**NYC Taxi Trip Time Prediction**

**Susovan Patra**

**Turab Ahmed**

**AlmaBetter, Bangalore**

**1.Introduction:**

New York City taxi rides paint a vibrant picture of life in the city. The millions of rides taken each month can provide insight into traffic patterns, road blockage, or large-scale events that attract many New Yorkers. With ridesharing apps gaining popularity, it is increasingly important for taxi companies to provide visibility to their estimated ride duration, since the competing apps provide these metrics upfront.

Predicting duration of a ride can help passengers decide when is the optimal time to start their commute, or help drivers decide which of two potential rides will be more profitable, for example. In order to predict duration, only data which would be available at the beginning of a ride was used. This includes pickup and dropoff coordinates, trip distance, number of passengers. Linear regression, Lasso, Ridge and XGBoost Regressor were used to predict duration of the ride.

**2.Data and Problem Statement:**

The data used in this study are all subsets of New York City Taxi, which contains observations on around 1.4 million+ of New York City taxi rides in the year 2016. To build the models, 1 million observations were used, of which 800,000+ were used for training and 200,000+ for validation.

The main objective is to build a predictive model, which could help in predicting the duration time of the ride. This would in turn help them in matching the right cabs with the right customers quickly and efficiently.

* 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
* pickup\_day - Day of the week
* dropoff\_day - Day of the week
* pickup\_day\_no - Day of the week in numerical representation
* dropoff\_day\_no - Day of the week in numerical representation
* pickup\_hours – The hour of the pickup time(24 hour format)
* dropoff\_hours– The hour of the pickup time(24 hour format)
* pickup\_month – Pick up month in numerical representation
* dropoff\_month - Pick up month in numerical representation
* pickup\_shift - Shift of the day
* dropoff\_shift - Shift of the day
* 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
* distance – The distance between the ride in KM.

**3. Steps involved:**

* **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 figuring out various aspects and relationships among the target and the independent variables. It gave us a better idea of which feature behaves in which manner compared to the target variable.

* **Outliers Treatment**

Our dataset contain less number of outliers which might tend to disturb our accuracy hence we dropped them at the beginning of our project in order to get a better result.

* **Encoding of Object/categorical columns**

We converted the object datatype column ‘pickup\_datetime‘ to date time datatype and segregated with the Week day number to produce integers of 0 to 6 encode our object/categorical features because categorical features that are in string/object format cannot be understood by the machine and needs to be converted to numerical format.

* **Feature Selection**

In these steps we used algorithms like XGBoost regressor, Ridge and Lasso to check the results of each feature i.e which feature is more important compared to our model and which is of less importance.

we used OLS ANOVA for numerical features to select the best feature which we will be using further in our model.

* **Standardization of features**

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.

The basic goal was to enforce a level of consistency or uniformity to certain practices or operations within the selected environment.

* **Fitting different models**

For modelling we tried various classification algorithms like:

1. **Linear Regression**
2. **Ridge Regression**
3. **Lasso Regression**
4. **Elastic Net Regression**
5. **XGBoost Regressor**

* **Tuning the hyperparameters for better accuracy**

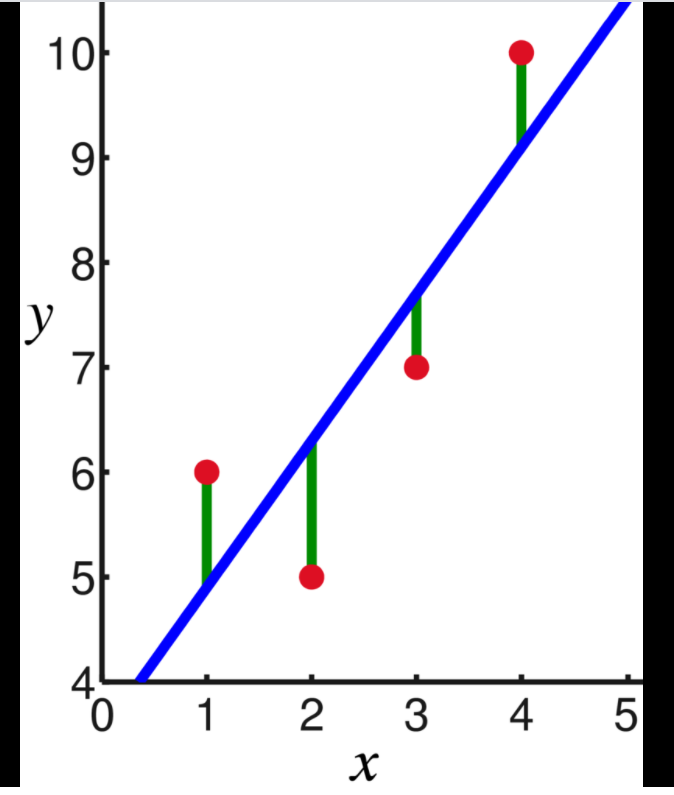
Tuning the hyperparameters of respective algorithms is necessary for getting better accuracy by selecting the lambda rate(regression strength)

**4. Algorithms:**

* 1. **Linear Regression:**

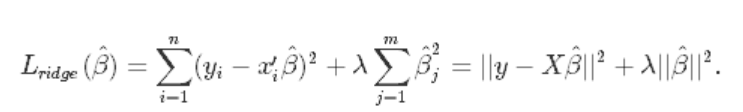
Linear Regression is [linear](https://en.wikipedia.org/wiki/Linearity) approach for modelling the relationship between a [scalar](https://en.wikipedia.org/wiki/Scalar_(mathematics)) response and one or more explanatory variables (also known as [dependent and independent variables](https://en.wikipedia.org/wiki/Dependent_and_independent_variables)). The case of one explanatory variable is called [simple linear regression](https://en.wikipedia.org/wiki/Simple_linear_regression); for more than one, the process is called multiple linear regression. This term is distinct from [multivariate linear regression](https://en.wikipedia.org/wiki/Multivariate_linear_regression), where multiple [correlated](https://en.wikipedia.org/wiki/Correlation_and_dependence) dependent variables are predicted, rather than a single scalar variable.

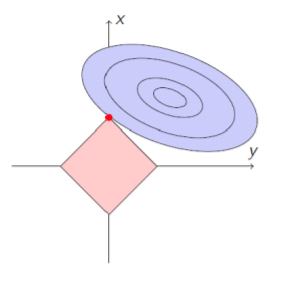




.**4.2 Ridge Regression:**

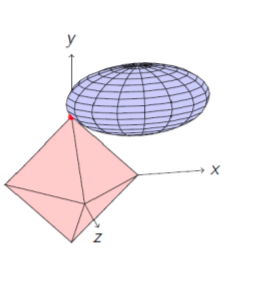
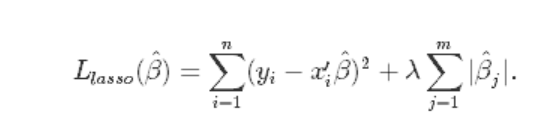
Ridge regression is a method of estimating the [coefficients](https://en.wikipedia.org/wiki/Coefficient) of multiple-[regression models](https://en.wikipedia.org/wiki/Regression_model) in scenarios where independent variables are highly correlated. Ridge regression was developed as a possible solution to the imprecision of least square estimators when linear regression models have some multicollinear (highly correlated) independent variables—by creating a ridge regression estimator (RR). This provides a more precise ridge parameters estimate, as its variance and mean square estimator are often smaller than the least square estimators previously derived.





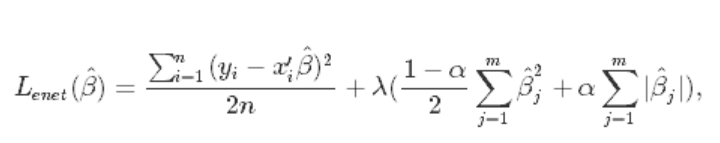
* 1. **Lasso Regression:**

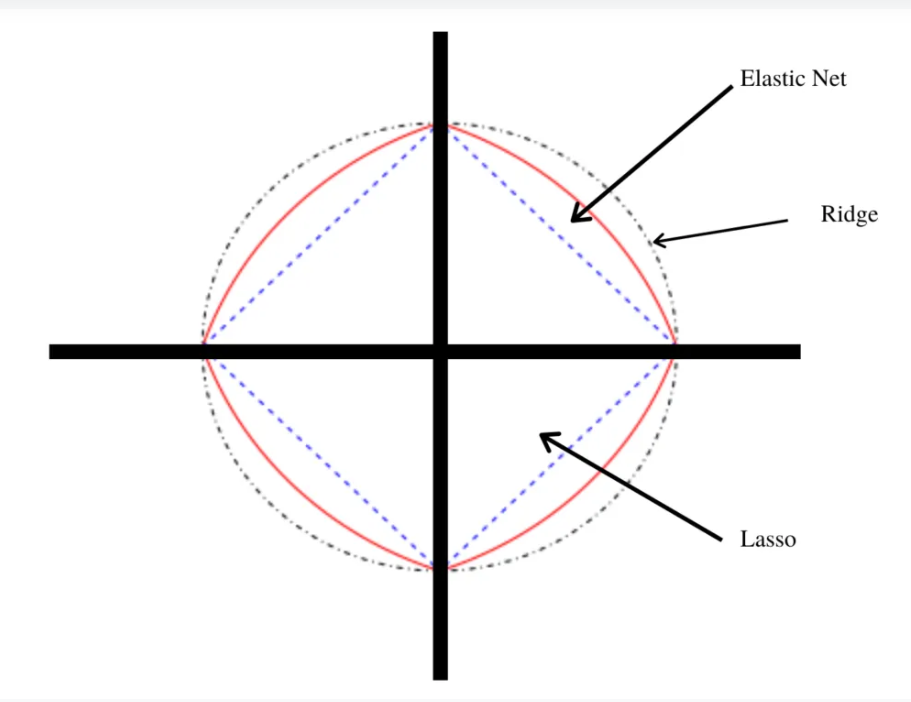
Lasso, or Least Absolute Shrinkage and Selection Operator, is quite similar conceptually to ridge regression. It also adds a penalty for non-zero coefficients, but unlike ridge regression which penalizes sum of squared coefficients (the so-called L2 penalty), lasso penalizes the sum of their absolute values (L1 penalty). As a result, for high values of *λ*, many coefficients are exactly zeroed under lasso, which is never the case in ridge regression.



* 1. **Elastic Net Regression:**

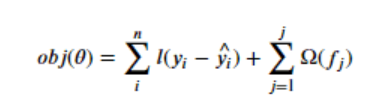
Elastic net linear regression uses the penalties from both the lasso and ridge techniques to regularize regression models. The technique combines both the [lasso](https://corporatefinanceinstitute.com/resources/knowledge/other/lasso/) and ridge regression methods by learning from their shortcomings to improve the regularization





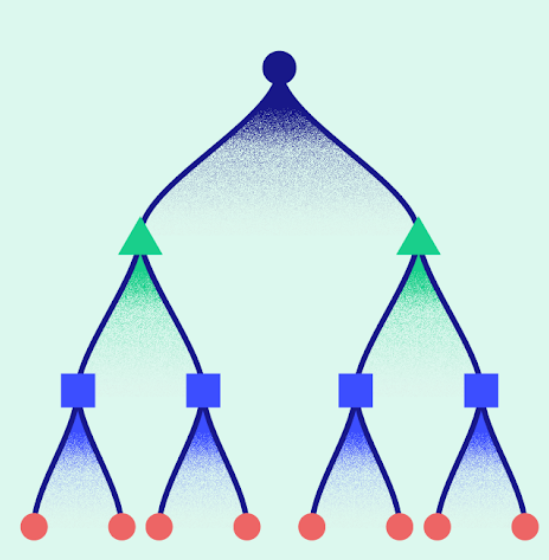
* 1. **XGBoost Regressor:**

XGBoost is a gradient boosting package that implements a gradient boosting framework. The algorithm is scalable for parallel computing



**4.5.1 Gradient Boosting:**

Gradient boosted trees consider the special case where the simple model is a decision tree



**XGBoost** is one of the fastest implementations of gradient boosting. 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). XGBoost 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.

**5. 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 for hyperparameter tuning. This also results in cross validation and in our case we divided the dataset into different folds.

**6.1 Grid Search CV:**

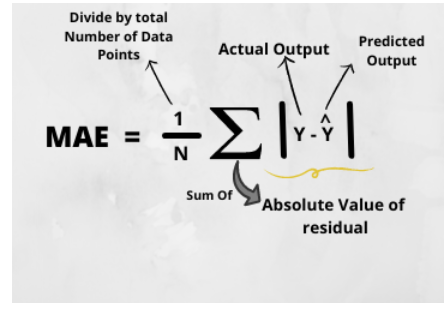
Grid Search combines a selection of hyperparameters established by the scientist and runs through all of them to evaluate the model’s performance. Its advantage is that it is a simple technique that will go through all the programmed combinations. The biggest disadvantage is that it traverses a specific region of the parameter space and cannot understand which movement or which region of the space is important to optimize the model.

**6. Model performance:**

The performance of the model can be evaluated using the following approaches

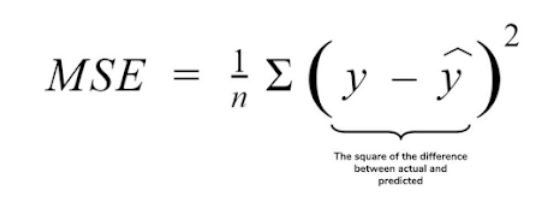
**6.1. Mean Absolute Error:**

MAE is a very simple metric which calculates the absolute difference between actual and predicted values.



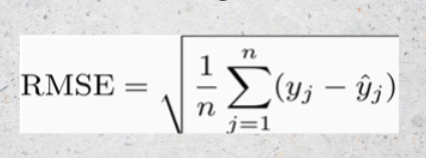
**6.2. Mean Squared Error:**

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.



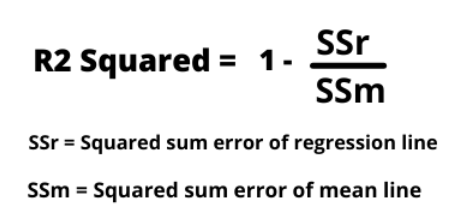
**6.3. Root Mean Squared Error:**

As RMSE is clear by the name itself, that it is a simple square root of mean squared error.



**6.4. R2 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.



**7. Conclusion:**

That's it! We reached the end of our exercise.

Starting with loading the data so far we have done EDA , Outliers treatment, encoding of categorical columns, feature selection and then model building.

In all of these models our accuracy revolves in the range of 40 to 69%.

And there is no such improvement in R2 score even after hyperparameter tuning.

So the accuracy of our best model is 69% which can be said to be good for this large dataset. This performance could be due to various reasons like: no proper pattern of data, too much data, not enough relevant features.

**References-**

1. Machine Learning Mastery
2. Wikipedia
3. Analytics Vidhya