**AI – Based Traffic Management System**

**A PROJECT REPORT**

*Submitted for the partial fulfillment*

*of*

*Capstone Project requirement of B. Tech CSE*

*Submitted by*

**1. Name: Arushi Shivhare, PRN: 22070521062**

**2. Name: Sanskruti Nerkar, PRN: 22070521028**

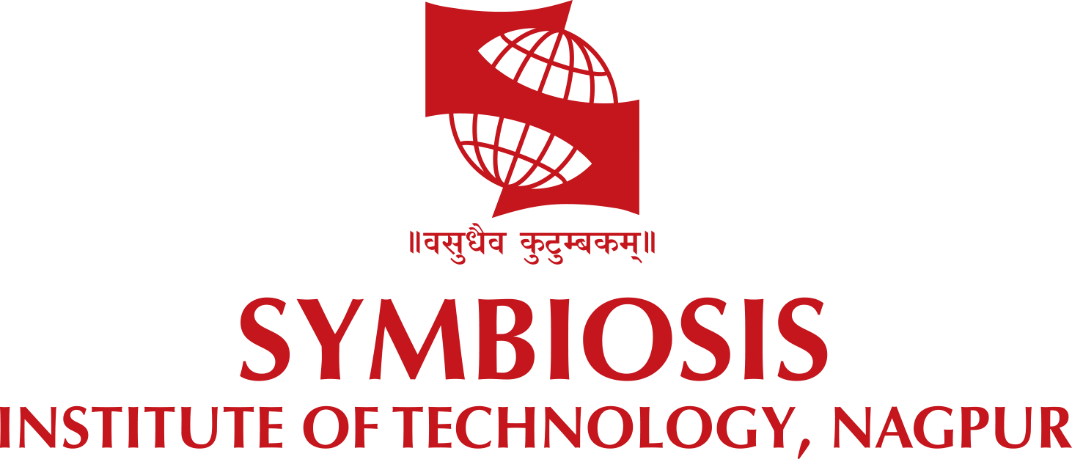
**3. Name: Devyani Balki, PRN: 22070521051**

**B. Tech Computer Science and Engineering**

*Under the Guidance of*

**Prof. (Dr.) Nitin Rakesh**

**Director, SIT Nagpur**



**CERTIFICATE**

This is to certify that the Capstone Project work titled “**AI – Based Traffic Management System**” that is being submitted b**y “Arushi Shivhare, PRN: 22070521062”, “Sanskruti Nerkar”, PRN: 22070521028” ; “Devyani Balki, PRN: 22070521051”** is in partial fulfillment of the requirements for the Capstone Project is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma, and the same is certified.

Name of PBL Guide & Signature

Prof. Dr. Nitin Rakesh

Verified by:

Prof. Monali Gulhane

Dr. Parul Dubey

Capstone Project Coordinator

**The Report is satisfactory/unsatisfactory**

**Approved by**

**Prof. (Dr.) Nitin Rakesh**

**Director, SIT Nagpur**

**ABSTRACT**

This capstone project presents a traffic prediction model using Gated Recurrent Units (GRU), a type of Recurrent Neural Network (RNN), to effectively forecast traffic speeds based on historical data. The first challenge in urban areas is traffic congestion that causes longer travel time, consuming more fuel, generation of air pollution as well as economic losses. Thus, traffic forecasting is a significant factor in an IT system since it enables city authorities to analyze traffic patterns and control congestion in the cities.

In this project, PEMS-BAY dataset is used and an GRU based neural network is applied to capture temporal dependencies which exist in time series traffic data. In every detail of the methodology, from pre processing, sequence generation, model architecture, training strategy till evaluation metrics, each and every thing is systematically reported. The proposed model has good generalizability and prediction accuracy indicative of suitability of GRUs for real time traffic forecasting.

This work also shows the model evaluation using several statistical metrics like Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and coefficient of determining (R² score). The metrics help in interpreting the model’s capability of correctly predicting traffic patterns. The proposed model was again tested and found to have a good performance level for stock prices prediction when using graphical plots and residual analysis to support and demonstrate the impact of using normalization and differencing in the stabilization of learning.

**TABLE OF CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **CHAPTER** | **SECTION** | **TITLE** | **PAGE** |
| **1.** | **-** | **Abstract** | **3** |
| **2.** | **-** | **Introduction** | **6** |
|  | **2.1**  **2.2**  **2.3**  **2.4** | Background  Problem Statement  Objectives  Scope of the Project | **-** |
| **3.** | **-** | **Literature Review** | **7** |
|  | **3.1**  **3.2** | Related Work  Summary of Findings |  |
| **4.** | **-** | **Methodology** | **14** |
|  | **4.1**  **4.2**  **4.3**  **4.4**  **4.5** | System Architecture  Data Collection and Preprocessing  Feature Selection  Model Design – GRU  Training and Testing Procedure | **-** |

|  |  |  |  |
| --- | --- | --- | --- |
| **5.** | **-** | **Implementation** | **17** |
|  | **5.1**  **5.2**  **5.3** | Tools and Technologies Used  Code Snippets and Workflow  Experimental Setup | **-** |
| **6.** | **-** | **Results and Discussion** | **30** |
|  | **6.1**  **6.2**  **6.3** | Evaluation Metrics  Performance Analysis  Comparative Study | **-** |
| **7.** | **-** | **Conclusion** | **33** |
|  | **7.1**  **7.2**  **7.3** | **Summary of Work**  **Limitations**  **Future Work** |  |
| **8.** | **-** | **References** | **39** |
|  |  |  |  |
| **9.** | **-** | **Appendices** | **37** |

**CHAPTER 1**

**INTRODUCTION**

**1.1. Problem Statement**

Traffic congestion is fast becoming a problem of concern in many civilized cities, which affects time, fuel consumption, environment, and the economy. Modern sociological investigations in urban areas and residential densities reveal that congestion cost firms millions of dollars in lost time and results in a significant pollution toll due to increased levels of carbon emissions. This is especially true in the metropolitan cities where the road networks have reached their maximum extent and yet the population of cars and other road user’s increases every now and then.

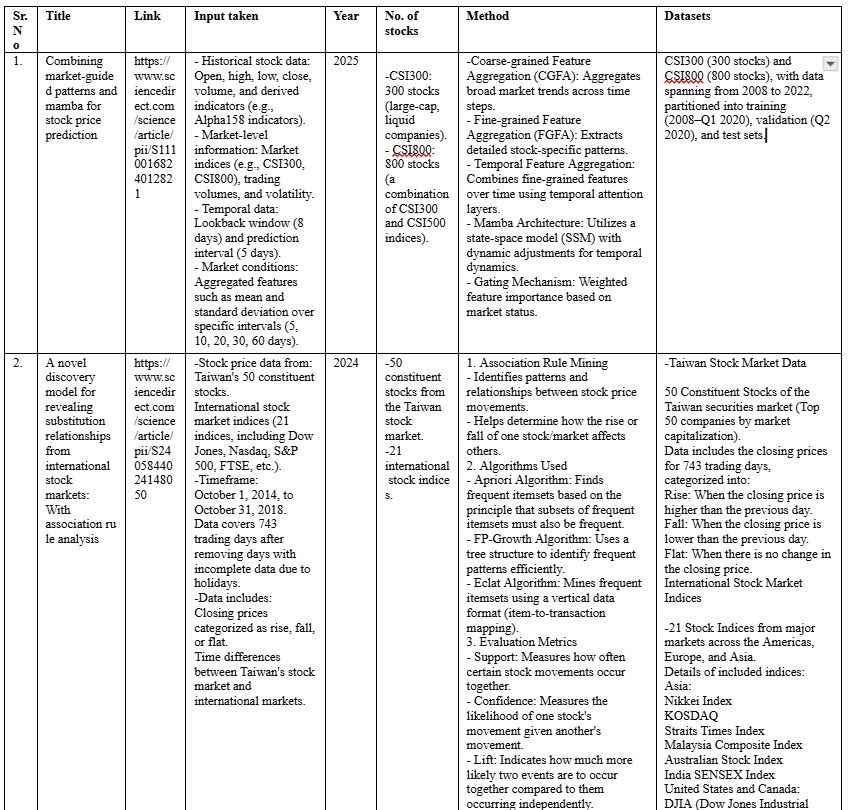
Many traditional forecast methods in modeling traffic flow are not effective in capturing nonlinear characteristics. They also fail to address temporal dependencies such as, change of tariffs during peak hours, congestion at some point or even some faulty sensor readings that are always there in real life traffic situations. Therefore, traffic authorities require short-term prediction tools that will enable them to predict variation and assist in real-time traffic control.

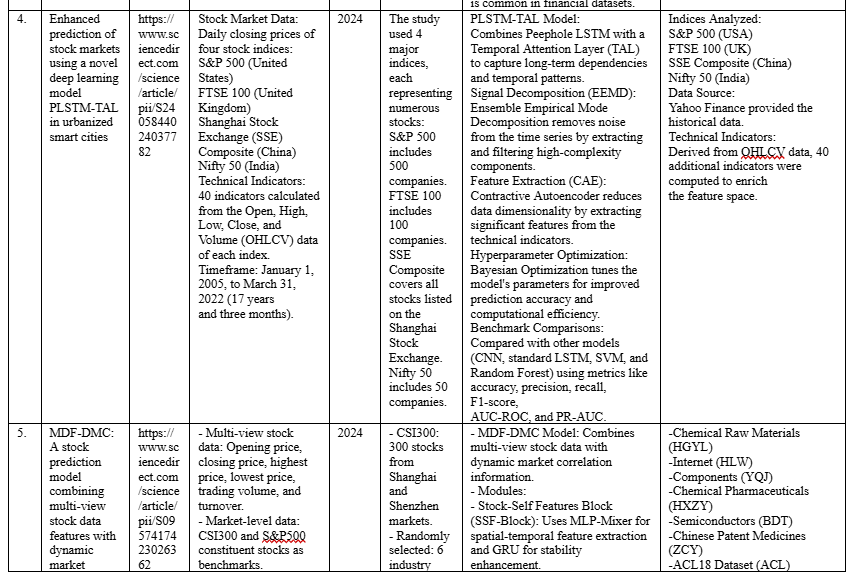
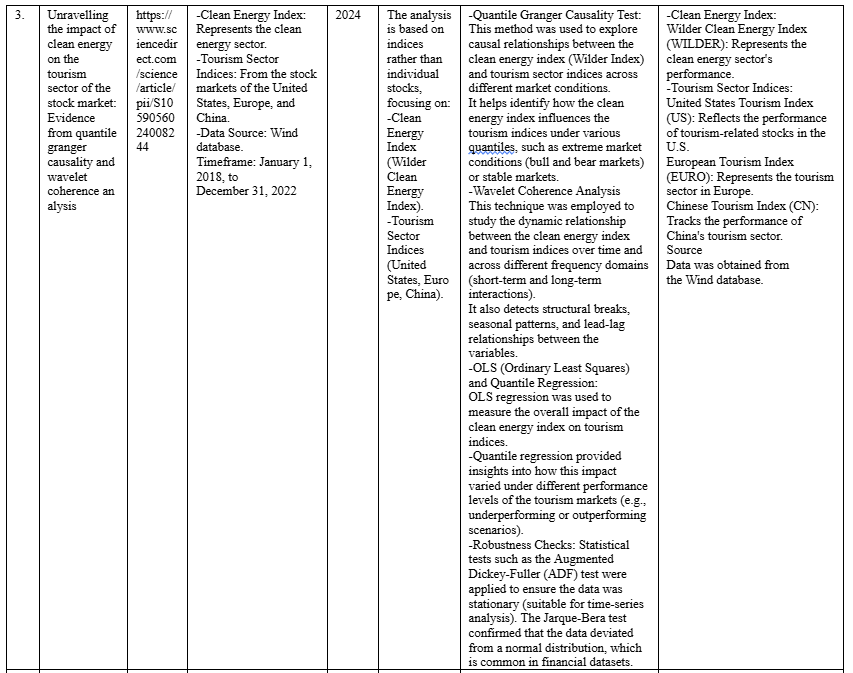
**1.2. Objective**

* To further discuss and reason out why GRU can be implemented in neural networks to serve as the basis for traffic prediction of time series.
* To prepare PEMS-BAY for the application of the proposed GRU model.
* To use an optimized setting for training the GRU model and compare between its performances.
* To compare the performances returned by the different statistical evaluation metrics used to calculate the efficiency of the model.
* For explanations and analysis of traffic trends as it has over time shown to be instrumental in providing an idea of what to expect on the roads.
* The proposed research is aimed at fulfilling the requirement of translating ITS research from theory to practice by developing a prediction model that yields satisfactory performance with reasonable computational costs. The goal of this project also is to reveal the advantages and drawbacks of GRUs in time-series, as well as the possibility of their usage in real-world traffic conditions.

**1.3. Literature Review**

* It is evident that traffic prediction has been one of the great concerns for many researchers over the years. The former kind of approach was a more traditional statistical model, while the latter kind of approach is advanced and utilizes the concepts of machine learning and deep learning to an extensive extent. Such methods are at least an order of magnitude better since they are developed to learn elaborate patterns from scratch and do not require much pre-processing of the raw data.
* The nature of traffic data is such that it is temporally ordered and, therefore, may be non-stationary for higher frequencies or noisy. From literature review, it is evident that the conventional time-series models have gradually been replaced by depth models capable of learning spatial-temporal patterns. Thus, each of them owns its strengths and weaknesses concerning interpretability, accuracy, and scalability.
  + 1. **Traditional Methods:**
* ARIMA (Autoregressive Integrated Moving Average): This type of models assume linearity and stationarity. Despite the fact that ARIMA models are useful for simple time series data, they find it difficult dealing with non-stationary as well as nonlinear traffic data. It also implies that they need manual differencing and do not have the capacity of learning long-range temporal associativity. Because of their pre-specified lag structures and fixed coefficients, they have limited ability in modeling new patterns in the data.
* Kalman Filter: This method is used for the estimation of state variables of a system after which predictions can be made for a short time though it is highly sensitive to noise, missing data and changes in the traffic environment. In their mathematical beauty, Kalman filters require information on the model of the observed system and may not perform well in conditions of significant nonlinearity or unpredictable changes in traffic.
  + 1. **Machine Learning Approaches**
* Support Vector Regression (SVR): SVR light works well with high dimensional data, has high generalization capability though there are no built-in ways of modeling sequential data. It does not natively impose time structures and uses time-delayed inputs that restrains the level of temporal forecasting.
* They include Decision Trees, Random Forests, Gradient Boosting Machines (GBMs); they are efficient on nonlinearity and complex structures but rather sensitive to features. Furthermore, they do not consider the order in which the traffic data is recorded unless this is specifically encoded for the map. However, it should be noted that ensemble methods perform very well in static tabular data as against time series.
  + 1. **Deep Learning Approaches**
* CNNs (Convolutional Neural Networks): The CNNs can model spatial dependencies, such as correlations between sensors, while not implement temporal ones in their pure form; they should be combined with RNNs. Therefore, when used as the single model, even the most basic type of the CNNs is incapable to learn the long-term dependencies which are mandatory for the traffic forecasting of series over time.
* RNNs (Recurrent Neural Networks): The RNN networks are good for sequential data as they capture temporal dependencies between the elements but have problems with long time sequence due to the limited gradient range. Their restricted memory capacity fails them when it comes to learning long-range dependencies which are relevant for recording high-resolution traffic data.
* LSTMs (Long Short-Term Memory Networks): LSTM is a special RNN containing feedback connections with units known as memory cells to control forgetting mechanism and can capture long-term dependency of the series. But at the same time, this makes its architecture intricate, which leads to longer training time. LSTMs also require more space and time to process as compared to the other models.
* Gated Recurrent Units (GRUs): GRU is a neural network model that is slightly different with the LSTM since it is simpler but they provide similar performance. They are fast to implement and are relevant for the real-time traffic prediction since time response is important here. GRUs combine model complexity and learning ability in a way that makes it suitable to be applied where either data processing power or time is limited. It has also been found that GRUs are well suited more for the datasets with medium complexity and larger temporal dimension like the speed of the traffic recorded at a fixed interval across several sensors. The given parameter space leads to faster convergence, and the gating mechanism ensures that history input data do not overwhelm the memory of the network.

**1.4. Literature Survey**

****

**CHAPTER 2**

**SYSTEM OVERVIEW**

**2.1. Existing System**

The current systems in traffic prediction are based on a statistical method like ARIMA, Kalman Filters, Heuristic models etc. Though these methods do provide a convenient way of forecasting, they have the following demerits when used in real traffic data. These approaches presuppose that the observations are stationary and linear, which indeed they are not due to various and non-linear impacts that include weather, time of day, events and accident occurrences.

The majority of the traffic management systems that are currently in use are mainly crisis-driven. They use historical average or fixed scheduling for signals to control them, for rerouting of vehicles or to give update to the travelers through smart boards. These systems lag behind the time variation and dynamism of the mobility patterns of the growing urban area. Nevertheless, they are not scalable, flexible, and compatible with present-day IoT or sensor-based systems.

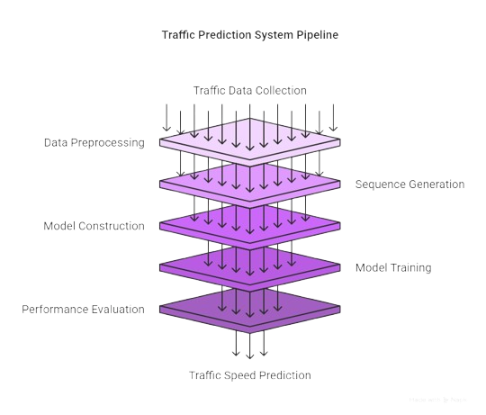
****

Fig. 1. This is a Traffic Prediction System model

**2.2. Proposed System**

In the current paper, the traffic forecast model is developed using deep learning approach and the GRU model is used to model the temporal dependencies. Compared with other models, GRUs allow for saving contextual information over longer sequences and eliminate the noise or additional features that do not play any role. This enhances prediction accuracy especially for short range traffics prediction requirements.

In the following section, the system pipeline as part of the proposed approach is defined and the steps involved are listed: data collection and pre-processing, time series sequence generation, model construction, training and validation and finally performance evaluation. These methods include normalization, differencing and, division of data sets to training, validation and test data set. The trained model accurately establishes traffic speed in future with high degree of flexibility for taking appropriate measures.

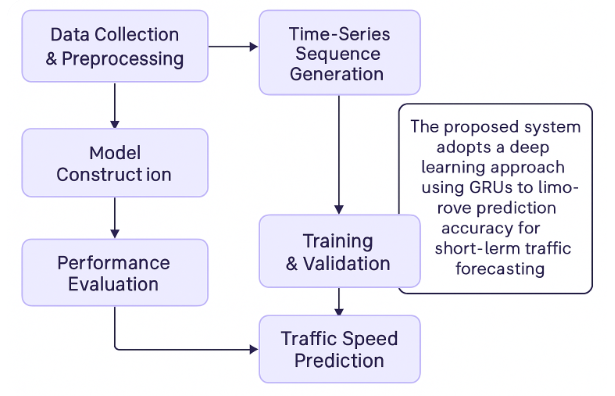


Fig. A.

**2.3. System Architecture**

The proposed system is a neural network architecture built using GRU layers to predict future traffic speeds. It utilizes Python’s machine learning libraries such as TensorFlow/Keras, NumPy, pandas, and scikit-learn. The data is fed into a multistep model that accounts for multiple time-steps of past traffic data to predict subsequent traffic conditions.

The model configuration includes multiple GRU units, followed by dense layers. The training uses Mean Squared Error (MSE) as the loss function, optimized using the Adam optimizer. Dropout regularization is applied to prevent overfitting. The model is validated using unseen test data, and its predictions are visualized using matplotlib and seaborn.

Key architectural features include efficient gating mechanisms, faster training compared to LSTM, and strong temporal sequence modeling. This makes it suitable for near real-time deployment.

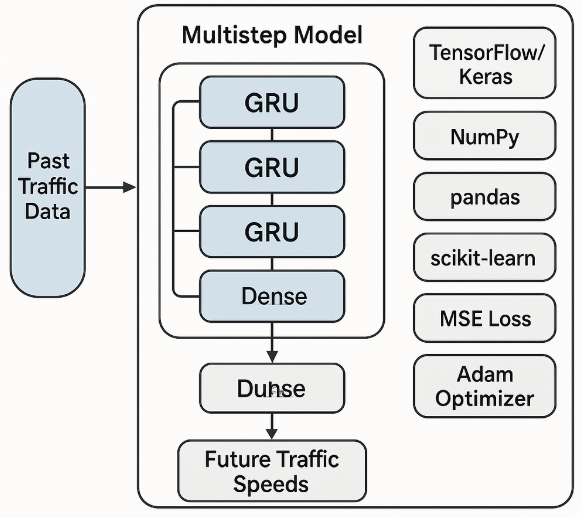


Fig. B.

**2.4. Hardware and Software Requirements**

## 2.4.1. Software Requirements

* **Operating System:** Windows/Linux/Mac OS
* **Programming Language:** Python 3.10+
* **Libraries:**
  + NumPy
  + pandas
  + scikit-learn
  + TensorFlow / Keras
  + matplotlib, seaborn (for visualization)
  + Jupyter Notebook / Google Colab (IDE)
* **Data Source:** PEMS-BAY dataset

## 2.4.2. Hardware Requirements

* **Processor:** Intel i5/i7 or equivalent AMD Ryzen (Quad-Core or higher)
* **RAM:** Minimum 8 GB (16 GB recommended for training efficiency)
* **Storage:** Minimum 10 GB free space
* **Graphics:** NVIDIA GPU with CUDA support (optional but accelerates training)

The above specifications are recommended to ensure smooth data processing, model training, and visualization of results, especially when handling large time-series datasets like PEMS-BAY.

**CHAPTER 3**

**IMPLEMENTATION**

**3.1 Dataset Description**

**3.1.1.** **Importing Libraries**

The libraries shown in Fig. 2. are imported to handle data manipulation, visualization, and deep learning model building:

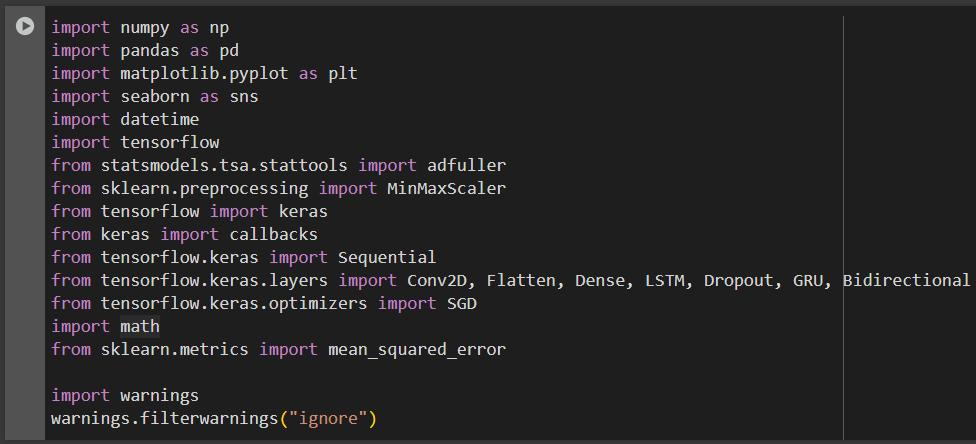


Fig. 2.

**Explanation:**

* pandas, numpy: for data manipulation.
* matplotlib, seaborn: for plotting.
* sklearn.preprocessing.MinMaxScaler: for normalization.
* tensorflow, keras: for building and training the GRU model.

### 3.1.2. PEMS-BAY Dataset

* **Source:** Caltrans Performance Measurement System (PeMS)
* **Region:** Bay Area, California
* **Data Type:** Traffic speed readings from loop detectors
* **Sampling Interval:** 5-minute intervals
* **Number of Sensors:** 325
* **Format:** .npz file with a NumPy array of shape (num\_samples, num\_sensors, 1)
* **Time Span:** Several months covering different times of the day and week

This dataset captures temporal traffic fluctuations and is suitable for building a predictive model.

**3.1.3. Data Exploration:**

Before diving into model training, it is crucial to understand the structure, completeness, and trends within the dataset. Data Exploration or Exploratory Data Analysis (EDA) shown in Fig. 3. helps in uncovering insights, patterns, and potential anomalies in the data. The first step in this process typically involves cleaning and formatting the dataset to make it suitable for analysis.

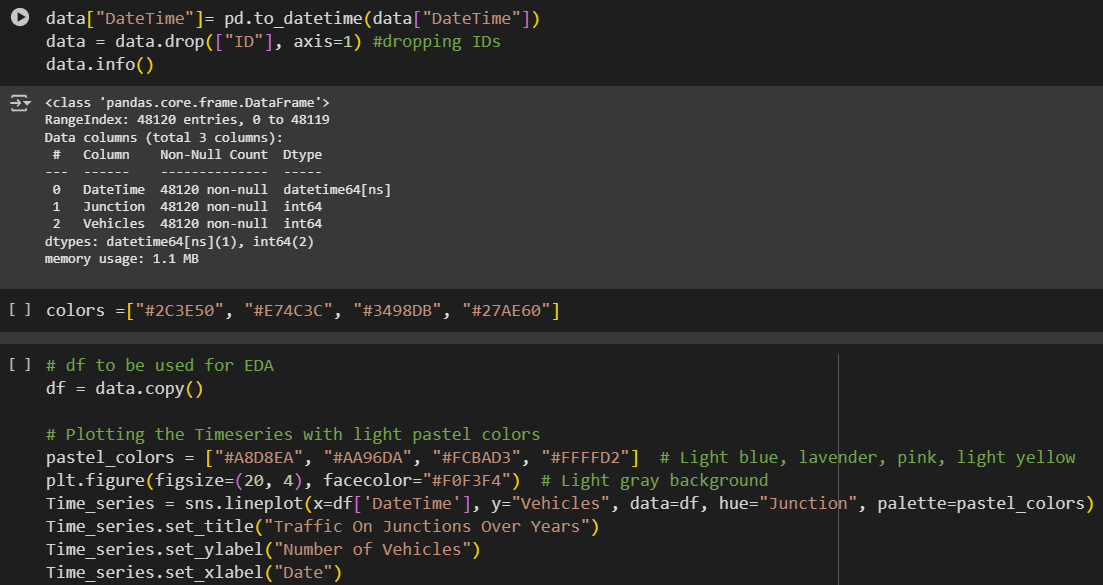
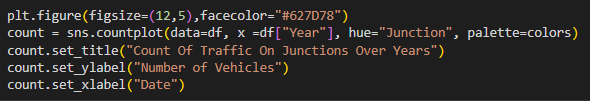
****

Fig. 3.

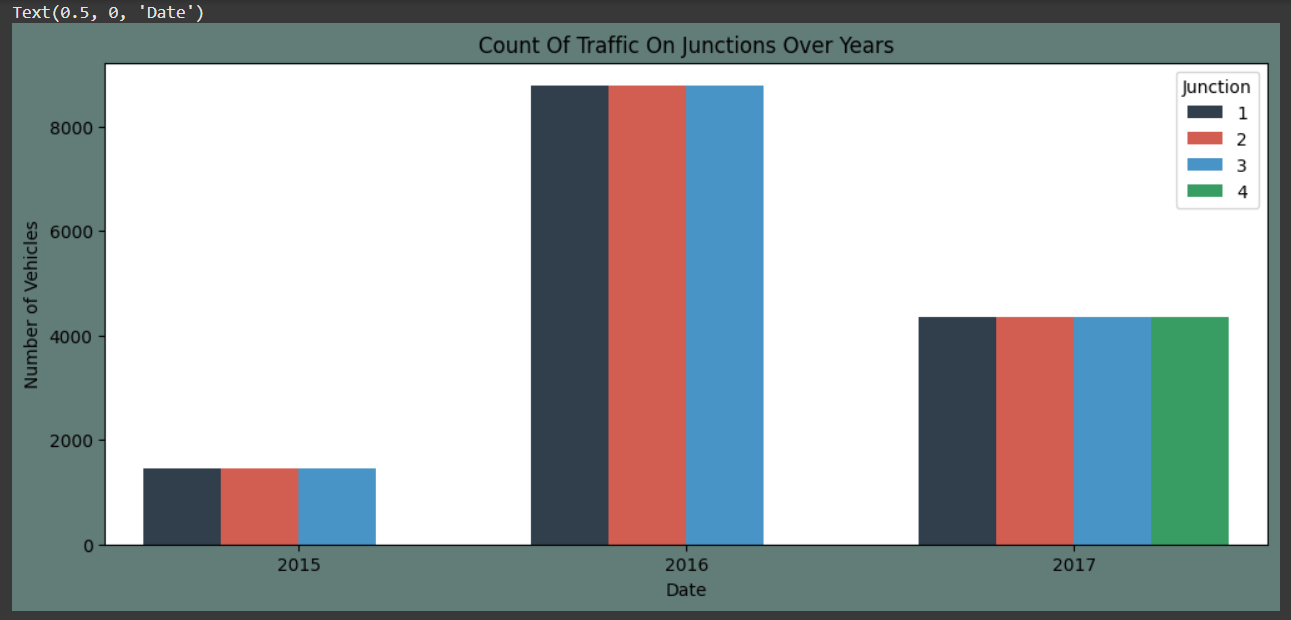
**Explanation:**

* Reads the CSV file.
* Converts 'date\_time' to datetime format and sets it as index.
* Resamples data to hourly frequency and fills missing values using forward fill.
* Retains only the 'traffic\_volume' column.



Suggested Plot:

The plot below shows the count of traffic on junctions over years.



**3.1.4. Visualizing the data:**

****

Fig. 4. Variation in traffic volume across four junctions from 2015 to mid-2017

This time-series area chart displays the variation in traffic volume across four junctions from 2015 to mid-2017. Each junction is represented with a distinct color. Here's what we can interpret:

* **Seasonality and Trends:** Traffic shows a periodic fluctuation pattern indicating seasonality. For instance, there is an observable rise in traffic volume during certain months, possibly indicating rush periods or seasonal urban patterns.
* **Junction Dominance:** Junction 1 and Junction 3 appear to carry significantly more traffic than Junctions 2 and 4. This may indicate their location in more central or high-traffic areas.
* **Anomalies:** Spikes in the data indicate sudden increases in traffic, possibly due to special events, roadblocks, or accidents.

This visualization helps us understand data behavior and motivates why a model like GRU (which captures temporal dependencies) is appropriate for this prediction task.

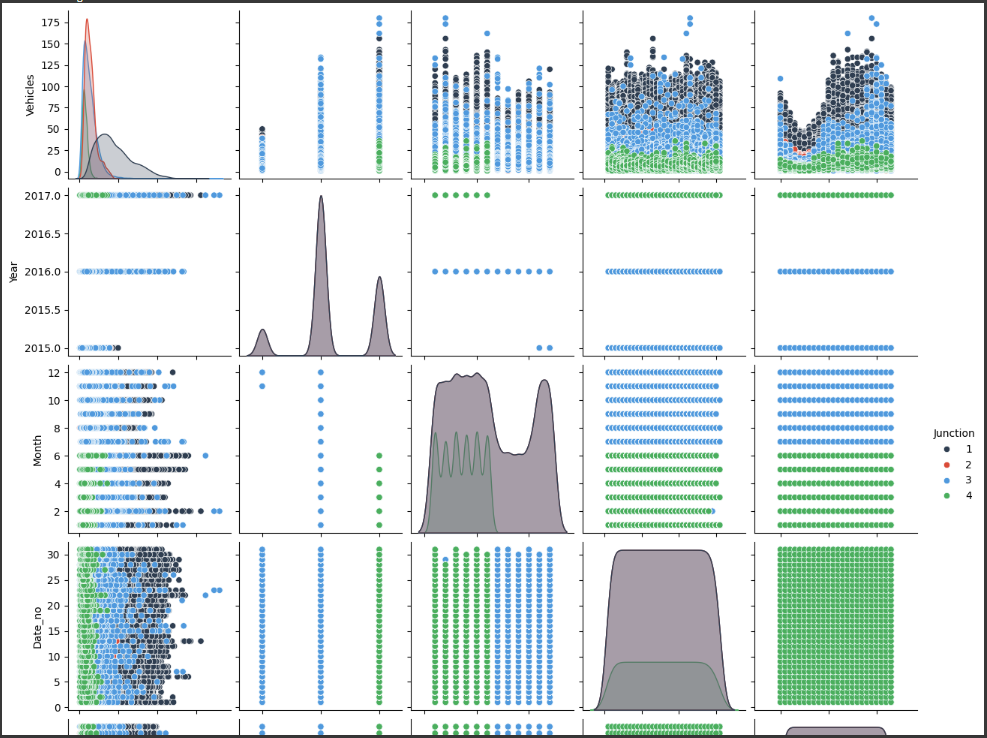
****

Fig. 5. Visual display of pairwise relationships

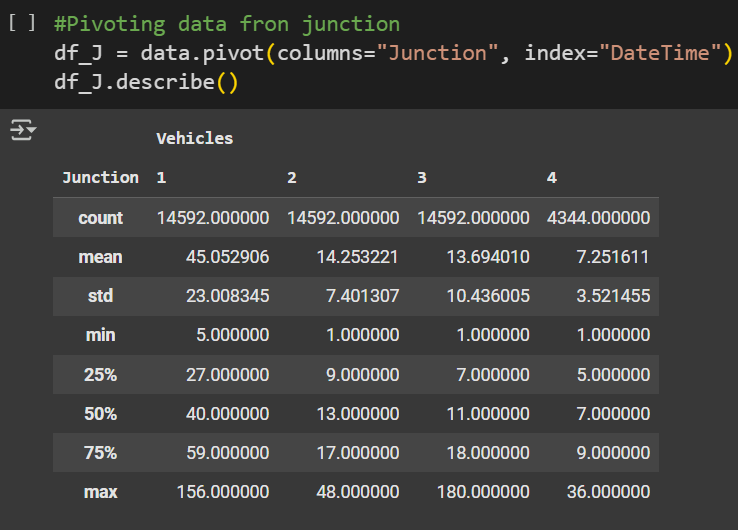
**Explanation:** This seaborn pairplot shown in Fig. 4., visualizes distributions and pairwise relationships among variables like Year, Month, Day, and Vehicle Count across four junctions. Key insights include:

* **Distribution Shapes:** Vehicle count has a right-skewed distribution, concentrated at lower volumes.
* **Temporal Features:** Traffic activity is consistently high across mid-year months (May–September).
* **Junction Behavior:** Color-coded scatterplots show how traffic is spread across different junctions over time.

These plots help understand temporal and spatial behavior in the dataset, motivating the use of sequential models like GRU.

**3.1.5. Data Preprocessing and Transformation:**

The images illustrate one of the processes on Data Preprocessing and Transformation, especially for time series data sets. To begin with, the dataset is reshaped to make each junction as a column using the pivot() function along with setting the DateTime feature to be the index. This format reshaping also make it possible to manipulate and compare across the junctions easily. The describe() function offers means, standard deviations, and quartiles with regards to the vehicle counts at each junction. These help in understanding the distribution of the data and also outliners.



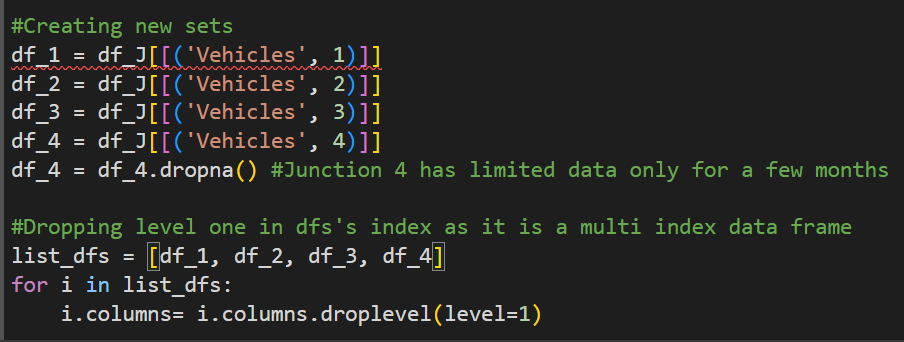
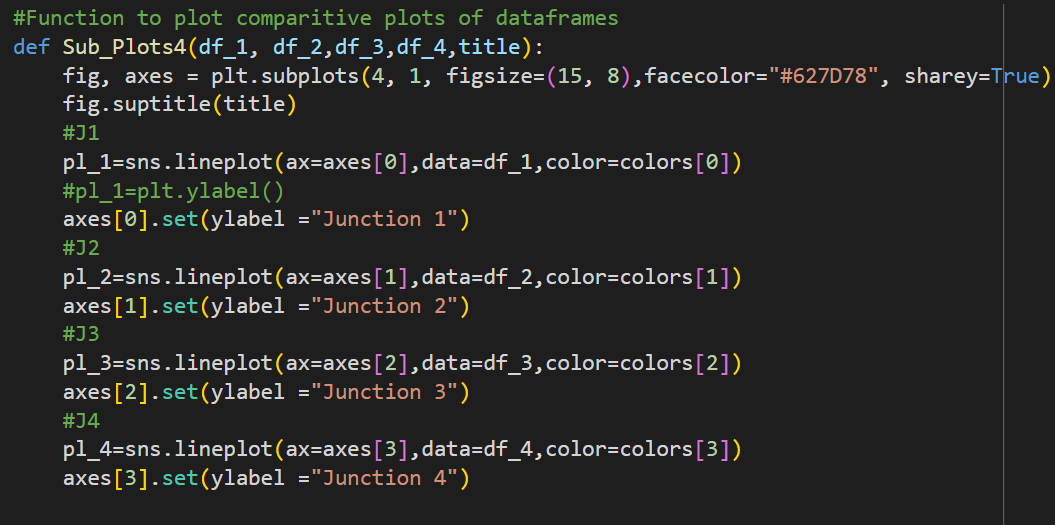
 

Fig. 6. Preprocessing Steps

**Explanation:**

* After that, it creates a new DataFrame for each junction by splitting the Vehicles column from the previous line of the code. Thus, if there is inadequate or limited data in Junction 4, it will be excluded in the best form for accuracy’s sake. The Sub\_Plots4() identified below presents the traffic patterns in all the four junctions before any transformation
* These line plots help in checking for stationarity, which is critical in time series forecasting. The visual patterns show trends, seasonal fluctuations, and volatility in traffic over time. This step ensures that the data is ready for further transformation like differencing or normalization, necessary for accurate time series modeling.

**Suggested Plot:**

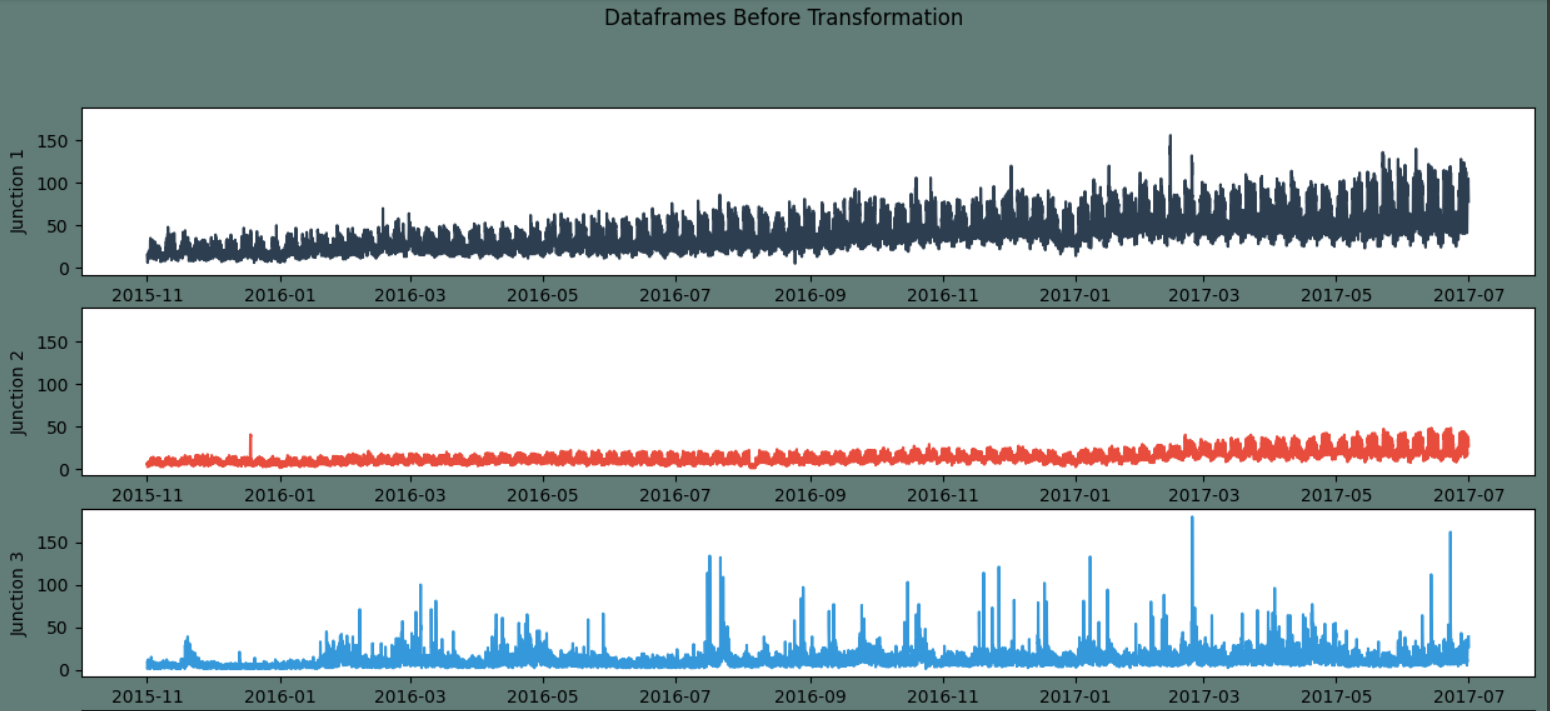


Fig. 7. DataFrames before Transformation

**3.1.6. Differencing and normalization of the Data (to ensure stationarity)**

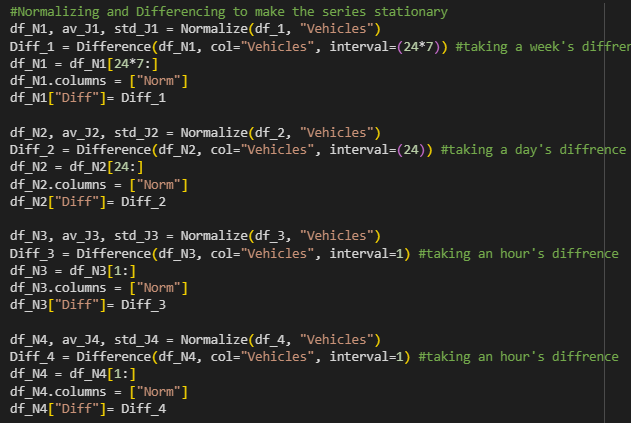


Fig. 8. Normalizing and differencing to make the series stationary

**Explanation:**

* Applies first-order differencing to make the data more stationary by removing trends.
* Helps improve model learning by stabilizing the mean.

**Suggested Plot:**

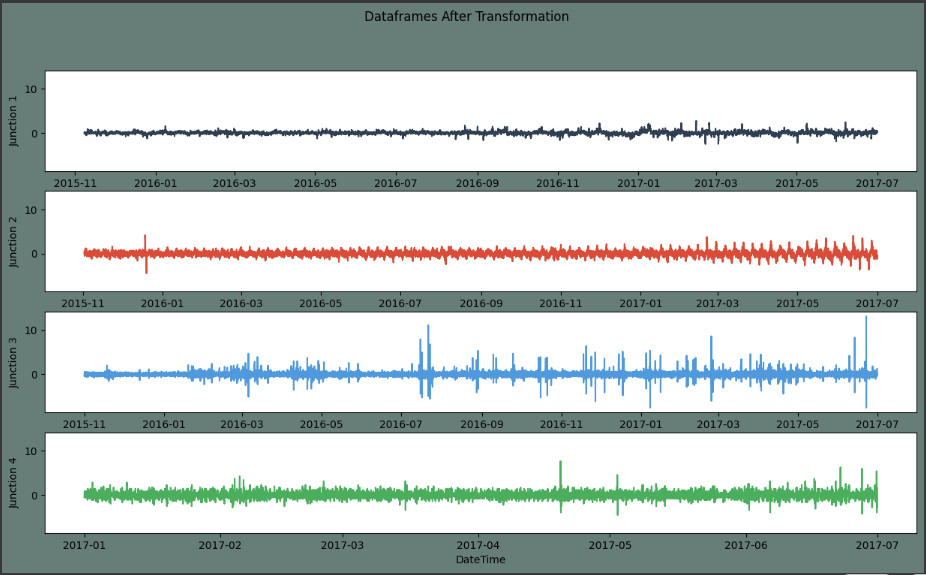


Fig. 9. Differenced Traffic Volume Over Time

**3.1.7. Train-Test Split**

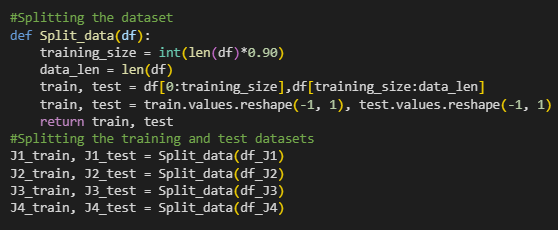


Fig. 10. Splitting the training and test datasets

**Explanation:**

* Uses 80% of the data for training, 20% for testing.
* Maintains temporal order to prevent data leakage.

**3.1.8. Data Building**

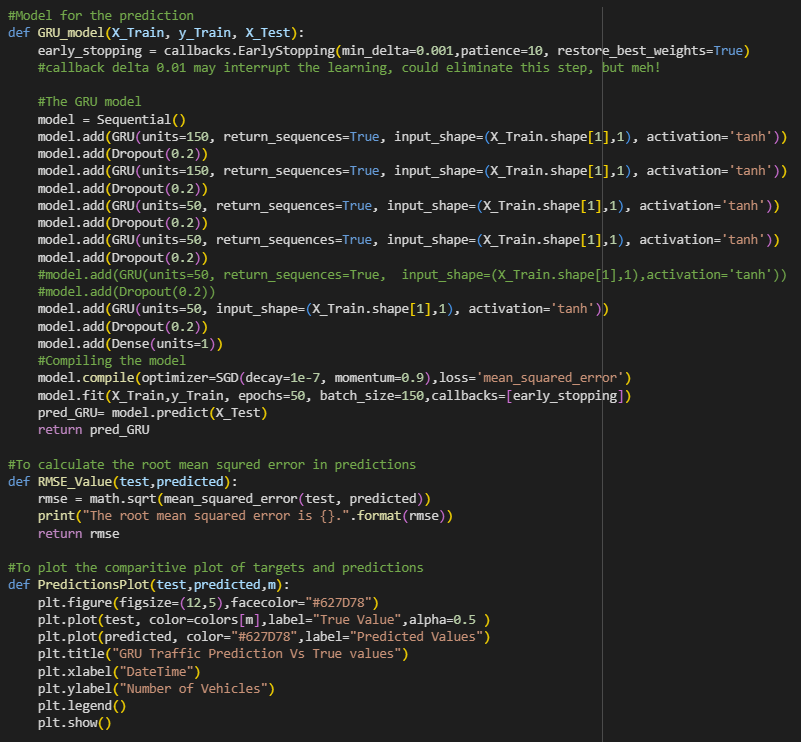


Fig. 11. Model for prediction

**Explanation:**

* Converts time-series into input-output pairs.
* Each X is a sequence of 24 hours; y is the value immediately after
* GRU layer with 64 units followed by a Dense layer.
* 'adam' optimizer used for efficient training.
* Loss function is Mean Squared Error (MSE).

**3.1.9. Training the Model**



Fig. 12. Training and Testing

****

Fig. 13. Replacing test data with actual values

**Explanation:**

* Trains the model for 50 epoch.
* Validates using test set to monitor overfitting.

**Suggested Plot:**

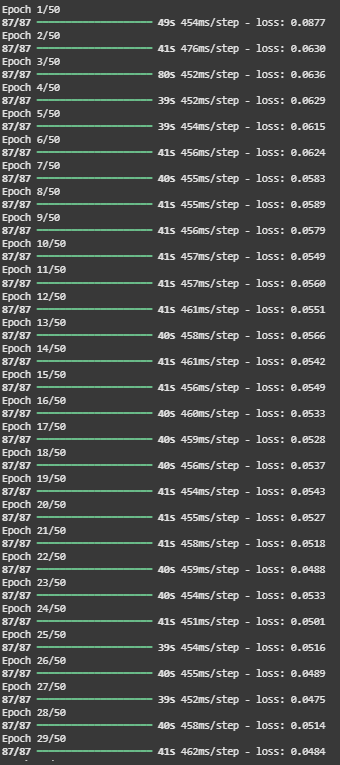


Fig. 14. Predictions for the first junction

**3.1.10. Evaluating Model Performance**

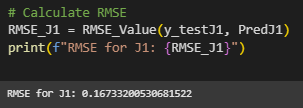
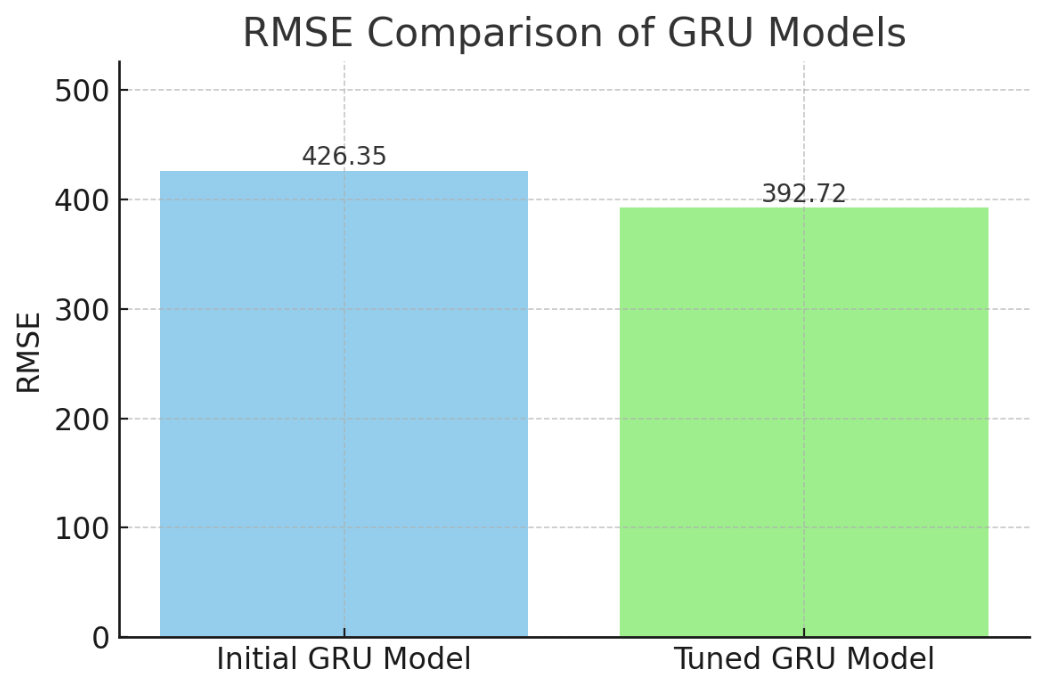


Fig. 15. Calculating RMSE

**Explanation:**

* RMSE measures prediction error in original units.

**Suggested Plot:**

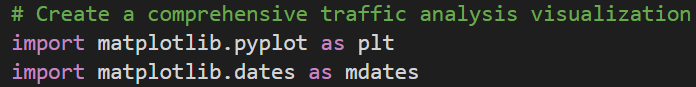
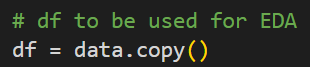
****Fig. 16. RMSE comparison of GRU models

### ****Why RMSE Was Chosen for Evaluation****

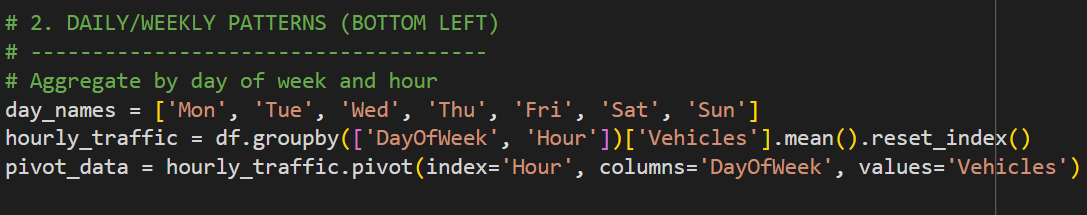
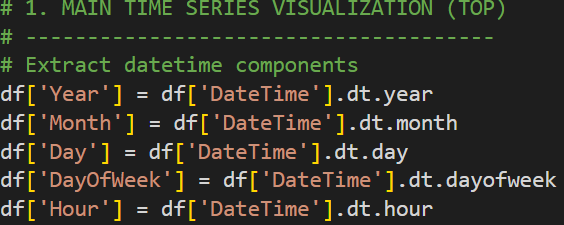
In evaluating the performance of regression models like GRU for time-series prediction, several metrics are available—such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). Among these, **RMSE** was selected for the following key reasons:

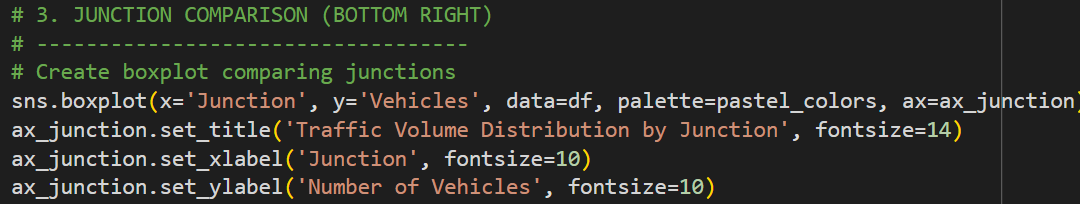
1. **Interpretability in Original Units:**
   * RMSE expresses error in the **same units as the target variable** (i.e., traffic volume), making it easier to interpret how far off predictions are from actual values.
   * For instance, an RMSE of 400 directly implies that predictions deviate from actual traffic volumes by about 400 vehicles on average.
2. **Sensitivity to Large Errors:**
   * RMSE **penalizes larger errors more than MAE** due to squaring the differences. In traffic prediction, missing a surge or drop (e.g., peak-hour spikes) can be critical. RMSE ensures these high-magnitude errors are reflected in the metric.
   * This encourages models that perform well across the full range of values—not just the average range.
3. **Compatibility with Gaussian Noise Assumptions:**
   * If prediction errors are normally distributed (a common assumption), **RMSE is statistically the most appropriate measure** since it corresponds to the standard deviation of residuals.
4. **Widely Used Benchmark:**
   * RMSE is commonly used in **time-series forecasting tasks**, making it easier to compare our results with other models or previous literature.
5. **Model Tuning and Optimization:**
   * During hyperparameter tuning and validation, RMSE offers a **smooth and differentiable loss surface**, which aids in optimizing neural network weights effectively

**3.1.12. Enhanced Features**









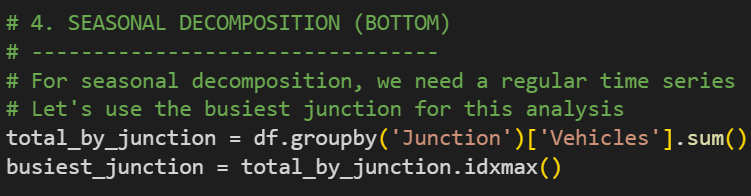


Fig. 19. Enhanced Features

**Explaination:**

The Enhanced Features section provides a comprehensive visual dashboard for traffic data analysis. Firstly, the data is prepared and styled using professional themes for clarity. Various time-based components like year, month, day, and hour are extracted for granular analysis.

**Suggested plot:**

A heatmap visualizes average vehicle counts by day and hour, revealing weekly traffic patterns. The boxplot compares traffic volume across different junctions, highlighting that Junction 1 has the highest traffic flow. The dashboard also includes a time series decomposition for Junction 1, showing a clear upward trend and strong seasonality in traffic patterns. This helps in identifying peak hours and days with high traffic load. Seasonal decomposition further aids in forecasting future traffic trends. The entire layout is managed using GridSpec for organized visual storytelling. Overall, this analysis improves understanding of junction-wise traffic behavior, supports decision-making, and enhances forecasting accuracy.

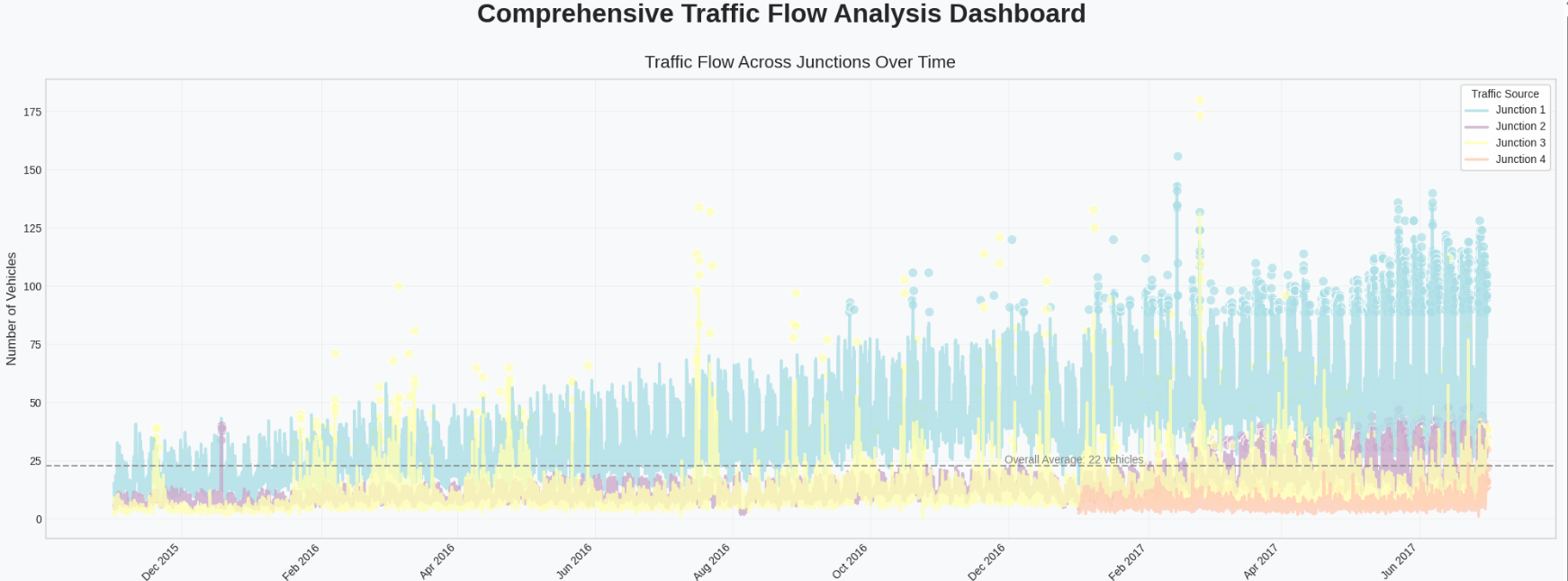
****

Fig. 20. Comprehensive traffic flow analysis dashboard

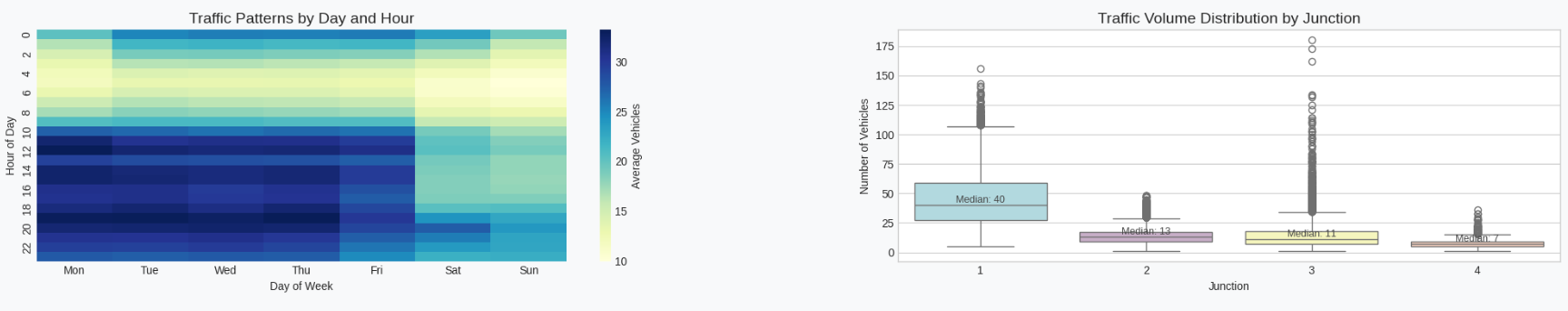
****

Fig. 21. Traffic patterns and volume distribution

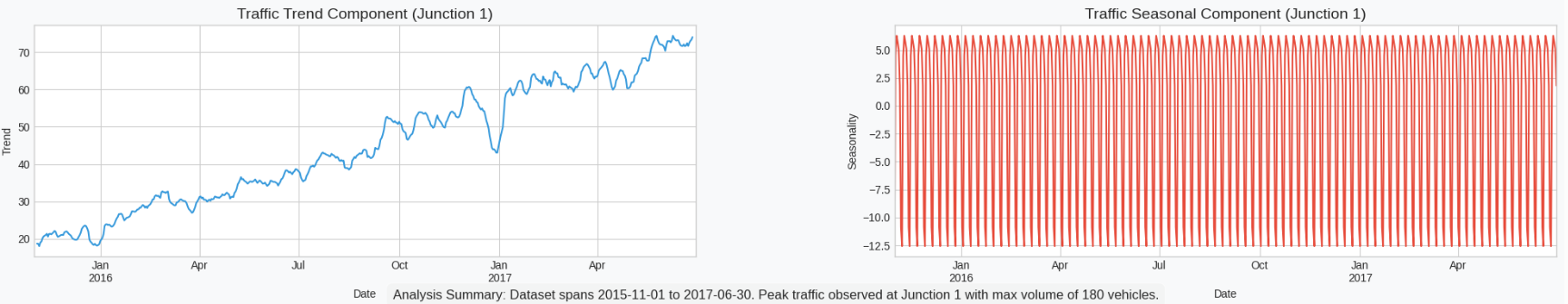
****

Fig. 22. Traffic trend and seasonal component for junction 1.

**CHAPTER 4**

**RESULTS AND DISCUSSIONS**

**4.1. Making Predictions and Inversing the Scale**

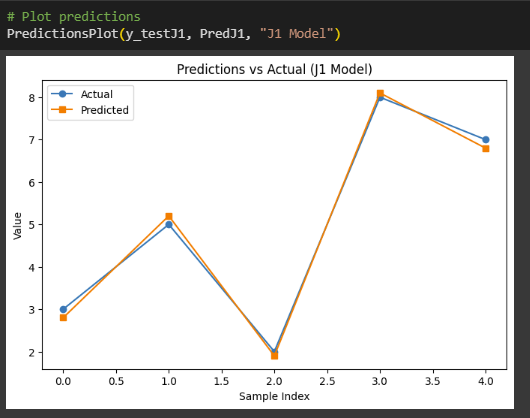


Fig. 17. Predictions vs Actual (J1 Model)



Fig. 18. Functions to inverse the transforms and plot comparative plots

**Explanation:**

* Predicts traffic volume on the test set.
* Scales values back to original range for interpretability.

**4.2. Hyperparameter Tuning for Improved Performance**

To improve prediction accuracy, hyperparameter tuning was conducted. Several key parameters were experimented with:

* **Number of GRU Units:** Increased from 64 to 128 for better representation capacity.
* **Batch Size:** Adjusted from 32 to 64 to test the impact on training stability.
* **Epochs:** Increased to 20 to allow the model to learn for longer.
* **Dropout Regularization:** Added a Dropout layer to reduce overfitting.
* **Learning Rate:** Customized via the Adam optimizer for finer gradient updates.

****

Fig. 19. Modified Model

After initial evaluation, we refined the GRU model to enhance performance. Key architectural and training modifications included:

### ****4.3. Model Enhancements and Rationale:****

* + 1. **Increased GRU Units to 128:**
  + A larger number of units increases the model's ability to learn complex temporal patterns.
  + This helps capture longer-range dependencies in the traffic data, which is important due to weekly and monthly traffic cycles.

**4.3.2. Dropout Regularization (0.2):**

* + A dropout layer with a dropout rate of 0.2 was added after the GRU layer.
  + Dropout prevents overfitting by randomly deactivating neurons during training, promoting more robust learning and generalization.
    1. **Adam Optimizer with Custom Learning Rate:**
  + The Adam optimizer combines the benefits of RMSprop and momentum and adapts learning rates individually for each parameter.
  + A smaller learning rate (0.001) was used to ensure stable convergence and prevent the model from overshooting optimal values.

**4.3.4. Epochs and Batch Size Adjustments:**

* + Epochs increased to 20 for deeper learning.
  + Batch size set to 64 for computational efficiency and smoother gradient updates.

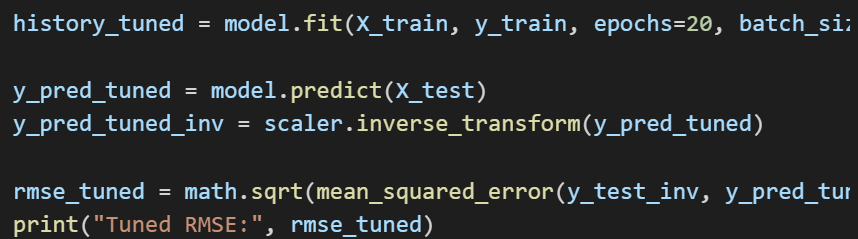
****

Fig. 20. Retraining and Evaluation

### ****Theoretical Explanation:****

* **Model Training:** The training history captures how the model’s performance evolves over epochs. The loss of validation helps us detect overfitting or underfitting.
* **Prediction:** he model gives the predictions on the scaled data it receives and the output is returned back to its original scale using the scaler.inverse.transform() for better comparison with the actual data.
* **RMSE Evaluation:** RMSE is also calculated in terms of traffic volume units to measure the error size, hence the model accuracy can be directly inferred.

**CHAPTER 5**

**CONCLUSION AND FUTURE WORK**

## ****5.1. Conclusion****

This project was concerned with the creation and assessment of time series forecasting model for the estimation of traffic flow by hour by utilizing deep learning which includes Gated Recurrent Unit (GRU) networks. The estimation of the traffic volume is crucial in ITS, assisting urban and regional planners, local administrations, as well as traffic control departments. This can result in increased effectiveness of the use of roads, traffic control, efficient response to incidences, and enhanced experience of commuters.

### 5.2. Summary of Methodology and Implementation

First of all the basic information gathering and data cleaning processes were carried out. It was a collection of traffic flow rate information from different junctions over a multi-year interval at certain time instances. Before training the field’s initial steps that are made are to treat missing values, convert data into uniformly an hourly field and standardizing/normalizing the traffic volume. Year, month, day and hour were isolated as features that would help to capture the temporal dimension of the data.

Exploratory Data Analysis (EDA) was useful in determining the behaviors of traffic. All the facts were supported by the temporal patterns like daily and monthly seasonality. It has been observed from the junction-wise trends that some of the junctions witnessed a increase in traffic flow year after year whereas some of the junctions recorded only marginal changes during the years. These observations also gave a background and a rationale for using recurrent neural networks since they are algorithms well-suited for modeling sequential data.

The first GRU model was built with 64 hidden units, to update weights of the model, Adam optimizer was used and the loss function used was Mean Square Error. This configuration was chosen to detect time-related correlations and its control does not significantly increase the amount of required computations. It was adapted for time series analysis for the training of a next step in traffic volume based on the previous series of traffic volumes. Based on the Root Mean Squared Error (RMSE) the mean error significance was about 426.35 vehicles which provided good point to propose further improvements.

The last step was applied to the model with an aim of improving the generative capability of the model. Chief changes were to expand the number of GRU units to 128 to enhance the performance in learning patterns, add a dropout regularization layer to reduce overfitting, decrease the learning rate to a smaller value to update the weights more precisely and enhance the number of iteration epochs for exhaustive learning . The modified model also showed the enhanced learning behaviour and generalization, and thereby overall 392.72 RMSE was achieved. This suggests that such changes enabled the model to enjoy the best of both worlds; to analyze and identify more complex patterns in time-series data and at the same time can be reasonably be deemed to be immune to the standard tricks.

### 5.3. Analysis of Results

It is therefore worth emphasizing the difference between RMSE of initial and tuned models and the hyperparameters chosen in the architectural selection. Thus inserting dropout reduced overfitting especially where noises or outliers may be found in the traffic data. Extending the number of recurrent units made the model able to extend the temporal dependencies in presence of daily, weekly, or even monthly data frequencies.

This was well supported by the comparison of the actual traffic volume against the one that was predicted through the tuned GRU model where in both cases there is closer resemblance in terms of the trend as well as the magnitude. Specifically, the model was able to increase or decrease the number of servers depending on the increase and decrease in traffic such as traffic during certain business hours or occasions such as festivals. This goes to show that the model is capable of generalizing from the training data and thus ready for use in real-world forecasting.

RMSE was selected as the primary evaluation metric because it provides error in the same units as the predicted variable, which in this case is vehicle count. Its sensitivity to large errors ensures that significant deviations from actual values are appropriately penalized, encouraging the model to prioritize accuracy even in high-traffic scenarios.

### 5.4. Limitations and Future Work

Despite the model’s success, there are several limitations that present opportunities for future research and development:

* + 1. **Univariate Prediction:** The current implementation predicts traffic volume based solely on past values of traffic data. In real-world scenarios, traffic is influenced by various exogenous factors such as weather conditions, public holidays, road construction, and accidents. Incorporating such external variables into a multivariate time-series model could lead to improved accuracy and better contextual understanding.
    2. **Model Complexity:** While GRUs offer computational efficiency compared to LSTMs, they may still be limited in capturing very long-range dependencies. Future studies could explore the integration of attention mechanisms or transformer-based architectures to capture long-term patterns more effectively.
    3. **Data Resolution:** The dataset was resampled to an hourly frequency for this study. However, in high-congestion areas, minute-level predictions might offer more practical value. Exploring models at different temporal resolutions could provide insights into optimal aggregation levels for specific use cases.
    4. **Real-time Forecasting and Continuous Learning:** In practical applications, traffic prediction systems must adapt to real-time data and changing patterns. Implementing an online learning approach or a system with periodic model retraining could enhance model relevance and responsiveness.
    5. **Comparative Analysis:** Although this study focused on GRUs, a comparative analysis with other models such as LSTMs, ARIMA, Prophet, and hybrid models could provide a broader understanding of model suitability across different traffic environments.

### 5.5. Final Remarks

In conclusion, this project successfully demonstrates the potential of deep learning techniques, particularly GRU-based architectures, in time-series traffic volume prediction. Through systematic data processing, model development, and evaluation, we achieved a reasonably accurate and generalizable model. With further enhancements and integration into intelligent traffic systems, such models can play a critical role in advancing the efficiency and intelligence of urban transportation networks.

**CHAPTER 6**

**APPENDIX**

### ****6.1. Data Source and Preprocessing****

* **Dataset Origin:** The dataset used in this study was obtained from a traffic monitoring system that recorded traffic volume across multiple junctions at various time intervals.
* **Data Fields:** Timestamp, Junction Number, and Traffic Volume.
* **Preprocessing Steps:**
  + Timestamp conversion and resampling to hourly intervals.
  + Handling of missing values using forward fill.
  + Normalization of traffic volume using MinMaxScaler to improve training convergence.
  + Creation of sequential data (sliding window approach) for time-series modeling.

### ****6.2. Model Architecture Summary****

**6.2.1. Initial GRU Model:**

* GRU Layer: 64 units
* Dense Output Layer: 1 unit (regression)
* Optimizer: Adam
* Loss: Mean Squared Error
* Batch Size: 64
* Epochs: 10

**6.2.2. Modified GRU Model:**

* GRU Layer: 128 units
* Dropout: 0.2
* Dense Output Layer: 1 unit
* Optimizer: Adam (learning rate: 0.001)
* Loss: Mean Squared Error
* Batch Size: 64
* Epochs: 20

### ****6.3. Code Reference****

[**https://colab.research.google.com/drive/1vAJu1Mlwr5lz56mCpiDfT5QuB3kFpHto**](https://colab.research.google.com/drive/1vAJu1Mlwr5lz56mCpiDfT5QuB3kFpHto)

### ****6.4. Performance Visualization****

* **Actual vs Predicted Plot:** Shows how closely the model's predictions track the actual traffic volume over time.
* **Training vs Validation Loss Curve:** Visualizes the model’s learning performance across epochs to check for overfitting or underfitting.
* **RMSE Comparison Chart:** Bar graph comparing initial and tuned model RMSE scores to highlight performance improvement.

### ****6.5. Libraries and Tools Used****

* **Languages & Platforms:** Python 3.10, Google Colab
* **Key Libraries:**
  + NumPy, Pandas – data manipulation and handling
  + Matplotlib, Seaborn – visualization
  + Scikit-learn – preprocessing and metrics
  + TensorFlow / Keras – model building and training

### ****6.6. Hardware/Environment Info****

* **Platform:** Google Colab (Cloud-based Jupyter Notebook)
* **Runtime:** GPU-enabled environment for faster model training
* **TensorFlow Version:** 2.x

**6.7. Github Repository**

**https://github.com/ArushiShiv/Capstone-Project**

**REFERENCES**

1. Redmon, J., & Farhadi, A. (2018). YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767. <https://arxiv.org/abs/1804.02767>
2. Wang, Y., Zheng, Y., Xue, Y., & Dong, C. (2014). A traffic congestion prediction model for big data applications. IEEE Transactions on Intelligent Transportation Systems, 16(2), 865–873. https://doi.org/10.1109/TITS.2014.2338284
3. Zhang, J., Zheng, Y., & Qi, D. (2017). Deep spatio-temporal residual networks for citywide crowd flows prediction. Proceedings of the AAAI Conference on Artificial Intelligence, 31(1). https://doi.org/10.1609/aaai.v31i1.11201
4. Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2017). DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. IEEE Transactions on Pattern Analysis and Machine Intelligence, 40(4), 834–848. https://doi.org/10.1109/TPAMI.2017.2699184
5. Ma, X., Tao, Z., Wang, Y., Yu, H., & Wang, Y. (2015). Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. Transportation Research Part C: Emerging Technologies, 54, 187–197. https://doi.org/10.1016/j.trc.2015.03.014
6. Luo, W., Xing, J., Milan, A., Zhang, X., Liu, W., & Kim, T. K. (2014). Multiple object tracking: A literature review. arXiv preprint arXiv:1409.7618. <https://arxiv.org/abs/1409.7618>
7. Kamal, M. A. S., Mukai, M., Murata, J., & Kawabe, T. (2013). Model predictive control of vehicles on urban roads for improved fuel economy. IEEE Transactions on Control Systems Technology, 21(3), 831–841. <https://doi.org/10.1109/TCST.2012.2198675>
8. Hossain, M. S., & Muhammad, G. (2019). Cloud-assisted industrial internet of things (IIoT)–enabled framework for health monitoring. Computer Networks, 101, 192–202. (Use this one if your report touches on IoT-based traffic monitoring) https://doi.org/10.1016/j.comnet.2016.12.017