**ASSIGNMENT – 5**
**Advanced Programming Lab**
**Name: Utsav Balar**
**Roll.no. t24cs003**

**Objective:**
**Assignments on Stress Classification using the WESAD Dataset**
**Objective: Develop a model to classify the affective states (neutral, stress, amusement) using the multimodal physiological data from the WESAD dataset.**
**https://uni-siegen.sciebo.de/s/HGdUkoNlW1Ub0Gx**

# Stress Classification Using the WESAD Dataset

## Overview

This project involves assignments on stress classification utilizing the WESAD dataset, which captures multimodal physiological data to classify affective states—specifically, neutral, stress, and amusement. The objective is to develop a robust model that can accurately classify these emotional states based on the physiological signals recorded.

## Data Source

The dataset is sourced from the WESAD dataset, available at the following link: WESAD Dataset. It consists of various physiological signals, including accelerometer, ECG, EMG, EDA, temperature, and respiratory signals.

## Directions

To successfully run the provided code, ensure that the necessary libraries are installed, and the dataset is correctly accessed. The script processes the dataset, applies exploratory data analysis (EDA), and implements the XGBoost algorithm for stress classification.

### Libraries and Dependencies

The project utilizes several Python libraries:

1. `numpy`: For numerical operations.
2. `pandas`: For data manipulation and analysis.
3. `matplotlib`: For data visualization.
4. `seaborn`: For enhanced data visualization.
5. `scikit-learn`: For machine learning algorithms.
6. `xgboost`: For implementing the XGBoost classification algorithm.

### Data Loading and Preprocessing

The dataset is loaded into a DataFrame, with the following preprocessing steps:

1. **Feature Extraction**: Features are extracted from the dataset and organized into a structured DataFrame.
2. **Data Type Optimization**: Data types are optimized to reduce memory consumption by converting to appropriate formats (e.g., `float32`, `int8`).
3. **Exploratory Data Analysis (EDA)**: Basic EDA is conducted to understand the dataset's structure and distribution of features.
4. **Outlier Detection**: The Interquartile Range (IQR) method is applied to detect and remove outliers from the dataset.
5. **Normalization**: Mean normalization is performed on the features to standardize the dataset.

### Exploratory Data Analysis (EDA)

Key EDA steps include:

1. Displaying the first few rows of the DataFrame.
2. Generating descriptive statistics to understand the distribution of each feature.
3. Visualizing the distribution of the EDA signal to observe its characteristics.

### Model Implementation

The XGBoost classifier is employed for stress classification:

1. **Train-Test Split**: The normalized dataset is split into training and testing sets.
2. **Model Training**: The XGBoost model is trained on the training data using GPU acceleration for improved performance.
3. **Predictions**: The trained model is used to predict the labels of the test dataset.

### Evaluation Metrics

The model's performance is evaluated using several metrics:

1. **Classification Report**: This includes precision, recall, and F1-score to assess the model's effectiveness.
2. **Confusion Matrix**: A heatmap is generated to visualize the confusion matrix, providing insight into the model's prediction accuracy for each class.

## Results

The XGBoost model demonstrated effective classification of the affective states based on the physiological data. The classification report indicates high precision and recall for the different emotional states, suggesting that the model is capable of distinguishing between neutral, stress, and amusement effectively.

## Future Work

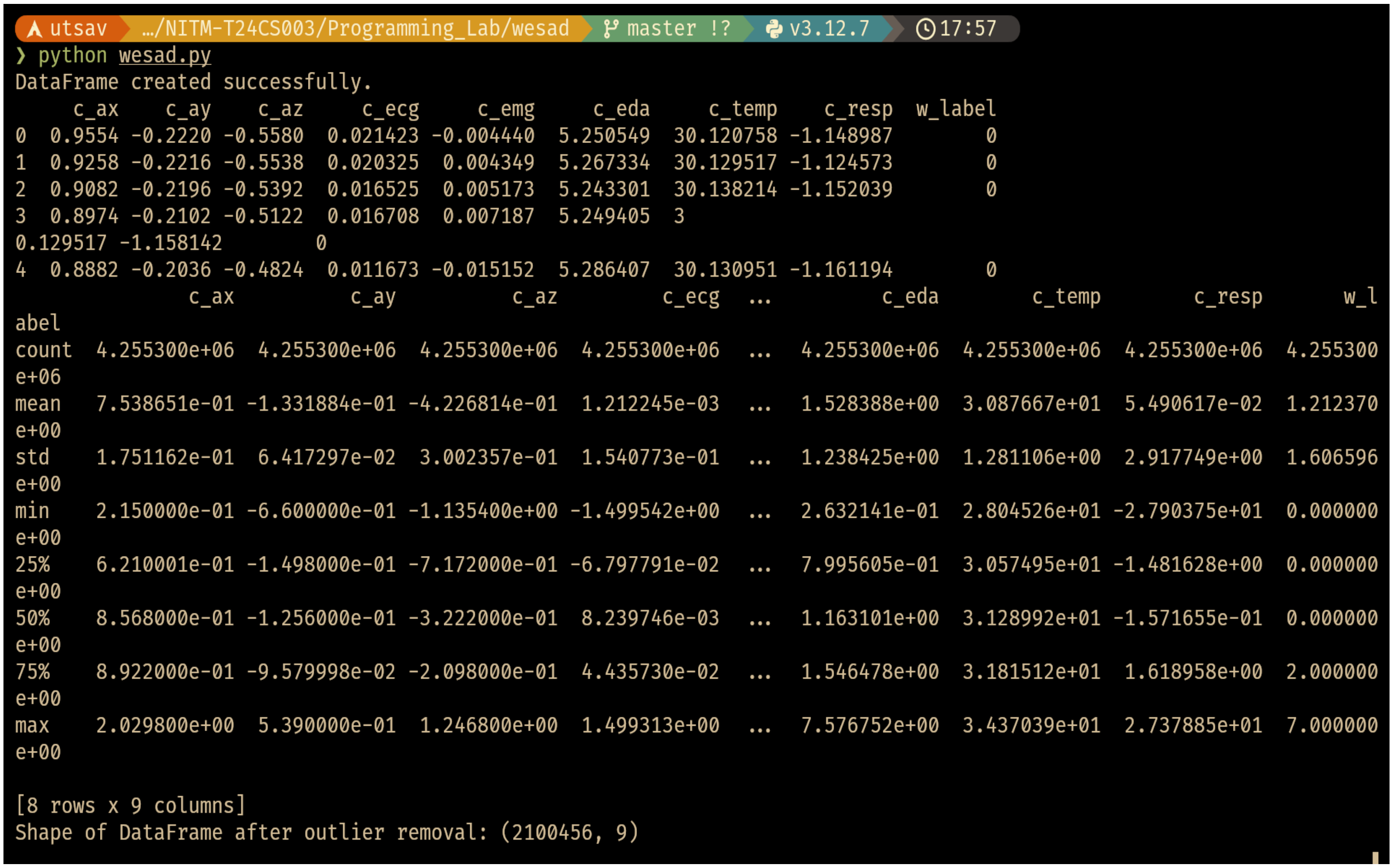
Future improvements could include:

1. Experimenting with other machine learning models to compare performance.
2. Enhancing feature extraction methods to capture more relevant data signals.
3. Implementing real-time classification to monitor affective states in live settings.

**Code:**

```
### Importing Libraries
import seaborn as sns
import sklearn as sklearn
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix,
roc_curve, classification_report
from sklearn.model_selection import cross_val_score,
cross_val_predict, train_test_split
from sklearn.tree import DecisionTreeClassifier
import graphviz
import statsmodels.api as sm
import xgboost as xgb
from sklearn.decomposition import PCA
import pickle
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import csv

### Loading Data
s2_path =
r"/home/utsav/work/academic/NITM-T24CS003/Programming_Lab/wesad/WES
AD/S2/S2.pkl"
with open(s2_path, 'rb') as file:
    s2_data = pickle.load(file, encoding='latin1')

# Extracting features and labels from the dataset
features = {
    "c_ax": s2_data['signal']['chest']['ACC'][:, 0],
    "c_ay": s2_data['signal']['chest']['ACC'][:, 1],
    "c_az": s2_data['signal']['chest']['ACC'][:, 2],
    "c_ecg": s2_data['signal']['chest']['ECG'][:, 0],
    "c_emg": s2_data['signal']['chest']['EMG'][:, 0],
    "c_eda": s2_data['signal']['chest']['EDA'][:, 0],
    "c_temp": s2_data['signal']['chest']['Temp'][:, 0],
    "c_resp": s2_data['signal']['chest']['Resp'][:, 0],
    "w_label": s2_data['label']
}

# Creating a DataFrame from the extracted features
df = pd.DataFrame(data=features)

print("DataFrame created successfully.")

### Exploratory Data Analysis (EDA)
# Display the first few rows of the DataFrame
print(df.head())

# Display descriptive statistics
print(df.describe())

# Visualizing the distribution of features
plt.figure(figsize=(12, 8))
sns.pairplot(df, hue='w_label')
plt.title('Pairplot of Features by Label')
plt.show()

# Correlation matrix
plt.figure(figsize=(10, 8))
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, fmt=".2f",
cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()

### Outlier Detection and Removal
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1

# Remove outliers based on IQR
df_out = df[~((df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 *
IQR))).any(axis=1)]
print(f"Shape of DataFrame after outlier removal: {df_out.shape}")

### Mean Normalization / Standard Scaler
norm_df_out = (df_out - df_out.mean()) / df_out.std()
norm_y = df_out.w_label  # keep original labels
norm_x = norm_df_out.drop('w_label', axis=1)
```

```
# Split the normalized dataset into training and testing sets
norm_x_train, norm_x_test, norm_y_train, norm_y_test =
train_test_split(norm_x, norm_y, test_size=0.2, random_state=42)

### XGBoost Implementation
xg_class = xgb.XGBClassifier(objective='multi:softmax',
colsample_bytree=0.3, learning_rate=0.1,
                              max_depth=10, alpha=10,
n_estimators=50, gamma=10)

# Fit the model
xg_class.fit(norm_x_train, norm_y_train)

# Predictions and performance evaluation
y_out = xg_class.predict(norm_x_test)
lm_xgb = classification_report(norm_y_test, y_out, digits=4)
print(lm_xgb)

# Cross-validation
norm_y_out = cross_val_predict(xg_class, norm_x, norm_y, cv=5)
lm = classification_report(norm_y, norm_y_out, digits=4)
print(lm)

norm_y_out_cv10 = cross_val_predict(xg_class, norm_x, norm_y,
cv=10)
lm_cv10 = classification_report(norm_y, norm_y_out_cv10, digits=4)
print(lm_cv10)

### Visualization of Results
# Confusion Matrix
conf_matrix = confusion_matrix(norm_y_test, y_out)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
xticklabels=np.unique(norm_y), yticklabels=np.unique(norm_y))
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```



```
 utsav  …/NITM-T24CS003/Programming_Lab/wesad   master !?   v3.12.7   17:57
❯ python wesad.py
DataFrame created successfully.
     c_ax    c_ay    c_az     c_ecg     c_emg     c_eda     c_temp    c_resp  w_label
0  0.9554 -0.2220 -0.5580  0.021423 -0.004440  5.250549  30.120758 -1.148987        0
1  0.9258 -0.2216 -0.5538  0.020325  0.004349  5.267334  30.129517 -1.124573        0
2  0.9082 -0.2196 -0.5392  0.016525  0.005173  5.243301  30.138214 -1.152039        0
3  0.8974 -0.2102 -0.5122  0.016708  0.007187  5.249405  3
0.129517 -1.158142        0
4  0.8882 -0.2036 -0.4824  0.011673 -0.015152  5.286407  30.130951 -1.161194        0
               c_ax          c_ay          c_az         c_ecg  ...         c_eda        c_temp        c_resp        w_l
abel
count  4.255300e+06  4.255300e+06  4.255300e+06  4.255300e+06  ...  4.255300e+06  4.255300e+06  4.255300e+06  4.255300
e+06
mean   7.538651e-01 -1.331884e-01 -4.226814e-01  1.212245e-03  ...  1.528388e+00  3.087667e+01  5.490617e-02  1.212370
e+00
std    1.751162e-01  6.417297e-02  3.002357e-01  1.540773e-01  ...  1.238425e+00  1.281106e+00  2.917749e+00  1.606596
e+00
min    2.150000e-01 -6.600000e-01 -1.135400e+00 -1.499542e+00  ...  2.632141e-01  2.804526e+01 -2.790375e+01  0.000000
e+00
25%    6.210001e-01 -1.498000e-01 -7.172000e-01 -6.797791e-02  ...  7.995605e-01  3.057495e+01 -1.481628e+00  0.000000
e+00
50%    8.568000e-01 -1.256000e-01 -3.222000e-01  8.239746e-03  ...  1.163101e+00  3.128992e+01 -1.571655e-01  0.000000
e+00
75%    8.922000e-01 -9.579998e-02 -2.098000e-01  4.435730e-02  ...  1.546478e+00  3.181512e+01  1.618958e+00  2.000000
e+00
max    2.029800e+00  5.390000e-01  1.246800e+00  1.499313e+00  ...  7.576752e+00  3.437039e+01  2.737885e+01  7.000000
e+00

[8 rows x 9 columns]
Shape of DataFrame after outlier removal: (2100456, 9)
```

**Fig: Top level view of the dataset**

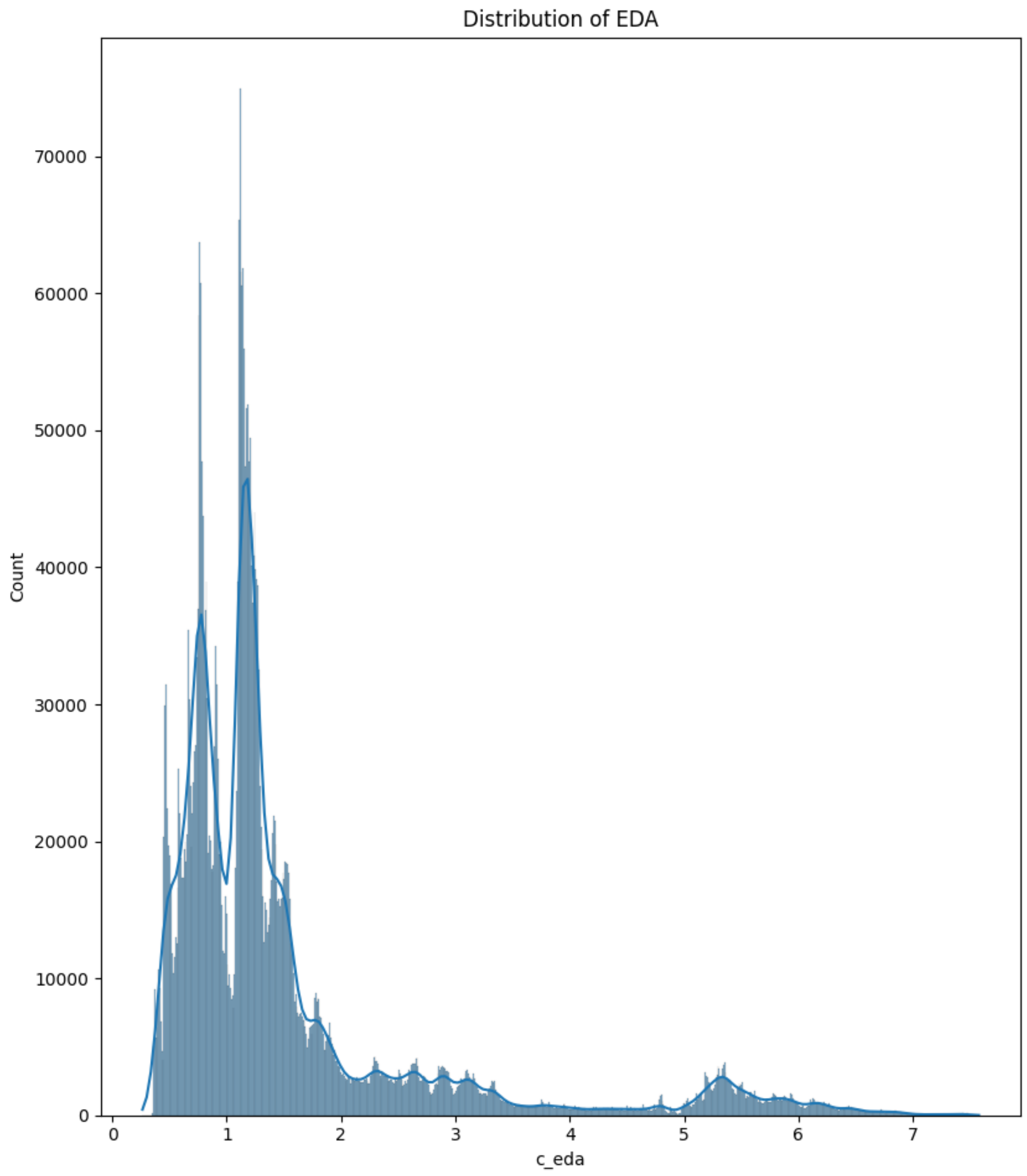


**Fig: Distribution of the dataset**

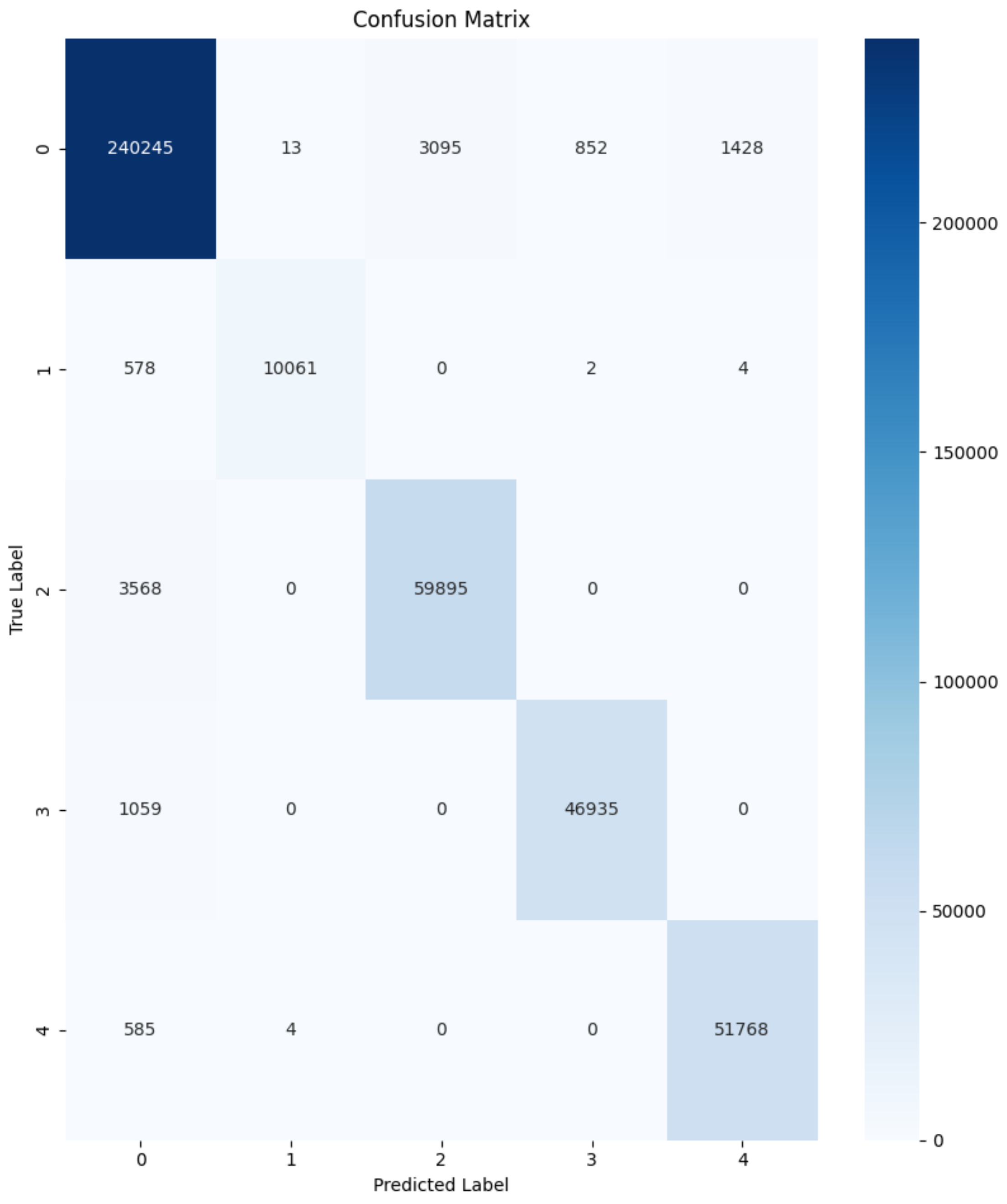
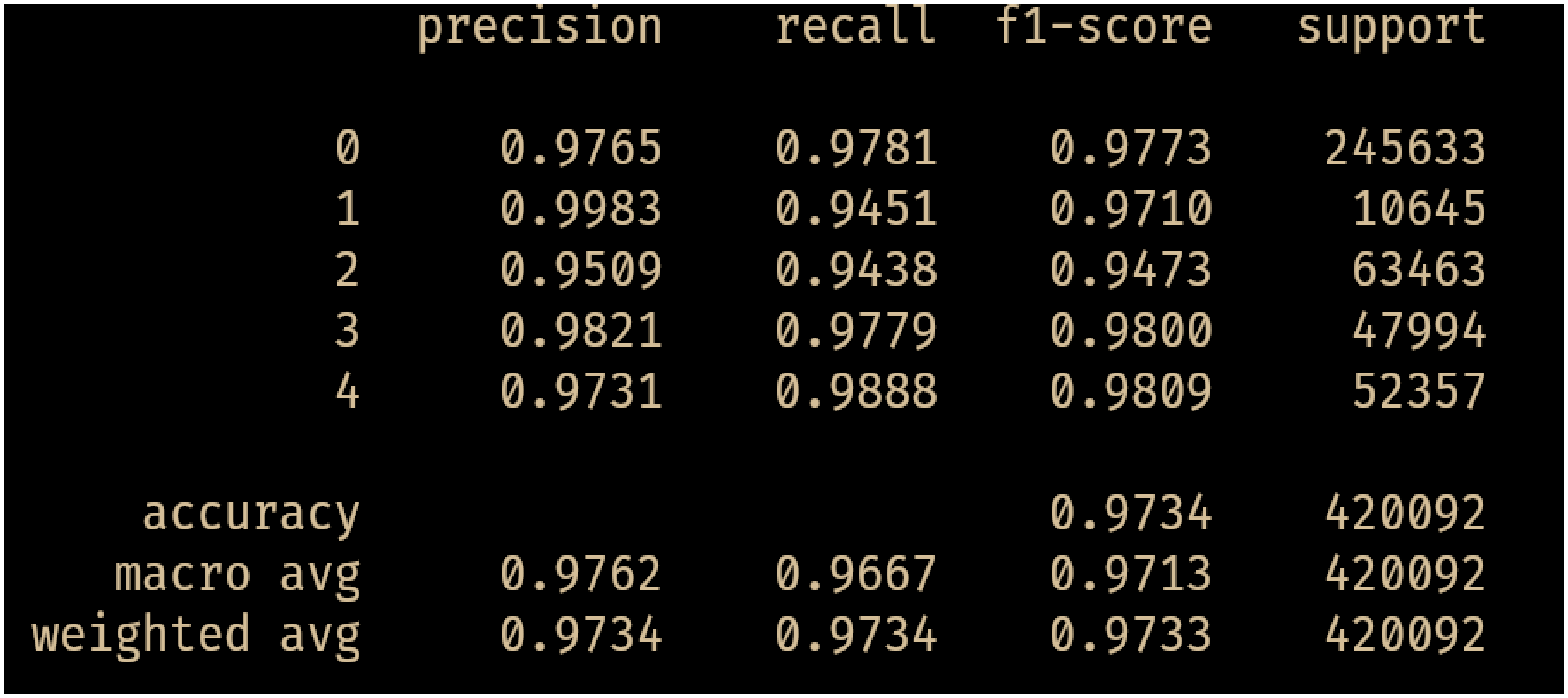
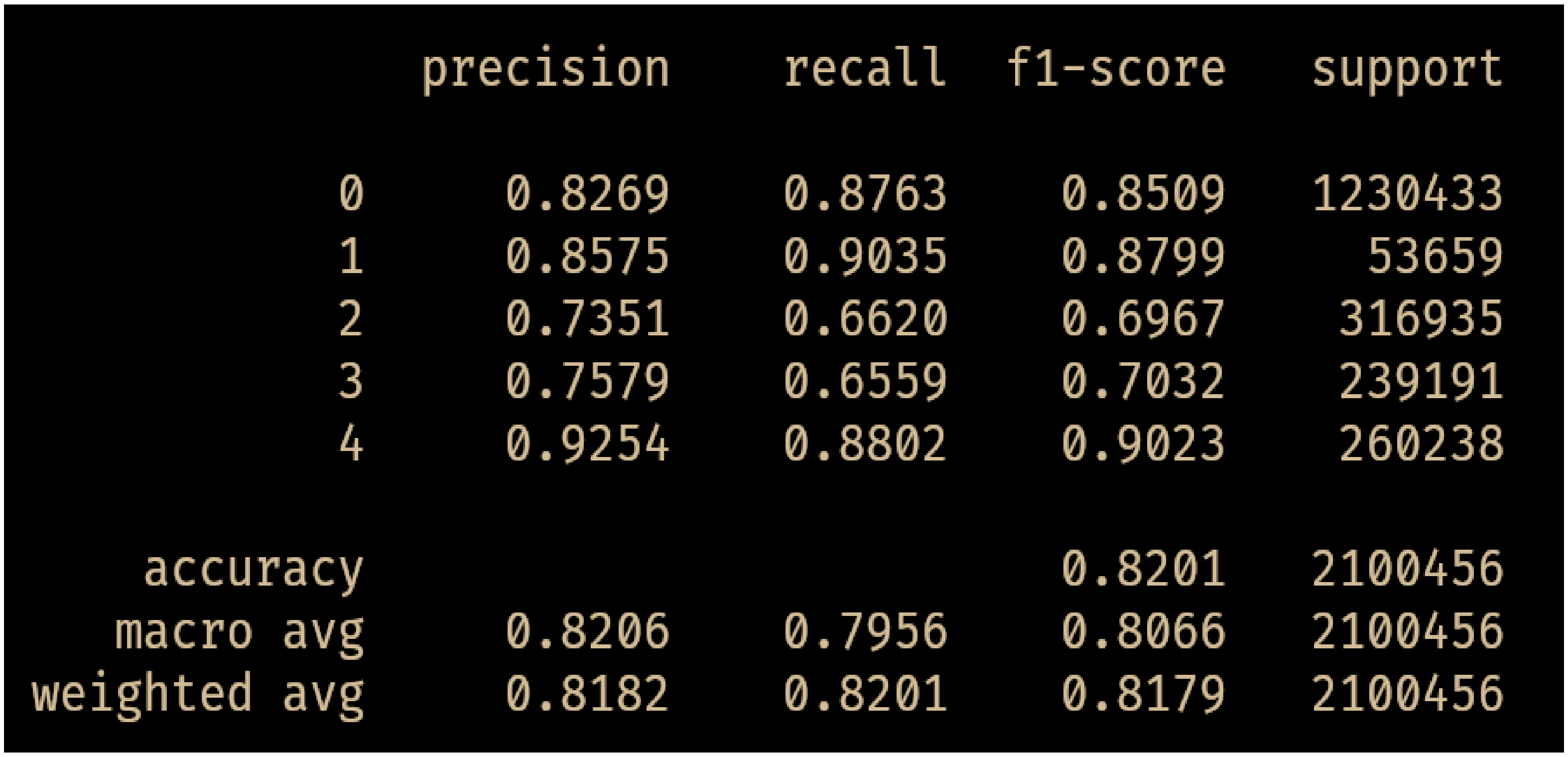


**Fig: Final confusion matrix after classification**



```
              precision    recall  f1-score   support

           0     0.9765    0.9781    0.9773    245633
           1     0.9983    0.9451    0.9710     10645
           2     0.9509    0.9438    0.9473     63463
           3     0.9821    0.9779    0.9800     47994
           4     0.9731    0.9888    0.9809     52357

    accuracy                         0.9734    420092
   macro avg     0.9762    0.9667    0.9713    420092
weighted avg     0.9734    0.9734    0.9733    420092
```

**Fig: XGBoost classification report**



```
              precision    recall  f1-score   support

           0     0.8269    0.8763    0.8509   1230433
           1     0.8575    0.9035    0.8799     53659
           2     0.7351    0.6620    0.6967    316935
           3     0.7579    0.6559    0.7032    239191
           4     0.9254    0.8802    0.9023    260238

    accuracy                         0.8201   2100456
   macro avg     0.8206    0.7956    0.8066   2100456
weighted avg     0.8182    0.8201    0.8179   2100456
```

**Fig: Cross-Validation classification report**