ISSN (ONLINE): **2454-9762**
ISSN (PRINT): **2454-9762**
***Available online at*** *www.ijarmate.com*

IJARMATE
Your ulti-MATE Research Paper !!!

***International Journal of Advanced Research in Management, Architecture, Technology and Engineering (IJARMATE)***
***Vol. 7, Issue 5, May 2021***

# Predicting Automobile Trip Duration using Machine Learning

Tanvi Baweja
*Dept. of Information Technology Engineering*
*Amity School of engineering and Technology, Guru Gobind Singh Indraprastha University*
New Delhi, India
tanvibaweja@gmail.com

***Abstract*—*Predicting automobile trip duration for future trips is a problem as real time traffic data is not available. However, an estimate can be predicted using the information available at hand before the start of the trip. The objectives of this paper are (a) To predict automobile trip duration using such factors like starting location, destination, date and time using various machine learning models and (b) To analyze and compare the performance of these models. Six machine learning models (i) Linear models- OLS, Ridge, Lasso and Elastic Net (ii) Random Forest (iii) Deep Neural Network have been used to predict the trip duration. Three performance metrics- $R^2$ Score, MSE and MAE have been used for comparison of these models. Deep Neural Network gives the lowest MSE, followed by OLS model.***

***Keywords*—*Machine Learning, Regression, Random Forest, Deep Neural Network, Trip Duration Prediction, Linear Models***

## I. Introduction

One can easily check for the real time estimated time of arrival for a trip we are about to take in the moment. The real problem exists when we want an estimate of duration of a trip we are planning to take in the future as no real time traffic updates can be made available. An approach to this problem could be to treat this problem as a regression problem and to use machine learning to predict trip duration, using trip information available to us before the beginning of the trip. In 2015, a feature for predictive estimates of future transit time was introduced in Google Maps API which lets developers estimate the travel time by adding the *departure_time* parameter [1]. The purpose of this paper is similar- to predict travel time using starting location, destination, date and time of the trip, and to compare the performance of different machine learning models designed for the same problem. Required and relevant features can be extracted using these four variables which could be treated as input to machine learning models. In [2], authors have used Linear Regression with model selection and Random Forest to work on a similar problem. In this paper, we extend the study further and compare the performance of (i)Linear models - OLS, Ridge, Lasso, Elastic Net, (ii) Random Forest and (iii) Deep Neural Network (DNN) using (i) R-Squared($R^2$) Score, (ii)Mean Squared Error (MSE) and (iii) Mean Absolute Error (MAE). The input features and the results of this paper are different from the work done by authors in [2].

While distance is a major factor in determining the trip duration; date and time can also play an important role in determining the trip duration. Information such as month, day of the week and time of the day can be extracted using date and time. Commuters might experience traffic congestion during peak hours. Traffic flow and road congestion might also vary during months of extreme weather conditions. Markou [3] analysed demand fluctuations in traffic networks and disruptive events scenarios like extreme weather conditions, public holidays, religious festivities and parades were correlated to these fluctuations. Information about distance, month, day of the week and time of the day can be useful to predict the trip duration in advance.

All models have been implemented using scikit-learn 0.24.0 [4] and Keras with their default parameters unless stated otherwise.

## II. Data

For this study, the database is based on 2016 NYC Yellow Cab trip record data provided by Kaggle for the Playground Prediction Challenge - New York City Taxi Trip Duration, which contains trip records from the month of January to September. The original dataset contains longitude and latitude of pickup and dropoff locations, time and date of pickup and dropoff and other variables (which are not relevant to this study). Data processing was done to extract separate features like pickup month, pickup day, pickup timezone, dropoff month, dropoff day, dropoff timezone. The timezones were defined as morning, midday, evening and late night. Furthermore, trip duration was calculated as the difference between pickup and dropoff time, and trip distance was calculated using the pickup and dropoff longitudes and latitudes. Corelations among these features were studied and none of these features shared a linear corelation with the dependent variable i.e., trip duration. To find a linear corelation between trip distance and trip duration, these two features were replaced by their natural logs (see Fig. 1 and Fig. 2). Finally, this categorical data was transformed through one-hot encoding. To build the models, anomalies were removed. The final dataset contains 1450690 rows and 37 columns (36 independent features and 1 dependent feature – natural log of trip duration). This dataset was split into training and testing sets in the ratio 8:2.

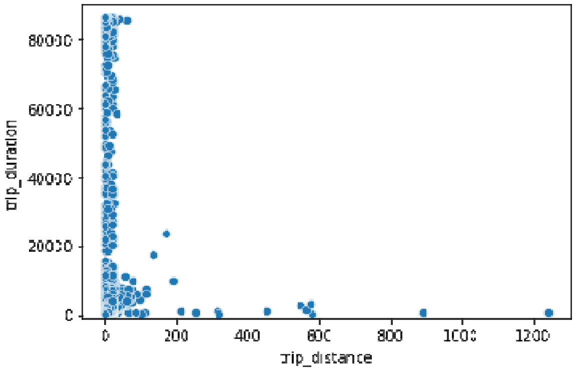

Fig. 1. Plot between trip duration and trip distance

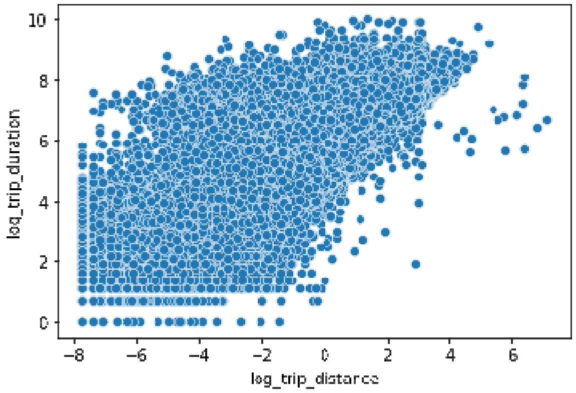

Fig. 2. Plot between natural log of trip duration and natural log of trip distance

## III. Methadology and Models

### A. Linear Models

Since one of the independent features- natural log of trip duration shares a linear relationship with the dependent feature- natural log of trip duration, linear models with different optimization techniques- Ordinary Least Squares (OLS), Ridge, Lasso and Elastic Net. Mathematically, linear regression is represented as

$$\hat{y}(w,x) = w_0 + w_1x_1 + \cdots + w_px_p \qquad (1)$$

where $\hat{y}(w,x)$ is the predicted value and $w_0, \ldots, w_p$ are the coefficients.

Regularization improves a model's performance by preventing overfitting. In some cases, even if the model does not suffer from overfitting, better performance can still be achieved with regularization. The method of Ordinary Least Squares (OLS) does not use any regularization. To check if regularization helps in achieving better accuracy, Ridge, Lasso and Elastic Net have also been implemented along with OLS. At first, models were fed only one independent feature – natural log of trip distance. On feeding other independent features, even though not significant, some improvement in the model performance was observed.

*1) Ordinary Least Squares:* OLS aims at minimizing the sum of squared residuals (SSR). Mathematically, the problem that is being solved is:

$$min_w ||Xw - y||_2^2 \qquad (2)$$

In [5], the author mentions that if the input features and the output feature share a true linear relationship, a low bias will be shown by OLS. If the number of observations, n (in this case, approximately 1160000) is much greater than the number of independent variables p (36), then the least squares estimates tend to also have low variance, and hence will perform well on test observations.
OLS has been implemented in python using class sklearn.linear_model.LinearRegression.

*2) Ridge Regression:* Ridge regression uses L2-norm regularization of coefficients. A penalized SSR is reduced by the ridge coefficients. The penalty term added to SSR is equivalent to the square of magnitude of coefficients:

$$SSR + \alpha||w||_2^2 \qquad (3)$$

This has been implemented in python using class sklearn.linear_model.Ridge.

*3) Lasso:* Lasso uses L1-norm regularization of coefficients. The penalty term added is equivalent to the absolute magnitude of coefficients. The objective function to minimize is :

$$min_w \frac{1}{2n_{samples}} ||Xw - y||_2^2 + \alpha||w||_1 \qquad (4)$$

Lasso has been implemented in python using the class sklearn.linear_model.Lasso (alpha = 0.1).

*4) Elastic Net:* Both L1 and L2-norm regularization of the coefficients is used by the Elastic Net regression. It works by minimizing the following objective function:

$$min_w \frac{1}{2n_{samples}} ||Xw - y||_2^2 + \alpha\rho||w||_1 + \frac{\alpha(1-\rho)}{2} ||w||_2^2 \qquad (5)$$

The convex combination of L1 and L2 is controlled using the *l1_ratio* parameter. If *l1_ratio*=1.0, Elastic Net acts as Lasso (no L2 penalty). Table 1 shows $R^2$ score, MSE, MAE for different *l1_ratio* for Elastic Net model. *l1_ratio* of 0.1 gives highest $R^2$ score and lowest MSE and MAE. Elastic Net model has been implemented in python using sklearn.linear_model.ElasticNet(alpha=0.1).

TABLE I. $R^2$ SCORE, MSE, MAE FOR DIFFERENT L1_RATIO FOR ELASTIC NET MODEL

| *l1_ratio* | **$R^2$ score** | **MSE** | **MAE** |
|---|---|---|---|
| 0.1 | 0.638 | 0.213 | 0.347 |
| 0.3 | 0.623 | 0.222 | 0.357 |

| | | | |
|---|---|---|---|
| 0.5 | 0.619 | 0.224 | 0.359 |
| 0.7 | 0.616 | 0.226 | 0.361 |
| 1.0 | 0.612 | 0.226 | 0.362 |

### B. *Random Forest*

Random Forest is a tree based ensemble model and has a nonlinear nature. Authors in [7] mention that different types of predictor variables can be handled by tree based ensemble methods. They prove to be good candidates when it comes to solving travel time prediction problems. In [2], Random Forest is used to account for nonlinear effect of location, and it outperforms all other models. However, in our study, effect of location is not accounted for and OLS outperforms Random Forest. This model has been implemented in python using the class sklearn.ensemble.RandomForestRegressor.

### C. *Deep Neural Network*

Artificial Neural Networks (ANN) are made up layers and nodes to mimic the network of neurons in the brain. ANNs are known for their ability to learn complex functions. If an ANN contains multiple hidden layers, it is termed as a Deep Neural Network (DNN). If the right hyperparameters are chosen, a DNN can learn both linear and nonlinear relationships between input and output. After numerous tests, the following combination of hyperparameters was selected for the deep learning model:

- No. of hidden layers: 2
- No. of nodes in hidden layers: 36 each
- Activation function: relu (hidden layers) & linear (final layer)
- Optimizer: Adam
- Learning Rate: 0.001
- Loss function: mean squared error
- Epochs: 100
- Batch size: 64

DNN has been implemented using tf.keras.Sequential class.

## IV. RESULTS

TABLE II. $R^2$ SCORE, MSE, MAE FOR DIFFERENT MODELS

| Model | $R^2$ score | MSE | MAE |
|---|---|---|---|
| OLS | 0.665 | 0.198 | 0.330 |
| Ridge | 0.645 | 0.207 | 0.340 |
| Lasso | 0.612 | 0.226 | 0.362 |
| Elastic Net(l1_ratio=0.1) | 0.638 | 0.213 | 0.347 |
| Random Forest | 0.611 | 0.229 | 0.360 |
| **Deep Neural Network** | **0.672** | **0.182** | **0.317** |

TABLE III. ACTUAL Y VS PREDICTED VALUES FOR DIFFERENT MODELS

| Actual y | 5.533 | 7.579 | 8.358 |
|---|---|---|---|
| OLS Prediction | 5.958 | 7.658 | 8.118 |
| Ridge Prediction | 5.963 | 7.670 | 7.975 |
| Lasso Prediction | 6.130 | 7.424 | 7.614 |
| Elastic Net Prediction | 6.058 | 7.541 | 7.764 |
| Random Forest Prediction | 6.033 | 7.431 | 7.702 |
| Deep Neural Network | 5.925 | 7.559 | 7.995 |

Table 2 contains values of $R^2$ score, MSE, MAE for different regression models implemented in this study. Among OLS, Ridge, Lasso and Elastic Net regression, OLS model performs best with highest $R^2$ score and lowest MSE and MAE. As discussed earlier, this is because the number of observations in the training set is much greater than the number of independent variables. Since a large number of training examples have been used, chances of overfitting were lower and regularization did not improve model performance. Ridge model, which uses L2 norm regularization, outperforms Lasso and Elastic Net.

In contrast to the results in [2], where nonlinearities in traffic and location effect is modeled by Random Forest, it did not outperform OLS model. In our study, although independent variables with nonlinear correlations have significance in improving model performance, the most significant independent variable – natural log of trip duration has high linear correlation with the dependent variable.

DNN outperforms all other models in this study. The model's performance on the testing set is given in table II. The validation set contained at least 500 examples. Model's performance was evaluated after each epoch using this validation set and $R^2$ score of 0.681, MSE of 0.175 and MAE of 0.309 was recorded after 100 epochs.

A comparison between actual value of natural log of trip duration (actual y) and predicted values by different regression models is given in Table 3. Comparing the prediction made by DNN with Google Maps with real time traffic updates, DNN predicts a trip duration of 26.89 minutes for a trip taken from Statue of Liberty to Empire State Building on 13 April 2021 at 18 hours 15 minutes, while Google maps show an ETA of 21, 23 and 25 minutes depending upon the route taken.

## V. CONCLUSION AND FUTURE WORK

Using features like trip distance, month, day of the week and time have been successful at predicting trip distance using regression models. Among all the six models, DNN gives the lowest MSE, followed by the OLS model.

In this study, trip distance is being calculated as the distance between two GPS coordinates along the curvature of the earth. Accuracy can be improved if trip distance is calculated according to the preferred route which can be chosen on the basis of other input features.

## REFERENCES

[1] Google Maps Platform: Predicting the Future with Google Maps APIs, https://maps-apis.googleblog.com/2015/11/predicting-future-with-google-maps-apis.html. Last accessed 10 Jan 2021

[2] Antoniades C, Fadavi D, Amon AF. Fare and duration prediction: A study of New York city taxi rides. Tech. Rep(2016)

[3] Markou, Ioulia, Filipe Rodrigues, and Francisco C. Pereira. "Use of taxi-trip data in analysis of demand patterns for detection and explanation of anomalies," Transportation Research Record 2643, pp.129-138,2017

[4] Pedregosa et al,"Scikit-learn: Machine Learning in Python," JMLR 12, pp. 2825-2830 ,2011

[5] G. James et al,"An introduction to statistical learning: with applications in R," Springer Texts in Statistics,Springer Science+Business Media,New York (2013) DOI 10.1007/978-1-4614-7138-7

[6] Scikit-learn 1.1Linear Models,https://scikit-learn.org/stable/modules/linear_model.html#linear-models. Last accessed 10 Jan 2021

[7] Zhang, Yanru, and Ali H," A gradient boosting method to improve travel time prediction," Transportation Research Part C: Emerging Technologies 58, pp. 308-324(2015)