TRAFFIC FORECASTING – TIME SERIES USING RNN

Reshma Jayaprakash
Scifor Technologies Trainee
ID: STB03011

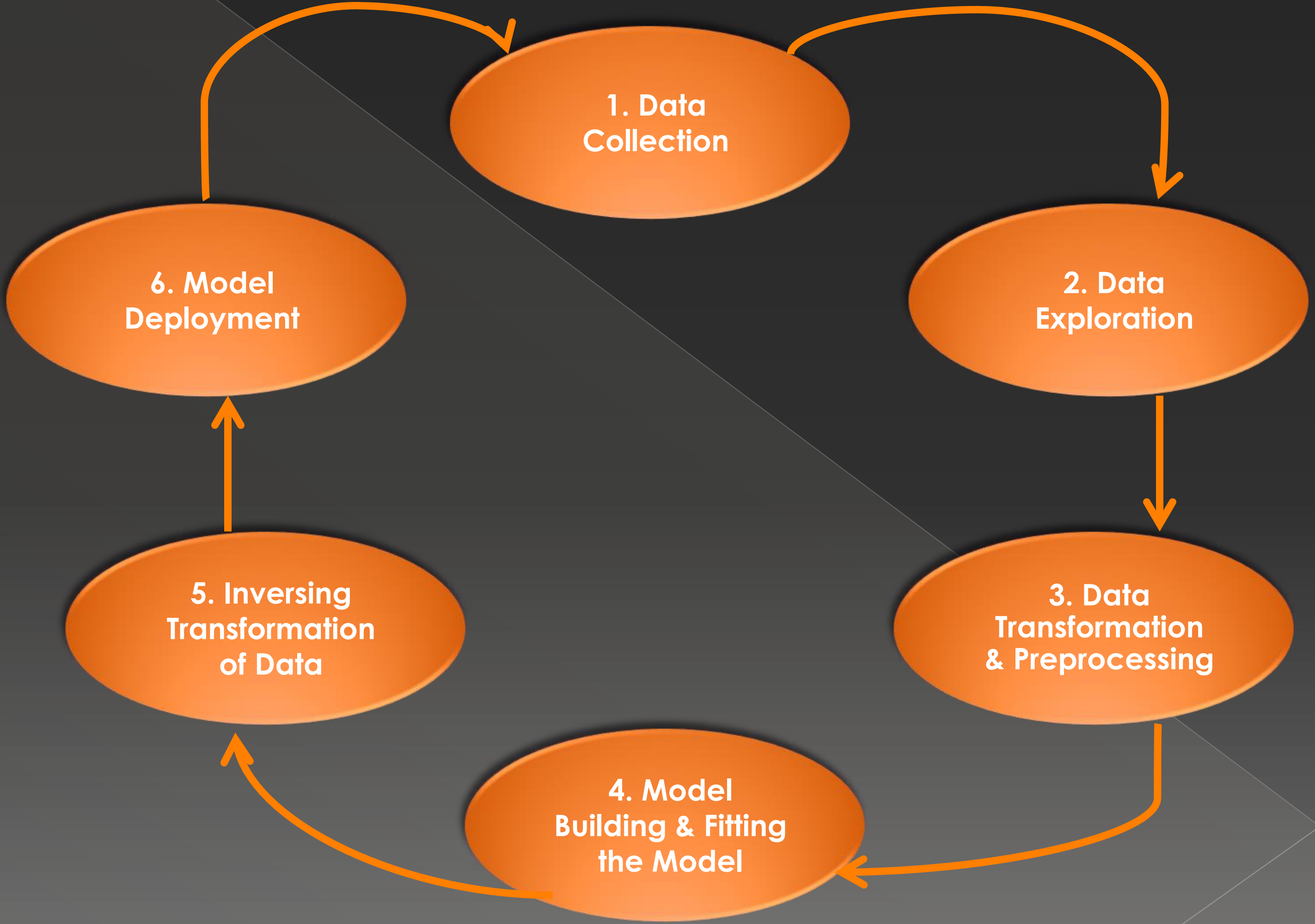
INTRODUCTION

- 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
- 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

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

PROJECT WORKFLOW


1. Data Collection
2. Data Exploration
3. Data Transformation & Preprocessing
4. Model Building & Fitting the Model
5. Inversing Transformation of Data
6. Model Deployment

DATA COLLECTION

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

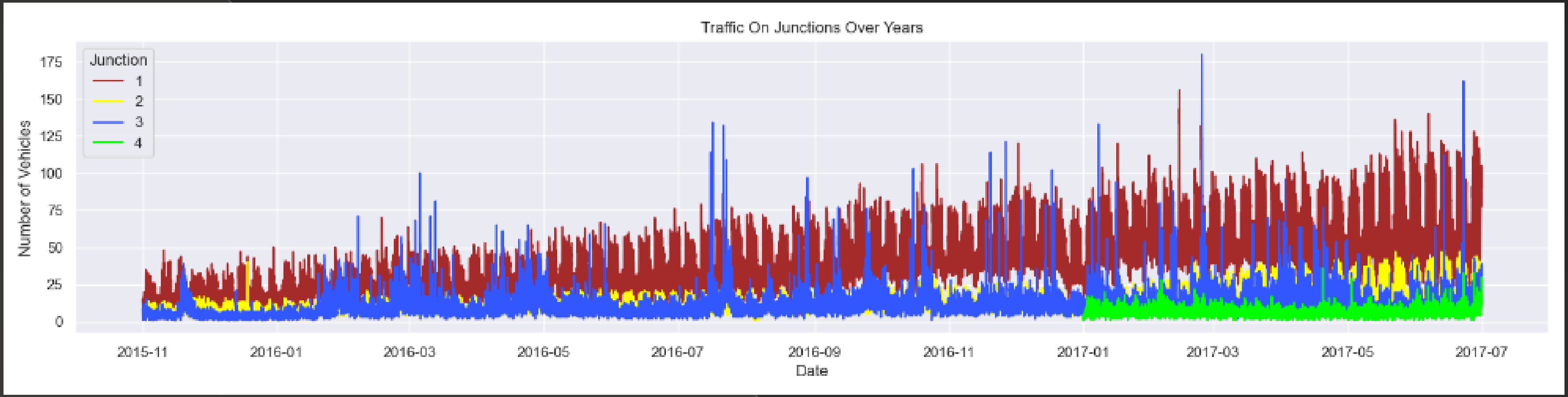
Initial Process,

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.



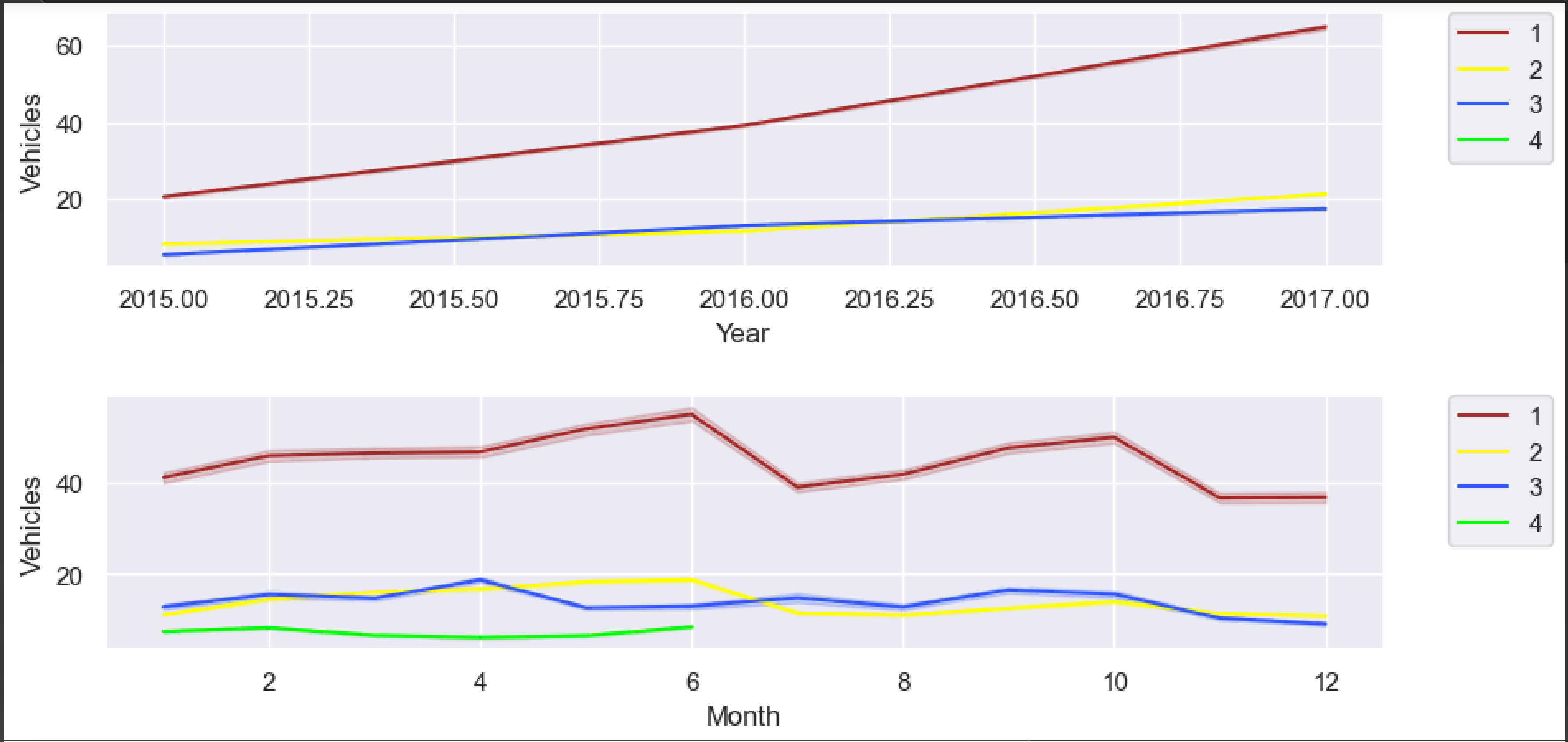
- Here, It can be seen that the Junction 1 is visibly having an upward trend
- The data for the Junction 4 is sparse starting only after 2017
- Seasonality is not evident from the above plot. So datetime composition has been explored to figure out more about it
- Trend and Seasonality are the components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

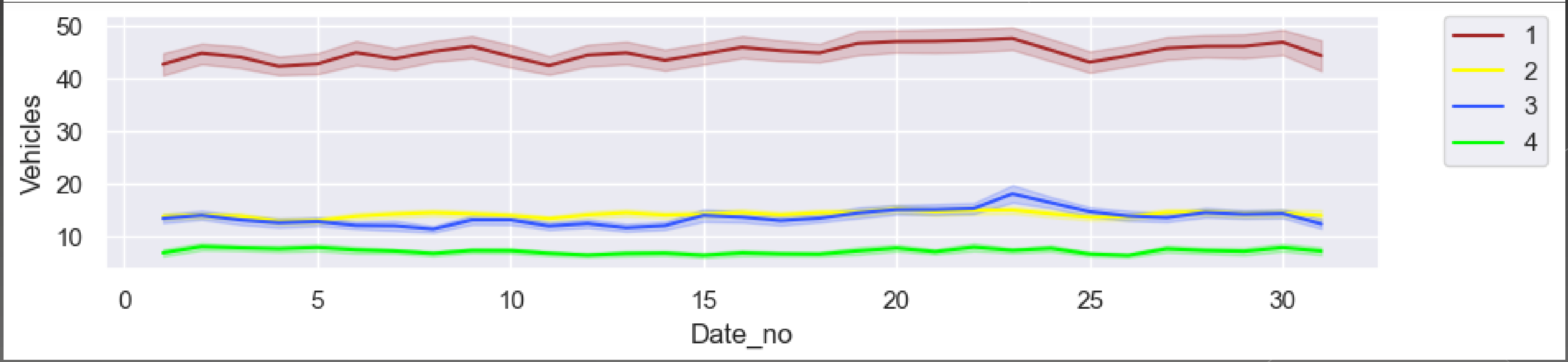
- Seasonality – is a periodic fluctuation that occurs in time series where predictable and regular patterns are exhibited at intervals
- Here, we have plotted seasonality graphs by doing Feature Engineering

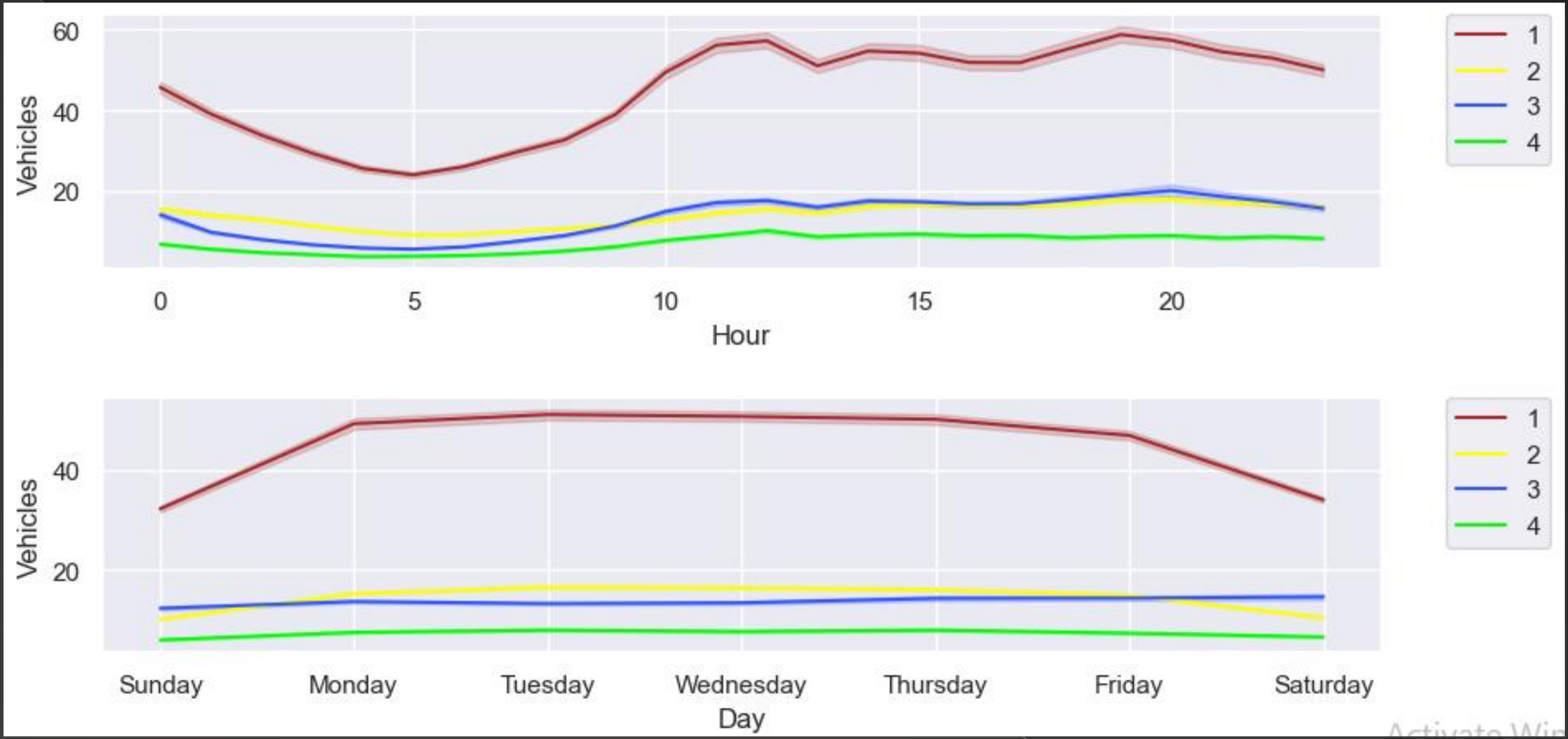
Feature Engineering for EDA – Created new features out of DateTime and plotted using seaborn library

New Features:

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


Vehicles
60
40
20
2015.00
2015.25
2015.50
2015.75
2016.00
2016.25
2016.50
2016.75
2017.00
Year
1
2
3
4
Vehicles
40
20
2
4
6
8
10
12
Month
1
2
3
4


Vehicles
50
40
30
20
10
0
5
10
15
20
25
30
Date_no
1
2
3
4



- 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

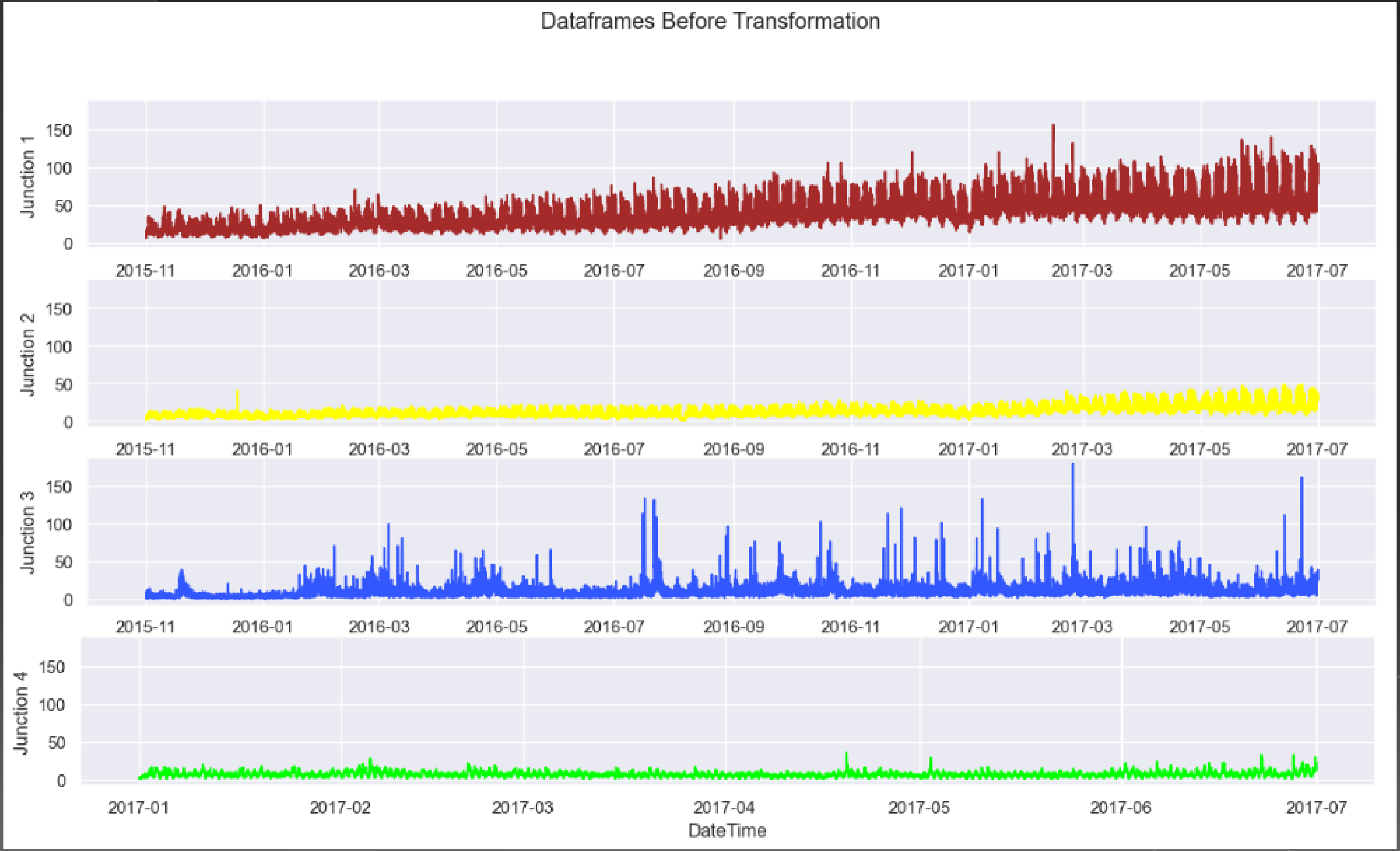
DATA TRANSFORMING & PREPROCESSING

- 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

Preprocessing Steps,

1. Created different frames for each Junction
2. Transforming the series
3. Performed the Augmented Dickey-Fuller test to check the seasonality of transformed series
4. Creating test and train sets

Different frames for each Junction

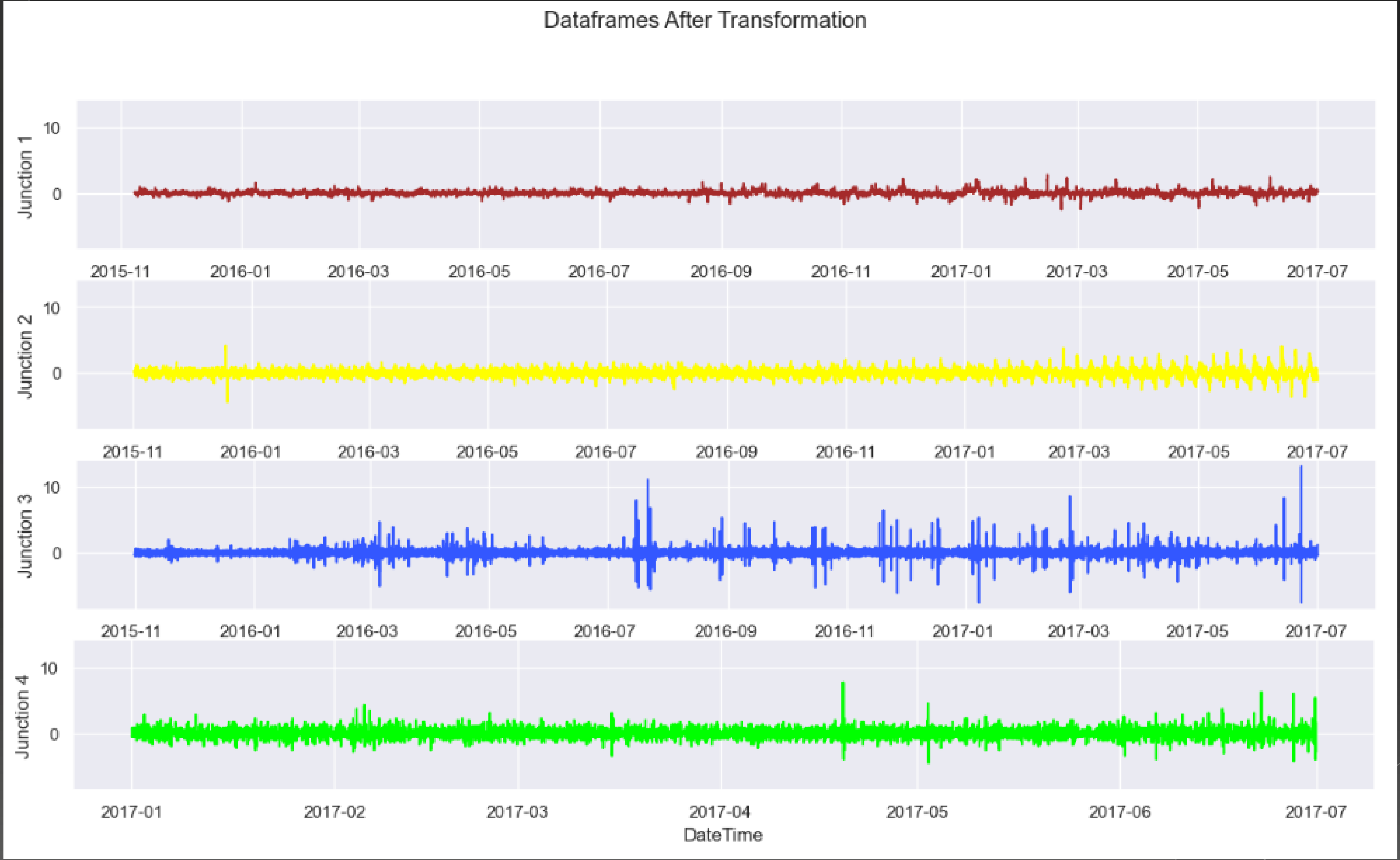


- A time series is stationary if it does not have a trend or seasonality
- From the analysis made, we have seen a weekly seasonality and an upwards trend over the years
- In the above plot, it is again seen that the Junctions one and two have an upward trend. If we limit the span we will be able to further see the weekly seasonality
- Hence Transforming – Normalizing and Differencing has applied

Transforming the series

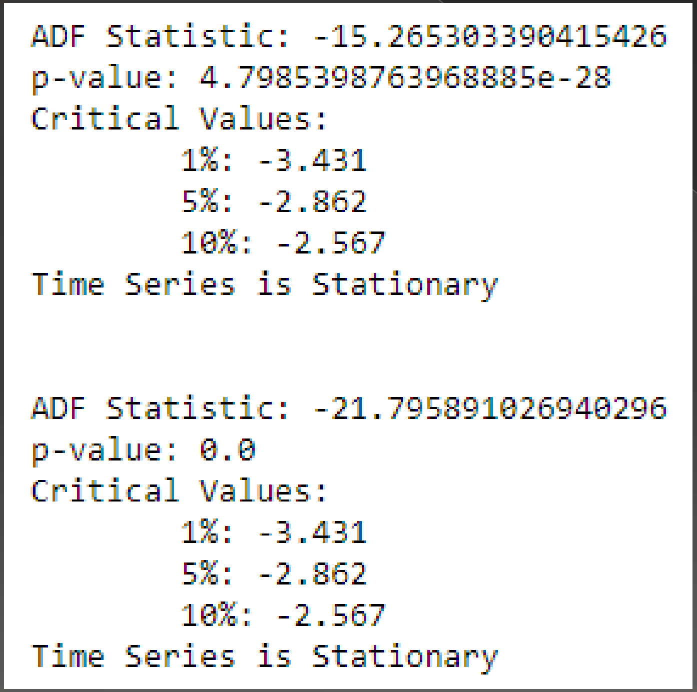
In accordance with the above observations, Differencing to eliminate the seasonality should be performed as follows:

- 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

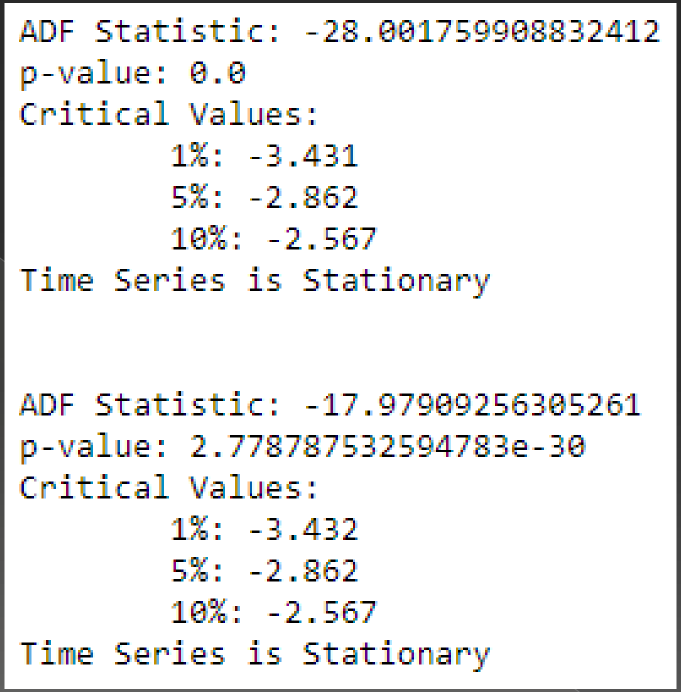

Dataframes After Transformation
Junction 1
Junction 2
Junction 3
Junction 4
10
0
2015-11
2016-01
2016-03
2016-05
2016-07
2016-09
2016-11
2017-01
2017-03
2017-05
2017-07
2017-02
2017-04
2017-06
DateTime

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: 4.798539876396885e-28
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: -28.001759908832412
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 & FITTING THE MODEL

- 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:

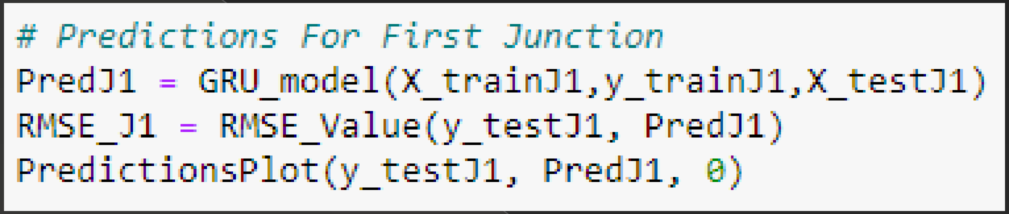
Rate: 0.2

Dense (Output) Layer:

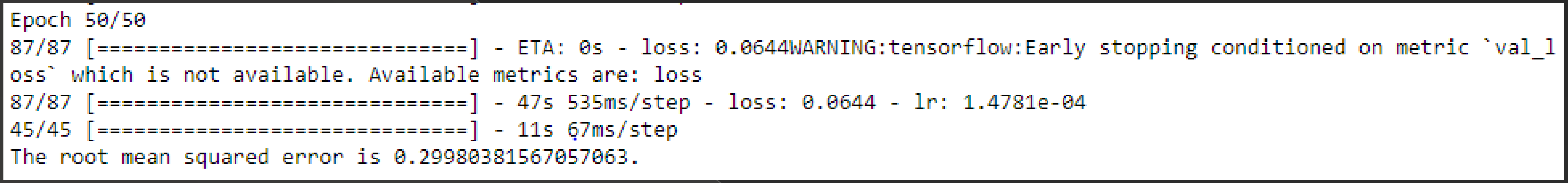
Number of Units: 1 (as it's a regression problem predicting a single value)

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.

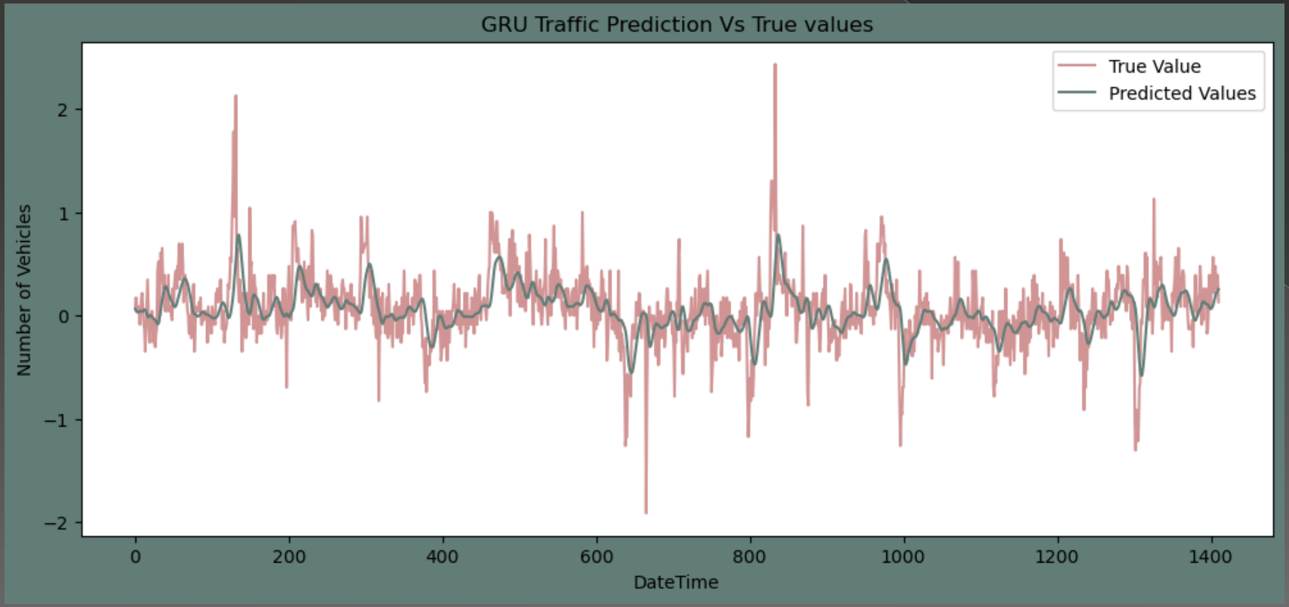
Model Fitting – Junction 1



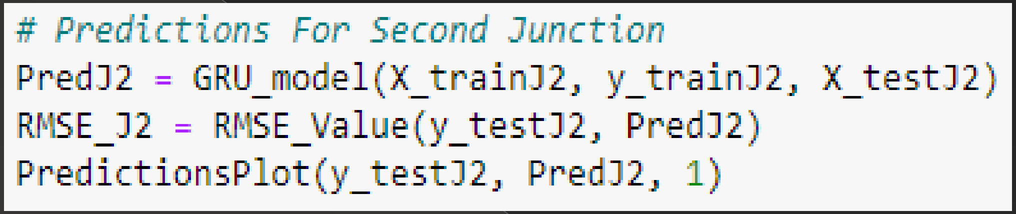
```
# Predictions For First Junction
PredJ1 = GRU_model(X_trainJ1,y_trainJ1,X_testJ1)
RMSE_J1 = RMSE_Value(y_testJ1, PredJ1)
PredictionsPlot(y_testJ1, PredJ1, 0)
```



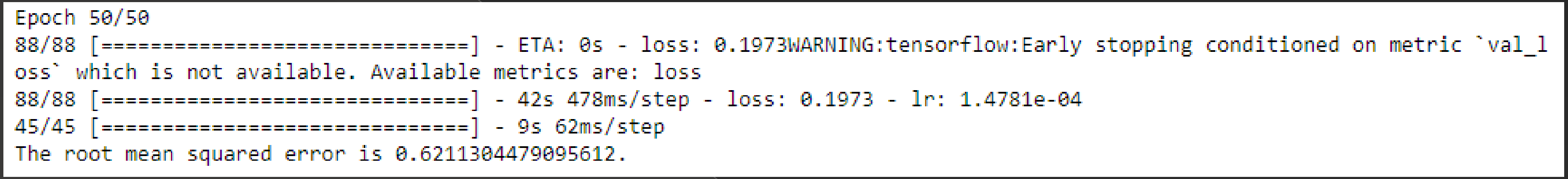
```
Epoch 50/50
87/87 [==============================] - ETA: 0s - loss: 0.0644WARNING:tensorflow:Early stopping conditioned on metric `val_l
oss` which is not available. Available metrics are: loss
87/87 [==============================] - 47s 535ms/step - loss: 0.0644 - lr: 1.4781e-04
45/45 [==============================]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



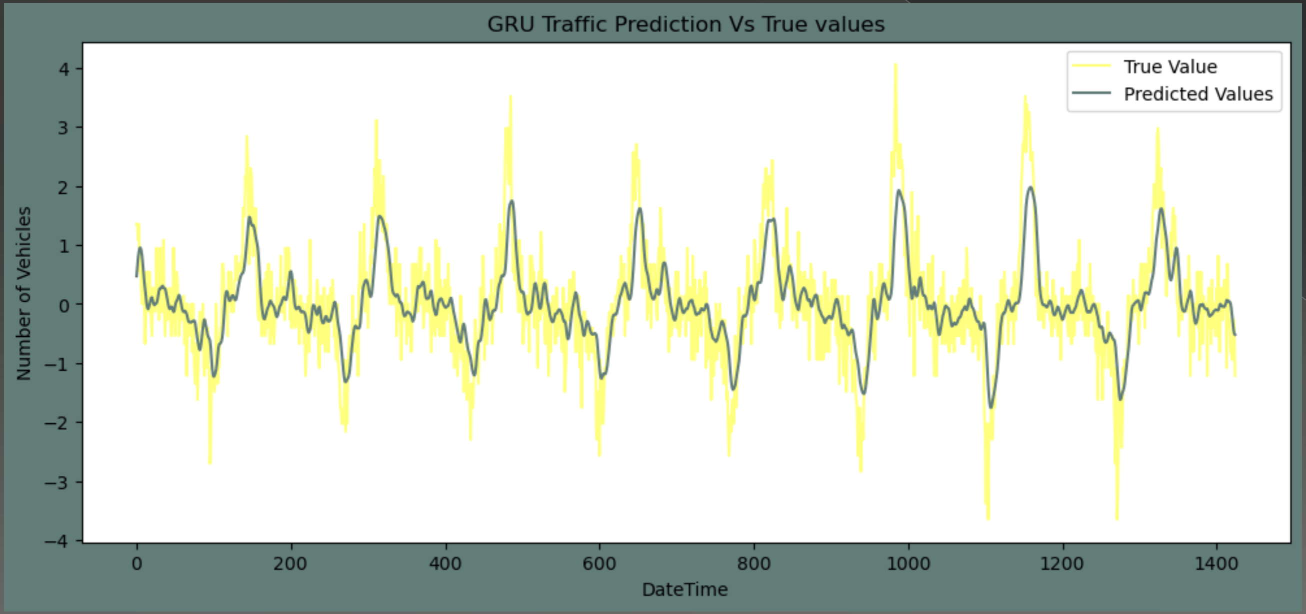
Model Fitting – Junction 2



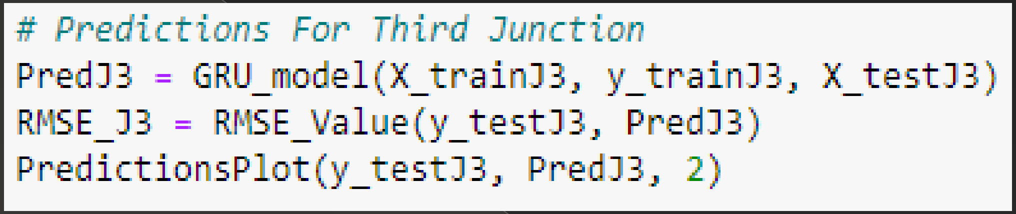
```
# Predictions For Second Junction
PredJ2 = GRU_model(X_trainJ2, y_trainJ2, X_testJ2)
RMSE_J2 = RMSE_Value(y_testJ2, PredJ2)
PredictionsPlot(y_testJ2, PredJ2, 1)
```



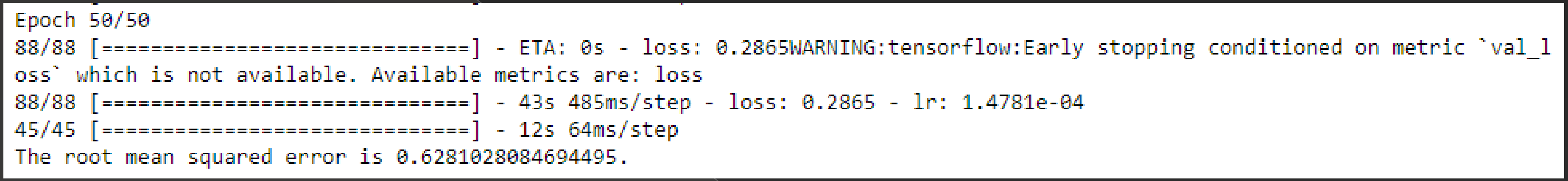
```
Epoch 50/50
88/88 [==============================] - ETA: 0s - loss: 0.1973WARNING:tensorflow:Early stopping conditioned on metric `val_l
oss` which is not available. Available metrics are: loss
88/88 [==============================] - 42s 478ms/step - loss: 0.1973 - lr: 1.4781e-04
45/45 [==============================]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



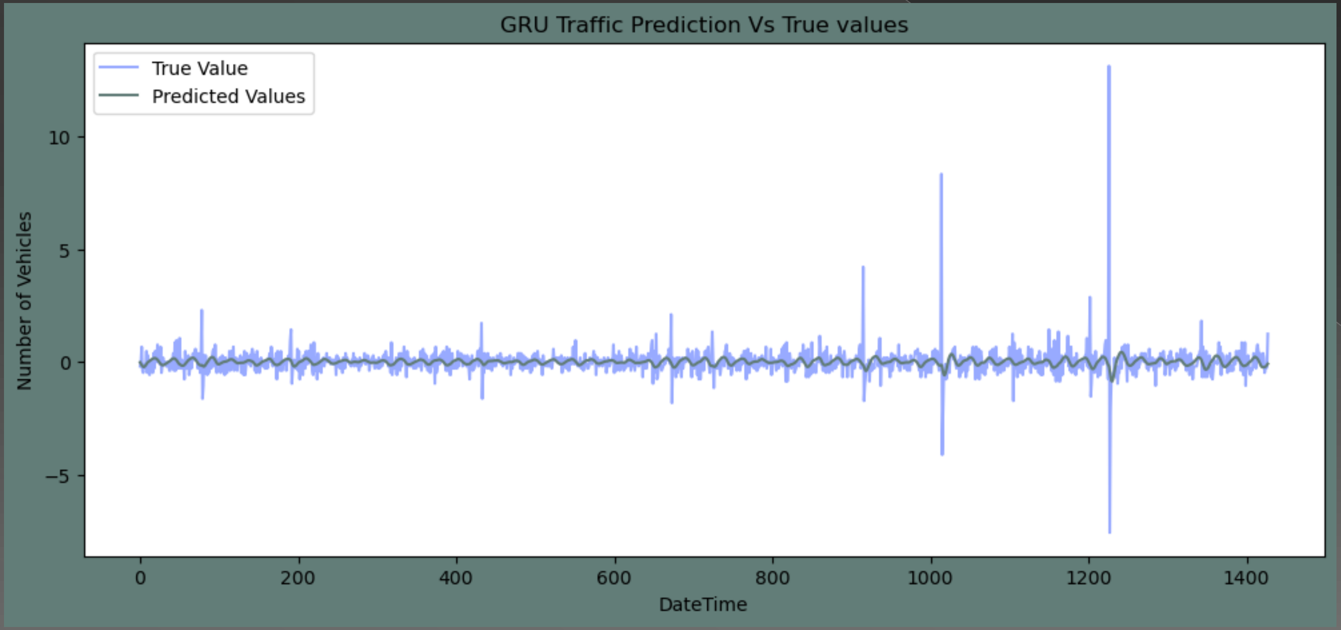
Model Fitting – Junction 3



```
# 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)
```



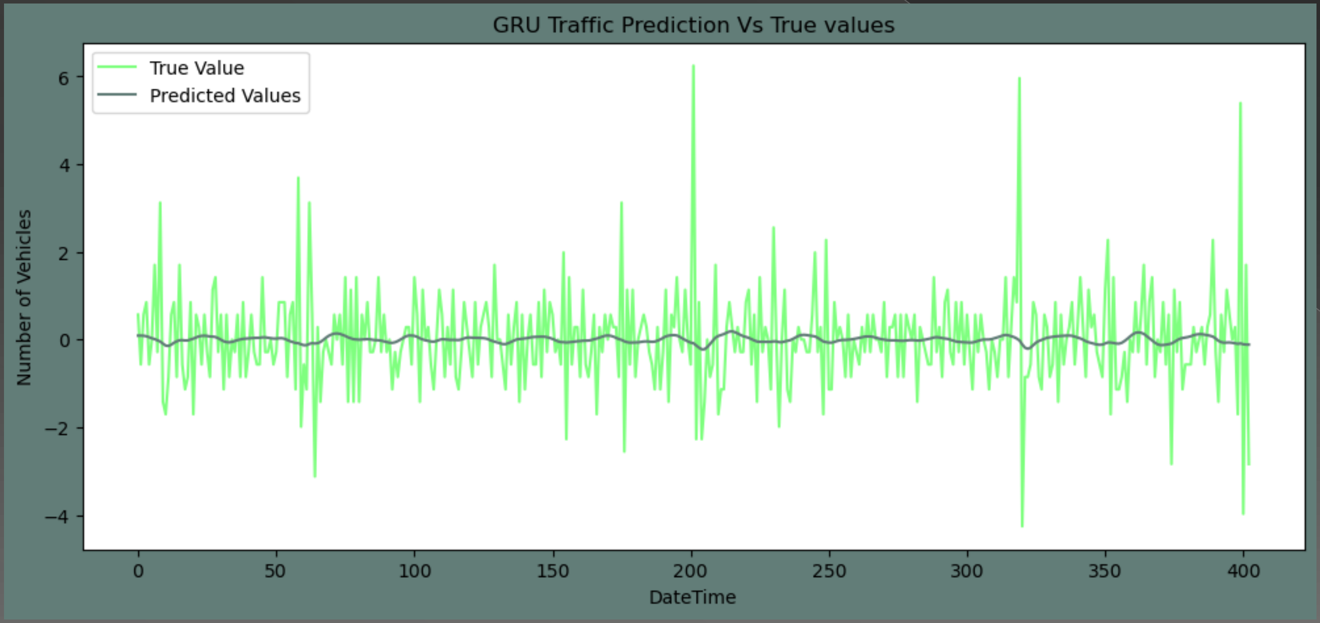
```
Epoch 50/50
88/88 [==============================] - ETA: 0s - loss: 0.2865WARNING:tensorflow:Early stopping conditioned on metric `val_l
oss` which is not available. Available metrics are: loss
88/88 [==============================] - 43s 485ms/step - loss: 0.2865 - lr: 1.4781e-04
45/45 [==============================]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



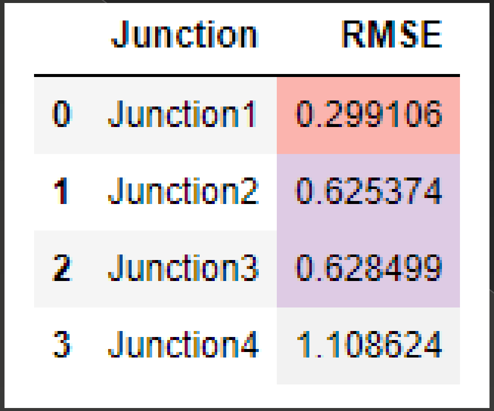
Model Fitting – Junction 4

```
# 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)
```

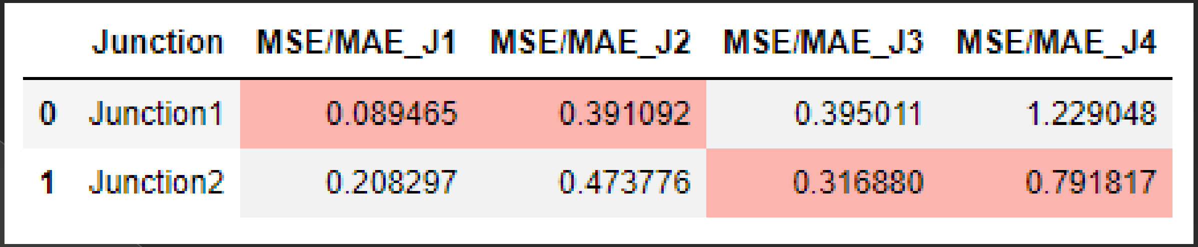
```
Epoch 50/50
26/26 [==============================] - ETA: 0s - loss: 0.6806WARNING:tensorflow:Early stopping conditioned on metric `val_l
oss` which is not available. Available metrics are: loss
26/26 [==============================] - 13s 496ms/step - loss: 0.6806 - lr: 1.4781e-04
13/13 [==============================]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



Model Evaluation - GRU

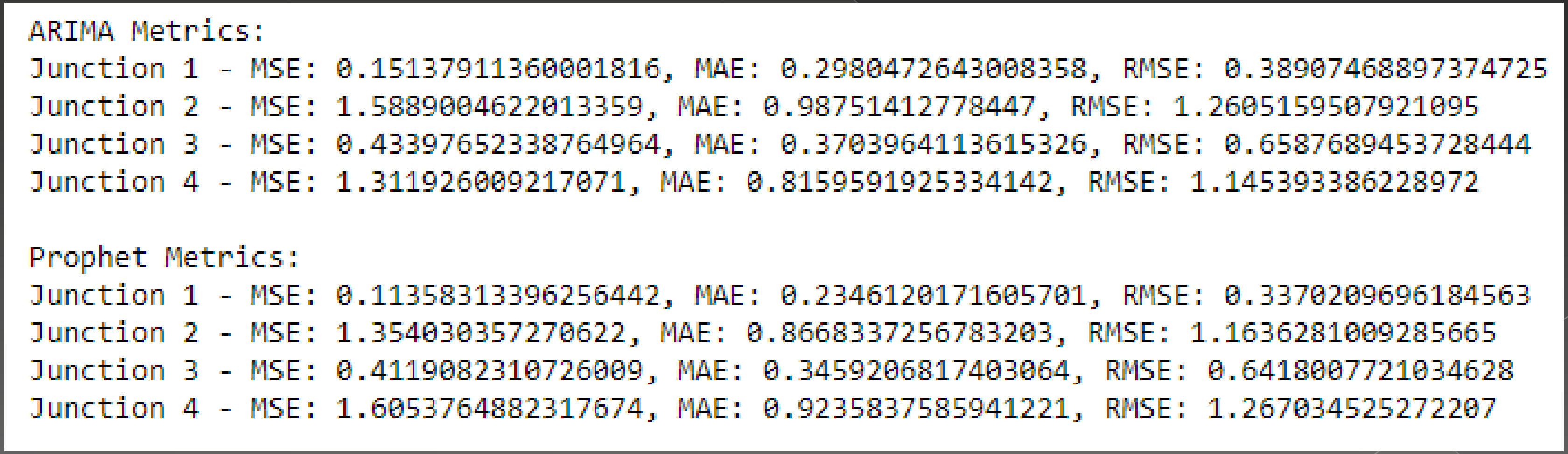


	Junction	RMSE
0	Junction1	0.299106
1	Junction2	0.625374
2	Junction3	0.628499
3	Junction4	1.108624



	Junction	MSE/MAE_J1	MSE/MAE_J2	MSE/MAE_J3	MSE/MAE_J4
0	Junction1	0.089465	0.391092	0.395011	1.229048
1	Junction2	0.208297	0.473776	0.316880	0.791817

Model Evaluation – Comparing with ARIMA and Prophet

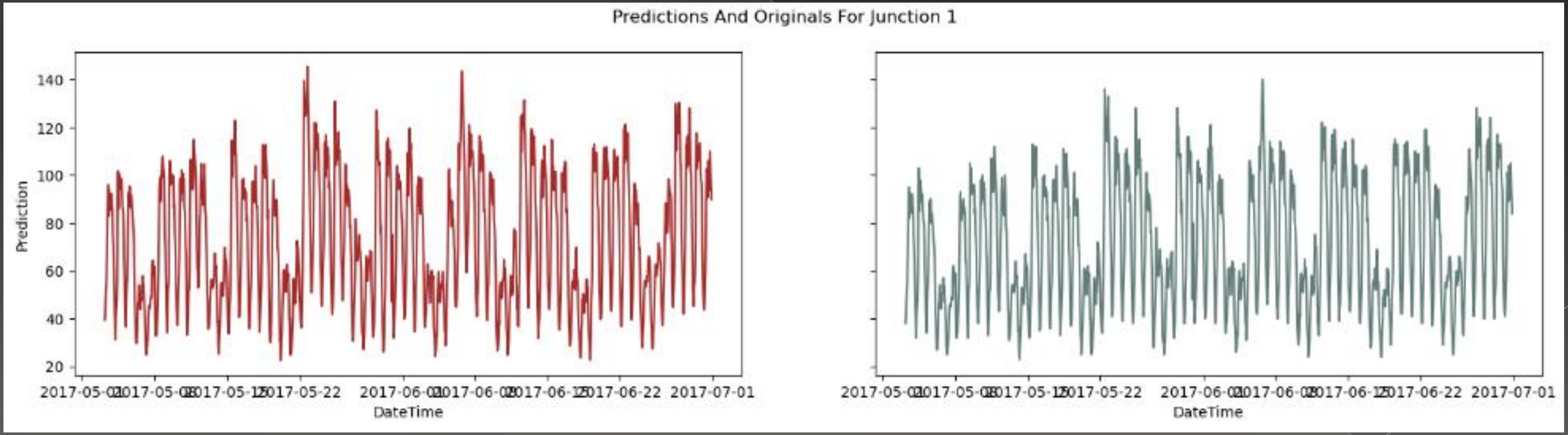


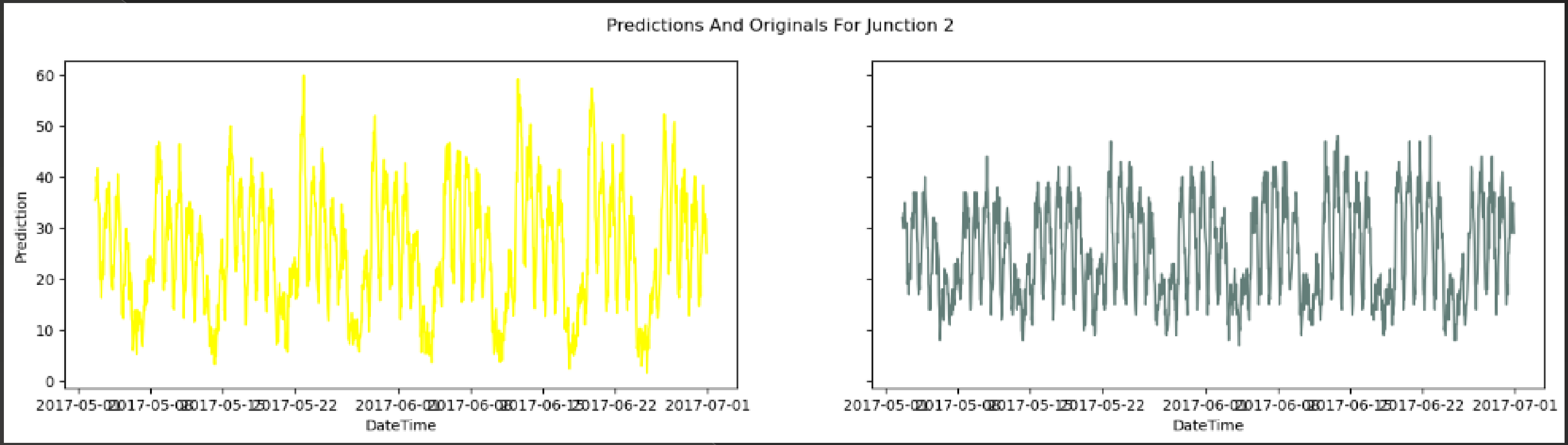
```
ARIMA Metrics:
Junction 1 - MSE: 0.15137911360001816, MAE: 0.2980472643008358, RMSE: 0.38907468897374725
Junction 2 - MSE: 1.5889004622013359, MAE: 0.98751412778447, RMSE: 1.2605159507921095
Junction 3 - MSE: 0.43397652338764964, MAE: 0.3703964113615326, RMSE: 0.6587689453728444
Junction 4 - MSE: 1.311926009217071, MAE: 0.8159591925334142, RMSE: 1.145393386228972

Prophet Metrics:
Junction 1 - MSE: 0.11358313396256442, MAE: 0.2346120171605701, RMSE: 0.3370209696184563
Junction 2 - MSE: 1.354030357270622, MAE: 0.8668337256783203, RMSE: 1.1636281009285665
Junction 3 - MSE: 0.4119082310726009, MAE: 0.3459206817403064, RMSE: 0.6418007721034628
Junction 4 - MSE: 1.6053764882317674, MAE: 0.9235837585941221, RMSE: 1.267034525272207
```

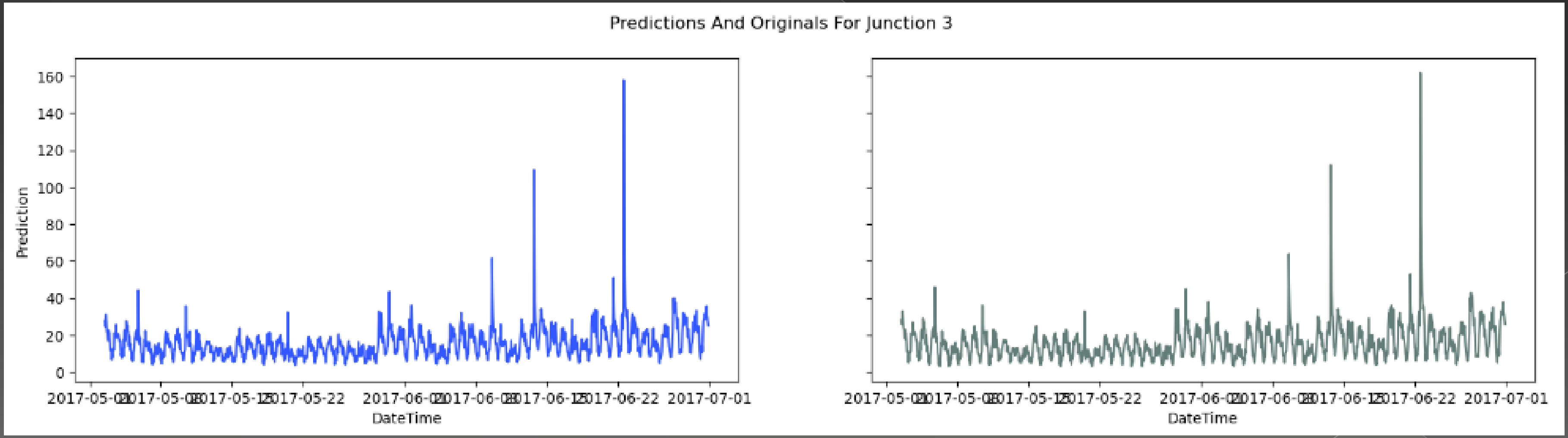
INVERSE TRANSFORMATION OF DATA

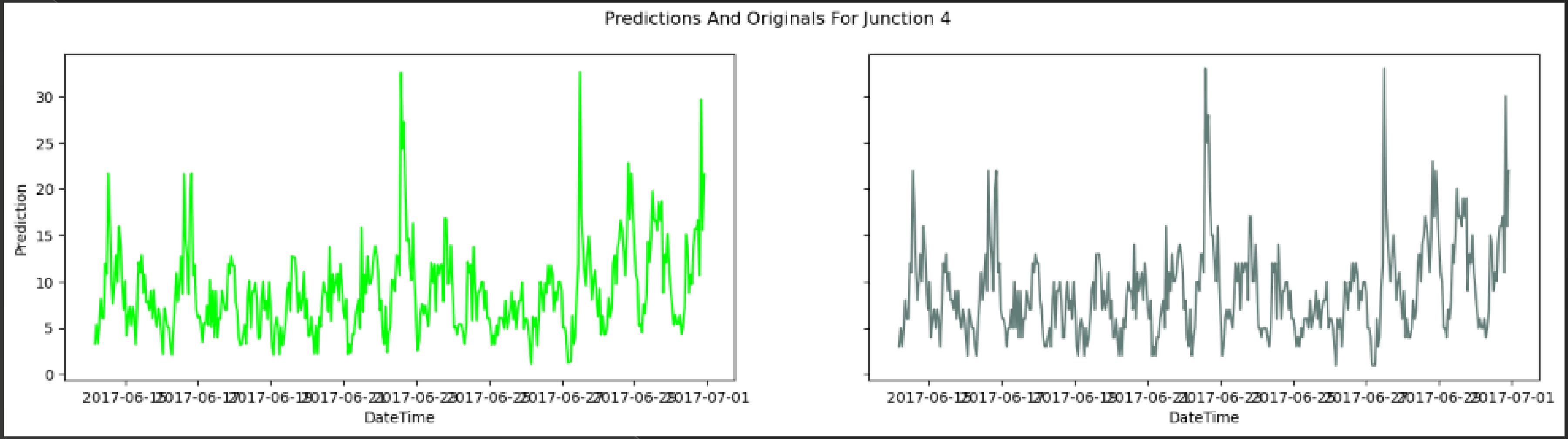
Inversed transforms that I applied to the datasets to remove the seasonality and trends. This step is performed to make the predictions get back on the accurate scale.

The model takes sequences of length 32 and predicts the next time step.




Predictions And Originals For Junction 2
Prediction
DateTime
0
10
20
30
40
50
60
2017-05-01
2017-05-08
2017-05-15
2017-05-22
2017-06-01
2017-06-08
2017-06-15
2017-06-22
2017-07-01


Predictions And Originals For Junction 3
Prediction
DateTime
0
20
40
60
80
100
120
140
160
2017-05-01
2017-05-08
2017-05-15
2017-05-22
2017-06-01
2017-06-08
2017-06-15
2017-06-22
2017-07-01


Predictions And Originals For Junction 4
Prediction
DateTime
DateTime
0
5
10
15
20
25
30
2017-06-15
2017-06-17
2017-06-19
2017-06-21
2017-06-23
2017-06-25
2017-06-27
2017-06-29
2017-07-01

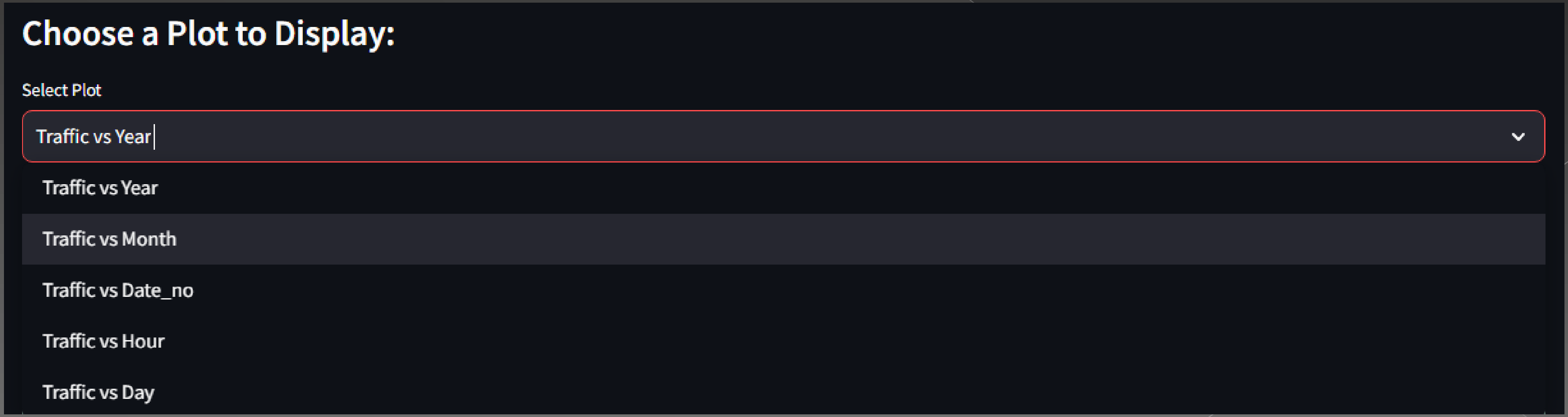
MODEL DEPLOYMENT

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

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

Features:

- Data Visualization of Traffic on Junctions over Years
- Time Series Analysis to understand Trend and Seasonality



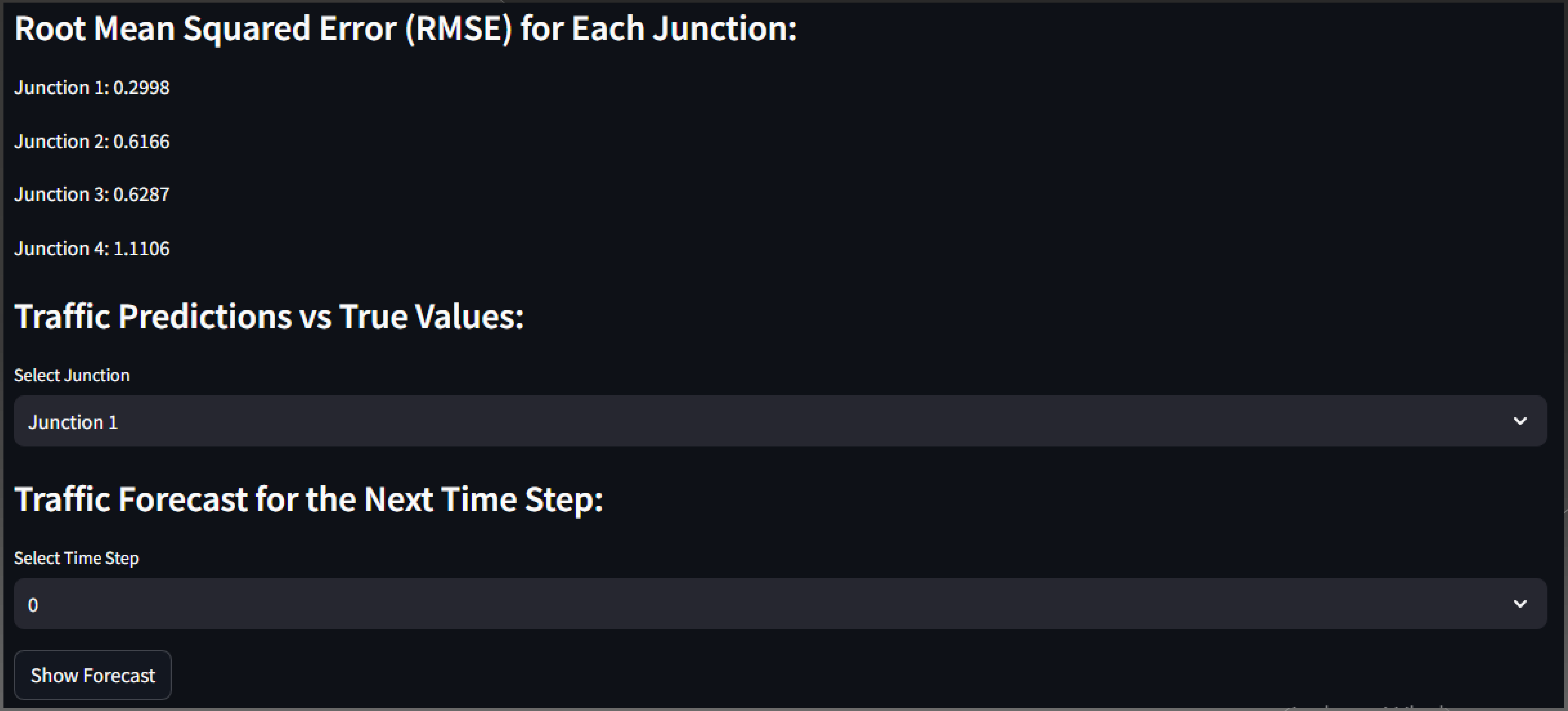
Choose a Plot to Display:

Select Plot

Traffic vs Year

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

- 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



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:

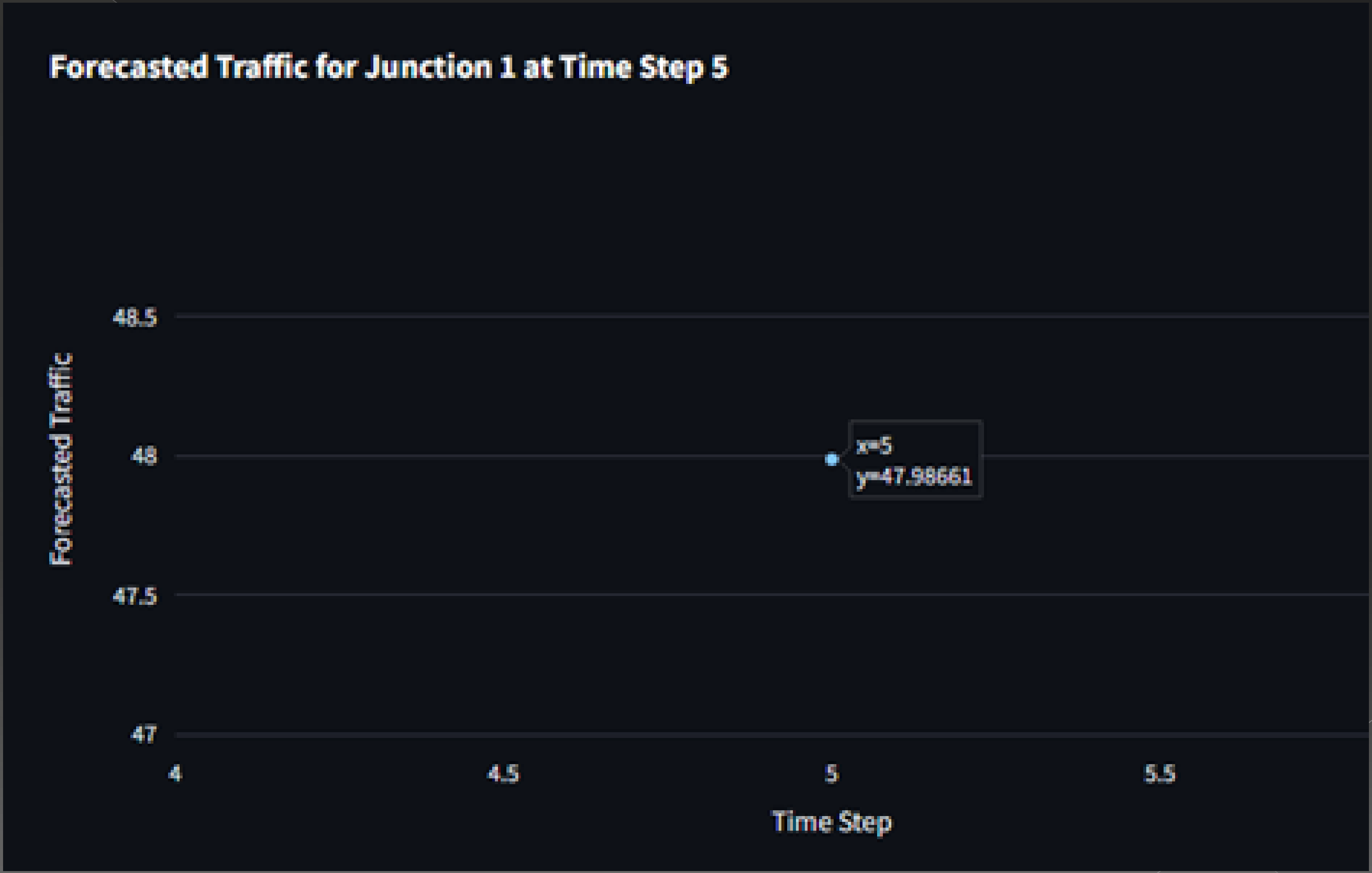
Select Junction

Junction 1

Traffic Forecast for the Next Time Step:

Select Time Step

0


Forecasted Traffic for Junction 1 at Time Step 5
Forecasted Traffic
48.5
48
47.5
47
4
4.5
5
5.5
Time Step
x=5
y=47.98661

SUMMARY

- In this project, I trained a GRU Neural network to predict the traffic on four junctions. I used a 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
- Comparing with ARIMA and Prophet model, GRU model gives a better output performance and results.

THANK YOU