Capstone Project

NYC Taxi Trip Duration

Individual

Bhaskar Mendhe

Points For Discussion

- About TLC
- Project Road Map
- Business Problem
- Purpose Of The Project
- Data Pipeline
- Data Summary
- Fixing The Outliers
- Feature Engineering
- Exploratory Data Analysis
    - Univariate Analysis
    - Bivariate & Multivariate Analysis
- Selection of the Features
- Preparing Dataset
- Machine Learning Models
- Model Evaluation And Selection
- Conclusion

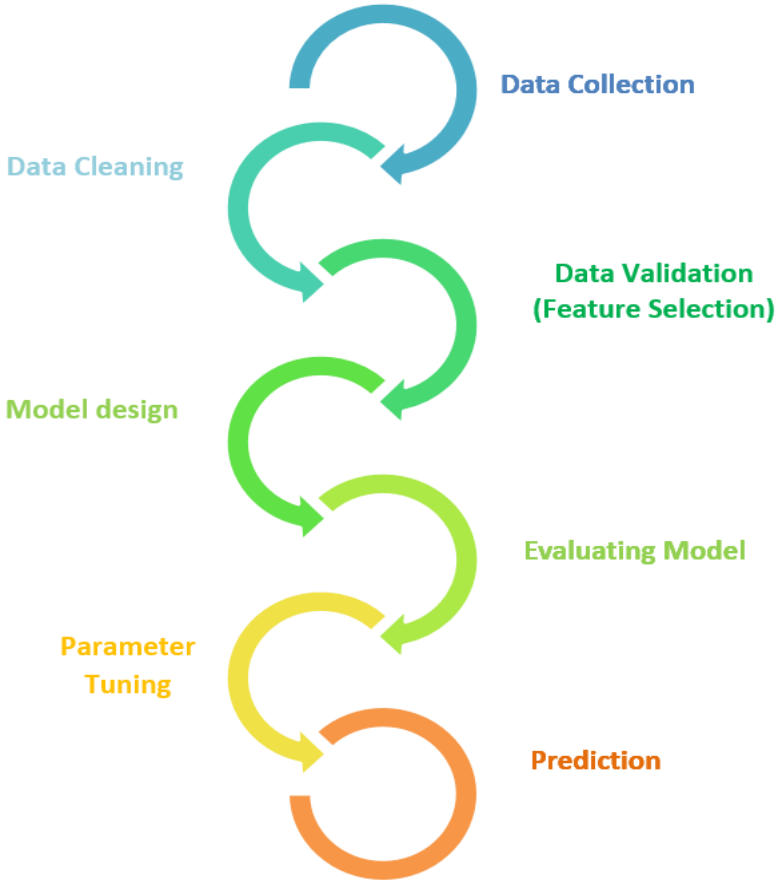
Taxi And Limousine Commission

- The New York City Taxi and Limousine Commission (TLC), created in 1971, is the agency responsible for licensing 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. Over 200,000 TLC licensees complete approximately 1,000,000 trips each day.

Yellow Cab

- Taxicabs are the only vehicles that have the right to pick up street-hailing and prearranged passengers anywhere in New York City. By law, there are 13,587 taxis in New York City and each taxi must have a medallion affixed to it. Other vehicles are For-Hire vehicles, Green cabs, Commuter vans, Paratransit vehicles are also in the service.

Project Road Map



Business Problem

The New York City Taxi and Limousine Commission wants to predict the trip duration of their taxi rides.

Purpose Of The Project

The purpose of this project is to build the model which will help to predict the trip durations of NYC taxis.

Data Pipeline

- Data Processing : In this we have explored the dataset and tried to find some inconsistencies also we took proper actions on them. We have looked for missing and null values Some of the data which were needed to be changed in required formats to retrieve some useful data. We also created some new features.

- EDA : In this we have analysed the data and tried to clean and filter it. Also, we found out some outliers and removed them which were causing inconsistency in dataset. As the need arrised we modified the data and drew some useful trends from it.

- Data Preparation : After cleaning and filtering the data we prepared the data for regression model by removing features which were highly correlated with each other. Then we selected the required features and standardized them for making them ready for the application of machine learning model.

- Regression Modelling : After preparation of data we applied types of regression model on the dataset as it is an iterative process. We started with simple regression model and after that other complex one also evaluated them for best one to choose.

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 forward 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.

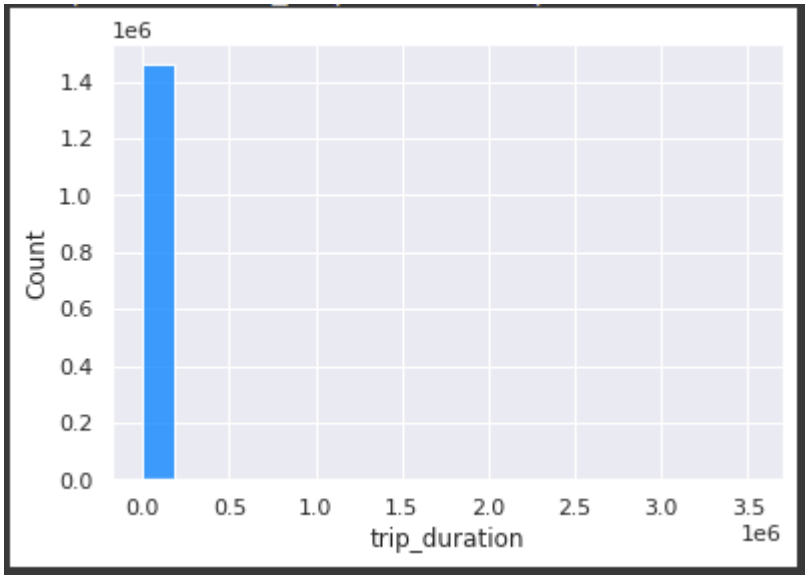
Fixing The Outliers

```
#Description of the data
nyct.describe()
```

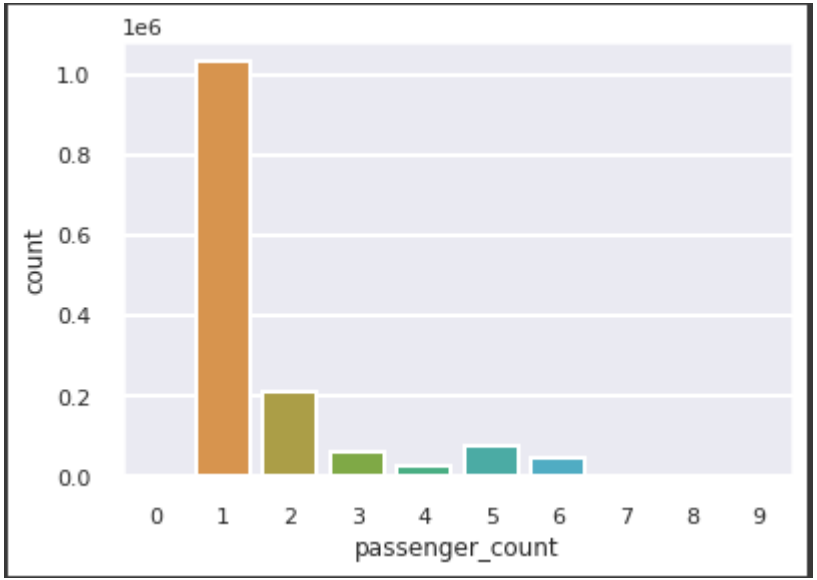
	vendor_id	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	trip_duration
count	1.458644e+06	1.458644e+06	1.458644e+06	1.458644e+06	1.458644e+06	1.458644e+06	1.458644e+06
mean	1.534950e+00	1.664530e+00	-7.397349e+01	4.075092e+01	-7.397342e+01	4.075180e+01	9.594923e+02
std	4.987772e-01	1.314242e+00	7.090186e-02	3.288119e-02	7.064327e-02	3.589056e-02	5.237432e+03
min	1.000000e+00	0.000000e+00	-1.219333e+02	3.435970e+01	-1.219333e+02	3.218114e+01	1.000000e+00
25%	1.000000e+00	1.000000e+00	-7.399187e+01	4.073735e+01	-7.399133e+01	4.073588e+01	3.970000e+02
50%	2.000000e+00	1.000000e+00	-7.398174e+01	4.075410e+01	-7.397975e+01	4.075452e+01	6.620000e+02
75%	2.000000e+00	2.000000e+00	-7.396733e+01	4.076836e+01	-7.396301e+01	4.076981e+01	1.075000e+03
max	2.000000e+00	9.000000e+00	-6.133553e+01	5.188108e+01	-6.133553e+01	4.392103e+01	3.526282e+06

The data had some outlier. Passenger count and Trip duration had outliers, passenger count has 0 count for passenger but the trips were generated for them it must be because of software error which we fixed minimum passenger count to 1 and maximum 6. Trip duration has outliers which was 3526282 seconds which was an outlier affecting the model so we restricted the trip duration as required.

Fixing The Outliers



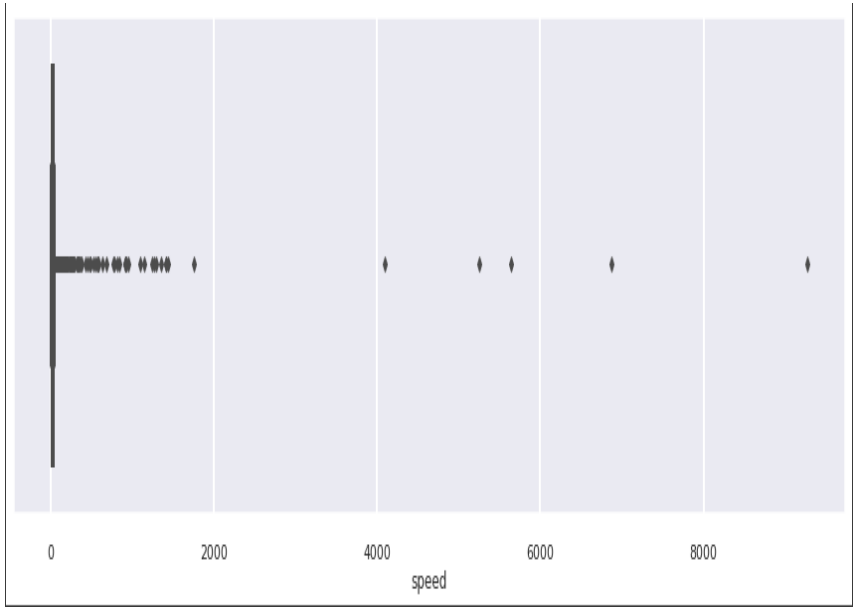
As we can see there is an outlier in trip duration also it is highly right skewed. It contained some duration more 5000 sec and some duration as low as 1 sec so we fixed the outliers for the consistency of the data.



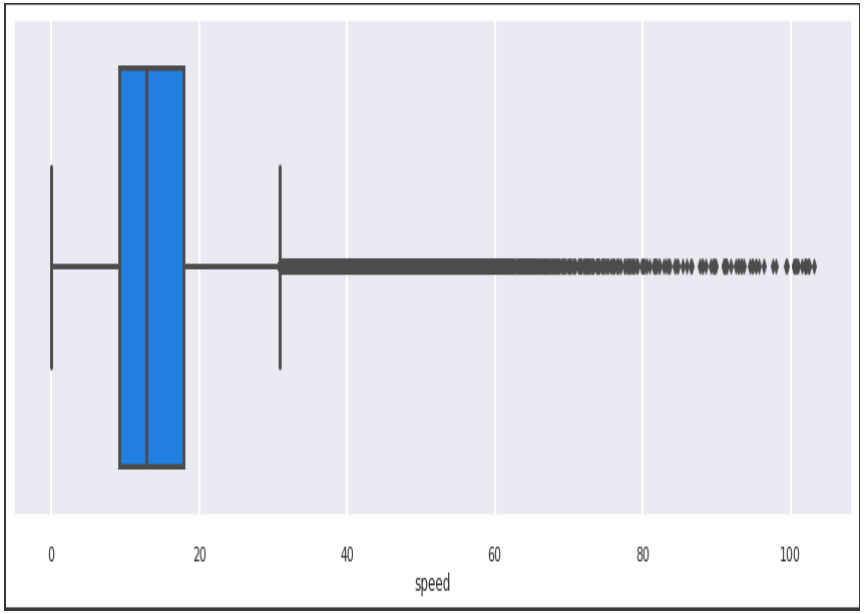
In trip duration as we can see the passenger count 0 has trips also passenger count of 7, 8, and 9 rarely has some trips so we dropped them for the consistency of the data.

Fixing The Outliers

Speed



Removing trips with the speed of over 200 km/hr for data consistency.



Most of the trips are done under 0-20 km/hr and above 30 km/hr.

Feature Engineering

We have created the following features :

- Pickup day & Dropoff day : Contains the name of the weekdays.
- Pickup day no & Dropoff day no: Converts weekdays into numbers that starts at Monday = 0 and ends at Sunday = 6.
- Pickup hour & Dropoff hour : Extracting hour from datetime columns.
- Pickup minute & Dropoff minute : Extracting minutes from datetime columns.
- Pickup month & Dropoff month : Extract month in number
- Distance : Calculate the distance with the help of latitude and longitude (km).
- Speed : Calculated speed distance/trip duration (km/hr).
- Day part : Dividing 24 hours into four parts
    - Morning (from 6:00 am to 11:59 A.M.)
    - Afternoon (from 12 noon to 3:59 P.M)
    - Evening (from 4:00 pm to 9:59 P.M.)
    - Late Night (from 10:00 pm to 5:59 A.M.)

Exploratory Data Analysis

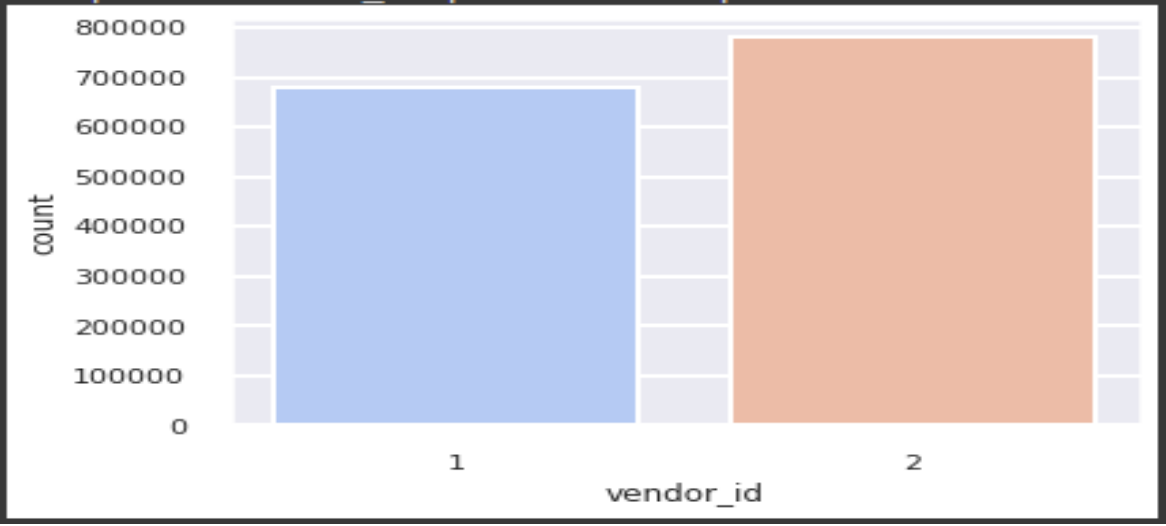
Univariate analysis : Univariate analysis explores each variable in a data set, separately. It looks at the range of values, as well as the central tendency of the values. It describes the pattern of response to the variable. It describes each variable on its own. Descriptive statistics describe and summarize data.

Bivariate Analysis : Bivariate analysis is stated to be an analysis of any concurrent relation between two variables or attributes. This study explores the relationship of two variables as well as the depth of this relationship to figure out if there are any discrepancies between two variables.

Vendor Id

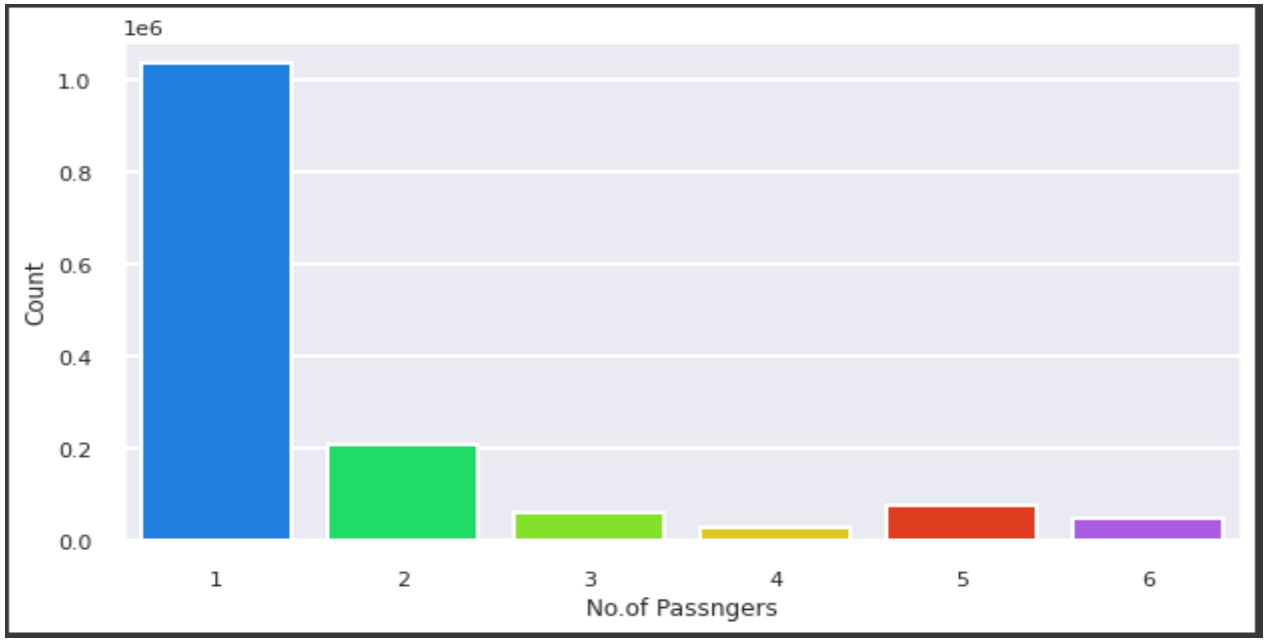
```
#Ploting count plot for vendor_id
sns.set_context('poster', font_scale = 0.5)
sns.countplot(x ='vendor_id', data = nyct , palette ='coolwarm')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fda3098b5d0>
```



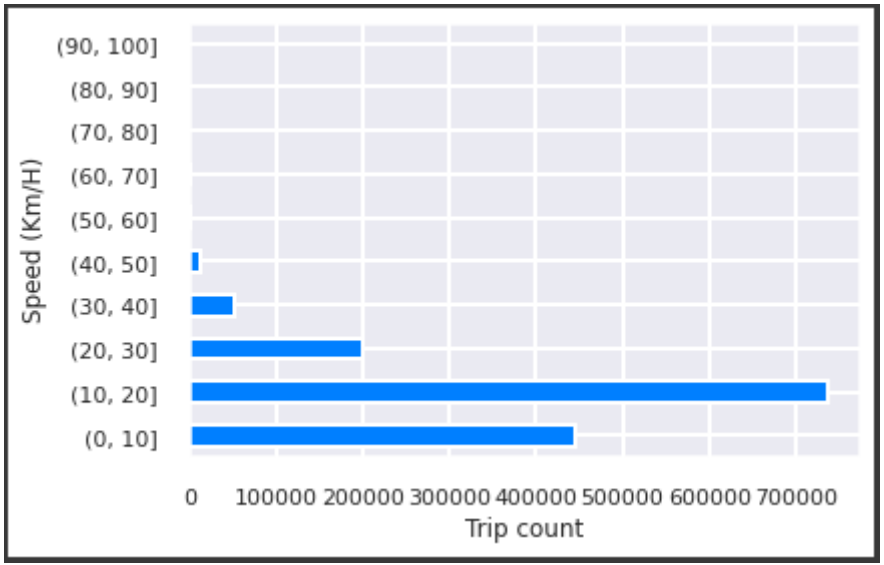
From the above plot we can say that most of the trips were taken by the 2nd vendor also we can say that there is not much difference between both of them.

Passenger Count



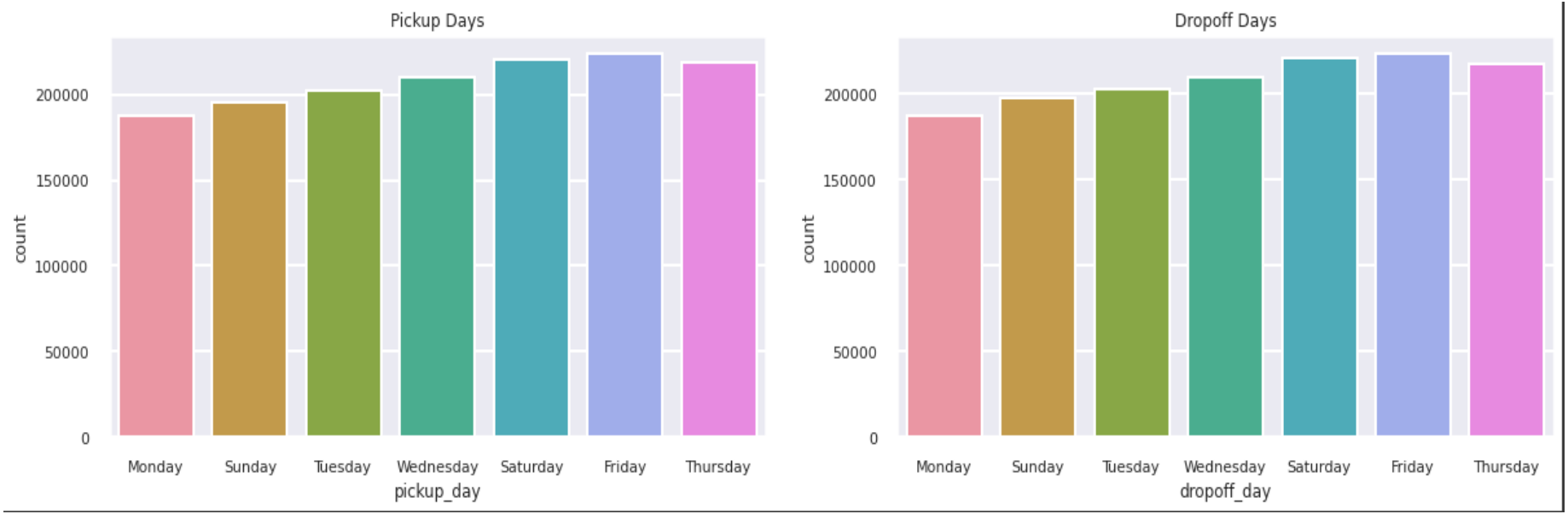
Here we can see that passenger count 1 has higher number of trips followed by passenger count 2. We can say that passenger count 1 and 2 only has the number of trips.

Speed limit From 0 - 104 km/hr



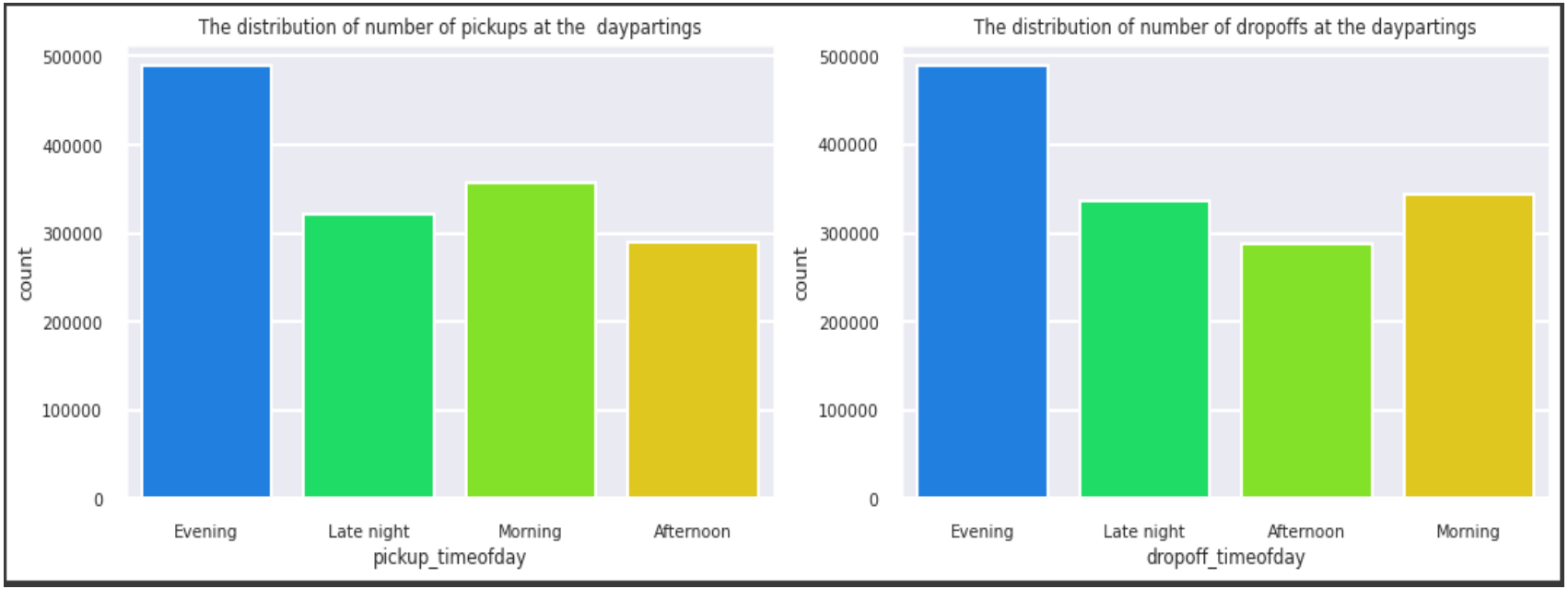
In the above graph most of the trips are with 10-20 km/hr range as mentioned earlier.

Pickup Day and Dropoff Day/Week



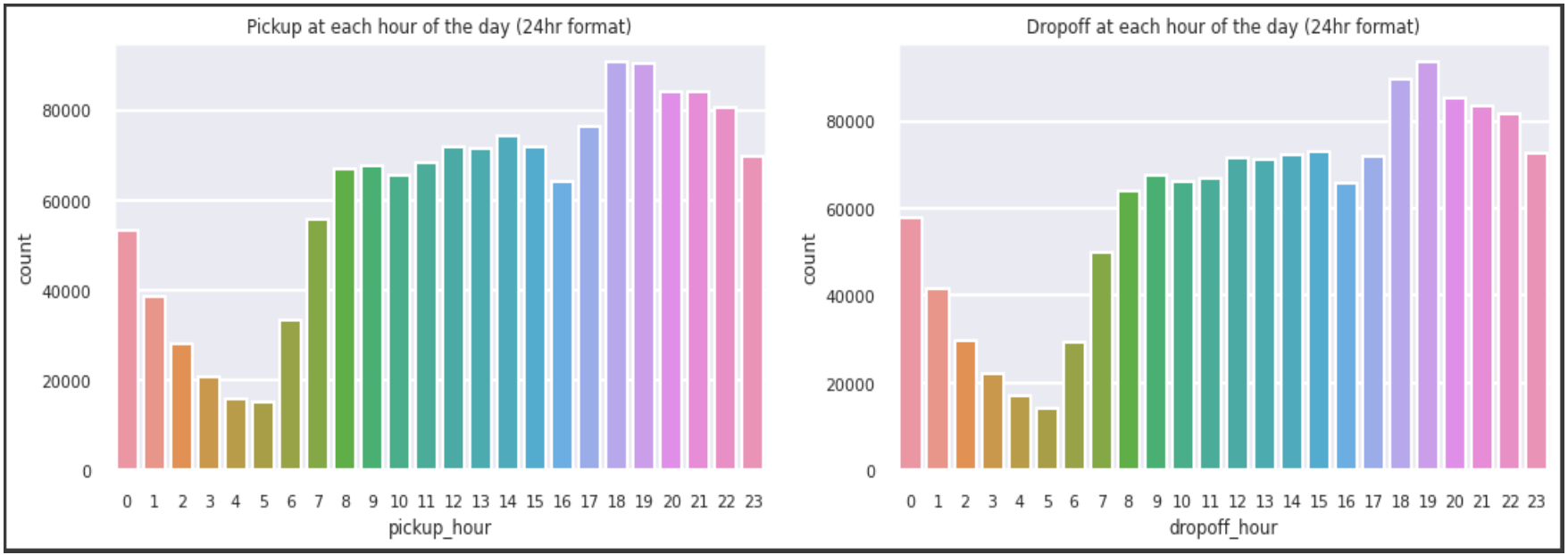
From above plot we can see that the pickup and dropoff were busiest at Friday followed by Saturday as it must be due to weekend that most of the peoples plan trips

Pickup and Dropoff During Day-Partings



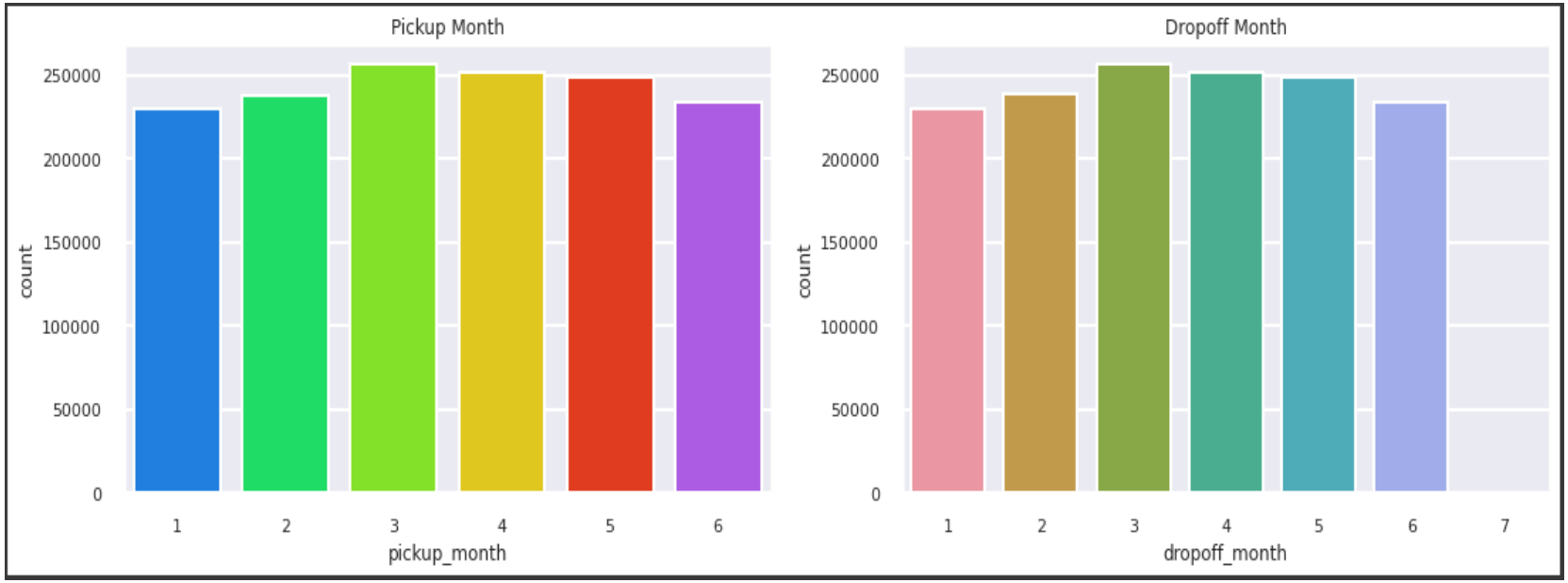
Here, we can see that most of the pickups and dropoffs took place at time of evening which is normal as people head to their home after they finished with their jobs.

Pickup and Dropoff At Each Hour Of The Day



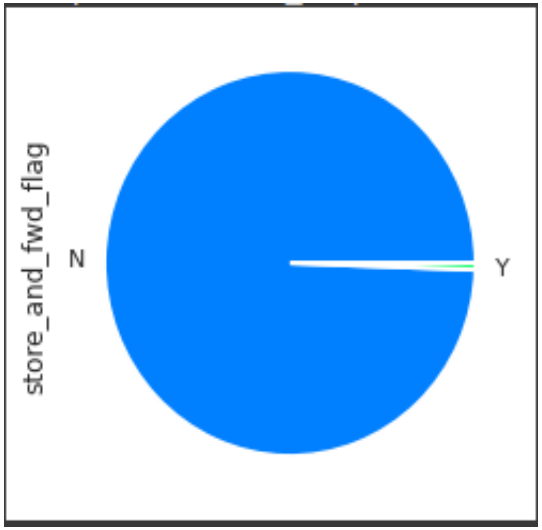
In pickup and dropoff at day partings we saw most of the trips took place at evening likely in pickup and dropoff at each hour of the day the maximum trips took place at 6 - 7 P.M.

Pickup and Dropoffs At Every Month



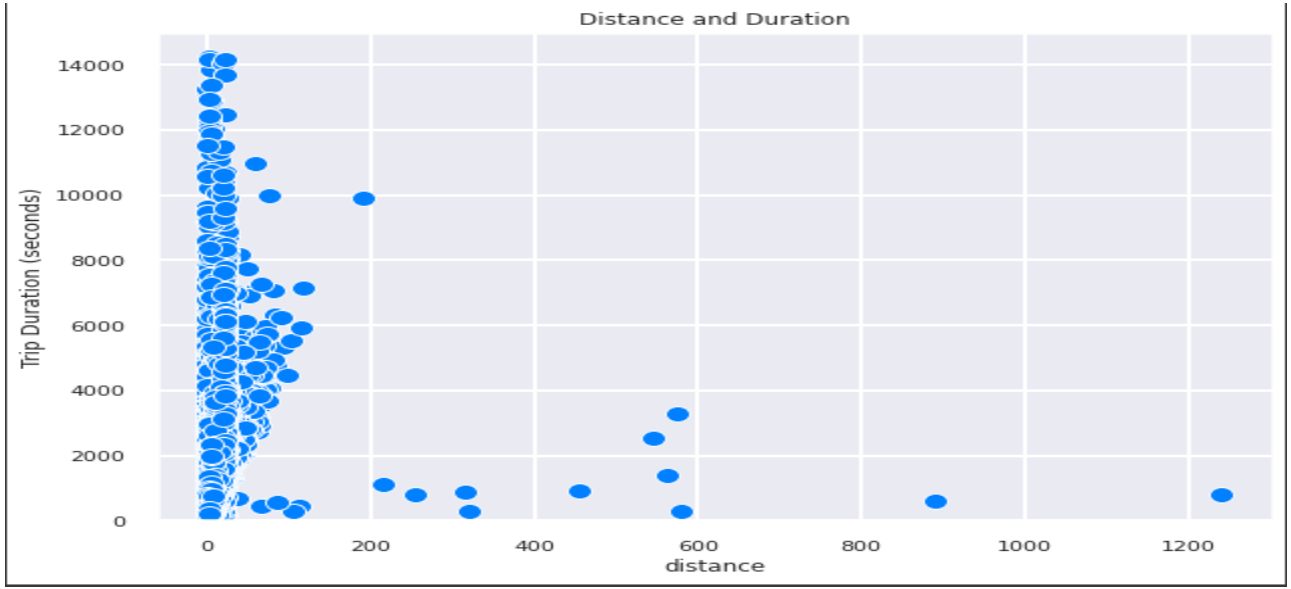
From above we can see that the month of march(3) and April(4) were busiest of booking cabs.

The Distribution Of Stored And Forward Flag



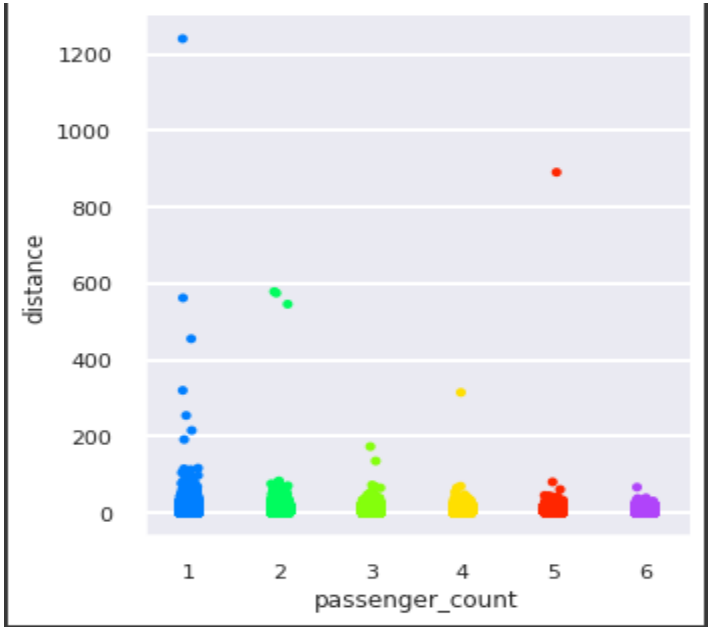
We can see that only about 1% of the trip details were stored in the vehicle memory first before sending it to the server. This might have occured because of the GPS or mobile device battery was down when the trip finished.

Distance And Trip Duration



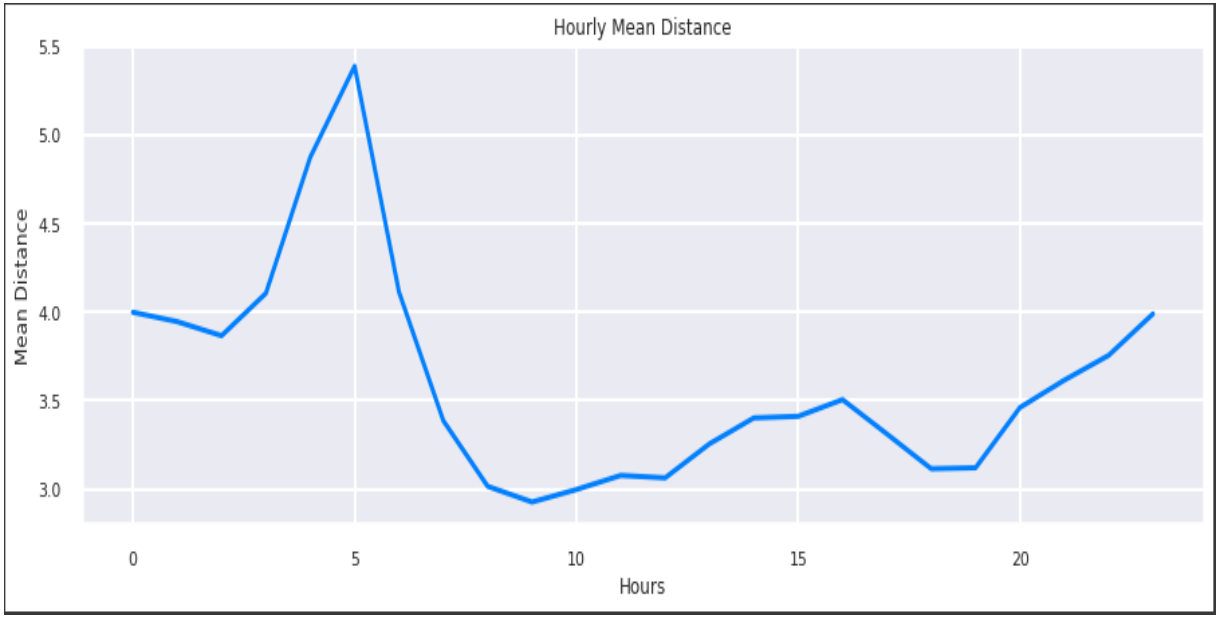
Here we can see that the distance from 0 - 100km has the dense trip duration.

Distance And Passenger Count



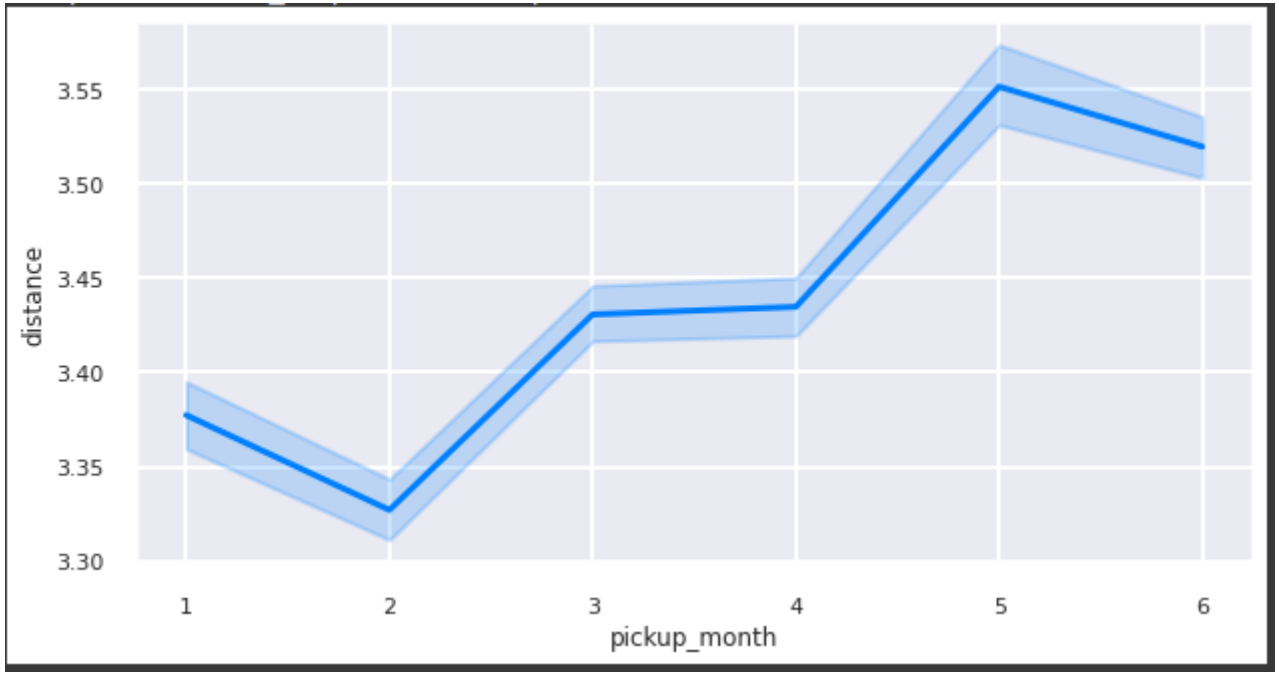
We see some of the longer distances are covered by either 1 or 2 or 4 passenger rides.

Hourly Mean Distance



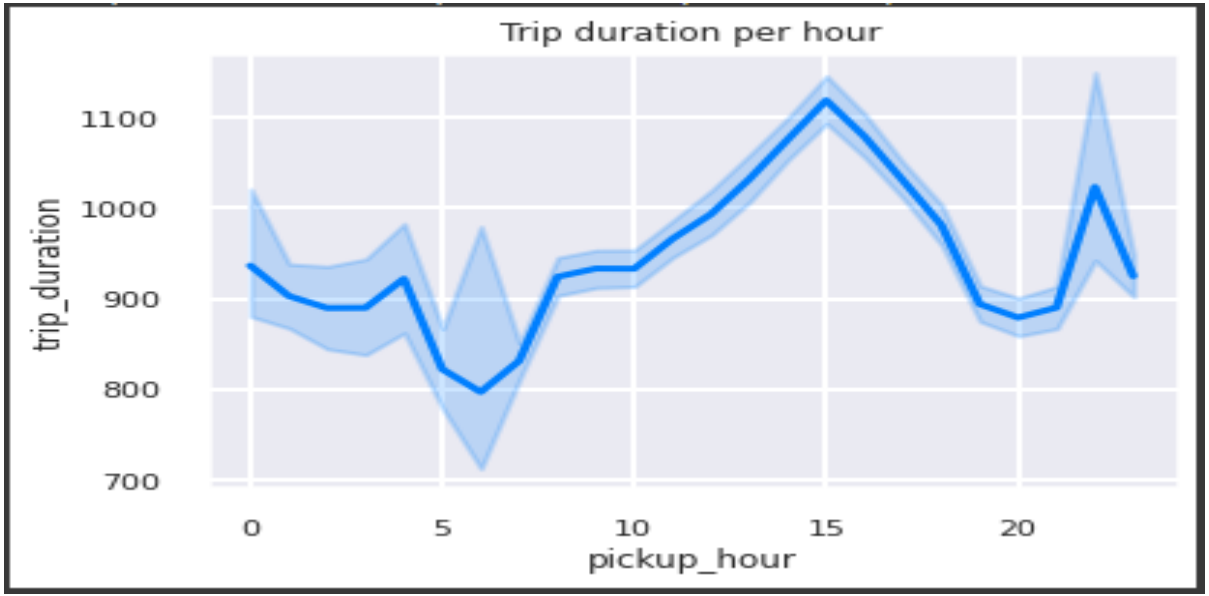
The longest distance travelled at late night to early morning.

Distance And Pickup Month



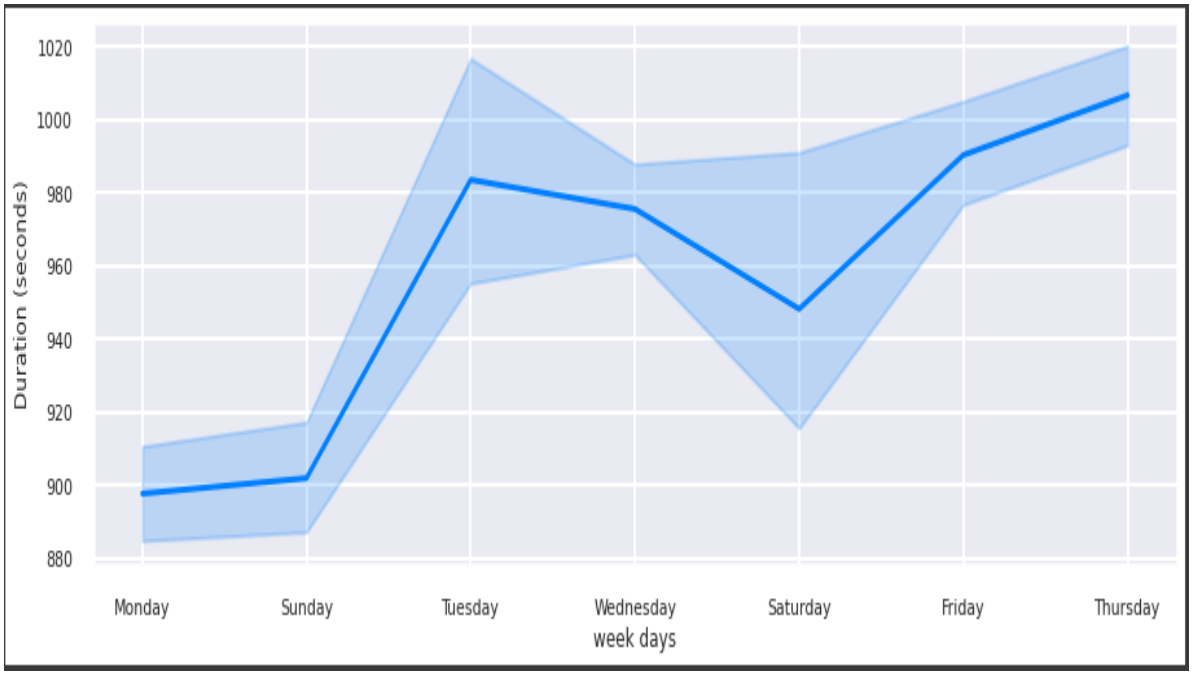
Here we can see that the maximum distance was covered in the month of may and least in february.

Trip Duration And Pickup Hour



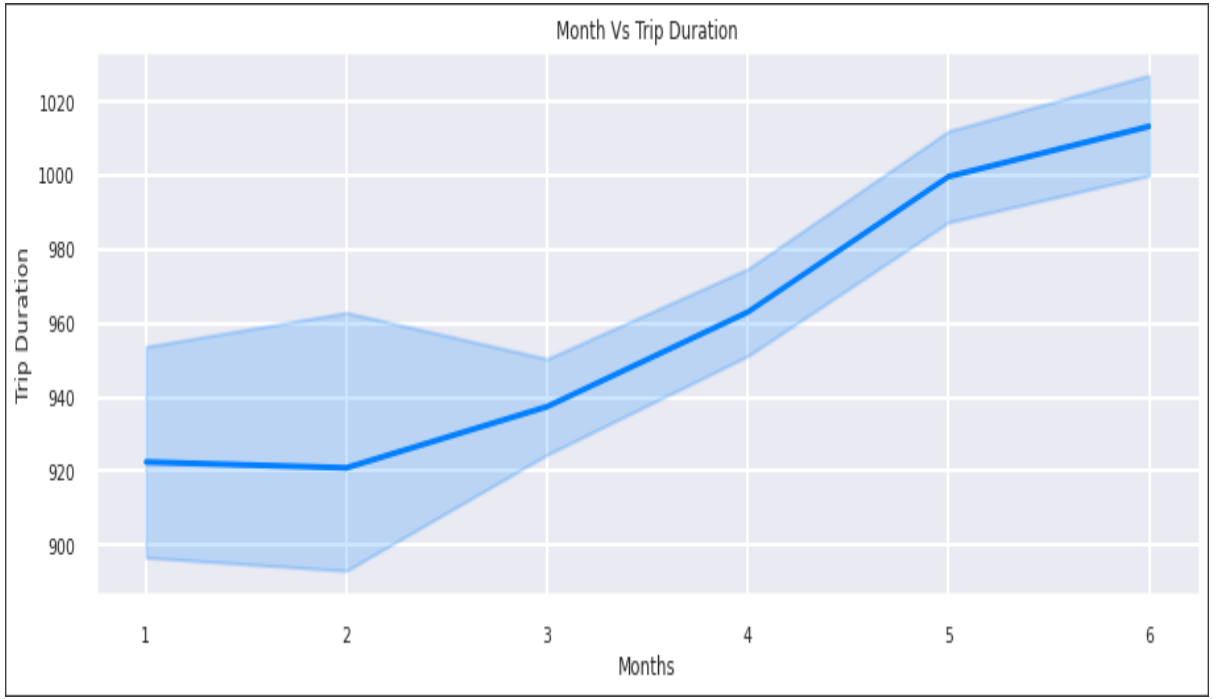
Here, we can see that the duration of the trips were busy during 3 P.M. may be due to heavy traffic. Trip duration were least around 5:30A.M – 6 A.M as light traffic.

Trip Duration Per Day



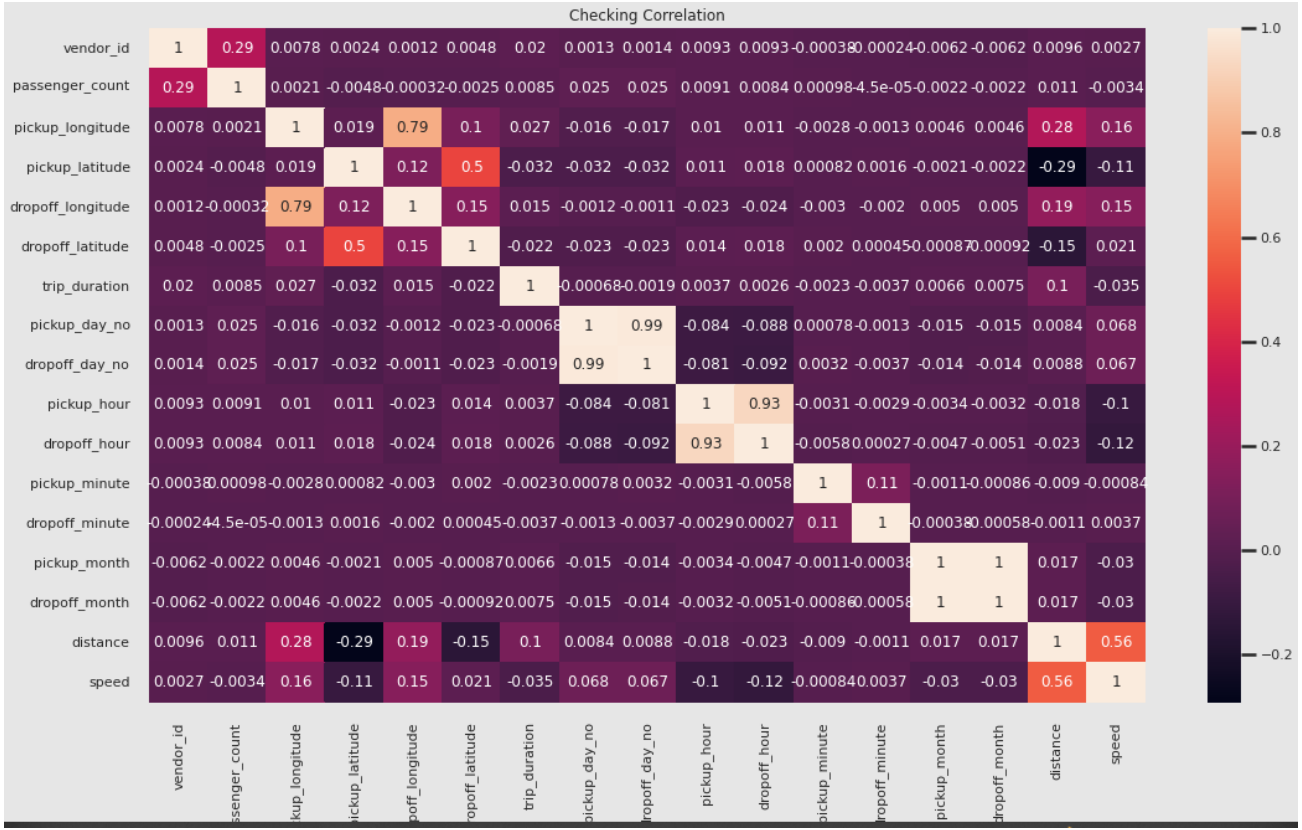
Here, we can see that the trip durations are maximum on working days i.e (Monday, Tuesday, Wednesday, Thursday and Friday). Trip duration least during weekoffs i.e. (Saturday and Sunday).

Trip Duration Per Month



Here, we can see that the trip duration start increasing from the month of march before that its quite constant

Heatmap For Correlation



Here we can see there are lot of correlated features that are required to be detected and to be dropped.

Selection Of The Features

Independent Variable

- Pickup day
- Dropoff day
- Pickup day number
- Dropoff day number
- Pickup hour
- Dropoff hour
- Distance
- Speed
- Pickup time of day
- Dropoff time of day
- Pickup latitude
- Pickup longitude
- Dropoff latitude
- Dropoff longitude

Dependent Variable

- Trip duration

Preparing Dataset

```
[88] # first we make a copy of  data  to a new variable
     new_nyct = nyct.copy()
```

```
[89] # seperating our data in feature and target variables
     x = new_nyct.drop(['trip_duration'], axis = 1)

     # taking our target variable into log form
     y = np.log(new_nyct['trip_duration']).values
```

Splitting our data into train and test variables

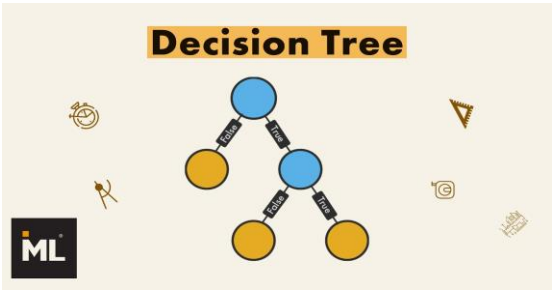
```
[ ]  # By using train test split from sklearn
     from sklearn.model_selection import train_test_split
```

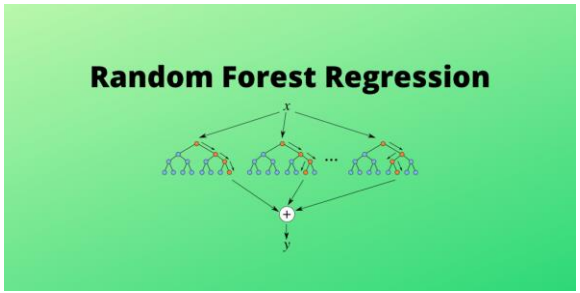
```
[ ]  # Keeping 30% data for testing and 70% for training the model

     x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 1)
```

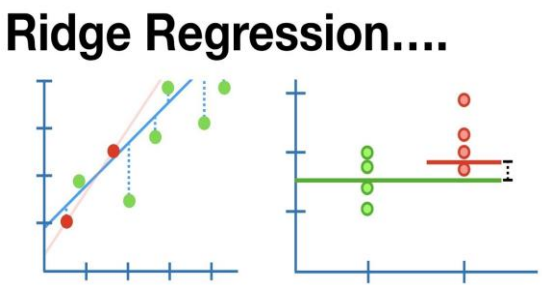
```
[ ]  x_train.shape, x_test.shape, y_train.shape, y_test.shape

     ((1020895, 19), (437527, 19), (1020895,), (437527,))
```

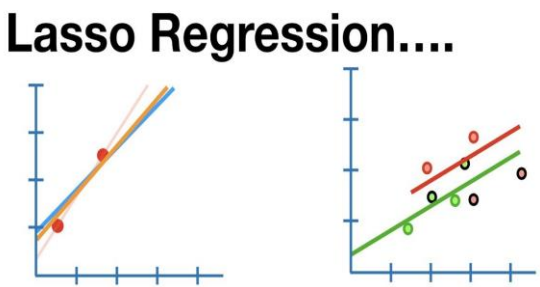
Here we have separated our features and target variable and we split them into train and test variables. Here we are preparing model without standardizing the feature to see how model performs without standardizing.

Machine Learning Models









Model Evaluation And Selection

	Training_Score	Testing_Score	R2_Score	ADJ_R2	MSE	RMSE
Linear Regression	0.591141	0.591825	0.591825	0.591807	0.259843	0.509748
Decision Tree Regression	0.976706	0.975439	0.975439	0.975438	0.015635	0.125042
Random Forest Regression	0.980628	0.979655	0.979655	0.979654	0.012952	0.113805
XGBOOST	0.996709	0.994279	0.994279	0.994279	0.003642	0.060348

Ridge Regressor

Without standardization of features

Lasso Regressor

```
Train MSE : 0.26066285565405756
Train R2 : 0.5911403145652487
Train Adjusted R2 :  0.5911403145652487
```

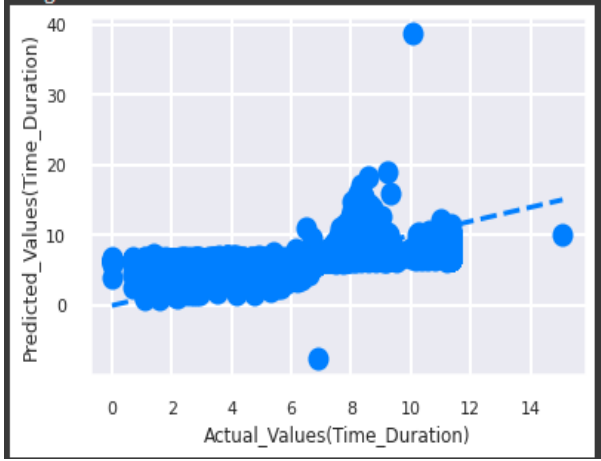
```
Train MSE : 0.2606622187242883
Train R2 : 0.5911413136140178
Train Adjusted R2 :  0.5911413136140178
```

```
Test MSE : 0.25984349359915115
Test R2 : 0.5918249518108839
Test Adjusted R2 :  0.5918072256352671
```

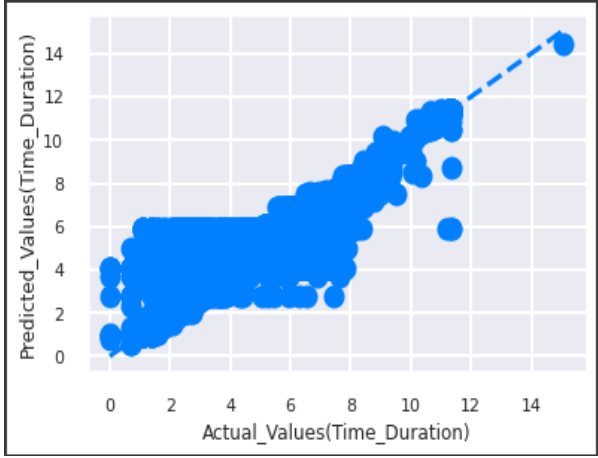
```
Test MSE : 0.25984336630529276
Test R2 : 0.5918251517703905
Test Adjusted R2 :  0.5918074256034574
```

Actual Vs Predicted Values Plots

```
Prediction by LinearRegression :
 [5.76233521 6.5469791  6.14837798 ... 6.56147306 6.66251138 6.50533633]
<Figure size 432x216 with 0 Axes>
```

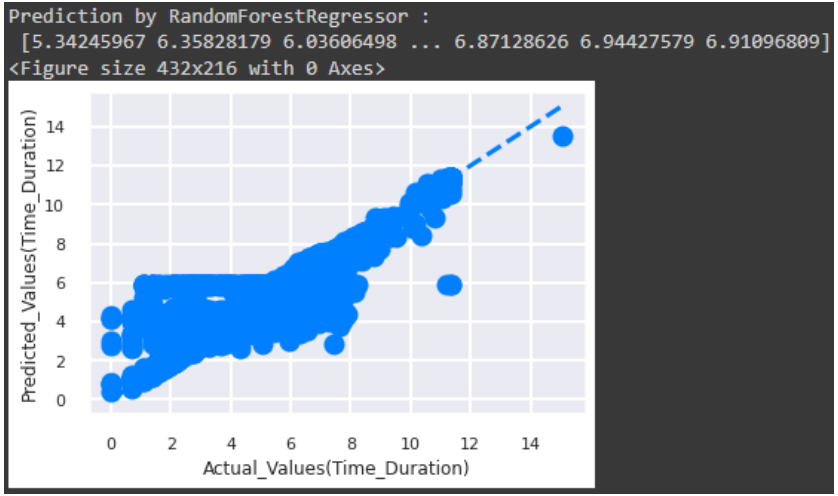


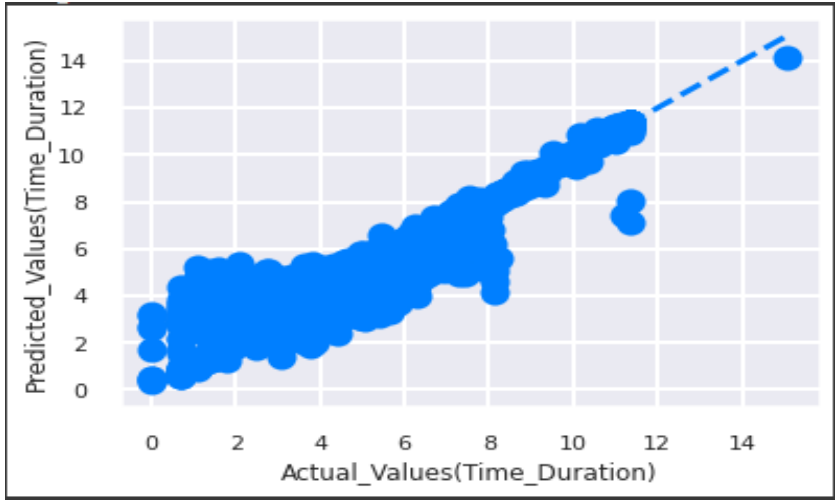

```
Prediction by DecisionTreeRegressor :
 [5.36369686 6.2559168  6.03155616 ... 6.94590581 7.01618777 6.95329929]
<Figure size 432x216 with 0 Axes>
```



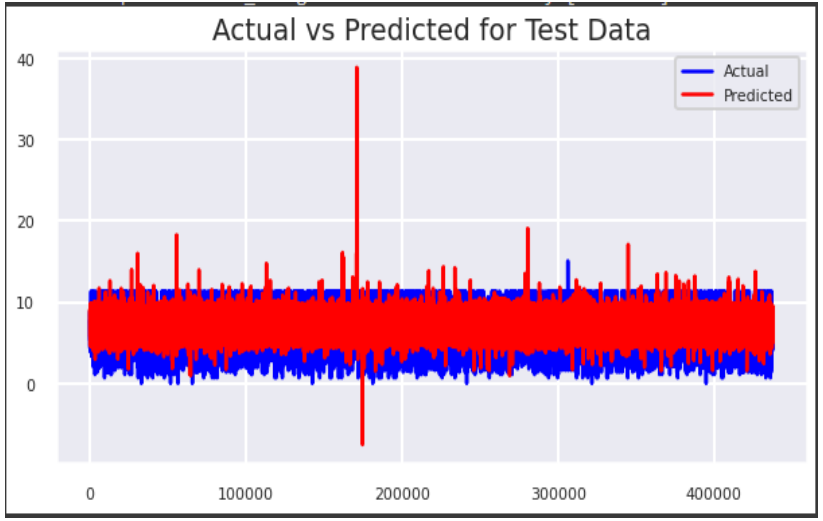
Actual Vs Predicted Values Plots

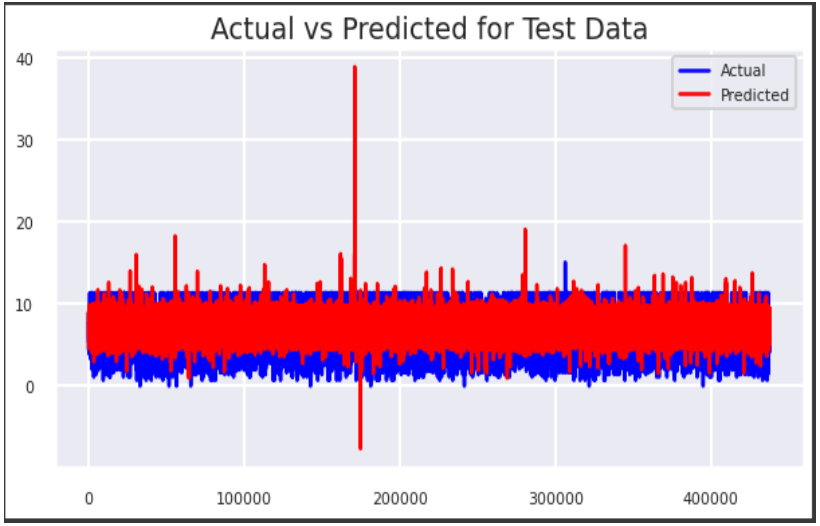
```
Prediction by RandomForestRegressor :
 [5.34245967 6.35828179 6.03606498 ... 6.87128626 6.94427579 6.91096809]
<Figure size 432x216 with 0 Axes>
```





Actual Vs Predicted Values Plots





Ridge Regression

Lasso Regression

Model Evaluation And Selection

	Training_Score	Testing_Score	R2_Score	ADJ_R2	MSE	RMSE
Linear Regression	0.589935	0.594624	0.594624	0.594607	0.259212	0.509129
Decision Tree Regression	0.976689	0.976821	0.976821	0.976820	0.014822	0.121744
Random Forest Regression	0.980014	0.980210	0.980210	0.980209	0.012655	0.112493
XGBOOST	0.996232	0.993903	0.993903	0.993902	0.003899	0.062441

Ridge Regressor

With standardization of the features

Lasso Regressor

```
Train MSE : 0.2609324896748073
Train R2 : 0.5899346607648064
Train Adjusted R2 :  0.5899346607648064
```

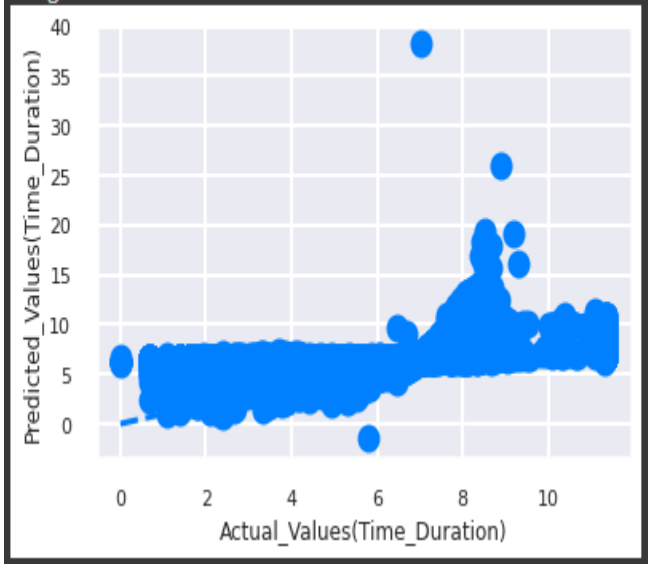
```
Test MSE : 0.2592138798185277
Test R2 : 0.5946218072533849
Test Adjusted R2 :  0.594604202539261
```

```
Train MSE : 0.2628142484146756
Train R2 : 0.5869774052808896
Train Adjusted R2 :  0.5869774052808896
```

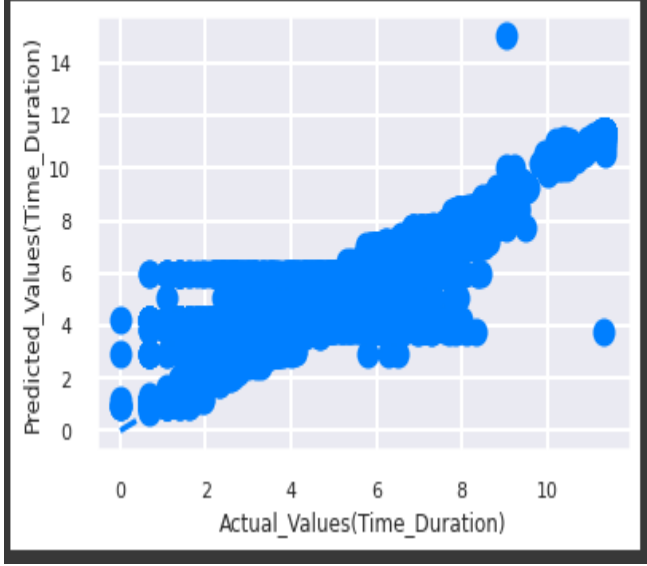
```
Test MSE : 0.261582862704362
Test R2 : 0.590917013352767
Test Adjusted R2 :  0.5908992477473108
```

Actual Vs Predicted Values Plots

```
Prediction by LinearRegression :
 [6.43536375 6.04990003 6.57308844 ... 6.28725913 9.69408054 6.52609705]
<Figure size 432x216 with 0 Axes>
```

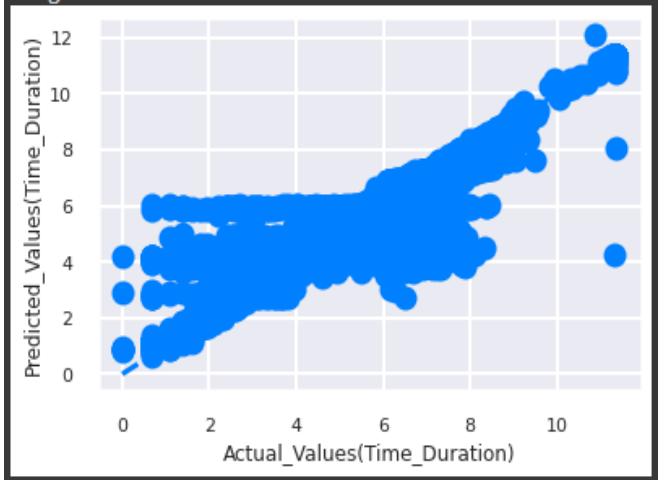



```
Prediction by DecisionTreeRegressor :
 [6.77549921 6.11102935 6.96454961 ... 5.89548384 8.28415983 6.58098929]
<Figure size 432x216 with 0 Axes>
```



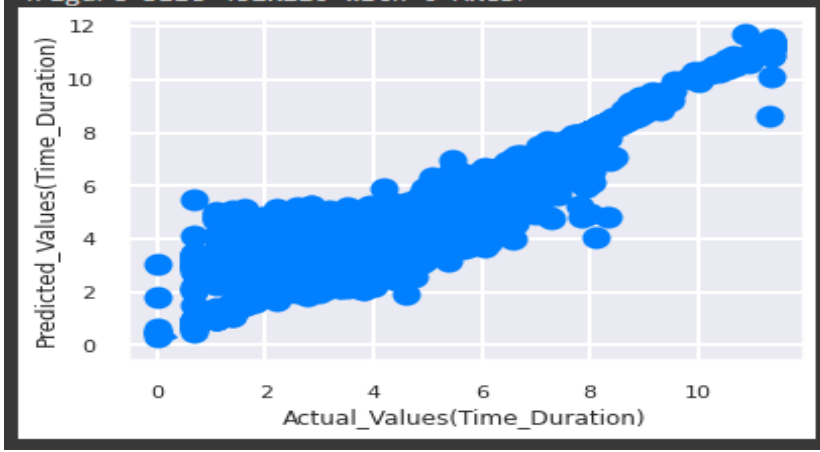
Actual Vs Predicted Values Plots

```
Prediction by RandomForestRegressor :
 [6.71985749 6.18704467 6.92324457 ... 5.92114585 8.22418346 6.6066948
<Figure size 432x216 with 0 Axes>
```

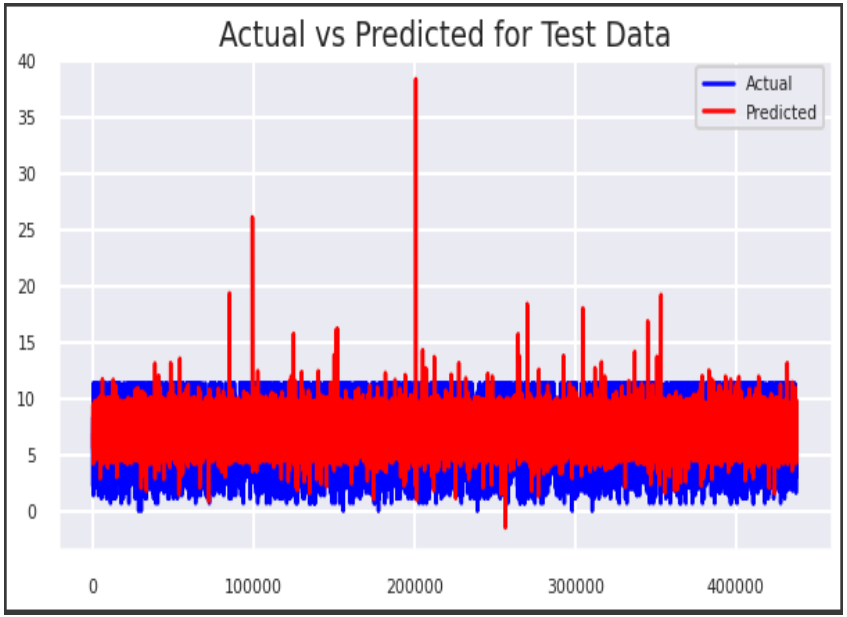


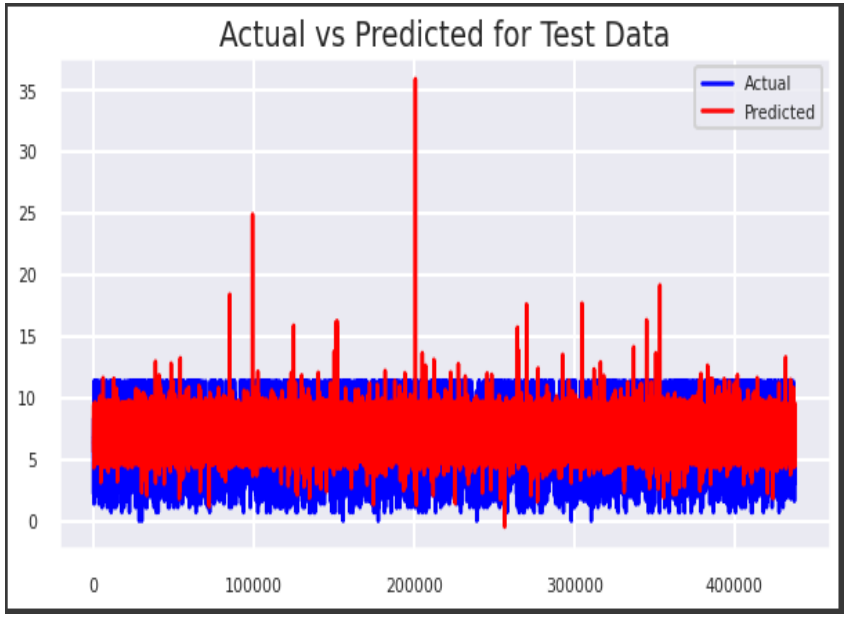

```
#XGBoost regression plot
train_vs_pred_plot(y_test,y_pred)
```

```
<Figure size 432x216 with 0 Axes>
```



Actual Vs Predicted Values Plots





Conclusion

•As until now we have tried multiple models with and without standardizing the features also compared them with all aspects and we can see XGBoost is performing good compared to linear, ridge, lasso, decision and random forest.

•Also the decision tree and random forest are performing good with or without standardizing the features.

•After standardizing the features there is no much difference in the accuracy or other metrics of models.

•Lasso and Rigde are not performing well after standardizing the features either. Their accuracy has been decreased by 1%.

•At last we can conclude that the XGBoost is the best fit model for good prediction rate and lesser error.


Thank You!