Industrial Internship Report on

”Forecasting of Smart City traffic patterns”

Prepared by

Rajitha Nair

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 about Forecasting Smart City Traffic Patterns. It focused on leveraging historical data and relevant factors to make accurate predictions about future traffic conditions in a city. The main objectives were to aid in urban planning, enhance traffic management, and optimize resource allocation. To achieve these goals, various machine learning (ML) techniques were explored and applied. One significant aspect of the project involved integrating ML models into the existing traffic management systems. This allowed for real-time adjustments of traffic signals, efficient rerouting of vehicles, and the optimization of overall traffic flow based on the predictions generated by the models. This will not only help drivers save time and fuel but also contributed to reducing traffic congestion and improving the overall efficiency of the city's transportation system.

This internship gave me a very good opportunity to get exposure to Industrial problems and design/implement solution for that. It was an overall great experience to have this internship.

TABLE OF CONTENTS

1 Preface

During my internship, I chose "Forecasting of Smart City traffic patterns" as my project. In the first week, I researched and figured out what was needed for the project. In the second week, I explored various Machine Learning techniques and learned about Python libraries that could help with the project.

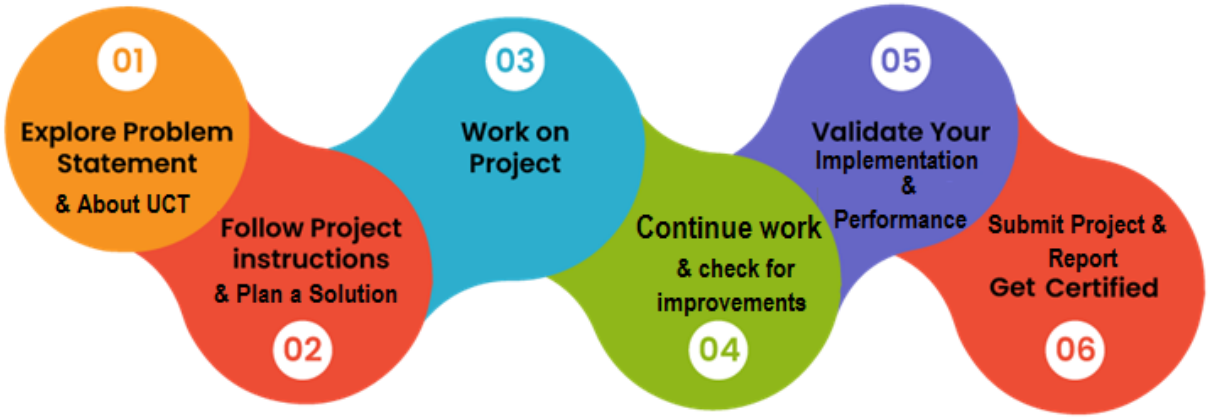
By the third week, I started working on the actual model and set up the necessary libraries. I also looked into other models to see if they could be a better fit. In the fourth and fifth weeks, I ran the model and fixed any errors that came up. I compared the outputs of three different models to see which one performed the best.

In the final week, I focused on improving the model by getting rid of unnecessary elements that could affect its performance. After careful consideration, I identified the best model for the given data.

Being a statistics and data science student, participating in an internship focused on use of machine learning techniques offers numerous advantages. It provides practical experience, enhances skills, and gives exposure to industry tools. Moreover, it creates networking opportunities, allows to tackle real data challenges, and will strengthen my resume.

The problem statement of the project is to develop a predictive model for forecasting smart city traffic patterns. The goal is to utilize historical traffic data and relevant factors to predict future traffic conditions accurately. The main objective is to aid in urban planning, optimize traffic management, and allocate resources effectively. By integrating various machine learning techniques, the project aims to create an efficient model that can anticipate traffic patterns in real-time, leading to reduced congestion, enhanced transportation efficiency, and an improved quality of life for residents and visitors.

Upskill USC/UCT has provided a wonderful opportunity for this internship, allowing me to delve into the practical aspects of Data Science and Machine Learning. Their support and guidance have been invaluable in enhancing my skills and gaining real-world experience. I am grateful for this enriching opportunity that has opened doors to new possibilities in my career journey. How Program was planned



During this internship, I had a truly enriching learning experience that exceeded my expectations. Exploring Python programming in a hands-on setting allowed me to grasp concepts with greater depth and practicality.

As I near the conclusion of my internship, I wanted to take a moment to extend my heartfelt gratitude to Upskill Campus and UCT for the exceptional learning experience they have provided me. I would also like to thank Brutus, Nitish Sharma and Kaushlendra Singh Sir for their mentorship throughout the internship.

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.

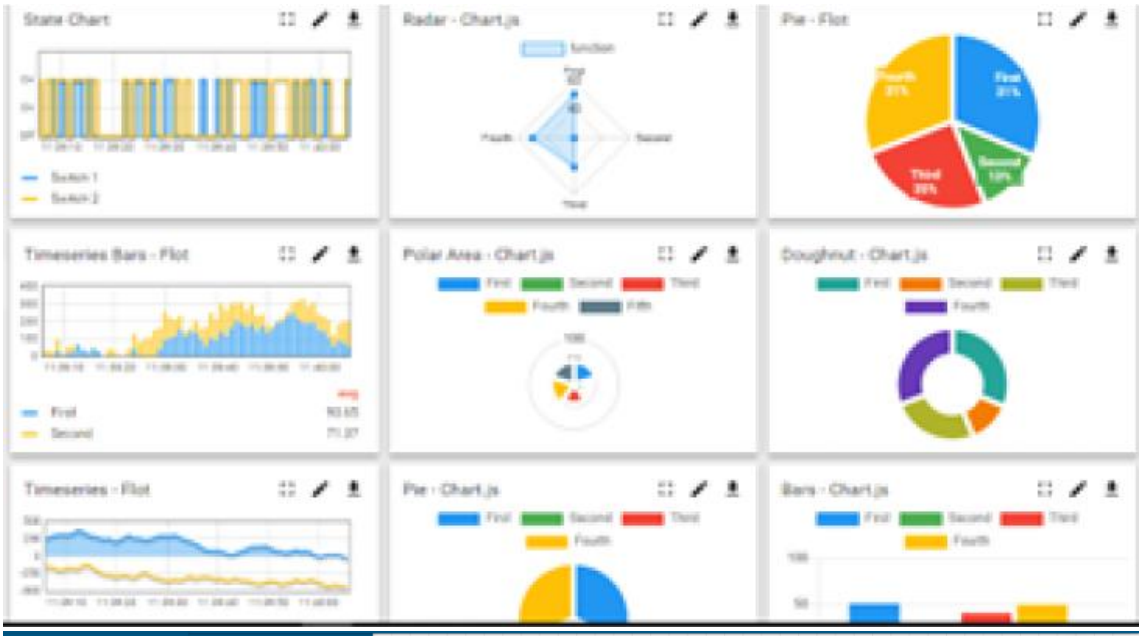
i. UCT IoT Platform (UCT Insight)

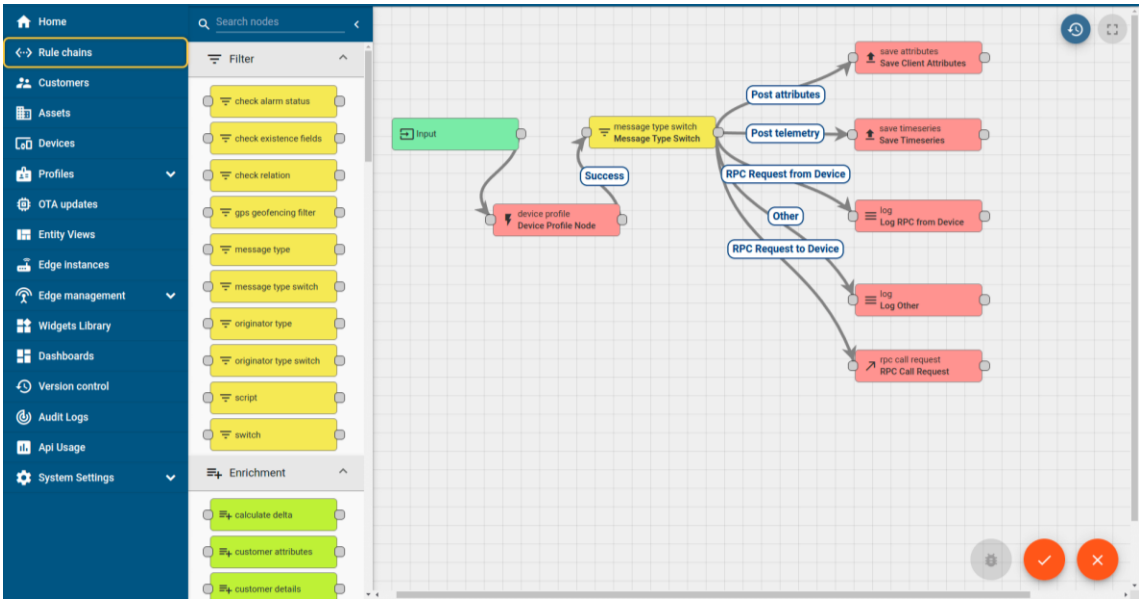
UCT Insight is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable "insight" for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

- 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.

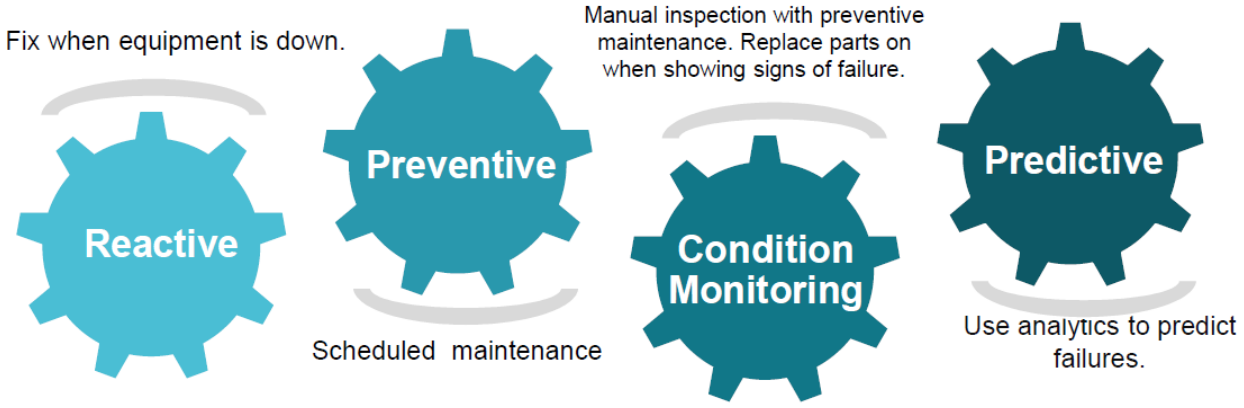
Machine	Operator	Work Order ID	Job ID	Job Performance	Job Progress		Output		Rejection	Time (mins)				Job Status	End Customer
					Start Time	End Time	Planned	Actual		Setup	Pred	Downtime	Idle		
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i
CNC_S7_81	Operator 1	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 LoRAWAN teschnology and providing solution in Agritech, Smart cities, Industrial Monitoring, Smart Street Light, Smart Water/ Gas/ Electricity metering solutions etc.

iv. Predictive Maintenance

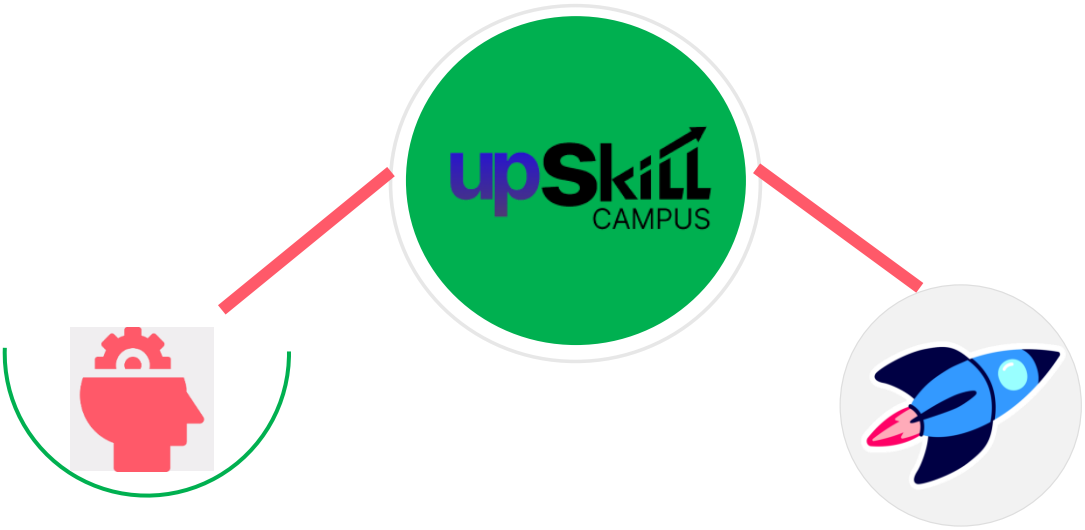
UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.



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.

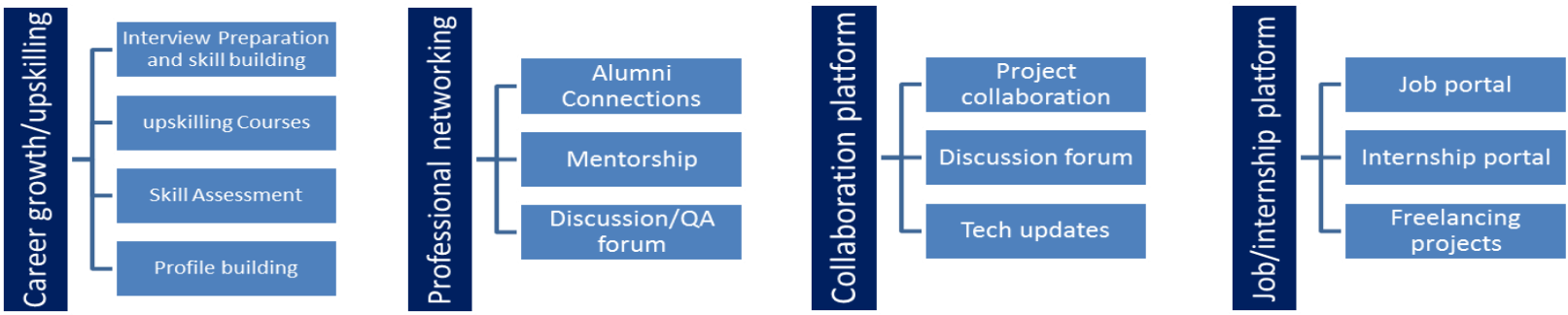
USC is a career development platform that delivers **personalized executive coaching** in a more affordable, scalable and measurable way.



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://link.springer.com/chapter/10.1007/978-3-031-08859-9_10

[2] https://www.researchgate.net/publication/363677388_Smart_City_Traffic_Patterns_Prediction_Using_Machine_Learning

2.6 Glossary

Terms	Acronym
Extreme Gradient Boosting	XGBoost
Support Vector Regression	SVR

3 Problem Statement

The government aims to implement a robust traffic management system for a city, focusing on four major junctions. The goal is to understand the traffic patterns at these junctions to efficiently handle traffic peaks and plan accordingly. The system needs to account for the variation in traffic between normal days and holidays, as traffic patterns differ on these occasions.

To achieve this, the government needs a traffic forecasting solution that can accurately predict the number of vehicles passing through each junction at different times of the day. The forecasts will help the city's traffic management authorities to allocate resources effectively, optimize signal timings, and plan traffic diversions during peak hours.

Key Objectives:

Accurate Traffic Forecasting: Develop a machine learning and data science model that accurately predicts the number of vehicles passing through each junction for different time intervals (e.g., hourly or daily) in the future.

Real-time Adaptability: The traffic forecasting system should be adaptable to real-time data and continuously update its predictions to respond to any sudden changes or unexpected events on the road.

Robustness and Scalability: The solution should be robust and scalable to handle data from multiple junctions and support expansion to more locations in the future.

The success of this project will enable the government to implement a data-driven and efficient traffic management system that reduces traffic congestion, improves commuting experiences for citizens, and ensures the smooth functioning of the city's transportation infrastructure.

4 Existing and Proposed solution

Existing traffic forecasting models: Some existing solutions use traditional time series forecasting models (e.g., ARIMA, SARIMA) to predict traffic patterns based on historical traffic data. However, these models may not effectively capture the complex and dynamic nature of traffic patterns, especially during special events or holidays. Traditional time series models may oversimplify the underlying traffic patterns, missing out on important factors that influence traffic, such as weather, events, or road closures.

We will employ advanced machine learning models like XGBoost, Random Forest and SVR to capture complex traffic patterns and relationships. These models will take into account historical traffic data, weather information, events, and other relevant factors to forecast future traffic.

The hybrid approach will provide accurate traffic predictions, enabling proactive planning and resource allocation for traffic management. The system's ability to integrate real-time data will ensure its relevance and effectiveness in handling dynamic traffic situations. The solution will be designed to accommodate data from multiple junctions, making it scalable for city-wide implementation.

4.1 Code submission (Github link)

https://github.com/Rajitha140/UpSkill-Campus/blob/main/Forecasting%20Smart%20City%20Traffic_RajithaNair_USC_UCT.py

4.2 Report submission (Github link) : first make placeholder, copy the link.

5 Proposed Design/ Model

The proposed design for the Smart City Traffic Forecasting Model aims to accurately predict traffic patterns at the city's junctions while considering variations between normal days and holidays. The model will be designed to handle time series data and make real-time predictions for proactive traffic management. Here's a high-level overview of the proposed design:

Data Preprocessing:

Handle Missing Values: Check for missing values in the dataset and apply appropriate imputation methods, if necessary.

Feature Engineering: Extract relevant features from the datetime information, such as day of the week, hour of the day, month, and year. Create binary features to indicate whether the day is a holiday or not.

Train-Test Split:

Split the preprocessed data into a training set and a test set. The training set will be used to train the traffic forecasting model, and the test set will be used for evaluation.

Model Selection:

3 machine learning models have been considered: XGBoost, Random Forest and Support Vector Regression (SVR).

Both models have been trained on the training data to predict the number of vehicles passing through each junction.

Evaluate each model's performance on the training set using appropriate evaluation metrics like Mean Absolute Error (MAE

Model Deployment:

Deploy the trained models and the forecasting system.

City authorities can access the predictions and insights easily through a user-friendly interface.

5.1 XGBOOST MODEL

XGBoost (Extreme Gradient Boosting) is a popular and powerful machine learning algorithm designed for supervised learning tasks, especially for regression and classification problems. It is an advanced implementation of the gradient boosting framework that leverages the strengths of boosting algorithms to create highly accurate and efficient predictive models. XGBoost has gained popularity in data science

and machine learning competitions due to its outstanding performance and ability to handle complex and large-scale datasets effectively.

Summary of XGBoost Implementation:

Imported the XGBRegressor class from the xgboost library to use XGBoost for regression.

Initialized an instance of XGBRegressor as model_xgb.

Trained the XGBoost model using the fit() method with the training data (X_train and y_train).

Made predictions on the test data (X_test) using the predict() method and stored the results in test_predictions_xgb.

Calculated the Mean Square Error (MSE) between the predicted and actual number of vehicles.

The implementation utilizes XGBoost as one of the machine learning models to forecast smart city traffic and evaluates its performance using the MSE metric.

5.2 RANDOM FOREST REGRESSOR MODEL

Random Forest is a popular machine learning algorithm used for both regression and classification tasks. Random Forest is an ensemble learning method that combines multiple decision trees to create a more accurate and robust predictive model. It belongs to the family of bagging algorithms, which aims to reduce overfitting and improve generalization.

Key Features of Random Forest:

Ensemble Learning: Random Forest builds a collection of decision trees, each trained on a random subset of the data and features. The final prediction is made by aggregating the predictions of individual trees (classification: majority vote, regression: averaging).

Random Subset Sampling: During the construction of each tree, Random Forest randomly selects a subset of the training data (bootstrapping) and a random subset of features. This introduces diversity among the trees and reduces the risk of overfitting.

Decision Tree Building: Each decision tree in the Random Forest is built by recursively partitioning the data into subsets based on different feature splits, resulting in a tree-like structure.

Feature Importance: Random Forest provides a measure of feature importance based on the average impurity reduction (Gini or entropy) caused by each feature across all trees. This feature importance metric helps in feature selection and understanding the model's behavior.

Summary of Random Forest Regressor Implementation:

Imported the RandomForestRegressor class from the sklearn.ensemble module to use Random Forest for regression.

Initialized an instance of RandomForestRegressor as model_rf.

Trained the Random Forest model using the fit() method with the training data (X_train and y_train).

Made predictions on the test data (X_test) using the predict() method and stored the results in test_predictions_rf.

Calculated the Mean Square Error between the predicted and actual number of vehicles.

The implementation utilizes Random Forest as one of the machine learning models to forecast smart city traffic and evaluates its performance using the MSE metric.

5.3 SUPPORT VECTOR REGRESSION

SVR stands for "Support Vector Regression." It is a variant of Support Vector Machines (SVM) used for regression tasks. While SVM is primarily used for classification problems, SVR extends the concept to handle continuous or numeric target variables, making it suitable for predicting real-valued outcomes.

Key Features of Support Vector Regression (SVR):

Kernel Trick: SVR uses the kernel trick, similar to SVM, to map the input features into a higher-dimensional space. This allows SVR to handle nonlinear relationships between the features and the target variable.

Support Vectors: In SVR, only a subset of the training data points known as support vectors are used to define the regression model. These support vectors lie closest to the decision boundary, and their distances play a crucial role in determining the regression function.

Epsilon-Tube and Loss Function: SVR introduces an epsilon-tube around the regression line. Data points within this tube are considered to be well-predicted, while points outside the tube incur a penalty based on a loss function. The loss function aims to minimize the error between the predicted and actual target values.

Regularization Parameter: SVR includes a regularization parameter (C) that controls the trade-off between fitting the training data and minimizing the model's complexity. Higher values of C emphasize

fitting the data, potentially leading to overfitting, while lower values increase the model's robustness but might result in underfitting.

Import the necessary libraries, including SVR from sklearn.svm, StandardScaler from sklearn.preprocessing, and mean_squared_error from sklearn.metrics.

Create an instance of SVR with default hyperparameters using SVR().

Scale the training and test data using StandardScaler.

Train the SVR model on the scaled training data using the fit() method.

Make predictions on the scaled test set using the predict() method.

Calculate the Mean Square Error (MSE) to evaluate the model's performance.

6 Performance Test

Used MSE to validate the models.

Root Mean Squared Error (RMSE): MSE calculates the average of squared differences between the predicted and actual values. It penalizes larger prediction errors more heavily than MAE.

6.1 Test Plan/ Test Cases

Test Plan for Smart City Traffic Forecasting Project:

Objective: The test plan aims to verify the accuracy and performance of the machine learning models (XGBoost, Random Forest and SVR) implemented for smart city traffic forecasting.

Test Cases:

Data Preprocessing:

Test Case 1: Verify if missing values are handled properly by the preprocessing steps.

Test Case 2: Ensure that the datetime feature is correctly split into year, month, day, and hour components.

Model Training and Prediction:

Test Case 3: Check if the XGBoost and Random Forest model is trained without errors.

Test Case 4: Verify that the both model makes predictions on the test set without errors.

Test Case 5: Ensure that the SVR model is trained without errors.

Test Case 6: Verify that the SVR model makes predictions on the test set without errors.

Model Performance Evaluation:

Test Case 7: Calculate MSE for all 3 models and compare the results.

6.2 Test Procedure

Data Preparation:

Load the dataset containing datetime, junction, ID, and vehicles features.

Verify that the dataset is correctly loaded, and check for any missing values.

Data Preprocessing:

Apply data preprocessing steps, such as handling missing values and extracting datetime components (year, month, day, hour).

Perform one-hot encoding for categorical features (Junction and ID).

Split the dataset into training and test sets.

Model Training and Prediction:

Train the XGBoost model on the training data using the XGBRegressor class.

Train the Random Forest model on the training data using the RandomForestRegressor class

Train the SVR model on the training data using the SVR class from scikit-learn.

Make predictions on the test set using both models.

Performance Evaluation:

Calculate the Mean Square Error (MSE) for all the models.

6.3 Performance Outcome

XGBoost (MSE = 25.5731965): The XGBoost model has a lower MSE compared to the other two models (Random Forest and SVR). A lower MSE indicates that the XGBoost model's predictions are closer to the actual values, resulting in better performance.

Random Forest (MSE = 16.786898992): The Random Forest model has the lowest MSE among the three models. This suggests that the Random Forest model's predictions are the closest to the actual values, making it the best-performing model based on MSE.

SVR (MSE = 79.1994868): The SVR model has the highest MSE among the three models, indicating that its predictions have higher errors compared to the actual values.

Based on the MSE values alone, **the Random Forest model appears to be the best performer**, as it has the lowest MSE.

7 My learnings

The project showcases the application of advanced machine learning techniques like XGBoost, Random Forest, and Support Vector Regression in solving real-world problems. The project's focus on forecasting traffic patterns in a smart city highlights its relevance and potential impact on urban planning and transportation management. By accurately predicting traffic trends, the project offers invaluable insights for designing efficient traffic systems, optimizing resources, and enhancing overall urban mobility. Working on this project involves implementing and comparing different machine learning algorithms such as XGBoost, Random Forest, and Support Vector Regression. Understanding the strengths and weaknesses of these models enhances the data scientist's knowledge and expertise in choosing the most appropriate algorithms for various tasks.

8 Future work scope

Hyperparameter Optimization: Perform an extensive hyperparameter tuning process using techniques like grid search or random search to fine-tune the model parameters. This can improve the models' performance by finding the optimal combination of hyperparameters.
Interactive Dashboard: Create an interactive web-based dashboard that allows users to visualize traffic forecasts, explore historical trends, and adjust input parameters to get personalized traffic predictions.