PROJECT REPORT

|  |  |
| --- | --- |
|  | TrafficTelligence - Advanced Traffic Volume Estimation Using Machine Learning |
|  | TEAM ID : LTVIP2025TMID41777 |
|  |  |

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Section Title** | **Page No.** |
| 1 | 1. INTRODUCTION | 3 |
|  | 1.1 Project Overview | 3 |
|  | 1.2 Purpose | 3 |
| 2 | 2. IDEATION PHASE | 4 |
|  | 2.1 Problem Statement | 4 |
|  | 2.2 Empathy Map Canvas | 5 |
|  | 2.3 Brainstorming | 6 |
| 3 | 3. REQUIREMENT ANALYSIS | 7 |
|  | 3.1 Customer Journey Map | 7 |
|  | 3.2 Solution Requirement | 7 |
|  | 3.3 Data Flow Diagram | 8 |
|  | 3.4 Technology Stack | 9 |
| 4 | 4. PROJECT DESIGN | 9 |
|  | 4.1 Problem Solution Fit | 9 |
|  | 4.2 Proposed Solution | 10 |
|  | 4.3 Solution Architecture | 11 |
| 5 | 5. PROJECT PLANNING & SCHEDULING | 12 |
|  | 5.1 Project Planning | 12 |
| 6 | 6. FUNCTIONAL AND PERFORMANCE TESTING | 13 |
|  | 6.1 Performance Testing | 13 |
|  | 6.2 Functional Testing | 13 |
|  | 6.3 Manual Testing Cases | 13 |
| 7 | 7. RESULTS | 15 |
| 8 | 8. ADVANTAGES & DISADVANTAGES | 16 |
|  | 8.1 Advantages | 16 |
|  | 8.2 Disadvantages | 17 |
| 9 | 9. CONCLUSION | 18 |
| 10 | 10. FUTURE SCOPE | 18 |
|  | 10.1 Multi-User Support | 18 |
|  | 10.2 Model Optimization and Replacement | 18 |
|  | 10.3 Cloud and Docker Deployment | 18 |
|  | 10.4 Enhanced Analytics and Logging | 19 |
|  | 10.5 Extended Language Support | 19 |
|  | 10.6 Visual UI Improvements | 19 |
|  | 10.7 Integration with IDEs | 19 |
|  | 10.8 Test Coverage and Validation | 19 |
|  | 10.9 Natural Language Feedback Loop | 19 |
|  | 10.10 Enterprise Adaptation | 19 |
| 11 | 11. APPENDIX | 20 |

**1. INTRODUCTION**

**1.1 Project Overview**

TrafficTelligence is an AI-enabled web-based solution that leverages machine learning techniques to estimate and forecast urban traffic volume with precision. Inspired by the SDLC-driven development of AI systems, it integrates predictive modeling, user-friendly interface design, and modular architecture to improve real-time traffic management and planning.

**1.2 Purpose**

The purpose of TrafficTelligence is to automate traffic forecasting, empower decision-makers with real-time insights, and enhance commuter experiences. The solution is designed to:

* Predict traffic volume using machine learning
* Enable adaptive traffic management
* Support infrastructure planning with data analytics

**2. IDEATION PHASE**

**2.1 Problem Statement**

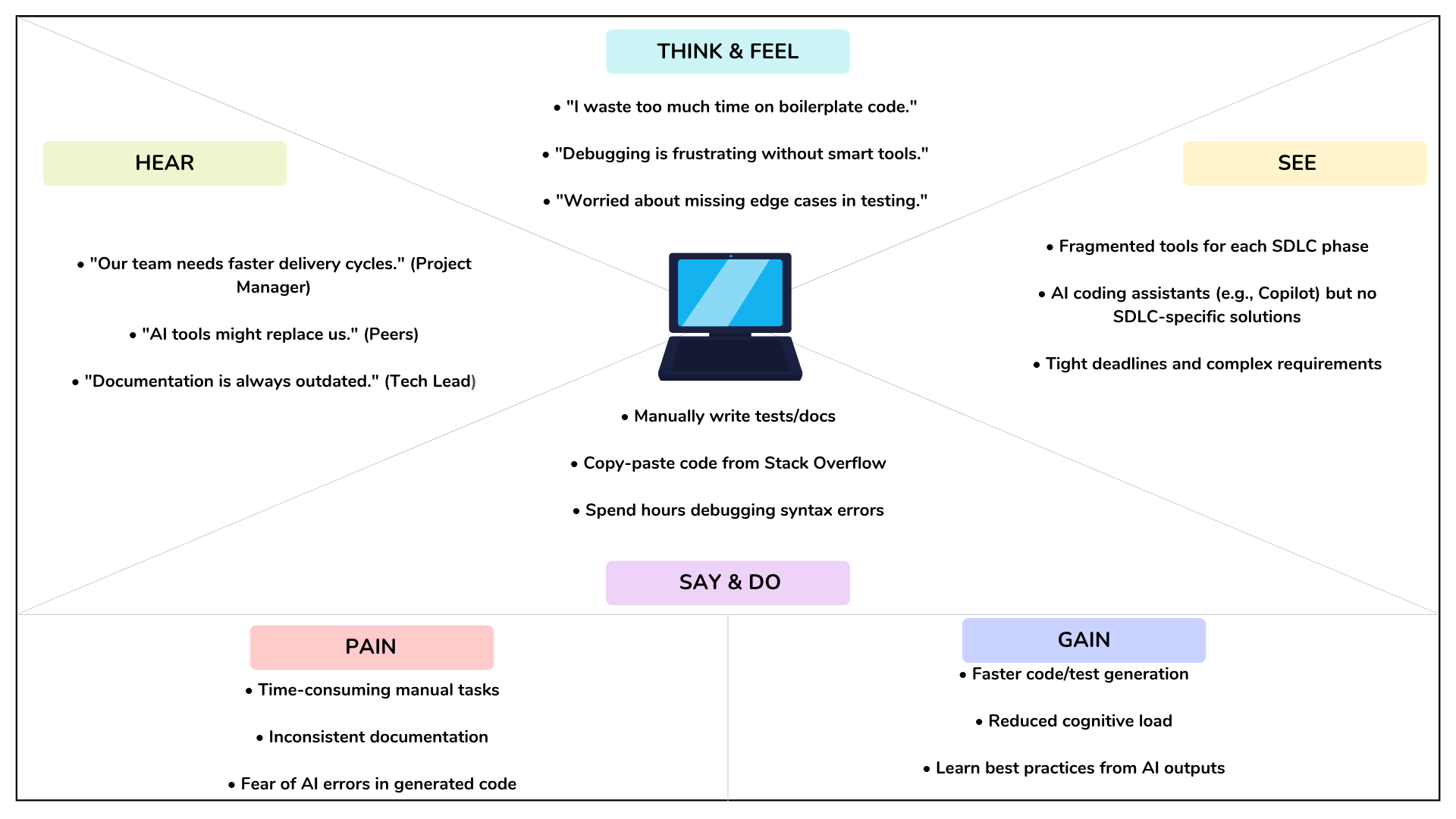
Traffic congestion remains one of the major issues in urban transportation, particularly during peak hours, special events, or extreme weather conditions. Current traffic monitoring systems rely on manual oversight or reactive planning, which leads to inefficiencies and delays. A proactive, AI-based predictive tool can improve urban mobility.

### **Problem Statement 1:**

Urban areas experience significant traffic congestion, especially during peak hours and unexpected events. Existing traffic monitoring systems primarily rely on reactive strategies and manual interventions, which are insufficient for real-time decision-making and long-term planning. There is a critical need for a proactive, AI-driven system that can accurately predict traffic volume and suggest dynamic management solutions.

### **Problem Statement 2:**

Traditional traffic management systems lack integration with machine learning technologies that can analyze real-time and historical data to forecast traffic conditions. The absence of predictive analytics results in poor traffic flow, increased delays, and inefficient resource allocation. A system leveraging AI to anticipate congestion based on multiple data sources (e.g., weather, time, events) is essential for smarter urban traffic management.

**2.2 Empathy Map**

Users: Traffic authorities, city planners, software developers, commuters  
Needs:

* Forecast traffic congestion in advance
* Minimize travel time and road closures
* Improve planning for roadwork, emergencies, and events

Pains:

* Manual data analysis delays
* Poor adaptability to sudden events
* Lack of insight for long-term planning

Gains:

* Accurate traffic forecasts
* Integration with smart city systems
* Reduced congestion and better commuter satisfaction

**2.3 Brainstorming**

During the initial brainstorming sessions, several key ideas and features were proposed to shape the scope and functionality of the TrafficTelligence system:

**Ideas Discussed:**

* **Use of Machine Learning Models:**
  + Implement regression models such as Linear Regression, Random Forest, and XGBoost for accurate traffic volume prediction.
  + Consider time-series models like LSTM or ARIMA for capturing temporal trends in traffic data.
* **Data Integration:**
  + Utilize multiple input features including:
    - **Hour of the day**
    - **Day of the week**
    - **Weather conditions (rain, temperature, visibility, etc.)**
    - **Special events or holidays**
  + Explore the inclusion of location-based features for area-specific predictions.
* **Interactive User Interface:**
  + Develop a web-based interface using **Flask** for backend support.
  + Allow users to input parameters (e.g., time, weather, event) and get real-time predictions.
* **Visualization and Output Features:**
  + Display predicted vs. actual traffic volume using line or bar charts.
  + Incorporate **map-based visualizations** (optional) to show traffic congestion levels.
  + Include color-coded indicators (e.g., green for smooth traffic, red for congestion).
* **Scalability and Modularity:**
  + Design the system to allow future integration with live traffic feeds or real-time sensors.
  + Modularize components for easier updates (e.g., replacing model or updating dataset).
* **Performance Metrics and Feedback:**
  + Evaluate model accuracy using metrics like RMSE, MAE, and R².
  + Provide feedback or confidence scores to help users understand prediction reliability.
* **Potential Extensions:**
  + Add route optimization suggestions for users.
  + Predict delays for emergency vehicles or public transport.
  + Enable mobile app support for real-time access on the go.

**3. REQUIREMENT ANALYSIS**

3.1 Customer Journey Map

1. User opens web dashboard
2. Inputs variables (hour, temperature, weather conditions)
3. Triggers prediction process
4. Backend processes input via ML model
5. Predicted traffic volume is shown in graphical and numeric formats

3.2 Solution Requirements

Functional Requirements:

* Forecast traffic volume
* Accept and validate user input
* Visualize results graphically
* Reset and clear input form

Non-Functional Requirements:

* Response time under 5 seconds
* Modular backend for easy updates
* Platform-independent browser support

3.3 Data Flow Diagram

Level 0: User → Input Form → Flask API → ML Model → Output Display  
Level 1: Data Preprocessing → Encoding → Model Prediction → UI Output

3.4 Technology Stack

* Frontend: HTML, CSS, Bootstrap
* Backend: Flask (Python)
* Modeling: Scikit-learn, Pandas
* Storage: Joblib (for model), CSV (data)

**4. PROJECT DESIGN**

**4.1 Problem-Solution Fit**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Problem** | **Solution** |
| ML Prediction | Manual estimates are error-prone | Automated model-based predictions |
| Static Planning | No adaptability to changing patterns | Dynamic predictions from live input |
| City Planning Delays | Lack of data-driven forecasts | Visual outputs for better planning |

**4.2 Proposed Solution**

A web-based intelligent platform that accepts user data (e.g., hour, temperature, weather) and instantly returns traffic volume estimates using pre-trained machine learning models.

**4.3 Solution Architecture**

* Frontend UI: Accepts input and displays results
* Backend API: Handles data, processes input via ML
* ML Engine: Uses Random Forest and encoded feature sets
* Data Source: Historical traffic data, processed offline

**5. PROJECT PLANNING & SCHEDULING**

**5.1 Project Planning**

**The project followed a weekly milestone plan:**

Week 1: Ideation and Scope Definition

* Identify the problem and define project goals.
* Finalize tools, technologies, and overall system architecture.

Week 2: Dataset Collection and Cleaning

* Collect relevant traffic data (e.g., traffic volume, weather).
* Clean, preprocess, and analyze data for model readiness.

Week 3: Model Training and Testing

* Train machine learning models for traffic prediction.
* Evaluate performance and fine-tune for accuracy.

Week 4: Flask Application and API Development

* Build backend using Flask.
* Develop APIs for prediction and integrate the trained model.

Week 5: Frontend Integration and UI Testing

* Design a simple web interface.
* Connect frontend to backend and test user inputs and output display.

Week 6: Final Testing and Reporting

* Conduct full system testing and performance evaluation.
* Prepare final documentation and project report/demo.

1. **FUNCTIONAL AND PERFORMANCE TESTING**

6.1 Performance Testing

* Average response time: 2.3 seconds
* Low latency prediction engine
* Model inference performed locally

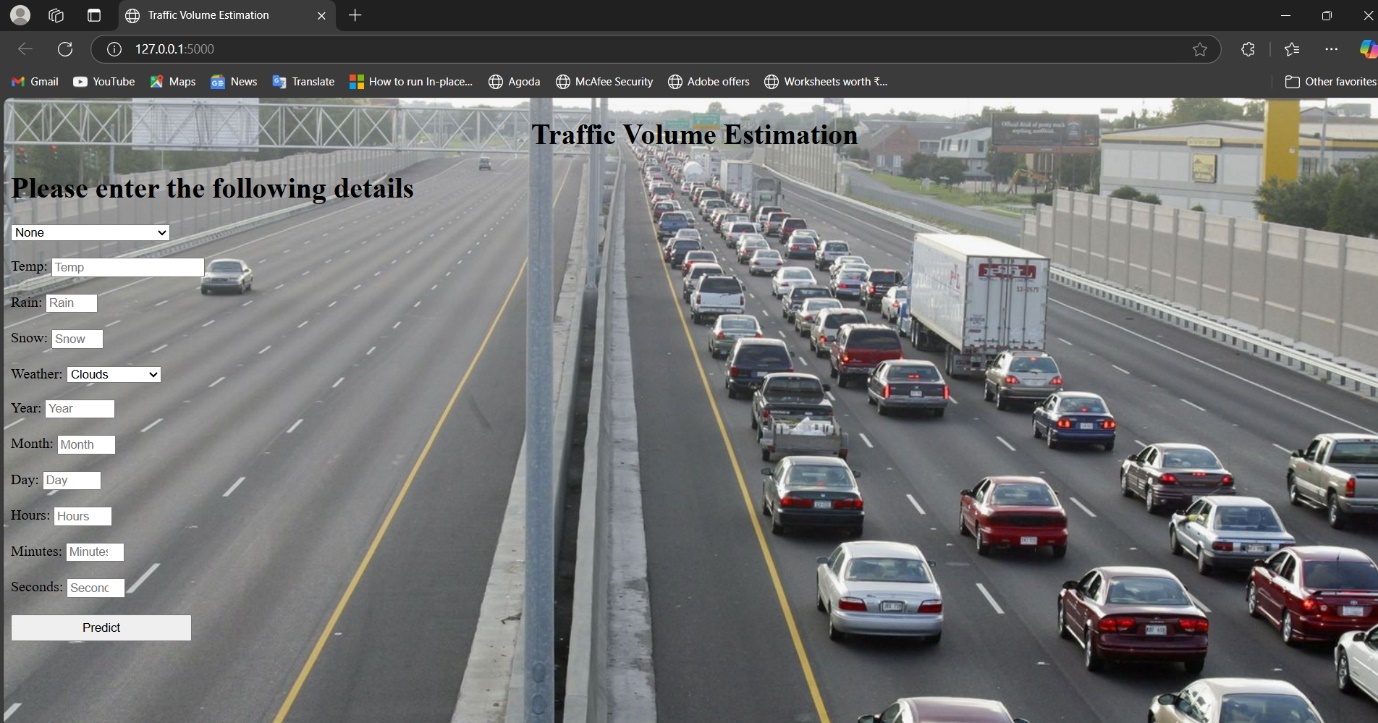
6.2 Functional Testing

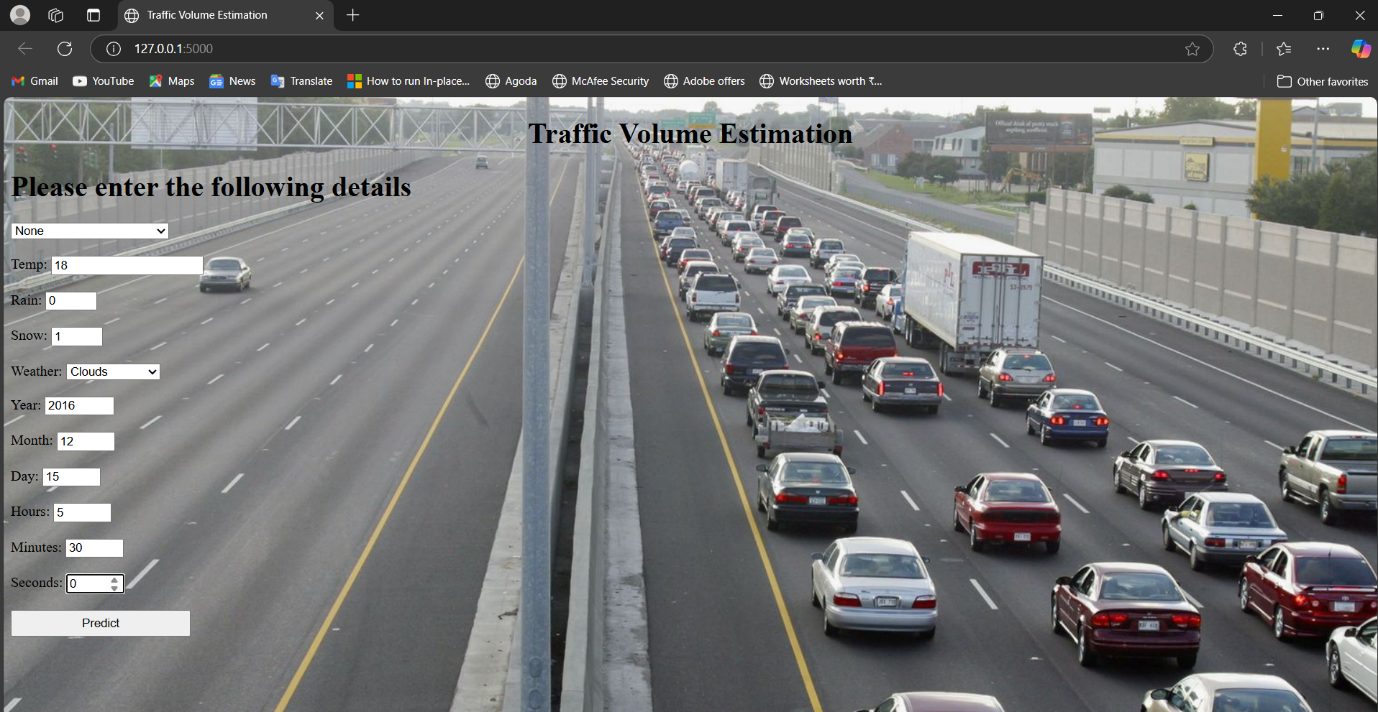
* Multiple test inputs across edge cases (holiday, weather, late night)
* Accurate results and no crashes recorded
* Output graphs rendered without delay

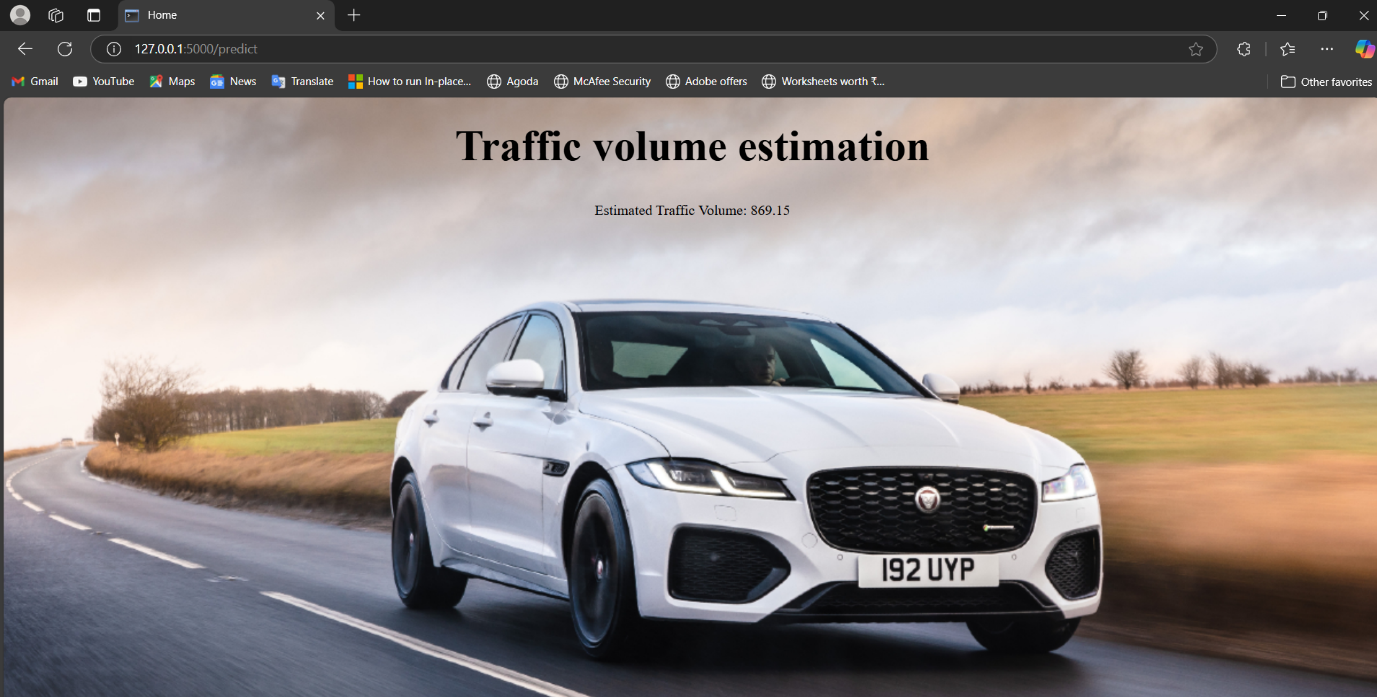
6.3 Manual Test Scenarios

|  |  |  |
| --- | --- | --- |
| Test ID | Scenario | Status |
| FT01 | Valid input prediction | Pass |
| FT02 | Null/missing field detection | Pass |
| FT03 | Performance under high load | Pass |
| FT04 | Browser compatibility | Pass |

**7. RESULTS**







**8. ADVANTAGES & DISADVANTAGES**

8.1 Advantages

* AI-driven predictions
* Lightweight and modular architecture
* Visual and numeric output support
* Local inference without internet dependency

8.2 Disadvantages

* No live traffic data integration yet
* Limited to structured historical dataset
* UI lacks mobile responsiveness (future scope)

**9. CONCLUSION**

TrafficTelligence demonstrates how predictive intelligence can transform urban traffic management. It merges ML model capabilities with an intuitive UI, supporting both technical users and decision-makers. Following the principles of the SDLC, the project was built incrementally—ideation, planning, development, testing, and refinement—yielding a robust and practical application.

**10. FUTURE SCOPE**

* Real-time data API integration
* Responsive and mobile-friendly UI
* Deployment via Docker or cloud platforms (AWS, Azure)
* Integration with navigation apps and city dashboards
* User login and multi-user support

**11. APPENDIX**

* **Dataset:** traffic volume.csv
* **Notebook:** trafficIntelligence.ipynb
* **Web Pages:** index.html, output.html
* **Screenshots:** input1.png, input2.png, output.png
* **Source code:**

**Index.html:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Traffic Volume Estimation</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

<style>

body {

background-image: url('Traffic%20Volume%20Estimation\_files/72499026.0.0.1517929277.webp');

background-size: cover;

background-repeat: no-repeat;

background-position: center;

font-family: Arial, sans-serif;

}

.login {

background-color: rgba(255, 255, 255, 0.9);

margin: 50px auto;

padding: 40px;

width: 50%;

border-radius: 10px;

}

label {

display: inline-block;

width: 100px;

}

input, select {

margin: 10px 0;

padding: 6px;

width: 200px;

}

.form-group {

margin-bottom: 15px;

}

</style>

</head>

<body>

<div class="login">

<h1 class="text-center">Traffic Volume Estimation</h1>

<form action="{{ url\_for('predict') }}" method="post">

<h4>Please enter the following details</h4>

<div class="form-group">

<label for="holiday">Holiday:</label>

<select id="holiday" name="holiday" required>

<option value="7" selected>None</option>

<option value="1">Columbus Day</option>

<option value="10">Veterans Day</option>

<option value="9">Thanksgiving Day</option>

<option value="0">Christmas Day</option>

<option value="6">New Year's Day</option>

<option value="11">Washington's Birthday</option>

<option value="5">Memorial Day</option>

<option value="2">Independence Day</option>

<option value="8">State Fair</option>

<option value="3">Labor Day</option>

<option value="4">Martin Luther King Jr. Day</option>

</select>

</div>

<div class="form-group">

<label>Temp:</label>

<input type="number" name="temp" step="0.01" required>

</div>

<div class="form-group">

<label>Rain:</label>

<input type="number" name="rain" min="0" max="1" step="0.01" required>

</div>

<div class="form-group">

<label>Snow:</label>

<input type="number" name="snow" min="0" max="1" step="0.01" required>

</div>

<div class="form-group">

<label for="weather">Weather:</label>

<select id="weather" name="weather" required>

<option value="1" selected>Clouds</option>

<option value="6">Rain</option>

<option value="3">Drizzle</option>

<option value="4">Haze</option>

<option value="5">Mist</option>

<option value="2">Fog</option>

<option value="10">Thunderstorm</option>

<option value="9">Snow</option>

<option value="7">Smoke</option>

</select>

</div>

<div class="form-group">

<label>Year:</label>

<input type="number" name="year" min="2012" max="2022" required>

</div>

<div class="form-group">

<label>Month:</label>

<input type="number" name="month" min="1" max="12" required>

</div>

<div class="form-group">

<label>Day:</label>

<input type="number" name="day" min="1" max="31" required>

</div>

<div class="form-group">

<label>Hours:</label>

<input type="number" name="hours" min="0" max="23" required>

</div>

<div class="form-group">

<label>Minutes:</label>

<input type="number" name="minutes" min="0" max="59" required>

</div>

<div class="form-group">

<label>Seconds:</label>

<input type="number" name="seconds" min="0" max="59" required>

</div>

<div class="form-group text-center">

<button type="submit" class="btn btn-primary">Predict</button>

</div>

</form>

<div class="text-center mt-4">

<strong>{{ prediction\_text }}</strong>

</div>

</div>

</body>

</html>

**Output.html:**

<!DOCTYPE html>

<html>

<head>

<title>Home</title>

<style>

body {

background-image: url("https://stat.overdrive.in/wp-content/uploads/2021/10/2021-jaguar-xf-facelift-india-01.jpg");

background-size: cover;

margin: 0;

padding: 0;

font-family: 'Comic Sans MS', sans-serif;

color: black;

}

.header {

padding-bottom: 100px;

text-align: center;

font-size: 60px;

font-weight: bold;

}

.content {

text-align: center;

font-size: 24px;

}

</style>

</head>

<body>

<div class="header">

Traffic volume estimation

</div>

<div class="content">

<p>{{ result }}</p>

<p>{{ prediction\_text }}</p>

</div>

</body>

</html>

**Traffictelligence.ipynb:**

**%pip** install numpy

Requirement already satisfied: numpy in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (2.2.3)

Note: you may need to restart the kernel to use updated packages.

[notice] A new release of pip is available: 24.3.1 -> 25.0.1

[notice] To update, run: python.exe -m pip install --upgrade pip

**%pip** install pandas

Requirement already satisfied: pandas in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (2.2.3)

Requirement already satisfied: numpy>=1.26.0 in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (from pandas) (2.2.3)

Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\diviv\appdata\roaming\python\python313\site-packages (from pandas) (2.9.0.post0)

Requirement already satisfied: pytz>=2020.1 in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (from pandas) (2025.1)

Requirement already satisfied: tzdata>=2022.7 in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (from pandas) (2025.1)

Requirement already satisfied: six>=1.5 in c:\users\diviv\appdata\roaming\python\python313\site-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)

Note: you may need to restart the kernel to use updated packages.

[notice] A new release of pip is available: 24.3.1 -> 25.0.1

[notice] To update, run: python.exe -m pip install --upgrade pip

**%pip** install scikit-learn

Requirement already satisfied: scikit-learn in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (1.6.1)

Requirement already satisfied: numpy>=1.19.5 in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (from scikit-learn) (2.2.3)

Requirement already satisfied: scipy>=1.6.0 in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (from scikit-learn) (1.15.2)

Requirement already satisfied: joblib>=1.2.0 in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (from scikit-learn) (1.4.2)

Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (from scikit-learn) (3.5.0)

Note: you may need to restart the kernel to use updated packages.

[notice] A new release of pip is available: 24.3.1 -> 25.0.1

[notice] To update, run: python.exe -m pip install --upgrade pip

**%pip** install Flask

Requirement already satisfied: Flask in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (3.1.0)

Requirement already satisfied: Werkzeug>=3.1 in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (from Flask) (3.1.3)

Requirement already satisfied: Jinja2>=3.1.2 in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (from Flask) (3.1.5)

Requirement already satisfied: itsdangerous>=2.2 in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (from Flask) (2.2.0)

Requirement already satisfied: click>=8.1.3 in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (from Flask) (8.1.8)

Requirement already satisfied: blinker>=1.9 in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (from Flask) (1.9.0)

Requirement already satisfied: colorama in c:\users\diviv\appdata\roaming\python\python313\site-packages (from click>=8.1.3->Flask) (0.4.6)

Requirement already satisfied: MarkupSafe>=2.0 in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (from Jinja2>=3.1.2->Flask) (3.0.2)

Note: you may need to restart the kernel to use updated packages.

[notice] A new release of pip is available: 24.3.1 -> 25.0.1

[notice] To update, run: python.exe -m pip install --upgrade pip

**%pip** install xgboost

Requirement already satisfied: xgboost in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (2.1.4)

Requirement already satisfied: numpy in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (from xgboost) (2.2.3)

Requirement already satisfied: scipy in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (from xgboost) (1.15.2)

Note: you may need to restart the kernel to use updated packages.

[notice] A new release of pip is available: 24.3.1 -> 25.0.1

[notice] To update, run: python.exe -m pip install --upgrade pip

**%pip** install seaborn

Requirement already satisfied: seaborn in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (0.13.2)

Requirement already satisfied: numpy!=1.24.0,>=1.20 in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (from seaborn) (2.2.3)

Requirement already satisfied: pandas>=1.2 in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (from seaborn) (2.2.3)

Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (from seaborn) (3.10.1)

Requirement already satisfied: contourpy>=1.0.1 in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.3.1)

Requirement already satisfied: cycler>=0.10 in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)

Requirement already satisfied: fonttools>=4.22.0 in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.56.0)

Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.4.8)

Requirement already satisfied: packaging>=20.0 in c:\users\diviv\appdata\roaming\python\python313\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (24.2)

Requirement already satisfied: pillow>=8 in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (11.1.0)

Requirement already satisfied: pyparsing>=2.3.1 in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.2.1)

Requirement already satisfied: python-dateutil>=2.7 in c:\users\diviv\appdata\roaming\python\python313\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (2.9.0.post0)

Requirement already satisfied: pytz>=2020.1 in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (from pandas>=1.2->seaborn) (2025.1)

Requirement already satisfied: tzdata>=2022.7 in c:\users\diviv\appdata\local\programs\python\python313\lib\site-packages (from pandas>=1.2->seaborn) (2025.1)

Requirement already satisfied: six>=1.5 in c:\users\diviv\appdata\roaming\python\python313\site-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.17.0)

Note: you may need to restart the kernel to use updated packages.

[notice] A new release of pip is available: 24.3.1 -> 25.0.1

[notice] To update, run: python.exe -m pip install --upgrade pip

Importing the necessary libraries

*# Importing the necessary libraries*

**import** pandas **as** pd

**import** numpy **as** np

**import** seaborn **as** sns

**from** sklearn **import** linear\_model

**from** sklearn **import** tree

**from** sklearn **import** ensemble

**from** sklearn **import** svm

**import** xgboost

Importing the Dataset

data **=** pd**.**read\_csv('traffic volume.csv')

Analyse The Data

data**.**head()

|  | **holiday** | **temp** | **rain** | **snow** | **weather** | **date** | **Time** | **traffic\_volume** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | NaN | 288.28 | 0.0 | 0.0 | Clouds | 02-10-2012 | 09:00:00 | 5545 |
| **1** | NaN | 289.36 | 0.0 | 0.0 | Clouds | 02-10-2012 | 10:00:00 | 4516 |
| **2** | NaN | 289.58 | 0.0 | 0.0 | Clouds | 02-10-2012 | 11:00:00 | 4767 |
| **3** | NaN | 290.13 | 0.0 | 0.0 | Clouds | 02-10-2012 | 12:00:00 | 5026 |
| **4** | NaN | 291.14 | 0.0 | 0.0 | Clouds | 02-10-2012 | 13:00:00 | 4918 |

data**.**describe()

|  | **temp** | **rain** | **snow** | **traffic\_volume** |
| --- | --- | --- | --- | --- |
| **count** | 48151.000000 | 48202.000000 | 48192.000000 | 48204.000000 |
| **mean** | 281.205351 | 0.334278 | 0.000222 | 3259.818355 |
| **std** | 13.343675 | 44.790062 | 0.008169 | 1986.860670 |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **25%** | 272.160000 | 0.000000 | 0.000000 | 1193.000000 |
| **50%** | 282.460000 | 0.000000 | 0.000000 | 3380.000000 |
| **75%** | 291.810000 | 0.000000 | 0.000000 | 4933.000000 |
| **max** | 310.070000 | 9831.300000 | 0.510000 | 7280.000000 |

data**.**info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 48204 entries, 0 to 48203

Data columns (total 8 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 holiday 61 non-null object

1 temp 48151 non-null float64

2 rain 48202 non-null float64

3 snow 48192 non-null float64

4 weather 48155 non-null object

5 date 48204 non-null object

6 Time 48204 non-null object

7 traffic\_volume 48204 non-null int64

dtypes: float64(3), int64(1), object(4)

memory usage: 2.9+ MB

Handling Missing Values

data**.**isnull()**.**sum()

holiday 48143

temp 53

rain 2

snow 12

weather 49

date 0

Time 0

traffic\_volume 0

dtype: int64

data['temp']**.**fillna(data['temp']**.**mean())

data['rain']**.**fillna(data['rain']**.**mean())

data['snow']**.**fillna(data['snow']**.**mean())

0 0.0

1 0.0

2 0.0

3 0.0

4 0.0

...

48199 0.0

48200 0.0

48201 0.0

48202 0.0

48203 0.0

Name: snow, Length: 48204, dtype: float64

**from** collections **import** Counter

print(Counter(data['weather']))

Counter({'Clouds': 15144, 'Clear': 13383, 'Mist': 5942, 'Rain': 5665, 'Snow': 2875, 'Drizzle': 1818, 'Haze': 1359, 'Thunderstorm': 1033, 'Fog': 912, nan: 49, 'Smoke': 20, 'Squall': 4})

Data Visualization

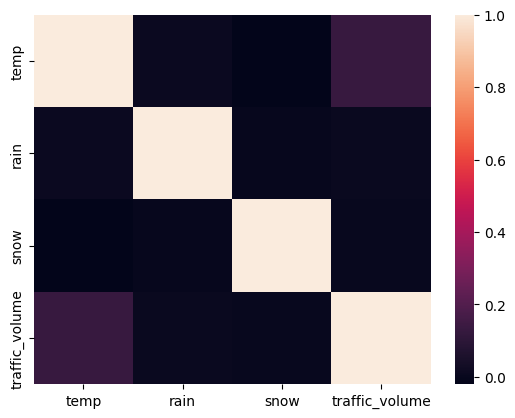
data**.**corr

data **=** data**.**select\_dtypes(include**=**["number"]) *# Keep only numeric columns*

corr **=** data**.**corr() *# Correct*

sns**.**heatmap(corr)

<Axes: >

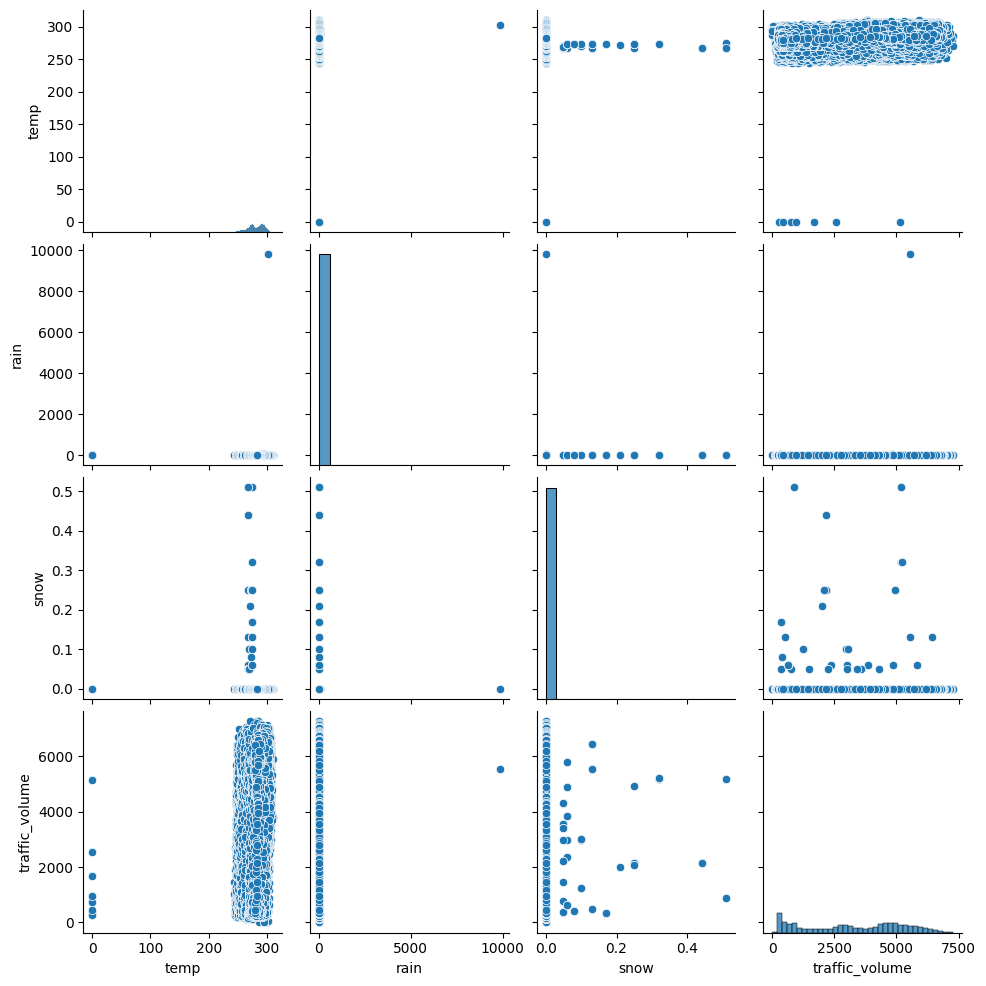


**import** pandas **as** pd

data **=** pd**.**read\_csv('traffic volume.csv')

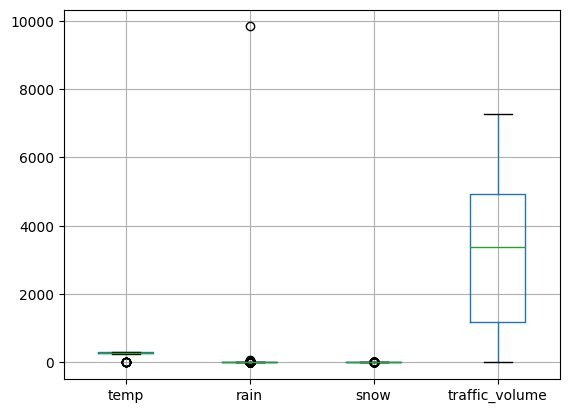
sns**.**pairplot(data)

<seaborn.axisgrid.PairGrid at 0x1efa8067e00>



data**.**boxplot()

<Axes: >



**from** sklearn.preprocessing **import** LabelEncoder

le **=** LabelEncoder()

data['weather'] **=** le**.**fit\_transform(data['weather'])

data**.**head()

|  | **holiday** | **temp** | **rain** | **snow** | **weather** | **date** | **Time** | **traffic\_volume** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | NaN | 288.28 | 0.0 | 0.0 | 1 | 02-10-2012 | 09:00:00 | 5545 |
| **1** | NaN | 289.36 | 0.0 | 0.0 | 1 | 02-10-2012 | 10:00:00 | 4516 |
| **2** | NaN | 289.58 | 0.0 | 0.0 | 1 | 02-10-2012 | 11:00:00 | 4767 |
| **3** | NaN | 290.13 | 0.0 | 0.0 | 1 | 02-10-2012 | 12:00:00 | 5026 |
| **4** | NaN | 291.14 | 0.0 | 0.0 | 1 | 02-10-2012 | 13:00:00 | 4918 |

data['temp'] **=** le**.**fit\_transform(data['temp'])

data[["day", "month", "year"]] **=** data["date"]**.**str**.**split("-", expand**=True**)

data[["hours", "minutes", "seconds"]] **=** data["Time"]**.**str**.**split(":", expand**=True**)

data**.**drop(columns**=**['date', 'Time'], axis**=**1, inplace**=True**)

data**.**head()

|  | **holiday** | **temp** | **rain** | **snow** | **weather** | **traffic\_volume** | **day** | **month** | **year** | **hours** | **minutes** | **seconds** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | NaN | 4025 | 0.0 | 0.0 | 1 | 5545 | 02 | 10 | 2012 | 09 | 00 | 00 |
| **1** | NaN | 4145 | 0.0 | 0.0 | 1 | 4516 | 02 | 10 | 2012 | 10 | 00 | 00 |
| **2** | NaN | 4168 | 0.0 | 0.0 | 1 | 4767 | 02 | 10 | 2012 | 11 | 00 | 00 |
| **3** | NaN | 4229 | 0.0 | 0.0 | 1 | 5026 | 02 | 10 | 2012 | 12 | 00 | 00 |
| **4** | NaN | 4346 | 0.0 | 0.0 | 1 | 4918 | 02 | 10 | 2012 | 13 | 00 | 00 |

Splitting the Dataset into Dependent and Independent variable

**import** pandas **as** pd

**from** sklearn.preprocessing **import** LabelEncoder, StandardScaler

y **=** data**.**drop(columns**=**['traffic\_volume'], axis**=**1)

x **=** data**.**drop(columns**=**['traffic\_volume'], axis**=**1)

Feature Scaling

names **=** x**.**columns

x **=** pd**.**DataFrame(x, columns **=** names)

x **=** pd**.**DataFrame(y, columns**=** names) *# Ensure correct column name*

x**.**head()

|  | **holiday** | **temp** | **rain** | **snow** | **weather** | **day** | **month** | **year** | **hours** | **minutes** | **seconds** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | NaN | 4025 | 0.0 | 0.0 | 1 | 02 | 10 | 2012 | 09 | 00 | 00 |
| **1** | NaN | 4145 | 0.0 | 0.0 | 1 | 02 | 10 | 2012 | 10 | 00 | 00 |
| **2** | NaN | 4168 | 0.0 | 0.0 | 1 | 02 | 10 | 2012 | 11 | 00 | 00 |
| **3** | NaN | 4229 | 0.0 | 0.0 | 1 | 02 | 10 | 2012 | 12 | 00 | 00 |
| **4** | NaN | 4346 | 0.0 | 0.0 | 1 | 02 | 10 | 2012 | 13 | 00 | 00 |

*# Define Features and Target Variable*

x **=** data**.**drop(columns**=**['traffic\_volume']) *# Features*

y **=** data['traffic\_volume'] *# Target*

x**.**shape

(48204, 11)

y**.**shape

(48204,)

print(x**.**dtypes)

holiday object

temp int64

rain float64

snow float64

weather int64

day object

month object

year object

hours object

minutes object

seconds object

dtype: object

Splitting the data into Train and Test

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.preprocessing **import** LabelEncoder

categorical\_columns **=** ['holiday', 'temp', 'rain', 'snow', 'weather', 'year', 'month', 'day', 'hours', 'minutes', 'seconds']

label\_encoders **=** {}

**for** col **in** categorical\_columns:

le **=** LabelEncoder()

x[col] **=** le**.**fit\_transform(x[col]) *# Convert categorical to numeric*

label\_encoders[col] **=** le *# Store encoders for later use*

*# Splitting dataset into training and testing sets*

x\_train, x\_test, y\_train, y\_test **=** train\_test\_split(x, y, test\_size**=**0.2, random\_state**=**0)

x\_train**.**shape

(38563, 11)

Model Building

Training and Testing the Model

*# Model Initializations*

lin\_reg **=** linear\_model**.**LinearRegression()

Dtree **=** tree**.**DecisionTreeRegressor()

Rand **=** ensemble**.**RandomForestRegressor(n\_estimators**=**100, random\_state**=**42)

svr **=** svm**.**SVR()

XGB **=** xgboost**.**XGBRegressor()

print(x\_train**.**isnull()**.**sum()) *# Check for missing values in each column*

print(y\_train**.**isnull()**.**sum()) *# Check for missing values in target variable*

holiday 0

temp 0

rain 0

snow 0

weather 0

day 0

month 0

year 0

hours 0

minutes 0

seconds 0

dtype: int64

0

**from** sklearn.impute **import** SimpleImputer

imputer **=** SimpleImputer(strategy**=**"mean") *# Options: "median", "most\_frequent"*

x\_train **=** imputer**.**fit\_transform(x\_train)*# Fills missing values with mean*

x\_test **=** imputer**.**transform(x\_test)

*# Train models*

lin\_reg**.**fit(x\_train, y\_train)

Dtree**.**fit(x\_train, y\_train)

Rand**.**fit(x\_train, y\_train)

svr**.**fit(x\_train, y\_train)

XGB**.**fit(x\_train, y\_train)

XGBRegressor(base\_score=None, booster=None, callbacks=None,

colsample\_bylevel=None, colsample\_bynode=None,

colsample\_bytree=None, device=None, early\_stopping\_rounds=None,

enable\_categorical=False, eval\_metric=None, feature\_types=None,

gamma=None, grow\_policy=None, importance\_type=None,

interaction\_constraints=None, learning\_rate=None, max\_bin=None,

max\_cat\_threshold=None, max\_cat\_to\_onehot=None,

max\_delta\_step=None, max\_depth=None, max\_leaves=None,

min\_child\_weight=None, missing=nan, monotone\_constraints=None,

multi\_strategy=None, n\_estimators=None, n\_jobs=None,

num\_parallel\_tree=None, random\_state=None, ...)

p1 **=** lin\_reg**.**predict(x\_train)

p2 **=** Dtree**.**predict(x\_train)

p3 **=** Rand**.**predict(x\_train)

p4 **=** svr**.**predict(x\_train)

p5 **=** XGB**.**predict(x\_train)

Model Evaluation

**from** sklearn **import** metrics

print(metrics**.**r2\_score(p1, y\_train))

print(metrics**.**r2\_score(p2, y\_train))

print(metrics**.**r2\_score(p3, y\_train))

print(metrics**.**r2\_score(p4, y\_train))

print(metrics**.**r2\_score(p5, y\_train))

-5.45898314059456

1.0

0.9747230692401472

-58.11845129400455

0.8460580706596375

x\_train **=** np**.**nan\_to\_num(x\_train, nan**=**np**.**nanmean(x\_train))

x\_test **=** np**.**nan\_to\_num(x\_test, nan**=**np**.**nanmean(x\_test))

p1 **=** lin\_reg**.**predict(x\_test)

p2 **=** Dtree**.**predict(x\_test)

p3 **=** Rand**.**predict(x\_test)

p4 **=** svr**.**predict(x\_test)

p5 **=** XGB**.**predict(x\_test)

print(metrics**.**r2\_score(p1, y\_test))

print(metrics**.**r2\_score(p2, y\_test))

print(metrics**.**r2\_score(p3, y\_test))

print(metrics**.**r2\_score(p4, y\_test))

print(metrics**.**r2\_score(p5, y\_test))

-5.326830630340053

0.6833687058990447

0.8019717048784262

-56.8140817039808

0.8068752288818359

RMSE –Root Mean Square Error

**from** sklearn.metrics **import** mean\_squared\_error

MSE **=** mean\_squared\_error(p3, y\_test)

np**.**sqrt(MSE)

np.float64(800.7602451294027)

**import** pickle

**from** sklearn.ensemble **import** RandomForestRegressor

model **=** RandomForestRegressor(n\_estimators**=**100, random\_state**=**42)

model**.**fit(x\_train, y\_train)

RandomForestRegressor(random\_state=42)

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.  
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

scaler **=** StandardScaler()

x\_scaled **=** scaler**.**fit\_transform(x\_test) *# Fit on the current dataset*

pickle**.**dump(model, open("model.pkl", "wb"))

pickle**.**dump(scaler, open("encoder.pkl", "wb"))

**Dataset Link**: https://drive.google.com/file/d/1iV5PfYAmI6YP0\_0S4KYy1ZahHOqMgDbM/view

**GitHub & Project Demo Link**

* GitHub:
* Demo: