📎11 Appendix – TrafficTelligence

The appendix includes additional resources, references, and supporting visuals that complement the core content of the project report.

🗀 A. Project File Structure

css

CopyEdit

TrafficTelligence/

├── app.py → Flask backend application

├── model_training.py → ML model training script

├── traffic_model.pkl → Saved trained model

├── traffic_volume.csv → Dataset used

├── templates/

│ └── index.html → Frontend UI (HTML form)

├── static/

│ └── style.css → Frontend styling

□ B. Sample Input & Output

Input Feature	Sample Value
Temperature (temp)	294.15 K
Rain in 1h (rain_1h)	0.0 mm
Snow in 1h (snow_1h)	0.0 mm
Clouds (clouds_all)	40 %

Output: Traffic Volume **≈ 4567 vehicles/hour**

🖼 C. Supporting Screenshots

- EDA Charts (Heatmap, Pairplot)

- UI with Prediction Result
- Code Snapshots (Model training, Flask setup)

📎 **D. Useful Links**

- Flask Documentation
- scikit-learn Documentation
- OpenWeatherMap API *(for future integration)*

11.1 SOURCE CODE

PYTHON CODE USED IN JUPYTER NOTEBOOK

```
# Importing the necessary 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"C:\Users\ganir\OneDrive\Desktop\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

data.corr()
sns.heatmap(data.corr())

data.head()

sns.pairplot(data)
```

```
data.boxplot()

data.corr()

# 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)
```

Let us build an app.py flask file which is a web framework written in python for server-side scripting. Let's see step by step procedure for building the backend application.

In order to develop web API with respect to our model, we basically use the Flask framework which is written in python.

Line 1-9 We are importing necessary libraries like Flask to host our model request

Line 12 Initialise the Flask application

Line 13 Loading the model using pickle

Line 16 Routes the API URL

Line 18 Rendering the template. This helps to redirect to the home page. In this home page,we give our input and ask the model to predict

In line 23 we are taking the inputs from the form

Line 28 Feature Scaling the inputs

Line 31 Predicting the values given by the user

Line 32-35 if the output is false render no chance template If the output is True render chance template

Line 36 The value of __name__ is set to __main__ when the module run as the main program otherwise it is set to the name of the module .

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=/0x78:3000x1766/1600x900/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>
```

11.2 🗁 Dataset Links – TrafficTelligence

The following datasets were used for building, training, and evaluating the traffic volume prediction model:

◈ 1. Main Dataset Used

- **Name:** Metro Interstate Traffic Volume

- **Source:** Kaggle Dataset Link
- **Description:** Contains hourly traffic volume data from the I-94 interstate highway in Minnesota, USA, along with weather conditions from 2012 to 2018.

🔹 2. Local File Used in Project

- **File Name:** traffic_volume.csv
- **Contents:** Preprocessed version of the Kaggle dataset, including selected features:
    - temp, rain_1h, snow_1h, clouds_all, and traffic_volume

📌 Usage in Project

- Used for feature extraction and model training.
- Split into training (80%) and testing (20%) for performance evaluation.
- Supports supervised learning using regression models.

11.3 PROJECT DEMO LINK:

https://drive.google.com/file/d/1ICBD-e83NQ70bCP_hQypkCxJEfh1mWF6/view?usp=sharing