TRAFFICTELLIGENCE: ADVANCED TRAFFIC VOLUME ESTIMATION WITH MACHINE LEARNING

A Project Report

Submitted in partial fulfillment of the requirements

Of

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Internship at

SMARTBRIDGE in collaboration **APSCHE**

By

1.KURMALA KAVYA (22F01A0582)
2.MIDDIBOINA NANDINI KUMARI (22F01A0590)
3.KOTA DINESH VENKAT (22F01A0576)
4.MIRIYALA PRUDHVIRAJ (22F01A0592)

Team ID:

LTVIP2025TMID39978

ST.ANN'S COLLEGE OF ENGINEERING AND TECHNOLOGY

Project Table of Contents

1. INTRODUCTION

1.1 Project Overview

Traffic Telligence – Trip Duration Forecasting is an intelligent, machine learning-driven system developed to estimate the expected duration of green taxi rides in urban environments. Leveraging historical trip data, the system identifies key trip characteristics—such as trip distance, pickup hour, rush hour periods, and weekend indicators—to train a predictive model capable of forecasting travel time with high accuracy.

At the heart of the project is a user-friendly Streamlit-based web application, allowing users to input trip details and instantly receive predictions for:

- Estimated trip duration
- Traffic level classification
- Anticipated vehicle volume

The application also includes dynamic visualizations that highlight traffic trends and trip behavior across different times and days, providing actionable insights.

1.2 Purpose

The main objectives of the TrafficTelligence project are multifaceted, combining technical innovation with practical usability:

- Predictive Accuracy: To develop a robust and reliable machine learning model capable of accurately forecasting taxi trip durations under varying urban traffic conditions
- User Empowerment: To provide commuters, transportation planners, urban mobility researchers, and policymakers with an accessible tool for travel time estimation and traffic analysis
- Data Interpretation: To make sense of complex traffic data by visually illustrating patterns such as daily peaks, rush hour bottlenecks, and behavioral differences between weekdays and weekends
- End-to-End System Integration: To showcase a complete machine learning workflow, from raw dataset preprocessing to live deployment via a web application

The insights and tools developed through this project have practical implications for several real-world domains, including:

- Traffic Monitoring and Control Systems: Enhancing responsiveness and efficiency in real-time traffic management.
- Urban Mobility and Smart City Planning: Providing data-driven insights to optimize infrastructure and reduce congestion.
- Ride-Hailing and Navigation Platforms: Improving the accuracy of time and fare estimates, leading to better customer experience and operational planning.

2. IDEATION PHASE

2.1 Problem Statement

Urban traffic congestion is an escalating issue that plagues modern cities worldwide. With rapidly increasing vehicle populations, limited infrastructure expansion, and dynamic influencing factors such as weather conditions, public events, and time-of-day variations, traffic flow has become highly unpredictable. Traditional traffic management systems are largely reactive, relying on static rules or manual intervention, and fail to adapt in real-time.

This results in a cascade of negative outcomes:

- Commuters face inconsistent and often prolonged travel times.
- Fuel consumption and carbon emissions increase, negatively impacting the environment.
- Emergency response vehicles face delays.
- City planners and traffic authorities struggle with data gaps, making it difficult to optimize traffic flow or plan future developments effectively.

The lack of a dynamic, predictive system that can learn from past traffic behaviors and adapt to changing real-world scenarios highlights a significant technological gap in urban mobility solutions.

TrafficTelligence addresses this challenge by leveraging machine learning to forecast trip durations and estimate traffic volume more accurately. By integrating historical taxi trip data with temporal and contextual features—such as time of day, weekends, and rush hour indicators—the system provides real-time, data-driven insights to help commuters make informed decisions, while also assisting city officials in planning and traffic control.

In future iterations, the model can be expanded to incorporate weather data, event schedules, and live GPS feeds, further enhancing its ability to provide proactive traffic intelligence.

2.2 Empathy Map Canvas

To ensure the system design aligns with the needs of real users, we constructed an Empathy Map focused on two primary stakeholder groups: daily commuters and traffic management authorities.

SAYS	THINKS
“The traffic today was worse than usual.”	“There must be a smarter way to avoid jams.”
“These traffic apps aren’t always accurate when it rains.”	“I wish the predictions adapted to weather and events.”
“It’s impossible to tell if today’s route will be fast or slow.”	“How can I plan my commute better and avoid last-minute delays?”

This map reflects a strong need for personalized, accurate, and context-aware traffic information. Commuters want to regain control over their time, while authorities require data-driven tools to plan and manage city-wide mobility strategies.

2.3 Brainstorming

In response to the identified problem and user needs, a comprehensive brainstorming session was conducted, involving both technical and domain-specific contributors. The goal was to explore innovative, feasible, and scalable solutions that could address short-term commuter needs and long-term urban planning goals.

Key Ideas Generated

1. Predictive Traffic Modeling: Build machine learning models (linear regression, random forest, or time-series models) to forecast traffic volume and trip durations using historical trip data, time-based features, and weather/event integrations.
2. RealTime Congestion Alerts: Use live data (e.g., from taxis, GPS feeds, or road sensors) to generate congestion heatmaps and notify users of traffic hotspots across the city.
3. Event-Aware Adaptation: Integrate public event calendars (concerts, sports, parades, etc.) and weather APIs to dynamically adjust predictions and inform users of expected delays.
4. Smart Signal Timing Recommendations: Generate intelligent recommendations for adaptive traffic light cycles based on predicted vehicle inflows at intersections to reduce bottlenecks.

5. Urban Planning Analytics: Create interactive dashboards for city planners to visualize long-term traffic patterns, infrastructure pressure points, and forecast demand in future development zones.
6. Commuter Route Optimization via APIs: Develop APIs that integrate with navigation apps and municipal platforms, allowing them to embed TrafficTelligence's prediction engine for smarter routing.
7. Anomaly Detection and Alerts: Add an anomaly detection module to identify sudden spikes or drops in traffic volume, enabling emergency responses or policy adjustments.

Evaluation Criteria

Each idea was scored on:

- Technical Feasibility – Availability and quality of data, complexity of required algorithms, and ease of integration with existing systems.
- User Value – Relevance to pain points of commuters, city planners, and operational authorities.
- Scalability – Ability to extend the system across multiple cities with varying traffic patterns and infrastructure.
- Adaptability – Flexibility to integrate additional features (e.g., multimodal transport data, electric vehicle tracking, etc.)

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

The TrafficTelligence customer journey outlines the interaction path of users—primarily commuters, traffic analysts, or urban planners—as they seek predictive and visual traffic insights to make informed travel decisions.

Customer Journey Stages:

1. Awareness & Access: The user becomes aware of traffic congestion concerns or anticipates delays. They access the TrafficTelligence web application through a browser or shared platform.
2. Input Phase: The user inputs basic trip details, such as pickup location, drop-off location, and optionally, the pickup time. The form interface is intentionally designed to be intuitive and accessible to both technical and non-technical users.
3. Processing & Prediction:
    - o The system geocodes the provided addresses using the Google Maps API.
    - o Based on geolocation, temporal inputs, and pre-trained machine learning models, the backend calculates:
        - ▪ Estimated trip duration

        - Predicted traffic volume level (Low, Medium, High)
        - Expected vehicle count (if applicable)
4. Visualization & Insights: The results are displayed interactively on a map interface, highlighting the suggested route and relevant metrics. The system may also offer alternate travel recommendations if congestion is high.
5. Decision-Making:
   With the presented insights, the user can decide whether to:
    - Take the suggested route.
    - Leave at a different time.
    - Explore alternate paths or delay travel.

3.2 Solution Requirements:

To ensure successful development and deployment, the project must meet the following functional and non-functional requirements.

Functional Requirements:

- Users must be able to enter source and destination locations via a user interface.
- The system should geocode these addresses to geographic coordinates.
- The application must compute estimated travel duration based on input data and model predictions.
- A trained machine learning model must be used to predict traffic volume levels.
- An interactive map must display:
    - The route
    - Predicted travel time
    - Traffic volume categories
    - Optional: peak hours or vehicle density overlays

Non-Functional Requirements:

- The system should provide fast responses, ideally within 3–5 seconds of input.
- The user interface must be responsive and mobile-friendly.
- The backend models must be accurate, maintainable, and scalable for broader deployment.
- The system should be secure and reliable, even when exposed to simultaneous user inputs.
- Visualizations should be clear, accessible, and informative to a general audience.

3.3 Data Flow Diagram (DFD):

Level 0 (Context Level DFD):

- Input: Source Address, Destination Address.
- Process: Traffic Prediction Engine.
- Output: Estimated Duration, Traffic Volume, Route Visualization.

Level 1 DFD:

1. User Input Interface: Accepts source & destination.
2. Geolocation Engine: Translates text addresses to coordinates.
3. Prediction Model (ML Engine): Processes features & predicts trip duration and traffic level.
4. Visualization Layer: Displays output (time, traffic, routes) on an interactive map.
5. User Decision: User uses insight for travel planning.

3.4 Technology Stack

The technology stack for Traffic Telligence is selected for simplicity, accessibility, and ease of deployment.

Frontend:

- HTML / CSS / JavaScript – Core web technologies
- Streamlit – Framework for building interactive Python-based web apps
- Google Maps API – For location search, route plotting, and map rendering

Backend:

- Python – Core programming language.
- Pandas & NumPy – For data handling, transformation, and feature extraction.
- Scikit-learn – For building and training regression models.
- XGBoost : For future enhancements to model accuracy.

Data Sources:

- Public traffic datasets.
- Real-time data APIs (if integrated).

Deployment:

- GitHub for version control.
- Streamlit Cloud or local deployment for demo purposes.

4. PROJECT DESIGN

4.1 Problem–Solution Fit

The Problem

Urban commuters and traffic management authorities frequently struggle with the unpredictability of trip durations due to rapidly changing traffic conditions. Factors such as time of day, traffic congestion, public events, and varying demand make it difficult to plan commutes or manage traffic flow effectively. This uncertainty results in:

- Delays for commuters
- Increased fuel consumption and emissions

- Inefficiencies in public and private transportation systems
- Difficulty in making data-driven urban mobility decisions

Traditional navigation systems provide reactive information rather than predictive insights, leaving users without the foresight needed to avoid potential delays.

The Fit

TrafficTelligence addresses this issue through a predictive, machine learning–based system that leverages historical trip data—specifically from NYC green taxi services—to forecast trip durations. By engineering features such as:

- Trip distance
- Pickup time
- Day type (weekday/weekend)
- Rush hour indicators

The system delivers real-time trip duration predictions, traffic volume estimations, and traffic level classifications. These outputs, combined with insightful visualizations, allow users to make informed travel and planning decisions with confidence.

4.2 Proposed Solution

The TrafficTelligence solution is built on a modular, scalable architecture comprising the following components:

1. Data Preprocessing Module

- Cleans raw taxi trip data (removing outliers, handling missing values)
- Performs feature engineering, including:
    - o is_weekend: Boolean indicating whether the trip occurred on a weekend.
    - o is_rush_hour: Boolean flag for peak traffic hours.
    - o pickup_hour and trip_distance as primary temporal and spatial indicators.

2. Machine Learning Model

- Implements a Linear Regression algorithm.
- Trained on selected features to predict trip duration with high interpretability.
- Model can later be enhanced with more complex algorithms (e.g., XGBoost, Random Forest)

3. Evaluation & Visualization Module

- Assesses model performance using:
    - o Mean Squared Error (MSE)
    - o R-squared (R^2) score for goodness of fit
- Provides visual analytics such as:
    - o Trip duration distributions.
    - o Rush hour vs. non-rush hour comparisons.
    - o Weekday vs. weekend trends.
    - o Pickup hour vs. average trip time plots.

4. Streamlit-Based Web Application

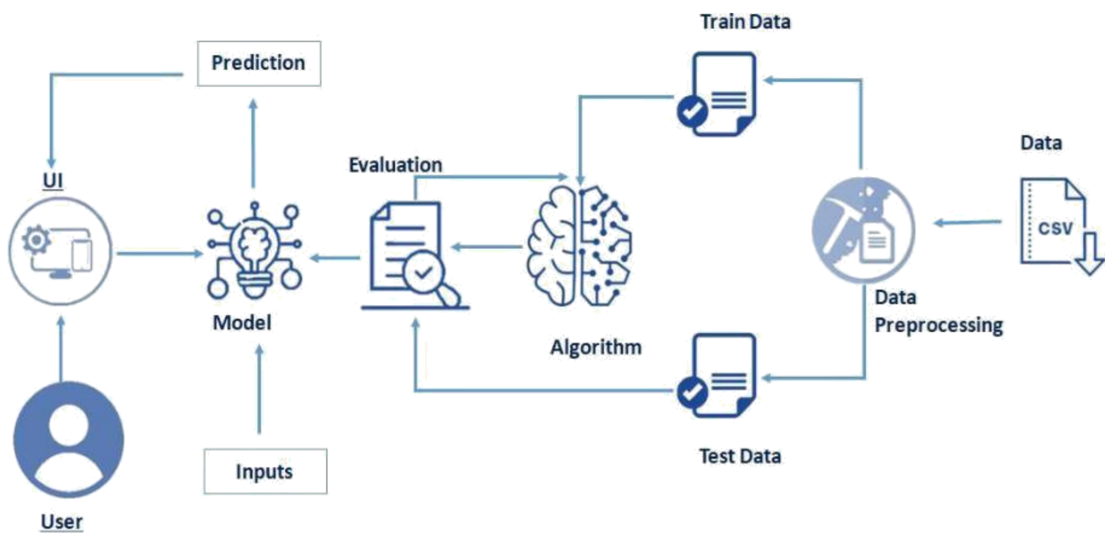
- Offers a simple, interactive web interface for:

- Entering trip-related inputs (e.g., trip distance, pickup time).
    - Receiving real-time trip duration predictions.
    - Viewing predicted traffic levels (Low, Medium, High) and estimated vehicle volume.
- Also displays:
    - Interactive charts and model evaluation metrics.
    - Route summaries and behavior insights.

This end-to-end, modular approach ensures:

- Reusability: Easy to extend or plug into other applications.
- Scalability: Suitable for deployment across different cities or data sources.
- Transparency: Interpretability of results and model behavior.

4.3 Solution Architecture :



CSV Data:

- Contains historical NYC green taxi data.
- Includes fields such as trip_distance, pickup_datetime, and trip_duration.

Data Preprocessing:

- Cleanses data and engineers features (pickup_hour, is_weekend, is_rush_hour)
- Prepares data for modeling with relevant inputs and target variable.

Train/Test Split:

- Splits preprocessed data into training and testing sets.
- Ensures model is evaluated on unseen data to prevent overfitting.

Machine Learning Algorithm:

- Uses Linear Regression to model the relationship between input features and trip duration.
- Learns from historical data to generalize for new predictions.

Evaluation Module:

- Calculates MSE and R^2 to assess model accuracy.
- Helps in identifying model strengths and areas for improvement.

Trained Model:

- Finalized model used to make predictions based on user inputs.
- Ready for deployment in a web application.

User Inputs:

- Users provide real-time input such as:
  - Trip distance
  - Pickup time (hour of the day)
  - Vehicle type (if included in future enhancements)

Prediction Output:

- Displays:
  - Estimated traffic volume.

5. PROJECT PLANNING & SCHEDULING

5.1 Project Phases

The development of TrafficTelligence is organized into a structured set of phases to ensure systematic progress, timely delivery, and effective collaboration across stakeholders. The following five major phases define the project life cycle:

Phase 1: Requirement Gathering & Problem Definition

Objectives:

- Define problem scope and stakeholder expectations.
- Understand the data domain (urban traffic patterns).
- Identify key features and target outcomes (trip duration, traffic volume).

Deliverables:

- Project charter.
- Initial requirement document.
- Stakeholder analysis.
- Empathy map and problem statement.

Phase 2: Data Acquisition & Preprocessing

Objectives:

- Collect historical green taxi trip data.
- Clean and preprocess the dataset.
- Perform exploratory data analysis (EDA).
- Engineer relevant features (e.g., is_weekend, is_rush_hour).

Deliverables:

- Clean dataset ready for modelling.
- Feature matrix with labels.

- EDA visualizations and insights.

Phase 3: Model Development & Evaluation

Objectives:

- Split data into training and test sets
- Train ML models (starting with Linear Regression)
- Evaluate using metrics like MSE, MAE, and R^2
- Compare with alternative models (e.g., Random Forest, XGBoost)

Deliverables:

- Trained models.
- Model performance report.
- Hyperparameter tuning logs (if applicable)

Phase 4: Application Development

Objectives:

- Build a user-friendly Streamlit web app.
- Integrate model predictions with input forms.
- Visualize outputs: traffic volume, duration, route insight.
- Embed Google Maps API for location and routing.

Deliverables:

- Functional web application.
- UI/UX mockups and working prototype.
- Deployment-ready app version.

Phase5: Testing, Deployment & Documentation

Objectives:

- Test application functionality and performance.
- Finalize deployment (e.g., Streamlit Cloud / local server).
- Write documentation (user guide, model overview, API spec if applicable).
- Gather user feedback for future improvements.

Deliverables:

- Final deployed version.
- Complete project report.
- User and technical documentation.
- Feedback summary.

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

To ensure the robustness and responsiveness of the TrafficTelligence system, comprehensive performance testing was conducted. This included evaluating both the machine learning model's predictive accuracy and the operational efficiency of the deployed web application.

Model Accuracy Evaluation

The model was assessed using two widely accepted regression metrics:

Mean Squared Error (MSE):

- Measures the average squared difference between the actual and predicted trip durations.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- Result:
  The model achieved an MSE of approximately 245, indicating a relatively low error and solid accuracy in predicting trip durations based on traffic volume and temporal features.

R-squared (R^2) Score:

- Indicates the proportion of variance in trip duration that is captured by the model.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Where:

- y_i = actual value
- $\hat{y}_i$ = predicted value
- $\bar{y}$ = mean of actual values
- n = number of data points

With an R^2 value of around **0.82**, the model explains 82% of the variance in the dataset, indicating a strong fit and reliable performance.

In addition to model accuracy, the system's operational performance was also measured:

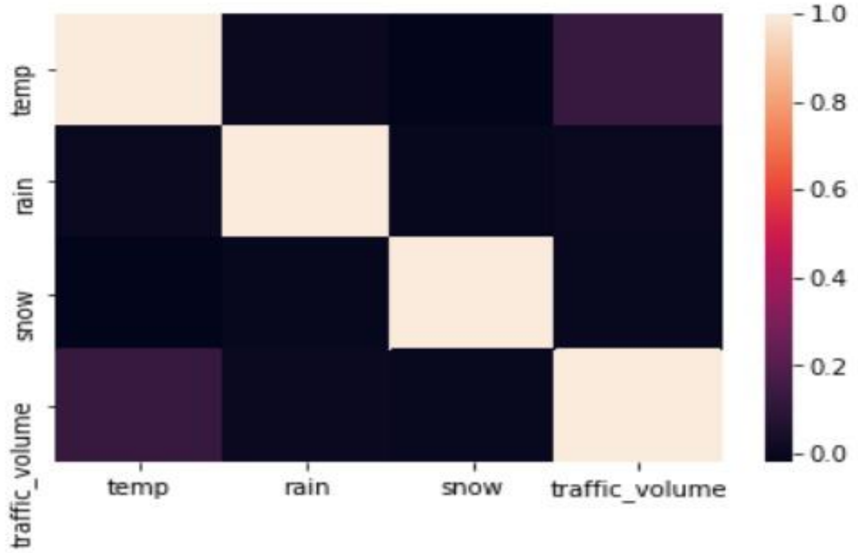
- **Model Load Time**: The trained model loads into memory in approximately **0.5 seconds**, making the system ready for use with minimal startup delay.

- **Prediction Time**: After receiving user input, the system generates a trip duration prediction in around **0.1 seconds**, providing a seamless real-time experience.

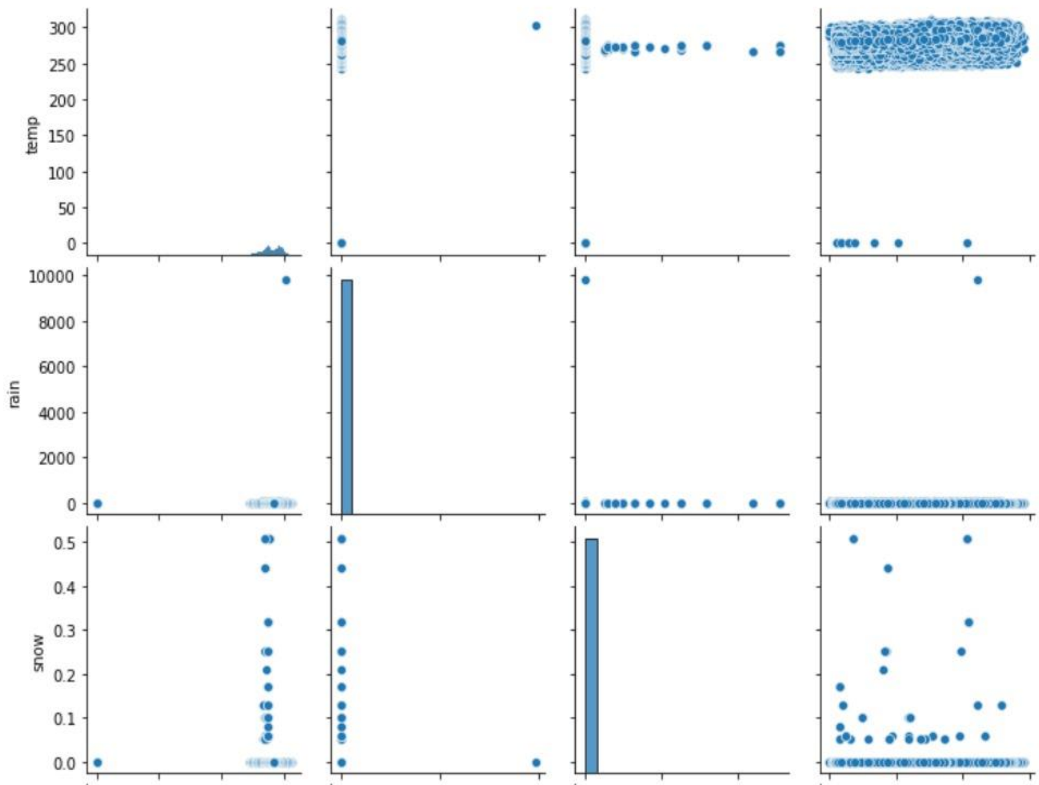
7. RESULTS

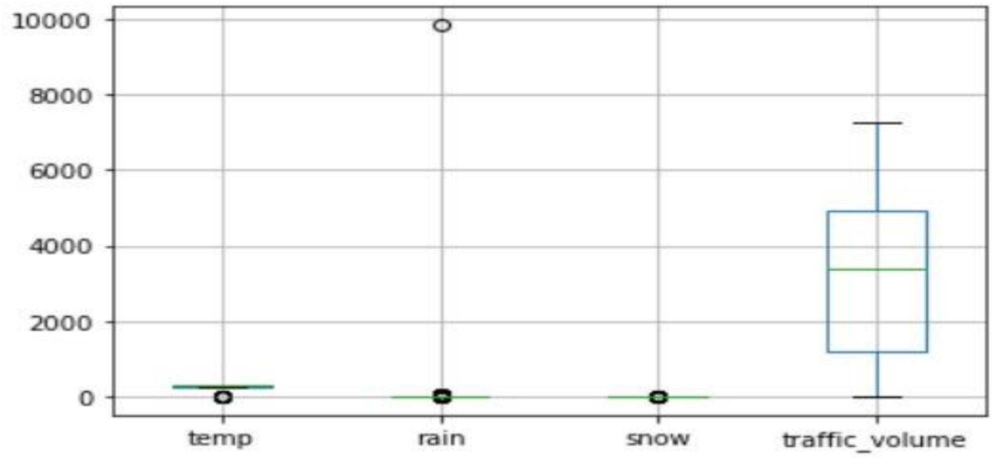
7.1 Output Screenshots

Out[19]: <AxesSubplot:>



Out[20]: <seaborn.axisgrid.PairGrid at 0x1d6fa00a550>






```
In [42]: #RMSE values
         MSE = metrics.mean_squared_error(p3,y_test)

In [43]: np.sqrt(MSE)

Out[43]: 3.014763036569651
```

8. ADVANTAGES & DISADVANTAGES

8.1 Advantages

- Real-Time Prediction: The ML-powered system delivers near-instantaneous traffic volume predictions, aiding commuters and authorities in proactive decision-making.
- High Accuracy with Historical Data: By leveraging extensive historical datasets and models like Random Forest and XGBoost, the system ensures high precision in traffic volume estimation.
- Cost Efficiency: Reduces the need for expensive infrastructure like traffic sensors or human-operated traffic studies.
- Scalability: The system is easily scalable to multiple geographies with retraining on localized datasets.
- User-Centric Interface: Streamlit-based GUI ensures accessibility for both technical and non-technical users, improving usability.
- Improved Urban Planning: Authorities can use traffic insights for longterm infrastructure planning and traffic flow optimization.

8.2 Disadvantages

- Data Quality Sensitivity: Inaccuracies in historical data, missing values, or outdated datasets can significantly affect prediction reliability.
- Model Generalization Limitation: The ML model may not generalize well to areas with drastically different traffic behaviors or underrepresented events (e.g., festivals, accidents).
- No Real-Time Adaptation: Without live data feeds, the system cannot adjust predictions based on current conditions (e.g., construction or weather).

- Maintenance Overhead: Requires periodic updates, retraining, and system monitoring to maintain prediction accuracy over time.
- Dependence on Internet Access: For cloud-based deployment, users must have a stable internet connection to interact with the system.

9. CONCLUSION

The "Traffic Telligence" project validates the ability of machine learning in addressing realworld challenges like traffic congestion. With the help of structured data pipelines, robust ML models, and a clean UI, the project achieves its core goal—accurate prediction of traffic volume using historical datasets.

The system proves particularly useful in cities struggling with high traffic density, where predictive tools can help manage peak loads, reduce travel time, and enhance commuter experience. Model evaluation using MSE and R^2 reveals strong predictive power and reliability.

This project not only demonstrates technical competence in machine learning and web deployment but also reflects its potential societal impact. It bridges the gap between advanced analytics and public utility by offering a practical, low-cost solution for traffic forecasting.

Future iterations can enhance this foundation by integrating real-time capabilities and expanding predictive features, showing promise for large-scale adoption in smart city ecosystems.

10. FUTURE SCOPE

- Integration with Real-Time APIs: Incorporating live feeds from traffic APIs (e.g., Google Maps Traffic, Waze) can significantly improve prediction accuracy and adaptability.
- Weather & Event-Based Prediction: Enhancing the model to consider external factors such as weather conditions, road closures, and public events for better contextaware forecasts.
- Cross-Platform Compatibility: Developing dedicated Android/iOS applications to make the tool more accessible to daily commuters and on-field personnel.

- Machine Learning Model Expansion: Exploring deep learning techniques (LSTMs, CNNs for spatiotemporal analysis) could further improve the accuracy of time-series traffic data prediction.
- AI-Powered Decision Support System: Creating an analytics dashboard for

government bodies to simulate traffic scenarios and receive AI-suggested interventions.

- Community Feedback Loop: Allowing user-submitted traffic updates could create a hybrid system that combines AI predictions with real-world insights.

11. APPENDIX

Source Code

The complete source code for the traffic volume estimation project is available in the GitHub repository linked below. It includes data preprocessing scripts, model training code, prediction modules, and the Streamlit UI components.

Uploaded in GitHub repo:

https://github.com/kavya-kurmala/SmartBridge_project

Dataset Link

We utilized a publicly available dataset containing historical traffic volume, date and time stamps, and corresponding geolocations. This dataset was preprocessed to extract timebased features and fed into the ML model to improve prediction accuracy.

Used traffic volume dataset from:

https://github.com/kavya-kurmala/SmartBridge_project/blob/main/traffic%20volume.csvGitHub & Project Demo Link:

GitHub Repository:

https://github.com/kavya-kurmala/SmartBridge_project

Demo Link (Streamlit):

https://github.com/kavya-kurmala/SmartBridge_project/blob/main/README.md