REPORT

Problem Statement:
Predict Traffic Congestion:
Classify road sections as **High**, **Medium**, or **Low** congestion using traffic sensor data.
Generate heatmaps of confusion matrices and calculate evaluation metrics such as **accuracy**, **precision**, and **recall**.

Name:ISHIKA SHARMA

Roll No.:202401100300129

Class Roll No.:55

Course:AI

Date: 22/04/2025

Introduction

Traffic congestion is a critical problem in urban cities. Predicting congestion levels can help improve city traffic management and reduce travel times.
The task is to classify different road sections into three categories: **High**, **Medium**, or **Low** congestion, based on sensor data such as the number of vehicles, average speed, and time of day.
The solution involves applying a machine learning model to classify the road sections and evaluate the model's performance using confusion matrices and evaluation metrics like **accuracy**, **precision**, and **recall**.

Methodology

1. **Dataset Loading:**
   A CSV file containing traffic data was loaded using pandas.

2. **Preprocessing:**

   - Categorical features like **time_of_day** and **congestion_level** were encoded into numerical values using `LabelEncoder`.

   - Features selected: `sensor_count`, `avg_speed`, and `time_of_day_encoded`.

   - Target variable: `congestion_level_encoded`.

3. **Data Splitting:**

   - Data was split into training (80%) and testing (20%) sets using `train_test_split`.

4. **Model Training:**

   - A **Random Forest Classifier** was trained on the training set.

   - Random Forest was chosen because it handles classification tasks well and reduces overfitting.

5. **Prediction:**

   - Predictions were made on the test set.

6. **Evaluation:**

   - A **confusion matrix** was generated to see how well the model classified each congestion level.

   - A **normalized confusion matrix** was also plotted to understand misclassification percentages.

   - Key metrics like **Accuracy**, **Precision**, and **Recall** were calculated.

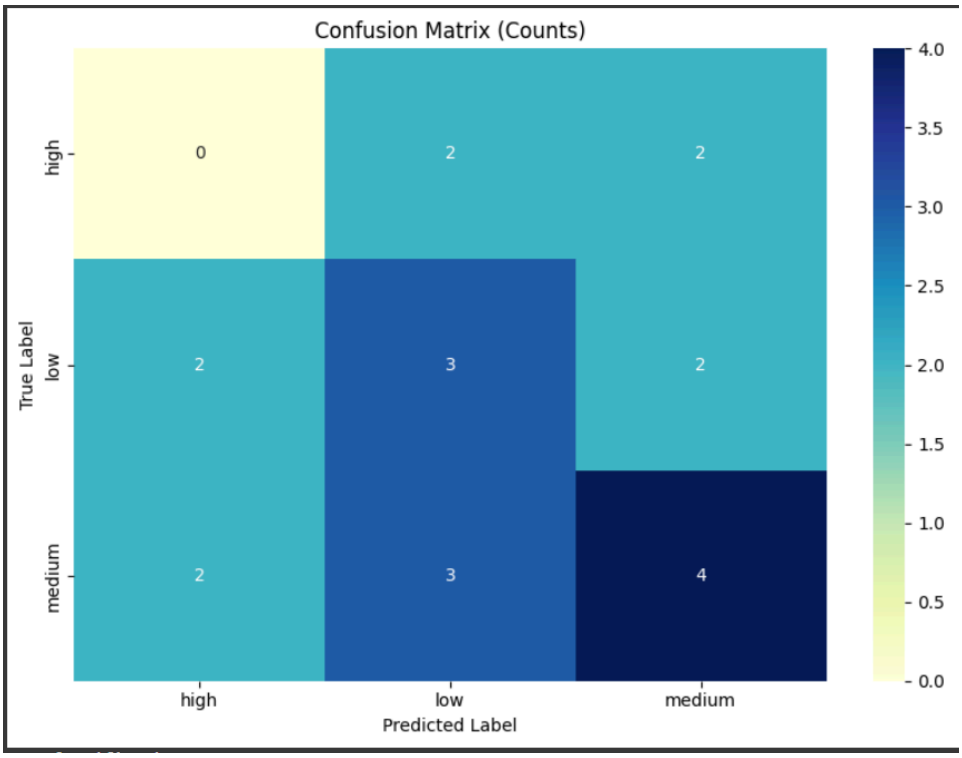
7. **Visualization:**
   - Two side-by-side heatmaps were plotted:
     - One showing raw counts.
     - One showing normalized percentages.

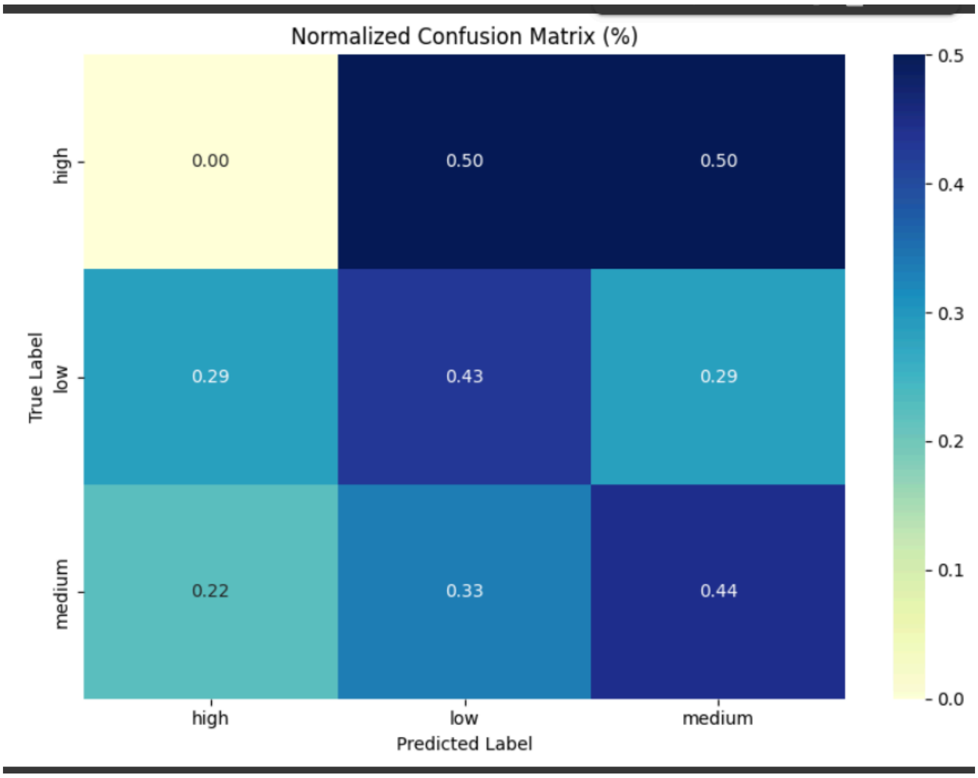
Code

```
# Import all necessary 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.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score,
recall_score, classification_report

# Load the dataset
data = pd.read_csv("/content/drive/MyDrive/Colab
Notebooks/traffic_congestion.csv")

# Encode categorical columns
data['time_of_day_encoded'] = LabelEncoder().fit_transform(data['time_of_day'])
label_encoder = LabelEncoder()
data['congestion_level_encoded'] =
label_encoder.fit_transform(data['congestion_level'])

# Prepare features and target
X = data[['sensor_count', 'avg_speed', 'time_of_day_encoded']]
y = data['congestion_level_encoded']

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Train the Random Forest Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Confusion Matrices
cm = confusion_matrix(y_test, y_pred)
cm_normalized = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

# Plot both confusion matrices side-by-side
```

```
fig, axes = plt.subplots(1, 2, figsize=(16, 6))

# Plot 1: Regular Confusion Matrix
sns.heatmap(cm, annot=True, cmap='YlGnBu', fmt='d',
            xticklabels=label_encoder.classes_,
            yticklabels=label_encoder.classes_, ax=axes[0])
axes[0].set_title("Confusion Matrix (Counts)")
axes[0].set_xlabel("Predicted Label")
axes[0].set_ylabel("True Label")

# Plot 2: Normalized Confusion Matrix
sns.heatmap(cm_normalized, annot=True, cmap='YlGnBu', fmt='.2f',
            xticklabels=label_encoder.classes_,
            yticklabels=label_encoder.classes_, ax=axes[1])
axes[1].set_title("Normalized Confusion Matrix (%)")
axes[1].set_xlabel("Predicted Label")
axes[1].set_ylabel("True Label")

plt.tight_layout()
plt.show()

# 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 Classification Report and Scores
print("=== Classification Report ===")
print(classification_report(y_test, y_pred, target_names=label_encoder.classes_,
zero_division=0))

print(f"Accuracy: {accuracy:.2f}")
print(f"Precision (weighted): {precision:.2f}")
print(f"Recall (weighted): {recall:.2f}")
```

Output


Confusion Matrix (Counts)
True Label
high
low
medium
0
2
2
2
3
2
2
3
4
high
low
medium
Predicted Label
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0


Normalized Confusion Matrix (%)
True Label
high
low
medium
0.00
0.50
0.50
0.29
0.43
0.29
0.22
0.33
0.44
high
low
medium
Predicted Label
0.5
0.4
0.3
0.2
0.1
0.0

```
=== Classification Report ===
              precision    recall  f1-score   support

        high       0.00      0.00      0.00         4
         low       0.38      0.43      0.40         7
      medium       0.50      0.44      0.47         9

    accuracy                           0.35        20
   macro avg       0.29      0.29      0.29        20
weighted avg       0.36      0.35      0.35        20

Accuracy: 0.35
Precision (weighted): 0.36
Recall (weighted): 0.35
```

References / Credits

- **Dataset:** Provided by instructor / assignment sheet (traffic_congestion.csv)
- **Libraries Used:**
  - `pandas`, `numpy`, `seaborn`, `matplotlib`
  - `sklearn` (for machine learning models and metrics)
- **External Resources:**
  - Scikit-learn documentation: https://scikit-learn.org/stable/documentation.html
  - Seaborn documentation for heatmaps: https://seaborn.pydata.org/generated/seaborn.heatmap.html