Industrial Internship Report on

"Smart City Traffic Patterns"

Prepared By:- Shabdprakash Thakkar

Executive Summary

This report provides details of the Industrial Internship provided by upskill Campus and The IoT Academy in collaboration with Industrial Partner UniConverge Technologies Pvt Ltd (UCT).

This internship was focused on a project/problem statement provided by UCT. We had to finish the project including the report in 6 weeks' time.

My project was that we are working with the government to transform various cities into smart cities. The vision is to convert it into a digital and intelligent city to improve the efficiency of services for the citizens. One of the problems faced by the government is traffic. You are a data scientist working to manage the traffic of the city better and to provide input on infrastructure planning for the future.

The government wants to implement a robust traffic system for the city by being prepared for traffic peaks. They want to understand the traffic patterns of the four junctions of the city.

TABLE OF CONTENTS

1 Preface 3

2 Introduction 4

2.1 About UniConverge Technologies Pvt Ltd 4

2.2 About upskill Campus 8

2.3 Objective 9

2.4 Reference 9

3 Problem Statement 10

4 Existing and Proposed solution 11

5 Proposed Design/ Model 14

5.1 High Level Diagram (if applicable) 15

5.2 Low Level Diagram (if applicable) 19,20

5.3 Interfaces (if applicable) 218

6 Performance Test 22

6.1 Test Plan/ Test Cases 24

6.2 Test Procedure 26

6.3 Performance Outcome 28

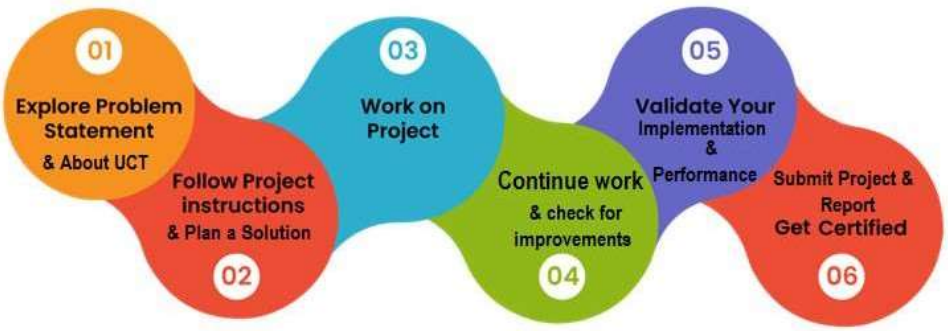
7 My learnings 30

8 Future work scope 31

1. Preface

India's position as the second-largest country in terms of population and economic growth brings forth significant traffic congestion issues in major cities. Unlike more affluent nations, India faces unique challenges due to limited space, high costs, and delayed infrastructure development, leading to a rapid increase in vehicle numbers that outpaces the expansion of road networks. Short-term traffic flow prediction, using real-time data to anticipate conditions for the next 5 to 20 minutes, becomes crucial for aiding travelers in navigating congested areas, optimizing traffic operations, and managing highway networks. This pressing need for effective traffic management systems is particularly acute in rapidly growing cities like Bangalore and Delhi, where population growth and expanding IT industries contribute to traffic congestion. Intelligent Traffic Systems (ITS) are essential solutions for India's urban areas to address these challenges.

Data science involves exploring vast datasets to extract valuable business insights, combining methods from mathematics, statistics, artificial intelligence, and computer engineering. It empowers data scientists to answer questions about past events, understand their causes, predict future occurrences, and derive actionable conclusions. Machine learning, a subset of artificial intelligence, enables computers to learn from data without explicit programming, while data science uses technology and tools to unlock meaning from data. With its capacity to uncover patterns and relationships within massive datasets, data science is a powerful tool not only for traffic prediction but also for diverse applications, such as predicting agricultural productivity.



2. Introduction

2.1 About UniConverge Technologies Pvt Ltd

A company established in 2013 and working in Digital Transformation domain and providing Industrial solutions with prime focus on sustainability and RoI. For developing its products and solutions it is leveraging various **Cutting Edge Technologies e.g. Internet of Things (IoT), Cyber Security, Cloud computing (AWS, Azure), Machine Learning, Communication Technologies (4G/5G/LoRaWAN), Java Full Stack, Python, Front end** etc.

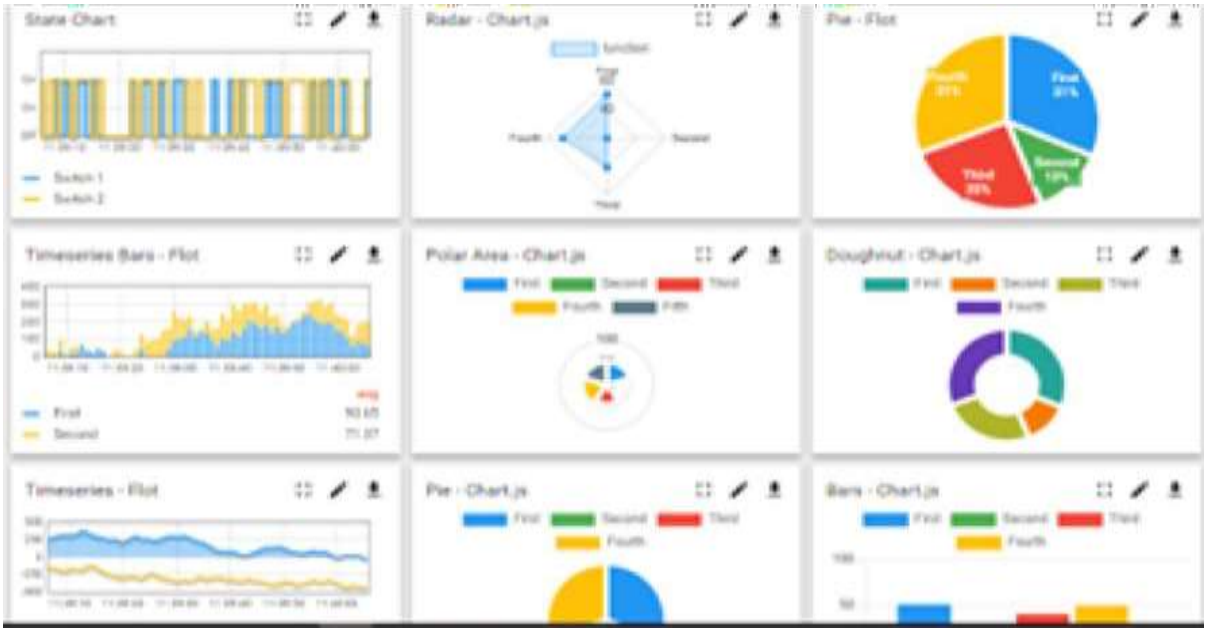
i. UCT IoT Platform(uct Insight)

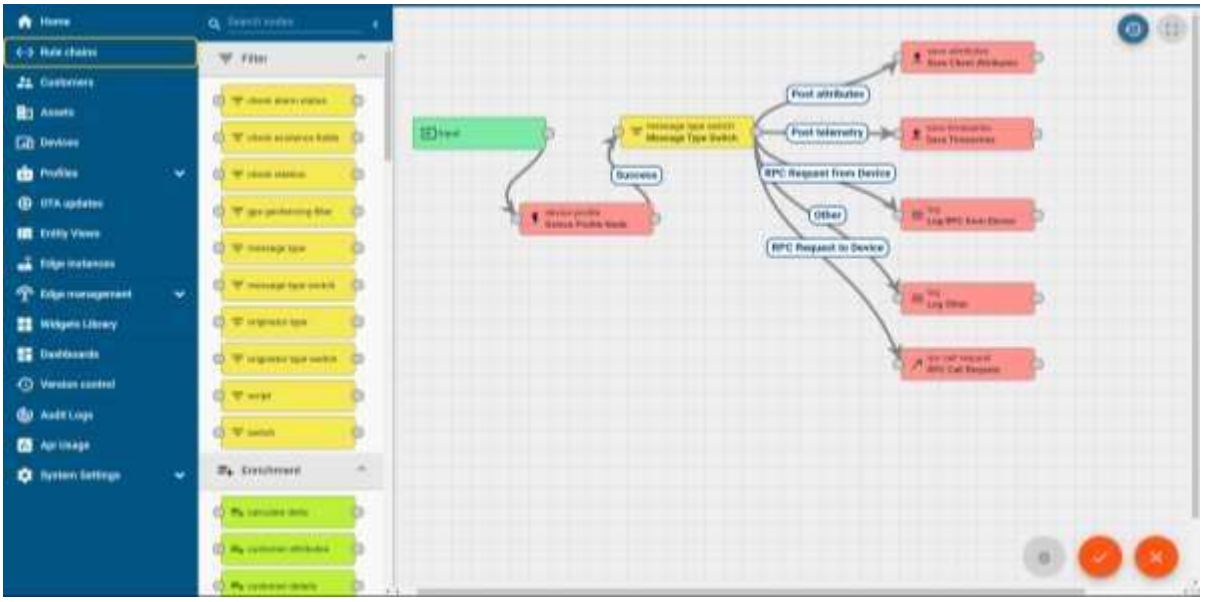
UCT Insight is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable "insight" for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

- It enables device connectivity via industry standard IoT protocols - MQTT, CoAP, HTTP, Modbus TCP, OPC UA
- It supports both cloud and on-premises deployments.

It has features to

- Build Your own dashboard
- Analytics and Reporting
- Alert and Notification
- Integration with third party application(Power BI, SAP, ERP)
- Rule Engine





ii. Smart Factory Platform (FACTORY WATCH)

Factory watch is a platform for smart factory needs.
It provides Users/ Factory

- with a scalable solution for their Production and asset monitoring
- OEE and predictive maintenance solution scaling up to digital twin for your assets.
- to unleased the true potential of the data that their machines are generating and helps to identify the KPIs and also improve them.
- A modular architecture that allows users to choose the service that they what to start and then can scale to more complex solutions as per their demands.

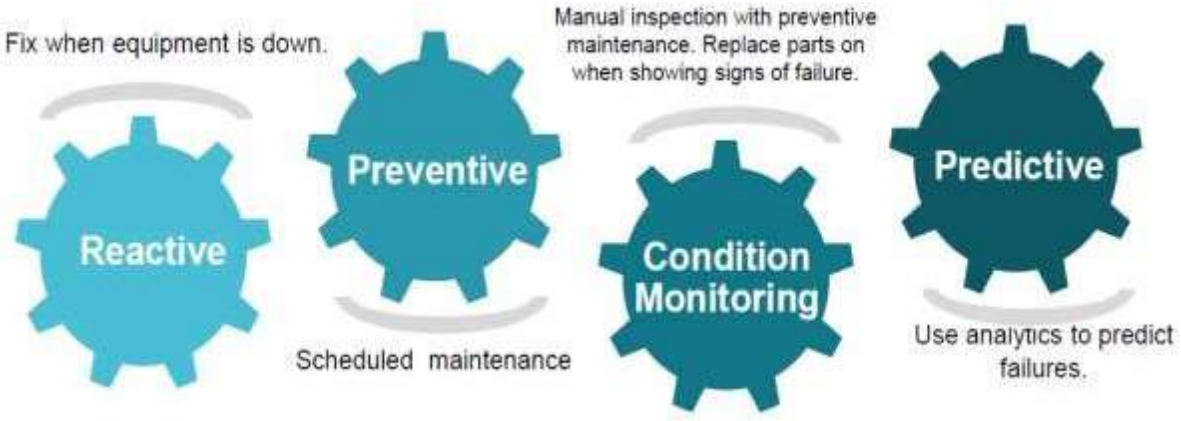
Its unique SaaS model helps users to save time, cost and money.

iii. LoRaWAN based Solution

UCT is one of the early adopters of LoRAWANteschnology and providing solution in Agritech, Smart cities, Industrial Monitoring, Smart Street Light, Smart Water/ Gas/ Electricity metering solutions etc.

iv. Predictive Maintenance

UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.



2.2 About upskill Campus (USC)

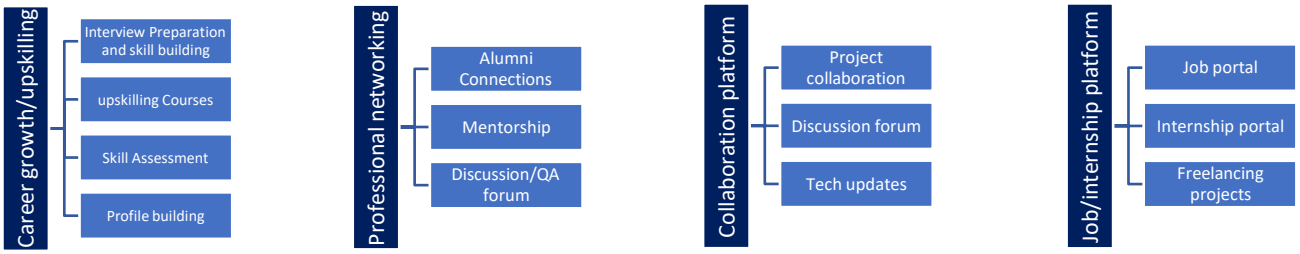
upskill Campus along with The IoT Academy and in association with Uniconverge technologies has facilitated the smooth execution of the complete internship process.

USC is a career development platform that delivers **personalized executive coaching** in a more affordable, scalable and measurable way.

Seeing need of upskilling in self paced manner along-with additional support services e.g. Internship, projects, interaction with Industry experts, Career growth Services

upSkill Campus aiming to upskill 1 million learners in next 5 year

https://www.upskillcampus.com/



2.3 The IoT Academy

The IoT academy is EdTech Division of UCT that is running long executive certification programs in collaboration with EICT Academy, IITK, IITR and IITG in multiple domains.

2.4 Objectives of this Internship program

The objective for this internship program was to

- ☛ get practical experience of working in the industry.
- ☛ to solve real world problems.
- ☛ to have improved job prospects.
- ☛to have Improved understanding of our field and its applications.
- ☛to have Personal growth like better communication and problem solving.

2.5 Reference

[1] https://learn.upskillcampus.com/s/mycourses

[2] https://www.uniconvergetech.in/

[3] https://upskillcourses.com/

3. Problem Statement

In today's world managing the increasing traffic system is a big problem. Traffic congestion is a major problem in many cities of India along with other countries. Failure of signals, poor law enforcement and bad traffic management has led to traffic congestion. One of the major problems with Indian cities is that the existing infrastructure cannot be expanded more, and thus the only option available is better management of the traffic.

As cities around the world continue to grow and evolve, traffic congestion has become a significant issue. This problem is further exacerbated by the rapid increase in population, the rise of urbanization, and the expanding number of vehicles on the roads. To address these challenges, smart city initiatives are being implemented to leverage technology and data-driven solutions.

One crucial aspect of smart city planning is the ability to forecast and predict traffic patterns accurately. Therefore, the problem statement revolves around developing effective forecasting models for smart city traffic patterns.

The goal is to create a system that can accurately forecast traffic patterns within a smart city environment. This involves predicting traffic congestion levels, travel times, and traffic flow across various routes, intersections, and road networks. The forecasts should consider multiple factors such as time of day, day of the week, weather conditions, special events, and historical traffic data.

The accuracy and reliability of the forecasting models can be evaluated using metrics such as mean absolute error (MAE), root mean square error (RMSE), or mean absolute percentage error (MAPE). These metrics quantify the differences between the predicted and actual traffic patterns.

4. Existing and Proposed solution

Provide summary of existing solutions provided by others, what are their limitations?

1. Traditional Statistical Models:
   Existing solutions often employ traditional statistical models such as autoregressive integrated moving average (ARIMA) or exponential smoothing techniques.

   Limitations: These models typically assume linear relationships and fail to capture the complexities of traffic patterns affected by various factors. They may struggle to handle large-scale data and real-time updates, limiting their accuracy and scalability.

2. Machine Learning (ML) Approaches:

   ML-based solutions utilize algorithms like support vector machines (SVM), random forests, or gradient boosting to predict traffic patterns.

   Limitations: These approaches often rely heavily on feature engineering, which requires domain expertise and can be time-consuming. They may struggle to handle dynamic and nonlinear relationships in traffic data, resulting in suboptimal predictions.

3. Deep Learning Models:

   Deep learning models, such as recurrent neural networks (RNNs) and convolutional neural networks (CNNs), have shown promise in traffic forecasting.

   Limitations: Deep learning models often require a substantial amount of labeled training data, which can be challenging to obtain in some smart city environments. Additionally, they can be computationally intensive and may lack interpretability, making it difficult to understand the underlying factors influencing traffic patterns.

4. Traffic Simulation Models:

Simulation models use traffic flow simulations to predict traffic patterns based on known road network layouts and traffic rules.

Limitations: These models rely on accurate input data, including road network details, traffic rules, and driver behavior models. Inaccurate or outdated input data can lead to inaccurate predictions. The computational complexity of simulation models can also limit their real-time applicability.

Existing solutions for forecasting smart city traffic patterns vary in their approaches and limitations. Challenges include handling large-scale and dynamic data, incorporating real-time updates, ensuring data quality and reliability, interpretability of models, and scalability for entire city environments. Future advancements may focus on developing hybrid models, improving data integration techniques, addressing real-time processing challenges, and enhancing interpretability and transparency in the predictions.

What is your proposed solution?

We go through many machine algorithms that are good for forecasting smart city traffic patterns and at the end decided that we will be implementing the Decision Tree algorithm of machine learning to forecast the traffic patterns.

A well-liked machine learning algorithm for classification and regression is the decision tree algorithm. The result is a model that takes the shape of a tree, with each internal node standing in for a feature or attribute, each branch standing in for a decision rule, and each leaf node standing in for an outcome or a class label.

Using RFID Tag, we can easily manage traffic and help to emergency vehicle by giving green signal. We place reader and antenna at every cross-road and RFID tag in every emergency vehicle (police, ambulance, etc..).

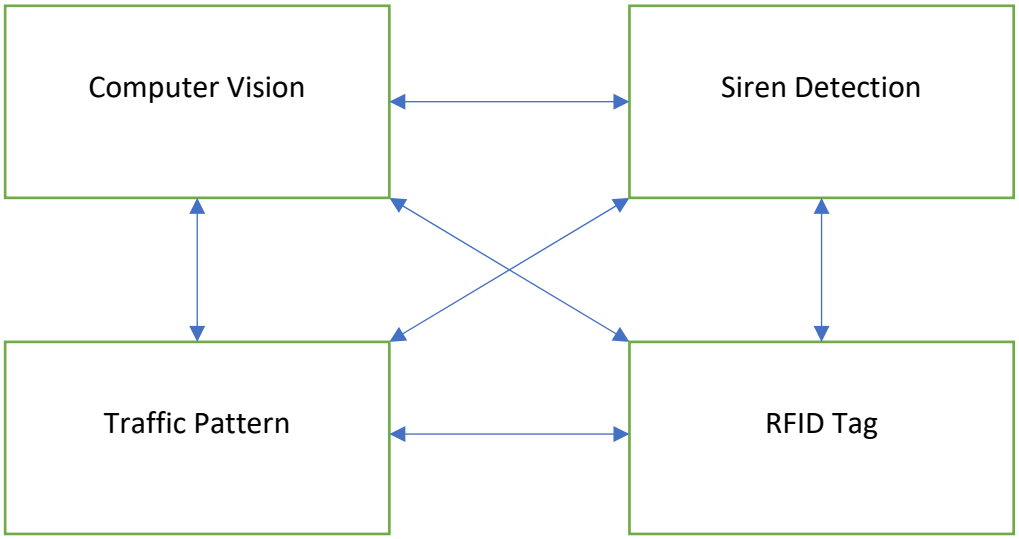
Now, when the emergency vehicle come near any antenna, then antenna detect it.

But that’s not the only solution we need to embedded a siren detection system to avoid any false signal and actives.

Also, we need to use computer vision Technology to avoid mistakes and fraud to make it more secure and reliable management.

So, here we need multiple Technology to create best traffic management system.

1. Computer Vision: To use to detect emergency vehicle.
2. Siren Detection: To detect a siren of the emergency vehicle.
3. RFID Tag: To Track the emergency vehicle.
4. Traffic Pattern Analysis: To Identified the traffic peak time and manage it by signal lights and traffic police team.



Code submission (Github link)

https://github.com/ShabdprakashThakkar/upskillcampus

5. Proposed Design/ Model

Designing a flow for forecasting traffic patterns using the decision tree algorithm in machine learning involves several steps.

Here's how the decision tree algorithm works:

Data Preparation: Create a labelled dataset with target labels that correlate to the input features (the dependent variable) as well as independent variables. The format of the data must be one that the decision tree algorithm can handle.

Attribute Selection: The best feature is chosen by the algorithm to divide the data at each internal node. To maximise the homogeneity of the target labels within each subset, the splitting is carried out depending on certain criteria, such as information gain or Gini impurity.

Building the Tree: The algorithm divides the data recursively according to the chosen feature, starting at the root node. Until a halting requirement is satisfied, this process goes on. The stopping criterion can be a minimal information gain threshold, a certain number of samples per leaf, or a predetermined depth limit.

Handling Categorical and Numerical Features: The algorithm divides the data recursively according to the chosen feature, starting at the root node. Until a halting requirement is satisfied, this process goes on. The stopping criterion can be a minimal information gain threshold, a certain number of samples per leaf, or a predetermined depth limit.

Handling Missing Values: Surrogate splits and assigning missing values to the most prevalent class or the majority class at that node are two ways that decision trees can deal with missing data.

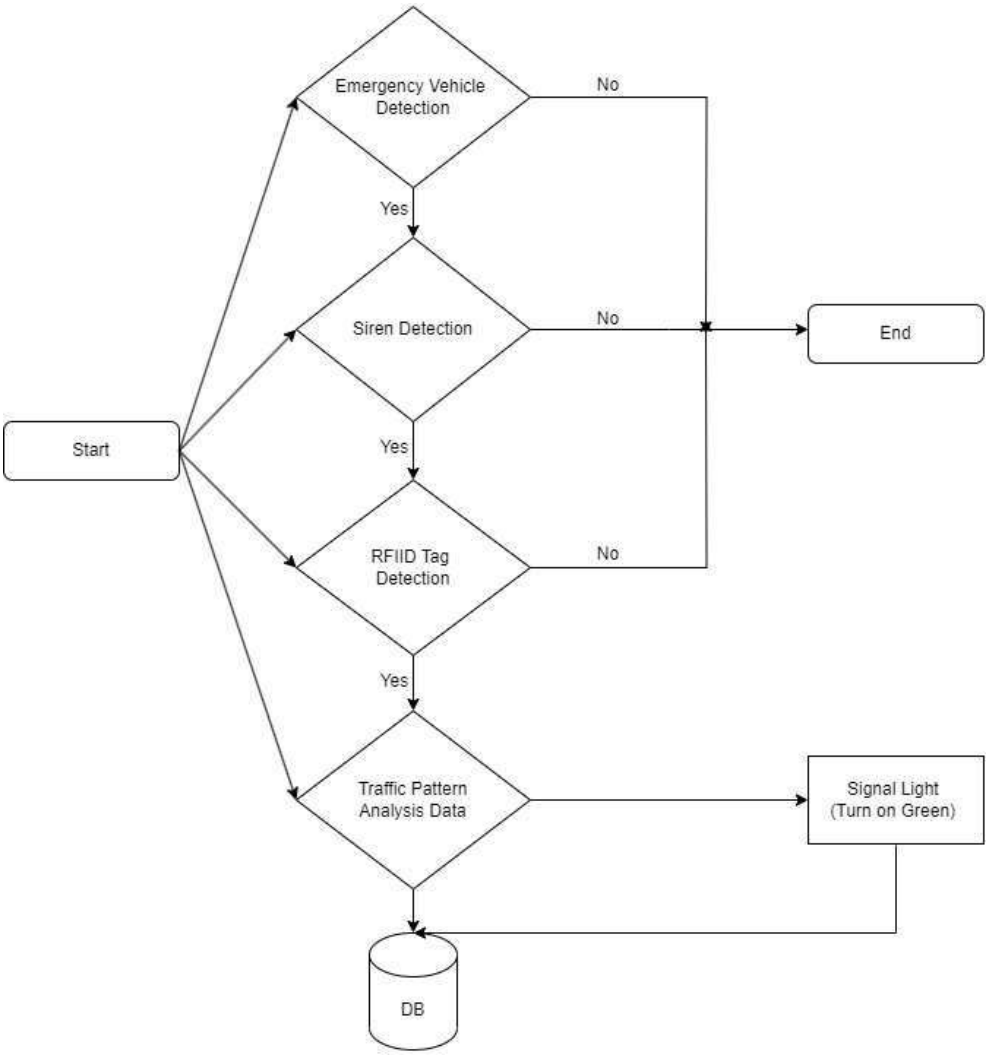
Pruning: After the tree has been constructed, overfitting can be avoided by using pruning strategies such cost complexity pruning (also known as alpha pruning or weakest link pruning). Pruning gets rid of extra branches or nodes that don't help the model perform well on unobserved data.

Prediction: The input data moves through the decision tree to create a forecast, adhering to the decision rules at each node until it reaches the leaf node. The input data is given the projected result or class label related to the leaf node.

Evaluation: Depending on the task at hand, determine the decision tree model's performance using the appropriate metrics. Metrics like accuracy, precision, recall, and F1-score are frequently employed in categorization. Metrics like mean squared error (MSE) or mean absolute error (MAE) are frequently used for regression.

Feature Importance: Decision trees can shed light on the significance of a feature. You can figure out which features have the biggest influence on the model's decisions by examining the splits and the resulting drop in impurity or information gain.

Ensemble Techniques: To improve prediction accuracy and tackle more challenging jobs, decision trees can be integrated into ensemble models like random forests or gradient boosting techniques.



Here's a proposed design flow to help you get started:

1. Data Collection

- Gather historical traffic data including features like time of day, day of the week, weather conditions,holidays, road construction, etc.

2. Data Preprocessing

- Clean the data by removing duplicates, handling missing values, and addressing outliers.
- Perform feature engineering to extract meaningful features.
- Convert categorical variables into numerical representations.

3. Splitting the Data

- Divide the preprocessed data into training and testing sets (e.g., 80:20 or 70:30 ratio).

4. Training the Decision Tree Model

- Apply the decision tree algorithm to the training data.
- Select the best feature to split the data at each internal node based on criteria like information gain or Gini impurity.
- Tune hyperparameters to avoid overfitting or underfitting.

5. Model Evaluation

- Evaluate the trained decision tree model using the testing data.
- Calculate metrics such as accuracy, precision, recall, or mean absolute error (MAE).

6. Model Deployment

- Deploy the decision tree model for traffic pattern forecasting using the entire dataset for training.
- Consider retraining or updating the model periodically as new data becomes available.

7. Prediction and Monitoring

- Use the deployed decision tree model to forecast traffic patterns based on new input data.
- Continuously monitor the performance of the model and consider necessary adjustments.

The decision tree algorithm can be effective for forecasting traffic patterns due to several reasons:

1. Interpretability: Decision trees offer an understandable and transparent model representation. The decision-making processes used to forecast traffic patterns may be understood and visualised thanks to the tree structure. The important elements impacting traffic patterns, such as the time of day, the weather, or holidays, can be identified with the use of this interpretability.

2. Handling Nonlinear Relationships: There are frequently intricate nonlinear linkages in traffic patterns. Such interactions can be captured by decision trees by using many splits on various features. The method can handle both categorical and continuous variables, which enables it to capture the variety of traffic data.

3. Feature Importance: Insights into the significance of various factors in anticipating traffic patterns can be gained using decision trees. You can determine which attributes have the greatest impact on traffic patterns by looking at the splits in the tree and the corresponding drop in impurity. Understanding the underlying causes of traffic fluctuations can be aided by this information.

4. Handling Interactions: Decision trees naturally record feature-feature interactions. For instance, a decision tree can discover that the day of the week and the time of day both affect traffic patterns during rush hour, with the impact of the time of day varying depending on whether it is a weekday or a weekend. Without requiring explicit feature engineering, the programme can automatically learn these interactions.

5. Robustness to Outliers: Decision trees are typically resistant to data outliers. Decision trees can manage outliers without severely distorting the general forecasts, even if they can have a considerable impact on traffic patterns. The algorithm can handle outliers and yet produce accurate predictions since it can split based on information gain or Gini impurity reduction.

6. Handling Missing Values: Due to a variety of factors, such as sensor malfunctions or data gathering problems, traffic statistics may have missing numbers. Decision trees can deal with missing values by either employing surrogate splits or assigning them to the most prevalent class. Due to its adaptability, the method can handle imperfect datasets that are frequently present in applications of real-world traffic forecasting.

7. Ensemble Techniques: Decision trees can be included in ensemble approaches, such as gradient boosting or random forests, to further improve the predictability of traffic patterns. Using ensemble approaches, you can average or mix different decision trees to lessen overfitting and increase the model's generalisation potential.

RFID Tag Model:

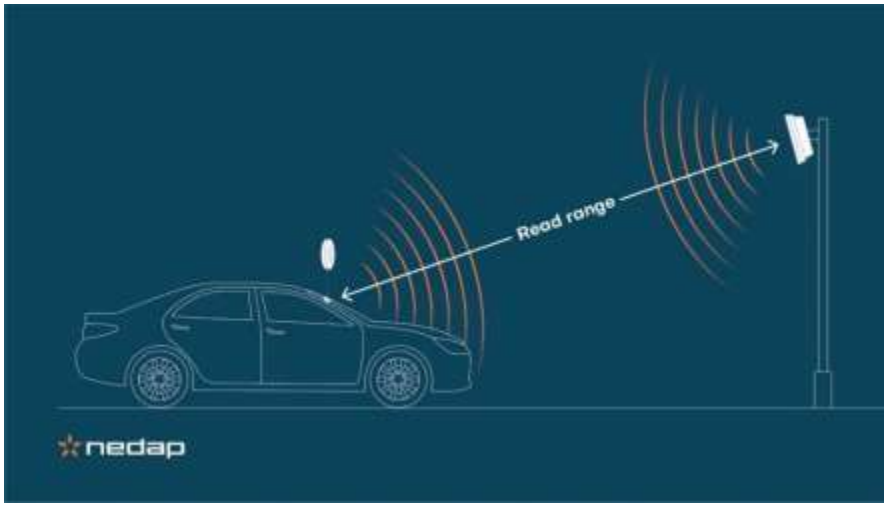
A RFID tag-based vehicle tracking system is a technology that uses radio frequency identification (RFID) tags to track and monitor vehicles. RFID tags are small devices that contain a unique identifier and can be attached to or embedded in vehicles. The tracking system uses RFID readers and antennas to detect and read these tags, allowing the system to identify and monitor the movement of vehicles.

1. RFID Tags: RFID tags used in vehicle tracking systems can come in various forms, as mentioned in the previous explanation. They have a unique identification number that is stored in a memory chip. This identifier allows each vehicle to be uniquely identified in the system.

2. RFID Readers: RFID readers are devices that send out radio waves to detect nearby RFID tags. When a tag comes into the range of the reader, it receives the radio waves, and the tag responds by sending back its unique identification number to the reader.

3. Antennas: Antennas are used to improve the range and efficiency of the RFID system. They are positioned strategically to ensure that the RFID reader can detect the tags efficiently.

4. RFID Tracking Software: The RFID reader is connected to a computer or a central system that has RFID tracking software installed. This software processes the information received from the RFID tags and stores the data in a database.



Setting up a RFID tag-based vehicle tracking system involves the following steps:

Step 1: Identify the Requirements: Determine the specific needs of your tracking system, such as the range of detection, the number of vehicles to track, and the type of data to be collected.

Step 2: Choose the RFID Tags: Select the appropriate RFID tags based on the requirements and the type of vehicles you want to track (e.g., windshield tags, hang tags, or license plate tags).

Step 3: Install RFID Readers and Antennas: Position the RFID readers and antennas at strategic locations, such as entry and exit points or overhead positions, to ensure optimal tag detection and read rates.

Step 4: Connect to Tracking Software: Connect the RFID readers to a computer or a central system where the RFID tracking software is installed. Ensure that the software is compatible with the RFID readers you are using.

Step 5: Configure the Tracking Software: Set up the RFID tracking software according to your specific tracking needs. This may include configuring data storage, setting up alarms or notifications, and customizing the user interface.

Step 6: Test the System: Test the RFID tracking system to ensure that it is working correctly. Verify that the readers can detect the tags accurately and that the software is processing the data as expected.

Step 7: Monitor and Maintain: Regularly monitor the system to ensure that it is functioning correctly. Perform maintenance checks on the RFID readers and antennas to keep them in good working condition.

Siren Detection:

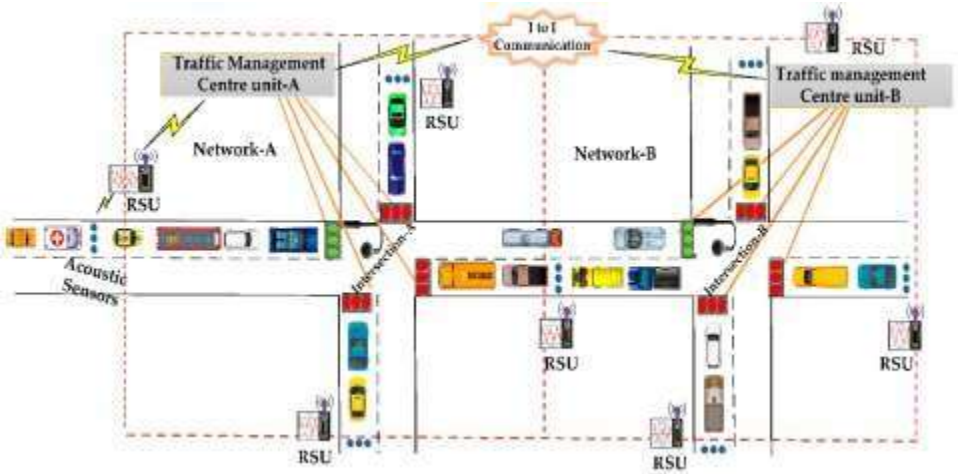
1. Audio Acquisition: The system starts by capturing audio from the environment using microphones or audio sensors installed at strategic locations along the roadways.

2. Sound Processing: The captured audio is then processed to filter out background noises and isolate the siren sounds. Advanced audio processing algorithms, such as digital signal processing (DSP) techniques or machine learning models, are often used for this purpose.

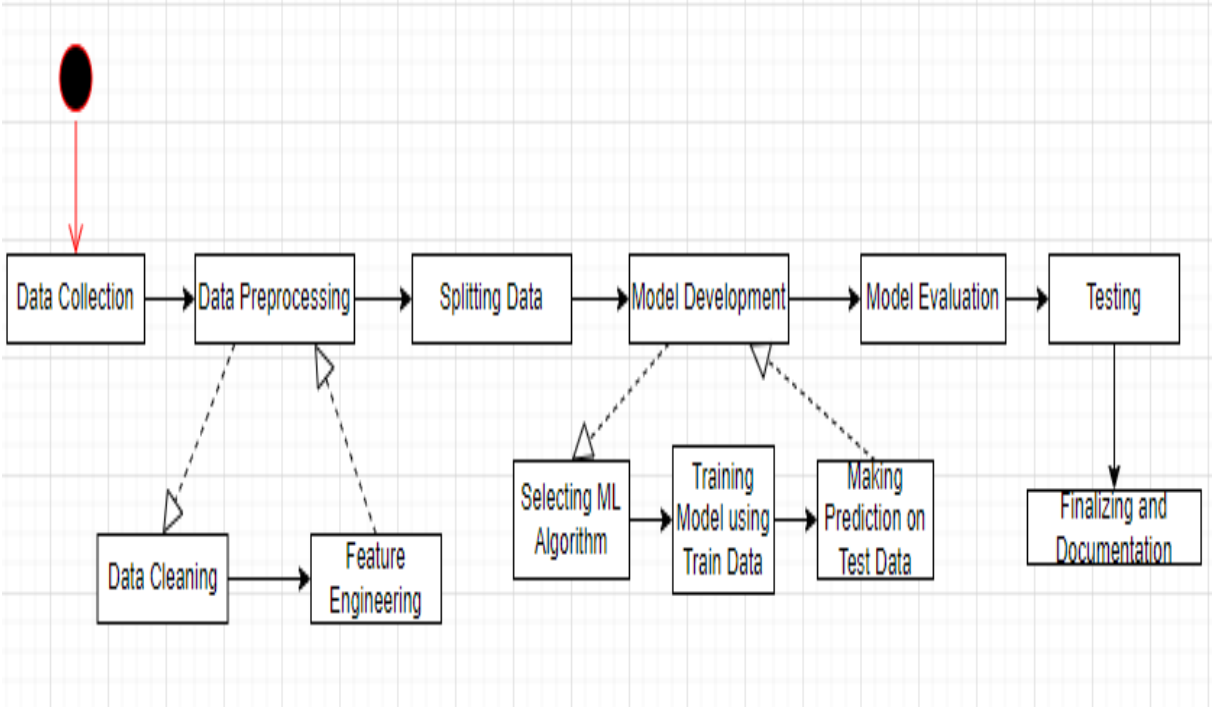
3. Siren Identification: Once the siren sound is isolated, the system identifies it as an emergency vehicle siren based on predefined patterns or characteristics of emergency sirens. This process may involve pattern recognition algorithms or machine learning classifiers trained to distinguish emergency vehicle sirens from other sounds.

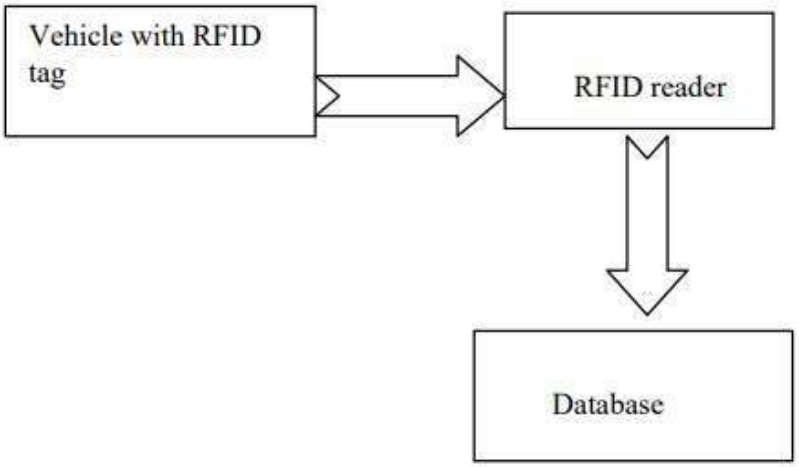
4. Activation of Response Mechanism: After successfully identifying the siren, the system triggers the appropriate response mechanism. This could include changing traffic signal timings to prioritize the path of the emergency vehicle, controlling traffic gates or barriers to allow the emergency vehicle to pass, or activating warning signs for other drivers to yield to the emergency vehicle.

5. Optional Notifications: Additionally, the system can send notifications to drivers' smartphones or in-vehicle communication systems, alerting them of the approaching emergency vehicle and instructing them to clear the way.



5.1 Interfaces (if applicable)





6. Performance Test

Although the decision tree algorithm has several benefits, it's crucial to remember that no one algorithm is always better. The precise properties of the data, the difficulty of the issue, and the resources available all play a role in the algorithm selection process. To choose the best strategy, it is advised to examine various algorithms and gauge their effectiveness in your traffic forecasting work.

The decision tree algorithm's architecture of a traffic forecasting system can be significantly impacted by identified constraints. Following are some typical restrictions and suggestions about how to handle them:

1. Limited Data Availability: The quality and dependability of the forecasting model may be hampered by a lack of past traffic data. To manage this constraint, some suggestions are as follows:

    - Acquiring additional data: See if there is a way to collect more historical traffic data, either by expanding the time frame or by obtaining information from several sources.
    - Data augmentation: Consider data augmentation methods like simulation or synthetic data generation to expand the dataset's size and diversity if getting new data is not practical.
    - Feature engineering: To obtain the most information possible from the little data, concentrate on extracting pertinent aspects.

2. Incomplete or Noisy Data: The decision tree model's performance may be significantly impacted by poor data quality, missing values, or outliers. To manage this constraint, some suggestions are as follows:

    - Data cleaning: To deal with missing values, outliers, and inconsistencies in the dataset, use effective data cleaning procedures. Think about outlier identification algorithms to handle extreme values and imputation techniques for handling missing numbers.
    - Feature selection: To reduce the impact of noisy or irrelevant data, give preference to the selection of robust and relevant attributes.
    - Robust modeling techniques: Utilise noise-resistant decision tree variations, such random forests or gradient boosting, which can manage noisy data and reduce overfitting.

3. Computation Time and Resource Constraints: The decision tree approach can be computationally and resource-intensive, especially when working with huge datasets or complicated features. To manage this constraint, some suggestions are as follows:

    - Sample or subset the data: Consider sampling or sub setting strategies to lessen processing costs while maintaining data integrity if the dataset is too vast.
    - Feature reduction: Reduce the number of features and make the decision tree construction process simpler by using feature selection or dimensionality reduction approaches.
    - Model optimization: To establish a balance between accuracy and computing economy, optimize the decision tree algorithm's hyperparameters and look for the optimum configurations.
    - Distributed computing or parallelization: Use parallelization methods or distributed computing frameworks to quicken the training and assessment procedures, especially when working with huge datasets.

4. Interpretability Requirements: The model's interpretability may be a constraint in some circumstances. Although interpretability of decision trees is fundamental, complex trees with numerous layers might be challenging to comprehend. To manage this constraint, some suggestions are as follows:

    - Limiting the tree depth: Control the decision tree's maximum depth to produce a model that is easier to understand. Pruning methods or hyperparameter manipulation can be used to achieve this.
    - Ensembling with simpler models: To keep the decision tree model understandable while increasing accuracy, combine it with other straightforward, comprehensible models like rule-based models or linear regression.

5. Evolving Traffic Patterns: Various factors, such as urbanisation, modifications to the road system, or changes in travel habits, can cause traffic patterns to vary over time. To manage this constraint, some suggestions are as follows:

    - Regular model updates: Retrain the decision tree model on a regular basis to account for changing traffic patterns.
    - Continuous monitoring: Put in place a monitoring system to track the model's performance in real-time and spot drifts or variations in the traffic patterns. If necessary, update the model or retrain it.

6.1. Test Plan/ Test Cases

Test Plan for Forecasting Traffic Patterns using Decision Tree Algorithm:

1. Test Objective: Verify the precision and dependability of the Decision Tree algorithm-based traffic pattern forecasting model.

2. Test Environment:
   - Programming language: Python
   - Libraries: scikit-learn, pandas, numpy
   - Traffic data set: Historical traffic data containing features such as date, time, weather conditions, etc.
3. Test Data:

   - Prepare a dataset with historical traffic data, including known patterns and corresponding outcomes.
   - Split the dataset into training and testing sets (e.g., 70% training, 30% testing).

4. Test Cases:

   a. Data Preprocessing:
   i. Verify that the dataset is loaded correctly, and all required features are present.
   ii. Check for missing or invalid values and ensure proper handling or imputation.
   iii. Validate the normalization or scaling of numerical features if applicable.

   b. Model Training:
   i. Train the Decision Tree model using the training dataset.
   ii. Verify that the model has been trained successfully without any errors.
   iii. Validate that the model has learned patterns and relationships from the training data.

   c. Model Evaluation:
   i. Apply the trained model to the testing dataset.
   ii. Compare the predicted traffic patterns with the actual patterns in the testing dataset.
   iii. Calculate evaluation metrics such as accuracy, precision, recall, and F1-score.
   iv. Ensure that the model performance meets the defined acceptance criteria.

d. Model Validation:

i. Use cross-validation techniques (e.g., k-fold cross-validation) to assess the model's generalization capabilities.

ii. Verify that the model performs consistently across different subsets of the data.

iii. Ensure that the model does not overfit or underfit the training data.

e. Performance Testing:

i. Measure the training and prediction times to ensure they are within acceptable limits.

ii. Evaluate the model's performance on large datasets to validate scalability.

f. Boundary and Edge Cases:

i. Test the model's behavior with extreme or outlier values in the input features.

ii. Verify that the model handles unexpected or novel traffic patterns gracefully.

g. Integration Testing:

i. Validate the integration of the Decision Tree algorithm with other components or systems.

ii. Verify the compatibility and data exchange between the traffic forecasting model and external systems.

6.2. Test Procedure

Test Procedure for Forecasting Traffic Patterns using Decision Tree Algorithm:

1. Set up the Test Environment

    - Install the required software and libraries (Python, scikit-learn, pandas, numpy).
    - Set up the development environment with the necessary dependencies.
    - Ensure that the historical traffic dataset is available for testing.

2. Identify Test Data:

    - Select a subset of the historical traffic dataset for testing.
    - Split the dataset into training and testing sets (e.g., 70% training, 30% testing).

3. Preprocessing:

    - Load the training dataset into the system.
    - Perform any necessary preprocessing steps, such as handling missing values, normalizing or scaling features, and encoding categorical variables.

4. Model Training:

    - Implement the Decision Tree algorithm using the scikit-learn library.
    - Train the Decision Tree model using the training dataset.
    - Ensure that the training process completes without any errors or exceptions.
    - Validate that the model has been trained successfully by inspecting its attributes and structure.

5. Model Evaluation:

- Load the testing dataset into the system.
- Apply the trained Decision Tree model to the testing dataset to predict traffic patterns.
- Compare the predicted traffic patterns with the actual patterns in the testing dataset.
- Calculate evaluation metrics such as accuracy, precision, recall, and F1-score to assess the model's performance.
- Ensure that the model meets the defined acceptance criteria.

6. Performance Testing:

- Measure the training and prediction times of the model to ensure they meet the performance requirements.
- Evaluate the model's performance on larger datasets to assess scalability.
- Identify any bottlenecks or performance issues and address them accordingly.

6.3. Performance Outcome

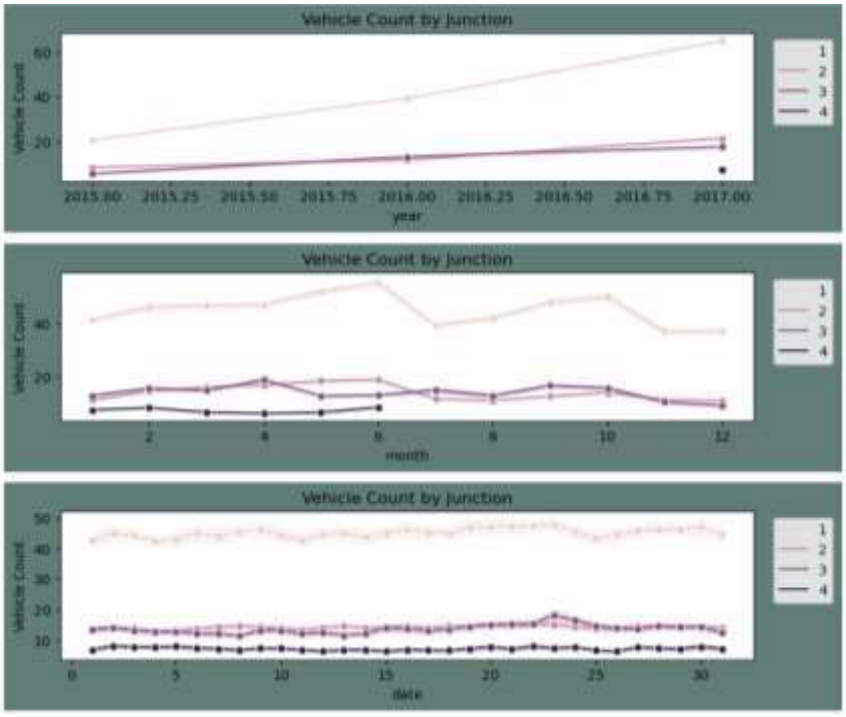
The performance outcome of a Decision Tree algorithm for forecasting traffic patterns can vary depending on several factors, including the quality of the data, the complexity of the traffic patterns, the choice of features, and the tuning of the model parameters.

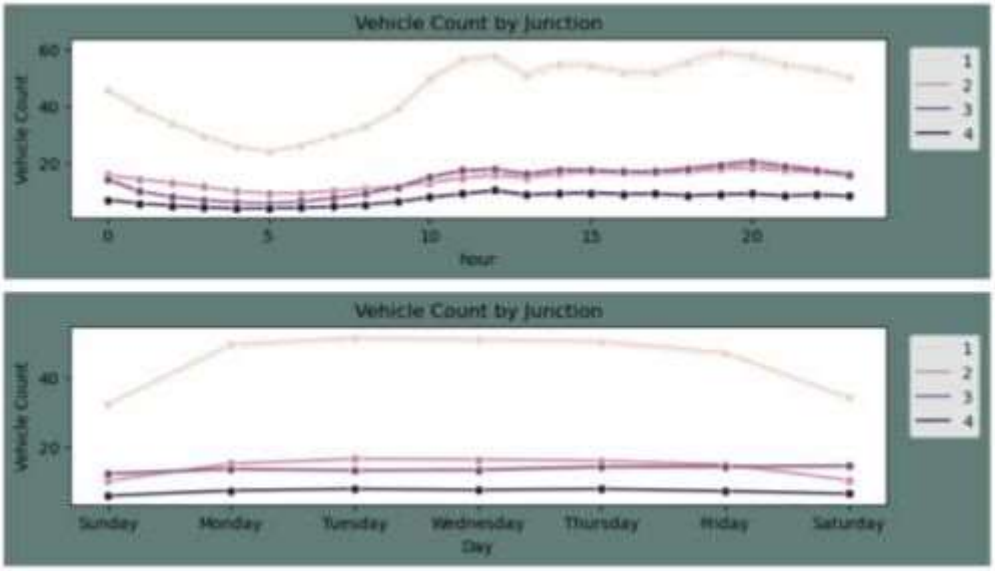
1. Prediction made by the Machine learning model using Decision tree algorithm:

	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
...	...	...	...	...
48115	2017-06-30 19:00:00	4	11	20170630194
48116	2017-06-30 20:00:00	4	30	20170630204
48117	2017-06-30 21:00:00	4	16	20170630214
48118	2017-06-30 22:00:00	4	22	20170630224
48119	2017-06-30 23:00:00	4	12	20170630234

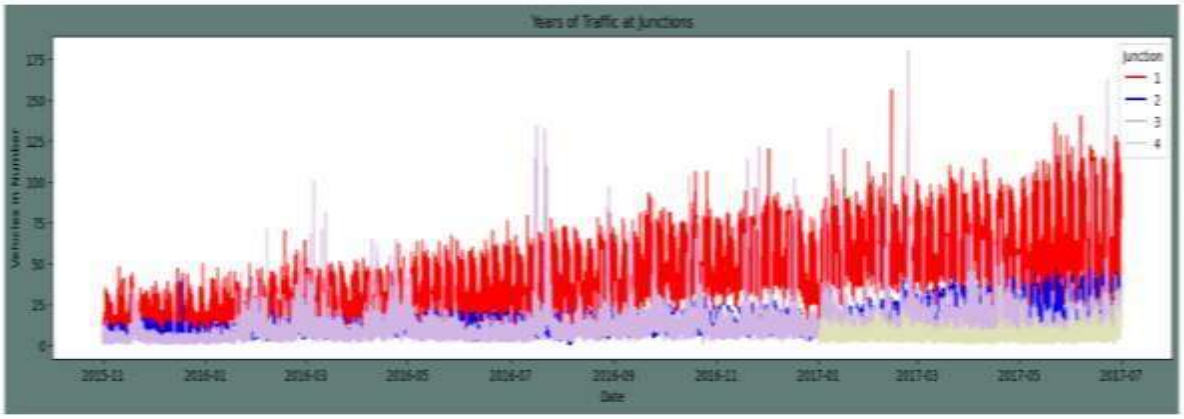
48120 rows × 4 columns

2. Visualizing Prediction of traffic at different Junction:





3. Visualizing the Training data and the prediction:



4. Statistics:

```
The test for  junction 1
ADF Statistic: -7.148077688286681
p-value: 3.19384299636279 86e-10
Critical Values:
        1%: -3.431
        5%: -2.862
        10%: -2.567
The test for  junction 2
ADF Statistic: -8.041077275597678
p-value: 1.8463683667709498e-12
Critical Values:
        1%: -3.431
        5%: -2.862
        10%: -2.567
The test for  junction 3
ADF Statistic: -8.236600353698044
p-value: 5.86729585586 87e-13
Critical Values:
        1%: -3.431
        5%: -2.862
        10%: -2.567
The test for  junction 4
ADF Statistic: -10.671803424843981
p-value: 4.148828810495 9944e-19
Critical Values:
        1%: -3.431
        5%: -2.862
        10%: -2.567
```

7. My learnings

During the immersive 6-week internship with UpSkill Campus and UniConverge Technologies Pvt. Ltd., I gained invaluable insights into the world of Python programming and its multifaceted applications, particularly in the realms of Artificial Intelligence, Data Science, and Machine Learning. This immersive experience allowed me to delve deep into the intricacies of these fields and discover the myriad algorithms that underpin modern solutions. The internship emphasized the significance of Machine Learning in our contemporary landscape, and I was fortunate to explore a range of algorithms and gain hands-on experience in implementing them for practical problem-solving.

A standout aspect of the internship was the comprehensive utilization of Python's powerful libraries. I effectively employed the pandas library to efficiently read and preprocess data, a crucial step in any Data Science endeavor. Additionally, the scikit-learn library facilitated the seamless division of data into training and testing sets, enabling the application of diverse machine learning algorithms. The internship also equipped me with the skills to visually analyze and comprehend data through the use of matplotlib and seaborn, enabling me to present insights in a clear and concise manner.

This transformative experience not only expanded my technical proficiency but also offered the unique opportunity to collaborate on a real-world project. I extend my gratitude to UpSkill Campus and UCT for providing this exceptional platform, which undoubtedly sets me on a path to a successful future in the field. The knowledge and practical skills I gained during this internship are invaluable, and I eagerly look forward to leveraging them as I advance in my career.

8. Future work scope

1. Feature Engineering: Enhance decision tree performance by crafting valuable input features. Include elements like time of day, day of the week, holidays, and traffic-related factors such as jams, collisions, and construction. This preprocessing creates a more robust model capable of capturing complex relationships.

2. Real-Time Data Integration: Improve prediction accuracy by incorporating real-time data like traffic flow, weather, and events. The decision tree adapts to current conditions, resulting in more precise traffic forecasts.

3. Hierarchical Decision Trees: Recognize hierarchical traffic patterns (daily, weekly, monthly) by building a decision tree hierarchy. Lower-level trees focus on shorter periods, while the top-level tree forecasts overall trends. This approach boosts accuracy across different time spans.

4. Spatial Dependencies: Consider geographic information system (GIS) data to account for spatial elements like road networks, congestion, and population density. The decision tree becomes more accurate with spatial variables, reflecting localized traffic patterns.

5. Dynamic Updating: Implement a dynamic updating mechanism to account for changing variables such as urban development, road system modifications, or significant events. Regularly retrain the decision tree with recent data to maintain forecast accuracy.

6. Ensemble Methods: Enhance accuracy with decision tree ensembles (e.g., gradient boosting, random forests). These methods mitigate overfitting, produce more reliable predictions, and handle noise and oscillations in data better than individual trees.