TrafficTelligence:
Smart Traffic Volume Estimation using Machine Learning

Submitted by:
Mummana venkata sai
Roll No: 23Q75A0218
Department of EEE
Avanthi Institute of Engineering and Technology

Team ID : LTVIP2025TMID37739

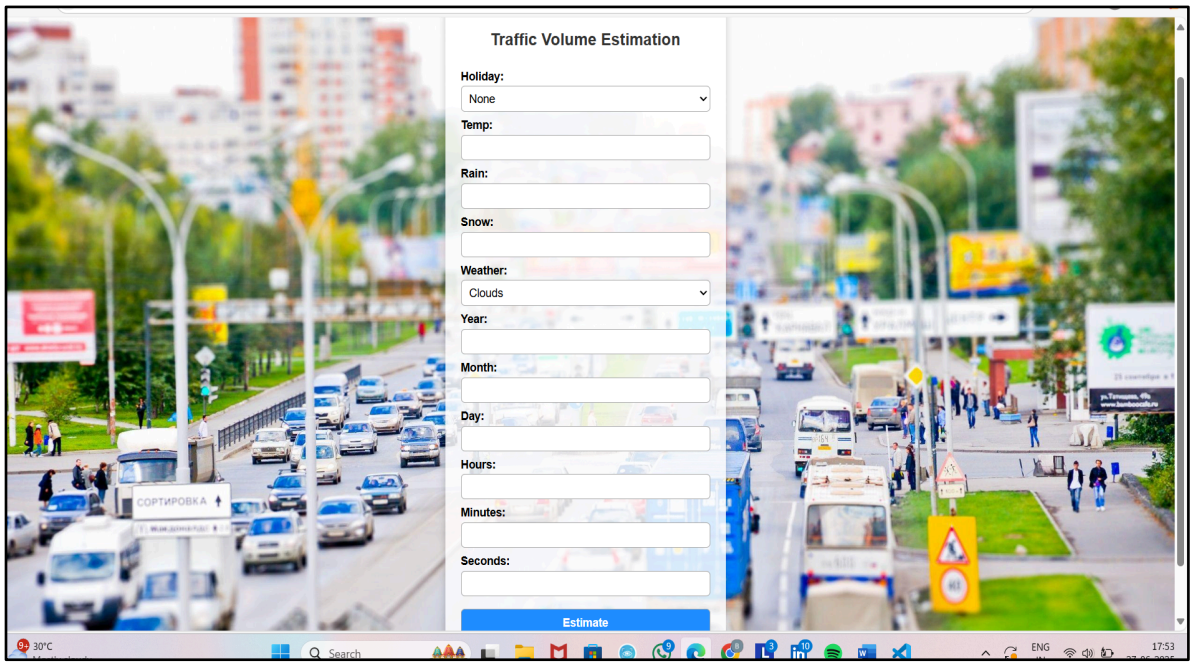
Introduction

This project focuses on developing an TrafficTelligence assistant using IBM Granite, TrafficTelligence is a system that uses machine learning to predict traffic volume. It takes information like weather, temperature, date, and time to estimate how much traffic might be on the roads. This helps reduce traffic jams and supports smart city planning.

Overview

TrafficTelligence is a machine learning-based system designed to estimate and predict traffic volume with high accuracy. It analyzes historical traffic data along with environmental and temporal factors (like weather, holidays, and time) to produce intelligent forecasts. This system supports traffic authorities, urban planners, and daily commuters in making smarter decisions.

Pre-requisites



To build and run this project, basic understanding of the following is required:

- Python programming
- Machine Learning concepts (regression, feature scaling, etc.)
- Flask web framework
- HTML/CSS basics

Required Python Packages

Install these packages using Anaconda Prompt or terminal:

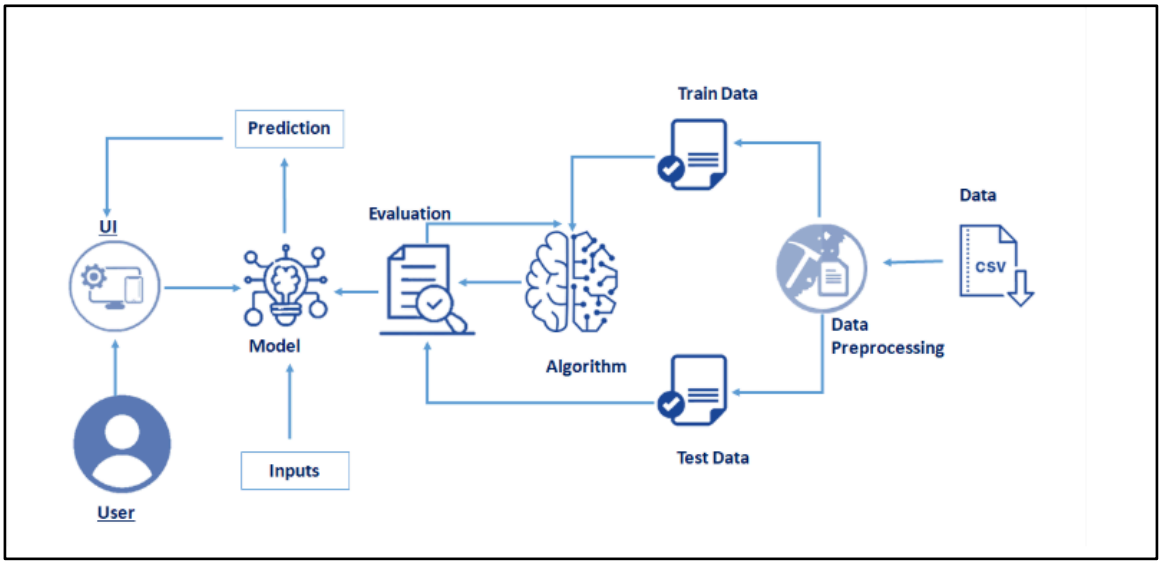
pip install numpy
pip install pandas
pip install matplotlib
pip install scikit-learn
pip install Flask
pip install xgboost

Project Objectives

- Identify if the problem is regression or classification.
- Clean and pre-process traffic datasets.
- Visualize data and extract insights.
- Apply and train suitable machine learning algorithms.
- Evaluate model accuracy.
- Integrate model with a web application using Flask.

Project Flow

1. User Input via Web Form



2. Input Processed by Machine Learning Model

3. Prediction Displayed on Web Interface

Project Structure

traffic_volume_app/

- app.py - model.pkl

- static/

- 3994546.jpg

- templates/

- index.html

Data Collection and Pre-processing

1. Collect Dataset: -

Use traffic data from public datasets (e.g., UCI, Kaggle).

2. Preprocessing Steps: -

Import libraries

- Load dataset using pandas

- Handle missing/null values

- Analyze data trends

- Normalize features with scaling

- Split data into training and testing sets

Model Building

- Use algorithms like Linear Regression or XGBoost

- Train model on historical traffic patterns

- Evaluate with metrics like Mean Absolute Error (MAE)

app.py > ...

```python
from flask import Flask, render_template, request

app = Flask(__name__)

# Simple dummy model: ju (parameter) features: Any puts
def dummy_model_predict(features):
    return int(sum(features) % 500)

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    try:
        inputs = [
            1 if request.form['holiday'] == "Holiday" else 0,
            float(request.form['temp']),
            float(request.form['rain']),
            float(request.form['snow']),
            1 if request.form['weather'] == "Clear" else 0,
            int(request.form['year']),
            int(request.form['month']),
            int(request.form['day']),
            int(request.form['hours']),
            int(request.form['minutes']),
            int(request.form['seconds']),
        ]
        prediction = dummy_model_predict(inputs)
        return render_template('index.html', result=prediction)
    except Exception as e:
        return render_template('index.html', result="Error: " + str(e))

if __name__ == "__main__":
    app.run(debug=True)
```

Application Building

- Frontend: Simple HTML form (index.html)

- Backend: Flask script (app.py)

- Output: Predicted traffic volume shown on screen

Example Scenarios

Scenario1: Dynamic Traffic Management Authorities use real-time predictions to optimize signal timings, reduce congestion, and manage lane control dynamically.

Scenario 2: Urban Planning Planners design road networks and public transport systems based on long-term traffic forecasts.

Scenario 3: Commuter Navigation Apps and drivers receive real-time predictions and reroute to avoid congested areas.

Technical Architecture

- Data Pipeline: Load and clean data

- Model: ML trained on historic + environmental features

- Web App: Flask handles requests and renders results

- UI: HTML + CSS with transparent design

Conclusion and Future Work

This project shows that we can use AI to predict traffic. In the future, we can use real-time data, show graphs, use a real model, and even create a mobile version.