (Day\_of\_week)\*24 + Hour\_of\_day. Our analysis shows that there are certain hours of week where the price estimate is significantly higher than the average. The chart below shows that hours 115 to 120 and 140 to 145 have clearly higher estimate as surge multiplier may have been active during those times.



## 

## **Regression Model:**

We have used a multivariable linear regression model to estimate the price based on our independent variables. In the least-squares model, the best-fitting line for the observed data is calculated by minimizing the sum of the squares of the vertical deviations from each data point to the line.

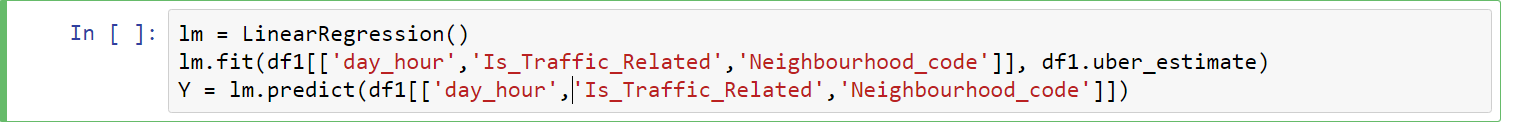
***y*i =** **0 +** **1*x*i1 +** **2*x*i2 + ...** **p*x*ip +** **i for *i* = 1,2, ... *n*.**

According to our prelimnary analysis the following 3 variables are considered independent and seem to have an impact on our estimate.

1. Neighborhood Code
2. TrafficRelatedTweet
3. Time of Week

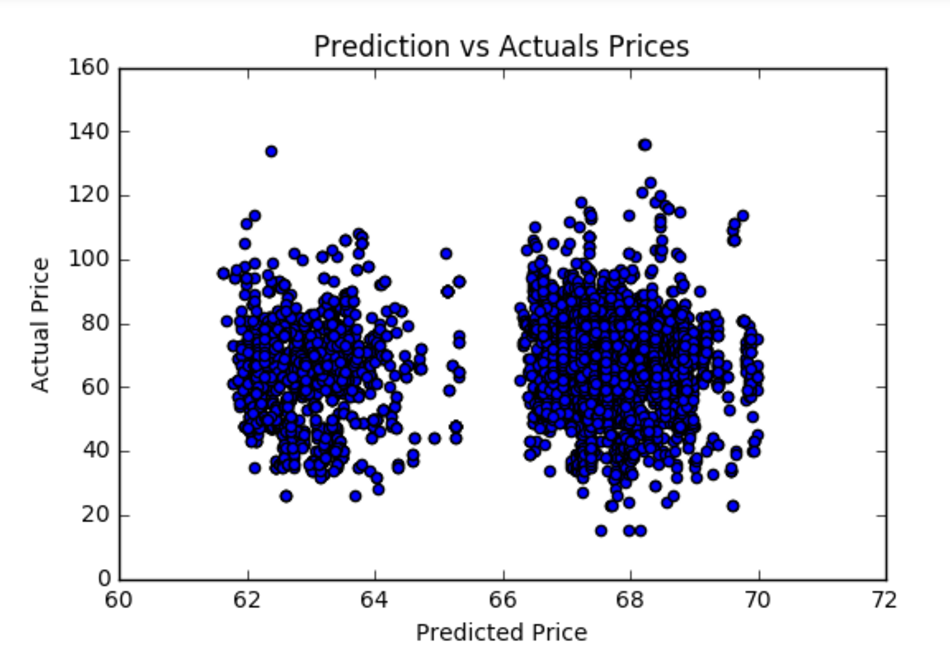
We started with the variable that had the highest correlation with the estimated price which was the Neighborhood code and calculated the adjusted R square to see which model is the best.

|  |  |  |
| --- | --- | --- |
| Model | Variables | Adj R-Square |
| Model 1 | Neighborhood | .002 |
| Model 1 | Traffic\_Related\_tweet | .013 |
| Model 3 | Hour\_of\_Week | .0091 |
| Model 4 | Neighborhood+ Hour\_of\_Week | .08 |
| Model 5 | Hour\_of\_Week + Traffic\_related\_tweet | .061 |
| Model 6 | Neighborhood+ Traffic\_related\_Tweet | .075 |
| Model 7 | Neighborhood+Hour\_of\_Week+traffic\_related\_tweet | .162 |



Model 7 which had all the 3 input variables had the best value of the adjusted R-square which is a statistical measure of how close the data is to the fitted regression line. The close the value is to 1 the better the model is.

**Further Analysis:**



The chart above shows the predicted output is always closer to the mean of $65 and is not able to predict high variations from the mean. This indicates that further analysis needs to be done to fine tune our feature vectors to better capture data points that vary a lot from the mean. The initial bag of words identified need to be expanded to find out more terms that may have a higher correlation with our desired outcome.

Some aspects of natural language are not captured by our sentiment analysis approach, more notably modifiers and negation: how do we deal with phrases like not bad or very good?

Another approach to consider is whether linear regression may not give a very accurate result as the independent variables may not have a linear relationship with the output. To solve for this other models like decision tree regression model, kernel regression or support vector classification model may be considered to better predict our outcome.

**Acknowledgements:**

I would like to than Hassan Kingravi for his extremely valuable guidance throughout this project in terms of not only how to approach such a problem but also provide crucial insights on making my model better.

**Addendum- Python Code:**

Get Twitter Data:

**from** **tweepy** **import** Stream

**from** **tweepy** **import** OAuthHandler

**import** **json**

**import** **jsonpickle**

**import** **pytz**

**import** **math**

**from** **pytz** **import** timezone

**from** **tweepy.streaming** **import** StreamListener

**from** **uber\_rides.session** **import** Session

**from** **uber\_rides.client** **import** UberRidesClient

auth = OAuthHandler(consumer\_key, consumer\_secret)

auth.set\_access\_token(access\_token, access\_secret)

*#sf coordinates*

*#loc= [-122.75,36.8,-121.75,37.8]*

*#clt coordinates*

*#loc=[-80.9138,35.1386,-80.7380,35.3539]*

*#NYC coordinates*

loc=[-73.984098,40.578532,-73.859795,40.927190]

uber\_session = Session(server\_token='-l3XEhlZRPidhD3pbwMAT44BAn6W-9ZdWqlV9EJU')

uber\_client = UberRidesClient(uber\_session)

dx = 10000

dy = 10000

**def** GetUBerEstimate(lat\_origin, long\_origin):

lat\_dest = lat\_origin + (180/math.pi)\*(dy/6378137)

long\_dest = long\_origin + (180/math.pi)\*(dx/6378137)/math.cos(lat\_origin)

**try**:

response = uber\_client.get\_price\_estimates(

start\_latitude=lat\_origin,

start\_longitude=long\_origin,

end\_latitude=lat\_dest,

end\_longitude=long\_dest,

seat\_count=2

)

estimate = response.json.get('prices')

high\_estimate\_uberx = estimate[1]['high\_estimate']

**except** **IndexError**:

print("Oops! That was no valid index")

high\_estimate\_uberx = -1

**except** **Exception** **as** e:

*#print(e.message)*

high\_estimate\_uberx = -1

**return** str(high\_estimate\_uberx)

eastern = timezone('US/Eastern')

utc = timezone('UTC')

fName = 'data/tweets\_streamed\_dataset.txt'

**class** **MyListener**(StreamListener):

**def** on\_status(self, status):

**if** (status.geo != **None**):

*#print(status.geo)*

tweet\_text=status.text

tweet\_id=jsonpickle.encode(status.id, unpicklable=**False**)

created\_at=jsonpickle.encode(status.created\_at, unpicklable=**False**)

utc\_created\_at = utc.localize(status.created\_at).astimezone(eastern)

tweet\_text = jsonpickle.encode(status.text, unpicklable=**False**)

tweet\_coordinates= jsonpickle.encode(status.geo['coordinates'],unpicklable=**False**)

lat\_origin=(tweet\_coordinates.split(",",1)[0].split("[",1)[1]).strip()

long\_origin=(tweet\_coordinates.split(",",1)[1].split("]",1)[0]).strip()

uber\_estimate = GetUBerEstimate(float(lat\_origin),float(long\_origin))

f.write(tweet\_id + '**\t**' + str(utc\_created\_at) + '**\t**' + tweet\_text +'**\t**' + tweet\_coordinates + '**\t**'+lat\_origin+'**\t**' +long\_origin+ '**\t**' + uber\_estimate + '**\n**')

f.flush()

**def** on\_error(self, status):

print(status)

**return** **True**

twitter\_stream = Stream(auth, MyListener())

*#twitter\_stream.filter(track=keyword\_list, languages=['en'])*

**with** open(fName, 'w',1) **as** f:

f.write("tweet\_id**\t**tweet\_time**\t**tweet\_text**\t**tweet\_coordinates**\t**lat\_origin**\t**long\_origin**\t**uber\_estimate**\n**")

twitter\_stream.filter(locations=loc)

Generate Bag of Words:

**import** **time**

**import** **re**

**import** **pytz**

**from** **pytz** **import** timezone

**from** **nltk.tokenize** **import** word\_tokenize

**from** **collections** **import** Counter

**from** **nltk.corpus** **import** stopwords

**import** **string**

**from** **nltk** **import** bigrams

**from** **nltk** **import** trigrams

**import** **json**

**import** **pandas** **as** **pd**

**from** **unicodedata** **import** normalize

punctuation = list(string.punctuation)

city\_specific\_stop\_words = ['York', 'Manhattan','NY','NewYork','NYC','The','A','#NewYork','#NYC','Brooklyn']

stop = stopwords.words('english') + punctuation + city\_specific\_stop\_words + ['RT', 'rt', 'via', 'I' , 'like','**\\**',',',':','in','…','**\ud83c**','**\ud83d**','°','u2026','ud83d','ud83c']

emoji\_pattern = re.compile("["

u"**\U0001F600**-**\U0001F64F**" *# emoticons*

u"**\U0001F300**-**\U0001F5FF**" *# symbols & pictographs*

u"**\U0001F680**-**\U0001F6FF**" *# transport & map symbols*

u"**\U0001F1E0**-**\U0001F1FF**" *# flags (iOS)*

"]+", flags=re.UNICODE)

emoticons\_str = r"""

(?:

[:=;] # Eyes

[oO\-]? # Nose (optional)

[D\)\]\(\]/\\OpP] # Mouth

)"""

regex\_str = [

emoticons\_str,

r'<[^>]+>', *# HTML tags*

r'(?:@[\w\_]+)', *# @-mentions*

r"(?:\#+[\w\_]+[\w\'\_\-]\*[\w\_]+)", *# hash-tags*

r'http[s]?://(?:[a-z]|[0-9]|[$-\_@.&amp;+]|[!\*\(\),]|(?:%[0-9a-f][0-9a-f]))+', *# URLs*

r'(?:(?:\d+,?)+(?:\.?\d+)?)', *# numbers*

r"(?:[a-z][a-z'\-\_]+[a-z])", *# words with - and '*

r'(?:[\w\_]+)', *# other words*

r'(?:\S)' *# anything else*

]

tokens\_re = re.compile(r'('+'|'.join(regex\_str)+')', re.VERBOSE | re.IGNORECASE)

emoticon\_re = re.compile(r'^'+emoticons\_str+'$', re.VERBOSE | re.IGNORECASE)

**def** tokenize(s):

**return** tokens\_re.findall(s)

**def** preprocess(s, lowercase=**False**):

tokens = tokenize(s)

**if** lowercase:

tokens = [token **if** emoticon\_re.search(token) **else** token.lower() **for** token **in** tokens]

**return** tokens

count\_all = Counter()

*#twitter\_dataset = pd.read\_table('tweets\_streamed\_dataset.txt', index\_col=None)*

twitter\_dataset = pd.read\_table('data/nyc\_streamed\_dataset.txt', index\_col=**None**)

**for** tweet **in** twitter\_dataset.tweet\_text:

*#tweet = normalize('NFKD', tweet).encode('ascii','ignore').decode('unicode\_escape')*

tweet = normalize('NFKD', tweet)

terms\_stop = [term **for** term **in** preprocess(tweet) **if** term **not** **in** stop **and** **not** term.startswith(('#', '@')) **and** **not** emoji\_pattern.search(term)]

*#terms\_bigram = bigrams(terms\_stop)*

terms\_trigram = trigrams(terms\_stop)

*#terms\_hash = [term for term in preprocess(tweet1) if term.startswith('#')]*

*#terms\_all = [term for term in preprocess(tweet) if term not in stop]*

count\_all.update(terms\_trigram)

*#count\_all.update(terms\_hash)*

*#count\_all.update(terms\_all)*

print(count\_all.most\_common(100))

Feature Analysis:

lm = LinearRegression()

lm.fit(df1[['day\_hour','Is\_weekend','Is\_Traffic\_Related','Neighbourhood\_code']], df1.uber\_estimate)

msePRC = np.mean((df1.uber\_estimate - lm.predict(df1[['day\_hour','Is\_weekend','Is\_Traffic\_Related','Neighbourhood\_code']])) \*\* 2)

print (msePRC)

print (lm.coef\_)

ESS = np.sum(lm.predict(df1[['day\_hour','Is\_weekend','Is\_Traffic\_Related','Neighbourhood\_code']]) - np.mean(df1.uber\_estimate)) \*\* 2 RSS = np.sum((df1.uber\_estimate - lm.predict(df1[['day\_hour','Is\_weekend','Is\_Traffic\_Related','Neighbourhood\_code']])) \*\* 2) R2 = ESS/(ESS+RSS)

print (R2)