Industrial Internship Report on

Forecasting of Smart city traffic patterns

Prepared By:- Bharath Merugu

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.

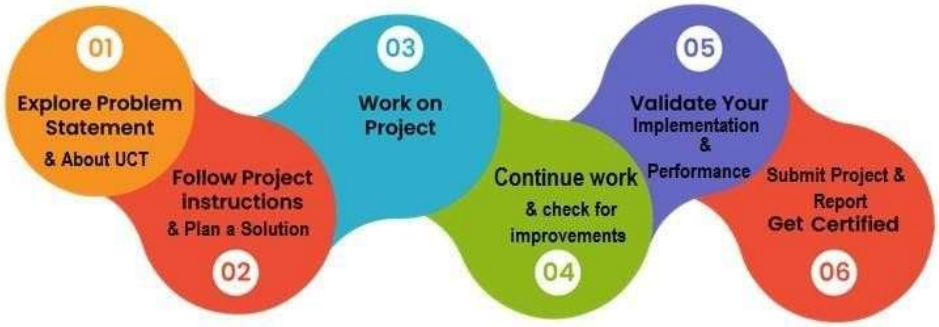
TABLE OF CONTENTS

Preface

In terms of population and economic growth, India is the second-largest country. India's major cities are plagued by issues with traffic congestion. In wealthy nations, sustaining Intelligent Transport Management Systems (ITMS) is a realistic challenge, and enhancing urban areas is also fraught with challenges. One of the most significant issues facing India is this one. This is brought on by limited space, high costs, and delayed infrastructure growth compared to the quick rise in vehicle numbers. Travellers require traffic flow information to aid in better travel choices in congested areas and to increase the effectiveness of traffic operations. To manage motorway networks, it will be more beneficial to forecast short-term traffic flow. This traffic flow prediction uses real-time data to forecast traffic conditions for the next 5 to 20 minutes. Every nation on earth strives to make their traffic management system more effective. Different techniques have been employed by researchers to forecast motorway traffic in urban settings. The most prevalent issue in many of the emerging cities, despite the growth of transit services, is traffic congestion. With the fast expansion of the IT industry, both the population and the number of automobiles rose in major cities like Bangalore and Delhi. To control traffic congestion and notify drivers to choose alternate routes to avoid congestion and save time, a smart and intelligent traffic information system is needed. This is why India's urban regions need intelligent traffic systems (ITS).

Data science is the study of data with the goal of gaining important business insights. In order to examine enormous amounts of data, this multidisciplinary approach incorporates ideas and methods from the domains of mathematics, statistics, artificial intelligence, and computer engineering. Data scientists can ask and receive answers to questions like what occurred, why it occurred, what will occur, and what can be done with the outcomes thanks to this study. Machine learning is a subset of artificial intelligence (AI) that enables computers to learn from data without being explicitly taught, whereas data science combines tools, methodologies, and technology to derive

meaning from data. Because it can find patterns and relationships in enormous volumes of data and base forecasts on these relationships, it is the perfect tool for predicting agricultural productivity.



Introduction

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

FACTORY WATCH

ii. Smart Factory Platform ()

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.

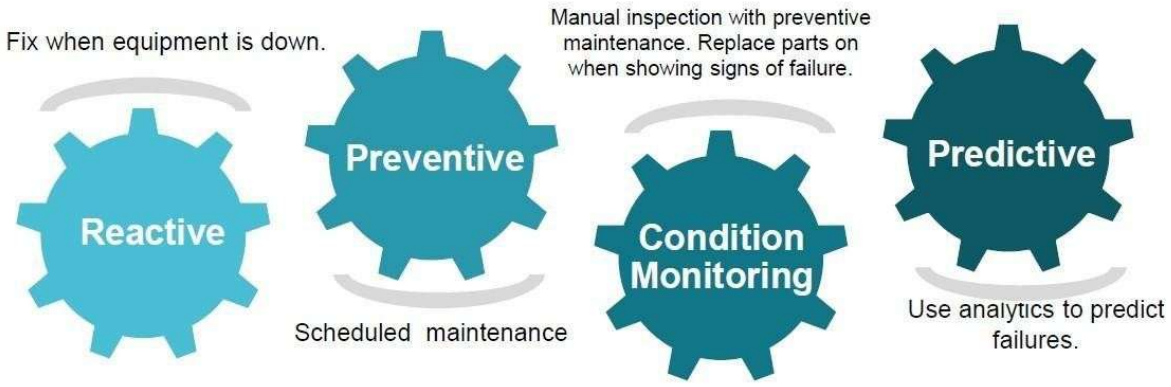
Machine	Operator	Work Order ID	Job ID	Job Performance	Job Progress		Output		Rejection	Time (mins)				Job Status	End Customer
					Start Time	End Time	Planned	Actual		Setup	Pred	Downtime	Idle		
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i

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.

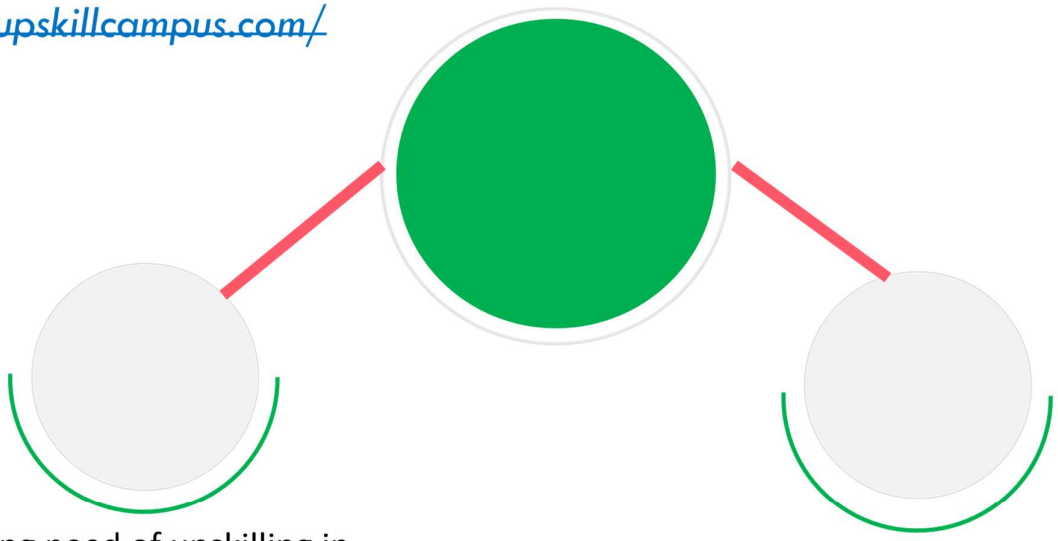


About upskill Campus (USC)

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.

https://www.upskillcampus.com/



Seeing need of upskilling in

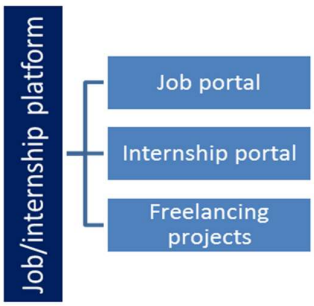
Self-paced manner along-with

additional support services e.g.

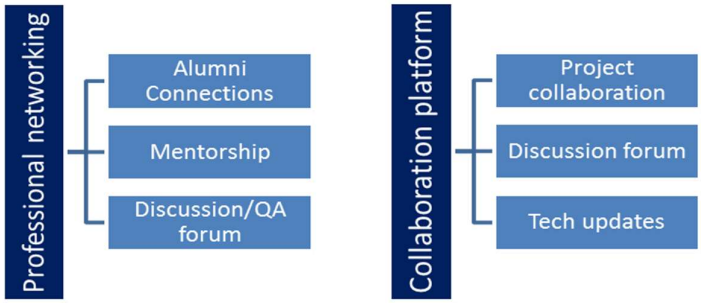
Internship, projects, interaction with Industry experts, Career

upSkill Campus aiming

to upskill 1 million learners in next 5 year



growth Services



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.

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.

Reference

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

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

[3] https://upskillcourses.com/

Problem Statement

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.

Forecasting smart city traffic patterns is a complex problem with significant implications for traffic management, city planning, and overall urban efficiency. By leveraging advanced data analytics techniques, machine learning algorithms, and real-time data processing, accurate traffic forecasts can be obtained. These forecasts can assist city officials in making

informed decisions, optimizing transportation systems, and improving the quality of life for citizens in the smart city environment.

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.

Existing and Proposed solution

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 realtime 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.

Code submission (Github Link)

https://github.com/MeruguBharath11/UpskillCampus02/blob/main/Traffic-Patterns.ipynb

Report submission (Github Link)

https://github.com/MeruguBharath11/UpskillCampus02/blob/main/Forecasting_of_smart_city_traffic_patterns_Bharath_USC_UCT.pdf

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.

Here's how the decision tree algorithm works:

1.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.

2.Attribute Selection: The best feature is chosen by the algorithm to divide the data at each internal node. To maximize 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.

3.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.

4.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.

5.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.

6.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.

7.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.

8.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.

9.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.

10.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.

The decision tree technique is renowned for its interpretability because the final model is simple to see and comprehend. However, if the data is noisy or the regularization is not done correctly, decision trees can also be vulnerable to overfitting. To increase generalization and robustness, it is crucial to carry out rigorous validation and apply strategies like pruning and ensemble methods.

What value addition are you planning?

The decision tree algorithm offers a flexible and understandable method for predicting traffic patterns, allowing you to obtain knowledge of the underlying variables affecting traffic and anticipate future behavior using data from the past.

Proposed Design/ Model

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

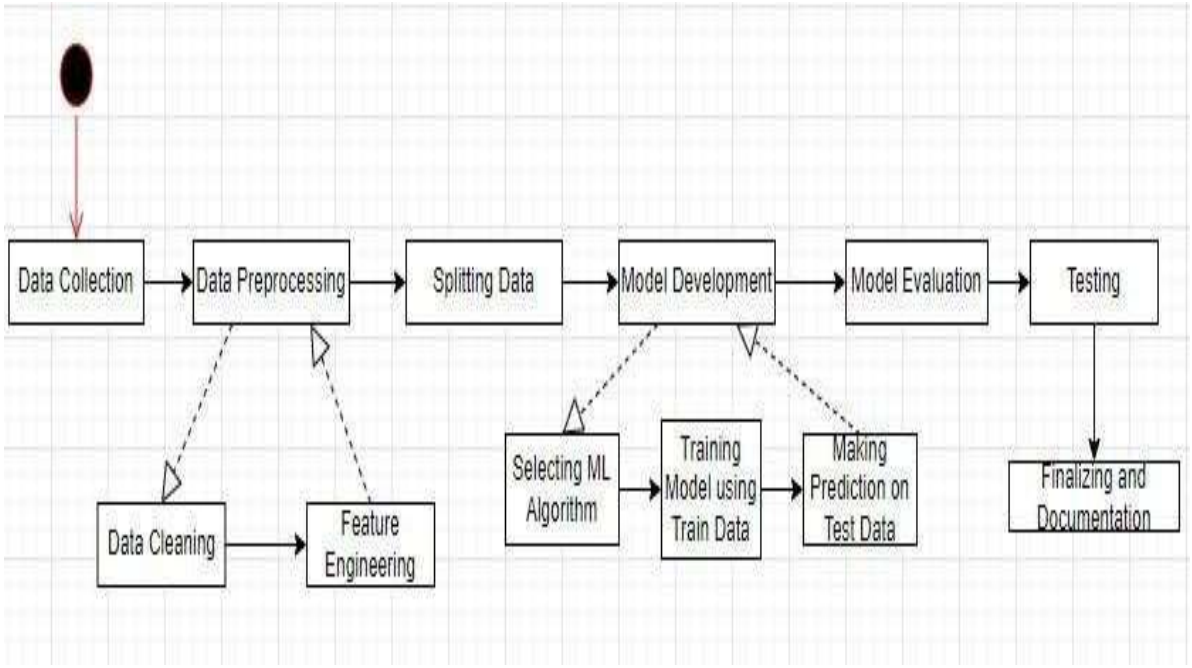
time of day, the weather, or holidays, can be identified with the use of this interpretability.

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.

Overall, the decision tree algorithm offers a flexible and understandable method for predicting traffic patterns, allowing you to obtain knowledge of the underlying variables affecting traffic and anticipate future behaviour using data from the past.

Interfaces (if applicable)



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 urbanization, 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.

To make wise design choices, it is crucial to recognize and comprehend the unique restrictions and requirements of the traffic forecasting problem. Based on the exact restrictions and resources available in your particular scenario, modify the preceding suggestions.

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.

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.

Performance Outcome

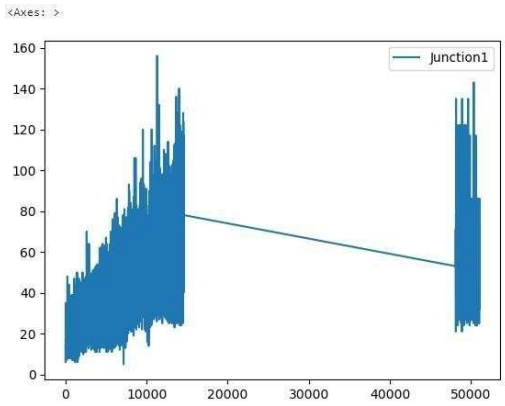
The performance outcome of a Decision Tree algorithm for forecasting traffic patterns can vary depending on several factors, including the quality of the data, the complexity of the traffic patterns, the choice of features, and the tuning of the model parameters.

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

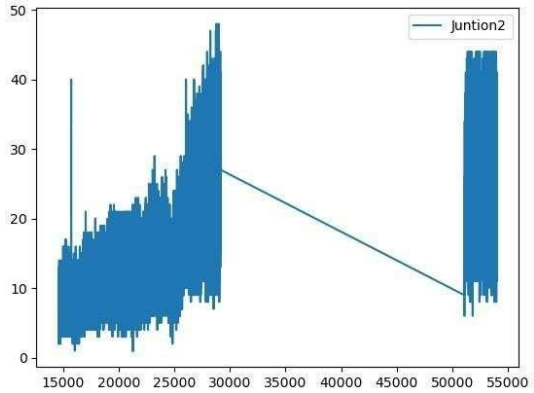
	ID	Vehicles
0	20170701001	53
1	20170701011	71
2	20170701021	37
3	20170701031	39
4	20170701041	31
...	...	...
11803	20171031194	30
11804	20171031204	30
11805	20171031214	16
11806	20171031224	22
11807	20171031234	12

2. Visualizing Prediction of traffic at different Junction:

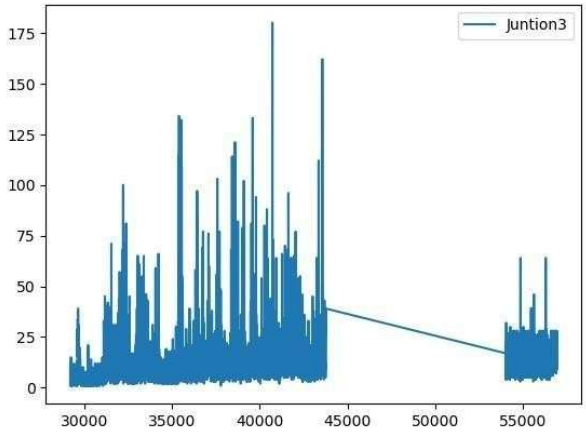
   - Junction 1:



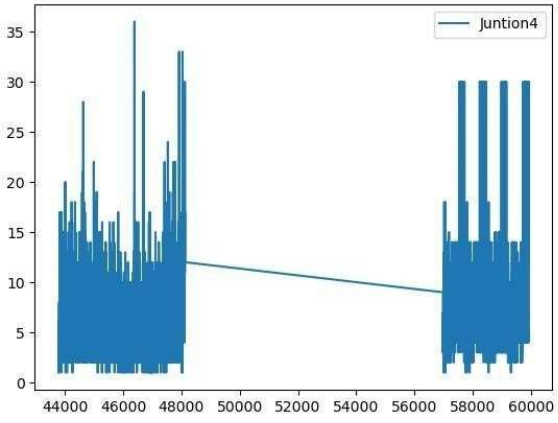
- Junction 2:



- Junction 3:



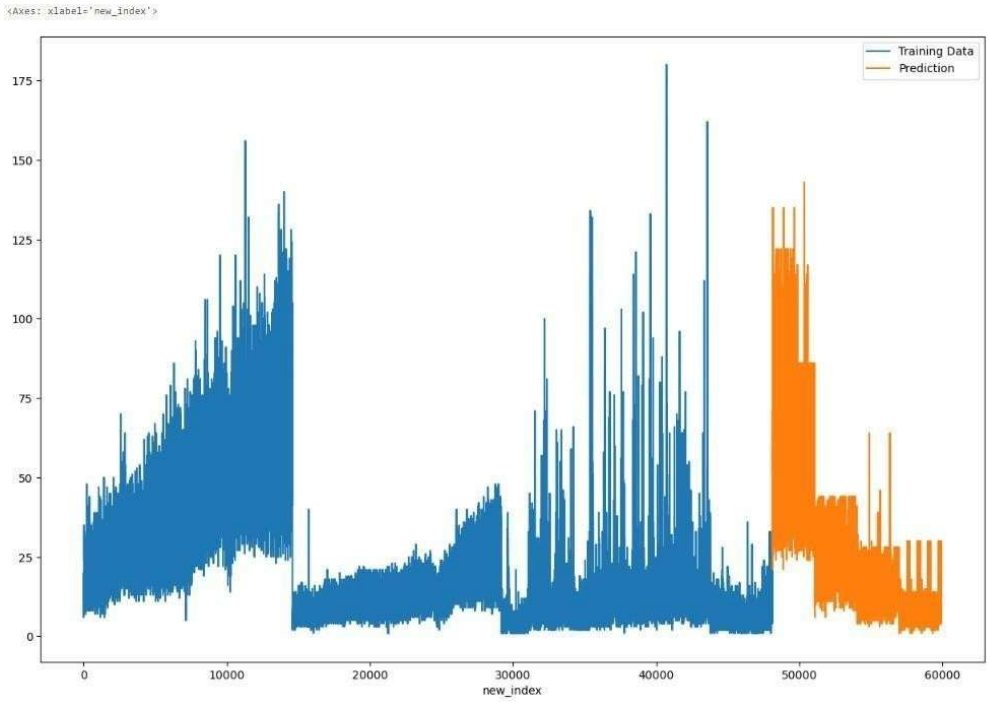
- Junction 4:



Prediction for Junction 1, Junction 2 and Junction 4 were quite accurate as shown in the graphical representation of the prediction.

Whereas prediction for Junction 3 were not as accurate as that of the prediction made for other junctions.

3. Visualizing the Training data and the prediction:



4. Calculation Metrics for the regression Problem:

- Mean absolute error:

```
metrics.mean_absolute_error(y_test, pred)
```

```
7.805694098088113
```

- Mean squared error:

```
metrics.mean_squared_error(y_test, pred)
```

```
166.12572734829592
```

- Root mean squared error:

```
np.sqrt(metrics.mean_squared_error(y_test, pred))
```

```
12.888976970585986
```

My learnings

We learned about the python programming and its various implementation in areas like Artificial Intelligence, Data Science and Machine Learning. Also, we study about the Data Science and Machine learning and get to know about the different algorithms used in this field that can be used for the solution of our project. We get to know the importance of Machine learning in today's world. We get to study about various algorithm that can be implemented in this project. We learned how to apply them and how its implementation can be done, and which will give the best result on implementation. We learned how to apply them and how its implementation can be done, and which will give the best result on implementation. We get to used pandas library in python to read our data and perform data cleaning, then we use scikit-learn library of machine learning to split train and test data and then apply the machine learning algorithm and we use matplotlib and seaborn for graphical representation of data to analyze and understand it.

This 6 Week Internship program with UpSkill Campus & UniConverge Technologies Pvt. Ltd. Was very much helpful. I learned so much about Python, Machine learning, Data Science and get to work on a real-world project. It was a great experience which will help me get ahead in my career in future. Thanks to UpSkill Campus and UCT for giving us this wonderful opportunity.

Future work scope

1. Feature engineering: The performance of decision trees can be improved by processing the input data beforehand and creating useful

characteristics. You could design elements like time of day, day of the week, holiday, and seasonality indicators for traffic predictions. Additionally, take into account including traffic-related elements as traffic jams, collisions, and construction activities. The decision tree algorithm can better anticipate outcomes by capturing complicated relationships through feature engineering.

2. Incorporating real-time data: Decision trees can be trained using historical traffic data, but real-time data, such as traffic flow, weather, and events, may increase predicting accuracy. The decision tree may adjust and create predictions based on the present circumstances by incorporating these variables, producing more precise traffic forecasts.
3. Hierarchical decision trees: Traffic patterns, like daily, weekly, and monthly trends, might show hierarchical systems. You can identify these patterns on several levels by building a hierarchy of decision trees. For instance, while lower-level trees can concentrate on weekly or monthly patterns, the top-level decision tree can forecast daily traffic trends. This strategy can increase forecasting precision across a range of time spans.
4. Spatial dependencies: Road networks, traffic congestion, and transportation infrastructure are examples of spatial elements that frequently affect traffic patterns. Consider using geographic information system (GIS) data to incorporate geographical dependencies into the decision tree algorithm. The decision tree can more accurately depict localized traffic patterns by using spatial variables like road connection, distance to important destinations, and population density.
5. Dynamic updating: Various variables, like urban development, modifications to the road system, or significant events, can cause changes in traffic patterns throughout time. Consider adding a dynamic

updating mechanism to the decision tree model to take these changes into account. Retrain the decision tree on a regular basis with the most recent data to account for changing traffic patterns and guarantee forecast accuracy.

6. Ensemble methods: Accuracy can be raised by using decision tree ensembles like gradient boosting and random forests. Ensemble approaches lessen overfitting and produce more reliable traffic forecasts by training many decision trees and pooling their predictions. Individual decision trees cannot manage noise and oscillations in the data as well as ensemble models, which leads to more accurate predictions.