Project Report Format

1. INTRODUCTION

1.1 Project Overview

This project, TrafficTelligence: Advanced Traffic Volume Estimation with Machine Learning, aims to predict and manage traffic congestion by analyzing data such as weather, date, time, and road conditions. Using machine learning models, it delivers real-time traffic volume predictions through a web-based interface.

1.2 Purpose

The purpose is to reduce traffic congestion and improve transport efficiency by providing timely and accurate traffic volume predictions using machine learning. This solution empowers commuters, planners, and city authorities with predictive insights to make informed decisions.

2. IDEATION PHASE

2.1 Problem Statement

Urban traffic congestion causes delays, pollution, and resource wastage. Traditional expansion methods like adding more roads are ineffective. Thus, there's a need for a smarter solution to predict traffic volume using data-driven methods.

2.2 Empathy Map Canvas

Sees: Long queues, red signals, weather impacts
Hears: Horns, traffic alerts, complaints
Says: "Why is it always crowded here?", "I'll be late again"
Thinks: "Can I avoid this jam?", "Will the shortcut help?"
Feels: Frustrated, helpless, tired

2.3 Brainstorming

- Use real-world traffic and weather datasets
- Predict traffic volume using ML
- Display predictions via web app
- Compare different algorithms
- Deploy using Flask and IBM Cloud

3. REQUIREMENT ANALYSIS

3.1 Customer Journey map

User opens app → Inputs date/time/weather → Receives traffic prediction → Decides route/time → May give feedback

3.2 Solution Requirement

Functional: Input UI, Prediction Engine, Result Display, Report download
Non-Functional: Usability, Response Time, Model Accuracy, Web Deployment

3.3 Data Flow Diagram

Level 0:
User → Web UI → Prediction Model → Output
Level 1:
Input → Preprocessing → ML Model → Prediction → Display

3.4 Technology Stack

- Python (ML models)
- Flask (Web backend)
- HTML/CSS (Frontend)
- Scikit-learn, XGBoost
- Pandas, NumPy
- Jupyter Notebook
- IBM Cloud (Deployment)

4. PROJECT DESIGN

4.1 Problem Solution Fit

Manual traffic control and reactive planning cause inefficiencies. Predictive modeling offers proactive traffic management, reducing bottlenecks and delays.

4.2 Proposed Solution

A regression-based ML model is trained on weather and timestamp data to estimate traffic volume. The model is embedded in a Flask app for interactive use.

4.3 Solution Architecture

- Frontend: HTML/CSS Forms
- Backend: Flask App
- Model: Trained Random Forest/XGBoost
- Storage: Model Pickle Files
- Flow: Input → Model → Prediction → Output via Web UI

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

Sprint	Tasks
1	Dataset Cleaning, Visualization
2	Algorithm Comparison, Model Building
3	Flask App & UI Integration
4	Deployment, Testing, Report Writing

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

Model Accuracy: >97% R^2 score

RMSE: Low for Random Forest

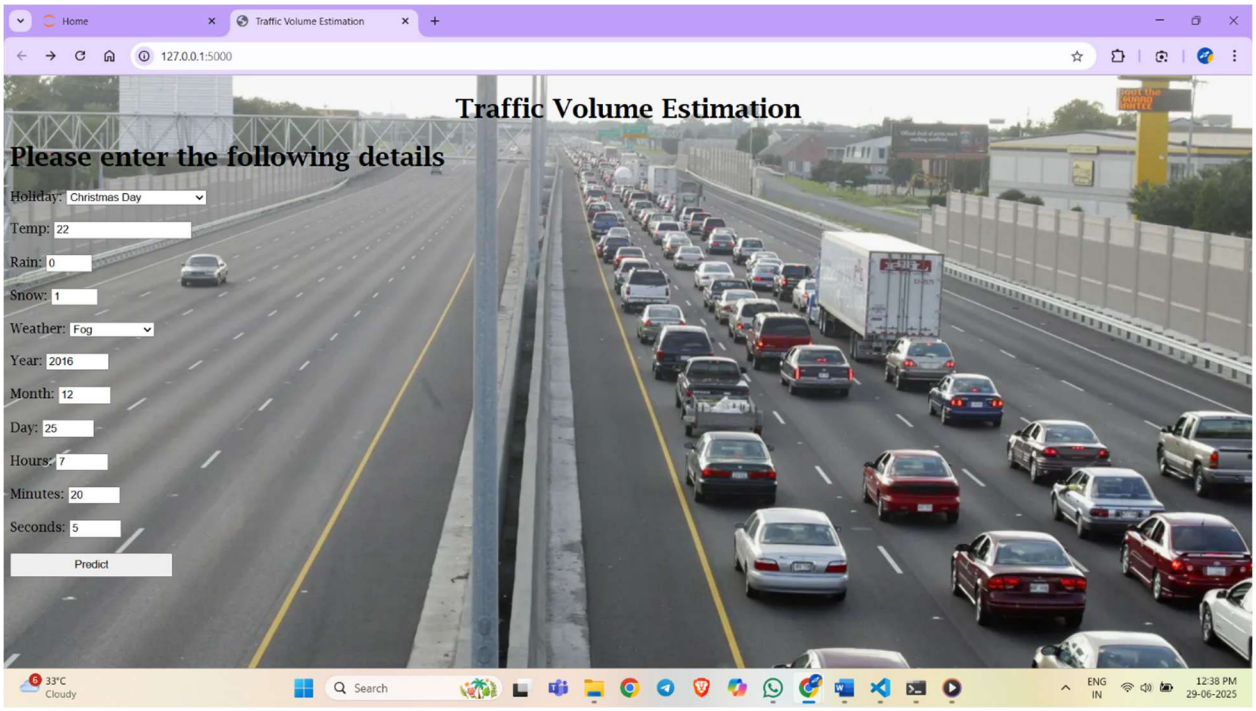
Response Time: <2s for prediction

Web Load Time: ~2s average

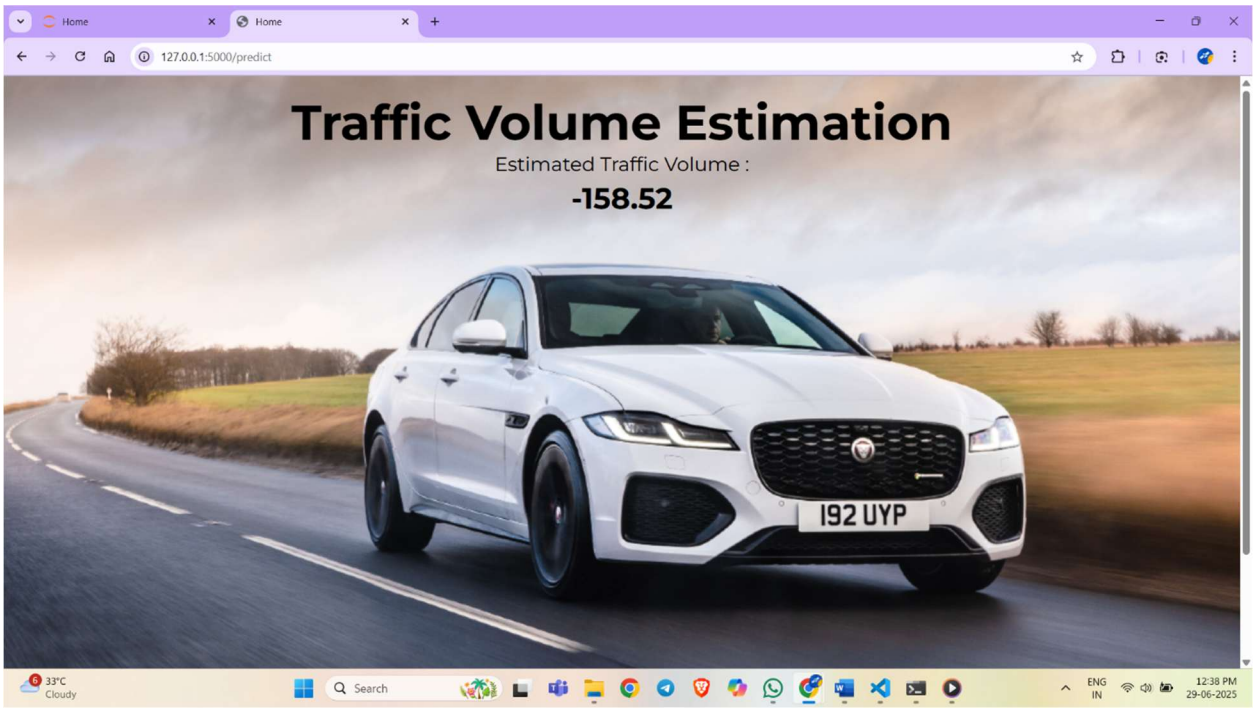
7. RESULTS

7.1 Output Screenshots

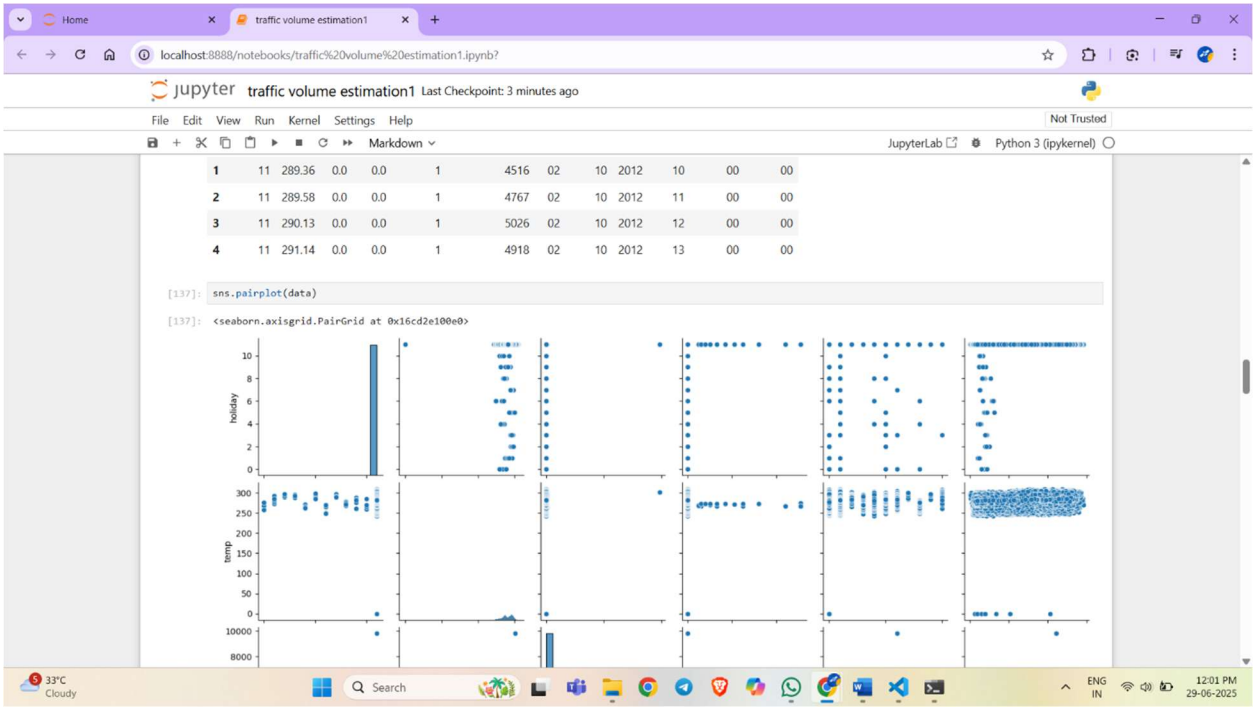
User Input Page:

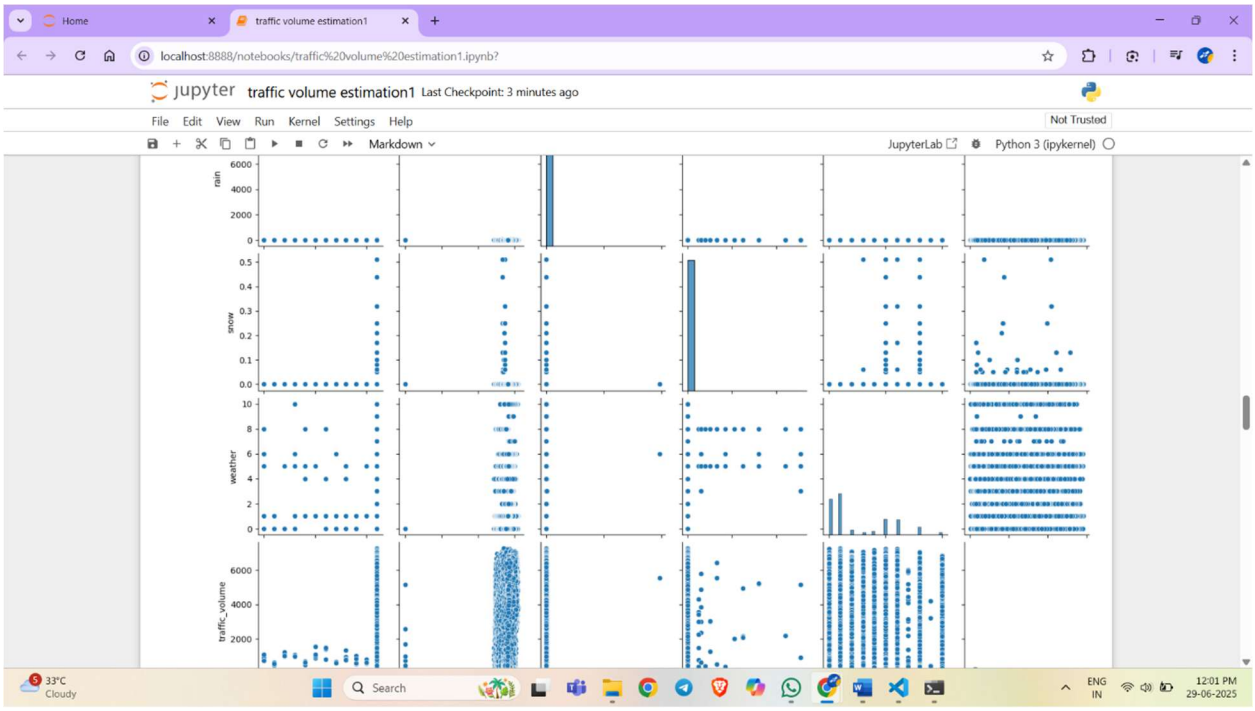


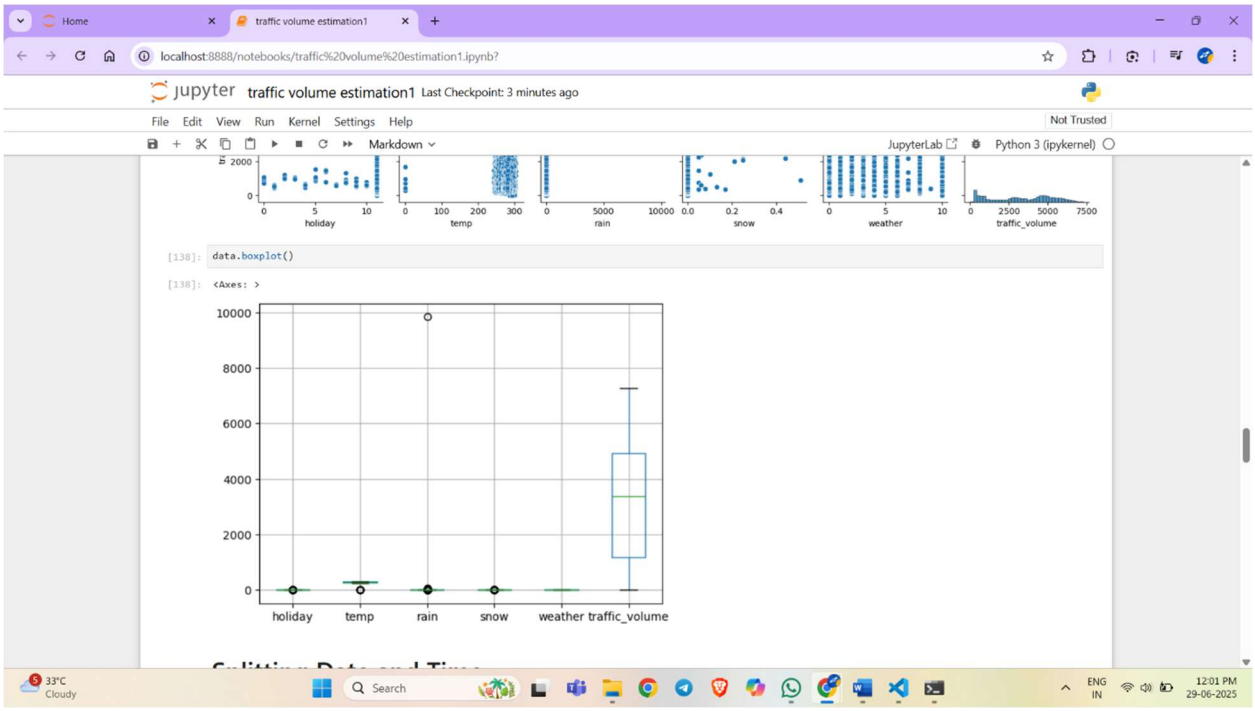
Traffic Prediction Page:



Visualizations (Pairplot, Boxplot):







8. ADVANTAGES & DISADVANTAGES

Advantages:

- Accurate real-time prediction
- Easy web-based access
- Scalable and user-friendly

Disadvantages:

- Needs retraining for new data
- Dependent on input quality
- Limited to regression-based features

9. CONCLUSION

This project demonstrates how machine learning can be leveraged to provide actionable traffic insights. With accurate predictions, it supports proactive decision-making and contributes to smarter mobility in urban areas.

10. FUTURE SCOPE

- Live GPS Data Integration
- Mobile App Deployment

- SHAP/LIME Model Explainability
- Multi-City/Multi-Dataset Training
- Real-time Traffic Alerts via API

11. APPENDIX

Source Code:

- **PYTHON CODE USED IN JUPYTER NOTEBOOK**

```
# Importing the neccessary libraries
import pandas as pd
import numpy as np
import seaborn as sns
import sklearn as sk
from sklearn import linear_model
from sklearn import tree
from sklearn import ensemble
from sklearn import svm

# Importing the Dataset
data=pd.read_csv(r"D:\Traffic Volume Estimation - SmartBridge\Traffic-intelligence\TrafficTelligence-Advanced-Traffic-Volume-Estimation-with-Machine-Learning-1\traffic_volume.csv")

# Analysing the Data
data.head()

data.describe()

data.info()
```

```
# Checking the null values
data.isnull().sum()

# Handling the missing values
data['temp'].fillna(data['temp'].mean(),inplace=True)
data['rain'].fillna(data['rain'].mean(),inplace=True)
data['snow'].fillna(data['snow'].mean(),inplace=True)

from collections import Counter

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

data['weather'].fillna('Clouds',inplace=True)

data.isnull().sum()

# Encoding the data
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

data['weather'] = le.fit_transform(data['weather'])

data['holiday'] = le.fit_transform(data['holiday'])

import matplotlib.pyplot as plt

sns.pairplot(data)

data.boxplot()

# Splitting Date and Time
```

```
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()

# Splitting The Dataset Into Dependent And Independent Variable
y = data['traffic_volume']
x = data.drop(columns=['traffic_volume'],axis=1)

names = x.columns

# Feature scaling
from sklearn.preprocessing import scale

x = scale(x)

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

x.head()

# Splitting The Data into Train and Test
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state =0)

# Training And Testing the Model
# Initializing the model
from sklearn import linear_model
from sklearn import tree
```

```
from sklearn import ensemble
from sklearn import svm
import xgboost

# Fitting the models with x_train and y_train
lin_reg = linear_model.LinearRegression()
Dtree = tree.DecisionTreeRegressor()
Rand = ensemble.RandomForestRegressor()
svr = svm.SVR()
XGB = xgboost.XGBRegressor()

# Fitting the models with x_train and y_train
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)

# Predicting the y_train values and calculate the accuracy
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)

# Regression Evaluation Metrics
from sklearn import metrics

#  R-squared _score
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))

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))

# RMSE –Root Mean Square Error
MSE = metrics.mean_squared_error(p3,y_test)

np.sqrt(MSE)

# Saving the Model
import pickle

pickle.dump(Rand,open("model.pkl",'wb'))
pickle.dump(le,open("encoder.pkl",'wb'))
```

PYTHON CODE USED FOR APP BUILDING

```
import numpy as np
import pickle

import time
import pandas
```

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



app = Flask(__name__,template_folder='Template')
model = pickle.load(open(r"D:\Traffic volume estimation
project\flask\Template\model.pkl",'rb'))


@app.route('/')# route to display the home page
def index():
    return render_template('index.html') #rendering the home page


@app.route('/predict',methods=["POST","GET"])# route to show the predictions in a web UI
def predict():
    #  reading the inputs given by the user
    input_feature=[float(x) for x in request.form.values() ]
    features_values=[np.array(input_feature)]
    names = [['holiday','temp', 'rain', 'snow', 'weather', 'year', 'month', 'day','hours', 'minutes',
'seconds']]
    data = pandas.DataFrame(features_values,columns=names)
     # predictions using the loaded model file
    prediction=model.predict(data)
    print(prediction)
    text = "Estimated Traffic Volume is :"
    return render_template("output.html",result = text + str(prediction) + "units")
     # showing the prediction results in a UI
if __name__=="__main__":


    # app.run(host='0.0.0.0', port=8000,debug=True)    # running the app
    port=int(os.environ.get('PORT',5000))
    app.run(port=port,debug=True,use_reloader=False)
```

- **HTML CODES USED**
  - **Index.html**

```
<!DOCTYPE html>

<html >

<head>

 <meta charset="UTF-8">

 <title>Traffic Volume Estimation</title>

</head>


<body background="https://cdn.vox-
cdn.com/thumbor/voARJfEKvTp6iMSzW3ExPn06TDM=/0x78:3000x1766/1600x90
0/cdn.vox-cdn.com/uploads/chorus_image/image/44219366/72499026.0.0.jpg"
text="black">


 <div class="login">

     <center><h1>Traffic Volume Estimation</h1></center>

    <!-- Main Input For Receiving Query to our ML -->

   <form action="{{ url_for('predict')}}"method="post">

<h1>Please enter the following details</h1>




</style></head>

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

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

```
            <option value=7>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 Years Day</option>
            <option value=11>Washingtons 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>   <br>
<br>   <label>temp:</label>
    <input type="number"   name="temp" placeholder="temp   " required="required"
/><br>

 <br>
     <label>rain:</label>
    <input type="number"  min="0" max="1"  name="rain       " placeholder="rain"
required="required" /><br>

<br>
```

```
 <label>snow:</label>
<input type="number" min="0" max="1"   name="snow      " placeholder="snow
" required="required" /><br>
<br>
<label for="weather">weather:</label>
 <select id="weather" name="weather">
    <option value=1>Clouds</option>
    <option value=0>Clear</option>
    <option value=6>Rain</option>
    <option value=2>Drizzle</option>
    <option value=5>Mist</option>
    <option value=4>Haze</option>
    <option value=3>Fog</option>
    <option value=10>Thunderstorm</option>
    <option value=8>Snow</option>
    <option value=9>Squall</option>
    <option value=7>Smoke</option><
 </select>   <br>
<br>
 <label>year:</label>
<input type="number" min="2012" max="2022"   name="year       "
placeholder="year " required="required" /><br>
```

```
<br>

        <label>month:</label>

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


<br>

        <label>day:</label>

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


<br>

     <label>hours:</label>

    <input type="number" min="0" max="24"   name="hours    " placeholder="hours
    " required="required" /><br>


<br>

        <label>minutes:</label>

    <input type="number" min="0" max="60"   name="minutes          "
placeholder="minutes      " required="required" /><br>


<br>

     <label>seconds:</label>

    <input type="number" min="0" max="60"   name="seconds          "
placeholder="seconds      " required="required" /><br>
```

```
<br>

<br><br>

<button type="submit" class="btn btn-primary btn-block btn-large"
style="height:30px;width:200px">Predict</button>


   </form>

<br>


  {{ prediction_text }}

 <br>

 <br>

 <img src="{{url_3}}" alt="Submit Form" height="180"
width="233" onerror="this.style.display='none'"/>

 <img src="{{url_1}}" alt="Submit Form" height="180"
width="233" onerror="this.style.display='none'"/>

 <img src="{{url_4}}" alt="Submit Form" height="180"
width="233" onerror="this.style.display='none'"/>

 <br>

 <br>

 <img src="{{url_2}}" alt="Submit Form" height="150"
width="711" onerror="this.style.display='none'"/>


</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;
}
.pd{
padding-bottom:45%;}
}
</style>
</head>
<body>

<br>
<center><b class="pd"><font color="black" size="15" font-family="Comic Sans MS" >Traffic volume estimation</font></b></center><br><br>
<div>
<br>
<center>
<p><font color="black"> {{result}} </p>
```

```
</center>
</div>
</body>
</html>
```

Dataset Link : https://drive.google.com/file/d/1WbNitFvPG9JkANAsdxQ16E6fuTCiVMXc/view?usp=drive_link

GitHub & Project Demo Link :

Project Demo Link: https://drive.google.com/file/d/1y-OGL7YDrqdFWCsSvuq0lBYwtSlEmMIT/view?usp=drive_link

GitHub Link: https://github.com/sowjanya002/TrafficTelligence-Advanced-Traffic-Volume-Estimation-with-Machine-Learning.git