TrafficTelligence: Advanced Traffic Volume Estimation with Machine Learning

1. Introduction

Project Title: TrafficTelligence: Advanced Traffic Volume Estimation with Machine Learning

Team ID: LTVIP2025TMID41765

Team Size: 4

• **Team Leader**: Navuru Akshitha - Project Coordination, Milestone Tracking, ML Model Development & Flask Integration

• **Team Member**: Guttla Sruthi – Data Collection, Cleaning, preprocessing pipeline

• **Team Member:** I Asha – Define Problem / Problem Understanding, project setup & Infrastructure

• **Team Member:** Kaki Dona – UI/UX Design, Documentation, Descriptive Statistics

2. Project Overview

2.1. Purpose

Traffic congestion is a significant challenge in modern urban areas, leading to delays, fuel wastage, and environmental impact. This project aims to develop a machine learning model that accurately estimates traffic volume using various sensor and environmental features. By analyzing these features, the system can support smarter traffic management, planning, and control strategies in real-time.

2.2. Features

• User input form with 41 traffic-related features

• Prediction using a trained Random Forest model

• Result displayed in a user-friendly web interface

• Error handling for missing or invalid data

• Instantly returns a prediction label – “High Traffic Volume” or “Low Traffic Volume” – after form submission

3. Architecture

3.1. Frontend

• Developed using HTML5, CSS3, and Jinja2 templates

• HTML form (index.html) for inputting 41 traffic-related features

• Styled using embedded CSS and images served from the /static directory

• Features a clean, responsive UI with gradient backgrounds and conditionally rendered prediction results

3.2. Backend

• Backend logic implemented using Python and Flask

• Model Training Pipeline:

• Model training, evaluation, and export handled in Google Colab

• Scripts for data cleaning, feature engineering, and training were executed in Colab

• Final .pkl files (model + tools) were trained and exported in Colab, then used in the Flask backend for real-time predictions

• **Model:**

A Random Forest Regressor trained on a traffic volume dataset. Model artifacts include:

• model.pkl: Trained model

• encoders.pkl: LabelEncoder instances

• normalizer.pkl: MinMaxScaler instance

3.3. Data Preprocessing

• Normalization with MinMaxScaler

• Label Encoding for categorical variables

• Feature extraction and transformation for traffic data such as time-of-day, weather conditions, and sensor types

• Encoders and scalers saved as .pkl files

3.4. Database

• No persistent database is used

• Prediction is done entirely in-memory based on form input

4. Setup Instructions

4.1. Prerequisites

• Python 3.10+

• Flask

• scikit-learn

• pandas, numpy

• Jupyter Notebook

4.2. Installation

git clone https://github.com/Akshitha1805/TrafficTelligence-Advanced-Traffic-Volume-Estimation-with-Machine-Learning

cd TrafficTelligence/Flask

pip install -r requirements.txt

python app.py

5. Folder Structure

5.1. Client (Flask Frontend)

```
├── templates/
│   └── index.html    # Main HTML form for user input
├── static/
│   └── images/
│       └── traffic-bg.jpg
│       └── low_traffic.png
```

| └── high_traffic.png

5.2. Server (Flask Backend)

├── app.py # Flask application script

├── train_model.py # Script for preprocessing data and training ML model

├── model.pkl # Trained Random Forest model

├── normalizer.pkl # Saved MinMaxScaler

├── encoders.pkl # Dictionary of saved label encoders

├── Traffic.ipynb # Notebook for model development and training in Colab

6. Running the Application

• Start the app:

cd Flask

python app.py

• Open in browser at: http://127.0.0.1:5000/

7. API Documentation

• Route: / (root URL)

• Method: POST

• Inputs: 41 traffic-related features such as time of day, weather, and sensor data

• Processing:

- Input is encoded using encoders.pkl

- Normalized using normalizer.pkl

- Sent to model.pkl for prediction

• Output:

- HTML page displaying prediction: “High Traffic Volume” or “Low Traffic Volume”

- Color-coded message with icons

• Error Handling:

- User-friendly errors for invalid or missing input

8. Authentication

Not applicable – This application does not implement user login or authentication as it's a prototype for traffic prediction.

9. User Interface

• Form with 40+ traffic-related input fields

• Background with traffic-themed image

• Result shown using color-coded icons and messages

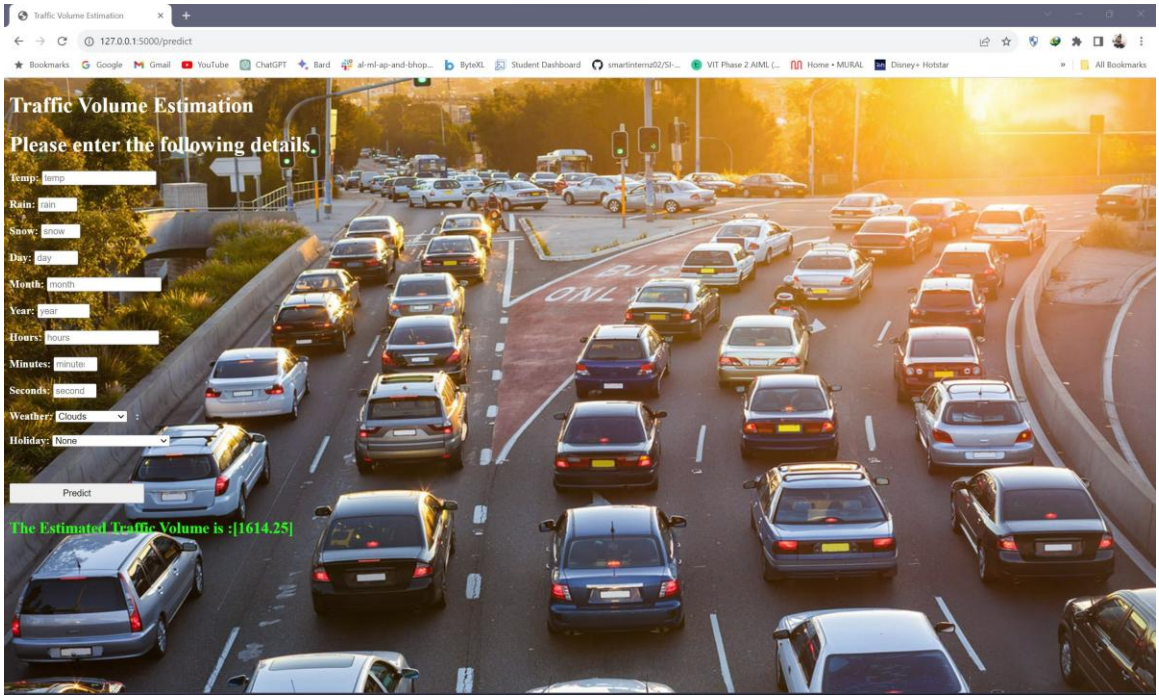
10. Testing

• Model accuracy validated using scikit-learn

• Manual testing for form input handling

• 80/20 stratified data split used for model validation

11. Screenshots or Demo



Demo Link: DEMO VIDEO.mp4

12. Known Issues

- Requires specific input formats
- No database or session tracking
- Limited interpretability of model results

13. Future Enhancements

- Add database to store inputs and predictions
- Generate downloadable PDF reports
- Improve accuracy using more data
- Integrate with city traffic systems
- Add login, history tracking, mobile support