NYC Taxi Time Prediction

By Saurabh Aradwad, Aman Guleria,
Rishika Rai , Ayush Sharma
Data science trainees
AlmaBetter, Bangalore

1. Abstract

The Taxi and Limousine Commission (TLC), established in 1971, is in charge of licencing and regulating New York City's Medallion (Yellow) taxi cabs, for-hire vehicles (community-based liveries, black cars, and luxury limousines), commuter vans, and paratransit vehicles.Every day, approximately 1,000,000 trips are completed by over 200,000 TLC licensees. Drivers who work for hire must first pass a background check, have a clean driving record, and complete 24 hours of driver training. TLC-licensed vehicles are inspected at TLC's Woodside Inspection Facility for safety and emissions.

The dataset is based on data from the 2016 NYC Yellow Cab trip records, which were made available in Big Query on Google Cloud Platform. The NYC Taxi and Limousine Commission first made the data public (TLC). For the purposes of this project, the data was sampled and cleaned. We should predict the duration of each trip in the test set based on individual trip attributes.

2. Problem Statement

To build a model that predicts the total ride duration of taxi trips in New York City. 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.

3. Introduction

In order to forecast the typical journey time and fare for a given Pickup location, Drop location, Date, and Time, the primary goal of this post is to build a fundamental machine learning model. In order to gain an advantage over rival businesses and give customers more value, every organisation today needs to

use data effectively. With the use of tools like data prep and sklearn, even non-programmers without prior domain knowledge or coding experience may create models for machine learning systems.

4. 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 metre was engaged
- **dropoff_datetime** — date and time when the metre was disengaged
- **passenger_count** — the number of passengers in the vehicle (driver entered value)
- **pickup_longitude** — the longitude where the metre was engaged
- **pickup_latitude** — the latitude where the metre was engaged
- **dropoff_longitude** — the longitude where the metre was disengaged
- **dropoff_latitude** — the latitude where the metre 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

4. Exploratory Data Analysis

- DataPrep library is a nice tool for basic EDA and EDA report creation hence its use here.
- Data Set is clean, there are no missing values.
- Some important correlations and countplots are shown as follows.

```
plot_missing(df)
```

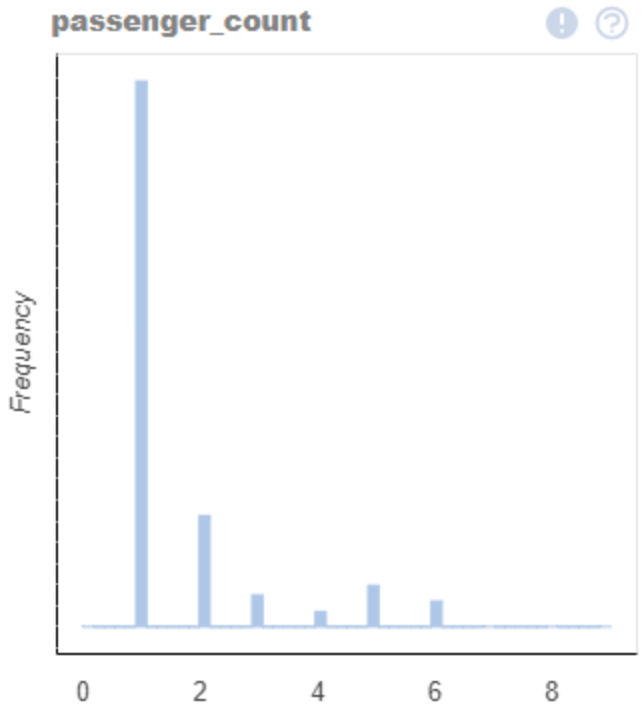
Stats | Bar Chart | Spectrum | Heat Map | Dendrogram

Missing Statistics

Missing Cells	0
Missing Cells (%)	0.0%
Missing Columns	0
Missing Rows	0
Avg Missing Cells per Column	0.0
Avg Missing Cells per Row	0.0

Dataset Statistics

Number of Variables	11
Number of Rows	1.4586×10^{06}
Missing Cells	0
Missing Cells (%)	0.0%
Duplicate Rows	0
Duplicate Rows (%)	0.0%
Total Size in Memory	461.8 MB
Average Row Size in Memory	332.0 B
Variable Types	Categorical: 5 Numerical: 6



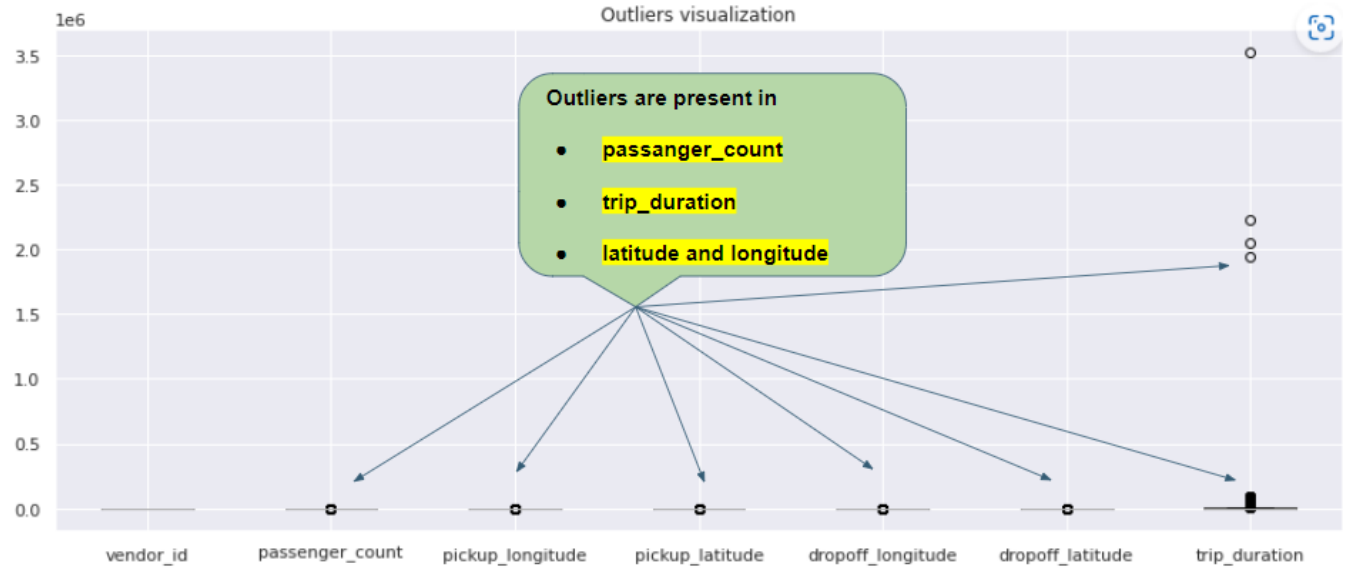
Dataset Insights

vendor_id has constant length 1	Constant Length
pickup_datetime has constant length 19	Constant Length
dropoff_dateti… has constant length 19	Constant Length
store_and_fwd_… has constant length 1	Constant Length
id has all distinct values	Unique
pickup_longitu… has 1458644 (100.0%) negatives	Negatives
dropoff_longit… has 1458644 (100.0%) negatives	Negatives

1 2

- passenger_count is having maximum count frequency at 1.
- vender_id and store_and_fwd_flag have constant length.
- id consists of all unique values; it means no duplicate data.

5. Dealing with Outliers



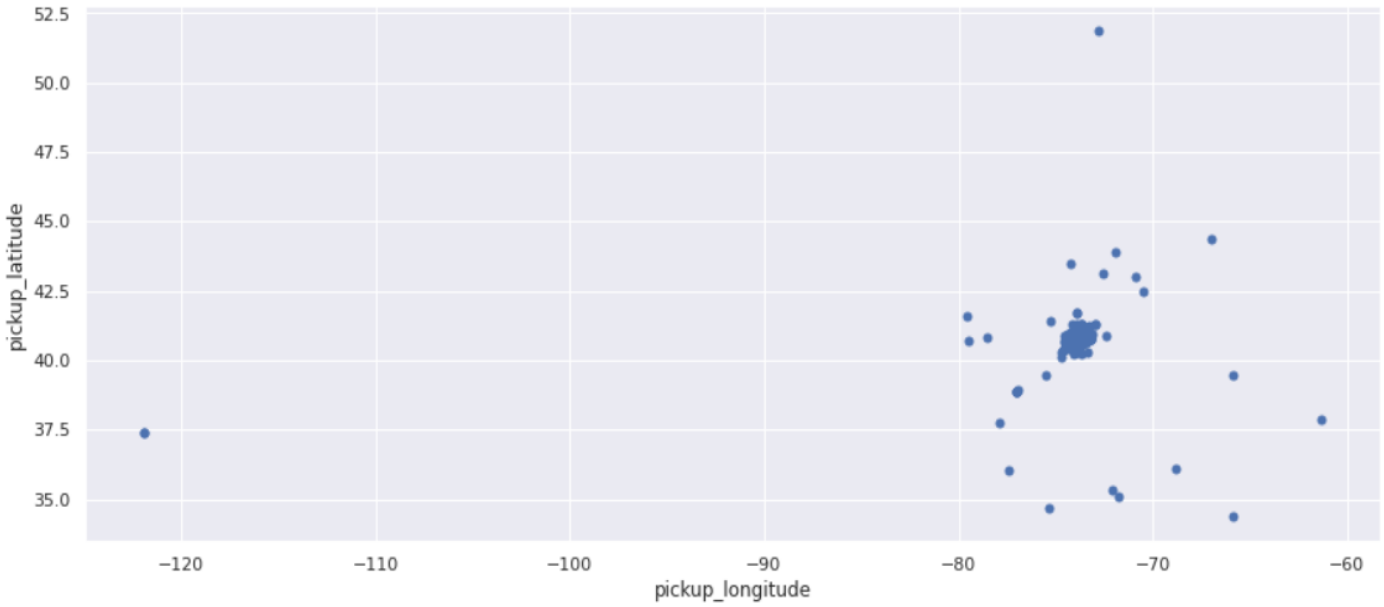
a) Trip Duration Outliers

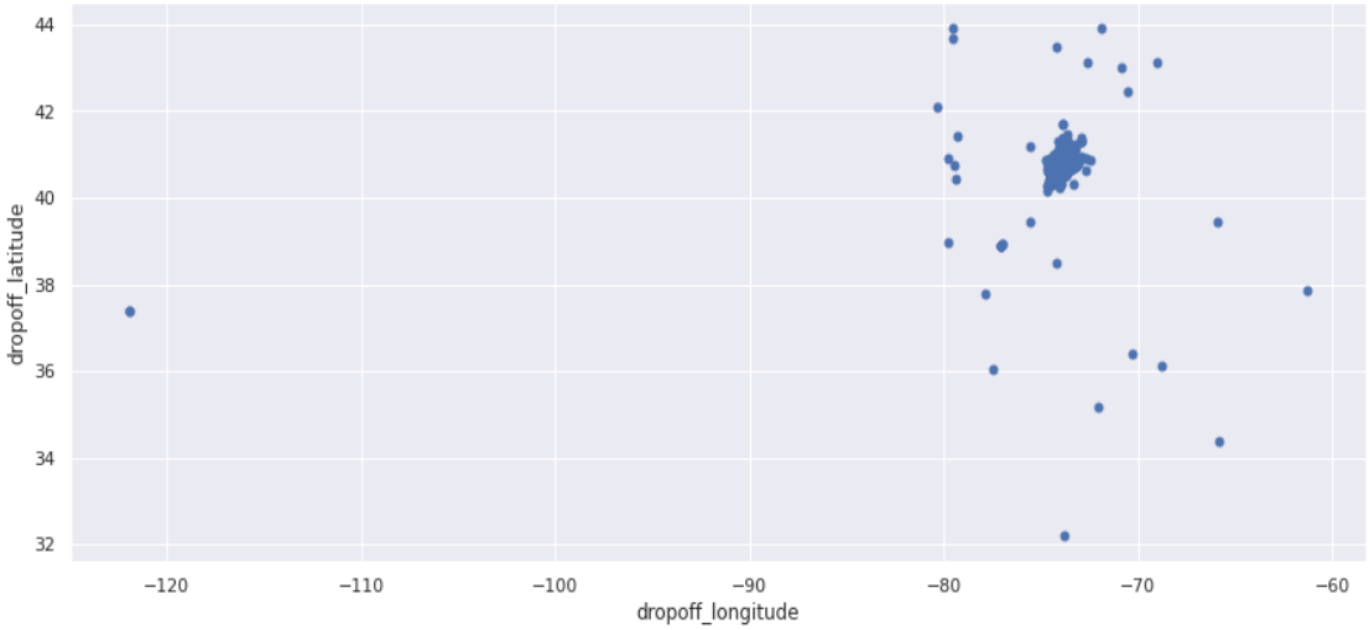
- There are outliers for our target value `trip_duration`.
- We can't find a proper interpretation and it will probably damage our model, so let's get rid of them.

b) Passenger Count Outliers

- Maximum count is shown with single passenger
- Let's get the actual trip_count w.r.t. number of passengers.

c) Outliers w.r.t latitude and longitude

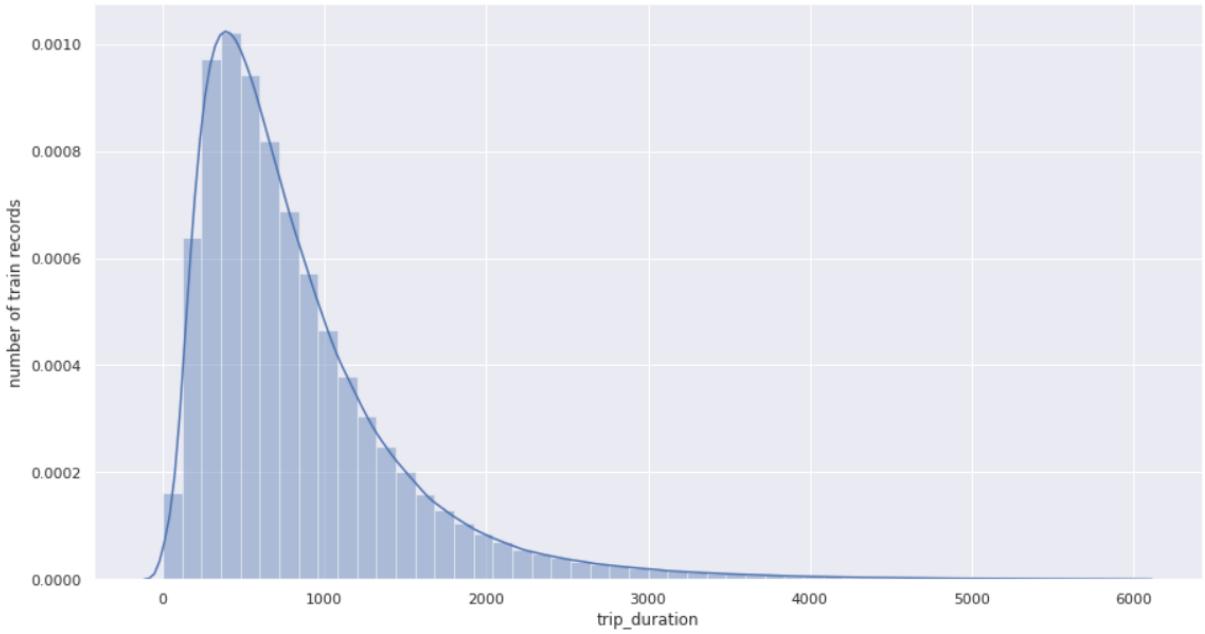




- There are some minor outliers in both pickup and dropoff data.
- It is critical for our target value prediction to remove outliers in coordinate data.

6. Feature Engineering

a) Attacking Target Values



- The distribution is right-skewed so we can consider a log-transformation of the `trip_duration` column.

b) One Hot Encoding

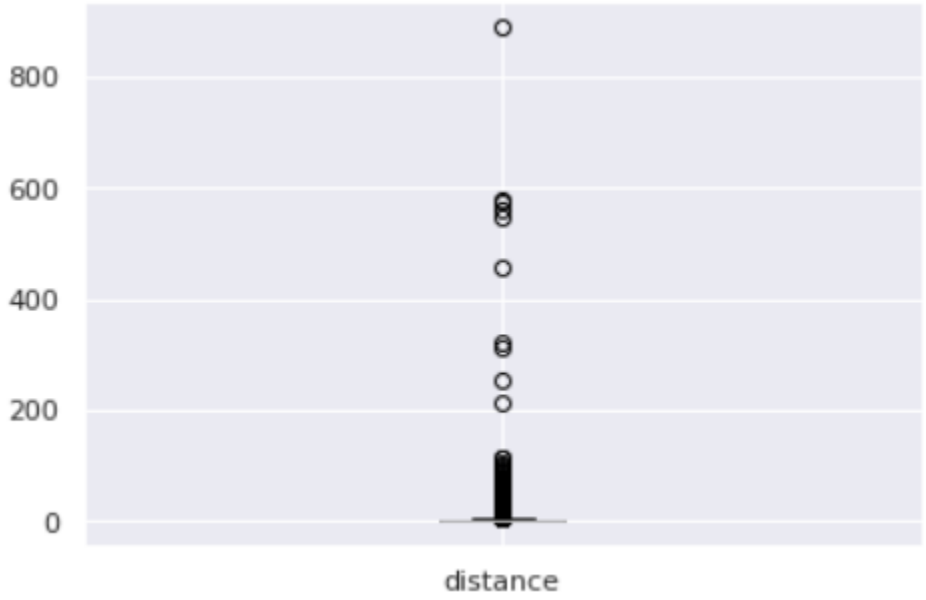
- object type data is duplicated and converted into uint8 Data type.
- Although these columns `store_and_fwd_flag` and `vendor_id` are not relevant to our target value, still it's better to treat them accordingly.

c) Dealing with Dates

- pickup_datetime` is an important feature that must be handled with pandas datetime extraction.
- We extracted the month, week, weekday, hour, and minute of the day as our important features using `pickup_datetime`.
- This information is critical for our regression model, so it is presented in integer format.
- We have removed the `dropoff_datetime` column because we need to predict the trip duration.

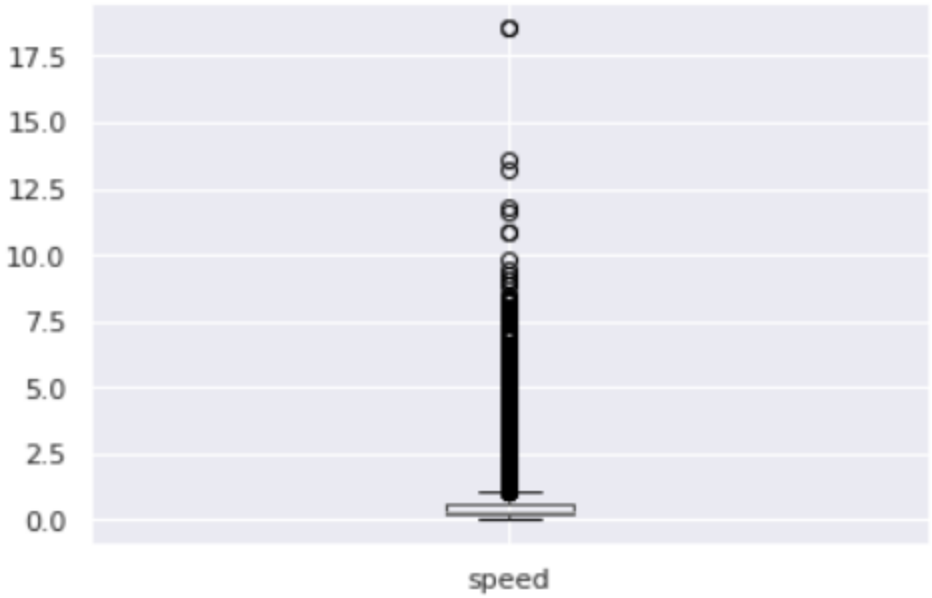
d) Distance and Speed Calculation

- During dataset research, we found that the pickup_longitude, pickup_latitude, dropoff_longitude & dropoff_latitude have coordinates in the form of longitude and latitude. But from it, we really can't infer anything or come to any conclusions.
- Therefore, distance is the most obvious feature we can draw from this. Let's carry it out.
- Geopy can calculate geodesic distance between two points using the geodesic distance or the great-circle distance, with a default of the geodesic distance available as the function
- For more Details refer link



- Again for distance we have some outliers. They may affect our model accuracy and efficiency hence we are going to remove them.
- We will keep observations which have distance > 200 for this specific model.

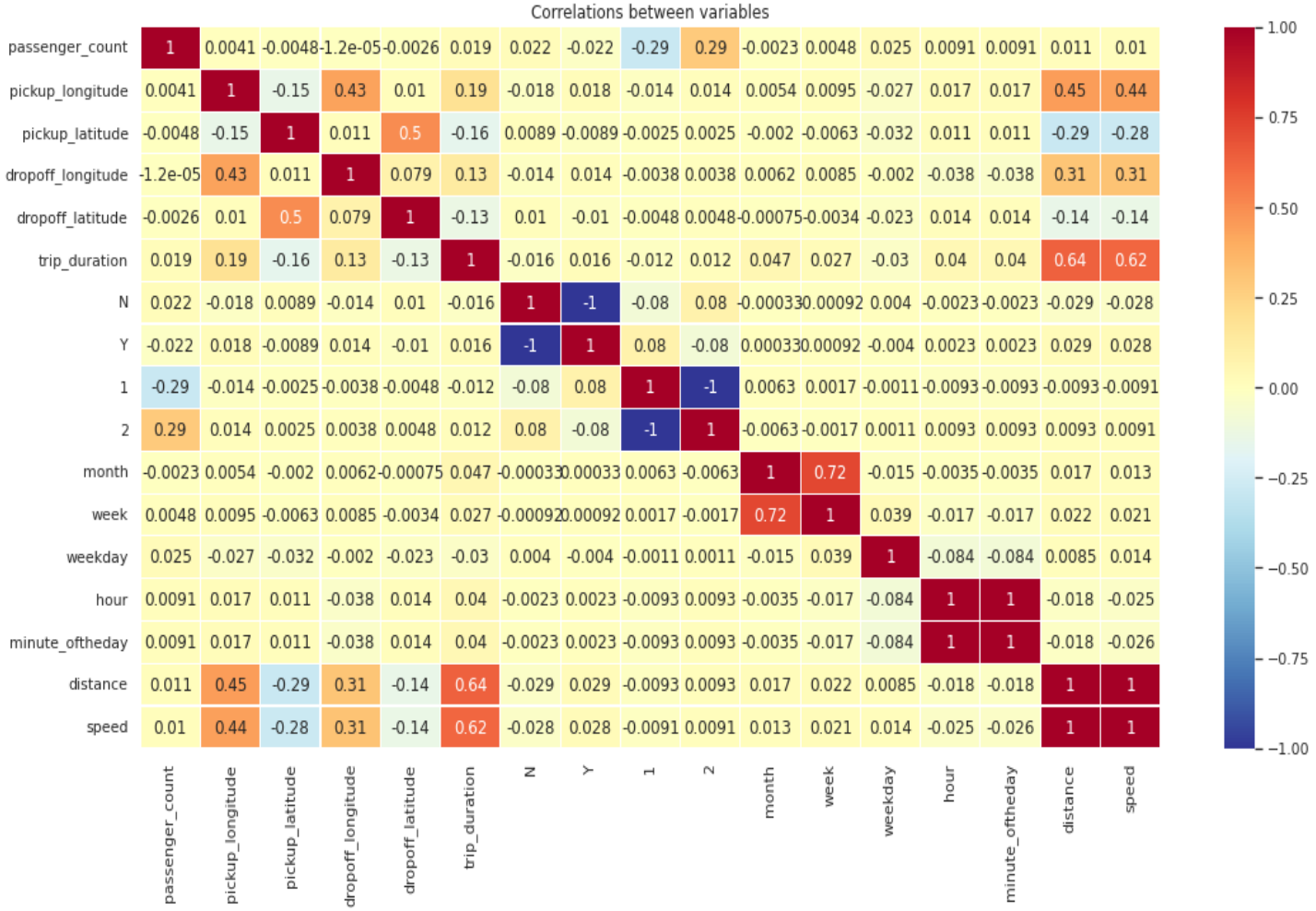
After this we have one more feature to create that is speed



We know speed = distance / time , this simple equation we will consider for the calculator of speed.

- From the above plot we can see that there are some outliers in the speed, we can remove them for better model performance.

7. Correlation Analysis



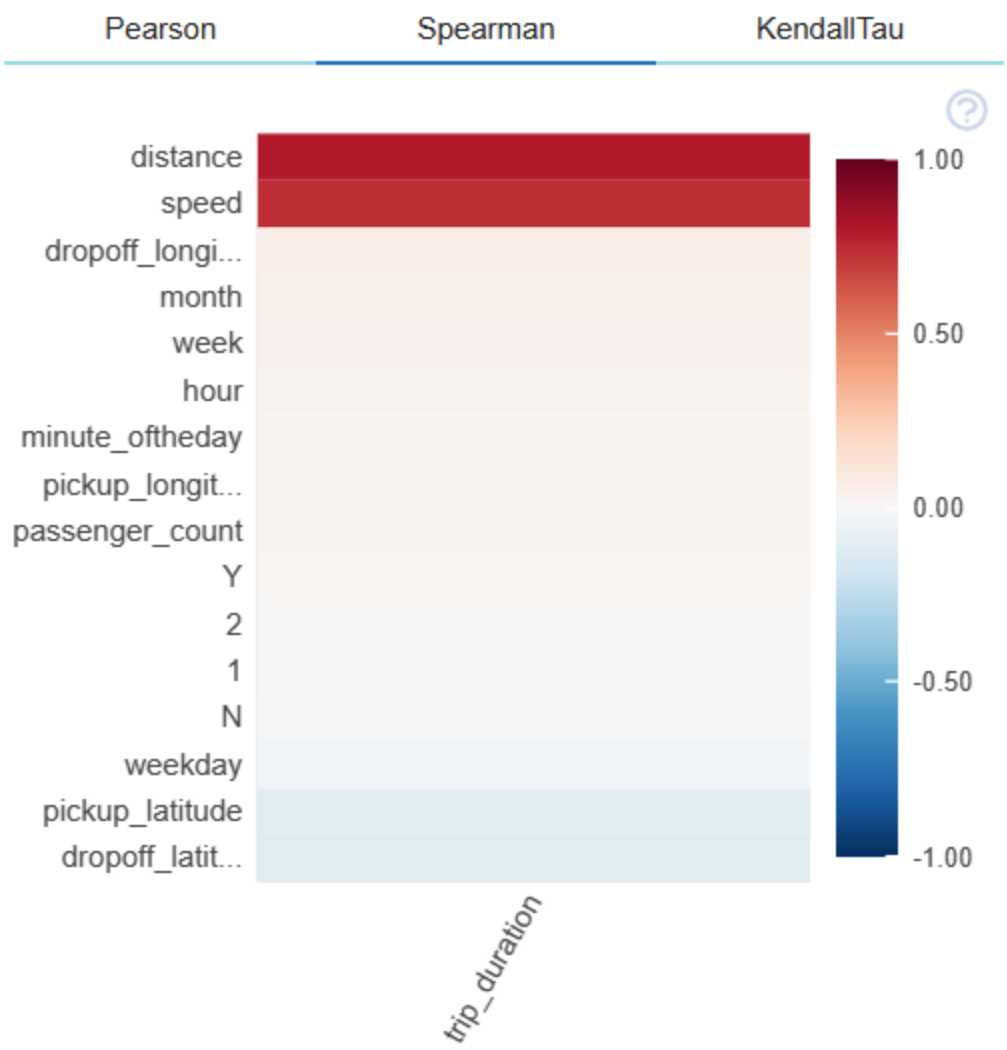
We can see this heat map shows correlation between all features with themselves.

- ❖ Our target variable is trip duration
- ❖ Maximum correlation is obtained for distance and speed i.e 0.64 and 0.62 respectively.
- ❖ These features are more critical for our regression model.

We will plot the graph for this correlation separately using DataPreps correlation plot method.

- ❖ This method takes dataframe and trip duration as input parameters.

Following fig shows the correlation for trip duration with all the other features.



For this graph also we have same values for correlation

- Distance = 0.64
- Speed = 0.62

8. Preparing Dataset For Modelling

a) Split

After correlation analysis the next task is to split the given data into train and test dataset. Before we split our data we will be dropping out trip duration from our dataframe as we are going to predict it in further steps.

The dataset is splitted at test_size = 0.2 and random state = 42

For x.shape and y.shpe we have (1455968, 16), (1455968,) respective values.

b) Metrics

❖ We used sklearn metrics for metrics, importing mean squared error, mean absolute error, and r2 score. With these, our models will be more accurate and avoid unnecessary errors.

9. Regression Models

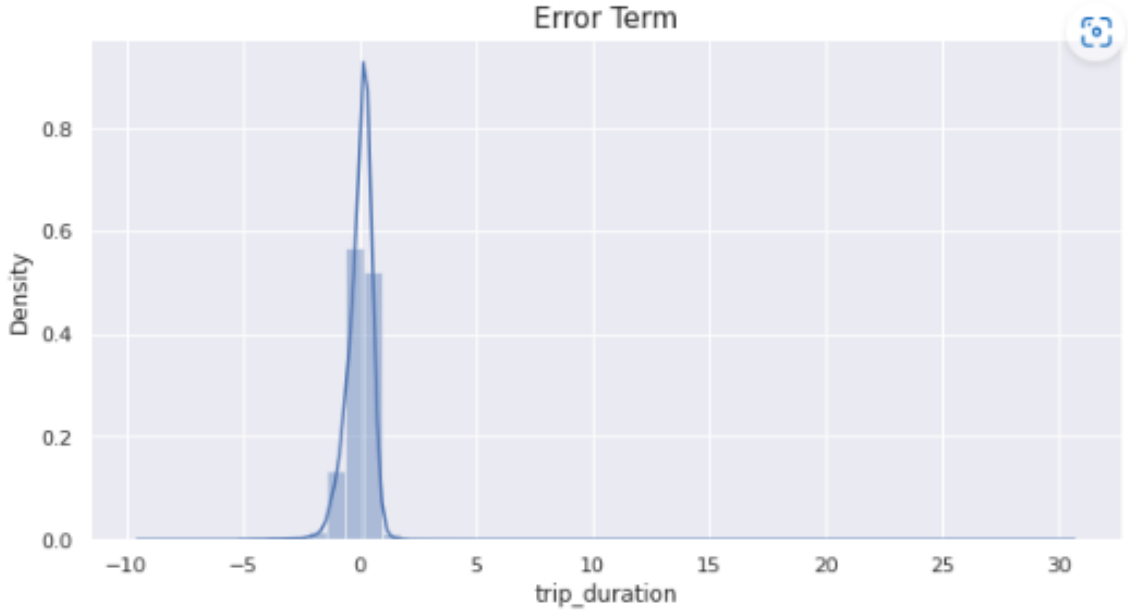
a) Try Different Models

Creating a reusable function ⚡ for different models.

- Step 1 : Fit the training data with fitting X_train and y_train.
- Step 2 : For the train and test datasets, use the `.score(X_train, y _train)` & `.score(X_test, y_test)` method to obtain the model score.
- Step 3 : Evaluate the root of the mean squared error w.r.t test and predicted values.
- Step 4 : Visualise the error distribution.

1) Linear Regression

```
Model score for train dataset is: 0.4577088802126631
Model score for test dataset is: 0.4581687677554299
Squareroot of MSE for model according to test dataset is: 0.5730803091406814
```




```
CPU times: user 3.4 s, sys: 686 ms, total: 4.09 s
Wall time: 3.06 s
```

Train Score: 0.4577088

Test Score: 0.4581687

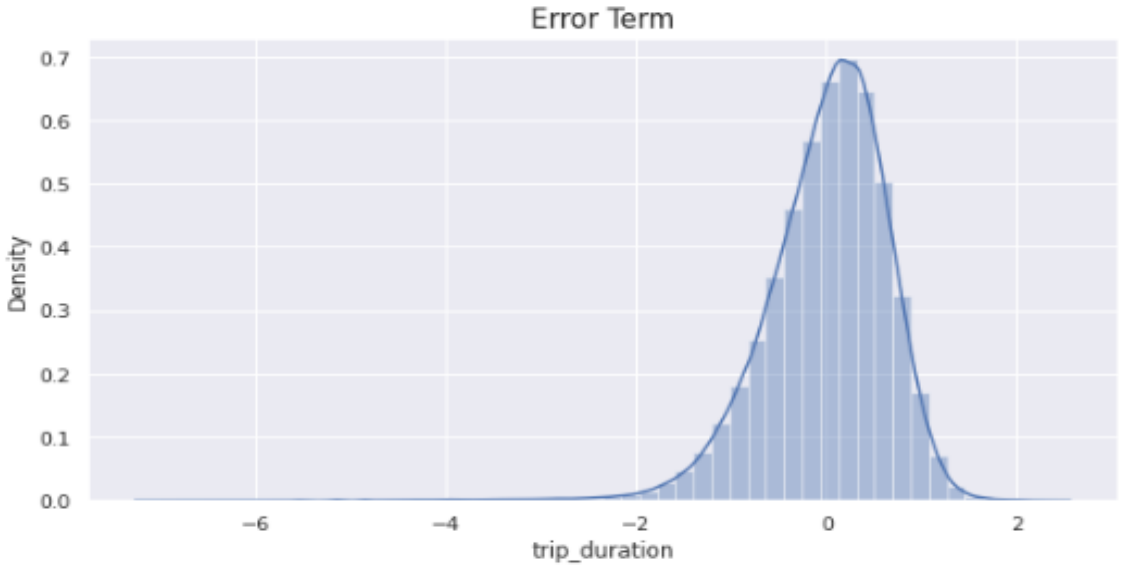
MSE Error: 0.5730803

Wall Time: 3.06 seconds

For this trained model train and test score are very less. The mean squared error is also high hence this model we cannot use for predicting our trip duration.

2) Lasso Regression

```
Model score for train dataset is: 0.30253789666998965
Model score for test dataset is: 0.30157200430059083
Squareroot of MSE for model according to test dataset is: 0.6506452874738975
```



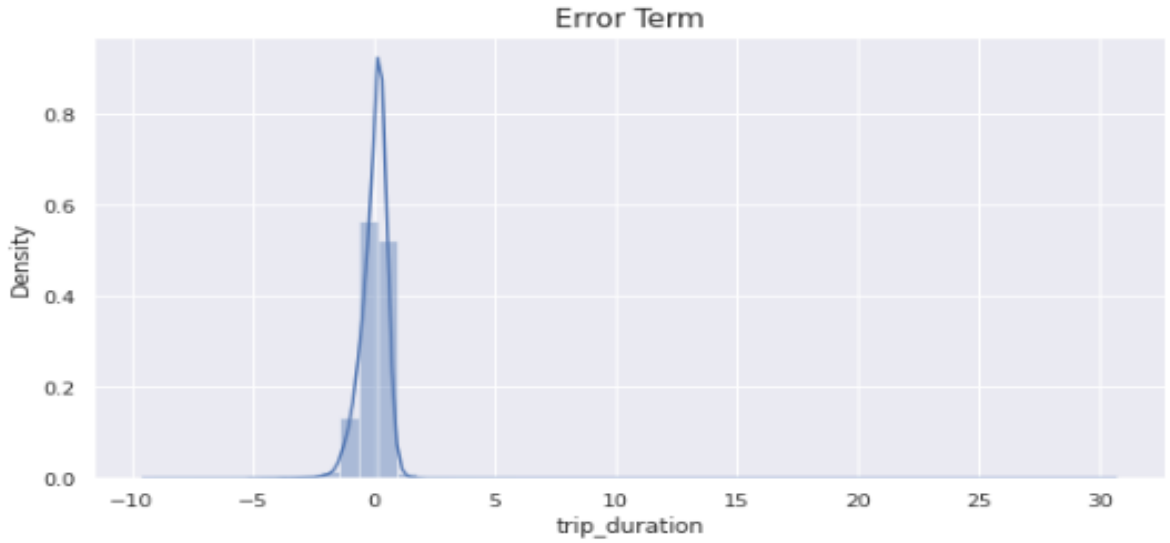

```
CPU times: user 2.72 s, sys: 516 ms, total: 3.23 s
Wall time: 2.58 s
```

Train Score:	0.302537
Test Score:	0.301572
MSE Error:	0.650645
Wall Time:	2.58 seconds

For this trained model train and test score are very less. The mean squared error is also high hence this model we cannot use for predicting our trip duration.
This model is the worst performing model.

3) Ridge Regression

```
Model score for train dataset is: 0.45770885692302576
Model score for test dataset is: 0.4581699664631613
Squareroot of MSE for model according to test dataset is: 0.5730796752198807
```



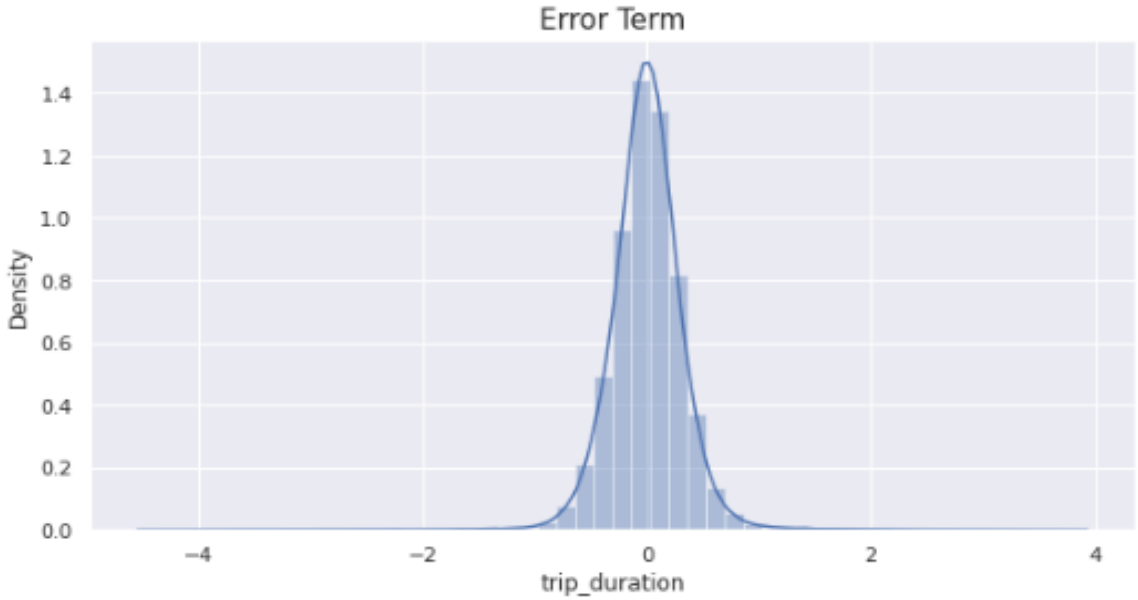

```
CPU times: user 2.4 s, sys: 545 ms, total: 2.95 s
Wall time: 2.33 s
```

Train Score:	0.4577088
Test Score:	0.4581699
MSE Error:	0.5730796
Wall Time:	2.33 seconds

For this trained model train and test score are very less. The mean squared error is also high hence this model we cannot use for predicting our trip duration.

4) Gradient Boosting

```
Model score for train dataset is: 0.8076601991528932
Model score for test dataset is: 0.807398428468072
Squareroot of MSE for model according to test dataset is: 0.34167514090485257
```



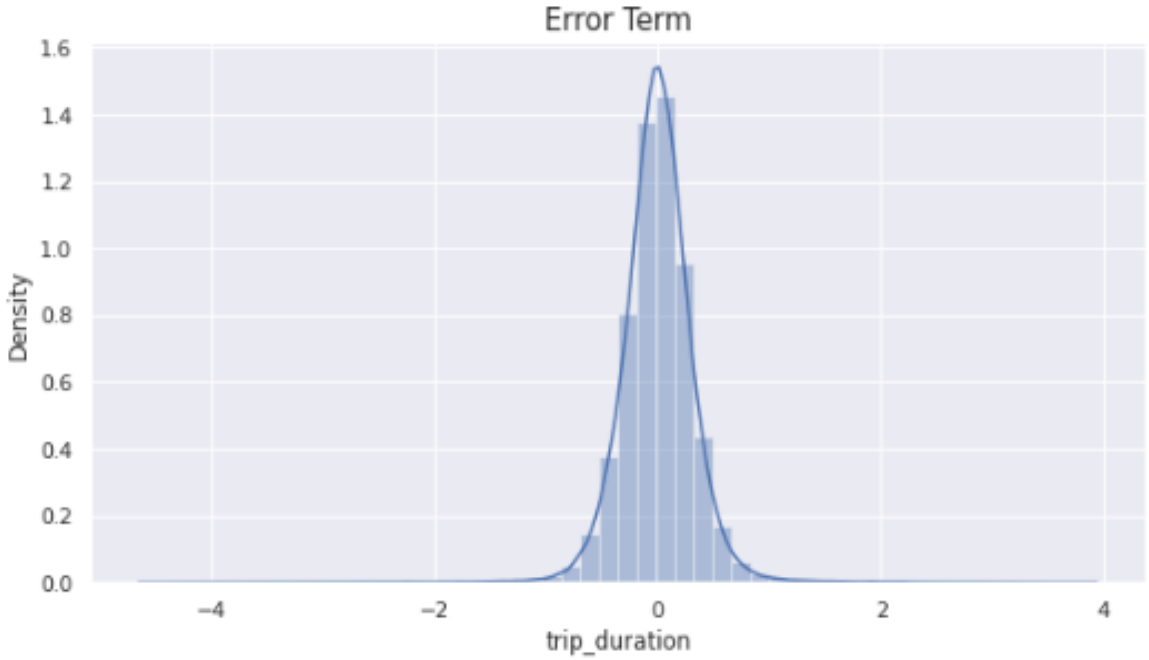

```
CPU times: user 10min 44s, sys: 1.25 s, total: 10min 45s
Wall time: 11min 28s
```

Train Score:	0.80766
Test Score:	0.80739
MSE Error:	0.341675
Wall Time:	11 min 28 seconds

For this trained model train and test scores are high compared to linear models. The mean squared error is also low hence this model we can use for predicting our trip duration .

5) XGBoost Regression

```
Model score for train dataset is: 0.8099197031292716
Model score for test dataset is: 0.8097427058647245
Squareroot of MSE for model according to test dataset is: 0.3395894009891341
```




```
CPU times: user 3min 18s, sys: 772 ms, total: 3min 19s
Wall time: 3min 25s
```

Train Score: 0.809919

Test Score: 0.809742

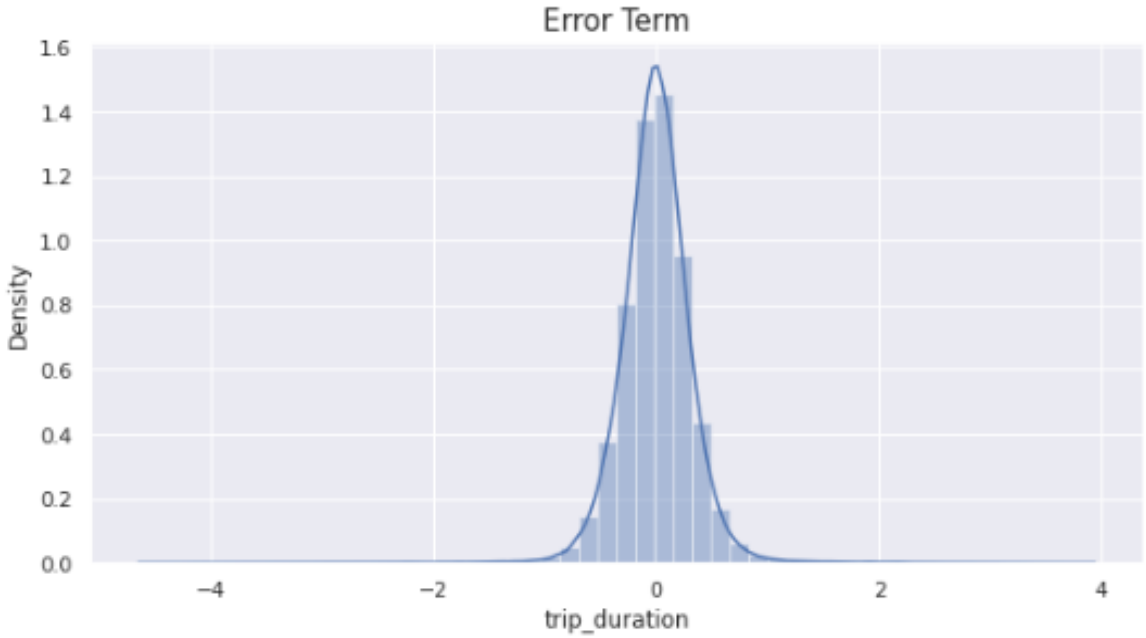
MSE Error: 0.339589

Wall Time: 3 min 25 seconds

For this trained model train and test scores are high compared to linear models. The mean squared error is also low hence this model we can use for predicting our trip duration .

6) LGB Regression

```
Model score for train dataset is: 0.8099197031292716
Model score for test dataset is: 0.8097427058647245
Squareroot of MSE for model according to test dataset is: 0.3395894009891341
```




```
CPU times: user 3min 15s, sys: 700 ms, total: 3min 16s
Wall time: 3min 26s
```

Train Score:	0.809919
Test Score:	0.809742
MSE Error:	0.339589
Wall Time:	3 min 26 seconds

For this trained model train and test scores are high compared to linear models. The mean squared error is also low hence this model we can use for predicting our trip duration.

This model shows the same score as our XGB model.

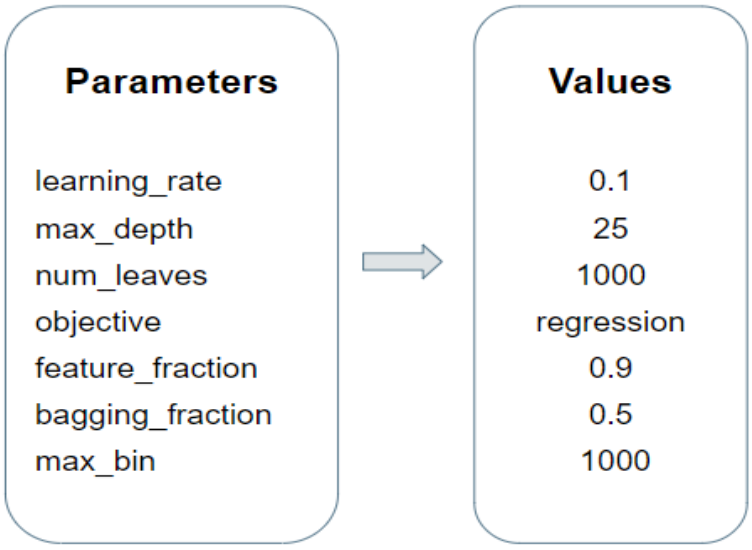
b) Model Comparison

	Model Name	Train MSE	Test MSE	Train R^2	Test R^2	Train Adjusted R^2	Test Adjusted R^2
0	LinearRegression()	0.326898	0.328421	0.457709	0.458169	0.457701	0.458139
1	Lasso()	0.420436	0.423339	0.302538	0.301572	0.302528	0.301534
2	Ridge()	0.326898	0.328420	0.457709	0.458170	0.457701	0.458140
3	GradientBoostingRegressor()	0.115944	0.116742	0.807660	0.807398	0.807658	0.807388
4	XGBRegressor()	0.114582	0.115321	0.809920	0.809743	0.809917	0.809732
5	LGBMRegressor()	0.042033	0.042141	0.930271	0.930475	0.930270	0.930472

Important takeouts

- MSE -This error is minimum for LGBM i.e. 0.042 equivalent to 4%.
- R2 - LGBM has the highest score 0.93 for both train and test dataset.
- Adjusted R2 - LGBN have highest accuracy about 93%.
- Best model - LGBM with 93% accuracy at our adjusted R2 score and with least error at MSE 0.042.

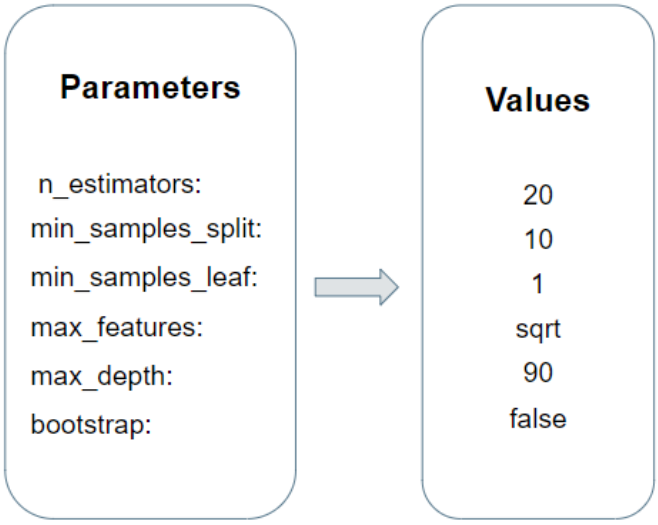
c) Cross-Validation



- Cross validation is required to ensure that the LGBMRegressor provides nearly identical accuracy.

- We used these parameters to test the model CV score for this LGMB model, which is cross validating 5 times.
- Maximum CV score : 0.93

d) Hyperparameter Tuning



- For Random grid n_estimators, max_features, max_depth, min_sample_split, max_sample_fit, bootstrap were used.
- Model was fitted and best_params_ were obtained
- With hyperparameter tuning and randomizedCVsearch, we can find similar CV scores with less training time.

e) Finalising Model

- When we run the LGB model with the default parameters and a slow learning rate, the error is very low and the only disadvantage is that it takes time.
- The total time taken by the LGB model with the default parameters is 9 minutes and 19 seconds.
- However, if we run this model with the best parameters in terms of CV score 5, we can get almost identical results while saving time.
- The total time taken by the LGM model with its best parameter is 5.95 seconds.

f) Results

	id	trip_duration	trip_duration_predicted	trip_duration_predicted_	percentage_error	percentage_error_
0	id2875421	455	452.147417	452.383246	0.626941	0.575111
1	id2377394	663	631.190911	661.107742	4.797751	0.285408
2	id3858529	2124	1328.808197	2128.534488	37.438409	0.213488
3	id3504673	429	439.678331	440.357180	2.489121	2.647361
4	id2181028	435	435.167455	435.438508	0.038495	0.100806

- With 93% accuracy from LGBMRegressor, our best taxi time prediction is here, and when compared to actual trip duration, we can see how close it is.
- percentage_error is more for when the model is trained fast with best parameters obtained with hyperparameter tuning and error is less if trained slowly(with less learning rate) with default LGBMs' parameters.

10. Conclusion

- LGBM models have the highest accuracy of 93% when compared to other models.
- The least accurate solutions for this problem are linear, lasso, and ridge regulation.
- Gradient Boost took a moderate time of 8 minutes and 7 seconds, while LGBM took the shortest time of
- LGBM's low learning rate increases accuracy while decreasing percentage error.
- If we want to reduce the training time of our model, we can use hyperparameter tuning to select tuned parameters. It will produce similar results in much less time.
- Logarithm treatment of trip duration is important, due to skewed data.
- Speed and distance calculation. We can confirm this with correlation heatmap and skewed.
- This trip duration prediction project is important for many cab service providers such as ola, uber etc. It helps them to improve their customer experience by showing expected trip duration time and arrival time.
- It will be helpful if you select a certain time from the day and select pickup and dropoff coordinates for your future trips and get a prediction for trip duration.