BMTC Travel Time Prediction

Ayush Yadav
ayush.yadav@iiitb.org
IMT2017009

Kaustubh Nair
kaustubh.nair@iiitb.org
IMT2017025

Sarthak Khoche
sarthak.khoche@iiitb.org
IMT2017038

***Abstract*—Travel time information is a vital component of many intelligent transportation systems applications. In this context, advanced public transportation systems are one of the most important ITS applications, which can significantly improve the traffic situation in India. This needs a real-time data collection technique, a quick and reliable prediction technique to calculate the expected travel time based on real-time data and informing the passengers regarding the same.**

***Index Terms*—Machine Learning, Grids, Linear Regression.**

I. Introduction

The BMTC Travel Time Prediction problem is a Real-World problem, provided by Bangalore Metropolitan Transport Corporation to predict the travel time of the trips undertaken by their buses, in the city of Bangalore. We attempt to solve the travel time prediction problem by introducing a novel method of segment creation, which are then localized to small grid sections in Bangalore, thus trying to learn the trend of traffic using these segments in all different grid-sections.

II. Data

The BMTC dataset comprises of geo-location data for all of the 6000 BMTC buses over the course of 6 days that are operational in the Bangalore metropolitan area. The location of each bus was monitored with a 10 second interval. The training data set contained 214 million data points. Each data point consists of the following attributes.

Data Attributes of training data

- $BusID$: The unique ID of each bus
- LAT: Latitude of the given bus
- LON: Longitude of the given bus
- $Angle$: The orientation of the bus, defined against a base value.
- $Speed$: Speed of the bus in kmph.
- $Timestamp$: Timestamp at which these values were recorded.

The test data set consists of multiple data points, where each data point consists of the $BusID$, the starting time of the trip and 100 geo-locations (latitude and longitude values) which correspond to the route taken by the bus.

Data Attributes of testing data

- $BusID$: The ID of the bus undertaking the trip
- $Timestamp$: Starting timestamp the trip
- LAT_1: First latitude of the trip
- LON_1: First longitude of the trip

$\vdots$

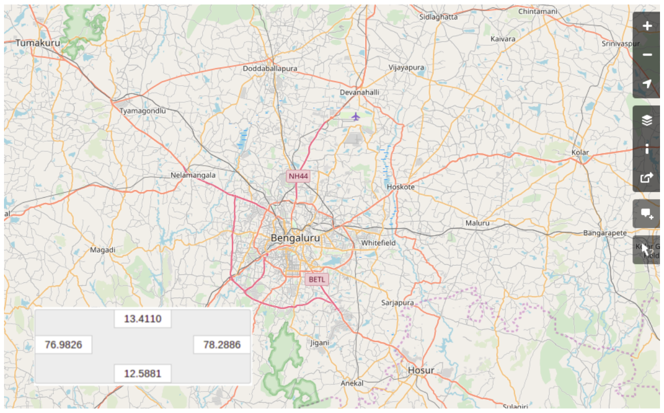

Fig. 1. Size of grid

- LAT_i: i^{th} latitude of the trip
- LON_i: i^{th} longitude of the trip

$\vdots$

- LAT_{100}: 100^{th} and the last latitude of the trip
- LON_{100}: 100^{th} and the last longitude of the trip

And our aim is to predict the trip duration for each of these data point given in the test dataset.

III. Exploratory Data Analysis

1) *Angle:* For a lot of buses, there are values that vary a lot even if the bus is not moving. For instance, it was observed that for some 10 consecutive rows, the LAT, LON values would remain same, but angle would keep changing. These faulty angle values can be found for about $\tilde{1}0\%$ of the dataset. To deal with this, the $Angle$ attribute was dropped and recalculated manually. (Further explained in *Feature Engineering*)

2) *Speed:* On plotting a histogram of the speeds of buses in the dataset, it was observed that majority of the values are 0kmph (*ref. Fig* 4), implying that either the bus is stationary at a bus stop or stuck in traffic. It was also noticed that for many datapoints the bus is not moving (i.e there is no difference in the LAT, LON) but the speed value of the bus is non-zero, this could be due to a lag in the transmission of speed values from the sensor. Also for a lot of buses, the values of speed are changing very slowly. In some cases, it can be found that a bus takes 120 seconds to decelerate from 20kmph to 0kmph.

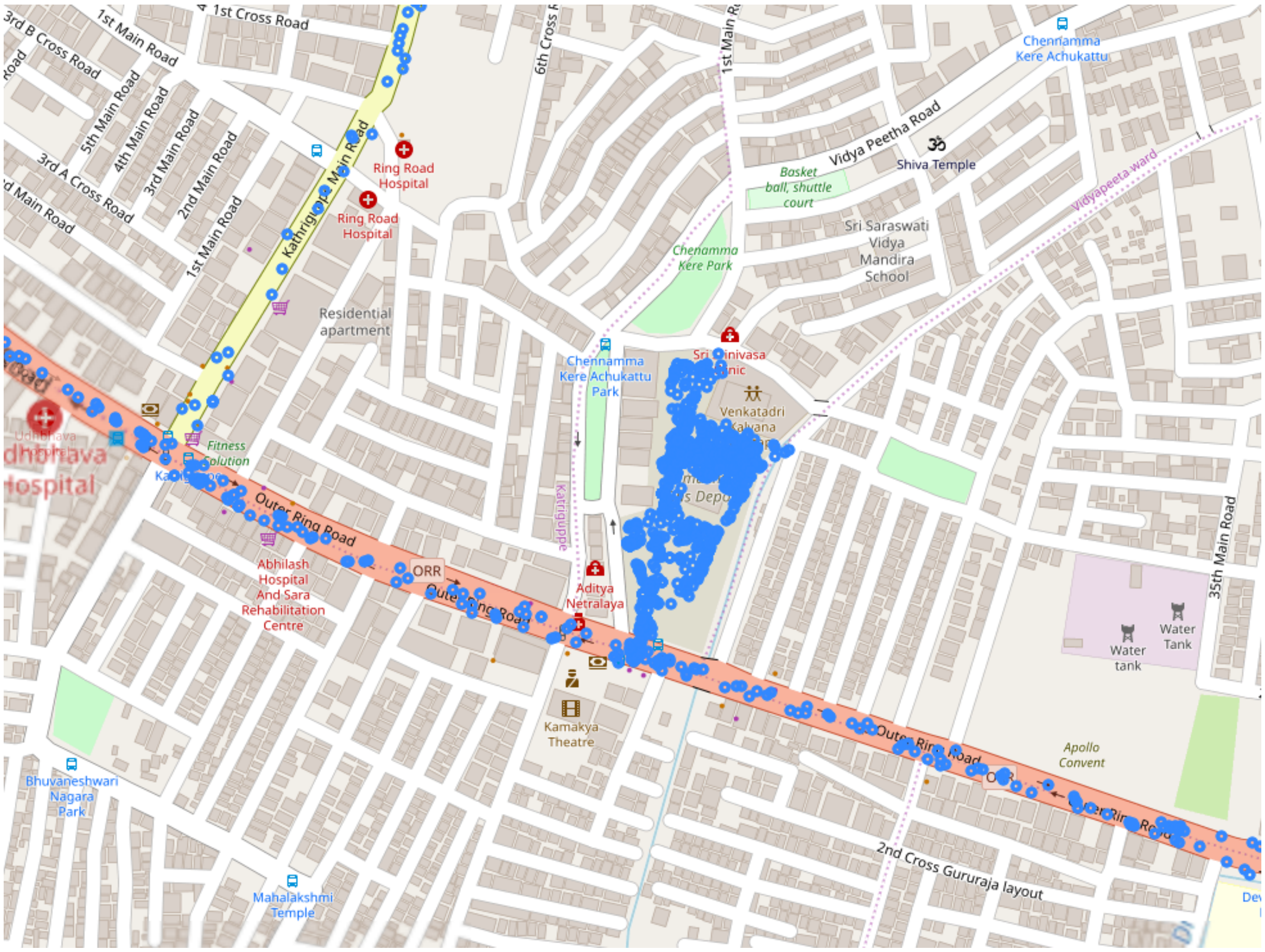
Fig. 2. Data points for a single bus concentrated around a bus depot.

Since this is highly unlikely, it is plausible that the speed sensors are not calibrated properly. To deal with this, the speed attribute was dropped and recalculated manually. (Further explained in *Feature Engineering*)

3) *Points around bus stops:* Even though there are points which span entire city, there are a lot data points that are heavily concentrated around popular bus stops in Bangalore. (Refer Fig. 2.). Moreover, a lot of these points have timestamps between 2 A.M. and 6 A.M., implying the GPS sensors were left functioning during the night. These points would lead to significant errors while modeling, since halt time for buses would not follow any set pattern.

4) *Inconsistent bus routes:* Taking inspiration from R.P.S. Padmanaban et.al.[8] we hypothesized every Bus ID will follow a specific route everyday. If this was true, we could have trained a model for every $BusID$, thus modeling information for every route separately. But after plotting routes for each day (Refer Fig. 3.), it can be seen that a bus does not travel on the same route for different days. To deal with this, we created multiple gridcells for modeling localized traffic conditions. (Further explained in *Model and Training*) Since, the $BusID$ attribute does not convey any kind of useful information. Therefore, the BusID attribute was subsequently dropped. This leads to further inaccuracies since, due to absent route information, it cannot be predicted whether a bus takes any special kind of road (For example, a flyover or service road) which will considerably affect travel time. But a solution for this issue is proposed in *Future Works*.

5) *Jagged datapoints:* The bus does not consistently move in a straight path. It keeps switching lanes acutely which leads to errors in angle. Also due to GPS inaccuracies, some points have incorrect values that do not fall in the path properly(Refer Fig. 5.). This can be fixed with an appropriate filtering algorithm such Kalman Filter

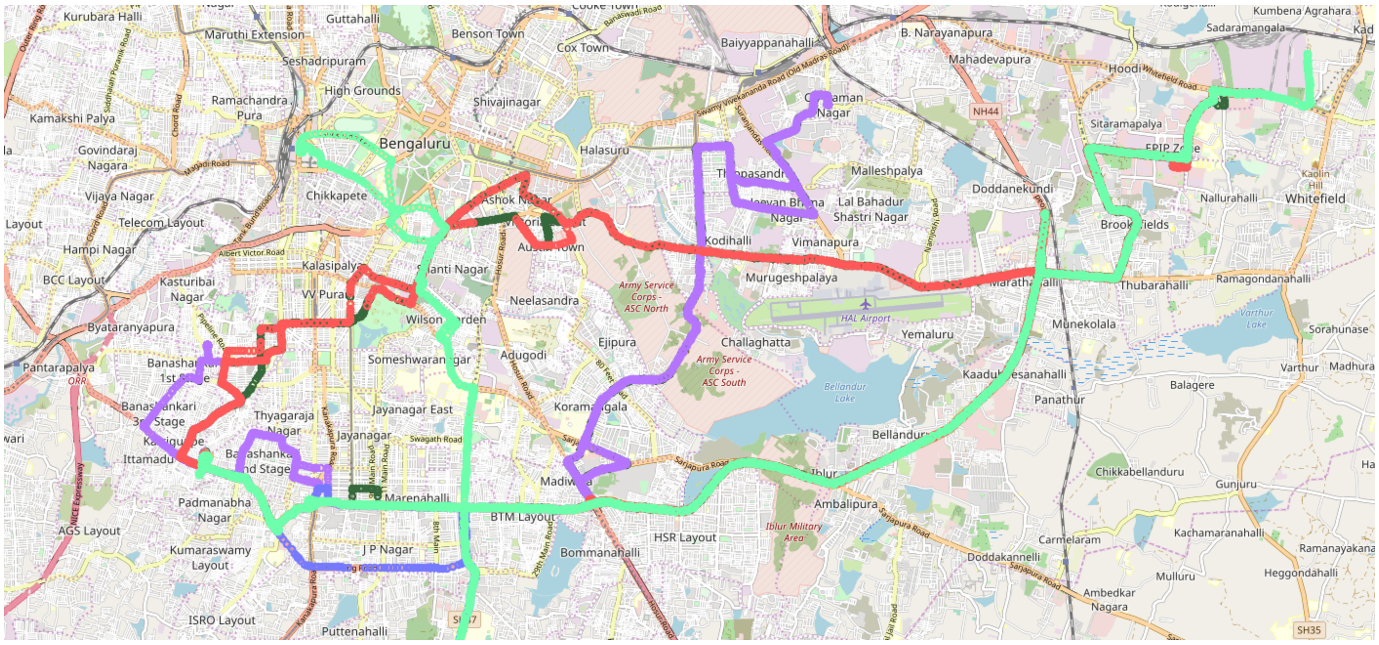

Fig. 3. Routes for a single $BusID$ where each color represents a different day of the week.

or Linear Least Squares fit[10]. (Further explained in *Future Works*)

6) *Data Point Density:* On plotting the entire dataset of points on the map of Bangalore city, it was seen that a large majority of the datapoints belong to central bangalore, thus giving us dense grid-cells (further explained in section VI) around in central bangalore and non dense gridcells otherwise.

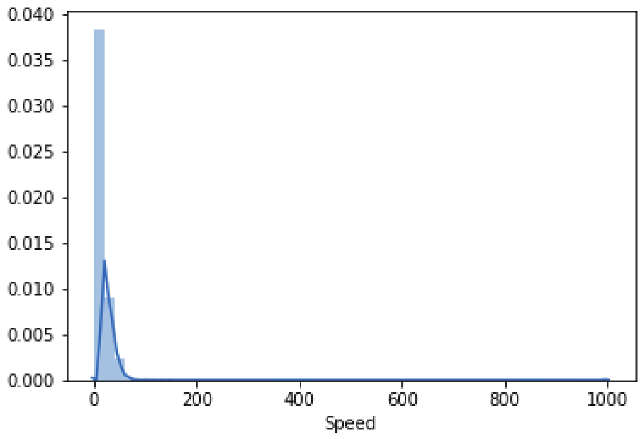

Fig. 4. Distribution of given speed values

IV. Preprocessing

The following preprocessing techniques were implemented on the training dataset.

1) *Removing outliers:* The following outliers in the data were removed.
   - *Coordinates:* Since the dataset consists of data points only in Bangalore city, we calculated a bounding box of latitude longitude values (LAT: 12.8 - 13.4, LON: 76.98 - 78.28) using an online tool[4] . All points which have latitude and longitude values outside this box were deleted under the assumption that they came from faulty sensor values. (Refer Fig. 1.)
   - *Speed:* All rows with speed values greater than 200 are absurd for BMTC buses. These rows were assumed to be corrupt and were subsequently deleted.
   - *Angle:* All rows in which the angle is not in the range [0, 360] were removed on account of the sensors transmitting corrupt data.

2) *Sorting:* The raw data was sorted by $Timestamp$. But since we required data based on the occurrence of routes, a nested sorting was done: first by $BusID$ and then by $Timestamp$ for each $BusID$. This method sorted the data on the basis of the route for every $BusID$, i.e consecutive points for a given $BusID$ represent the path taken by the Bus at the given time .

3) *Removing duplicate data points:* The dataset consists of multiple rows which have the same LAT, LON, Speed and Angle values, but have different timestamps. These rows imply that a bus is either stuck in traffic, or halting at a bus stop. All such duplicate rows, except for the first and last duplicate row were removed. The first row

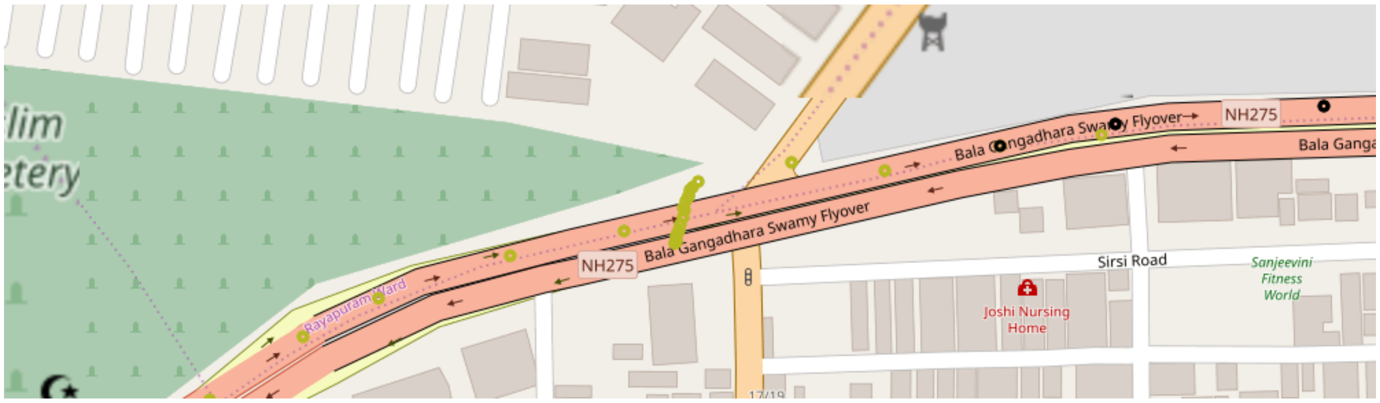

Fig. 5. Inaccuracies in GPS location.

(which contains the start time of the halt period) and last row (which contains the end time of the halt period) were retained because they convey information about the halting time of the bus.

V. Feature Engineering

1) *Creating segments:* Every row in the dataset was converted from a point, to a segment. For doing this, we took every pair of consecutive rows in the sorted dataset for a given BusId and appended the second datapoint to the first datapoint. For every such pair of datapoints, we created a new feature, for which the attributes were as follows:
   - LAT1 (the latitude of the first datapoint)
   - LON1 (the longitude of the first data point)
   - LAT2 (the latitude of the second data point)
   - LON2 (the longitude of the second data point)
   - Speed (elaborated in section 5.5)
   - Duration (elaborated in section 5.3)
   - Distance (elaborated in section 5.2)
   - Direction (one hot encoded, elaborated in section 5.4)
2) *Calculating Distance and Angle:* Based on the tuples created, the distance and angle between each pair of coordinates was computed. The estimation for distance(in km) was based on the Haversine Formula[5] (Refer Fig. 6), instead of using the conventional Euclidean method, since the Haversine Formula gives us a more accurate measure by considering the curvature of the earth. Similar considerations were taken for calculating the direction where the bus is heading[6].

3) *Calculating Time and Duration:* The $Timestamp$ attribute present in the dataset is a string, according to the UTC Date-Time Format[7]. This was split into two attributes for Day which represents the day of the week, and $Time$ which represents the time at the specific day. Day is an integer between 0-6 and $Time$ is expressed in seconds.
   This was done for both points in the segment, which presented the $start\_time$ and end_time of the segment. These two were subtracted to obtain the $duration$ representing the time taken to travel through each segment.

4) *Calculating Direction:* Based on the angle calculated between the pair of coordinates, it was mapped to 4 direction: $East, West, North, South$. These directions were encoded using One Hot Encoding. The east-direction was assigned a value 1 if the angle came out to be in the range [0, 45] or [315, 360]. Similar computations were done for rest of directions as well.

5) *Calculating Speed:* Based on the duration predicted for every segment and the length of every segment, the speed with which the bus moved on the segment was calculated, based on the simple formula : speed = distance/time.

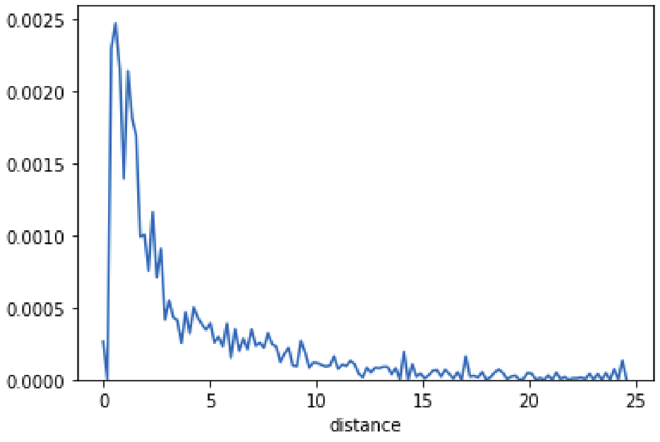

Fig. 6. Distribution of distance between two points

VI. Model and Training

A. Method 1 (implemented)

Dividing Bangalore in grids : Due to the complex nature of traffic in different parts of Bangalore, it will be ineffective to train a single model for the entire city, as a single regression

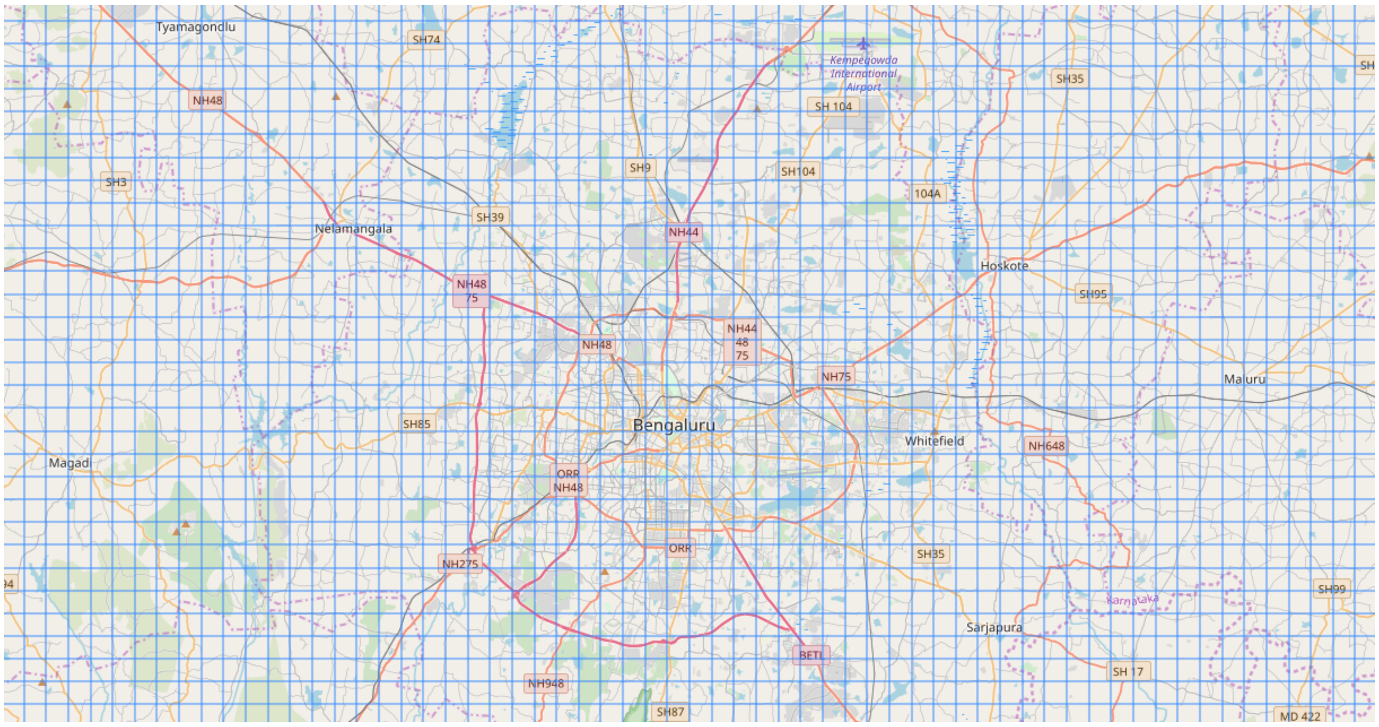

Fig. 7. Bangalore divided into small grid cells for modeling traffic better.

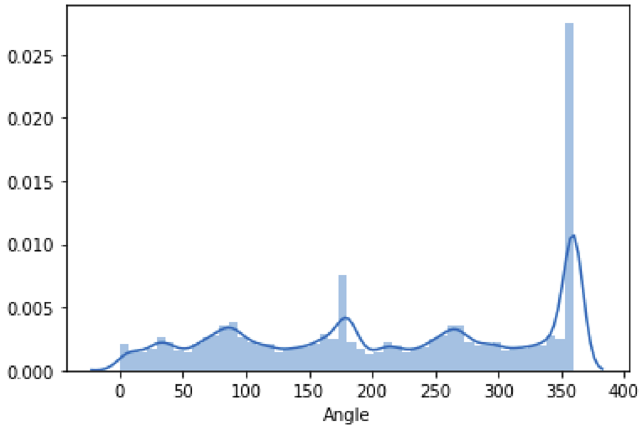

Fig. 8. Distribution of calculated angle

model would be incapable of capturing the erratic trend of the traffic conditions throughout Bangalore. Hence, a grid based approach was followed to train different models. The city of Bangalore was split into around 5000 gridcells of size 1.5km $\times$ 1.5km each.

Initially, we decided to split Bangalore into small grids of dimension 500m $\times$ 500m. But that resulted in each gridcells having an average of around 8000 points which is very low. Additionally, since the data is clustered around central Bangalore the number of datapoints per grid in grids not around central Bangalore were around 2000. Thus, to have a big enough dataset to train on we decided to try the model the trend with 1.5km width grids. (Refer Fig. 7)

Implementation of gridcells: Each gridcell was indexed by a tuple (i, j) where the bottom left grid corresponds to $(0, 0)$

The increase in latitude corresponding to 1.5km ,

$$\Delta_{\text{LAT}} = 0.013491$$

Similarly, for longitude,

$$\Delta_{\text{LON}} = 0.013824$$

The coordinate of the bottom left corner of the grid are $(12.6, 76.98)$. (This was calculated while *Removing outliers* in *Preprocessing*)

The index (i, j) of a point k can be calculated by

$$i = \left\lfloor \frac{LAT_{\text{k}} - 12.6}{\Delta_{\text{LAT}}} \right\rfloor$$

$$j = \left\lfloor \frac{LON_{\text{k}} - 76.98}{\Delta_{\text{LON}}} \right\rfloor$$

This resulted in the creation of around 5192 unique grids which are labeled as (i, j)s. Since a lot of gridcells (which represent the corners of the Bangalore metropolitan area) had no points, theses grids were discarded (i.e those (i, j) values were not trained for), this left us with

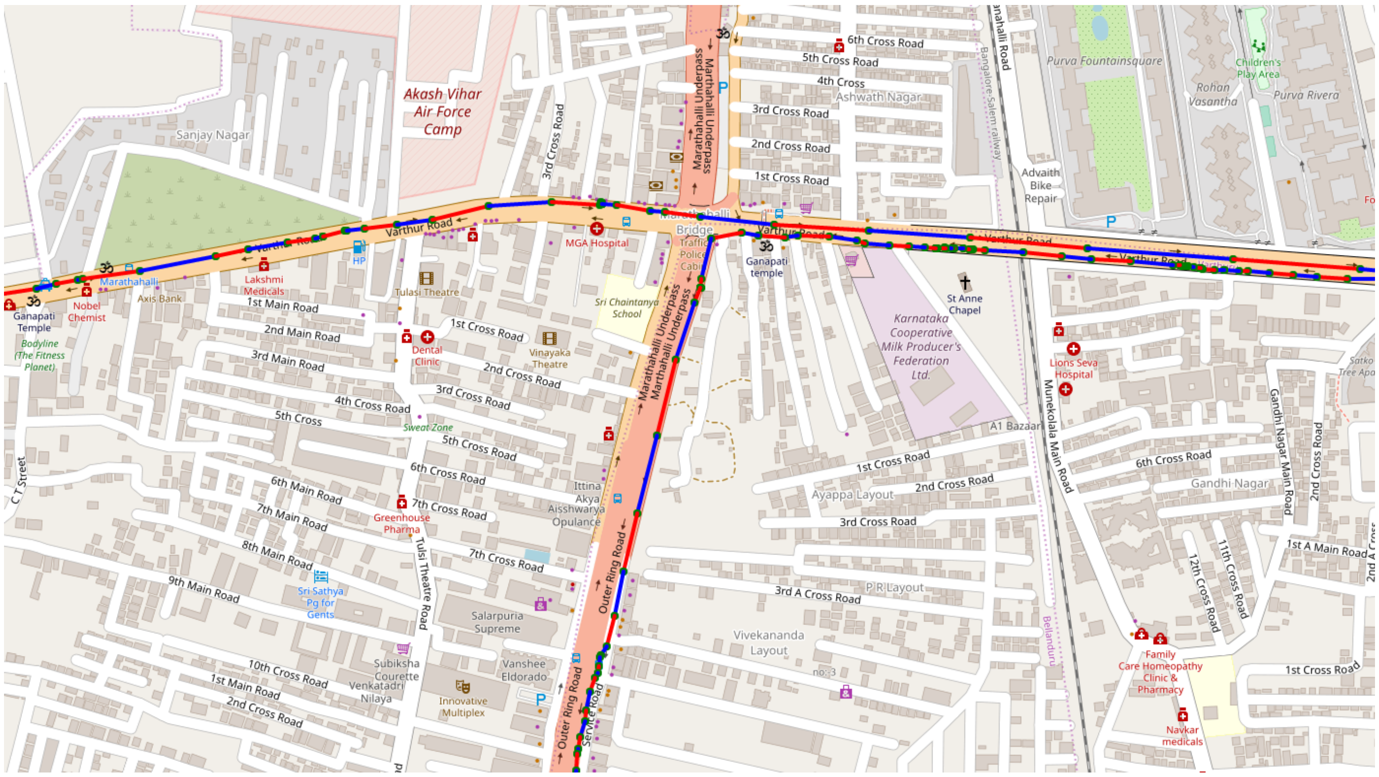

Fig. 9. Points converted to segments. Each segment is colored alternately for better visibility

around 1800 gridcells which had sufficient datapoints in them.

Assigning a segment to a gridcell: After feature engineering, the dataset contains segments (each of which contains 2 LAT-LON values, representing the start and the end of a segment) instead of single pair of LAT-LON values, hence segregating segments to grids is a non-trivial task. This was done using the following method:

- *Start and end of a segment lie in the same segment:* Segregating segments to gridcells is a trivial task if the segment begins and ends in the same grid. The segment is assigned the gridcell number in which it resides. (refer fig. 7)

- *Start and end of the segment lies in different segments:* If the start of a segment lies in one grid and it's end lies in another grid, we approximated and split the given segment into two segments, where each segment lies in one of the given two grids.

  Segment lies in two horizontally/vertically adjacent segments: For such a case, a point of intersection of the segment with the edge of the grid is considered to be the "$breakpoint$". Using this $breakpoint$, the existing segment is broken into two segments, where the $breakpoint$ is the ending point of one segment, and also the starting point of the other segment. Duration for both these segment is calculated by taking the ratio of the lengths of the two new segments and dividing the duration in the same ratio.

  Segment lies in two diagonally adjacent segments: For this case, the common corner between the two diagonally adjacent grids is considered to be the $breakpoint$. Using this $breakpoint$, the existing segment is approximated to two segments, where the $breakpoint$ is the ending point for one segment, and the starting point for another segment. Duration assignment is the same as that in the above case.

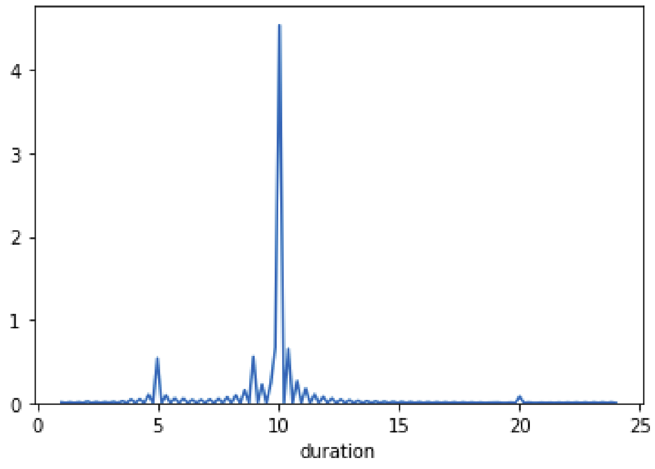

Fig. 10. Distribution of duration of each segment

The idea of splitting a segment works under the condition that the start and end point of the segment lie in adjacent grids, (i.e the distance between the starting and the ending point of the segment is less than 3 kms). As seen in section 3 (EDA) the length of a segment is on average 5 or 6 meters, and thus works well with our assumption.

Sorting and Segregating : After the grid-indexes (i, j) were computed, the entire dataset was sorted based on these (i, j) values, so that all the data points corresponding to a particular grid are consecutive to each other.

Training a model for each Gridcell : After sorting and segregating, a model was trained for every gridcell (i, j). The training points for each gridcell were the segments that lie in the gridcell. The parameters for each model are:

- $LAT1$: Latitude of start of segment
- $LON1$: Longitude of start of segment
- $LAT2$: Latitude of end of segment
- $LON2$: Longitude of end of segment
- $Day1$: Day of segment
- $Time1$: Time in minutes of starting point
- $distance$: Length of segment
- $Direction - E$: Boolean attribute for East direction
- $Direction - W$: Boolean attribute for West direction
- $Direction - N$: Boolean attribute for North direction
- $Direction - S$: Boolean attribute for South direction

Based on these features, a linear regression model and a random forest model was trained for every gridcell on these features, to predict the average speed for each gridcell. The model parameters for each model were stored in a pickle file, so that it can be accessed later while testing.

The pickle file for the linear regressor can be found on this link

VII. Testing

The testing data which was indexed by BusIDs contained a set of 100 consecutive Latitudes and Longitudes, which were to be considered as the path taken by that Bus and hence we were supposed to predict duration for each trip.

Based on the coordinates of the first set of Latitude and Longitude, the corresponding grid to which the coordinates belonged was found and the model corresponding to the grid was retrieved from the pickle file. The set of coordinates along with timestamp and direction were passed as features to the model and average speed was predicted. Based on the speed and distance calculated between the points suing Haversine[5], the duration was calculated as : duration = distance / avg speed. This calculated duration was added to the timestamp which was again passed as an input to some other model based on its grid number.

This procedure was done for all 100 coordinates and the final duration which was the sum of all individual duration indexed using BusID was stored and submitted as *sub.csv* ‘

VIII. Results

The table below shows the models that we used along with their RMSE values.

Model	**RMSE**
Linear Regression	15436.41918
Random Forest	8433.51998

IX. Future Work

1) Sub-dividing the grids: The size of the considered grid is too large. Better results can be achieved with smaller grids. The disadvantage of using smaller grids is that the number of training points per grid is very less for small grids. To solve this issue a dynamic grid width can be defined, such that for a given grid-cell (i,j) of width 1.5km, if there are significantly many datapoints inside the grid-cell, then the gridcell can be further divided into 4 gridcell, thus changing the gridcell numbering system to (i, j, k). Here k represents the index of the smaller grids inside the bigger grid. A 0 index can represent that the grid has not been divided into smaller grids.
This would be a significant help, since the traffic trends of a 500m wide grid would be far more significant than those of a 2km wide grid
2) Clustering for flyovers: In the instances of gridcells having a flyover, we might get a large variation in the average speed for that grid, primarily segregated into two clusters in a high dimension space. This might be because of the fact the average speed above and below the flyover are marginally different. The idea is to cluster all the datapoints in clusters such that each cluster gives a different average speed. The cluster with a lower average speed will correspond to all the points below the flyover, whereas the cluster with higher average speed will correspond to the points above the flyover. Hence, for a training point in this gridcell, we need to find the cluster it belongs to, based on the data we have and then using the model corresponding to that cluster to predict the speed and then the duration.
3) *Modify the segments passed for training:* The average length of segments in the training set as soon in the preprocessing section is shown to be around 20m, but the average segment distance in the testing dataset is around 1800m. This difference in test and train data is a possible reason for 8433.51998 RMSE value on the testing data.

   Since all consecutive segments create a path, several segments can be merged together to create larger segments. Now, with segments of length greater than 1500m, possibly better results can be obtained. A downside to this approach would be the the linear estimation of 1500m paths, i.e all paths are approximated to straight lines, and though this assumption may be valid for 20-50m distance, there will be loss of information when approximated for 1500m paths.

4) *Kalman Filter or Linear Least Squares fit:* These filters can be used to smoothen the path taken by the bus, by smoothing the values sent by the GPS sensors on the Bus.[9, 10]

Acknowledgment

We sincerely thank Prof G Srinivasa Raghavan for his guidance and support. Discussions with Prof G Srinivasa Raghavan's always left us with new ideas and new thoughts on how to tackle the problem. We would also like to thank our teaching assistant Vaishali for her valuable time and encouragement.

References

[1] https://scikit-learn.org/stable/documentation.html
[2] https://pandas.pydata.org/pandas-docs/stable/
[3] https://matplotlib.org/3.1.1/contents.html
[4] https://www.openstreetmap.org/
[5] https://en.wikipedia.org/wiki/Haversine_formula
[6] https://www.igismap.com/formula-to-find-bearing-or-heading-angle-between-two-points-latitude-longitude/
[7] https://www.w3.org/TR/NOTE-datetime
[8] Development of a real-time bus arrival prediction system for Indian traffic conditions: (https://ieeexplore.ieee.org/document/5558887)
[9] https://en.wikipedia.org/wiki/Kalman_filter
[10] Jakkaraju Pavan Kumar, Namit Rarotra, Uma Maheswari. Design and Implementation of Kalman Filter for GPS Receivers