# Caesar's Taxi Prediction Services

Predicting NYC Taxi Fares, Trip Distance, and Activity

Paul Jolly, Boxiao Pan, Varun Nambiar

***Abstract*—In this paper, we propose three models each predicting either taxi fare, activity or trip distance given a specific location, day of week and time using NYC taxi data. We investigated three approaches for each of the models - Random Forest, Fully Connected Neural Network and Long Short-Term Memory Network. We explored increasing data granularity by applying K-Means to group coarse location data into finer level clusters. In the end, we also plotted heatmaps of all three outputs as the final deliverable to drivers. Empirical analysis shows that our models can capture the relative magnitudes of labels among different locations very well. The models discussed in this paper aim to help taxi drivers find the most profitable trips.**

## I. Introduction

For an NYC taxi driver, being at the right place at the right time is often what makes or breaks a day. One may naively assume that the right spot corresponds to a place where demand is high, for instance, the New York financial district at the end of a work day. However, this may not necessarily be the best place to be. Taxi drivers might find themselves spending the whole day stuck in traffic traveling far away from the high activity zone thereby reducing overall profits for the day. Sometimes a better option might be for the driver to go to an area that is slightly less popular where people are making many short local trips, which would accumulate to a handsome sum over the course of the day.

To assist drivers in deciding where to be, we used NYC Yellow Taxi data provided by the NYC Taxi and Limousine Commission (TLC) [1] to create three different prediction models. Each model outputs either the taxi fares, activity or expected trip distances given a certain location, day of week, and time of the day in 30 minute intervals. For each model, we implemented three different approaches: Random Forest Classifier, Fully-Connected Neural Network and Long Short-Term Memory (LSTM) network. In order to make the best of our data, we also applied K-Means to increase the granularity of the data. And in the end, we generated heatmaps to show the results more intuitively. It turned out that our model can capture the relative magnitudes of labels among different locations very well.

In this paper, we discuss the design and results for each of these approaches and outline the next steps that would lead to a successful tool.

## II. Related Works

A few groups have tried to predict NYC Taxi Activity and Fares with a variety of features. One group used Random Forest Regressors and K-Nearest Neighbors Regression to predict pickup density using 440 million trips [2]. They used features such as pickup lat / long, time, day, month, year, weather, etc. Despite using more data, the model outputted activity maps with high levels of noise. Differently, we approached this problem by trying to solve a classification task. We discretized all the outputs so that we can improve the accuracy of the models. In addition, the group didn't use any of the newer and relevant taxi pickup data. Our approach attempts to provide a solution for training on both sets of data and providing pickup spots to a taxi driver that have a smaller area than the location IDs present in the newer data.

Recently there was a Kaggle competition that attempted to solve the fare prediction problem using features such as pickup / drop-off locations in lat / long, time, date, and number of passengers. Some of the approaches used include LGBM [3], XGBoost [3], Random Forests [4], etc. However the problem we're attempting to solve is different from the one in the competition. The competition aims to predict a fare given all the features about the trip including drop-off location whereas we attempt to forecast activity, fares, and trip distances using time of day, location information, and day of week. As a result we can use some of the methods used for the competition but we can not compare our model to the ones used in the competition.

The problem we are attempting to solve does not have much literature and most of the approaches we used have not been tested before.

## III. Data and Features

### A. Raw Data Description

The NYC Taxi and Limousine Commission (TLC) provides a large amount of trip data covering over a

billion trips from the years 2009 to 2018. Each trip sample contains a lot of information from which we chose pickup date and time, pickup location, drop-off date and time, drop-off location, fare amount, and trip distance. For data prior to July 2016 (referred to as older data), the pickup and drop-off locations are reported as latitude and longitude, while for data July 2016 onward (referred to as newer data), they are reported as location IDs - IDs that correspond to larger geographic regions. Other than this, the data is consistent over the years.

### B. Data Pre-Processing

Due to computational and storage constraints, we used a sample of the full data set from July 2014 to June 2018 which amounted to 2 million trips. For each of the models, we used a 90% / 5% / 5% split for training, validation, and testing respectively.

*1) Obtaining Activity:* To derive the activity feature from the provided data, we aggregated the trips for each location, day of week, and time interval and repeated for each week over the entire sampled data set (208 weeks). This resulted in 208 data points of activity for each location, day of week and time interval.

*2) Label Bucketing:* To convert this problem into a classification task, we discretized each of the labels - fare, activity and trip distance - into buckets. We first attempted to do this using K-Means clustering, however this resulted in a non-ideal spread of clusters in which some groups would be clumped too close together. This did not practically make sense since it would be more useful to a driver to split some of the larger clusters into smaller buckets. Instead, we used the K-means clusters as a guide to discretizing the labels into bins. Table I shows the discretized buckets for each of the labels. Both fare and trip distance were split into 8 buckets, while Activity was split into 9 buckets.

TABLE I: Bucketed Labels

| Bucket ID | Fare ($) | Trip Distance (Miles) | Activity (# Trips) |
|---|---|---|---|
| **0** | <0 | <0.5 | <2 |
| **1** | 0-5 | 0.5-1.0 | 2-5 |
| **2** | 5-10 | 1.0-1.5 | 5-7 |
| **3** | 10-15 | 1.5-2.0 | 7-10 |
| **4** | 15-25 | 2.0-3.0 | 10-15 |
| **5** | 25-50 | 3.0-5.0 | 15-25 |
| **6** | 50-60 | 5.0-10.0 | 25-35 |
| **7** | >60 | >10.0 | 35-45 |
| **8** | - | - | >45 |

### C. Feature Extraction

Feature extraction is divided into two steps: constructing non-time series set for the Random Forest and FCNN models and constructing time-series set for the LSTM model.

*1) Constructing Non-Time Series Data:* In this case, since every time we are inputting data at a single time point, so we simply used each data piece as an example, and splitted them into different subsets.

*2) Constructing Time Series Data:* By using sequential model, we hope our model can discover a relationship timewise, so we loop through the data and put the next $seq\_len$ data pieces as one single example only when these $seq\_len$ pieces and the following one piece are of the same location ID. Here $seq\_len$ is a hyper-parameter which denotes the length of input sequence, and through cross-validation we set it to 5. The corresponding output label for that example is the label (fares, activity or trip distance) of the next one data piece right after all $seq\_len$ pieces.

## IV. Methods

Our work employed various data pre-processing and modeling techniques. We go over the methodologies in this section.

### A. K-means Clustering and Non-Uniform Distribution

We cluster our Location IDs into 10 smaller clusters to increase the granularity of the model. In the newer data set, the pickup / drop-off locations are specified by large location IDs defined by the NY Taxi Commission. However the regions the commission chose were quite large and we wanted to reduce the area to a few blocks. K-Means algorithm works as follows: (1) Randomly assign cluster centroids in a given location ID. (2) Iteratively compute the cluster centroids to minimize the cost function. For every cluster $i$ compute:

$$c^{(i)} := \arg\min_j \left\| x^{(i)} - \mu_j \right\|^2 \tag{1}$$

For each $j$ set:

$$\mu_j := \frac{\sum_{j=1}^{m} 1\left\{c^{(i)} = j\right\} x^{(i)}}{\sum_{j=1}^{m} 1\left\{c^{(i)} = j\right\}} \tag{2}$$

Clustering works by first randomly assigning cluster centroids and then iteratively computing the cluster centroids to minimize the L2-norm between all the points in the data set and their respective centroid. This approach works well for our case because we are attempting to clump pickups by their spatial locations. This allows us to roughly estimate activity in a few blocks rather than a large neighborhood. Example outputs are shown in figure 1.

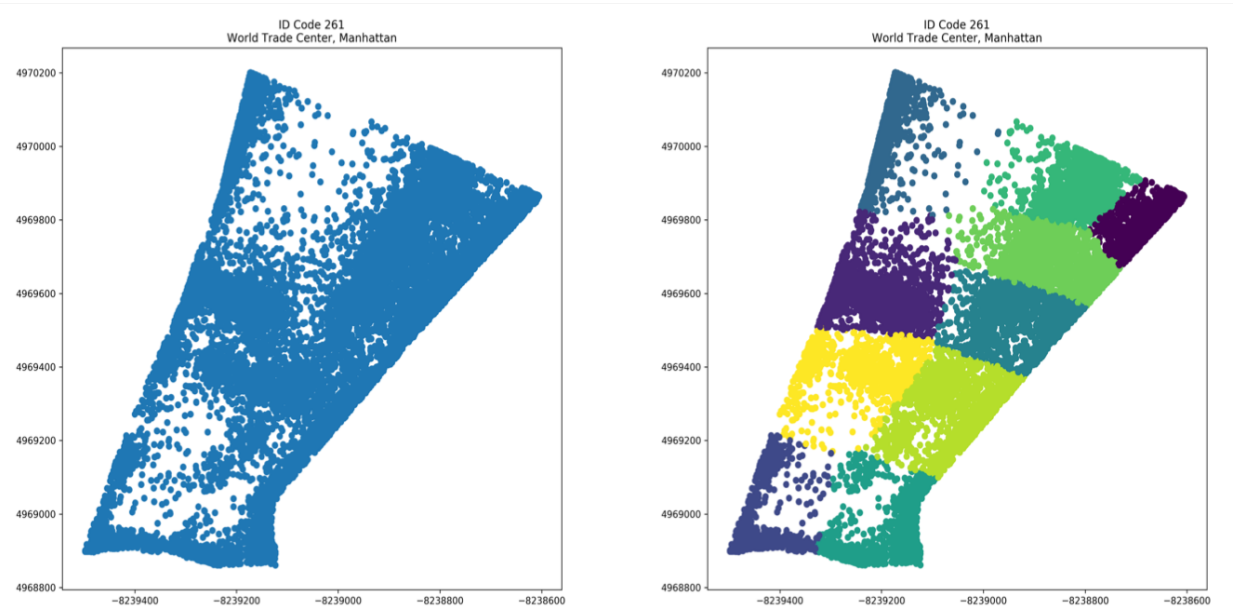


Fig. 1: Example of *K*-means clustering of location ID 261. The left image shows the pickup locations latitude and longitude that fall within location ID 261. The right image shows the result of *K*-means clustering of these trips into 10 clusters.

Once the clusters were created, we computed a probability distribution for the clusters within each location ID by computing the fraction of the total trips in a specific location ID that fell within a given cluster. Using this distribution, we assigned each trip in the newer data set to a cluster based on their location ID. This homogenization allowed us to combine the older and newer data sets together.

### *B. Models*

*1) Random Forest Classifier:* We used random forest classifiers with varying hyper-parameters to predict fare, activity and trip distance. Random forests construct an ensemble of decision trees that randomly compare features and assign a classification to a given input by calculating the mode classification from the outputs of the decision trees [5]. This method guarantees that the model does not overfit to the training data. We used the Gini loss for the random forests criterion shown in equation 3.

$$L_{gini} = \sum_{c} \hat{p}_c(1 - \hat{p}_c) \tag{3}$$

The gini criterion tries to minimize the missclassification loss and is generally less computationally expensive than the entropy criterion.

*2) Fully Connected Neural Network:* For the case of activity prediction, we used a fully connected layer consisting of 4 hidden layers and 1 output layer, while

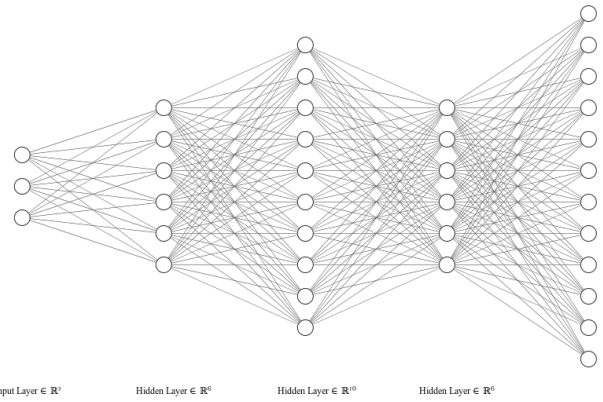


Fig. 2: Architecture of Our Fully-Connected network

the hidden layers contain 6, 10, 6 and 12 hidden neurons, respectively (see figure 2). All neurons in the hidden layers use ReLU as the activation function, while for the final output layer we used Softmax as the activation. We will not detail the network structure for all three types of outputs due to the space limitations. For the FCNN, we used the cross entropy loss function (equation 4).

$$L_{cross} = -\sum_{c} p_c(\log \hat{p}_c) \tag{4}$$

*3) Long Short-Term Memory Network:* A LSTM network is a variant of Recurrent Neural Networks (RNN), which was proposed by Hochreiter and Schmidhuber in

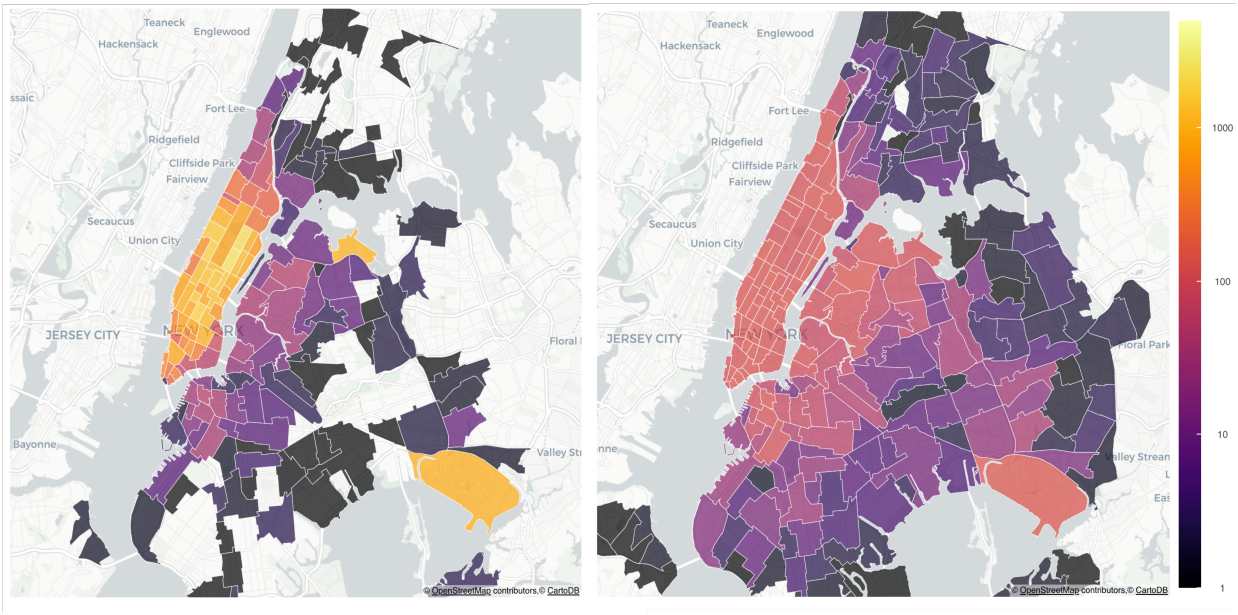


Fig. 3: Heatmap representation of ground truth activity (left) versus predicted activity (right). The lighter the color, the heavier the activity. Despite not being able to predict the exact magnitude of activity well, our model is able to capture the relative activity between different locations.

1997 [6]. RNNs are Artificial Neural Networks (ANN) in which unit connections form a directed cycle [7]. This cycle allows information to be passed from one step of the network to the next [8], which makes it good at dealing with data that possesses sequential patterns. LSTMs can achieve incredibly good performance on a large variety of problems that deal with sequential data such as Machine Translation, Sentiment Analysis and Natural Language Conversation, thus being widely used. LSTM redesigns the structure of hidden layers specifically to tackle the long-term dependency problem of RNN [6].

Since in our case we assume there is some relationship across time, we decided to adopt a sequential model, and used the widely adopted LSTM. For the example of activity prediction, we used a Long Short-Term Memory Network (LSTM) consisting of 4 LSTM layers and a final Dense layer on top. The output size for all LSTM layers is 128. Like the FCNN, we used cross-entropy loss as the loss function (equation 4).

## V. Results and Discussion

### A. Hyperparameter Tuning

*1) Random Forests:* We tuned the hyperparamaters of our model with the validation set. Some of the parameters we tweaked include number of trees, tree depth, and loss criterion. In our final models we used 40 trees, the gini criterion, and tree depths of two (for activity prediction) and three (for fare and distance predictions).

*2) FCNN/LSTM:* We tuned these hyperparameters on the validation set.

### B. K-means Clustering Output

Figure 1 shows an example result of $K$-means clustering of the World Trade Center (ID 261). The left image is a plot of the pickup latitudes and longitudes that occurred over two years at the World Trade Center, while the right image shows the same trips distributed into the 10 $K$-means clusters of the World Trade Center.

### C. Accuracy

Table II shows the accuracy results for each of the approaches for each model. RFC performs the best in all three models, but overall we see the performance is poor.

TABLE II: Prediction Accuracy for Each Model

| Model | Activity | Fare | Trip Distance |
|---|---|---|---|
| **RFC** | 51.02% | 45.71% | 30.08% |
| **FCNN** | 35.67% | 45.72% | 23.16% |
| **LSTM** | 26.65% | 27.72% | 23.10% |

### D. Heat Maps

With the predictions, we were able to produce heatmaps for each of the models to visualize how they vary with region for a given day of week and time

interval. Figure 3 shows an example heatmap output for activity from our RFC approach (right) compared to the corresponding ground truth (left). We see that despite the colors not matching completely (low accuracy), the relative magnitudes between regions are captured quite well. Therefore, our model is able to rank regions in terms of activity.

### *E. Discussion*

From Table II, we can see that the model does not perform well when predicting exact numbers. But the heatmaps show that our model can capture the relative differences between regions well. Our goal was to show drivers heatmaps so that they could get a rough understanding of the differences between regions. This is better than providing exact numbers since exact numbers would not be as meaningful to them. So in this sense our model has reached our expectation.

However, we have carried out some error analysis steps to analyze the poor performance of the models. First, we analyzed the K-Means clustering/non-uniform distribution used for distributing the newer data into the cluster IDs. We assumed that the clustering would have a low error but after seeing our results we tested it. We split 5% of the older data into a validation set, and compared the cluster IDs assigned to them (using Location ID and non-uniform distributions) versus their ground truth cluster IDs (we got this by directly mapping their lat / long to cluster ID). After running tests, the accuracy of our K-Means algorithm was only 11%. In order to test the effect of this poor performance on our prediction model, we further trained the model using only newer data (i.e., location ID rather than cluster ID). However, the results turned out to be approximately the same as with cluster ID. This suggests that our prediction model inherently doesn't do well. We attribute the poor performance of our prediction model mainly to two aspects. First, our models only use three features to tackle the problem and this might not be enough. Second, since we sampled 1% of the data from each month uniformly (due to computation issues), the time buckets may be too sparsely distributed. This sparse distribution may be causing models such as LSTM to not capture the temporal pattern in the data.

## VI. Conclusion and Future Work

In this project, we approached the problem of forecasting NYC taxi fares, activity and trip distance for taxi drivers. Specifically, we applied K-Means to increase the granularity of our data, implemented different models such as Random Forest, Fully-Connected Neural Network, and Long Short-Term Memory Network to do the prediction. Then we generated heatmaps using the predictions for taxi drivers. We also carried out error analysis steps to help diagnose the focus of our future work:

1) Extract more related features for our prediction model.
2) Currently our loss function penalizes all misclassifications equally. However some misclassfications may be better than others. A weighted loss function could help improve our model's accuracy. These weights could be determined by the distance between the actual classification and the predicted classification.
3) Even though yellow taxis are allowed to operate in all the boroughs of New York, there are regions where the taxis do not frequently pickup customers from. In a future iteration of our model we could disregard those regions where the taxis do not pick up from frequently. That way our model has plenty of data for a given location ID.
4) Using one model to understand the Taxi Pickup Patterns of 5 boroughs may be infeasible. We are making an assumption that the behaviors of people across all the boroughs can be predicted using one model. A possible future implementation could focus a single model on one of the boroughs. This may improve performance.
5) Sample more data.

## References


[1] "Tlc trip record section." http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml.

[2] S. Daulton, S. Raman, and T. Kindt, "Nyc taxi data prediction." Kaggle. https://sdaulton.github.io/TaxiPrediction/.

[3] "Cleansing eda modelling(lgbm xgboost starters)." Kaggle. https://www.kaggle.com/madhurisivalenka/cleansing-eda-modelling-lgbm-xgboost-starters.

[4] D. Sterling. Kaggle. https://www.kaggle.com/dster/nyc-taxi-fare-starter-kernel-simple-linear-model.

[5] T. K. Ho, "Random decision forests," in *Document analysis and recognition, 1995., proceedings of the third international conference on*, vol. 1, pp. 278–282, IEEE, 1995.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] "Recurrent neural network," Dec 2018.

[8] C. Olah, "Understanding lstm networks." http://colah.github.io/posts/2015-08-Understanding-LSTMs/.