TAXI DEMAND PREDICTION

Team Members:

1. Shyam Nair: smn387
2. Gaurang Gatlewar: gvg228
3. Vish Akella: vua201
4. Jayesh Mhatre: jkm437

GitHub Link: https://github.com/ShyamNair9/Taxi_Demand_Predictor

1. Business Understanding

For over a century, Medallion (Yellow) Cabs have been the lifeblood of transportation for New Yorkers. Every month, yellow cabs make an average of 300,000 trips, earning $5 million. Often, people looking for a cab don't get to hail one as the cabs are unavailable at that specific place and time. Among several factors that affect demand for cabs, scheduling of local events, weather conditions, recurring patterns, population density, etc. plays a significant role. Usually there is a disparity in the taxi demand and availability leading to dearth of cabs in some areas and customers in other areas. A cab driver must attempt to optimize his location to increase his sales. This will enable him/her to meet the peak demand and reduce the idle time by efficiently finding customers within the time and traffic constraints and the customers can also be served promptly.

We are attempting to help this situation by supplying cabs with information on locations where the chances of getting a customer is very high (more specifically, a place where an event is scheduled to take place on that day, the weather conditions, etc.). We have generated a regression model that predicts a range of cabs required at any place and time, based on the local events and weather conditions. This will in turn provide a cab driver the tool to solve this problem and improve his/her availability at the right place and at the right time, also benefitting the customers.

2. Data Understanding

For the prediction feeder models, we have used 3 primary categories of data - Taxi trip records, Weather Records and SeatGeek events. The table below indicates the data sources we have explored to decide on the final inputs for our models.

Category	Shape	Source	Extraction Method
Events	12500	https://api.seatgeek.com, http://api.eventful.com/	API Request
Weather	3000	https://www.wunderground.com	Website scraping
Taxi	9000000	https://www.nyc.gov	Open Source

Taxi Data

It contains approximately 10M rows of transactional data, each corresponding to a single cab trip. The cab location was provided in 265 discrete location IDs representing the entire NYC area. The taxi data contains features like pickup and drop-off points, start time, end time, fare, tip, number of passengers, etc.

Weather Data

The dataset contains temperatures and categorical information about the nature of the weather like rain, snow, sunny etc. The metrics were measured at specific locations. For our model, we used the information pertaining to NYC. The weather data was collected by scraping the Weather Underground website as it was the most reliable source providing all the historical data including features like temperature, precipitation, weather conditions which proved to be significant in demand prediction.

Event Data

Event data was obtained using multiple APIs like Facebook Graph API, SeatGeek API and Eventful API. The event data included features such as the location, popularity and score of the event, event type, artists and the statistical data. Lastly, cab data was obtained from the Open data source provided by NYC Cab website and was acquired by merging the yellow and green cab data. The data extracted lies in the period July 2017 - November 2017.

Since the data was acquired individually from independent sources, the raw data required extensive cleaning and modifications before merging into a single dataset.

3. Data Preparation

The integration process of the 3 datasets involved the following steps.

3.1. Taxi Data

- Data was aggregated based on a 1hr time frame and location to get the Taxi count
- The event location was provided in the form of contiguous latitude-longitude coordinates. In order to reconcile the two sets of information, the Event locations were mapped to the cab Location IDs.

- To consolidate the spatial information, we performed clustering on the locations ids to generate a list of 100 new location ids using the kNN clustering algorithm. This New ID column was used in the prediction model.
- A high percentage of the taxi usage follows a fixed weekly pattern governed by office timings, peak hours, etc. Hence, we segregated the datetime field to get the Month, Day, Weekday and Hour of the data and was used as a major factor in prediction.

3.2. Weather Conditions

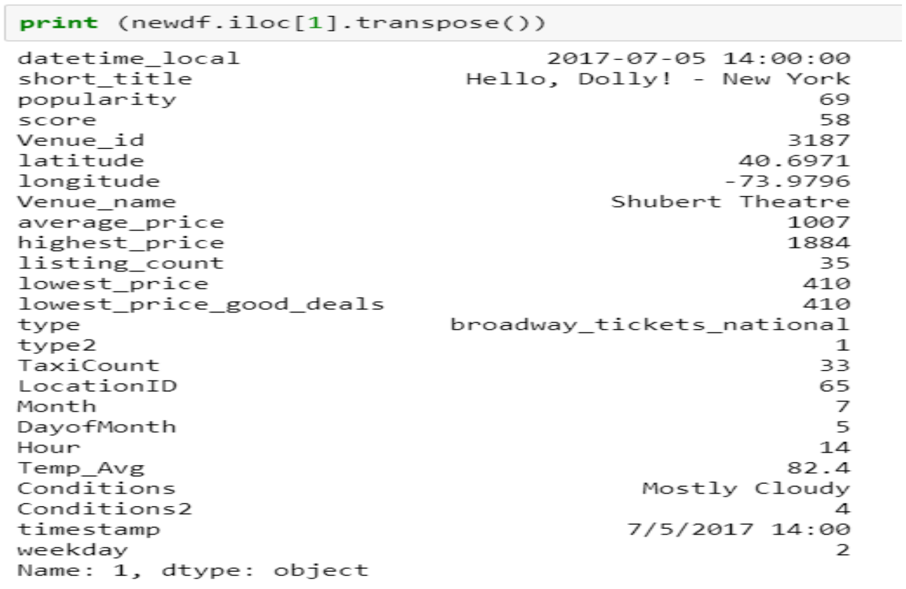
- The weather conditions contained a lot of redundant categories such as Rain, little rain and squalls, overcast and cloudy, fog and haze etc. which were grouped into 6 broad categories. We mapped the categorical variables to integers to obtain the final training data.

3.3 Event Data

- The event data set consisted of a parameter 'Event type' which mentioned the type of the event such as Broadway Ticket, Classical Opera, Classical Dance, Soccer, Football, Baseball (Around 16 types) which were classified into 6 broad categories such as Broadway shows, Comedy, Sports, Concert, Theatre etc. These were further replaced by integers.
- Event Data with missing values of end times were replaced by event start time + 3 hours as it was observed that majority of the events had a duration of 3 hours.

3.4. Combining Different Datasets:

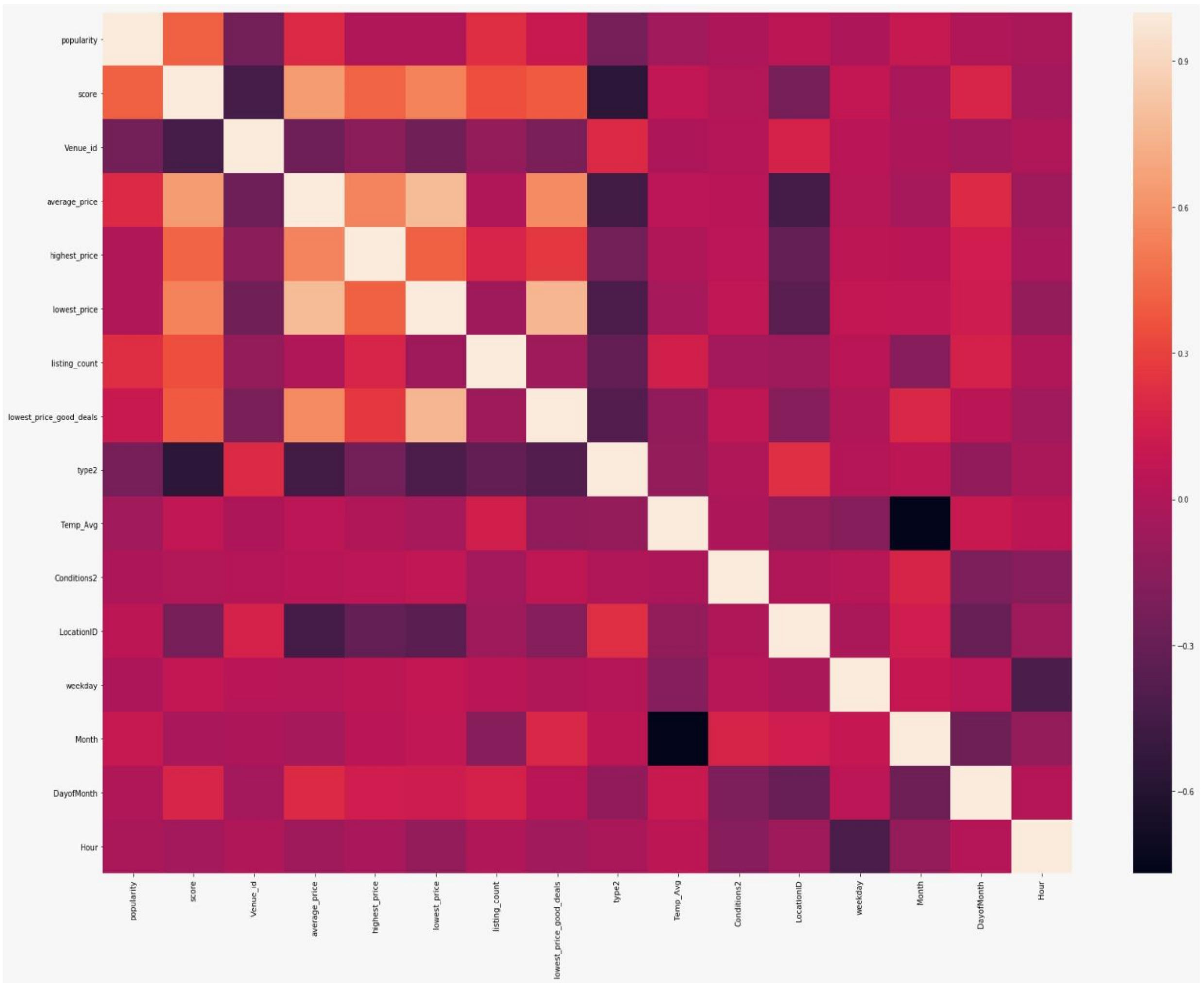
- The event and cab datasets were merged based on datetime (day, month and hour) and Location ID which was further merged with the weather dataset on the datetime column using MongoDB.
- All the categorical values and non-integral values were replaced by integers to be used for the prediction model and the empty, NaN and duplicate values were dropped using functions provided by the Pandas DataFrame module. Also the features such as score and popularity were re-scaled for maximizing the prediction accuracy.
- The target variable for this model is the number of cabs expected in the vicinity of an event depending on the weather conditions as well. We are estimating the demand, which is a contiguous variable, in a predetermined locality with a deviation of 10 cabs on either side.
- A single instance of the dataset is as follows:



```
print (newdf.iloc[1].transpose())
```

```
datetime_local                     2017-07-05 14:00:00
short_title                   Hello, Dolly! - New York
popularity                                          69
score                                               58
Venue_id                                          3187
latitude                                       40.6971
longitude                                     -73.9796
Venue_name                             Shubert Theatre
average_price                                     1007
highest_price                                     1884
listing_count                                       35
lowest_price                                       410
lowest_price_good_deals                            410
type                         broadway_tickets_national
type2                                                1
TaxiCount                                           33
LocationID                                          65
Month                                                7
DayofMonth                                           5
Hour                                                14
Temp_Avg                                          82.4
Conditions                               Mostly Cloudy
Conditions2                                          4
timestamp                               7/5/2017 14:00
weekday                                              2
Name: 1, dtype: object
```

- The correlation matrix of all the features is shown below based on which latitude and longitude were not considered in generating the training set because Location ID had a greater impact in prediction.



4. Modeling and Evaluation

4.1. Model Choices

Our primary objective is to estimate the number of taxis available at a given Venue where an event was being held. To achieve this, we used an approach based on classification where the finite number (positive integer values) of taxi availability was trained using the NYC Taxi frequency. Though the number of available taxis could theoretically be infinity, our observation from the historical data is that this would be a countably small number that can be estimated using a classifier. Given the nature of the prediction problem and the availability of pre-existing Taxi data, Supervised algorithms offer an estimation approach which is easily understood and can be improved upon, iteratively.

We trained our datasets on three Supervised Classification approaches to pick the best performing algorithm -

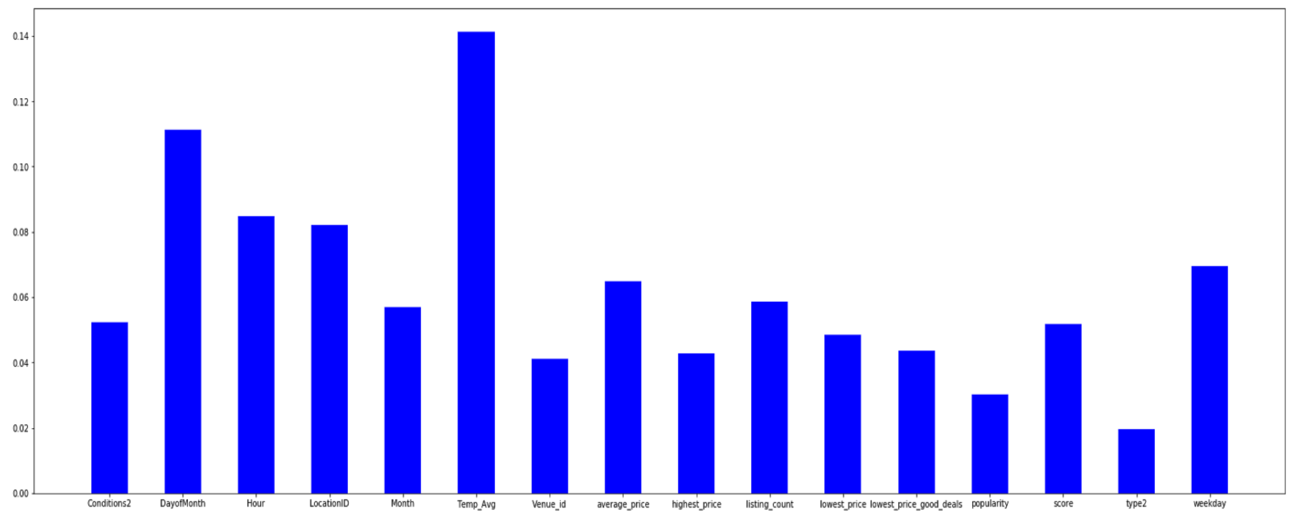
- **Logistic Regression** - One-Vs-Rest, linear classification plane
- **Decision Trees** - non-linear hyper planes, but prone to overfitting
- **Random Forests** - an ensemble of Decision Trees

We have a supervised learning problem with a continuous target variable and hence, we have to use a Regression model instead of a classification model. For this particular problem, Logistic Regression, Bernoulli Naive Bayes, Random forest, Gradient Boosting and Support Vector Machines usually perform well.

4.2. Performance Analysis

Our analysis of the accuracy indicates that, among the alternatives considered, a Random Forest model has the highest accuracy and least Mean squared error. Details of this are submitted in the notebook.

We tuned each of the three classification models using **GridSearchCV** to optimise the parameters. On the optimal model, we evaluated the performance of each parameter and their contribution. The most important parameter for demand prediction is 'Temp_Avg' followed by 'DayOfMonth'.



4.3. Evaluation Framework

In estimating the Classification's usefulness, we considered the following priorities -

- value numerical closeness of the predicted to the actual value. This would directly translate into usefulness as a directional pointer to the taxi driver.

- minor differences in the number of cars predicted to the actual does not carry a huge penalty
- the model leaves out some sources of variance like Traffic conditions, public transportation access to event areas, etc. Some of these are hard to measure and the others can be added incrementally, at a later stage

Considering the above reasons, we opted for accuracy as the measure of model performance. Because of minor penalty, precision (reproducibility) is not critical for this model. Also, the intended purpose of this model is to be a useful support tool, not a high criticality feature.

4.4. Analysis of Algorithms

We are trying to predict the taxi demand in such a way as to maximise the number of instances where the demand range correctly matches the actual value as well as trying to optimize the total number of cabs required over the total area. Hence, we require the model to have high precision as well as very low Mean Square error. Thus, we have taken both of these features into account while deciding on the final prediction model.

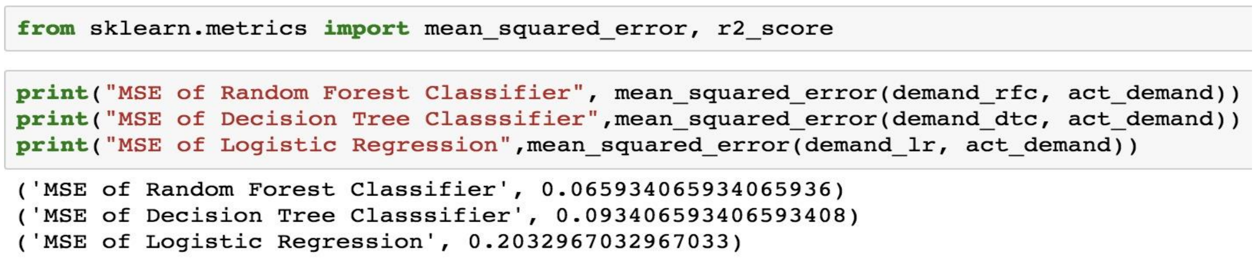
As we had limited number of parameters, we performed hyper-tuning on all 3 algorithms before choosing the baseline model. The table shown below describes the number of times the model correctly predicts the count of taxis for the test instances as well as the cross-validated score of the model generated by choosing the optimal parameters determined by GridSearchCV. This also ensures prevents the model from overfitting.

Classifier	Original Precision	Cross-Validated precision
Logistic Regression	0.6978	0.7967
Decision tree	0.9176	0.9065 (maybe overfitting)
Random Forest	0.9286	0.9341

As we can see from the table, Random Forest has the highest precision of all the 3 models.

Mean Squared Error (MSE):

We have calculated the mean squared error for all 3 models and the comparison can be seen in the code snippet and graphs below:

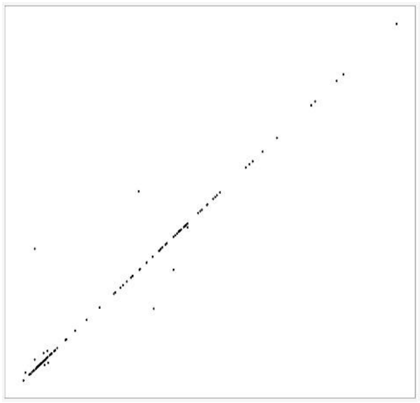


```
from sklearn.metrics import mean_squared_error, r2_score
```

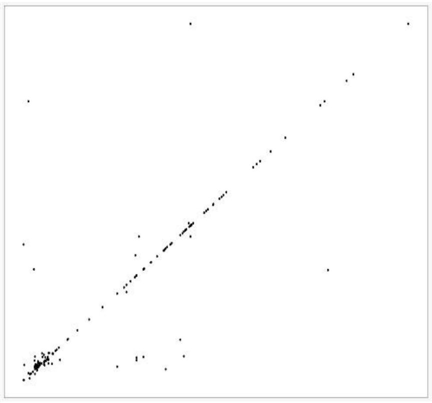
```
print("MSE of Random Forest Classifier", mean_squared_error(demand_rfc, act_demand))
print("MSE of Decision Tree Classsifier",mean_squared_error(demand_dtc, act_demand))
print("MSE of Logistic Regression",mean_squared_error(demand_lr, act_demand))
```

```
('MSE of Random Forest Classifier', 0.065934065934065936)
('MSE of Decision Tree Classsifier', 0.093406593406593408)
('MSE of Logistic Regression', 0.2032967032967033)
```

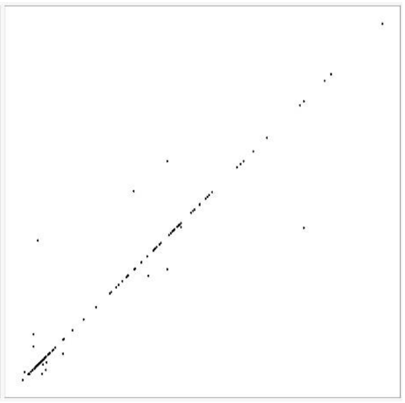
Random Forest



Logistic Regression



Decision Tree

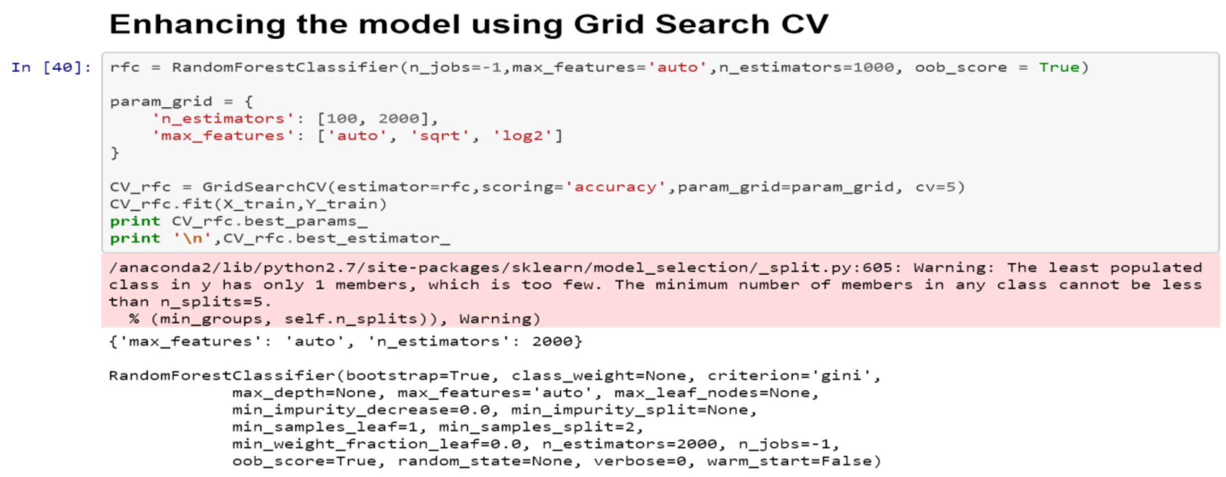


As we can see from the values, Random Forest has the minimum MSE of all the 3.

We can see that Random Forest algorithm outperforms the other algorithms in both precision and MSE. Hence, we have chosen Random Forest as our final baseline model for demand prediction.

4.5 Optimization on Baseline model

Further, we tuned the best performing Random Forest Model using GridSearchCV to optimise the parameters. The optimal parameters and the best estimators provided by GridSearchCV are as shown below:


Enhancing the model using Grid Search CV

```
In [40]: rfc = RandomForestClassifier(n_jobs=-1,max_features='auto',n_estimators=1000, oob_score = True)

         param_grid = {
             'n_estimators': [100, 2000],
             'max_features': ['auto', 'sqrt', 'log2']
         }

         CV_rfc = GridSearchCV(estimator=rfc,scoring='accuracy',param_grid=param_grid, cv=5)
         CV_rfc.fit(X_train,Y_train)
         print CV_rfc.best_params_
         print '\n',CV_rfc.best_estimator_
```

```
/anaconda2/lib/python2.7/site-packages/sklearn/model_selection/_split.py:605: Warning: The least populated
class in y has only 1 members, which is too few. The minimum number of members in any class cannot be less
than n_splits=5.
  % (min_groups, self.n_splits)), Warning)
{'max_features': 'auto', 'n_estimators': 2000}

RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=2000, n_jobs=-1,
            oob_score=True, random_state=None, verbose=0, warm_start=False)
```

The accuracy of Random forest model was improved from 0.9286 to 0.9341 after hyper-tuning using GridSearchCV.

4.6 Business Impact

The primary insight from the model is the number of taxis available at a particular venue conditional on parameters pertaining to the event, location and weather. There are two scenarios where this can be used -

- WHOLESALE: Allow a taxi operator to plan the deployment of the fleet optimally on days when events are scheduled
- RETAIL: Allow an individual driver to position himself so that he improves his chances of being hailed, in person or on apps like Uber and Lyft

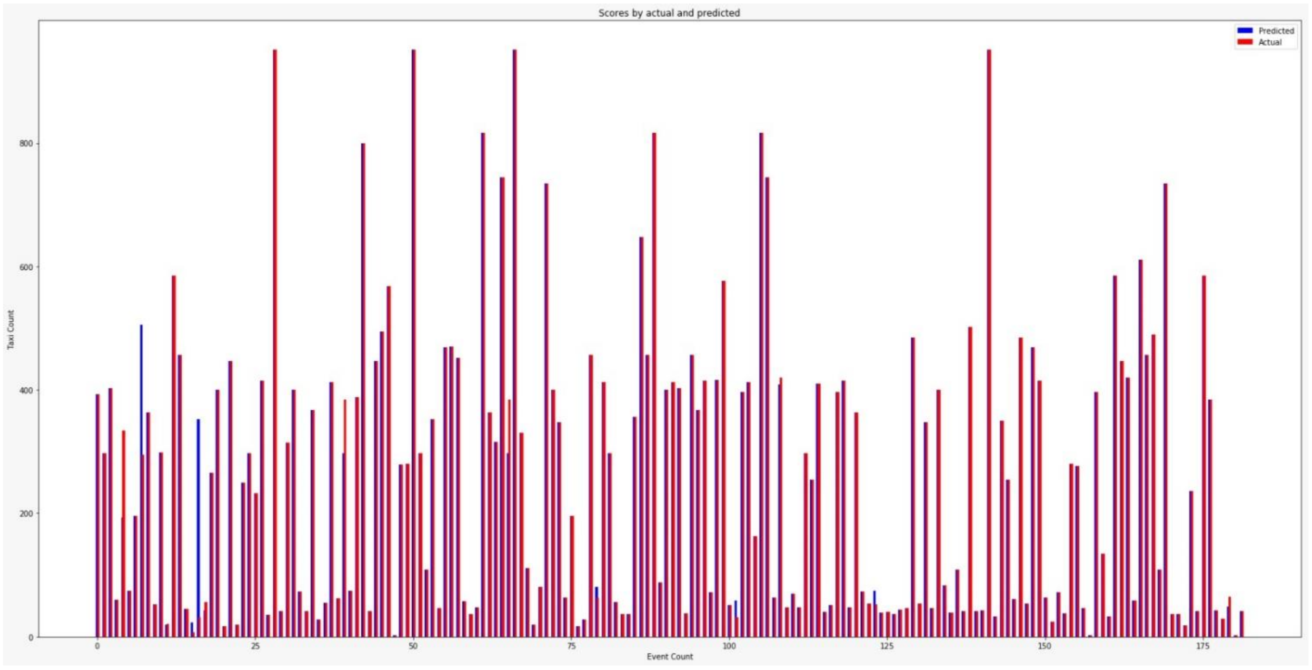
The mode of delivery would be through a browser based application for the wholesale clients. Integration with analytics like fueling locations, shift times, maintenance costs are important to these clients.

For the retail users, a mobile app with tight integration for traffic and resting stops would be more important. Also, a real time feedback of how quickly the demand is being met at a particular venue would help drivers avoid crowding at venues where demand has already been fulfilled.

4.7 Evaluation Metric

We quantified the distance between the actual and predicted Taxi demand using the mean squared error. We did this in order to penalize both positive and negative errors equally. Also, we want to penalize large outliers. This is important as large deviations would pose a reputational risk to the product.

Below is a chart displaying the actual vs. Predicted performance for Random Forest classifier.



5. Deployment

The current prediction model is based only on data gathered in 4 months (July - November). The model can perform better with larger datasets updated in real time. The model heavily relies on the periodic trends of the change in traffic (‘Hour’ and ‘DayofMonth’ are important factors in demand prediction) and can perform better if we have at least one year’s worth of data. To implement such a model, we need Cloud/AWS services and big data technologies such as Hadoop, MapReduce and Spark.

One of the major risks of this model is that, the cabs would overcrowd the high demand areas and desert the low demand areas further aggravating the problem that this solution intended to mitigate. Consider a scenario where the taxi demand predicted by our model for a particular event is 20 and 50 cab drivers use our model and go to the event location. This can prove to be a great loss to the cab drivers. This problem can be solved by updating the location of the cabs in real time and managing the fleet centrally.

Another risk with the model is that it portrays the actual cabs hired rather than the demand for the cabs thus rendering it unable to measure demand correctly in some cases. In the cases where the cabs were not hired due to unavailability rather than demand, the model would further pile on the error increasing it exponentially with time. The model's limitation of prediction in such cases can be eliminated by assuming a dummy value and predicting the demand in a manner similar to that of Laplace smoothing.

6. Citations/References:

1. http://platform.seatgeek.com/ : Event data extraction

2. http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml : NYC Cab data

3. https://www.wunderground.com/weather/us/ny/new-york : Weather data source

4. http://scikit-learn.org/stable/ : Used for modelling and evaluation

5. https://pandas.pydata.org/pandas-docs/stable/ : Used for data preparation

6. https://developers.facebook.com/docs/graph-api/ : Event data source

7. http://api.eventful.com/ : Event data source

8. https://matplotlib.org/contents.html : Used for plotting graphs

9. https://stackoverflow.com/ : Debugging/Q&A

10. https://docs.mongodb.com/v3.4/ : Data Aggregation