Taxi Trip Time Prediction

Technical Documentation

By

Prabin Deb Nath

Introduction:

Taxi services play an important role in daily commute for people in NYC, according to Wikipedia 1.6% of the overall population rely on taxis for their daily travelling.

With the increasing use of taxis companies try to provide their services as fast as possible. These services do come at a cost as they collect data and analyze it to find the factors that affect the trip durations, which then help in predicting the same.

1.Problem Statement:

We have been provided with a dataset containing around 1458644 trip records in it regarding the NYC taxi trips.

Our task is to build a model that predicts the total ride duration of taxi trips in New York City. The 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.

2. About the Data:

The dataset is based on the 2016 NYC Yellow Cab trip record data made available in Big Query on 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. It contains 1458644 rows and 11 columns.

- Dependent Feature:

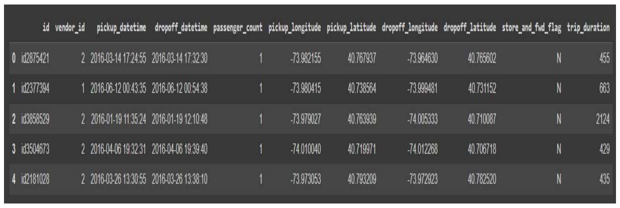
1. **trip_duration**: duration of the trip in seconds.

- Independent Features:

1. **id** - a unique identifier for each trip
2. **vendor_id** - a code indicating the provider associated with the trip record
3. **pickup_datetime** - date and time when the meter was engaged
4. **dropoff_datetime** - date and time when the meter was disengaged
5. **passenger_count** - the number of passengers in the vehicle (driver entered value)
6. **pickup_longitude** - the longitude where the meter was engaged
7. **pickup_latitude** - the latitude where the meter was engaged
8. **dropoff_longitude** - the longitude where the meter was disengaged
9. **dropoff_latitude** - the latitude where the meter was disengaged
10. **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

3. Exploratory Data Analysis:

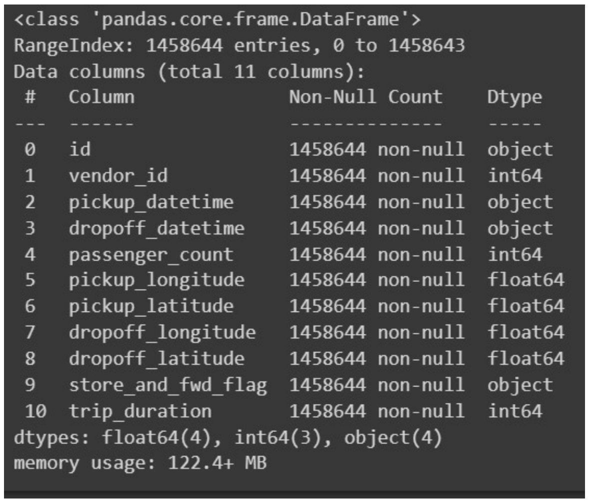
Let's get a quick glance on the data we would be working with:



	id	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	store_and_fwd_flag	trip_duration
0	id2875421	2	2016-03-14 17:24:55	2016-03-14 17:32:30	1	-73.982155	40.767937	-73.964630	40.765602	N	455
1	id2377394	1	2016-06-12 00:43:35	2016-06-12 00:54:38	1	-73.980415	40.738564	-73.999481	40.731152	N	663
2	id3858529	2	2016-01-19 11:35:24	2016-01-19 12:10:48	1	-73.979027	40.763939	-74.005333	40.710087	N	2124
3	id3504673	2	2016-04-06 19:32:31	2016-04-06 19:39:40	1	-74.010040	40.719971	-74.012268	40.706718	N	429
4	id2181028	2	2016-03-26 13:30:55	2016-03-26 13:38:10	1	-73.973053	40.793209	-73.972923	40.782520	N	435

The first five records of our data give us the basic idea about what we will be working with.

Lets use .info() to understand the features more clearly:



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1458644 entries, 0 to 1458643
Data columns (total 11 columns):
 #   Column              Non-Null Count    Dtype
---  ------              --------------    -----
 0   id                  1458644 non-null  object
 1   vendor_id           1458644 non-null  int64
 2   pickup_datetime     1458644 non-null  object
 3   dropoff_datetime    1458644 non-null  object
 4   passenger_count     1458644 non-null  int64
 5   pickup_longitude    1458644 non-null  float64
 6   pickup_latitude     1458644 non-null  float64
 7   dropoff_longitude   1458644 non-null  float64
 8   dropoff_latitude    1458644 non-null  float64
 9   store_and_fwd_flag  1458644 non-null  object
 10  trip_duration       1458644 non-null  int64
dtypes: float64(4), int64(3), object(4)
memory usage: 122.4+ MB
```

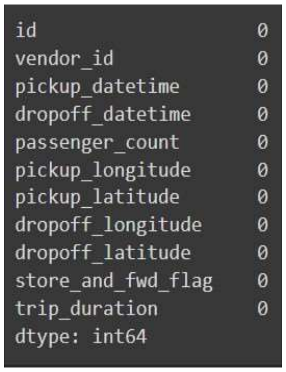
This provides a lot of information about the features:

- id, which is a categorical feature will be removed as it would be any help to us.
- pickup_datetime and dropoff_datetime are in object format which will be converted to datetime format so that the useful information can be extracted from it.

As seen before store_and_fwd_flag is a categorical feature which will be transformed or removed.

Now let's check if the data contains null values:

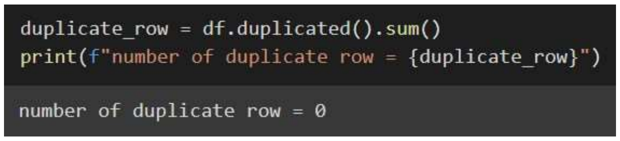
We will check for null values using the function df.isnull().sum() which will provide the sum of null values in each feature.



```
id                    0
vendor_id             0
pickup_datetime       0
dropoff_datetime      0
passenger_count       0
pickup_longitude      0
pickup_latitude       0
dropoff_longitude     0
dropoff_latitude      0
store_and_fwd_flag    0
trip_duration         0
dtype: int64
```

As we see there are no null values in the data.

In order to check with the duplicate records we use the function df.duplicated().sum()

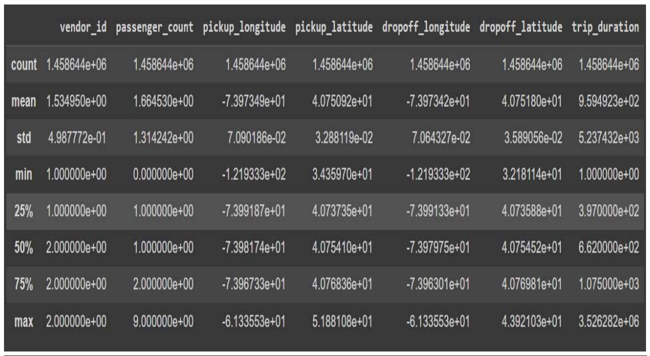


```
duplicate_row = df.duplicated().sum()
print(f"number of duplicate row = {duplicate_row}")

number of duplicate row = 0
```

As shown above there are no duplicate records.

Get some statistics about the data using the .describe() function:

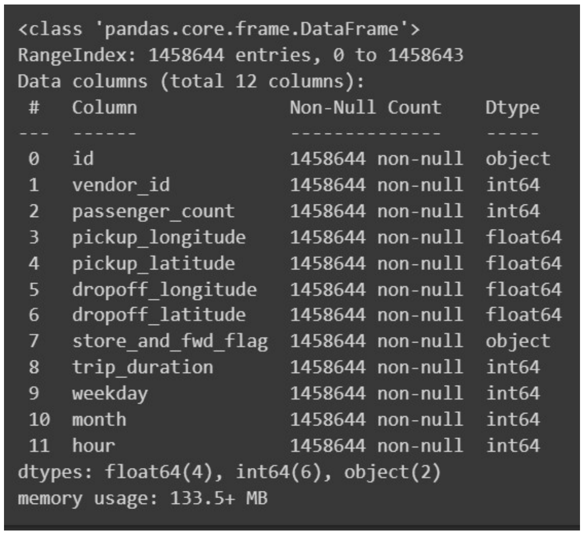


	vendor_id	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	trip_duration
count	1.458644e+06	1.458644e+06	1.458644e+06	1.458644e+06	1.458644e+06	1.458644e+06	1.458644e+06
mean	1.534950e+00	1.664530e+00	-7.397349e+01	4.075092e+01	-7.397342e+01	4.075180e+01	9.594923e+02
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

Points to be noted from the statistics:

- duration is given in seconds we will convert it into minutes, to do so we will divide the current values with 60.
- vendor_id consists of two values 1 and 2.
- passenger_count ranges from 0-9, the difference between the 75th percentile and the max value shows the presence of outliers.
- trip_duration_minutes also contains outliers.
- As we had already seen pickup_datetime and dropoff_datetime are in object format due to which it didn't show up in the statistics therefore before proceeding we will convert it into datetime and extract **day, month, pickup and hours** from the pickup_datetime.

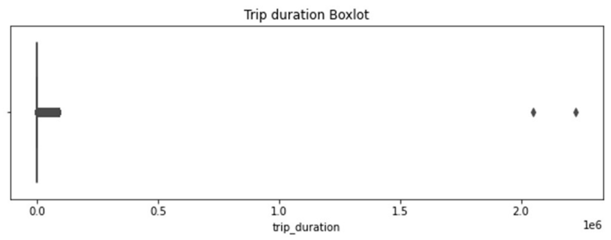
This is how the dataset will be after the above transformation:



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1458644 entries, 0 to 1458643
Data columns (total 12 columns):
 #   Column              Non-Null Count    Dtype
---  ------              --------------    -----
 0   id                  1458644 non-null  object
 1   vendor_id           1458644 non-null  int64
 2   passenger_count     1458644 non-null  int64
 3   pickup_longitude    1458644 non-null  float64
 4   pickup_latitude     1458644 non-null  float64
 5   dropoff_longitude   1458644 non-null  float64
 6   dropoff_latitude    1458644 non-null  float64
 7   store_and_fwd_flag  1458644 non-null  object
 8   trip_duration       1458644 non-null  int64
 9   weekday             1458644 non-null  int64
 10  month               1458644 non-null  int64
 11  hour                1458644 non-null  int64
dtypes: float64(4), int64(6), object(2)
memory usage: 133.5+ MB
```

Now let's look at each column:

1. Trip duration:

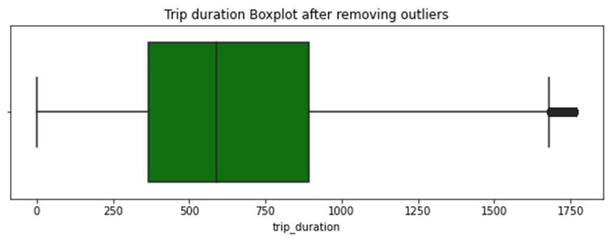


From the boxplot the we can see that there are many outliers in the data, for removing the outliers we use IQR range.

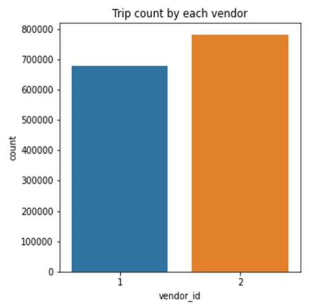
IQR = Q3 - Q1
lower_range = Q1 - (1.5 * IQR)
upper_range = Q3 - (1.5 * IQR)

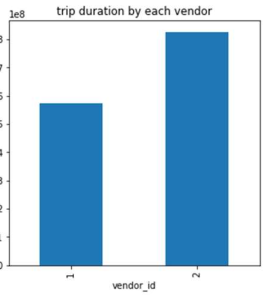
since distance cannot be less than 0 we use lower limit as 0 and upper limit from the upper range we got from the IQR method.

After removing the outliers this is how the boxplot looks like



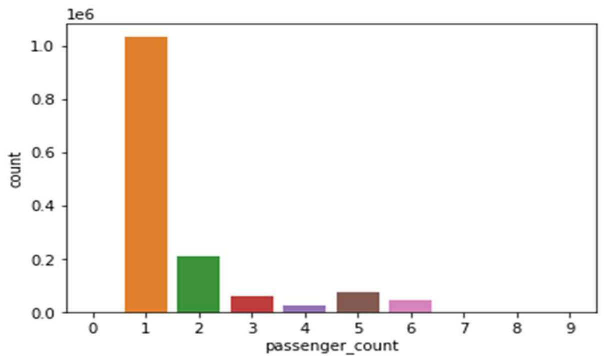
2. Vendor_id:





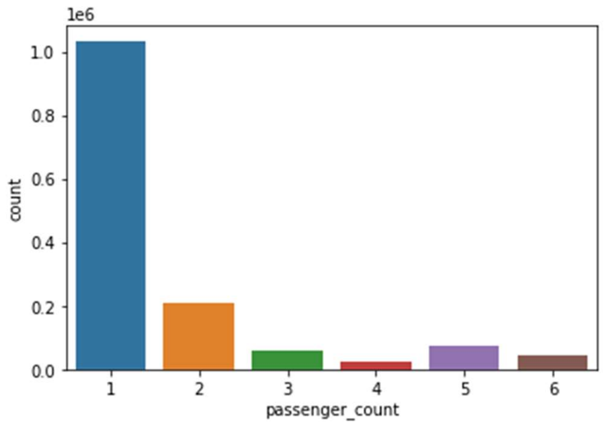
- Trip count of vendor 2 is more than that of trip count of vendor 2.
- also, maximum duration is covered by vendor 2

3. Passenger_count:



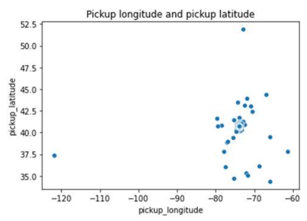
- some values are zero which mean either the trip was cancelled or there was an error in the data entry.
- 7, 8, 9 are extreme cases considering the capacity of a car, so we will get rid of them.

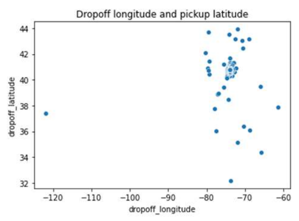
After removing the unwanted passenger count this is how it looks like.



4. Pickup and drop off longitude and latitude:

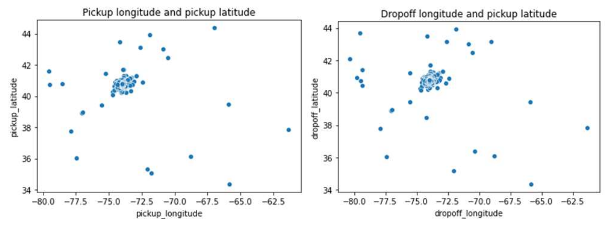
We create scatterplot for the longitude and latitude for pickup and drop off columns.



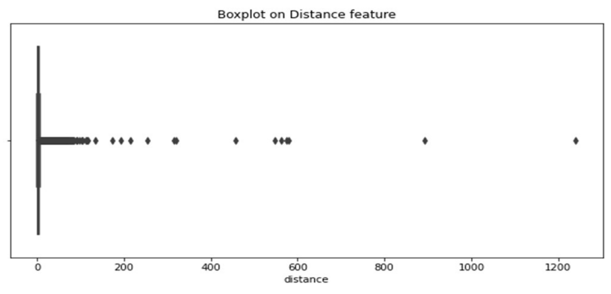


There are some outliers in the data, and we get rid of them.

After removing the outliers this is how the figures look likes.



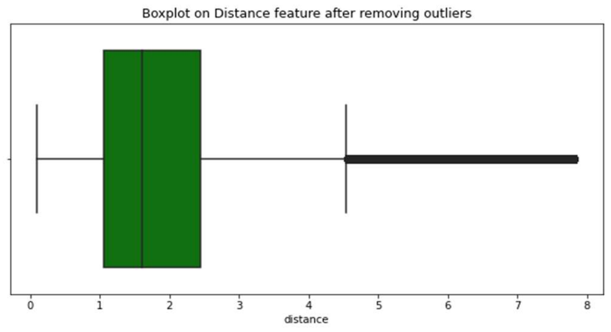
Now, using pickup and drop off longitude and latitude we create the **Distance** feature:



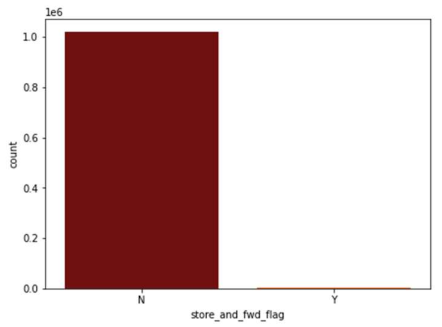
From analyzing the distance, we see that there are many outliers in the data, for removing the outliers we use IQR range

since distance can not be less then 0 we use lower limit as 0 and upper limit from the upper range we got from the IQR method.

After removing the outliers this is how the boxplot looks like

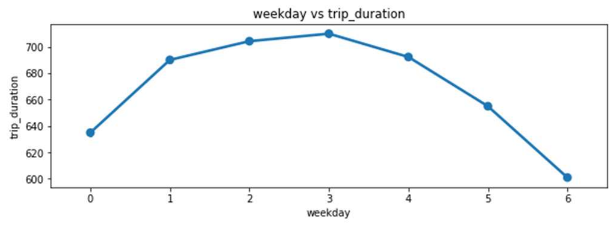


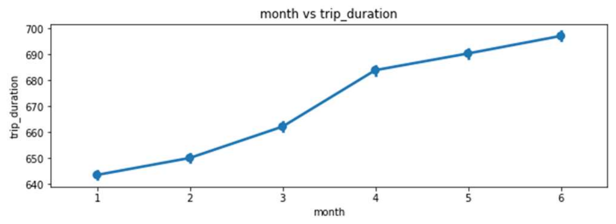
5. Store_and_fwd-flag

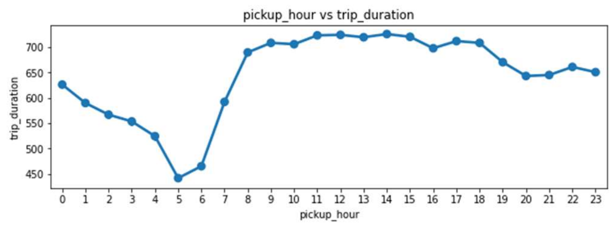


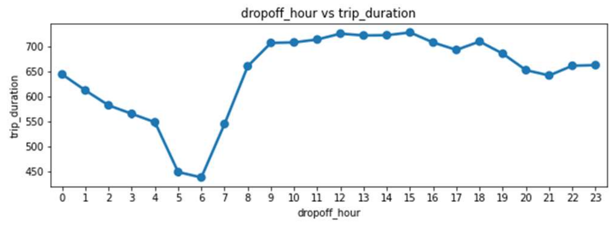
- most of the data values are N and only few values are Y, which means most of the data was uploaded directly without storing it and forwarding.
- This is a categorical variable which will be converted to numerical using dummies.

6. Pickup_datetime/dropoff_datetime:









- Commute is least in early morning and late night.
- Trip duration decreases as the weekend approaches, it makes sense as most of the people either stay at home or go for vacations.
- Trip duration increases after February.

4. Feature Engineering and data preparation:

Transform pickup hours.

Since there are 24 different values hour columns it would be better to categorize it.
between 7 - 9 = morning(7-10),
between 11 - 15= afternoon(11-15)
between 16 - 19 = evening(16-19),
between 20 - 23 = night(20-23),
between 0 - 6 = latenight(0-6)

We create new features from the pickup_hour and dropoff_hour named pickup_time and dropoff_time,

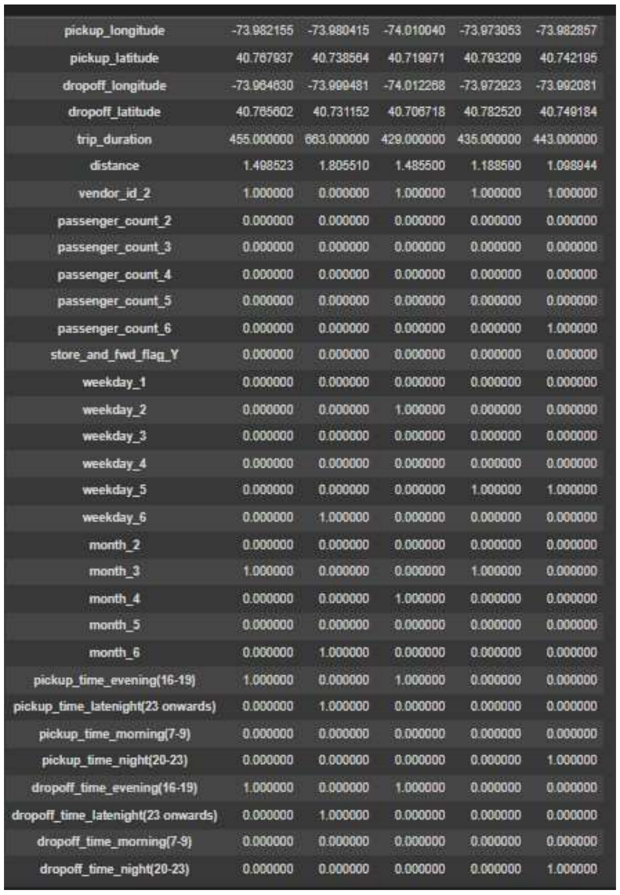
Get dummies for all the categorical variables:

We got dummy variables for the categorical features:

['vendor_id', 'passenger_count', 'store_and_fwd_flag', 'weekday', 'month', 'pickup_time', 'dropoff_time']

Now we will prepare the data before we start building models to train and test it.

After creating the dummy variables and dropping the un necessary columns this is transpose view of our data.



pickup_longitude	-73.982155	-73.980415	-74.010040	-73.973053	-73.982857
pickup_latitude	40.767937	40.738564	40.719971	40.793209	40.742195
dropoff_longitude	-73.964630	-73.999481	-74.012268	-73.972923	-73.992081
dropoff_latitude	40.765602	40.731152	40.706718	40.782520	40.749184
trip_duration	455.000000	663.000000	429.000000	435.000000	443.000000
distance	1.498523	1.805510	1.485500	1.188590	1.098944
vendor_id_2	1.000000	0.000000	1.000000	1.000000	1.000000
passenger_count_2	0.000000	0.000000	0.000000	0.000000	0.000000
passenger_count_3	0.000000	0.000000	0.000000	0.000000	0.000000
passenger_count_4	0.000000	0.000000	0.000000	0.000000	0.000000
passenger_count_5	0.000000	0.000000	0.000000	0.000000	0.000000
passenger_count_6	0.000000	0.000000	0.000000	0.000000	1.000000
store_and_fwd_flag_Y	0.000000	0.000000	0.000000	0.000000	0.000000
weekday_1	0.000000	0.000000	0.000000	0.000000	0.000000
weekday_2	0.000000	0.000000	1.000000	0.000000	0.000000
weekday_3	0.000000	0.000000	0.000000	0.000000	0.000000
weekday_4	0.000000	0.000000	0.000000	0.000000	0.000000
weekday_5	0.000000	0.000000	0.000000	1.000000	1.000000
weekday_6	0.000000	1.000000	0.000000	0.000000	0.000000
month_2	0.000000	0.000000	0.000000	0.000000	0.000000
month_3	1.000000	0.000000	0.000000	1.000000	0.000000
month_4	0.000000	0.000000	1.000000	0.000000	0.000000
month_5	0.000000	0.000000	0.000000	0.000000	0.000000
month_6	0.000000	1.000000	0.000000	0.000000	0.000000
pickup_time_evening(16-19)	1.000000	0.000000	1.000000	0.000000	0.000000
pickup_time_latenight(23 onwards)	0.000000	1.000000	0.000000	0.000000	0.000000
pickup_time_morning(7-9)	0.000000	0.000000	0.000000	0.000000	0.000000
pickup_time_night(20-23)	0.000000	0.000000	0.000000	0.000000	1.000000
dropoff_time_evening(16-19)	1.000000	0.000000	1.000000	0.000000	0.000000
dropoff_time_latenight(23 onwards)	0.000000	1.000000	0.000000	0.000000	0.000000
dropoff_time_morning(7-9)	0.000000	0.000000	0.000000	0.000000	0.000000
dropoff_time_night(20-23)	0.000000	0.000000	0.000000	0.000000	1.000000

- Now let's separate the dependent and the independent variables, where y is the dependent variable trip_duration and X contains rest of the features in our dataset.
- Once we have X and y, we will do the train test split to separate the training and the testing data that we will use to build and validate our regression models.
- Finally, we will scale the data using StandardScaler. This is done to standardize the data before feeding them to the algorithms.

5. Model Selection:

We have prepared our data now its time to select the best performing model:

For model selection we run the following algorithm without tuning it:

1. **Linear Regression**
2. **Decision Tree Regressor**
3. **XG Boost Regressor.**
4. **Hist Gradient Boosting Regressor**
5. **AdaBoost Regressor.**

1. Linear Regression:

Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data. One variable is an explanatory variable, and the other is a dependent variable.

2. Decision Tree Regressor:

Decision tree regression observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output. Continuous output means that the output/result is not discrete, i.e., it is not represented just by a discrete, known set of numbers or values.

3. XG Boost Regressor:

XGBoost is a gradient boosting algorithm, a common technique in ensemble learning, ensemble learning is a type of machine learning that enlists many models to make predictions together.
XGBoost operates on decision trees, models that construct a graph that examines the input under various "if" statements (vertices in the graph). Whether the "if" condition is satisfied influences the next "if" condition and eventual prediction. XGBoost the Algorithm progressively adds more

and more "if" conditions to the decision tree to build a stronger model.

4. Hist Gradient Boosting Regressor:

Gradient boosting works by building simpler (weak) prediction models sequentially where each model tries to predict the error left over by the previous model. It works on the principle of improving mistakes of the previous learner through the next learner.

Hist Gradient Boosting Regressor or Histogram-based Gradient Boosting Regression is much faster than **Gradient Boosting Regressor** for big datasets. This estimator has native support for missing values (NaN). During training, the tree grower learns at each split point whether samples with missing values should go to the left or right child, based on the potential gain. When predicting, samples with missing values are assigned to the left or right child consequently. If no missing values were encountered for a given feature during training, then samples with missing values are mapped to whichever child has the most samples.

It is faster than Gradient Boosting since the bottleneck of a gradient boosting procedure is building the decision trees. Building a traditional decision tree requires sorting the samples at each node (for each feature). Sorting is needed so that the potential gain of a split point can be computed efficiently. Splitting a single node has thus a complexity of $\mathcal{O}(n_{\text{features}} \times n\log(n))$ where *n* is the number of samples at the node.

Hist Gradient Boosting Regressor in contrast, do not require sorting the feature values and instead use a data-structure called a histogram, where the samples are implicitly ordered. This reduces the time taken to build models.

5. Ada Boost Regressor:

AdaBoost also called Adaptive Boosting is a technique in Machine Learning used as an Ensemble Method. The most common algorithm used with AdaBoost is decision trees with one level that means with Decision trees. What this algorithm does is that it builds a model and gives equal weights to all the data points. It then assigns higher weights to points that are wrongly classified. Now all the points which have higher weights are given more importance in the next model. It will keep training models until and unless a lower error is received.

1. **Root Mean Squared Error (RMSE):**
   It used measure of the differences between values (sample or population values) predicted by a model or an estimator and the values observed.

   The RMSE of a predicted model with respect to the estimated variable x_{model} is defined as the square root of the mean squared error.

$$\textbf{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(X_{obs,i}-X_{model,i})^2}{n}}$$

   Where, X_{obs} is observed values, X_{model} is modelled values at time i.

   The lower the error better the fit.

2. **Coefficient of Determination (R^2 score):**

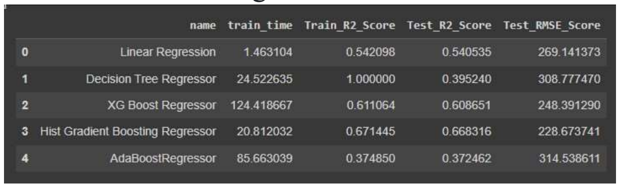
   R-squared (R^2) is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model.

$$\mathbf{r} = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[n^*(\sum x^2 - (\sum x)^2)] * [n^*(\sum y^2 - (\sum y)^2)]}}$$

- r = The Correlation coefficient
- n = number in the given dataset
- x = first variable in the context
- y = second variable

Selecting the best performing model:

In order to select the best performing model, we will use the following metrics:



	name	train_time	Train_R2_Score	Test_R2_Score	Test_RMSE_Score
0	Linear Regression	1.463104	0.542098	0.540535	269.141373
1	Decision Tree Regressor	24.522635	1.000000	0.395240	308.777470
2	XG Boost Regressor	124.418667	0.611064	0.608651	248.391290
3	Hist Gradient Boosting Regressor	20.812032	0.671445	0.668316	228.673741
4	AdaBoostRegressor	85.663039	0.374850	0.372462	314.538611

From the above table we see that performance metrices of Hist Gradient Boosting is Better then the rest of the Algorithms and also time required for this is also low compared to others.

So, we now tune the Hyper parameters of Hist Gradient Boosting Regressor Algorithm.

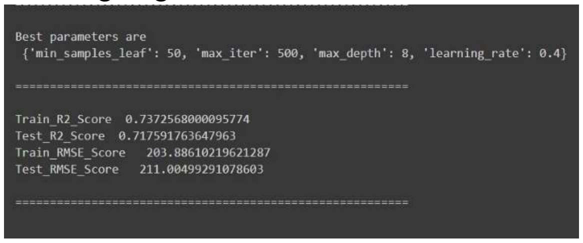
7. Hyperparameter Tuning:

Hyperparameters are sets of information that are used to control the way of learning an algorithm. Their definitions impact parameters of the models, seen as a way of learning, change from the new hyperparameters. This set of values affects performance, stability, and interpretation of a model. Each algorithm requires a specific hyperparameters grid that can be adjusted according to the business problem. Hyperparameters alter the way a model learns to trigger this training algorithm after parameters to generate outputs.

We will use **Randomized Search CV** for hyperparameter tuning and cross validations to select the best values to improve the performance of our model.

8. Conclusion:

So far, we have done EDA, Outlier Treatment, Feature Engineering, Encoding of Categorical variables, and Model Building with the data that was provided to us.
In all these models our score (r2) revolves in the range of 46 to 58%.
With Hyperparameter Tuning we were able to attain following r2 score for Hist Gradient Boosting Regressor.



```
Best parameters are
 {'min_samples_leaf': 50, 'max_iter': 500, 'max_depth': 8, 'learning_rate': 0.4}

==========================================================

Train_R2_Score  0.7372568000095774
Test_R2_Score  0.717591763647963
Train_RMSE_Score   203.88610219621287
Test_RMSE_Score   211.00499291078603

==========================================================
```

Hence, we can proceed with the Hist Gradient Boosting Regressor for future predictions of trip duration of NYC taxis.

References:

- Geek for Geeks.
- Analytics Vidhya.
- Wikipedia.
- Towards Data science.
- Sk Learn Documentation.