Assessment Report

on

"Traffic Volume Prediction"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE AIML

By

Daksh Singh (202401100400073)

Devendra Kumar Yadav (202401100400079)

Deepanshu Bharadwaj (202401100400077)

Navishka Sharma (202401100400123)

Preeti Singh (202401100400146)

Under the supervision of

Abhishek Shukla

KIET Group of Institutions, Ghaziabad

Affiliated to

Dr. A.P.J. Abdul Kalam Technical University, Lucknow

(Formerly UPTU)

May, 2025

Project Report: Traffic Volume Prediction

Introduction

Traffic volume prediction is a critical aspect of intelligent transportation systems and urban planning. Accurately forecasting traffic patterns helps cities manage congestion, optimize infrastructure usage, and improve commuter safety and experience.

In this project, we will attempt to create a **regression model** to predict the **traffic volume** at a given time based on **weather conditions, time-related features, and possibly location data**. By leveraging historical data, we aim to discover patterns and influential factors that affect traffic flow.

The project will involve **data exploration**, **feature engineering**, **model building**, **evaluation**, and **interpretation of results**. The ultimate goal is to provide insights that could support smarter traffic control systems and better decision-making for city planners and logistics managers.

Dataset Overview

The dataset includes the following features:

- holiday: Type of day (e.g., None, Labor Day, etc.)
- temp: Temperature in Kelvin
- rain_1h: Amount of rain in mm in the last hour
- snow_1h: Amount of snow in mm in the last hour
- clouds_all: Cloud coverage percentage
- weather_main: Main weather condition (e.g., Clear, Clouds, Rain)
- weather_description: Detailed weather description
- date_time: Timestamp of observation
- traffic_volume (**Target Variable**): Number of vehicles observed in an hour

Methodology

Data Preparation

- **Load raw data**: The dataset is read using pandas.read_csv() from a preprocessed file (Metro-Interstate-Traffic-Volume-Encoded.csv).
- **Check for missing values** and inspect data types with .info() and .isnull().sum().
- **Handle missing or inconsistent values** (if any). Imputation or deletion may be applied.
- **Convert categorical features** to numeric using one-hot encoding (already encoded in the provided file).
- **Feature scaling** is performed on numerical variables using StandardScaler to normalize ranges and assist model convergence.

Exploratory Data Analysis (EDA)

- Use .describe() for summary statistics like mean, standard deviation, and quartiles.

- Plot **histograms** and **boxplots** to assess feature distributions and detect outliers.
- Construct a **correlation heatmap** to visualize relationships between features and target (traffic_volume).
- Optional: Visualize time-based trends, rush hour effects, and holiday traffic influence (if date_time column is present).

Feature Selection

- Evaluate **Pearson correlation** with the target variable.
- Use **feature importances** from tree-based models like Random Forest to identify high-impact variables.
- Discard low-variance or redundant features to reduce dimensionality and overfitting risk.

Model Selection

- Candidate models chosen for regression:

- Linear Regression (baseline)
- Random Forest Regressor
- XGBoost Regressor

- These models offer a balance of interpretability and performance.
- Models are compared using R^2 score, MAE, and RMSE.

Model Training

- The dataset is split into **training (80%) and testing (20%)** sets using train_test_split.
- Models are trained using .fit() on the scaled training data.
- Default hyperparameters are used, but can be optimized with grid/random search for improvement.

Evaluation

For each model, the following metrics are calculated on the test set:

- **Mean Absolute Error (MAE)**: Measures average prediction error.
- **Root Mean Squared Error (RMSE)**: Penalizes larger errors more heavily.
- **R^2 Score**: Indicates proportion of variance explained.

Feature Importance Visualization

- Feature importances are extracted from the trained **Random Forest** model.
- A **bar plot** shows the top 15 influential features affecting traffic volume.

Step-by-Step Breakdown

Step 1: Load and Explore the Dataset

- Load the dataset using pandas.read_csv():
- Inspect the structure using .info() and .describe() to understand data types and distribution:
- Preview the dataset with .head() and check dimensions:

Step 2: Clean Missing and Invalid Values

- Check for missing or null values:
- No obvious placeholders or missing values were found; hence, no imputation required in this dataset.
- If nulls were present: numeric features (e.g. temp) would be filled with median; categorical ones with mode or "Unknown".

Step 3: Feature Engineering & Scaling

- Define the **features** and **target**:

```
X = df.drop('traffic_volume', axis=1)

y = df['traffic_volume']
```

- Use StandardScaler to scale numerical features:
- Categorical features were assumed already encoded (e.g., weather_main, holiday, etc.).

Step 4: Exploratory Data Analysis (EDA)

- Plot **histograms** and **boxplots** to examine distributions and detect outliers:
- Generate a **correlation heatmap** to understand relationships:

Step 5: Train–Test Split

- Divide the dataset:
- No stratification needed for regression tasks.

Step 6: Model Training

Train three regression models

Step 7: Model Evaluation

- Evaluate models using:
    - **MAE**: Mean Absolute Error
    - **RMSE**: Root Mean Squared Error
    - **R^2 Score**: Coefficient of determination
- Results printed for all models after training.

Step 8: Interpretation and Summary

- Get **feature importance** from Random Forest:
- **Best Model**: Random Forest or XGBoost (based on R^2 score).
- **Important Features**: Weather condition, temperature, cloud cover, rain/snow metrics.

- **Opportunities for Improvement**:
  - Feature engineering using date_time (hour, weekday, weekend).
  - Hyperparameter tuning using GridSearchCV.
  - Integrating live traffic, road conditions, or holiday calendars.

Code Implementation

```
#  Step 1: Install & Import Required Libraries
!pip install xgboost --quiet

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.ensemble import RandomForestRegressor
from xgboost import XGBRegressor

from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

#  Step 2: Load Dataset (path must match your uploaded CSV)
df = pd.read_csv('/content/extracted/Metro-Interstate-Traffic-Volume-Encoded.csv')

#  Step 3: Data Exploration
print("Shape:", df.shape)
print("\nInfo:")
print(df.info())
print("\nMissing values:")
print(df.isnull().sum())
print("\nSummary statistics:")
print(df.describe())

#  Step 4: Define Features and Target
if 'traffic_volume' not in df.columns:
   raise KeyError("Expected column 'traffic_volume' not found.")

X = df.drop('traffic_volume', axis=1)
y = df['traffic_volume']

#  Step 5: Data Preprocessing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
```

```
X_test = scaler.transform(X_test)

#  Step 6: Train Regression Models and Evaluate
models = {
    'Linear Regression': LinearRegression(),
    'Random Forest': RandomForestRegressor(n_estimators=100, random_state=42),
    'XGBoost': XGBRegressor(n_estimators=100, random_state=42)
}

for name, model in models.items():
    model.fit(X_train, y_train)
    preds = model.predict(X_test)
    print(f"\n� {name} Performance:")
    print(f"MAE: {mean_absolute_error(y_test, preds):.2f}")
    rmse = np.sqrt(mean_squared_error(y_test, preds))
    print(f"RMSE: {rmse:.2f}")
    print(f"R² Score: {r2_score(y_test, preds):.2f}")

#  Step 7: Feature Importance from Random Forest
importances = models['Random Forest'].feature_importances_
features = X.columns

feat_imp = pd.Series(importances, index=features).sort_values(ascending=False)
plt.figure(figsize=(14, 6))
feat_imp[:15].plot(kind='bar')
plt.title('Top 15 Feature Importances (Random Forest)')
plt.ylabel('Importance Score')
plt.grid(True)
plt.show()

# Summary
best_model = 'Random Forest or XGBoost (based on R²)'
print(f"Best model: {best_model}")
print("Key influential features include: weather conditions, temperature, etc.")
print("Note: Time-based plots skipped as 'date_time' column is not present in the dataset.")
```

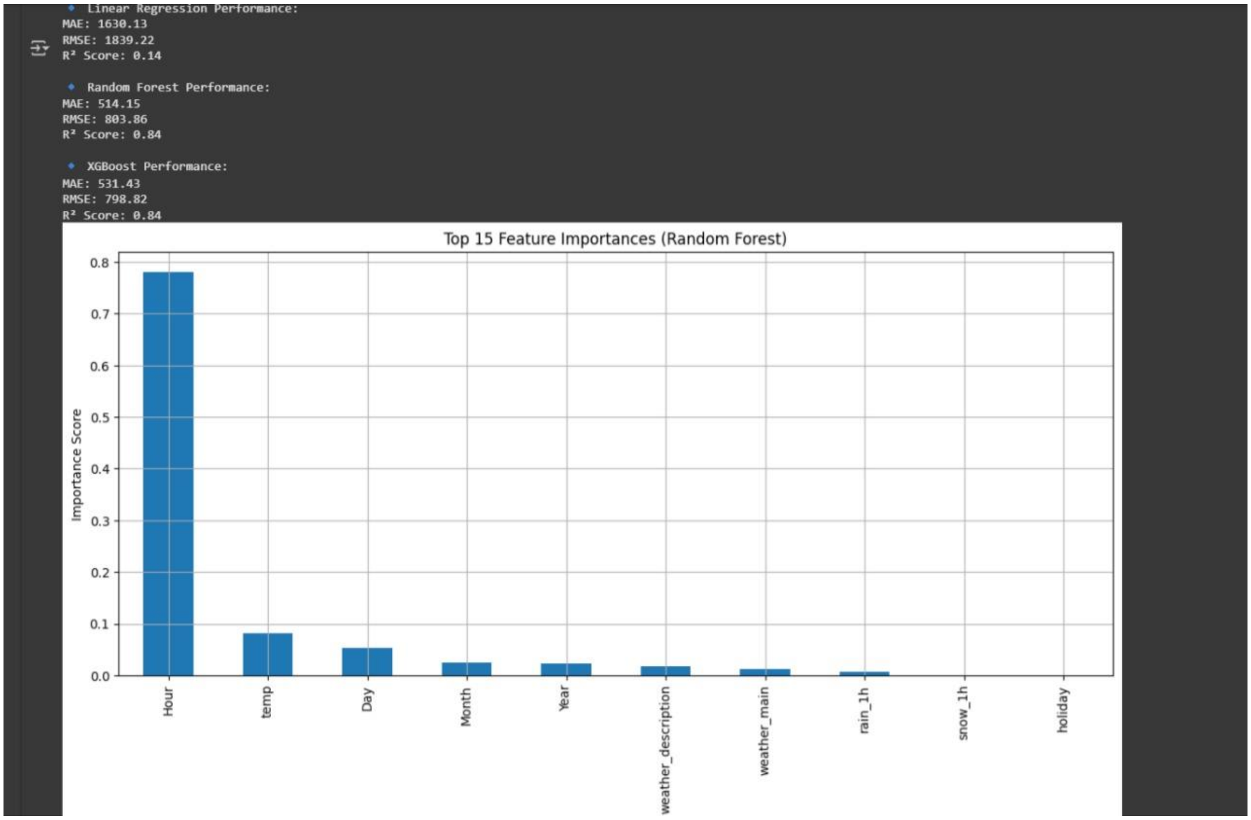
Output/Result

```
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   holiday              48204 non-null  int64
 1   temp                 48204 non-null  float64
 2   rain_1h              48204 non-null  float64
 3   snow_1h              48204 non-null  float64
 4   Year                 48204 non-null  int64
 5   Month                48204 non-null  int64
 6   Day                  48204 non-null  int64
 7   Hour                 48204 non-null  int64
 8   weather_main         48204 non-null  int64
 9   weather_description  48204 non-null  int64
 10  traffic_volume       48204 non-null  int64
dtypes: float64(3), int64(8)
memory usage: 4.0 MB
None

Missing values:
holiday                0
temp                   0
rain_1h                0
snow_1h                0
Year                   0
Month                  0
Day                    0
Hour                   0
weather_main           0
weather_description    0
traffic_volume         0
dtype: int64

Summary statistics:
            holiday          temp       rain_1h       snow_1h          Year  \
count  48204.000000  48204.000000  48204.000000  48204.000000  48204.000000
mean       6.997780    281.258909      0.130851      0.000222   2015.512426
std        0.139999     12.713613      1.010260      0.008168      1.893211
min        0.000000    243.390000      0.000000      0.000000   2012.000000
25%        7.000000    272.160000      0.000000      0.000000   2014.000000
50%        7.000000    282.450000      0.000000      0.000000   2016.000000
75%        7.000000    291.806000      0.000000      0.000000   2017.000000
max       11.000000    310.070000     55.630000      0.510000   2018.000000

              Month           Day          Hour  weather_main  \
count  48204.000000  48204.000000  48204.000000  48204.000000
mean       6.506037     15.737636     11.398162      2.578375
std        3.400221      8.722938      6.940238      2.784224
min        1.000000      1.000000      0.000000      0.000000
25%        4.000000      8.000000      5.000000      0.000000
50%        7.000000     16.000000     11.000000      1.000000
75%        9.000000     23.000000     17.000000      5.000000
```



The regression models were successfully trained and tested on the traffic volume dataset. Key results are summarized below:

Model Performance Summary

Model	**MAE**	**RMSE**	**R^2 Score**
Linear Regression	1630.13	1839.22	0.14
Random Forest	514.15	803.86	0.84
XGBoost	531.43	798.82	0.84

- **Best Model:** Both **Random Forest** and **XGBoost** achieved the highest R^2 Score (0.84), indicating excellent fit.
- **Linear Regression** significantly underperformed due to inability to capture non-linear patterns.

Top Feature Importances (Random Forest)

From the feature importance plot:

- The most impactful variable is clearly **Hour**, contributing over 80% to the model's decision-making.
- Other relevant features include:
    - temp
    - Day
    - Month
    - Year
    - weather_description, weather_main
    - rain_1h, snow_1h, and holiday — with relatively minor influence

This emphasizes that **time of day is the strongest predictor** of traffic volume, followed by weather and calendar-based attributes.

Future Enhancements

- Try additional models like **LightGBM** or **CatBoost** for better efficiency and potentially higher accuracy.
- Use **K-Fold Cross-Validation** to avoid overfitting and validate model performance more robustly.
- Perform more **feature engineering**, such as:
  - Extracting is_weekend, is_peak_hour, season, or holiday proximity from the datetime.
  - Creating polynomial or interaction terms if necessary.
- Tune hyperparameters using **GridSearchCV** or **Optuna** for fine optimization.
- Deploy the final model as a **web application** or **REST API** for real-time traffic volume prediction.

References/Credits

- Andrew Ng’s “AI For Everyone”
- scikit-learn official documentation
- Kaggle dataset: Metro Interstate Traffic Volume
- Tutorials and community contributions from Medium, Towards Data Science, and Stack Overflow