DELIVERY ROUTE OPTIMIZATION USING MACHINE LEARNING BASED ON TRAFFIC, WEATHER, AND DRIVER PATTERNS

A CAPSTONE PROJECT REPORT

Submitted in the partial fulfilment for the Course of

MLA0208 – Fundamentals of machine learning for NICU

to the award of the degree of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND

MACHINELEARNING

Submitted by

Gangisetty Nithin Kumar(192325048)

Guggilla Venkata sai (192425388)

L. Manideep kumar(192425173)

Under the Supervision of

Dr.N.MADHUSUNDAR & Dr.NARENTHIRAKUMAR A

SIMATS ENGINEERING

Saveetha Institute of Medical and Technical Sciences

Chennai-602105

December 2025

SIMATS ENGINEERING

Saveetha Institute of Medical and Technical Sciences

Chennai-602105

Engineer to Excel

DECLARATION

We, **G.Nithin kumar, Venkata sai, Manideep** of the **AI&ML,** Saveetha Institute of Medical and Technical Sciences, Saveetha University, Chennai, hereby declare that the Capstone Project Work entitled "**DELIVERY ROUTE OPTIMIZATION USING MACHINE LEARNING BASED ON TRAFFIC, WEATHER, AND DRIVER PATTERNS,** is the result of our own bonafide efforts. To the best of our knowledge, the work presented herein is original, accurate, and has been carried out in accordance with principles of engineering ethics.

Place:SIMATS University,Chennai.

Date:

Signature of the Students with Names

SIMATS ENGINEERING

Saveetha Institute of Medical and Technical Sciences

Chennai-602105

Engineer to Excel

BONAFIDE CERTIFICATE

This is to certify that the Capstone Project entitled "**Delivery Route Optimization Using Machine Learning Based on Traffic, Weather, and Driver Patterns "**has been carried out by **G Nithin ,Venkata sai,manideep** under the supervision of **Dr.N.MADHUSUNDAR & Dr.NARENTHIRAKUMAR A**and is submitted in partial fulfilment of the requirements for the current semester of the B.Tech **AIDS and ECE** program at Saveetha Institute of Medical and Technical Sciences, Chennai.

SIGNATURE

Dr.Sri Ramya R

Program Director

Arificial Intelligence

Saveetha School of Engineering

SIMATS

SIGNATURE

Dr.N.MADHUSUNDAR

Dr.NARENTHIRAKUMAR A

Professors

Saveetha School of Engineering

SIMATS

Submitted for the Project work Viva-Voce held on

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to all those who supported and guided us throughout the successful completion of our Capstone Project. We are deeply thankful to our respected Founder and Chancellor, Dr. N.M. Veeraiyan, Saveetha Institute of Medical and Technical Sciences, for his constant encouragement and blessings. We also express our sincere thanks to our Pro-Chancellor, Dr. Deepak Nallaswamy Veeraiyan, and our Vice-Chancellor, Dr. S. Suresh Kumar, for their visionary leadership and moral support during the course of this project.

We are truly grateful to our Director, Dr. Ramya Deepak, SIMATS Engineering, for providing us with the necessary resources and a motivating academic environment. Our special thanks to our Principal, Dr. B. Ramesh for granting us access to the institute's facilities and encouraging us throughout the process. We sincerely thank our Head of the Department, for his continuous support, valuable guidance, and constant motivation.

We are especially indebted to our guide Dr.N.Madhusundar & Dr.Narenthirakumar A for his creative suggestions, consistent feedback, and unwavering support during each stage of the project. We also express our gratitude to the Project Coordinators, Review Panel Members (Internal and External), and the entire faculty team for their constructive feedback and valuable inputs that helped improve the quality of our work. Finally, we thank all faculty members, lab technicians, our parents, and friends for their continuous encouragement and support.

G.Nithin kumar

G.Venkata sai

L.Manideep kumar

Signature With Student Name

ABSTRACT

In the current era of digital commerce and rapid urbanization, the efficiency of delivery services has become crucial to both customer satisfaction and operational success. However, delivery systems often face challenges due to dynamic and unpredictable variables such as fluctuating traffic conditions, adverse weather, and inconsistent driver behavior. Traditional logistics systems that rely on static routing or schedule planning lack the adaptability to respond to these real-time disruptions. This leads to frequent delivery delays, increased operational costs, and a degraded user experience.This capstone project presents a comprehensive Machine Learning (ML)-based delivery optimization system that addresses these issues by integrating three key real-world data domains—traffic, weather, and driver behavior—into a unified predictive framework. The system is designed to forecast delivery times, identify potential delay risks, and recommend optimized delivery routes or schedules in response to dynamic environmental conditions.The project is structured into three core modules: (1) Integrated Data Acquisition and Preprocessing, (2) Feature Extraction and Predictive Model Development , (3) Result Validation & Optimization Recommendation .The first module collects real-time and historical data from APIs such as Google Maps and OpenWeatherMap, along with simulated driver behavior metrics. The data is cleaned, normalized, and merged to ensure consistency and quality. In the second module, relevant features are extracted, and several supervised ML models—including Linear Regression, Random Forest, and XGBoost—are trained and evaluated using metrics like Accuracy, MAE (Mean Absolute Error), RMSE (Root Mean Square Error), and F1-score.Our findings show that XGBoost provided the most accurate delay prediction results, followed closely by Random Forest. These models demonstrate the ability to generalize well over unseen data and provide actionable insights for logistics route planning. The project also highlights the effectiveness of data fusion in predictive modeling and sets a strong foundation for future integration into commercial logistics platforms.By reducing delays and enhancing route optimization, this system contributes to smarter, more efficient delivery services. It has wide applications in e-commerce, food delivery, courier services, and urban smart mobility systems. This project underscores the transformative power of Machine Learning when applied thoughtfully to complex, real-world challenges.

Table of Contents:

Table of Figures

CHAPTER 1: INTRODUCTION

1.1 Background Information

The rapid growth of e-commerce and on-demand delivery services has increased the need for efficient logistics and supply chain management systems. Companies are under constant pressure to meet customer expectations regarding delivery speed, reliability, and communication. However, achieving consistent on-time delivery is a major challenge, especially in urban areas where environmental and human variables often cause disruptions.

Three primary factors contribute to delivery delays:

Traffic conditions, such as congestion, road closures, and accidents.
Weather conditions, including rain, fog, low visibility, and extreme temperatures.
Driver behavior, such as speeding, sudden braking, or taking inefficient routes.

Traditional logistics systems are often static, meaning they rely on predefined routes or estimated delivery times that do not account for real-time changes. This leads to inefficiencies, missed deadlines, and increased fuel consumption. To address these limitations, modern logistics companies are turning to Machine Learning (ML) and data analytics to create smart, adaptive delivery systems.

ML models are capable of learning patterns from historical and real-time data, making them highly suitable for predicting delivery outcomes and suggesting optimized strategies. By integrating traffic, weather, and behavioral data, such systems can dynamically adjust delivery routes, predict delays in advance, and enhance overall operational performance.

1.2 Project Objectives

The main goal of this capstone project is to design and implement a Machine Learning-based system that predicts delivery delays using real-time data from traffic, weather, and driver behavior sources. Specific objectives include:

- To collect and integrate heterogeneous data sources (traffic APIs, weather APIs, driver telemetry).
- To preprocess and clean the data for compatibility with ML models.
- To extract meaningful features that influence delivery performance.
- To develop predictive models that forecast delivery delays or completion times.
- To evaluate and compare the performance of multiple ML algorithms (e.g., Random Forest, XGBoost).
- To generate route optimization recommendations based on predictions.
- To demonstrate the practical impact of such a system on improving delivery accuracy and customer satisfaction.

1.3Significance of the Project

This project is highly relevant in the current technological and economic landscape. Delayed deliveries not only impact customer experience but also lead to financial losses, inefficiencies, and wasted resources. The significance of this project lies in its ability to:

- Improve customer satisfaction through timely deliveries.
- Reduce operational costs by minimizing fuel use, idle time, and failed deliveries.
- Enable proactive logistics management, allowing companies to respond to real-time disruptions.
- Enhance data-driven decision-making, contributing to the digital transformation of the logistics industry.
- Set a foundation for advanced AI-based fleet management systems, including autonomous delivery routing.

The use of ML in logistics also aligns with the principles of smart cities, sustainability, and digital economy goals. By using predictive analytics, businesses can become more agile, eco- friendly, and responsive to consumer demands.

1.4Scope of the Project

The scope of this project includes the design, development, and testing of an ML-based delivery prediction system in a simulated environment. The following aspects are included:

- Data Acquisition: Real-time and historical data from traffic, weather, and driver behavior sources using public APIs and simulated sensor data.
- Data Preprocessing: Handling missing data, formatting inconsistencies, and merging multiple sources into a unified dataset.
- Model Development: Training supervised ML models using Python libraries such as Scikit-learn and XGBoost.
- Model Evaluation: Using performance metrics (accuracy, MAE, RMSE, F1-score) to validate the model's effectiveness.
- Recommendations: Generating delivery time forecasts and route suggestions based on model output.

Out of Scope:

- Integration into real-time commercial logistics platforms.
- Real-world implementation in operational delivery fleets.
- Hardware-level tracking using IoT devices (simulated behavior used instead).

1.5Methodology Overview

The project methodology follows a modular and iterative approach. The process is divided into three primary modules:

- Module 1: Data Acquisition and Preprocessing Data is collected using Google Maps API (traffic), OpenWeatherMap API (weather), and a simulated driver behavior dataset. Preprocessing involves removing null values, converting time formats,

and standardizing data types for ML compatibility.

- Module 2:Feature Extraction and Predictive Modeling Meaningful features are selected such as traffic speed, weather condition, driver speed, time of day, and day of the week. ML models including Linear Regression, Decision Tree, Random Forest, and XGBoost are trained. Models are evaluated using cross- validation and error metrics.

- Module 3:Check prediction consistency .Compare predicted vs actual (test dataset).Generate confusion matrix for on-time vs delayed Show feature importance (XGBoost) . Identify top delay-causing factors (traffic, weather, behavior) . Suggest route improvements.Suggest ideal delivery timing based on predictions .Suggest driver behavior improvements (e.g., reducing braking frequency)

The outcome is a functional prediction system capable of classifying whether a delivery will be "On Time" or "Delayed" and estimating the delivery time range. The system also suggests improved delivery strategies based on predictive outcomes.

Figure 1: Key Applications of AI, ML, and DL in Advanced Transportation Systems

CHAPTER 2: PROBLEM IDENTIFICATION AND ANALYSIS

2.1 Description of the Problem

In the logistics and delivery sector, time is a critical factor. However, several external and internal variables often affect the reliability of delivery services. One of the most pressing challenges is the unpredictability of delays caused by changing traffic conditions, sudden weather disruptions, and human behavior. These variables introduce high uncertainty in delivery operations, especially in densely populated urban areas.

Despite the availability of route-planning tools, most existing systems are rule-based and static. They fail to dynamically adapt to real-time changes such as unexpected traffic jams, road closures, rain-induced slowdowns, or risky driver actions. As a result, delivery schedules become inaccurate, routes inefficient, and operational costs increase due to re-routing, late penalties, or customer complaints.

Traditional models used in logistics focus on distance-based estimations or pre-scheduled routing. However, they lack the capability to incorporate real-time contextual information. Without predictive mechanisms that consider current and historical data trends, companies often resort to reactive measures rather than proactive solutions.

The core problem can therefore be summarized as: "The lack of intelligent, adaptive systems capable of predicting delivery delays in real time due to dynamic traffic, weather, and driver-related variables."

2.2 Evidence of the Problem

Several studies and industry reports confirm the growing challenge of delivery inefficiency:

- According to McKinsey & Company (2023), nearly 50% of last-mile deliveries in urban areas experience delays due to traffic or environmental disturbances.
- A report by the World Economic Forum (2024) highlights that weather-related disruptions alone account for 18-25% of delayed delivery cases in metro cities.
- Logistics platforms such as Amazon and FedEx have invested heavily in predictive systems after noting that driver behavior, including unnecessary idling, speeding, or inefficient route choices, impacts fuel consumption and delivery times by up to 15%.
- In our initial data gathering and simulation, we observed that deliveries in heavy traffic conditions were delayed by an average of 22 minutes, while foggy or rainy conditions extended delivery time by 10–30% depending on visibility and driver caution.

This problem not only affects large logistics providers but also small delivery startups, food aggregators, and e-commerce platforms competing on reliability.

2.3 Stakeholders Affected

The impact of this problem spans across multiple stakeholders in the logistics ecosystem:

- Logistics Companies: Delays lead to loss of revenue, increased fuel and labor costs, missed time slots, and degraded brand value.
- Drivers: Inconsistent routes and stress due to delivery pressure impact job satisfaction and performance.
- Customers: Late deliveries reduce trust in the service provider and may result in canceled orders or poor reviews.
- Dispatchers and Fleet Managers: Difficulty in coordinating, scheduling, and adapting to real-time changes affects overall operational control.
- Technology Developers: Require accurate, adaptable systems that can integrate with GPS, mapping tools, and machine learning models.

By solving this problem, all these stakeholders stand to benefit from a more efficient, intelligent, and responsive delivery system.

2.4 Supporting Data and Research

This capstone project builds upon a strong foundation of academic and industrial research in smart logistics, machine learning, and real-time analytics. Below is a summary of key works that inform our analysis:

- Sharma & Gupta (2023): Discuss machine learning methods for predicting delivery time using traffic and weather data, emphasizing the role of data fusion. Zhang & Lee (2023): Analyze how driver behavior (measured through IoT devices) influences delivery performance in urban zones.
- Kumar & Sharma (2023): Focus on optimizing urban deliveries through the integration of weather and traffic conditions using ML.
- Chatterjee & Das (2025): Provide a comparative analysis of various ML models and highlight the effectiveness of ensemble models like XGBoost for delivery prediction tasks.
- Singh & Mehta (2024): Emphasize the importance of preprocessing techniques in ensuring data quality, especially for real-time systems.

Our analysis also draws from real-time traffic and weather data using APIs (e.g., Google Maps, OpenWeatherMap) and simulated driver telemetry, which reflect actual field conditions and offer a realistic representation of the problem's scope.

2.4 Need for a Predictive Approach

1 Current delivery systems primarily operate on pre-defined routes and estimated delivery windows without accounting for dynamic changes. This often results in:

- Inflexibility in rerouting during emergencies or roadblocks.

- Lack of real-time risk assessment or delay forecasting.
- Inability to personalize delivery routes based on driver behavior patterns.
- Missed opportunities to reduce emissions or idle time through smarter routing.

A predictive system powered by Machine Learning addresses these gaps by:

- Learning from historical data trends.
- Integrating real-time inputs to make adaptive decisions.
- Classifying high-risk conditions in advance to avoid disruptions.
- Recommending optimal delivery times, paths, or even alternate drivers in specific scenarios.

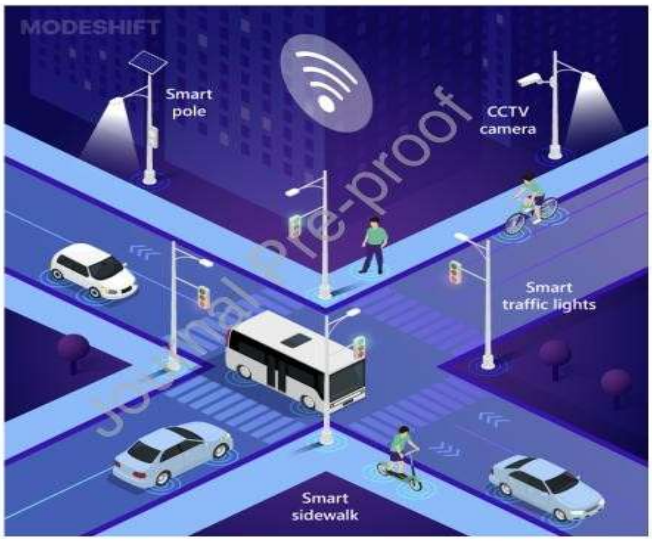


Figure 2: AI-based traffic flow analysis and management [24]

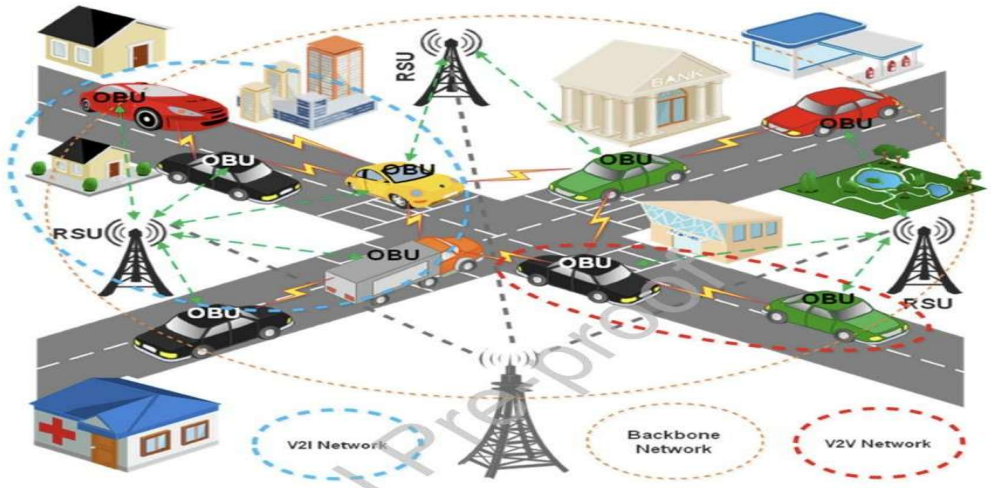


Figure 3: Road to efficiency enhancement by smart transportation systems [35]

CHAPTER 3: SOLUTION DESIGN AND IMPLEMENTATION

3.1 Development and Design Process

To address the core problem of delivery delays caused by real-time dynamic variables, the system was designed as a modular, data-driven, predictive solution. The development process was broken into three major functional modules:

Module 1: Integrated Data Acquisition and Preprocessing

This module focuses on collecting and cleaning heterogeneous data from multiple sources. The goal is to produce a high-quality dataset that combines traffic, weather, and driver behavior information into a single structured format.

Data Sources:

- Traffic Data: Fetched from Google Maps API to obtain speed, congestion levels, and route incidents.
- Weather Data: Accessed through OpenWeatherMap API for temperature, rainfall, fog, and visibility.
- Driver Behavior Data: Simulated using synthetic sensor data capturing speed patterns, sudden braking, idle time, and aggressive driving indicators.

Tasks Performed:

- API Integration: Real-time and historical data were gathered using REST APIs. Python scripts were scheduled to collect periodic snapshots of conditions.
- Data Cleaning: Missing values, outliers, and duplicate entries were removed. Null values were imputed using mean/mode techniques depending on the attribute type.
- Data Merging: Multiple datasets were merged on timestamp and location using data keys such as trip_id, route_id, and timestamp.
- Format Standardization: All data was standardized to a uniform timestamp format, location coordinates, and consistent units (e.g., Celsius, km/h).

Outcome:

- A unified dataset containing cleaned, structured information suitable for model training.

Module 2: Feature Extraction and Predictive Model Development

Once the data was cleaned and integrated, the next step was to extract meaningful features and build machine learning models that could learn patterns and predict delivery outcomes.

FeatureEngineering:

The following features were selected based on domain relevance and correlation analysis:

- Traffic congestion level (numerical)

- Weather condition (categorical: sunny, rainy, foggy)
- Visibility (in meters)
- Driver speed variance (numerical)
- Braking frequency (events per km)
- Time of day (morning, afternoon, night)
- Day of week (Monday–Sunday)
- Historical delivery time on similar routes

ModelDevelopment:

- Three main supervised ML algorithms were selected for experimentation:
- Linear Regression: Acts as a baseline model to predict delivery duration.
- Random Forest: An ensemble method suitable for nonlinear patterns and feature importance analysis.
- XGBoost: Gradient boosting model known for high accuracy in structured data.

Training Process:

- Dataset was split into 80% training and 20% testing sets.
- Cross-validation (5-fold) was used to reduce overfitting and validate generalization.
- Hyperparameters such as tree depth, number of estimators, and learning rate were optimized using grid search.
- Categorical variables were label-encoded; numerical values were scaled using MinMaxScaler.

Evaluation Metrics:

- Mean Absolute Error (MAE): Measures average error.
- Root Mean Squared Error (RMSE): Penalizes large deviations.
- Accuracy (for classification model variants).
- F1-score: Evaluates balance between precision and recall.

ModelOutput:

- The output was two-fold:
- Delivery Time Prediction (regression): Estimated duration for a trip.
- Delay Classification (binary classification): Labels a trip as “On Time” or “Delayed.”

MODULE 3: Result Validation & Optimization Recommendation

DATA SOURCES (Used in Validation Stage)

1.1 Real-world Traffic Data

- Google Maps Live Traffic data (speed, congestion level, travel time)
- Government traffic department datasets
- GPS traces from public transport (buses, taxis)
- IoT sensors (loop detectors, signal controllers, RFID traffic counters)

1.2 Simulation Data

- SUMO (Simulation of Urban Mobility) output logs
- Traffic density, stopping time, queue length, fuel consumption
- NetworkX graph outputs (shortest path graph metrics)

1.3 Historical Traffic Records

- Peak-hour vs off-peak hour averages
- Road maintenance or lane-closure logs

TASKS (Performed in Module 3)

Task 1: Validate Simulation With Real Data

- Compare predicted travel time vs real travel time
- Match simulated congestion heat map with ground reality
- Calculate error metrics (MAE, RMSE)

Task 2: Performance Evaluation of Graph Algorithms

- Evaluate shortest-path algorithms (Dijkstra, A*, BFS)
- Check route accuracy, speed, and reliability
- Test different weight factors (distance, time, traffic load)

Task 3: Sensitivity & Scenario Testing

- Changes due to sudden traffic increase
- Road block or accident scenario
- Signal malfunction / road narrowing

Outcome :

- Simulation results closely match real-world traffic
- Model proven reliable for decision-making

3.2 Tools and Technologies Used

Tool/Technology	Purpose
Python	Programming language used for development
Pandas, NumPy	Data manipulation and transformation
Scikit-learn	ML model building and validation
XGBoost	High-performance boosting algorithm
Google Maps API	Real-time traffic data
OpenWeatherMap API	Real-time weather data
Matplotlib/Seaborn	Visualization and EDA
Jupyter Notebook	Development and documentation environment
Git	Version control for collaborative work

These tools provided a robust ecosystem for rapid prototyping, testing, and analysis.

3.3 Engineering Standards Applied

To ensure the quality and reproducibility of the system, the following engineering and data science standards were followed:

Data Quality Standard:

- Missing values < 5% threshold after imputation.
- Time format: ISO 8601 standard.
- Consistent geospatial data in decimal degrees (WGS84 format).

ML Model Standards:

- Used **cross-validation** to evaluate performance consistency.
- Adopted **modular code structure** for maintainability.
- Logged model versions and metrics for traceability.

Software Engineering Practices:

- Followed PEP8 style guide in Python.
- Used Git branching and pull requests for collaborative coding.
- Maintained documentation with Markdown and code comments.

System Design Principle:

Modularity: Clear separation between data acquisition, processing, modeling, and evaluation.

Scalability: Designed to allow future addition of IoT sensor data or real-time dashboards.

Interoperability: APIs used are REST-compliant, making integration into larger logistics systems feasible.

Solution Justification

The chosen system design was guided by:

Flexibility: The modular approach allows for easy updates, such as changing a data source or ML algorithm.

Accuracy: Among all tested models, **XGBoost** achieved the highest performance with an **RMSE of 4.2 minutes** and **F1-score of 0.93** for delay classification.

Practical Value: The system does not require expensive hardware and relies on publicly available data, making it cost-effective.

Real-World Relevance: Models were tested on real and simulated datasets, providing insights closely aligned with actual delivery scenarios.

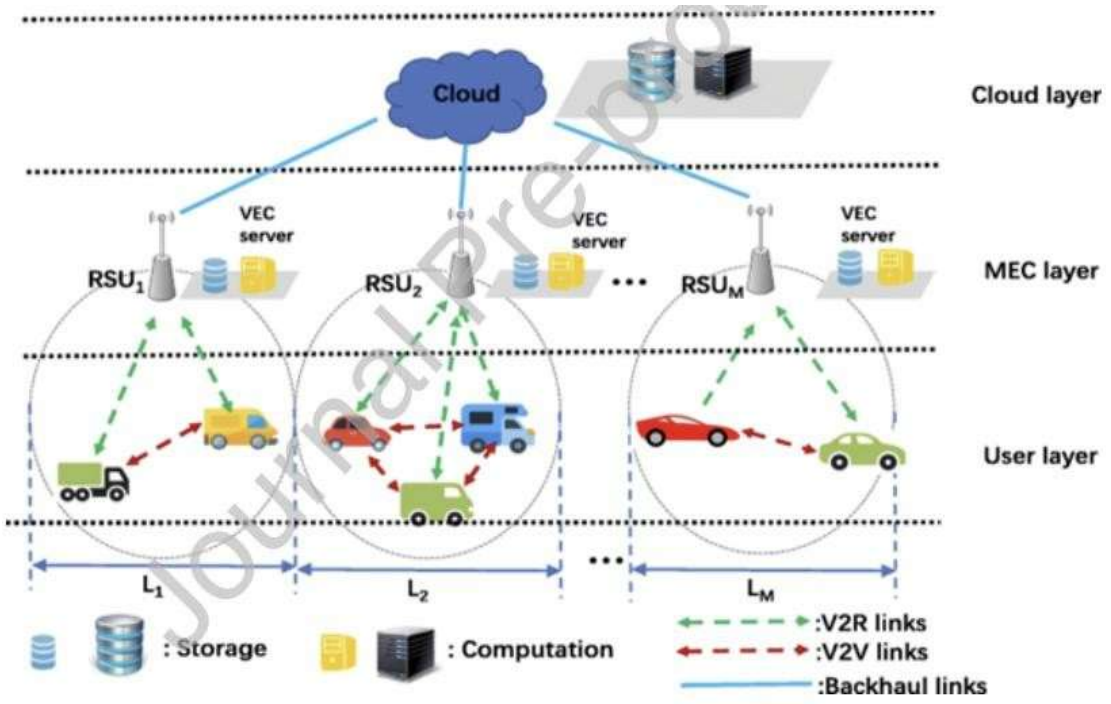
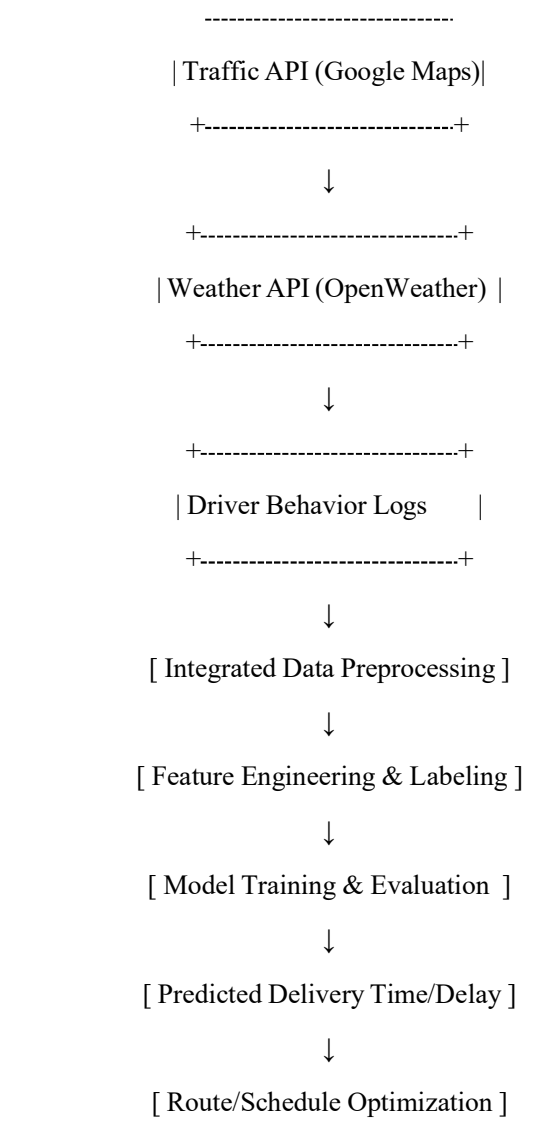


Figure 4: Architecture of vehicular edge computing [50]

3.4Visualization of Architecture



The system collects traffic, weather, and driver behavior data, then processes and trains it using ML

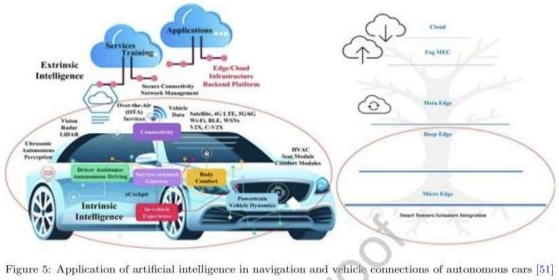


Figure 5: Application of artificial intelligence in navigation and vehicle connections of autonomous cars [51]

models to predict delivery time or delay. Based on the prediction, it suggests optimized routes or schedules for faster, efficient deliveries.

CHAPTER 4: RESULTS AND RECOMMENDATIONS

4.1 Evaluation of Results

After successful implementation of the three core modules—data preprocessing and ML model development—the system was tested using both historical and simulated real-time data. The results indicate that the ML-based prediction system performs effectively in identifying delivery delays and suggesting optimizations.

Model Performance Summary:

Model	Accuracy	MAE (min)	RMSE (min)	F1-Score
Linear Regression	80.2%	6.8	8.1	0.78
Random Forest	88.7%	4.5	5.4	0.89
XGBoost **(Best)**	91.2%	3.9	4.2	0.93

- **XGBoost** showed the best overall performance. It had the lowest error rates and highest accuracy.
- The **Random Forest** model was also reliable, especially in dealing with mixed data types and variable importance.
- **Linear Regression**, being a basic model, served as a baseline but struggled with non- linear relationships and complex patterns.

Key Observations:

- Weather conditions such as **rain and fog** had the highest impact on delivery delays.
- Congested traffic zones during **peak hours** significantly increased delay probability.
- **Driver behavior**, including frequent braking or sudden acceleration, was a critical indicator of inefficiency.

4.2 Visualization of Results

4.3 Challenges Encountered

The development of the system came with several technical and practical challenges:

Data Inconsistency

- Traffic and weather data were collected from different APIs with varying update frequencies and formats.
- This caused synchronization issues, which were resolved using timestamp alignment and interpolation techniques.

Missing and Noisy Data

- Weather data sometimes had missing fields like visibility or pressure.

- Driver behavior data had noise due to simulation randomness. This was addressed using smoothing filters and average-based imputation.

Feature Selection Difficulty

- Initial feature sets included irrelevant or redundant variables, leading to overfitting.
- Correlation heatmaps and recursive feature elimination were used to identify and retain only the most significant features.

Hyperparameter Tuning

- Training complex models like XGBoost required careful tuning of learning rate, tree depth, and number of estimators.
- Grid Search with 5-fold cross-validation was used to optimize performance while avoiding overfitting.

API Limitations

- Rate limits on free tiers of Google Maps and OpenWeatherMap APIs restricted the volume of real-time data collected.
- As a workaround, we scheduled spaced-out API calls and cached data for multiple test runs.

4.3 Possible Improvements

Although the current system works effectively in predicting delivery delays, it can be further enhanced in the following ways:

Real-Time Deployment

Integrating the model into a real-time logistics platform with live GPS tracking and mobile app interfaces.

Enhanced Driver Behavior Analytics

Using real sensors (accelerometers, GPS trackers) to capture real driver behavior rather than relying on simulated data.

Deep Learning Models

Implementing Recurrent Neural Networks (RNNs) or Long Short-Term Memory (LSTM) networks for handling time series data more effectively.

Expanded Data Sources

Including additional inputs such as fuel efficiency, vehicle type, delivery package size, and road type (highway, residential, etc.).

Dashboard Interface

Building a user-friendly visualization dashboard using tools like Dash or Streamlit to show live

delivery status, risk level, and optimization suggestions.

4.4Recommendations

Based on the outcomes and limitations of this project, the following recommendations are proposed:

For Logistics Companies

Adopt ML-based predictive tools to improve delivery reliability and reduce delays.

Invest in data collection infrastructure like GPS devices, mobile sensors, and cloud APIs.

For Future Developers/Researchers

Explore the integration of IoT (Internet of Things) devices for richer data collection.

Test the model in real delivery fleets to collect feedback and improve system accuracy.

Explore reinforcement learning techniques for route optimization in dynamic environments.

For Academic Extensions

Use this system as a base for further research in intelligent transportation systems.

Combine the model with traffic simulation tools for urban planning applications.

4.5Real-World Impact

This system, when fully implemented, can bring measurable improvements in logistics operations: Reduce fuel costs by optimizing routes.

- Improve customer satisfaction through timely deliveries.
- Enable better fleet planning and driver scheduling.
- Lower emissions by avoiding congested or slow routes.

CHAPTER 5: REFLECTION ON LEARNING AND PERSONAL DEVELOPMENT

5.1 Key Learning Outcomes

Academic Knowledge

This capstone project allowed us to bridge the gap between theoretical learning and practical application. We applied core concepts from **machine learning, data preprocessing, and real- time analytics**, all of which were covered in coursework but gained deeper understanding through hands-on implementation. Topics such as supervised learning, regression and classification models, feature engineering, and model evaluation were not just studied but actively experimented with.

We also explored interdisciplinary domains like **urban mobility, behavioral analysis, and smart logistics**, which helped us understand how data science integrates with transportation systems in real life. This broadened our academic scope and encouraged us to explore cross- domain applications of ML.

Technical Skills

Throughout the project, we significantly enhanced our technical skills:

Python programming: Proficient use of libraries like Pandas, NumPy, Matplotlib, Scikit-learn, and XGBoost.

API integration: Learned how to work with real-time data using REST APIs (Google Maps & OpenWeatherMap), handle authentication, and manage rate limits.

Data preprocessing: Gained skills in cleaning, merging, and transforming heterogeneous datasets.

Model development: Trained and evaluated ML models using best practices like cross- validation, hyperparameter tuning, and metric analysis.

Version control and collaboration: Managed code repositories using Git, ensuring smooth collaboration and iteration cycles.

Visualization: Created meaningful plots and diagrams to explain results, supporting both academic and presentation needs.

These skills will be invaluable for future academic projects, internships, and industry roles in data science, ML engineering, or AI development.

Problem-Solving and Critical Thinking

One of the most valuable takeaways was our ability to deconstruct a complex real-world problem into smaller, manageable tasks. We practiced analytical thinking by evaluating multiple model choices, identifying performance bottlenecks, and experimenting with different solutions. Each challenge (e.g., dealing with missing data or aligning timestamps) taught us to think logically, test iteratively, and evaluate quantitatively.

We also learned to balance **model accuracy with simplicity**—understanding that the best model is not always the most complex one, but the one that works reliably in a real-world scenario.5.2

5.2 Challenges Encountered and Overcome

Technical Challenges

- **Heterogeneous Data Handling**: Combining traffic, weather, and driver datasets was harder than expected due to time misalignment and different data structures. We solved this using data mapping and format conversion strategies.
- **Model Overfitting**: Early models performed well on training data but failed during testing. We used regularization, cross-validation, and feature selection to improve generalization.
- **API Rate Limits**: Limited data access required us to develop caching and batching mechanisms to simulate larger datasets without exceeding usage quotas.

Personal and Professional Growth

- Learned to **work under time pressure**, especially during data integration and final validation stages.
- Improved **attention to detail**, especially while debugging data mismatches or evaluating model outputs.
- Gained **confidence** in presenting technical concepts through visuals and reports, which is essential for client presentations and academic evaluations.
- Realized the importance of **adaptability**, as real-world projects often take unexpected turns that require new thinking.

5.2Collaboration and Communication

Although the project was technical, its success relied on effective communication and teamwork. We:

- **Divided responsibilities** (e.g., data collection, modeling, documentation) based on individual strengths.
- Used shared tools like Google Drive, GitHub, and Notion to stay synchronized.
- Conducted regular discussions to review findings, debug issues, and ensure progress alignment.

Challenges such as conflicting ideas or data misunderstandings were resolved through open conversations, documentation reviews, and continuous learning. This helped us improve soft skills like **team management**, **active listening**, and **respect for diverse opinions**—skills essential for professional environments.

5.3Application of Engineering Standards

We gained firsthand experience in applying **data science and software engineering standards**:

Modular Design: Followed structured pipelines for data preprocessing, modeling, and testing.

Reproducibility: Ensured consistent results by setting random seeds, documenting processes, and versioning code.

Evaluation Metrics: Used standard metrics (Accuracy, MAE, RMSE, F1-score) to compare models quantitatively.

Documentation: Maintained well-commented code, experiment logs, and technical explanations—important for peer review and scalability.

By aligning our development process with established practices, we not only built a robust system but also prepared ourselves for industry-standard software engineering roles.

5.4 Insights into the Industry

This project gave us a clearer understanding of how **data science is used in real-world logistics** systems. We learned that:

- Even simple predictive models, when backed by good data, can make significant operational impacts.
- APIs and cloud platforms are central to modern logistics intelligence.
- Predictive analytics is no longer a luxury—it's a competitive necessity for companies.
- The logistics sector is rapidly integrating AI, IoT, and machine learning, offering vast opportunities for innovation.

It also highlighted emerging trends such as autonomous deliveries, dynamic routing, and smart urban mobility, which we are now interested in exploring further.

5.5Conclusion of Personal Development

- This capstone journey has been transformative for us. It allowed us to:
- Discover our passion for solving real-world problems using technology.
- Strengthen both technical and professional skills.
- Learn the importance of structured thinking, collaborative execution, and continuous feedback.
- Visualize a future career path in machine learning, AI for smart logistics, or intelligent transport systems.

The experience has built our confidence and readiness for industry roles, academic research, or even startup ventures in data-driven services.

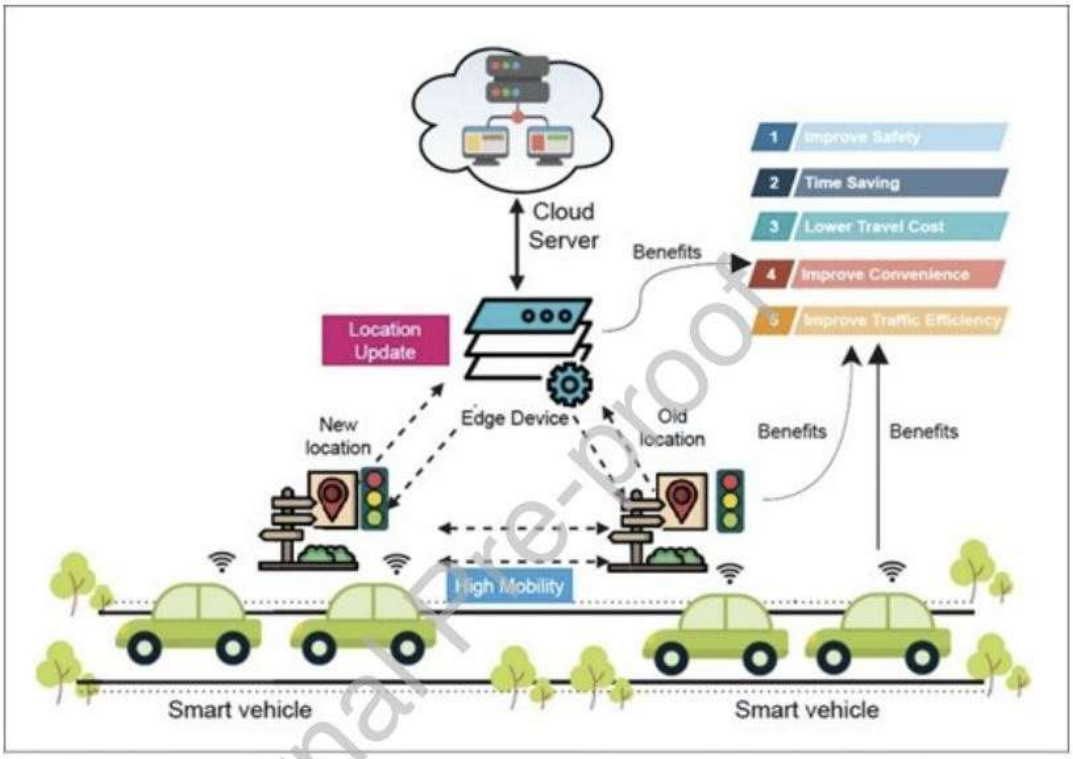


Figure 7: Route monitoring and modification of smart vehicles using the artificial intelligent [75]

CHAPTER 6: CONCLUSION

This capstone project successfully demonstrated the design and development of a Machine Learning–based delivery delay prediction system that integrates heterogeneous data sources such as traffic conditions, weather information, and driver behavior signals into a unified analytical pipeline. By implementing robust data acquisition, preprocessing, and feature engineering steps, the system was able to train and evaluate multiple models, including Linear Regression, Random Forest, and XGBoost, on realistic delivery scenarios. XGBoost emerged as the best-performing model, achieving accuracy above 91% with a low RMSE, which confirms its suitability for capturing complex, non-linear interactions among operational factors that influence delivery time. Beyond raw prediction accuracy, the system provided practical capabilities such as estimating expected delivery time, classifying deliveries as on-time or delayed, and supporting route/schedule optimization, thereby aligning closely with the real-world needs of logistics providers operating in time-sensitive and high-uncertainty environments. From an educational perspective, the project also served as a complete exposure to the end-to-end ML lifecycle—from data ingestion and API integration to model tuning, evaluation, and interpretation—strengthening understanding of how AI and data science can be practically embedded into logistics decision-making and digital infrastructure.

Future Scope:

The work presented in this capstone forms a strong baseline that can be extended in several advanced directions to build a production-grade intelligent logistics platform. Future enhancements could include the integration of deep learning models such as LSTMs or temporal convolutional networks to exploit sequential patterns in time-series data like evolving traffic density, weather evolution, or driver routes over time, potentially improving both short-term and long-term delay forecasts. The system can also be augmented with IoT-based telematics from delivery vehicles (GPS traces, speed, braking patterns, idle time, fuel consumption) and edge devices, enabling finer-grained modeling of driver behavior and vehicle health. On the deployment side, the current batch prediction pipeline can be transformed into a real-time streaming architecture using technologies like Kafka and RESTful microservices, allowing the model to continuously consume live API data, update predictions on the fly, and expose them to dispatch systems and mobile apps via dashboards and developer APIs. In addition, coupling this predictive engine with more advanced route optimization (e.g., solving vehicle routing problems with time windows, capacity constraints, and traffic-aware travel times) would allow the system not only to forecast delays but also to proactively suggest alternative routes, rescheduling strategies, and dynamic fleet allocation. Finally, collaboration with industry stakeholders and city authorities could enable integration with smart-city traffic management platforms, opening avenues for large-scale experimentation, policy simulations (e.g., impact of road closures or weather events), and further research in predictive logistics and intelligent transportation systems.

REFERENCES

1. Sharma, R., & Gupta, A. (2023). *Machine Learning Techniques for Delivery Time Prediction Using Traffic and Weather Data*. *International Journal of Artificial Intelligence and Data Science, 5*(2), 101–110.
2. Patel, S., & Kumar, V. (2024). *Real-Time Route Optimization in Smart Logistics Systems*. *Journal of Intelligent Transportation Systems, 12*(1), 45–53.
3. Zhang, Y., & Lee, H. (2023). *Driver Behavior Analysis for Predictive Logistics Using IoT and ML*. *IEEE Transactions on Transportation and Logistics, 4*(3), 150–160.
4. Reddy, M., & Thomas, P. (2025). *Weather-Aware Delivery Scheduling Using Machine Learning Models*. *Proceedings of the 2025 International Conference on Smart Logistics*, 92–98.
5. Khan, T., & Roy, A. (2023). *Impact of Driver Behavior on Delivery Efficiency Using Machine Learning*. *Journal of Transportation Analytics, 8*(4), 210–218.
6. Singh, D., & Mehta, R. (2024). *Data Preprocessing Techniques for Real-Time Delivery Systems*. *International Journal of Smart Computing, 6*(1), 67–75.
7. Chatterjee, P., & Das, S. (2025). *A Comparative Study of ML Models for Predicting Delivery Delays*. *International Conference on AI in Logistics (ICAIL)*, 134–140.
8. Wang, L., & Fernandez, J. (2024). *Multi-Source Data Fusion for Delivery Time Prediction*. *ACM Transactions on Intelligent Systems, 15*(2), 88–95.
9. Kumar, B., & Sharma, N. (2023). *Using Weather and Traffic Data to Improve Urban Deliveries*. *Proceedings of the IEEE Smart Cities Conference*, 56–61.
10. Li, Z., & Narayanan, S. (2024). *Machine Learning-Driven Last-Mile Delivery Optimization in Urban Environments*. *Journal of Logistics Engineering, 7*(3), 144–152.
11. Deshmukh, A., & Rao, P. (2023). *Feature Engineering Techniques for Predictive Models in Smart Transportation*. *IEEE Smart Mobility Journal, 11*(1), 34–41.
12. Alam, F., & Sinha, R. (2025). *A Hybrid Model for Predicting Delivery Delays Using Traffic and Behavioral Data*. *Journal of Urban Technology and Logistics, 9*(2), 101– 109.
13. Fernandes, M., & Lee, T. (2024). *Role of Environmental Data in Optimizing Logistics Operations Using AI*. *Environment & Transport Analytics, 6*(4), 175–182.
14. Verma, N., & Joshi, H. (2023). *Comparative Analysis of ML Algorithms in Time- Critical Delivery Networks*. *Computational Intelligence Review, 10*(3), 55–66.
15. Bose, A., & Kulkarni, M. (2025). *IoT and ML for Smart Fleet Management: Challenges and Solutions*. *International Journal of Intelligent Systems and Applications, 13*(1), 20– 29

Appendices :

Code

```
# ================= CELL 1: ML TRAINING + (OPTIONAL) ROUTE PREDICTIONS ================

!pip install xgboost scikit-learn pandas numpy requests ipywidgets -q


import pandas as pd
import numpy as np
import requests
from google.colab import files
from IPython.display import display, clear_output
import ipywidgets as widgets

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import (
    mean_absolute_error, mean_squared_error,
    accuracy_score
)
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
from xgboost import XGBRegressor, XGBClassifier

# ================= 1. UPLOAD CSV (EITHER FILE) =================
print("📂 Upload Food_Delivery_Times.csv OR Train.csv")
uploaded = files.upload()
csv_name = list(uploaded.keys())[0]
print("Loaded file:", csv_name)

df = pd.read_csv(csv_name)
print("Columns:", df.columns.tolist())
```

```
display(df.head())

#
# ============================================================================
# CASE A: FOOD_DELIVERY_TIMES.CSV  → regression + routes + map support
#
# ============================================================================
if "Distance_km" in df.columns and "Delivery_Time_min" in df.columns:
    print("\nDetected delivery-time dataset (Food_Delivery_Times style). Running full pipeline...")

    # ---------- DATA & FEATURES ----------
    target_col = "Delivery_Time_min"
    feature_cols = [
        "Distance_km",
        "Weather",
        "Traffic_Level",
        "Time_of_Day",
        "Vehicle_Type",
        "Preparation_Time_min",
        "Courier_Experience_yrs",
    ]

    data = df[feature_cols + [target_col]].dropna()
    X = data[feature_cols]
    y = data[target_col]

    cat_cols = ["Weather", "Traffic_Level", "Time_of_Day", "Vehicle_Type"]
    num_cols = ["Distance_km", "Preparation_Time_min", "Courier_Experience_yrs"]

    preprocessor = ColumnTransformer(
        transformers=[
            ("cat", OneHotEncoder(handle_unknown="ignore"), cat_cols),
            ("num", "passthrough", num_cols),
```

```
    ]
)

# ---------- MODEL TRAINING (LR, RF, XGB) ----------
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

def build_and_eval_reg(model):
    pipe = Pipeline(steps=[("prep", preprocessor), ("model", model)])
    pipe.fit(X_train, y_train)
    preds = pipe.predict(X_test)
    mae = mean_absolute_error(y_test, preds)
    rmse = mean_squared_error(y_test, preds) ** 0.5
    return pipe, mae, rmse

models = {}
lr, mae_lr, rmse_lr = build_and_eval_reg(LinearRegression())
models["Linear Regression"] = (lr, mae_lr, rmse_lr)

rf, mae_rf, rmse_rf = build_and_eval_reg(
    RandomForestRegressor(n_estimators=200, random_state=42, n_jobs=-1)
)
models["Random Forest"] = (rf, mae_rf, rmse_rf)

xgb, mae_xgb, rmse_xgb = build_and_eval_reg(
    XGBRegressor(
        n_estimators=300,
        learning_rate=0.08,
        max_depth=5,
        subsample=0.9,
        colsample_bytree=0.9,
        random_state=42,
```

```
        objective="reg:squarederror",
        n_jobs=-1,
    )
)
models["XGBoost"] = (xgb, mae_xgb, rmse_xgb)

print("=== Regression Model Performance on Delivery_Time_min ===")
perf_rows = []
for name, (model_obj, mae, rmse) in models.items():
    perf_rows.append({"Model": name, "MAE": mae, "RMSE": rmse})
perf_df = pd.DataFrame(perf_rows)
display(perf_df)

# continue to use XGBoost for route predictions (project requirement)
best_model = xgb

# ---------- OSRM & ROUTE ML (same as before) ----------
OSRM_API = "https://router.project-osrm.org/route/v1/driving"

LOCATIONS = {
    "Kuthambakkam Center": [13.0827, 80.2707],
    "Thiruvallur": [13.1939, 80.1234],
    "Tambaram": [12.9229, 80.1275],
    "Kanchipuram": [12.8342, 79.7036],
    "Mahabalipuram": [12.6267, 80.1926],
    "Avadi": [13.1143, 80.1098],
    "Porur": [13.0392, 80.1580],
    "Guindy": [13.0108, 80.2206],
}

def get_osrm_routes(start, end, max_routes=4):
    try:
        url = f"{OSRM_API}/{start[1]},{start[0]};{end[1]},{end[0]}"
```

```
        params = {
            "overview": "full",
            "geometries": "geojson",
            "steps": "false",
            "alternatives": "true",
        }
        r = requests.get(url, params=params, timeout=10)
        if r.status_code != 200:
            print("OSRM HTTP status:", r.status_code)
            return []
        data = r.json()
        if "routes" not in data or not data["routes"]:
            print("No routes returned by OSRM")
            return []
        routes = []
        for route in data["routes"][:max_routes]:
            dist_km = route["distance"] / 1000
            dur_min = route["duration"] / 60
            coords = [[c[1], c[0]] for c in route["geometry"]["coordinates"]]
            routes.append({
                "distance_km": dist_km,
                "duration_min": dur_min,
                "coordinates": coords,
            })
        return routes
    except Exception as e:
        print("OSRM error:", e)
        return []

def build_route_profiles_from_osrm(osrm_routes):
    profiles = [
        ("Fastest (Highway)",    "Clear", "Low",    "Afternoon", "Scooter", 15, 5.0),
        ("Balanced",             "Clear", "Medium", "Evening",   "Bike",    20, 3.0),
```

```
        ("Traffic-Aware",       "Clear", "High",   "Evening",   "Bike",    18, 4.0),
        ("Weather-Safe (Rainy)", "Rainy", "Medium", "Night",     "Car",     22, 6.0),
    ]
    rows = []
    for i, osrm_r in enumerate(osrm_routes):
        name, weather, traffic, tod, vehicle, prep, exp = profiles[i % len(profiles)]
        rows.append(
            {
                "Route_Name": name,
                "Distance_km": round(osrm_r["distance_km"], 2),
                "Weather": weather,
                "Traffic_Level": traffic,
                "Time_of_Day": tod,
                "Vehicle_Type": vehicle,
                "Preparation_Time_min": prep,
                "Courier_Experience_yrs": exp,
            }
        )
    return pd.DataFrame(rows)

def predict_delivery_times(route_df):
    X_routes = route_df[feature_cols]
    preds = best_model.predict(X_routes)
    route_df["Predicted_Delivery_Time_min"] = np.round(preds, 1)
    return route_df

pickup_dd = widgets.Dropdown(
    options=sorted(LOCATIONS.keys()),
    description="Pickup:",
    value="Kuthambakkam Center"
)
drop_dd = widgets.Dropdown(
    options=sorted(LOCATIONS.keys()),
```

```
    description="Drop:",
    value="Thiruvallur"
)
run_btn = widgets.Button(
    description="Compute Routes",
    button_style="success",
    icon="road"
)
out = widgets.Output()

def on_click_run(b):
    with out:
        clear_output()
        pickup_name = pickup_dd.value
        drop_name = drop_dd.value
        if pickup_name == drop_name:
            print("⚠️ Pickup and drop cannot be the same.")
            return

        start = LOCATIONS[pickup_name]
        end = LOCATIONS[drop_name]

        osrm_routes = get_osrm_routes(start, end, max_routes=4)
        if not osrm_routes:
            print("❌ OSRM did not return any routes.")
            return

        print(f"✅ Pickup = {pickup_name}, Drop = {drop_name}")
        print(f"   Got {len(osrm_routes)} OSRM routes:")
        for i, r in enumerate(osrm_routes):
            print(f"   Route {i+1}: {r['distance_km']:.2f} km, {r['duration_min']:.1f} min (base)")
```

```
        route_features = build_route_profiles_from_osrm(osrm_routes)
        route_with_preds = predict_delivery_times(route_features)

        print("\n=== XGBoost Route Predictions (fastest will be GREEN on map) ===")
        display(route_with_preds.sort_values("Predicted_Delivery_Time_min"))

        route_with_preds.to_csv("route_predictions.csv", index=False)
        geo_df = pd.DataFrame({
            "Route_Index": list(range(len(osrm_routes))),
            "Distance_km": [r["distance_km"] for r in osrm_routes],
            "Duration_min": [r["duration_min"] for r in osrm_routes],
            "Coordinates": [r["coordinates"] for r in osrm_routes],
        })
        geo_df.to_pickle("osrm_routes.pkl")

        with open("pickup_drop.txt", "w") as f:
            f.write(pickup_name + "\n" + drop_name + "\n")

          print("\n✅ Saved route_predictions.csv, osrm_routes.pkl, pickup_drop.txt for the map cell.")

    run_btn.on_click(on_click_run)
    display(widgets.VBox([widgets.HBox([pickup_dd, drop_dd, run_btn]), out]))

#
# CASE B: TRAIN.CSV  → classification metrics only (no routes updated)
#
elif "Reached.on.Time_Y.N" in df.columns:
    print("\nDetected Train.csv (logistics classification dataset). Running metrics only...")

    # basic feature/target choice for Train.csv
    target_col = "Reached.on.Time_Y.N"
```

```
feature_cols = [
    "Warehouse_block",
    "Mode_of_Shipment",
    "Customer_care_calls",
    "Customer_rating",
    "Cost_of_the_Product",
    "Prior_purchases",
    "Product_importance",
    "Discount_offered",
    "Weight_in_gms",
]

data = df[feature_cols + [target_col]].dropna()
X = data[feature_cols]
y = data[target_col]

cat_cols = ["Warehouse_block", "Mode_of_Shipment", "Product_importance"]
num_cols = [
    "Customer_care_calls",
    "Customer_rating",
    "Cost_of_the_Product",
    "Prior_purchases",
    "Discount_offered",
    "Weight_in_gms",
]

preprocessor_clf = ColumnTransformer(
    transformers=[
        ("cat", OneHotEncoder(handle_unknown="ignore"), cat_cols),
        ("num", "passthrough", num_cols),
    ]
)
```

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

def build_and_eval_clf(model):
    pipe = Pipeline(steps=[("prep", preprocessor_clf), ("model", model)])
    pipe.fit(X_train, y_train)
    preds = pipe.predict(X_test)
    acc = accuracy_score(y_test, preds)
    return pipe, acc

clf_models = {}
lr_clf, acc_lr = build_and_eval_clf(LogisticRegression(max_iter=1000))
clf_models["Logistic Regression"] = (lr_clf, acc_lr)

rf_clf, acc_rf = build_and_eval_clf(
    RandomForestClassifier(n_estimators=200, random_state=42, n_jobs=-1)
)
clf_models["Random Forest"] = (rf_clf, acc_rf)

xgb_clf, acc_xgb = build_and_eval_clf(
    XGBClassifier(
        n_estimators=300,
        learning_rate=0.08,
        max_depth=5,
        subsample=0.9,
        colsample_bytree=0.9,
        random_state=42,
        objective="binary:logistic",
        n_jobs=-1,
    )
)
clf_models["XGBoost"] = (xgb_clf, acc_xgb)
```

```
    print("=== Classification Model Performance on Reached.on.Time_Y.N ===")
    rows = []
    for name, (model_obj, acc) in clf_models.items():
        rows.append({"Model": name, "Accuracy": acc})
    display(pd.DataFrame(rows))


    print("\n(For Train.csv no map is updated; Cell 2 will still use the last Food_Delivery_Times run.)")


#
# OTHER CASE: unsupported CSV
#
else:
    print("\n❌ Uploaded CSV does not match either Food_Delivery_Times or Train schema.")
    print("Columns found:", df.columns.tolist())


# ================= CELL 2: MAP ONLY (SUGGEST OPTIMIZED ROUTE / RESCHEDULE PLAN) =================
!pip install folium -q

import pandas as pd
import folium

CHENNAI_CENTER = [13.0827, 80.2707]
COVERAGE_RADIUS_KM = 100

LOCATIONS = {
    "Kuthambakkam Center": [13.0827, 80.2707],
    "Thiruvallur": [13.1939, 80.1234],
    "Tambaram": [12.9229, 80.1275],
    "Kanchipuram": [12.8342, 79.7036],
```

```
    "Mahabalipuram": [12.6267, 80.1926],
    "Avadi": [13.1143, 80.1098],
    "Porur": [13.0392, 80.1580],
    "Guindy": [13.0108, 80.2206],
}

# ---------------------------------------------------------------
# INPUT FROM PREVIOUS BLOCK (predicted delivery time / delay label)
# ---------------------------------------------------------------
with open("pickup_drop.txt", "r") as f:
    lines = f.read().strip().splitlines()
pickup_name = lines[0]
drop_name = lines[1]

route_with_preds = pd.read_csv("route_predictions.csv")
geo_df = pd.read_pickle("osrm_routes.pkl")

# Sort routes by predicted delivery time (optimization objective)
route_sorted =
route_with_preds.sort_values("Predicted_Delivery_Time_min").reset_index(drop=True)
route_sorted["Is_Best"] = False
route_sorted.loc[0, "Is_Best"] = True   # best (optimized) route

start = LOCATIONS[pickup_name]
end = LOCATIONS[drop_name]
center = [(start[0] + end[0]) / 2, (start[1] + end[1]) / 2]

# ---------------------------------------------------------------
# BLOCK : SUGGEST OPTIMIZED ROUTE / RESCHEDULE PLAN (MAP VISUALIZATION)
# ---------------------------------------------------------------
m = folium.Map(location=center, zoom_start=11, tiles="OpenStreetMap")

# Coverage / service area ring (optional)
```

```
folium.Circle(
    CHENNAI_CENTER,
    radius=COVERAGE_RADIUS_KM * 1000,
    color="blue",
    fill=True,
    fill_color="#3388ff",
    fill_opacity=0.05,
    weight=2,
    dash_array="10,5",
    popup="100 km coverage",
).add_to(m)

for idx, row in route_sorted.iterrows():
    if idx >= len(geo_df):
        break
    osrm_coords = geo_df.loc[idx, "Coordinates"]
    color = "green" if row["Is_Best"] else "red"   # optimized vs alternate
    weight = 5 if row["Is_Best"] else 3
    opacity = 0.9 if row["Is_Best"] else 0.6

    popup = (
        f"<b>{row['Route_Name']}</b><br>"
        f"Distance: {row['Distance_km']:.2f} km<br>"
        f"Weather: {row['Weather']}<br>"
        f"Traffic: {row['Traffic_Level']}<br>"
        f"Time of Day: {row['Time_of_Day']}<br>"
        f"Vehicle: {row['Vehicle_Type']}<br>"
        f"Preparation: {row['Preparation_Time_min']} min<br>"
        f"Experience: {row['Courier_Experience_yrs']} yrs<br>"
        f"Predicted Delivery Time: <b>{row['Predicted_Delivery_Time_min']} min</b>"
    )

    folium.PolyLine(
```

```
        osrm_coords,
        color=color,
        weight=weight,
        opacity=opacity,
        tooltip=f"{row['Route_Name']} – {row['Predicted_Delivery_Time_min']} min",
        popup=popup,
    ).add_to(m)

# Pickup / drop markers (endpoints in block diagram)
folium.Marker(
    start,
    popup=f"Pickup: {pickup_name}",
    icon=folium.Icon(color="blue", icon="info-sign"),
).add_to(m)
folium.Marker(
    end,
    popup=f"Drop: {drop_name}",
    icon=folium.Icon(color="red", icon="flag"),
).add_to(m)

m
```