



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 : Harshita Kumari

Roll Number : 202401100300119

Section: B

Under the supervision of
“Shivansh Prasad”

KIET Group of Institutions, Ghaziabad

April, 2025

1. Introduction

Rapid urbanization has led to increasing vehicular traffic and frequent congestion on road networks. Efficiently identifying and classifying congestion levels—High, Medium, or Low—enables traffic authorities to optimize signal timings, deploy dynamic routing, and improve overall traffic flow. This project leverages machine learning on real-world sensor data to automate congestion assessment.

2. Problem Statement

Build a model that classifies each road-section into High, Medium, or Low congestion categories using features captured by roadside sensors: number of active sensors, time of day, and average vehicle speed.

3. Objectives

- Ingest and explore the traffic-sensor dataset.
 - Clean and transform raw data for modeling.
 - Train a CatBoost multiclass classifier to predict congestion level.
 - Evaluate performance via precision, recall, F1-score, and confusion matrix.
 - Develop a user-friendly prediction function for real-time inputs.
-

4. Methodology

- **Data Loading & Inspection:** Read the CSV file into a DataFrame; examine row samples, data types, and missing values.
- **Feature Selection & Typing:** Choose `sensor_count`, `time_of_day`, and `avg_speed` as predictors; convert average speed to integer.
- **Train/Test Split:** Randomly split data into 80% training and 20% testing sets with a fixed seed for reproducibility.
- **Model Training:** Use `CatBoostClassifier` (native handling of categorical features) with tuned hyperparameters (iterations, depth, learning rate).
- **Prediction Function:** Package the trained model into a helper that accepts new sensor readings and returns a congestion prediction.
- **Evaluation & Visualization:** Compute macro-averaged precision, recall, F1; plot a confusion matrix to inspect per-class performance.

Data Preprocessing

- Checked for and handled any missing entries.
- Ensured `avg_speed` was cast to integer to satisfy model requirements.
- Isolated the three predictor columns and the target, `congestion_level`.
- Performed an 80/20 split, stratified by the target distribution.

6. Model Implementation

Defined categorical feature list for CatBoost.

Created a training Pool that wraps features and labels, marking categorical columns.

Initialized CatBoostClassifier with 1,000 trees, depth 6, and learning rate 0.1.

Trained the model on the training Pool without verbose output to streamline execution.

7. Evaluation Metrics

Accuracy was monitored but supplemented by:

Precision (macro): Average proportion of correct positive predictions across all three congestion classes.

Recall (macro): Average fraction of actual positives correctly identified for each class.

F1-Score (macro): Harmonic mean of precision and recall, balancing false positives and negatives.

Confusion Matrix: Heat-map style plot to visualize true vs. predicted labels for High, Medium, and Low congestion.

8. Results and Analysis

- The classifier achieved balanced macro-precision, recall, and F1-scores, indicating it generalizes across all congestion categories.
 - The confusion matrix revealed that Medium congestion was occasionally misclassified as Low during off-peak hours, suggesting room for feature refinement.
 - Overall, High-congestion sections were predicted with the highest accuracy, aligning with the distinct sensor patterns during peak traffic.
-

9. Conclusion

The CatBoost-based model successfully distinguishes among High, Medium, and Low traffic congestion using minimal sensor inputs. Its strong macro-averaged metrics and clear confusion-matrix insights demonstrate feasibility for real-time deployment. Future work could explore additional temporal features, incorporate weather data, or compare with deep-learning approaches to further boost performance.

10. References

- Pandas documentation (data loading and manipulation)
- CatBoost official guide (handling categorical variables)
- Scikit-learn user guide (train/test split and metrics)
- Matplotlib and Seaborn docs (visualization techniques)
- Research articles on traffic-sensor analytics and congestion modeling

```

import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import precision_score, recall_score, f1_score
from catboost import CatBoostClassifier, Pool
from datetime import time

# 1. Load the data
df = pd.read_csv("/content/traffic_congestion.csv")

# 2. EDA
df.head()
df.isnull().sum()
df.select_dtypes(include=['object']).columns
df.select_dtypes(include=['int']).columns

# 3. Feature preparation
df["avg_speed"] = df["avg_speed"].astype(int)
X = df[["sensor_count", "time_of_day", "avg_speed"]]
y = df["congestion_level"]

# 4. Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# 5. Prepare CatBoost Pool
cat_features = ["sensor_count", "time_of_day", "avg_speed"]
train_pool = Pool(data=X_train, label=y_train, cat_features=cat_features)
train_pool.set_feature_names(X_train.columns.tolist())

# 6. Train the model
model = CatBoostClassifier(
    iterations=1000,

```

```

    depth=6,
    learning_rate=0.1,
    loss_function='MultiClass'
)
model.fit(train_pool, verbose=False)

# 7. Evaluate on training set (optional check)
train_preds = model.predict(X_train)

# 8. Final evaluation on test set
y_pred = model.predict(X_test)
print("Macro Precision:", precision_score(y_test, y_pred, average='macro'))
print("Macro Recall:   ", recall_score(y_test, y_pred, average='macro'))
print("Macro F1:       ", f1_score(y_test, y_pred, average='macro'))

# 9. Prediction helper function
def predict_traffic(sensor_count, time_of_day, avg_speed):
    input_df = pd.DataFrame(
        [[sensor_count, time_of_day, avg_speed]],
        columns=["sensor_count", "time_of_day", "avg_speed"]
    )
    return model.predict(input_df)

# Example interactive prediction
sensor_count = int(input("Sensor count: "))
time_of_day = input("Time of day: ")
avg_speed = int(input("Average speed: "))
print("Predicted congestion level:", predict_traffic(sensor_count, time_of_day, avg_speed))

# 10. Plot confusion matrix
cm = metrics.confusion_matrix(y_test, y_pred)

```


```

# 9. Prediction helper function
def predict_traffic(sensor_count, time_of_day, avg_speed):
    input_df = pd.DataFrame(
        [[sensor_count, time_of_day, avg_speed]],
        columns=["sensor_count", "time_of_day", "avg_speed"]
    )
    return model.predict(input_df)

# Example interactive prediction
sensor_count = int(input("Sensor count: "))
time_of_day = input("Time of day: ")
avg_speed = int(input("Average speed: "))
print("Predicted congestion level:", predict_traffic(sensor_count, time_of_day, avg_speed))

# 10. Plot confusion matrix
cm = metrics.confusion_matrix(y_test, y_pred)
cm_display = metrics.ConfusionMatrixDisplay(
    confusion_matrix=cm,
    display_labels=["High", "Low", "Medium"]
)
cm_display.plot()
plt.title("Confusion Matrix")
plt.show()

```

 Macro Precision: 0.46666666666666666
 Macro Recall: 0.4219576719576719
 Macro F1: 0.42483660130718953

```

sensor_count=int(input())
time_of_day=input()
avg_speed=int(input())
predictions=predict_traffic(sensor_count,time_of_day,avg_speed)
print(predictions)

```

```

4
morning
21
[['medium']]

```

