ADITYA COLLEGE OF ENGINEERING AND TECHNOLOGY

Department of Artificial Intelligence and Machine Learning

Surampalem

INTERNSHIP REPORT

ON

"TrafficTelligence: Advanced Traffic Volume Estimation with Machine Learning"

Submitted in partial fulfillment of the requirements of the

Virtual Internship Program

Organized by

SMART BRIDGE

Submitted by

Jalla Shalini (23P31A42G3)

Medapati Sai Sri Harsha (23A91A61A0)

D. Venkatesh (23A91A0112)

Chamarti Deva Naga Sai(23P31A6173)

TEAM ID:

LTVIP2025TMID34566

Under the Mentorship of

Mr. Anji Babu

Smart Bridge

June 2025

Index

BRAINSTORMING &IDEATION

Objective:

Problem Context

TrafficTelligence is a machine learning-based project aimed at predicting traffic volume using historical data and contextual features such as weather conditions, holidays, and timestamps. The system analyzes patterns in road usage to estimate how many vehicles are likely to be on the road at a given time, which can assist in traffic management and urban planning. A Jupyter notebook contains the data analysis and model training pipeline, while a Flask web application allows users to interact with the model through a user-friendly interface. The project uses pre-trained models and encoders to process inputs and make real-time predictions.

Purpose of the Project

Key Points:

1. Collect Traffic Data
   Gather historical data on vehicle traffic along with corresponding details like weather, date, time, and holidays.

2. Clean and Prepare the Data
   Format and process the data to make it usable for machine learning—handle missing values, encode categories (like weather or holidays), etc.

3. Train a Prediction Model
   Use the cleaned data to train a machine learning model that learns how traffic volume changes with different conditions.

4. Evaluate Model Accuracy
   Test how well the model performs on unseen data to ensure it makes accurate predictions.

5. Build a Web App
   Create a user-friendly interface using Flask where users can input date, time, weather, and holiday status to get a traffic volume estimate.

6. Deploy and Use
   Allow city planners, commuters, or analysts to use the app to forecast traffic and plan accordingly.

➢ Problem Statement

Too Much Traffic: Cities often have traffic jams that waste time and fuel.

Hard to Predict: Traffic changes a lot depending on weather, time, and holidays.

Need for Prediction: If we could **predict traffic volume** in advance, we could plan better.

Smart Solution: Use **machine learning** to learn from past traffic data and make predictions.

Easy Access: Build a **web app** where users can enter date, time, weather, etc., and get a traffic estimate.

➢ Proposed Solution

- **Collect Data:**
  Use past traffic data that includes weather, date, time, and holiday information.
- **Clean & Prepare Data:**
  Organize the data, remove errors, and convert text (like weather conditions) into numbers the computer can understand.
- **Train the Model:**
  Use machine learning to teach a model how traffic volume changes with different conditions.
- **Test the Model:**
  Check if the model gives good predictions using new, unseen data.

- **Build a Web App:**
  Create a simple website using Flask where users can input weather, date, and time to get traffic volume predictions.
- **Make it Useful:**
  Let city planners, drivers, or anyone use the app to plan better and avoid heavy traffic.

Target Users

- **City Traffic Planners**
  To manage traffic flow and make better road and signal timing decisions.
- **Government and Municipal Authorities**
  For planning infrastructure improvements and reducing congestion in busy areas.
- **Commuters and Drivers**
  To check traffic predictions and choose the best time or route to travel.
- **Logistics & Delivery Companies**
  To optimize delivery schedules and avoid traffic delays.
- **Public Transport Services**
  To plan bus/train timings based on expected traffic conditions.
- **Researchers and Data Analysts**
  To study traffic patterns and explore urban mobility trends.

Expected Outcomes

1. **Accurate Traffic Predictions:**
   A machine learning model that can estimate traffic volume based on weather, time, and holiday information.
2. **User-Friendly Web Application**:
   A simple web interface where users can input details and instantly see predicted traffic volume.
3. **Smarter Planning:**
   Better decisions for travel, road use, and city traffic management.
4. **Reduced Congestion:**
   By using predictions, users can avoid high-traffic times, reducing overall traffic jams**.**

5. **Time and Fuel Savings:**
   Helps commuters and logistics companies save time and fuel by avoiding busy routes.
6. **Support for Future Projects:**
   Provides a foundation for further improvements in smart city and transportation systems.

REQUIREMENT ANALYSIS

Objective:

Functional Requirements:

The TrafficTelligence Project includes several key functional requirements to ensure it operates effectively. The system must provide a user-friendly interface where users can input relevant details such as date, time, weather conditions, and whether it is a holiday. These inputs are then processed by the backend, which prepares the data by encoding values appropriately before sending it to a pre-trained machine learning model. The model predicts the expected traffic volume based on the given inputs. The results must be displayed clearly on the web interface, allowing users to understand traffic conditions easily. Additionally, the web application should be responsive and accessible on both desktop and mobile devices. It must also include proper error handling to notify users of any invalid inputs or system issues during prediction**.**

Technical Requirements:

The TrafficTelligence Project requires several technical components to function efficiently. It must utilize a machine learning framework such as scikit-learn for training and saving the traffic volume prediction model. The project should be developed in Python, leveraging Jupyter Notebook for data analysis and model building. The web interface must be built using the Flask framework to enable interaction between the user and the prediction model. Data preprocessing tools like Pandas and NumPy are essential for handling and transforming traffic data. The system should also include necessary model files such as model.pkl and encoder files like encoder_weather.pkl and encoder_holiday.pkl for transforming categorical

inputs. Furthermore, the platform must support deployment on a server environment capable of running Python-based web applications. Basic front-end technologies such as HTML, CSS, and JavaScript may also be used to enhance the web interface's usability and responsiveness..

Key points:

➢ Technical Requirements:

- **Python Language**
  The project must be developed using Python.
- **Machine Learning Libraries**
  Use libraries like **scikit-learn**, **Pandas**, and **NumPy** for data processing and model building.
- **Model Files**
  Include trained model files like model.pkl and encoders (encoder_weather.pkl, encoder_holiday.pkl).
- **Jupyter Notebook**
  Use Jupyter Notebook for analyzing data and training the machine learning model.
- **Flask Framework**
  Build the web app using Flask to connect the model with a user-friendly interface.
- **Frontend Technologies**
  Use basic **HTML**, **CSS**, and optionally JavaScript to design the web interface.
- **Deployment Environment**
  A system or server that can run Python and Flask applications is needed for hosting.

➢ Functional Requirements:

- **User Input Form**
  Allow users to enter date, time, weather conditions, and holiday status.
- **Predict Traffic Volume**
  Use a trained machine learning model to predict traffic volume based on user inputs.
- **Display Output**
  Show the predicted traffic volume clearly on the web page.
- **Process Input Data**
  Convert user inputs (like weather and holidays) into machine-readable format using encoders.

- **Connect Frontend to Model**
  Link the user interface with the backend model using Flask.
- **Responsive Design**
  Ensure the web app works well on both computers and mobile devices.
- **Handle Errors Gracefully**
  Show helpful messages when something goes wrong or input is missing.

Constraints & Challenges:

- **Data Quality Issues**
  Incomplete or inaccurate traffic, weather, or holiday data can affect prediction accuracy.
- **Limited Features**
  Using only basic inputs like weather and holidays may not capture all factors affecting traffic (e.g., accidents, events).
- **Model Accuracy**
  Machine learning models may not always predict traffic volume correctly, especially in unusual conditions.
- **Real-time Input Limitation**
  The system doesn't fetch live weather or traffic updates — users must manually input the data.
- **Scalability**
  The current setup may need upgrades to handle large-scale deployment or heavy user traffic.
- **User Input Errors**
  Users may enter incorrect or inconsistent data, which could impact prediction reliability.
- **Deployment Constraints**
  Hosting the Flask web app and model online securely and reliably may require additional server resources.

PROJECT DESIGN

Objective:

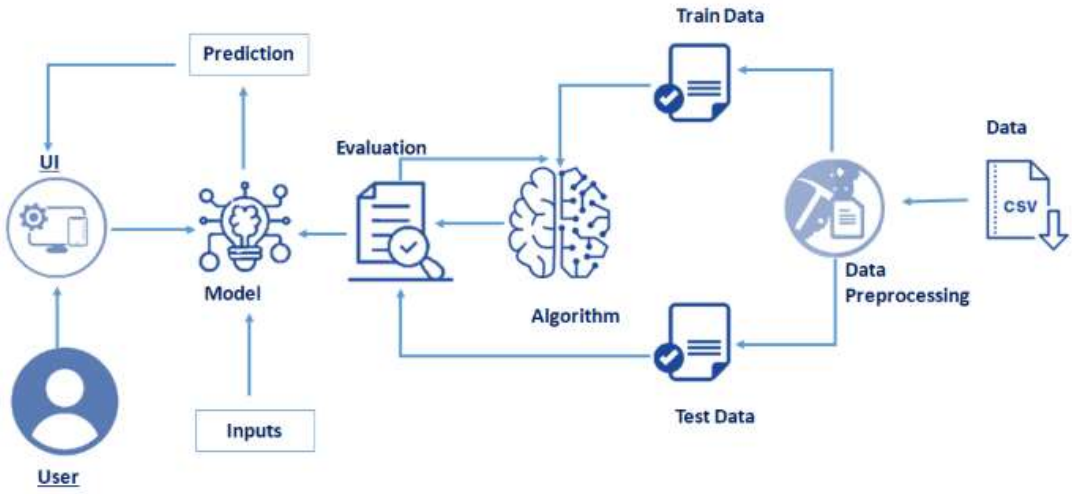
The objective of the TrafficTelligence Project is to develop an intelligent traffic volume prediction system using machine learning techniques. By analyzing historical traffic data along with weather conditions, time, and holiday

information, the system aims to provide accurate traffic volume forecasts. This helps users—including city planners, commuters, and logistics companies—make better travel and planning decisions through a user-friendly web application.

Key points:

The objective of the TrafficTelligence Project is to build a smart system that can accurately predict traffic volume using machine learning techniques. By analyzing real-world factors such as weather conditions, time of day, and holiday information, the project aims to provide reliable traffic forecasts. These predictions are intended to help users—including city planners, logistics companies, and everyday commuters—make better decisions about travel and traffic management. To make the system accessible, a simple and user-friendly web interface is developed, allowing users to input data and receive instant traffic volume predictions.

- **System Architecture Diagram:**



- **User flow: TrafficTelligence**
  - A user runs the Flask application, starting the local server.
  - They open the provided URL in their browser, leading to the **index.html** page.
  - On this page, they see a clean form where they input relevant traffic parameters (such as date, hour, weather, or location data).
  - Upon clicking the **Predict** button, the form data is sent as a POST request to the backend Flask route /predict.
  - The trained ML model processes this input and predicts the expected traffic volume with precision.
  - The results are rendered on **result.html**, displayed clearly at the /predict URL endpoint for the user to view.
  - The user can choose to return to the home page to make further predictions or close the application when done.

- **UI/UX Consideration:**

  TrafficTelligence is designed with a clean, responsive, and user-friendly interface for traffic analysts and planners. The input form uses Bootstrap for compatibility across devices, ensuring ease of use on desktops, tablets, and mobiles. High-contrast colours and readable font sizes support accessibility, while input fields are clearly labelled for intuitive data entry. Results are displayed clearly with minimal clutter, enabling quick interpretation and decision-making. Toast notifications or alert messages confirm submissions or highlight errors, enhancing user confidence. The overall design prioritises simplicity, clarity, and efficient workflow to support accurate traffic volume analysis.

PROJECT PLANNING

Objective:

The objective of the Project Planning phase is to structure the development of the TrafficTelligence platform into manageable tasks and timeframes using an Agile methodology. By organising work into focused sprints and distributing responsibilities across the team, the project ensures steady progress, timely feedback, and successful completion of each module. This structured plan includes sprint goals, task allocation, and milestone tracking.

Key Points:

➢ Sprint Planning:

- **Sprint 1: Project Setup**
  • Initialise Flask project with app.py, requirements.txt, and set up virtual environment.
  • Integrate necessary libraries like pandas, scikit-learn, pickle.
- **Sprint 2: Data Preprocessing & Model Training**
  • Clean and prepare traffic dataset.
  • Train ML model (RandomForestRegressor or XGBoost) and save as model.pkl.
- **Sprint 3: Prediction Module Development**
  • Create prediction route in app.py.
  • Develop input form in index.html and result display page.
- **Sprint 4: UI/UX Enhancement**
  • Design responsive frontend using Bootstrap.
  • Add clear labels, colours, and toast notifications for feedback.
- **Sprint 5: Database Integration (Optional)**
  • Set up SQLite/MySQL to store user queries and predictions for trend analysis.
- **Sprint 6: Testing & Optimisation**
  • Perform unit and integration testing.
  • Optimise model loading and prediction time.
- **Sprint 7: Documentation & Deployment**
  • Finalise README, add code comments, prepare deployment configs for GitHub or cloud.
  • Demo with sample data and screenshots.

Task Allocation:

Members	Tasks

Jalla Shalini	• Flask project setup • Integrate ML model with Flask routes • Final testing and bug fixing
Medapati Sai Sri Harsha	• Data cleaning and preprocessing • Model training and evaluation • Preparing model.pkl for integration
D. Venkatesh	• Frontend development: index.html form and result.html page • Adding toast notifications or alert messages
Chamarthi Deva Naga Sai	• Research and implement graphs or data visualisation. • Testing entire user flow • Preparing detailed documentation, screenshots, and presentation slides

TimeLine and Milestones:

Date	Milestone
Week-1	• Flask project setup • Data preprocessing • Model training
Week-2	• Prediction module development • Frontend input form and result page
Week 3	• UI/UX enhancements

	• Testing, bug fixes
Week 4	• Final documentation and deployment

PROJECT DEVELOPMENT

Objective:

The objective of the Project Development phase is to implement, integrate, and test all modules of the TrafficTelligence platform, transforming design into a fully functional traffic volume prediction system. This includes building the ML model, integrating it with Flask, developing a user-friendly frontend, and ensuring smooth input-output workflows.

Key Points:

➢ Technology Stack Used:

- **Backend Framework:** Python with Flask (for routing and integration)
- **Frontend Technologies:** HTML, CSS, JavaScript
- **Machine Learning:** scikit-learn (RandomForestRegressor for traffic prediction)
- **Data Handling:** pandas, numpy for preprocessing
- **Model Serialization:** pickle for saving and loading trained models
- **Version Control:** Git and GitHub for collaborative development
- **Deployment (Optional):** Local server, GitHub Pages (frontend demo), or cloud hosting

➢ Development Process:

➢ **Flask Project Initialization:**

- Created the basic Flask structure with app.py and requirements.txt.
- Installed necessary dependencies including Flask, scikit-learn, pandas, and pickle.

➢ **Data Preprocessing & Model Training:**

- Cleaned and prepared the traffic dataset to handle missing values and irrelevant columns.
- Trained the ML model using RandomForestRegressor for optimal accuracy.
- Saved the trained model as model.pkl for integration with Flask.

➢ **Prediction Module Implementation:**

- Developed the /predict route in app.py to handle user inputs and generate predictions.
- Loaded the saved model in the route to ensure real-time prediction capability.

➢ **Frontend Development:**

- Designed index.html with input fields for user data entry and a submit button.
- Created result.html to display the predicted traffic volume clearly.
- Added clear labels and buttons for intuitive user navigation.

➢ **Testing and Final Integration:**

- Performed module-wise testing for model accuracy and Flask route handling.
- Integrated frontend forms with backend prediction logic, ensuring smooth data flow.
- Conducted end-to-end testing to validate input submission, model prediction, and output display.

➢ Challenges & Fixes:

➢ **Model Integration Errors:**

Faced pickle loading errors initially due to mismatched library versions. Fixed by ensuring consistent environments during model saving and loading.

➢ **Input Format Mismatch:**

User inputs didn’t align with model feature expectations. Resolved by mapping frontend input values to the exact feature order and data types.

➢ **Frontend Styling Issues:**

The layout was misaligned on mobile screens. Fixed by implementing responsive design practices.

- **Prediction Delays:**
  Initial model loading took time during the first request. Optimised by loading the model once at app startup instead of per request.
- **Git Conflicts:**
  Merge conflicts occurred during collaborative updates. Resolved through regular commits, proper branching, and clear task allocation.

FUNCTIONAL & PERFOMANCE TESTING

Objective:

The primary objective of the Functional and Performance Testing phase is to ensure that all features of the TrafficTelligence platform work as intended and that the system performs reliably under expected user conditions. This phase involved testing each module individually and then in combination, checking for prediction accuracy, UI responsiveness, and overall stability. The goal was to identify and resolve any bugs, inconsistencies, or performance bottlenecks before final submission.

Key Points:

➢ Test Cases Executed:

- **Flask App Routes:**
  Verified that the home page (`/`) loads correctly and the prediction route (`/predict`) accepts inputs and returns outputs without errors.
- **Model Prediction:**
  Tested with multiple input scenarios to check if the trained model predicts traffic volume within acceptable accuracy ranges.
- **Input Form Handling:**
  Checked input field validations, clear labels, and submission functionality on `index.html`.
- **Result Display:**
  Ensured that predictions are displayed clearly on `result.html` without UI breaks.

- **Performance:**
  Measured prediction response times (<2s after model is loaded) and confirmed the UI remains responsive during interactions.

➢ Bug Fixes & Improvements:

Several issues were identified and fixed during testing:

- **Model Loading Delay:**
  Loading the model on each request caused delays. Fixed by loading the model once during app initialization.
- **Input Mismatch Errors:**
  Some input values caused type errors. Resolved by ensuring frontend data is correctly converted to the required numeric types.
- **UI Layout Misalignment:**
  The input form and result display were misaligned on mobile screens. Fixed by adjusting CSS styling.
- **Incorrect Predictions:**
  Initially, the model output was unrealistic for certain inputs due to insufficient preprocessing. Improved by reprocessing data with correct scaling and retraining.

➢ Final Validation:

After thorough testing, the TrafficTelligence platform was confirmed to:

- Accurately predict traffic volume for tested inputs
- Provide a clean, responsive user interface
- Handle input submissions and display results without errors
- Load predictions quickly and efficiently
- Meet the functional requirements defined during the planning phase

➢ Deployment:


Traffic Volume Estimation
Please enter the following details:
Predict


Traffic Volume Prediction Result
192 UYP