NYC Taxi Trip Time Prediction

Sandesh Gaikwad
Data Science Trainee,
AlmaBetter, Bangalore

Abstract:

Estimated travel time is a valuable metric in the traveling domain. In this project, we use a machine-learning algorithm to predict the taxi ride duration in New York City. For model training, We use the historical trip data released by the NYC Taxi and Limousine Commission which includes pickup time, geo-coordinates, number of passengers, and several other variables.

We tried multiple regression techniques such as Linear regression, Regularised linear regression, and Tree-based regressor approaches to find the best model.

Keywords: machine learning, regression, predictive modeling, trip time prediction

Problem Statement:

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.

The dataset is based on the 2016 NYC Yellow Cab trip record data made available in Big Query on the 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, we need to predict the duration of each trip in the test set.

- **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 it 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

Steps:

● Exploratory Data Analysis:

After loading the dataset we performed this method by comparing our target variable that is trip_duration with other independent variables. This process helped us figure out various aspects and relationships between the target and the independent variables. It gave us a better idea of which feature behaves in which manner compared to the target variable. Also, We plotted the box plots to know which features are consisting the outliers.

● Null values handling:

We checked and didn't find any Null values in our dataset.

● Encoding of categorical columns:

We used One Hot Encoding to produce binary integers of 0 and 1 to encode our categorical features because categorical features that are in string format cannot be understood by the machine and needs to be converted to the numerical format.

● Feature Engineering

Our main motive through this step was to scale our data into a uniform format that would allow us to utilize the data in a better way while performing fitting and applying different algorithms to it. We used normalization methods to get data into Gaussian distribution and also calculated direction and duration parameters

● Fitting different models

For modeling, we tried various classification algorithms like

1. Linear Regression
2. Lasso Regression
3. Ridge Regression
4. Decision Tree Regressor
5. Random Forest Regressor
6. xgboost Regreesor

● Hyperparameters tuning

Tuning the hyperparameters of respective algorithms is necessary for getting better performance and to avoid overfitting in the case of tree-based models like Decision Tree and Random Forest Regressors.

Algorithms:

1. Linear Regression:

Linear regression is a linear model, the model that assumes a linear relationship between the input variables (x) and the single output variable (y). More specifically, that y can be calculated from a linear combination of the input variables (x). When there is a single input variable (x), the method is referred to as simple linear regression. When there are multiple input variables, literature from statistics often refers to the method as multiple linear regression.

Different techniques can be used to prepare or train the linear regression equation from data, the most common of which is called Ordinary Least Squares. It is common to therefore refer to a model prepared this way as Ordinary Least Squares Linear Regression or just Least Squares Regression

2. Lasso Regression

Lasso Regression is a popular type of regularized linear regression that includes an L1 penalty. This has the effect of shrinking the coefficients for those input

variables that do not contribute much to the prediction task. This penalty allows some coefficient values to go to the value of zero, allowing input variables to be effectively removed from the model, and providing a type of automatic feature selection.

3. Ridge Regression:

It is a popular type of regularized linear regression that includes an L2 penalty. This has the effect of shrinking the coefficients for those input variables that do not contribute much to the prediction task.

4. Decision Tree:

It is a decision-making tool that uses a flowchart-like tree structure or is a model of decisions and all of their possible results, including outcomes, input costs, and Utility. Decision-tree algorithm falls under the category of supervised learning algorithms. It works for both continuous as well as categorical
output variables.

5. Random Forest Regressor:

Random Forest is a bagging type of Decision Tree Algorithm that creates a number of decision trees from a randomly selected subset of the training set, collects the labels from these subsets, and then averages the final prediction depending on the number of times a label has been predicted out of all the predictions.

6. xgboost 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, much like a neural network.

Evaluation of Performace

The model can be evaluated by various metrics such as

- **R-squared (R2)** -

It is the proportion of variation in the outcome that is explained by the predictor variables. In multiple regression models, R2 corresponds to the squared correlation between the observed outcome values and the predicted values by the model. The Higher the R- squared, the better the model.

- **Mean Absolute Error (MAE)** :

It calculates the average difference between the calculated values and actual values. It is also known as scale-dependent accuracy as calculates errors in observations taken on the same scale.

- **Mean Square Error (MSE):** -

It assesses the average squared difference between the observed and predicted values. When a model has no error, the MSE equals zero. As model error increases, its value increases. The mean squared error is also known as the mean squared deviation (MSD).

Results from different models

Model Name	Evaluation Metrics		
	R-Squared	RMSE	MAE
Linear Regression	0.55	372	241
Lasso Regression	0.55	372	241
Ridge Regression	0.55	372	241
Decision Tree	0.61	346	230
Random Forest	0.6	348	232
Xgboost	0.62	341	224

Conclusions:

- Linear, Lasso, and Ridge regressions are giving similar results.
- In the Decision Tree regressor, the results are slightly improved.
- Out of all tried models, Random Xgboost Regressor is giving the best result.

Future challenges:

- Distance and time of day are not sufficient to predict the time of ride it requires a thorough understanding of traffic and that the actual path of the ride is important.
- Deep Neural Networks can improve this performance.

References:

- Kaggle
- Geeks for Geeks
- Medium blogs