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:**
  1. Yalavarthi Sireesha
  2. Shaik Abdulgani Mohammedsabir
  3. Shaik Shoaib Malik
  4. Shaik Irfan

2. Project Overview

- ### Purpose:

The purpose of the TrafficTelligence project is to design and develop a machine learning–based web application that accurately predicts traffic volume based on time, date, weather conditions, and holiday data. This system aims to assist urban commuters, traffic authorities, and smart city planners in making informed, proactive decisions to reduce congestion, improve travel efficiency, and optimize traffic management resources.

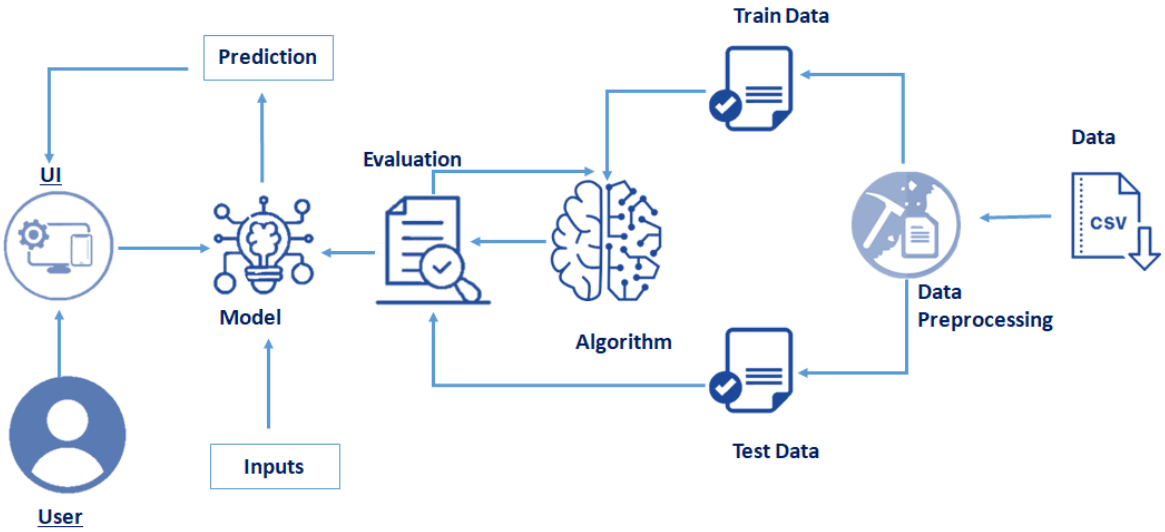
Traffic congestion is a persistent issue in urban areas, often leading to delays, fuel wastage, and increased stress among commuters. The TrafficTelligence project addresses this challenge by leveraging historical traffic data and contextual factors such as weather and holidays to train a predictive machine learning model. Integrated into a user-friendly web application, this tool allows users to enter specific conditions and receive real-time traffic volume estimates. The ultimate goal is to support smarter travel planning, enhance public safety, and contribute to more intelligent transportation systems.

- ### Goals:

- Build a machine learning model (e.g., RandomForestRegressor) that accurately predicts traffic volume based on inputs such as weather, holiday, date, and time.
- Provide a simple and responsive web interface using Flask that allows users to input relevant conditions and receive predictions in real time.
- Enable traffic authorities or planners to use prediction outputs for resource allocation and congestion planning.
- Design the system to be easily expandable with new features like heatmaps, charts, or API integration. Prevent app crashes by validating all inputs and providing clear feedback for errors.

- **Features:**
  - **Traffic Volume Prediction:** Predicts traffic volume using date, time, holiday, and weather as inputs.Built using RandomForestRegressor for high accuracy.
  - **Date & Time-Based Input:** Users can select specific Date (year, month, day) Time (hours, minutes, seconds) Enables future traffic planning.
  - **Weather & Holiday Integration:** Factors like weather conditions (e.g., Rain, Snow, Clear) and holidays (e.g., Columbus Day, Christmas) are encoded and used for prediction.
  - **ML Model Integration:** Uses trained and saved model **(**model.pkl**)** and OneHotEncoder **(**encoder.pkl**)** for consistent prediction logic.
  - **Web-Based UI (Flask):** Simple and interactive form for users to input required parameters Instant output after clicking Predict.
  - **Scalable Architecture:** Can be extended to include Live weather APIs Real-time location data Visualizations like traffic maps or line graphs.
  - **User-Friendly Web Interface:** Clean and simple interface built using HTML + Flask Users can enter inputs through a web form Instant output display after submission.
  - **Result Display:** Predicted traffic volume shown clearly.Optional Result can be visualized with color or graphs (e.g., light/medium/heavy).

3. Architecture



4. Setup Instructions

- **Prerequisites:**

To complete and run the TrafficTelligence project, the following software and Python packages are required:

- **Software Requirements:**
  - Visual Studio Code (VS Code) or any Python-supported IDE
  - Python 3.10 (recommended for best compatibility with all packages)
- **Python Packages Installation:**
  - Open your VS Code terminal or command prompt and run the following commands:
  - pip install numpy
  - pip install pandas
  - pip install scikit-learn
  - pip install matplotlib
  - pip install scipy
  - pip install seaborn
  - pip install Flask

Note: You do not need tensorflow or Pillow for this project unless you add advanced ML or image processing in the future.

- ## Installation & Setup Instructions:

1.Create a Virtual Environment (Recommended):

- python -m venv .venv
- .venv\Scripts\activate (For Windows)
- source .venv/bin/activate (For Mac/Linux)

2. Install Required Packages:

- pip install numpy
- pip install pandas
- pip install scikit-learn
- pip install matplotlib
- pip install scipy
- pip install seaborn
- pip install Flask

3. Prepare the Dataset:

- Place **traffic volume.csv** in your project root or data/ folder.

This dataset should contain relevant features like:

- holiday, weather, temp, rain, snow, year, month, day, hours, minutes, seconds, and traffic_volume.

4. Preprocess & Train the Model:

Use your preprocessing and training script (e.g., train_model.py) or run your notebook:

- python train_model.py

- Apply **OneHotEncoding** to categorical features
- Train a **RandomForestRegressor**
- Save:
    - model.pkl
    - encoder.pkl

5. Run the Flask Web Application:

- Make sure **app.py** is in the project root, then run:
    - python app.py

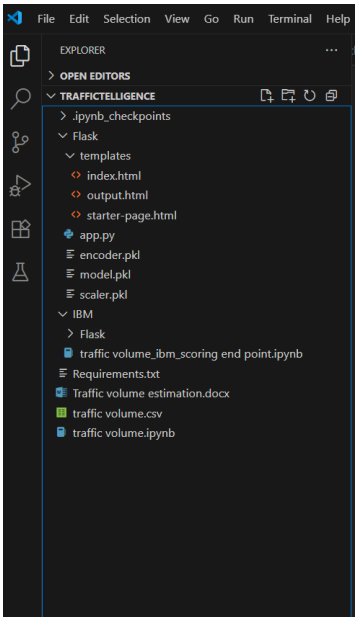
6. Access the Web App:

Open your browser and go to:

- http://127.0.0.1:5000

You can now input date, time, weather, and holiday info to predict the traffic volume.

5. Folder Structure



6. Running the Application

A. Backend (Flask Application):

The Flask backend serves:

- The ML prediction API for traffic volume
- The **frontend templates** (HTML pages) like **`index.html`**

Steps to Run the Backend:

1. Open terminal (inside your project directory).
2. Activate your virtual environment (if created):

   - **source .venv/Scripts/activate # For Windows**
   - **source .venv/bin/activate # For Mac/Linux**

3. Run the Flask application:

   - python app.py

If everything is set up correctly, the server will start at:

- http://127.0.0.1:5000

B. Frontend:

The project uses Flask's **Jinja2 HTML templates** for the user interface:

- `index.html` – Main prediction form and results display
- (Optional future pages like `about.html`, `visualize.html`)

No separate command is required to start the frontend.

Once the Flask server is running, open your browser and visit:

- http://127.0.0.1:5000

You can input:

- Date & Time
- Holiday
- Weather
- Temperature, Rain, Snow

Then submit to get a **real-time traffic volume prediction**.

7. API Documentation

a)Home Page:

- **URL**: /
- **Method**: GET
- **Description**: Displays the landing page of the application, where users can input traffic-related data (weather, holiday, time, etc.) for prediction.
- **Response**: Returns the index.html template (form interface for traffic prediction).

b)About Page *(optional)*

(Include if you create an `about.html` page)

- **URL**: `/about`
- **Method**: `GET`
- **Description**: Provides information about the TrafficTelligence project, its purpose, and how it works.
- **Response**: Returns the `about.html` template.

c)Prediction Page

- **URL**: `/predict`
- **Method**: `POST`

- **Description**: Accepts user input from the form, processes the data (encoding + model prediction), and returns the predicted traffic volume.
- **Response**: Renders the same `index.html` page with the prediction result embedded.

Name	Type	Description
Holiday	String	Name of the holiday (e.g. "Christmas Day")
Weather	String	Weather condition (e.g. "Clear", "Rain")
Temp	Float	Temperature value
Rain	Float	Rain level
Snow	Float	Snow level
Year	Int	Year component of date
Month	Int	Month component of date
Day	Int	Day component of date
Hours	Int	Hour of the day
Minutes	Int	Minute of the hour
Seconds	Int	Seconds
Month	Int	Month component of date
Day	Int	Day component of date
Hours	Int	Hour of the day

Example Request Using HTML Form:

```
    <form method="POST" action="/predict">
<label>Holiday:</label>
<input type="text" name="holiday" required>

<label>Weather:</label>
<input type="text" name="weather" required>

<label>Temperature:</label>
<input type="number" step="any" name="temp" required>
```

```
  <label>Rain:</label>
  <input type="number" step="any" name="rain" required>

  <label>Snow:</label>
  <input type="number" step="any" name="snow" required>

  <label>Year:</label>
  <input type="number" name="year" required>

  <label>Month:</label>
  <input type="number" name="month" required>

  <label>Day:</label>
  <input type="number" name="day" required>

  <label>Hour:</label>
  <input type="number" name="hours" required>

  <label>Minute:</label>
  <input type="number" name="minutes" required>

  <label>Second:</label>
  <input type="number" name="seconds" required>

  <button type="submit">Predict</button>
</form>
```

Example Response (Rendered on index.html Page):

Upon successful prediction, the following output is shown:

- User-submitted inputs (holiday, weather, time, etc.)
- **Predicted traffic volume** (e.g., Traffic volume is estimated at 3875 vehicles)
- Optionally, traffic condition tag (e.g., *"Moderate"*, *"Heavy"*, *"Low"*)

Example Backend Response (if Converted to JSON API):

If your app were using an API endpoint and returning a JSON response instead of rendering HTML:


```
{
 "predicted_traffic_volume": 3875,
 "traffic_condition": "Moderate",
 "input_summary": {
  "holiday": "Columbus Day",
  "weather": "Clear",
  "temp": 17.5,
  "rain": 0.0,
  "snow": 0.0,
  "year": 2025,
  "month": 6,
  "day": 26,
  "hours": 14,
  "minutes": 30,
  "seconds": 0
 }
}
```


8. Authentication

Current Status:

The current version of the **TrafficTelligence** project does **not implement authentication or authorization mechanisms**.
It is designed as a **publicly accessible traffic volume prediction tool** intended for **demonstration and academic purposes**, allowing any user to:

- ✔ Access the website
- ✔ Input weather, date, and time details for traffic prediction
- ✔ View traffic volume predictions without login or registration

Future Scope for Authentication (Optional Enhancements):

To enhance **security** and **user-specific features**, the following authentication and authorization methods can be added in future versions:

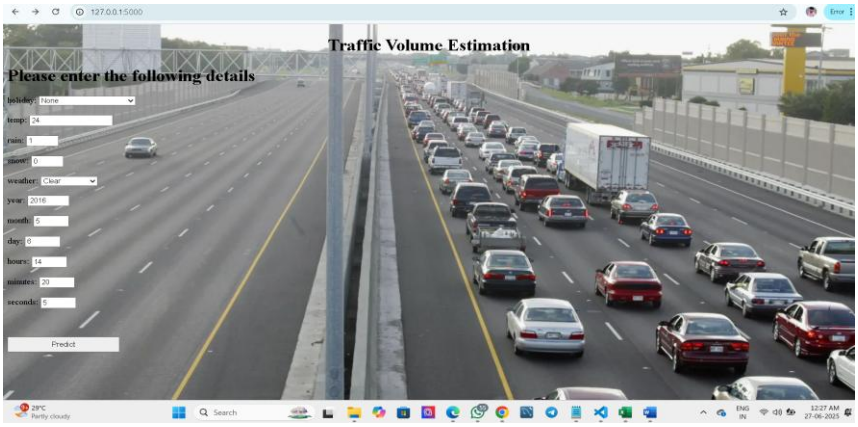
Method	Description
Session-Based Authentication	User credentials are verified at login, and a secure session is maintained using cookies.
Token-Based Authentication (JWT)	Users receive a JSON Web Token upon successful login, which must be sent with each request to access protected endpoints (e.g., `/predict`).
Role-Based Access Control (RBAC)	User roles like *Admin*, *Traffic Officer*, *City Planner* can be defined to restrict access to sensitive features (e.g., model retraining or analytics).

Recommended Future Features:

- ❖ **Admin Login**: Only authorized city planners or admins can upload new data or retrain the model.
- ❖ **User Dashboard**: Logged-in users can view and track prediction history over time.
- ❖ **API Access Tokens**: Enable secure, token-protected access for mobile apps or third-party platforms integrating the prediction system.

❖ **Login via Gmail/OAuth**: Allow easy access using secure Google authentication for trusted users.

9. User Interface



10. Testing

Model Evaluation Testing

- Evaluated the trained **RandomForestRegressor** using separate **train-test splits** of the dataset.
- Generated performance metrics such as:
  - **Mean Absolute Error (MAE)**
  - **Root Mean Squared Error (RMSE)**
  - **R² Score**

- Ensured the model generalizes well to unseen traffic data (weather + time-based).

External Data Testing

- Manually tested the model by modifying inputs like holidays, weather conditions, and different time combinations **not used in training**.
- Observed predictions to evaluate how well the model generalizes to real-world-like combinations.

Web Application Functional Testing

- Verified that all pages (Home, Predict) render correctly via the browser.
- Confirmed form inputs accept valid data and reject invalid types.
- Ensured:
    - All fields are required
    - Backend handles missing fields gracefully
    - Correct traffic volume prediction is shown
- Tested end-to-end flow: input → predict → output display

Tool / Library	**Purpose**
scikit-learn	Model training, evaluation, and performance metrics (MAE, R^2, etc.)
pandas, numpy	Data manipulation and statistical summaries
matplotlib, seaborn	Visualization of feature importance and prediction distribution
Flask (Debug Mode)	Real-time testing of frontend-backend integration
Manual Testing	Validated form functionality, prediction output, and user experience through the browser

11. Screenshots or Demo

The complete execution of the **TrafficTelligence** application is shown below through a series of step-by-step screenshots:

Step 1: Running the Flask Application

- Run the `app.py` file from your terminal or VS Code.
- You will see the Flask server starting message.
- The application will be accessible at:
    - **http://127.0.0.1:5000**

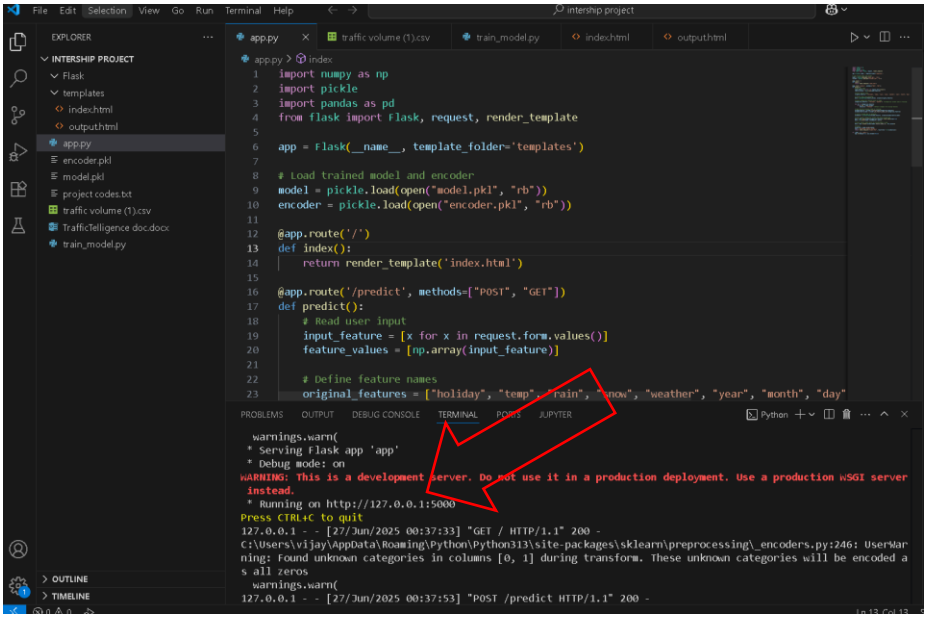


Fig 1:code running in terminal

Step 2: Opening the Application in Browser

Open your browser and navigate to:

- http://127.0.0.1:50000

The **Home Page** will appear with a clean and simple interface to input:

- Holiday name
- Weather condition
- Temperature, rain, snow values
- Date and time details

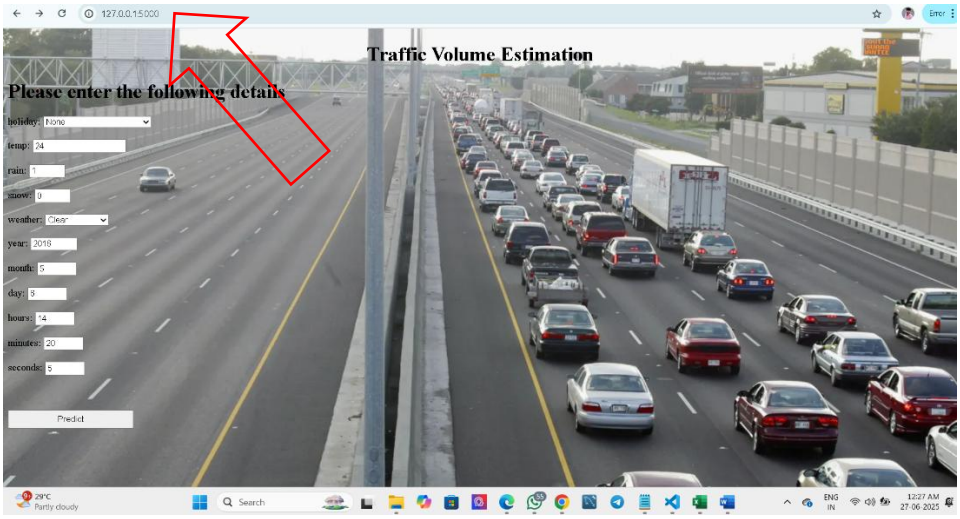


Fig 1.1:Traffic Volume Estimation Interface

Step 3: Submitting the Form for Prediction

- Fill in the input fields and click **Predict**.
- The backend processes the request using the trained model and encoder.

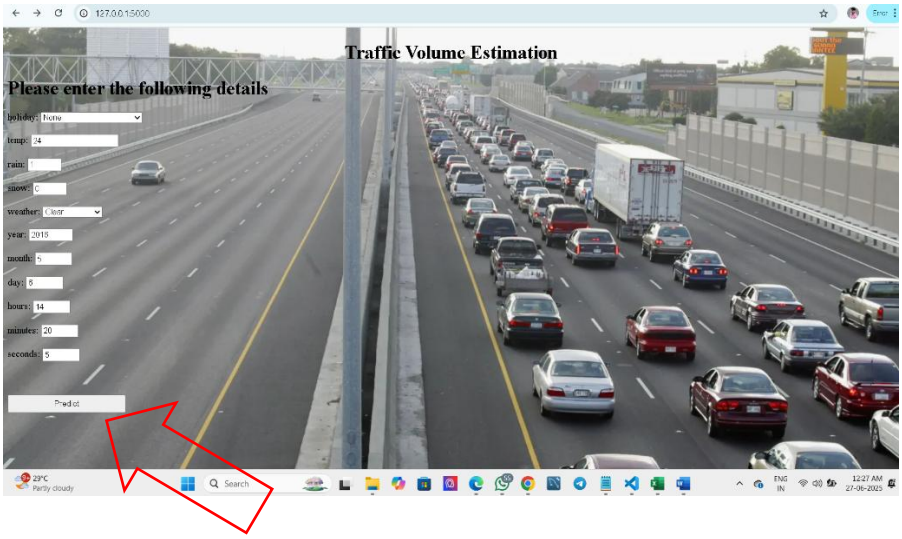


Fig 1.3:click predict option

Step 4: Viewing the Prediction Output

- The predicted traffic volume is displayed on the same page.
- This output is generated by the `model.pkl` and formatted with `Jinja2`.

- **Screenshot Example:**
- 📸 *Prediction result shown below the form: "Estimated traffic volume: 1373.94"*

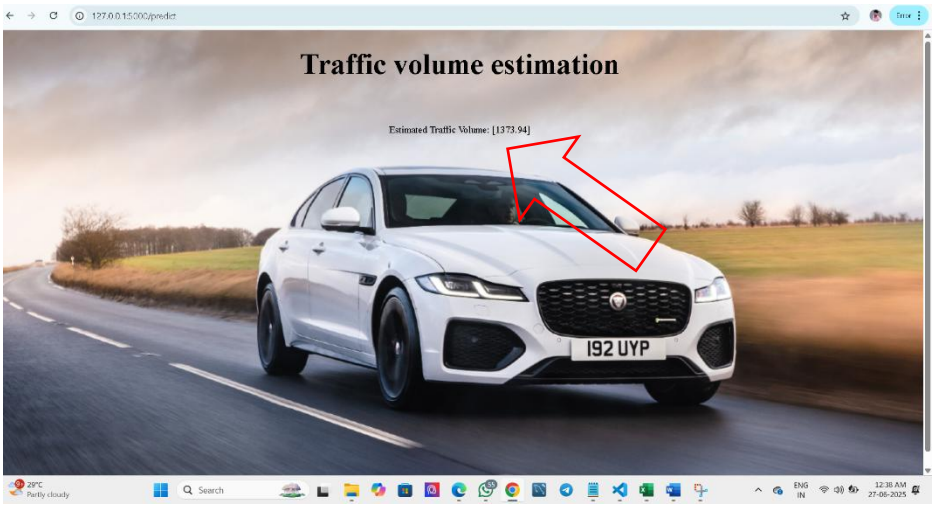


Fig 1.4:Predict the Estimated Traffic Volume

Project Demo Link:

12. Known Issues

Functional Issues

- **Inaccurate Predictions on Edge Cases**
  The model may provide inaccurate traffic volume predictions when:
  - Holidays are combined with extreme weather conditions
  - Rare or unseen holiday names are entered
  - Input patterns are significantly different from training data
- **Limited Generalization**
  The model may underperform on combinations it hasn't seen often during training, such as regional holidays or unusual weather-hour combinations.

Data Handling Issues

- **No Dropdown Selection for Categorical Inputs**
  The `holiday` and `weather` fields are free-text, which can lead to misspellings and invalid inputs that break the encoder.
- **No Input Error Handling**
  If any numeric or categorical field is left blank or incorrectly formatted, the app may crash or return an unhelpful error.

UI/UX Issues

- **No Labels for Predicted Volume**
  Users receive a numeric value (e.g., 3921 vehicles) without context. There's no traffic condition label such as *Low*, *Moderate*, or *High*.
- **Lack of Input Hints or Examples**
  Users are not guided with tooltips, placeholders, or examples for what to type in fields like "holiday" or "weather".
- **No Data Preview Before Submission**
  Users cannot review or confirm the inputs they entered before submitting for prediction.

Security Issues

- **Open Access Without User Verification**
  Anyone can access the prediction interface. There's no login, rate limiting, or access control.
- **No Input Sanitization or Validation**
  Inputs are directly passed into processing logic without sanitation, which may pose a risk in the future if the app connects to a database or logging system.

13. Future Enhancements

To increase the utility, accuracy, and scalability of the TrafficTelligence system, the following future enhancements are proposed:

Functional Improvements

- **Add Traffic Severity Labels**
  Classify predictions into categories such as *Low*, *Moderate*, *High*, or *Extreme* congestion for better user interpretation.
- **Visualization Dashboard**
  Display real-time prediction trends using interactive graphs (e.g., line charts, heatmaps, hourly histograms).
- **Location-Based Predictions**
  Integrate geolocation or city selection to offer area-specific traffic forecasts.

Machine Learning Enhancements

- **Model Upgrade with XGBoost or LSTM**
  Use advanced models like **XGBoost** or **time-series models (e.g., LSTM)** for better temporal predictions.
- **Model Retraining Portal**
  Build an admin-only interface for uploading new datasets and retraining models directly from the browser.

Frontend Enhancements

- **Auto Date/Time Picker**
  Replace manual input with a date-time selector to reduce user error.
- **Holiday & Weather Dropdowns**
  Add selectable dropdown menus with valid holiday names and weather conditions instead of free-text input.
- **Responsive Design**
  Make the web app mobile-friendly using Bootstrap or TailwindCSS.

Security & Access Control

- **User Authentication System**
  Implement login and registration features with Flask-Login or OAuth (Google, Facebook).
- **Role-Based Access Control (RBAC)**
  Assign roles like Admin, Analyst, or General User to restrict access to sensitive model training features.

Integration & Deployment

- **API Integration**
  Expose prediction endpoints via a RESTful API for integration with mobile apps or smart city platforms.
- **Cloud Deployment**
  Deploy the project on platforms like Render, Heroku, or AWS EC2 for public access.

- **Mobile App Companion (Optional)**
  Create a lightweight Android/iOS version to allow users to check traffic predictions on the go.