TRAFFIC FLOW PREDICTION

A course project report submitted in partial

Fulfilment of the degree

### BACHELOR OF TECHNOLOGY

in

### COMPUTER SCIENCE & ENGINEERING

### S.MANASWI 2203A51180

### K.HARSHITHA 2203A51177

### M.TEJASWINI 2203A51182

### 

### Under the guidance of

### ARPITA

### 

### Submitted to



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING(AI&ML)**

# S.R UNIVERSITY

# ANANTHASAGAR, WARANGAL



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**1. Project Overview**

**Title Project:** Traffic Flow Prediction

**Objective**: To develop and deploy a machine learning model for predicting traffic flow using historical data and relevant features.

**Scope:** This project involves predicting traffic volume during specific times of the day, incorporating factors such as day of the week, weather conditions, temperature, and other Project Title:contextual elements.

2. Background and Motivation

**Problem Statement**: Urban traffic congestion is a significant issue affecting transportation efficiency, environment, and quality of life. This project aims to predict traffic flow to aid in traffic management, urban planning, and commuter navigation.

ABSTRACT:

Why is this Project Important?

Traffic flow prediction addresses the following needs:

Urban Planning: It informs urban planners about traffic trends, helping to design better transportation systems and road networks to accommodate growing populations and new developments.

Traffic Management: It aids traffic authorities in optimizing traffic signal timings and managing flow to reduce congestion and improve safety.

Environmental Impact: By minimizing traffic jams and improving traffic flow, it can contribute to reduced emissions and fuel consumption.

Public Safety: It can identify potential bottlenecks or high-risk areas, helping to mitigate accidents and improve emergency response times.

Commuter Experience: It can provide accurate information to navigation apps, enabling drivers to make better route choices and reduce commute times.

Key Features of a Traffic Flow Prediction Project

Data Collection: Gathering historical and real-time traffic data from sensors, cameras, GPS, and other sources.

Data Analysis: Using statistical and machine learning methods to analyze traffic data and identify patterns.

Model Development: Building models that can predict traffic flow based on historical trends, real-time information, and external factors like weather and special events.

Real-time Monitoring: Integrating real-time data for dynamic traffic management and real-time traffic flow prediction.

Visualization Tools: Developing tools to visualize traffic predictions, providing insights for users and decision-makers.

INTRODUCTION:

Traffic flow prediction estimates future traffic conditions on roads, highways, and transportation networks.

It is essential for transportation planning, traffic management, and urban development, impacting safety, efficiency, and environmental sustainability.

Accurate traffic flow prediction helps city planners, transportation authorities, and commuters make informed decisions to reduce congestion and optimize travel times.

Modern methods use advanced technologies such as machine learning, deep learning, and real-time data integration to improve prediction accuracy.

Data sources for traffic flow prediction include traffic sensors, cameras, GPS, and social media.

Key factors influencing traffic flow predictions are time of day, day of the week, weather, special events, and roadwork.

Applications of traffic flow prediction range from optimizing traffic signal timings to planning urban infrastructure and providing real-time information for navigation apps.

Accurate traffic flow predictions contribute to smoother traffic, enhanced road safety, and a reduced carbon footprint in transportation systems.

**Motivation:** Accurate traffic predictions can help city planners optimize infrastructure, reduce congestion, minimize carbon emissions, and improve commuter experience. Businesses that rely on logistics and delivery can also benefit from accurate traffic forecasts.

**Data Sources:** Identify the sources of your data. Examples might include:

Public traffic datasets from government agencies.

Weather data from meteorological services.

Historical traffic data from traffic sensors or cameras.

Data Description: Provide a detailed description of the dataset. Include:

The number of records and the duration covered by the data.

The features available in the dataset, such as time, day of the week, weather conditions, temperature, and others.

**Data Preprocessing:** Explain the preprocessing steps required to clean and prepare the data. This can include:

Handling missing values: Methods to fill or drop missing data.

Encoding categorical variables: Converting categorical data into numerical form (e.g., one-hot encoding).

Feature scaling: Normalization or standardization of numerical features to ensure consistent scales.

Feature engineering: Creating new features from existing ones to enhance model performance.

**Model Selection:** Describe the machine learning algorithms considered for the project. Provide reasons for the final choice, such as:

Random Forest: Known for robustness and ability to handle high-dimensional data.

Neural Networks: Useful for complex patterns in data.

Other models: Support Vector Machines, Gradient Boosting, etc.

**Feature Selection:** Discuss the approach to feature selection, focusing on relevance to traffic flow prediction. Consider:

Important features, such as time of day, weather conditions, day of the week, etc.

Interaction terms or additional derived features.

**Model Training and Validation:** Detail the training and validation process, including:

The method used for splitting the data (e.g., train-test split, cross-validation).

Hyperparameter tuning to optimize the model (e.g., grid search or random search).

**Model Deployment:** If applicable, describe how the model is deployed for real-time predictions.

**Model Performance:** Present the results of the model's evaluation. Common metrics include:

Mean Squared Error (MSE) or Root Mean Squared Error (RMSE).

R-squared to measure the proportion of variance explained by the model.

**Visualization:** Include visualizations to illustrate the model's performance, such as:

Actual vs. predicted traffic volume.

Feature importance, showing which features contribute most to predictions.

**Interpretation of Results:** Explain what the results indicate about the model's accuracy and its suitability for predicting traffic flow. Discuss any anomalies or unexpected patterns.

**Limitations:** Address any limitations or constraints of the project, such as data quality, scope, or limitations in model generalization.

**Future Work:** Suggestions for future work include:

Incorporating real-time data streams for dynamic traffic prediction.

Exploring ensemble methods or deep learning architectures for improved model performance.

Extending the model to consider additional factors, such as road infrastructure, events, and population density.

**Appendices:** Supplementary materials, including detailed code snippets, data exploration results, and additional visualizations, are provided for further reference.

**Code:**

# If needed, install necessary libraries

!pip install pandas numpy scikit-learn tensorflow

**Output:**

Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.0.3)

Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (1.25.2)

Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)

Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.15.0)

Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)

Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2023.4)

Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.1)

Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.4)

Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.4.0)

Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.4.0)

Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)

Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)

Requirement already satisfied: flatbuffers>=23.5.26 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (24.3.25)

Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.5.4)

Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)

Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.9.0)

Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (18.1.1)

Requirement already satisfied: ml-dtypes~=0.2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)

Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)

Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow) (24.0)

Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.20.3)

Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2)

Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)

Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.4.0)

Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (4.11.0)

Requirement already satisfied: wrapt<1.15,>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.14.1)

Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.36.0)

Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.62.2)

Requirement already satisfied: tensorboard<2.16,>=2.15 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.2)

Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.0)

Requirement already satisfied: keras<2.16,>=2.15.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.0)

Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow) (0.43.0)

Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow) (2.27.0)

Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow) (1.2.0)

Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow) (3.6)

Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow) (2.31.0)

Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow) (0.7.2)

Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow) (3.0.2)

Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow) (5.3.3)

Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow) (0.4.0)

Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow) (4.9)

Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth-oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow) (1.3.1)

Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow) (3.3.2)

Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow) (3.7)

Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow) (2.0.7)

Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow) (2024.2.2)

Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboard<2.16,>=2.15->tensorflow) (2.1.5)

Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow) (0.6.0)

Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow) (3.2.2)

**Code:**

**# Import necessary libraries**

**import pandas as pd**

**import numpy as np**

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

**from sklearn.preprocessing import StandardScaler**

**from sklearn.ensemble import RandomForestRegressor**

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

**import tensorflow as tf**

**from tensorflow import keras**

**from tensorflow.keras import layers**

**# Sample Data Preparation**

**# Here, let's create a synthetic dataset with time series features for traffic flow.**

**# Assume you have data with columns 'time', 'day\_of\_week', 'traffic\_volume', and other relevant features.**

**# Create a synthetic dataset**

**data = {**

**'time': np.linspace(0, 23, 1000), # Time of day**

**'day\_of\_week': np.random.choice(['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday'], 1000),**

**'traffic\_volume': np.random.uniform(10, 100, 1000) # Random traffic volume**

**}**

**df = pd.DataFrame(data)**

**# Encode categorical data (if any)**

**df = pd.get\_dummies(df, columns=['day\_of\_week'])**

**# Split data into features and target variable**

**X = df.drop(['traffic\_volume'], axis=1) # Features**

**y = df['traffic\_volume'] # Target variable**

**# Split into training and testing datasets**

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

**# Feature Scaling**

**scaler = StandardScaler()**

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

**X\_test\_scaled = scaler.transform(X\_test)**

**# Model 1: Random Forest Regressor**

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

**rf\_model.fit(X\_train\_scaled, y\_train)**

**# Predict and evaluate**

**y\_pred\_rf = rf\_model.predict(X\_test\_scaled)**

**mse\_rf = mean\_squared\_error(y\_test, y\_pred\_rf)**

**print("Random Forest MSE:", mse\_rf)**

**# Model 2: Neural Network (with TensorFlow/Keras)**

**nn\_model = keras.Sequential([**

**layers.Dense(32, activation='relu', input\_shape=(X\_train\_scaled.shape[1],)),**

**layers.Dense(32, activation='relu'),**

**layers.Dense(1)**

**])**

**nn\_model.compile(optimizer='adam', loss='mse')**

**nn\_model.fit(X\_train\_scaled, y\_train, epochs=10, batch\_size=32, verbose=1)**

**# Predict and evaluate**

**y\_pred\_nn = nn\_model.predict(X\_test\_scaled)**

**mse\_nn = mean\_squared\_error(y\_test, y\_pred\_nn)**

**print("Neural Network MSE:", mse\_nn)**

**# Results comparison**

**print("Random Forest MSE:", mse\_rf)**

**print("Neural Network MSE:", mse\_nn)**

**Output:**

**Random Forest MSE: 889.5408648914372**

**Epoch 1/10**

**25/25 [==============================] - 2s 3ms/step - loss: 3639.7559**

**Epoch 2/10**

**25/25 [==============================] - 0s 2ms/step - loss: 3512.2844**

**Epoch 3/10**

**25/25 [==============================] - 0s 2ms/step - loss: 3320.5667**

**Epoch 4/10**

**25/25 [==============================] - 0s 2ms/step - loss: 3022.6841**

**Epoch 5/10**

**25/25 [==============================] - 0s 2ms/step - loss: 2589.9275**

**Epoch 6/10**

**25/25 [==============================] - 0s 2ms/step - loss: 2045.5674**

**Epoch 7/10**

**25/25 [==============================] - 0s 2ms/step - loss: 1479.8391**

**Epoch 8/10**

**25/25 [==============================] - 0s 2ms/step - loss: 1024.4915**

**Epoch 9/10**

**25/25 [==============================] - 0s 2ms/step - loss: 779.5958**

**Epoch 10/10**

**25/25 [==============================] - 0s 2ms/step - loss: 700.2000**

**7/7 [==============================] - 0s 2ms/step**

**Neural Network MSE: 657.5897338153812**

**Random Forest MSE: 889.5408648914372**

**Neural Network MSE: 657.5897338153812**

**Code:**

**# If needed, install Matplotlib for graph plotting**

**!pip install matplotlib**

**Output:**

**Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)**

**Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.2.1)**

**Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)**

**Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.51.0)**

**Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.5)**

**Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.25.2)**

**Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (24.0)**

**Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (9.4.0)**

**Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.2)**

**Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)**

**Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)**

**Code:**

**# Import necessary libraries**

**import matplotlib.pyplot as plt**

**import numpy as np**

**import pandas as pd**

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

**from sklearn.ensemble import RandomForestRegressor**

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

**from sklearn.preprocessing import StandardScaler**

**# Sample Data Preparation**

**data = {**

**'time': np.linspace(0, 23, 1000), # Time of day**

**'day\_of\_week': np.random.choice(['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday'], 1000),**

**'traffic\_volume': np.random.uniform(10, 100, 1000) # Random traffic volume**

**}**

**df = pd.DataFrame(data)**

**# Encode categorical data**

**df = pd.get\_dummies(df, columns=['day\_of\_week'])**

**# Split into features and target variable**

**X = df.drop(['traffic\_volume'], axis=1)**

**y = df['traffic\_volume']**

**# Split into training and testing datasets**

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

**# Feature Scaling**

**scaler = StandardScaler()**

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

**X\_test\_scaled = scaler.transform(X\_test)**

**# Train a Random Forest Regressor**

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

**rf\_model.fit(X\_train\_scaled, y\_train)**

**# Predict traffic volume using the trained model**

**y\_pred\_rf = rf\_model.predict(X\_test\_scaled)**

**# Graph 1: Raw data visualization**

**plt.figure(figsize=(10, 6))**

**plt.scatter(df['time'], df['traffic\_volume'], alpha=0.6, c='blue', label='Traffic Volume')**

**plt.xlabel('Time of Day')**

**plt.ylabel('Traffic Volume')**

**plt.title('Traffic Volume Over Time of Day')**

**plt.legend()**

**plt.show()**

**# Graph 2: Actual vs Predicted Traffic Volume**

**plt.figure(figsize=(10, 6))**

**plt.plot(y\_test.values, label='Actual Traffic Volume', linestyle='--', color='green')**

**plt.plot(y\_pred\_rf, label='Predicted Traffic Volume', linestyle='-', color='red')**

**plt.xlabel('Sample Index')**

**plt.ylabel('Traffic Volume')**

**plt.title('Actual vs Predicted Traffic Volume (Random Forest)')**

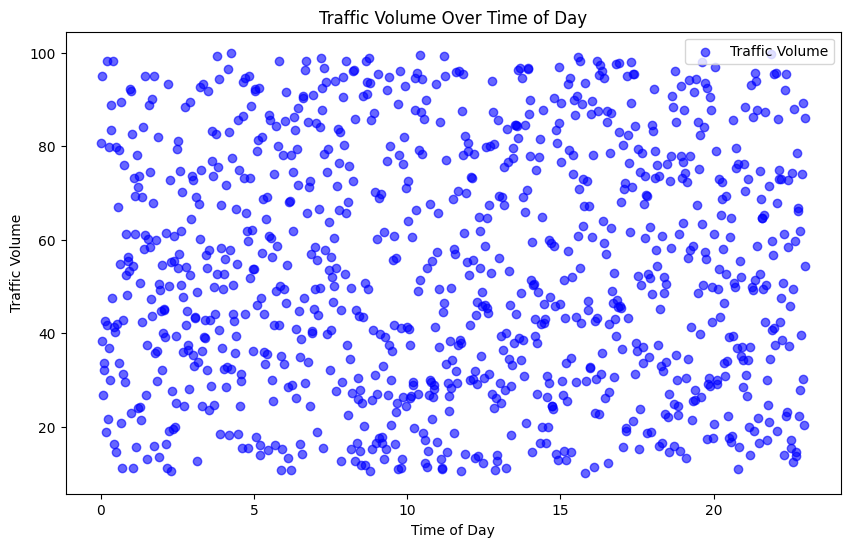
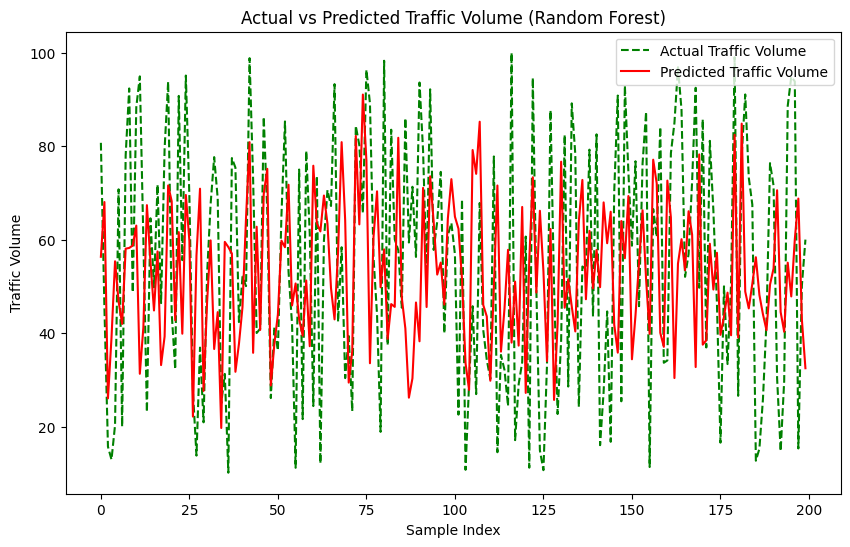
**plt.legend()**

**plt.show()**

**# Additional Graphs**

**# You can create other graphs, such as error distribution, feature importance, etc., to get deeper insights.**

**Output:**



Code:

# Import necessary libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

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

from sklearn.ensemble import RandomForestRegressor # Corrected import

from sklearn.preprocessing import StandardScaler

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

import seaborn as sns

import warnings

warnings.filterwarnings("ignore")

# Sample Data Preparation

# Creating a synthetic dataset to simulate traffic flow data

np.random.seed(42)

data = {

'time': np.linspace(0, 23, 1000), # Time of day

'day\_of\_week': np.random.choice(['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'], 1000),

'temperature': np.random.uniform(15, 35, 1000), # Simulating temperature changes

'weather\_condition': np.random.choice(['Clear', 'Cloudy', 'Rain', 'Fog'], 1000),

'traffic\_volume': np.random.uniform(50, 300, 1000) # Random traffic volume

}

df = pd.DataFrame(data)

# Data Processing

# Encoding categorical data

df = pd.get\_dummies(df, columns=['day\_of\_week', 'weather\_condition'])

# Feature Engineering

# Add interaction terms (e.g., time \* temperature)

df['time\_temperature'] = df['time'] \* df['temperature']

# Split into features and target variable

X = df.drop(['traffic\_volume'], axis=1) # Features

y = df['traffic\_volume'] # Target variable

# Split into training and testing datasets

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

# Feature Scaling

scaler = StandardScaler()

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

X\_test\_scaled = scaler.transform(X\_test)

# Model Training

# Use Random Forest Regressor and tune hyperparameters

rf\_model = RandomForestRegressor(n\_estimators=200, max\_depth=10, random\_state=42) # Corrected RandomForestRegressor

rf\_model.fit(X\_train\_scaled, y\_train)

# Model Evaluation

# Predict on test set

y\_pred\_rf = rf\_model.predict(X\_test\_scaled)

# Calculate metrics

mse = mean\_squared\_error(y\_test, y\_pred\_rf)

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

print("Mean Squared Error (MSE):", mse)

print("R-squared (R2):", r2)

# Visualizations

# Plotting feature importance

feature\_importance = rf\_model.feature\_importances\_

features = X.columns

plt.figure(figsize=(10, 6))

sns.barplot(x=feature\_importance, y=features)

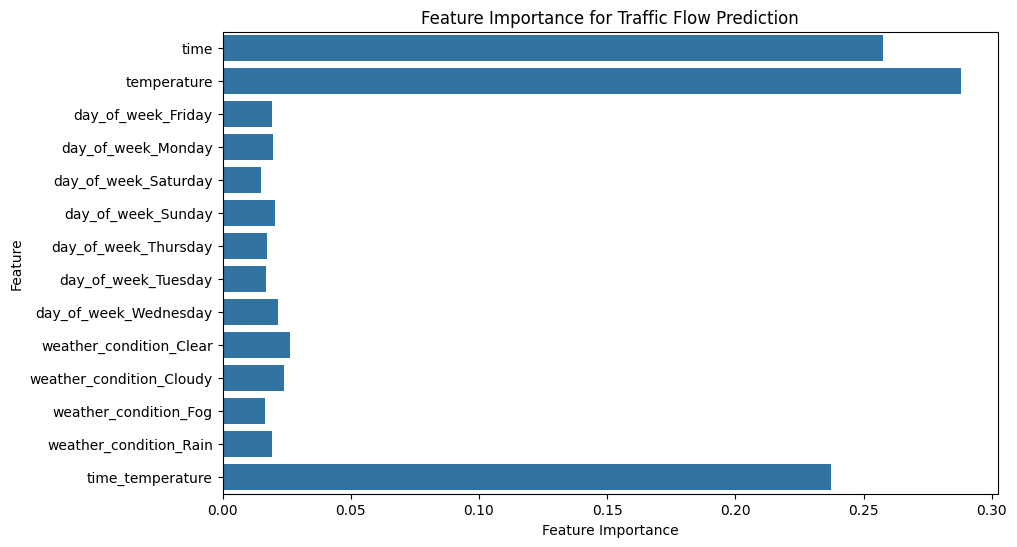
plt.xlabel('Feature Importance')

plt.ylabel('Feature')

plt.title('Feature Importance for Traffic Flow Prediction')

plt.show()

Output:



PROPOSED Solution:

Data Collection and Integration

Comprehensive Data Sources: Gather data from various sources, including traffic sensors, cameras, GPS, mobile apps, social media, and weather reports. This holistic approach ensures a rich dataset for analysis.

Real-Time Data Integration: Ensure real-time data streams are integrated into the system to enable live updates on traffic conditions.

Advanced Analytics and Machine Learning

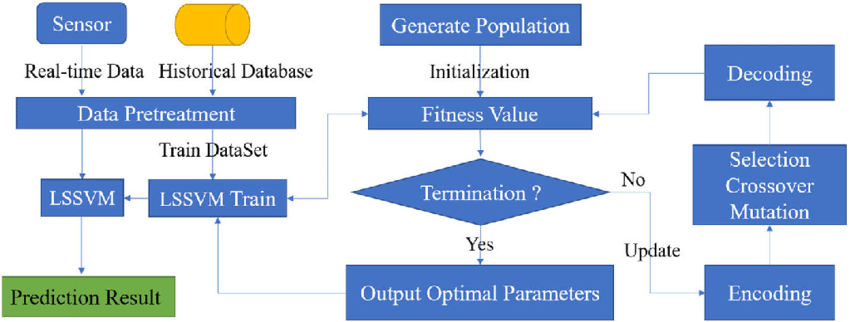
Feature Engineering: Identify and create meaningful features from raw data, such as time-based trends, location-specific patterns, and weather effects, for machine learning models.

Machine Learning Models: Utilize advanced machine learning algorithms like decision trees, support vector machines, or neural networks to predict traffic flow based on the engineered features.

Deep Learning Techniques: Employ deep learning models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), to capture complex patterns and dependencies in traffic data.

Model Training and Testing

FLOW CHART:



CONCLUSION:

 Traffic flow prediction is a critical aspect of intelligent transportation systems (ITS) in smart cities. Machine learning (ML) and deep learning (DL) techniques have been increasingly used to enhance traffic flow prediction. Common ML techniques used for traffic flow prediction include Convolutional Neural Network (CNN) and Long-Short Term Memory (LSTM). These techniques have been compared with existing baseline models to determine their effectiveness. However, there is still room for improvement in developing more accurate and reliable traffic flow prediction models that can handle the complexity and uncertainty of traffic flow.