New York City Taxi Fare Prediction

Project ID: 201812-31

Di Wu[1], Shuhao Qiao[1], Yunfei Wang[1]
[1]The Fu Foundation School of Engineering and Applied Science
Columbia University
dw2834@columbia.edu, sq2205@columbia.edu, yw3157@columbia.edu

Abstract—**Predicting taxi fares has always been important for our daily life. In this course project we use different predictors and different algorithms to build several models to predict taxi fare in New York City. After finding out the best model we achieve, we analyze the model in many ways and find out the important predictors in the model as well as how each model affects the prediction results. Finally, we make a demo Python application for users to predict taxi fares.**

Keywords: transportation, taxi fare, prediction, LightGBM, SHAP

I. Introduction

Taxis have been making our lives easier than ever before, and it has been quite familiar for all of us to take a taxi or enjoy the service from car-hailing platforms such as Uber, Lyft and Didi, especially in big cities like New York. Yet unlike the platforms that present us with an estimated car fee before we taking the ride, we can only know the exact amount of taxi when we arrive at the destination. This awkwardness makes us curious about the prediction of it, and a precise estimation will be of great help for us to control the daily budget.

Motivated by the facts above, we mainly focus on three points in this project. The first point is to build a prediction system to provide relatively accurate predicted taxi fare for given pick-up and drop-off locations in New York City. The second point is to investigate how each predictor affects the prediction result. The third point is to develop a Python application for users to predict the fare.

The rest of this report is organized as follows. Section II introduces some related works concerning taxi-fare prediction. Section III gives an overview of our system, as well as showing our dataset. Section IV introduces the algorithms we use in this project briefly. Section V presents the experiment results of our prediction system. Section VI makes some analysis for the prediction as well as the dataset. Section VII introduces a demo Python application we developed to predict taxi fare. And Section VIII makes a conclusion.

II. Related Works

Obviously, the taxi fare is highly related to the mileage and during of a specific ride. While the former is relatively easy to calculate or estimate, the latter is much harder, where actually attracts a lot of researches in this field [1].

Some researches focus on dealing the problem with the help of real-time data. Vanajakshi et al. use real-time GPS data from buses on the road [2], while Biagioni et al. use real-time smartphone data on the road [3]. However, some researches consider adopting machine learning methods to solve the problem. Wu et al. and Van et al. use Support Vector Regression (SVR) and Neural Network respectively [4-5].

III. System Overview

We first briefly introduce the dataset we use. Both our train set and test set are from Kaggle Playground Prediction Competition *New York City Taxi Fare Prediction* [6]. The size of the train set is more than 55 million and the size of the test set is 9914.

The data is consisted of 8 parts: “key”, “fare_amount”, “pickup_datetime”, “pickup_longitude”, “pickup_latitude”, “dropoff_longitude”, “dropoff_latitude” and “passenger_count”. The column “key” actually serves as an ID, which is different for each line of data. The column “fare_amount” is a float number which acts as a ground truth of our input. We are going to use this column of the training set as output of our system to fit the model. The rest parameters make sense as we can understand the meaning from their names. “pickup_datetime” let us know the exact time when the passenger was picked up by the taxi. And “pickup_longitude”, “pickup_latitude”, “dropoff_longitude” and “dropoff_latitude” tell us the position of starting and ending position of the ride. Finally, “passenger_count” records the number of the passengers on the taxi.

Before training, we need to do some preprocessing-related work. We remove the data with null value as well as with negative fare. We also remove the data with pick-up or drop-off locations that is outside of New York City.

For a better training and prediction, we also add some features to the data. One of those features is the distance between pick-up and drop-off locations. Note that here we use great circle distance calculated by the longitude and latitude of the two positions. We also separate the timestamp into different columns and each column represents one element in the timestamp, such as year, hour and etc.

Moreover, we add another three features "JFK", "EWR" and "LGA" standing for the distance of the route to three airports in New York City. Here we adopt the minimum distance of either from the pick-up position or from the drop-off location to the airport as the value of the specific feature. This variable is useful to show if a trip starts or ends at an airport, which would be useful in prediction. After finding out the locations of the three airports, the distances are easy to be calculated.

Before training and prediction, we make a graph presenting all pick-up points in our training dataset on a map of the center of New York City, which is shown in Figure 1. We find that the data is quite accurate that we can even figure out the streets and avenues from the data points. Thus, we believe the data is sufficient for us to train and predict.

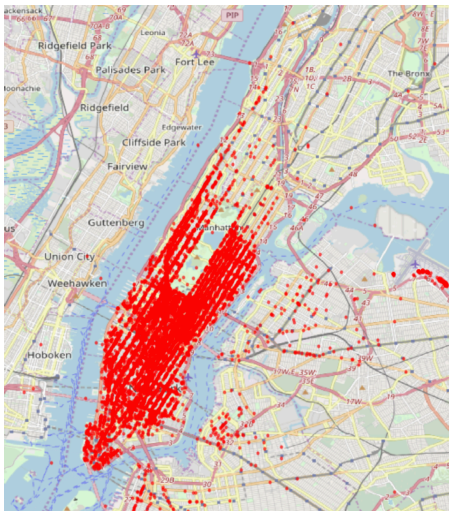


Figure 1. Pick-up points in the training dataset

We use different predictor combinations as well as algorithms to build models to make the prediction. The algorithms and experiment results of the models are demonstrated in the next two sections.

IV. ALGORITHM

After the preprocessing step, we have the dataset with all strange lines removed and predictors added. In this chapter, we briefly introduce the algorithms and tools we use in this project, including Linear Regression, Decision Tree and Random Forest in PySpark, as well as LightGBM.

Linear regression is a basic approach in machine learning. We are given some predictors as input and we wish to predict the value of an unknown variable. For a direct view of Linear Regression, what we aim to do is to find a single line to make as much dots as possible near to it. There are several ways to decide the variables including least square criterion and gradient decent method. The former method makes calculating variables much easier but has some restrictions to deal with input in matrices. While the main idea of the latter approach is to find the minimum value in the fastest descending direction by assigning the step value and a maximum iteration time. Note that for Multiple Linear Regression, we simply change the number of variables, and the principles are exactly the same.

We also use Decision Tree regression whose main idea is to build a regression tree with deterministic statements on each node. Every time we reach a node, we see if the current condition is matched. Either choice will lead us to a depth further. By the time we reach the leaves of the tree, we can get our result. The Decision Tree regression has an advantage over linear regression in that it is very easy to interpret. However, it may result in overfitting and result in a very low training error but a high testing error, whose main reason is that there is always noise in the training data and a decision tree simply took the noise as part of the partition criterion, making it hard to present the real characters of the data. Also, the lack of representative data may also cause a large testing error since a class of data may not be fitted properly.

Another method is called Random Forest. It is an important ensemble learning method based on bagging which could be used in both regression and classification. We pick n training samples from the original training set by bootstrapping the data to gain k bootstrapped training sets. Note that bootstrapping is a randomized pick with replacement, which may result in duplicate data. Then we train k models and use the average of the results as the final output. Introduced randomness of the model, the algorithm is less likely to overfit the model. It is also of better resistance to the noise also because of the randomness. This algorithm is also able to deal with data of high dimensions without having to make feature selection. Yet it also has a drawback of requiring large time and space complexity when there are many trees in the forest.

LightGBM is a framework developed by Microsoft Research. As is said in its document, it is a gradient boosting framework that uses tree-based learning algorithms and is designed to be distributed and efficient. The most important feature of LightGBM is that it adopts leaf-wise tree growth method instead of level-wise tree growth

method. It has many fantastic features such as higher speed, capable with larger dataset, lower memory and more accuracy. However, since it is quite sensitive to overfit, it may be only useful to deal with large dataset. In our project, we adopt it as our final model, which will be discussed later.

V. Experiment Results

As is indicated above, we have data for training and testing in this project. The evaluation matric we adopt is root mean-squared error (RMSE) between our prediction results and the ground truth of the testing data. A lower RMSE value indicates that the model is better.

First, we use PySpark to build some machine learning models. We try different combinations of predictors including distance between pick-up and drop-off locations, the longitude and latitude of both pick-up and drop-off locations, the time information (hour and weekday), and the number of passengers on the ride, as is shown in Table 1. We also try different algorithms including linear regres sion, decision tree and random forest. The experiment results are shown in Table 2.

Table 1. The combination of predictors in models

# Model	Distance	Locations	Time	# passenger
1-3	Yes	No	No	No
4-5	Yes	Yes	No	No
6	Yes	Yes	Yes	Yes

Table 2. The RMSE value of each model

# Model	Algorithm	RMSE
1	Linear Regression	5.79968
2	Decision Tree	4.55898
3	Random Forest	4.50127
4	Decision Tree	4.28111
5	Random Forest	4.62682
6	Decision Tree	4.25675

As is shown above, the RMSE value of Model 1 is the highest in Models 1-3, indicating Linear Regression cannot achieve a good prediction under this circumstance. With adding more predictors, models with both Random Forest and Decision Tree algorithms have lower RMSE and better accuracy. However, it is still far away from our expectation.

We build another two models using LightGBM. The first model adopts the same predictor as Model 6, which is the best model we achieve using PySpark. The RMSE value of this additional model is 3.56563, which is much better than that of Model 6. The second model using LightGBM has all predictors in Model 6 and more time information (Year, Month, Day), and the information concerning the three airports in New York City. This final model has the best RMSE value 2.96126 ever before in our project, which is a pretty decent accuracy for us.

Next, we show some prediction results of our final model. Here we have a pair of examples both from Columbia University to Time Square but at different time, the predicted fares are $17.93 and $21.45 respectively. And another route from Columbia University Morningside Heights to Manhattanville, whose predicted price is $4.80. Thus, it can be found that the distance and time really matters to the prediction result.

We have another pair of examples, where the first one is from Columbia University to JFK International Airport, and the other one is from Battery Park to JFK, at the same time. The distance of the two routes are 14.50mi and 12.50mi respectively, whose difference is around 1/6. However, the predicted fares are $60.73 and $57.23 respectively, which are really close to each other. This is because of the flat fare rule of JFK, i.e., the taxi fare from/to JFK and to/from any position in Manhattan is fixed at around $60. Obviously, our model learns this information quite well.

VI. Analysis

The tool SHAP is applied for interpreting this final LightGBM model. We compute the mean absolute SHAP value for each predictor in the model, which presents the importance of each predictor indirectly, and the result is shown in Figure 2 below.

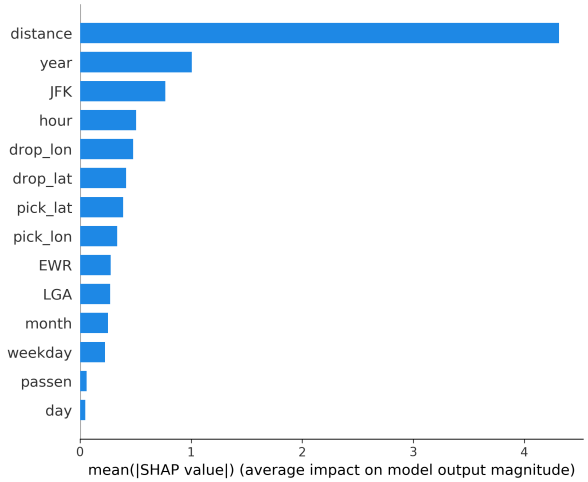


Figure 2. Mean SHAP value of each predictor in the final model

As can be found in Figure 2, the top four important predictors are “distance”, “year”, “JFK” and “hour”. It is common sense that distance would be the most important predictors, and the importance of year indicates that the price per mile is rising or the road becomes more crowded these years. The distance to JFK International Airport is also important may be partly because of its flat fare rule, and it is reasonable that price is higher at night and during rush hours. The importance of all position predictors ranks from

5 to 8, followed by the distance to another two airports and other time information. The number of passengers and the date are the two least important predictors.

Further, we investigate how these predictors affect the prediction result in our final model. The result is shown in Figure 3.

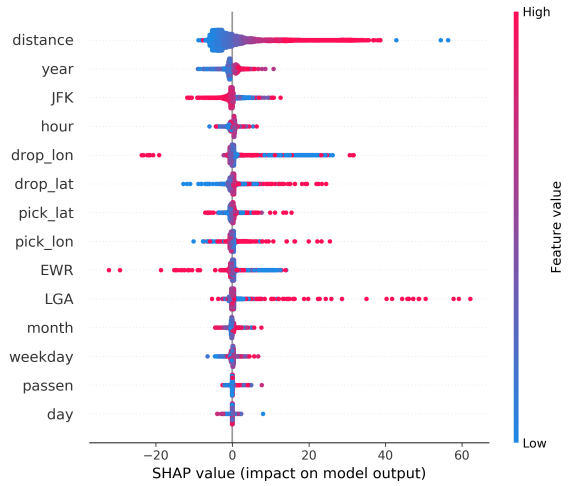


Figure 3. SHAP value with feature value of each predictor in the final model

In Figure 3, the x-axis indicates the SHAP value, where a predictor with a positive SHAP value means that the predictor helps to raise the predicted result. The color in the figure shows the value of the predictor.

Let us focus on the five predictors with obvious features below. We can find out that in general the larger the distance of the trip is, the higher SHAP value it has, which indicates that the prediction result is quite related to the distance. The similar pattern appears on the predictor "year", where we can infer that the price of the taxi may be more expensive per mile, or it is more crowded on the road these years. The other three predictors are airport-related predictors. As we can see, the pattern of "JFK" and "EWR" is quite similar with each other, while that of "LGA" is obviously different. Actually, a higher value of airport-related predictor indicates that both the pick-up point and the drop-off point are far from the airport. Since JFK International Airport is far from the city center, a higher value on "JFK" means that the trip is mostly in the center part of the city, which usually indicates that it is only a short journey with low price. However, since LGA Airport is near to the center of city, the pattern is different and actually more complex and non-interpretable.

We also use html, D3 package and JavaScript to make a day-hour heatmap, as is shown in Figure 4, where a darker block means that there are more people taking taxis at the specific time slot. Note that only the relative values make sense in the heatmap since we only use a part of data in our train set to make the graph. There are many interesting facts that we can figure out from the figure. First, people in New York City work really hard on Monday and Tuesday since the routine of these two weekdays seems quite normal. Next, there seems many activities held on Wednesday to Friday, no matter working or enjoying parties, as the period of rush hours on these days are longer than others. Besides, people play really hard on weekend nights that even it is really late, there are still many people taking taxis.

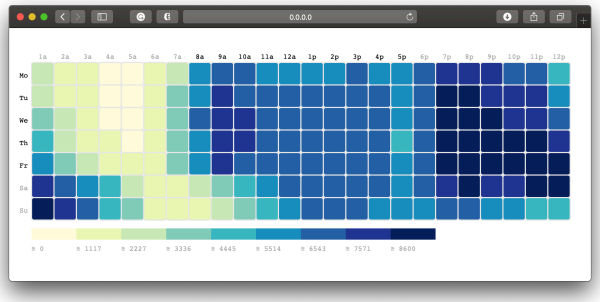
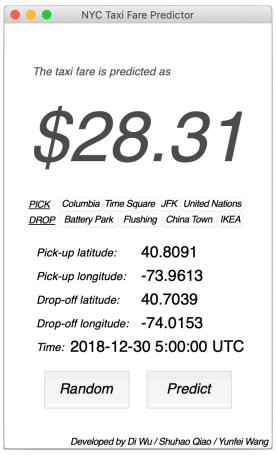


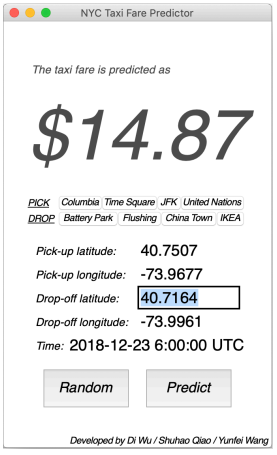
Figure 4. Screenshot of a webpage showing the day-hour heatmap of taking taxis numbers

VII. Software Package Description

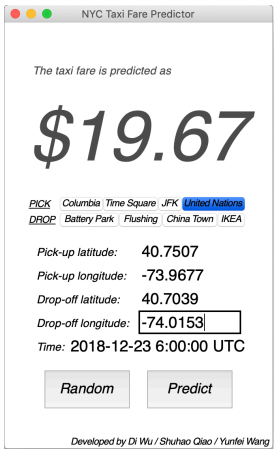
We build an application using Python in this project. Users can get to know the predicted taxi fare from and to any two points in New York City. Since it is just a demo version, users can assign pick-up and drop-off locations by clicking the "Random" button, entering the longitude and latitude by themselves, or choosing the giving spots in New York City. After assigning and clicking the "Predict" button, the prediction result will show on the above. Figure 5 (a)-(c) show the interface of the application as well as how user assign locations using the above methods.



(a) "Random" button



(b) Entering positions



(c) Choosing given spots

Figure 5. Interface of Python application and methods to assign locations

VIII. Conclusion

In this course final project, we use New York City taxi fare data to make prediction and analysis. We first do some overview throughout the data, and do some data cleaning work. We try different combinations of predictors and different algorithms in PySpark to build models and predict the fare, while all models give unsatisfactory results. We then add more predictors such as the airport information as well as try other algorithms such as LightGBM framework where we get a decent prediction result, and it is selected as our final model. We use SHAP tool to analyze and interpret the final model, where we figure out important predictors to the model as well as how predictors affect the prediction results. And we also build a web page using JavaScript showing the day-hour heatmap of taking taxis numbers, where we derive a lot of interesting facts concerning people's daily life. Finally, we make a Python application to help users predict the New York City taxi fares.

Since we are not quite familiar with the parameters in LightGBM before, the future work may include trying to tune these parameters to get better models. Meanwhile, several other powerful algorithms may even have better prediction. Embedding several Neural Network algorithms to make better predictors could be a possible solution. We are also going to add some other features such as weather condition and holiday into consideration, to make more precious and reasonable models, since these features sometimes really matters. Finally, we are going to try to build a web application for people in New York City to predict taxi fares.

Acknowledgment

The authors would like to thank the professor and all teaching assistants for providing the experiment platforms and their patience teaching as well as assistance.

Appendix

Below is the individual contribution of each group member. Di Wu is responsible for most of coding and programming task, including PySpark, html and application part. Shuhao Qiao is responsible for the proposal of this project, and does SHAP analysis as well. Yunfei Wang is responsible for PySpark and LightGBM parts coding. Each team member takes part in the preparation of the final presentation and the project final report.

References

[1] Christophoros Antoniades, Delara Fadavi, Antoine Foba Amon Jr., "Fare and Duration Prediction: A Study of New York City Taxi Rides", Standford Course CS229 Report, 2016.

[2] L. Vanajakshi, S. C. Subramanian and R. Sivanandan, "Travel time prediction under heterogeneous traffic conditions using global positioning system data from buses", IET Intelligent Transport Systems, vol. 3, no. 1, pp. 1-9, March 2009.

[3] James Biagioni, Tomas Gerlich, Timothy Merrifield and Jakob Eriksson, "EasyTracker: Automatic Transit Tracking, Mapping, and Arrival Time Prediction Using Smartphones", Proceedings of the 9th International Conference on Embedded Networked Sensor Systems, pp. 68-81, 2011.

[4] Chun-Hsin Wu, Jan-Ming Ho and D. T. Lee, "Travel-time prediction with support vector regression," in IEEE Transactions on Intelligent Transportation Systems, vol. 5, no. 4, pp. 276-281, Dec. 2004.

[5] J.W.C.van Lint, S.P.Hoogendoorn and H.J.van Zuylen, "Accurate freeway travel time prediction with state-space neural networks under missing data", Transportation Research Part C: Emerging Technologies, Volume 13, Issues 5–6, pp. 347-369, 2005.

[6] https://www.kaggle.com/c/new-york-city-taxi-fare-prediction, Kaggle Playground Prediction Competition: *New York City Taxi Fare Prediction*