Traffic Jam: Predicting People's Movement into Nairobi

PROBLEM DESCRIPTION,

- This challenge asks you to build a model that predicts the number of seats that Mobiticket can expect to sell for each ride, i.e. for a specific route on a specific date and time. There are 14 routes in this dataset.
- All of the routes end in Nairobi and originate in towns to the North-West of Nairobi towards Lake Victoria.
- The routes from these 14 origins to the first stop in the outskirts of Nairobi takes approximately 8 to 9 hours from time of departure.

PROBLEM STATEMENT

- From the first stop in the outskirts of Nairobi into the main bus terminal, where most passengers get off, in Central Business District, takes another 2 to 3 hours depending on traffic.
- The three stops that all these routes make in Nairobi (in order) are:
    1. Kawangware: the first stop in the outskirts of Nairobi
    2. Westlands
    3. Afya Centre: the main bus terminal where most passengers disembark
- Passengers of these bus (or shuttle) rides are affected by Nairobi traffic not only during their ride into the city, but from there they must continue their journey to their final destination in Nairobi wherever that may be.

PROBLEM STATEMENT

- Traffic can act as a deterrent for those who have the option to avoid buses that arrive in Nairobi during peak traffic hours. On the other hand, traffic may be an indication for people's movement patterns, reflecting business hours, cultural events, political events, and holidays.

DATA DESCRIPTION

- **Nairobi Transport Data.csv (zipped)** is the dataset of tickets purchased from Mobiticket for the 14 routes from “up country” into Nairobi between 17 October 2017 and 20 April 2018.
- This dataset includes the variables: ride_id, seat_number, payment_method, payment_receipt, travel_date, travel_time, travel_from, travel_to, car_type, max_capacity.
- Uber Movement provided historic hourly travel time between any two points in Nairobi.

DATA DESCRIPTION

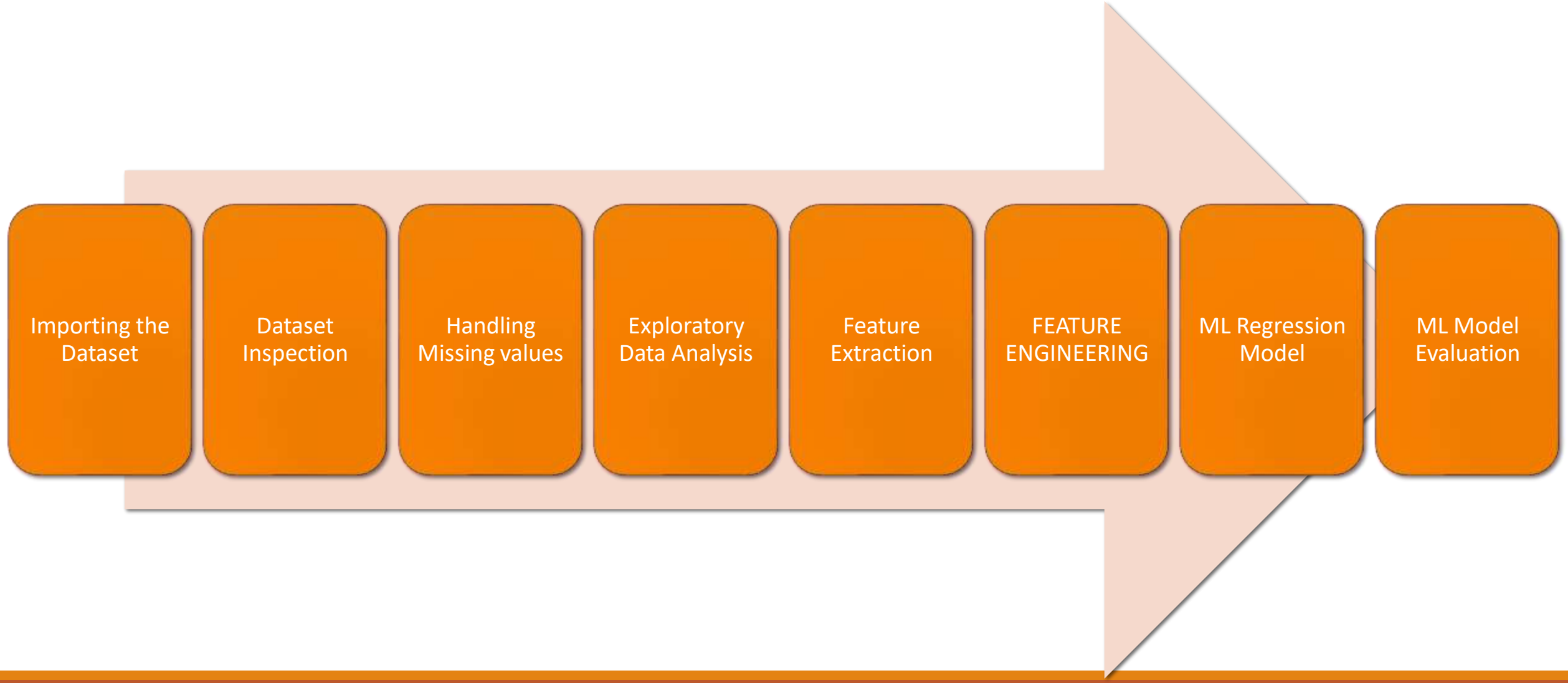
Variables description:

- **ride_id:** unique ID of a vehicle on a specific route on a specific day and time.
- **seat_number:** seat assigned to ticket
- **payment_method:** method used by customer to purchase ticket from Mobiticket (cash or Mpesa)
- **payment_receipt:** unique id number for ticket purchased from Mobiticket
- **travel_date:** date of ride departure. (MM/DD/YYYY)
- **travel_time:** scheduled departure time of ride. Rides generally depart on time. (hh:mm)
- **travel_from:** town from which ride originated
- **travel_to:** destination of ride. All rides are to Nairobi.
- **car_type:** vehicle type (shuttle or bus)
- **max_capacity**: number of seats on the vehicle

CONTENTS

- Visualization Analysis
- Preprocessing the Data
- ML-Regression Models
- ML-Evaluation Metrics
- conclusion

STEPS INVOLVED



DATA SUMMARY

Name of the Dataset	Train_revised
Number of variables	10
Number of observations	51645
Duplicate rows	0
Total size in memory	5 MB
Missing Data (Columns)	0

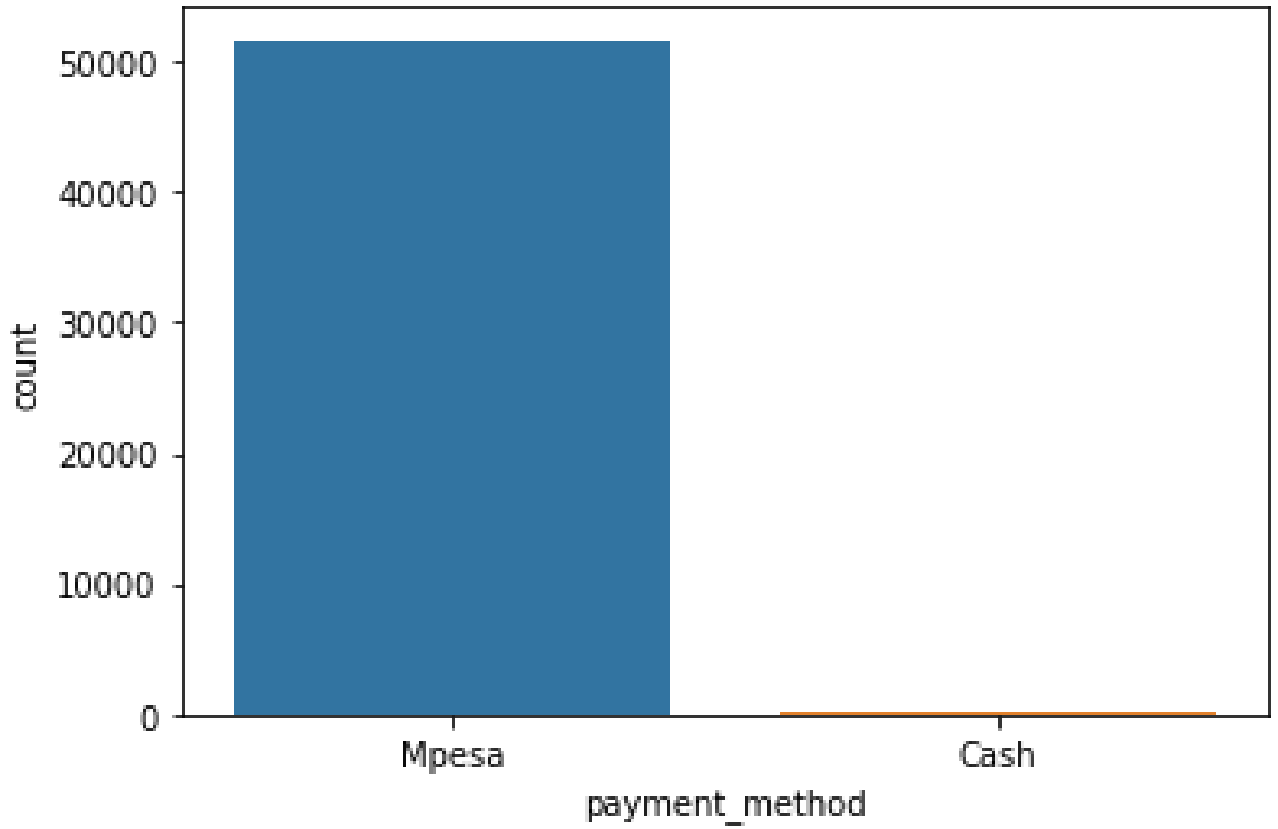
VARIABLE DATA TYPE

Date Type	Column
Numeric - int64	0 ride_id 9 max_capacity
String - object	1 seat_number 2 payment_method 3 payment_receipt 4 travel_date 5 travel_time 6 travel_from 7 travel_to 8 car_type

DATASET INSPECTION

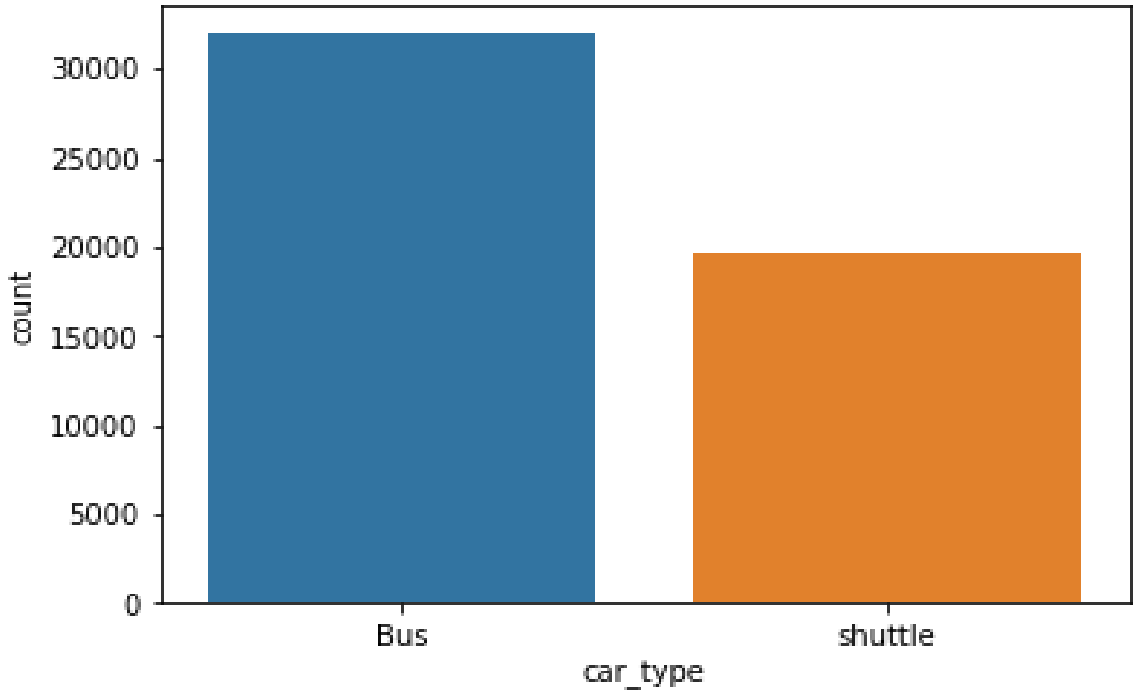
- There are total 61 unique seats in this dataset.
- Travelers have used 2 types of payment method and most of the people have used Mpesa to pay for their ticket.
- The record of 149 days out of 2 year is present in this dataset.
- There are 2 different types of car and most of them are bus.

PAYMENT METHODS



There are two type of payment methods people have used to buy the tickets.

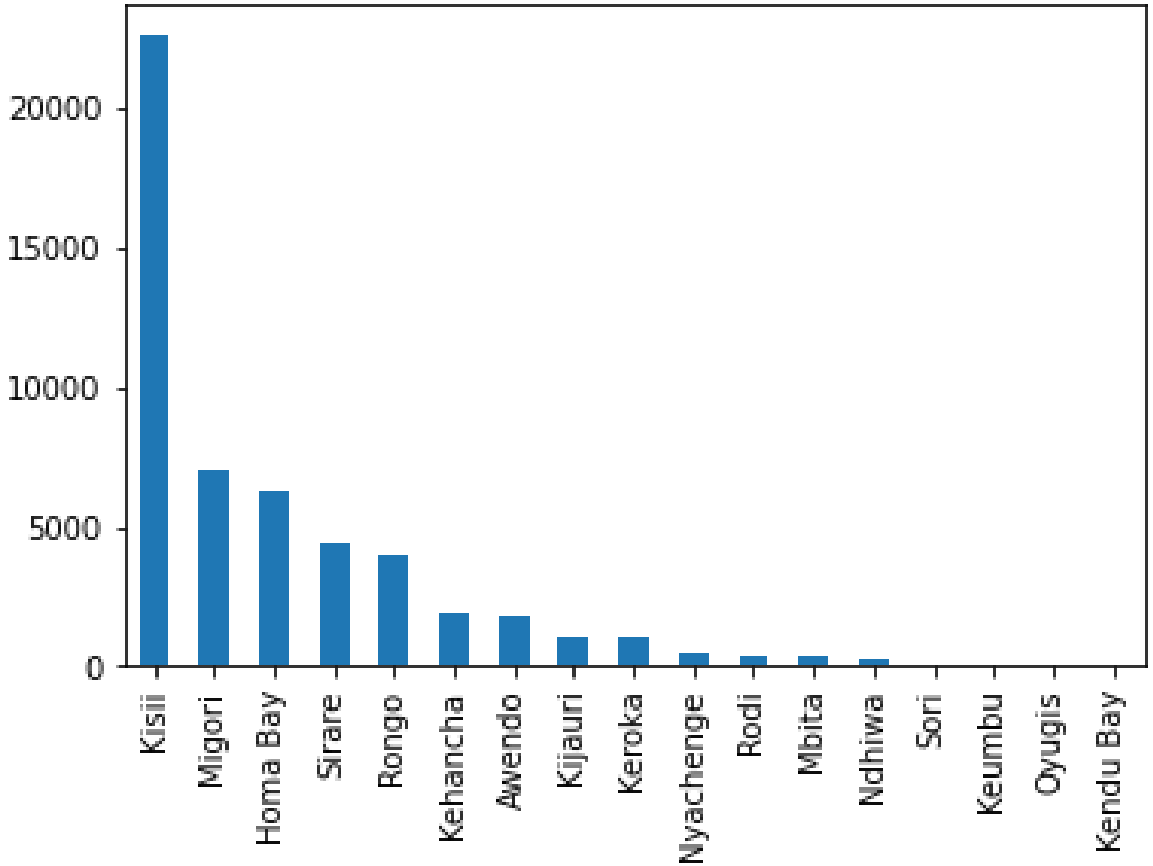
CAR TYPE



There are two type of cars

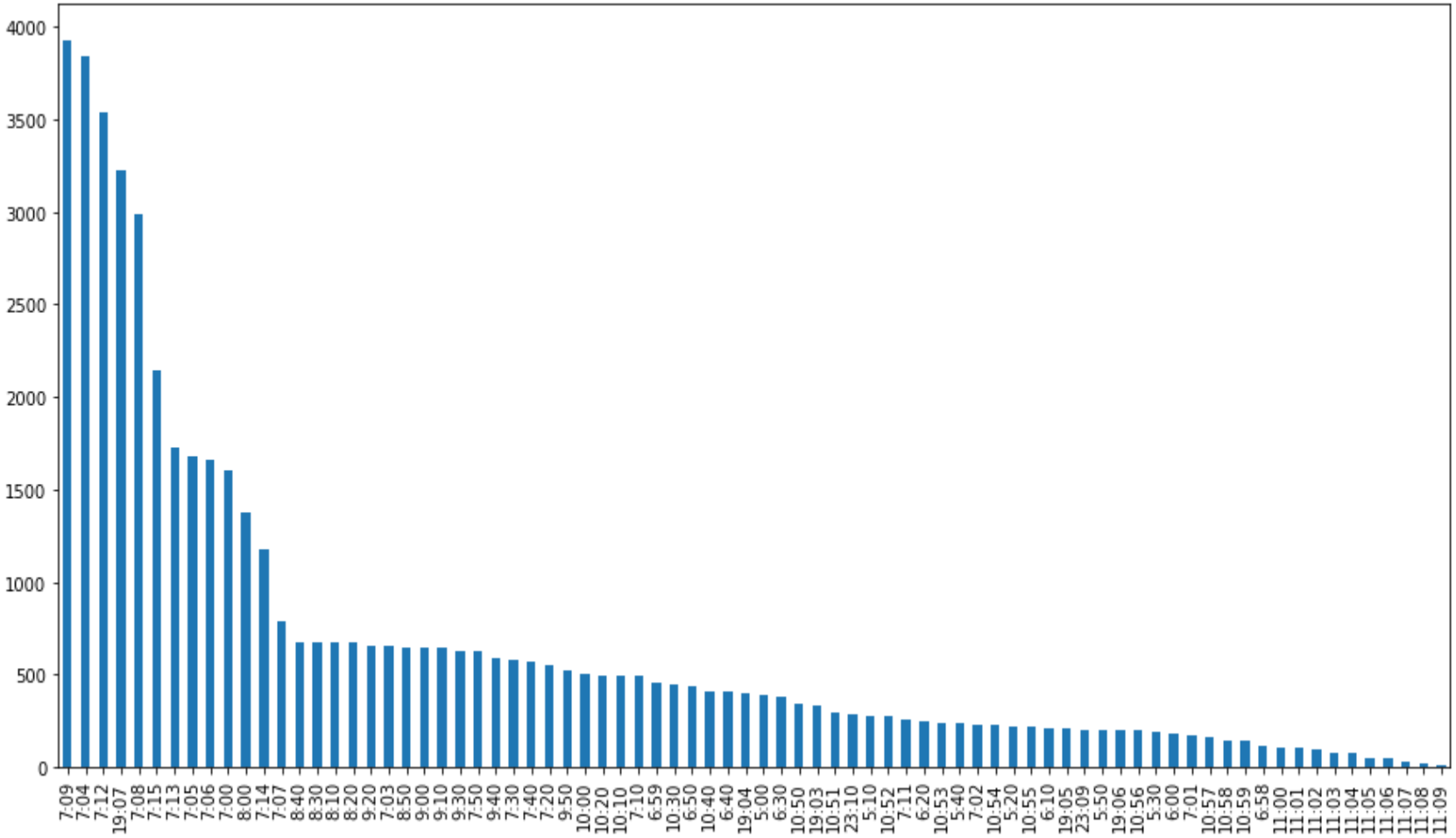
Bus and shuttle.

TRAVEL DETAILS – ORIGIN PLACE



From the above bar graph, we can say most of the travelers traveling from Kisii to Nairobi.

TRAVEL DETAILS -TIME



From the above bar graph, we can say most of the travelers travel at 7:09 from the origin.

RIDE ID

```
[ ]   1   # We can use the unique() method
      2   len(dataset['ride_id'].unique())

      6249
```

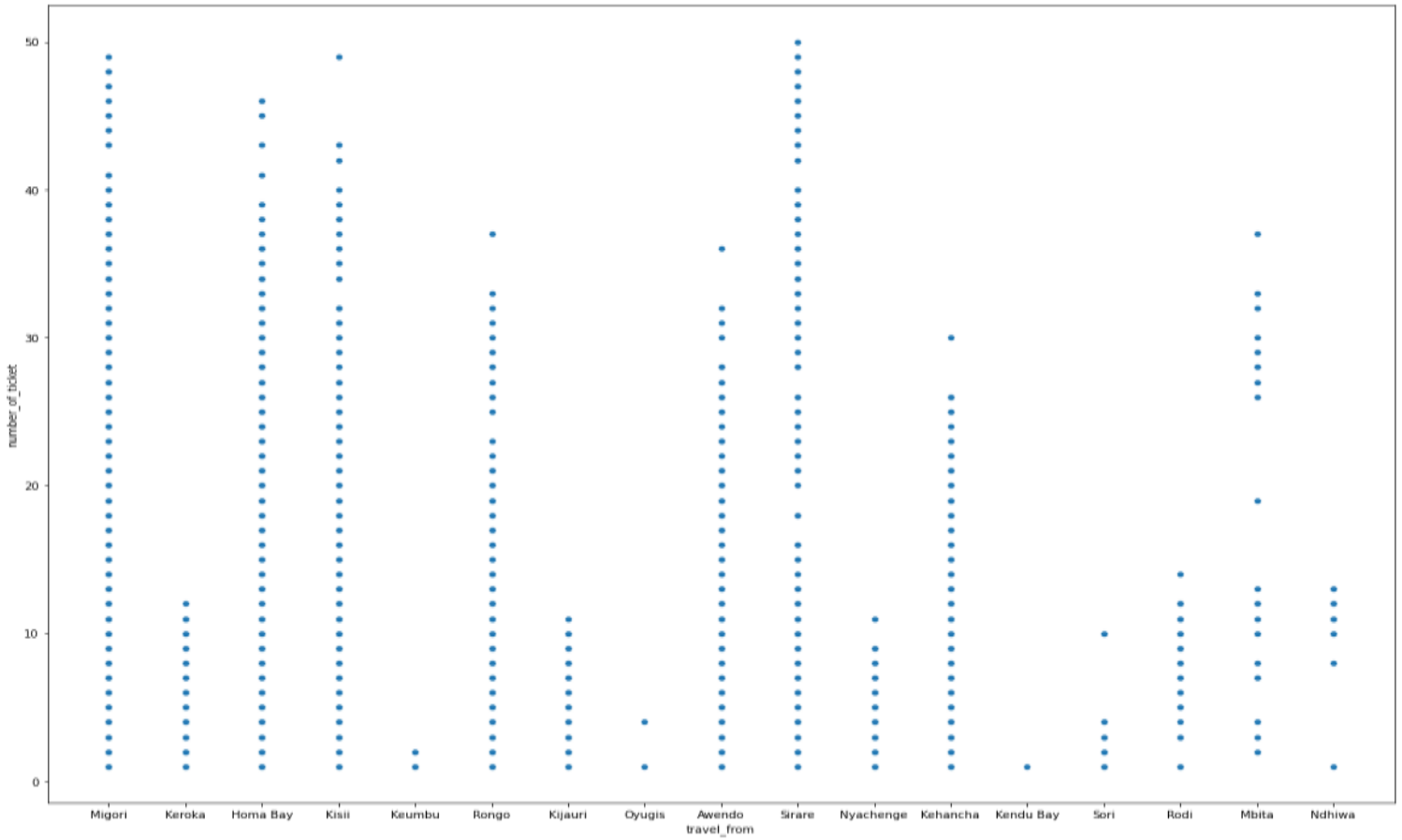
We see there are 6249 unique ride_id.

TARGET VARIABLE

- Since we are not given the target variable so we need to find target variable first.

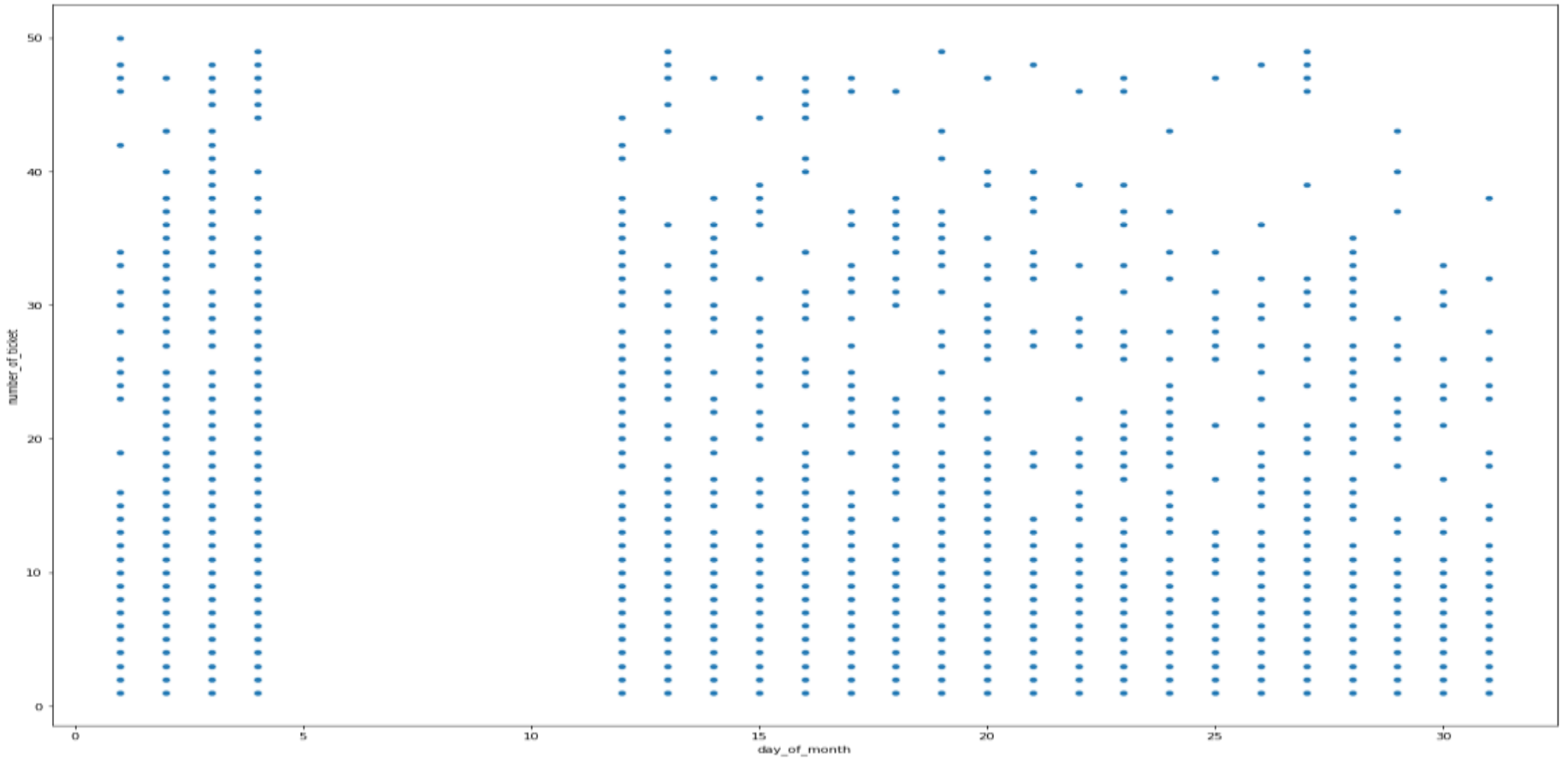
- There might be many ways of finding the target variable but here I am using one way that is I will find the count of each ride_id and that will be the number_of_ticket as our target variable.

NUMBER OF TICKETS BOOKED FROM ORIGIN CITY



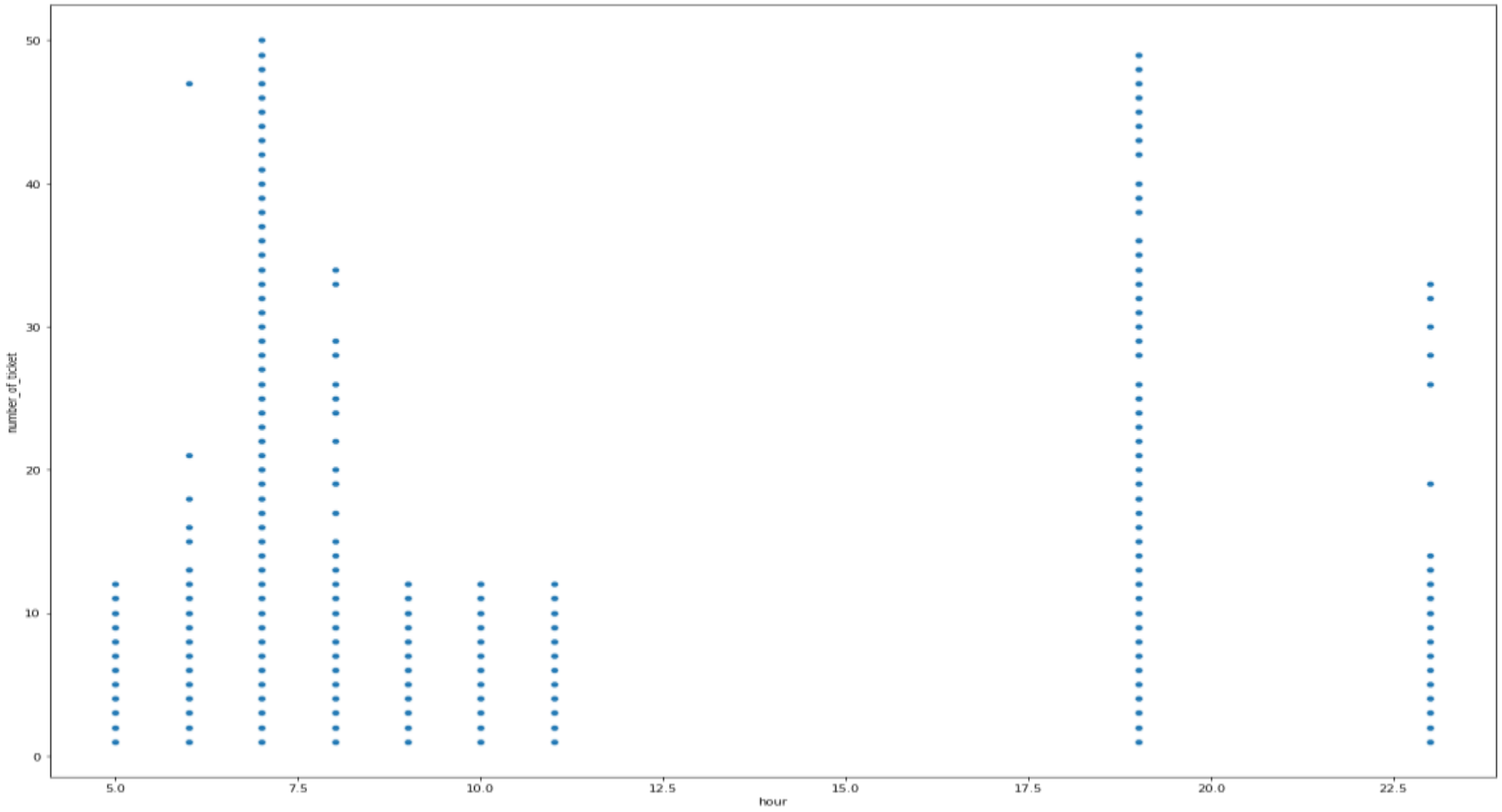
From the graph, peoples from Migori, Sirare has booked most number of tickets.

TRAVEL DETAILS BASED ON THE DAY OF THE MONTH



We can see that there is the gap between 5 to 11 in the day of the month. We can assume that there is official holiday of public transport between these days. we can also say that the number of tickets in all the days of month are same

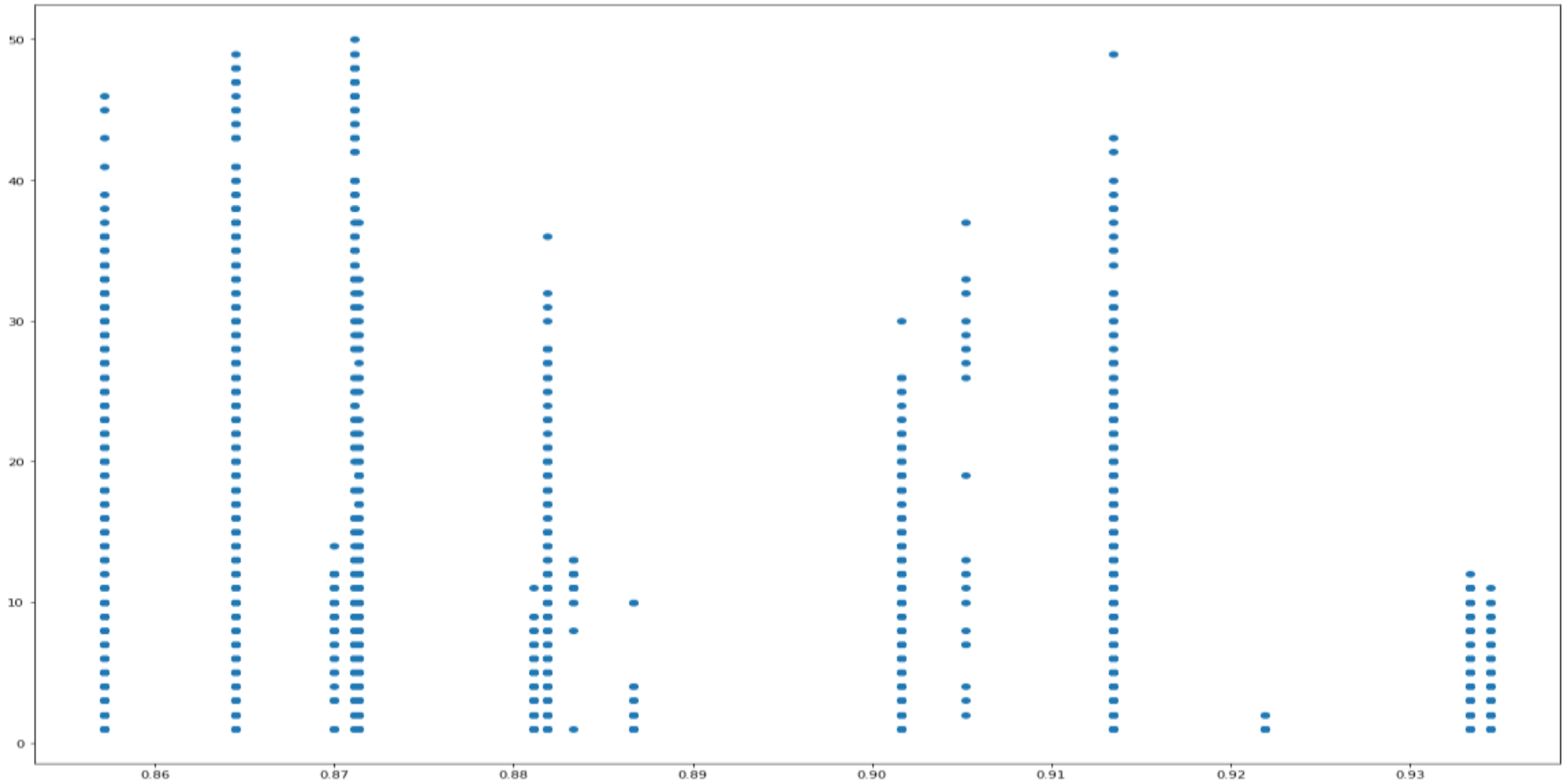
TRAVEL DETAILS BASED ON THE HOUR OF THE DAY



We can see that most of the tickets were sold at 7 AM and 8 PM. And that seems true because in the morning most of the people go to the work and office.

From the above we can say that there is not ride between 12pm to 5.30Pm.

SPEED TO REACH THE DESTINATION



Scatter plot shows the speed of the vehicle and Number of tickets booked.

Higher the speed , lesser number of tickets booked.

ENCODING CATEGORICAL FEATURES

Encoding Categorical features

```
[ ]  1  from sklearn import preprocessing #Import LabelEncoder
     2  data = pd.get_dummies(data, columns=['travel_from','day_of_month','month'])
     3  label_enc = {'Bus':1,'shuttle':0}
     4  data.replace(label_enc,inplace=True)
```

Encoding for categorical features are done by using get_dummies method.

The shape of the final data set is (6042,52)

The features considered to create ML- Regression are

{'car_type', 'day_of_week', 'day_of_year', 'hour', 'minute', 'is_weekend', 'year', 'quarter', 'hourly_travelers', 'daily_travelers', 'Time_gap_btw_0_1_next_bus', 'Time_gap_btw_0_1_previous_bus', 'Time_gap_btw_0_2_next_bus', 'Time_gap_btw_0_2_previous_bus', 'Time_gap_btw_0_3_next_bus', 'Time_gap_btw_0_3_previous_bus', 'Time_gap_btw_next_previous_bus', 'travel_from_distance', 'travel_from_time', 'Speed', 'hod_arrived_date', 'minute_arrived_date', 'is_rush_hour', 'travel_from_Awendo', 'travel_from_Homa Bay', 'travel_from_Kehancha', 'travel_from_Keroka', 'travel_from_Keumbu', 'travel_from_Kijauri', 'travel_from_Kisii', 'travel_from_Mbita', 'travel_from_Migori', 'travel_from_Ndhiwa', 'travel_from_Nyachenge', 'travel_from_Rodi', 'travel_from_Rongo', 'travel_from_Sirare', 'travel_from_Sori', 'day_of_month_1', 'day_of_month_2', 'day_of_month_3', 'month_1', 'month_2', 'month_3'}

ML MODEL

- For Train –Test data split up , we have considered 0.33 as threshold, So that train data will be around 4000 rows to train the model.

- To predict the number of tickets will be booked , following Regression models are created
  - Linear Regression
  - Random Forest Regressor
  - Extreme Gradient Boosting
  - Lasso Regression
  - Gradient Boosting Regressor

ML - REGRESSION MODEL

S.No	Regression Model	MSE	RMSE	RMSPE	MAE	MAPE
1	**Linear Regression**	48.5	6.96	346.44	4.75	165.80
2	**Lasso Regression**	56.12	7.49	341.44	5.437	185.59
3	**Gradient Boosting Regressor**	29.38	5.42	240	3.53	113.30
4	**XGboost**	24.9	4.99	206.98	3.16	96.5

- In Random Forest, Model seems to be over fitted because of

```
Train score: 0.951206584831083 Test score: 0.6480112690449429
```

HYPERPARAMETER TUNING

GRID SEARCH CV ON RANDOM FOREST

The best parameters for random forest are

```
1   rf_grid.best_estimator_.get_params()
```

```
{'bootstrap': True,
 'ccp_alpha': 0.01,
 'criterion': 'squared_error',
 'max_depth': 6,
 'max_features': 'auto',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 40,
 'min_samples_split': 100,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_jobs': None,
 'oob_score': False,
 'random_state': None,
 'verbose': 0,
 'warm_start': False}
```

The Evaluation metrics for random forest are

```
Train Score:  0.6145552472612785
Test Score:  0.6088897641458114
Mean Squared Error (MSE):  29.352445303861824
Mean Absolute Error (MAE):  3.467483290682997 8
Root Mean Squared Error (RMSE):  5.417789706500412
RMSPE is 236.36070981724467
MAPE is 111.06793360880727
R2 Score:  0.6088897641458114
R2_train Score:  0.6145552472612785
Adjusted R2 Score:  0.6003578601410597
```

HYPERPARAMETER TUNING

GRID SEARCH CV ON XGBOOST ALGORITHM

The best parameters for XGBOOST are

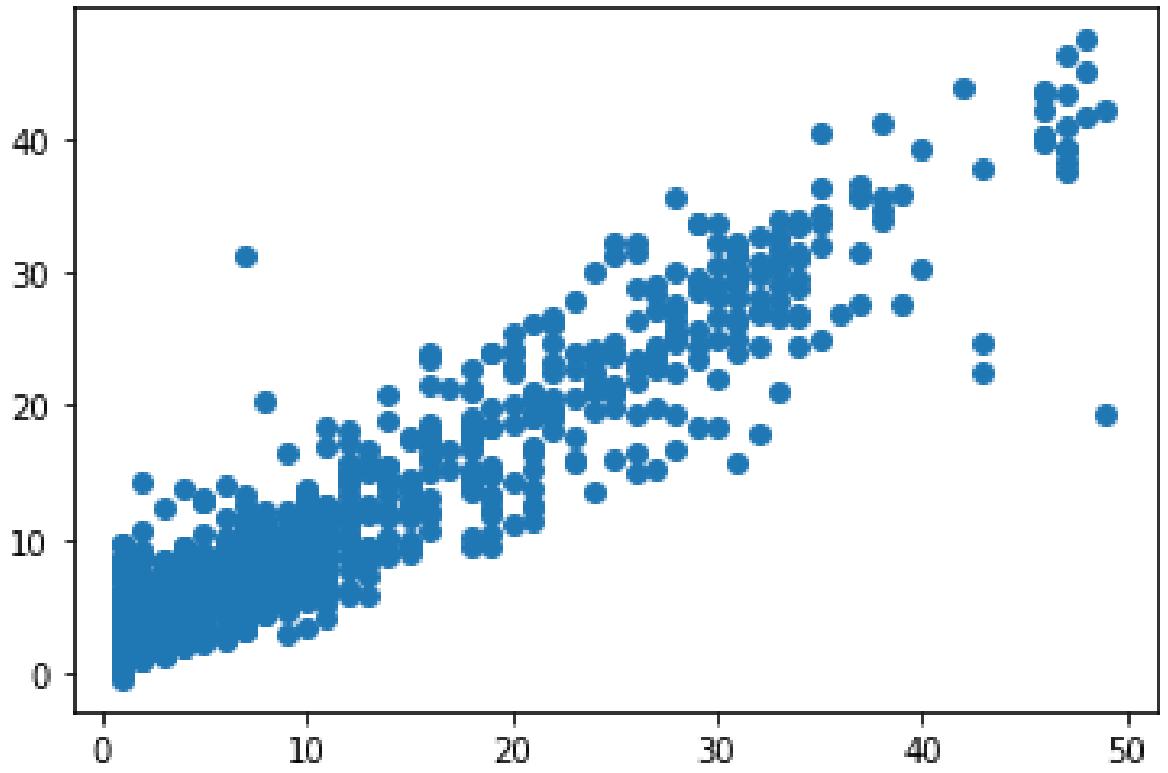
```
1   reg_gs.best_estimator_.get_params()
```

```
{'base_score': 0.5,
 'booster': 'gbtree',
 'colsample_bylevel': 1,
 'colsample_bynode': 1,
 'colsample_bytree': 0.7,
 'gamma': 0,
 'importance_type': 'gain',
 'learning_rate': 0.1,
 'max_delta_step': 0,
 'max_depth': 9,
 'min_child_weight': 10,
 'missing': None,
 'n_estimators': 100,
 'n_jobs': 1,
 'nthread': None,
 'objective': 'reg:linear',
 'random_state': 0,
 'reg_alpha': 0,
 'reg_lambda': 1,
 'scale_pos_weight': 1,
 'seed': None,
 'silent': None,
 'subsample': 1,
 'verbosity': 1,
 'eta': 0.004}
```

The Evaluation metrics for XGBOOST are

```
Mean Squared Error (MSE):  7.895139880481771
Mean Absolute Error (MAE):  1.8254969427587915
Root Mean Squared Error (RMSE):  2.8098291550344783
RMSPE is 103.24096713742972
MAPE is 50.56801221726399
R2 Score:  0.8948002461535709
Adjusted R2 Score:  0.8925053581172582
```

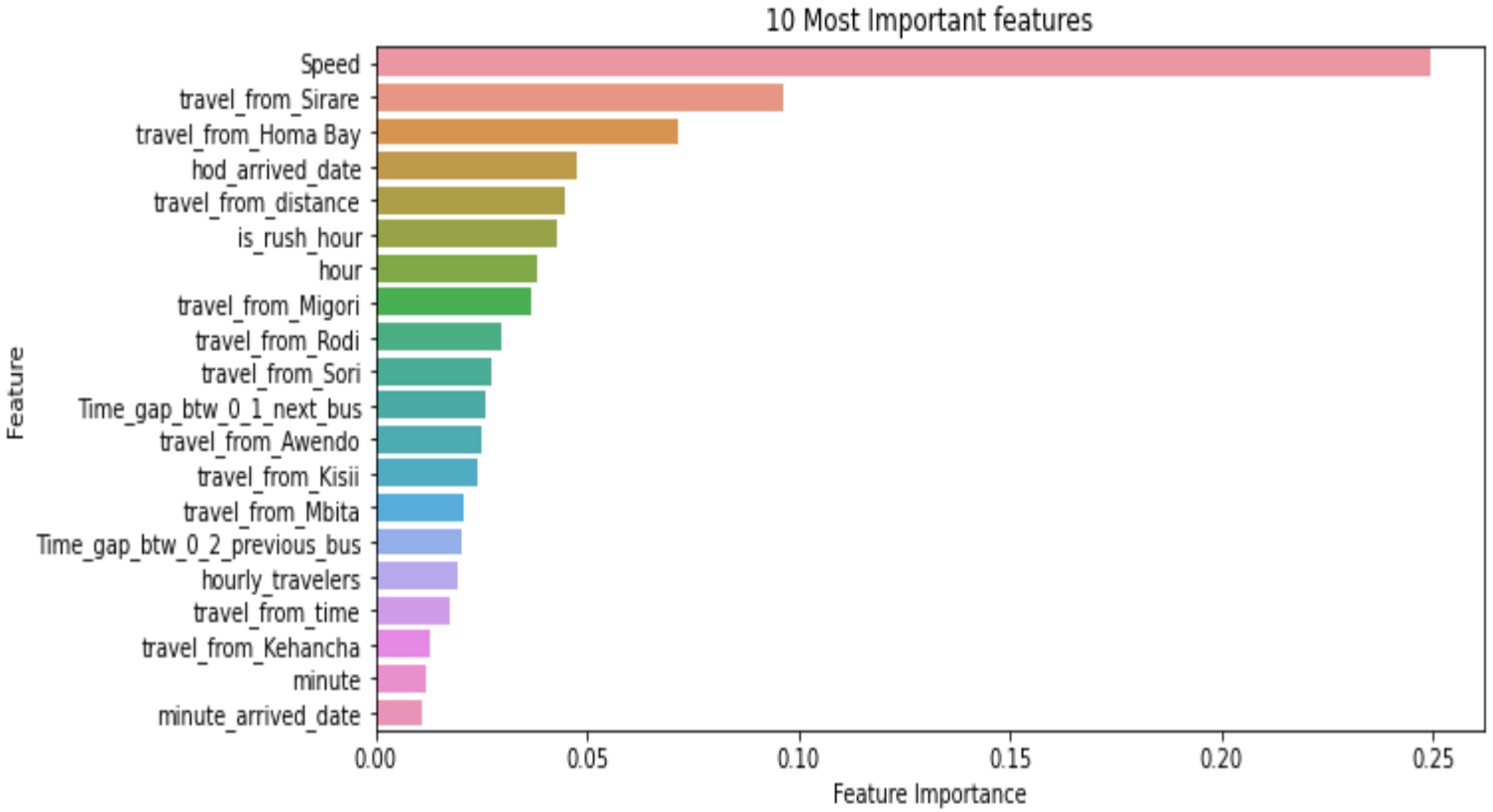
RESIDUAL ANALYSIS



Residual analysis is used to assess the appropriateness of a linear regression model by defining residuals and examining the residual plot graphs.

Residual (e) refers to the difference between observed value (y) vs predicted value (y ^). Every data point have one residual

FEATURE IMPORTANCE



From the graph, Speed is the most important feature of our data set.

CONCLUSION

We used different type of regression algorithms to train our model like, Linear Regression, Regularized linear regression (Lasso), GBM, Random Forest Regressor, XGboost regressor. and Also we tuned the parameters of Random forest regressor and XGboost regressor and also found the important features for training the model. Out of them XGboost with tuned hyperparameters gave the best result.