NYC Taxi Trip Time Prediction

Sanjay Yadav
Data science trainee,
AlmaBetter, Bangalore

Abstract:
New York City taxi rides form the core of the traffic in the city of New York. The many rides taken every day by New Yorkers in the busy city can give us a great idea of traffic times, road blockages, and so on. Predicting the duration of a taxi trip is very important since a user would always like to know precisely how much time it would require him to travel from one place to another.
Given the rising popularity of app-based taxi usage through common vendors like Ola and Uber, competitive pricing has to be offered to ensure users choose them. Prediction of duration and price of trips can help users to plan their trips properly, thus keeping potential margins for traffic congestions. It can also help drivers to determine the correct route which in-turn will take lesser time as accordingly.
Moreover, the transparency about trip duration will help to attract users at times when popular taxi app-based vendor services compare trip duration for a destination.

- ***Keywords: regression, machine learning, trip duration, location***

Problem Statement :

Our task is to build a model that predicts the total ride duration of taxi trips in New York City. 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. Based on individual trip attributes, you should predict the duration of each trip in the test set.

NYC Taxi Data.csv - the training set (contains 1458644 trip records)

Data fields :

- 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

Target Variable :

- trip_duration - duration of the trip in second

Introduction :

New York City is one of the highly advanced cities of the world with extensive use of taxi services. Along with a vast population, the requirement of commonly available transportation serves the common purpose as it provides a very large transportation system. New York facilitates one of the largest subway systems in the world and comprises various green and yellow cabs which approximately count around 14,000 taxis.

Of all people who commute to work in New York City, 39% use the subway, 23% drive alone, 11% take the bus, 9% walk to work, 7% travel by commuter rail, 4% carpool, 1.6% use a taxi, 1.1% ride their bicycle to work, and 0.4% travel by ferry.

Most of the population of New York depends upon public transport, and it has been estimated that 54 percent of the people do not own a car or a personal vehicle. As a matter of fact, it accounts for almost 200 million taxi trips per year

How Taxi Meters Work :

Taximeters determine the distance traveled and the elapsed time of your taxi ride. The apparatus uses electric pulses to calculate the distance. A sensor attached to the car’s transmission system sends pulses to the taximeter each time the taxi travels a certain distance.

As soon as the driver starts his meter, it records the time at which it was activated. Once the driver stops the meter, the computer calculates the time in between.

Steps involved:

- **Importing Libraries and Loading Data**

  To proceed with the problem, first we will import all the libraries we are going to use in our project and then load our dataset that is given to us in .csv file into a data frame. Mount the drive and load the csv file into a data frame.

- **Exploratory Data Analysis**

  After loading the dataset I did preliminary univariate and bivariate analysis of independent and target variable.

- **Feature Decomposition**

  New distance feature was added after calculating distance between each trip using longitude and latitude features. Day, month and hour columns were added from analyzing datetime feature.

- **Null values Treatment**
  There were no null values in our dataset.

- **Removing Outliers**

  We have done outlier treatment on variables like pickup and drop-off coordinates, duration, passenger count and trip duration. This was done by removing values greater than three standard values. We have done outlier treatment to prevent high errors that were influenced by outliers.

- **Encoding of categorical columns**

  We used pandas dummies for replacing the values of categorical variables . This was done to provide more data set to our model while also keeping the relationship of variables with trip duration into consideration.

- **Feature Selection**

  Feature selection is the process of identifying and selecting a subset of input variables that are most relevant to the target variable.

- **Splitting Data**

  Splitting the data into training and test sets can be done either in the beginning of the project or after EDA , the former method ensures no data leakage but since we are not deploying our model on cloud and using only provided dataset we can do it before model building.

- **Scaling Features**

  We have to scale the selected features using Standard Scalar or MinMax Scalar to bring the distribution within the same range as other features.

Building different models

For modeling we tried following regression algorithms :

- Multiple Regression
- Ridge Regressor
- Lasso Regressor
- Decision Trees Regressor
- XGB Regressor
- LGBM Regressor
- Catboost Regressor

4.1. Algorithms:

Linear Regression:

Linear regression is a type of model where the relationship between an independent variable and a dependent variable is assumed to be linear. The estimate of variable "y" is obtained from an equation, y'-y_bar = byx(x-x_bar)……(1) and estimate of variable "x" is obtained through the equation x'-x_bar = bxy(y-y_bar)…..(2). The graphical representation of linear equations on (1) & (2) is known as Regression lines. These lines are obtained through the Method of Least Squares.

The Linear Regression Model we have used is Multiple Linear Regression - A linear regression model with more than one independent variable and one dependent variable.

Assumptions of Linear Regression are :

- Heteroscedasticity is absent
- Linear Relationships exist between the variables.
- Independent Sample observations.
- No multicollinearity & auto-correlation
- Independent Sample observations.

Pros

- Simple method
- Good interpretation
- Easy to implement

Cons

- Assumes linear relationship between dependent and independent variables, which is incorrect in most cases
- Sensitive to outliers
- If the number of observations are less, it leads to over fitting, it starts considering noise.

Ridge Regression:

To understand Ridge Regression we first need to get through the concept of Regularization.
Regularization: There are two types of Regularization, L1 regularization & L2 regularization.

L1 regularization adds an L1 penalty equal to the value of coefficients to restrict the size of coefficients, which leads to the removal of some coefficients. On the other hand, L2 regularization adds a penalty L2 which is equal to the square of coefficients.
Using the above method Regularization solves the problem of a scenario where the model performs well on training data but underperforms on validation data.

A few key points about Ridge Regression:

- ★ The assumptions of this regression are the same as least squared regression except normality is not to be assumed.
- ★ It shrinks the value of coefficients but doesn't reach zero, which suggests no feature selection feature.

Pros

- Trades variance for bias (i.e. in presence of collinearity, it is worth having biased results, in order to lower the variance.)
- Prevents overfitting

Cons

- Increases bias
- Need to select perfect alpha (hyper parameter)
- Model interpret-ability is low

<u>Lasso Regression:</u>

LASSO (Least Absolute Shrinkage and Selection Operator) is a regression technique that was introduced first in geophysics. The term "Lasso" was coined by Professor Robert Tibshirani. Just like Ridge Regression, it uses regularization to estimate the results. Plus it also uses variable selection to make the model more efficient.
Pros

- Select features, by shrinking co-efficient towards zero.
- Avoids overfitting

Cons

- Selected features will be highly biased.
- For n<<p (n-number of data points, p-number of features), LASSO selects at most n features.
- LASSO will select only one feature from a group of correlated features, the selection is arbitrary in nature.
- For different bootstrapped data, the feature selected can be very different.

- Prediction performance is worse than Ridge regression.

Decision Tree :

A decision tree is a tree-like graph with nodes representing the place where we pick an attribute and ask a question; edges represent the answers the to the question; and the leaves represent the actual output or class label. They are used in non-linear decision making with simple linear decision surfaces.

Decision trees classify the examples by sorting them down the tree from the root to some leaf node, with the leaf node providing the classification to the example. Each node in the tree acts as a test case for some attribute, and each edge descending from that node corresponds to one of the possible answers to the test case. This process is recursive in nature and is repeated for every subtree rooted at the new nodes.

Pros

- Does not require standardization and normalization.
- Easy to implement
- Less data preparation work
- Missing values has no impact

Cons

- Doesn't work for smooth boundaries
- Doesn't work when variables are uncorrelated
- Due to greedy strategy, it has high variance
- Higher time to train the model
- Can become complex

XGB Regressor :

Gradient boosting refers to a class of ensemble machine learning algorithms that can be used for classification or regression predictive modeling problems.

Ensembles are constructed from decision tree models. Trees are added one at a time to the ensemble and fit to correct the prediction errors made by prior models. This is a type of ensemble machine learning model referred to as boosting.

Models are fit using any arbitrary differentiable loss function and gradient descent optimization algorithm. This gives the technique its name, "gradient boosting," as the loss gradient is minimized as the model is fit.

Some important features of XGBoost are:

Parallelization: The model is implemented to train with multiple CPU cores.
Regularization: XGBoost includes different regularization penalties to avoid overfitting. Penalty regularizations produce successful training so the model can generalize adequately.
Non-linearity: XGBoost can detect and learn from non-linear data patterns.
Cross-validation: Built-in and comes out-of-the-box.
Scalability: XGBoost can run distributed thanks to distributed servers and clusters like Hadoop and Spark, so you can process enormous amounts of data. It's also available for many programming languages like C++, JAVA, Python, and Julia.
Advantages:

- XGB consists of a number of hyper-parameters that can be tuned — a primary advantage over gradient boosting machines.
- XGBoost has an in-built capability to handle missing values.
- It provides various intuitive features, such as parallelization, distributed computing, cache optimization, and more.

Disadvantages:

- Like any other boosting method, XGB is sensitive to outliers.
- Unlike LightGBM, in XGB, one has to manually create dummy variable/ label encoding for categorical features before feeding them into the models.

<u>LGBM Regressor :</u>

Light GBM is a fast, distributed, high-performance gradient boosting framework based on a decision tree algorithm, used for ranking, classification and many other machine learning tasks.
Since it is based on decision tree algorithms, it splits the tree leaf wise with the best fit whereas other boosting algorithms split the tree depth wise or level wise rather than leaf-wise. 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. Also, it is surprisingly very fast, hence the word 'Light'.
Advantages of Light GBM

- Faster training speed and higher efficiency: Light GBM uses histogram based algorithm i.e. it buckets continuous feature values into discrete bins which fasten the training procedure.

- Lower memory usage: Replaces continuous values to discrete bins which result in lower memory usage.
- Better accuracy than any other boosting algorithm: It produces much more complex trees by following leaf-wise split approach rather than a level-wise approach which is the main factor in achieving higher accuracy. However, it can sometimes lead to overfitting which can be avoided by setting the max_depth parameter.
- Compatibility with Large Datasets: It is capable of performing equally good with large datasets with a significant reduction in training time as compared to XGBOOST.
- Parallel learning supported.

Catboost Regressor :

CatBoost is a recently open-source machine learning algorithm from Yandex. It can easily integrate with deep learning frameworks like Google's TensorFlow and Apple's Core ML. It can work with diverse data types to help solve a wide range of problems that businesses face today. To top it up, it provides best-in-class accuracy.

It is especially powerful in two ways:

It yields state-of-the-art results without extensive data training typically required by other machine learning methods, and

Provides powerful out-of-the-box support for the more descriptive data formats that accompany many business problems.

"CatBoost" name comes from two words "Category" and "Boosting".

As discussed, the library works well with multiple Categories of data, such as audio, text, image including historical data.

"Boost" comes from the gradient boosting machine learning algorithm as this library is based on a gradient boosting library. Gradient boosting is a powerful machine learning algorithm that is widely applied to multiple types of business challenges like fraud detection, recommendation items, forecasting and it performs well also. It can also return very good results with relatively less data, unlike DL models that need to learn from a massive amount of data.

Advantages of CatBoost Library

- Performance: CatBoost provides state of the art results and it is competitive with any leading machine learning algorithm on the performance front.
- Handling Categorical features automatically: We can use CatBoost without any explicit pre-processing to convert categories into numbers. CatBoost converts categorical values into numbers using various statistics on combinations of categorical features and combinations of categorical and numerical features. You can read more about it here.
- Robust: It reduces the need for extensive hyper-parameter tuning and lowers the chances of overfitting also which leads to more generalized models.Although,
  CatBoost has multiple parameters to tune and it contains parameters like the number of trees, learning rate, regularization, tree depth, fold size, bagging temperature and others. You can read about all these parameters here.
- Easy-to-use: You can use CatBoost from the command line, using an user-friendly API for both Python and R.

4.2. Model Performance Metrics:

- **Mean Squared Error(MSE)**
  MSE is a most used and very simple metric with a little bit of change in mean absolute error. Mean squared error states that finding the squared difference between actual and predicted value.
  So, above we are finding the absolute difference and here we are finding the squared difference.
  What actually the MSE represents? It represents the squared distance between actual and predicted values. We perform squared to avoid the cancellation of negative terms and it is the benefit of MSE.
  Advantages of MSE
  The graph of MSE is differentiable, so you can easily use it as a loss function.
  Disadvantages of MSE
  The value you get after calculating MSE is a squared unit of output. for example, the output variable is in meter(m) then after

calculating MSE the output we get is in meter squared.
If you have outliers in the dataset then it penalizes the outliers most and the calculated MSE is bigger. So, in short, It is not Robust to outliers which were an advantage in MAE.

$$MSE = \frac{1}{n}\Sigma\underbrace{\left(y - \widehat{y}\right)}_{\text{The square of the difference between actual and predicted}}^{2}$$

- **Root Mean Squared Error(RMSE)**
  As RMSE is clear by the name itself, that it is a simple square root of mean squared error.
  Advantages of RMSE
  The output value you get is in the same unit as the required output variable which makes interpretation of loss easy.
  Disadvantages of RMSE
  It is not that robust to outliers as compared to MAE.

- **R Squared (R2)**
  R2 score is a metric that tells the performance of your model, not the loss in an absolute sense that how many wells did your model perform.
  In contrast, MAE and MSE depend on the context as we have seen whereas the R2 score is independent of context.
  So, with help of R squared we have a baseline model to compare a model which none of the other metrics provides. The same we have in classification problems which we call a threshold which is fixed at 0.5. So basically R2 squared calculates how must regression line is better than a mean line.
  Hence, R2 squared is also known as Coefficient of Determination or sometimes also known as Goodness of fit.

$$\textbf{R2 Squared} = 1 - \frac{SSr}{SSm}$$

SSr = Squared sum error of regression line

SSm = Squared sum error of mean line

- **Adjusted R Squared**

  The disadvantage of the R2 score is while adding new features in data the R2 score starts increasing or remains constant but it never decreases because It assumes that while adding more data variance of data increases.
  But the problem is when we add an irrelevant feature in the dataset then at that time R2

sometimes starts increasing which is incorrect.

Hence, To control this situation Adjusted R Squared came into existence.

Now as K increases by adding some features so the denominator will decrease, n-1 will remain constant. R2 score will remain constant or will increase slightly so the complete answer will increase and when we subtract this from one then the resultant score will decrease. so this is the case when we add an irrelevant feature in the dataset.

And if we add a relevant feature then the R2 score will increase and 1-R2 will decrease heavily and the denominator will also decrease so the complete term decreases, and on subtracting from one the score increases.

$$R_a^2 = 1 - \left[\left(\frac{n-1}{n-k-1}\right) \times (1 - R^2)\right]$$

where:

n	= number of observations
k	= number of independent variables
R_a^2	= adjusted R^2

4.3. 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.

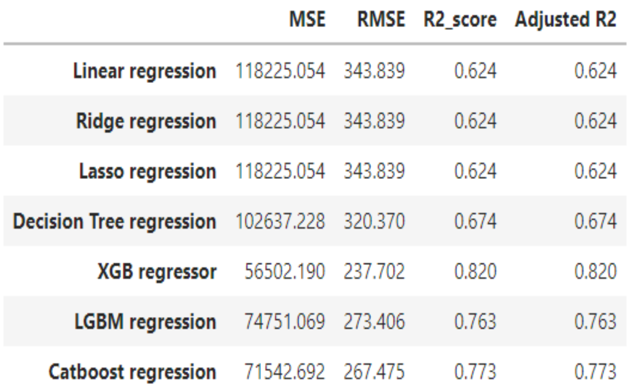
We used Grid Search CV, Randomized Search CV and Bayesian Optimization for hyperparameter tuning. This also results in cross validation and in our case we divided the dataset into different folds. The best performance improvement among the three was by Bayesian Optimization.

- **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 common hyperparameters which we extracted were n_estimators, max_depth, verbose=2 and cv = KFold.

- **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

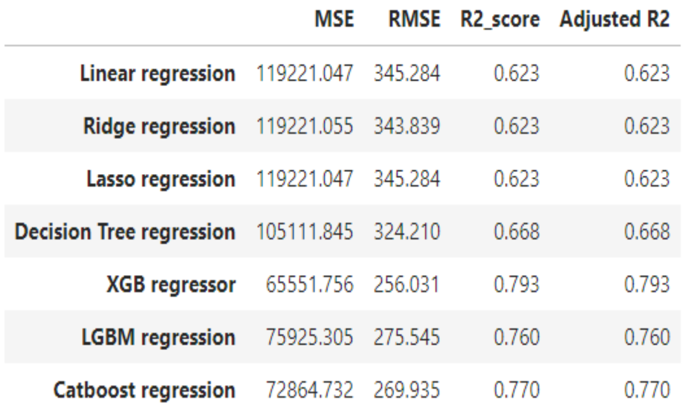
5. Conclusion:

Training model :



	MSE	RMSE	R2_score	Adjusted R2
Linear regression	118225.054	343.839	0.624	0.624
Ridge regression	118225.054	343.839	0.624	0.624
Lasso regression	118225.054	343.839	0.624	0.624
Decision Tree regression	102637.228	320.370	0.674	0.674
XGB regressor	56502.190	237.702	0.820	0.820
LGBM regression	74751.069	273.406	0.763	0.763
Catboost regression	71542.692	267.475	0.773	0.773

Test model :



	MSE	RMSE	R2_score	Adjusted R2
Linear regression	119221.047	345.284	0.623	0.623
Ridge regression	119221.055	343.839	0.623	0.623
Lasso regression	119221.047	345.284	0.623	0.623
Decision Tree regression	105111.845	324.210	0.668	0.668
XGB regressor	65551.756	256.031	0.793	0.793
LGBM regression	75925.305	275.545	0.760	0.760
Catboost regression	72864.732	269.935	0.770	0.770

We can conclude that boosting models are performing much better for given dataset and XGBoost is the best performing model with an adjusted R2 score of 82% on training data and 79.3% on test data.

References :

towardsdatascience.com
machine learning mastery
analytics vidhya