NYC Taxi Time Prediction Analysis

JUNAID A R

Abstract:

In New York City, taxi rides form the backbone of New York City traffic. Estimating the taxi ride time is very important because users always want to know exactly how long it will take to get from one place to another. Duration forecasting helps users to plan their trips properly, thus maintaining potential margin for traffic congestion. It also helps the driver to determine the correct route and thus takes less time. Transparency of travel duration will also help attract users.

We used data from six months of 2016, which includes data on customers who provide information at the start of a trip or when booking a trip.

Introduction:

New York is one of the most developed cities in the world where taxi services are widely used. The need for public transport along with its huge population serves a common purpose as it provides a very large transport system. New York City has one of the largest subway systems in the world, comprising approximately 13,000 different green and yellow cabs. A significant portion of New York City's population relies on public transportation, and an estimated 54% do not own a car or personal vehicle.

As a matter of fact, it accounts for almost 200 million taxi trips per year. Our goal here is to build a predictive model, which could help in predicting the time duration of each trip.

Problem Statement:

Task is to build a model that predicts the total ride duration of taxi trips in New York City. Your 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.

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.

Steps :

Data Pre-processing:

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. It involves the following:

Check Missing Data: No null, NaN or missing values found in the dataset.

Create new variables: New variables were derived using datetime variable and also distance and speed using coordinates and trip duration.

Encoding Categorical Data: The categorical variables such as store and forward flag, months and days are encoded using one hot encoding.

Data Correlation Reduction: This includes a correlation heatmap and some features are removed based on it to keep collinearity to a minimum. Split the

dataset into training and test sets: The data is split in an 80-20 format. That is, 80% for training and 20% for testing.

Exploratory Data Analysis:

Exploratory Data Analysis is the critical process of conducting an initial exploration of data to discover patterns, detect anomalies, test hypotheses, and test assumptions through summary statistics and graphical representations. says

This includes one-dimensional and two-dimensional analysis.

Univariate Analysis: Univariate analysis is the simplest form of data analysis. Each variable is analyzed individually.

Bivariate Analysis: Bivariate analysis is stated to analyze the parallel relationship between the dependent variable, trip_duration, and another independent variable.

Model implementation:

Model implemented for this dataset:

- Linear regression
- Lasso regression
- Ridge regression
- Decision tree regressor
- XGB regressor

Cross-validation and hyperparameter tuning:

Cross-validation and hyperparameter tuning using grid search CV Perform hyperparameter tuning.

This is done to reduce overfitting of the model.

Algorithm:

Linear Regression:

Linear regression 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. A variable that is used to predict the value of another variable is called an independent variable.

Data Preparation for Linear Regression:

Linear Assumption: Linear regression assumes that the relationship between independence and dependence is linear.

Remove Outlier: Linear regression assumes that your independent and dependent variables are not noisy.

Remove Collinearity: Linear regression will over-fit your data when you have highly correlated input variables.

Gaussian Distributions: Linear regression will make more reliable predictions if your independent and dependent variables have a Gaussian distribution.

Rescale Inputs: Linear regression will often make more reliable predictions if you rescale input variables using standardization or normalization.

Regularisation

Regularized linear regression models are very similar to least squares, except that the coefficients are estimated by minimizing a slightly different objective function.

We minimize the sum of the RSS and the "penalty period" that penalizes the odds size.

Lasso regression compresses the coefficients to zero and removes them from the model.

$$\text{RSS} + \lambda \sum_{j=1}^{p} |\beta_j|$$

Ridge regression reduces coefficients to zero, but rarely reaches zero.

$$\text{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2$$

Decision Tree Regressor

Decision Tree builds regression or classification models in the form of tree structures. Decision trees are developed incrementally while dividing the data set into smaller and smaller subsets. The final result is a tree with decision nodes and leaf nodes.

Decision Tree Terminology:

Root Node: Represents the entire population or sample and is further divided into two sets.

Split: The process of splitting a node into two subnodes.

Decision Node: When a child node splits into additional child nodes, it is called a decision node.

Leaf/Leaf Node: A node that is not split is called a leaf or leaf node.

Pruning: When removing decision node child nodes, this process is called pruning. We can say the reverse process of division.

Branch/Subtree: A subsection of the whole tree is called a branch or subtree.

Parent and child nodes: A node divided into subnodes is called the parent node of subnodes, and subnodes are children of the parent node.

Entropy:

This is just a measure of the uncertainty or disorder of the data set. Calculated using,

$$\text{Entropy} = -p \log_2 p - q \log_2 q$$

Information Gain: The decrease in entropy after

division is called Information Gain. Steps to compute split information to split:

Calculate entropy of parent node.

Calculation of the entropy of each individual split node and calculation of the weighted average of all subnodes available in the split. Calculate the difference in entropy before and after the

split.

Gini:

Gini says, if we select two items from a population at random then, if the population is pure, they must be of same class and probability for this is 1.

Steps to calculate Gini for a split:

Calculate Gini for sub-nodes, using formula sum of square of probability for success and failure (p^2+q^2).

Calculate Gini for split using weighted Gini score of each node of that split

XG Boost Regressor

Decision trees are prone to overfitting. ensemble learning is one way to tackle bias-variance trade-off. There are various ways to ensemble weak learners to come up with strong learners:

Bagging

Boosting

Stacking

We are focusing on boosting since it is about XG Boost.

Boost: The boost matches a set of weak learners (models slightly better than random guesses like small decision trees) to a weighted version of the data. More attention is paid to cases that were misclassified in previous rounds.

Gradient Boosting Machine The gradient boosting machine (GBM), like other boosting methods, builds a model step by step and generalizes it, allowing optimization of any differentiable loss function.

Steps involved:

Data set {(x1,y1),(x2,y2),(x3,y3),..

..,(xn,yn)} Choose a loss function such as

MSE. We call this model F0(x) by fitting the

naïve model to a dataset, a simple tree, or taking only $\bar{y}$.

First iteration:

Get the residuals of all predictions. regression tree) remainder {(x1,r11),(x2,r21),(x3,r31),..

..,(xn1,rn1)}, let's call this model h1(x).

New predictor $F1(x)=F0(x)+\gamma 1h1(x)$. Find $\gamma 1$ that minimizes the standard deviation.

Second iteration:

Get the residuals of all predictions (ri2(x)=yi-F1(x) p32).

...,(xn2,rn2)}, name this model h2(x).

New predictor $F2(x)=F1(x)+\gamma 2h2(x)$. Find the $\gamma 2$ that minimizes the standard deviation.

and so on...

get residuals of all predictions, rim(x)=yi-Fm-1(x)

residuals {(x1,r1m),(x2 ,r2m) ,(x3,r3m),..

..,(xnm,rnm)}, let's call this model hm(x)

. The final predictor is $Fm(x)=Fm-1(x)+\gamma mhm(x)$. Find γm that minimizes MSE.

XG Boost is an implementation of Gradient Boosted Decision Trees.

It does this by tackling one of the major inefficiencies of gradient boosted trees: considering the potential loss for all possible splits to create a new branch (especially if you consider the case where there are thousands of features, and therefore thousands of possible splits). XG Boost tackles this inefficiency by looking at the distribution of features across all data points in a leaf and using this information to reduce the search space of possible feature splits.

Cross Validation and Hyperparameter tuning

In cross-validation, we run our modelling process on different subsets of the data to get multiple measures of model quality. For example, we could have 5 folds or experiments. We divide the data into 5 pieces, each being 20% of the full dataset.

Hyperparameters are sets of information used to control how an algorithm learns. Each algorithm requires a specific grid of hyperparameters that can be customized to fit your business problem.

: I used the GridSearchCV provided in the science kit. Mesh search combines a set of hyperparameters and goes through them all to evaluate the model's performance. Its advantage is that it is a simple technique that goes through all 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.

Model Performance

A regression model performance can be evaluated using the following metrics.

R^2

Adjusted R^2

Mean Squared Error

Root Mean Squared Error

Model Analysis of Train and Test data using above metrics

Train Data:

Model Name	Train MSE	Train RMSE	Train R^2	Train Adjusted R^2
Linear Regression	0.032211	0.179475	0.674067	0.674059
Lasso Regression	0.032211	0.179475	0.674069	0.674498
Ridge Regression	0.032211	0.179475	0.674069	0.674061
DecisionTree Regressor	0.000981	0.031314	0.990078	0.990078
XGBoost Regressor	0.000123	0.011091	0.998755	0.998755

Test Data:

Model Name	Test MSE	Test RMSE	Test R^2	Test Adjusted R^2
Linear Regression	0.032213	0.179479	0.674528	0.674496
Lasso Regression	0.032213	0.179479	0.674530	0.674498
Ridge Regression	0.032213	0.179479	0.674528	0.674496
DecisionTree Regressor	0.001030	0.032094	0.989593	0.989592
XGBoost Regressor	0.000438	0.020939	0.995570	0.995570

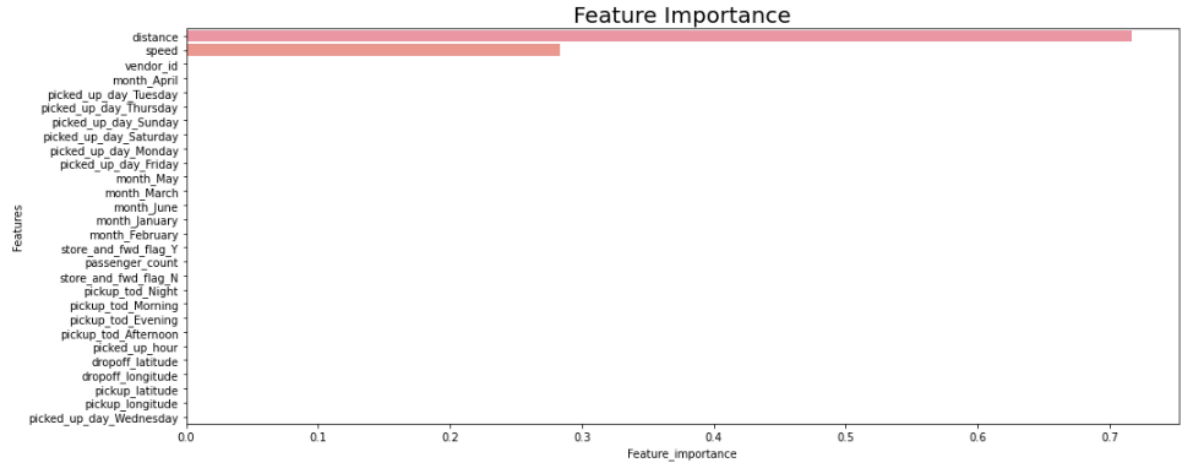
Decision Tree Regressor and XGB Regressor are the best models.

Feature Importance

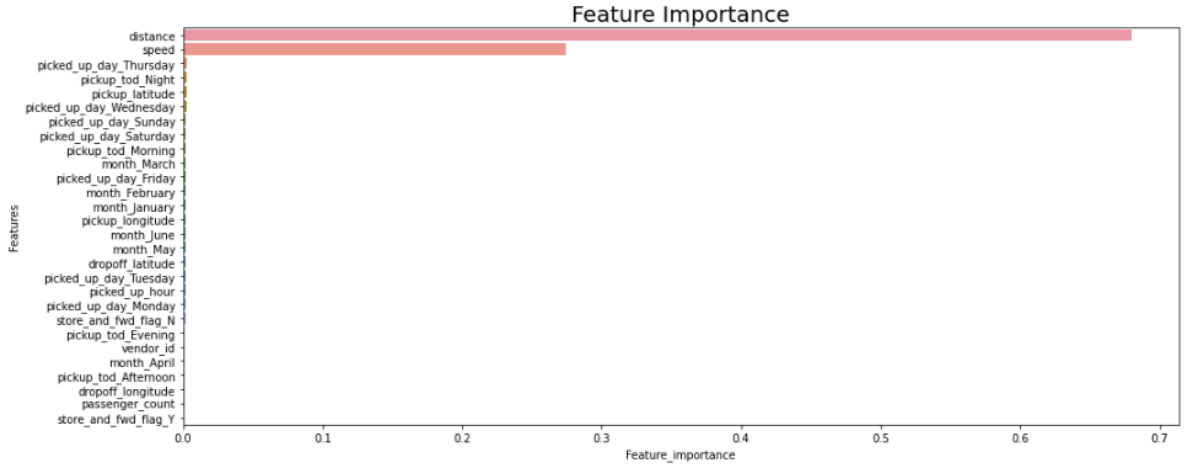
Feature importance is calculated as the decrease in node impurity weighted by the probability of reaching that node. The node probability can be calculated by the number of samples that reach the node, divided by the total number of samples.

The higher the value the more important the feature.

Decision Tree



XGB Regressor



From the above feature importance graph of both Decision Tree and XGB Regressor, we can say that distance and speed are the variables that were given importance.

Conclusion:

Linear, Ridge and Lasso regression overall have the same score even after cross validation and hyperparameter tuning of lasso and ridge.

Decision Tree Regression is better performing than the above-mentioned models, but still when compared to XGB Regressor it is somewhat less significant.

From the above analysis based on the regression metrics also the advantages of XGB Regressor, we can say that XGB Regressor is best performing model and can be used to predict the trip duration.