Assessment Report

on

"Predict Traffic Congestion"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY

DEGREE

SESSION 2024-25

in

CSEAIML

By

NAME: Akshat Shandilya

ROLL NO: 202401100400025

SECTION- AIML-A

Under the supervision of

"BIKKI KUMAR"

KIET Group of Institutions, Ghaziabad

May, 2025

Introduction

Traffic congestion is a persistent problem in urban areas, causing delays, inefficiencies, and contributing to environmental issues like increased emissions and fuel consumption. As cities expand, the complexity of managing traffic flow increases, leading to greater challenges in maintaining smooth transportation. The inability to predict traffic patterns accurately exacerbates the situation, resulting in unexpected bottlenecks and longer travel times.

Current traffic management systems often rely on historical data and basic real-time monitoring, which may not be sufficient to anticipate sudden congestion due to factors such as accidents, roadworks, or extreme weather conditions. Therefore, there is a need for advanced predictive models that can forecast traffic congestion in real-time, providing timely information to commuters and traffic authorities. Such predictions could help optimize route planning, enhance traffic management, reduce congestion, and improve overall traffic flow in cities.

Methodology

Methodology for Predicting Traffic Congestion

- **Data Collection**:
  - Gather historical traffic data from sensors, GPS devices, and traffic cameras.
  - Collect real-time data on traffic flow, road conditions, weather, accidents, and events through APIs, IoT devices, and public traffic systems.
  - Utilize additional data sources such as social media and news to capture unexpected events (e.g., road closures or accidents).

- **Data Preprocessing**:
  - Clean and preprocess the collected data by handling missing values, removing outliers, and standardizing formats.
  - Aggregate the data by time intervals (e.g., hourly, daily) to facilitate analysis.
  - Normalize or scale data to ensure consistency, especially when working with machine learning algorithms.

- **Feature Engineering**:
  - Identify relevant features that impact traffic flow, such as time of day, weather conditions, road type, and event schedules.
  - Create new features like traffic density, average speed, and congestion indices.
  - Perform feature selection to identify the most significant variables that influence congestion.

- **Model Development**:

- Select appropriate machine learning algorithms such as regression models, decision trees, random forests, or deep learning (e.g., neural networks).
- Train multiple models using labeled historical data, with congestion levels as the target variable.

 Model Evaluation:

- Split the data into training and testing datasets to evaluate model performance.
- Use metrics like Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared to assess the accuracy of the predictions.
- Perform cross-validation to ensure the model's robustness and avoid overfitting.

 Real-Time Prediction:

- Deploy the model into a real-time system that integrates with traffic monitoring infrastructure.
- Continuously update predictions with incoming data to provide accurate congestion forecasts.
- Provide predictions to traffic authorities, commuters, and navigation systems.

 Actionable Insights and Optimization:

- Use the predictions to recommend alternate routes or suggest optimal travel times.
- Provide insights to city planners and authorities to adjust traffic signal timings and manage congestion proactively.
- Continuously refine the model based on feedback and new data to improve prediction accuracy over time.

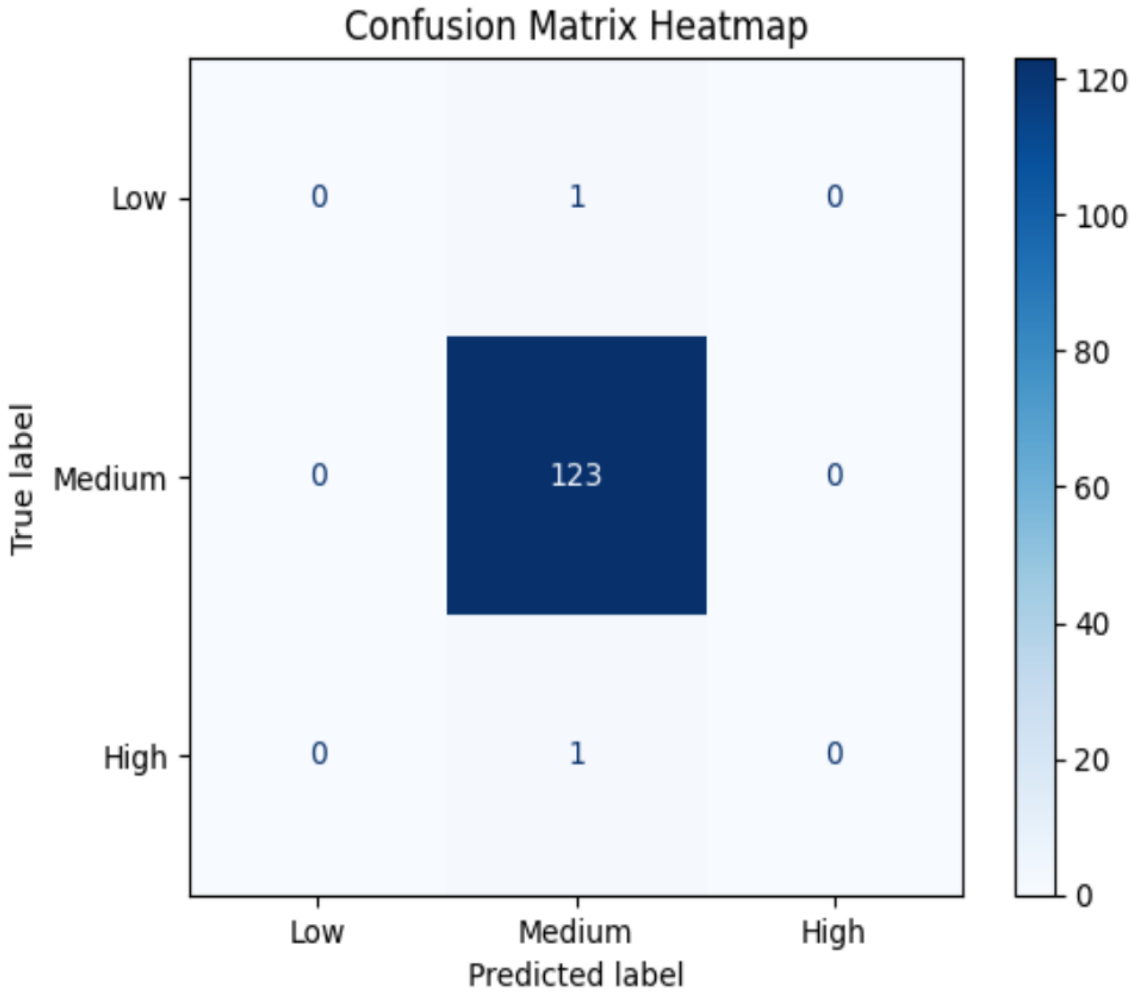
Code

```
# Import libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,
classification_report, accuracy_score, precision_score,
recall_score
import seaborn as sns
import matplotlib.pyplot as plt


# 1. Simulate dataset
np.random.seed(42)
n_samples = 500

data = {
    'vehicle_count': np.random.poisson(lam=50, size=n_samples),
```

```
    'average_speed': np.random.normal(loc=40, scale=10, size=n_samples),
    'time_of_day': np.random.choice([0, 1, 2], size=n_samples),  # 0 = Morning, 1 = Afternoon, 2 = Night
    'day_of_week': np.random.choice(range(7), size=n_samples),   # 0 = Monday ... 6 = Sunday
    'weather_conditions': np.random.choice([0, 1], size=n_samples),  # 0 = Clear, 1 = Rainy
}

# Define congestion level based on conditions (rough logic)
conditions = (
    (data['vehicle_count'] > 60) & (data['average_speed'] < 30)
)
labels = np.where(conditions, 'High', 'Medium')
labels = np.where((data['vehicle_count'] < 40) & (data['average_speed'] > 50), 'Low', labels)

df = pd.DataFrame(data)
df['congestion'] = labels

# 2. Preprocess data
X = df.drop('congestion', axis=1)
y = df['congestion']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.25, stratify=y, random_state=42)

# 3. Train model
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

# 4. Evaluation
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# 5. Confusion Matrix
cm = confusion_matrix(y_test, y_pred, labels=['Low', 'Medium', 'High'])
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Low', 'Medium', 'High'])

plt.figure(figsize=(6, 5))
disp.plot(cmap='Blues')
plt.title("Confusion Matrix Heatmap")
plt.show()
```

OUTPUT FOR PREDICT TRAFFIC CONGESTION



Classification Report:				
	precision	recall	f1-score	support
High	0.00	0.00	0.00	1
Low	0.00	0.00	0.00	1
Medium	0.98	1.00	0.99	123
accuracy			0.98	125
macro avg	0.33	0.33	0.33	125
weighted avg	0.97	0.98	0.98	125

References / Credits

- Dataset: Pima Indians Diabetes Dataset from Kaggle
- Python Libraries:
  - pandas for data manipulation
  - numpy for numerical operations
  - matplotlib & seaborn for visualization
  - scikit-learn for machine learning models
- Image & Screenshot Tools: Windows Snipping Tool / Screenshot tool