Traffic Analyser - Report

Traffic analysis is the study of the movement of vehicles and pedestrians on roads, highways, and other transportation networks. Traffic data is collected through a variety of methods, including traffic cameras, sensors embedded in the road, and manual counts. This data is used by traffic engineers and planners to optimise traffic flow, improve safety, and reduce congestion.

One of the primary motivations for a traffic analysis machine learning project is to develop predictive models that can accurately forecast traffic conditions based on historical data. These models can be used to optimise traffic signal timing, route planning, and other traffic management strategies. They can also be used to inform transportation policy decisions, such as the placement of new roadways or the implementation of tolls or congestion charges.

Overall, a traffic analysis machine learning project has the potential to make significant contributions to transportation planning and management, by providing more accurate and timely data-driven insights into traffic patterns and behaviours.

About the Tool:

This tool takes traffic data and analyses the dataset and gives us the pattern of traffic conditions throughout the timeline, and predicts on how the traffic flow is gonna be.

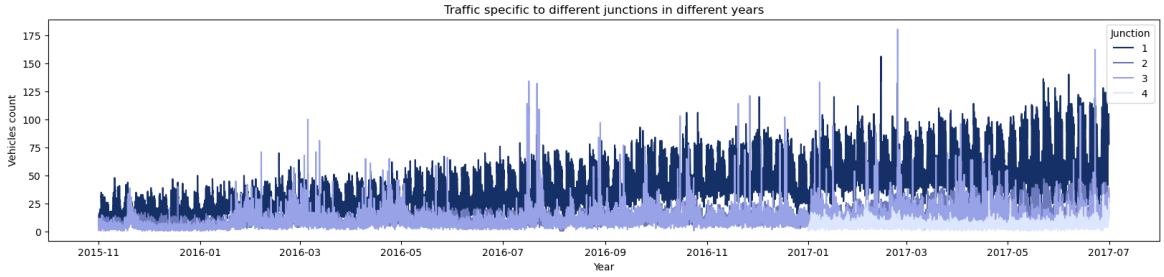
▼ About the Dataset:

This dataset is a collection of numbers of vehicles at four junctions at an hourly frequency.
The csv file provides four features:

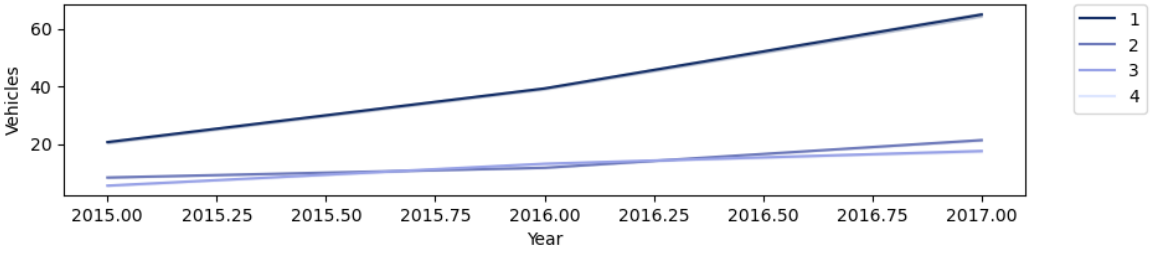
- Date & Time
- Junctions
- Vehicles
- ID

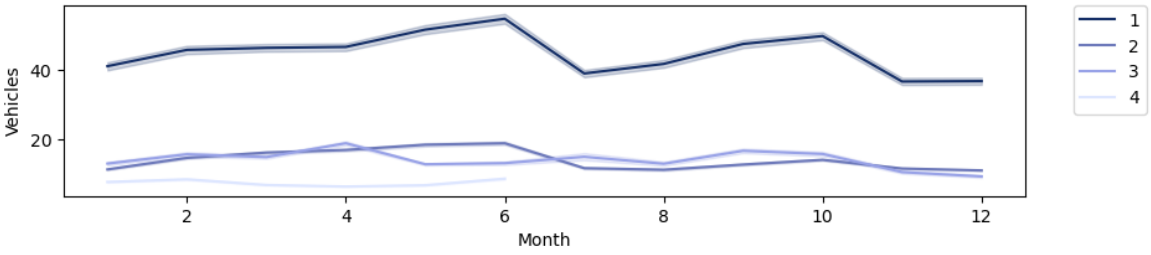
Analysing the Data:

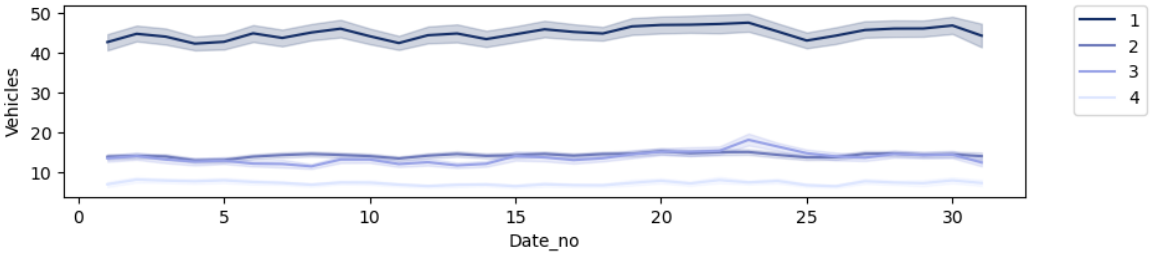
Before proceeding to the traffic condition, we tried to understand the dataset we had. Here's a few analysis we did from the data.

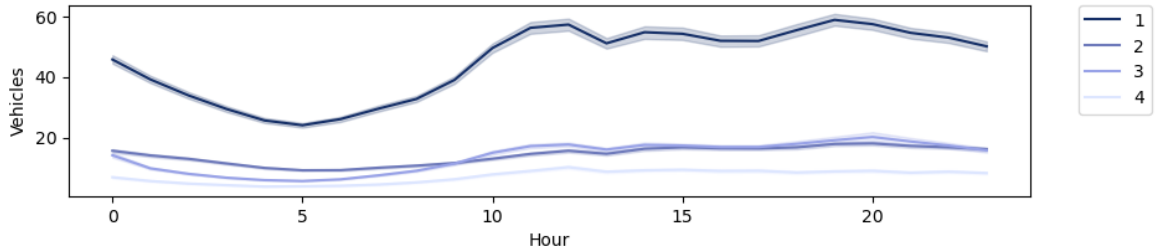


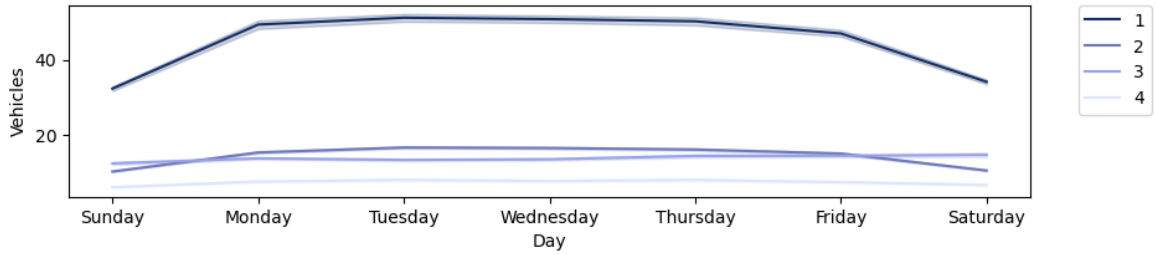
EDA with the data:



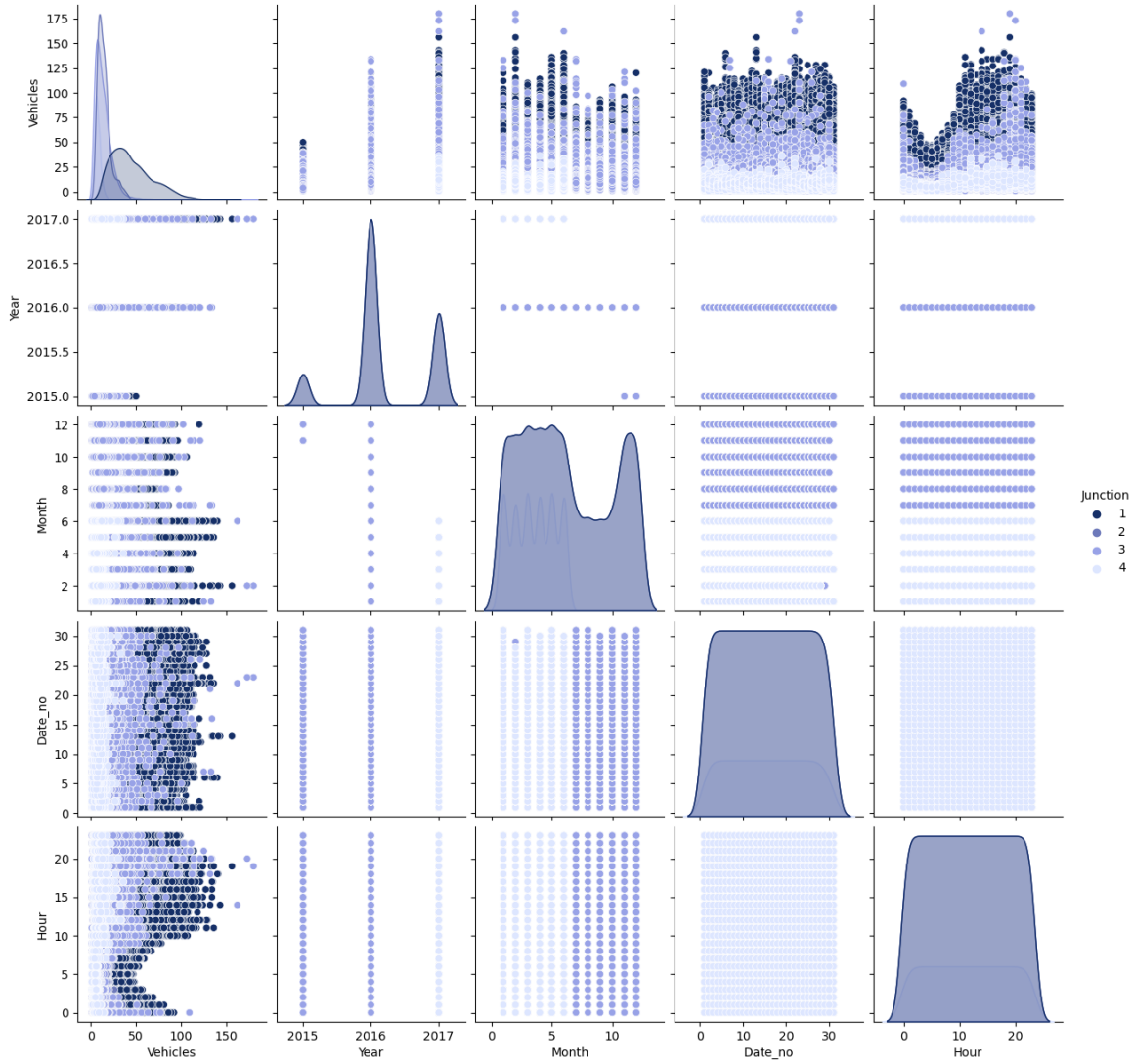








Pair Plot - Overall Representation:



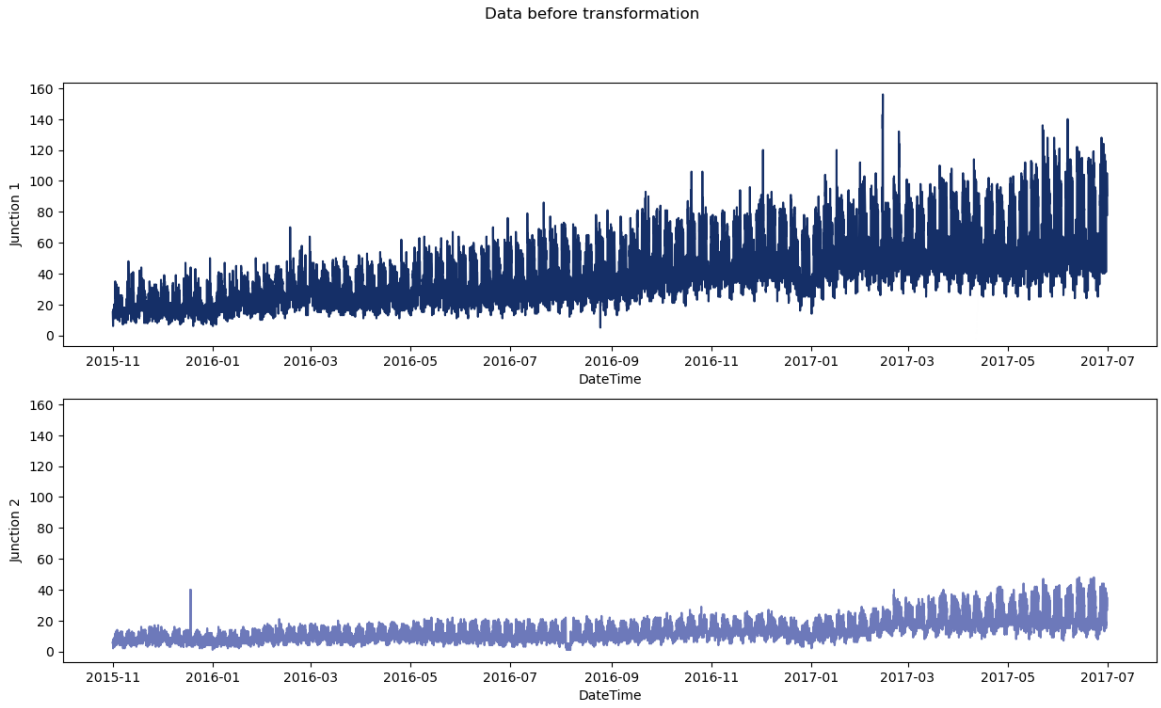
From the EDA, we can conclude few points:

- 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 (trend of varying).
- Junction number one has a more strong weekly seasonality in comparison to the other junctions.

  For the above-postulated reasons, we can think that junctions must be transformed as per their individual needs.

▼ Data Pre-Processing:

Here, we perform transformation of the data and normalisation to make sure the data will be in the format we need and will be useful for us.



Now as the data is stationary, we can start training for our model.

▼ Model & Outputs:

We have used a model called GRU Model, which is Gated Recurrent Unit model to train our data and get the required prediction.

Gated Recurrent Unit:

GRU or Gated Recurrent Unit, is a type of neural network that is used for processing sequences of data, such as text, speech, or music. The GRU has a special feature called "gates" that help it to selectively remember or forget information from previous time steps.

The GRU is commonly used in natural language processing and speech recognition tasks, where long-term dependencies need to be modeled. Overall, the GRU is a powerful tool for processing sequences of data, and its gates allow it to selectively remember or forget information.

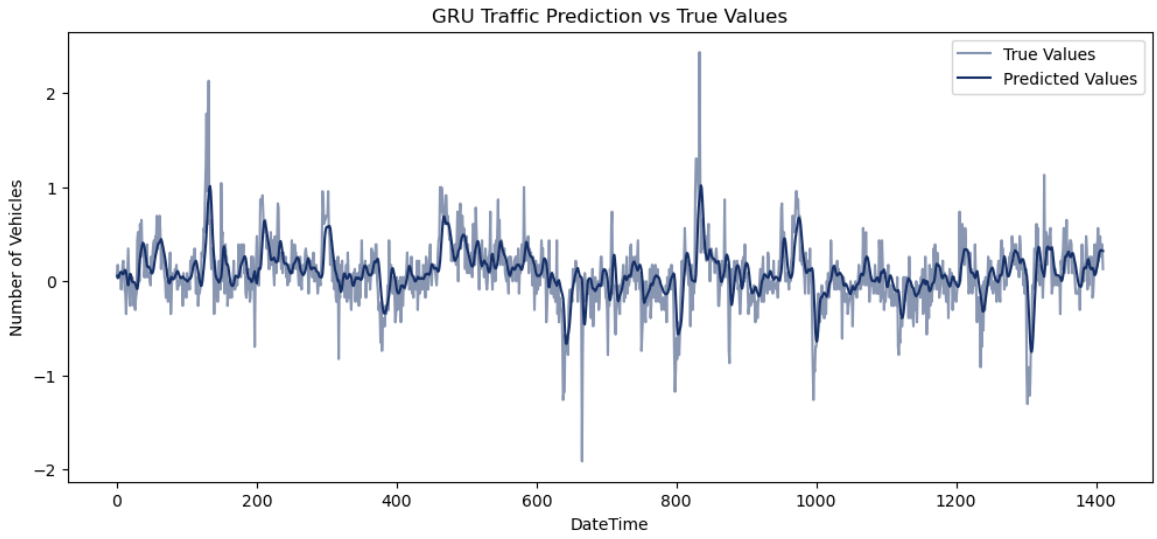
Hence, we decided to use this model and many have used this for different analysis which has given very good results.

Small glimpse of the code:

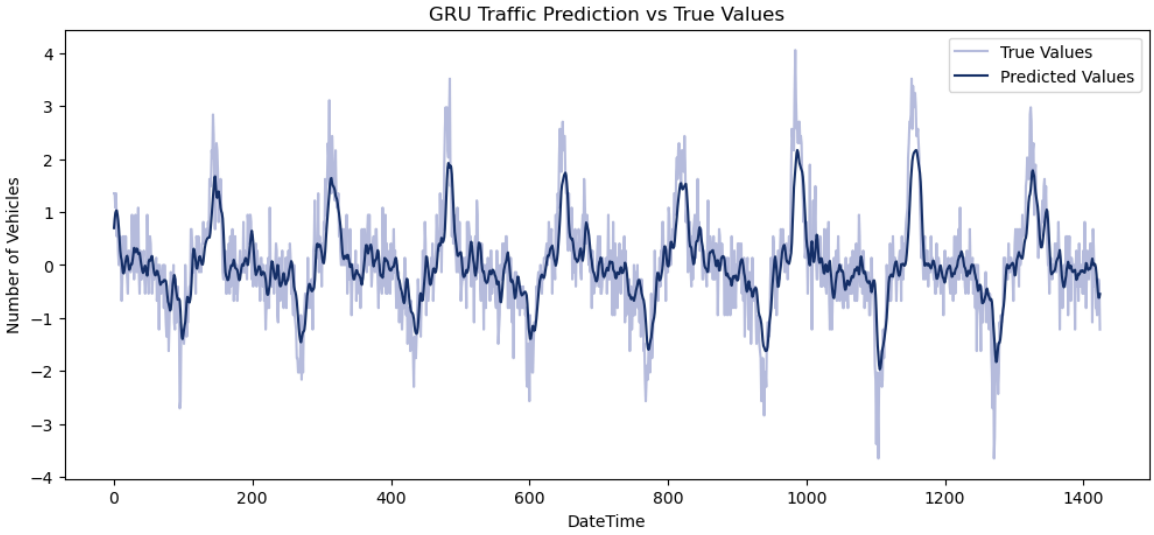
```
def GRU_model(X_Train, y_Train, X_Test):
    early_stopping = callbacks.EarlyStopping(min_delta=0.001,patience=10, restore_best_weights=True)

    model = Sequential()
    model.add(GRU(units=150, return_sequences=True, input_shape=(X_Train.shape[1],1), activation='tanh'))
    model.add(Dropout(0.2))
    model.add(GRU(units=150, return_sequences=True, input_shape=(X_Train.shape[1],1), activation='tanh'))
    model.add(Dropout(0.2))
```

```
    model.add(GRU(units=50, return_sequences=True, input_shape=(X_Train.shape[1],1), activation='tanh'))
    model.add(Dropout(0.2))
    model.add(GRU(units=50, return_sequences=True, input_shape=(X_Train.shape[1],1), activation='tanh'))
    model.add(Dropout(0.2))
    model.add(GRU(units=50, input_shape=(X_Train.shape[1],1), activation='tanh'))
    model.add(Dropout(0.2))
    model.add(Dense(units=1))
    model.compile(optimizer=SGD(decay=1e-7, momentum=0.9),loss='mean_squared_error')
    model.fit(X_Train,y_Train, epochs=10, batch_size=150,callbacks=[early_stopping])
    pred_GRU= model.predict(X_Test)
    return pred_GRU

def RMSE_Value(test,predicted):
    rmse = math.sqrt(mean_squared_error(test, predicted))
    print("RMSE Error {}.".format(rmse))
    return rmse

def PredictionsPlot(test,predicted,m):
    plt.figure(figsize=(12,5))
    plt.plot(test, color=colors[m],label="True Values",alpha=0.5 )
    plt.plot(predicted, color="#152F67",label="Predicted Values")
    plt.title("GRU Traffic Prediction vs True Values")
    plt.xlabel("DateTime")
    plt.ylabel("Number of Vehicles")
    plt.legend()
    plt.show()
```

For Junction-1:



For Junction-2:



Hence, by this way we can use the GRU Model to visualise the traffic data from the given dataset and predict the traffic flow at a certain area throughout the time line.

▼ Conclusion:

Hence, with the help of this tool, one can use it to understand how the traffic flow has been happening in a given area. Based on this, the government can take respective actions to ensure the flow rate in the traffic.

Future Aspects:

- We can extend this tool as an application which can used to feed the traffic data and give the prediction.
- We can also link it with OSM or GMaps API and give out good results about the traffic in a specific area.
- We can also try to develop the model more efficient towards the new data which can be implied as a live mode.