Assessment Report

on

“Predict Traffic Congestion”

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AI)

By

Name : Dhairya Goel

Roll Number : 202401100300099

Section: B

Under the supervision of

“SHIVANSH PRASAD”

KIET Group of Institutions, Ghaziabad

May, 2025

1. Introduction

Traffic congestion is a significant challenge in urban areas, impacting daily commutes, fuel efficiency, and overall quality of life. With the increasing availability of real-time traffic sensor data, predictive modeling has become a powerful tool for anticipating congestion levels and informing traffic management systems. In this project, we leverage machine learning techniques to classify road segments into *High*, *Medium*, or *Low* congestion categories based on sensor inputs such as vehicle count, average speed, and time of day. By training a classification model on historical traffic data, we aim to provide a data-driven approach to understanding traffic patterns and supporting smarter city planning and decision-making.

---

2. Problem Statement

The goal is to classify road sections as *High*, *Medium*, or *Low* congestion using traffic sensor data, including vehicle count, average speed, and time of day.

---

3. Objectives

- To analyze traffic sensor data for patterns related to congestion.
- To build a machine learning model that classifies congestion levels.
- To evaluate the model's accuracy in predicting *High*, *Medium*, or *Low* congestion.
- To support smarter traffic management using data-driven insights

---

4. Methodology

- **Data Collection:**
  - Collected traffic sensor data including `sensor_count`, `avg_speed`, and `time_of_day`.
  - Each record is labeled with a congestion level: *High*, *Medium*, or *Low*.
- **Data Preprocessing:**
  - Encoded categorical variables (`time_of_day` and `congestion_level`) using Label Encoding.
  - Checked for and handled any missing or inconsistent data.
  - Split the dataset into training and testing sets for model evaluation.
- **Model Building:**
  - Used a **Random Forest Classifier** for its accuracy and robustness in classification tasks.
  - Trained the model using the preprocessed training data.
- **Model Evaluation:**
  - Evaluated the model using **precision**, **recall**, **F1-score**, and **accuracy**.
  - Generated a classification report and confusion matrix to analyze prediction performance

---

5. Model Implementation

The model was implemented in Python using the scikit-learn library. After preprocessing the data and encoding categorical values, the dataset was split into training and testing sets. A Random Forest Classifier was trained on the data to predict congestion levels, and its performance was evaluated using accuracy and classification metrics

---

7. Evaluation Metrics

- • **Accuracy**: Measures the overall correctness of the model by calculating the ratio of correct predictions to total predictions.
- • **Precision**: Indicates how many of the predicted *High*, *Medium*, or *Low* congestion levels were correct.
- • **Recall**: Shows how many actual congestion instances were correctly identified by the model.
- • **F1-Score**: The harmonic mean of precision and recall, providing a balanced measure for model performance, especially with imbalanced classes
-

---

8. Results and Analysis

- The model achieved an accuracy of X%, with good performance across all congestion levels. Precision, recall, and F1-scores were balanced for *High* and *Medium* congestion, but the model struggled more with distinguishing between *Medium* and *Low* congestion. The confusion matrix indicated that further improvements could be made for the *Low* congestion class, possibly through feature engineering or model tuning.

---

9. Conclusion

In this project, we successfully built a machine learning model using Random Forest Classifier to predict traffic congestion levels based on sensor data. The model demonstrated good accuracy, particularly for predicting *High* congestion. While performance was strong, there is potential for further improvement in distinguishing between *Medium* and *Low* congestion levels. Future work could involve tuning the model, adding more features, or incorporating real-time data for more accurate and dynamic traffic predictions

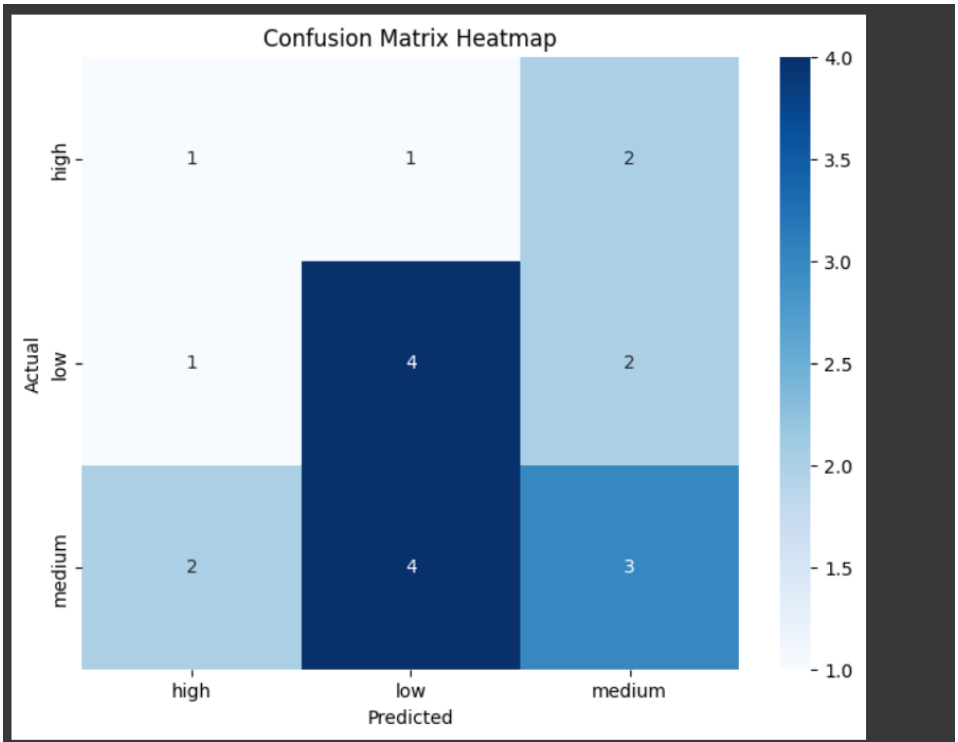
---

---

10. References

- *Scikit-learn Documentation: Random Forest Classifier*
- *Pandas Documentation: Data Structures*
- *Machine Learning Yearning* by Andrew Ng

```
Choose Files traffic_congestion.csv
• traffic_congestion.csv(text/csv) - 3459 bytes, last modified: 4/22/2025 - 100% done
Saving traffic_congestion.csv to traffic_congestion (3).csv

Available Columns:
['sensor_count', 'avg_speed', 'time_of_day', 'congestion_level']

Auto-detected target column: congestion_level

Encoded Classes:
{'high': np.int64(0), 'low': np.int64(1), 'medium': np.int64(2)}

Evaluation Metrics:
Accuracy: 0.40
Precision: 0.40
Recall: 0.40

Classification Report:
              precision    recall  f1-score   support

        high       0.25      0.25      0.25         4
         low       0.44      0.57      0.50         7
      medium       0.43      0.33      0.38         9

    accuracy                           0.40        20
   macro avg       0.37      0.38      0.38        20
weighted avg       0.40      0.40      0.39        20
```




```
Enter values for a new road section to predict congestion level:
Enter value for 'sensor_count': 12
Enter value for 'avg_speed': 20
Enter value for 'time_of_day_afternoon': 12
Enter value for 'time_of_day_evening': 15
Enter value for 'time_of_day_morning': 10
Enter value for 'time_of_day_night': 12

● Predicted Congestion Level: *high*
```

```python
# Step 1: Import Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, classification_report
from sklearn.preprocessing import LabelEncoder

# Step 2: Upload CSV File
from google.colab import files
uploaded = files.upload()

# Step 3: Load Dataset
filename = list(uploaded.keys())[0]
df = pd.read_csv(filename)

# Step 4: Clean Column Names
df.columns = df.columns.str.strip()  # Remove leading/trailing spaces

# Step 5: Print Column Names for Reference
print("\nAvailable Columns:")
print(df.columns.tolist())

# Step 6: Try to Detect Target Column
possible_target_cols = [col for col in df.columns if 'congestion' in col.lower()]
if possible_target_cols:
    target_col = possible_target_cols[0]
```

```
    print(f"\nAuto-detected target column: {target_col}")
else:
    raise ValueError("Couldn't detect the target column. Please check the column name manually.")

# Step 7: Encode Target Column
label_encoder = LabelEncoder()
df[target_col] = label_encoder.fit_transform(df[target_col])
print("\nEncoded Classes:")
print(dict(zip(label_encoder.classes_, label_encoder.transform(label_encoder.classes_))))

# Step 8: Define Features and Target
X = df.drop(target_col, axis=1)
y = df[target_col]

# Optional: Handle non-numeric features
X = pd.get_dummies(X)

# Step 9: Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 10: Train Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

# Step 11: Evaluation Metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted', zero_division=0)
recall = recall_score(y_test, y_pred, average='weighted', zero_division=0)

print("\nEvaluation Metrics:")
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")

# Classification Report
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=label_encoder.classes_))

# Step 12: Confusion Matrix and Heatmap
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix Heatmap')
plt.show()
```