TRAFFIC FORECASTING – TIME SERIES USING RNN

By

Reshma Jayaprakash

Scifor Technologies Trainee

ID: STB03011

Submitted to the Organization Scifor Technologies, Department of Data Science

Under the guidance of Ms. Urooj Khan

Scifor Technologies

Bengaluru, Karnataka

ABSTRACT

Traffic congestion is a pervasive issue in urban areas, and effective traffic forecasting is crucial for proactive management. This project aims to develop a robust Traffic Forecasting System based on Time Series Analysis using Gated Recurrent Unit (GRU) models. The focus is on forecasting traffic conditions for four junctions, leveraging historical data to provide insights into future traffic patterns.

To understand and analyze the traffic conditions over the years, an application will be developed based on the features like, Data Visualization of Traffic on Junctions over Years and Time Series Analysis to understand Trend and Seasonality. Then various options to Displays Model Evaluation results, option to visualize Traffic Predictions Vs. True Values by selecting the dropdown option of each Junctions and to visualize Traffic Forecast by selecting the Next Time Step.

INTRODUCTION

Urban traffic poses significant challenges, necessitating advanced tools for prediction and analysis. Traffic congestion has always been a crucial aspect of urban planning but has become a serious issue that must be addressed due to the rapid increase in the number of vehicles and transportation demand.

Traffic signal control is an important tool in traffic flow management as it is considered as one of the most effective ways to reduce traffic congestion at intersections.

Because of vehicle flow interactions within the network, human behavioral considerations, stochastic traffic demand, and traffic accidents, the Intersection Traffic Signal Control Problem (ITSCP) is a complex problem. The ITSCP is further complicated by not only the randomness of vehicular arrivals at an intersection, but also by the various configurations and numbers of intersections, types of vehicles in the network.

Traditional time series forecasting methods are often inadequate for capturing the complexities of traffic patterns. Hence, this project employs GRU, a type of recurrent neural network (RNN), renowned for its effectiveness in modeling sequential data.

PROBLEM STATEMENT

The project addresses the need for accurate and timely traffic forecasts for urban junctions. The challenges include handling diverse traffic patterns, accommodating fluctuations due to various factors, and providing a user-friendly interface for understanding and analyzing the forecasts. The application built using Streamlit aims to empower users with valuable insights into future traffic conditions, enabling better decision-making and proactive traffic management strategies.

OBJECTIVE

To Build a Gated Recurrent Unit (GRU) model tailored for time series analysis of traffic data.

Develop an interactive and user-friendly application using Streamlit for visualizing and understanding traffic forecasts.

Provide users with actionable insights into future traffic conditions, fostering informed decision-making.

Enhance traffic management strategies by enabling a proactive approach to address potential congestion issues.

By achieving these objectives, the project contributes to the development of efficient and accessible tools for urban traffic forecasting, ultimately leading to improved traffic management and enhanced urban mobility.

TOOLS AND TECHNOLOGY

Software Requirement:

1. Python IDE
2. Streamlit Community
3. Github
4. MS Office

Hardware Requirement:

1. CPU: i5 Processor 64-bit
2. RAM: 4GB

PROPOSED METHOD

The proposed method leverages a Gated Recurrent Unit (GRU) model for accurate and efficient time series analysis of traffic data. It combines advanced neural network architecture with user-friendly application development to deliver a comprehensive solution for accurate and accessible traffic forecasting. The incorporation of GRU models, coupled with feature engineering and interactive visualization, aims to provide users with a powerful tool for understanding and managing urban traffic.

The method encompasses the following key steps:

Data Preprocessing:

Load and preprocess historical traffic data, including date-time conversion and exploration of relevant features. Address missing values and normalize the data to ensure consistency in the model.

Time Series Analysis with GRU:

Implement a GRU-based neural network architecture to capture temporal dependencies in traffic patterns effectively. Train the model using historical traffic data, emphasizing sequence learning for accurate forecasting.

Streamlit Application Development:

Utilize Streamlit to create an interactive and user-friendly application for visualizing traffic forecasts. Integrate the GRU model into the application, allowing users to select specific junctions and timeframes for analysis.

Feature Engineering for Enhanced Forecasting:

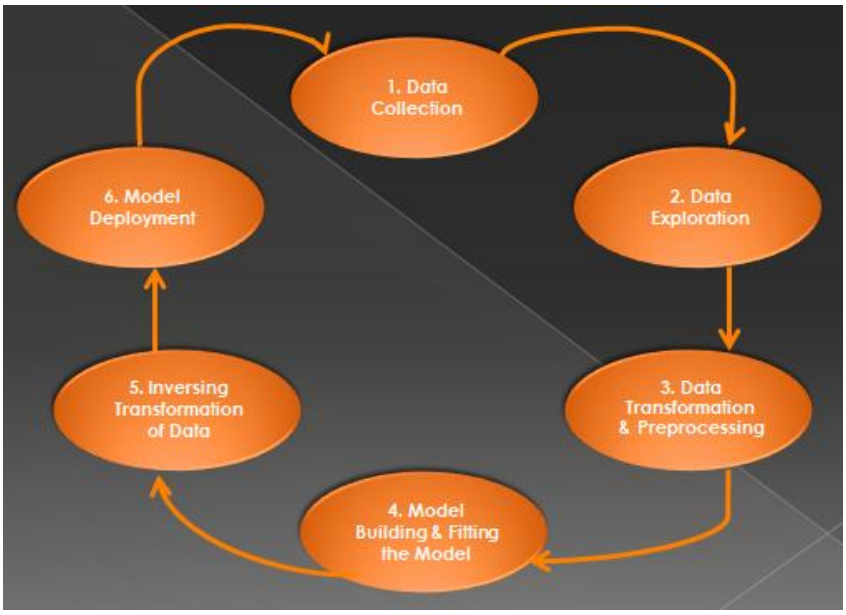
Explore additional features such as year, month, day, and hour to enhance the model's understanding of temporal variations.

Evaluation and Fine-Tuning:

Assess the performance of the GRU model using evaluation metrics such as Mean Root Mean Squared Error (RMSE). Fine-tune the model parameters based on performance metrics to optimize forecasting accuracy.

METHODOLOGY

Below block diagram shows the project workflow,



Data Collection

Data Source: https://www.kaggle.com/datasets/fedesoriano/traffic-prediction-dataset This dataset is a collection of numbers of vehicles at four junctions at an hourly frequency.

The CSV file provides four features:

1. DateTime
2. Junctions
3. Vehicles
4. ID

- The sensors on each of these junctions were collecting data at different times, hence the traffic data from different time periods
- Some of the junctions have provided limited or sparse data
- This data consist of 48120 instances

Data Exploration

- This collection of data starts from 11/01/2015, 00:00 and ends at 06/30/2017, 23:00
- Junction 1, Junction 2 and Junction 3 has total of 14592 entries out of 48120
- In Junction 3 there is only limited data, that is of 4344 entries

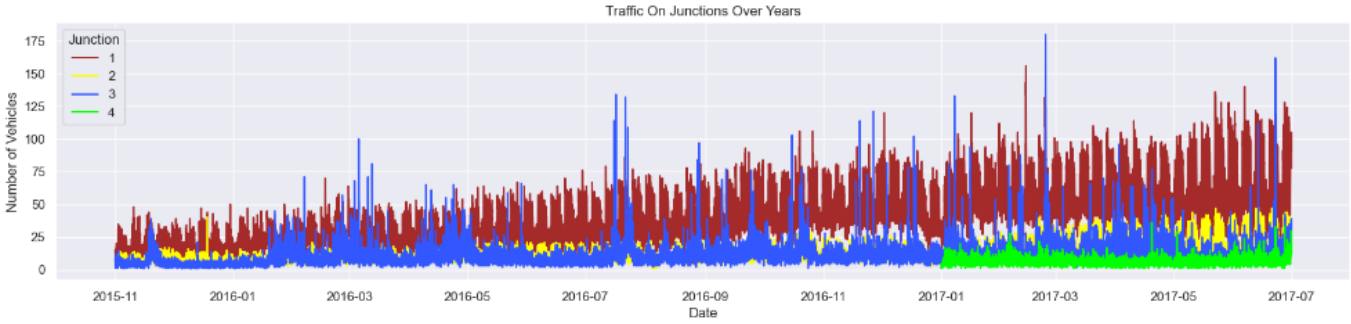
Initial Process,

1. Parsing Dates
2. Plotting Time Series
3. Feature Engineering for EDA

Parsing Dates – It is a process of converting date and time information in a dataset from a string format to a datetime format.

In Python, the pandas library provides a convenient method to parse dates using the pd.to_datetime() function. This conversion allows for easier handling and analysis of time-related data.

Plotting Time Series - Traffic on Junctions over Years has been plotted using Seaborn library.



- Here, It can be seen that the Junction 1 is visibly having an upward trend
- The data for the Junction 4 is sparse starting only after 2017
- Seasonality is not evident from the above plot. So datetime composition has been explored to figure out more about it

Trend and Seasonality are the components of Time Series

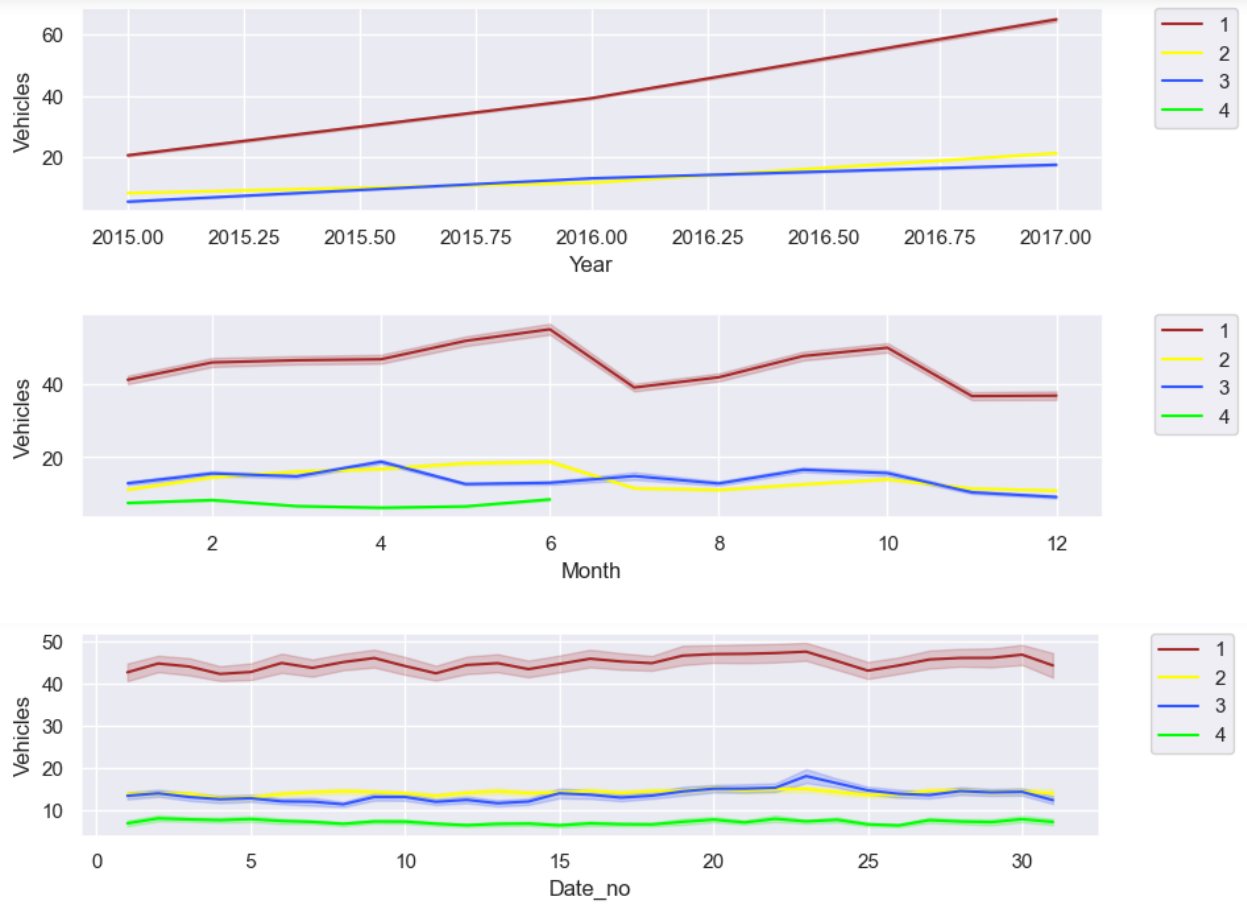
- Trend – is a long term movement observed in time series that changes over time which can be either positive (increasing) or negative (decreasing)

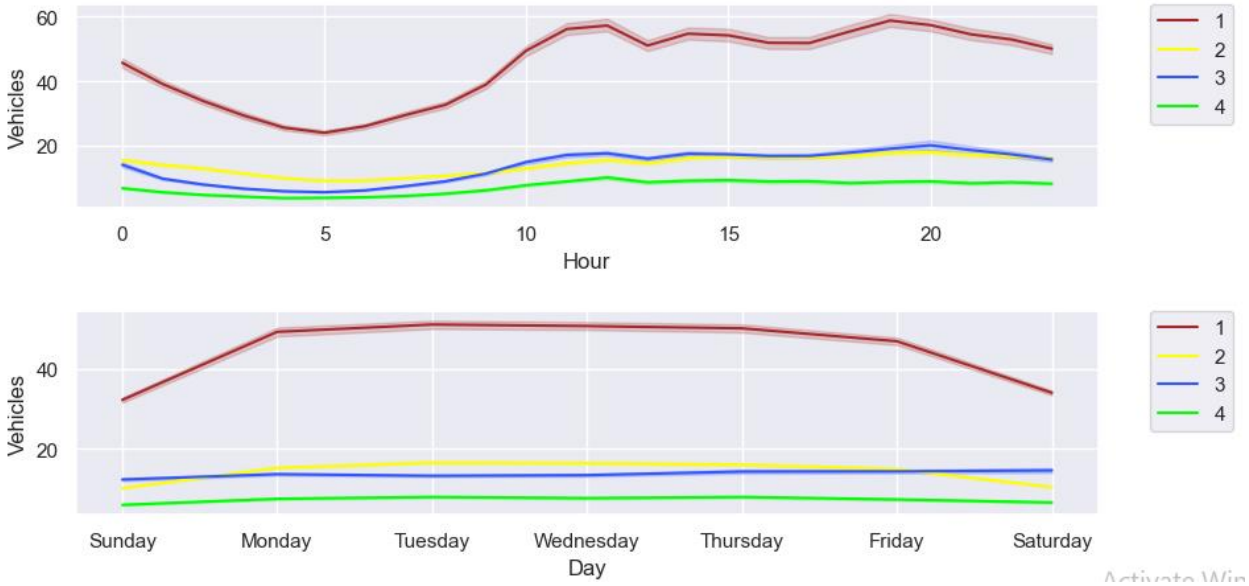
- Seasonality – is a periodic fluctuation that occurs in time series where predictable and regular patterns are exhibited at intervals
- Here, we have plotted seasonality graphs by doing Feature Engineering

Feature Engineering for EDA – Created new features out of DateTime and plotted using seaborn library

New Features:

1. Year
2. Month
3. Date in the given month
4. Days of week
5. Hour





- Yearly, there has been an upward trend for all junctions except for the fourth junction due to the sparse data in Junction 4
- We can see that there is an influx in the first and second junctions around June, this may be due to summer break and activities around the same
- Monthly, throughout all the dates there is a good consistency in data
- For a day, we can see that there are peaks during morning and evening times and a decline during night hours
- For weekly patterns, Sundays enjoy smoother traffic as there are lesser vehicles on roads. Whereas Monday to Friday the traffic is steady

Data Transforming and Preprocessing

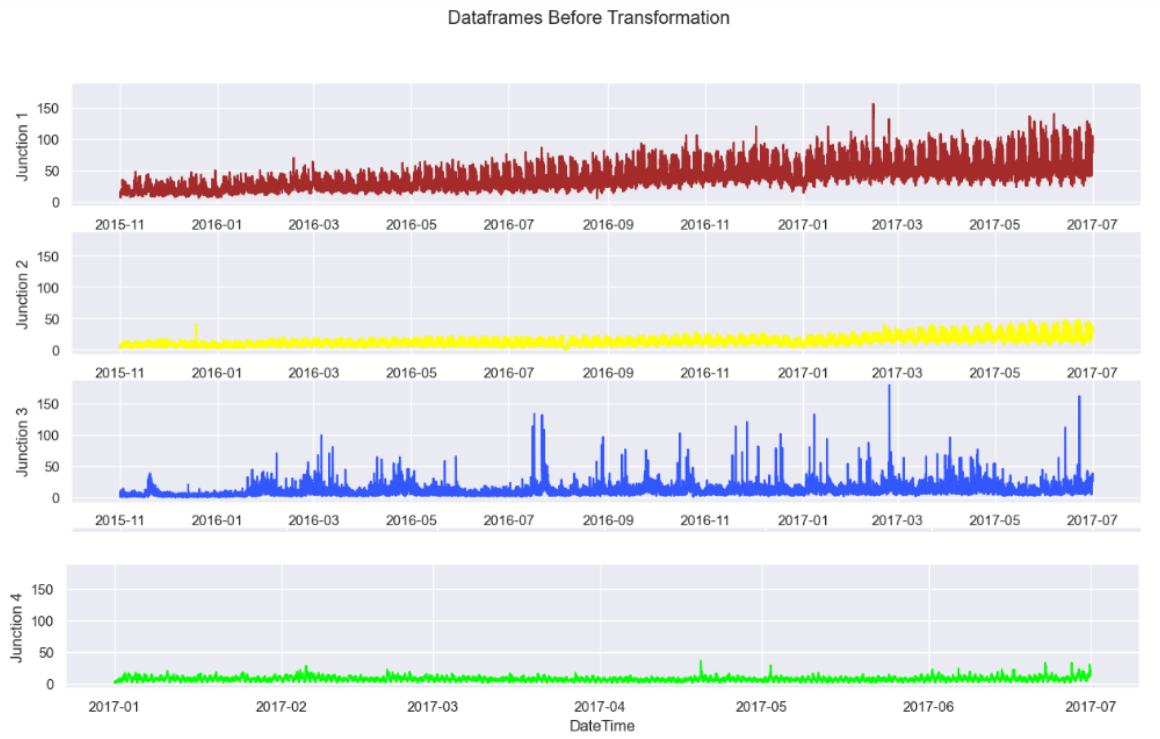
The span of data from all four junctions is not the same. Data provided for the fourth junction is limited to only 2017. The yearly trend for Junctions one, two and three, have different slopes.

Junction 1 has a more strong weekly seasonality in comparison to the other junctions. So here, transforming is applied to transform non-stationary time series to stationary by using the method differencing.

A time series is stationary if it does not have a trend or seasonality. From the analysis made, we have seen a weekly seasonality and an upwards trend over the years.

In the below plot, it is again seen that the Junctions one and two have an upward trend. If we limit the span we will be able to further see the weekly seasonality.

Hence Transforming – Normalizing and Differencing has applied



Transforming the series

In accordance with the above observations, Differencing to eliminate the seasonality should be performed as follows:

- For Junction 1, I will be taking a difference of weekly values.
- For Junction 2, the difference of consecutive days is a better choice
- For Junctions 3 and 4, the difference of the hourly values will serve the purpose

Augmented Dickey-Fuller test

This test is a statistical hypothesis test used to determine the presence of a unit root in a time series dataset. A unit root suggests that a time series has a stochastic trend and is non-stationary. The ADF test assesses whether differencing the series can make it stationary.

```
ADF Statistic: -15.265303390415426
p-value: 4.7985398763968885e-28
Critical Values:
        1%: -3.431
        5%: -2.862
        10%: -2.567
Time Series is Stationary
```

```
ADF Statistic: -28.001759908832412
p-value: 0.0
Critical Values:
        1%: -3.431
        5%: -2.862
        10%: -2.567
Time Series is Stationary
```

```
ADF Statistic: -21.795891026940296
p-value: 0.0
Critical Values:
        1%: -3.431
        5%: -2.862
        10%: -2.567
Time Series is Stationary
```

```
ADF Statistic: -17.97909256305261
p-value: 2.778787532594783e-30
Critical Values:
        1%: -3.432
        5%: -2.862
        10%: -2.567
Time Series is Stationary
```

Creating Test and Train Data

- Preprocessing the data for Neural Network
- Splitting the Test and Train sets
- Assigning X as Features and y as Target
- Reshaping data for Neural Network

90% of the data is used for training (train), and the remaining 10% is used for testing (test). The TnF function is defined to create features (X) and targets (y) from the differenced time series data. The steps parameter is used to determine the size of the sliding window to create features. It has been set to 32. The shapes of X_train and X_test are adjusted to be compatible with the expected input shape for a neural network model. Features (X_train, X_test) and targets (y_train, y_test) are assigned for each junction (J1, J2, J3, J4).

Model Building and Fitting

- For model building, I have used Gated Recurrent Unit (GRU). Created a function for the neural network to call on and fit the data frames for all four junctions
- It uses the Sequential model from Keras to stack layers
- The model architecture consists of multiple GRU layers with dropout to prevent overfitting
- The last layer is a Dense layer with one unit, as it's a regression problem predicting a single value
- The model is compiled using Stochastic Gradient Descent (SGD) as the optimizer and Mean Squared Error (mean_squared_error) as the loss function
- Training is done with the specified number of epochs, batch size, and learning rate schedule

Neural Network Structure

Input Layer:

Type: GRU layer

Number of Units: 150

Activation Function: Hyperbolic Tangent (tanh)

Dropout Layer:

Rate: 0.2 (to prevent overfitting)

GRU Layer:

Number of Units: 150

Activation Function: Hyperbolic Tangent (tanh)

Dropout Layer:

Rate: 0.2

GRU Layer:

Number of Units: 50

Activation Function: Hyperbolic Tangent (tanh)

Dropout Layer:

Rate: 0.2

GRU Layer:

Number of Units: 50

Activation Function: Hyperbolic Tangent (tanh)

Dropout Layer:

Rate: 0.2

GRU Layer (Final):

Number of Units: 50

Activation Function: Hyperbolic Tangent (tanh)

Dropout Layer:

Rate: 0.2

Dense (Output) Layer:

Number of Units: 1 (as it's a regression problem predicting a single value)

The model is compiled using Stochastic Gradient Descent (SGD) as the optimizer with a learning rate of 0.01. The training is done for 50 epochs with a batch size of 150. Additionally, early stopping with a patience of 10 is used to prevent overfitting, and a learning rate scheduler is applied to adjust the learning rate during training.

Neural Network Layers

In a neural network, the input layer receives the input data, and the output layer produces the final prediction. The hidden layers, which can include various types of layers, process the input

data and extract features to make accurate predictions. **GRU (Gated Recurrent Unit)** and **Dropout layers** are two types of layers used here.

GRU (Gated Recurrent Unit):

GRU is a type of recurrent neural network (RNN) layer that is designed to capture and learn dependencies in sequential data. It is especially useful for working with time-series data, where the order of the data points matters. GRUs have gating mechanisms that control the flow of information through the network, helping in the learning of long-term dependencies.

Dropout Layer:

Dropout is a regularization technique used to prevent overfitting in neural networks. During training, a fraction of randomly selected neurons (nodes) are "dropped out" or ignored during forward and backward passes. This prevents the network from relying too much on specific neurons and improves its generalization to unseen data.

In the implementation for the traffic prediction model, GRU layers are used to capture temporal patterns in the traffic data over time, and Dropout layers are employed to prevent overfitting during training. The combination of these layers helps build a robust and generalizable model for predicting traffic patterns.

Hyperparametes

GRU Units:

units=150 and units=50: These specify the number of memory units or neurons in each GRU layer. The model has two GRU layers with 150 units and two with 50 units.

Dropout Rate:

Dropout(0.2): This adds a Dropout layer with a dropout rate of 0.2 after each GRU layer. It means that 20% of the neurons in each layer will be randomly set to 0 during training.

Learning Rate:

SGD(learning_rate=0.01): Stochastic Gradient Descent (SGD) optimizer is used with a learning rate of 0.01. This controls the step size during the optimization process.

Epochs and Batch Size:

epochs=50 and batch_size=150: These define the number of training epochs (passes through the entire dataset) and the size of each mini-batch during training.

Early Stopping:

callbacks.EarlyStopping(patience=10, restore_best_weights=True): Early stopping is employed to halt training if there is no improvement in the validation loss for 10 consecutive epochs. The best weights are restored when training is stopped.

Learning Rate Scheduler:

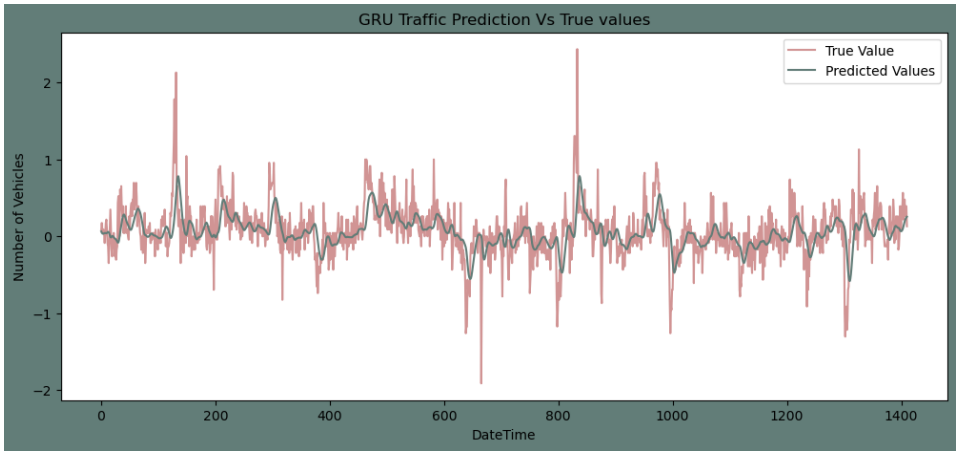
callbacks.LearningRateScheduler(lr_schedule): A learning rate scheduler is used to adjust the learning rate during training. The lr_schedule function is defined to reduce the learning rate after the first 10 epochs.

These hyperparameters play a crucial role in the performance of the neural network model. Depending on the specific dataset and problem, these values may need to be tuned for optimal results.

Model Fitting – Junction 1

```
# Predictions For First Junction
PredJ1 = GRU_model(X_trainJ1,y_trainJ1,X_testJ1)
RMSE_J1 = RMSE_Value(y_testJ1, PredJ1)
PredictionsPlot(y_testJ1, PredJ1, 0)
```

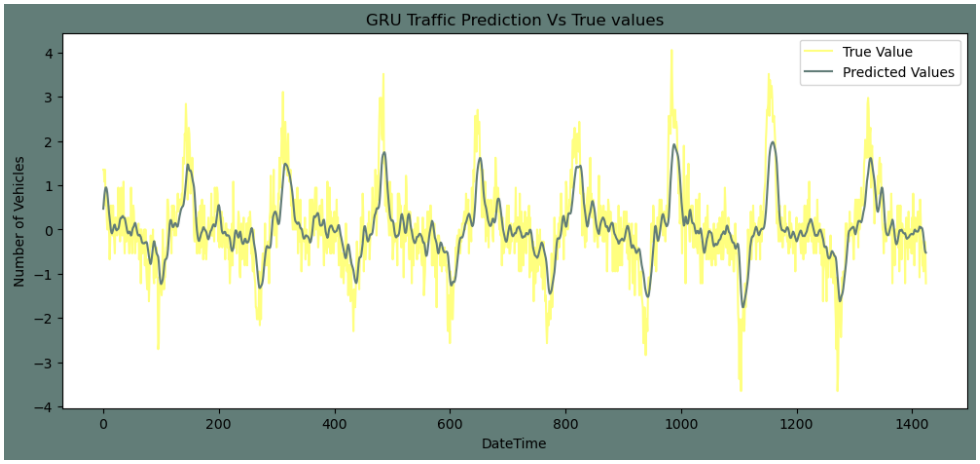
```
Epoch 50/50
87/87 [==============================] - ETA: 0s - loss: 0.0644WARNING:tensorflow:Early stopping conditioned on metric `val_l
oss` which is not available. Available metrics are: loss
87/87 [==============================] - 47s 535ms/step - loss: 0.0644 - lr: 1.4781e-04
45/45 [==============================] - 11s 67ms/step
The root mean squared error is 0.29980381567057063.
```



Model Fitting – Junction 2

```
# Predictions For Second Junction
PredJ2 = GRU_model(X_trainJ2, y_trainJ2, X_testJ2)
RMSE_J2 = RMSE_Value(y_testJ2, PredJ2)
PredictionsPlot(y_testJ2, PredJ2, 1)
```

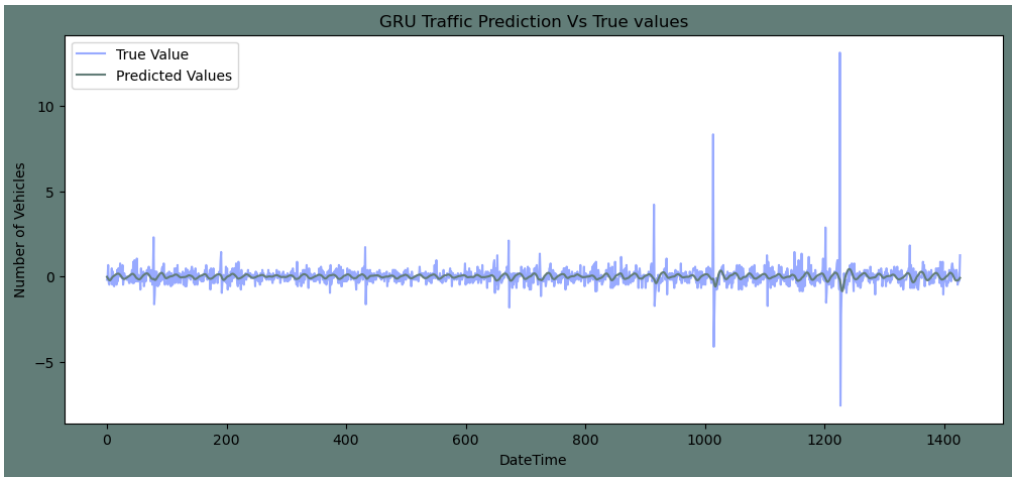
```
Epoch 50/50
88/88 [==============================] - ETA: 0s - loss: 0.1973WARNING:tensorflow:Early stopping conditioned on metric `val_l
oss` which is not available. Available metrics are: loss
88/88 [==============================] - 42s 478ms/step - loss: 0.1973 - lr: 1.4781e-04
45/45 [==============================] - 9s 62ms/step
The root mean squared error is 0.6211304479095612.
```



Model Fitting – Junction 3

```
# Predictions For Third Junction
PredJ3 = GRU_model(X_trainJ3, y_trainJ3, X_testJ3)
RMSE_J3 = RMSE_Value(y_testJ3, PredJ3)
PredictionsPlot(y_testJ3, PredJ3, 2)
```

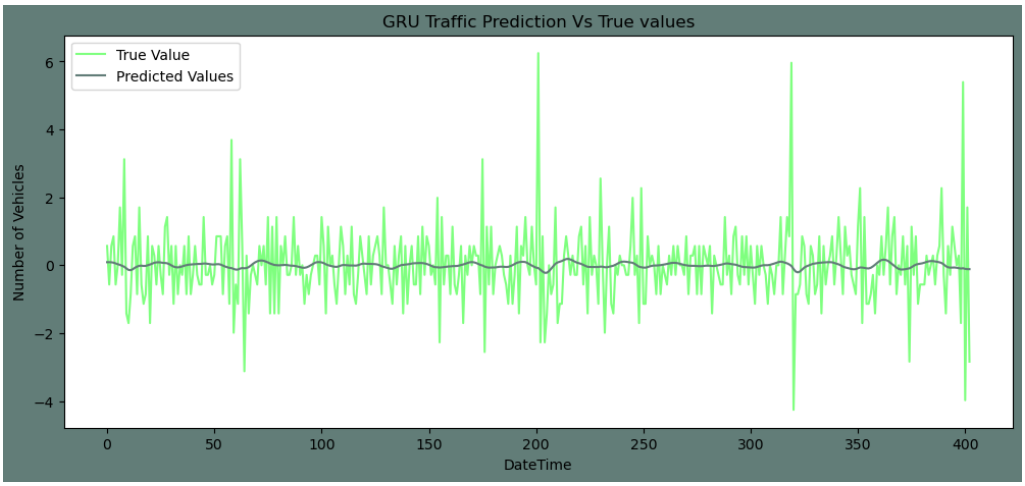
```
Epoch 50/50
88/88 [==============================] - ETA: 0s - loss: 0.2865WARNING:tensorflow:Early stopping conditioned on metric `val_l
oss` which is not available. Available metrics are: loss
88/88 [==============================] - 43s 485ms/step - loss: 0.2865 - lr: 1.4781e-04
45/45 [==============================] - 12s 64ms/step
The root mean squared error is 0.6281028084694495.
```



Model Fitting – Junction 3

```
# Predictions For Fourth Junction
PredJ4 = GRU_model(X_trainJ4, y_trainJ4, X_testJ4)
RMSE_J4 = RMSE_Value(y_testJ4, PredJ4)
PredictionsPlot(y_testJ4, PredJ4, 3)
```

```
Epoch 50/50
26/26 [==============================] - ETA: 0s - loss: 0.6806WARNING:tensorflow:Early stopping conditioned on metric `val_l
oss` which is not available. Available metrics are: loss
26/26 [==============================] - 13s 496ms/step - loss: 0.6806 - lr: 1.4781e-04
13/13 [==============================] - 6s 63ms/step
The root mean squared error is 1.1092825473039842.
```

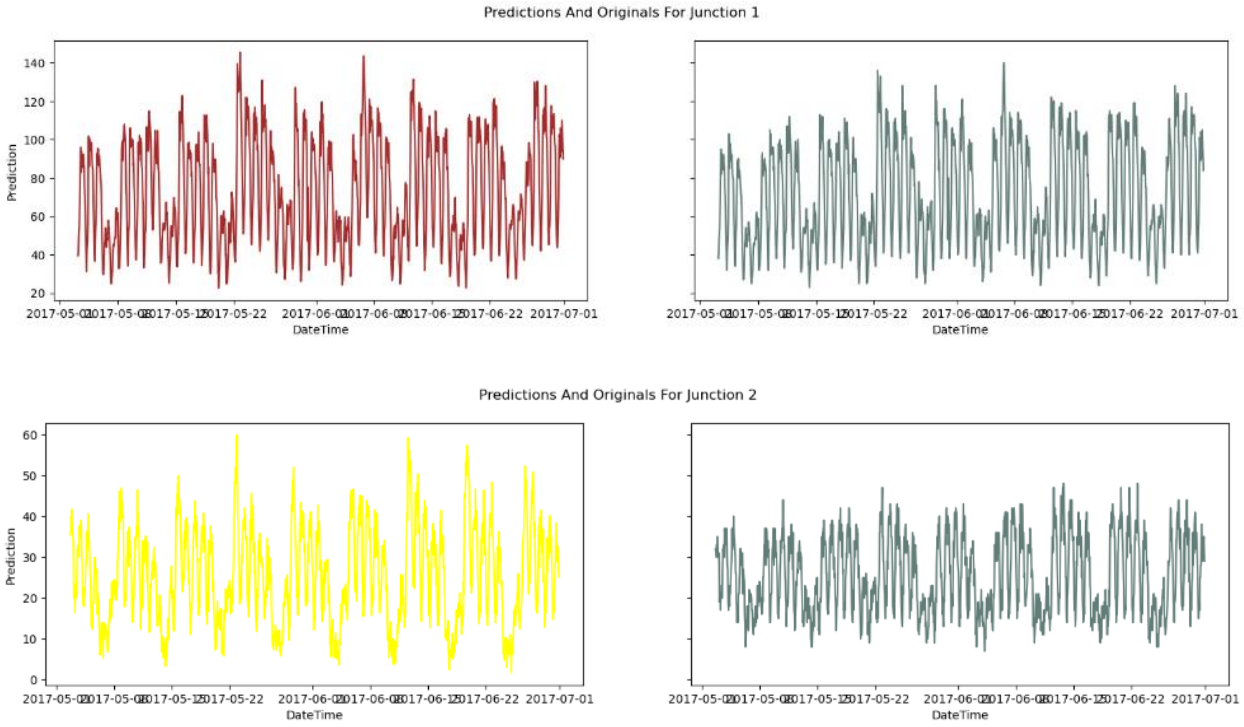


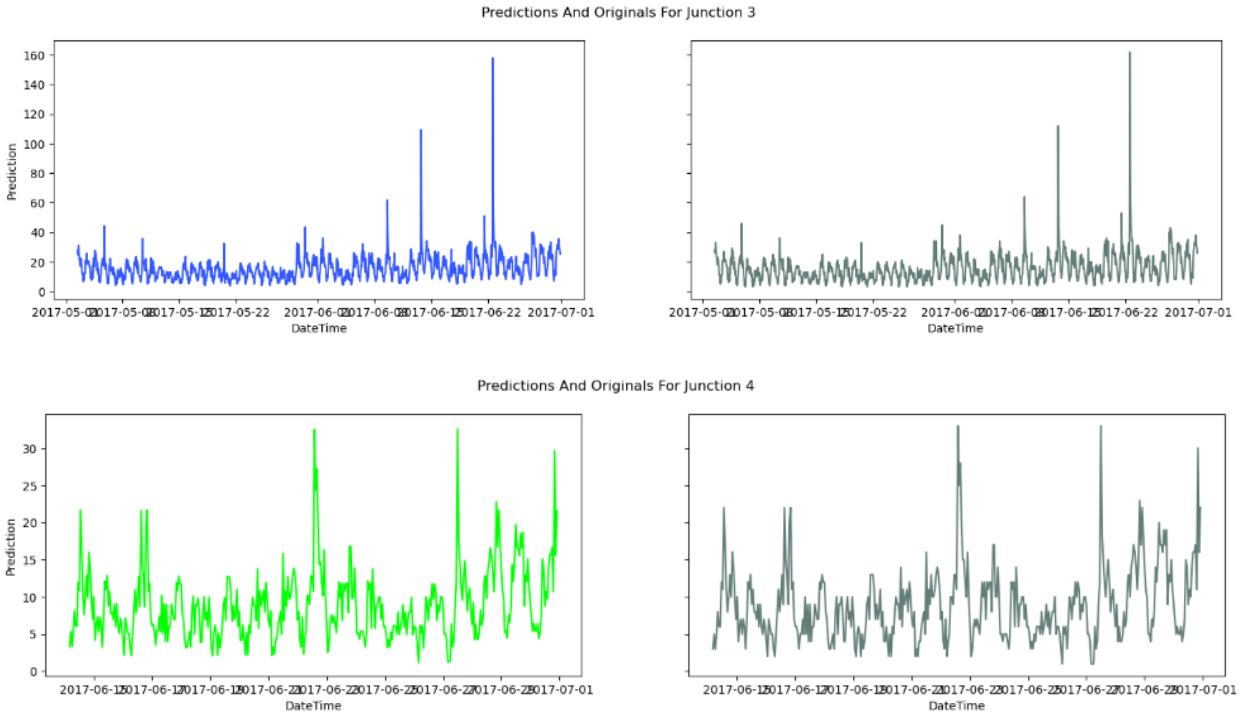
Model Evaluation

	Junction	RMSE
0	Junction1	0.299804
1	Junction2	0.616637
2	Junction3	0.628668
3	Junction4	1.110561

Inverse Transformation of Data

Inverse transformation in time series forecasting refers to converting the forecasted values back to their original scale after applying some transformations or normalization during the modeling process. In the implementation, the data has undergone differencing and normalization during the modeling phase. The inverse transformation is applied to bring the forecasted values back to the original scale for comparison with the actual values.





Model Deployment

A Streamlit application has been built to understand and analyze Traffic Predictions and Forecasting.

Dashboard URL: https://traffic-forecasting-reshmajp.streamlit.app/

Features:

- Data Visualization of Traffic on Junctions over Years
- Time Series Analysis to understand Trend and Seasonality
- Displays Model Evaluation results
- Option to visualize Traffic Predictions Vs. True Values by selecting the dropdown option of each Junctions
- Option to visualize Traffic Forecast by selecting the Next Time Step

Choose a Plot to Display:

Select Plot

Traffic vs Year

- Traffic vs Year
- Traffic vs Month
- Traffic vs Date_no
- Traffic vs Hour
- Traffic vs Day

Root Mean Squared Error (RMSE) for Each Junction:

Junction 1: 0.2998

Junction 2: 0.6166

Junction 3: 0.6287

Junction 4: 1.1106

Traffic Predictions vs True Values:

Select Junction

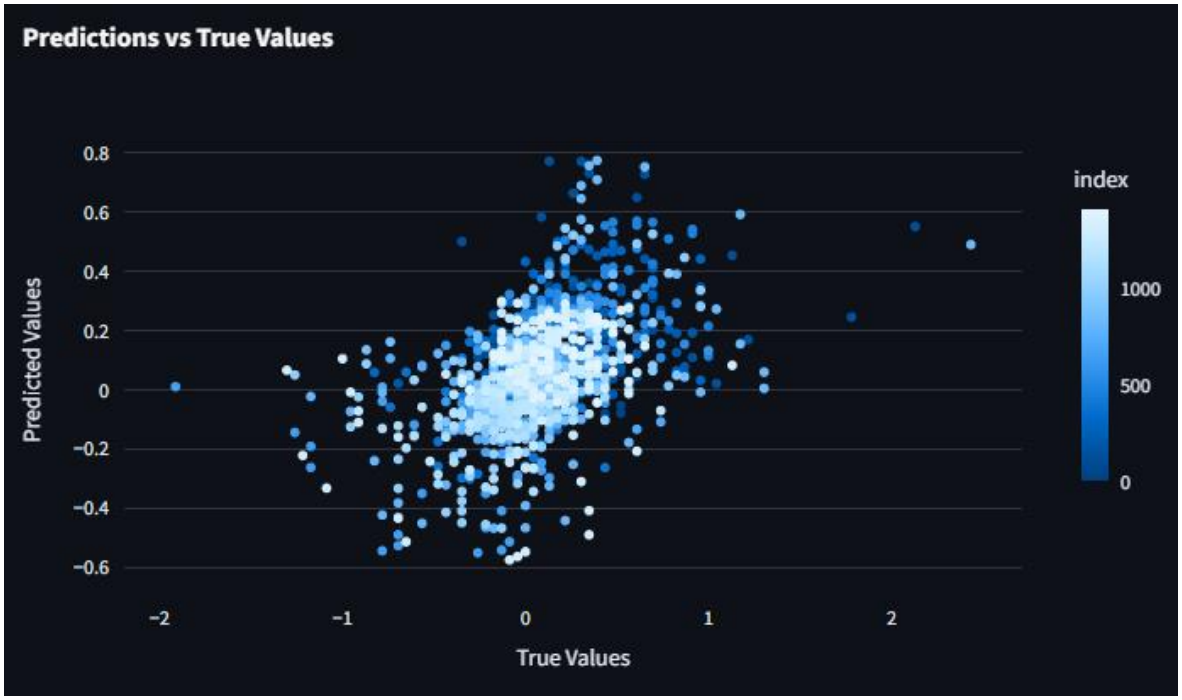
Junction 1

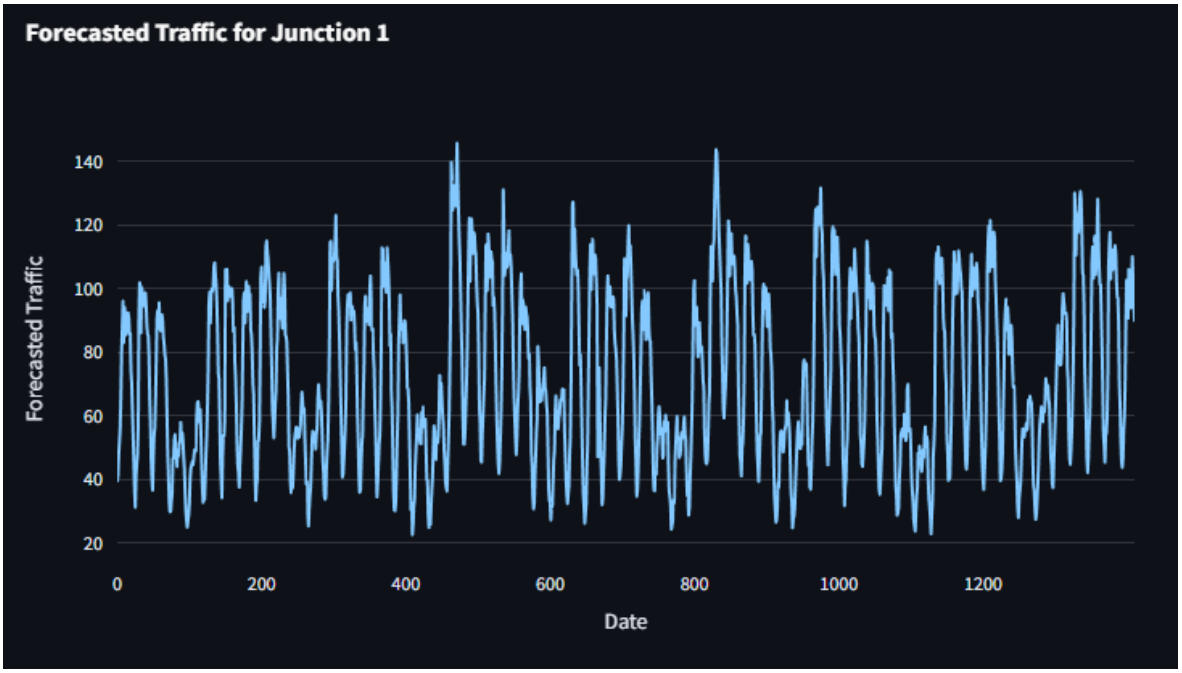
Traffic Forecast for the Next Time Step:

Select Time Step

0

Show Forecast


Predictions vs True Values
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
Predicted Values
−2
−1
0
1
2
True Values
index
1000
500
0


Forecasted Traffic for Junction 1
140
120
100
80
60
40
20
Forecasted Traffic
0
200
400
600
800
1000
1200
Date

APPLICATIONS

Urban Traffic Management:

Provide city planners and traffic management authorities with real-time and forecasted traffic information to optimize traffic light timings and reduce congestion in urban areas.

Route Planning and Navigation:

Integrate traffic forecasts into navigation applications to help users choose the most efficient routes, avoiding congested areas and minimizing travel time.

Public Transportation Planning:

Assist public transportation agencies in optimizing bus schedules and routes based on predicted traffic conditions, leading to improved service reliability.

Emergency Services Optimization:

Support emergency service vehicles by providing insights into traffic patterns, enabling more efficient and timely response to incidents.

Smart City Initiatives:

Contribute to smart city initiatives by providing data-driven insights for traffic planning, reducing pollution, and enhancing overall urban living conditions.

Infrastructure Development:

Support infrastructure planning by identifying areas with consistently high traffic, aiding in the design and construction of roads and transportation facilities.

Predictive Maintenance for Roads:

Use traffic forecasts to predict areas with high traffic loads, enabling proactive road maintenance and repairs to ensure road safety.

CONCLUSION

In this project, I trained a GRU Neural network to predict the traffic on four junctions. I used Normalization and Differencing transform to achieve a stationary Time Series. As the Junctions vary in Trends and Seasonality, I took different approach for each junction to make it stationary. I applied the Root Mean Squared error as the evaluation metric for the model. In addition to that I plotted the Predictions alongside the original test values.

The Number of vehicles in Junction 1 is rising more rapidly compared to Junction 2 and 3. Due to the limitation in data of Junction 4 fails to come up with a conclusion on the same.

The Junction 1's traffic has a stronger weekly seasonality as well as hourly seasonality. Whereas, other junctions are significantly linear.

In conclusion, the development and implementation of the Gated Recurrent Unit (GRU) model for traffic forecasting, coupled with the Streamlit-based application, represent a significant stride towards enhancing urban mobility and transportation management. Through a comprehensive analysis of historical traffic data from four junctions, our model has demonstrated its efficacy in capturing temporal patterns, trends, and seasonality inherent in traffic flow.

Dashboard Link: https://traffic-forecasting-jpreshma.streamlit.app/

FUTURE SCOPE

Multi-Modal Traffic Analysis:

Extend the model to analyze and predict traffic conditions for multiple transportation modes, including public transit and non-motorized transportation.

Evaluation of Alternative Models:

Explore alternative time series models, machine learning techniques, or ensemble methods to assess their effectiveness in comparison to the GRU-based model.

Incorporation of Additional Features:

Integrate additional relevant features, such as holidays, special events, or real-time social media data, to enhance the model's predictive capabilities.

Deployment in Other Urban Areas:

Adapt the model for deployment in diverse urban areas, considering variations in traffic patterns, infrastructure, and city dynamics.

Real-Time Feedback Mechanism:

Implement a feedback loop mechanism for users to provide real-time feedback on predicted versus actual traffic conditions, improving model performance.

REFERENCE

https://www.kaggle.com/datasets/fedesoriano/traffic-prediction-dataset

https://youtu.be/A-MqzClQXl0?si=pYXzFBIq3nJM8mYW

https://www.influxdata.com/time-series-forecasting-methods/#:~:text=Time%20series%20forecasting%20is%20a,hold%20similar%20to%20historical%20trends.

https://www.codingninjas.com/studio/library/time-series-prediction-with-gru

https://towardsdatascience.com/predictive-analytics-time-series-forecasting-with-gru-and-bilstm-in-tensorflow-87588c852915