CS658: Topics in Malware Analysis & Intrusion Detection

Group 9

Homework 2: PASAD Assignment Report

Fall Semester 2024

Ankit Kaushik	**220158**
Mohsin Khan	**241110612**
Naman Kumar Jaiswal	**220687**
Sumit Rojaria	**221104**

Q1: Implementing PASAD on SWaT and TE Datasets

Dataset Analysis and Plotting:

For the implementation of the PASAD paper, we were provided the following datasets:

- **SWaT Dataset:** Captures normal and attack behavior from a water treatment system. It has time-series data for a total of 55 sensors, for our analysis we have chosen the ultra-filtration feed water tank level sensor LIT301 sensor readings for our analysis.
- **TE Dataset:** Tennessee Eastman process control dataset which contains 41 sensor readings for Chemical Protocol readings. The dataset contains a total of 5 datasets (3 for stealth attack and 2 for direct attack). We will be using sensor Xmeas 10 (Sensor 10) for our analysis.

PASAD Steps:

- **Embedding:** Construct Hankel matrices from training and test data.
- **SVD:** Decompose Hankel matrix into U, S, V^T; calculate cumulative singular value share.
- **Projection:** Project data onto lower dimensions using orthonormal basis from U.
- **Departure Score:** Compute scores for training and testing; set threshold for zero false alarms.
- **Sensor Plot:** Visualize sensor data and departure scores, marking anomalies.

Conclusion: This PASAD implementation efficiently detects anomalies in sensor data by computing departure scores and visualizing thresholds, with a focus on industrial anomaly detection.

We have reached the following results with our PASAD implementation:

Performance Comparison:

We updated our algorithm to use the transformation matrix as UU^T instead of U^T to see the following result. We majorly did not see a major increase in detection capabilities, but as a matter of fact Linear map took much less time for training and detection.
For SA1 dataset: For Projection Matrix, the total time taken for 3802 departure score computations is 125.16287660598755 Making the average time 0.032920272647550645.
For Linear Map, the total time taken for 3802 departure score computations is 0.6710443496704102 Making the average time 0.0001764977247949527

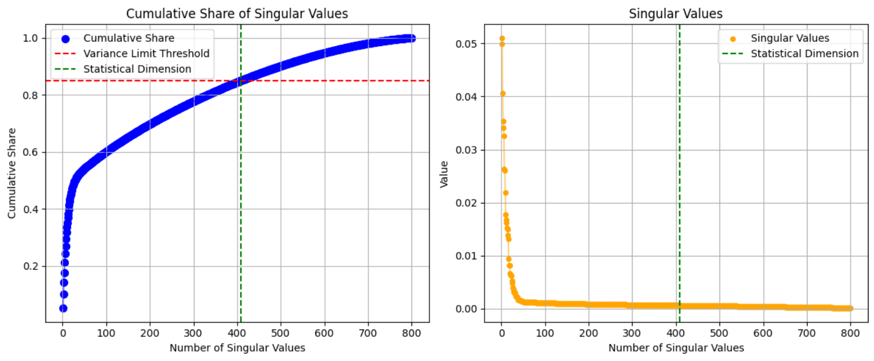

Figure 1: Scree Plot for Stealth Attack 1 Dataset

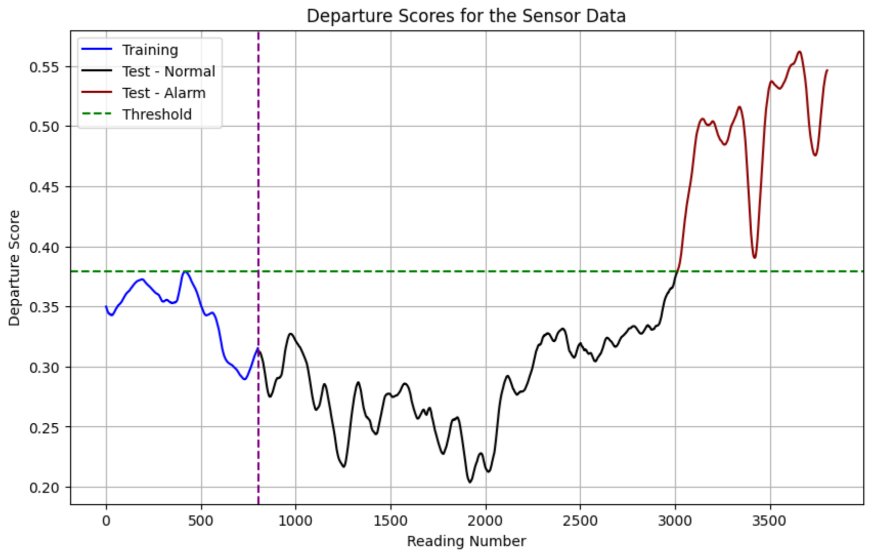

Figure 2: Departure Scores for the Sensor Data Stealth Attack 1 Dataset

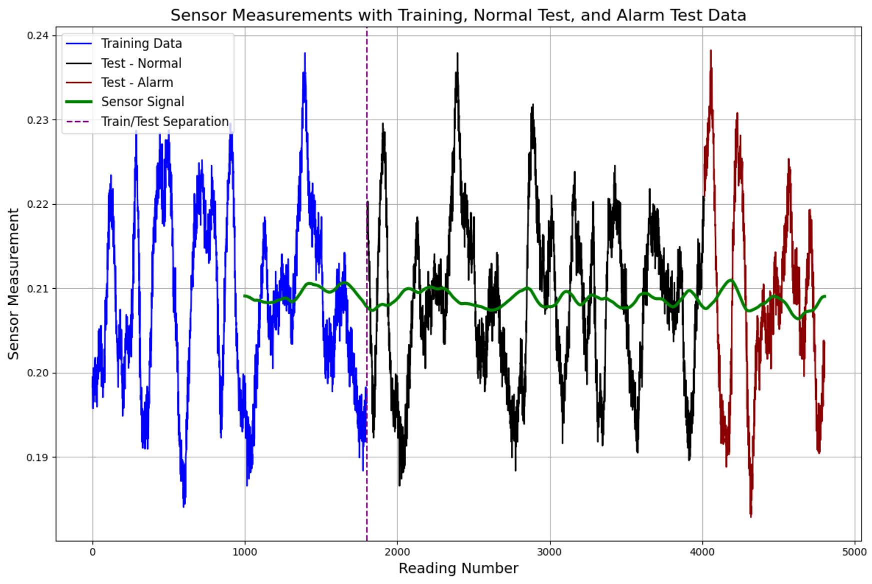

Figure 3: Sensor Readings for Stealth Attack 1 Dataset

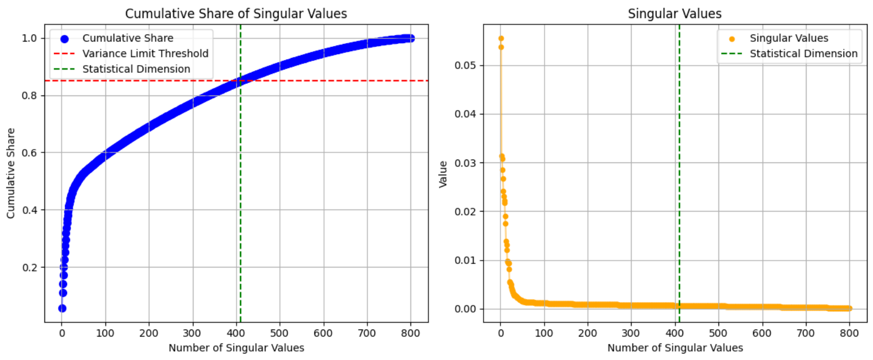

Figure 4: Scree Plot for Stealth Attack 2 Dataset

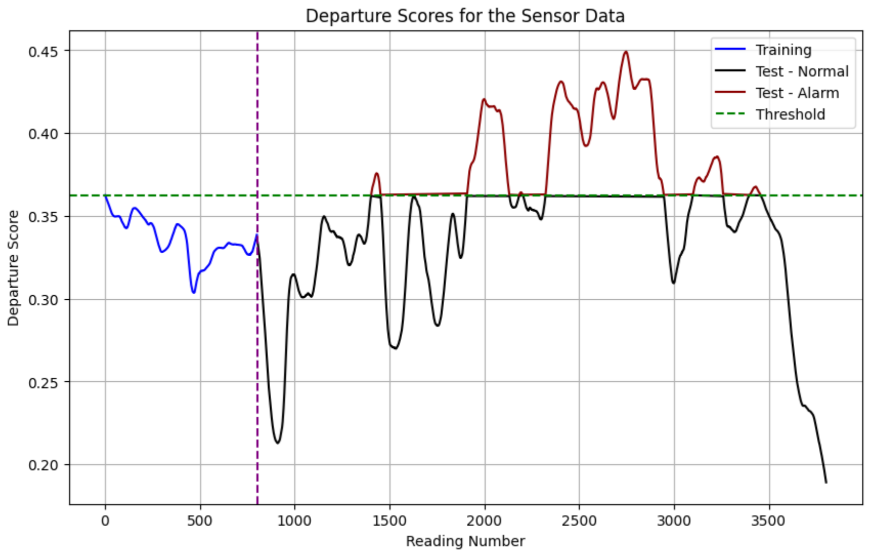

Figure 5: Departure Scores for Stealth Attack 2 Dataset

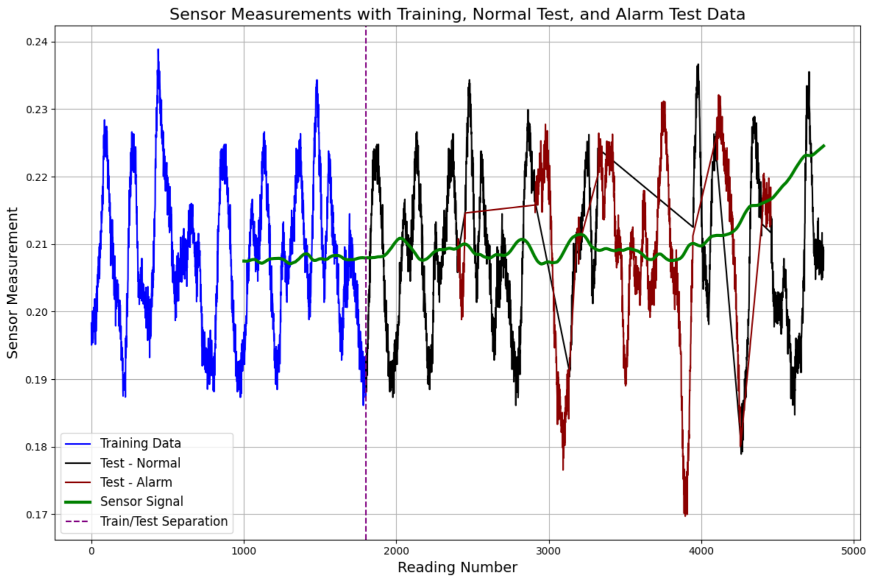

Figure 6: Sensor Readings for Stealth Attack 2 Dataset

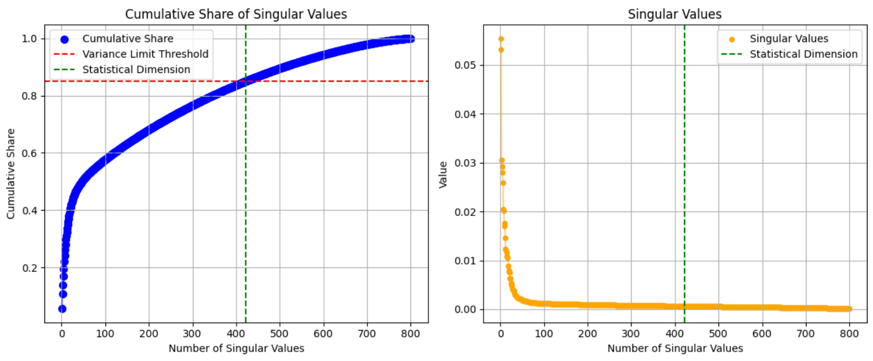

Figure 7: Scree Plot for Stealth Attack 3 Dataset

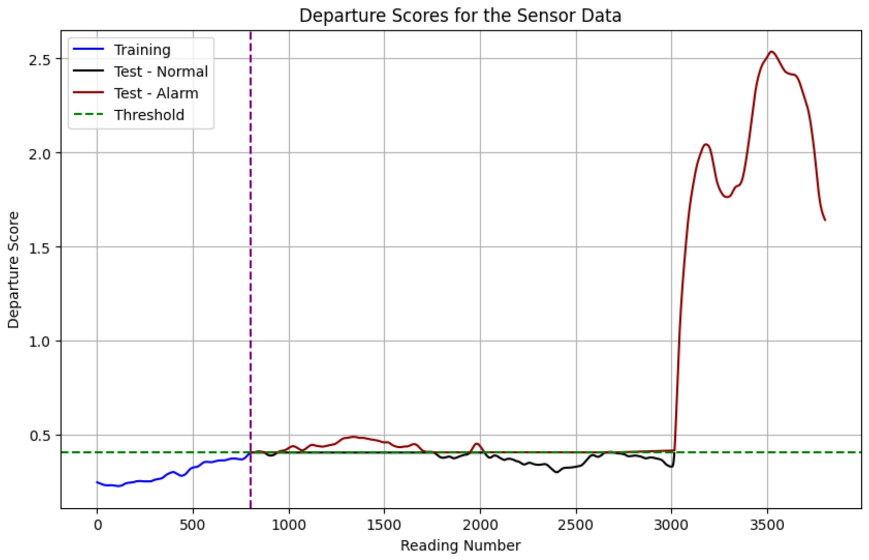

Figure 8: Departure Scores for Stealth Attack 3 Dataset

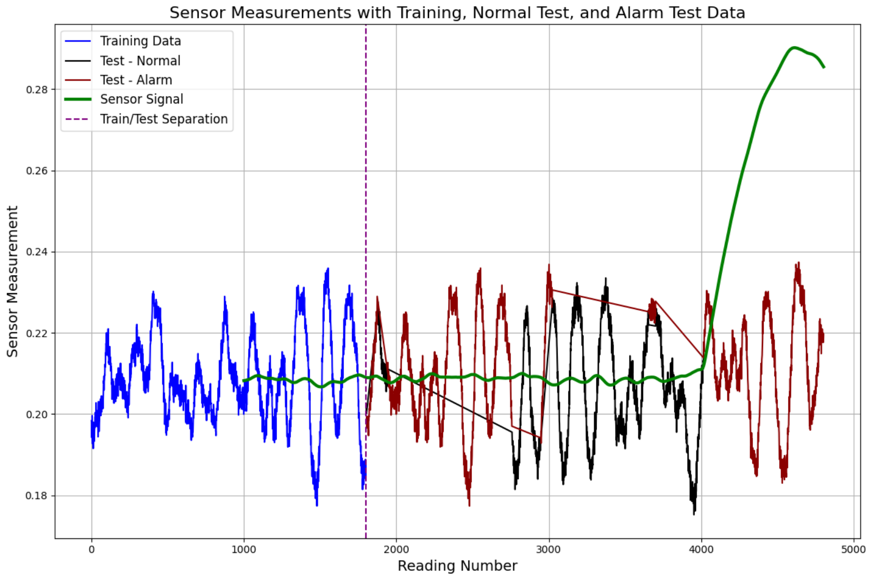

Figure 9: Sensor Readings for Stealth Attack 3 Dataset

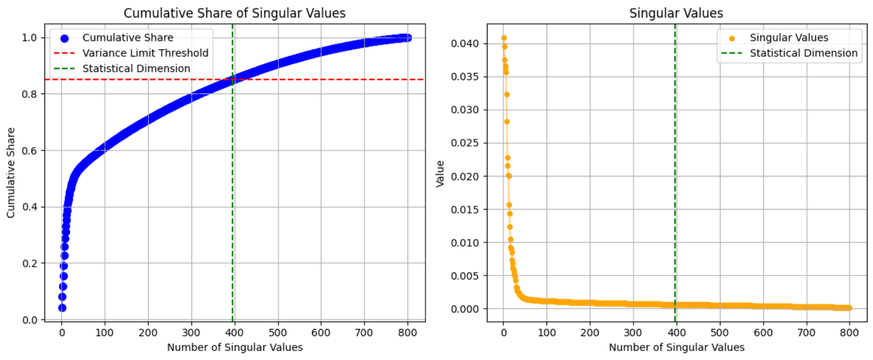

Figure 10: Scree Plot for Direct Attack 1 Dataset

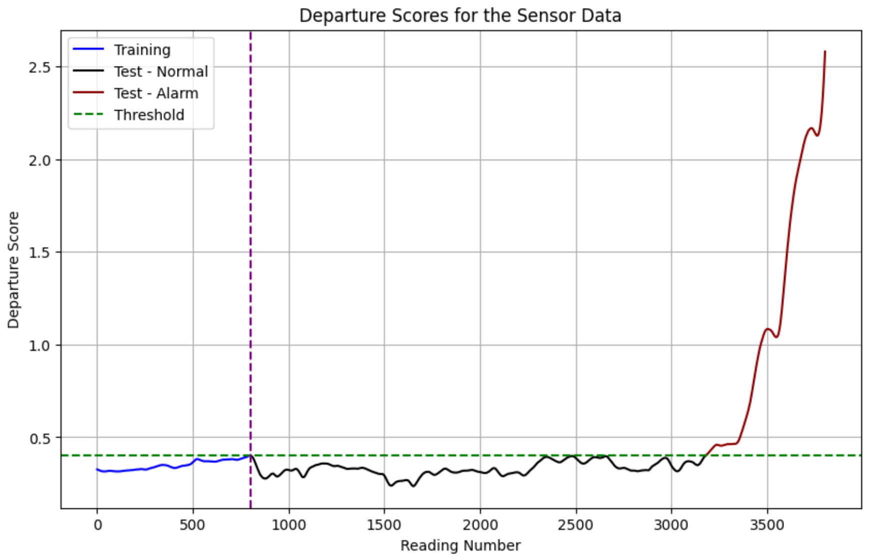

Figure 11: Departure Scores for Direct Attack 1 Dataset

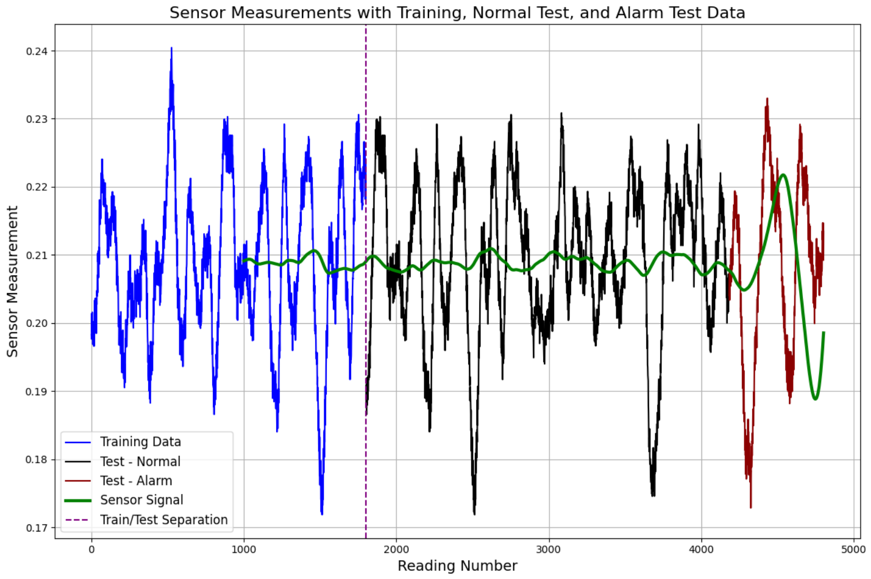

Figure 12: Sensor Readings for Direct Attack 1 Dataset

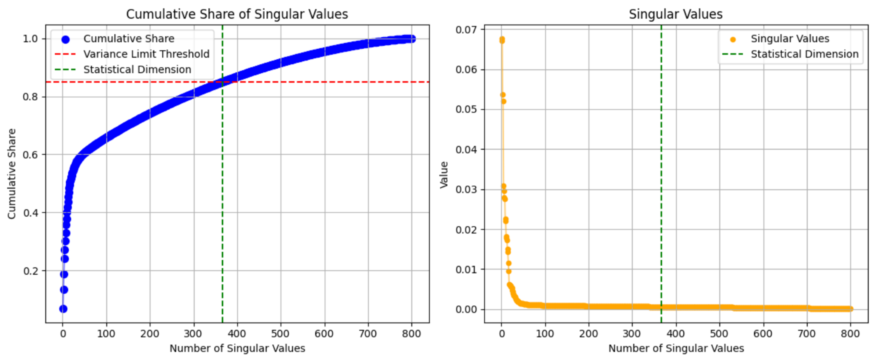

Figure 13: Scree Plot for Direct Attack 2 Dataset

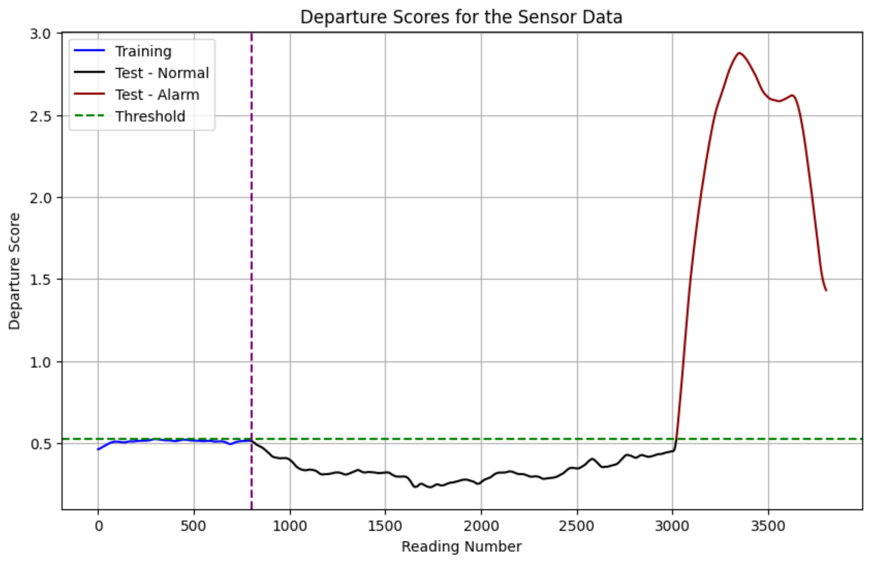

Figure 14: Departure Scores for Direct Attack 2 Dataset

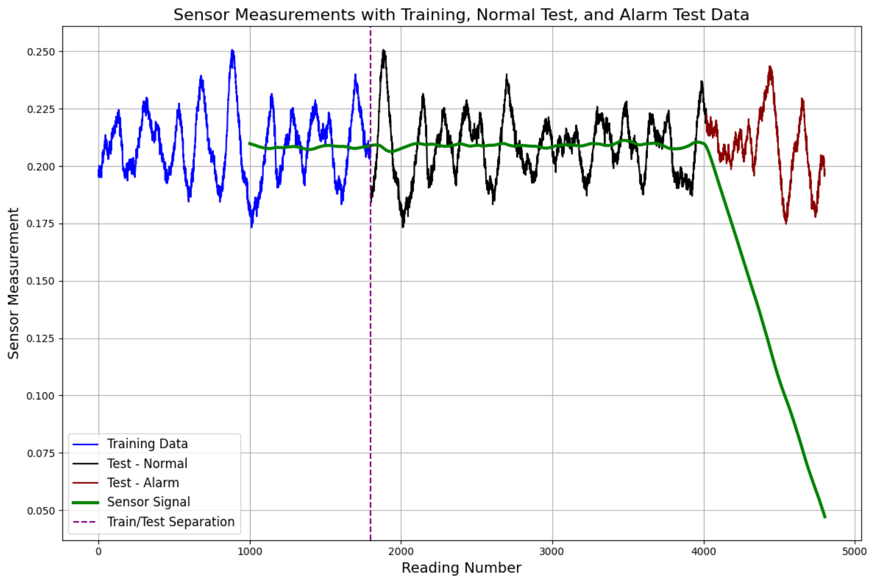

Figure 15: Sensor Readings for Direct Attack 2 Dataset

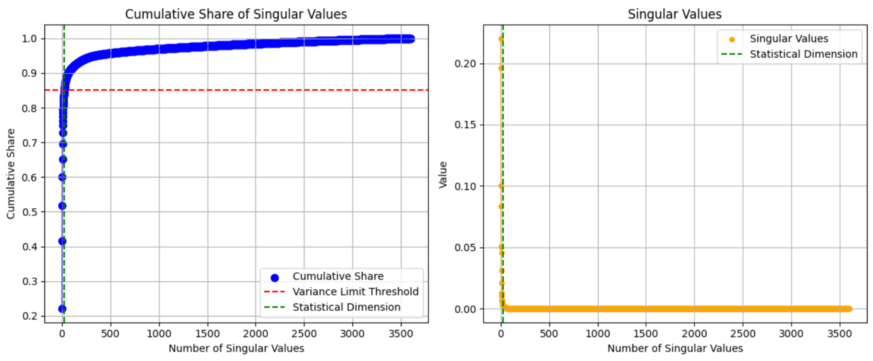

Figure 16: Scree Plot for SWaT Dataset

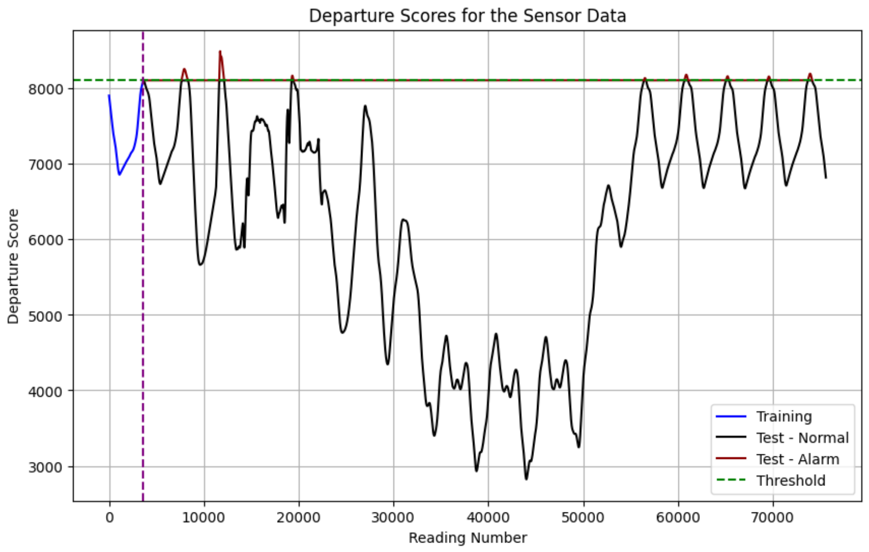

Figure 17: Departure Scores for SWaT Dataset

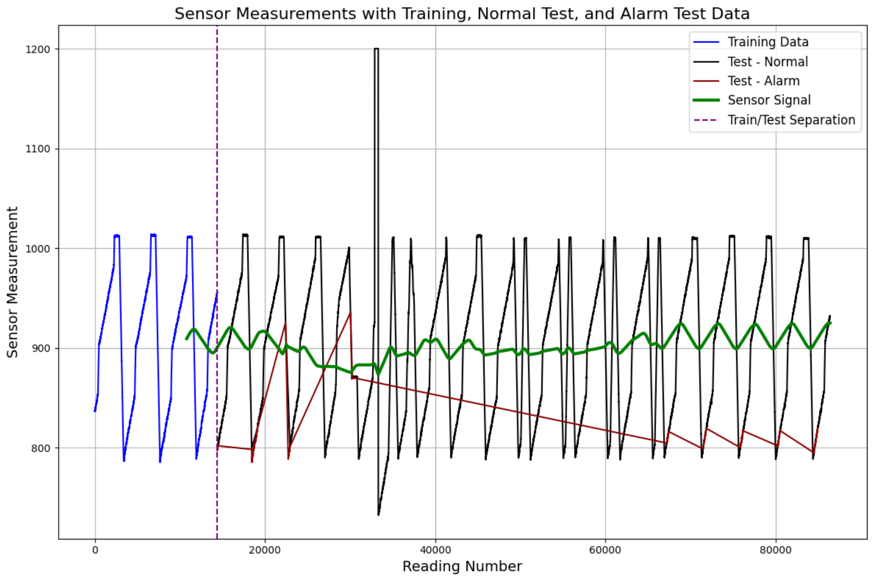

Figure 18: Sensor Readings for SWaT Dataset

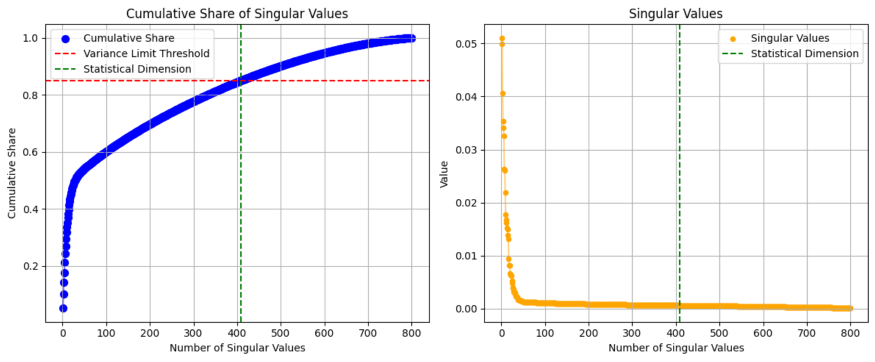

Figure 19: Scree Plot for Stealth Attack 1 Dataset, Projection Map

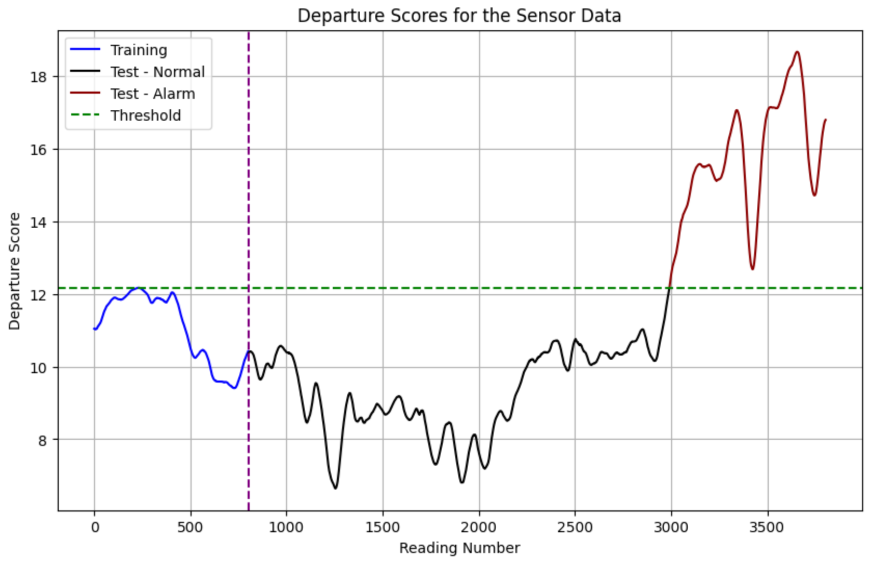

Figure 20: Departure Scores for Stealth Attack 1 Dataset, Projection Map

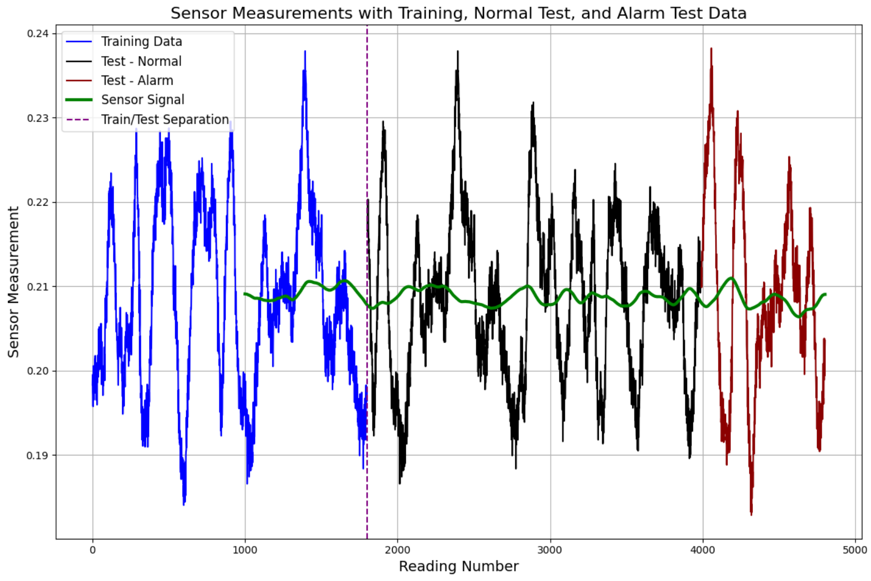

Figure 21: Sensor Readings for Stealth Attack 1 Dataset, Projection Map

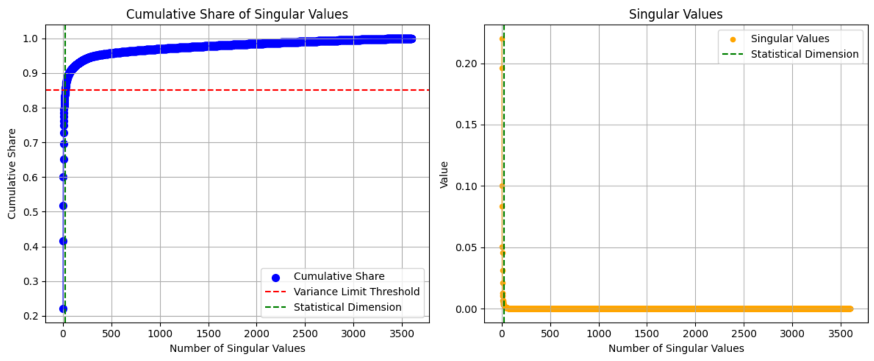

Figure 22: Scree Plot for SWaT Dataset, Projection Map

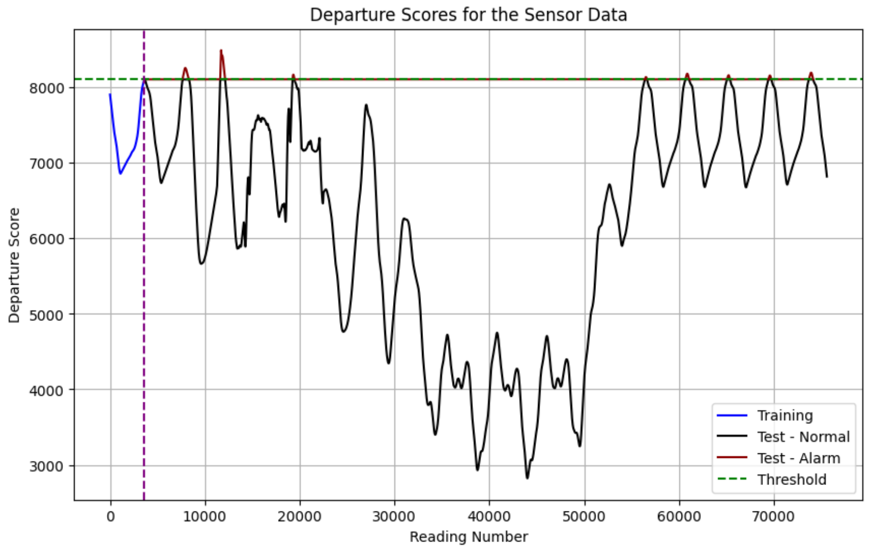

Figure 23: Departure Scores for SWaT Dataset, Projection Map

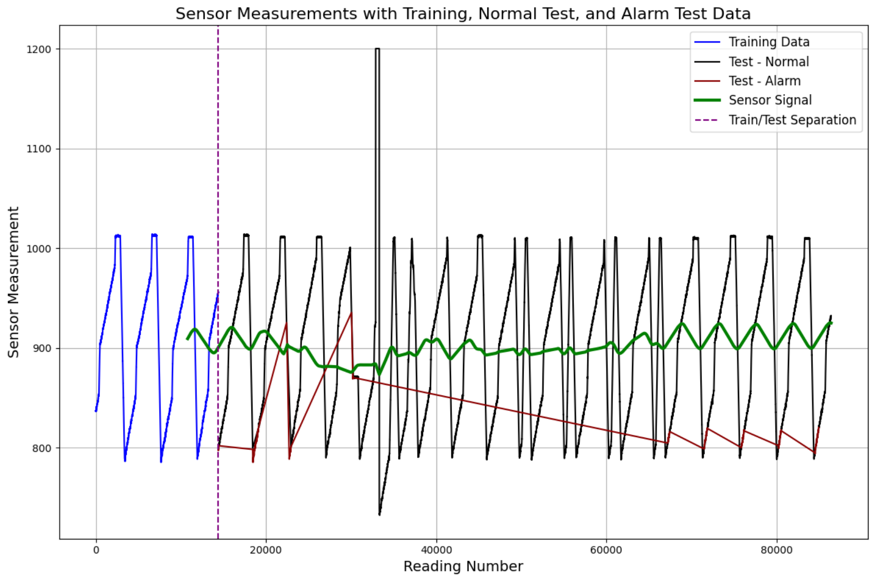

Figure 24: Sensor Readings for SWaT Dataset, Projection Map

Attack Scenario Analysis in TE Dataset:

While running the Attack Scenario for each of the data set in Tennessee Eastman Process, we achieve the following Alarm Counts for the normal Pasad implementation. The alarm Counts are 'SA1': 790, 'SA2': 1134, 'SA3': 1795, 'DA1': 621, 'DA2': 783

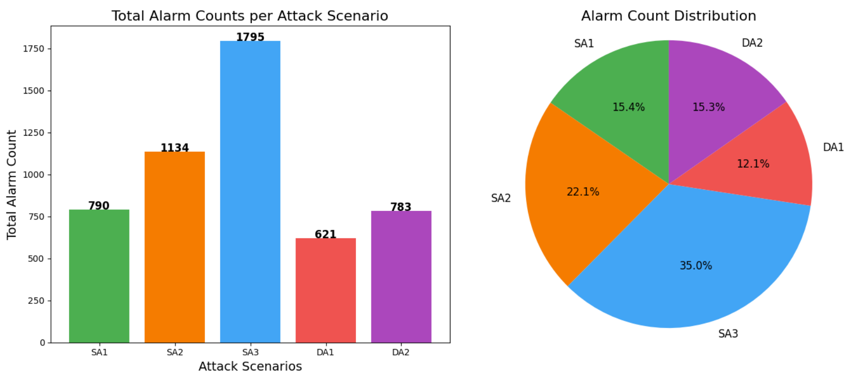

Figure 25: Plot of Alarm Counts and Corresponding Dataset

Q2: Exploring the Use of Centroid Instead of Mean of the Cluster

Justification:

In PASAD, the mean represents the average of all data points, calculated as:

$$\text{Mean} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

It minimizes squared deviations and is sensitive to outliers. The centroid, in this case, refers to the geometric center, calculated as:

$$C = \frac{\min(X) + \max(X)}{2}$$

This geometric center is less influenced by the density of points and is better suited for unevenly distributed data.

For PASAD, if the data is unevenly distributed without significant outliers, the centroid could improve performance by offering a more balanced representation of the data. However, for data with outliers or more uniform distributions, the arithmetic mean may still be more reliable. Centroid looks for minimum and maximum value, which even though computationally simpler, can be more easily swayed by outliers. Centroid looks at the spatial center, while mean looks the density center.

For unevenly distributed sensor data with minimal outliers, the centroid improves PASAD's performance by providing a more accurate center. However, if the data has outliers, the centroid can degrade performance since it may be influenced by these extreme values, making the mean a more reliable choice in such scenarios.

Implementation and Comparison:

We updated our definition of Central Tendency in the Pasad code and implemented it on each of the datasets to see the following Alarm Counts versus Attack Scenario Plot. We can draw the following conclusions upon comparing the performance of the new centroid.

1. The new central tendency computed generated more alarms for almost all the attack scenarios except the direct attack 2.
2. The increase in alarm counts is nearly 15% from the original Pasad implementation
3. The percentage of alarms of one dataset is nearly same.

The alarm Counts are 'SA1': 812, 'SA2': 1386, 'SA3': 2014, 'DA1': 718, 'DA2': 781

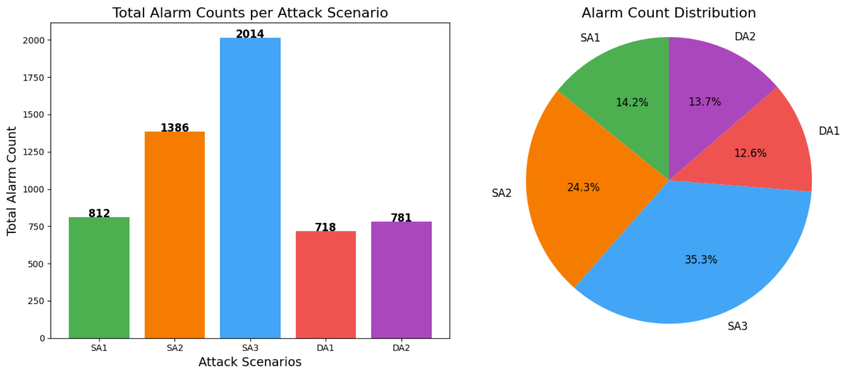

Figure 26: Plot of Alarm Counts and Corresponding Dataset for Spatial Center

Q3: Exploring the Use of Mahalanobis Distance

Justification:

Euclidean distance measures the straight-line distance between two points and is calculated using the formula

$$D_{\text{euclidean}} = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$$

or more generally

$$D_{\text{euclidean}} = \sqrt{(x - y)^T (x - y)}$$

in matrix form. It assumes that features are independent and equally scaled, making it a simple and intuitive distance metric. However, this assumption leads to inaccuracies when data points are correlated or exhibit different variances.

Mahalanobis distance, on the other hand, accounts for correlations and the variance within the data, making it more suitable for multivariate distributions. The formula for Mahalanobis distance is

$$D_{\text{mahalanobis}} = \sqrt{(x - \mu)^T \Sigma^{-1} (x - \mu)}$$

where Σ^{-1} is the inverse of the covariance matrix. This normalization allows Mahalanobis to handle features with different scales and interdependencies more effectively than Euclidean distance.

Euclidean distance works best on spherical data distributions, where the variance is similar across dimensions. Mahalanobis distance, however, captures elliptical data distributions, where the spread varies along different directions. While Mahalanobis is more accurate for complex, correlated data, it is computationally more demanding, especially for high-dimensional datasets, due to the need to compute the inverse of the covariance matrix.

In conclusion, Euclidean distance is simpler and works well for uncorrelated, evenly distributed data. However, Mahalanobis distance is superior when dealing with multivariate, correlated data, though its computation is more complex.

Implementation and Comparison:

We updated our definition of departure score to include mahabalonis distance, we computed the necessary covariance matrix, in the Pasad code and implemented it on each of the datasets to see the following Alarm Counts versus Attack Scenario Plot. The central tendency is still the mean, but similar conclusions can be drawn for centroid as well. We can draw the following conclusions upon comparing the performance of the new distance.

1. The new central tendency computed generated more alarms for almost all the attack scenarios except the direct attack 2.
2. The increase in alarm counts is nearly 15% from the original Pasad implementation
3. The percentage of alarms of one dataset is nearly same.

The alarm Counts are 'SA1': 790, 'SA2': 1134, 'SA3': 1795, 'DA1': 621, 'DA2': 783

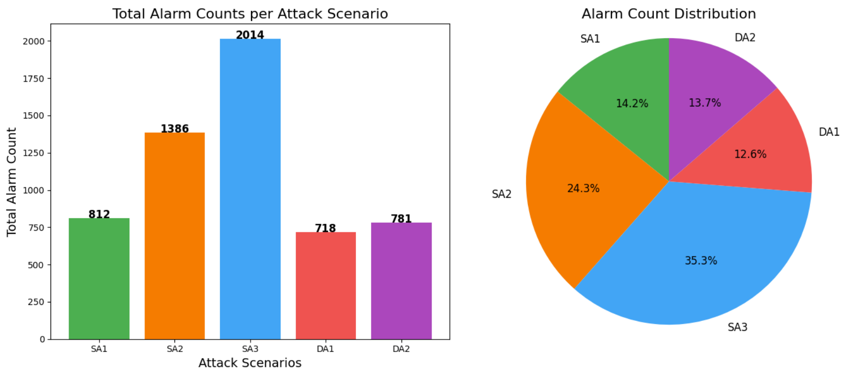

Figure 27: Plot of Alarm Counts and Corresponding Dataset for Mahabalonis Distance

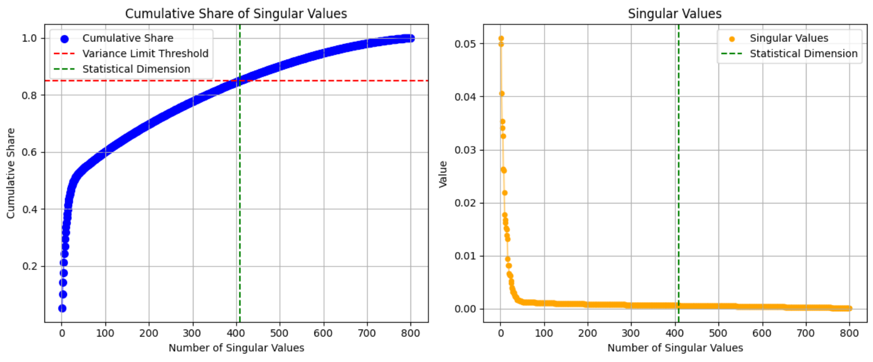

Figure 28: Scree Plot for SA1 Dataset

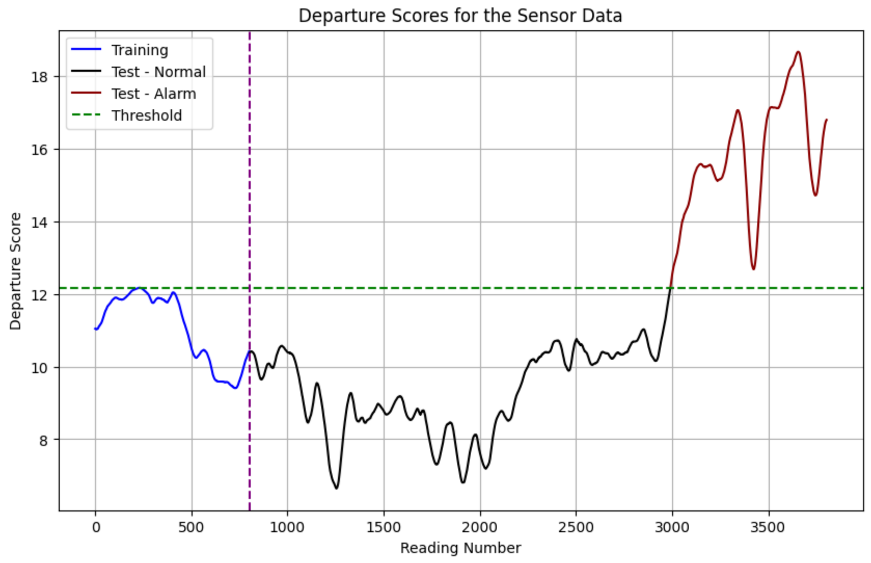

Figure 29: Departure Scores for SA1 Dataset

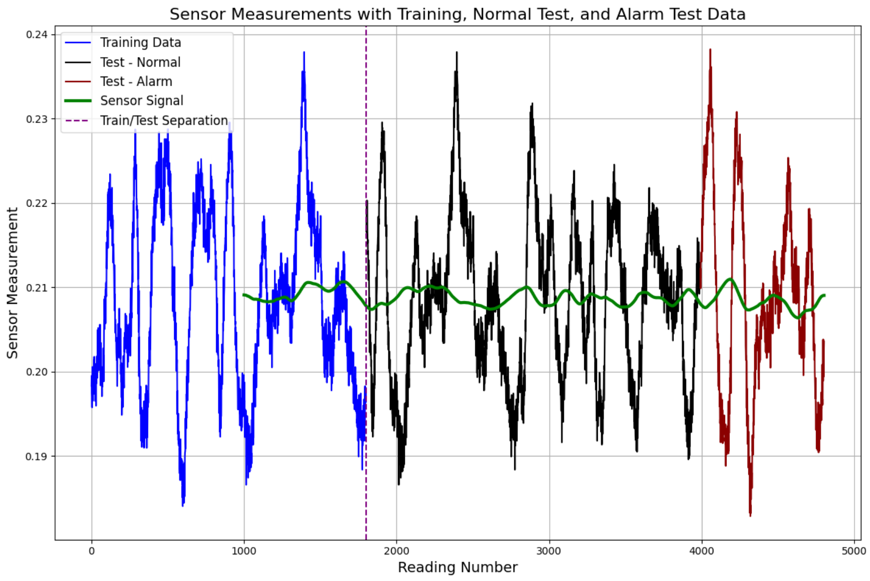

Figure 30: Sensor Readings for SA1 Dataset

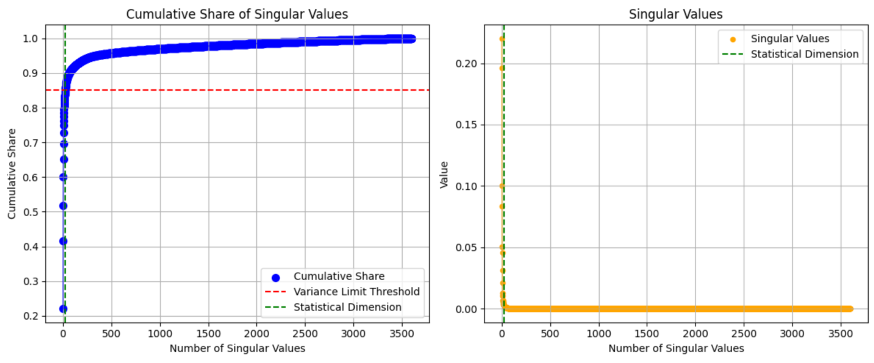

Figure 31: Scree Plot for SWaT Dataset

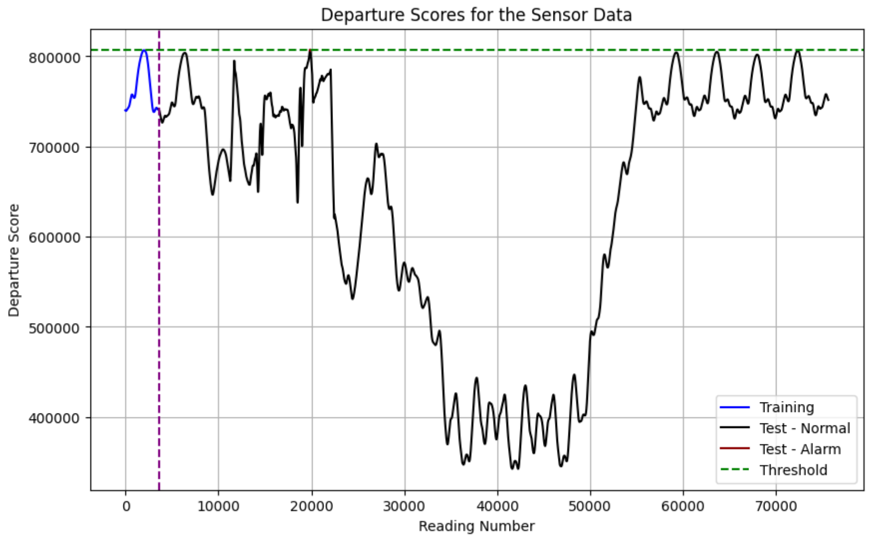

Figure 32: Departure Scores for SWaT Dataset

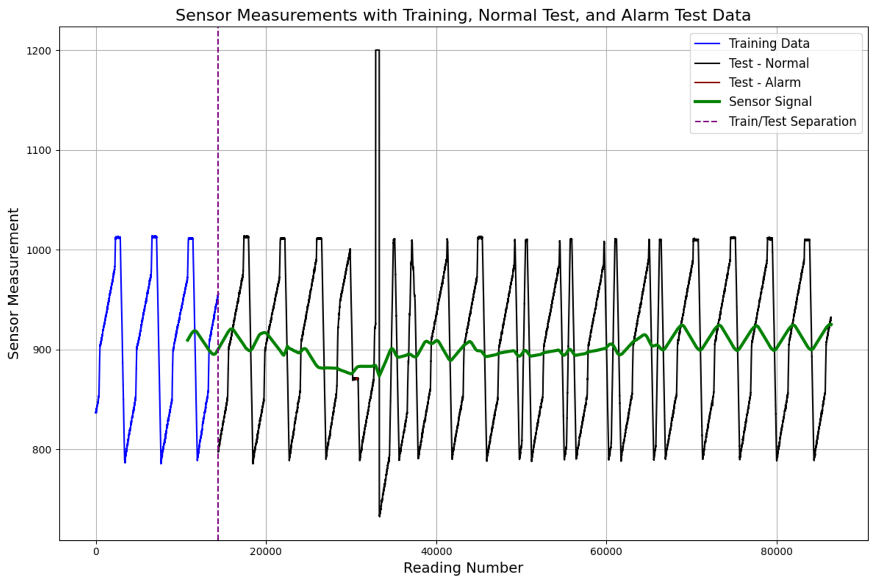

Figure 33: Sensor Readings for SWaT Dataset

Runtime Analysis:

For Mahalanobis distance, the total time taken for 3802 departure score computations is 0.9862837791442871 Making the average time 0.0002594118303903964.
For Euclidean distance, the total time taken for 3802 departure score computations is 0.6710443496704102 Making the average time 0.0001764977247949527
The Mahalanobis distance computation takes much more time than Euclidean distance.