**SUPERVISED MACHINE LEARNING- REGRESSION**

**ON**

**NYC TAXI TRIP TIME PREDICTION**

****

**Submitted by:**

**Shreesha K, Data Science Trainee, Almabetter**

**E-mail:** [**shreeshasrisha@gmail.com**](mailto:shreeshasrisha@gmail.com)

**Abstract**

In the era of the ever-increasing population and number of vehicles, many Urban residents find it difficult to reach their destinations on time. Being one of the largest cities in the world, New York is also not an exception. There might be numerous factors upon which the travel duration depends. Hence, it is important to analyze the effect of various factors that do affect the travel duration, by which travelers can plan accordingly and drivers can also choose the routes with lesser congestion. Meanwhile, different other parameters like the season of the month, specific parts of the month, and weekends have their impact on the trip duration. We need to make proper observations based on all these factors to build an optimum Machine Learning model which can make accurate predictions. In this way, users of the service can decide how to minimize the effort along with the cost utilized for their travel.

The Dataset was provided by NYC Taxi and Limousine Commission. This dataset contains variables like vendor id, passenger count, pickup-drop-off latitudes, and longitudes along with several other variables.

Our prime motive is to analyze the data to get some insights about the data, to perform feature engineering to generate some new variables from the given data, and finally to build an ML model. We have used Linear Regression, Decision Tree, Lasso, and Ridge Regression as well as some boosting algorithms like XGBoost.

***KeyWords: Machine Learning, trip\_duration, Supervised Models***

1. **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). Our task is to build a predictive model, which can make accurate predictions of the Trip duration. It enables the customers to plan their trips and also, it helps the Taxi vendors to match the right services to the right customers.

The various features or columns in the dataset are (same information directly taken from the data):

* + 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

1. **Introduction:**

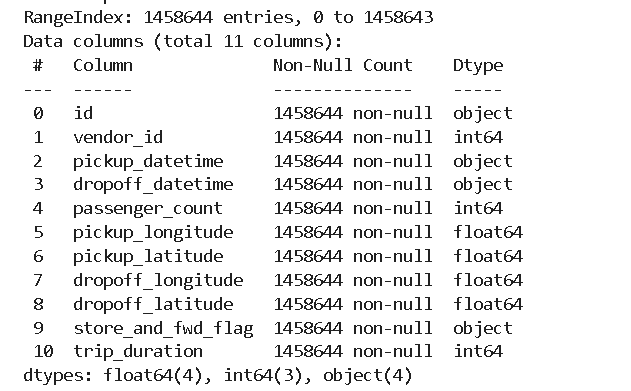
With the advancement of technology and the digital revolution, people often prefer to travel using taxis. New York, being an American city, has a large population that entirely depends upon taxis for their commute. Nearly 55% percent of the total population of the city does not own a personal vehicle, which increases the demand for taxis. As per the official source, 13,587 taxis are currently permitted. However, the time taken for commutation depends upon many factors.

Generally, the trip duration can be calculated by distance divided by the average speed. But there are other factors which can affect the time taken. Unfavorable weather conditions, occasional events or festivals, and peak hour traffic congestion may have an impact on the duration.

1. **Data Handling And Feature Engineering:**

**3.1. Data loading and Checking**

The data is published in 2016 by NYC Taxi and Limousine Commission and made available to the public via the Google clouds platform.



We loaded the data from the drive using pandas.read\_csv function. Our data had 1458664 rows and 11 Columns.

The dataset contains data of type object, int, and float.

**3.2. Checking for Null values and Duplicates**

As we checked, it was noted that there were no NULL values throughout. Hence it was not needed to clean them.

There were no duplicate entries.

The pickup\_datetime and dropoff\_datetime columns were given in the object format. But it would be better to handle them if they are in date time format. Therefore, we have changed them to the appropriate format.

* 1. **. EDA**

As the first step, we checked the relation between variables vendor id, (stored and fwd flag) and no of trips. It was found that the vendor with id 2 has taken more trips. There was a very less number of trips that were stored in the memory of the vehicle before the commencement of the trip. Then we analyzed the number of passengers in a trip. It was found that most people travel alone. By using the Great Circle method, we found out the distances. We plotted boxplots for numerical features and identified the outliers. By segregating the duration into different classes it was found that most of the trips were of duration between one min and one hour. By using Interquartile ranges and the concept of outliers we tried to remove outliers. The latitudes and longitudes outside the city limit were not considered for further analysis. We created some new features such as pickup hour, pickup minute, and pick up second to draw some more inferences. We found out the peak time of travel, the month with the maximum number of trips, the part of the day in which demand for the taxis is high, variation of trip duration along with weekdays and weekends.

**3.4. Handling categorical features**

It is mandatory to encode the categorical features into numerical values otherwise the scikit library won’t recognize them. By using dictionary mapping, we successfully converted those variables into the appropriate format.

**3.5. Feature Standardization**

In the dataset, there is a vast difference between the range of independent features. To get a better performing model we need to convert them into the same scale. For this purpose, we have used MinMaxScaler.

**3.6. Fitting Different Models**

We used many ML models to get better results. We used Linear Regression along with regularization methods Lasso and Ridge, Decision Trees, and Xgboost. To get the best parameters in each model, Cross-validation is performed. It ensures that we use the best hyperparameters in the model to get desired results.

1. **Algorithms:**

In this dataset, we have one dependent variable whose value shall be predicted and many independent input features. The dependent variable is continuous, so this is the case of a Supervised Regression Model. Hence, we have used some of the most commonly used Regression algorithms in this project.

**4.1. Linear Regression**

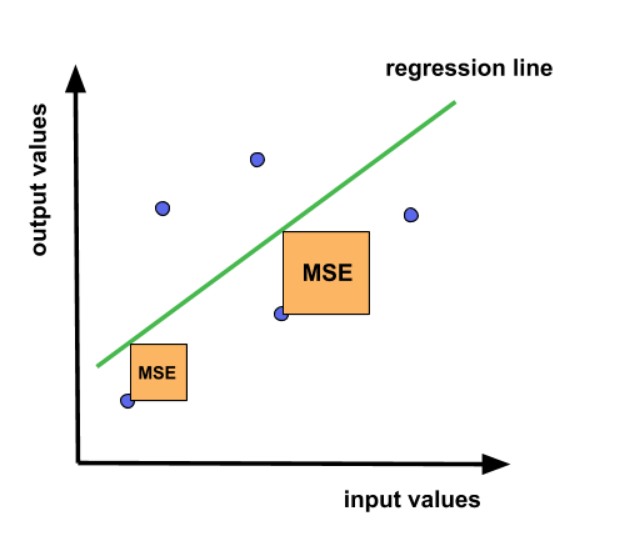
This is the algorithm which works on the assumption that the relationship between the input and output variables in linear. If y is the dependent variable and x1 , x2 ….., xn

are the independent input variables, the relationship is given by

**y= B0+B1x1+B2x2+…….. +Bnxn**

The curve obtained by this equation minimizes the sum of squared errors i.e.

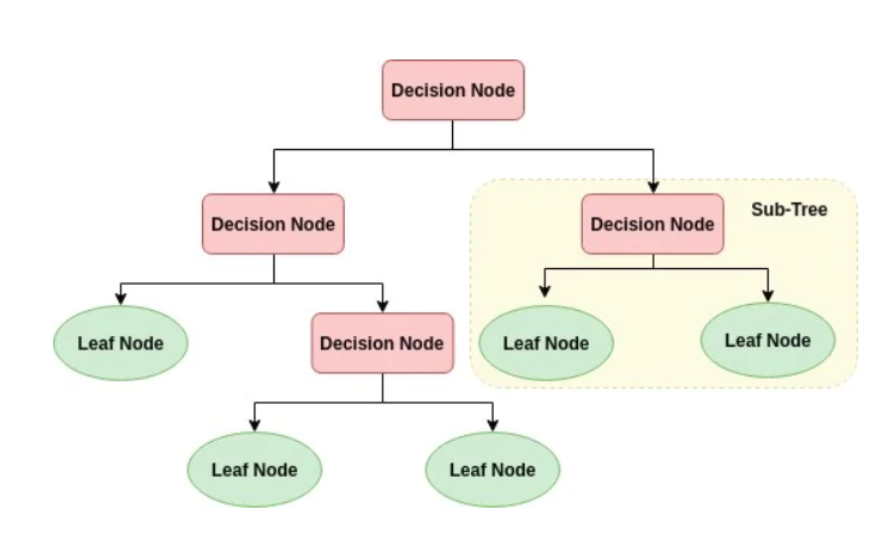
**Sum of (yactual-ypredicted)2**



To reduce the tendency of overfitting in this model, different regularization methods are used such as Lasso(L1) and Ridge(L2). These models often provide good results in comparison with linear regression.

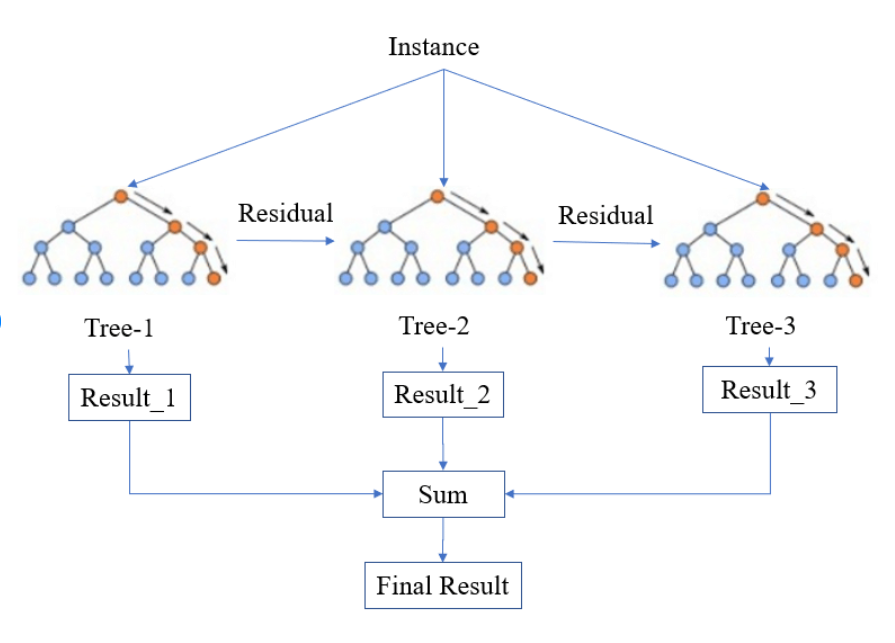
**4.2. Decision Tree**

It is an algorithm that makes numerous tree-like flowchart structures based on some common features and predicts the output by considering these common features. This model, if well-tuned, provides good results. Normally Decision Trees are assumed to be low biased, high variance models. By tuning hyperparameters appropriately we can eliminate this tendency.



**4.3. XGBoost**

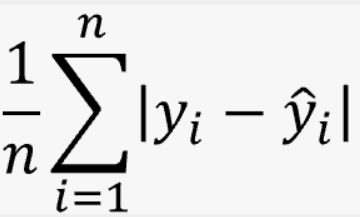
It stands for Extreme Gradient boosting. Here, several decision trees are trained sequentially. The weights are assigned to independent variables and decision trees are constructed. Weights of values predicted wrongly are increased and the next decision tree is built. It results in a reliable, low bias and low-variance model.



1. **Performance Parameters:**

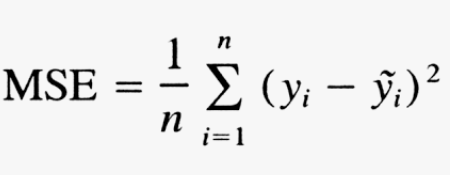
**5.1. Mean Absolute Error (MAE)**

It is the mean of the absolute errors, i.e., **|actual value- predicted value|.** It is given by the formula,



**5.2. Mean Squared Error (MSE)**

It is the average of square of the difference between predicted and actual values. It is one of the most widely used performance parameters in the industry. It can be expressed as



The problem with the MSE is that its unit is not same as that of the output feature. Hence another performance criterion is developed, known as RMSE.

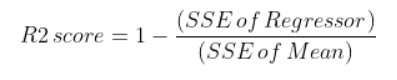
**5.3. Root Mean Squared Error (RMSE)**

It is the square Root of the MSE. One of the advantages of RMSE is that its value has the same unit as that of output variable, which helps in the interpretation of data.

**RMSE= √MSE**

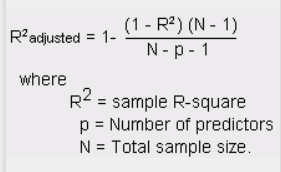
**5.4. R2 Score**

It indicates how much percentage deviation in the dependent variable can be expressed by the deviation in the input variables. It also explains the extent to which the model is better compared to the mean line.



**5.5. Adjusted R2 Score**

It imposes a penalty over the R2 score. Generally, R2 score tends to increase if the number of input parameters is increased. But Adjusted R2 score reduces this tendency to overestimate and I might be decreased if the addition of a new feature is irrelevant to the data. It is given by,



Its value is always lesser than or equal to R2 score. If its value is one, it indicates a perfect model. If it is zero, it implies that there is no relationship between input and output parameters.

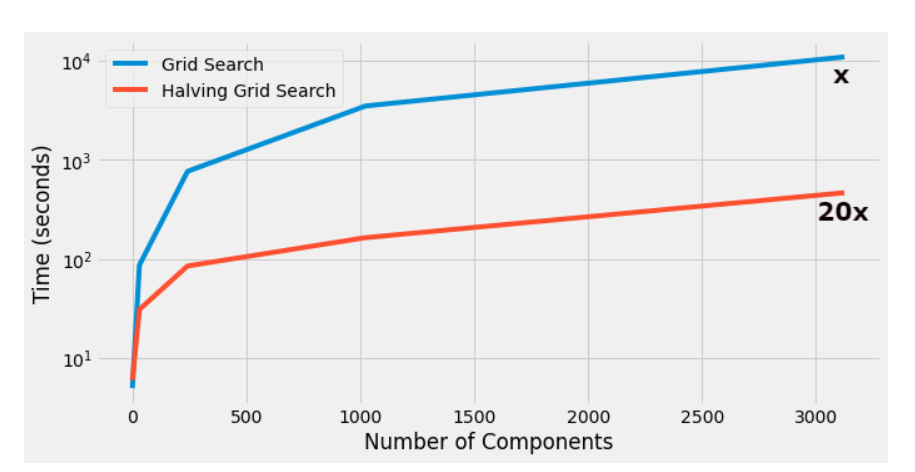
1. **Cross Validation and Hyper Parameter Tuning:**

Cross Validation is a method by which the model is trained by using the subset of the dataset and tested by using the complementary subset. It is performed to ensure that the model catches the correct pattern of the data and becomes robust to the random noises within the data.

Hyperparameters are those variables that affect the way by which an ML model gets trained. By adjusting these hyperparameters we can modify the performance, stability, and interpretation of the model. The commonly used methods for this purpose are GridSearchCV and RandomSearchCV. Being a large dataset, Gridsearch Cross validation takes too much time. So sometimes we have used HalvingGridsearchCV, which is a little bit inaccurate, but multiple times faster than the GridSearchCV. GridsearchCV evaluates the performance of the model for all the combinations of different hyperparameters. On the other hand, RandomSearchCV chooses these combinations randomly, and obtaining the best model will be a hit or miss.



For larger datasets, GridsearchCV results in a lot of possible combinations of parameters. It takes a lot of time. An approach towards reducing the time is by using HalvingGridsearchCV. It follows a successive halving approach. It first trains t a subset of the data for all possible combinations. Then it finds the best-performing combinations. A larger subset of the data is trained using these top combinations. As time progresses, the parameter counts decreases, and the size of the training data increases. Since it follows a successive halving approach, its time complexity is too less compared to the conventional methods.



1. **Conclusion:**

We have loaded the data, treated the NULL values, performed feature engineering, performed exploratory data analysis, and built ML models. After the evaluation, we observed that the adjusted R2 score ranges between 0.41 to 0.83. Some of the models didn’t perform well even after hyperparameter tuning. The regularization methods applied to linear regression didn’t provide significant improvement. XGBoost is the best-performing model among all and it is recommended. Its accuracy is almost nearer to 82% for train data and 74% for test data, which is good for this large dataset.

1. **References:**

* GeeksForGeeks
* AnalyticsVidya
* TowardsDataScience