International Institute of Information Technology, Bangalore

GEN - 511 Machine Learning Project

NEW YORK TAXI FARE PREDICTION

SHUBHAM KUMAR MT2019109
SHUBHAM LAMICHANE MT2019110
SUMIT RANA MT2019118

November 30, 2019

Abstract

In this project we wanted to predict the taxi fare for a ride in New York City. Taxi services has been around for many decades and in recent times a lot of new companies like Uber, Ola have come up with their improved digital services. Also the fact that most of the fares have had receipt attached with them, the collection of data in this case became easier. By making a prediction about the expected fare from one location to another, the companies help consumers to manage and plan their travels much more efficiently. So if a model can be created which predicts the fare taking into account various parameters like location and other constraints, we can surely make the life of common people much easier.

Introduction

In this challenge we were given 55 million taxi trips collected from the year 2009 as a training data and 11 million test data. We are given information about pick-up location, drop-off location, time and date of travel, passenger count and the fare that has been paid. 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.

Dataset Description

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

Hypothesis generation

The next step to solve any analytics problems is to list down a set of hypothesis, which in our case are factors that will affect the cost of a taxi trip.

1. Trip distance : If the distance to be traveled is more, then fare should be higher.

2. Time of Travel : During peak traffic hours, the taxi fare may be higher.

3. Day of Travel : Fare amount may differ on weekdays and weekends

4. Is it a trip to/from the airport : Trips to/from the airport generally have a fixed fare.

5. Pickup or Drop-off Neighborhood : Fare may be different based on the kind of neighborhood.

6. Availability of taxi : If a particular location has a lot of cabs available, the fares may be lower.

Data Exploration

For exploratory data analysis these are our considerations -

1. Does the number of passengers affects the fare?
2. Does the date and time of pickup affect the fare?
3. Does the day of the week affect the fare?
4. Does the distance travelled affect the fare?

First, let's split the datetime field 'pickup_datetime' to the following -

- year
- month
- date
- hour
- day of week

Using these we shall calculate the day of the week and come to our conclusions about how pickup_location affects the fare. Also, create a new field 'distance' to fetch the distance between the pickup and the drop.

We can calculate the distance in a sphere when latitudes and longitudes are given by Haversine formula

haversine(θ) = $\sin^2(\theta/2)$

Eventually, the formula boils down to the following where φ is latitude, λ is longitude, R is earth's radius (mean radius = 6,371km) to include latitude and longitude coordinates (A and B in this case).

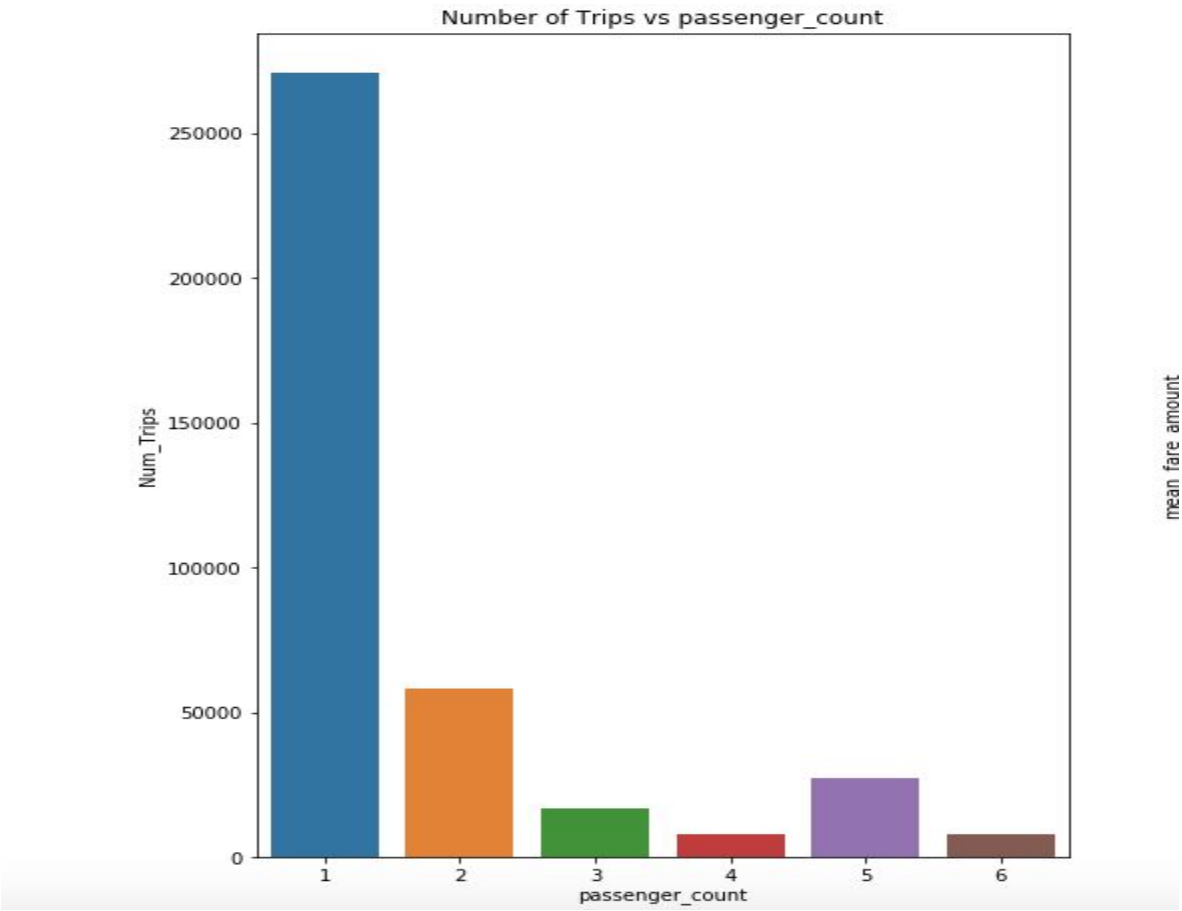
$a = \sin^2((\varphi B - \varphi A)/2) + \cos \varphi A \,.\, \cos \varphi B \,.\, \sin^2((\lambda B - \lambda A)/2)$

c = 2 * atan2(√a, √(1−a))

d = R · c

d = Haversine distance

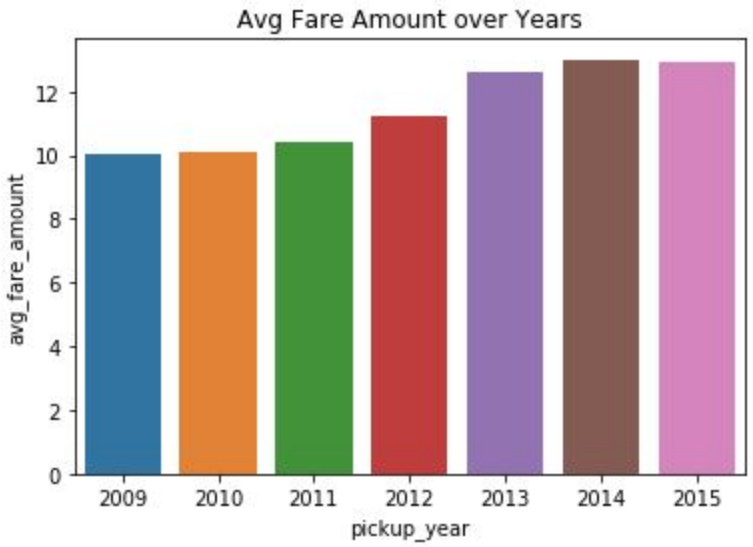
Does the number of passengers affects the fare?



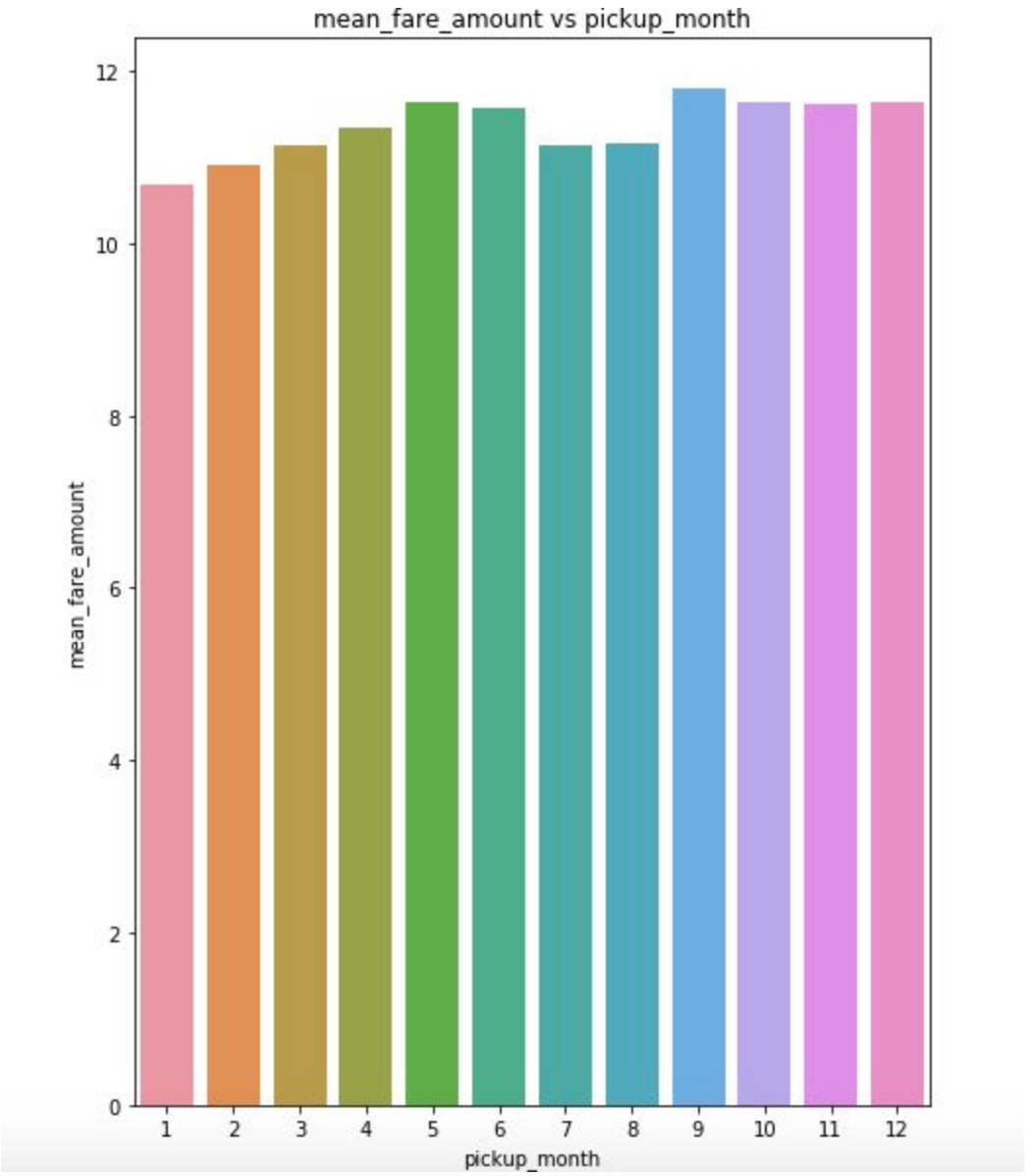
From the above graphs we can see that single passengers are the most frequent travellers, and the highest fare also seems to come from cabs which carry just 1 passenger.

Distribution of Pickup date time

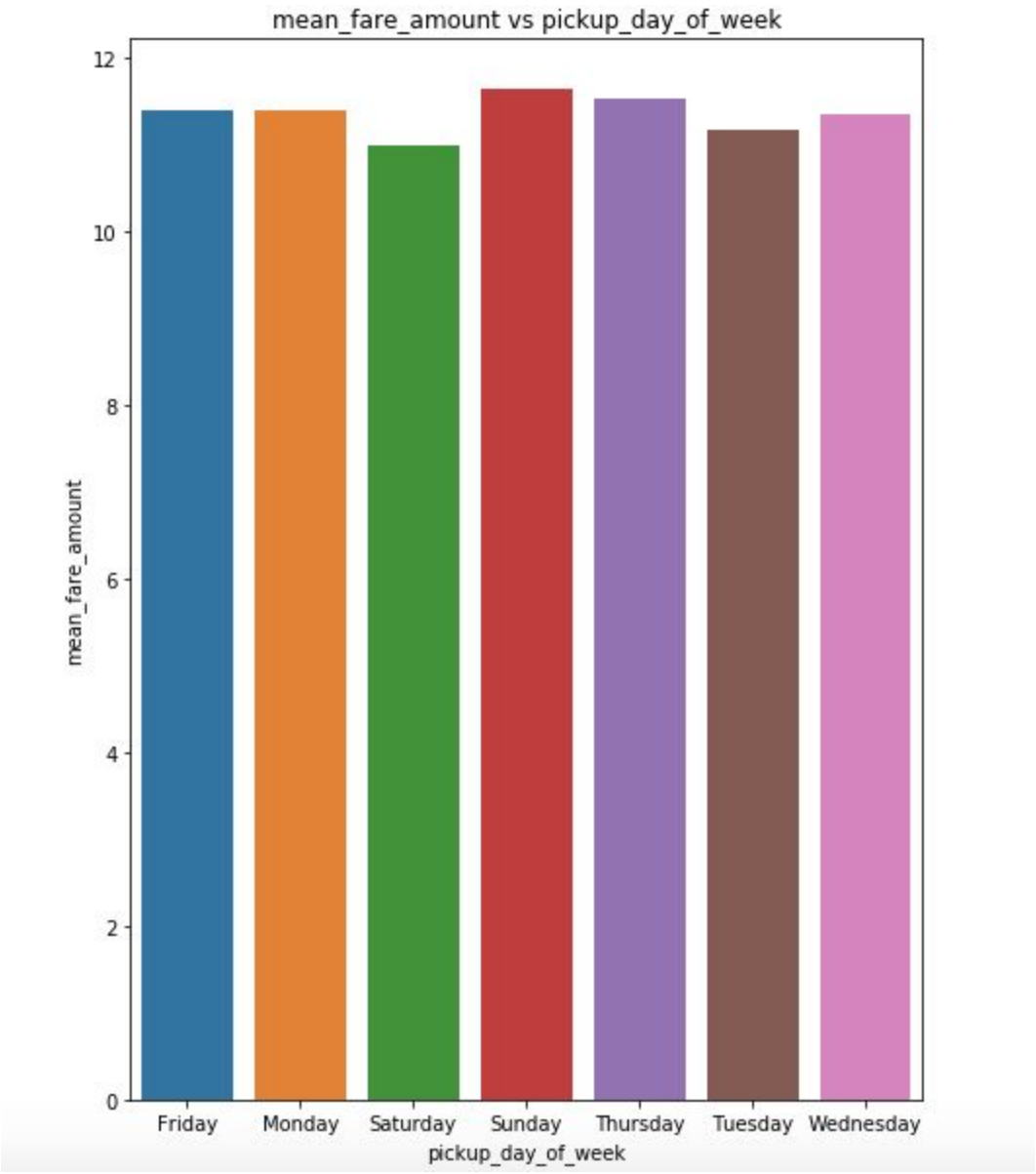
The first step to analyse how the fares have changed over time, is to create features like hour, day of the week, day, month, year from pickup datetime.



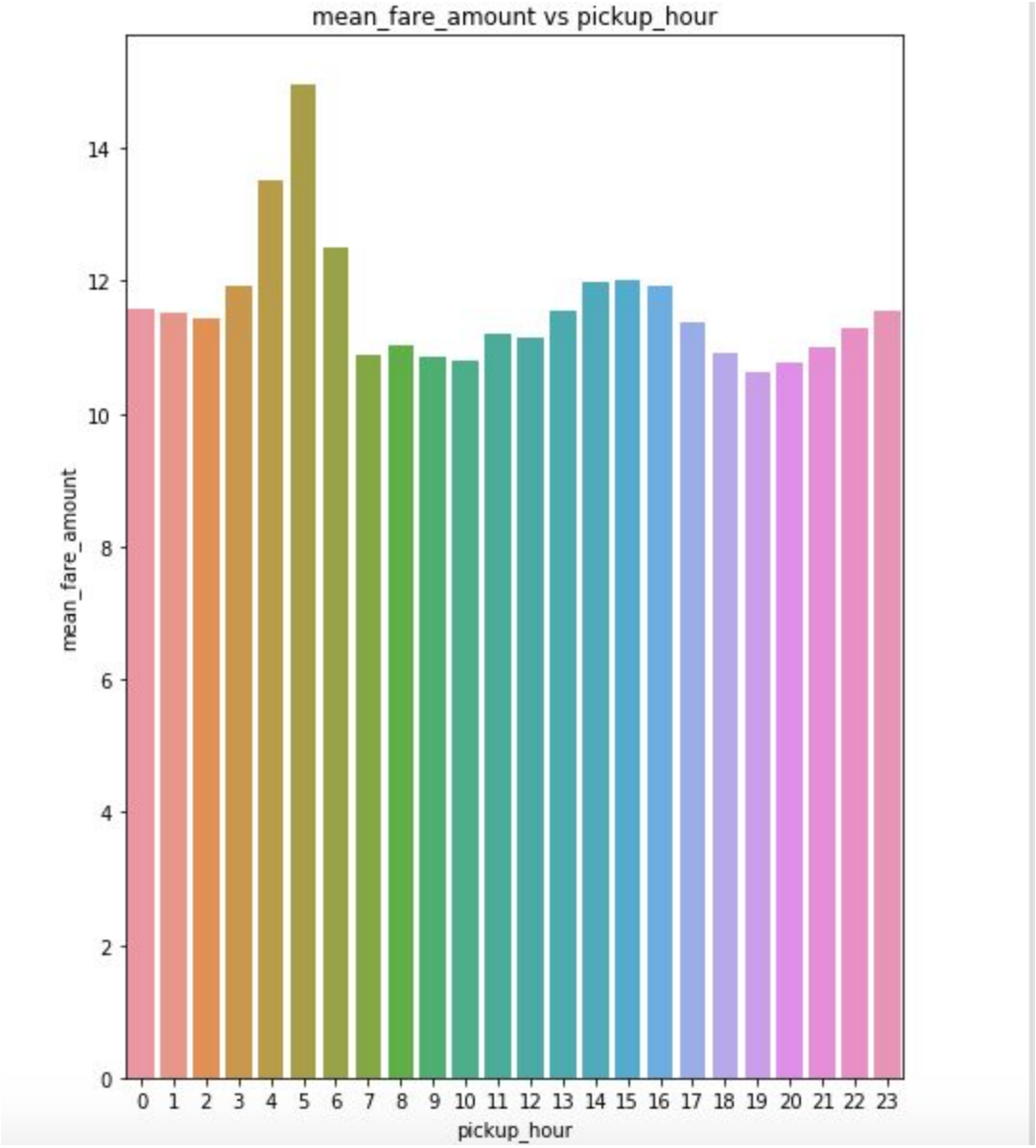
As expected, over the years the average taxi fare has increased.



Over months, though there have been fewer pickups from July to December, the average fare is almost constant across months.



We observed that though the number of pickups are higher on Saturday, the average fare amount is lower. On Sunday and Monday, though the number of trips are lower, avg fare amount is higher.



The average fare amount at 5 am is the highest while the number of trips at 5 am are the least. This is because, at 5 AM 83% of the trips are to the airport.The number of trips are highest in 18 and 19 hours.

Roadmap

After the Data Cleaning and Exploratory Analysis phase, we have finally arrived at the Model Building phase. The quality of results at the end of this phase depends on the data quality and features used for modelling. In this article, we are going to understand the below steps in detail:

1. Data Preparation for model building
2. Creating a baseline prediction
3. Building Models without Feature Engineering
4. Building Models with Feature Engineering

Data Preparation

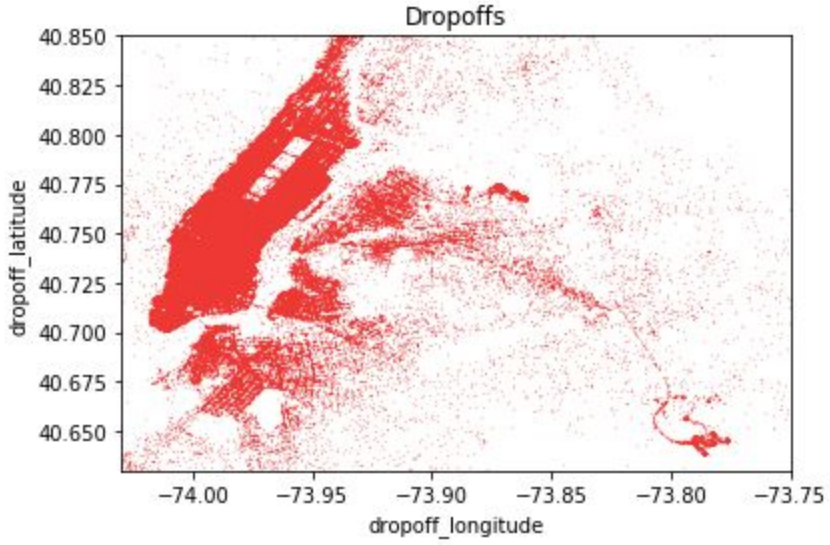
This step involves cleaning the data, dropping unwanted columns, converting the categorical variables to machine-understandable format, and finally splitting of training data into training and validation sets.

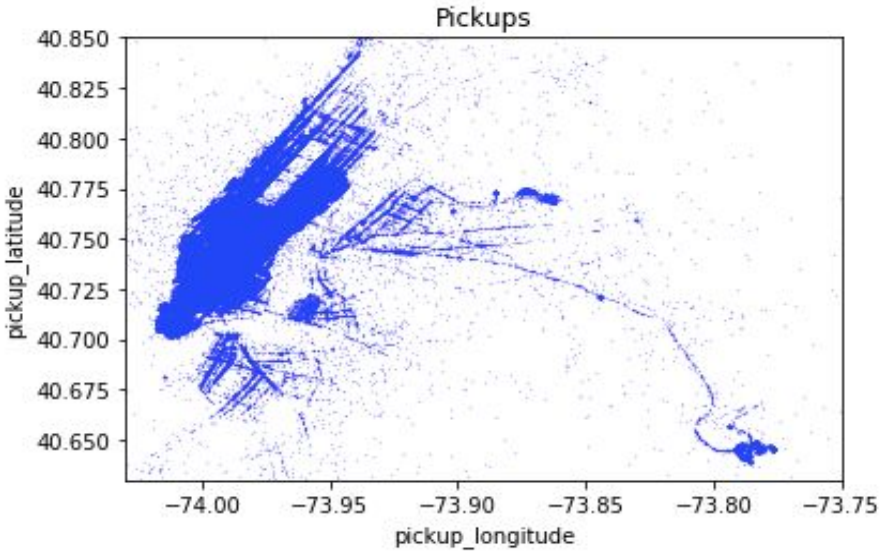
Since the dataset was huge, removing corrupted rows wasn't a problem. On the other hand, replacing it with some value could have resulted in false positive predictions.

We removed all negative fare amount and fare greater than $500(which is not possible within New York City), and passenger counts less than 0 and greater than 8.

Also few of the pick-up and drop-off locations were absurd, like in different countries etc. So we also removed all the latitudes and longitudes that were beyond New York City and its outskirts. We restricted the latitudes from

(40.57, 41.73) and longitudes from (-72.98, -74.26) as these are the coordinates of New York City. The distribution of pickup and dropoff locations are given below:





The date-time attribute which was given was not really helpful in its original form. So we parsed the data-time and converted it into several parameters like day of the week, month, year, hour of the travel etc. This way we could get more correlation with these attributes and the fare column.

Baseline Model

A baseline model is a solution to a problem without applying any machine learning techniques. **Any model we build must improve upon this solution.** Some ways of building a baseline model are taking the most common value in case of classification, and calculating the average in a regression problem. In this analysis, since we are predicting fare amount (which is a quantitative variable)— we will predict the average fare amount. **This resulted in an RMSE of 12.52333. So any model we build should have an RMSE lower than 12.52333.**

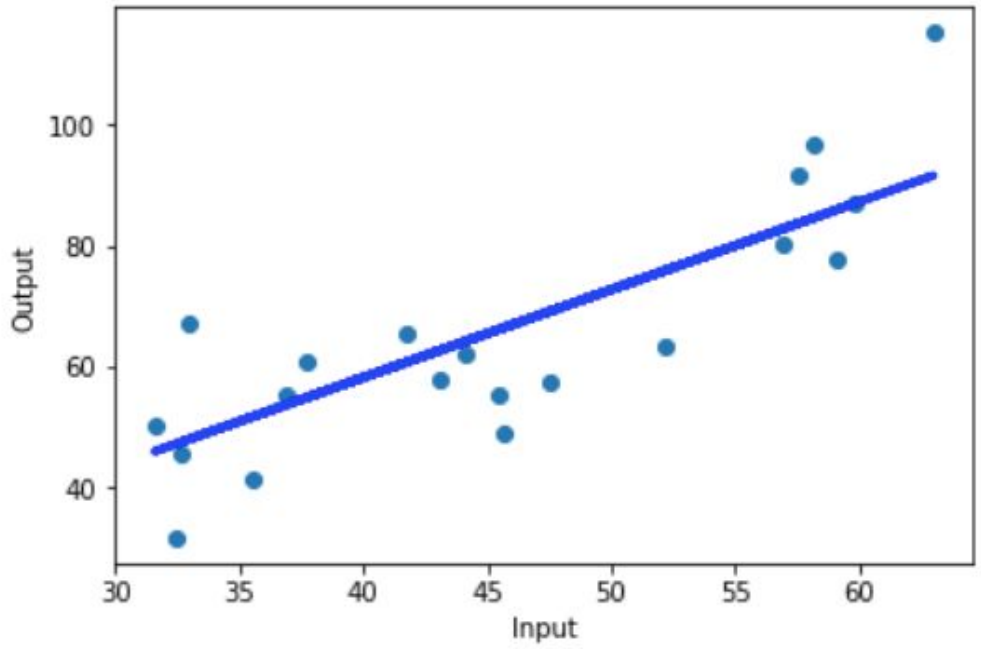
Model Building and Evaluation

To understand and evaluate the models, we will consider the following ML algorithms:

- Linear Regression
- Random Forest
- Light GBM

Linear Regression:

It is used to find a linear relationship between the target and one or more predictors. The main idea is to identify a line that best fits the data. The best fit line is the one for which the prediction error is the least. This algorithm is not very flexible, and has a very high bias. Linear Regression is also highly susceptible to outliers as it tries to minimize the sum of squared errors.




```
lm = LinearRegression()
lm.fit(X_train,y_train)
y_pred=np.round(lm.predict(X_test),2)
lm_rmse=np.sqrt(mean_squared_error(y_pred, y_test))
```

The test RMSE for Linear Regression model was **9.81667**, and the training RMSE was **9.81667**. This model was an improvement on the baseline prediction. Still, the error rate was very high in this model. So we moved to more complex model next.

Random Forest:

Random Forest is far more flexible than a Linear Regression model. This means lower bias, and it can fit the data better. Complex models can often memorize the underlying data and hence will not generalize well. Parameter tuning is used to avoid this problem.

```
rf = RandomForestRegressor(n_estimators = 100, random_state = 883,n_jobs=-1)
rf.fit(X_train,y_train)
```

The test RMSE for **Random Forest** model was **7.3312**, and the training RMSE was **5.2156**. This model is an improvement on the baseline prediction.

LightGBM:

LightGBM is a boosting tree based algorithm. The difference between Light GBM and other tree-based algorithms, is that Light GBM grows leaf-wise instead of level-wise. This algorithm chooses the node which will result in maximum delta loss to split. Light GBM is very fast, takes quite less RAM to run, and focuses on the accuracy of the result.

Parameters we used in lgbm:

```
params = {
        'boosting_type':'gbdt',
```

```
        'objective': 'regression',
        'nthread': 4,
        'num_leaves': 31,
        'learning_rate': 0.05,
        'max_depth': -1,

       'subsample': 0.8,
        'bagging_fraction' : 1,
        'max_bin' : 5000 ,
        'bagging_freq': 20,
        'colsample_bytree': 0.6,
        'metric': 'rmse',
        'min_split_gain': 0.5,
        'min_child_weight': 1,
        'min_child_samples': 10,
        'scale_pos_weight':1,
        'zero_as_missing': True,
        'seed':0,
        'num_rounds':50000
    }
```

GBDT is an ensemble of decision trees trained in a sequence where the errors from the previously trained tree are added to the new decision tree in the next decision tree in the next iteration. This means every subsequent learner will try to learn the difference between the actual output and the predictions.

The metric we used for error detection here was root mean squared error. The test RMSE for **LGBM** model was **5.73434**, and the training RMSE was **3.5321**. This model is an improvement on the baseline prediction.

Conclusion

We started with exploratory data analysis trying to find attributes that best fits the data. We found out some new attributes which had very high correlation with the final output. Taking those attributes we tried fitting three models to our data: linear regression, random forest and lgbm. LGBM did the best with rmse of 5.73 on test data.