Capstone project 2-
(NYC TAXI TRIP TIME PREDICTION)

NAME -***Vivek Kumar Soni***

CONTENT

INTRODUCTION

- In New York City, due to traffic jams, construction or road blockage etc. user will need to know how much time it will take to commute from one place to other.
- Increasing popularity of app-based taxi such as ola or uber and there competitive pricing levels made user decisive to choose based on trip pricing and duration.
- Taxi Drivers also have to choose best route having lesser trip time.
- So here we will be building a model which will be predicting the trip duration of taxies running in NewYork. This prediction will help customers to select the taxi based on trip duration and driver to select optimum route to their destination.

PROBLEM STATEMENT

We have the dataset Which is based on the 2016 NYC Yellow Cab trip record data made available in Big Query on the Google Cloud Platform. The data was originally published by the NYC Taxi and Limousine Commission (TLC). The data was sampled and cleaned for the purposes of this project. Based on individual trip attributes, we should predict the duration of each trip in the test set.

DATA SUMMARY

- id - a unique identifier for each trip
- vendor_id - a code indicating the provider associated with the trip record
- pickup_datetime - date and time when the meter was engaged
- dropoff_datetime - date and time when the meter was disengaged

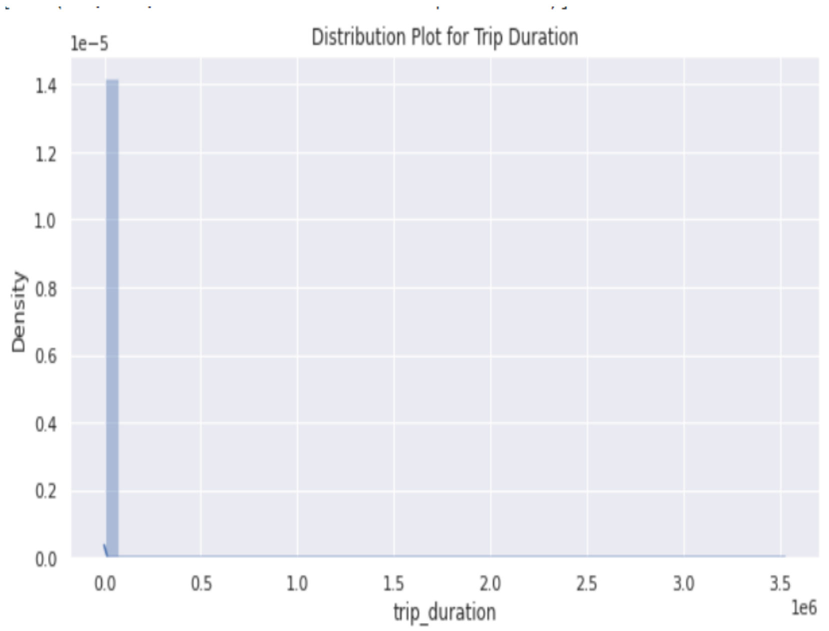
- passenger_count - the number of passengers in the vehicle (driver

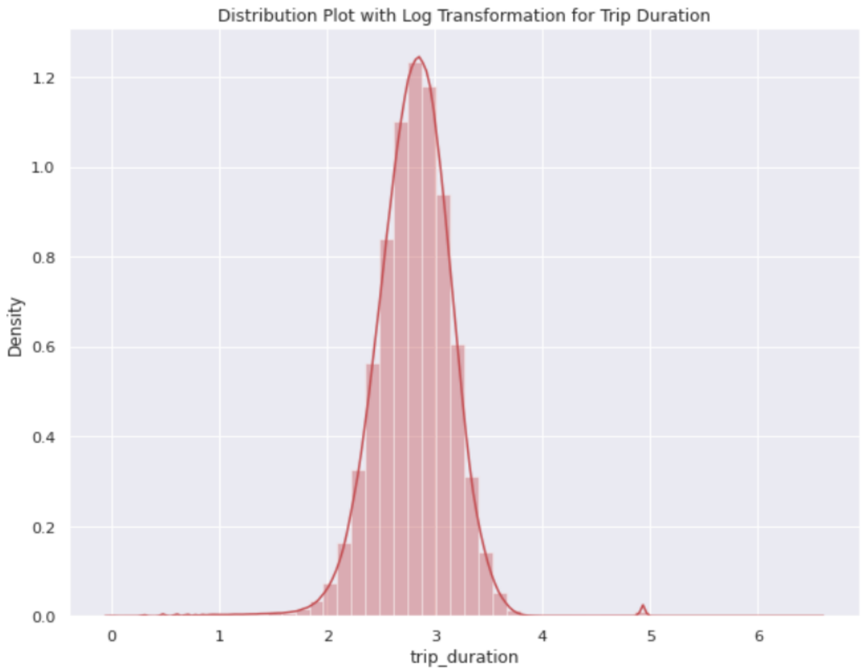
entered value)
- pickup_longitude - the longitude where the meter was engaged
- pickup_latitude - the latitude where the meter was engaged
- dropoff_longitude - the longitude where the meter was disengaged
- dropoff_latitude - the latitude where the meter was disengaged
- store_and_fwd_flag - This flag 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 the trip in seconds

BASIC EXPLORATION

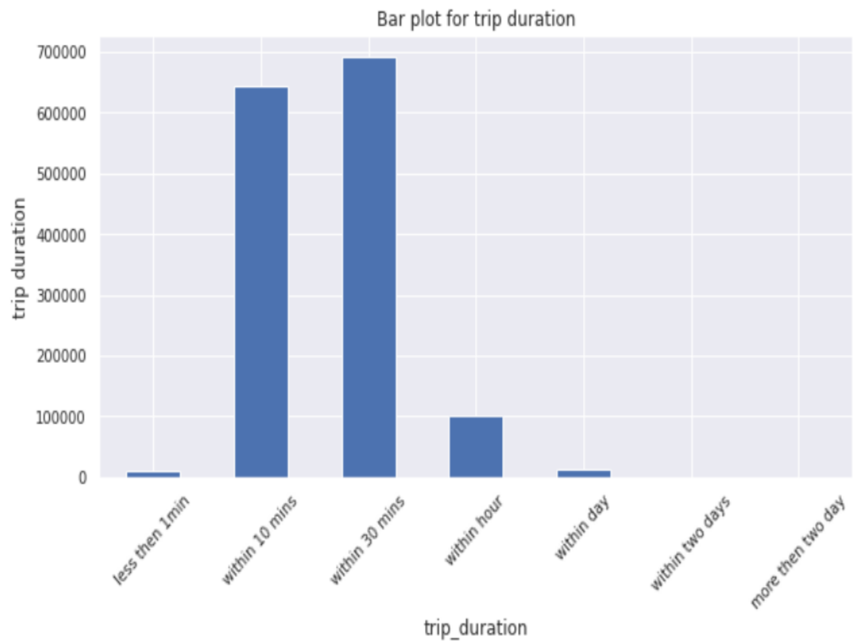
- The dataset contains 1458644 rows and 11 columns.
- Two categorical features ‘store_and_fwd_flag’ and ‘vendor_id’
- Outliers present in all numerical features
- Data formating steps required for datetime features
- No null values present
- Passenger_count, Vendor_id and trip_duration are having integer value.
- pickup_datetime,dropoff_datetime is a datetime variable
- pickup_longitude,pickup_latitude,dropoff_longitude,dropoff_latitude are real numbers having float as data type *store_and_fwd_flag and Id belongs to a string data type.

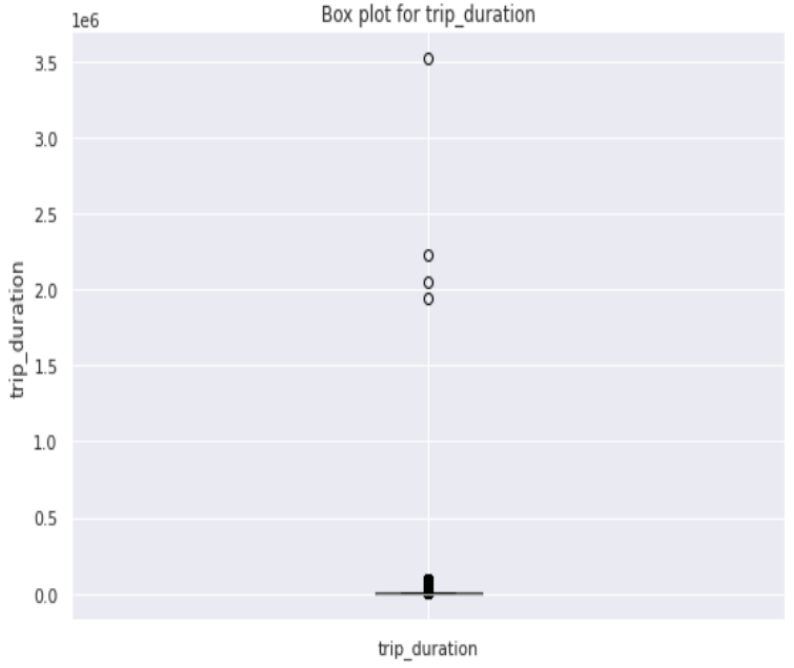
TRIP DURATION DATA ANALYSIS





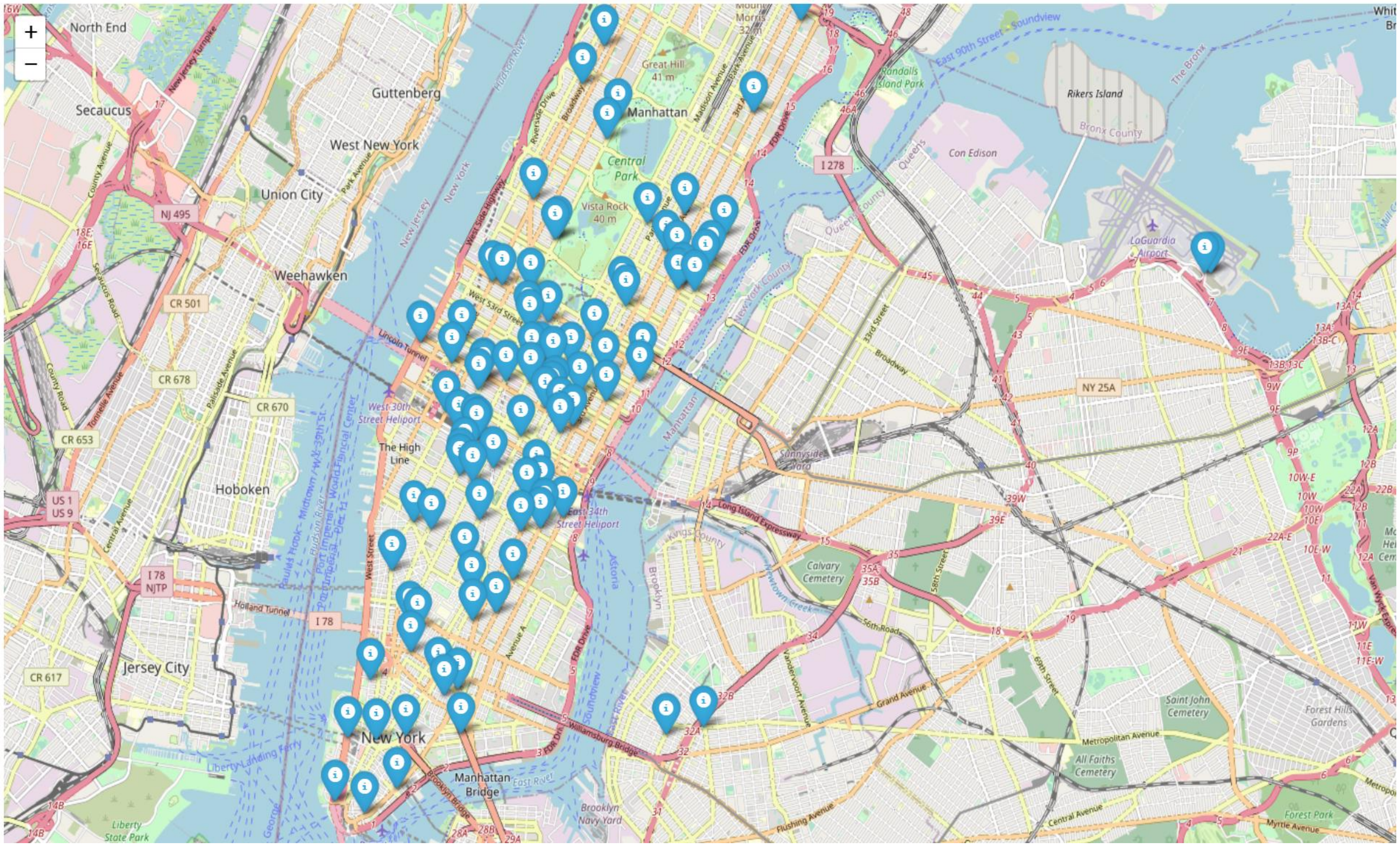
TRIP DURATION DATA ANALYSIS



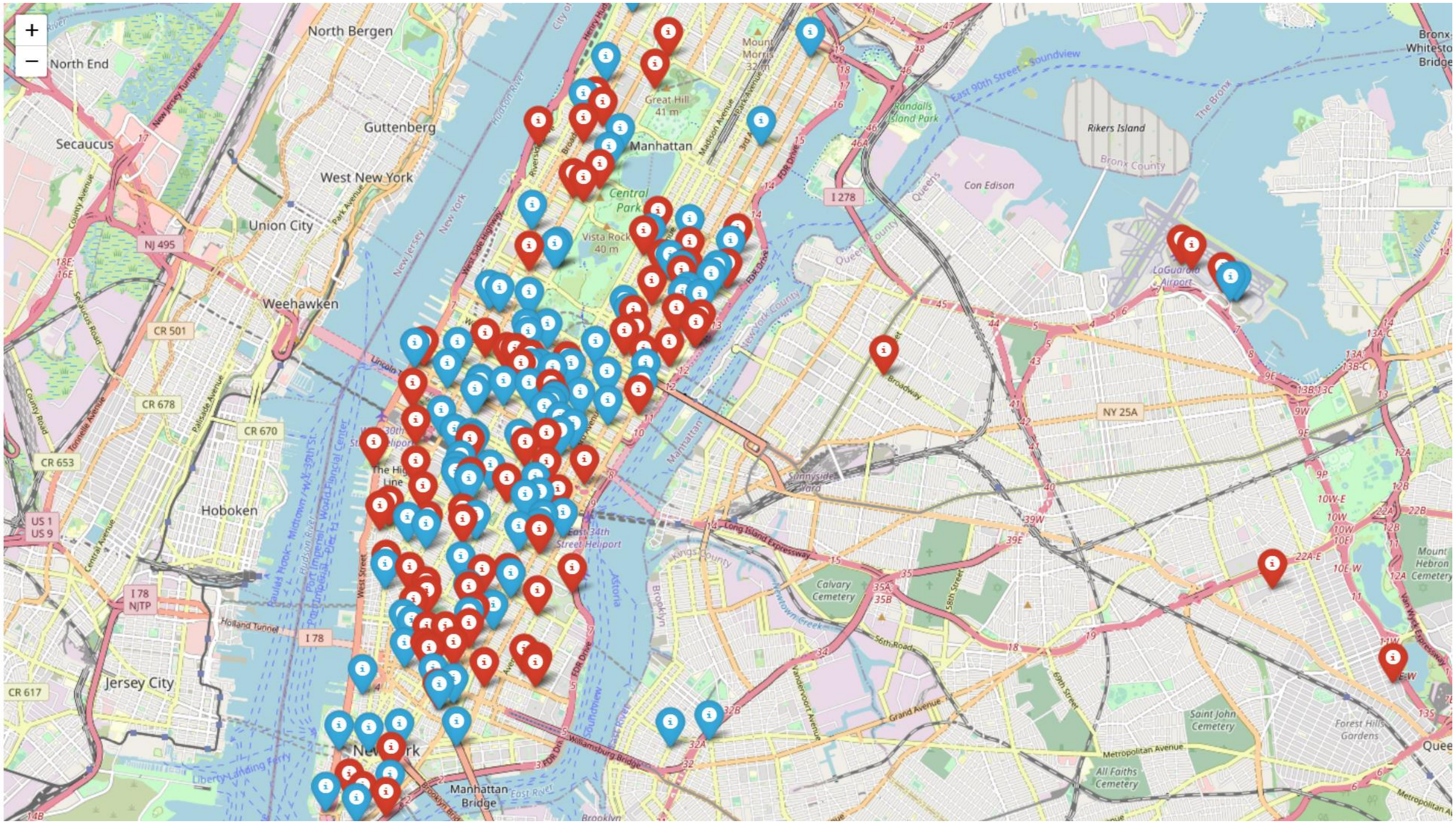


From the figure above we can conclude that most of the trip are general for 10 min to 1 hour.

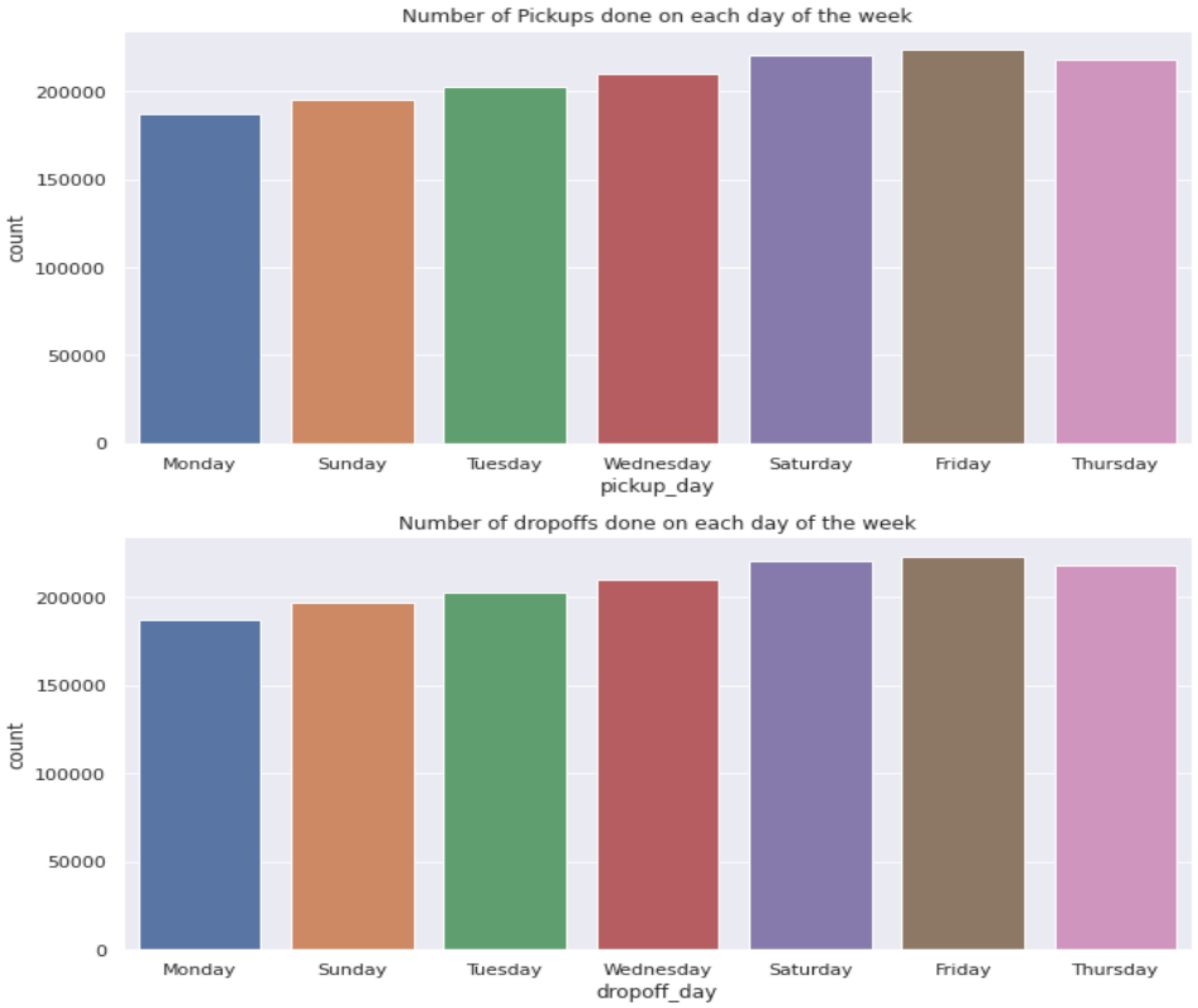
Distribution of Pickup Latitude and longitude



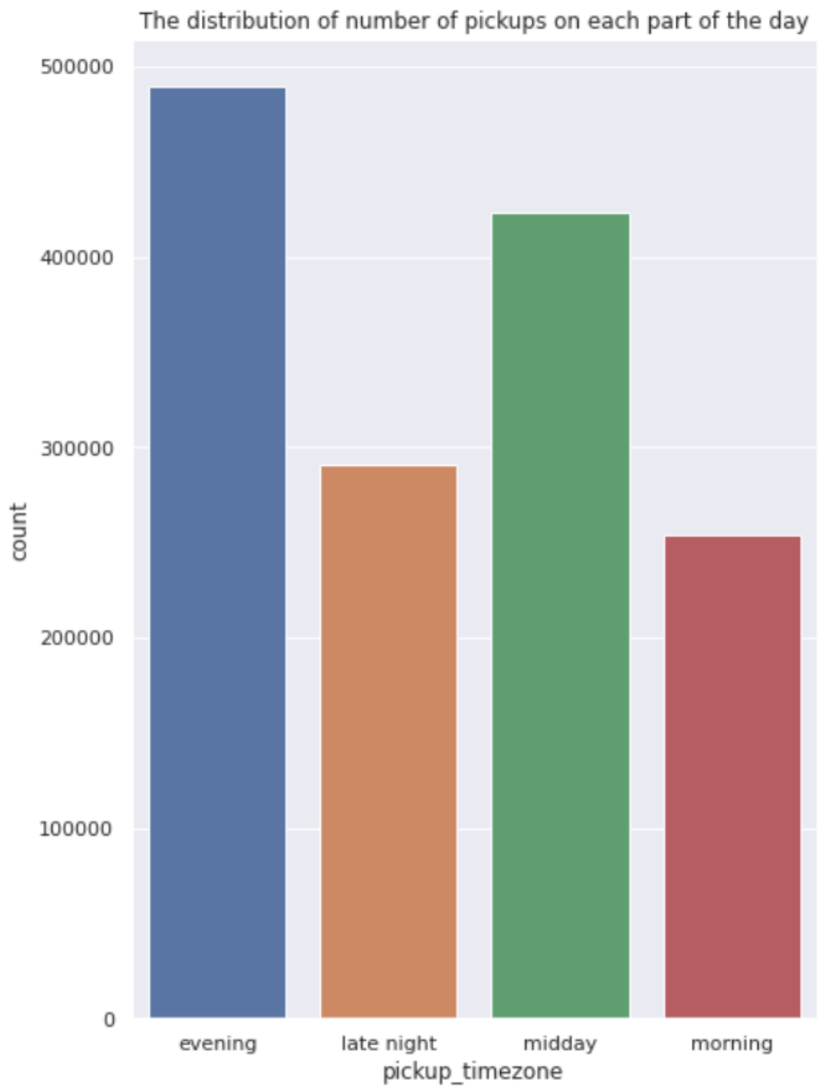
Distribution of Dropoff Latitude and longitude

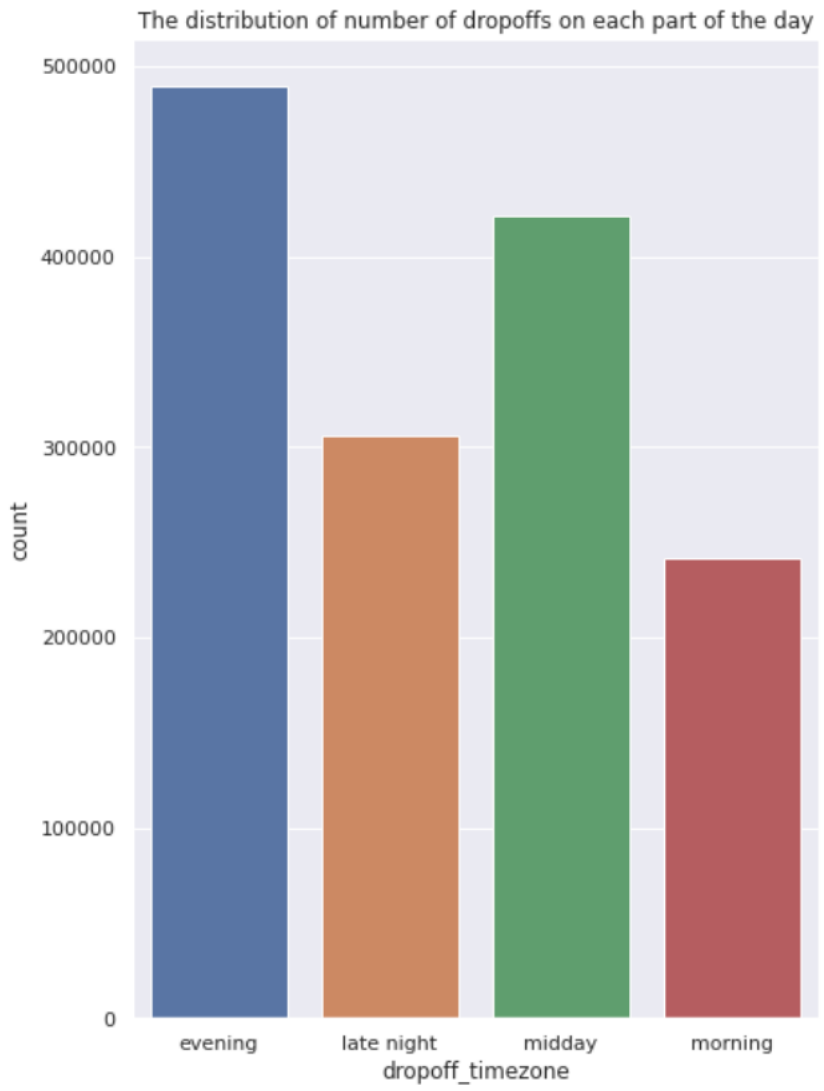


Number of Pickups and Drop-offs with in a Week



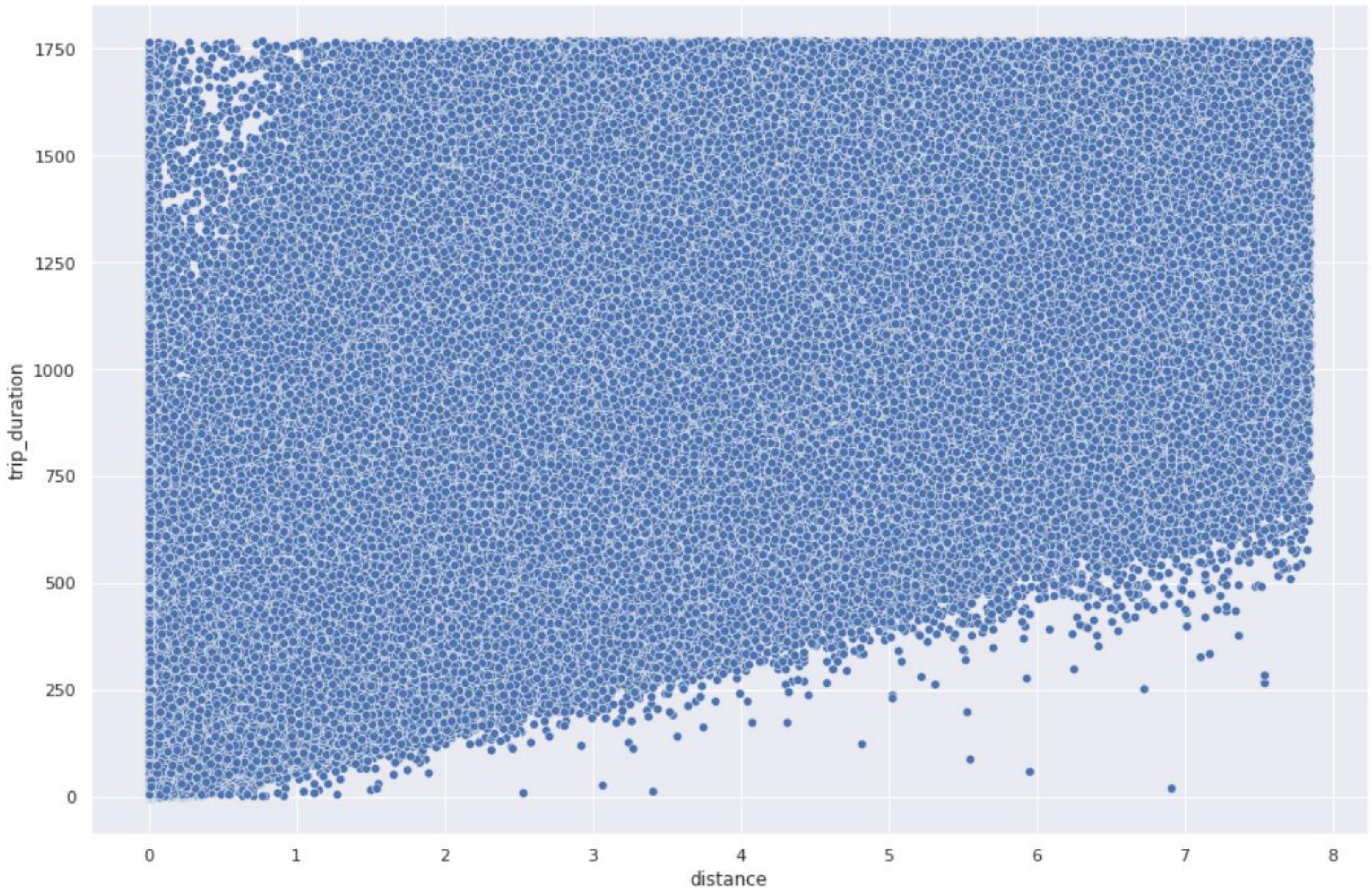
Number of Pickups and Drop-offs with in a Day



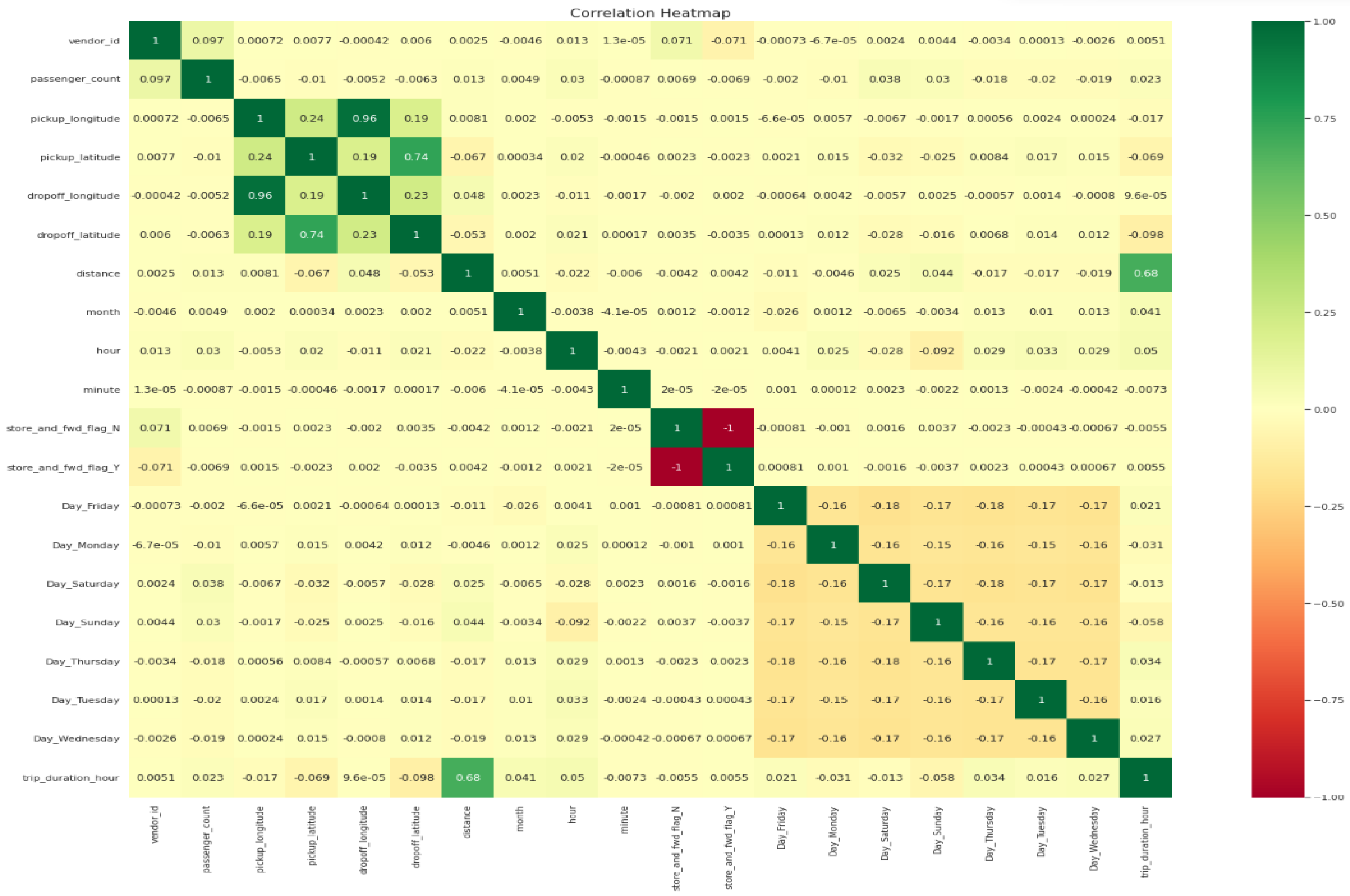


Relation Between Trip duration and Distance

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fecbb21a4d0>
```



Correlation Heatmap



	vendor_id	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	distance	month	hour	minute	store_and_fwd_flag_N	store_and_fwd_flag_Y	Day_Friday	Day_Monday	Day_Saturday	Day_Sunday	Day_Thursday	Day_Tuesday	Day_Wednesday	trip_duration_hour
vendor_id	1	0.097	0.00072	0.0077	-0.00042	0.006	0.0025	-0.0046	0.013	1.3e-05	0.071	-0.071	-0.00073	-6.7e-05	0.0024	0.0044	-0.0034	0.00013	-0.0026	0.0051
passenger_count	0.097	1	-0.0065	-0.01	-0.0052	-0.0063	0.013	0.0049	0.03	-0.00087	0.0069	-0.0069	-0.002	-0.01	0.038	0.03	-0.018	-0.02	-0.019	0.023
pickup_longitude	0.00072	-0.0065	1	0.24	0.96	0.19	0.0081	0.002	-0.0053	-0.0015	-0.0015	0.0015	-6.6e-05	0.0057	-0.0067	-0.0017	0.00056	0.0024	0.00024	-0.017
pickup_latitude	0.0077	-0.01	0.24	1	0.19	0.74	-0.067	0.00034	0.02	-0.00046	0.0023	-0.0023	0.0021	0.015	-0.032	-0.025	0.0084	0.017	0.015	-0.069
dropoff_longitude	-0.00042	-0.0052	0.96	0.19	1	0.23	0.048	0.0023	-0.011	-0.0017	-0.002	0.002	-0.00064	0.0042	-0.0057	0.0025	-0.00057	0.0014	-0.0008	9.6e-05
dropoff_latitude	0.006	-0.0063	0.19	0.74	0.23	1	-0.053	0.002	0.021	0.00017	0.0035	-0.0035	0.00013	0.012	-0.028	-0.016	0.0068	0.014	0.012	-0.098
distance	0.0025	0.013	0.0081	-0.067	0.048	-0.053	1	0.0051	-0.022	-0.006	-0.0042	0.0042	-0.011	-0.0046	0.025	0.044	-0.017	-0.017	-0.019	0.68
month	-0.0046	0.0049	0.002	0.00034	0.0023	0.002	0.0051	1	-0.0038	-4.1e-05	0.0012	-0.0012	-0.026	0.0012	-0.0065	-0.0034	0.013	0.01	0.013	0.041
hour	0.013	0.03	-0.0053	0.02	-0.011	0.021	-0.022	-0.0038	1	-0.0043	-0.0021	0.0021	0.0041	0.025	-0.028	-0.092	0.029	0.033	0.029	0.05
minute	1.3e-05	-0.00087	-0.0015	-0.00046	-0.0017	0.00017	-0.006	-4.1e-05	-0.0043	1	2e-05	-2e-05	0.001	0.00012	0.0023	-0.0022	0.0013	-0.0024	-0.00042	-0.0073
store_and_fwd_flag_N	0.071	0.0069	-0.0015	0.0023	-0.002	0.0035	-0.0042	0.0012	-0.0021	2e-05	1	-1	-0.00081	-0.001	0.0016	0.0037	-0.0023	-0.00043	-0.00067	-0.0055
store_and_fwd_flag_Y	-0.071	-0.0069	0.0015	-0.0023	0.002	-0.0035	0.0042	-0.0012	0.0021	-2e-05	-1	1	0.00081	0.001	-0.0016	-0.0037	0.0023	0.00043	0.00067	0.0055
Day_Friday	-0.00073	-0.002	-6.6e-05	0.0021	-0.00064	0.00013	-0.011	-0.026	0.0041	0.001	-0.00081	0.00081	1	-0.16	-0.18	-0.17	-0.18	-0.17	-0.17	0.021
Day_Monday	-6.7e-05	-0.01	0.0057	0.015	0.0042	0.012	-0.0046	0.0012	0.025	0.00012	-0.001	0.001	-0.16	1	-0.16	-0.15	-0.16	-0.15	-0.16	-0.031
Day_Saturday	0.0024	0.038	-0.0067	-0.032	-0.0057	-0.028	0.025	-0.0065	-0.028	0.0023	0.0016	-0.0016	-0.18	-0.16	1	-0.17	-0.18	-0.17	-0.17	-0.013
Day_Sunday	0.0044	0.03	-0.0017	-0.025	0.0025	-0.016	0.044	-0.0034	-0.092	-0.0022	0.0037	-0.0037	-0.17	-0.15	-0.17	1	-0.16	-0.16	-0.16	-0.058
Day_Thursday	-0.0034	-0.018	0.00056	0.0084	-0.00057	0.0068	-0.017	0.013	0.029	0.0013	-0.0023	0.0023	-0.18	-0.16	-0.18	-0.16	1	-0.17	-0.17	0.034
Day_Tuesday	0.00013	-0.02	0.0024	0.017	0.0014	0.014	-0.017	0.01	0.033	-0.0024	-0.00043	0.00043	-0.17	-0.15	-0.17	-0.16	-0.17	1	-0.16	0.016
Day_Wednesday	-0.0026	-0.019	0.00024	0.015	-0.0008	0.012	-0.019	0.013	0.029	-0.00042	-0.00067	0.00067	-0.17	-0.16	-0.17	-0.16	-0.17	-0.16	1	0.027
trip_duration_hour	0.0051	0.023	-0.017	-0.069	9.6e-05	-0.098	0.68	0.041	0.05	-0.0073	-0.0055	0.0055	0.021	-0.031	-0.013	-0.058	0.034	0.016	0.027	1

Evaluation Metrics

Evaluation metrics are used to measure the quality of the statistical or machine learning model. Evaluating machine learning models or algorithms is essential for any project. There are many different types of evaluation metrics available to test a model.

Why We require Evaluation Metrics?

Most beginners and practitioners most of the time do not bother about the model performance. The talk is about building a well-generalized model, Machine learning model cannot have 100 per cent efficiency otherwise the model is known as a biased model. which further includes the concept of overfitting and underfitting.

It is necessary to obtain the accuracy on training data, But it is also important to get a genuine and approximate result on unseen data otherwise Model is of no use.

So to build and deploy a generalized model we require to Evaluate the model on different metrics which helps us to better optimize the performance, fine-tune it, and obtain a better result.

If one metric is perfect, there is no need for multiple metrics. To understand the benefits and disadvantages of Evaluation metrics because different evaluation metric fits on a different set of a dataset.

Different Evaluation metrics

1) Mean Absolute Error(MAE)
MAE is a very simple metric which calculates the absolute difference between actual and predicted values.

2) Mean Squared Error(MSE)
MSE is a most used and very simple metric with a little bit of change in mean absolute error. Mean squared error states that finding the squared difference between actual and predicted value.

3) Root Mean Squared Error(RMSE)
As RMSE is clear by the name itself, that it is a simple square root of mean squared error.

4) **R Squared (R2)**
R2 score is a metric that tells the performance of your model, not the loss in an absolute sense that how many wells did your model perform.

5) **Adjusted R2**

Adjusted R2 is a corrected goodness-of-fit (model accuracy) measure for linear models. It identifies the percentage of variance in the target field that is explained by the input or inputs.

Linear Regression

Linear Regression is a regression of dependent variable on independent variable. It is a linear model that assumes a linear relationship between dependent
(y) and independent variables (x). The dependent variable (y) is calculated by l inear combination of independent variable (x).

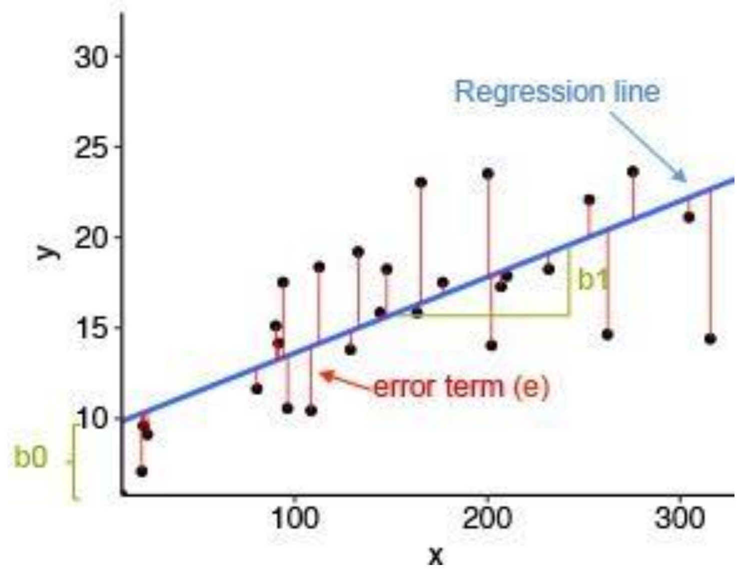
$$y=b_0+b_1x_1+b_2x_2$$

The cost function for linear regression is given by:

Mean of sum of square error

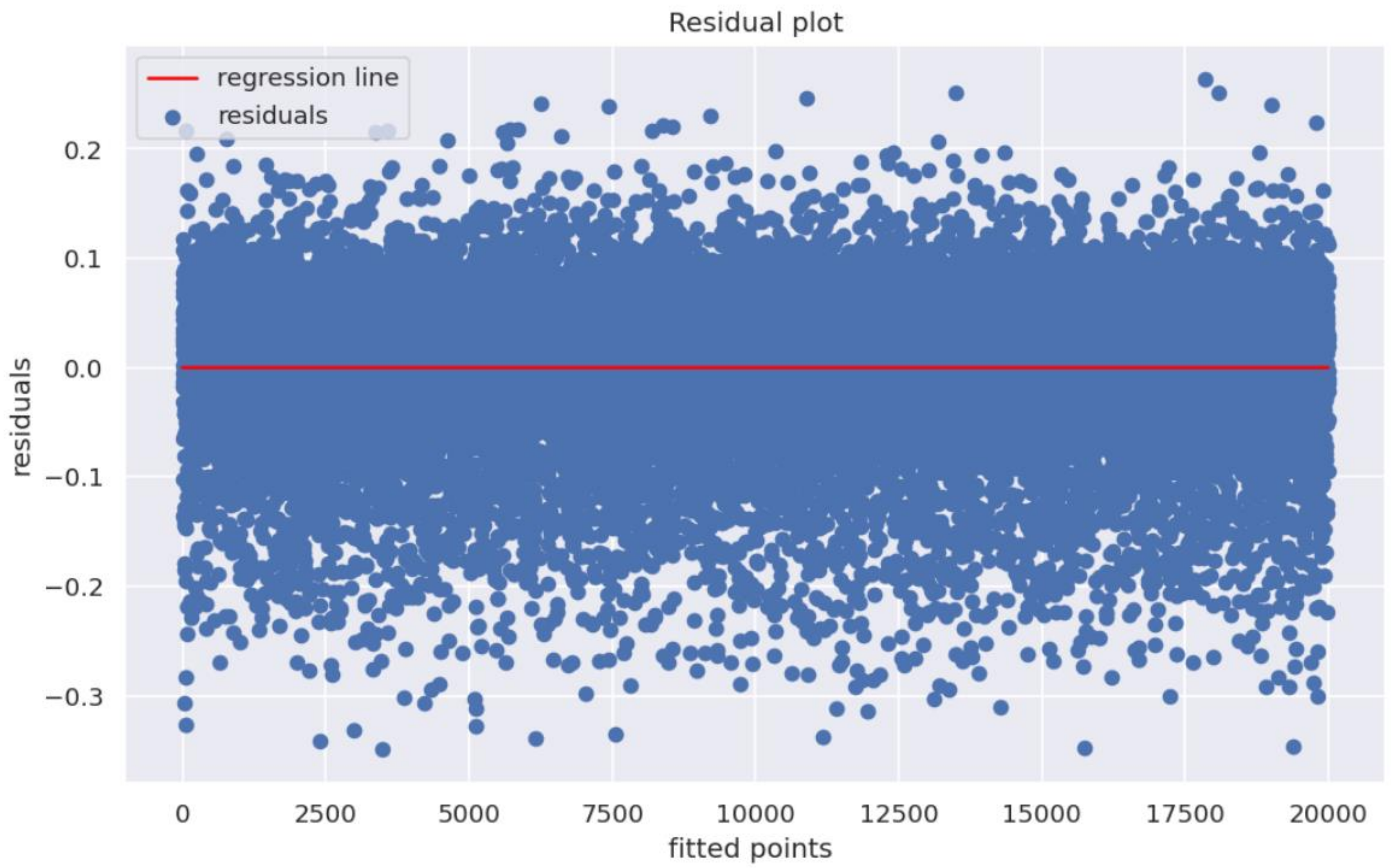
$$MSE = \frac{\sum(y_i - \hat{y}_i)^2}{n}$$

Where:
- y_i is the ith observed value.
- $\hat{y}_i$ is the corresponding predicted value.
- n = the number of observations.



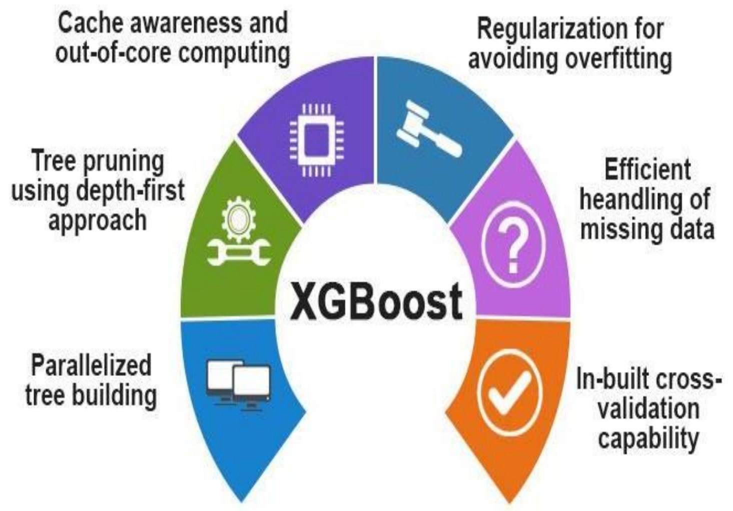
MSE	72764.0061
RMSE	269.748041
R2	0.484766
Adjusted_R2	0.484758

Homoscedasticity check



XGBoost

- XGBoost comes under boosting and is known as extra gradient boosting.
- GBM first calculates the model using X and Y then after the prediction is obtain.
- It will again calculates the model based on residual of previous model
- loss function will give more weightage to error of previous model. and this process continuous until MSE gets minimizes.


Cache awareness and out-of-core computing
Regularization for avoiding overfitting
Tree pruning using depth-first approach
Efficient heandling of missing data
XGBoost
Parallelized tree building
In-built cross-validation capability

XGBoost is just an extension of GBM with following advantages.

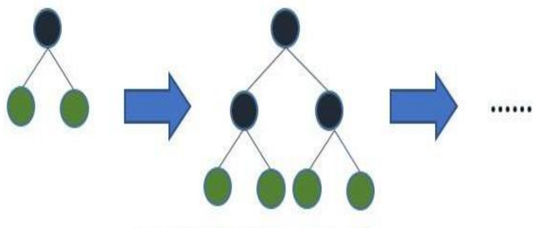
- Regularization
- Parallel Processing
- High Flexibility
- Handles Missing values
- Tree pruning
- Buitin cross validation
- Continuous on existing model

MSE	0.001963
RMSE	0.044311
R2	0.822151
Adjusted_R2	0.822109

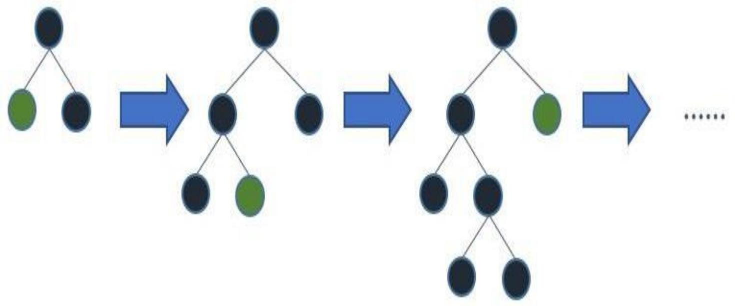
LightGBM

- LightGBM is a fast, distributed high performance gradient boosting framework.
- LightGBM is based on decision tree algorithm. But it splits the tree leaf wise rather then level wise like other boosting algorithm. So when growing on the same leaf in Light GBM, the leaf-wise algorithm can reduce more loss than the level-wise algorithm and hence results in much better accuracy which can rarely be achieved by any of the existing boosting algorithms.

MSE	56602.8129
RMSE	237.913456
R2	0.599202
Adjusted_R2	0.599196



Level-wise tree growth



Leaf-wise tree growth

Feature Importance

Feature Importance refers to techniques that calculate a score for all the input features for a given model — the scores simply represent the “importance” of each feature. A higher score means that the specific feature will have a larger effect on the model that is being used to predict a certain variable.

Let’s take a real-life example for a better understanding. Suppose you have to buy a new house near your workplace. While purchasing a house, you might think of different factors. The most important factor in your decision making might be the location of the property, and so, you’ll likely only look for houses that are near your workplace. Feature importance works in a similar way, it will rank features based on the effect that they have on the model’s prediction.

Feature Importance

Why is Feature Importance so Useful?

Feature Importance is extremely useful for the following reasons:

1) Data Understanding.

Building a model is one thing but understanding the data that goes into the model is another. Like a correlation matrix, feature importance allows you to understand the relationship between the features and the target variable. It also helps you understand what features are irrelevant for the model.

2) Model Improvement.

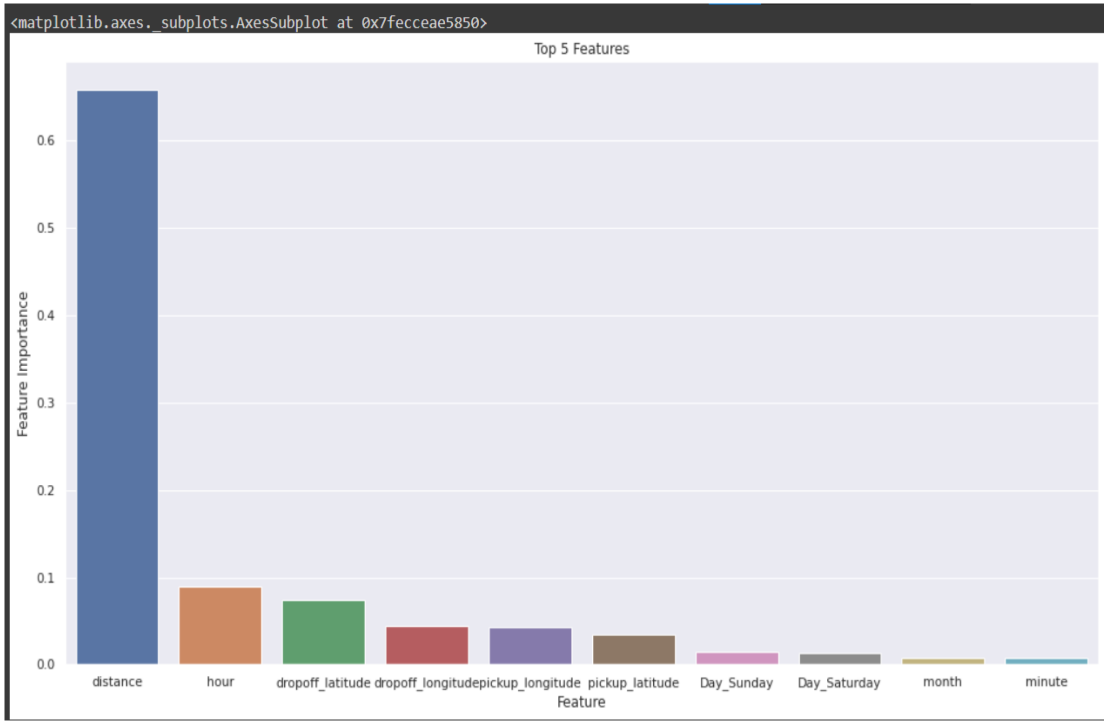
When training your model, you can use the scores calculated from feature importance to reduce the dimensionality of the model. The higher scores are usually kept and the lower scores are deleted as they are not important for the model. This not only makes the model simpler but also speeds up the model's working, ultimately improving the performance of the model.

3) Model Interpretability.

Feature Importance is also useful for interpreting and communicating your model to other stakeholders. By calculating scores for each feature, you can determine which features attribute the most to the predictive power of your model.

Feature Importance

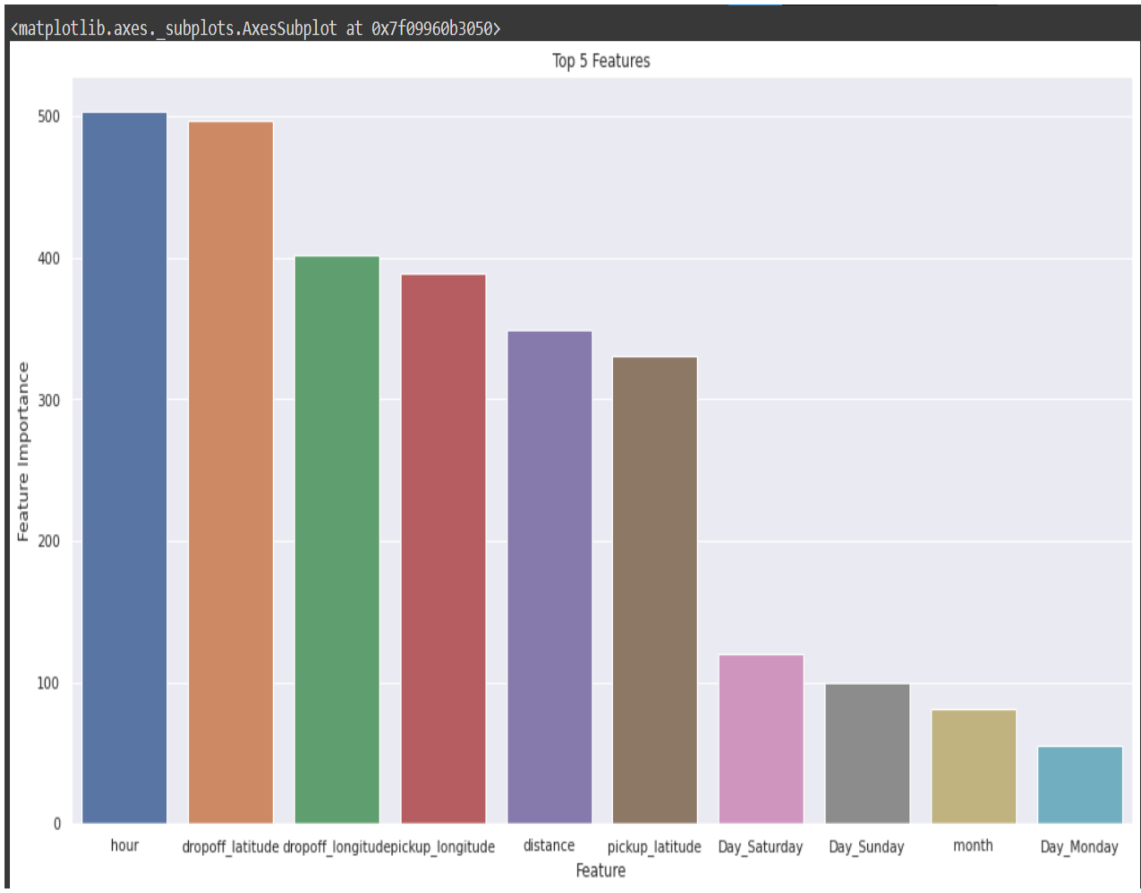
- Fig above illustrate that the most important feature detect by the XGBOOST algorithm is distance



```
<matplotlib.axes._subplots.AxesSubplot at 0x7fecceae5850>
```

Feature Importance

- Fig above illustrate that the most important feature detect by the LightGBM algorithm is hour

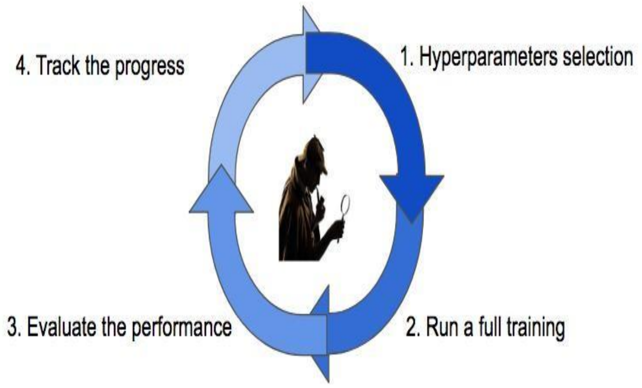


```
<matplotlib.axes._subplots.AxesSubplot at 0x7f09960b3050>
```

Hyperparameter Tuning

Hyperparameters are sets of information that are used to control the way of learning an algorithm.

We used Grid Search CV, for hyperparameter tuning. This also results in cross-validation and in our case we divided the dataset into different folds



Hyperparameters	XGB	LightGBM
n_estimator	[5,10,20]	[5,10,20]
max_depth	[5,7,9]	[5,7,9]
min_samples_split	[40,50]	[40,50]
cv	3	3
eval_Score	R2	R2

Metric	XGB	LightGBM
MSE	103672.2	103515.51
RMSE	321.98	321.74
R2	0.6933	0.6938
Adjusted_R2	0.6857	0.6862

Final Metrics Conclusion

Algorithms	Test MSE	Test RMSE	Test R2	Train Adjusted R2
Linear Regression	0.005472	0.073978	0.495791	0.495312
Lasso Regression	0.005470	0.073965	0.495963	0.495484
Ridge Regression	0.005470	0.073965	0.495966	0.495486
Decision Tree Regressor	0.004235	0.065081	0.609772	0.609401
XGB Regressor	0.003144	0.056076	0.710290	0.710014
Gradient Boosting	0.003121	0.055870	0.712415	0.712142
Light GBM	0.003418	0.058470	0.685021	0.684722

Conclusion

- In this project, we tried to predict the trip duration of a taxi in NYC.
- We are mostly concerned with the information of pick up latitude and longitude and drop off latitude and longitude, to get the distance of the trip.
- Hyperparameter tuning doesn’t improve much accuracy.
- Linear regression gives 60.89 % accuracy,XGBoost gives 68.56% accuracy, LightGBM gives 71.42% on the test set.
- LightGBM is more fitter and efficient than XGBoost for taxi trip duration-based predictions
- LightGBM will be the best model to predict the trip duration for a particular taxi.

Challenges

- Handling Large Dataset
- Feature Engineering
- Computation Time
- Optimising The Model

<u>THANKYOU</u>