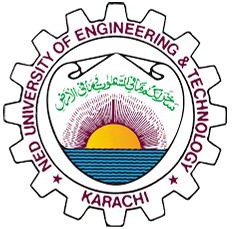
NED University of Engineering & Technology

****

**Machine Learning (CT-354)**

**Group Members:**

Anas Ahmed (CTAI-22013)

Faisal Shahid (CTAI-22011)

Sajeel Tariq (CTAI-22029)

Faraz Ahmed (CTAI-22028)

# 1. Introduction

Urban traffic congestion continues to be a major challenge, leading to increased travel time, fuel consumption, and environmental pollution. Traditional traffic monitoring systems often rely on static cameras and human analysis, which are inefficient in handling large-scale real-time traffic data. These limitations lead to delays in identifying congestion hotspots, predicting traffic build-ups, and responding to potential roadblocks. There is an urgent need for an AI-powered system that can monitor traffic conditions, predict congestion, and provide meaningful insights to city authorities and commuters.

This project's main objective was to design and implement an AI-Powered Smart Traffic Monitoring and Congestion Prediction System. The system utilizes machine learning and real-time data analytics to provide detailed traffic insights. It integrates live traffic feeds from the TomTom API, weather conditions from the Open-Meteo API, road incident reports (also via TomTom), and leverages a substantial historical dataset from Kaggle for training its predictive models. The goal was to develop a system capable of fulfilling Course Learning Outcome 2 (CLO 2 (C3)) by identifying appropriate machine learning techniques to solve classification and prediction problems of moderate complexity related to traffic management.

The scope includes the development of a conceptual model requiring innovative analysis (Complex Computing Problem Attribute CP 2) and the application of in-depth computing and domain knowledge (CP 3).

# 2. Methodology

The project followed a structured methodology encompassing the following key phases:

Problem Definition and Requirement Analysis: Clearly understanding the urban traffic congestion problem as stated and the system requirements outlined in the CCA.

Data Sourcing:

Identifying and integrating with specific APIs for real-time data collection: Open-Meteo for weather, TomTom for live traffic and incidents, and OpenRouteService for route coordinate data.

Acquiring and evaluating a large-scale historical dataset ("US Traffic Congestions 2016-2022" from Kaggle) for training the congestion prediction model.

Data Preprocessing: Cleaning, transforming, feature engineering, and preparing both the real-time API data and the historical Kaggle dataset to make them suitable for machine learning model training and system input.

Model Selection and Training: Selecting and training appropriate machine learning models. Specifically, an LGBM Classifier was trained on the Kaggle dataset for congestion prediction. Other models were explored for real-time detection. Development and experimentation were primarily conducted using Jupyter Notebooks.

System Development (Backend): Implementing the core logic in Python, including algorithms for data ingestion from APIs, real-time processing, applying the trained LGBM model for predictions, alert generation, and alternate route recommendations using OpenRouteService.

Frontend Development (Interface): Designing and developing a user-friendly map-based interface using React to visualize live traffic conditions, congestion levels, incidents, and potential disruptions.

Testing and Evaluation: Rigorously testing individual modules, the trained models (e.g., evaluating the LGBM classifier's accuracy), and the integrated system to ensure functionality and performance.

Documentation: Compiling this comprehensive technical report detailing the project's objectives, design, implementation, results, and challenges.

# 3. Dataset Collection and Preprocessing

The system utilizes both real-time data from APIs and historical data for model training.

## 3.1. Real-time Data Sources (APIs)

Live Traffic Congestion and Incidents: Real-time traffic flow, speed, density, and information on accidents or road construction were obtained using the TomTom API. This API served as the primary source for dynamic traffic events.

Weather Conditions: Current and forecasted weather data (e.g., rain, fog, temperature), crucial for understanding traffic pattern deviations, were sourced from the Open-Meteo API.

Route Coordinate Data: To facilitate alternate route planning, lists of latitudes and longitudes defining paths between two points were retrieved using the OpenRouteService API.

## 3.2. Historical Data for Prediction Model Training (Kaggle)

For training the congestion prediction model, the "US Traffic Congestions (2016-2022)" dataset by Sobhan Moosavi, available on Kaggle, was utilized.

Dataset Source: <https://www.kaggle.com/datasets/sobhanmoosavi/us-traffic-congestions-2016-2022/data>

Description: This dataset contains extensive records of traffic events across the US from 2016 to 2022. Key features likely include 'ID', 'Severity', 'Start\_Lat', 'Start\_Lng', 'StartTime', 'EndTime', 'Distance(mi)', 'DelayFromTypicalTraffic(mins)', 'DelayFromFreeFlowSpeed(mins)', 'Congestion\_Speed', 'Description', 'Street', 'City', 'County', 'State', 'Country', 'ZipCode', 'LocalTimeZone', 'WeatherStation\_AirportCode', 'WeatherTimeStamp', 'Temperature(F)', 'WindChill(F)', 'Humidity(%)', 'Pressure(in)', 'Visibility(mi)', 'WindDir', 'WindSpeed(mph)', 'Precipitation(in)', 'Weather\_Event', 'Weather\_Conditions'

This rich historical data is invaluable for identifying patterns and training a robust prediction model.

## 3.3. Data Preprocessing Steps

The raw data, from both APIs and the Kaggle dataset, underwent several crucial preprocessing steps within our Python environment, often initially explored in Jupyter Notebooks:

Data Cleaning: Addressing missing values (e.g., through imputation strategies like mean imputation or removal of records/features with excessive missingness), and correcting inconsistencies.

Feature Engineering: Creating new relevant features from the raw data. For the Kaggle dataset, this included is\_weekend and is\_rushhour. It showed noticeable correlation with the target column. For API data, examples include categorizing weather conditions from Open-Meteo.

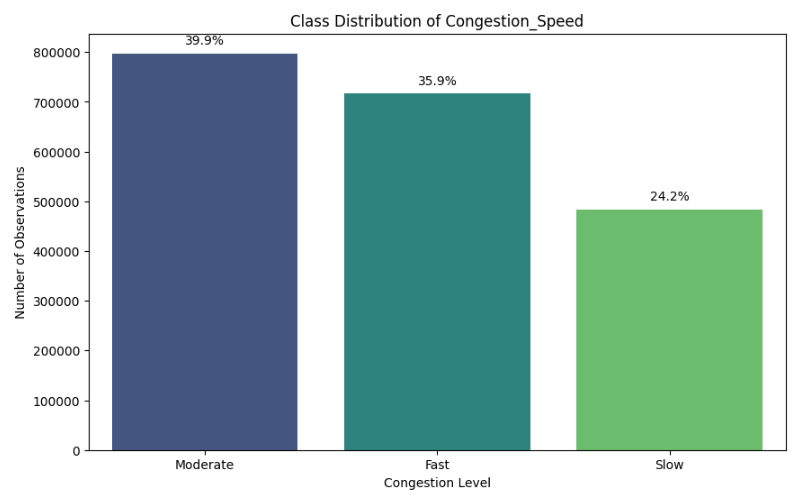
Data Transformation & Scaling: Standardizing data formats from different sources. Applying techniques standardization to numerical features to ensure they contribute appropriately during model training.

Handling Imbalanced Data: LightGBM makes sure the imbalanced data is not a pre-processing step.

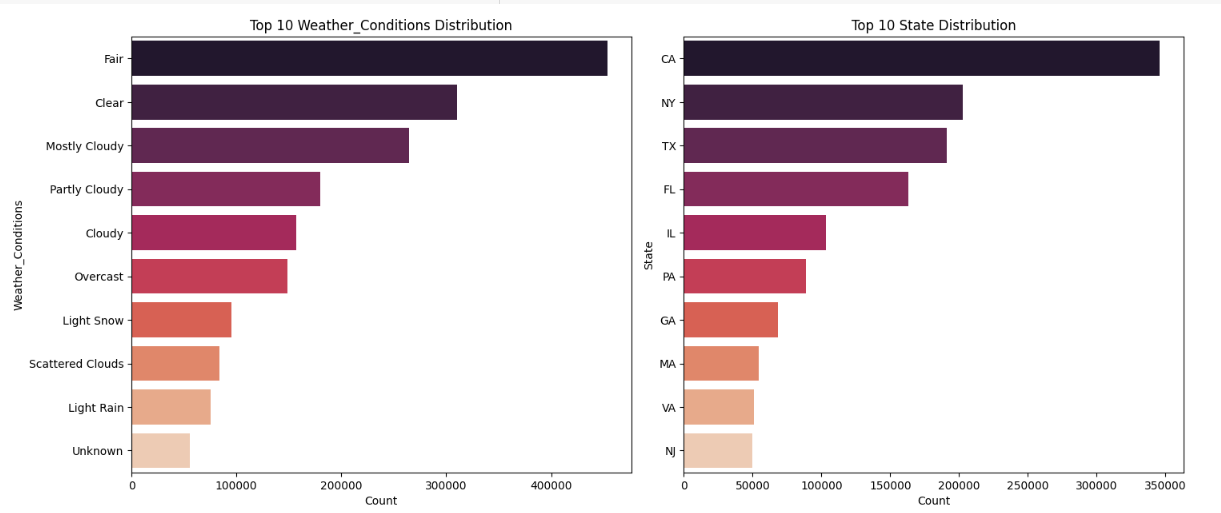
## 3.4. Exploratory Data Analysis

We performed EDA of Univariate Analysis, Bivariate Analysis, and Multivariate Analysis.

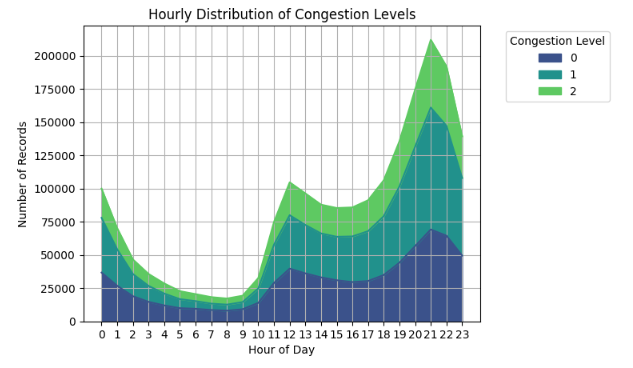
**3.4.1 Univariate Analysis**

****

* **Moderate congestion** is most common (39.9%), followed by **Fast** (35.9%) and **Slow** (24.2%).
* Majority of traffic is in **moderate to fast** conditions (~76%).
* **Severe congestion (slow)** is relatively less frequent.

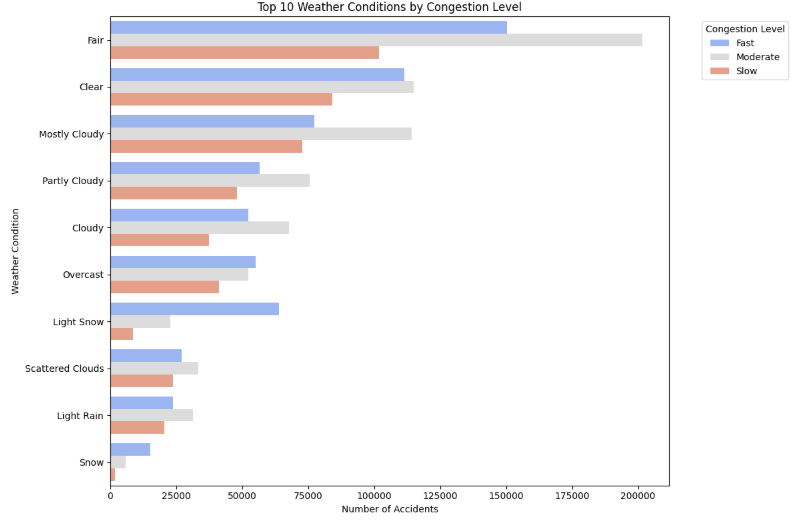


* **Weather Conditions:** Most incidents occur under **Fair, Clear, and Mostly Cloudy** conditions. Less frequent but notable conditions include **Light Snow, Scattered Clouds, and Light Rain**.
* **States:** **California, New York, and Texas** have the highest incident counts, followed by **Florida, Illinois, and Pennsylvania**.



* **Peak congestion** occurs around **8–9 AM** and **5–8 PM**, with the **evening peak being higher**.
* **Slow traffic** significantly increases in the evening, indicating heavier congestion.
* **Early morning (12–5 AM)** has minimal congestion.
* **Midday (10 AM–3 PM)** shows moderate and stable traffic.
* **Fast traffic** is present throughout the day, more during off-peak hours.

**3.4.2 Bivariate Analysis**



**Dominant Weather Patterns**

Fair and Clear conditions account for ~75% of all observations, suggesting these are the most common driving conditions in the dataset.

**Congestion Distribution**

Fast traffic appears most frequently during Clear weather (peaking at ~175K occurrences)

Moderate congestion dominates in adverse conditions (Light Snow, Light Rain, Snow)

Slow traffic shows consistent distribution across all weather types

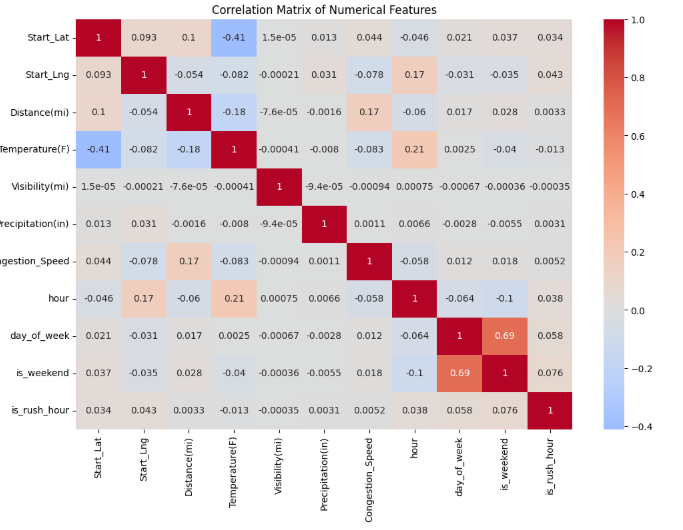
**Weather Impact**

Precipitation (Snow/Rain) correlates with reduced Fast traffic occurrences

Cloudy conditions show balanced distribution across all congestion levels

Extreme conditions (Snow) have 3x more Moderate than Fast traffic

**3.4.3 Multivariate Analysis**



* **Positive Correlations:** is\_weekend and day\_of\_week (0.69).
* **Negative Correlations:** Start\_Lat and Temperature(F) (-0.41).
* **Moderate Correlations:** hour and Temperature(F) (0.21), Distance(mi) and Congestion\_Speed (0.17).

# 4. Machine Learning Models Used

.

## 4.1. Congestion Prediction (LGBM Classifier)

To predict future traffic congestion (e.g., severity levels), a LightGBM (LGBM) Decision Tree Classifier was employed.

Training Data: The model was trained on the preprocessed "US Traffic Congestions (2016-2022)" Kaggle dataset.

Reason for Choice: LGBM was chosen due to its efficiency and high performance with large datasets, its gradient boosting framework which often yields excellent results, its ability to handle categorical features well, and its relatively faster training times compared to other gradient boosting models.

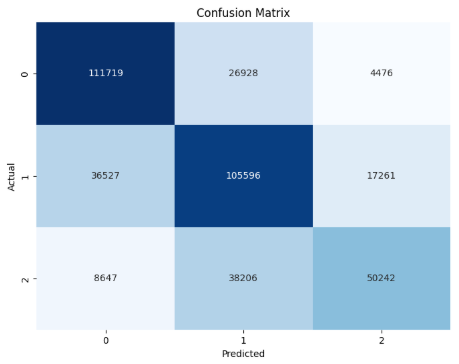
## 4.2. Alternate Route Recommendation Logic

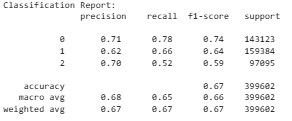
While not a standalone ML model, the recommendation logic utilized outputs from the LGBM congestion prediction model, real-time TomTom traffic data, weather data from Open-Meteo, and route geometries from OpenRouteService API to rank and suggest routes.

## 4.5. Model Training and Evaluation Overview

LGBM Classifier Training: The LGBM model was trained using 80/20 train-test split on the Kaggle dataset.

Evaluation Metrics: The primary metric for the LGBM Classifier was accuracy. Other metrics such as precision, recall, and F1-score for different congestion classes were also considered. Confusion Matrix is also showed.





# 5. System Design and Implementation

The system comprises a Python-based backend for data processing and machine learning, and a React-based frontend for user interaction and visualization.

## 5.1. Python Code Implementation & Core Functionalities

The backend, developed using Python and extensively utilizing Jupyter Notebooks for initial development, data exploration, model training (especially the LGBM model), and experimentation, performs the following:

API Interaction Layer: Modules to robustly connect to and fetch data from Open-Meteo, TomTom, and OpenRouteService APIs, including error handling and data parsing.

Data Processing Engine: Scripts for preprocessing and transforming incoming API data and for preparing the Kaggle dataset for training and prediction.

Machine Learning Core:

Implementation of the trained LGBM Classifier to make predictions on future congestion based on relevant input features.

Implementation of models/logic for real-time congestion detection.

Alerting Mechanism: Logic to trigger alerts when predefined congestion thresholds are met (based on real-time data or predictions) or significant incidents are reported by TomTom.

Alternate Route Generation: A module that:

Takes origin and destination from the user (via the React frontend).

Fetches potential route geometries from OpenRouteService.

Evaluates these routes based on current traffic (from TomTom), predicted congestion (from the LGBM model), and weather conditions (from Open-Meteo).

Recommends the most optimal alternate route(s).

## 5.2. Visualization: User-Friendly Map-Based Interface (React)

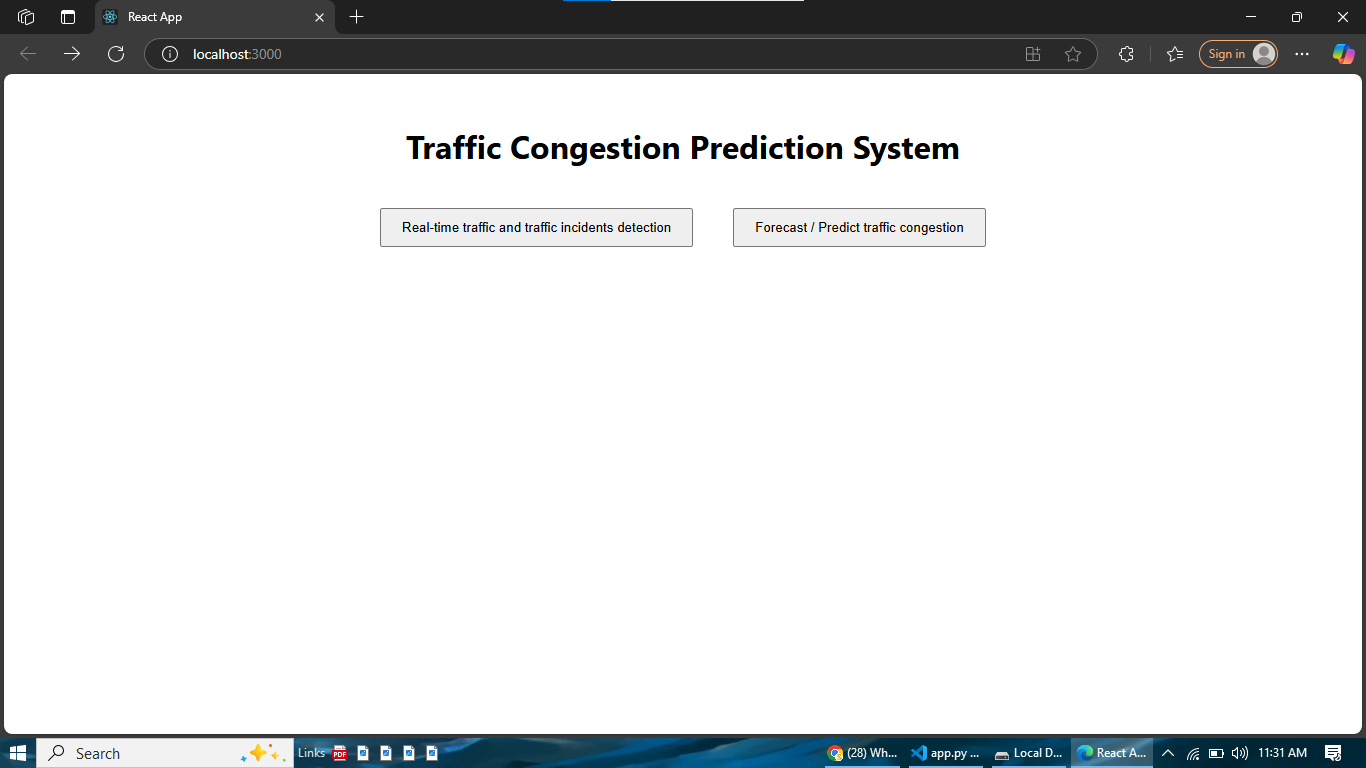
To enhance user interaction and ensure seamless communication between users and the AI system, a React-based frontend interface was developed. This web interface allows users to easily visualize real-time traffic data and view future congestion predictions.

#### Home Page (Home.jsx)

This is the main navigation page of the application, offering users two key options:

* View real-time traffic and incident data.
* Forecast traffic congestion and get alternate route suggestions.

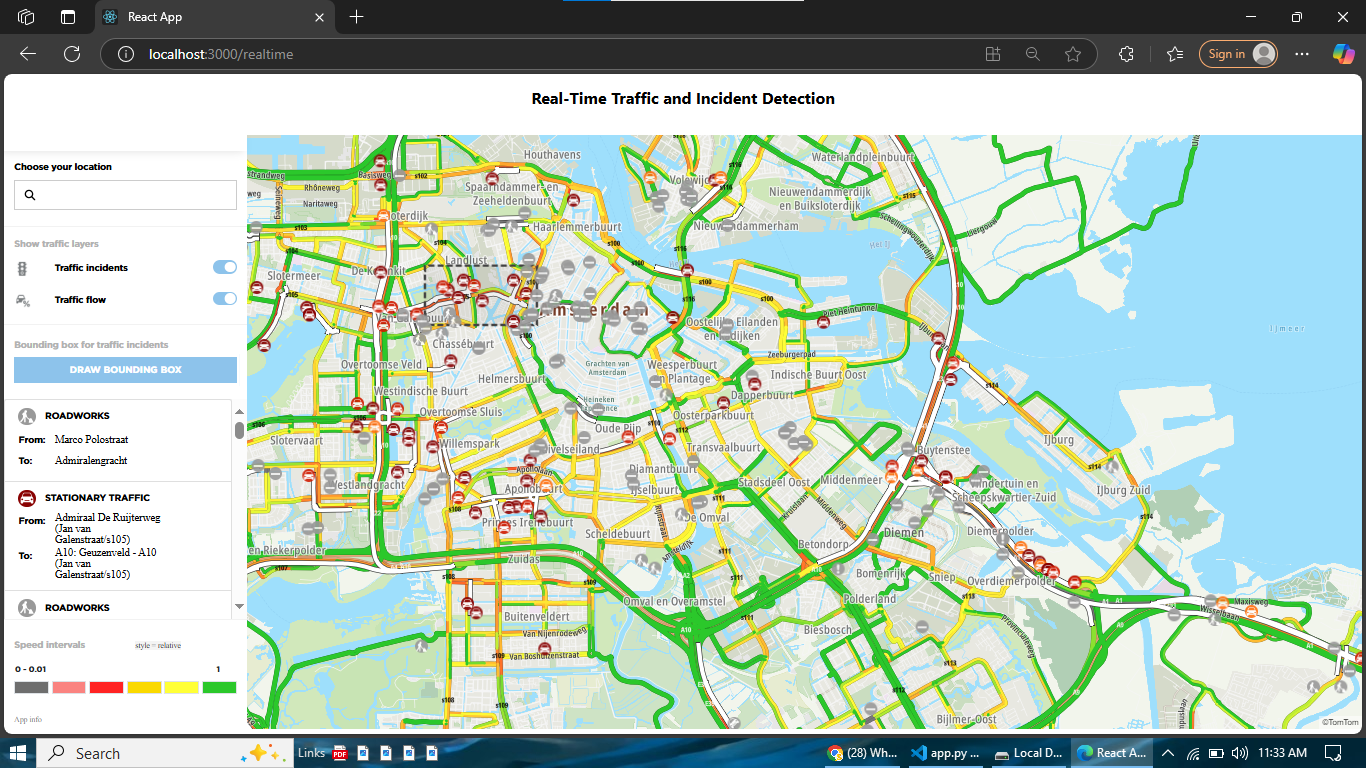




#### ****Real-Time Traffic Viewer (RealTime.jsx)****

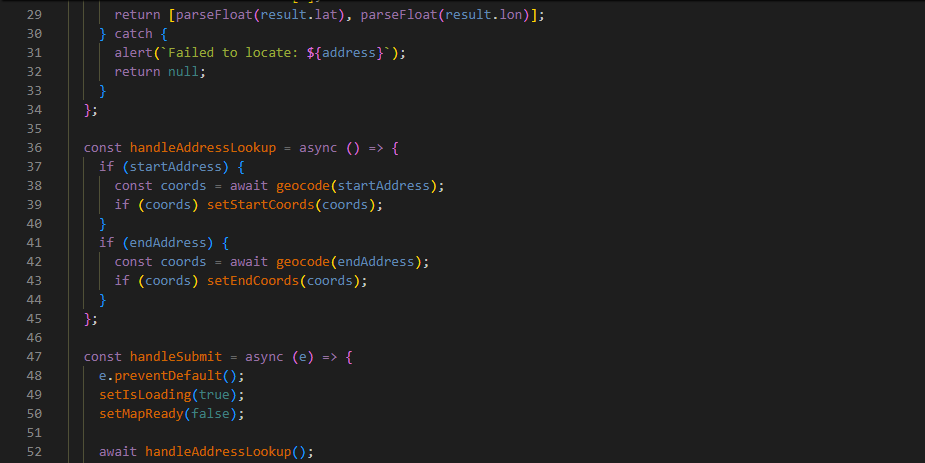
This page embeds a custom HTML page (traffic\_page/traffic.html) using an iframe. The HTML file is generated with the help of Python and Leaflet.js, and displays real-time traffic data including traffic incidents, congestion zones, and road closures using the TomTom API.

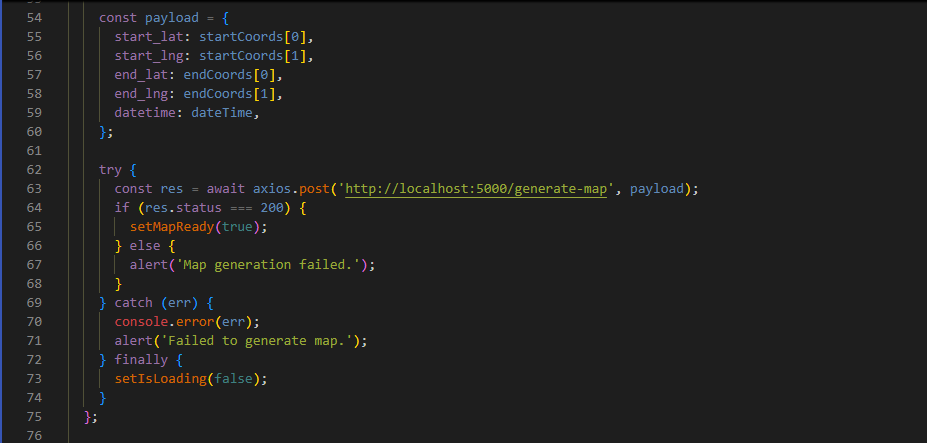


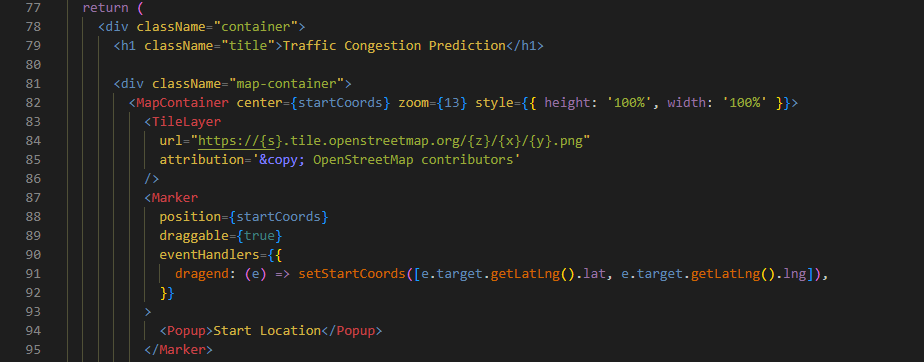


#### Congestion Forecast Interface (Forecast.jsx)

This page (assumed name: Forecast.jsx) interfaces with the backend API to allow users to input or select coordinates and receive congestion predictions powered by the trained LightGBM model. This also includes visual overlays on a Leaflet map to indicate the congestion level.

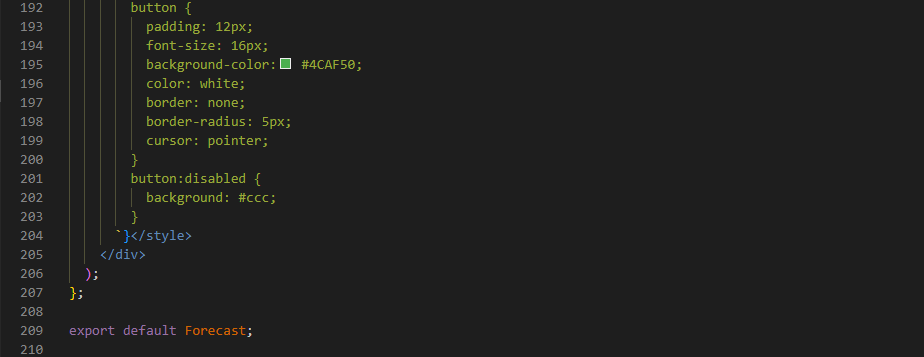
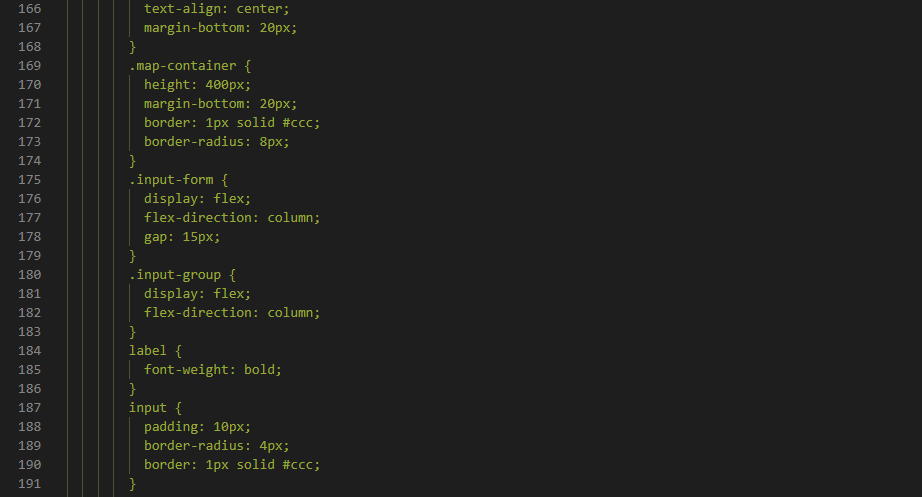
  


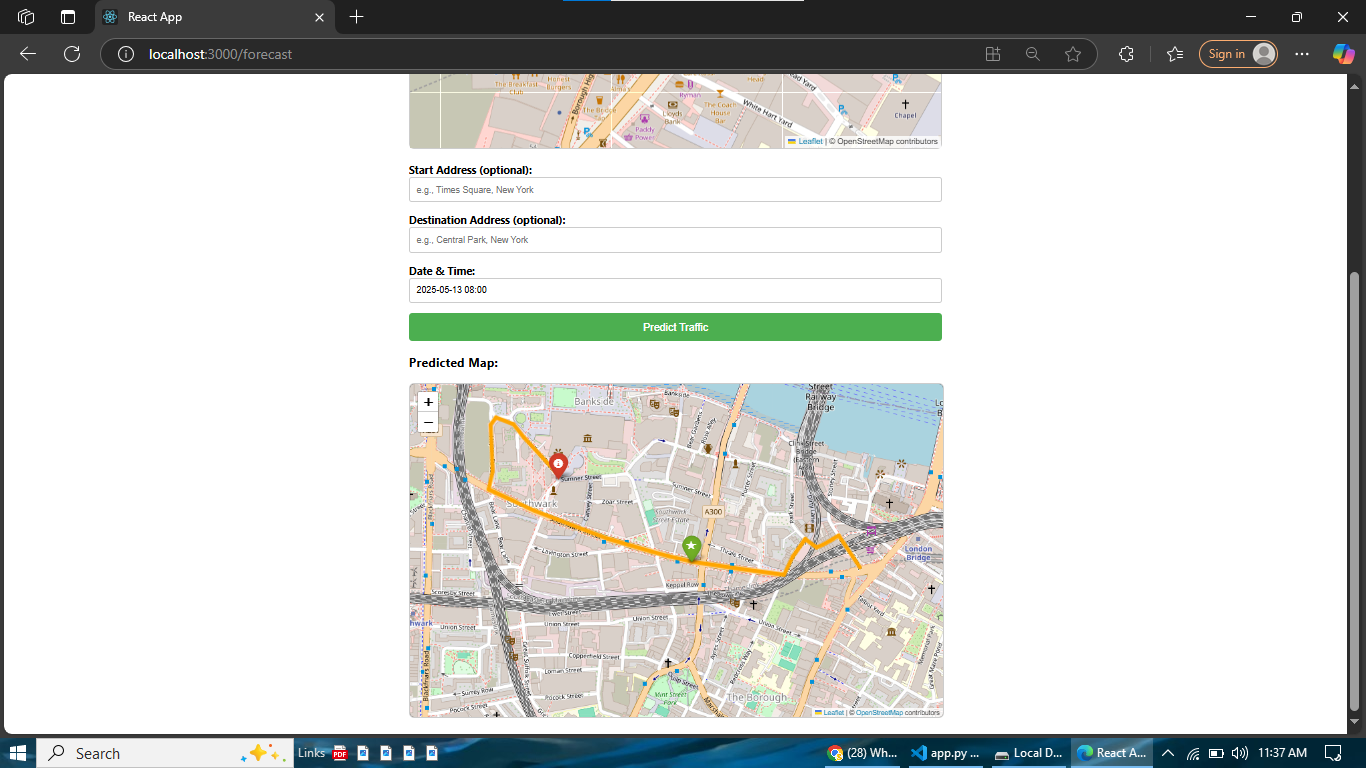
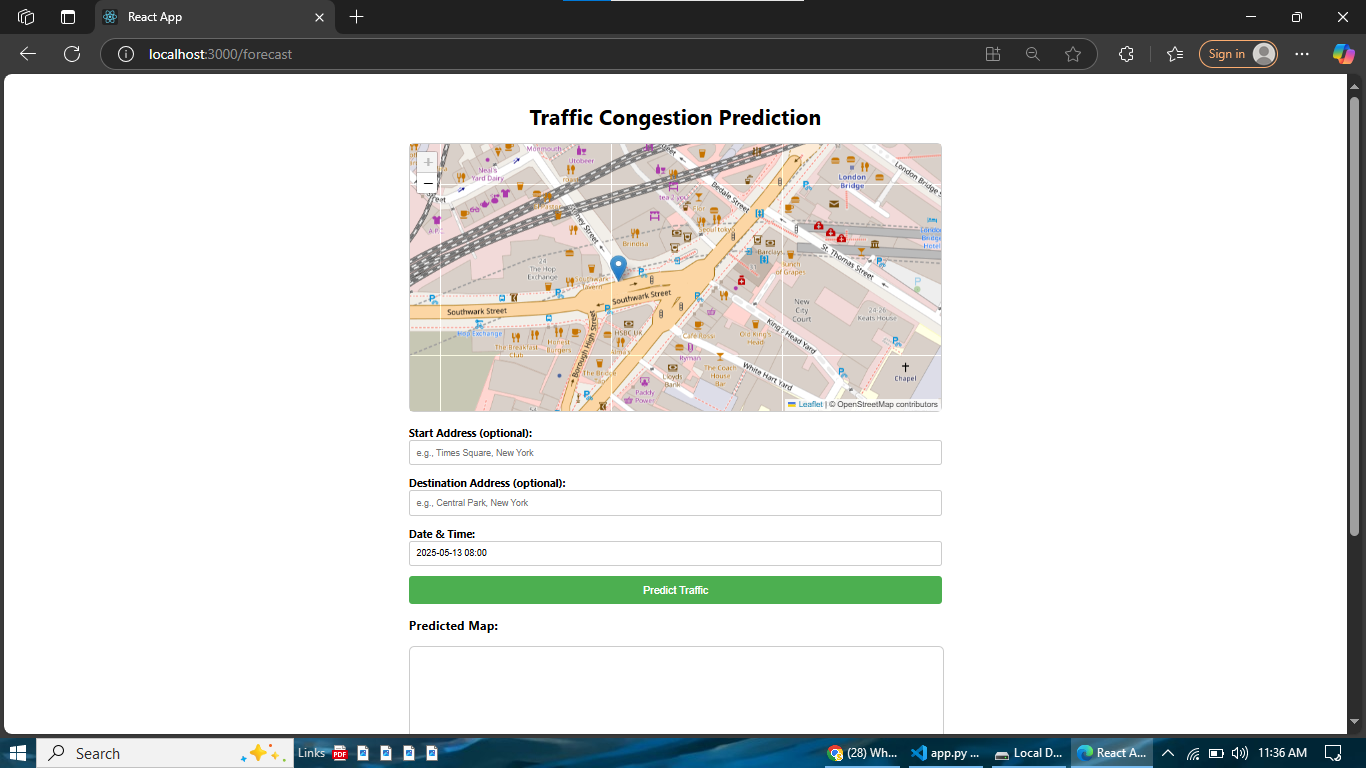










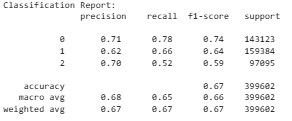


# 6. Results and Discussion

## 6.1. Model Performance

Congestion Prediction (LGBM Classifier):

The LGBM Classifier, trained on the "US Traffic Congestions (2016-2022)" Kaggle dataset, achieved an accuracy of 0.67 on the held-out test set.



## 6.2. System Functionality

The system successfully integrated data from Open-Meteo, TomTom, and OpenRouteService APIs. The LGBM model provided predictions based on historical patterns, and the React interface delivered effective visualization and interaction capabilities for users. Alerts were generated as per defined conditions.

## 6.3. Discussion of Insights

The accuracy of 0.67 for the LGBM Classifier in predicting traffic congestion indicates a model that has learned significant patterns from the historical Kaggle data. While this level of accuracy can provide useful indications, especially for a complex multi-class problem, there is potential for further improvement. Factors influencing this accuracy could include the inherent randomness in traffic events, the specific features engineered, and the complexity of real-world traffic dynamics that are hard to capture even in large datasets.

# 7. Challenges Faced During Development

During the project, several challenges were encountered:

API Integration and Rate Limits: Managing requests and data consistency across Open-Meteo, TomTom, and OpenRouteService APIs, each with its own structure and potential usage limitations.

Kaggle Dataset Preprocessing: Cleaning and feature engineering the large "US Traffic Congestions" dataset was a significant task, requiring careful handling of missing values and derivation of meaningful features for the LGBM model.

Model Tuning and Performance: Achieving higher accuracy with the LGBM model required iterative tuning of hyperparameters and feature selection. Balancing model complexity with performance was key.

Real-time Processing: Designing the system to process incoming API data and run predictions with minimal latency.

Computational Resources: Training the LGBM model on the large Kaggle dataset required adequate computational resources.

# 8. Ethical Considerations and Limitations

The development and deployment of this system considered the following:

Data Privacy: The system adheres to data privacy regulations. TomTom and OpenRouteService provide anonymized or aggregated data. The Kaggle dataset, while public, was used responsibly. No personally identifiable information beyond what is publicly available in these datasets was sought or stored. Vehicle license plates and specific personal location data are anonymized as per the CCA.

Accuracy and Reliability: While the LGBM model achieved an accuracy of 0.67, predictions are probabilistic. Users should be aware that recommendations are based on model outputs and available data, but unforeseen events can alter conditions.

Bias in Data: The Kaggle dataset is US-centric, so models trained on it might not generalize perfectly to other regions without retraining or transfer learning. Efforts were made during preprocessing to ensure fairness where possible.

Adherence to Local Laws: System recommendations must align with local traffic laws.

Limitations:

API Dependency: Functionality relies on the continued availability and terms of Open-Meteo, TomTom, and OpenRouteService APIs.

Data Quality: Accuracy is linked to the quality of API data and the representativeness of the Kaggle dataset.

Model Generalizability: The LGBM model, trained on US data, might require adjustments or retraining for optimal performance in different geographical contexts (e.g., Karachi traffic patterns).

Network Availability: Real-time features depend on stable internet connectivity.

Scope of Prediction: The system predicts common patterns; highly unusual events may not be accurately forecasted.

Accuracy Level: While 0.67 accuracy is a good starting point, for critical decisions, higher accuracy might be desired, necessitating further model refinement.

# 9. Conclusion and Future Work

Conclusion: This project successfully designed and implemented an AI-Powered Smart Traffic Monitoring and Congestion Prediction System. It leverages real-time data from Open-Meteo, TomTom, and OpenRouteService APIs, and a predictive model (LGBM Classifier trained on a substantial Kaggle dataset with 0.67 accuracy) to offer insights into traffic conditions. The React-based user interface provides an effective means of visualization and interaction. The project addressed the core requirements of the CCA, demonstrating the application of machine learning (developed via Jupyter Notebooks) to a significant urban challenge.

Future Work: To further enhance the system, the following avenues could be explored:

Improving Prediction Accuracy: Experimenting with more advanced models (e.g., ensemble methods, deep learning architectures like LSTMs or Transformers tailored for sequential/tabular data), more extensive feature engineering on the Kaggle data, or incorporating more localized datasets.

Real-time Model Adaptation: Investigating techniques for the LGBM model to adapt or be retrained more frequently with incoming live data to capture evolving traffic patterns.

Hybrid Modeling: Combining the strengths of the historically trained LGBM model with more agile real-time anomaly detection or short-term forecasting models.

User Feedback Integration: Allowing users to provide feedback on the accuracy of predictions or the quality of route recommendations to create a learning loop.

Explainable AI (XAI): Integrating techniques to provide explanations for why the LGBM model makes certain predictions, increasing trust and utility.

Contextualization for Local Environment: Fine-tuning or retraining models specifically for local traffic conditions (e.g., Karachi) if deploying in such an environment.

Enhanced Scalability and Deployment: Optimizing the system for robust cloud deployment.