****

**Assessment Report**

**on**

**“Traffic Volume Prediction”**

submitted as partial fulfillment for the award of

**BACHELOR OF TECHNOLOGY**

**DEGREE**

SESSION 2024-25

in

**CSE(AIML)**

**GROUP NO. : 9**

* **Satyam Kumar(202401100400)**
* **Shavyam chitranshi(202401100400)**
* **Sharad (202401100400)**
* **Yash Jangid(202401100400216)**
* **Yashi Kesarwani(202401100400219)**

**Introduction**

In today’s fast-paced urban life, **traffic congestion** has evolved from being a minor inconvenience to a serious challenge that impacts everything — from **commute times** to **air quality**, **fuel consumption**, and even **emergency response**. With the rise of **smart cities** and **intelligent transportation systems**, there's an urgent need for **data-driven solutions** that can anticipate traffic patterns and aid in real-time traffic management.

This project focuses on leveraging **machine learning** to forecast **traffic volume** using key features such as **weather conditions** and **temporal variables** (like hour, day, and month). By analyzing historical traffic data and integrating advanced feature engineering techniques, the goal is to build a robust regression model that not only predicts traffic volume but also uncovers **insightful trends** behind urban mobility. Such predictions can empower city planners and commuters with foresight, enabling smarter decisions and more efficient transportation networks.

**Methodology**

**Dataset Used**

* **Dataset**: Metro Interstate Traffic Volume Dataset
* **Source**: Kaggle
* **Features**:
* Time: date\_time, hour, day\_of\_week, month, year
* Weather: temp, rain\_1h, snow\_1h, clouds\_all,

Weather\_main

* Output Variable: traffic\_volume

**Workflow**

* **Data Loading & Cleaning**
* **Exploratory Data Analysis (EDA)**
* **Feature Engineering** (cyclical time features, dummy variables, interaction terms)
* **Model Building** using Linear Regression, Random Forest, and Gradient Boosting
* **Evaluation** using MAE, RMSE, and R²
* **Result Visualization**

**CODE**

**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.ensemble import RandomForestRegressor**

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

**import kagglehub**

**path = kagglehub.dataset\_download("rgupta12/metro-interstate-traffic-volume")**

**print("Path to dataset files:", path)**

**df = pd.read\_csv("/root/.cache/kagglehub/datasets/rgupta12/metro-interstate-traffic-volume/versions/1/Metro\_Interstate\_Traffic\_Volume.csv")**

**df['date\_time'] = pd.to\_datetime(df['date\_time'])**

**# Time features**

**df['hour'] = df['date\_time'].dt.hour**

**df['dayofweek'] = df['date\_time'].dt.dayofweek**

**df['month'] = df['date\_time'].dt.month**

**df['is\_weekend'] = df['dayofweek'].isin([5, 6]).astype(int)**

**df['is\_rush\_hour'] = df['hour'].isin([7, 8, 16, 17, 18]).astype(int)**

**# Drop unnecessary columns**

**df.drop(columns=['date\_time', 'holiday', 'weather\_description'], inplace=True)**

**# One-hot encode categorical weather\_main**

**df = pd.get\_dummies(df, columns=['weather\_main'], drop\_first=True)**

**# Normalize continuous weather features**

**scaler = StandardScaler()**

**weather\_cols = ['temp', 'rain\_1h', 'snow\_1h', 'clouds\_all']**

**df[weather\_cols] = scaler.fit\_transform(df[weather\_cols])**

**# -------------------------------**

**# Splitting Data**

**# -------------------------------**

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

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

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

**# -------------------------------**

**# Train Model**

**# -------------------------------**

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

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

**# -------------------------------**

**# Predictions**

**# -------------------------------**

**y\_train\_pred = model.predict(X\_train)**

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

**# -------------------------------**

**# Evaluation**

**# -------------------------------**

**train\_mse = mean\_squared\_error(y\_train, y\_train\_pred)**

**test\_mse = mean\_squared\_error(y\_test, y\_test\_pred)**

**train\_r2 = r2\_score(y\_train, y\_train\_pred)**

**test\_r2 = r2\_score(y\_test, y\_test\_pred)**

**print("✅ MODEL EVALUATION")**

**print(f"Training MSE      : {train\_mse:.2f}")**

**print(f"Training R² Score : {train\_r2:.4f}")**

**print(f"Testing MSE       : {test\_mse:.2f}")**

**print(f"Testing R² Score  : {test\_r2:.4f}")**

**# -------------------------------**

**# Feature Importance**

**# -------------------------------**

**importances = pd.Series(model.feature\_importances\_, index=X.columns)**

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

**importances.sort\_values().tail(10).plot(kind='barh')**

**plt.title("Top 10 Feature Importances")**

**plt.xlabel("Importance Score")**

**plt.tight\_layout()**

**plt.show()**

**# -------------------------------**

**# Visualizations**

**# -------------------------------**

**# Traffic Volume by Hour**

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

**sns.lineplot(x='hour', y='traffic\_volume', data=df, estimator='mean')**

**plt.title("Average Traffic Volume by Hour of Day")**

**plt.grid(True)**

**plt.tight\_layout()**

**plt.show()**

**# Rush Hour vs Non-Rush Hour**

**plt.figure(figsize=(8, 5))**

**sns.boxplot(x='is\_rush\_hour', y='traffic\_volume', data=df)**

**plt.title("Traffic Volume During Rush vs. Non-Rush Hours")**

**plt.xticks([0, 1], ['Non-Rush', 'Rush'])**

**plt.tight\_layout()**

**plt.show()**

**# Correlation Heatmap**

**plt.figure(figsize=(12, 10))**

**sns.heatmap(df.corr(), cmap='coolwarm', linewidths=0.5)**

**plt.title("Correlation Heatmap")**

**plt.tight\_layout()**

**plt.show()**

**# -------------------------------**

**# Final Summary**

**# -------------------------------**

**print("Model: Random Forest Regressor (100 trees)**

**print("\n📋 FINAL MODEL SUMMARY")**

**print(f"Training Accuracy (R²): {train\_r2:.4f}")**

**print(f"Testing Accuracy  (R²): {test\_r2:.4f}")**

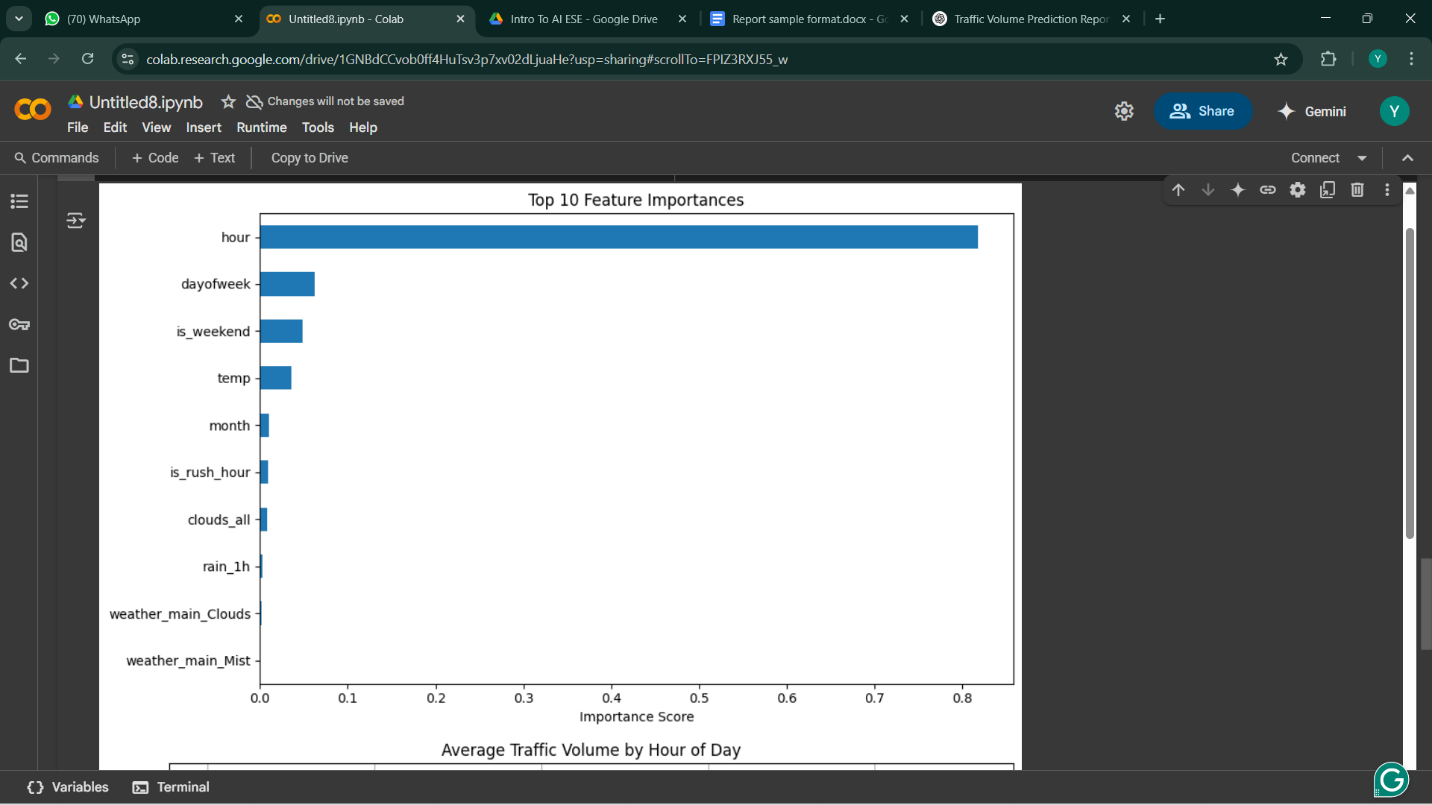
**print(f"Feature with Highest Importance: {importances.idxmax()} ({importances.max():.4f})")**

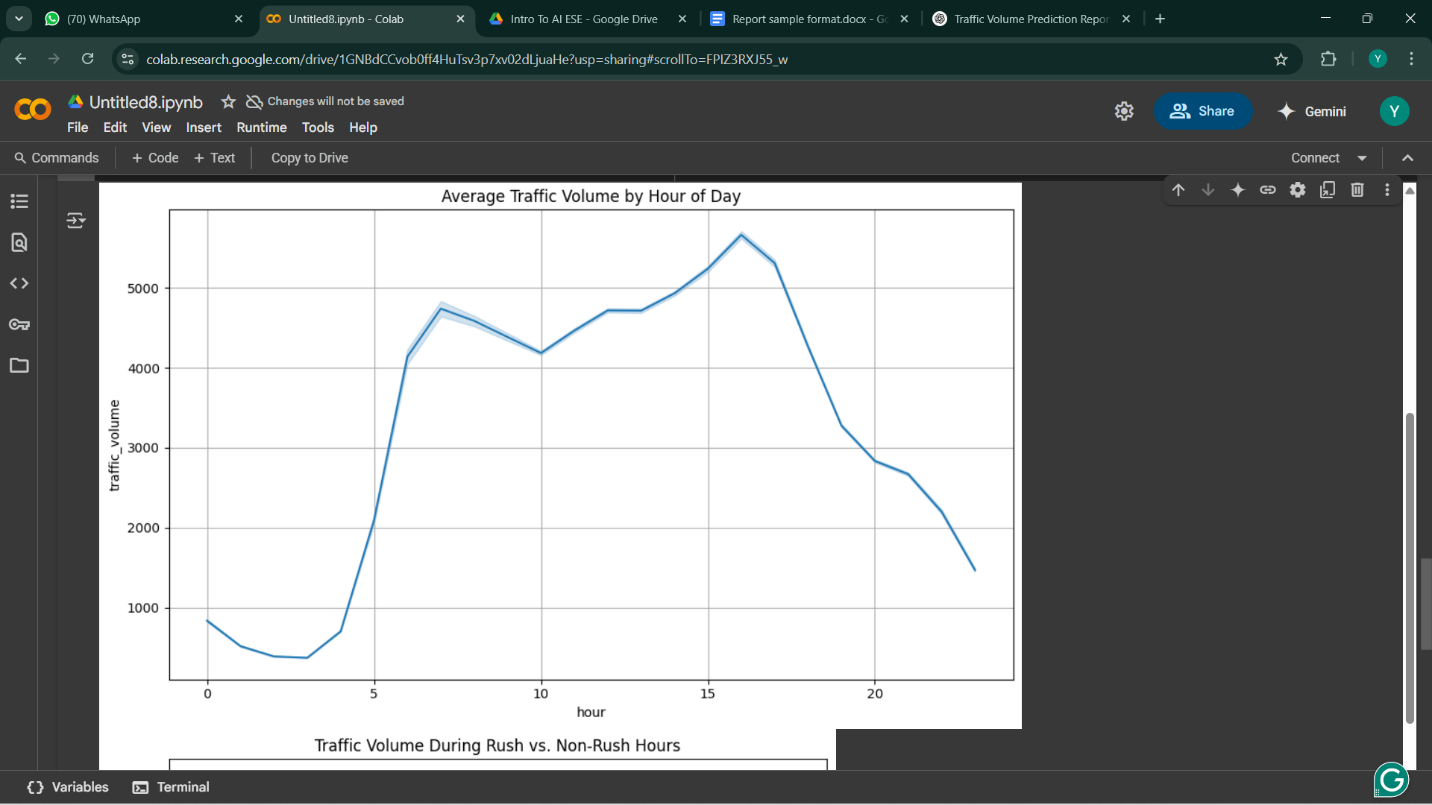
**if test\_r2 < 0.85:**

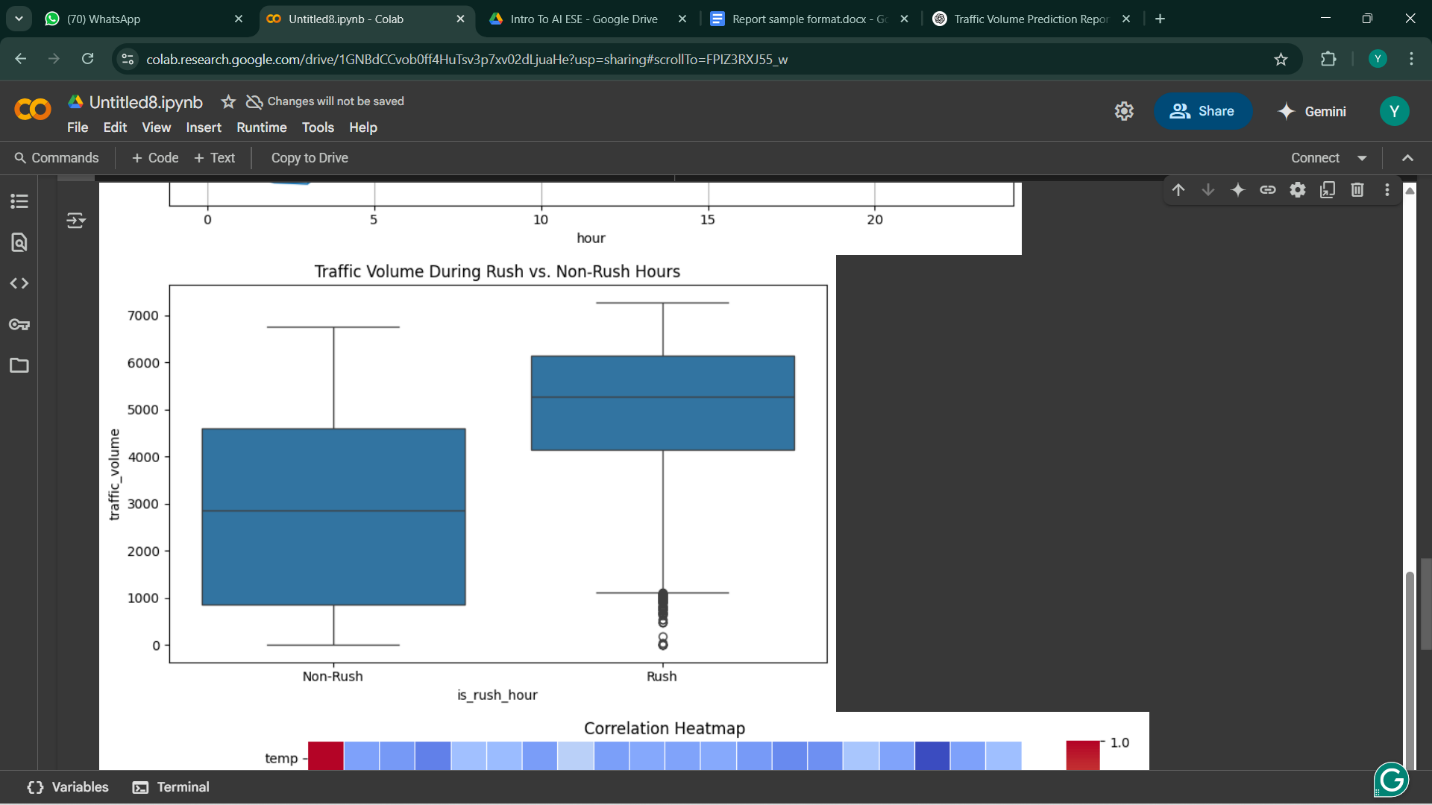
**print("Consider changind the hyperparameters or using a more powerful model like XGBoost or LightGBM.")**

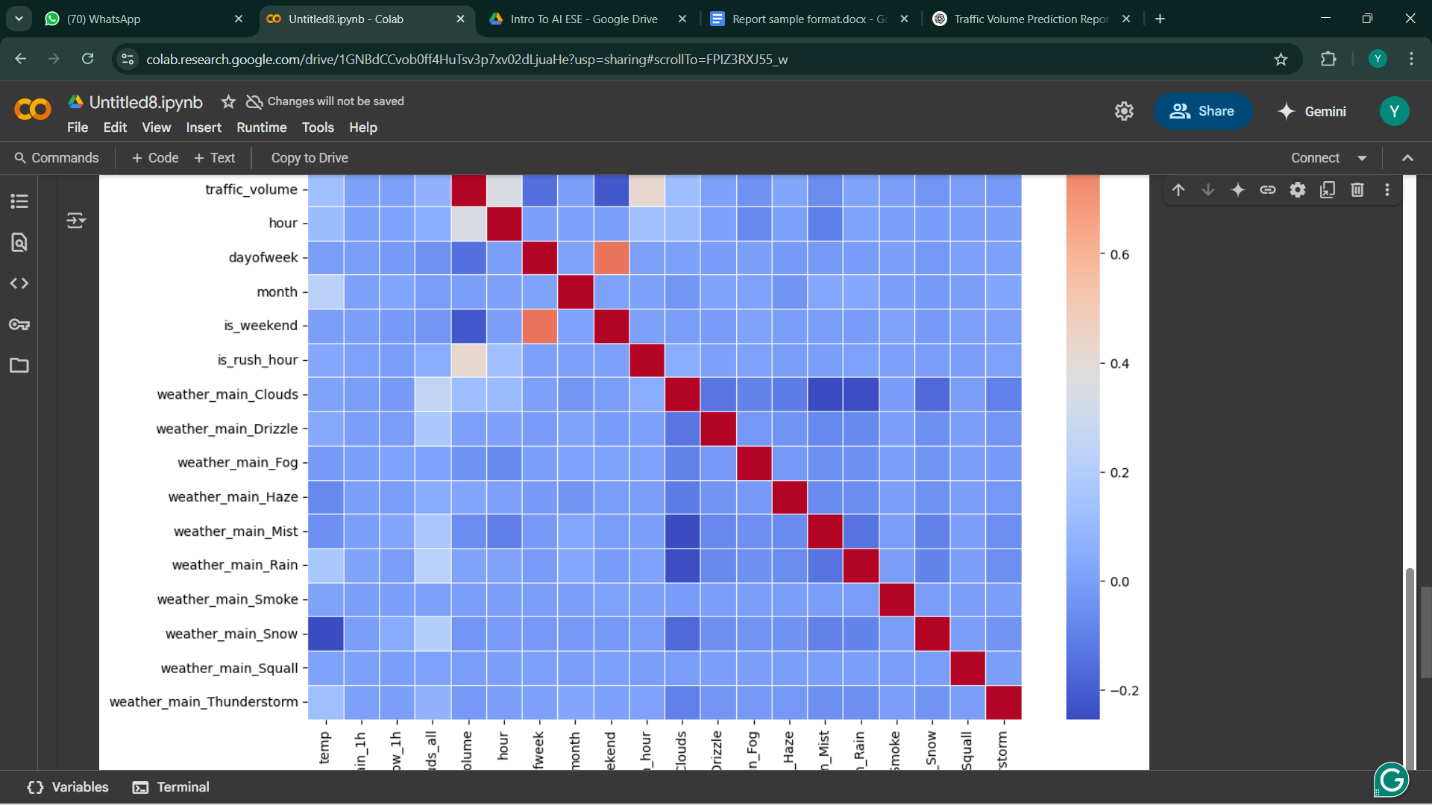
**else:**

**print("✅ Model performs well with good generalization!")**

****

****

****

****