PROJECT REPORT

ON

"NEW YORK TAXI FARE PREDICTION"

Submitted to

International Institute of Information Technology
Bangalore

BY

HARSH KUMAR	**MT2019508**
MD. ZAHID FAIZ	**MT2019514**
VIVEK GUPTA	**MT2019137**

Acknowledgements

We are profoundly grateful to **our Prof. G. Srinivasaraghvan , Prof. Neelam Sinha and TA Anuj Shah** for their expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion.

We would like to express deepest appreciation towards them because it helped us to understand how machine learning could be implemented in real time scenario.

At last we must express our sincere heartfelt gratitude to all our freinds,seniors etc who helped us directly or indirectly in clearing our doubts during this course of work.

HARSH KUMAR (MT2019508)
VIVEK GUPTA (MT2019137)
MD. ZAHID FAIZ (MT2019514)

ABSTRACT

Predicting taxi fare is definitely not as flourishing as predicting airline fares. However, since we do not currently have airline fare open data available.

In this task, we are going to predict the fare amount for a taxi ride in New York City, given the pick up, drop off locations and the date time of the pick up.

Contents

Chapter 1

Introduction

1.1 DEFINITION

1.1.1 PROJECT OVERVIEW

Making a prediction based on unknown data can be a deceptively difficult problem. What would be a simple calculation in hindsight, when you have the benefit of all the variables, often proves to be much more complicated when several variables cannot be known in advance.

Specifically, this project creates a submission to the New York Taxi Fare Prediction Kaggle competition. The goal is to accurately predict a rider's taxi fare using only factors that could be known when booking the ride, such as pickup and dropoff location but not unknowns like future traffic conditions or the route the driver is going to take.

1.1.2 PROBLEM STATEMENT

In this challenge we are given a training set of 55M Taxi trips in New York in the train data and 11M records in the test data.

The goal of this challenge is to predict the fare of a taxi trip given information about the pickup and drop off locations, the pickup date time and number of passengers travelling.

1.1.3 METRICS

The Kaggle competition uses the Root Mean Square Error (RMSE) as the evaluation metric, which seems to be a perfect match as it is similar to the Mean Absolute Error (MAE, which gives an average of all the errors without considering their direction), but it amplifies the magnitude of large errors. This favors model consistency as large errors, even if infrequent, will greatly increase RMSE.

MSE has similar characteristics to RMSE, but is not as human friendly, as the values are on on a different scale and thus not directly comparable. Meanwhile, an RMSE of 4.0 would give you an idea that most results are within $3-4 of the true price.

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{1.1}$$

Chapter 2

Analysis

2.1 DATA DESCRIPTION

In this project we are given 3 csv files, out of which one is train.csv which contains input features as well as the target fare_amount values for the training set and one is test.csv which consists of input features for the test set (about 10K rows). In addition to this there is sample_submission.csv which a sample submission file in the correct format (columns key and fare_amount). This file 'predicts' fare_amount to be $11.35 for all rows, which is the mean fare_amount from the training set.

Data fields

ID

- key - Unique string identifying each row in both the training and test sets. Comprised of pickup_datetime plus a unique integer, but this doesn't matter, it should just be used as a unique ID field. Required in your submission CSV. Not necessarily needed in the training set, but could be useful to simulate a 'submission file' while doing cross-validation within the training set.

Features

- pickup_datetime - timestamp value indicating when the taxi ride started.
- pickup_longitude - float for longitude coordinate of where the taxi ride started.
- pickup_latitude - float for latitude coordinate of where the taxi ride started.
- dropoff_longitude - float for longitude coordinate of where the taxi ride ended.
- dropoff_latitude - float for latitude coordinate of where the taxi ride ended.

- passenger_count - integer indicating the number of passengers in the taxi ride.

Target

- fare_amount - float dollar amount of the cost of the taxi ride. This value is only in the training set; this is what you are predicting in the test set and it is required in your submission CSV.

2.2 DATA EXPLORATION AND VISUALISATION

The train dataset is provided for the Kaggle competition is a 55 million row subset taken from the total 1,108,779,463 cab trips nyc-tlc dataset publicly hosted on kaggle Within the Kaggle subset, are columns for all the information you'd expect to receive from a rider:

- pickup and dropoff location (separate longitude and latitude columns with floats)
- date/time (timestamp)
- number of people (integer)

ID column was created with a key value (string) consisting of the timestamp and an integer for uniqueness, and the train set contains the fare_amount as that's that target value to predict

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
count	2.000000e+06	2.000000e+06	2.000000e+06	1.999988e+06	1.999988e+06	2.000000e+06
mean	1.133507e+01	-7.249770e+01	3.991464e+01	-7.249619e+01	3.991103e+01	1.684338e+00
std	9.783112e+00	1.228673e+01	8.380023e+00	1.206387e+01	9.842548e+00	1.334463e+00
min	-1.000000e+02	-3.313387e+03	-3.458665e+03	-3.412653e+03	-3.461541e+03	0.000000e+00
25%	6.000000e+00	-7.399208e+01	4.073490e+01	-7.399140e+01	4.073406e+01	1.000000e+00
50%	8.500000e+00	-7.398183e+01	4.075262e+01	-7.398016e+01	4.075315e+01	1.000000e+00
75%	1.250000e+01	-7.396710e+01	4.076709e+01	-7.396368e+01	4.076812e+01	2.000000e+00
max	4.500000e+02	2.857840e+03	3.310364e+03	3.457622e+03	3.345917e+03	2.080000e+02

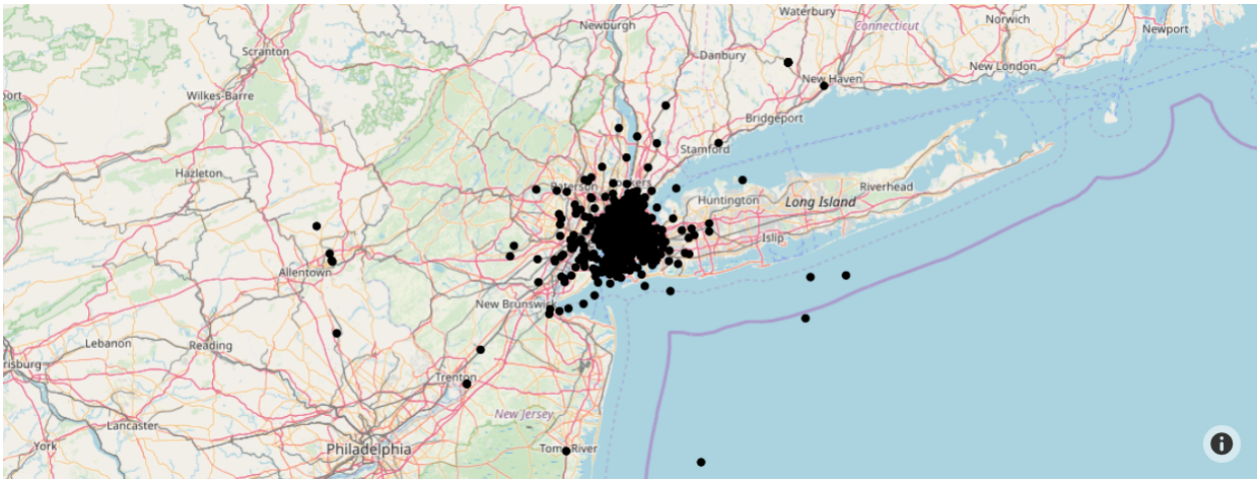
A quick summary of the train dataset's statistics reveals some questionable numbers. Within the first 2M rows, there are fares as low as -$62, which could be due to refunds or just erroneous, but I don't suspect the test dataset contains refunds or otherwise negative fares. According to the TLC , they charge an initial $2.50 for every trip, so there shouldn't be any fares under that amount, and all records below that threshold were removed. Next, there are some trips with 0 passengers, which might be due to a delivery or something, but since the test data doesn't contain trips with 0 passengers, those offending records were also removed. Finally, from these latitudes and longitudes, you'd think that New York city spanned half the globe; I don't think the big apple is quite that big.

A lot of them just have 0s for coordinates or other meaningless values, but a few (955 out of the first 2M records) contained coordinates that completely (both pickup and dropoff) fell within an inverse of this bounding box, which is geographically located in Antarctica and probably doesn't have many taxi cabs. Considering that these data points otherwise seem legitimate and could simply be due to a misconfigured data logger or an import error into the main dataset, we'll reverse the ordering on these coordinates and re-incorporate them back into the training data.

To fix this, a mask was created of the same section of the NYC map, with all visible water manually removed, then using the mask, all data-points within a water section were dropped from the dataset.

The next step is to calculate the straight line distance in kilometers between the pickup and dropoff points using the Haversine formula to account for the spherical nature of the earth (deal with it flat-earthers), which is a bigger factor at longer distances, and add it as a column on the dataset. While straight-line is not nearly as useful as the actual route the taxi took, it wouldn't be feasible for us to predict that without using a 3rd party tool, which is not an option for this project. This should result in much greater route variation for shorter rides due to the extra distances added from turns accounting for a larger proportion of the total route distance, but the straight distance is still a very helpful indicator here.

Also notice that a good number of the data points in the train set are at or near 0 km distances, so we'll filter out the data into very short distances (less than 0.1 km) and longer distances. Looking at the statistics, nothing sticks out about these points that indicates a pattern useful to correlate them. While that data could be accurate, it's hard to imagine how these data-points could be useful under these circumstances. But also notice that the test dataset also contains data in this category, which is slightly concerning, since so far the test set has seemed to contain perfect data. However, this seems unavoidable, so we'll just remove them from the training data and hope the model prediction gets close enough based on the other inputs in the test data or in real world usage Also, since the test data's validity is in question, we also check it for points in the water and find 2 out of the 9914, which isn't too bad.



Other data exploration was done,including charting the density of pickups/dropoffs on the map and many other things which were helpful for understanding the data on a deeper level, they don't represent any transformations that we should make to the data, but rather they signal factors that we hope the model will pick up on and work into its calculations without us having to hard code into our data. The timestamp was also split off into several columns: date , month and year ; but no filtering of the data was performed based on these factors since they all seem like valid values (and should be to have been a recognized timestamp). Similarly, while this time column certainly equate to deviation in fares, due to different rates at night/day/rush-hour and rate increases over the years, these are the types of things that we want our estimator to correlate from the data and factor into its predictions.

Chapter 3

Methodology

3.1 Data Preprocessing

Much of the data preprocessing was performed in conjunction with the data exploration and discussed in detail above, but that was only done on a small subset (the first 2 million rows) of the total dataset due to memory constraints (the full 55M row CSV is over 4.7GB, but it'd be much larger as a Pandas DataFrame held in memory).

Explore the test data It is very important to explore the test data before the train data, as we can decide the cleaning metric for the train data base on the data range in the test data.

```
pickup_datetime       0
pickup_longitude      0
pickup_latitude       0
dropoff_longitude    82
dropoff_latitude     82
passenger_count       0
dtype: int64
pickup_datetime           0
pickup_longitude     211541
pickup_latitude      210847
dropoff_longitude    210909
dropoff_latitude     210328
passenger_count       39005
dtype: int64
```

There are some null and zero values in the train data.

Explore the train data It is very important to explore the train data before the training model, as we can decide the cleaning metric for the train data based on the data range in the train data.

```
fare_amount          0
pickup_datetime      0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    12
dropoff_latitude     12
passenger_count      0
dtype: int64
fare_amount             66
pickup_datetime          0
pickup_longitude     38295
pickup_latitude      38167
dropoff_longitude    38338
dropoff_latitude     38237
passenger_count       7047
dtype: int64
```

There are some null and zero values in the train data. This step is very import, as these values can influence the training result significantly

In the end it wasn't feasible to load the entire train dataset into memory to process all at once and the preprocessing needed to be systematized into something that is nearly ready for deploying into a production model.

So every step that transformed the data in some way was converted into a function or a series of functions. This also organized the various steps of preprocessing so that if when it came time to create the production model, the functions needed to format the user input data would be easy to find and copy.

First we need to remove the outliers could be :

1. Some values in passenger count is more then 12 which is not possible, some values are negative which is also not possible.
2. Also some are zero this means that they could be carrying lagguage.
3. Some values in fare count is also negative which is also physically not possible.
4. Some fare are zero even if the pickup location is not zero. It could mean that the customer cancelled the cab after it has reached the loacation .
5. And the most important outliers are the ones where the pickup and dropoff locations are wayy too outside NewYork, which may be the result of the incorrect data input

Other then that there are some fare values which are above 400 but this could be because that could be fix price of the cabs.

For removing the outliers we will be doing these things :

1. Remove the passenger count above 12
2. Keep the zero passenger count rows as they contain the fare amount but drop the negative values
3. Drop the negative fare amount rows
4. And we will drop the values which are way out of NewYork (who takes cabs to travel to other continents)

Taxi fares for NYC according to its government is

- $2.50 initial charge.
- Plus 50 cents per 1/5 mile when traveling above 12mph or per 60 seconds in slow traffic or when the vehicle is stopped.
- Plus 50 cents MTA State Surcharge for all trips that end in New York City or Nassau, Suffolk, Westchester, Rockland, Dutchess, Orange or Putnam Counties.
- Plus 30 cents Improvement Surcharge.
- Plus 50 cents overnight surcharge 8pm to 6am.
- Plus $1.00 rush hour surcharge from 4pm to 8pm on weekdays, excluding holidays.
- Plus New York State Congestion Surcharge of $2.50 (Yellow Taxi) or $2.75 (Green Taxi and FHV) or 75 cents (any shared ride) for all trips that begin, end or pass through Manhattan south of 96th Street.
- Plus tips and any tolls.

New York Coordinates

- min_lat - 40.5774
- max_lat - 40.9176,
- min_long - -74.15,
- max_long - -73.7004

Passenger count cannot be more then 12 considering everyone as kid, still more then 12 is physically impossible

3.2 Model

We've used Random Forest Regressor model as it is highly effective algorithm for our dataset.A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

Random Forest Regressor has a hyperparameter - "warm_state" which is useful in our methodology of training the model in chunks . And actually it had the least RMSE score among the others. So, we applied this algorithm.

We also tuned some of the hyperparameters to have a better model. The pruning is required for better predictions and random growing trees and not good. The hypreparameters we played around with are : n_estimators : This the number of trees in our forest.

- We worked with small data and plotted one graph between the RMSE values and the n_estimators The range in which we got the least RMSE is in our model which is in between 60-70.
- We then pruned the leaves using the same technique as above and changed the value of max_features . The value we got is for max_features = 0.5 . Means that the minimum number of sample leaves required to be a leaf node is 0.5
- Other then that we tested the depth of the trees and the default value worked fine for us.
- The number of features to consider when looking for the best split should never be over 0.5 . But plotting the graph showed us that the best value for our model is coming more than 0.5 . So we took the last value whch had least RMSE that is 0.45 .
- We also worked with the randomly selecting samples by bootstrapping our data set . And it was also giving some off guesses in Fare_amount. So we worked with whole data set without bootstarp.

After then we've used the pickle library to store our trained model to a pickle file named model.pkl.

Chapter 4

Conclusion and Future Scope

4.1 Result

The best RMSE score that we got after training the model is 8.8.

4.2 Conclusion

The map displays just how concentrated most of the data is, which makes sense since the hot areas pretty closely match up with NYC city limits, but keep in mind that the TLC serves a much wider region, just with much less demand in the other parts of the metro area. Also, notice the scale on the side, which is log(1 + of datapoints), meaning that it takes around 100-1000 rides within a given square KM to start making any kind of visible impact on this map, with the noticeable areas being on the order of 10,000-1M rides per KM . Furthermore, note that this was pulled from just the first 2 million records in the train data, and that these bounds were chosen due to the test dataset containing datapoints as far reaching as the borders of this map. In reality,the entire 55M train dataset contained at least a few datapoints from just about all valid areas of this map.The fractions of zero values in the hidden layer.

4.3 Future Scope

Although we tried a number of different techniques and implemented a complete solution, there are still steps to take that can improve the model. The things to work on are :

- Considering the airport fare as separate feature.

- Using any boosting algorithm like Adaboost or XGboost in random forest would have made our model more accurate.