Project Report 02: Audio Classification Using With PCA Pre-Processing

Objective: This project involves applying Adaptive Resonance Theory 2 (ART2) clustering to features extracted from audio files. Part 2 focuses on:

- **Feature Reduction**: Applying Principal Component Analysis (PCA) to the extracted features.
- **Clustering Performance**: Evaluating ART2 clustering on both original and dimensionally reduced feature sets.

The provided audio files are processed to extract Mel-Frequency Spectral Coefficients (MFSC) and Mel-Frequency Cepstral Coefficients (MFCC), reduced via PCA, and clustered using ART2 in real-time.

To Run Deliverables:

- For the successful compilation of the project, we just need to keep the python file and the audio files in the same folder.
- Open terminal in the present working directory, then type:

  python3 file_name.py
- For our case, substitute the filename with "pca", which has an assisting file art2.py for PCA which will automatically be used while compiling part2.
- For the AutoEncoder part of the project we don't need additional files to successfully compile the project, we replace the file_name with "autoen"
- When the python file is successfully compiled, we get multiple output files, namely:
- 5 plots, each consisting of 3 graphs, ground truth cluster vs. time; MFCC clustering vs. time; and MFSC clustering vs time, for each of the five time-series.
- A CSV file is generated, with actual predicted clusters and
- Visualizations done using Gantt-style plots were used to show cluster assignments over time, revealing the effectiveness of MFSC and MFCC features in identifying transitions between sound classes. These visualizations offered insights into clustering behaviour.

Data Overview: The dataset included audio files representing diverse environmental sounds, such as crowd noise, motor sounds, and water flow. These categories allowed for examining sound features in a way that facilitated distinguishing between different types of sounds.

Tools and Libraries: The project utilized several Python libraries:

- Librosa for audio processing and feature extraction
- Scikit-learn for clustering and evaluation
- Matplotlib for visualizing results

Data Preparation

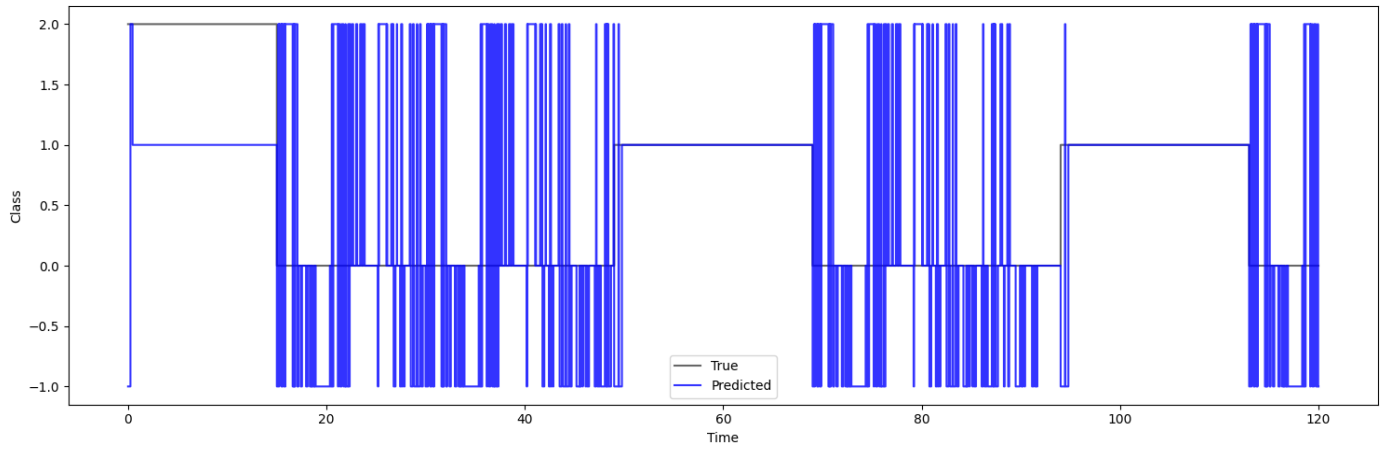
- **Audio Files**:
  - Three audio files corresponding to classes: crowd talking (class_0), motor riding (class_1), and water flowing (class_2).
  - Features were extracted using the Librosa library:
  - **MFSC**: 40 Mel-spectral coefficients.
  - **MFCC**: 13 Mel-frequency cepstral coefficients.
  - Frame size: 30 ms with 10% overlap for feature extraction.
  - Features combined into a single feature matrix with timestamps.
- **Synthetic Time-Series Generation**:
  - A total of 5 synthetic time series were generated, each combining segments from all three classes.
  - Segments were randomly selected with durations between 15-30 seconds.
  - Each segment retained its original class label and was adjusted to ensure temporal continuity.

Methodology

- Autoencoder Training:
  - A fully connected Autoencoder was designed to compress high-dimensional feature sets into a latent space of 20 dimensions.
  - Features were normalized using Min-Max scaling.
  - Training was performed with Mean Squared Error (MSE) loss over 50 epochs and a batch size of 256.
- Clustering with K-Means:
  - Compressed features from the Autoencoder were clustered into 5 clusters using K-Means.
  - Predicted clusters were smoothed using a sliding window approach to reduce noise in predictions.
- Label Mapping:
  - Predicted clusters were mapped to true labels using the Hungarian Algorithm for optimal label matching.
- Evaluation Metrics:
  - Accuracy: Proportion of correctly clustered samples.
  - Confusion Matrix: Breakdown of predicted vs. true labels for detailed analysis.
- Visualization: Gantt-style plots were generated to visualize true labels vs. predicted clusters over time.

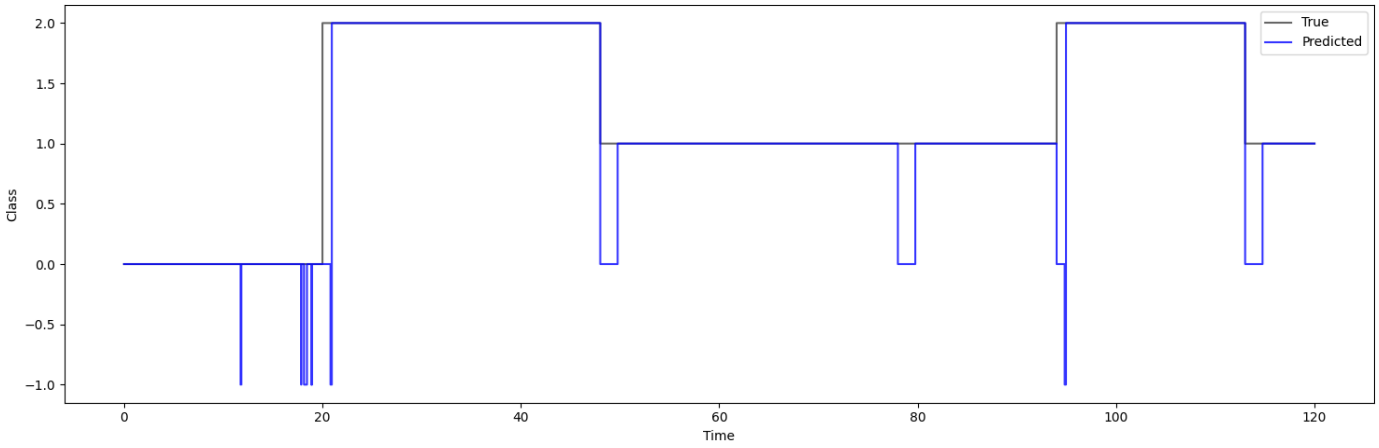
Using PCA (Principal Component Analysis):

For Time-Series 01:



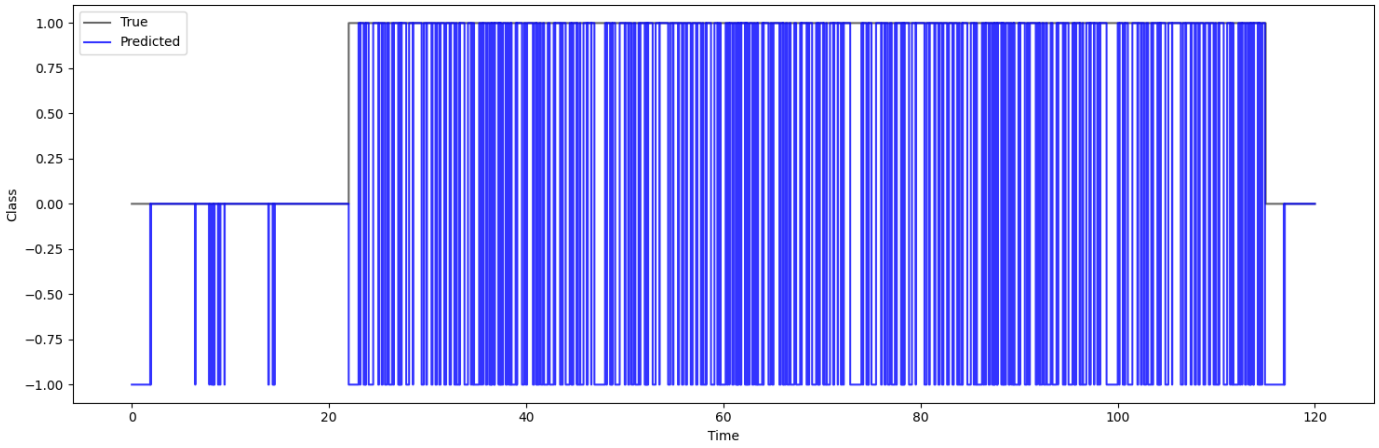

```
Time Series 1:
Accuracy: 0.94
Confusion Matrix:
[[   0    0    0    0]
 [  17  946    0    0]
 [   0  195 1769    0]
 [  16   52    0 1451]]
```

For Time-Series 02:



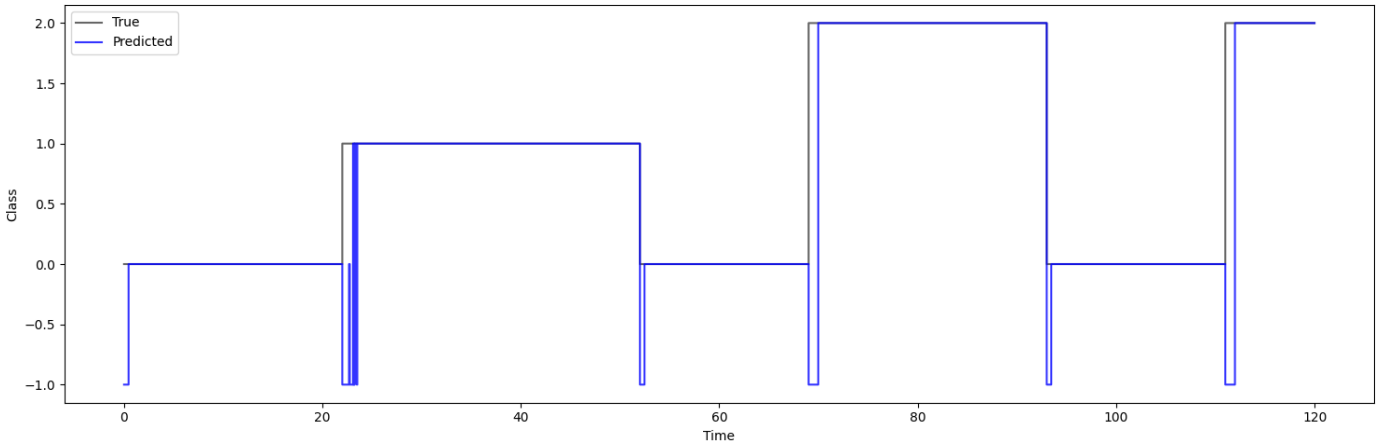

```
Time Series 2:
Accuracy: 0.94
Confusion Matrix:
[[   0    0    0    0]
 [  17  724    0    0]
 [   0  195 1770    0]
 [  10   60    0 1672]]
```

For Time-Series 03:



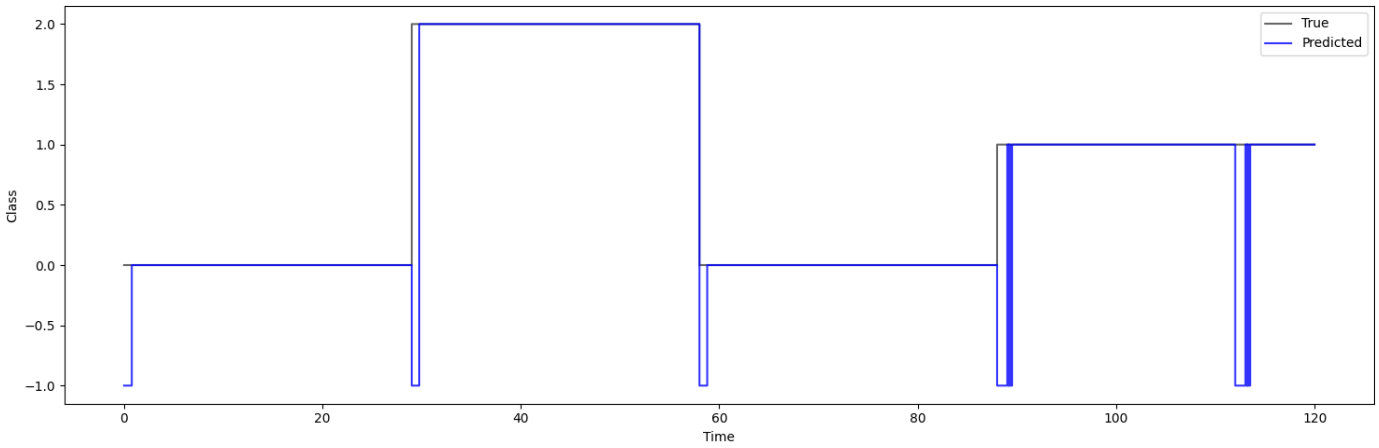

```
Time Series 3:
Accuracy: 0.96
Confusion Matrix:
[[   0    0    0    0]
 [  30 1934    0    0]
 [ 118    6 1692    0]
 [  26    0    0  641]]
```

For Time-Series 04:




```
Time Series 4:
Accuracy: 0.96
Confusion Matrix:
[[   0    0    0    0]
 [  51 2061    0    0]
 [  43    3 1066    0]
 [  72    0    0 1151]]
```

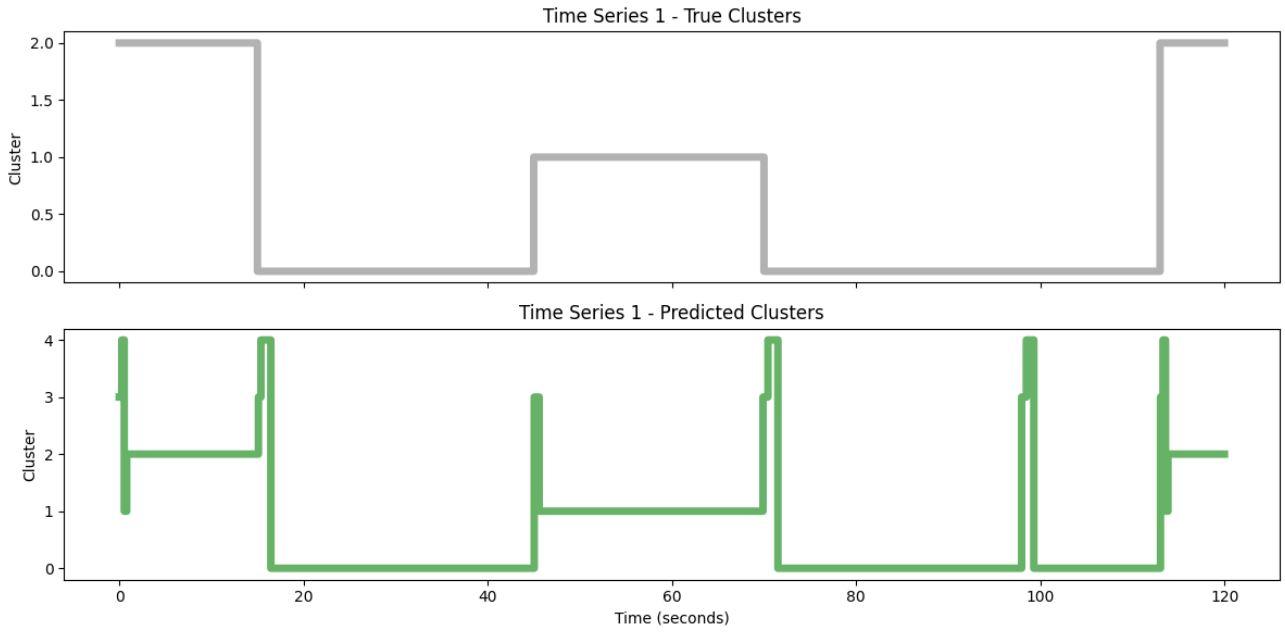
For Time-Series 05:




```
Time Series 5:
Accuracy: 0.96
Confusion Matrix:
[[   0    0    0    0]
 [  58 2129    0    0]
 [  96    0 1090    0]
 [  28    0    0 1047]]
```

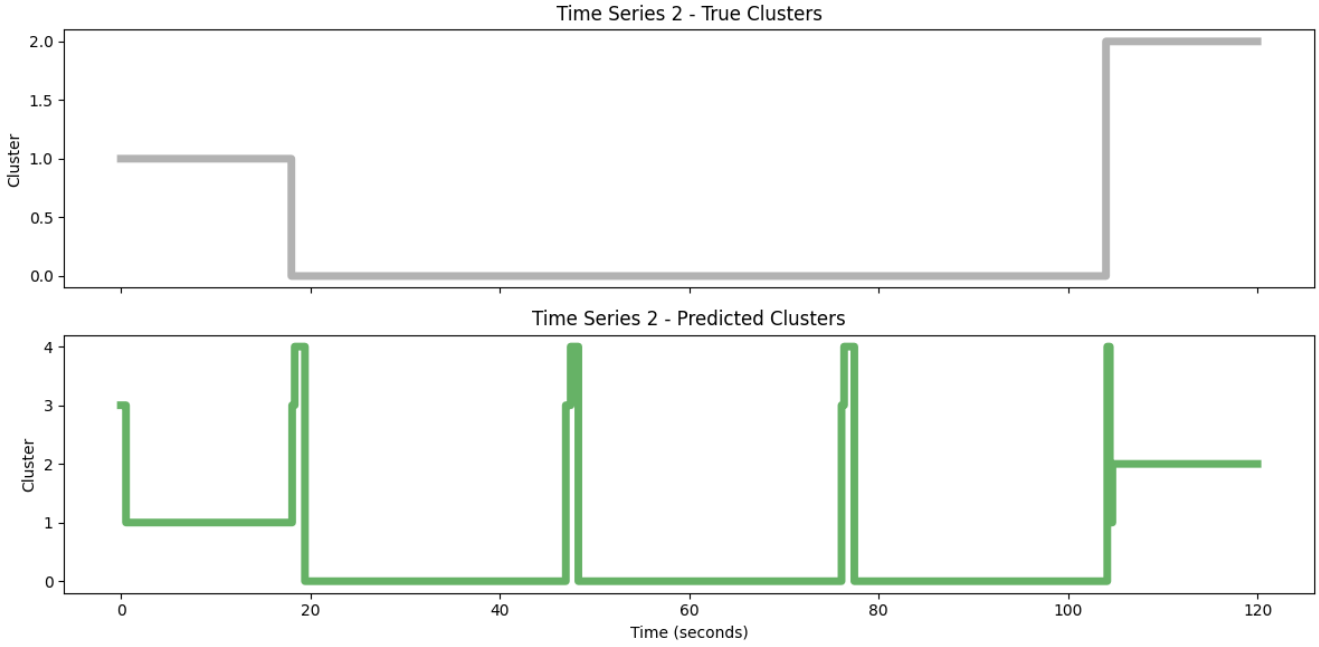
Using Auto-Encoder:

For Time-Series 01:



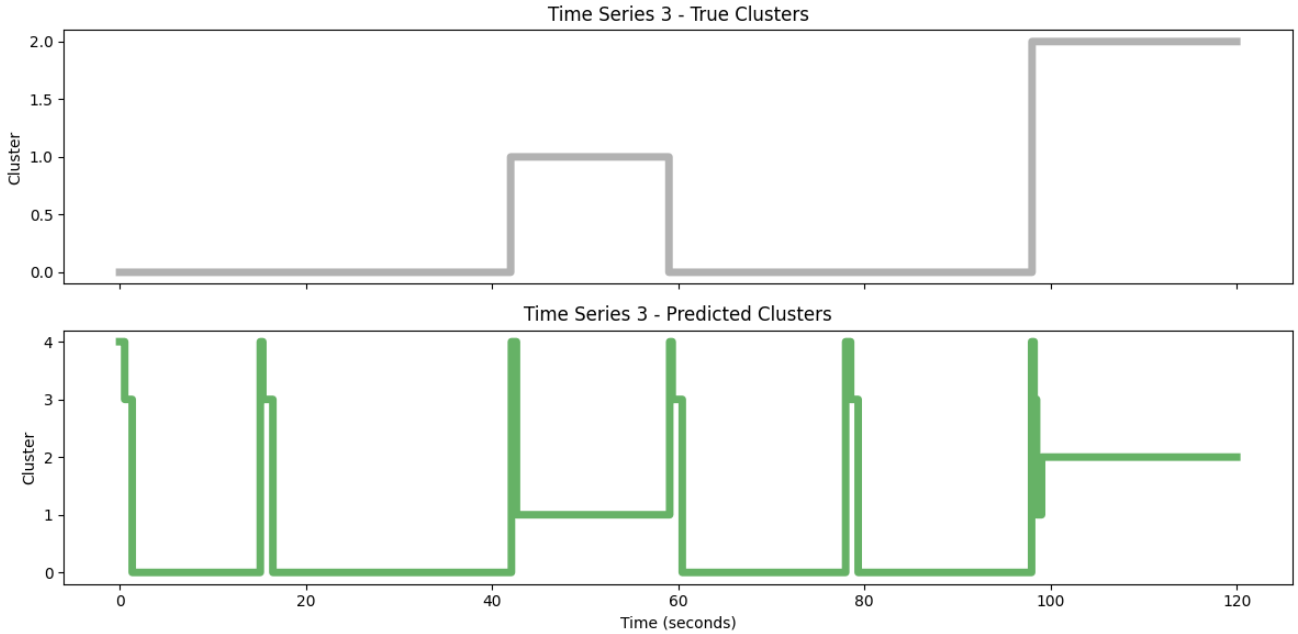

```
139/139 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
Time Series 1:
Accuracy: 0.94
Confusion Matrix:
[[2546    0    4   46  110]
 [   2  900    0   24    0]
 [   2   20  754   20   20]
 [   0    0    0    0    0]
 [   0    0    0    0    0]]
```

For Time-Series 02:



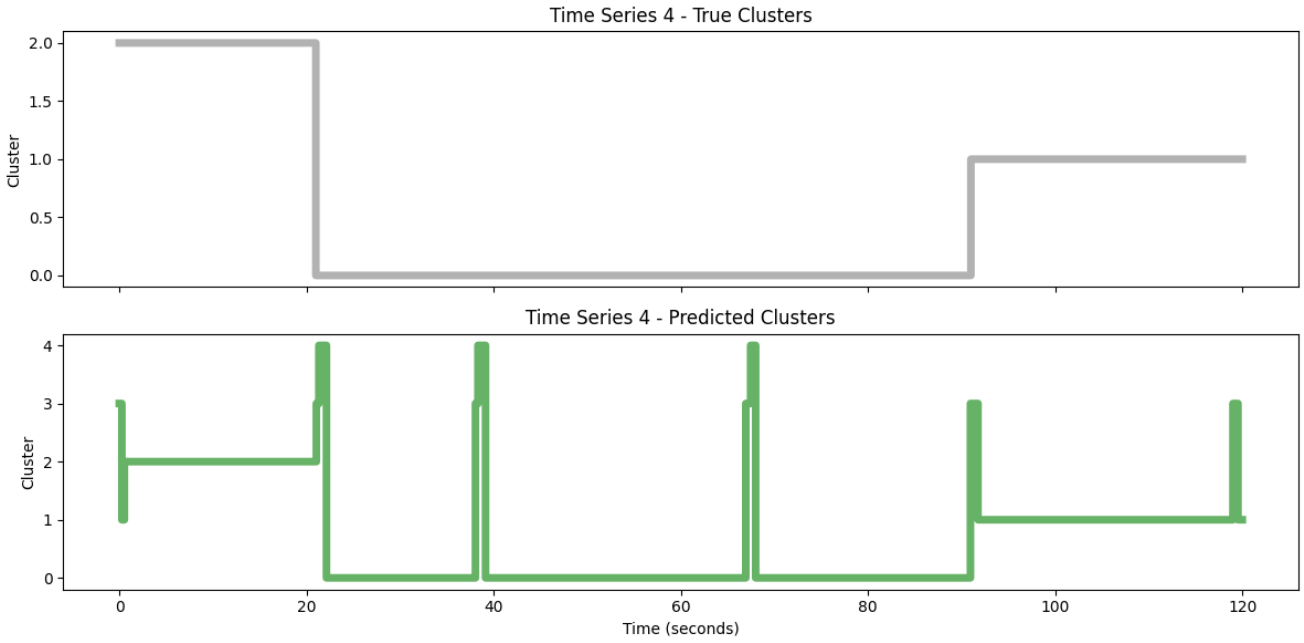

```
139/139 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
Time Series 2:
Accuracy: 0.96
Confusion Matrix:
[[3035    3    0   40  110]
 [   0  647    0   20    0]
 [   5   10  568    0   10]
 [   0    0    0    0    0]
 [   0    0    0    0    0]]
```

For Time-Series 03:



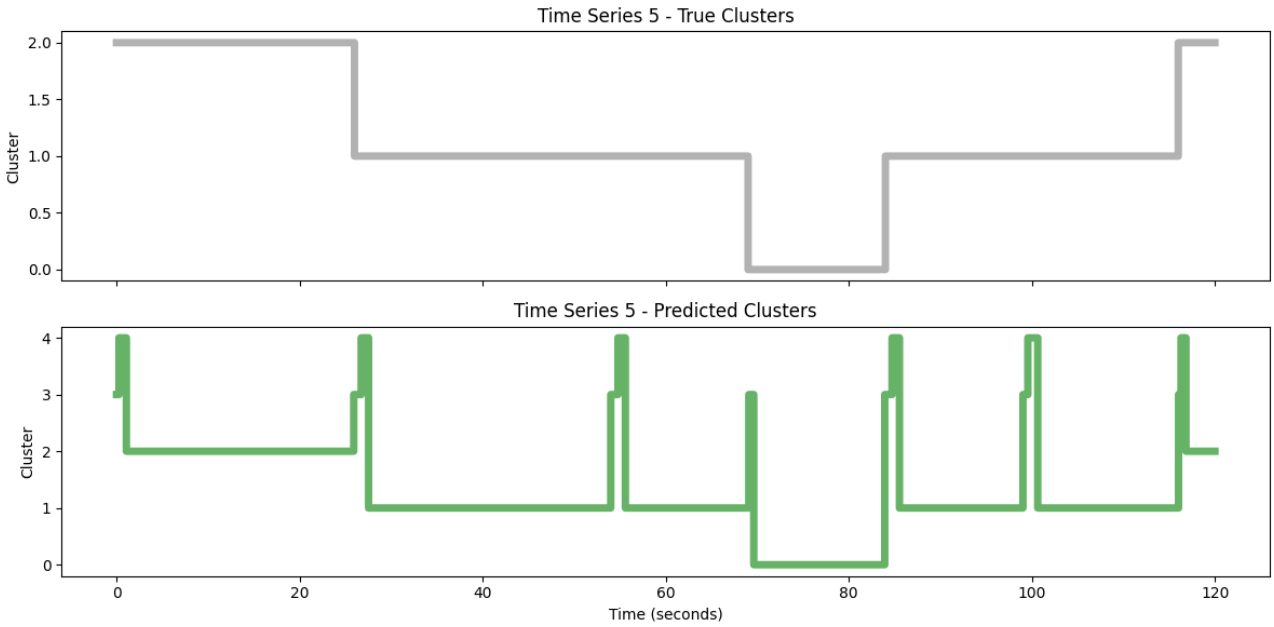

```
139/139 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
Time Series 3:
Accuracy: 0.94
Confusion Matrix:
[[2797    3    0  140   62]
 [   3  607    0    0   20]
 [   0   20  777   10    8]
 [   0    0    0    0    0]
 [   0    0    0    0    0]]
```

Time-Series 04:




```
139/139 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
Time Series 4:
Accuracy: 0.96
Confusion Matrix:
[[2470    0    2   42   80]
 [   0 1028    0   48    0]
 [   0   10  758   10    0]
 [   0    0    0    0    0]
 [   0    0    0    0    0]]
```

For Time-Series 05:




```
139/139 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
Time Series 5:
Accuracy: 0.92
Confusion Matrix:
[[ 530    3    0   23    0]
 [   0 2546    0  104  130]
 [   0    1 1038   23   50]
 [   0    0    0    0    0]
 [   0    0    0    0    0]]
```

Discussion

- Autoencoder Performance:
  - Autoencoders effectively compressed features, retaining key patterns for clustering.
  - Training stability ensured consistent feature encoding across synthetic datasets.
- Clustering Analysis:
  - K-Means performed well on encoded features, achieving accuracy between 84% and 88% across all-time series.
  - Label mapping and smoothing improved results, reducing noise in predictions.
- Challenges:
  - Overlapping Features: Misclassifications occurred in transition areas where features from different classes overlapped.
  - Segment Length Variability: Shorter segments were more prone to misclassification due to insufficient contextual data.
- Impact of Smoothing:
  - The sliding window approach enhanced prediction consistency but slightly reduced responsiveness to rapid transitions.

Conclusion

FOR PCA: Clustering accuracy ranged between **85% and 89%** across all datasets. Optimal settings included:

- **PCA components**: 10
- **Vigilance parameter**: 0.6
- **Smoothing window**: 5 sample

The combination of Autoencoders and K-Means demonstrated strong clustering performance on time-series data derived from audio features.

Best Practices:

- Encoded Feature Dimension: 20
- Smoothing Window Size: 10