Travel Planning Assistant

Yiting Li, Hongyi Li, Liutong Zhou
Department of Electrical Engineering
Columbia University
yl3407@columbia, hl2915@columbia.edu, lz2484@columbia.edu

Abstract — The big data analysis in transportation industry has shown great importance in travel planning and distribution. In this paper, we propose a travel planning system that predicts the traveling cost and taxi pick-up hotspots from large sets of NYC taxi data. In order to achieve the best prediction in New York City area, we studied and implemented machine learning models including: 1) Random Forest Regression; 2) Linear Regression; 3) K-Nearest Neighbor. Finally, the best trained model from each algorithm are combined to build the Travel Planning System Website. Experimental results demonstrate the correctness of our proposed method for travel planning and cost prediction.

Keywords: *Big Data, Travel Planing, Linear Regression, Random Foreset Regression*

I. Introduction

People usually do not have a plan when they are traveling or commuting in their daily life, so the motivation of our project is to help people make decisions in traveling as we called it travel planning system. We basically have three main objectives:1) save money and time on travel; 2) recommend optimal traveling plans for our users; 3) predict traveling costs based on our machine learning model. After the system is built, we believe our system would have some deliverables: 1) for single users, they could use our system to predict cost and avoid rush hours to take taxi on a specific location; 2) for taxi companies, they could use the prediction model for developing long-term policies for taxis distribution.

We briefly discuss previous work on travel planning in II, give an overview of our method and describe our data in III, and give a detailed description of our software description and each steps of algorithms in IV-V and the evaluation of each algorithms in VI. The main conclusion is summarized in VII.

II. Related Works

Some related researches were conducted to recommend users best time to travel[1]. These works will tell users the most efficient method to travel for saving money and time. However, some problems may arise. The users may wish to travel in certain range of time. If the system recommends a time out of this range, then this recommendation will make no sense. To address this issue, we choose another way to view this problem. We predict the taxi pickup density in a time range queried by users so that user can see the taxi volume in next few hours. In addition, the traditional method may fail to tell users the best location to take taxi. We will show the pickup density in the map so that the user can choose the nearest location with higher pickup density to take taxi.

III. System Overview

A. Spark

We run Spark on a personal workstation in pseudo distributed mode. The taxi trip datasets were stored in Hadoop Distributed File System. We used PySpark, which has been successfully integrated with Jupyter Notebook and Spyder IDE using FindSpark, for manipulating data and daya analysis and model training. Some middle and post results are analyzed or generated using Python and Matlab.

B. Data Processing

1) Data cleaning

Our team decided to employ the raw NYC taxi trip data that *is* available to public. We use 3 months' records *(*over 3 million records*)*. The advantages to use this dataset are that the dataset provided the location of pick up & drop down, time, and costs. These attributes are vital parameters to our model. However, the dataset is not quite ready to use. We found that there are some dirty records, as some of those records have latitude and longitude which are out of NYC and some have $ 0 cost. Thus, we only keep records with cost above 0 and with latitude between 40.5 and 41.1 and longitude between -74.1 and -73.6.

2) Data Transformation and Processing

After cleaning the data, we further found that the records during midnight is significantly less than records during daytime and saw-like data happened. These will probably cause two problems. The first one is that the size of midnight records is very small, which is likely to rise overfitting problem. Secondly, we planned to use the random forest algorithm which expects data to better have smooth transition. Based on this, we decided to encode the time series data to sine and cosine so that we can get smooth records and we can easily decode the time back based on its sine and cosine value. We found that if group by GPS points to analyze as location attribute does not make sense. Because the pick-up GPS location may not be exactly the same even in the same building, we planned to hash the GPS location to cluster nearby GPS. The function we use is an open source python script called Geohash.py which can be used to encode GPS point.

3) Traning set and test set:

We need to split the data set into training and testing. Inappropriate percentage of training and testing will rise problem. Smaller training percentage will cause overfitting problem and larger percentage will rise the testing accuracy inaccurate. Based on some researches, we decided to assign testing percentage to be 0.8 and test percentage to be 0.2 and save it into training set RDD and test set RDD. We programmed another script to transform those RDD into csv files.

IV. Algorithm

A. *Random Forest Regression*

The random forest regression is a machine learning algorithm that will construct multiple decision trees during training process and can be used to make predictions based on majority vote manner. We use this model to predict the average pick-up density in NYC in a day. The reasons to use this algorithm are that it is quick to train, capable to deal with large number of parameters and can judge importance of parameters.

During each iteration of the training, the decisions tree will be constructed from the updated training set and response set. Each decision tree is trained to be responsible for judging a very limited range of features. According to formula,

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B}\hat{f}_b(x')$$

(where $\hat{f}$ is the final output $\widehat{fb}$ is the decision made by each decisions)

The decisions tree will be assigned to make decisions of the pick-up density in terms of its assigned attribute and the system will weight their importance to make decision and export its output, the system will then average the output from each decision tree to make predictions.

The details of our training methods are as following

1) Training Input: time in sine, time in cosine, latitude, longitude, pickup times
2) Training Parameter: max depth is 30, number of iteration is 50
3) Sklearn APIs: We employ the python Sklearn API called RandomForestRegressor to train
4) Output: original records with additional column for predicted pickup times

B. *K-Nearest Neighbor*

The KNN (k-nearest neighbor algorithm) algorithm is a machine learning algorithm that make decisions or predictions based on its nearest k neighbor's property, that is, if its majority neighbors belong to a cluster, then it belongs to that cluster. We decide to utilize this model to predict pickup density based on the input time and location.

The calculation process can be divided by three steps. The first step is to calculate distance to each data. In our implementation, our team calculated distance based on the Cosine similarity instead of Euclidean distance because Euclidean have weaker performance when used into large variance of parameters.

The second step is to find nearest neighbors to test and the third steps is to do the classification. In order to make our model have better performance when applied to real life, we decided to append the training set columns to cover every 30 minutes from 0:00 to 23:59 so that users can use time as input to predict density

The details of our training methods:

1) Training Input: time slot with 30 minutes' intervals, pickup density

2) Training Parameter: k = 3, neighbors are 1,2,5

3) Sklearn APIs: KNeighborRegressor

4) Output: Models to predict pickup density

C. *Linear Regression*

1) Ordinary Least Square Linear Model

a) Feature Selection

A simple linear regression model is chosen for predicting the traveling cost. Firstly, we explore the features by visualizing the response variable against predictors in Tableau. Any observation that is out of the 99.7% confidence interval is taken as an outlier and is then removed from the dataset. After several iterations of cleaning, we are able to visually choose the most important features which contribute to the linear model most. Finally, two 2 out of 21 features, whose correlation coefficient are greater than 0.8, are chosen to be predictors in the linear model. Figure 1 A matrix plot of traveling cost vs predictors, which is generated using 1000 records, shows the linearity between the total price and the most critical features, namely traveling distance and traveling duration.

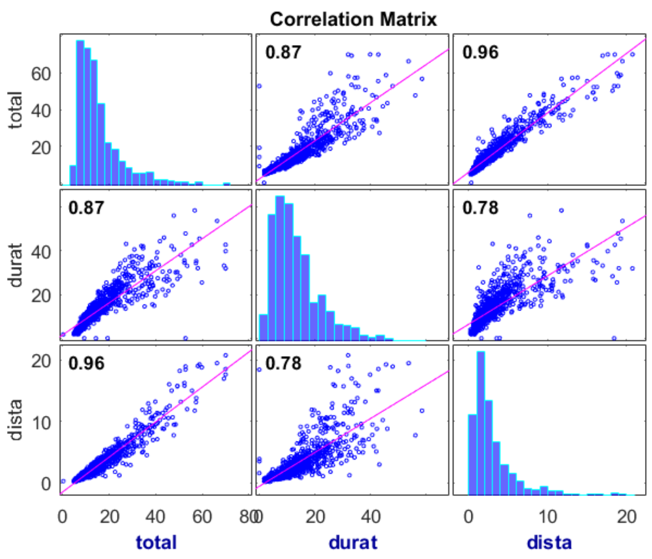


Figure 1 A matrix plot of traveling cost vs predictors

b) Training the Linear Regreesion Model

The final version of our model has the following form

$$\text{price} = \text{intercept} + \beta_1 \text{ distance} + \beta_2 \text{ duration}$$

1) Price: traveling cost by taxi in USD
2) Distance: traveling distance in miles
3) Duration: trip duration in minutes

The linear regression model class in PySpark MLlib solves the model by minimizing the following loss function.

$$\beta = \arg\min \text{Loss}$$
$$\text{loss} = \|price - \text{intercept} + \beta_1 \text{ distance} + \beta_2 \text{ duration}\|_2^2$$

The size of the training dataset is 1.73GB after data cleaning. One of the most important advantages of training a linear model using a large dataset is that the model tends to be unbiased and have minimum variance, which is guaranteed by law of large numbers and asymptotic optimality of MLE.

The output of the model, after hours of training, is
$\text{price} = 3.4 + 2.08 * \text{distance} + 0.43 * \text{duration}$

V. Software Package Description

We published all of our trained model and code on the Github, so that people could easily access to our work. We have also built a website to allow users visualizing our model and get predictions. The Travel Planning System website has three main functions.

Figure 2

A. Heat Map

Our website allows user a function to visualize the Heat Map, which is generated from the results of our machine learning model. The Geo Region Pick-up Prediction shows a time-lapse video for the pick-up times in every geohash flashing point range from 0 to 100.

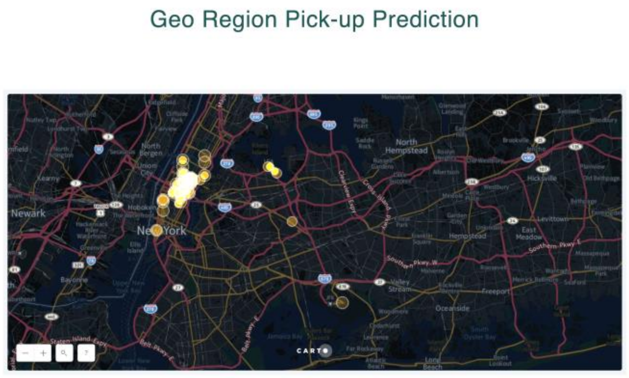
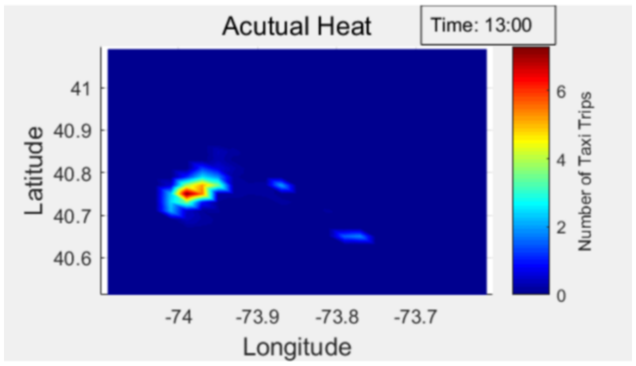
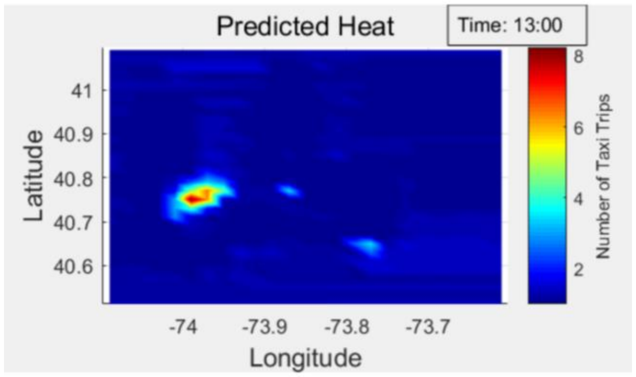


Figure 3

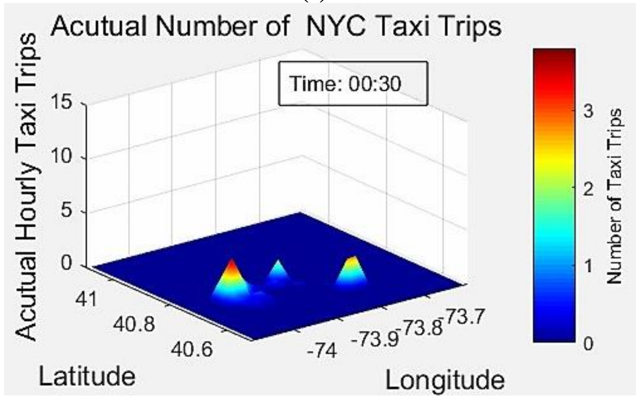
We also provided a Heat Map time-lapse video of Temporal-Spatial Pick-up Prediction for the user in our website. The x-axis and y-axis represent the longitude and Latitude. Both left and right side represent the actual and predicted heat. The deeper color hotspot means higher pick-up times in that area.



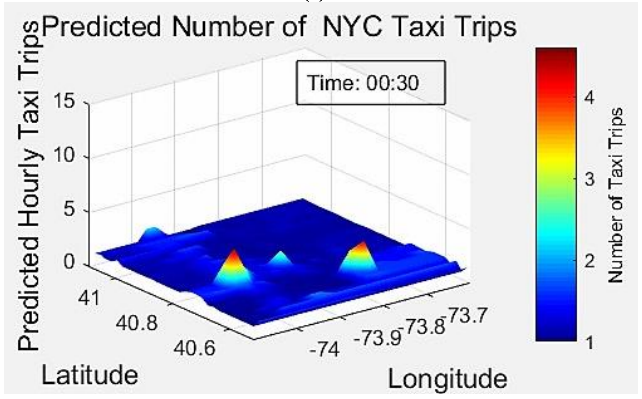
(a)



(b)



(c)



(d)

Figure 4 Predicting temporal-spatial distribution of NYC taxi trip volumes (a) Actual Heat Region over Time (b) Predicted Heat Region over Time (c) The actual number of taxi trips (d) The predicted number of taxi trips

B. *Recommendation*

Our website also provided a Recommend function to allow users enter their starting point, destination and a time duration. This function helps users to visualize the pick-up density in a certain time period from the current time. The chart is generated using google chart API, and as the front-end send the user inputs to the backend, the backend function would generate a data object that contains the predicted pick-up density from the model. As shown in Figure 5, we have tested the starting point from LaGuardia to JFK in the next 8 hours, and the system returns a chart that shows the predicted density of the starting area. Based on the chart, our user is able to avoid rush hours and save more time on the trip.

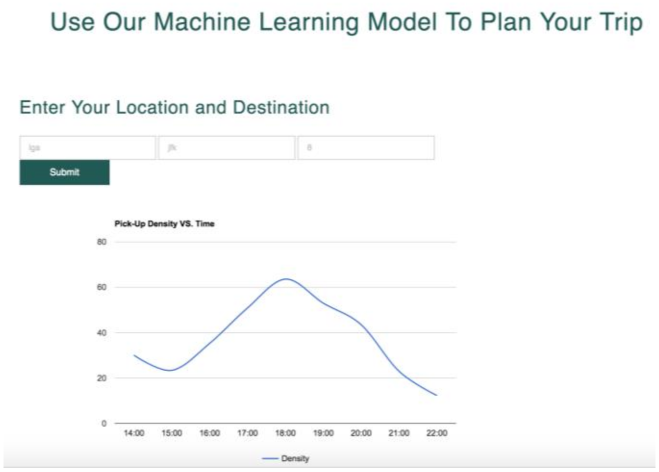
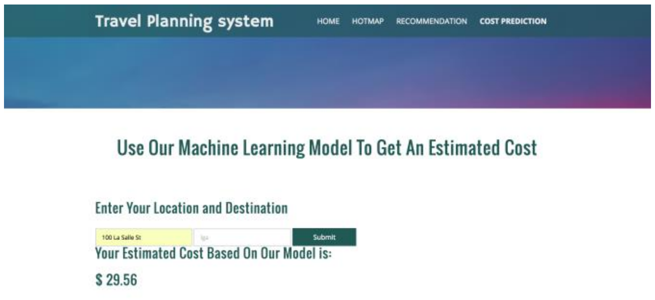

Use Our Machine Learning Model To Plan Your Trip
Enter Your Location and Destination
Submit
Pick-Up Density VS. Time
Density

Figure 5

C. *Cost Prediction*

The Travel Planning System website, we provided a Cost Prediction function which allows user to enter their starting point and destination. So our system could use our trained linear regression model to get a predicted cost and return back to the website. As shown in the Figure, the user entered “100 La Salle St” as the starting point and destination is LGA, after submission, the system would show an estimated cost which is 29.56 dollar.


Travel Planning system
HOME
HOTMAP
RECOMMENDATION
COST PREDICTION
Use Our Machine Learning Model To Get An Estimated Cost
Enter Your Location and Destination
Submit

100 La Salle St

Your Estimated Cost Based On Our Model is:

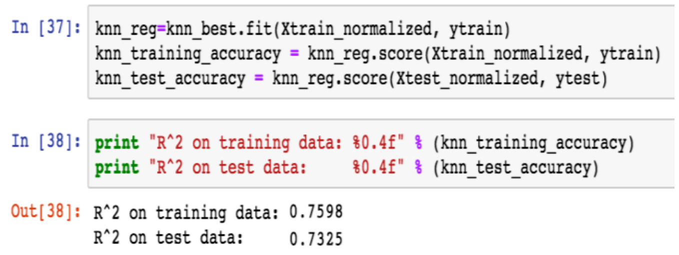
$ 29.56

Figure 6

VI. Experiment Results

a) Evaluating the Random Forest Regression and K-Nearest Neighbor

The model trained by the KNN achieved accuracy to around 76%. The accuracy is not that accurate as the training process of a large dataset is long (more than 1 day) and consume large amount of CPU. We have to adjust some parameters to make the process faster. We have to adjust the k be a small number and narrow the input columns and cut some records from original dataset. The testing result can be further improved if more dataset can be trained and number of k can be larger.

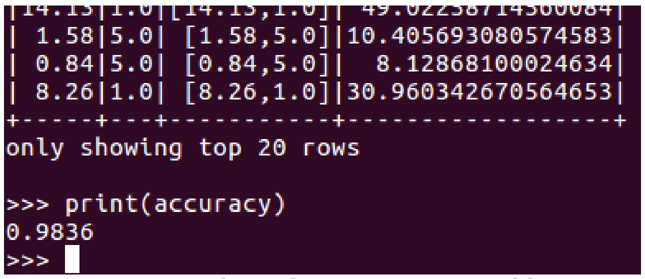


```
In [37]: knn_reg=knn_best.fit(Xtrain_normalized, ytrain)
         knn_training_accuracy = knn_reg.score(Xtrain_normalized, ytrain)
         knn_test_accuracy = knn_reg.score(Xtest_normalized, ytest)

In [38]: print "R^2 on training data: %0.4f" % (knn_training_accuracy)
         print "R^2 on test data:     %0.4f" % (knn_test_accuracy)

Out[38]: R^2 on training data: 0.7598
         R^2 on test data:     0.7325
```

Figure 7 Testing Accuracy

B. *Evaluating the Linear Regression Model*

The performance of the linear regression model is evaluated using a 5% held-out test set. Figure 8 Evaluating the Linear Regression Model shows the prediction accuracy of the model trained using large sample to be as high as 98.36% percent. The in sample prediction error estimated using held-out test set is as small as 10.9752.



```
|14.13|1.0|[14.13,1.0]| 49.02238714360084|
| 1.58|5.0| [1.58,5.0]|10.405693080574583|
| 0.84|5.0| [0.84,5.0]|  8.12868100024634|
| 8.26|1.0| [8.26,1.0]|30.960342670564653|
+-----+---+-----------+------------------+
only showing top 20 rows

>>> print(accuracy)
0.9836
>>> 
```

Figure 8 Evaluating the Linear Regression Model

VII. CONCLUSION

We proposed a Travel Planning System that used big-data machine learning models to predict taxi trip volumes, discover traffic patterns like NYC hotspots, predict traveling costs, provide travel information to users to make travel plans.

By using large dataset and appropriate models and features, we achieved high prediction accuracy in both predicting the taxi trip volumes and predicting the traveling cost.

Our team members are Liutong Zhou, Hongyi Li and Yiting Li. Hongyi Li implemented the random forest regression algorithm and KNN algorithm for predicting traffic volumes in PySpark, Yiting Li designed the user interface and developed the front end, Liutong Zhou implemented the ordinary least square model for predicting traveling cost in PySpark and created the visualizations based on the output results. All members have collaborated closely in making every submission material and have contributed to the project from different aspects.

ACKNOWLEDGMENT

We would like to thank Professor Ching-Yung Lin for giving us wonderful lectures, imparting us necessary knowledge about big data machine learning algorithms and techniques and the all TAs for answering questions.

References

[1] Chen RC., Chen CT., Li JY. (2012) A Genetic Algorithm for Planning Travel Route with Mimimum Transportation Carbon Footprint. In: Qu X., Yang Y. (eds) Information and Business Intelligence. Communications in Computer and Information Science, vol 268. Springer, Berlin, Heidelberg