**NYC-Taxi-Trip-Duration Prediction**

**A PROJECT REPORT**

**Submitted in partial fulfilment of the requirements for the award of the degree**

**of**

**MASTERS OF COMPUTER APPLICATION (MCA)**

**in**

**Artificial Intelligence & Machine Learning**

**Submitted by**

**Naveen Rai**

**LNCCMCA11180**

**Under the Supervision of**

**VED LAD**

**LNCT UNIVERSITY, BHOPAL (M.P.)**

# Image result for lnctuniversity logo

MCA A.I.M.L

**LNCT UNIVERSITY, BHOPAL (M.P.)**

**BHOPAL**  
Madhya Pradesh, INDIA

MAY 2023

**MCA – Artificial Intelligence & Machine Learning**

LNCT UNIVERSITY BHOPAL (MADHYA PRADESH), INDIA

BHOPAL

Date: 04 - July - 2023

**CERTIFICATE**

This is to certify that the project titled **NYC-Taxi-Trip-Duration Prediction** is a record of bonafide the work done by **Naveen Rai** (LNCCMCA11180) submitted in partial fulfilment of the requirements for the award of the Degree of Masters of Computer Application (MCA) in **Artificial Intelligence & Machine Learning** of LNCT University BHOPAL MP, during the academic year 2022-23.

**VED LAD sir**

*Project Guide, Department of Computer Application*

*LNCT UNIVERSITY BHOPAL*

**Dr. Kavita Kanathey**

*Head, Department of MCA AIML*

*LNCT University BHOPAL*

(*On company letterhead)*

BHOPAL

Date: 04 - July - 2023

**CERTIFICATE**

This is to certify that the project entitled **NYC-Taxi-Trip-Duration Prediction (regression)** was carried out by **Naveen Rai** (LNCCMCA11180) at **SAMATRIX.IO, BHOPAL** under my guidance during **MAY, 2023** to **JUNE 2023**.

**VED LAD sir**

Designation,

Organisation Name, City

**ACKNOWLEDGMENTS**

This section should contain the acknowledgements due to the Dean/Director, Dept HOD, Project supervisor, company personnel, department guide, department Project Coordinator and Co-coordinator, Laboratory In-charge (where the work was carried out) and other faculty members whose assistance was sought during the project work.

**(Naveen Rai)**

**(LNCCMCA11180)**

**ABSTRACT**

In New York City many of people commute to different regions of city via taxi. A lot of streets and roads in New York city are quite busy due to traffic jams, construction, or road blockage etc. Therefore, it is very important to predict the trip duration of taxi so that the user will know how much time it will take to commute from one place to other. Also, due to the increasing popularity of app-based taxi such as ola or uber and there competitive pricing levels. Decisions has to be taken by the user for opting which one to choose based on trip pricing and duration. This prediction also helps drivers to choose route having lesser trip time. We were provided with dataset which is released by NYC Taxi and Limousine Commission. This dataset contains pickup time, drop-off time, geo- coordinates, number of passengers, trip duration and several other variables.

Our primary motives are to analyse the dataset, perform feature engineering to comes up with suitable independent features and building a good model that will help us in predicting the trip duration of NYC taxi.

Here, for prediction the taxi trip duration we have applied a linear regression, lasso, and ridge regression and then we have applied XGBoost and LightGBM. To find out which will give better acuracy and with lesser amount of prediction time. At last, a comparison of the two mentioned algorithms facilitates us to decide that XGBoost is more fitter and efficient than Multi-Layer Perceptron for taxi trip duration-based predictions

***Keywords: machine learning, surge pricing, dynamic pricing, classified labels***

# **Introduction**

More than 7 billion people exist on earth. With necessities of food, water and shelter there also a key requirement of commutating from one place to other. Rapid advancement in technology in the last two decades leads to adaption of a more efficient way of transportation via internet and app-based transport system. New York city is one of such advanced city with extensive use of transportation via subways, buses and taxi services. New York has more then 10,000 plus taxi and nearly 50% of population doesn’t have a personal vehicle. Due to this facts most people used taxi has a there primary mode of transport and it accounts for more than 100 millions taxi trips per year.

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). This dataset contains around 1458644 records and 11 features.

Out of numerous machine learning algorithm we have selected Xgboost and LightGBM repressors for our used case. More accurately prediction will lead to make better taxi trip

duration prediction not in New York but also applicable to other city as well in future and make user taking better decision for choosing right taxi for there commute.

# **Observations:**

# We can observe that both the models shows somewhat similar learning rate but with visible differences in error rates.

# Gradient boosting performed very well out pof all the models

# XGBoost training curve on the other hand starts quite low and further improves with the increase in the training size and it too plateau towards the end.

# Validation curve seems to show similar trend in both the models i.e. XGBoost curve learning is quite fast and more accurate as compared to the RF one.

# Both the models seems to suffer from high variance since the training curve error is very less in both the models.

# The large gap at the end also indicates that the model suffers from quite a low bias i.e. over fitting the training data.

# Also, both the model’s still has potential to decrease and coverage towards the training curve by the end.

# At this point, here are a few things we could do to improve our models:

# Add more training instances to improve validation curve in the XGBoost model. Increase the regularization for the learning algorithm. This should decrease the variance and increase the bias towards the validation curve, it will build less complex models.

# **Conclusion:**

That's it! We reached the end of our exercise.

Starting with loading the data so far, we have done EDA, null values treatment, encoding of categorical columns, feature selection and then model building.

In all these models our accuracy revolves in the range of 70 to 74%.

And there is no such improvement in accuracy score even after hyperparameter tuning.

So, the accuracy of our best model is 73% which can be said to be good for this large dataset. This performance could be due to various reasons like no proper pattern of

data, too much data, not enough relevant features.

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No** | **Figure Title** | **Page No** |
|  |  |  |
| 1 | Bar plot for Trip duration | 17 |
| 2 | Drop offs longitude | 18 |
| 3 | Numbers of drop offs each weeks and days | 19 |
| 4 | Co – relation heat map | 20 |
| 5 | Residual Fitted line – trip duration hours | 21 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CONTENTS** | | | | | |
|  | | | | | Page No |
| Acknowledgement | | | |  | i |
| Abstract | | | |  | ii |
| List of Figures | | | |  | iii |
|  | | | |  |  |
|  | | | | | |
| **Chapter 1** | | | **INTRODUCTION** | |  |
|  | **1.1** | Problem Statement | | | 9 |
|  | **1.2** | Introduction | | | 9 |
|  | **1.3** | Trip duration & Trip Duration Variation | | | 10 |
|  | | | | | |
| **Chapter 2** | | | BACKGROUND THEORY | |  |
|  | **2.1** | Feature Engineering | | | 11 |
|  | **2.2** | Exploratory Data Analysis | | | 11 |
|  | | | | | |
| **Chapter 3** | | | **METHODOLOGY** | |  |
|  | **3.1** | Algorithms | | | 12 |
|  | **3.2** | Linear Regression: XGBoost: Gradient Boosting: LightGBM: | | | 13-14 |
|  | | | | | |
| **Chapter 4** | | | **RESULT ANALYSIS** | |  |
|  | **4.1** | Model performance | | | 13 |
|  | **4.2** | Hyper parameter tuning: | | | 14-15 |
|  | | | | | |
| **Chapter 5** | | | **CONCLUSIONS & FUTURE SCOPE** | | 15 |
|  | **5.1** | Code/ Result Analysis | | | 15-30 |
|  | **5.2** | Work Conclusions Future Scope of Work | | | 25-30 |
|  | | | | | |
| **REFERENCES** | | | | | 28 |
| **ANNEXURES (OPTIONAL)** | | | | | 24-30 |
| **PROJECT DETAILS** | | | | | 32 |

# **Problem Statement**

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 main objective is to build a predictive model, which could help them in predicting the trip duration of taxi. This would in turn help them in matching the right cabs with the right customers quickly and efficiently.

* + 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

# **Introduction**

More than 7 billion people exist on earth. With necessities of food, water and shelter there also a key requirement of commutating from one place to other. Rapid advancement in technology in the last two decades leads to adaption of a more efficient way of transportation via internet and app-based transport system. New York city is one of such advanced city with extensive use of transportation via subways, buses and taxi services. New York has more then 10,000 plus taxi and nearly 50% of population doesn’t have a personal vehicle. Due to this facts most people used taxi has a there primary mode of transport and it accounts for more than 100 millions taxi trips per year.

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). This dataset contains around 1458644 records and 11 features.

Out of numerous machine learning algorithm we have selected Xgboost and LightGBM repressors for our used case. More accurately prediction will lead to make better taxi trip

duration prediction not in New York but also applicable to other city as well in future and make user taking better decision for choosing right taxi for there commute.

## Trip duration & Trip Duration Variation

Trip duration normally be calculated based on the distance between pickup and drop-off point and average speed of the vehicle covering this distance. However, there are many factors which affects the trip duration. Following are some of the factors:

* Peak hours: there are certain hours where route are might get busy due to moment of peoples commutating from office to home or vice versa.
* bad weather conditions (rain, snow, etc)
* big events or festivals
* traffic conditions

# **Feature Engineering:**

## Data Loading and general check-ups

We have loaded the data from the given csv files using a function from pandas library. Then we checked the general information about data. We observed that the data contains 1458644 records and 11 features. We see that our data contains three different data types i.e. floats, strings and date time objects.

## Null values Treatment

We inspected the dataset and found out that our dataset has no null value present in it. So, no need to do null value treatment.

## Exploratory Data Analysis

We begin our EDA by first checking the distribution of our dependent variable i.e. trip duration. We observed that the data is highly positively skewed. We also plotted the box plot and observed that there are many outliers present in the variable. To cross check this trip duration we have calculated the difference in pick and drop off timing and matched with trip duration we observed no difference. Thus, there is no miscalculation or falsified entries. To eliminate the outliers, we have segregated the data variable into different segment and observed that majority of trip duration is within an hour some observations are within two days but a very few observation are having more than two days. We eliminate such values from out dataset.

We removed id variable as it doesn’t give much interpretation. We then calculated the distance based on haversine formula from pickup and drop-off latitude and longitude. Then we plotted the box plot for the variable and observed there are many outlier so we segregate this variable and see that most of the trip are within 10km, some trip are within 50km while a very few trip crosses 50km. so we eliminate trip with 0 and above 50km distance.

We then checked for categorical variable store\_and\_fwd\_flag and passenger\_count. We observed the store and fwd. flag contain majority of one category. So we drop this feature. Passenger count variable has entries from 0 to 9. Since there is no trips with 0 passenger either this a miss entry or the driver forgot to enter passenger count of that trip. Also in a taxi maximum six person are allowed to sit including minor. So we eliminate 0 and 7-9 records from our dataset.

We also created some more feature i.e. pickup month, pickup weekday and pickup hour. To get a good insight of trip duration and drop pickup date and drop-off time column. Then we checked for correlation between variables and observed that geographic coordinates are very less correlated and VIF is also high between this variables so we drop off this variable from our data set.

## Encoding of categorical columns

Since some of our categorical variable are in string format. So we cannot passed this variable to our model directly so we have to use one hot encoding to convert it into numerical variable having binary integers 0 and 1.

## Standardization of features

This is one of the important step for getting good accuracy as you can see there are some columns having different ranges of values then other column. Therefore. It is important to do scaling the data so that our data set will have uniformity and we get good accuracy. So, here we use MinMaxscaler function.

## Fitting different models

For modelling we tried various classification algorithms like:

1. **Linear Regression**
2. **XGBoost classifier**
3. **LightGBM**

## Tuning the hyperparameters for better accuracy

Tuning the hyper parameters of respective algorithms is necessary for getting better accuracy.

# **Algorithms:**

## 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 linear combination of independent variable (x).

Y=B0+B1x1+B2x2

The cost function for linear regression is given by:

Minimum sum of square error MSSE= (𝑌 𝑎𝑐𝑡 − 𝑌 𝑝𝑟𝑒𝑑)2

1

∑ 𝑖 𝑖

**Chart, scatter chart

Description automatically generated**

## XGBoost:

Sometime in building a model. We cannot just rely on the result of a single model. Ensemble offer a systematic solution for this by combining the prediction of multiple model. The resultant model is superior then individual model called base learner and is obtained from aggregation of base learner prediction. Bagging and boosting are two types of ensemble method.

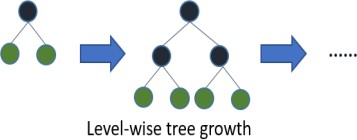
XGBoost comes under boosting and is known as extra gradient boosting. GBM first calculates the model using X and Y then after the prediction is obtain. It will again calculates the model based on residual of previous model, here loss function will give more weightage to error of previous model. and this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

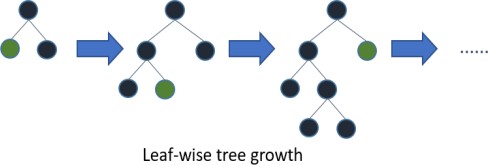
* Regularization
* Parallel Processing
* High Flexibility
* Handles Missing values
* Tree pruning
* Built in cross validation
* Continuous on existing model

## LightGBM:

Sometime in building a model Light GBM is a fast, distributed high performance gradient boosting framework. It is widely used for ranking, classification, regression, and many other machines learning task.

Light GBM 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.





# **Model performance:**

The model performance can be evaluated by various regression metrics such as:

## Mean Squared Error (MSE):

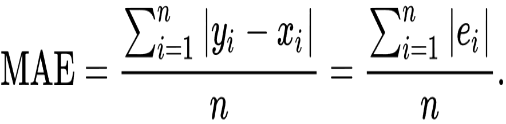
Mean squared error is the most widely used evaluation metric for regression task. It is the average of squared difference between actual and predicted value of dependent variable

A picture containing text, clock, watch, gauge

Description automatically generated

## Mean Absolute Error (MAE):

**Mean absolute error** (**MAE**) is a measure of [errors](https://en.wikipedia.org/wiki/Error_(statistics)) between paired observations expressing the same phenomenon. Examples of *Y* versus *X* include comparisons of predicted versus observed, subsequent time versus initial time, and one technique of measurement versus an alternative technique of measurement. MAE is calculated as the **sum of absolute errors** divided by the [sample size](https://en.wikipedia.org/wiki/Sample_size).

****

{\displaystyle \mathrm {MAE} ={\frac {\sum \_{i=1}^{n}\left|y\_{i}-x\_{i}\right|}{n}}={\frac {\sum \_{i=1}^{n}\left|e\_{i}\right|}{n}}.}

## R2 Score :

Coefficient of determination also called as R2 score is used to evaluate the performance of a linear regression model. It is the amount of the variation in the output dependent attribute which is predictable from the input independent variable(s). It is used to check how well-observed results are reproduced by the model, depending on the ratio of total deviation of results described by the model.

R2= 1- SSres / SStot

Where,

SSres is the sum of squares of the residual errors.

SStot is the total sum of the errors.

## Adjusted R2 Score:

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.

R2 tends to optimistically estimate the fit of the linear regression. It always increases as the number of effects are included in the model. Adjusted R2 attempts to correct for this overestimation. Adjusted R2 might decrease if a specific effect does not improve the model.

Adjusted R squared is calculated by dividing the residual mean square error by the total mean square error (which is the sample variance of the target field). The result is then subtracted from 1.

Text

Description automatically generatedAdjusted R2 is always less than or equal to R2. A value of 1 indicates a model that perfectly predicts values in the target field. A value that is less than or equal to 0 indicates a model that has no predictive value. In the real world, adjusted R2 lies between these values.

# **Hyper parameter 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 used Grid Search CV, Randomized Search CV and Bayesian Optimization for hyperparameter tuning. This also results in cross validation and in our case we divided the dataset into different folds. The best performance improvement among the three was by Bayesian Optimization.

## Grid Search CV-Grid Search combines a selection of hyperparameters established by the scientist and runs through all of them to evaluate the model’s performance. Its advantage is that it is a simple technique that will go through all the programmed combinations. The biggest disadvantage is that it traverses a specific region of the parameter space and cannot understand which movement or which region of the space is important to optimize the model.

## Randomized Search CV-

## In Random Search, the hyperparameters are chosen at random within a range of values

## that it can assume. The advantage of this method is that there is a greater chance of finding

## regions of the cost minimization space with more suitable hyperparameters, since the choice

## for each iteration is random. The disadvantage of this method is that the combination

## of hyperparameters is beyond the scientist’s control.

## The first paragraph should outline the importance of the work /topic in the present day scenario, hence leading to the objective of the project work.

**Code/Result Analysis**

import numpy as np

import pandas as pd

from numpy import math

from haversine import haversine

import xgboost

from sklearn.preprocessing import MinMaxScaler

from sklearn.preprocessing import MinMaxScaler

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.tree import DecisionTreeRegressor

from sklearn.model\_selection import GridSearchCV

from sklearn.metrics import r2\_score

from sklearn.metrics import mean\_squared\_error

from sklearn.linear\_model import Lasso

from sklearn.linear\_model import Ridge

import seaborn as sns

from datetime import datetime

import warnings

warnings.filterwarnings("ignore")

import warnings

from pylab import rcParams

%matplotlib inline

import matplotlib.pyplot as plt

import seaborn as sns; sns.set()

warnings.filterwarnings('ignore')

**mount to a drive**

from google.colab import drive

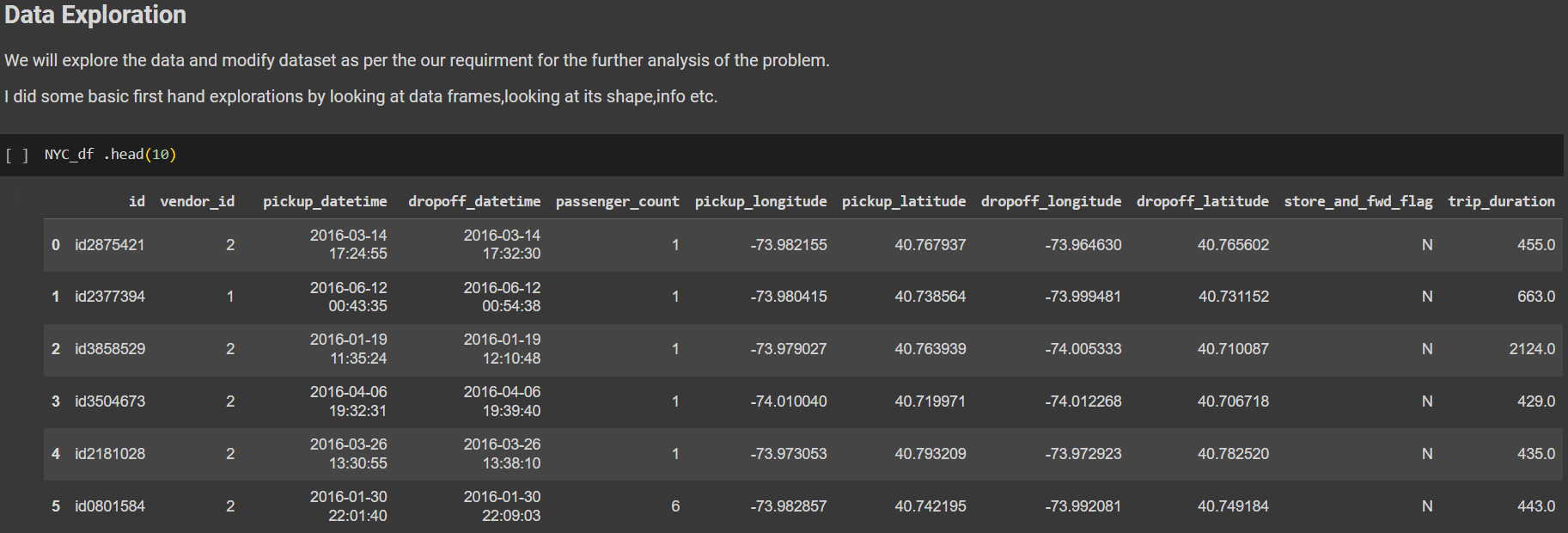
drive.mount('/content/drive')

**Load dataset**

path ="/content/NYC Taxi Data.csv"

NYC\_df = pd.read\_csv(path)

**Data Exploration**



To checked for outlier we segregate our dependent variable into different categories i.e trip whose duration is less then 1min, within 10mins,within hour,within day, within two day and more than two day

plt.figure(figsize=[10,5])

labels=['less then 1min','within 10 mins','within 30 mins','within hour','within day','within two days','more then two day']

NYC\_df.groupby(pd.cut(NYC\_df['trip\_duration'],bins=[0,60,600,1800,3600,86400,86400\*2,10000000],labels=labels))['trip\_duration'].count().plot(kind='bar',fontsize=10)

plt.title("Bar plot for trip duration")

plt.ylabel("trip counts")

plt.ylabel("trip duration")

plt.xticks(rotation=45)

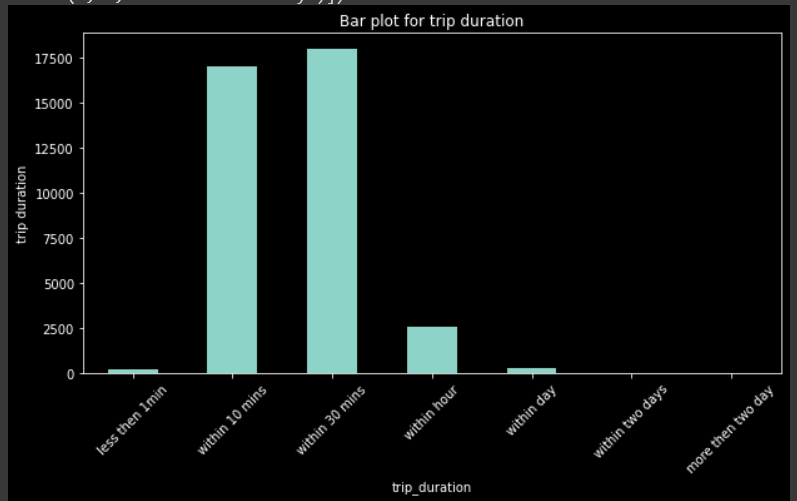


Fig. 01

## Map Visualization

We shall visualize the Taxi pickup locations by placing long and lat marker on the MAP of the US. So that we can analyze below questions:

* Are all pickups constrained to NYC and it's surrounding areas?
* Is there any unusual location of the pickup?
* Are the lat long constrained to the land area of the US and nowhere else?

city\_long\_border = [-74.03, -73.75]

city\_lat\_border = [40.63,40.85]

NYC\_df.plot(kind='scatter', x='dropoff\_longitude',y='dropoff\_latitude',

          color='Red',

          s=0.2, alpha =.6)

plt.title('Dropoffs')

plt.ylim(city\_lat\_border)

plt.xlim(city\_long\_border)

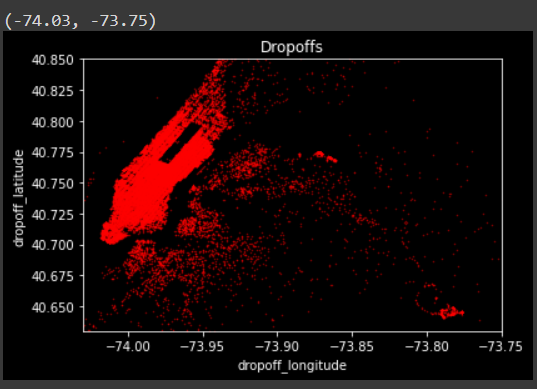


Fig 02

# **FEATURE ENGINEERING**

Feature engineering is a machine learning technique that leverages data to create new variables that aren’t in the training set. It can produce new features for both supervised and unsupervised learning, with the goal of simplifying and speeding up data transformations while also enhancing model accuracy.

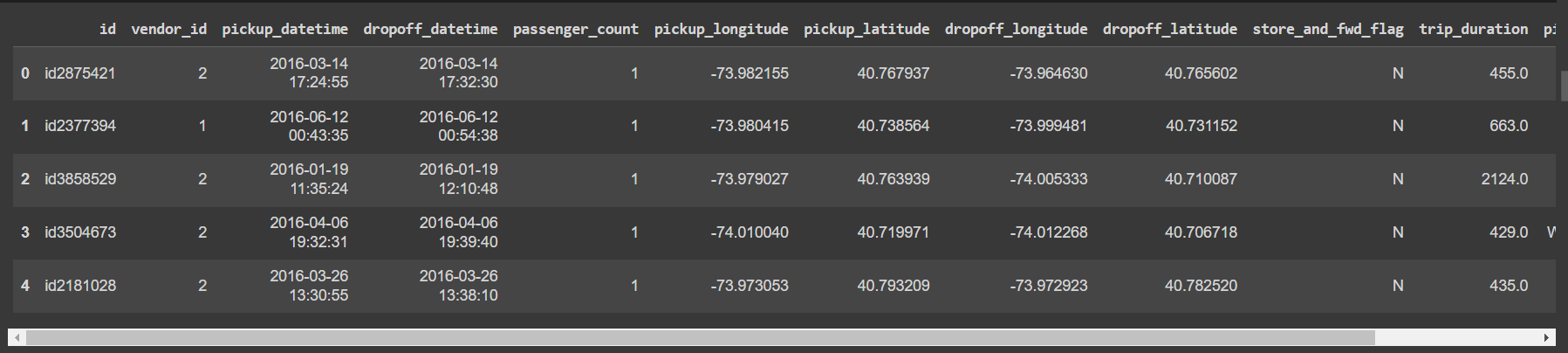
#Calculate and assign new columns to the dataframe such as pickupday,

#dropoffday which will help us to gain more insights from the data.

NYC\_df['pickup\_day']=NYC\_df['pickup\_datetime'].dt.day\_name()

NYC\_df['dropoff\_day']=NYC\_df['dropoff\_datetime'].dt.day\_name()

NYC\_df.head()



#Number of Pickups and Dropoff on each day of the week

figure,ax=plt.subplots(nrows=2,ncols=1,figsize=(10,10))

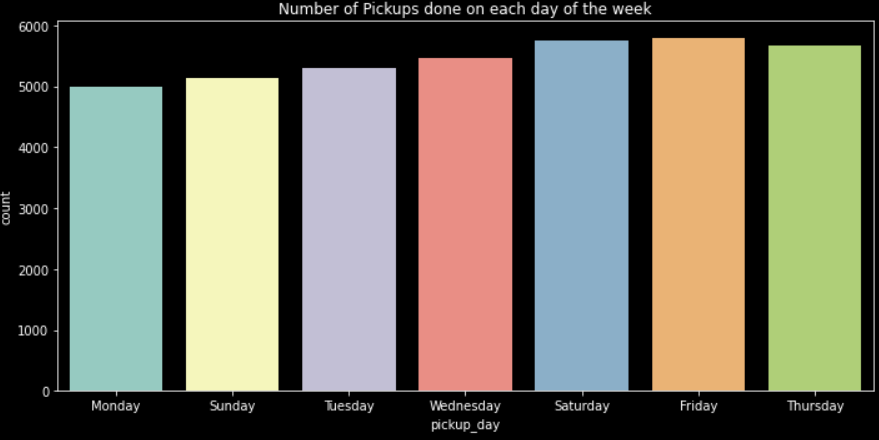
sns.countplot(x='pickup\_day',data=NYC\_df,ax=ax[0])

ax[0].set\_title('Number of Pickups done on each day of the week')

sns.countplot(x='dropoff\_day',data=NYC\_df,ax=ax[1])

ax[1].set\_title('Number of dropoffs done on each day of the week')

plt.tight\_layout()



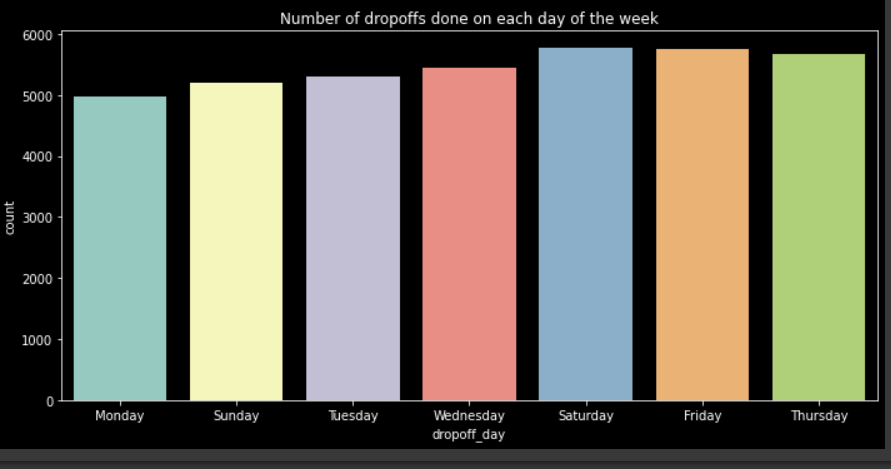


Fig.03

### Outlier Detection using IQR Method

IQR=Q3-Q1

lower\_limit\_outlier=Q1-1.5\*IQR

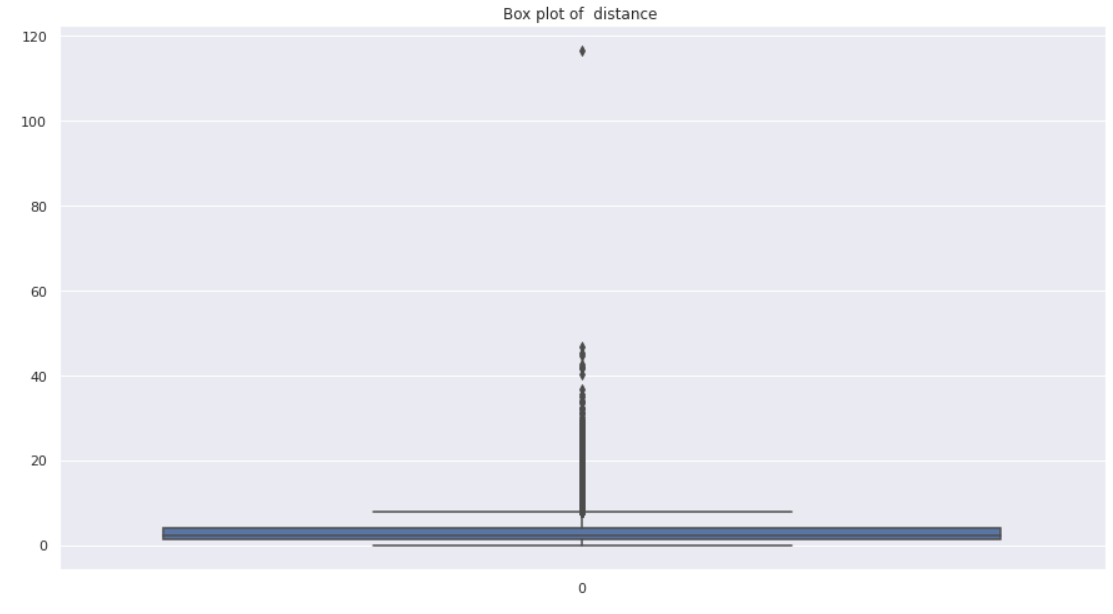
upper\_limit\_outlier=Q3+1.5\*IQR¶

so we have to take the values which is greater then lower limit outlier and less then upper limit outlier remove the outlier present in distance

plt.figure(figsize=(15,8))

plt.title("Box plot of distance ")

ax = sns.boxplot(data=NYC\_df['distance'], orient="v")



percentile\_q1 = np.percentile(NYC\_df['distance'],25)

print(percentile\_q1)

percentile\_q2 = np.percentile(NYC\_df['distance'],50)

print(percentile\_q2)

percentile\_q3 = np.percentile(NYC\_df['distance'],75)

print(percentile\_q3)

1.2303056255050369

2.0773105511108407

3.8432264798565123

iqr=percentile\_q3 - percentile\_q1

lower\_limit\_outlier=percentile\_q1-1.5\*iqr

upper\_limit\_outlier=percentile\_q3+1.5\*iqr

print("lower limit for outlier :",lower\_limit\_outlier)

print("Upper limit for outlier :",upper\_limit\_outlier)

lower limit for outlier : -2.689075656022177

Upper limit for outlier : 7.762607761383726

### Split Data

Lets split our data first before scaling the features

*#For Standarization apply z-score*

from scipy.stats import zscore

*#Train test split*

X = nyc\_df[features].apply(zscore)[:100000]

y=nyc\_df['trip\_duration\_hour'][:100000]

*# Importing train\_test\_split*

from sklearn.model\_selection import train\_test\_split

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y,test\_size=0.2,random\_state=42)

print(X\_train.shape,y\_train.shape)

print(X\_test.shape,y\_test.shape)

(25828, 18) (25828,)

(6458, 18) (6458,)

**Co-reallation**

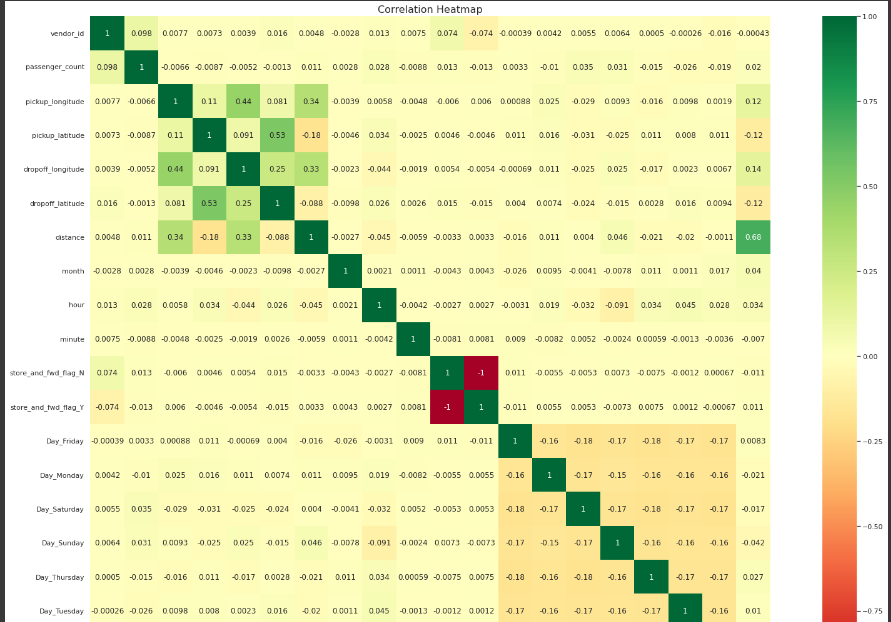
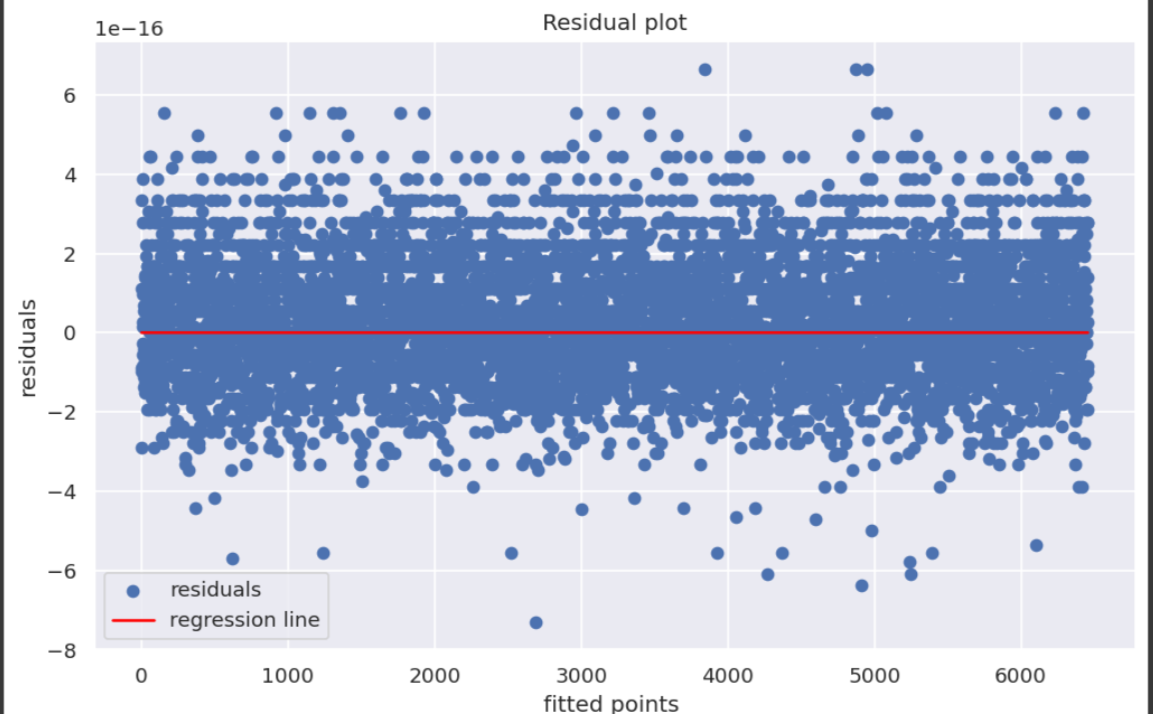


Fig 04

# **Linear Regression**

Linear regression analysis is used to predict the value of a variable based on the value of another variable. The variable you want to predict is called the dependent variable. The variable you are using to predict the other variable's value is called the independent variable.



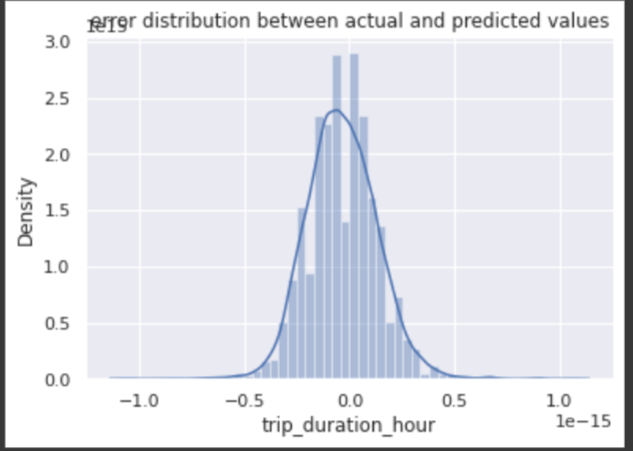
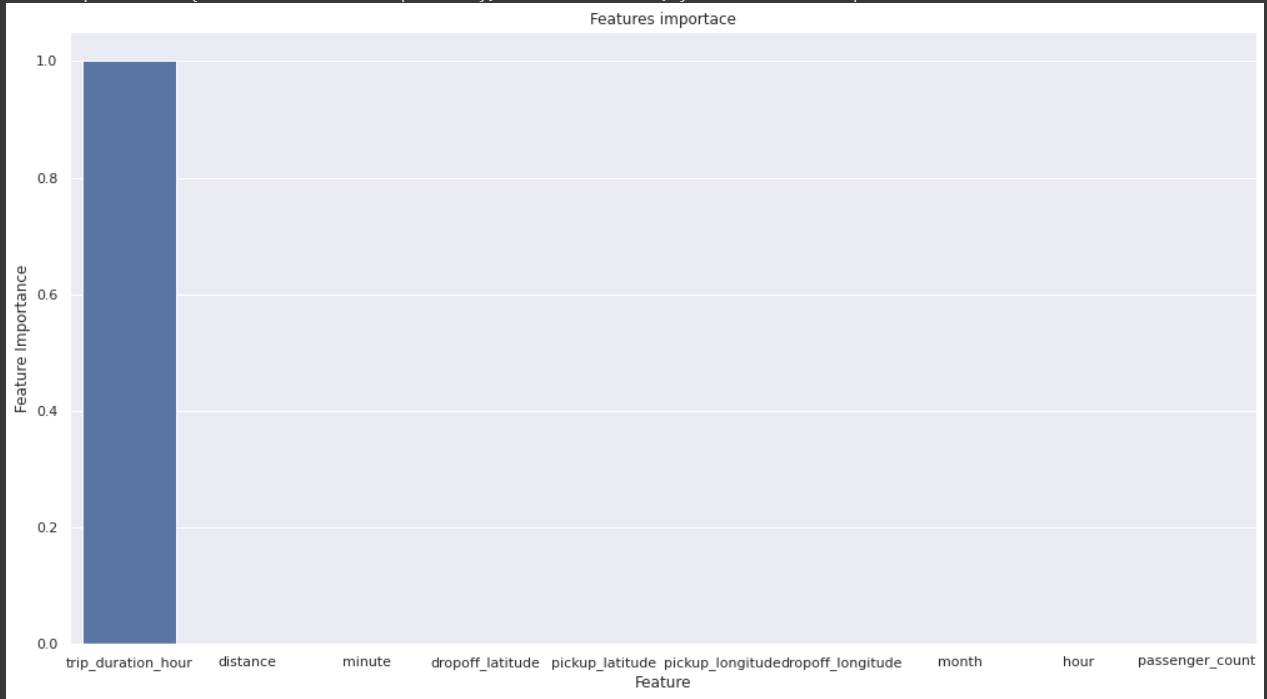


Fig. 05

### XGBoost

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way



### GradientBoosting

Gradient boosting is a machine learning technique used in regression and classification tasks, among others. It gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision trees.When a decision tree is the weak learner, the resulting algorithm is called gradient-boosted trees

*# Number of trees*

n\_estimators = [100,120]

*# Maximum depth of trees*

max\_depth = [5,8,10]

*# Minimum number of samples required to split a node*

min\_samples\_split = [50,80]

*# Minimum number of samples required at each leaf node*

min\_samples\_leaf = [40,50]

*# HYperparameter Grid*

param\_gb = {'n\_estimators' : n\_estimators,

'max\_depth' : max\_depth,

'min\_samples\_split' : min\_samples\_split,

'min\_samples\_leaf' : min\_samples\_leaf}

*# Create an instance of the GradientBoostingRegressor*

from sklearn.ensemble import GradientBoostingRegressor

gb\_model=GradientBoostingRegressor()

*# Grid search*

gb\_grid = GridSearchCV(estimator=gb\_model,

param\_grid = param\_gb,

cv = 3, verbose=2, scoring='r2')

gb\_grid.fit(X\_train,y\_train)

### Light GBM

LightGBM is a gradient boosting framework that uses tree based learning algorithms. It is designed to be distributed and efficient with the following advantages:

Faster training speed and higher efficiency.

Lower memory usage.

Better accuracy.

Support of parallel, distributed, and GPU learning.

Capable of handling large-scale data.

from lightgbm import LGBMRegressor

In [ ]:

*# Applying LightGBM*

n\_estimator=[5,10,20] *# No. of tree*

max\_depth=[5,7,9] *# max depth of tree*

min\_samples\_split=[40,50]

params={"n\_estimator":n\_estimator,"max\_depth":max\_depth,"min\_samples\_split":min\_samples\_split}

lgb=LGBMRegressor()

gs\_lgb=GridSearchCV(lgb,params,cv=3,verbose=2,scoring='r2')

gs\_lgb.fit(X\_train,y\_train)

print(gs\_lgb.best\_score\_)

print(gs\_lgb.best\_params\_)

gs\_lgb.best\_estimator\_

LGBMRegressor(max\_depth=5, min\_samples\_split=40, n\_estimator=5)

gs\_lgb\_opt\_model = gs\_lgb.best\_estimator\_

y\_preds\_lgb = gs\_lgb\_opt\_model.predict(X\_test)

y\_pred\_lgb\_train=gs\_lgb\_opt\_model.predict(X\_train)

*#Evaluation metrics for Test set*

EvaluationMetric(X\_test,y\_test,y\_preds\_lgb)

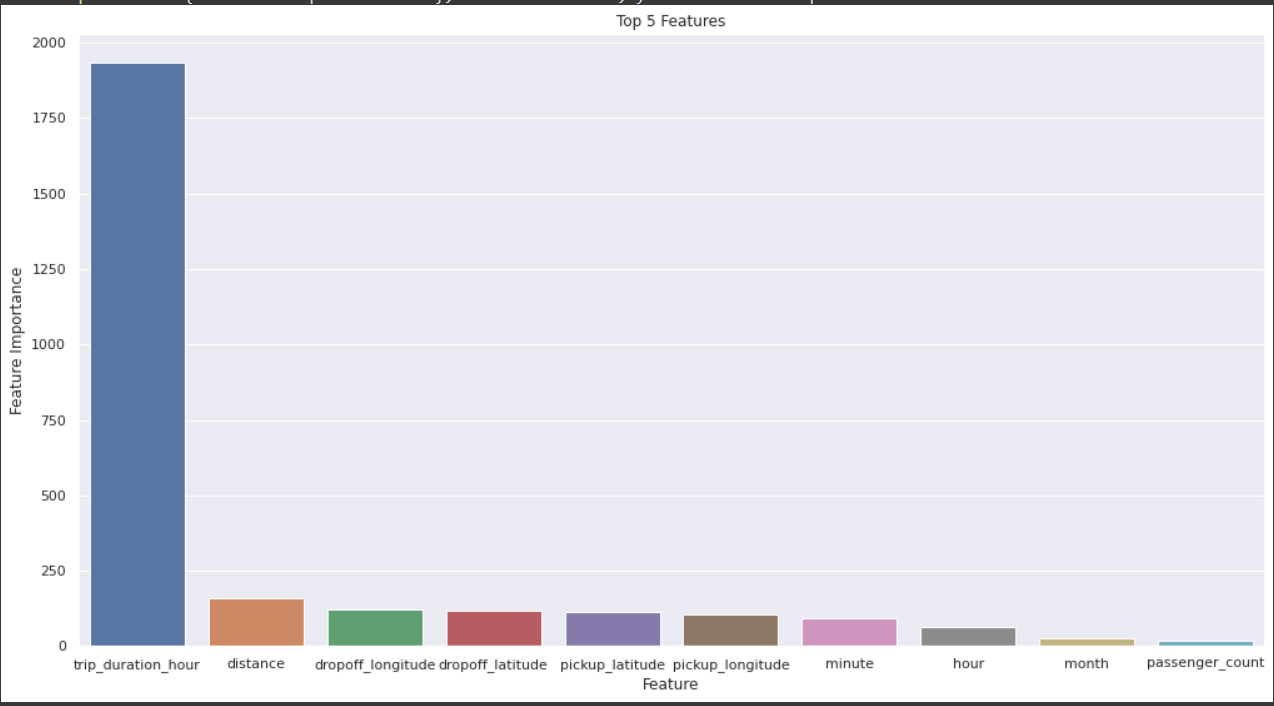
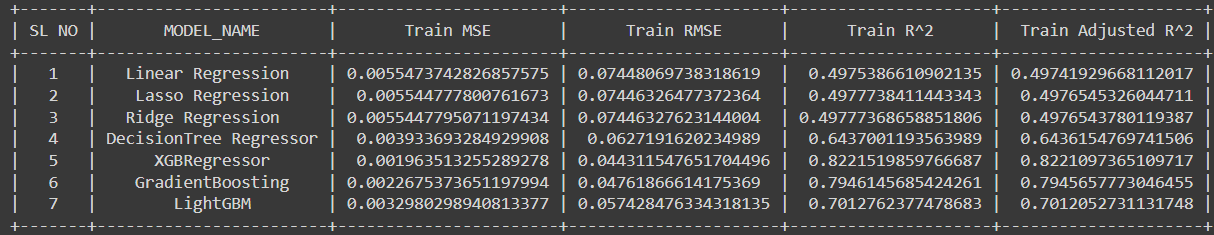
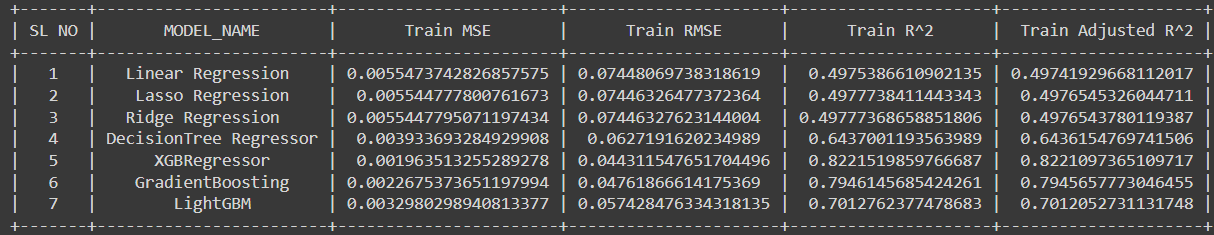


Fig 06





Hence we can conclude that Gradient boosting perform well.

## Observations:

We can observe that both the models shows somewhat similar learning rate but with visible differences in error rates.

Gradient boosting performed very well out pof all the models

XGBoost training curve on the other hand starts quite low and further improves with the increase in the training size and it too plateau towards the end.

Validation curve seems to show similar trend in both the models i.e. starts very high but improves with the training size with some differences in error rate i.e. XGBoost curve learning is quite fast and more accurate as compared to the RF one.

Both the models seems to suffer from high variance since the training curve error is very less in both the models.

The large gap at the end also indicates that the model suffers from quite a low bias i.e. overfitting the training data.

Also, both the model's still has potential to decrease and converge towards the training curve by the end.

**At this point, here are a few things we could do to improve our model:**

Add more training instances to improve validation curve in the XGBoost model. Increase the regularization for the learning algorithm. This should decrease the variance and increase the bias towards the validation curve. Reduce the numbers of features in the training data that we currently use. The algorithm will still fit the training data very well, but due to the decreased number of features, it will build less complex models. This should increase the bias and decrease the variance.

## End Notes

In this project we covered various aspects of the Machine learning development cycle. We observed that the data exploration and variable analysis is a very important aspect of the whole cycle and should be done for thorough understanding of the data. We also cleaned the data while exploring as there were some outliers which should be treated before feature engineering. Further we did feature engineering to filter and gather only the optimal features which are more significant and covered most of the variance in the dataset. Then finally we trained the models on the optimum featureset to get the results.

## Further Scope..

There's always a room for the improvement and a lot more to explore, and if this helped you in any way, I'd like to see One Upvote!. Also, please leave comments about any further improvements to this notebook!! Your feedback or any constructive criticism is highly appreciated.

A taxi company faces a common problem of efficiently assigning the cabs to passengers so that the service is smooth and hassle free. So, the topic for the supervised machine learning capstone project is NYC taxi-trip time prediction in this project our target variable is trip time predictionso our goal to predict when the cab will be free for the next trip.

Our first task is to prepare dataset for our machine learning models. After loading the dataset, we started with the Data Cleaning (it involves Nan value Checking and Duplicated value checking) after this we performed Exploratory Data Analysis by comparing our target variable that is trip duration with other independent variables and visualized it using seaborn and matlibplot library. This process helped us figuring out various aspects and relationships among the target and the independent variables. We will do certain steps like dropping unnecessary columns and do the one hot encoding for the required columns.

After data handling and performing EDA on it we get the important feature for our machine learning model then we fit our Machine learning models like Linear regression, XGBOOST, LightGBM the data. After applying the ML Model, we determine the key feature of the data set and perform cross-validation and hyper parameter tuning so as to find out the optimal parameter at which the error would be less for the training and testing dataset and the model performance is high and with the help of ML Evaluation matrices like R2 score, RMSE, MSE, Adjusted R2 score we decide that which machine learning model is the best fit for our dataset.

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.

After applying various algorithm, it is found that LightGBM perform the best in predict the trip duration for a particular taxi.

# **Conclusion:**

This paper presented a study on the topic of vehicle travel time estimation. Specifically, the paper tackled the problem of taxi trip prediction using a public dataset of taxi trips in New York City. First, a detailed analysis of the different features available for developing predictive models was performed. Next, classical machine learning models namely SVM, Random Forest, XGBoost, and MLP were trained using the extracted features. This step of the work provided both a predictive framework for taxi trip durations as well as insights into the proper feature-spaces that can be used for this purpose. Finally, fully connected deep neural networks were developed, which outperformed the classical machine learning models and predicted the travel durations with high accuracy. Future work will aim to develop more powerful models on larger datasets, attaining better representation of city traffic.

That's it! We reached the end of our exercise.

Starting with loading the data so far, we have done EDA, null values treatment, encoding of categorical columns, feature selection and then model building.

In all these models our accuracy revolves in the range of 70 to 74%.

And there is no such improvement in accuracy score even after hyper parameter tuning.

So, the accuracy of our best model is 73% which can be said to be good for this large dataset. This performance could be due to various reasons like no proper pattern of

Data, too much data, not enough relevant features.

# **References:**

* 1. Machine Learning Mastery
  2. Geeks for Geeks
  3. Analytics Vidhya

PROJECT DETAILS

|  |  |  |  |
| --- | --- | --- | --- |
| ***Student Details*** | | | |
| **Student Name** |  | | |
| Register Number | 7000936517 | Section / Roll No | 80 |
| Email Address | Naveenrai9878@gmail.com | | |
| Phone No: |  | |  |
|  | | | |
| Project Details NYC Taxi Trip Time Prediction(regression model) machine learning | | | |
| **Project Title** | NYC Taxi Trip Time Prediction | | |
| Project Duration | 2 months | Date of reporting | 04-07-2023 |
|  |  | | |
| ***Organization Details*** | | | |
| **Organization Name** |  | | |
| Full postal address with pin code | Bhopal 462023  Ashoka Garden , Bhopal | | |
| Website address |  | | |
|  |  | | |
| ***Supervisor Details*** | | | |
| **Supervisor Name** |  | | |
| Designation | Samatrix.io Data Analysis & Pro. Ved Lad | | |
| Full contact address with pin code | Bhopal | | |
| Email address |  | | |
| Phone No (M) |  | |  |
|  |  | | |
| *Internal Guide Details -* | | | |
| **Faculty Name** | **VED LAD** | | |
| Full contact address with pin code | - | | |
| Email address | - | | |
| Phone No (M) | - | | |