TrafficTelligence: Advanced Traffic Volume Estimation with Machine Learning

Project Documentation Format

1. Introduction

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

Team Members:

Baireddy Mahendra Reddy [22HM1A0506]

Boya Nikhil Naidu [23HM5A0503]

Guduru Vyshnavi [22HM1A0538]

Bandi Kavitha [22HM1A0508]

2. Project Overview

Purpose:

The purpose of the TrafficTelligence project is to develop an intelligent system that can accurately estimate and predict traffic volume using machine learning techniques. This helps in making better decisions for traffic control, urban planning, and commuter convenience by analyzing various factors like historical traffic data, weather, and events. The goal is to reduce congestion, improve road safety, and support the development of smart cities.TrafficTelligence is a smart system designed to estimate and predict traffic volume using machine learning. It uses data from the past, like how many vehicles were on the road at different times, to understand traffic patterns. It also looks at things like weather, holidays, and local events that might affect traffic. By studying all these factors together, the system can give accurate predictions about how busy the roads will be. This helps traffic managers make better decisions and reduce traffic jams.

Goals:

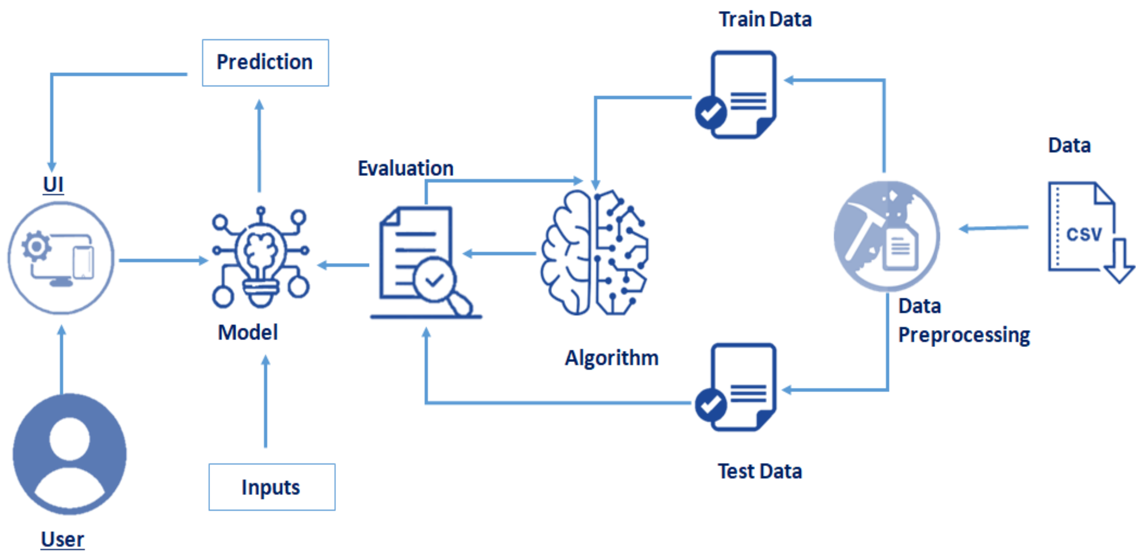
- Reduce traffic congestion and improve the flow of vehicles.
- Monitor traffic in real time using smart sensors and data.
- Improve road safety by detecting accidents and hazards quickly.
- Optimize traffic signals to reduce waiting time at intersections.
- Support emergency vehicles by giving them clear routes.
- Collect and analyze traffic data to make better decisions.
- Provide live traffic updates to drivers and city authorities.
- Lower fuel use and pollution by minimizing idle time.

Key Features:

1. **Real-Time Traffic Monitoring**
   Continuously tracks vehicle flow and traffic conditions using sensors or cameras.
2. **AI-Powered Traffic Management**
   Uses artificial intelligence to analyze data and make smart traffic decisions.
3. **Smart Signal Control**
   Automatically adjusts traffic light timings based on real-time traffic density.
4. **Data Collection & Analysis**
   Gathers traffic data to identify patterns, peak hours, and problem areas.
5. **Accident & Incident Detection**
   Detects accidents, roadblocks, or unusual activity and sends instant alerts.
6. **Live Traffic Updates**
   Provides live updates and route suggestions to drivers via mobile or dashboard apps.
7. **Emergency Vehicle Priority**
   Gives priority to ambulances, fire trucks, and police vehicles through dynamic signal control.

8. **Eco-Friendly Approach**
   Reduces fuel consumption and emissions by minimizing idle time and traffic jams.
9. **Cloud Integration**
   Stores and processes traffic data on the cloud for easy access and scalability.
10. **Scalable for Smart Cities**
    Easily adaptable and expandable for use in different city sizes or conditions.

3. Architecture:



4. Setup Instructions:

Prerequisties:

To complete this project, you must require the following softwares and packages.

Software Requirements:

Visual Studio Code(VS Code) or any Python supported IDE

Annaconda navigator (Jupyter notebook) for suitable for all packages

Python Packages:

To install the python packages on VS Code follow these steps:

Open anaconda prompt as administrator.

- Type “pip install numpy” and click enter.
- Type “pip install pandas” and click enter.
- Type “pip install matplotlib” and click enter.
- Type “pip install scikit-learn” and click enter.
- Type “pip install Flask” and click enter.
- Type “pip install xgboost” and click enter.

Installation:

1. Create a Virtual Environment (Optional but Recommended):
   ```
   python -m venv .venv
   source .venv/Scripts/activate    For Windows
   source .venv/bin/activate        For Mac/Linux
   ```
2. Install Required Packages:
   Ensure you run:
   ```
   pip install numpy
   pip install pandas
   pip install scikit-learn
   pip install matplotlib
   pip install scipy
   pip install seaborn
   pip install tensorflow
   pip install Flask
   pip install Pillow
   ```
3. Download Dataset:
   Place the dataset inside the `data` folder or run your `dataset_downloader.py` to fetch from Kaggle.
4. Prepare the Dataset:
   ```
   python data_preparation.py
   ```
5. Clean the Dataset (Optional for Transparency Issues):
   ```
   python rgb_cleaner.py
   ```

6. Train the Model:

```
python model_training.py
```

7. Test the Model (Optional):

```
python model_testing.py
```

8. Run the Flask Web Application:

```
python app.py
```

9. Access the Application:

   Open your browser and visit: [http://127.0.0.1:5050]

.

6. Running the Application

A. Backend (Flask Application):

The Flask backend serves both the API and the frontend templates (HTML pages).

Commands to run the backend:

Open the terminal (preferably inside your project directory).

Activate the virtual environment if created:

```
source .venv/Scripts/activate   # For Windows

source .venv/bin/activate       # For Mac/Linux
```

Run the Flask application:

```
python app.py
```

The server will start at:

```
http://127.0.0.1:5050
```

B. Frontend:

The project uses Flask's Jinja templates (index.html, about.html, inspect.html) for the frontend. No separate command is needed for the frontend. Once the Flask server starts, the frontend will be accessible through your web browser at:

http://127.0.0.1:5050

The following endpoints are exposed by the Flask backend for the application:

7. API Documentation

a) **Home Page**

- **URL:** /about
- **Method:** GET
- **Description:** Displays the landing page of the application.
- **Response:** Returns the index.html template.

b) **About Page**

- **URL:** /about
- **Method:** GET
- **Description:** Displays information about the project.
- **Response:** Returns the about.html template.

8. Authentication

Current Status:

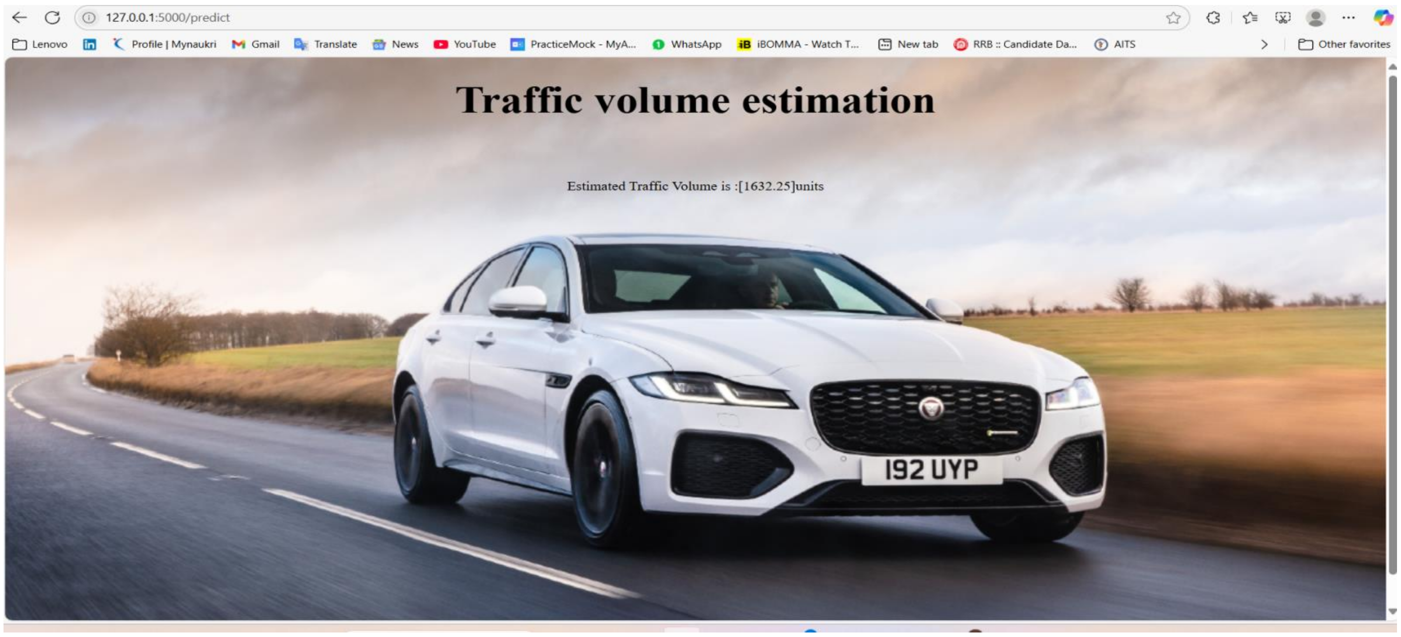
The current version of the project does not implement authentication or authorization mechanisms. The application is designed as a publicly accessible image classification tool intended for demonstration purposes, allowing any user to:

✓ Access the website

✓ View results without requiring login or registration

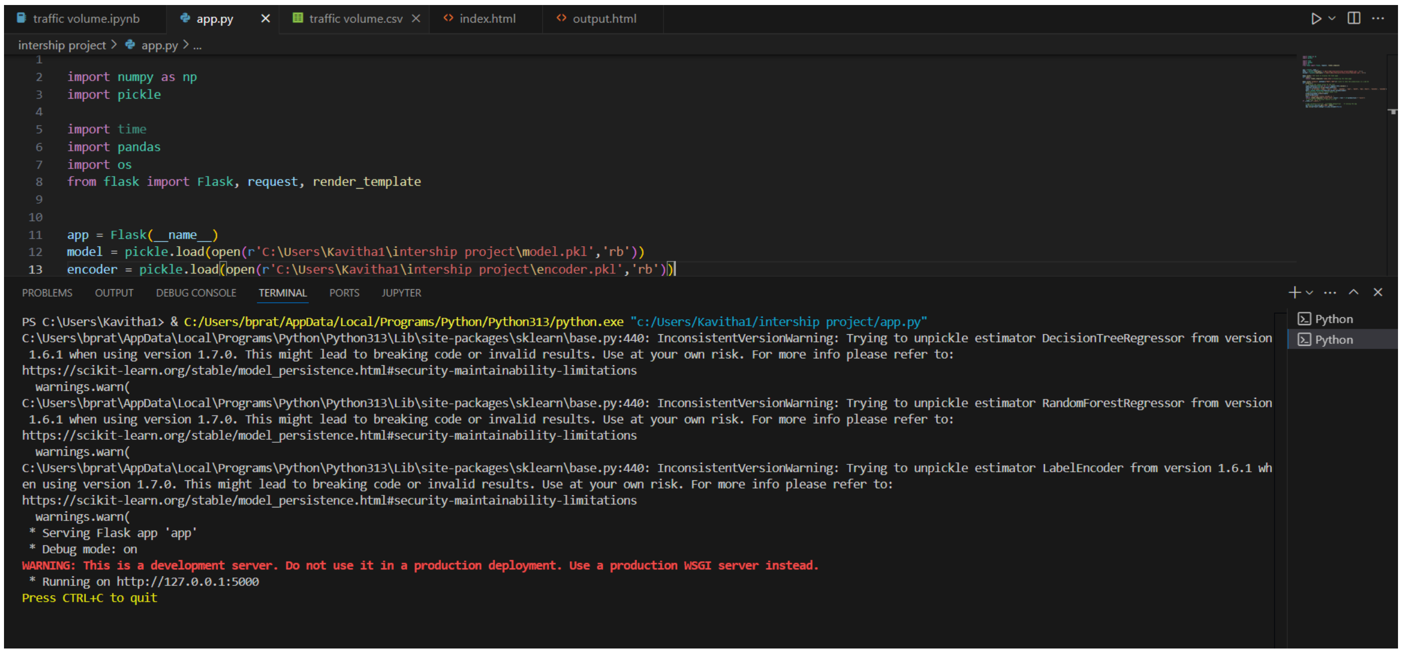
1. **Username & Password** – Basic login system for accessing dashboards or control panels.
2. **Two-Factor Authentication (2FA)** – Adds extra security using SMS, email, or authenticator apps.
3. **Biometric Authentication** – Uses fingerprint, face recognition, etc., for high-security roles.
4. **API Key Authentication** – For secure communication between systems (e.g., traffic sensors and servers).
5. **Role-Based Access Control (RBAC)** – Different access levels for admin, users, maintenance staff, etc.

9.User Interface



Estimated Traffic Volume is :[1632.25]units

10. Testing



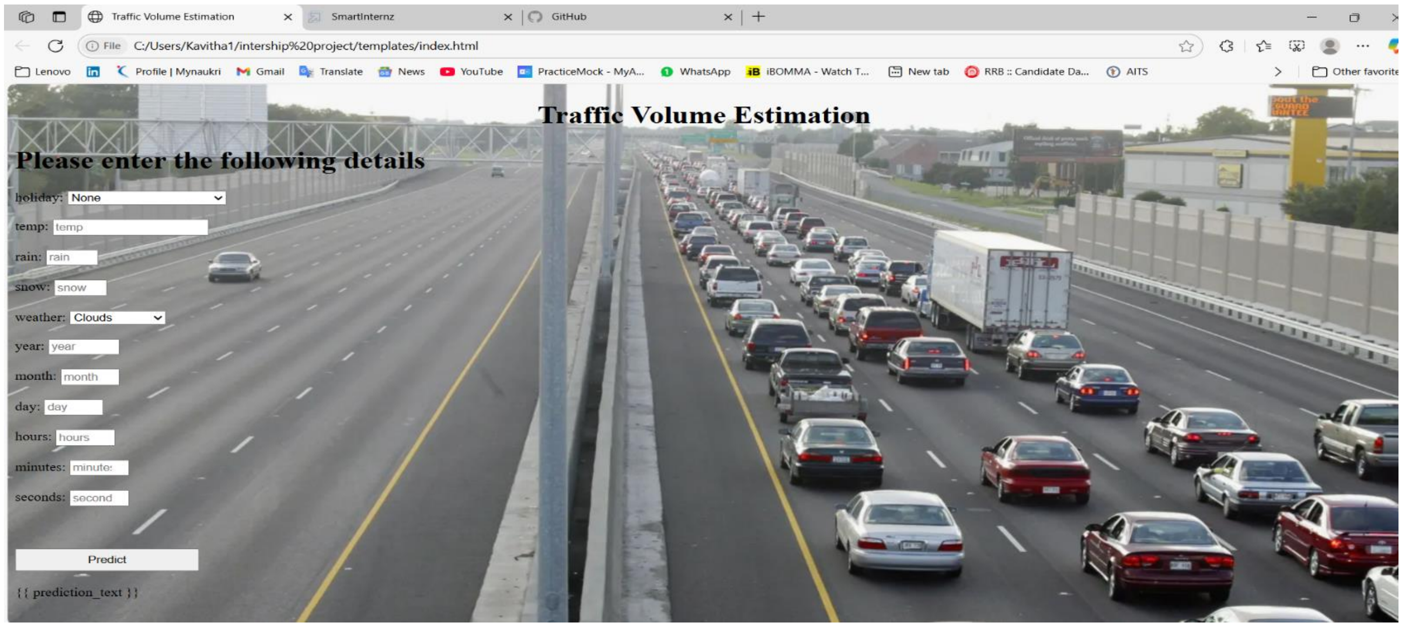
```python

import numpy as np
import pickle

import time
import pandas
import os
from flask import Flask, request, render_template


app = Flask(__name__)
model = pickle.load(open(r'C:\Users\Kavitha1\intership project\model.pkl','rb'))
encoder = pickle.load(open(r'C:\Users\Kavitha1\intership project\encoder.pkl','rb'))
```

```
PS C:\Users\Kavitha1> & C:/Users/bprat/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/Kavitha1/intership project/app.py"
C:\Users\bprat\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\base.py:440: InconsistentVersionWarning: Trying to unpickle estimator DecisionTreeRegressor from version
 1.6.1 when using version 1.7.0. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
C:\Users\bprat\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\base.py:440: InconsistentVersionWarning: Trying to unpickle estimator RandomForestRegressor from version
 1.6.1 when using version 1.7.0. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
C:\Users\bprat\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\base.py:440: InconsistentVersionWarning: Trying to unpickle estimator LabelEncoder from version 1.6.1 wh
en using version 1.7.0. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```



Traffic Volume Estimation

Please enter the following details

holiday: None

temp: temp

rain: rain

snow: snow

weather: Clouds

year: year

month: month

day: day

hours: hours

minutes: minutes

seconds: second

Predict

{ { prediction_text } }

11. Screenshots or Demo

Traffic volume estimation

{ {result} }

Demo Link:
https://drive.google.com/file/d/1tAIXAQQCtEHAkc3Mx_p1KqmW1kSUfpHd/view?usp=sharing

12.Known Issues

1. **Sensor/Data Inaccuracy**
   Inconsistent or incorrect data due to faulty sensors, weather interference, or low visibility.
2. **Network Connectivity Issues**
   Real-time traffic systems rely on stable internet; poor connectivity can delay data transmission.
3. **Data Privacy and Security Concerns**
   Collecting real-time traffic and vehicle data may raise privacy or cybersecurity issues.
4. **AI Model Limitations**
   AI may struggle with unusual traffic patterns or unexpected events not present in training data.
5. **Power Dependency**
   The system may fail during power outages if not backed by reliable power sources.
6. **High Initial Setup Cost**
   Installing sensors, cameras, and servers can be expensive and may not be affordable for all cities.
7. **Integration Challenges**
   Difficulty integrating with existing infrastructure, like old traffic lights or outdated road systems.
8. **Limited Accuracy in Low-Traffic Areas**
   In areas with less traffic, the system might not gather enough data to function effectively.
9. **User Resistance or Lack of Awareness**
   Drivers or authorities might not use or trust the system unless properly trained or informed.
10. **Need for Continuous Testing and Updates**
    Regular updates and monitoring are needed to keep the system reliable and effective.

13. Future Enhancements

□ **Vehicle-to-Infrastructure (V2I) Communication**
Enable direct communication between vehicles and traffic systems for faster, smarter responses.

 Advanced AI and Machine Learning
Use deep learning models to better predict traffic patterns, accidents, and peak congestion.

 Driver Mobile App Integration
Develop a dedicated app to give drivers personalized route suggestions, traffic alerts, and eco-driving tips.

 GPS and Navigation System Integration
Connect with popular GPS systems (like Google Maps or Waze) to improve route guidance and traffic flow.

 Computer Vision for Traffic Analysis
Use AI-powered cameras to recognize vehicle types, detect rule violations, and track pedestrian movement.

 Emergency Routing System
Add smart routing for ambulances, fire trucks, and police vehicles to reduce emergency response times.

 Multi-City/Regional Control System
Expand to manage traffic across multiple cities or regions using a centralized control center.

 Solar-Powered IoT Devices
Use solar energy to power sensors and devices, reducing electricity costs and increasing sustainability.

 Blockchain for Secure Data Sharing
Implement blockchain to securely share traffic data between agencies and maintain transparency.

 Voice Assistant Integration
Add voice-based traffic information for drivers using smart assistants like Alexa or Google Assistant.