**1.Introduction**

This report describes the Stage 1 Clustering Pipeline developed to identify and categorize manufacturing bottlenecks using time‑series clustering techniques. The primary goal was to transform raw stoppage events into meaningful clusters that capture similar bottleneck behavior across production lines, stoppage reasons, and shifts. This pipeline supports downstream analysis by revealing patterns in equipment downtime, enabling targeted process improvements.

**2. Data Loading & Initial Cleanup**

We begin by importing production data from an Excel workbook containing timestamped stoppage events and their durations. Key steps include:

* Column Verification: Ensuring presence of Line, Stoppage Category, Stoppage Reason, Shift Id, and Bottleneck Duration Seconds columns; missing columns trigger an explicit error .
* Missing Duration Removal: Rows lacking Bottleneck Duration Seconds are dropped, as they cannot contribute to meaningful clustering.
* Active State Flag: A binary is\_active column (1 = active bottleneck, 0 = idle) is derived from Stoppage Category ≠ "Not Occupied".

These steps ensure a reliable foundation for time‑series extraction.

**3. Outlier Removal**

To mitigate the influence of extreme durations, we apply the Interquartile Range (IQR) method:

1. Q1 = df['Bottleneck Duration Seconds'].quantile(0.25)

2. Q3 = df['Bottleneck Duration Seconds'].quantile(0.75)

3. IQR = Q3 - Q1

4. lower, upper = Q1 - 1.5\*IQR, Q3 + 1.5\*IQR

5. df = df[df['Bottleneck Duration Seconds'].between(lower, upper)]

6.

Any duration outside [Q1 – 1.5·IQR, Q3 + 1.5·IQR] is excluded, reducing noise from anomalous stoppages.

**4. Summary Statistics & Time‑Series Extraction**

We group data by (Line, Stoppage Reason, Shift Id) to compute basic statistics (mean, std, min, max, count) for each group. Simultaneously, we build a dictionary of raw time‑series sequences:

1. stats = df.groupby([...])['Bottleneck Duration Seconds']

2. .agg(['mean','std','min','max','count'])

3.

4. # Only groups with >1 event

5. ts\_dict = { key: grp['Bottleneck Duration Seconds'].values

6. for key, grp in df.groupby([...]) if len(grp)>1 }

7.

This dual output (stats and ts\_dict) supports both tabular summaries and sequence‑based clustering.

**5. Distance Matrix Computation (DTW)**

To compare time series of varying lengths, we use Dynamic Time Warping (DTW):

1. Padding & Scaling: Convert list of sequences into a padded 3D array via to\_time\_series\_dataset, then normalize each series to zero mean and unit variance with TimeSeriesScalerMeanVariance.
2. DTW Distances: Compute pairwise DTW using cdist\_dtw, yielding a symmetric distance matrix D.
3. Visualization: A heatmap of D highlights clusters of similar behavior (Figure 1) filecite.

1. X = to\_time\_series\_dataset(list(ts\_dict.values()))

2. X\_scaled = scaler.fit\_transform(X)

3. D = cdist\_dtw(X\_scaled)

4.

**6. Clustering Methods**

We explore two unsupervised algorithms on DTW distances:

* Agglomerative Hierarchical Clustering (AHC) with complete linkage on the precomputed D.
* Density‑Based Clustering (DBSCAN/HDBSCAN) using D as a precomputed distance metric.

**6.1. Silhouette‑Based Tuning**

To select the optimal cluster count for AHC, we evaluate silhouette scores for k = 2…7:

1. scores = []

2. for k in range(2,8):

3. labels = AgglomerativeClustering(n\_clusters=k,

4. metric='precomputed',

5. linkage='average')

6. .fit\_predict(D)

7. sil = silhouette\_score(D, labels, metric='precomputed')

8. scores.append((k, sil))

9. opt\_k = max(scores, key=lambda x: x[1])[0]

10.

The maximum silhouette selects k = 2 for our data.

**6.2. Final Clustering**

With k=2, we assign each (Line, Reason, Shift) to one of two clusters via AHC. We also run HDBSCAN (fallback to DBSCAN if HDBSCAN unavailable) for density‑based labels.

**7. Evaluation Metrics We compute:**

* Silhouette Score on D for AHC and density clusters.
* Davies–Bouldin and Calinski–Harabasz are omitted due to incompatibility with non‑Euclidean distances.

The AHC clusters achieve a silhouette of ~0.50 vs. ~0.18 for density-based, indicating tighter, more distinct grouping under the AHC solution.

**8. Visualizations**

* Dendrogram: Hierarchical tree for k=2 clusters to illustrate merge distances.
* Silhouette Plot: Distribution of silhouette coefficients by cluster, with average line to assess cohesion.
* Sample Time Series: Up to five representative sequences per cluster, exposing characteristic bottleneck patterns.

**9. Results Export The final artifacts include:**

* clustering\_metrics\_stage1.csv: Silhouette scores per method.
* cluster\_assignments\_stage1.csv: Table of group keys with assigned cluster labels.
* ts\_dict\_stage1.npy: Serialized dictionary of time series for downstream dashboards.
* dendrogram\_snapshot.png: Static dendrogram visualization.

**10. Conclusion Stage 1** of the clustering pipeline successfully identifies two major bottleneck behaviours patterns across production lines and shifts. The DTW‑based AHC approach outperforms density methods by silhouette score, suggesting strong, cohesive clusters. These insights enable targeted troubleshooting and continuous process optimization in subsequent stages.