**TrafficTelligence: Advanced Traffic Volume Estimation With Machine Learning**

**1. Pre-requisites**

* Python (3.x)
* Jupyter Notebook or Google Colab
* Basic ML knowledge (regression models, metrics)
* Libraries: pandas, numpy, matplotlib, seaborn, scikit-learn, joblib, flask (for app)

**2. Prior Knowledge**

* Understanding of datasets (CSV format)
* Supervised learning (especially regression)
* Data cleaning, visualization
* Model deployment using Flask (optional but helpful)

**3. Project Objectives**

* Predict hourly traffic volume using real-world sensor data
* Apply preprocessing, modeling, and deployment techniques
* Visualize data trends and traffic patterns
* Build a simple web app to demonstrate prediction

**4. Project Flow**

1. Data Collection
2. Data Preprocessing
3. Exploratory Data Analysis
4. Model Building
5. Model Evaluation
6. Model Deployment (Flask + HTML)
7. Application Output

**5. Project Structure**

pgsql

CopyEdit

TrafficTelligence/

├── dataset/

│ └── Metro\_Interstate\_Traffic\_Volume.csv

├── model/

│ └── traffic\_model.pkl

├── templates/

│ └── index.html

├── app.py

├── traffic\_analysis.ipynb

└── requirements.txt

**6. Data Collection**

* Dataset: [Metro Interstate Traffic Volume Data (UCI)](https://archive.ics.uci.edu/ml/datasets/Metro+Interstate+Traffic+Volume)
* Features: weather, holiday, date\_time, etc.
* Target: traffic\_volume

**7. Download The Dataset**

Store it in dataset/Metro\_Interstate\_Traffic\_Volume.csv

**8. Data Pre-Processing**

**8.1 Import Necessary Libraries**

python

CopyEdit

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import r2\_score, mean\_squared\_error

import joblib

**8.2 Importing The Dataset**

python

CopyEdit

df = pd.read\_csv("dataset/Metro\_Interstate\_Traffic\_Volume.csv")

df.head()

**8.3 Analyse The Data**

python

CopyEdit

df.info()

df.describe()

df.isnull().sum()

**8.4 Handling Missing Values**

python

CopyEdit

df.dropna(inplace=True) # or fillna method

**9. Data Visualization**

python

CopyEdit

sns.histplot(df['traffic\_volume'], bins=50)

plt.xticks(rotation=45)

sns.boxplot(data=df, x='holiday', y='traffic\_volume')

sns.heatmap(df.corr(), annot=True, cmap='coolwarm')

**10. Splitting The Dataset Into Dependent And Independent Variable**

python

CopyEdit

X = df[['temp', 'rain\_1h', 'snow\_1h', 'clouds\_all']] # or other features

y = df['traffic\_volume']

**11. Feature Scaling**

python

CopyEdit

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

**12. Splitting The Data Into Train And Test**

python

CopyEdit

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X\_scaled, y, test\_size=0.2, random\_state=42)

**13. Model Building**

python

CopyEdit

model = LinearRegression()

**14. Training And Testing The Model**

python

CopyEdit

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

**15. Model Evaluation**

python

CopyEdit

r2 = r2\_score(y\_test, y\_pred)

rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

print(f"R² Score: {r2:.2f}")

print(f"RMSE: {rmse:.2f}")

**16. Save The Model**

python

CopyEdit

joblib.dump(model, 'model/traffic\_model.pkl')

joblib.dump(scaler, 'model/scaler.pkl')

**17. Application Building**

A simple web app using Flask.

**18. Build HTML Code (templates/index.html)**

html

CopyEdit

<!DOCTYPE html>

<html>

<head>

<title>TrafficTelligence</title>

</head>

<body>

<h2>Traffic Volume Estimation</h2>

<form action="/predict" method="post">

Temperature: <input type="text" name="temp"><br>

Rain (1h): <input type="text" name="rain"><br>

Snow (1h): <input type="text" name="snow"><br>

Clouds All: <input type="text" name="clouds"><br>

<input type="submit" value="Predict">

</form>

</body>

</html>

**19. Main Python Script (app.py)**

python

CopyEdit

from flask import Flask, render\_template, request

import joblib

import numpy as np

app = Flask(\_\_name\_\_)

model = joblib.load('model/traffic\_model.pkl')

scaler = joblib.load('model/scaler.pkl')

@app.route('/')

def home():

return render\_template('index.html')

@app.route('/predict', methods=['POST'])

def predict():

features = [float(request.form['temp']),

float(request.form['rain']),

float(request.form['snow']),

float(request.form['clouds'])]

scaled\_features = scaler.transform([features])

prediction = model.predict(scaled\_features)[0]

return f"<h3>Estimated Traffic Volume: {int(prediction)} vehicles/hour</h3>"

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**20. Run The App**

bash

CopyEdit

python app.py

Navigate to http://127.0.0.1:5000

**21. Output**

You will see a simple form in the browser. After entering values, the app predicts **estimated traffic volume** in vehicles/hour.