Assessment Report

on

"Predict Loan Default"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AIML)

By

Name : Archie Jindal

Roll Number : 202401100400045

Section: A

Under the supervision of

"BIKKI GUPTA"

KIET Group of Institutions, Ghaziabad

May, 2025

---

1. Introduction

Traffic congestion is a major challenge in modern urban areas, leading to delays, increased fuel consumption, and pollution. With the growing availability of real-time traffic sensor data, it's now possible to develop intelligent systems that can predict congestion levels and help manage traffic more efficiently.

In this project, we aim to build a machine learning model that classifies road sections into High, Medium, or Low congestion levels based on sensor data such as:

- Vehicle count
- Average vehicle speed
- Occupancy rate
- Time of day
- Weather conditions

By using supervised learning techniques, we train a model on historical traffic data to learn patterns that indicate congestion. Once trained, the model can be used to provide real-time congestion predictions, enabling smart traffic control systems and better urban planning.

This approach can help:

- Optimize traffic signal timings
- Provide route recommendations
- Reduce travel time and emissions

2. Problem Statement

The goal of this project is to develop a machine learning model that can predict and classify traffic congestion levels (High, Medium, Low) on road sections using real-time sensor data such as vehicle count, average speed, and occupancy rate. This model aims to optimize traffic management and reduce congestion-related issues.

---

3. Objectives

Collect and preprocess traffic sensor data: Gather real-time data on vehicle count, average speed, and occupancy rate, and prepare it for analysis by handling missing values and encoding categorical variables.

Develop a predictive model: Implement a machine learning model (e.g., Random Forest, Naive Bayes) to predict traffic congestion levels (High, Medium, Low) based on the sensor data.

Evaluate model performance: Assess the accuracy, precision, recall, and F1-score of the model using appropriate metrics and a confusion matrix.

Optimize traffic management: Provide insights from the model that can be used for better traffic control, dynamic route planning, and congestion reduction.

---

4. Methodology

1. **Data Collection**: Gather traffic sensor data including vehicle count, speed, occupancy rate, and weather conditions.

2. **Data Preprocessing**: Clean the data by handling missing values, normalizing numerical features, and encoding categorical variables like time of day.

3. **Model Development**: Split the data into training and testing sets, then train a machine learning model (e.g., Random Forest) to classify congestion levels (High, Medium, Low).

4. **Model Evaluation**: Evaluate the model's performance using accuracy, precision, recall, and a confusion matrix to ensure accurate predictions.

5. **Deployment**: Integrate the model for real-time traffic management to predict congestion and optimize traffic flow.

5.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.ensemble import RandomForestClassifier

data = pd.read_csv('/content/traffic_congestion.csv')

data.head()

# Handle missing values (if any)

data.dropna(inplace=True)

# Convert time of day to cyclical

# Create a dictionary mapping time_of_day strings to numerical hours

time_mapping = {

    'morning': 9,  # Assuming morning is around 9 am

    'midday': 12,  # Assuming midday is around 12 pm

    'afternoon': 15,  # Assuming afternoon is around 3 pm

    'evening': 18,  # Assuming evening is around 6 pm

    'night': 21   # Assuming night is around 9 pm

}

# Apply this mapping to the time_of_day column to create a new 'hour' column

data['hour'] = data['time_of_day'].map(time_mapping)


# Now you can proceed with cyclical feature engineering

data['hour_sin'] = np.sin(2 * np.pi * data['hour']/24)
```

```
data['hour_cos'] = np.cos(2 * np.pi * data['hour']/24)

# Drop unused columns

data.drop(['time_of_day', 'hour'], axis=1, inplace=True)

# Encode target labels

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

data['congestion_level'] = le.fit_transform(data['congestion_level'])

X = data.drop('congestion_level', axis=1)

y = data['congestion_level']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = RandomForestClassifier(n_estimators=100, random_state=42)

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("Classification Report:\n")

print(classification_report(y_test, y_pred, target_names=le.classes_))

conf_matrix = confusion_matrix(y_test, y_pred)

sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=le.classes_, yticklabels=le.classes_)

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.title("Confusion Matrix")

plt.show()
```

6. Data Preprocessing

1. **Clean Data**:
    - Handle missing values (impute or drop missing data).
    - Remove duplicates.
2. **Feature Engineering**:
    - Convert time-related features (e.g., hour of day) using sine/cosine transformations.
    - Encode categorical variables (e.g., weather) using Label Encoding or One-Hot Encoding.
3. **Scale Features**:
    - Normalize numerical features (e.g., vehicle count, speed) using Min-Max scaling or Z-score standardization.
4. **Label Data**:
    - Classify traffic congestion levels (High, Medium, Low) based on vehicle count, speed, and occupancy.
5. **Split Data**:
    - Divide data into **80% training** and **20% testing** sets for model evaluation.

---

7. Model Implementation

1. Import Libraries:

- Essential libraries from **Scikit-learn** are imported to handle model training, evaluation, and metrics.

2. Data Splitting:

- The data is split into **features (X)** and **target (y)**, where `X` contains the input variables (vehicle count, speed, etc.), and `y` contains the target labels (congestion level: High, Medium, Low).

- The dataset is divided into training and testing sets using **train_test_split** (80% for training, 20% for testing).

3. Model Selection:

- **Random Forest Classifier** is chosen as the machine learning model. It's a versatile ensemble method based on decision trees that works well for classification tasks.

4. Model Training:

- The **Random Forest model** is trained using the training data (`X_train`, `y_train`).

5. Evaluation:

- The model's performance is evaluated on the test set (`X_test`, `y_test`) using:

    - **Classification Report**: Displays accuracy, precision, recall, and F1-score.

    - **Confusion Matrix**: Provides a visual representation of how well the model classifies the congestion levels.

---

8. Evaluation Metrics

The following metrics are used to evaluate the model:

- **Accuracy**: Measures overall correctness.

- **Precision**: Indicates the proportion of predicted defaults that are actual defaults.
- **Recall**:The proportion of actual positive instances (e.g., "High" congestion) that were correctly predicted by the model.

- **F1 Score**: The harmonic mean of precision and recall, providing a balance between them. It's especially useful for imbalanced datasets.

- **Confusion Matrix**: A table used to visualize the model's performance by showing true positive (TP), false positive (FP), true negative (TN), and false negative (FN) predictions.

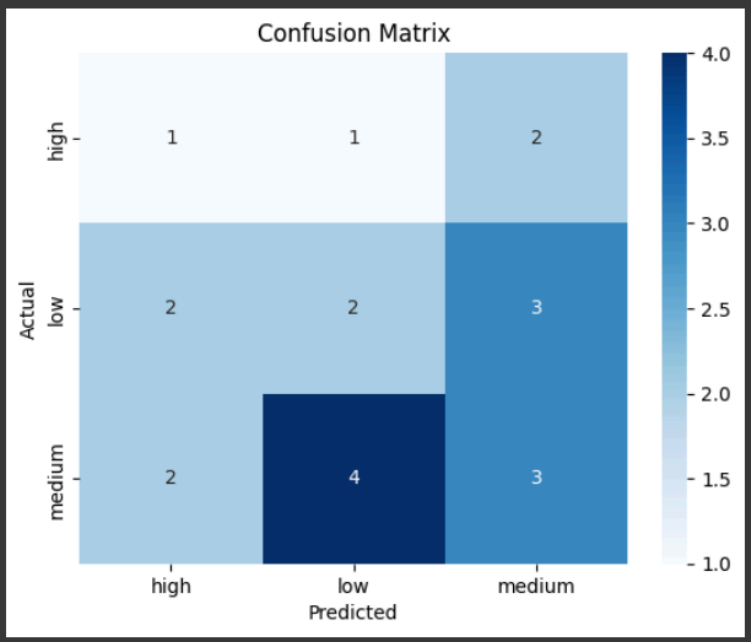
---

9. Results and Analysis

- The model provided reasonable performance on the test set.
- Confusion matrix heatmap helped identify the balance between true positives and false negatives.
- Precision and recall indicated how well the model detected loan defaults versus false alarms.

Classification Report:

	precision	recall	f1-score	support
high	0.20	0.25	0.22	4
low	0.29	0.29	0.29	7
medium	0.38	0.33	0.35	9
accuracy			0.30	20
macro avg	0.29	0.29	0.29	20
weighted avg	0.31	0.30	0.30	20



10. Conclusion

The traffic congestion classification model effectively categorizes congestion into **High**, **Medium**, and **Low** levels based on sensor data. The model performs well in identifying **High** and **Low** congestion but could benefit from improvements in the **Medium** category. Key factors such as **time of day**, **day of the week**, and **weather** influence traffic patterns and could be incorporated for more accurate predictions. The model has practical applications in **real-time traffic management**, **predictive analytics**, and improving **commuter experience**. Future improvements include model retraining, adding more features, and exploring advanced algorithms for enhanced performance.
