A retrospective on Taxi Hailing methodology

Sanskar Shah

Student, Department of Information Technology, K J Somaiya College of Engineering, Maharashtra, India

---***---

Abstract - *This paper analyses the NYC Yellow Taxi Cabs data. The data is multidimensional in form and gives an acute insight into many different features of a taxi trip like the Pick Up point, Drop Off point, Trip Distance, fare Amount, Taxable Amount, etc. Here the goal is to analyze these features to develop the best model that predicts accurately for this given dataset while also giving an overview into the algorithms used with some details. Using this, users can benefit by forecasting the fare amount that is charged for each trip and which location will be cheaper for them to hail a taxi from.*

Key Words: fare predictor, NYC TLC, Linear Regression, XGBoost

1. INTRODUCTION

The rise of Uber as the global alternative taxi operator is interesting. More interestingly since most features that Uber offers to the day to day traveler are indeed present in the more conventional taxi-cab ride. Let us take an example, every Uber ride starts with your phone, you entering your Pickup and Drop off location, checking the ride price, comparing the same with any competitor like Ola or Lyft depending on where you stay and then booking the cab. From there on, an available and willing driver is assigned to your ride, you need to wait for the driver to arrive, pick you up and have a smooth ride. In a more conventional means of transport, it would be a lot simpler, you will simply go down and hail a cab depending on whether a driver is willing or not. When we look at both approaches they are quite similar, only difference for a customer being the increased accessibility and the ability of compare prices on the downside it means that the customer has to wait for a driver and then explain where he/she is when the driver finally arrives. None of these problems are in the conventional approach to hailing taxis nevertheless Uber seems to be on the rise. All of this and the only additional datapoint is the ability to check the price. So, let us remedy that.

1.1 Problem Definition

Our primary goal is to be able to show any user what a taxi ride will cost him using a traditional means of transport, like hailing a regular taxi from down the road.

This can be done by analyzing the everyday data of public transport and one of the publicly available datasets can be used to do this. The New York Yellow Cabs dataset provides information regarding fares paid by the taxi customers for each trip, pickup location, drop-off location, time and the distance travelled during a taxi ride. Our objective is to use this data and to predict the fare amount for a given pickup and drop-off location based on the distance between the two locations. To do this also there are many methodologies that have been proposed over the years but this paper focusses on analyzing the data initially and then giving a prediction of the cost for the journey by checking which model is best suited for such a use case.

In this paper, we use various regression models to predict the fare of a taxi ride based on the distance, pickup location and drop-off location and those are:

- Linear Regression
- Linear Regression with fourier features
- XGBoost Regression

1.2 Dataset

The dataset used is massive and hence, to process that data to generate information useful to us will take a considerable amount of time. To trade off between the processing time and accuracy, to make our model efficient, we chose the most relevant attributes to develop our model. For example, Currently the model does not use the datetime parameters but in the future the same could be used to allow for taking the trip duration into consideration or take into consideration price hikes due to demand at same point in time. For the purpose of analysis in this paper we have restricted ourselves to 4 columns only – PULocationID, DOLocationID, Trip_distance, Fare_amount.

Any dataset from the TLC contains around 10000000 rows per month and 17 columns that are described below. The dataset used by us contains 78,32,546 rows. The description of the rows we used are as follows:

Table -1: Dataset Description

Field Name	Description
Trip distance	The distance of the journey for each time the taximeter is reset.
PULocationID	TLC Taxi Zone where the taximeter was started. (journey start location ID)
DOLocationID	TLC Taxi Zone where the taximeter was stopped. (journey stop location ID)

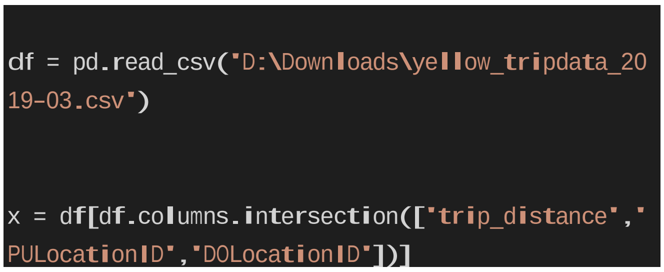
Fare amount	The fare amount based on the time and distance elapsed during journey

2. Proposed Methodology

This approach is applicable universally across any city although choosing the city of New York was an easy choice. The availability of data there and the huge dataset that they provide free of charge along with the API support provided for the same helped us a lot during the length of this paper.

2.1 Data Cleaning

Since the extent of the data is massive, we decided that having any kind of errors or inaccurate data is unacceptable. Also, taking advantage of this large amount of data, we could easily neglect rows rather than giving them any kind of 0 values or wrong input. We used the following code:



```
df = pd.read_csv("D:\Downloads\yellow_tripdata_20
19-03.csv")


x = df[df.columns.intersection(['trip_distance','
PULocationID','DOLocationID'])]
```

What the above code snippet does is that it reads the data into a pandas type data frame and then using the function intersect, it only allows for data to be read if it actually exists across all 3 values – 'trip distance', 'PULocationID', 'DOLocationID'. If any of these values are not present it will skip that row entirely hence providing us with a more accurate result with no compromise in accuracy of the model itself. This method provides all numeric data and no NA values which will hinder our model development.

2.1 Data Analysis

We have used three of the more common algorithms which are efficient and accurate. In order to decide which one of them will provide us with the best output, we tried them all and decided which gave us the highest accuracy.

1. *Linear regression:*

Linear Regression is the most primitive machine learning algorithm and was developed based on mathematical Regression analysis. Similar to most regression models, it predicts a dependent attribute using values from other independent attributes from the dataset. It is mostly used to determine the relationship if it exists among attributes and for predicting values based on statistics. It works by determining a linear/non-linear relationship among the attributes assuming that they are dependent and then uses it to predict the value for a dependent attribute.

What Linear regression does is to predict a dependent attribute value (y) based on multiple input variables (x[]). So, using this technique, we find out a linear relationship between x[] and y. On finding this relation we can generalize it into a general equation which can be used for everything. Therefore, it is known as Linear Regression.

$$y = \theta_1 + \theta_2.x$$

We are given the following while training the model:

X: matrix/vector of independent variables

Y: vector of dependent variable/label

Depending on the learning algorithm used (Gradient descent, etc.), the number of iterations and the learning rate, etc., the training model fits a line to predict the value of Y for given value(s) of X. Then the model gets the best regression fitting. After this θ1 and θ2 can be used to predict all values required.

θ1: intercept

θ2: coefficient of x/ slope of best fit line

As we find the best values for θ1 and θ2, we get the best fit line. Therefore, when we are using the model for forecasting, it will forecast the value of y for an input of x.

2. *Linear regression using Fourier Features:*

"A continuous kernel κ(x, y) = κ(x − y) on $\Re^d$ is positive definite only when κ(δ) is Fourier transform of a non-negative measure." This theorem suggests that if κ(δ) is scaled properly, its Fourier transform p(ω) is a proper probability distribution.

The random feature set has random Fourier bases, cos(ω T x + b), where ω ϵ ℜd and b ϵ ℜ. Data points are projected into a random chosen line and then through a sinusoidal function, the resulting scalar is passed. Then address every component of the projection. Starting with ω. Assume a given, shift invariant kernel, which is to be approximated, κ(x,y) = κ(x – y). At last, we have x, y ϵ ℜD. Now we get, p(ω) as the Fourier transform of κ(δ), and its inverse is:

$$p(\omega) = \frac{1}{2 * \pi^D} \int_{\Re^D} \kappa(\delta) e^{-i\omega^T \delta} d\delta$$

$$\kappa(\boldsymbol{x} - \boldsymbol{y}) = k(\delta) = \int p(\omega) e^{i\omega^T(\boldsymbol{x}-\boldsymbol{y})} d\omega$$

Now we assume tha $\zeta_\omega(\boldsymbol{x}) = e^{i\omega^T \boldsymbol{x}}$ then:

$$\kappa(\boldsymbol{x} - \boldsymbol{y}) = E_\omega[\zeta_\omega(\boldsymbol{x}) \zeta_\omega^*(\boldsymbol{y})]$$

where the random variable ω follows the probability distribution function p(ω). Since, κ(δ) and p(ω) are both real, the integral converges when the complex exponentials are replaced with cosines. Therefore, we expect to obtain a real-valued mapping that satisfies the condition. So, we set:

$$Z_\omega(\boldsymbol{x}) = \sqrt{2}cos(\boldsymbol{\omega}^T\boldsymbol{x} + b),$$

where ω is drawn from $p(\omega)$ and b is drawn uniformly from $[0, 2\pi]$. Then,

$$\kappa(\boldsymbol{x} - \boldsymbol{y}) = E_{\omega,b}[Z_\omega(\boldsymbol{x})Z_\omega(\boldsymbol{y})]$$

Random Fourier Features:

Select a shift invariant kernel (e.g. Gaussian Kernel) and the respective parameters.

$$\boldsymbol{x} \mapsto Z_\omega(\boldsymbol{x}) = \frac{\sqrt{2}}{\sqrt{D}} \begin{pmatrix} cos(\boldsymbol{\omega}_1^T\boldsymbol{x} + b_1) \\ cos(\boldsymbol{\omega}_2^T\boldsymbol{x} + b_2) \\ \cdot \\ \cdot \\ \cdot \\ cos(\boldsymbol{\omega}_D^T\boldsymbol{x} + b_D) \end{pmatrix}$$

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) \approx Z_\omega(\boldsymbol{x})Z_\omega(\boldsymbol{y})$$

Calculate the Fourier transform p(ω) of κ(·, ·).

• Choose D random elements, ω1, ω2, ..., ωD from p(ω)

• Choose D random elements, b1, b2, ..., bD uniformly from [0, 2π]

3. *XGBoost*

XGBoost stands for eXtreme Gradient Boosting. It is an algorithm build for efficiency and performance and is one of the most used algorithms for any kind of data science application in today's world. It is a Machine Learning algorithm that uses a gradient boosting framework. The main advantages of this algorithm reflect upon its deep integration with the system where it provides invaluable features like parallelism, distributed computing, out-of-core computing and cache optimization which lets the algorithm run quickly even on massive databases.

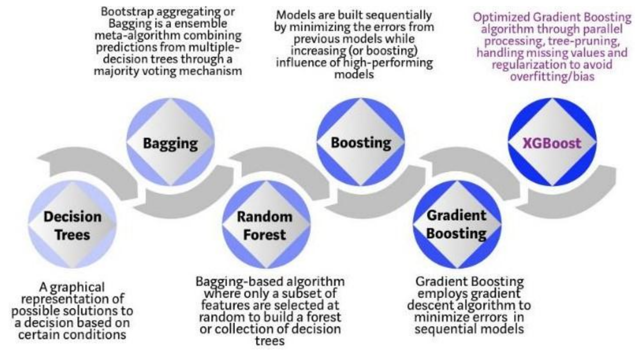

Bootstrap aggregating or Bagging is a ensemble meta-algorithm combining predictions from multiple-decision trees through a majority voting mechanism
Models are built sequentially by minimizing the errors from previous models while increasing (or boosting) influence of high-performing models
Optimized Gradient Boosting algorithm through parallel processing, tree-pruning, handling missing values and regularization to avoid overfitting/bias
Bagging
Boosting
XGBoost
Decision Trees
Random Forest
Gradient Boosting
A graphical representation of possible solutions to a decision based on certain conditions
Bagging-based algorithm where only a subset of features are selected at random to build a forest or collection of decision trees
Gradient Boosting employs gradient descent algorithm to minimize errors in sequential models

Fig -1: XGBoost

Algorithmic Enhancements:

a. Regularization: It uses 2 levels of filtering in the form of LASSO and Ridge algorithms to avoid having large complex datasets and overfitting curves.

b. Sparsity Awareness: Whenever XGBoost knows there is lack of data, it can fill it up appropriately and not with random numbers so that the training loss is kept to the minimum.

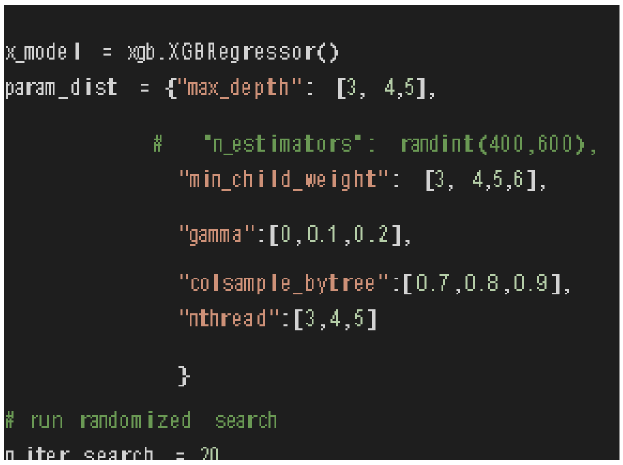
c. Weighted Quantile Sketch: Sometimes when the databases are weighted, XGBoost has the ability to divide the database in parts on its own according to the weights.

d. Cross-validation: In other algorithms we need to specify the number of times we have to run it or how long we need to run any algorithm when training, XGBoost has an inbuilt capacity to understand the same.

The implementation of the algorithm is such that it is economical in computation time as well as memory resources. Some key algorithm implementation features include:

- Sparse Aware implementation with automatic handling of missing, "NA', "None" data values.
- Block Structure to support the parallelization of tree construction to improve on the processing time.
- Continued Training so that you can boost a fitted model on new data to increase performance.

Code for the same is as follows:



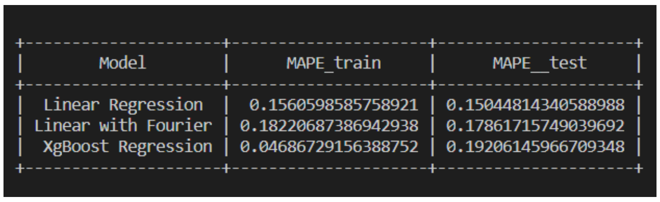
```
x_model  = xgb.XGBRegressor()
param_dist  = {"max_depth":  [3,  4,5],

                #   "n_estimators":  randint(400,600),
                  "min_child_weight":  [3,  4,5,6],

                  "gamma":[0,0.1,0.2],

                  "colsample_bytree":[0.7,0.8,0.9],
                  "nthread":[3,4,5]

                  }
# run randomized  search
n_iter_search  = 20
```

3. RESULTS

Due to lack of processing power, we were only able to calculate the accuracy using 580 rows of the database. When we calculate the Mean Absolute Percentage Error (MAPE) of each of the algorithm we can observe a fitting between the datapoints. We can see that the best fitting – where the training and testing data prove to have the closest MAPE – is found in Linear Regression model which proves it to be the preferred method for handing a dataset of this type.

$$\mathrm{M} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|,$$

The image below shows our findings



```
+---------------------+---------------------+---------------------+
|        Model        |      MAPE_train     |      MAPE__test     |
+---------------------+---------------------+---------------------+
|  Linear Regression  |  0.1560598585758921 | 0.15044814340588988 |
| Linear with Fourier | 0.18220687386942938 | 0.17861715749039692 |
|  XgBoost Regression | 0.04686729156388752 | 0.19206145966709348 |
+---------------------+---------------------+---------------------+
```

Fig -2: Accuracy measure

Note that the least error for the test data is also noted by the Linear Regression making it the most accurate too. If this was not the case, then it would just so happen that we would have to choose between reliability and higher accuracy. As in the figure it can also be observed that Even though XGBoost has had the best training performance with only 4% errors, it does not fare well in the testing phase while both Linear Regression methods hold their own with relative ease and also produce quite good fittings.

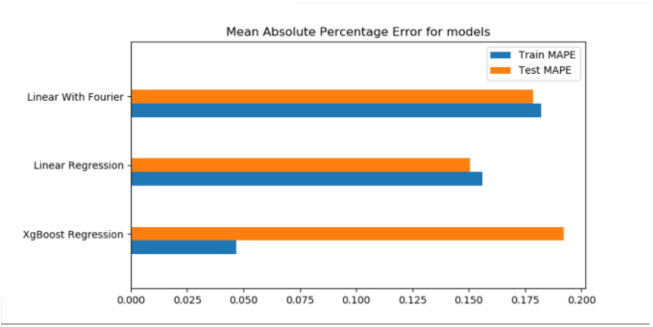


Chart -1: Accuracy for train and test sets

4. CONCLUSIONS

When intricately comparing normal Taxicabs with hailing Taxis from apps like Uber, the major differentiating factor is having the ability to see what you pay. The ability to be able to know beforehand what you will be paying for a taxi ride which is in particular a satisfying feeling for any human being, being able to save the most by choosing the cheapest option or the most heavily discounted option (As noted by Dan Ariely on many occasions). Here we look at the same and level the playing field for the traditional options. Interesting observations and future work can be based of this research as we might get a chance to observe if customers choose to have additional ease of use or the least price offered in the future. Using various classifiers, we have also determined which of them is the optimal classifier that can be used in this circumstance as we move towards a world shaped more and more by Machine Learning.

REFERENCES

[1] https://medium.com/analytics-vidhya/building-a-linear-regression- model-on-the-new-york-taxi-trip-duration-dataset-using-python-2857027c54f3

[2] https://medium.com/@ranasinghiitkgp/taxi-demand-prediction-in- new-york-city-916cde6a3492

[3] https://towardsdatascience.com/new-york-taxi-data-set-analysis- 7f3a9ad84850

[4] https://github.com/gauravtheP/Taxi-Demand-Prediction-New-York-City/

[5] http://web.mit.edu/ariely/www/MIT/Papers/interactivity.pdf

[6] https://pergamos.lib.uoa.gr/uoa/dl/frontend/file/lib/default/data/13245 73/theFile/1324574