Road Traffic Prediction and Analysis

Rakesh Muppala	**Rishabh Agarwal**	**Shubham Bansal**	**Shubham Dua**
(rmuppal)	(ragarwa9)	(sbansal6)	(sdua2)
Group P38	Group P38	Group P38	Group P38
NC State University	NC State University	NC State University	NC State University
rmuppal@ncsu.edu	ragarwa9@ncsu.edu	sbansal6@ncsu.edu	sdua2@ncsu.edu

1 Background

1.1 Problem

Finding a way to dodge or avoid traffic is one of the most annoying things that we, as human beings, have to go through on a daily basis. On average, an American commuter wastes 54 extra hours a year because of traffic delays. The use of navigation apps to help find the fastest possible route has grown exponentially; almost 77% of smart-phone owners use navigation apps.

For this project we are going to find traffic trends and make predictions according to those trends for all the states across the United States. We had two main project tasks - the first was to predict whether a given station lies on an urban road or a rural road, and the second task was to calculate the hourly traffic volume given the geographic coordinates.

1.2 Literature Survey

A lot of research is currently being done on analyzing and predicting traffic trends. [1] uses a multilayer perceptron (MLP) and ARIMA models as the baseline models, and build a hybrid model H-ARIMA for traffic trends prediction.[3] investigates the factors that have a significant impact on the forecasting accuracy of traffic per hour using a non-linear time series traffic prediction model. [4] uses classical statistical models like feature-based models, Gaussian process models, and state-space models and deep learning models like CNNs and RNNs.

2 Methods

2.0.1 Support Vector Machines (SVM)

SVM's are a set of supervised learning algorithms; they are very widely used because they can be used for classification tasks, regression tasks, or even outlier detection. For binary classification, SVM's draw a decision boundary (called hyperplane) between the two classes in order to classify them. Figure 1 shows a simple diagram of an SVM model.

2.0.2 K-Nearest Neighbors (KNN)

KNN is a supervised learning algorithm. It's a very simple learning algorithm (also called a lazy algorithm). To implement a KNN model, we calculate the distance between the point whose class we want to find with the rest of the data points. The KNN model makes the classification based on the K-nearest neighbors of that particular data point. Figure 2 shows a simple diagram of a KNN model.

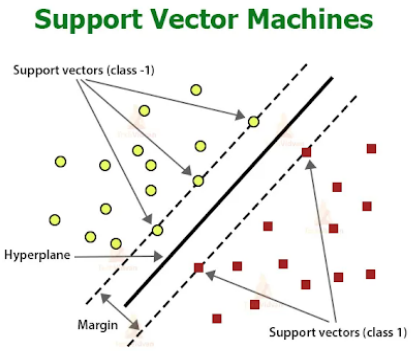

Figure 1: SVM

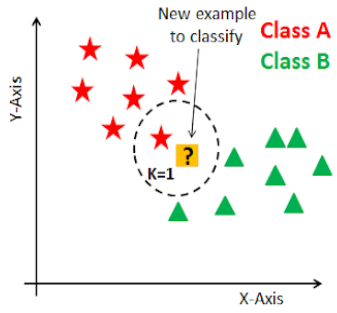

Figure 2: KNN

2.0.3 Naive Bayes

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. There is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. Figure 3 shows a simple diagram of a Naive Bayes model.

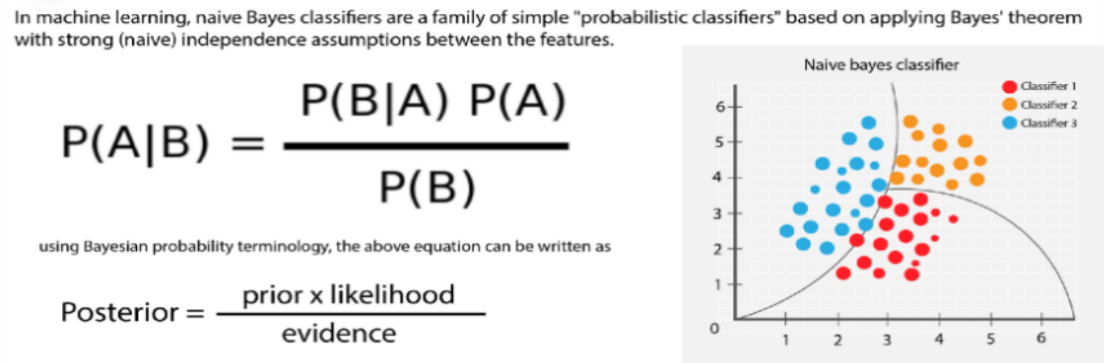

Figure 3: Naive Bayes

2.0.4 Decision Trees

A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements. Figure 4 shows a simple diagram of a Decision Tree model.

2.0.5 Long Short Term Memory (LSTM)

LSTMs are a popular version of RNNs. They were introduced as a solution to the vanishing gradient problem faced by RNNs. That is, if the previous state that is influencing the current prediction is not in the recent past, the RNN model may not be able to accurately predict the current state. To remedy this, LSTMs have “cells” in the hidden layers of the neural network, which have three gates–an input gate, an output gate, and a forget gate. These gates control the flow of information which is needed to predict the output in the network. Figure 5 shows a simple diagram of a LSTM model.

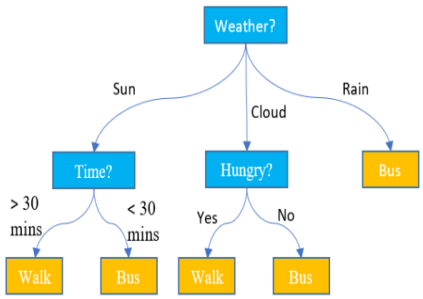

Figure 4: DT

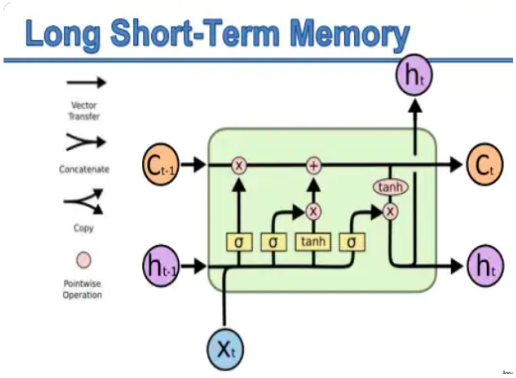

Figure 5: LSTM

2.1 Rationale

Upon extensive research, we found most of the traffic management system applications focus more on what route to take. However, our focus on this project was on hourly traffic volume and its analysis. For the first task, we used classification techniques to find if a road belongs to a rural or an urban setting, which is unique. For the second task, we used an LSTM to predict hourly traffic volume for traffic management purposes. The data for this task is time-series data, and that's why we use an LSTM model.

Figure 6 below is a graph we plotted showing the time series data (and this is why we used an LSTM). This graph shows the traffic volume for the first six hours of the day. Because all the entries of the first six hours show a similar pattern, we used an LSTM.

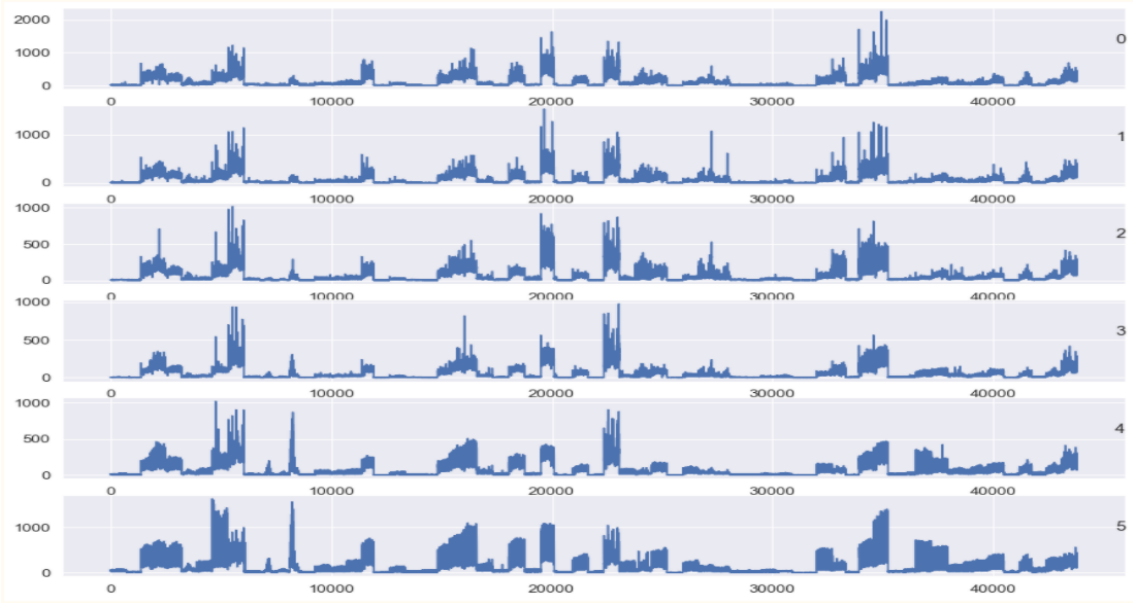

Figure 6: Patterns in data

3 Plan & Experiment

3.1 Dataset

The dataset used is the US Traffic, 2015. The dataset can be found at `https://www.kaggle.com/jboysen/us-traffic-2015`. The dataset consists of 7.1 million observations by hour and direction for different stations acroos the country for the year 2015. There are two files in the dataset.

- The first file contains the daily traffic volume counts for the 24 hourly bins along with station_id, date, traffic direction and type of road.
- The second file contains deeper location(latitude and longitude coordinates) and historical data on individual observation stations, cross-referenced by station_id.

This dataset was compiled by the US Department of Transportation and available on Google BigQuery.

Figure 7 below is a heatmap plotted between the hourly traffic and the day of the week.

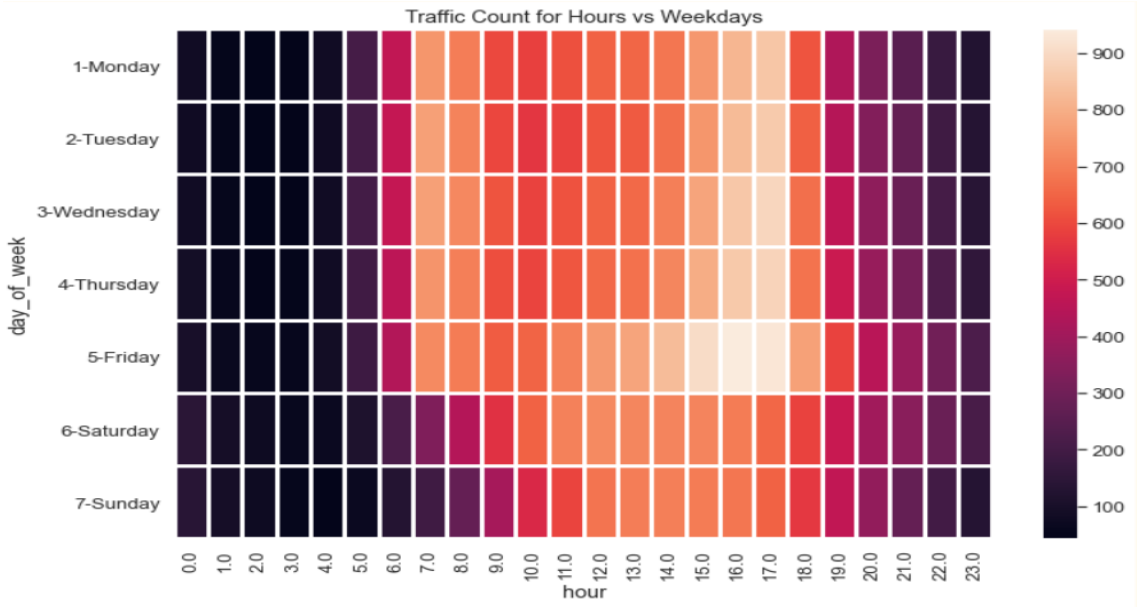

Figure 7: Heatmap - Day of week vs hour

3.2 Hypotheses

The main questions we tried to address are:

- Where are the heaviest traffic volumes based on time of the day, type of road and day of the week?
- What are the seasonal patterns observed in traffic data?

We usually expect the traffic count to be higher in cities and downtown areas and the traffic count to be higher in weekdays compared to the weekends because of a lot of people commuting to get to work. To learn about this, we tried visualizing a heat map to identify the traffic trends for the different weekdays. We tried a classification task to identify the roads as an urban area road or a rural area road using the hourly traffic counts, latitude, latitude and the state as the input attributes.

We thought that the traffic count at a given station for a particular hour depends on the traffic count of the previous few hours. To analyze this, we tried to create a regression model to predict the traffic count for a given hour based on the previous few hours, location, date and direction of travel.

3.3 Experimental Design

3.3.1 Dataset preparation & Pre-processing

The first dataset has 38 columns - 38 input attributes for our binary classification task. We used data preprocessing to reduce the dimensionality of the dataset by dropping 11 attributes from the first dataset (that were not affecting our predictions much). These are the 11 attributes dropped after preprocessing - the day the data was recorded, the functional classification name, the direction of travel (North, South, East, West), the record type, the restrictions, the year, date, and month the data was recorded, the day of the week, the direction of travel name, and the lane of travel.

For the next step, we grouped the data together based on three attributes - the name of the state, the station ID, and the functional classification. The 'functional classification' attribute gives information about the type of the road, whether it is a rural road, an urban road, a state highway, an inter-state highway etc. The remaining 24 input attributes are the traffic volume attributes (one for each hour of the day). After grouping the data according to the three attributes above, we took the mean of the hourly traffic volume attributes to maintain the homogeneity of the dataset.

We were only interested in five attributes from the second dataset - the name of the state, the station ID, the latitude and longitude for that station ID, and whether the given road is a highway or not. We merged the two datasets using an inner join on two attributes - the name of the state and the station ID. The attribute to check if a given road is a highway or not is the output attribute (the attribute to be predicted) for the binary classification task. The aim to combine the two datasets was just to map each station ID with its corresponding latitude and longitude coordinates. After removing all the duplicate entries, we have 6904 records as the data we gave to train our models. The input data (now consisting of 27 attributes) was normalized to transform the data points between the values of 0 and 1.

3.3.2 Classification

After preprocessing, we used four simple binary classification models - SVM, KNN, Naive Bayes and Decision trees - to calculate the accuracy of both the models.

We tested different kinds of splits to split the data. We used 80-20 split with 80 percent training data and 20 percent test data. We included a validation set to increase the accuracy of both these classification models where we split the data as 70 percent training data, 15 percent validation data and 15 percent test data. Then we also used Cross-Validation with different number of folds to train our models.

3.3.3 Regression

For the second task, entries for the state of North Carolina were used. The records were sorted based on stations followed by date and time. For each record, 24 new rows were created with each hourly count as the y label and the previous 4 hour traffic counts as input attributes. Other attributes such as location, date, day of week and direction of travel were also used as input attributes. Using this data, an LSTM model is used to predict the hourly traffic count for a given inputs.

4 Results

4.1 Task 1

We split the data three ways - for the first split, the data consisted of 80% training data and 20% test data; for the second split was 70% training data, 15% validation data, and the remaining 15% to be

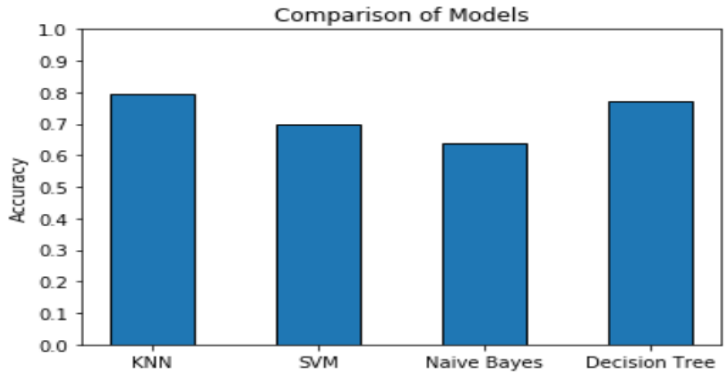

Figure 8: Train-test split

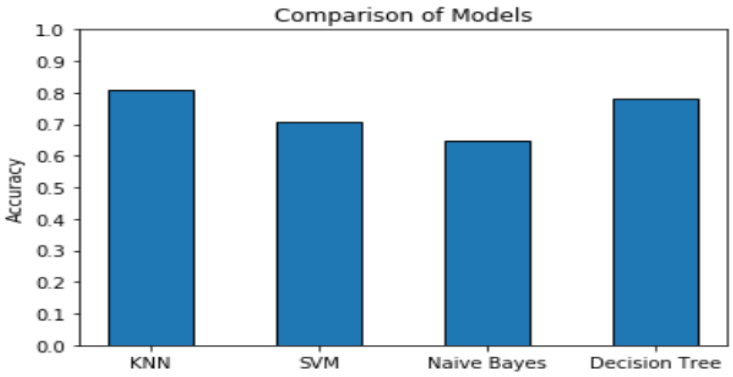

Figure 9: Train-test-validation split

test data; for the third split, we used K-fold cross-validation (ranging from 2 to 10 folds). We used the four classification models explained above (KNN, SVM, Naive Bayes, and Decision Trees).

For **split 1 (80-20 split)**,

- KNN model has a best accuracy of 0.7936 (with K=10)
- SVM model has a best accuracy of 0.6966
- Naive Bayes has a best accuracy of 0.6371
- Decision Tree has a best accuracy of 0.7733 (with criterion 'Entropy')

Figure 8 shows a graph comparing the accuracy of the four different models for the first split.

For **split 2 (70-15-15 split)**,

- KNN model has a best accuracy of 0.8089 (with K=10)
- SVM model has a best accuracy of 0.6699
- Naive Bayes has a best accuracy of 0.6486
- Decision Tree has a best accuracy of 0.7828 (with criterion 'Entropy')

Figure 9 shows a graph comparing the accuracy of the four different models for the second split.

There is a slight improvement in the performance of the models after including a validation split. However, we cannot say that for certain since the data points for the two splits are different.

For **split 3**, we used K-Fold Cross Validation (ranging from 2 to 10 folds)

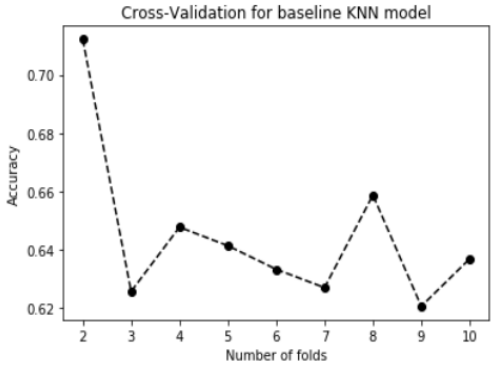

Figure 10: KNN model

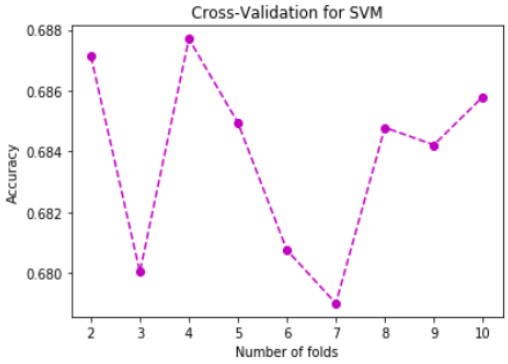

Figure 11: SVM model

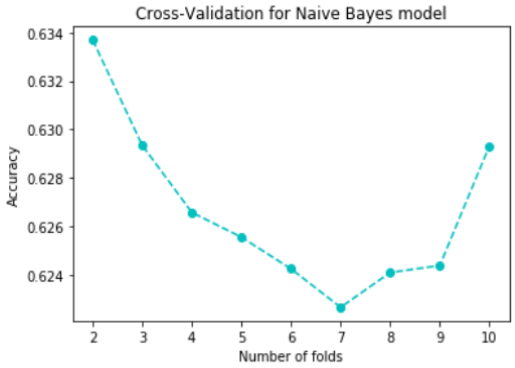

Figure 12: NB model

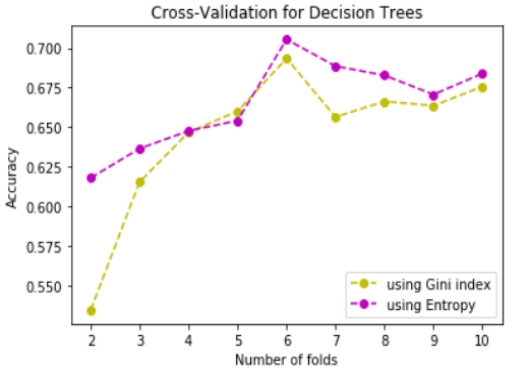

Figure 13: DT model

Figure 10 shows a graph indicating the accuracy of KNN model for number of folds ranging from 2 to 10. The baseline KNN model (with K=10) has a best accuracy of 0.7122 with 2 folds.

Figure 11 shows a graph indicating the accuracy of SVM model for number of folds ranging from 2 to 10. The SVM model has a best accuracy of 0.6877 with 4 folds.

Figure 12 shows a graph indicating the accuracy of Naive Bayes model for number of folds ranging from 2 to 10. The Naive Bayes model has a best accuracy of 0.6337 with 2 folds

Figure 13 shows a graph indicating the accuracy of Decision Tree for number of folds ranging from 2 to 10. The Decision Tree (with criterion 'Entropy') has a best accuracy of 0.7055 with 6 folds

4.2 Task 2

We used a basic LSTM model to calculate the hourly traffic volume according to a given station's geographic coordinates. 80% of the data was used as training data, and the remaining data as test data. After training the model for 50 epochs, the model started fitting to the data well and gave an RMSE value of 0.029.

Figure 14 shows a graph indicating the RMSE values for train and test data over a period of 50 epochs.

5 Conclusion

- For the first project task, we predicted whether a given road is a National highway or not using four models - KNN, SVM, Naive Bayes and Decision Trees

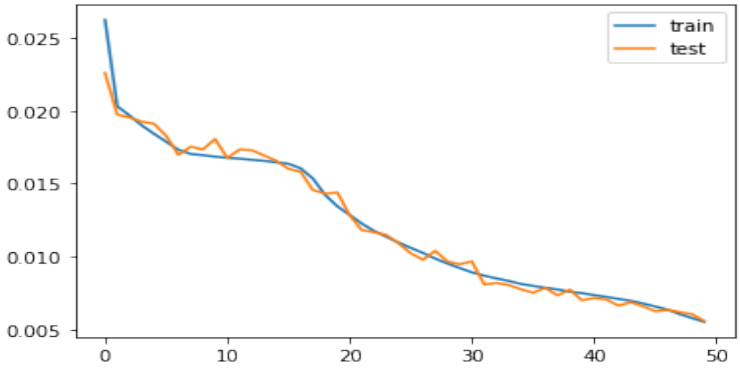

Figure 14: LSTM - Training loss vs Test loss

- We split the data three different ways; an 80-20 train-test split, a 70-15-15 train-validation-test split, and K-fold cross validation. All three splits give their best accuracy for KNN model (with 10 neighbors).
- We were not able to tune these classification models further due to time constraints, but there is a pretty good chance that tuning these models can improve the accuracy of these models.
- For the second project task, we calculated the hourly traffic volume for stations in South Carolina and North Carolina.
- An LSTM was used to predict the hourly traffic volume and the model had a Test RMSE of 0.029.
- These encouraging results for such a basic LSTM model mean that if we are able to tune the model further, we might get even better results.
- There are a few ways that we can use to improve the performance of the LSTM model and lower the RMSE value. We could increase the number of training data points by augmenting data or we could tune the model by adding more depth to the model or by adding dropout or batch normalization layers. Another way to get better results is to use pre-trained models like BERT or ELMo.
- With a low RMSE value, this model can be used in navigation systems or any application that predict (or analyzes) traffic trends by the hour.

6 References

[1] Pan, B., Demiryurek, U. and Shahabi, C., 2012, December. Utilizing real-world transportation data for accurate traffic prediction. In 2012 IEEE 12th International Conference on Data Mining (pp. 595-604). IEEE.
[2] Min, W. and Wynter, L., 2011. Real-time road traffic prediction with spatio-temporal correlations. Transportation Research Part C: Emerging Technologies, 19(4), pp.606-616.
[3] Ishak, S. and Al-Deek, H., 2002. Performance evaluation of short-term time-series traffic prediction model. Journal of transportation engineering, 128(6), pp.490-498.
[4] Yin, X., Wu, G., Wei, J., Shen, Y., Qi, H. and Yin B. (2021) 'Deep Learning on Traffic Prediction: Methods, Analysis and Future Directions'. IEEE Transactions on Intelligent Transportation Systems

The github repo is `https://github.ncsu.edu/ragarwa9/engr-ALDA-fall2021-P38`

7 Appendix

7.0.1 Old Problem Statement

- Predicting traffic volume at a station for a particular date-time. Visualizing hourly traffic volume at a station/route/region for different days of the week and calculating peak congestion hours for different regions.
- Analysing traffic trends and discovering clusters of stations having similar traffic volume at a time, and discovering routes having high traffic congestion.

7.0.2 New Problem Statement

- Implement binary classification model to predict whether a given road is part of an Urban road system or Rural road system.
- Implement regression model to find out the hourly traffic volume at any given latitude and longitude coordinates.