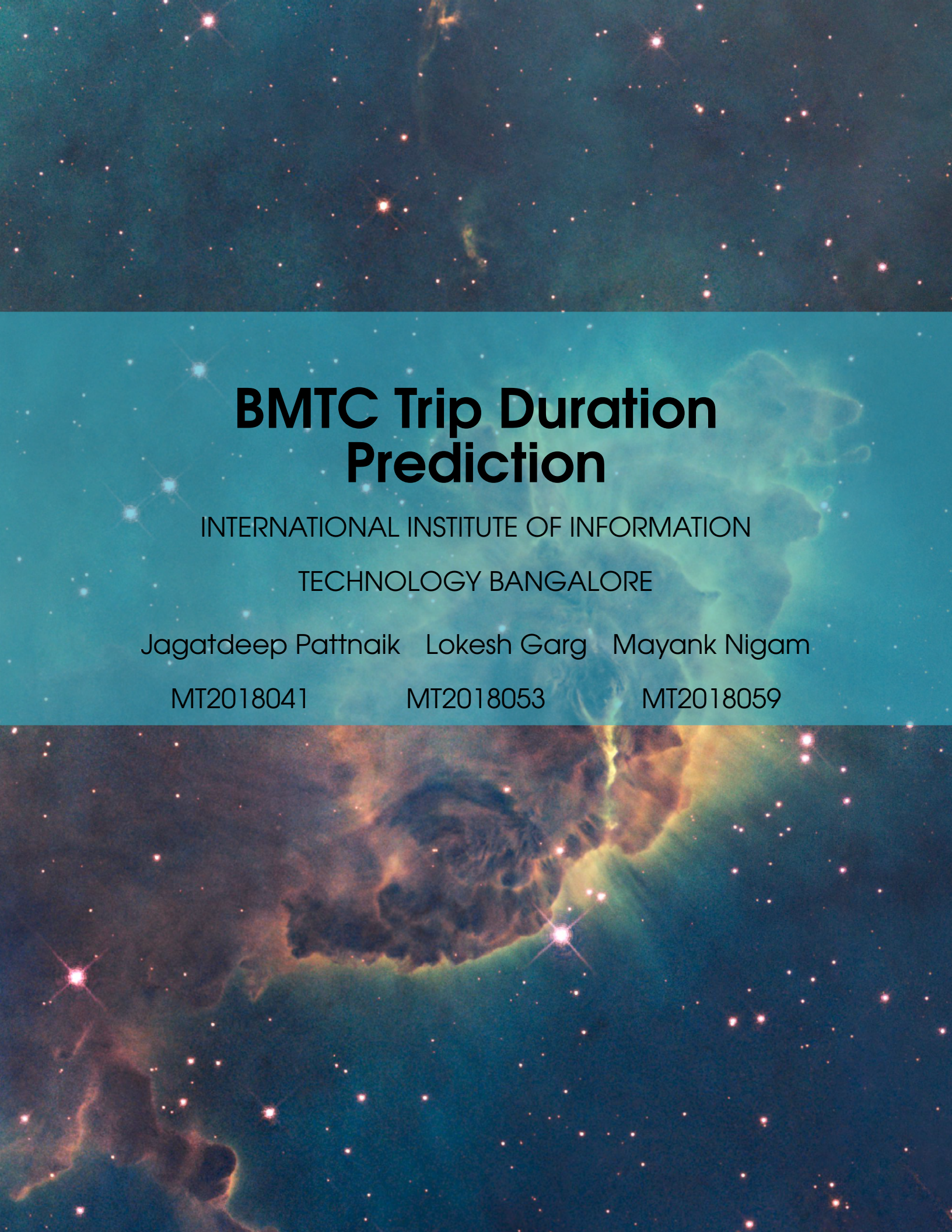


BMTC Trip Duration Prediction

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY BANGALORE

Jagatdeep Pattnaik MT2018041

Lokesh Garg MT2018053

Mayank Nigam MT2018059

ABSTRACT

Predicting the travel time with considerable accuracy and reliability is critically important for advanced traffic management and route planning in Intelligent Transportation Systems (ITS). Travel time prediction uses real travel time values within a sliding time window to predict travel time one or several time step(s) in future. However, the non stationary properties and abrupt changes of travel time series make challenges in obtaining accurate and reliable predictions. To benefit the commuters and promote the usage of public transportation, the BMTC(Bengaluru Metropolitan Transport Corporation) launched its Intelligent Transport System to provide the real time information about the arrival of bus. To predict the arrival time of buses we used different machine learning algorithms.

Contents

1. Introduction

The Bengaluru Metropolitan Transport Corporation is the sole public bus transport provider for Bengaluru, serving urban, sub-urban and rural areas. BMTC is committed to provide quality, safe, reliable, clean and affordable travel. The testimony of its success lies in increasing passenger trips everyday by a wide range of customer base. In an effort to modernize its services for commuter comfort, BMTC strives to strengthen information systems and improve processes through introduction of intelligent technology solution, make capacity enhancement through infrastructure development, user-friendly interchange facilities, fleet upgradation and augmentation, apart from its core activities, which includes fare structuring, route network optimization, planning and monitoring.

To benefit the commuters and promote the usage of public transportation, the BMTC launched its Intelligent Transport System (ITS) on 25 May 2016. Under this project, BMTC buses were equipped with GPS in a phased manner which would transmit the location of the bus to the ITS control room. Now BMTC wants to facilitates the commutors by providing travel time or trip time. So for this it is the requirement for the prediction of the trip time with high accuracy.

Travel-time calculation depends on vehicle speed, traffic flow and occupancy, which are highly sensitive to weather conditions and traffic incidents. These features make travel-time predictions very complex and difficult to reach optimal accuracy. Nonetheless, daily, weekly and seasonal patterns can still be observed at a large scale. For instance, daily patterns distinguish rush hour and late night traffic, weekly patterns distinguish weekday and weekend traffic, while seasonal patterns distinguish winter and summer traffic. The time-varying feature germane to traffic behavior is the key to travel-time modeling.

2. Discovering what to do...

2.1 First ideas

To predict the travel time we get the extracted data from the Intelligent Transport System (ITS) implemented by BMTC. The data collected is of the date from 1st of Auguust to 6th of August 2016 i.e the data of buses of one week. Now the very first challenging problem infront of us is size of data i.e 15 Giga Bytes(GB) which is very difficult to deal on the machine of 4GB RAM. So we overcome it by chunking the dataset into the chunk size of 1 lakh each and we will remain with 2150 number of chunks.

As we know the data is the mixture of many buses so the requirement comes to sort the data bus id wise. We come to know that there are total of 6597 buses are there in dataset. So we divided the data into bus id wise which then turn to sorted into date and time wise.

Now we observe thet the data is to noisy which require lot of to model the data. For the preprocessing of data we use various techniques, which includes Scalling, Data scrapping, Normalization, data visualization and Feature extraction.

After that we apply various machine learning regression models by using predefined libraries provided by Scikit Learn. In this project we use Linear Regression, Random Forest, XG Boost, Gradient Boosting algorithms to build our model which can able to predict travel time.

3. Understand your data

Before continuing, first and most importantly you must select the *raw* data you are going to process and later after you acquire experience with an specific dataset the idea is to expand the algorithms to any kind of dataset. The important things are to learn how to input the data correctly, establish the right *learning parameters* in the selected algorithm and find the best way to visualize your results and interpret them correctly.

Features of Dataset:

- Bus Id
- Latitude
- Longitude
- Time Stamp

3.1 Latitude

In geography, latitude is a geographic coordinate that specifies the north–south position of a point on the Earth's surface. Latitude is an angle (defined below) which ranges from 0 at the Equator to 90 (North or South) at the poles. Lines of constant latitude, or parallels, run east–west as circles parallel to the equator. Latitude is used together with longitude to specify the precise location of features on the surface of the Earth. On its own, the term latitude should be taken to be the geodetic latitude as defined below. Briefly, geodetic latitude at a point is the angle formed by the vector perpendicular (or normal) to the ellipsoidal surface from that point, and the equatorial plane. Also defined are six auxiliary latitudes which are used in special applications.

3.2 Longitude

Longitude is a geographic coordinate that specifies the east–west position of a point on the Earth's surface. It is an angular measurement, usually expressed in degrees and denoted by the Greek letter lambda (). Meridians (lines running from pole to pole) connect points with the same longitude. By convention, one of these, the Prime Meridian, which passes through the Royal Observatory,

Greenwich, England, was allocated the position of 0 longitude. The longitude of other places is measured as the angle east or west from the Prime Meridian, ranging from 0 at the Prime Meridian to +180 eastward and 180 westward. Specifically, it is the angle between a plane through the Prime Meridian and a plane through both poles and the location in question. (This forms a right-handed coordinate system with the z-axis (right hand thumb) pointing from the Earth's center toward the North Pole and the x-axis (right hand index finger) extending from the Earth's center through the Equator at the Prime Meridian.)

Latitude and Longitude represents the particular location of the bus at given time instance.

3.3 Time Stamp

A timestamp is a sequence of characters or encoded information identifying when a certain event occurred, usually giving date and time of day, sometimes accurate to a small fraction of a second. From the dataset we observe that entry for a particular bus is taken for every 10 second.

Bus Id	Latitude	Longitude	Time Stamp
150218715	13.013666	77.50882	2016-07-01 00:00:03
150811210	12.957109	77.862457	2016-07-01 00:10:09
150813654	13.067285	77.425148	2016-07-01 00:10:12
150811427	13.025971	77.631737	2016-07-01 00:10:14
150221120	12.955228	77.595551	2016-07-01 00:10:13

Table 3.1: DataSet

3.4 Preprocessing And Visualization of Data

Data preprocessing is an important step in the data mining process. The phrase "garbage in, garbage out" is particularly applicable to machine learning projects. Data-gathering methods are often loosely controlled, resulting in out-of-range values (e.g., Latitude: 100), impossible data combinations (e.g., Latitude : 13.012312, Longitude : 0.00), missing values, etc. Analyzing data that has not been carefully screened for such problems can produce misleading results. Thus, the representation and quality of data is first and foremost before running an analysis. Often, data preprocessing is the most important phase of a machine learning project. // Various techniques for preprocesssing used :-

3.4.1 Scalling

Feature scaling is a method used to standardize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step.

3.4.2 Data Scrapping

The key element that distinguishes data scraping from regular parsing is that the output being scraped is intended for display to an end-user, rather than as input to another program, and is therefore usually neither documented nor structured for convenient parsing. Data scraping often involves ignoring binary data (usually images or multimedia data), display formatting, redundant labels, superfluous commentary, and other information which is either irrelevant or hinders automated processing.

3.4.3 Normalization

Normalization refers to the creation of shifted and scaled versions of statistics, where the intention is that these normalized values allow the comparison of corresponding normalized values for different datasets in a way that eliminates the effects of certain gross influences, as in an anomaly time series. Some types of normalization involve only a rescaling, to arrive at values relative to some size variable. In terms of levels of measurement, such ratios only make sense for ratio measurements (where ratios of measurements are meaningful), not interval measurements.

3.4.4 Visualize the Trip Duration given using log-scale Distplot in SeaBorn

We are asked to predict trip duration of the test set, so we first check what kind of trips durations are present in the dataset. First We plotted it on a plain scale and not on a log scale, and some of the records have very long trip and very small durations. Such trips are making all another trip invisible in the histogram on plain scale => We go ahead with the log scale. Another reason of using the log scale for visualizing trip-duration on the log scale is that this competition uses rmsle matrix so it would make sense to visualize the target variable in log scale only.

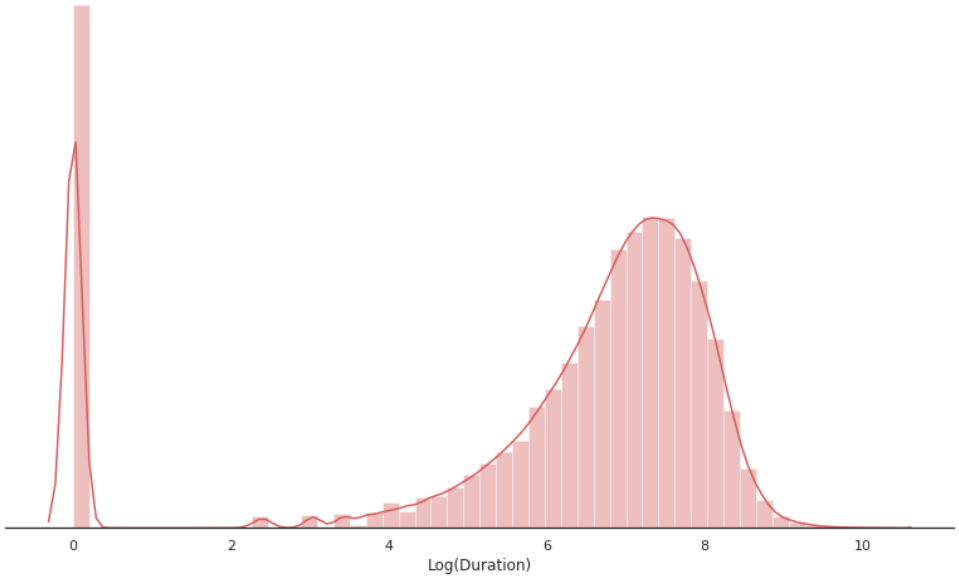


Figure 3.1: It is clear with the above histogram and kernel density plot that the trip durations are like Gaussian and few trips have very small duration while most of the trips are e^4 = 1 minute to e^9 120 minutes. Let's check the lat-long distributions are then used them to have a heat map kind of view of given lat-longs. But, we also see that most of the durations are 0 minutes. So we decided to remove that one.

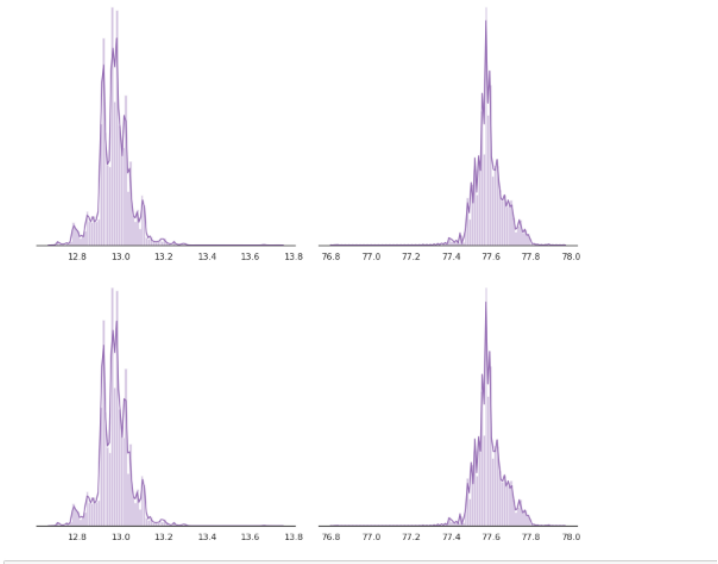


Figure 3.2: From the plot above it is clear that pick and drop latitude are centered around 12.7 to 13.4, and longitude are situated around -77.3 ton-77.8. We are not getting any histogram kind of plots when we are plotting lat-long as the distplot function of sns is getting affected by outliers, some trips which are very far from each other are taking very long time, and have affected this plot such that it is coming off as a spike. Let's remove those large duration trip by using a cap on lat-long and visualize the distributions of latitude and longitude given to us.

3.4.5 Visualize the Trip Duration given using log-scale Distplot in SeaBorn(After Removing Outliers)

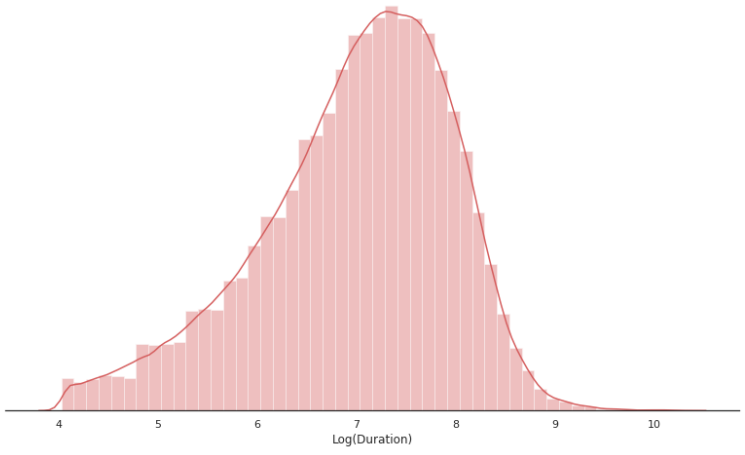


Figure 3.3: It is clear with the above histogram and kernel density plot that the trip durations are like Gaussian and few trips have very small duration while most of the trips are e^4 = 1 minute to e^9 120 minutes. Let's check the lat-long distributions are then used them to have a heat map kind of view of given lat-longs. Here We can see that the duration with 0 seconds are removed.

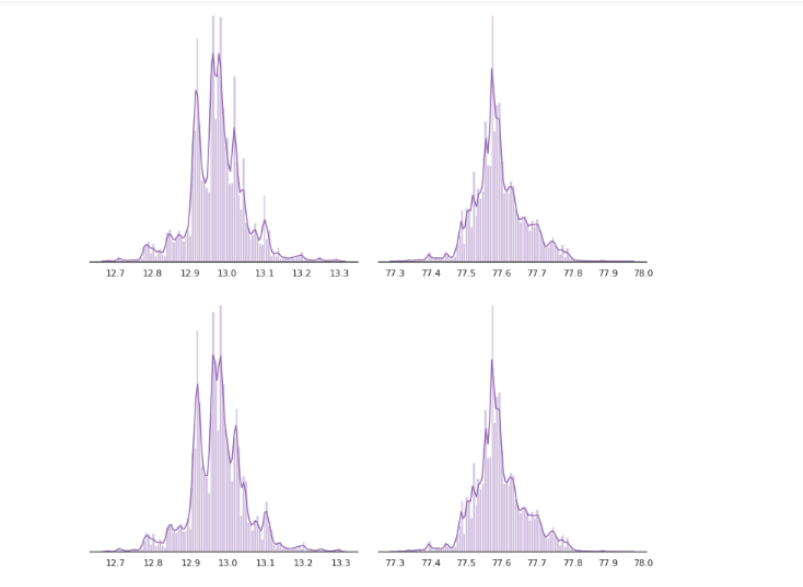
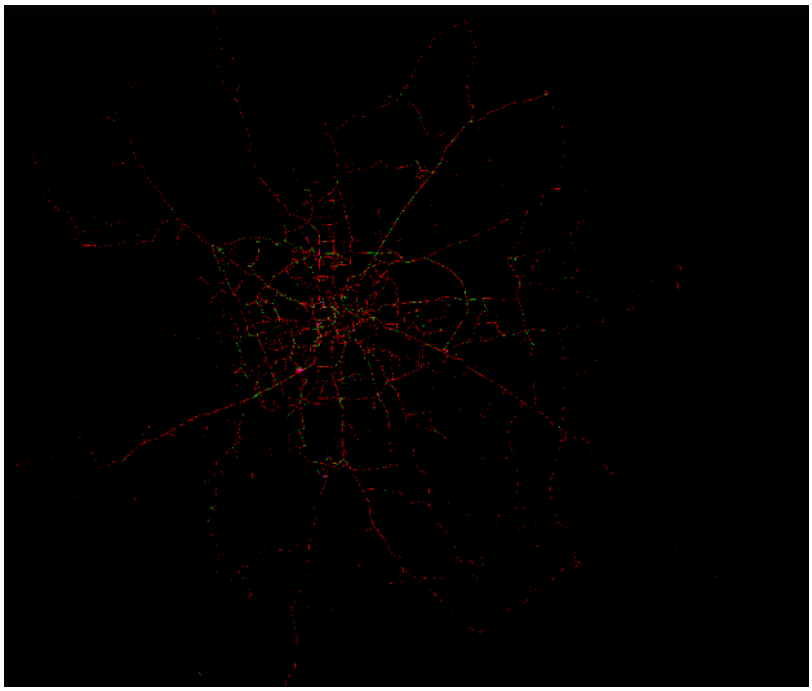


Figure 3.4: We put the following caps on lat-long: a) Latitude should be between 12.7 to 13.4 b) Longitude should be between -77.3 ton-77.8 .We get that the distribution spikes becomes as distribution in distplot (distplot is a histogram plot in seaborn package), we can see that most of the trips are getting concentrated between these lat-long only. Let's plot them on an empty image and check what kind of a city map we are getting as we can't use gmaps and folium on kaggle kernel for visualizations.



3.4.6 Features' Exploration (Checking if we have any explainable pattern)

Even if we don't visualize these features we can make model and model will predict the trip duration, then why are we visualizing? - Because - it will give us an explanation of model's output and will give us some pattern which may even guide if we should make multiple models, or one model if we should include that particular variable or there is no sense in including that variable. in short - it will give us new ideas to make a model.

Box-Plots

Interpretation

- Most popular plots to check the distribution of variables
- Box covers data from second and third quadrant and rest is shorn by bars
- Dots on the both side of bars shows outliers

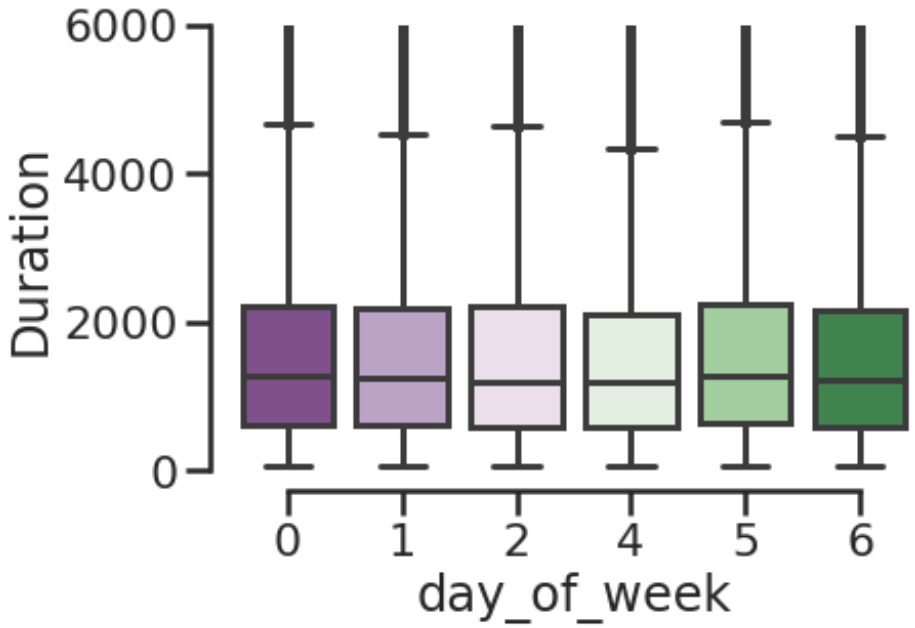


Figure 3.5: From the boxplot above we can see that 75%ile of trip takes around 2000 seconds. i.e. around 33 minutes

3.4.7 Feature Extraction

In machine learning, pattern recognition and in image processing, feature extraction starts from an initial set of measured data and builds derived values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations. Feature extraction is a dimensionality reduction process, where an initial set of raw variables is reduced to more manageable groups (features) for processing, while still accurately and completely describing the original data set. Here we extracted following features from the given data set.

Distance

Let's calculate the distance (km) between pickup and dropoff points. Currently Haversine is used. We could check the Manhattan (L1) distance too.

pd.DataFrame.apply() would be too slow so the haversine function is rewritten to handle arrays. We extraxt the middle of the path as a feature as well.

Lets create the "Neighborhoods"

One might think it necessary to have a map handy to do this, but not really. This will intuitively work as KMeans will cluster the data points into their own neighborhoods. This is pretty straight forward since Numpy helps create a vertically stacked array of the pickup and dropoff coordinates, and using 'sklearn''s MiniBatchKMeans module it's easy to set up the parameters to create the clusters.

There are three steps to preparing the data: create the coordinates stack, configure the KMeans clustering parameters, and create the actual clusters.

This shows as a trip would differ from point A to point B, in various parts of Banglore.

Date Extraction

Part of the reasoning behind extracting the different parts of the date for each trip is to enable us to do one hot encoding. This involves changing categorical variables into binary variables. It makes it easier to use when training ML models, since logically a machine can much better understand 1's and 0's rather than 'January' or 'Febuary', for example. To make sure we can use the features we extract from the dates, we need to check if both data sets has the same size (i.e. same number of months, days, hours, etc.).

Creating Dummy Variables

At this point it's possible to train our model, but often in Data Science and analysing different datasets, we need to look for alternative sources of data that will add accuracy to the models we build. And we still have the opportunity to create dummy variables that can add to model's accuracy. So for this step we get to the one hot encoding we spoke of earlier. Generally speaking you can do this in a few ways, but luckily Pandas helps us out again. A simple function that changes categorical data into dummy/indicator variables.

3.4.8 Visualization of Data after Feature Extraction

After feature extraction the following features are available now:

- ID
- Start Time
- End Time
- Start Latitude
- Start Longitude
- End Latitude
- End Longitude
- Duration
- pickup date
- pickup weekday
- pickup hour
- pickup month
- pickup day

- haversine distance
- pickup cluster
- dropoff cluster

Visualization of Distance Traveled By Buses

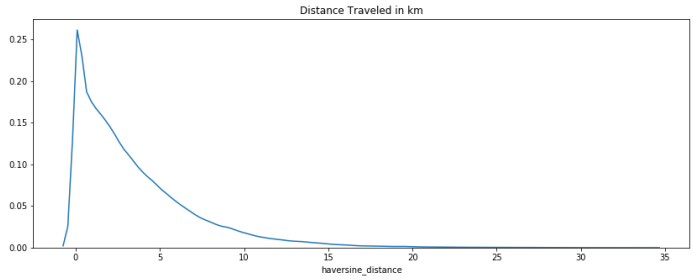


Figure 3.6: The above graph represents the haversine distance traveled by the buses during each trip.

Visualization of No. of trips per Day

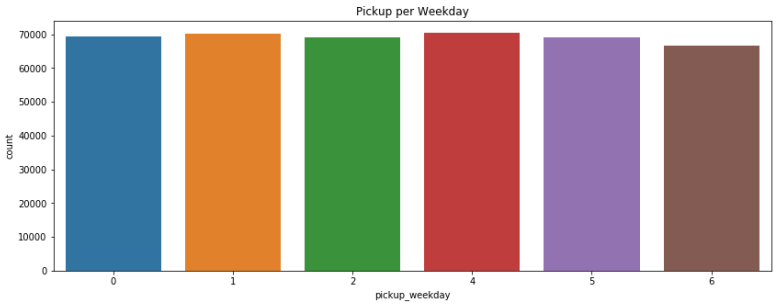


Figure 3.7: The above graph represents the No. of pick up(Here we assume that Start Latitude and Longitude of every trip is a pick up point) per day.

Visualization of No. of trips per Each hour(All Days)

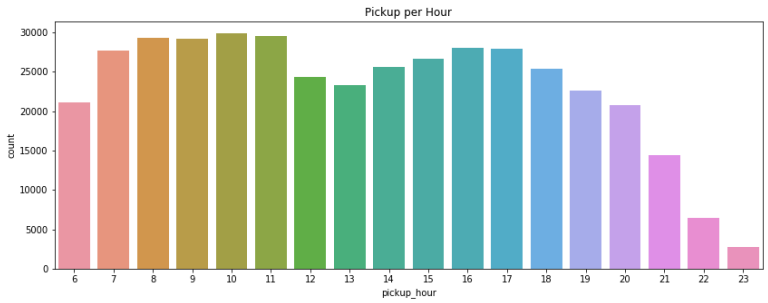


Figure 3.8: The above graph represents the No. of trips per each hour.

Visualization of Average Trip Duration(Day Wise)

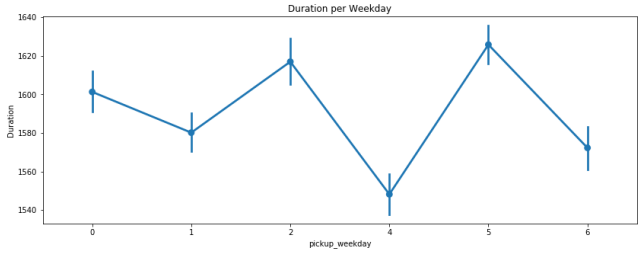


Figure 3.9: The above graph represents the average trip duration on a particular day.

Visualization of Average Trip Duration(Hour Wise)

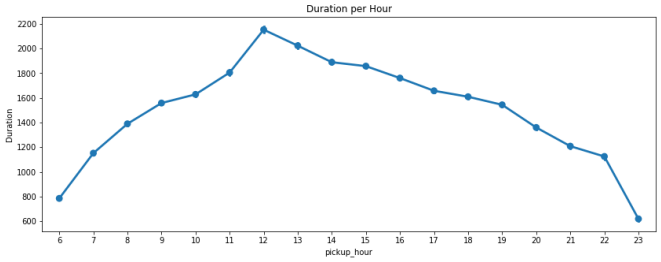


Figure 3.10: The above graph represents the average trip duration on a particular hour.

Correlation Matrix

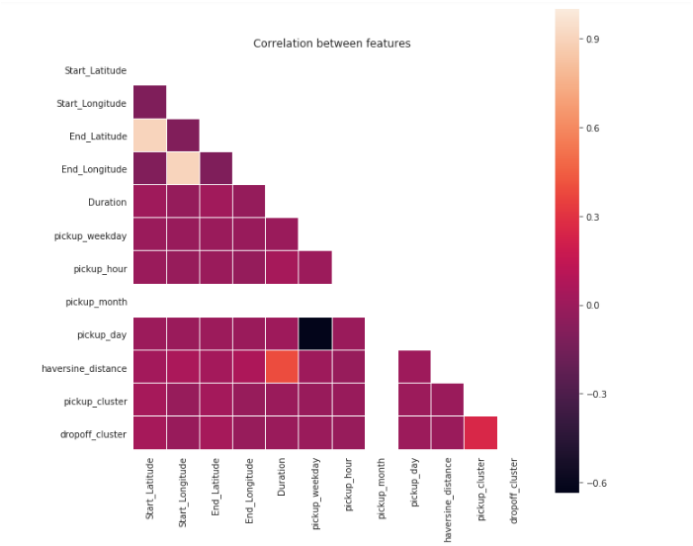


Figure 3.11: The above graph represents the correlation between various feature vectors.

Feature Importance

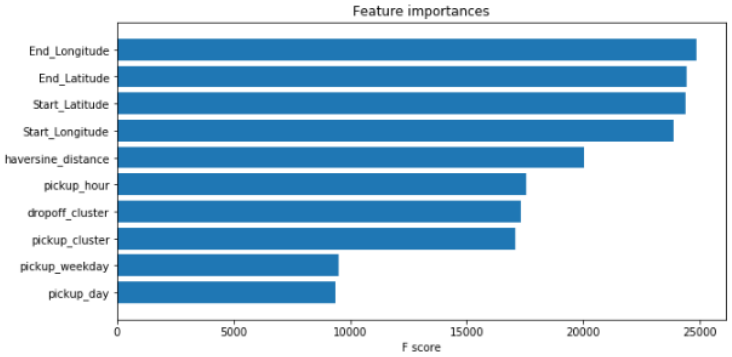


Figure 3.12: The above graph shows the importance of different features.

3.5 Software Use

There are a bunch of python libraries are available for Machine Learning. Here, we use numpy, pandas, matplotlib, seaborn and scikit learn etc.. libraries for preprocessing and visualization of data and for learning different models.

- NumPy: It provides an abundance of useful features for operations on n-arrays and matrices in Python.
- Pandas: Pandas is a Python package designed to do work with “labeled” and “relational” data simple and intuitive. Pandas is a perfect tool for data wrangling. It designed for quick and easy data manipulation, aggregation, and visualization.
- Matplotlib: matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy.
- Seaborn: Seaborn is mostly focused on the visualization of statistical models; such visualizations include heat maps, those that summarize the data but still depict the overall distributions. Seaborn is based on Matplotlib and highly dependent on that.
- SciKit-Learn: Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

4. Experimenting

There are various libraries available to implement machine learning methods. After research, the Scikit-learn library for Python seemed to be a more suitable option for this project and we try different regression models and analyze our model on different test data files to check accuracy of these models. The analysis includes measurable qualities, such as run time and memory requirements, but also less quantifiable properties such as ease of use. Near the end of this chapter we will discuss these results in the context of the prediction accuracy on test data. In the last section we will give our recommendations for these models.

4.1 Errors

To compare our regression results we have picked two methods for the error calculation.

4.1.1 Mean Squared Error

This measure uses the difference between the prediction Y_i , and the actual data y_i. Here $Y_i - y_i$ is the error. We have no knowledge about the value of the error, this can either be positive or negative, therefore we square every error and receive the squared error $(Y_i - y_i)^2$. Since we can have multiple data points to calculate the error over (we use time series so we can calculate te error over a longer period of time), we want to know something about the overall error. This over all error can be calculated by taking the mean of all errors in the made predictions.

The final measure we get is thus the Mean Squared Error:

$$MSE = \frac{1}{n}\Sigma_{i=1}^{n}\left(\frac{Y_i - y_i}{\sigma_i}\right)^2$$

4.1.2 Root Mean Squared Error

The RMSE is an extension of the MSE. Squaring the errors can be deceiving, big errors become exponentially large whilst errors below one become exponentially small. This draws errors further apart. To overcome this the RMSE can be used. This measure takes the root of the MSE which is not

exactly the same as the absolute error, but brings the errors closer together. The final measure we get is thus the Root Mean Squared Error:

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\left(\frac{Y_i - y_i}{\sigma_i}\right)^2}$$

4.2 Regression Models

In statistical modeling, regression analysis is a set of statistical processes for estimating the relationships among variables. It includes many techniques for modeling and analyzing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables (or 'predictors'). More specifically, regression analysis helps one understand how the typical value of the dependent variable (or 'criterion variable') changes when any one of the independent variables is varied, while the other independent variables are held fixed.

Most commonly, regression analysis estimates the conditional expectation of the dependent variable given the independent variables – that is, the average value of the dependent variable when the independent variables are fixed. Less commonly, the focus is on a quantile, or other location parameter of the conditional distribution of the dependent variable given the independent variables. In all cases, a function of the independent variables called the regression function is to be estimated. In regression analysis, it is also of interest to characterize the variation of the dependent variable around the prediction of the regression function using a probability distribution.

Regression analysis is widely used for prediction and forecasting, where its use has substantial overlap with the field of machine learning. Regression analysis is also used to understand which among the independent variables are related to the dependent variable, and to explore the forms of these relationships. In restricted circumstances, regression analysis can be used to infer causal relationships between the independent and dependent variables. However this can lead to illusions or false relationships, so caution is advisable.

4.2.1 Linear Regression

linear regression is a linear approach to modelling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). The case of one explanatory variable is called simple linear regression. For more than one explanatory variable, the process is called multiple linear regression.[1] This term is distinct from multivariate linear regression, where multiple correlated dependent variables are predicted, rather than a single scalar variable.

Linear regression was the first type of regression analysis to be studied rigorously, and to be used extensively in practical applications. This is because models which depend linearly on their unknown parameters are easier to fit than models which are non-linearly related to their parameters and because the statistical properties of the resulting estimators are easier to determine.

$classsklearn.linear_model.LinearRegression(fit_intercept = True, normalize = False, copy_X = True, n_jobs = None)$.

RMSE on test data :-

Public : 6023.0902

Private : 15004.49588

4.2.2 Random Forest Regressor

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True (default).

$classsklearn.ensemble.RandomForestRegressor(n_estimators='warn', criterion='mse', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False)$

RMSE on test data :-

Public : 5587.17456

Private : 15920.76183

4.2.3 Gradient Boosting Regressor

Gradient Boosting builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function. $classsklearn.ensemble.GradientBoostingRegressor(loss='ls', learning_rate=0.1, n_estimators=100, subsample=1.0, criterion='friedman_mse', min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_depth=3, min_impurity_decrease=0.0, min_impurity_split=None, init=None, random_state=None, max_features=None, alpha=0.9, verbose=0, max_leaf_nodes=None, warm_start=False, presort='auto', validation_fraction=0.1, n_iter_no_change=None, tol=0.0001)$

RMSE on test data :-

Public : 7396.5495

Private : 14441.13236

4.2.4 XG Boost

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. $classxgboost.XGBRegressor(max_depth=3, learning_rate=0.1, n_estimators=100, silent=True, objective='reg:linear', booster='gbtree', n_jobs=1, nthread=None, gamma=0, min_child_weight=1, max_delta_step=0, subsample=1, colsample_bytree=1, colsample_bylevel=1, reg_alpha=0, reg_lambda=1, scale_pos_weight=1, base_score=0.5, random_state=0, seed=None, missing=None, importance_type='gain', **kwargs)$

RMSE on test data :-

Public : 5981.57856

Private : 15293.45141

5. Conculsion

5.1

We explored the impact on the predictive performance of a simple local linear regression. We find that the additional features can lead to improved performance and we have also confirmed that local linear modelling is preferable to global linear modelling for the task of travel time prediction.

In our experiments we observe that rush hours and night time are more difficult for prediction. The errors during the night time are less important and we exclude it, since they are related to low traffic density and high variation in vehicle speed (free flow), which is not relevant to routing applications, since a static model of average speed based on link lengths is often sufficient when traffic is in free flow.

To further improve the accuracy of prediction, it can be considered using other models such as the linear model using link flow and occupancy. We can also try to consider other factors that affect travel time such as the traffic information for the crossing highways.

5.2 References

1. John Rice, Erik van Zwet, "A simple and effective method for predicting travel times on freeways", Proc. 4th IEEE Conference Intelligent Transportation Systems, pp. 229-234, 2001.
2.Xiaoyan Zhang, John. A. Rice, "Short-term travel time prediction", Transport Research, vol. 11, pp. 187-210, 2003.
3.Wu Chun-Hsin, Jan-Ming Ho, D. T. Lee, "Travel-time prediction with support vector regression", Intelligent Transportation Systems IEEE Transactions on, vol. 5.4, pp. 276-281, 2004.
4.KarelVerhoeven,NewYork city taxi travel time prediction,https://www.kaggle.com/karelrv/nyct-from-a-to-z-with-xgboost-tutorial,2017