NYC Taxi Trip Time Prediction by Regression Analysis

Debanjan Ganguly

Data Science Trainee

AlmaBetter, Bengaluru

CONTENTS

Introduction:

Most of the ride service providing company faces a common problem of efficiently assigning the rides to passengers so that the service is smooth and hassle free. One of main issue is determining the duration of the current trip so it can predict when the cab will be free for the next trip.

It is important to predict how long a driver will have his taxi occupied. If a dispatcher or system got estimates about the taxi driver's current ride time, they could better recognize which driver to allocate for each pickup request which results to be less waiting time which means less cancellation from client side and increase in profit margin, as well as customer base.

Machine learning has been of significant help as it has helped businesses in abundant ways. We will use Machine Learning to efficiently build a model with a real world dataset of Yellow Taxi Service of NYC which will predict the estimated time duration of a tax trip for a given Pick up location, Drop location, Date, and Time.

Problem Statement:

Our task is to build a model that predicts the total ride duration of taxi trips in New York City. Our primary dataset is one released by the NYC Taxi and Limousine Commission, which includes pickup time, geo-coordinates, number of passengers, and several other variables.

Dataset Analysis:

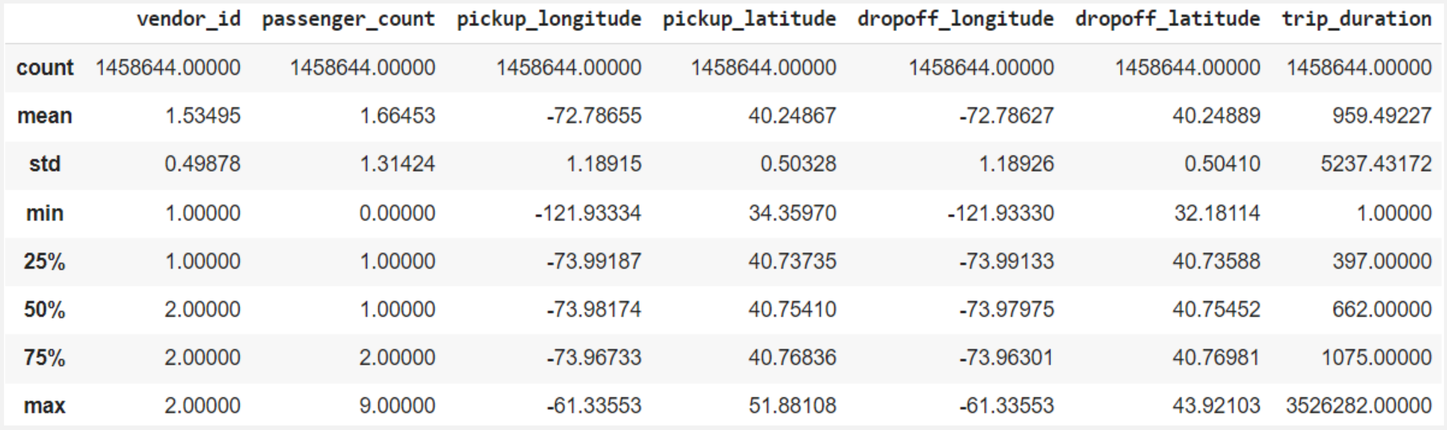
The dataset contains 1458644 observations with 11 features. It contains features to find out location pickup and drop off points, timestamp of trip events, passenger traveled with, necessary metrics to make predictions and draw conclusions. Let us look through our features,

- Id: a unique identifier for each trip
- vendor_id: code representing the provider associated with trip
- pickup_datetime: - date and time when the meter was engaged
- dropoff_datetime: date and time when the meter was disengaged
- passenger_count: the number of passengers in the vehicle (driver entered value)
- pickup_latitude: the latitude where meter was engaged
- pickup_longitude: the latitude where meter was engaged
- dropoff_latitude: the latitude where meter was disengaged
- dropoff_longitude: the latitude where meter was disengaged
- store_and_fwd_flag: flag indicates whether the trip record was held in vehicle memory before sending to the vendor because the vehicle did not have a connection to the server - Y=store and forward; N=not a store and forward trip
- trip_duration: duration of trip in seconds (target variable)

Observations about data are,

- id is nominal as well as vendor_id, but we will check if vendor_id is of any use as a feature as it contains only values i.e., 1, 2
- pickup/dropoff_latitude and pickup_dropoff_longitude have represented a co-ordinate
- store_and_fwd_flag is categorical column with Y & N as values
- There is pickup/dropoff_datetime column which we've to convert in datetime format

The distribution of numerical columns are as follows,



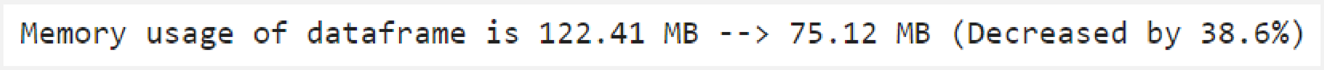
	vendor_id	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	trip_duration
count	1458644.00000	1458644.00000	1458644.00000	1458644.00000	1458644.00000	1458644.00000	1458644.00000
mean	1.53495	1.66453	-72.78655	40.24867	-72.78627	40.24889	959.49227
std	0.49878	1.31424	1.18915	0.50328	1.18926	0.50410	5237.43172
min	1.00000	0.00000	-121.93334	34.35970	-121.93330	32.18114	1.00000
25%	1.00000	1.00000	-73.99187	40.73735	-73.99133	40.73588	397.00000
50%	2.00000	1.00000	-73.98174	40.75410	-73.97975	40.75452	662.00000
75%	2.00000	2.00000	-73.96733	40.76836	-73.96301	40.76981	1075.00000
max	2.00000	9.00000	-61.33553	51.88108	-61.33553	43.92103	3526282.00000

Fig 1. Statistical Distribution of Numerical Features

From statistical distribution of fig 1 we can observe,

- Trip duration with a range of 1 second and ~979 hrs., there must've some extreme outliers in data.
- From the range of co-ordinates, there is plenty of outstation trips
- Though there is max no of 9 people in any trip, but 1 or 2 is the avg passenger count in most of the trips

There are no missing or duplicated values present in our dataset. Also, to reduce memory usage a function has been applied over dataset which converted normal integer and floats to numpy.int or numpy.float32. After applying, we've reduced 38.6% memory usage.



```
Memory usage of dataframe is 122.41 MB --> 75.12 MB (Decreased by 38.6%)
```

Exploratory Data Analysis:

We will perform EDA to discover key understandings such as:

- What can we learn from the co-ordinates? Is there any specific area with high demand of ride service?
- What can we learn from predictions? (ex: locations, prices, reviews)
- Which hosts are the busiest and why?
- Is there any noticeable difference of traffic among different areas and what could be the reason for it?

We will plot different graphs and perform univariate and bi variate analysis. First, we will observe the distributions of numerical variables.

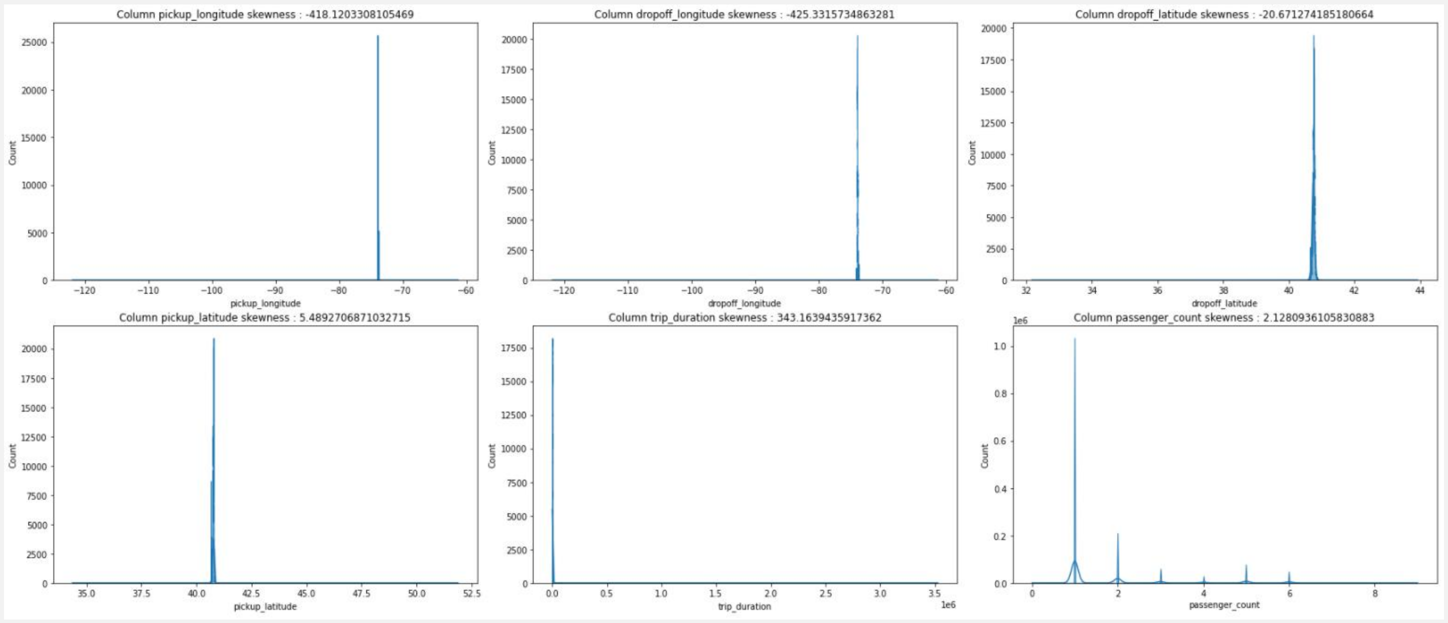


Fig 2. Distribution plot of Numerical Features

From fig 2, we can observe target variable has extreme positive skewness, so we will treat it with log transformation.

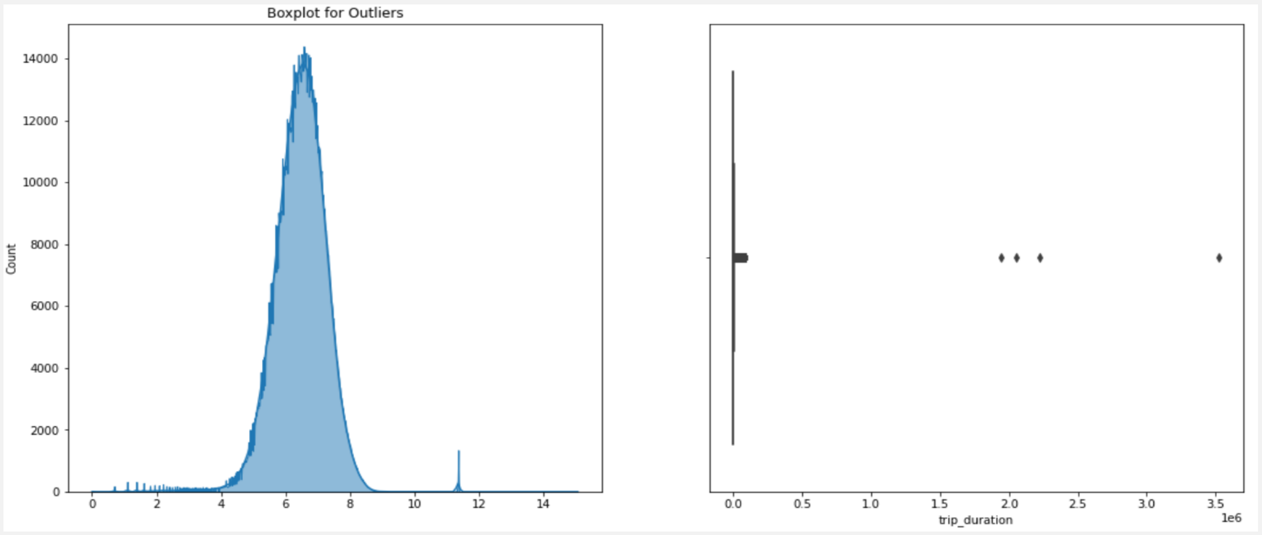


Fig 3. Distribution with log(trip_duration) & boxplot for outliers

From the above fig 3, it looks like a normal distribution on log scale with most of the trips between ~54 sec (exp 4) and 82 mins (exp 8). We can notice four records which clearly no way looks similar to other records.

As previously we've noticed the co-ordinates of pickup and dropoff locations are clearly from a particular range, so we considered 1st & 3rd quantiles of data to plot the below graph with ±0.10 noise.

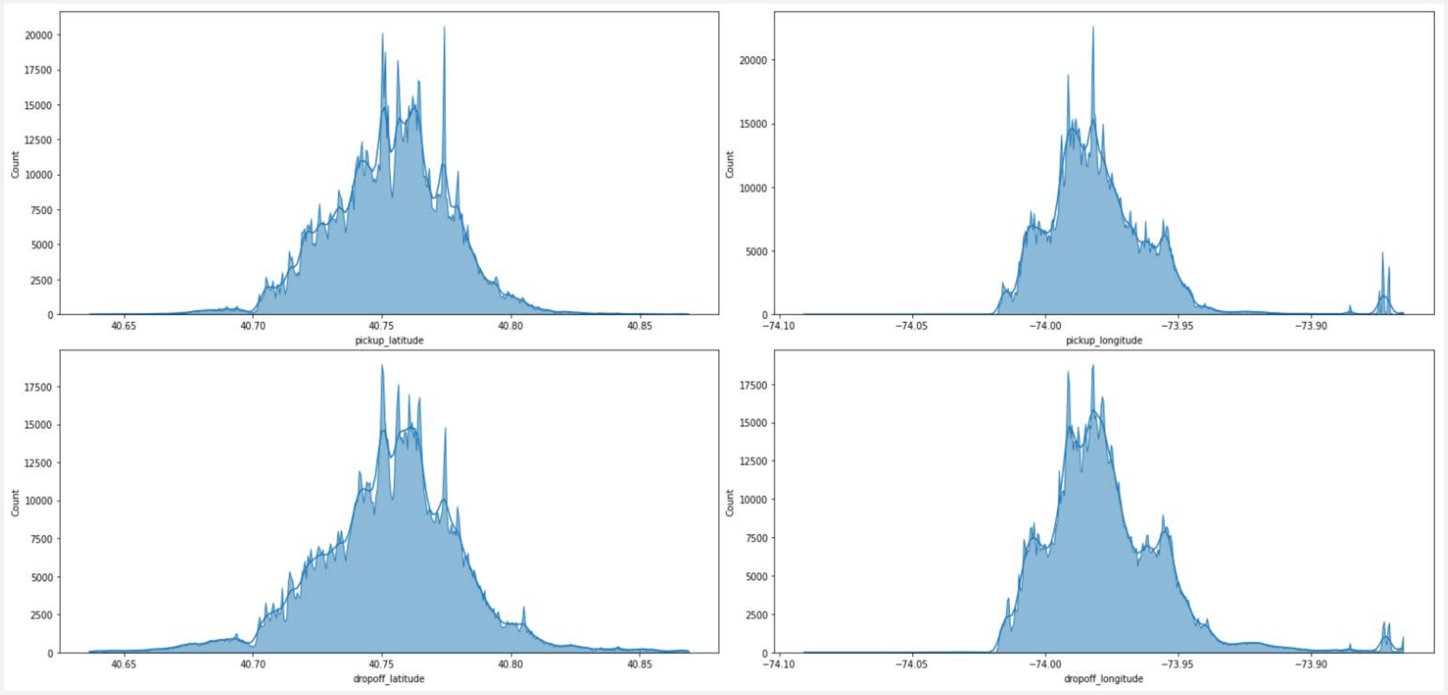


Fig 4. Distribution of pickup/dropoff co-ordinates

We can notice from fig 4 that pick and drop latitude are centered between 40 - 41 and longitude are situated around -74 to -73. Trips which are significantly far from each other within current range, have affected this plot such that it is coming off as a spike.

For Better visualization on traffic routes of rides, we've plotted spatial density graph of pickup and dropoff location which looked similarly as a satellite image. The brightest area with higher density is Manhattan. The graph shows most of the trips originated from Manhattan area, also there are some outstation trips too. Also, we can notice an uneven distribution of sources and destinations.

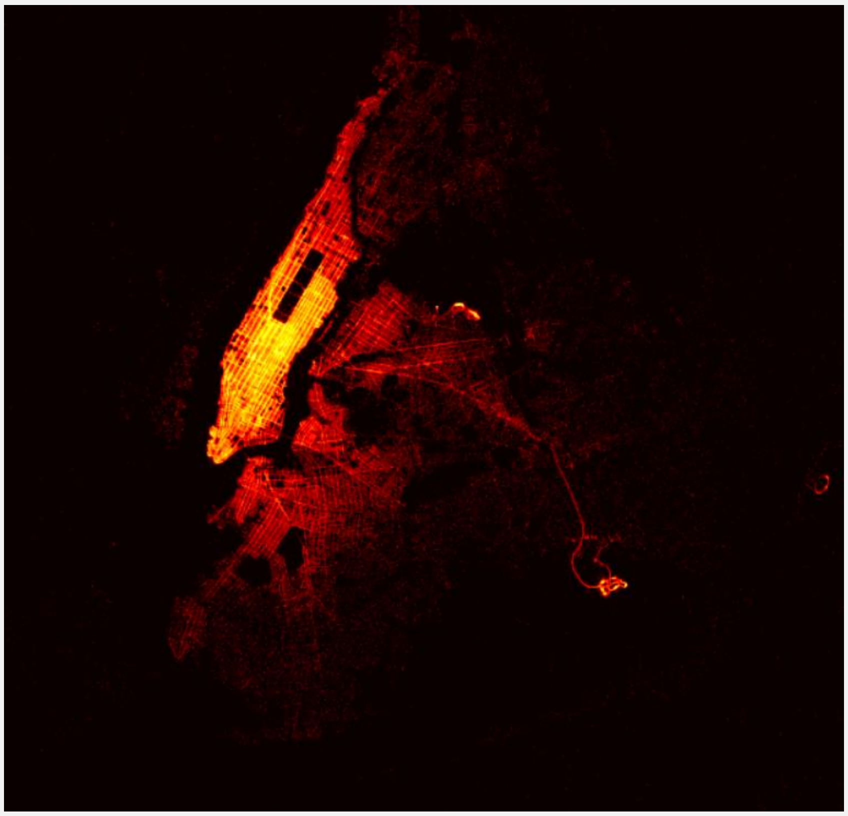


Fig 5. Spatial Density Graph of Location

Now we will perform bivariate analysis against other features, vendor_id, store_and_fwd_flag & passenger_count respectively.

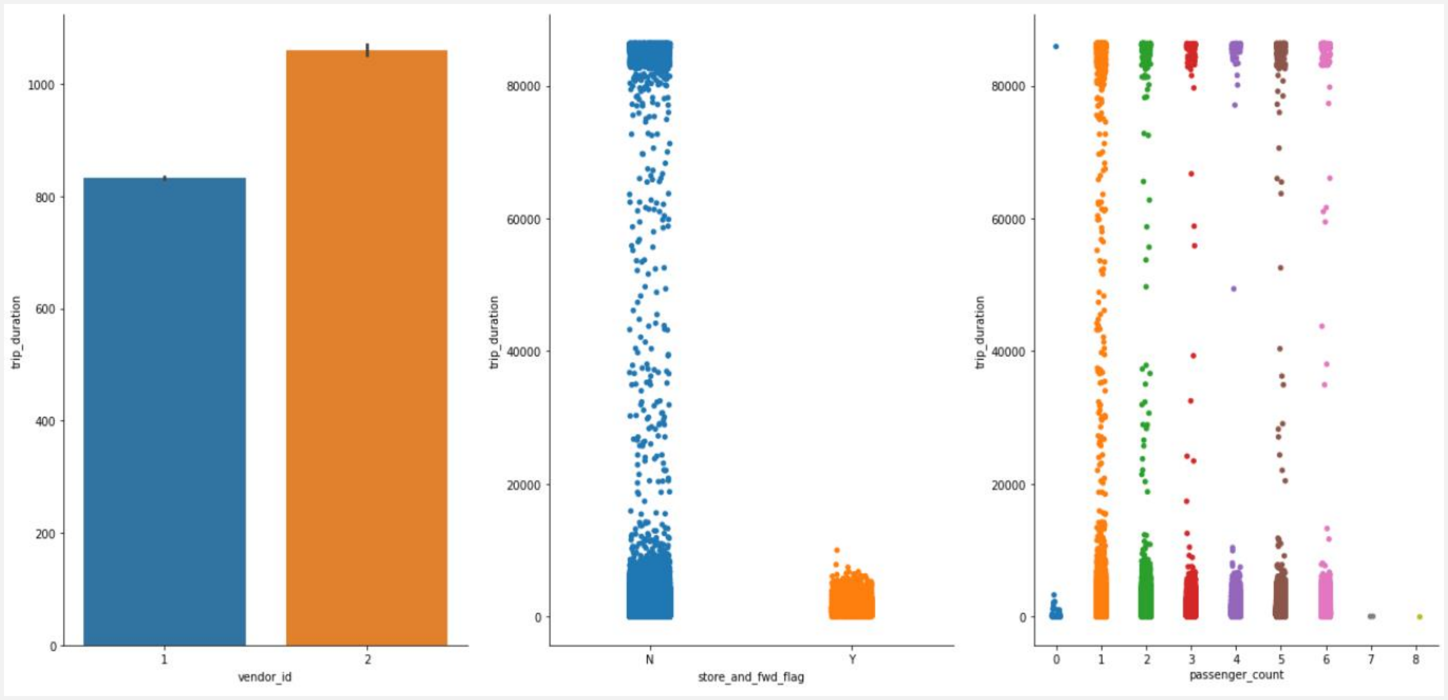


Fig 6. Bivariate analysis against vendor_id, store_and_fwd_flag & passenger_count

The observations from above plot are,

- It looks like vendor 1 mostly provide or prefer shorter ride than vendor 2.
- Most of the rides have flags not stored, also it has been stored mostly for short duration trips.
- There is no significant relation between trip duration and passenger counts. But it worth the notice that no long trips have been conducted taking passenger count of 7 or more.
- There are several trips with 0 passenger, the possible reason might be the cab is called to some particular location to arrange pickup by customer.

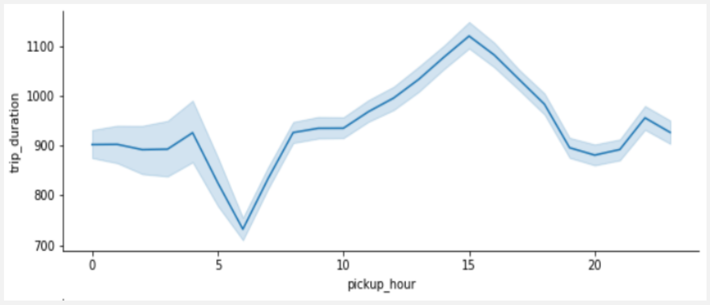


Fig 7. trip_duration vs pickup_hour

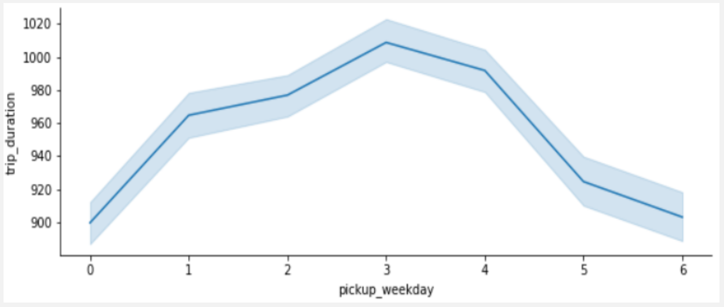


Fig 8. Trip_duration vs pickup_weekday

We can see the least number of requests comes from time ~4 AM to 6 AM, and from ~8 AM to 3 PM the requests were moderate in number, and then gradually till ~1 AM in night, the requests for cabs increased. Between mid-week trip duration is high than other weekday, normally it lowest on Sunday, people use cab mostly for professional purposes.

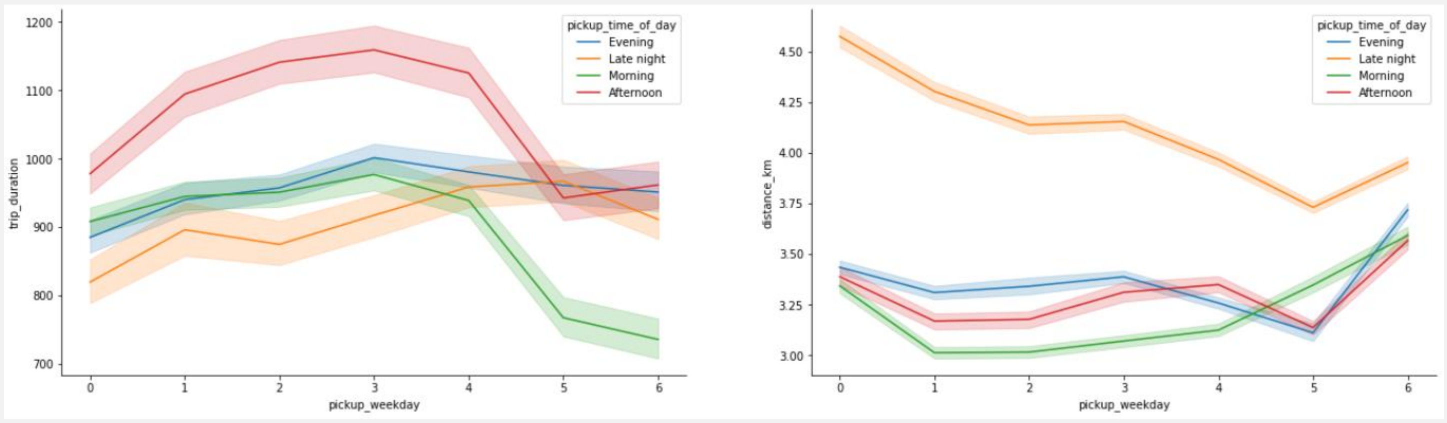


Fig 9. trip_duration & distance_km vs pickup_weekday

We can notice that, duration of travel after pickup at afternoon is high, might be roads are busy, office duration ends etc. Similarly, duration of travel at midnight are somehow low, but the distances are relatively high because traffics are pretty low in midnight.

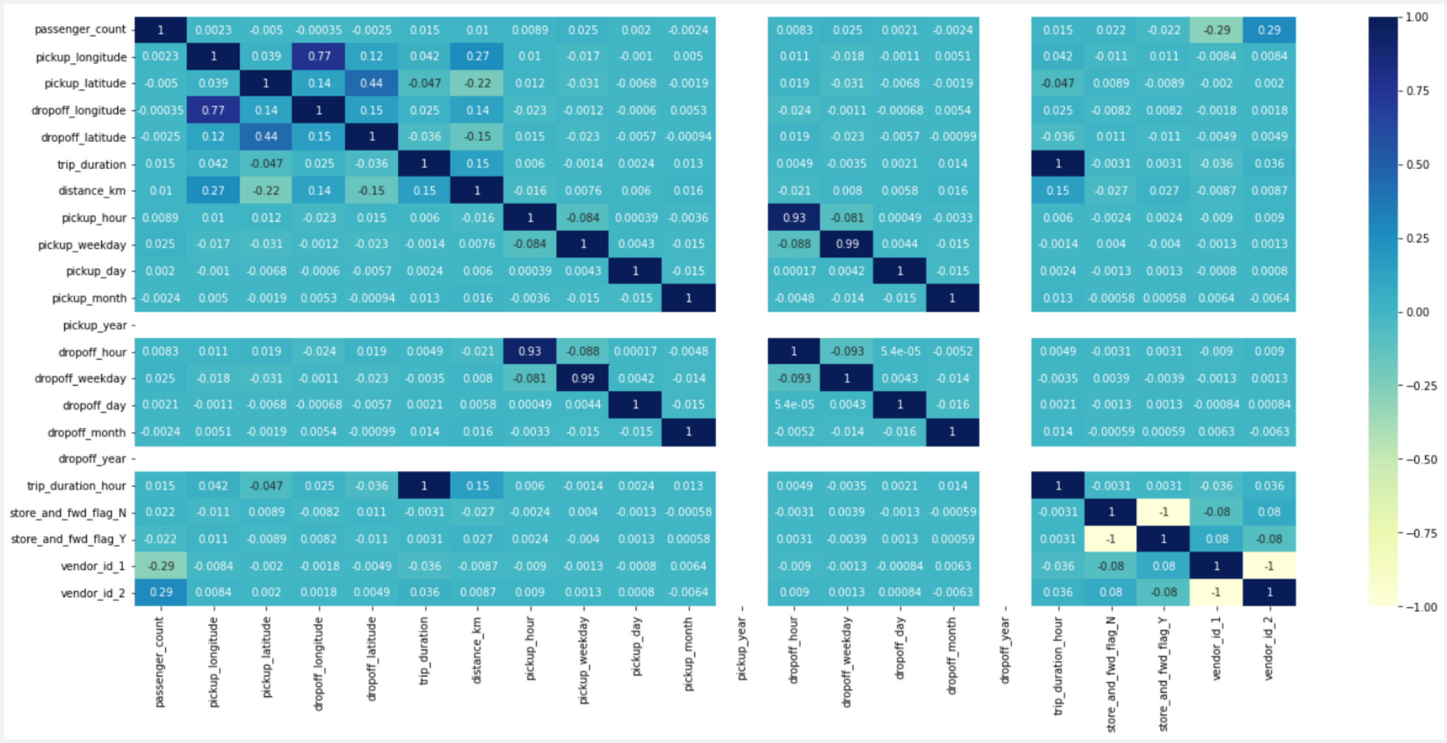


Fig 10. Correlation Matrix

Data has only one pickup/dropoff_year i.e. 2016. So, we will drop it. With our target variable trip_duration maximum correlation is with distance i.e. -0.15. We'll keep pickup_hour and drop dropoff_hour due to very high correlation. For the same reason we'll remove pickup_weekday.

Modelling:

Feature Preparation:

Two columns vendor_id & store_and_fwd_flag are categorical, have been one hot encoded. The dataset is split into 3 parts which are train, test & validation. We've 1039279 records in our training data, 346427 records in test data and 72932 in validation dataset. The validation dataset has not been modified by any feature engineering techniques.

Train and test set has been standardized using StandardScaler from sklearn, and target variable has been log transformed. Let us plot the graphs of train and test set.

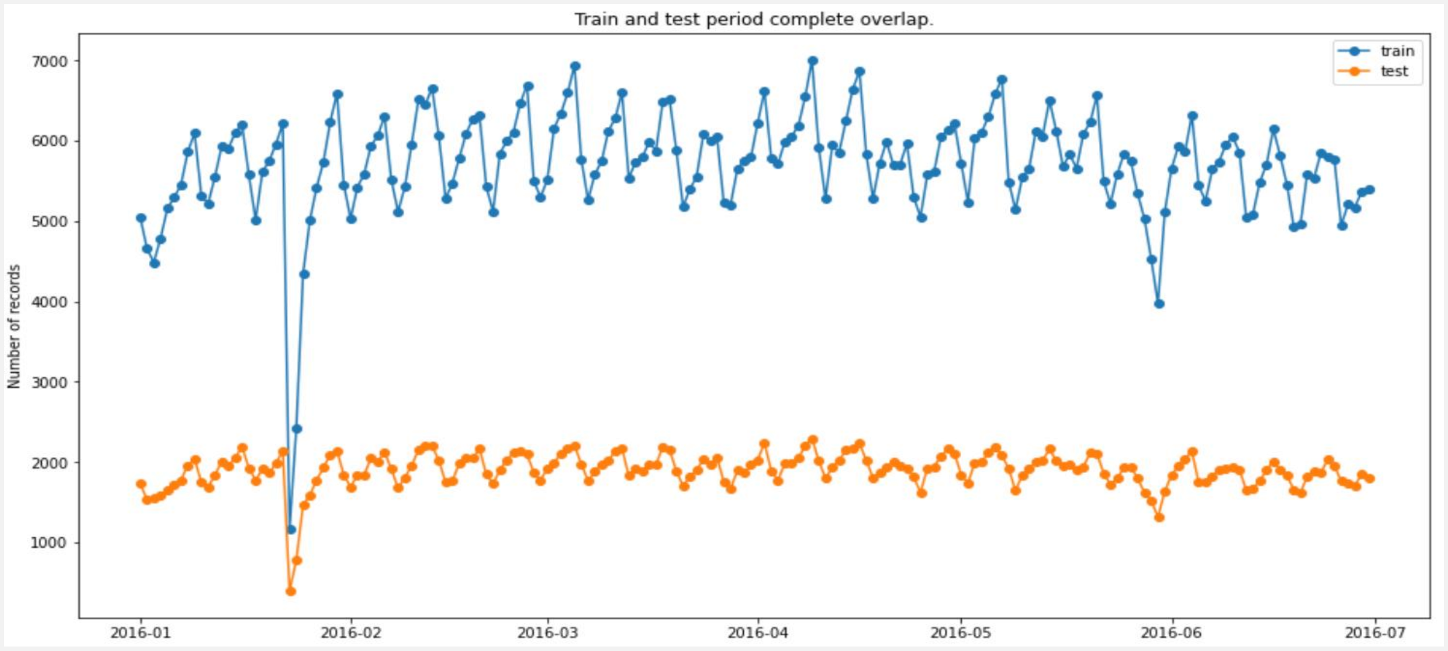


Fig 11. Train and test period complete overlap

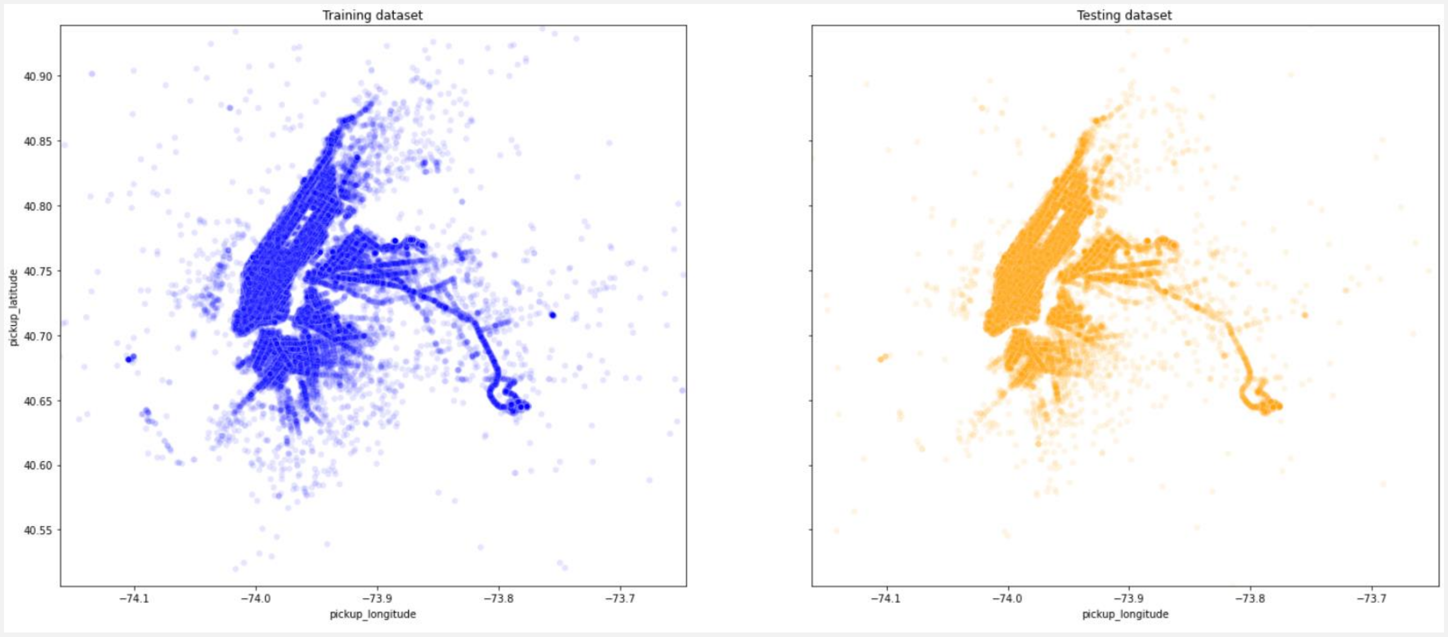


Fig 12. Plotting of coordinates of both dataset

Training:

We've approached with 6 models with hyper parameter tuning over all these.

1. **LinearRegression:**

   Linear regression is one of the very basic forms of machine learning where we train a model to predict the behavior of your data based on some variables. In the case of linear regression as you can see the name suggests linear that means the two variables which are on the x-axis and y-axis should be linearly correlated.

Mathematically, we can write a linear regression equation as:

$$y = a + bx$$

Where a and b given by the formulas:

$$b\,(slope) = \frac{n\sum xy - (\sum x)(\sum y)}{n\sum x^2 - (\sum x)^2}$$

$$a\,(intercept) = \frac{n\sum y - b(\sum x)}{n}$$

2. **LassoRegression:**

   The acronym "LASSO" stands for Least Absolute Shrinkage and Selection Operator. Lasso regression is a type of linear regression that uses shrinkage. Shrinkage is where data values are shrunk towards a central point, like the mean. The lasso procedure encourages simple, sparse models (i.e. models with fewer parameters). This particular type of regression is well-suited for models showing high levels of multicollinearity or when you want to automate certain parts of model selection, like variable selection/parameter elimination. It performs L1 Regularization.

$$\sum_{i=1}^{n}(y_i - \sum_{j} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

3. **RidgeRegression:**

   Ridge regression is a model tuning method that is used to analyze any data that suffers from multicollinearity or when the number of predictor variables in a set exceeds the number of observations. This method performs L2 regularization. When the issue of multicollinearity occurs, least-squares are unbiased, and variances are large, this results in predicted values being far away from the actual values. It performs L2 Regularization.

4. DecisionTreeRegressor:

Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. Decision trees regression normally use mean squared error (MSE) to decide to split a node in two or more sub-nodes. The algorithm pick a variable and tried to split using a value such that two groups are as different as possible.

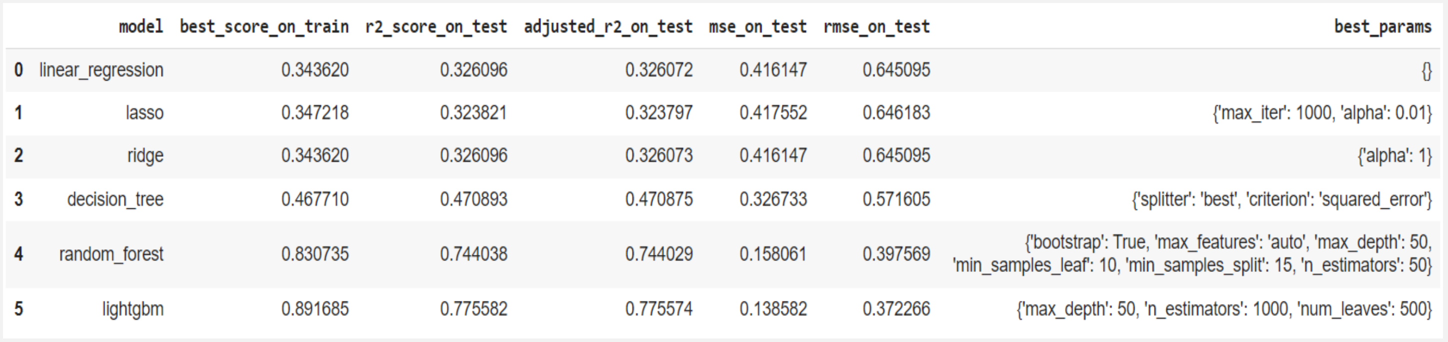
5. RandomForestRegressor:

Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model. It first picks a random ***k*** data points from the training set and build a decision tree associated to these ***k*** data points. A Random Forest Regression model is powerful and accurate. It usually performs great on many problems, including features with non-linear relationships. Disadvantages, however, include the following: there is no interpretability, overfitting may easily occur, we must choose the number of trees to include in the model.

6. LGBMRegressor:

LightGBM stands for Light Gradient Boosting Machine, this algorithm extends the gradient boosting algorithm by adding a type of automatic feature selection as well as focusing on boosting examples with larger gradients. This can result in a dramatic speedup of training and improved predictive performance. It takes less memory to run and is able to deal with large amounts of data.

After fitting the models with training data, we get a maximum r2_score of 0.775582 on LightGBM model. The training statistics are as of below,



	model	best_score_on_train	r2_score_on_test	adjusted_r2_on_test	mse_on_test	rmse_on_test	best_params
0	linear_regression	0.343620	0.326096	0.326072	0.416147	0.645095	{}
1	lasso	0.347218	0.323821	0.323797	0.417552	0.646183	{'max_iter': 1000, 'alpha': 0.01}
2	ridge	0.343620	0.326096	0.326073	0.416147	0.645095	{'alpha': 1}
3	decision_tree	0.467710	0.470893	0.470875	0.326733	0.571605	{'splitter': 'best', 'criterion': 'squared_error'}
4	random_forest	0.830735	0.744038	0.744029	0.158061	0.397569	{'bootstrap': True, 'max_features': 'auto', 'max_depth': 50, 'min_samples_leaf': 10, 'min_samples_split': 15, 'n_estimators': 50}
5	lightgbm	0.891685	0.775582	0.775574	0.138582	0.372266	{'max_depth': 50, 'n_estimators': 1000, 'num_leaves': 500}

Fig 13. Error metrics obtained from training

Evaluation:

We've predicted target using our validation set on our LightBGM model as it is the best performing model trained by us. On our validation set, we obtained adjusted r2_score of ~0.78 and root mean squared error of ~0.367.

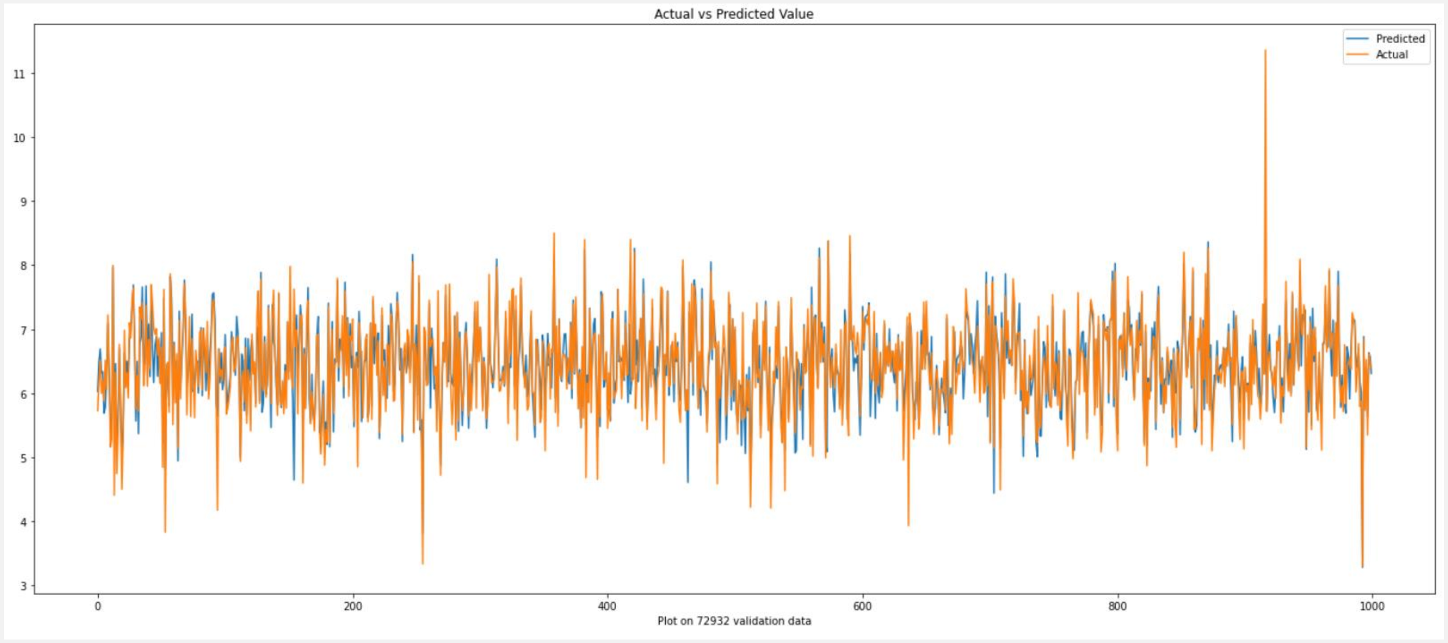


Fig 14. Actual vs Predicted value on validation dataset

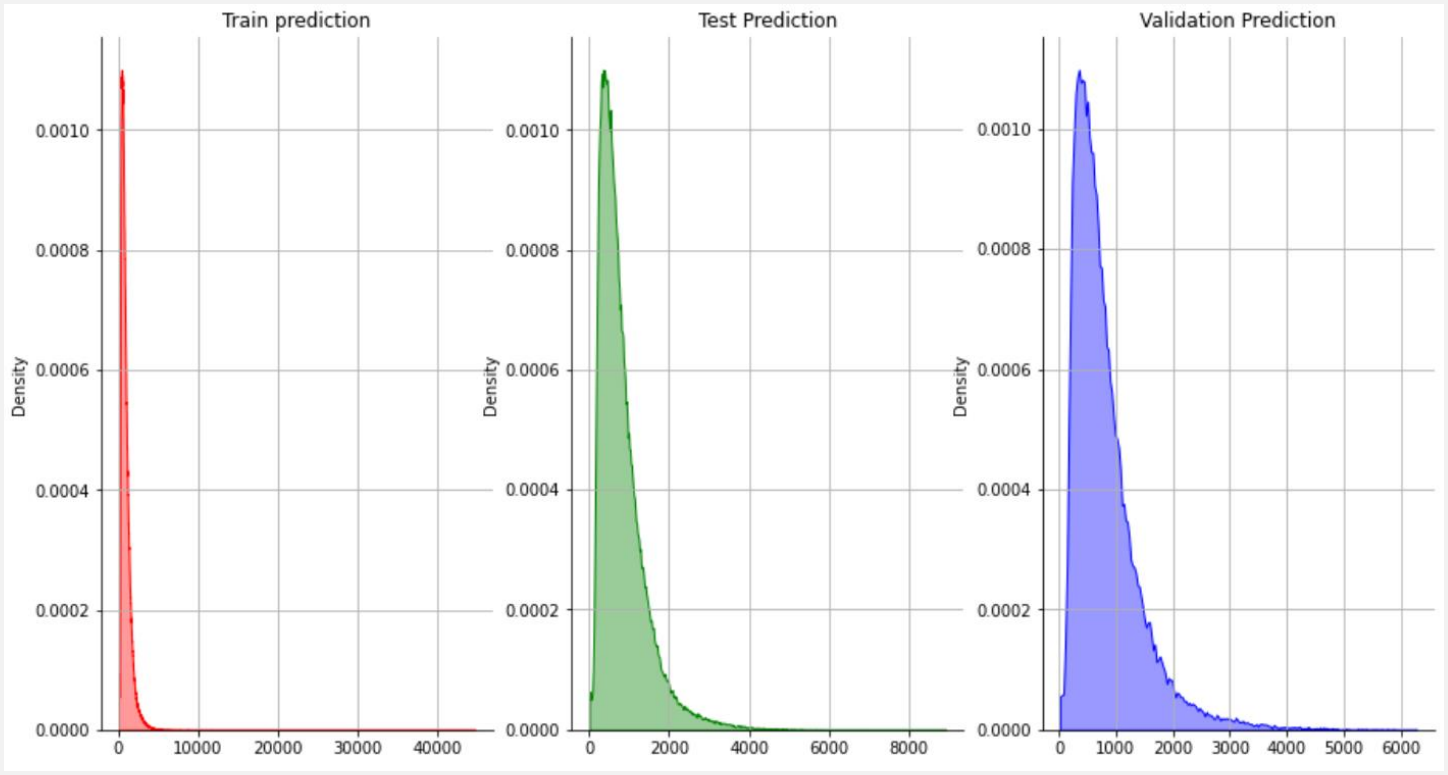


Fig 15. Comparison of prediction on different dataset

Limitation:

Real world data can be and is a messy one, it's also a no exception in our dataset. In our case, data as it was very raw, and we needed to clean it first to move further. It has many anomalies we noticed about the time duration and distance travelled, though we've removed it but still many more is present there, we live it as it is to add a bit noise.

We don't have proper information of traffic and season, that'd be a great addition in this dataset. Also, the rating of cab or driver would also do great, people normally provide better rating with clean driving and punctuality in rush hours.

Scope of Improvement:

We can add more features to our dataset, like ratings, driver's ability on some scale or experience (experienced driver is normally better at driving than newbies), seasons information or can feed live traffic data. Also, more extensive EDA is required with help of external datasets.

Try out XGBoost model, done hyper parameter tuning over XGBoost model. XGBoost always gives more importance to functional space when reducing the cost of a model while Random Forest tries to give more preferences to hyperparameters to optimize the model. Also we need to perform PCA to reduce dimensions.

Conclusion:

From the entire regression analysis, it can be concluded that,

- We've achieved r2_score > ~0.78 on validation set, though it is not that much of good, especially when we scale prediction from log to normal scale, noise will be too high, but still, we have scope of improvement with addition of few other features (can be from other sources to merge with), more accurate hyper parameter tuning, using XGBoost or performing PCA etc.
- Almost all long delays happen for the cab of vendor 2, even though vendor 2 has only a slightly higher percentage of rides overall. This discrepancy is definitely significant enough to explain most of the effect.
- Driver's ability can be an important factor some drivers got higher skills comparatively than others, so trip duration can also vary with driver's ability as well.
- Outside of Work hours traffics are relatively low, majority of trips are fast.
- sklearn CV options (RandomizedSearchCV or GridSearchCV) for usual mid-to-large dataset the training for a single model could take hours, also reducing memory is another challenging

THE END