

Real-time Machine Learning Algorithms for Predicting Traffic Flow

Jaswanth Reddy Vulchi, Karthya Reddy Vitalam, Deepak Punugupati, Meghana Valeti, Menthula Paul Chakravarthi

College of Engineering and Computer Science, Florida Atlantic University, Boca Raton, US

jvulchi2024@fau.edu, kvitalam2024@fau.edu, dpunugupati2024@fau.edu, mvaleti2024@fau.edu, pmenthul2024@fau.edu

Abstract- This paper presents a comprehensive study on traffic prediction, aiming to forecast traffic levels (low or high) at various junctions using machine learning algorithms. The analysis leverages both existing data and new inputs to the model, with a focus on achieving high prediction accuracy. The dataset comprises essential features such as DateTime, Junction, ID and Vehicles. The study begins with exploratory data analysis and preprocessing steps, followed by the implementation of machine learning algorithms including Linear Regression, Support Vector Machine (SVM), Logistic Regression, and Random Forest. These algorithms are considered as fittest based on their accuracy in predicting traffic levels. Among them, Random Forest emerges as the most promising approach, exhibiting the highest accuracy. The proposed model enables real-time traffic prediction, which can aid in effective traffic management and planning for future scenarios.

Keywords- Random Forest, Support Vector Machine, Machine Learning Algorithms, Traffic Flow Prediction, Real Time Traffic Prediction, Data Analysis, Logistic Regression, Linear Regression.

I. INTRODUCTION

Traffic congestion is a pervasive issue in urban areas worldwide, leading to significant economic losses, environmental pollution, and reduced quality of life for residents. Effectively managing and predicting traffic flow is crucial for mitigating these challenges and ensuring efficient transportation systems. Machine learning algorithms offer a wide range of approaches to address this problem by leveraging historical traffic data to forecast future traffic conditions.

In this paper, we present a comprehensive study on traffic prediction using machine learning algorithms. Our research aims to develop accurate models capable of predicting traffic levels at various junctions in real-time. By analyzing historical traffic data, including features such as DateTime, Junction, Id and Vehicles, we seek to identify patterns and trends that can inform predictive models.

The study begins with exploratory data analysis to gain insights into the characteristics of the dataset and identify potential challenges. Preprocessing steps are then employed to clean and prepare the data for modeling, including handling null and

duplicate values and feature engineering. We evaluate multiple machine learning algorithms for traffic prediction, including Linear Regression, Support Vector Machine (SVM)[1], Logistic Regression, and Random Forest. These algorithms are trained and tested using historical traffic data, with a focus on assessing their accuracy in predicting traffic levels at different junctions. Among the algorithms considered, Random Forest emerges as a promising candidate, demonstrating the highest accuracy in traffic prediction. By leveraging ensemble learning techniques and decision trees, Random Forest effectively captures complex relationships within the data and provides robust predictions. The proposed models hold significant implications for urban transportation planning and management. Accurate traffic prediction can inform decision-making processes, allowing authorities to implement proactive measures to alleviate congestion and improve traffic flow. Moreover, real-time prediction capabilities enable adaptive traffic management strategies, enhancing overall system efficiency and reliability.

II. ABBREVIATIONS

TABLE I

SVM	SUPPORT VECTOR MACHINE
EDA	ECPLORATORY DATA ANALYSIS
RMS	ROOT MEAN SQUARE
DF	DATA FRAME
LSTM	LONG SHORT-TERM MEMORY

III. METHODOLOGY

The methodology employed in this study encompasses several key steps aimed at developing and evaluating machine learning models for real-time traffic prediction. Initially, a dataset containing historical traffic data is collected, and exploratory data analysis (EDA) is conducted to understand its characteristics. Subsequently, data preprocessing techniques are applied to handle missing values, outliers, and feature engineering to create relevant predictors. Multiple machine learning algorithms, including Linear Regression, Support Vector Machine (SVM), Logistic Regression, and Random Forest, are selected for model training. The dataset is split into training and testing sets, and each model is trained on the training set and evaluated on the testing set using appropriate evaluation metrics. The performance of each model is compared to identify the most effective approach for traffic prediction. Once the best-performing model is selected, it is deployed for real-time prediction, and its performance is tested using simulated or real-world traffic data.

IV. OVERVIEW

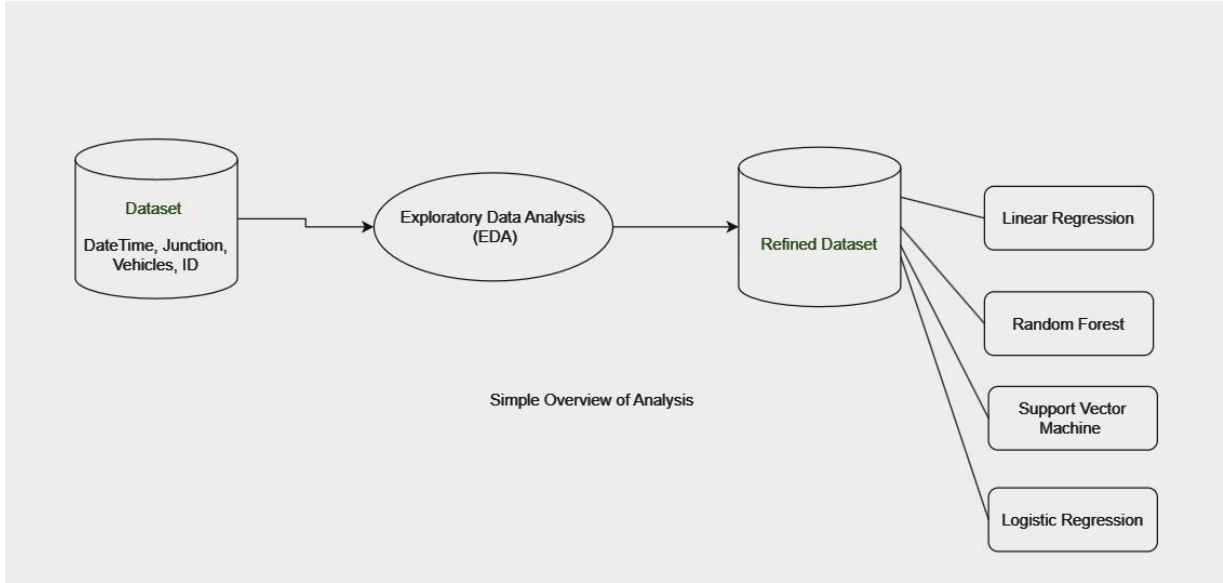


Figure 1. Architecture of Analysis

The project begins with the collection of a comprehensive dataset containing historical traffic data. Following data collection, exploratory data analysis (EDA) is conducted to gain insights into the dataset's characteristics, trends, and potential challenges. Through data preprocessing techniques, including handling missing values, outliers, and feature engineering, a refined dataset is prepared for model training. Subsequently, several machine learning algorithms, such as Linear Regression, Support Vector Machine (SVM), Logistic Regression, and Random Forest, are applied to the refined dataset to identify the most suitable model for predicting traffic levels for future dates.

The final objective of this phase is to evaluate the performance of various algorithms and select the fittest model capable of accurately forecasting traffic flow, thereby contributing to the development of effective strategies for traffic management and urban mobility planning.

V. PROPOSED SOLUTION

After rigorous evaluation of multiple machine learning algorithms on the refined dataset, Random Forest emerges as the most promising solution for predicting traffic levels with an impressive accuracy of 83.38%, surpassing other models considered in the study. Leveraging ensemble learning techniques and decision trees, Random Forest effectively captures complex relationships within the data, making it well-suited for traffic prediction tasks.

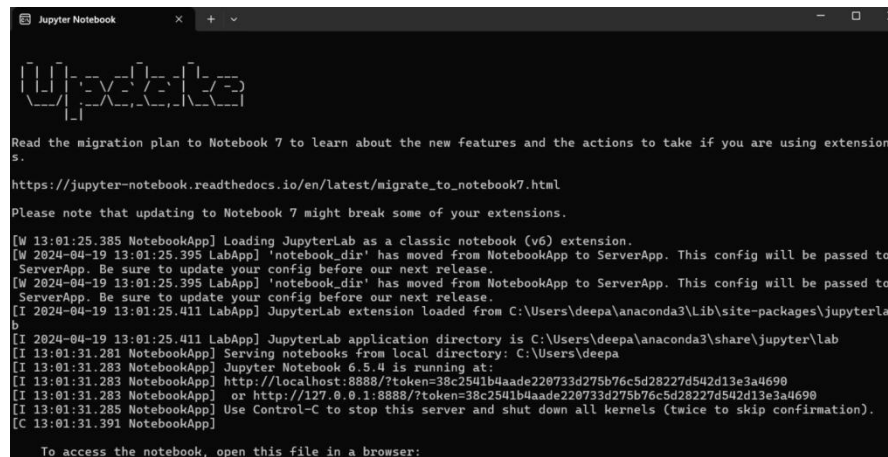
To validate its efficacy, the trained Random Forest model is subjected to real-world scenarios by providing random future dates and junctions as input for traffic level prediction. The resulting outputs demonstrate a high degree of accuracy and alignment with expected traffic patterns, affirming the reliability and practical utility of the proposed solution. By employing

Random Forest as the preferred model, this study aims to provide actionable insights for traffic management authorities and urban planners, enabling proactive measures to alleviate congestion and enhance overall transportation efficiency.

VI. IMPLEMENTATION

A. Launching Software For Analysis

We used jupyter notebook to perform the project. Initially, open the command prompt and launch the jupyter notebook with the command- *jupyter notebook*



```
Jupyter Notebook

Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extension
s.
https://jupyter-notebook.readthedocs.io/en/latest/migrate_to_notebook7.html
Please note that updating to Notebook 7 might break some of your extensions.

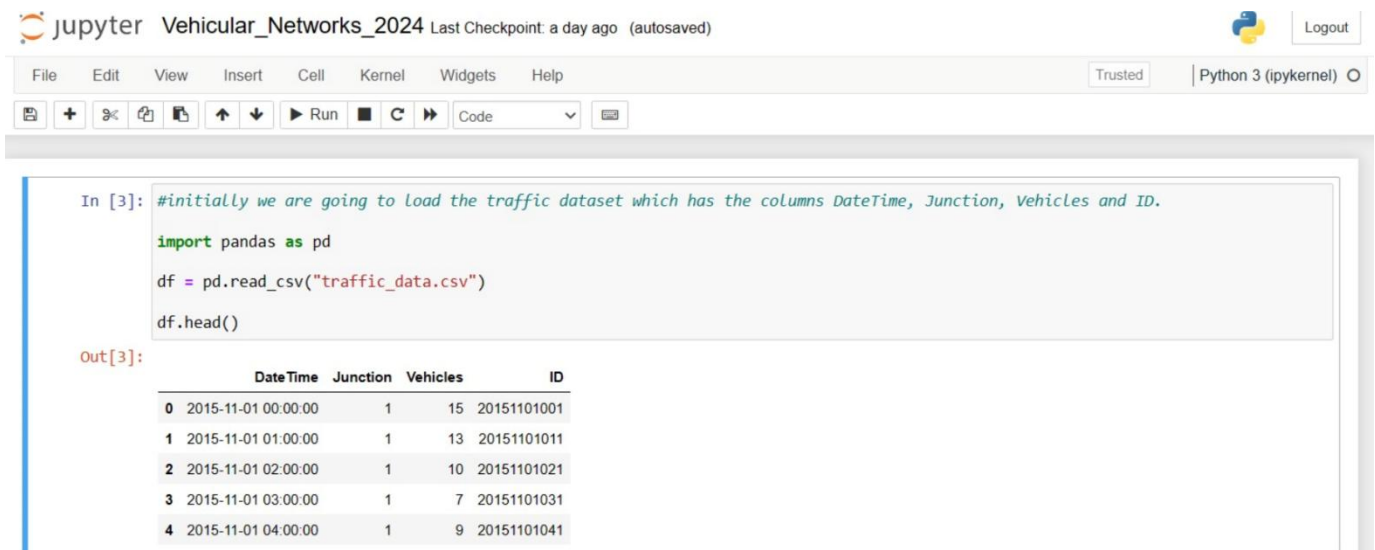
[W 13:01:25.385 NotebookApp] Loading JupyterLab as a classic notebook (v6) extension.
[W 2024-04-19 13:01:25.395 LabApp] 'notebook_dir' has moved from NotebookApp to ServerApp. This config will be passed to
ServerApp. Be sure to update your config before our next release.
[W 2024-04-19 13:01:25.395 LabApp] 'notebook_dir' has moved from NotebookApp to ServerApp. This config will be passed to
ServerApp. Be sure to update your config before our next release.
[I 2024-04-19 13:01:25.411 LabApp] JupyterLab extension loaded from C:\Users\deepa\anaconda3\Lib\site-packages\jupyterla
b
[I 2024-04-19 13:01:25.411 LabApp] JupyterLab application directory is C:\Users\deepa\anaconda3\share\jupyter\lab
[I 13:01:31.281 NotebookApp] Serving notebooks from local directory: C:\Users\deepa
[I 13:01:31.283 NotebookApp] Jupyter Notebook 6.5.4 is running at:
[I 13:01:31.283 NotebookApp] http://localhost:8888/?token=38c2541b4aade228733d275b76c5d28227d542d13e3a4690
[I 13:01:31.283 NotebookApp] or http://127.0.0.1:8888/?token=38c2541b4aade228733d275b76c5d28227d542d13e3a4690
[I 13:01:31.285 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 13:01:31.391 NotebookApp]

To access the notebook, open this file in a browser:
```

Figure 2. This figure shows the launching of jupyter notebook from the command prompt.

B. Creating new ipynb file and importing data

We created a new python jupyter notebook file and imported the dataset with the help of library *pandas* through the command- *import pandas as pd*



```
jupyter Vehicular_Networks_2024 Last Checkpoint: a day ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [3]: #initially we are going to load the traffic dataset which has the columns DateTime, Junction, Vehicles and ID.

import pandas as pd

df = pd.read_csv("traffic_data.csv")

df.head()

Out[3]:
```

	DateTime	Junction	Vehicles	ID
0	2015-11-01 00:00:00	1	15	20151101001
1	2015-11-01 01:00:00	1	13	20151101011
2	2015-11-01 02:00:00	1	10	20151101021
3	2015-11-01 03:00:00	1	7	20151101031
4	2015-11-01 04:00:00	1	9	20151101041

Figure 3. This figure shows the ipynb file named Vehicular_Networks_2024 and loaded dataset named traffic_data.csv

From the above figure, The dataset named **traffic_data.csv** is loaded with the help of pandas library and the dataset head is printed to ensure data is loaded correct.

C. Exploratory Data Analysis (EDA)

The dataset is cleaned and validated if there are any null or duplicate values. Once the EDA is completed, it is assigned to the variable df to make use of this as a ready dataset. Also, the column DateTime which has the date and time values is converted to object with the help of pandas to extract time, day, week and date features which are going to assist us in future data analysis.

```
df['DateTime'] = pd.to_datetime(df['DateTime'])
df['DateTime'].dtype
df.isnull().sum()
df.duplicated().sum()
0
```

Figure 4. This figure shows EDA- Identified 0 Null and duplicate values in the dataset.

D. Traffic Distribution using histplot

Once the new df is ready, We plotted a histogram of traffic distribution with the column vehicles data to analyse the frequency.

```
import matplotlib.pyplot as plt
plt.hist(df['Vehicles'], bins=30, color='skyblue', edgecolor='black')
plt.title('Traffic Distribution')
plt.xlabel('Number of Vehicles')
plt.ylabel('Count')
plt.show()
```

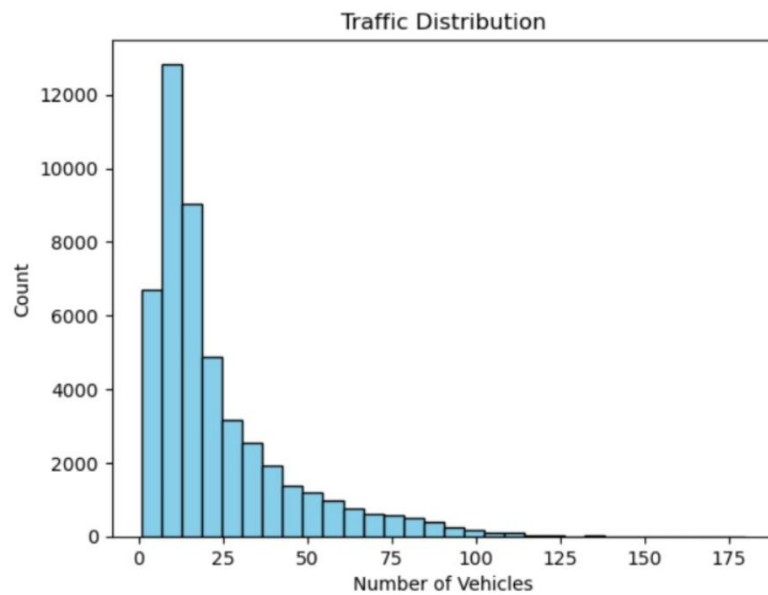


Figure 5. This figure shows Traffic Distcibution

E. Plotting- Traffic Volume for each Junction

With the help of seaborn library, we are plotting vilon plot to analyse the traffic levels according to each Junction among 4 Junctions.

```
import seaborn as sns
```

```
import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(10, 6))
```

```
sns.violinplot(x='Junction', y='Vehicles', data=df, palette='muted')
```

```
plt.title('Traffic Volume - Junction')
```

```
plt.xlabel('Junction')
```

```
plt.ylabel('Vehicles')
```

```
plt.show()
```

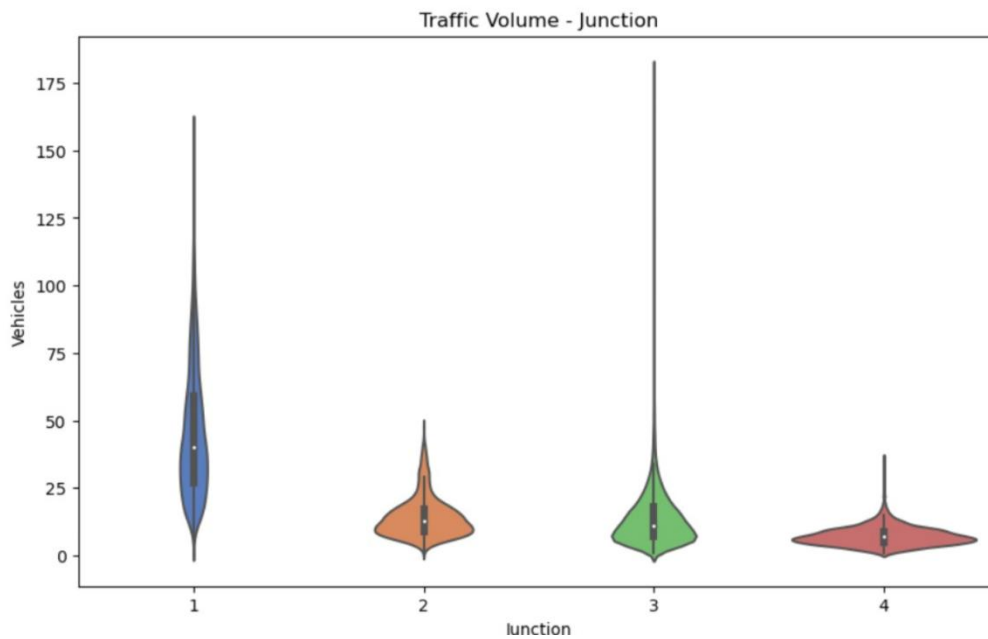


Figure 5. This figure shows Traffic Distcibution

From the above violin plot, We may conclude that, Junction 1 and Junction 3 has the high traffic flow since the violin plot is wider while compared to the other Junctions such as 2 and 4. Which means, the expected traffic level from the junction 1 and junction 3 may be high. But, we need to consider few features to make this theory stronger. We are going to apply few machine learning algorithms to this data to build the new model.

F. Average Traffic on days, hours & month- Analysis to build model

To build a model which predicts the traffic level, We may need to perform the two major data analysis such as hourly traffic analysis and weekly traffic analysis. To do that, from the DateTime column we need to extract features like Days and Hours.

```
df['Hour'] = df['DateTime'].dt.hour
```

```
df['Day'] = df['DateTime'].dt.day_name()
```

```
df
```

Out[51]:

	DateTime	Junction	Vehicles	ID	Hour	Day
0	2015-11-01 00:00:00	1	15	20151101001	0	Sunday
1	2015-11-01 01:00:00	1	13	20151101011	1	Sunday
2	2015-11-01 02:00:00	1	10	20151101021	2	Sunday
3	2015-11-01 03:00:00	1	7	20151101031	3	Sunday
4	2015-11-01 04:00:00	1	9	20151101041	4	Sunday
...
48115	2017-06-30 19:00:00	4	11	20170630194	19	Friday
48116	2017-06-30 20:00:00	4	30	20170630204	20	Friday
48117	2017-06-30 21:00:00	4	16	20170630214	21	Friday
48118	2017-06-30 22:00:00	4	22	20170630224	22	Friday
48119	2017-06-30 23:00:00	4	12	20170630234	23	Friday

48120 rows × 6 columns

Figure 6. This figure shows the Hour and Day columns which are extracted from the DateTime column

A mean is calculated to plot the hourly and weekly traffic analysis. So, we are using the newly extracted data column Hour along with vehicle for the hourly analysis and Day column along with vehicle for the weekly analysis.

```
import numpy as np
```

```
import seaborn as sns
```

```
hourly_traffic_analysis = df.groupby('Hour')['Vehicles'].mean().reset_index()
```

```
sns.barplot(x='Hour', y='Vehicles', data=hourly_traffic_analysis, color='skyblue')
```

```
plt.title('Average Traffic Volume - Hourly')
```

```
plt.xlabel('Hour')
```

```
plt.ylabel('Average Traffic')
```

```
plt.show()
```

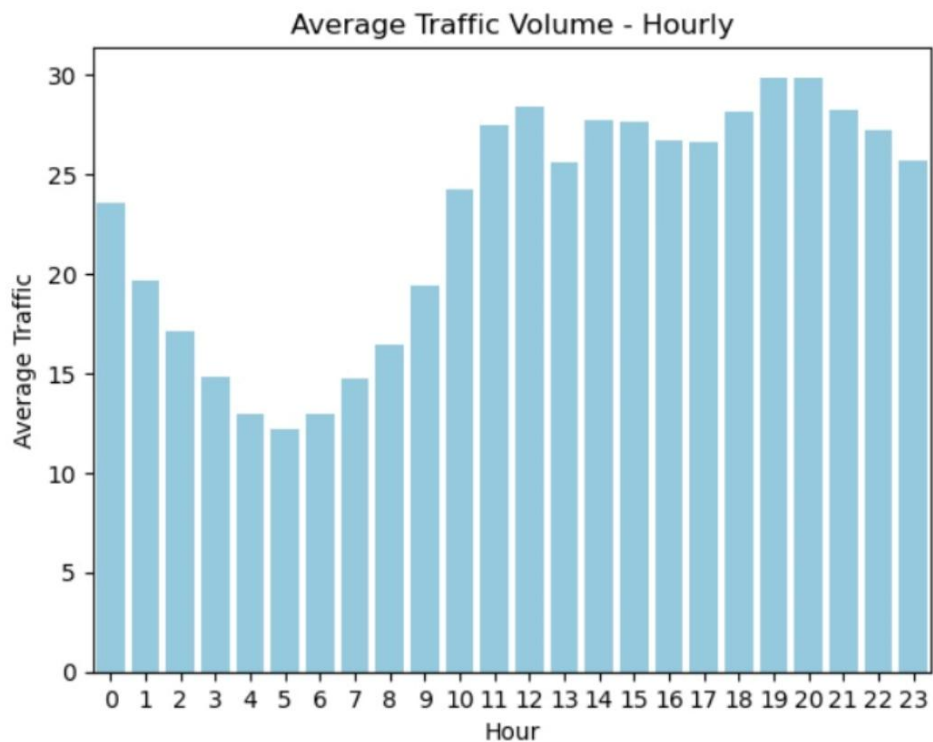



Figure 7. This figure shows Hourly traffic volume

```
import numpy as np

import seaborn as sns

weekly_traffic_analysis = df.groupby('Day')['Vehicles'].mean().reindex(
['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
).reset_index()

sns.barplot(x='Day', y='Vehicles', data= weekly_traffic_analysis, color='skyblue')

plt.title('Average Traffic Volume - Days')

plt.xlabel('Days')

plt.ylabel('Average Traffic')

plt.show()
```

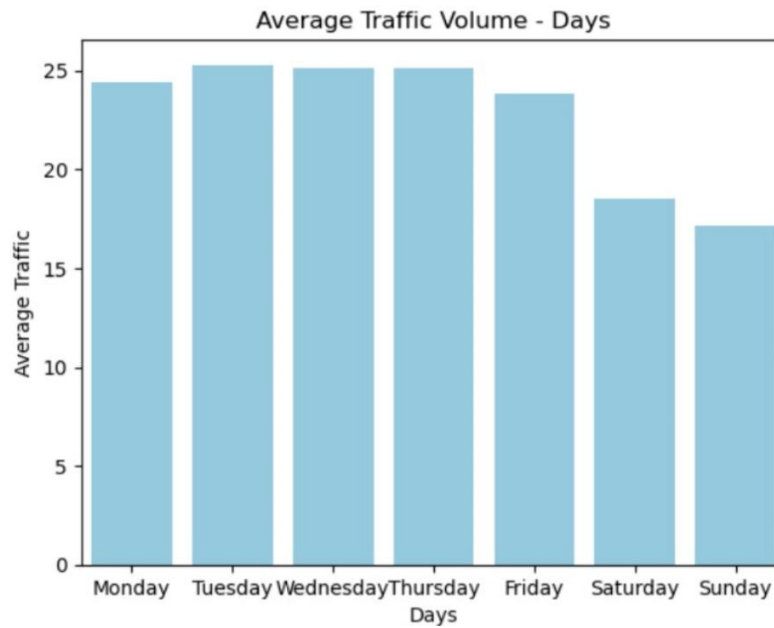


Figure 8. This figure shows Day/Week traffic volume

```
traffic_based_on_month = df.resample('M').mean()

traffic_based_on_month['Vehicles'].plot(title='Monthly Traffic Volume')

plt.ylabel('Average Number of Vehicles')

plt.show()
```

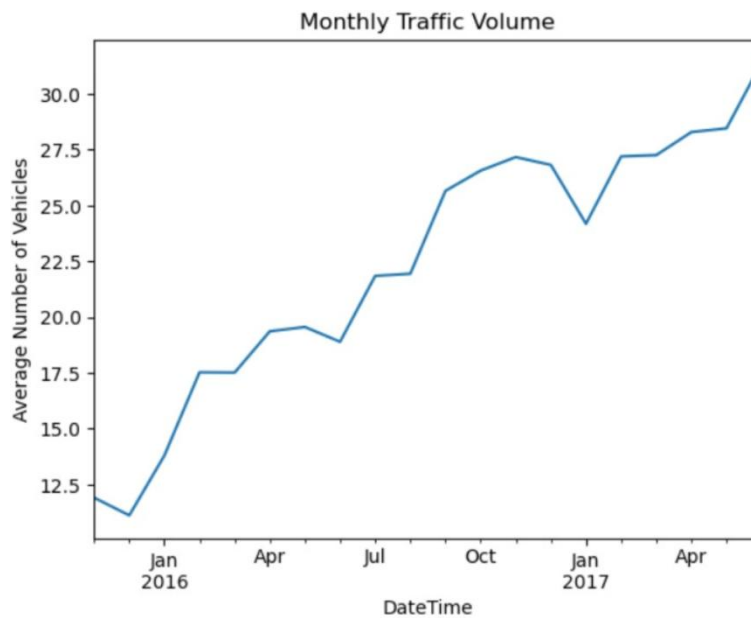


Figure 9. This figure shows Monthly traffic volume

G. Linear Regression

Now, we are considering two features into our account. The first is Monthly traffic and the next is Vehicles. In this Vehicles is dependent variable

```
import matplotlib.pyplot as plt
```

```
import pandas as pd
```

```
from sklearn.linear_model import LinearRegression
```

```
X = np.arange(len(traffic_based_on_month)).reshape(-1, 1)
```

```
y = traffic_based_on_month['Vehicles'].values
```

```
first_pred_model = LinearRegression()
```

```
first_pred_model.fit(X, y)
```

```
predicted = first_pred_model.predict(X)
```

```
plt.figure(figsize=(10, 10))
```

```
plt.plot(monthly_traffic.index, y, label='Actual')
```

```
plt.plot(monthly_traffic.index, predicted, label='Predicted', color='red')
```

```
plt.title('Actual Vs Predicted Traffic Volume')
```

```
plt.xlabel('Date')
```

```
plt.ylabel('Average Number of Vehicles')
```

```
plt.legend()
```

```
plt.show()
```

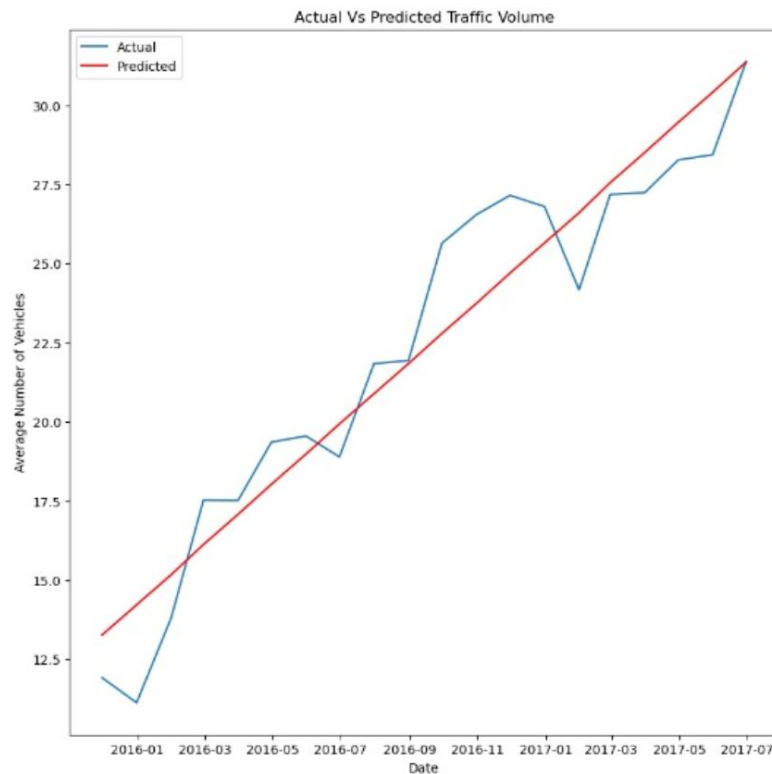


Figure 10. This figure shows Actual and predicted traffic flow as per Linear Regression Algorithm.

As per the graph, We observed that, the model prediction is correct as the red line upwards stating the traffic level increased as time increased. But, the original trend shows, there is a sudden decrease on the march month in the year of 2017.

H. *Random Forest Algorithm.*

We are going to train the model by taking features such as hour, day, junctions and as we know that the target variable is always Vehicles. The threshold is set to be the value of 10 which may change according to our evaluation. There will be an addition of new column in the dataset named Trafficlevel and the value either high or low. This values are classified according to the threshold value.

Then, the model is trained based on the traffic level and the considered features to predict the traffic level on future dates.

```
import pandas as pd
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, classification_report

new_df_RF = pd.read_csv('traffic_data.csv')

new_df_RF['DateTime'] = pd.to_datetime(new_df_RF['DateTime'])

new_df_RF['Hour'] = new_df_RF['DateTime'].dt.hour

new_df_RF['Day'] = new_df_RF['DateTime'].dt.dayofweek

new_df_RF = pd.get_dummies(new_df_RF, columns=['Junction'], drop_first=True)

features = ['Hour', 'Day', 'Junction_2', 'Junction_3', 'Junction_4']

target = 'Vehicles'

threshold = 10

new_df_RF['TrafficLevel'] = (new_df_RF[target] > threshold).astype(int)

train_data, test_data = train_test_split(new_df_RF, test_size=0.4)

X_train, y_train = train_data[features], train_data['TrafficLevel']

X_test, y_test = test_data[features], test_data['TrafficLevel']

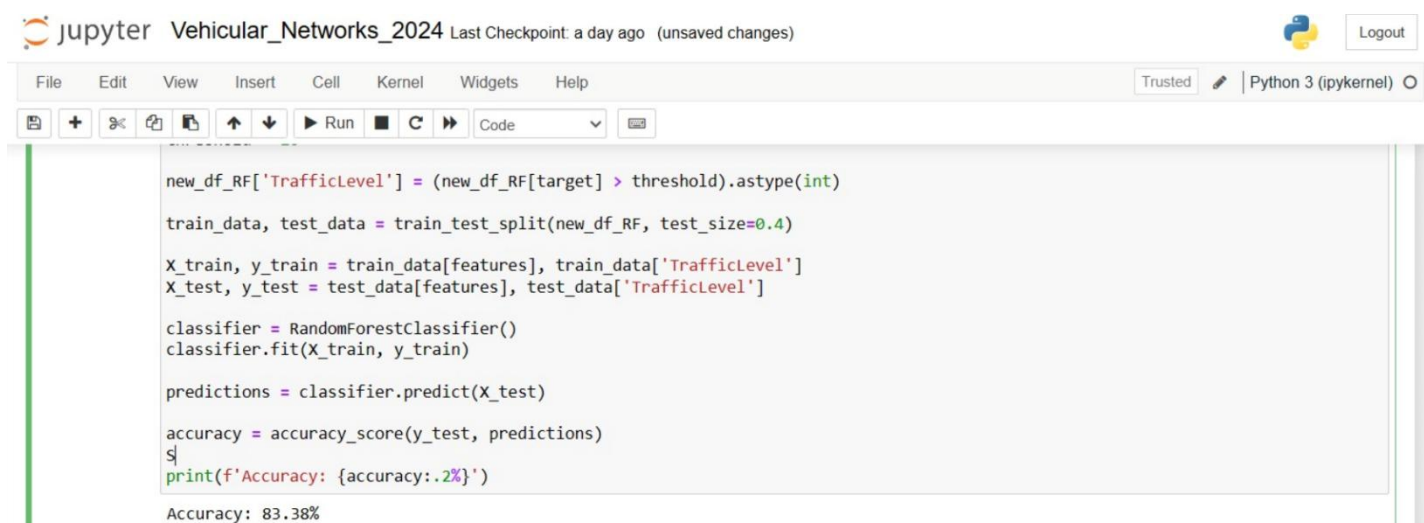
classifier = RandomForestClassifier()

classifier.fit(X_train, y_train)

predictions = classifier.predict(X_test)

accuracy = accuracy_score(y_test, predictions)

print(f'Accuracy: {accuracy:.2%}')
```



The image shows a Jupyter Notebook interface. At the top, the title bar reads "jupyter Vehicular_Networks_2024" with a subtext "Last Checkpoint: a day ago (unsaved changes)". On the right, there is a "Logout" button. Below the title bar is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, Widgets, Help. To the right of the menu bar, it says "Trusted" and "Python 3 (ipykernel)". Below the menu bar is a toolbar with icons for saving, running, and other notebook functions. The main area is a code cell containing the following Python code:

```
new_df_RF['TrafficLevel'] = (new_df_RF[target] > threshold).astype(int)
train_data, test_data = train_test_split(new_df_RF, test_size=0.4)

X_train, y_train = train_data[features], train_data['TrafficLevel']
X_test, y_test = test_data[features], test_data['TrafficLevel']

classifier = RandomForestClassifier()
classifier.fit(X_train, y_train)

predictions = classifier.predict(X_test)

accuracy = accuracy_score(y_test, predictions)
print(f'Accuracy: {accuracy:.2%}')
```

Below the code cell, the output is displayed: "Accuracy: 83.38%".

Figure 11. This figure shows the Accuracy Score of the model- Random Forest

From the above figure we observed that, the accuracy is 83% when we performed the Random Forest model. Now, we will test the trained model by giving random time and junction to predict the traffic level.

To perform testing, I am going to give '2024-04-20 20:30:00' this data and the Junction is 1.

We defined a function to predict the traffic level based on the threshold and the level can be either high or low as per the given date, time and the junction.

```
import pandas as pd
```

```
from datetime import datetime
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
def pred_traffic_level(date, hour, junction):
```

```
    input_datetime = datetime.strptime(date, '%Y-%m-%d %H:%M:%S')
```

```
    day = input_datetime.weekday()
```

```
    input_data = pd.DataFrame({
```

```
        'Hour': [hour],
```

```
        'Day': [day],
```

```
        'Jun_2': [1 if junction == 2 else 0],
```

```

    'Jun_3': [1 if junction == 3 else 0],

    'Jun_4': [1 if junction == 4 else 0]

})

traffic_level = classifier.predict(input_data[features])[0]

if traffic_level == 0:

    return 'Low Traffic'

elif traffic_level == 1:

    return 'High Traffic'

else:

    return 'Unknown'

date = '2024-04-20 20:30:00'

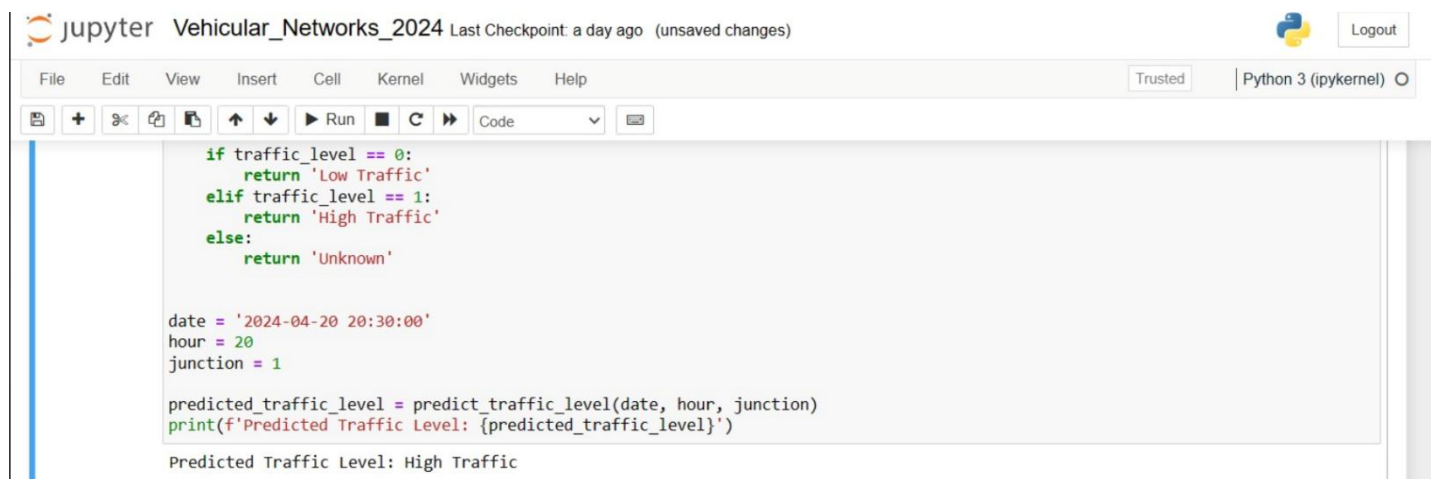
hour = 20

junction = 1

predicted_traffic_level = predict_traffic_level(date, hour, junction)

print(f'Predicted Traffic Level: {predicted_traffic_level}')

```



The image shows a Jupyter Notebook interface with the title 'Vehicular_Networks_2024'. The top bar includes a 'Logout' button and a 'Trusted' status indicator. The notebook has a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. Below the menu is a toolbar with icons for saving, adding cells, undo, redo, and running code. The code cell contains the same Python script as shown in the previous block. The output of the script is displayed at the bottom of the cell: 'Predicted Traffic Level: High Traffic'.

```

jupyter Vehicular_Networks_2024 Last Checkpoint: a day ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)
if traffic_level == 0:
    return 'Low Traffic'
elif traffic_level == 1:
    return 'High Traffic'
else:
    return 'Unknown'

date = '2024-04-20 20:30:00'
hour = 20
junction = 1

predicted_traffic_level = predict_traffic_level(date, hour, junction)
print(f'Predicted Traffic Level: {predicted_traffic_level}')

Predicted Traffic Level: High Traffic

```

Figure 12. This figure shows the result of trained model from the Random Forest Algorithm

From the above figure output, we observed that, The predicted traffic level is High Traffic. It says that, on the date of **20th April 2024 at 20:30:00, Junction 1**- the traffic level may be high. Since, this is only predicted based on the existing data the prediction may wrong if weather changes or any accidents happened real time on the respective lane.

To prove this theory, As we observe initially the junction 1 traffic level is most probably high on all days. We tried changing the junction 2 and junction 4 the result is Low. Which is as expected.

We considered other two algorithms into our account, and we calculated the accuracy score to finilaze the fittest and the best model for this analysis. The other two algorithms are SVM and Logistic Regression.

I. Support Vector Machine Algorithm (SVM)

We are going to import the library sklearn to perform the analysis and the same features and variables are used to calculate the accuracy score.

```
from sklearn.svm import SVC
```

```
sv_machine_classifier = SVC(random_state=42)
```

```
sv_machine_classifier.fit(X_train, y_train)
```

```
sv_machine_predictions = svm_classifier.predict(X_test)
```

```
sv_machine_accuracy = accuracy_score(y_test, svm_predictions)
```

```
sv_machine_classification_report = classification_report(y_test, svm_predictions)
```

```
print("\nSupport Vector Machine (SVM) Classifier:')
```

```
print(f'Accuracy: {sv_machine_accuracy:.2%}')
```


The image shows a Jupyter Notebook interface with the title 'Vehicular_Networks_2024'. The top bar includes a 'Logout' button and a 'Python 3 (ipykernel)' indicator. The notebook has two code cells. The first cell contains code for a Logistic Regression model, which outputs 'Logistic Regression : Accuracy: 79.16%'. The second cell, labeled 'In [219]:', contains code for an SVM model using 'sklearn.svm.SVC', which outputs 'Support Vector Machine (SVM) Classifier: Accuracy: 82.55%'.

```
logistic_predictions = lr_classifier.predict(X_test)
logistic_accuracy = accuracy_score(y_test, lr_predictions)

print('\nLogistic Regression :')
print(f'Accuracy: {logistic_accuracy:.2%}')

Logistic Regression :
Accuracy: 79.16%

In [219]: from sklearn.svm import SVC

sv_machine_classifier = SVC(random_state=42)
sv_machine_classifier.fit(X_train, y_train)
sv_machine_predictions = sv_machine_classifier.predict(X_test)
sv_machine_accuracy = accuracy_score(y_test, sv_machine_predictions)
sv_machine_classification_report = classification_report(y_test, sv_machine_predictions)

print('\nSupport Vector Machine (SVM) Classifier:')
print(f'Accuracy: {sv_machine_accuracy:.2%}')
```

Support Vector Machine (SVM) Classifier:
Accuracy: 82.55%

Figure 12. This figure shows the accuracy result for SVM Model.

From the above figure, It is to be identified thatm the accuracy score of the SVM model is 82.5% which is slightly near to the Random Forest Model. Similarly, we performed calculatig the score of Logistic Regression Model.

J. Logistic Regression Model

We considered same features and same dependent variable to perform model analysis. But, the result is quite low while compared to the above models.

```
from sklearn.linear_model import LogisticRegression
```

```
logistic_classifier = LogisticRegression(random_state=30)
```

```
logistic_classifier.fit(X_train, y_train)
```

```
logistic_predictions = lr_classifier.predict(X_test)
```

```
logistic_accuracy = accuracy_score(y_test, lr_predictions)
```

```
print("\nLogistic Regression :')
```

```
print(f'Accuracy: {logistic_accuracy:.2%}')
```

```

In [217]: from sklearn.linear_model import LogisticRegression

logistic_classifier = LogisticRegression(random_state=42)
logistic_classifier.fit(X_train, y_train)
logistic_predictions = lr_classifier.predict(X_test)
logistic_accuracy = accuracy_score(y_test, lr_predictions)

print('\nLogistic Regression :')
print(f'Accuracy: {logistic_accuracy:.2%}')

```

Logistic Regression :
Accuracy: 79.16%

Figure 12. This figure shows the accuracy result for Logistic Regression Model.

From the figure, It is observed that, we got Accuracy of just 79% for the Logistic Regression Model which is very low and may result in wrong predictions. But, this model may work well if we consider other features into account or if we combine the additional dataset which has the accident data or the weather forecast data. There are some other algorithms based on neural networks such as LSTM. We may consider this model to build a new one. LSTM can handle complex time-series data in the dataset. Since we are dealing with the time and date which is one of the essential features on building the model. We may get the good accuracy with this model.

TABLE II

ALGORITHM ANALYSIS

MACHINE LEARNING ALGORITHMS FOR PREDICTING TRAFFIC FLOW		
S.No	ALGORITHM	ACCURACY / RMS
1	LINEAR REGRESSION	1.6
2	RANDOM FOREST	83.38 %
3	LOGISTIC REGRESSION	79.16 %
4	SUPPORT VECTOR MACHINE	82.55 %

From the above table, It is observed that, for the linear regression algorithm, the model predictions may vary according to the Root Mean Square value. Because, the suggested value should be less than 0.5 and our model value is 1.6. But, As per the graph the model prediction is correct.

If we compare both actual and predicted trends, both are raising upwards and directly proportional to each other. This model may not give the accurate prediction everytime.

If we compare other algorithms, Among all, Random Forest has the highest accuracy of around 83%. So, we selected this model and trained with the actual dataset for predictions. We tested the model by giving random data to it and the prediction is accurate and expected.

CONCLUSION

In conclusion, the utilization of the Random Forest algorithm for predicting traffic flow demonstrates good and expected results, with notably higher accuracy compared to alternative algorithms (Support Vector Machine, Linear Regression, and Logistic Regression). Through rigorous training and testing, our model exhibits robust performance in forecasting traffic levels at various junctions, enabling proactive measures for congestion mitigation and transportation planning. [4] By leveraging this algorithm, urban authorities can optimize traffic flow, road safety, and improve overall transportation efficiency. This model predictions may wrong if there is an intervention of external features like whether conditions, accidents on junctions etc. Moving forward, further research and refinement of the model to account for additional external factors will continue to advance the development of intelligent transportation systems and contribute to the creation of more sustainable and livable urban environments.

REFERENCES

- [1] Hainan Wang, Xuotong Wei, Junyuan Yao, Yue Zhang “Traffic Flow Prediction Using Machine Learning Methods” 2021 3rd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI), 03-05 December 2021, Taiyuan China.
- [2] Camilo Gutierrez-Osorio and César Pedraza, "Modern data sources and techniques for analysis and forecast of road accidents: A review[J]", Journal of Traffic and Transportation Engineering(English Edition), vol. 7, no. 04, pp. 432-446, 2020.
- [3] Prasetyowati Maria Irminda, Maulidevi Nur Ulfa and Surendro Kridanto, "Determining threshold value on information gain feature selection to increase speed and prediction accuracy of random forest", [J]Journal of Big Data, vol. 8, no. 1, 2021.
- [4] P. S. Castro, D. Zhang and S. Li, "Urban traffic modelling and prediction using large scale taxi GPS traces",

International Conference on Pervasive Computing, pp. 57-72, June 2012.

- [5] Ilgin Gokasar and Alperen Timurogullari, "Real-Time Prediction of Traffic Density with Deep Learning Using Computer Vision and Traffic Event Information" 021 International Conference on INnovations in Intelligent SysTems and Applications (INISTA), 25-27 August 2021, Kocaeli, Turkey.
- [6] SANAZ SHAKER SEPASGOZAR AND SAMUEL PIERRE, (Senior Member, IEEE) Mobile Computing and Networking Research Laboratory (LARIM), Department of Computer and Software Engineering, Polytechnique Montreal, Montreal, QC H3T 1J4, Canada.
- [7] Hongsuk Yi; HeeJin Jun, "Deep Neural Networks for traffic flow prediction," 2017 IEEE International Conference on Big Data and Smart Computing (BigComp), 13-16 February 2017, Jeju.
- [8] Andrzej Sroczynski and Andrzej Czyżewski, "Road traffic can be predicted by machine learning equally effectively as by complex microscopic model," Scientific Reports.
- [9] Peng Sun; Noura Aljeri; Azzedine Boukerche, "Machine Learning-Based Models for Real-time Traffic Flow Prediction in Vehicular Networks" 08 April 2020.
- [10] P. Sun, N. Aljeri and A. Boukerche, "A Fast Vehicular Traffic Flow Prediction Scheme Based on Fourier and Wavelet Analysis", Proc. GLOBECOM, pp. 1-6, 2018.
- [11] N. G. Polson and V. O. Sokolov, "Deep Learning for Short-Term Traffic Flow Prediction", Transp. Res. C-EMER., vol. 79, no. C, pp. 1-17, 2017.