**Project Report Format**

1. **INTRODUCTION**

1.1 Project Overview

TrafficTelligence aims to revolutionize traffic volume estimation by leveraging machine learning for dynamic, real-time, and predictive analytics. Traditional traffic systems often rely on static models and outdated sensors, leading to inefficiencies. This project uses modern data sources—like GPS, IoT sensors, and historical data—processed through machine learning algorithms to optimize urban mobility.

1.2 Purpose

To design and develop a scalable, intelligent system capable of accurately predicting traffic volumes, enabling smart infrastructure planning, signal optimization, and smoother commuter experiences.

2. **IDEATION PHASE**

2.1 Problem Statement

Urban mobility suffers from inaccurate traffic forecasts, resulting in congestion, delays, and frustration. Municipal authorities lack adaptive, cost-effective tools that can predict and manage traffic flow in real time.

2.2 Empathy Map Canvas

2.3 Brainstorming

Real-time congestion forecasting ML-based adaptive signal control

* Visual heatmaps for high-traffic zones
* Public transport & event integration
* Emergency vehicle routing optimization

3. **REQUIREMENT ANALYSIS**

3.1 Customer Journey map



3.2 Solution Requirement

* Input: Live traffic feeds, GPS, historical flow data Output: Real-time volume prediction, visual dashboards
* Functional: Live updates, visualization, alerts
* Non-Functional: Scalability, low latency, fault tolerance

3.3 Data Flow Diagram

| Sources +-------->+ Data Processor +------->+ Database +-------->+ UI/Dashboard

(ML Engine + Logic(Visualization) |

↑ ↓ ↑ ↑

| [Data Cleaning, [Model Results, [User queries,

| Preprocessing, Feature Raw + Refined] report generation,

| Extraction] traffic alerts]

Real-time

Traffic APIs,

IoT Sensors,

Historic Logs,

GPS Feeds

3.4 Technology Stack

Frontend – React.js, Angular, or HTML/CSS/JavaScript  
 Backend – Python with Flask or FastAPI  
 Machine Learning – Scikit-learn, TensorFlow, XGBoost  
 Database – PostgreSQL or MongoDB  
 Data Ingestion – REST APIs, Apache Kafka  
 Cloud Hosting – AWS, Azure, or Google Cloud  
 Visualization – Plotly, Dash, or D3.js

**4. PROJECT DESIGN**

4.1 Problem Solution Fit

The system directly addresses the inefficiency of current traffic models by introducing an adaptable, predictive, real-time solution using ML.

4.2 Proposed Solution

* Ingest real-time traffic data Clean and transform data streams
* Predict traffic volume using ML models
* Trigger alerts and visualize traffic load dynamically

4.3 Solution Architecture

1. Data Collection Layer (IoT Sensors, APIs) Data Processing & ML Layer
2. Data Storage Layer
3. Visualization & Notification Layer (UI + Alert Systems

**5. PROJECT PLANNING & SCHEDULING**

5.1 Project Planning

* **Week 1–2**: Define the problem, finalize scope, research current systems **Week 3–4**: Collect data (historical and real-time), clean and preprocess it
* **Week 5–6**: Train and evaluate machine learning models for traffic prediction
* **Week 7–8**: Develop backend APIs and connect models to the system
* **Week 9**: Build the frontend dashboard with visualizations
* **Week 10**: Test functionality, performance, and fix bugs
* **Week 11–12**: Document the project, prepare demo, finalize report and screenshots.

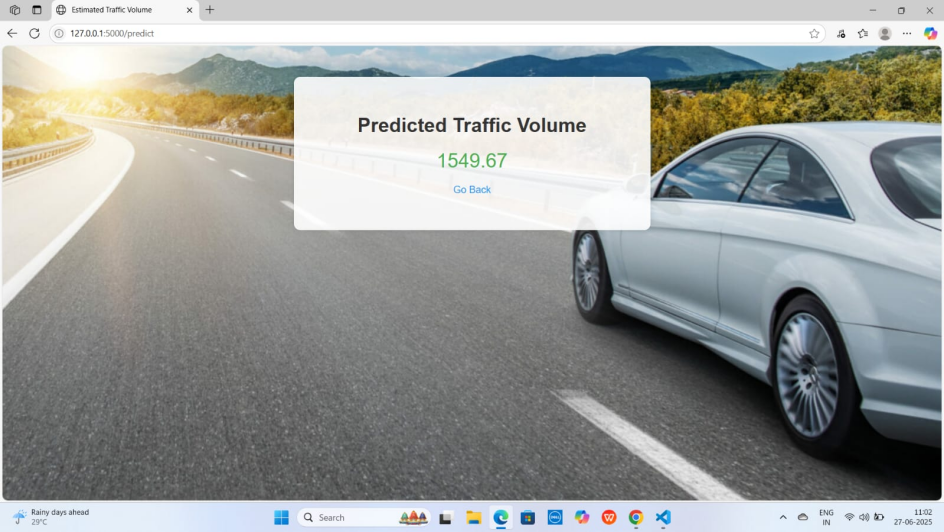
**6. FUNCTIONAL AND PERFORMANCE TESTING**

6.1 Performance Testing

* Simulated traffic datasets under high-load scenarios Stress testing ML model prediction times
* Latency analysis for real-time dashboards

**7. RESULTS**

7.1 Output Screenshots



**8. ADVANTAGES & DISADVANTAGES**

**Advantages**

* **Accurate real-time predictions**
* **Scalable architecture**
* **Helps reduce congestion and emissions**
* **Actionable insights for city planners**

**Disadvantages**

* **Dependent on data availability and sensor health**
* **Complex integration with existing infrastructure**
* **May need city-specific retrainin**

**9. CONCLUSION**

**TrafficTelligence presents a scalable, intelligent solution to modern urban mobility problems. It bridges the gap between traditional static systems and the need for adaptive, predictive traffic forecasting using machine learning**

**10. FUTURE SCOPE**

* **Integration with autonomous vehicle routing Drone-based traffic monitoring**
* **Reinforcement learning for self-optimizing signal control**
* **Citizen-facing travel suggestion app**

**11. APPENDIX**

Source Code(if any)

import pandas as pd

import numpy as np

from sklearn import linear\_model, tree, ensemble, svm, metrics

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

from sklearn.preprocessing import scale

import xgboost

import joblib

import os

# Load dataset

data = pd.read\_csv("C:/AI lab/traffic volume - traffic volume.csv")

# Handle missing values

data['temp'] = data['temp'].fillna(data['temp'].mean()) # still used as a feature

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

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

data['weather'] = data['weather'].fillna('Clouds')

data['holiday'] = data['holiday'].fillna('None')

# Feature engineering

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'], inplace=True)

# Convert time features to integers

for col in ['day', 'month', 'year', 'hours', 'minutes', 'seconds']:

data[col] = data[col].astype(int)

# One-hot encoding for categorical features

data = pd.get\_dummies(data, columns=['holiday', 'weather'], drop\_first=False)

# Define features and target

target = 'traffic\_volume'

features = data.columns.drop(target)

X = data[features]

y = data[target]

# Normalize numeric features

numeric\_cols = X.select\_dtypes(include=np.number).columns

X\_scaled = pd.DataFrame(scale(X[numeric\_cols]), columns=numeric\_cols)

X\_categorical = X.drop(columns=numeric\_cols).reset\_index(drop=True)

X\_final = pd.concat([X\_scaled, X\_categorical], axis=1)

# Train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_final, y, test\_size=0.2, random\_state=42)

# Define models

models = {

"Linear Regression": linear\_model.LinearRegression(),

"Decision Tree": tree.DecisionTreeRegressor(),

"Random Forest": ensemble.RandomForestRegressor(),

"SVR": svm.SVR(),

"XGBoost": xgboost.XGBRegressor()

}

# Train and evaluate models

print("Train R² Scores:")

for name, model in models.items():

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

score = model.score(X\_train, y\_train)

print(f"{name}: {score:.4f}")

print("\nTest R² Scores:")

for name, model in models.items():

predictions = model.predict(X\_test)

score = metrics.r2\_score(y\_test, predictions)

print(f"{name}: {score:.4f}")

# Save best model (example: Random Forest)

joblib.dump(models['Random Forest'], 'C:/AI lab/TrafficTelligence Advanced Traffic Volume Estimation with Machine Learning/model.joblib')

joblib.dump(list(X\_final.columns), 'C:/AI lab/TrafficTelligence Advanced Traffic Volume Estimation with Machine Learning/feature\_order.joblib')

Dataset Link

https://drive.google.com/file/d/1iV5PfYAmI6YP0\_0S4KYy1ZahHOqMgDbM/view?usp=sharing

GitHub & Project Demo Link

<https://github.com/Kambhampati-Sruthi/traffictelligence-advanced-traffic-volume-estimation-with-machine-learning>

[traffictelligence-project.mp4](https://drive.google.com/file/d/1d0Ns0EQd0pYqAcTq7ToLN5NHM5GawgQI/view?usp=sharing)https://drive.google.com/file/d/1d0Ns0EQd0pYqAcTq7ToLN5NHM5GawgQI/view?usp=sharing