**NYC Taxi Time Prediction Analysis**

**Ajay Pradeep M**

**Abstract:**

New York City taxi rides form the core of the traffic in the city of New York. 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 of him to travel from one place to another. Prediction of duration 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.

We used six months data, of the year 2016 which contains data of customers who would provide at the start of a ride, or while booking a ride.

**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 of around 13,000 taxis. 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. 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. (DV)

**Steps involved:**

**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.
* **Reducing data correlation:** This involves a correlation heatmap and based on that some features are removed in order to reduce collinearity to minimum.
* **Splitting dataset into training and test set:** The data is split into 80-20 format i.e. 80% for training and 20 % for testing purpose.

**Exploratory Data Analysis:**

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations. This includes univariate and bivariate analysis.

* **Univariate Analysis:** Univariate analysis is the simplest form of analysingdata. Each variable is analyzed separately.
* **Bivariate Analysis:** Bivariate analysis is stated to be an analysis of anyconcurrent relation between the dependent variable trip\_duration and other independent variables.

**Model Implementation:**

The models implemented for this dataset are:

* Linear Regression
* Lasso Regression
* Ridge Regression
* Decision Tree Regressor
* XGB Regressor

**Cross Validation and Hyperparameter Tuning:**

The cross validation and hyperparameter tuning are done using Grid Search CV. This is to reduce overfitting of the models.

**Algorithms:**

**Linear Regression:**

Linear regression analysis is used to predict the value of a variable based on the valueof another variable.

The variable you want to predict is called the dependent variable. The variable you are using to predict the other variable's value is called the independent variable.



**Data Preparation for Linear Regression:**

**Linear Assumption**: Linear regression assumes that the relationship between your independent and dependent 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 RSS and a "penalty term"** that penalizes coefficient size.

### ****Lasso regression**** shrinks coefficients all the way to zero, thus removing them from the model.

### 

### ****Ridge regression**** shrinks coefficients toward zero, but they rarely reach zero.

### 

### Visualizing Regularization

### 

### For **ridge**, this region is a **circle** because it constrains the square of the coefficients. Thus, the intersection will not generally occur on an axis, and so the coefficient estimates will be typically be non-zero.

### For **lasso**, this region is a **diamond** because it constrains the absolute value of the coefficients. Because the constraint has corners at each of the axes, and so the ellipse will often intersect the constraint region at an axis. When this occurs, one of the coefficients will equal zero. In higher dimensions, many of the coefficient estimates may equal zero simultaneously. In the figure above, the intersection occurs at β1=0, and so the resulting model will only include β2.

### Decision Tree Regressor

### Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.

### Decision Trees Terminologies:

* **Root Node:** It represents entire population or sample and this further gets divided into two sets.
* **Splitting:** It is a process of dividing a node into two sub-nodes.
* **Decision Node:** When a sub-node splits into further sub-nodes, then it is called decision node.
* **Leaf/ Terminal Node:** Nodes do not split is called leaf or terminal node.
* **Pruning:** When we remove sub-nodes of a decision node, this process is called pruning. You can say opposite process of splitting.
* **Branch / Sub-Tree:** A sub section of entire tree is called branch or sub-tree.
* **Parent and Child Node:** A node, which is divided into sub-nodes is called parent node of sub-nodes whereas sub-nodes are the child of parent node.

### Entropy:

### It is nothing but the uncertainty in our dataset or measure of disorder. It is calculated using

### 

**Information Gain:**

The decrease is entropy after split is called **Information Gain**. Steps to calculate information split for a split:

* Calculate entropy of parent node
* Calculate entropy of each individual node of split and calculate weighted average of all sub-nodes available in split.
* Calculate the difference in entropy before and after 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**

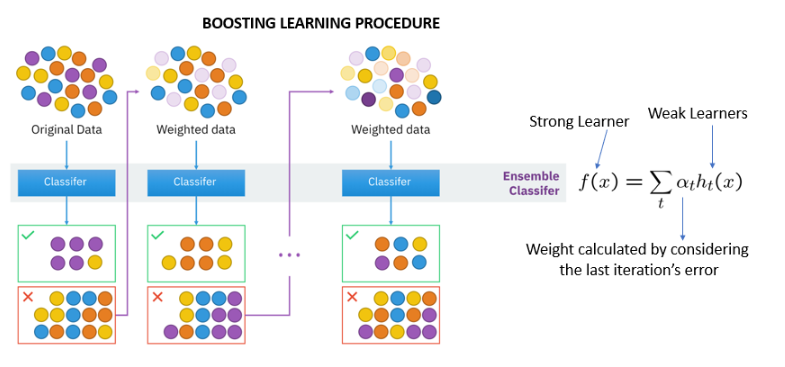
Decision trees are prone to overfitting. **ensemble learning is one way to tackle bias-variance trade-off.** There are various ways to ensemble weak learners to come up with strong learners:

* Bagging
* Boosting
* Stacking

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

**Boosting:**

Boosting fit a sequence of weak learners − models that are only slightly better than random guessing, such as small decision trees − to weighted versions of the data. More weight is given to examples that were misclassified by earlier rounds.



### ****Gradient Boosting Machine****

Gradient Boosting Machine (GBM) builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

#### **The steps involved are:**

* Let's take a dataset {(x1,y1),(x2,y2),(x3,y3),....,(xn,yn)}
* Choose a loss function, let's say MSE.
* Fit a naive model on the dataset, a simple tree or just take y¯, call this model F0(x)

**First iteration:**

* Get residuals of all predictions, ri1(x)=yi−F0(xi)
* Fit a model (can be regression tree) on residuals {(x1,r11),(x2,r21),(x3,r31),....,(xn1,rn1)}, call this model h1(x)
* New predictor is F1(x)=F0(x)+γ1h1(x). Find γ1 which minimizes MSE.

**Second iteration:**

* Get residuals of all predictions, ri2(x)=yi−F1(x)
* Fit a model (can be regression tree) on residuals {(x1,r12),(x2,r22),(x3,r32),....,(xn2,rn2)}, call this model h2(x)
* New predictor is F2(x)=F1(x)+γ2h2(x). Find γ2 which minimizes MSE.

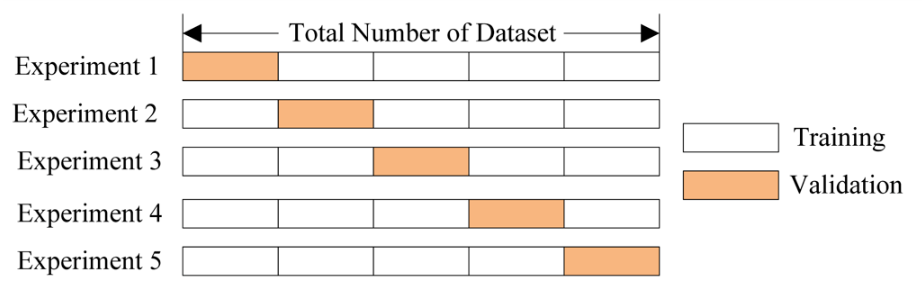
**and so on...**

* Get residuals of all predictions, rim(x)=yi−Fm−1(x)
* Fit a model (can be regression tree) on residuals {(x1,r1m),(x2,r2m),(x3,r3m),....,(xnm,rnm)}, call this model hm(x)
* **Final predictor** is Fm(x)=Fm−1(x)+γmhm(x). Find γm which 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 that are used to control the way of learning an algorithm. Each algorithm requires a specific hyperparameters grid that can be adjusted according to the business problem.

We have used GridSearchCV which is available in sci-kit learn. Grid Search combines a selection of hyperparameters 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 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.

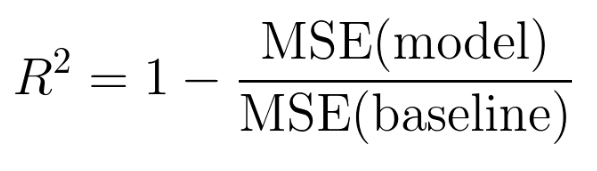
* R^2
* Adjusted R^2
* Mean Squared Error
* Root Mean Squared Error

**R^2:**

This metric helps us to compare our current model with a constant baseline and tells us how much our model is better.

The constant baseline is chosen by taking the mean of the data and drawing a line at the mean.

R² is a scale-free score that implies it doesn't matter whether the values are too large or too small, the R² will always be less than or equal to 1.

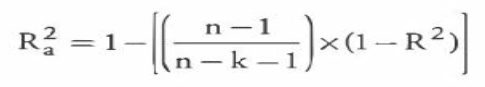


**Adjusted R^2:**

Adjusted R² depicts the same meaning as R² but is an improvement of it.

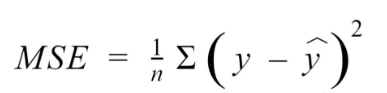
R² suffers from the problem that the scores improve on increasing terms even though the model is not improving which may misguide the researcher.

Adjusted R² is always lower than R² as it adjusts for the increasing predictors and only shows improvement if there is a real improvement.



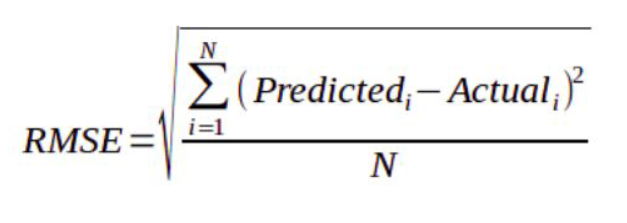
**Mean Squared Error:**

MSE or Mean Squared Error is one of the most preferred metrics for regression tasks. It is simply the average of the squared difference between the target value and the value predicted by the regression model.



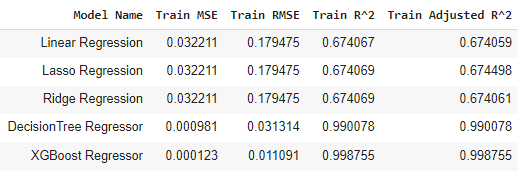
**Root Mean Squared Error:**

RMSE is the most widely used metric for regression tasks and is the square root of the averaged squared difference between the target value and the value predicted by the model. It is preferred more in some cases because the errors are first squared before averaging which poses a high penalty on large errors.

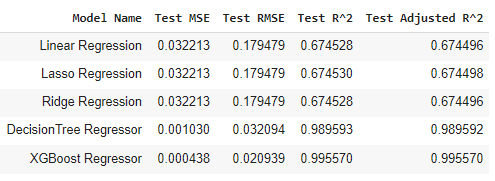


**Model Analysis of Train and Test data using above metrics**

**Train Data:**



**Test Data:**

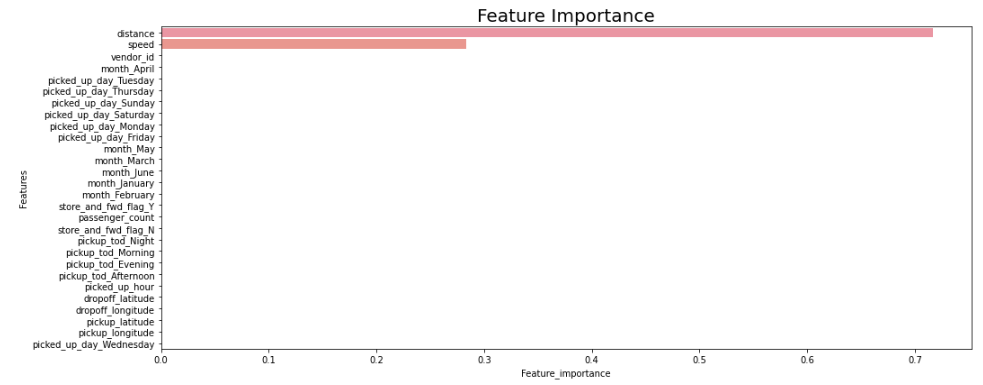


Decision Tree Regressor and XGB Regressor are the best models.

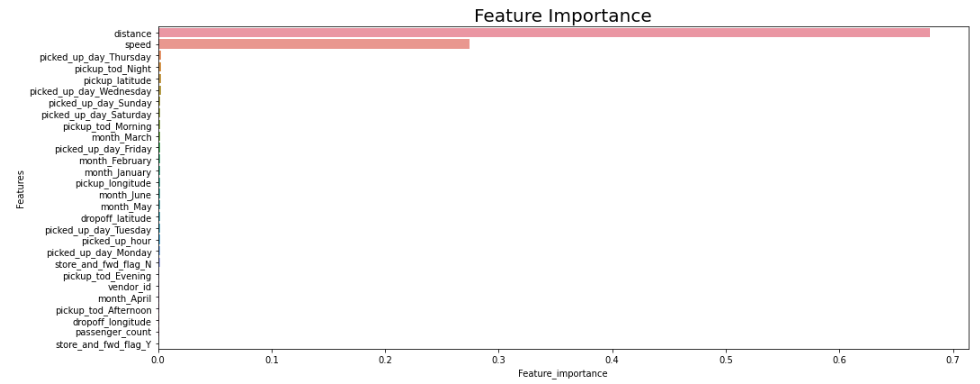
**Feature Importance**

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