👋 Introduction

**Automatic Data Recognition Project - Anomaly Detection using LSTM Neural Networks**

Daniel Hamama.

318652252.

Holon Institute of Technology.

Student of BSc Applied Mathematics.



**Introduction:**

This project aims to develop an anomaly detection model for a manufacturing process using time-series data from various sensors.

Using EDA and Clustering to get a better understanding of the dataset while applying Data Preprocessing to keep the data as clean as possible.

The focus is on applying Long Short-Term Memory (LSTM) neural networks, which are well-suited to capture temporal dependencies in sequential data and thus identify patterns indicative of faults or abnormalities.

This anomaly detection solution could support predictive maintenance, reduce equipment downtime, and optimize manufacturing processes by identifying and addressing issues before they escalate.

🎯 Objective

**Objective:**

To develop an LSTM-based model that accurately detects anomalies in manufacturing time-series data by learning from sequential patterns while dealing with the challenges posed by data imbalance.

This is done by first cleaning the data, understanding the data using EDA and Clustering, then finally detecting anomalies using the LSTM model and assessing performance with classification metrics and MSE on both training and test data.

⚙️ Methods

**Methods:**

* **Data Preprocessing**: Data was preprocessed by dropping irrelevant features, checking for null values, reducing RAM usage, checking and changing data types of certain features and Label Encoding.
* **Exploratory Data Analysis:** EDA was done in order to get a better understanding of the data. It was achieved with visualizations, temporal graphs and highlighting correlations in the data.
* **Clustering:** Unsupervised clustering was applied to group data points and identify patterns within the dataset.
* **Model Architecture**: A multi-layer LSTM model was constructed with two LSTM layers of varying units, followed by dropout layers to mitigate overfitting. An L2 regularization was applied to the LSTM and Dense layers to further prevent overfitting, and a Dense layer with sigmoid activation was used for binary classification.
* **Class Imbalance Handling**: Given the imbalanced nature of the data, class weights were adjusted to give higher importance to anomalies during training.
* **Evaluation**: The model was evaluated using both Mean Squared Error (MSE) and classification metrics (accuracy, precision, recall, F1-score) in order to get a detailed view of its effectiveness.

📋 Data Description

**Data Description:**

The Training Dataset comprises sensor readings and operational settings from various stages of an industrial ion beam etching process. Each sample represents a time-series measurement collected from different sensors and equipment settings within the process. The dataset consists of 27 parameters, including both categorical identifiers and numeric sensor readings and 3,716,092 data points where only 13,693 are initially flagged as anomalies.

**Features:**

* **S1 - time**: Time of measurement
* **S2 - Tool**: Tool ID (categorical)
* **S3 - stage**: Processing stage of the wafer (categorical)
* **S4 - Lot**: Wafer lot ID (categorical)
* **S5 - runnum**: Tool run count
* **S6 - recipe**: Tool settings for processing (categorical)
* **S7 - recipe\_step**: Recipe step in the process (categorical)
* **S8 - IONGAUGEPRESSURE**: Pressure in the main chamber (sensor)
* **S9 - ETCHBEAMVOLTAGE**: Voltage on the beam plate
* **S10 - ETCHBEAMCURRENT**: Ion current on the beam grid
* **S11 - ETCHSUPPRESSORVOLTAGE**: Voltage on the suppressor plate
* **S12 - ETCHSUPPRESSORCURRENT**: Ion current on the suppressor plate (sensor)
* **S13 - FLOWCOOLFLOWRATE**: Helium flow rate in the flowcool circuit
* **S14 - FLOWCOOLPRESSURE**: Helium pressure in the flowcool circuit (sensor)
* **S15 - ETCHGASCHANNEL1READBACK**: Argon flow rate in source assembly
* **S16 - ETCHPBNGASREADBACK**: Argon flow rate in PBN assembly
* **S17 - FIXTURETILTANGLE**: Wafer tilt angle setting
* **S18 - ROTATIONSPEED**: Wafer rotation speed setting
* **S19 - ACTUALROTATIONANGLE**: Measured wafer rotation angle (sensor)
* **S20 - FIXTURESHUTTERPOSITION**: Shutter setting for wafer shielding
* **S21 - ETCHSOURCEUSAGE**: Grid assembly usage counter
* **S22 - ETCHAUXSOURCETIMER**: Chamber shields usage counter
* **S23 - ETCHAUX2SOURCETIMER**: Chamber shields auxiliary usage counter
* **S24 - ACTUALSTEPDURATION**: Step duration for a particular step (sensor)
* **S25 - FlowCool Pressure Dropped Below Limit:**  Indication for when the pressure is too low (flag).
* **S26 - FlowCool Pressure Too High Check Flowcool Pump:** Indication for when pressure is too high (flag).
* **S27 - Flowcool leak**: Indication for when there’s a leak in Flowcool (flag).

The Testing Dataset consists of 1,270,704 data points with 24 features, all of which are the same as in the training data minus the last 3 label features.

🛠️ Data Preprocessing

**Data Preprocessing:**

The data preprocessing phase was very important in preparing the dataset for clustering and for the LSTM model. Given the time-series nature of the data and the presence of class imbalance, the following preprocessing steps were applied:

**Data Cleaning and Feature Selection**:

* **Missing Values**: Any missing values in the dataset were handled through imputation or exclusion.
* **Feature Selection**: Based on domain knowledge and initial exploration, specific features (such as Flowcool Pressure) were selected as key indicators of anomalies, while irrelevant or redundant columns were excluded.

**Scaling**:

* **Min-Max Scaling**: All numeric sensor readings were scaled to a range between 0 and 1 using MinMaxScaler.
* **Categorical Encoding**: Categorical columns such as Tool, stage, recipe, etc., were converted to numeric formats where necessary. This encoding allowed the model to interpret these identifiers without adding bias.

**Time-Windowing (Sequence Generation)**:

* **Sliding Window Approach**: To capture temporal dependencies in the data, a sliding window technique was used to create sequences of 30 timesteps for each feature. Each sequence represents 30 consecutive data points, allowing the LSTM model to learn patterns over time.

**Class Imbalance Handling**:

* **Class Weights**: Given that anomalies are rare compared to normal observations, class weights were assigned to penalize misclassification of the minority class (anomalies). This approach ensures that the model learns to recognize fault conditions effectively without being biased towards the majority class.

**Train-Test Split**:

* **Training and Validation Sets**: The dataset was divided into training and validation sets, with 80% used for training and 20% for validation. This split allowed for model evaluation on unseen data and helped in tuning model parameters.
* **Test Set Preparation**: The test set, which consists mainly of normal data, was preprocessed similarly but without target labels. This setup allows for model evaluation on real-world, unlabeled data to assess its anomaly detection capabilities.

📈 EDA

**Exploratory Data Analysis:**

The purpose of the EDA step is to gain initial insights into the dataset, understand feature distributions, identify any correlations, and uncover potential patterns or anomalies that might affect model performance.

This section includes analyses of feature distributions, labeled fault count, target variable (Flowcool Pressure) analysis and correlations.

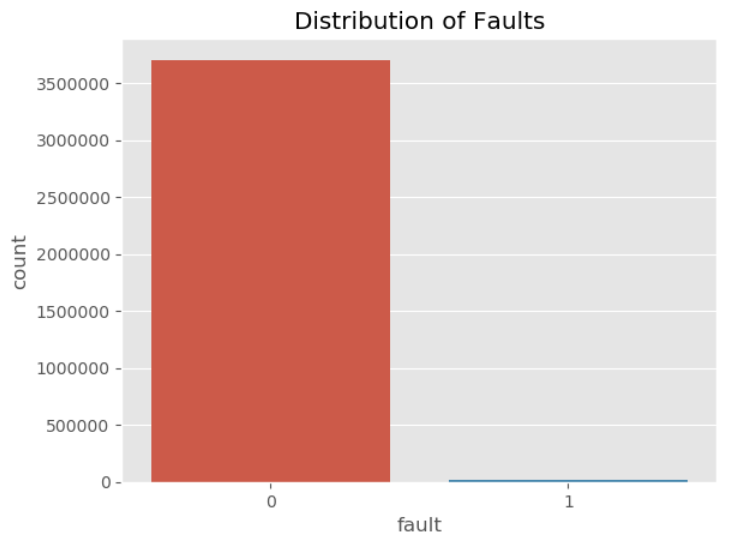
We are going to use visualizations such as barplots, boxplots, distribution plots, heatmaps and graphs in order to get a better grasp of how the data behaves.

I did not include descriptive statistics as it is too confusing when applying to the entire dataset, later in the clustering phase the statistics are easier to understand.

Fault Distribution

**Fault Label Distribution:**

Using the ‘countplot’ method of the seaborn library we can see that there exists a major imbalance between the data flagged as anomaly and the normal data.

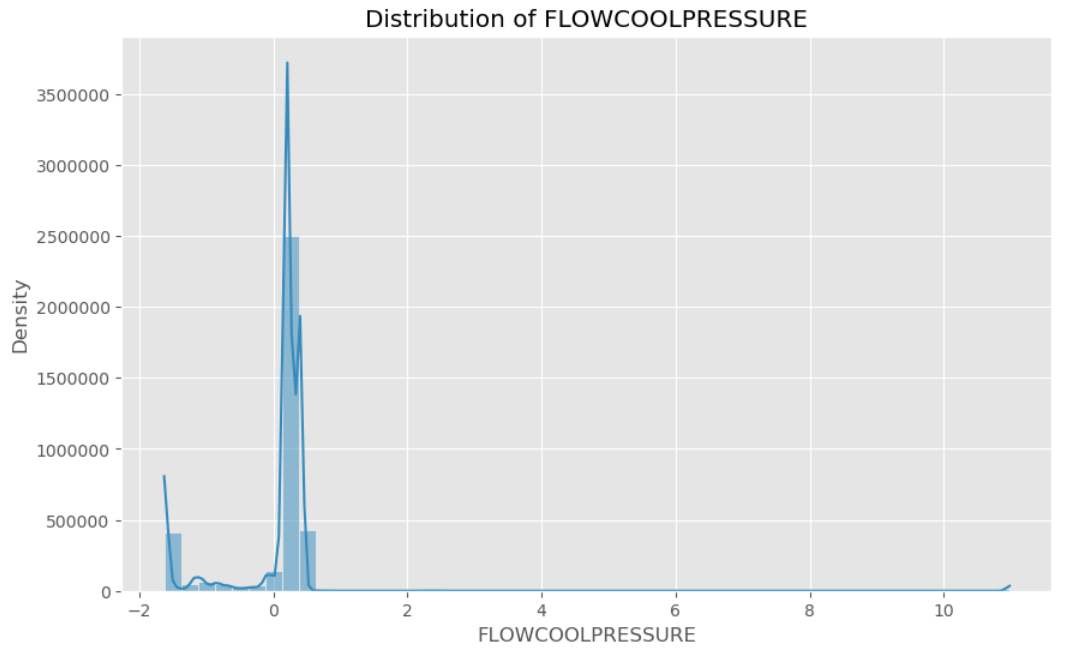


The count for fault data is 13,693 in this case.

Target Variable Distribution

**Target Variable Distribution:**

Using the ‘histplot’ method of the seaborn library we can clearly see how our target variable data is distributed across the dataset and its density in certain values.

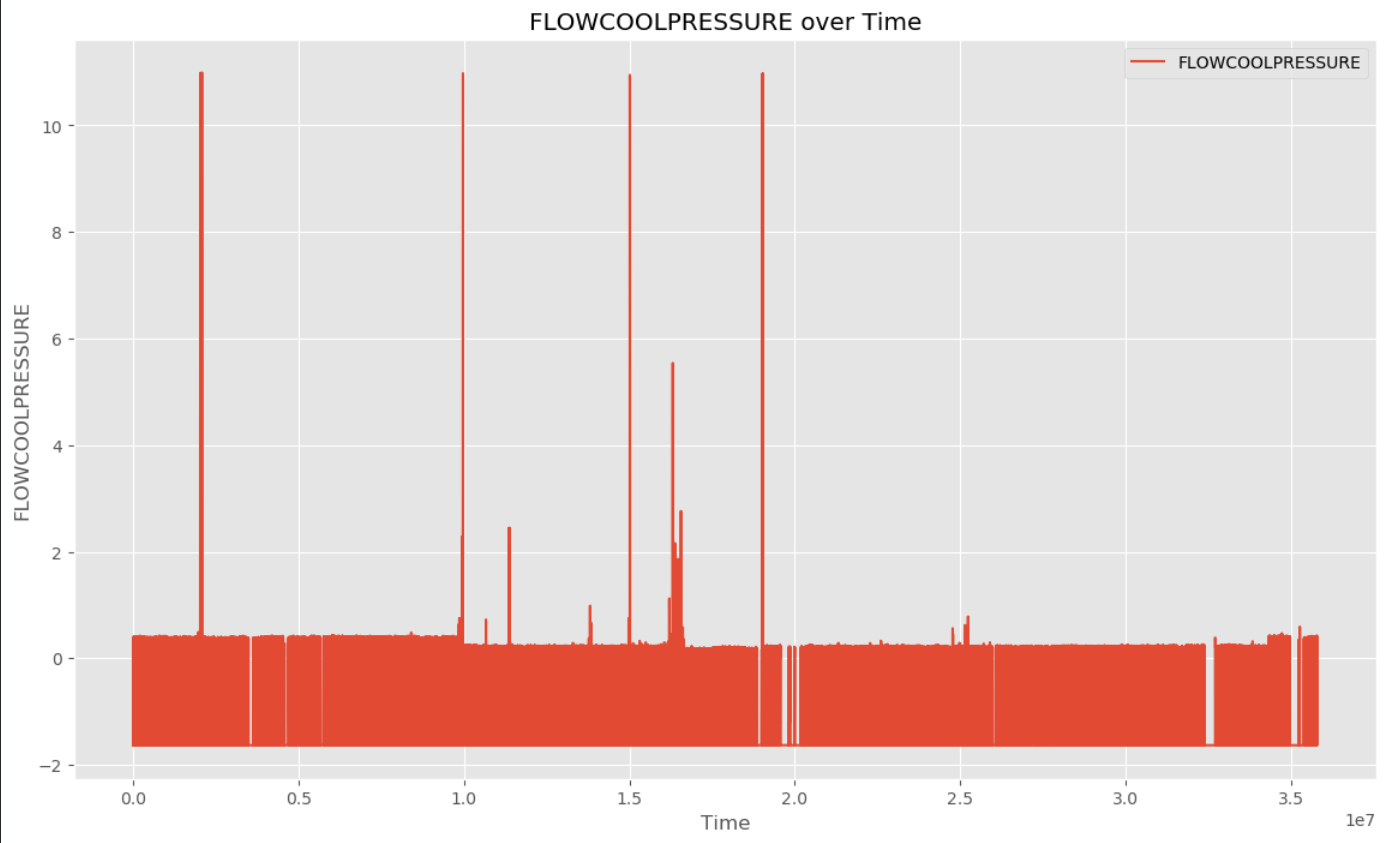


As we can see there is a high concentration of the target variable around the 0 value.

Target Variable Over Time

**Target Variable Over Time:**

Using the ‘plot’ method of the ‘matplotlib’ library we can see a simple graph that shows that the target variable behaves differently in certain points across time.

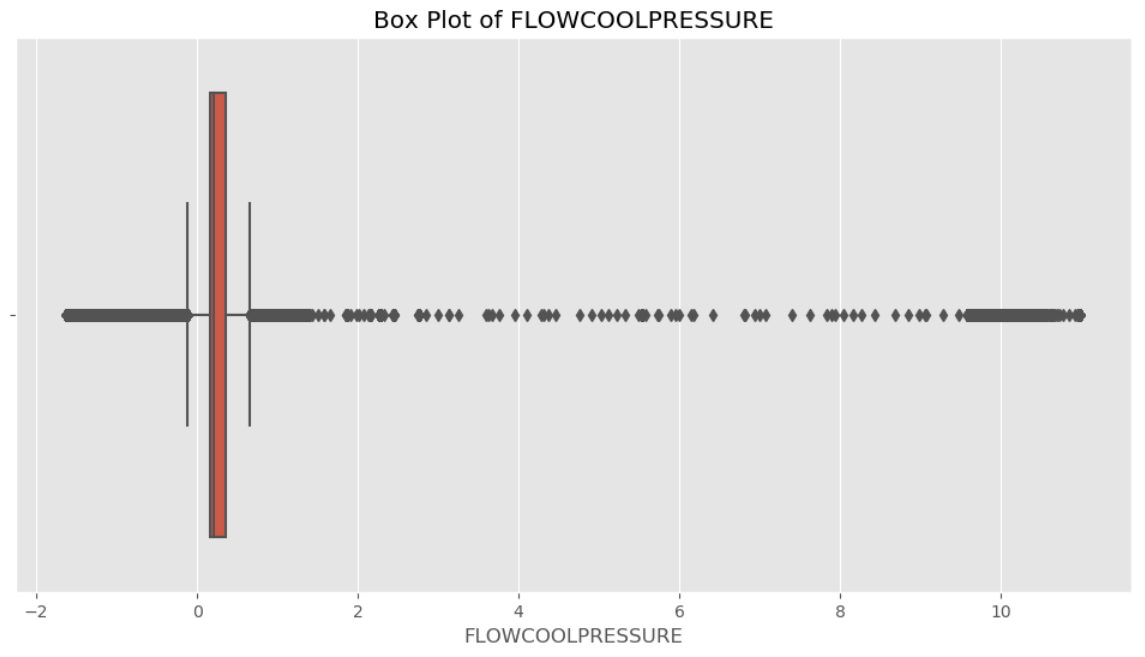


In some points in time the value spikes and in others the value sinks. This indicates that at a certain point in time something happened that affected the target variable and made it behave differently thus making it an anomaly.

Outliers in Target Variable

**Outliers in Target Variable:**

Here we use the ‘boxplot’ method of the ‘seaborn’ library in order to see how the data behaves statistically.



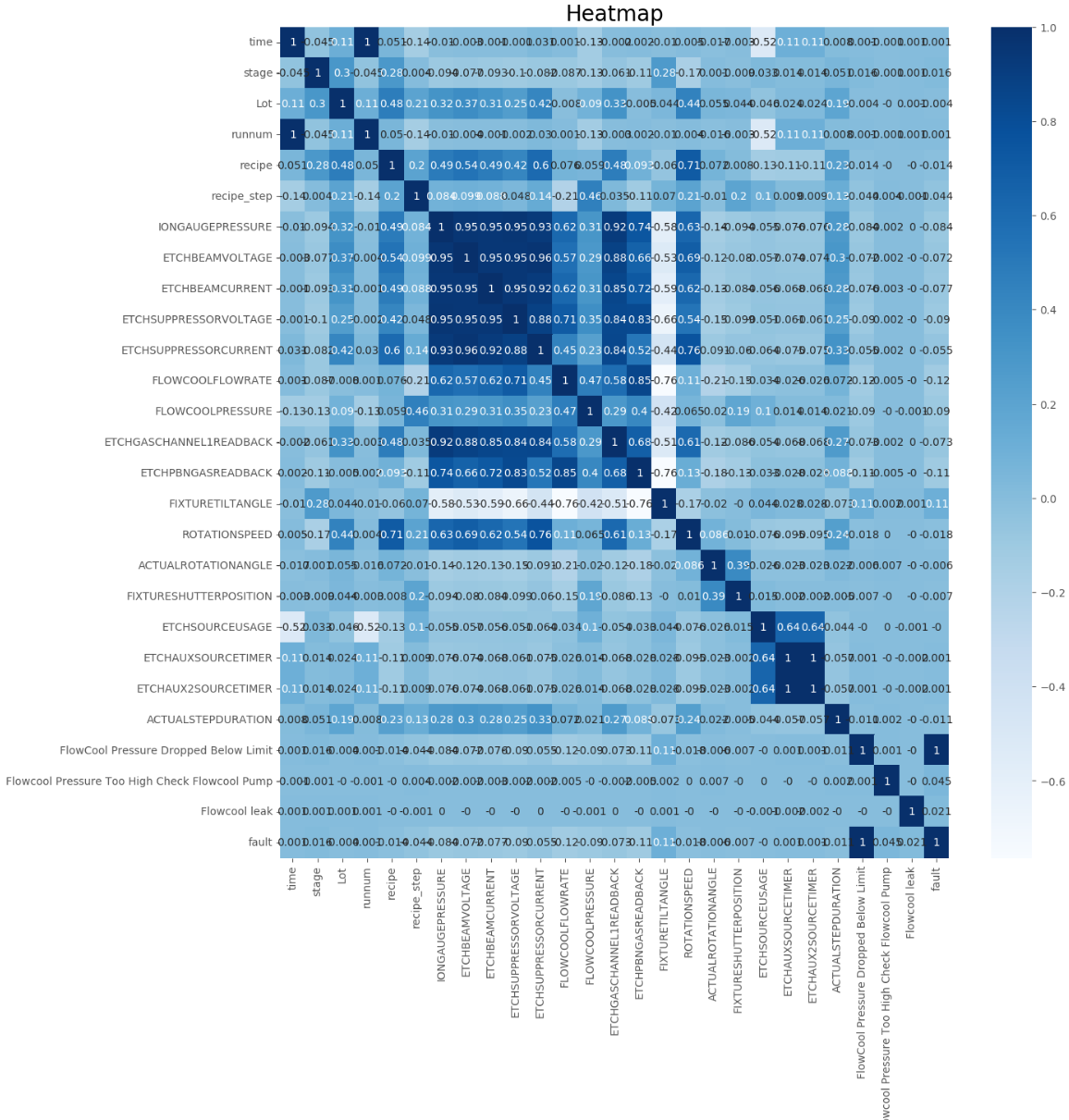
As we can see most of the data is concentrated above the 0 value with the upper and lower thresholds not far from it.

We can also see many outliers in the target variable, which probably indicates the anomalies in the data.

Correlation Heatmap

**Correlation Heatmap:**

Here we use the ‘heatmap’ method in order to create a heatmap of how the features of the data correlate with each other.



As we can see some features are highly correlated with each other such as IONGAUGEPRESSURE with ETCHBEAMVOLTAGE, a correlation of 0.95.

Now we can also see here how the features correlates with the target variable:



This might give us important information we can use later on in clustering and anomaly detection.

📊 Clustering

**Clustering:**

The clustering step aims to identify natural groupings in the dataset, which may reflect different operational states or machine conditions that could relate to anomalies.

Furthermore while the data is grouped into different clusters it is easier to understand due to the smaller scale and it’s simpler to see how the data behaves in the different clusters.

While we wanted to use clustering in order to help us with anomaly detection, it came to mind that a better approach may be more suitable. Therefore the clustering process will serve as an exploratory tool to help us get a better understanding of the data.

In this section we will go over the methods used, algorithms chosen, cluster interpretation, statistics, visualizations and findings.

Feature Selection

**Feature Selection:**

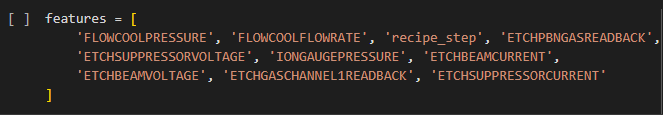
Earlier in the EDA section we checked for correlation with the target variable **‘*FLOWCOOLPRESSURE*’,** a known **anomaly** indicator.

In this section we will be using the highest correlated variables in order to get a better understanding on what affects the target variable and minimize unnecessary noise in the data and thus keeping the results as ‘pure’ as possible.

The selected features are the following:

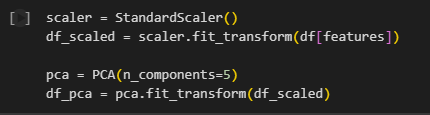
* **FLOWCOOLPRESSURE**
* **FLOWCOOLRATE**
* **recipe\_step**
* **ETCHPBNGASREADBACK**
* **ETCHSUPPRESSORVOLTAGE**
* **IONGAUGEPRESSURE**
* **ETCHBEAMCURRENT**
* **ETCHBEAMVOLTAGE**
* **ETCHGASCHANNEL1READBACK**
* **ETCHSUPPRESSORCURRENT**

This selection ensures that the clustering analysis captures meaningful patterns related to anomaly detection while reducing dimensionality.



Scaling and PCA

**Scaling and PCA:**

****

**Standardization**:

Each feature was standardized using StandardScaler to achieve a mean of 0 and standard deviation of 1. Standardization prevents features with larger scales from disproportionately influencing the clustering algorithm.

**Principal Component Analysis (PCA)**:

**PCA** was applied to reduce dimensionality, capturing the primary variance in the dataset with fewer components. This improves computational efficiency and reduces noise.

Five principal components were retained, accounting for the most relevant variance while preserving interpretability.

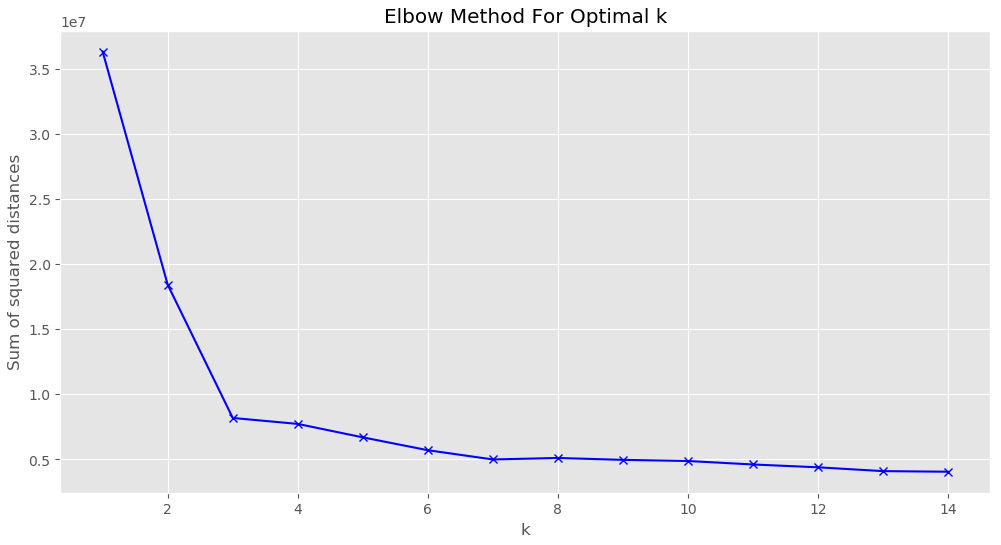
The reduced dataset **simplifies** the clustering process and focuses it on the most informative features, facilitating clearer cluster separation.

Elbow Method

**Determining the Optimal Number of Clusters (Elbow Method):**

The elbow method was used to identify the optimal number of clusters by evaluating the within-cluster sum of squares (WCSS) for various cluster counts.

WCSS values were plotted against the number of clusters. The optimal number was chosen based on the “elbow” point in the graph, where adding more clusters yields diminishing returns in reducing WCSS.

****

As we can see there's a major change at point 3 and a slight change at point 4.

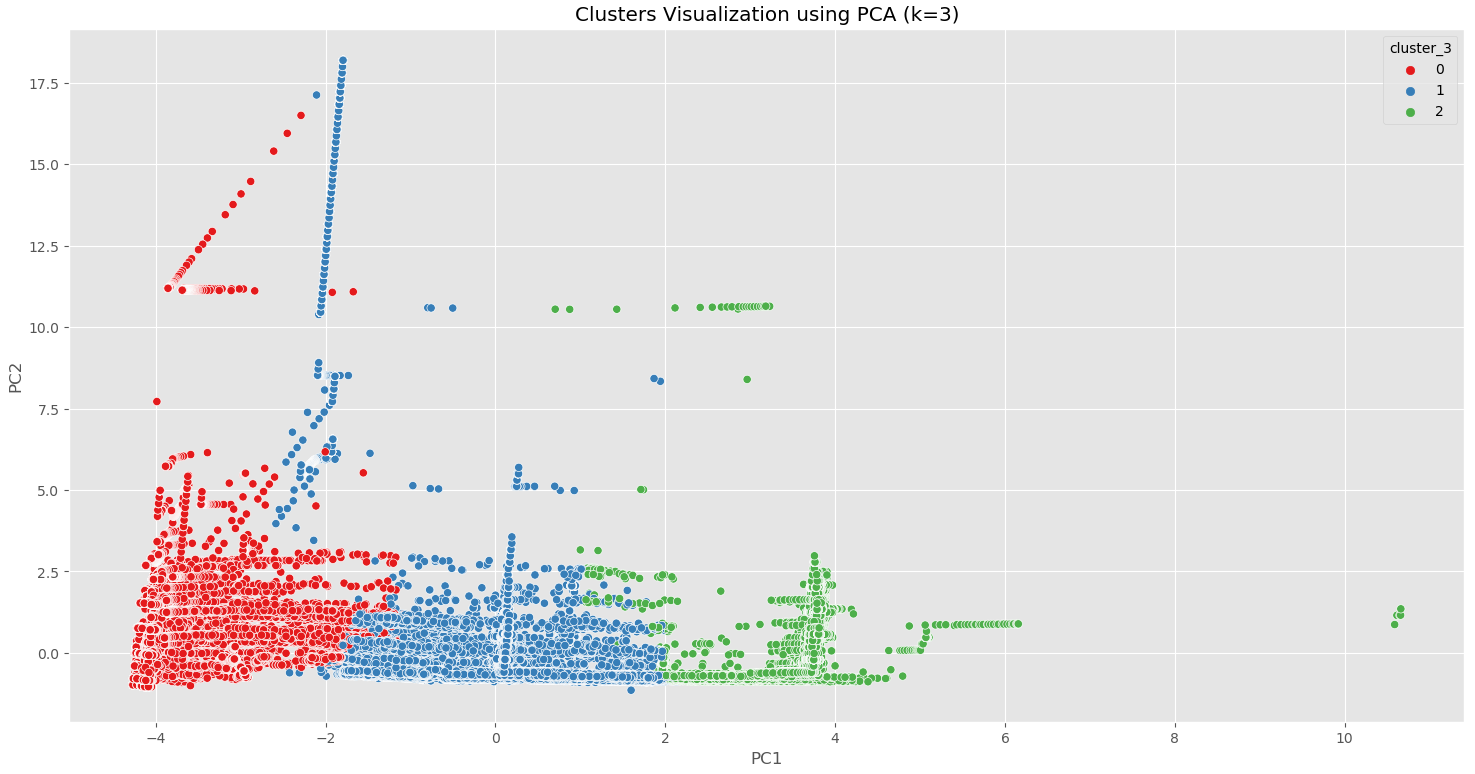
Next we analyze both points in order to see which one yields better results using the MiniBatchKMeans algorithm.

**Comparison of Clustering with k=3 and k=4:**

After narrowing down the cluster count options, clustering was performed with both k=3 and k=4 to assess how each configuration influences the clustering quality and interpretability.

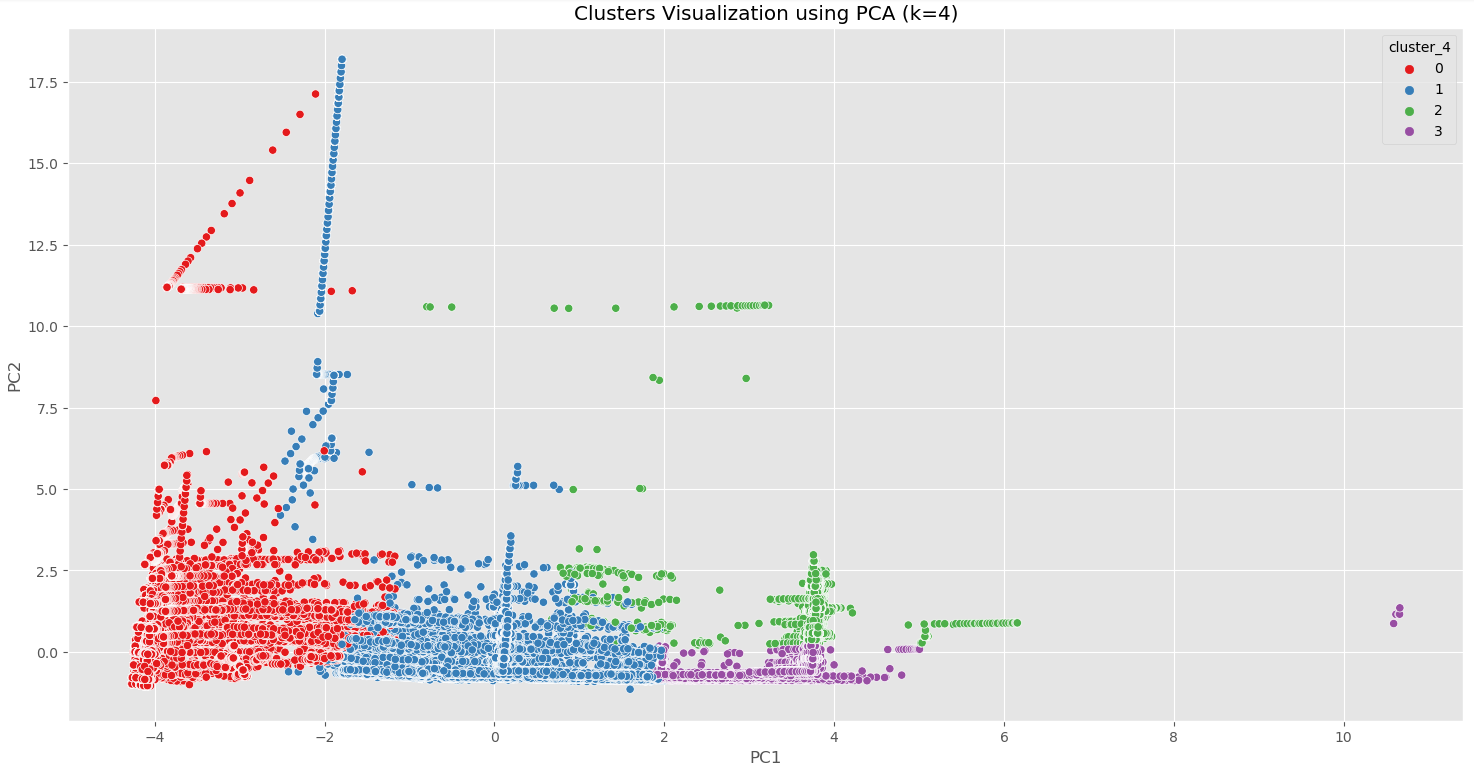
Scatter plots of PCA components for both k=3 and k=4 configurations were generated, illustrating how clusters are distributed in the reduced dimensional space. In these plots:

***PCA projection of clusters for k=3:***



**k=3 Plot**: Shows larger, more distinct clusters with broader definitions, making it simpler to interpret each cluster’s meaning.

***PCA projection of clusters for k=4:***



**k=4 Plot**: Displays smaller, more nuanced clusters, which might capture additional variations but may also risk overlap between clusters.

This provides a clear comparison between the two configurations.

From here on out we are working with **K = 3** for the rest of the clustering process.

Feature Importance

**Feature Importance**

#### **Introduction to Feature Importance:**

Feature importance was analyzed to determine which variables most significantly influenced the clustering process. By understanding these relationships, we could gain insights into the dominant factors that define cluster characteristics and behavior.

**Methodology:**

The analysis was performed using the following features. The features were chosen for their correlation value with the presumed target variable.

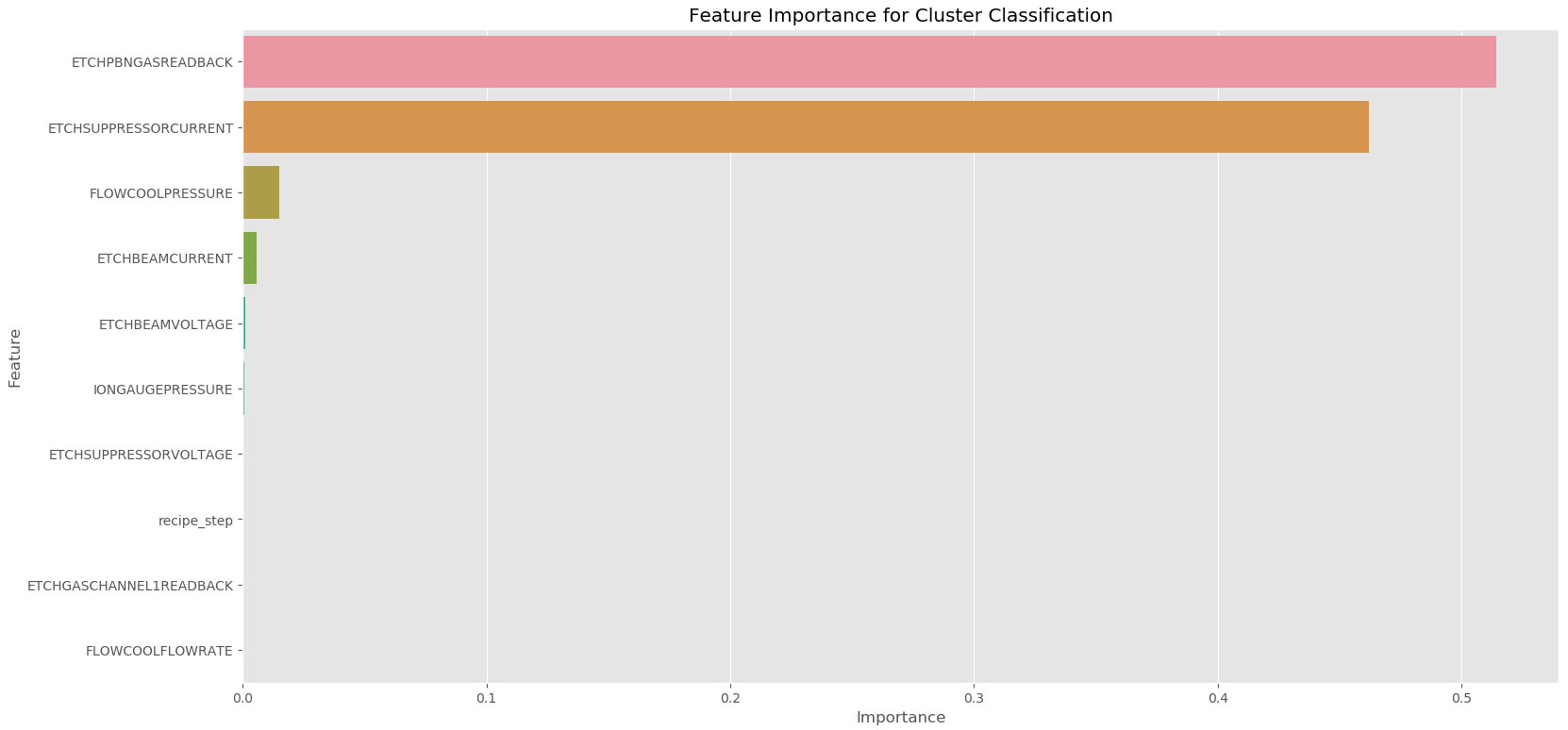
Features used:

* FLOWCOOLPRESSURE
* FLOWCOOLRATE
* Recipe\_step
* ETCHPBNGASREADBACK
* ETCHSUPPRESSORVOLTAGE
* IONGAUGEPRESSURE
* ETCHBEAMCURRENT
* ETCHBEAMVOLTAGE
* ETCHGASCHANNEL1READBACK
* ETCHSUPPRESSORCURRENT

**Classification Model:**

A Decision Tree Classifier was used to evaluate feature importance by measuring the reduction in impurity caused by each feature split.

**Visualization:**A bar plot was created to illustrate the importance of each feature.



#### **Results:**

The top three features influencing clustering were:

1. **ETCHPBNGASREADBACK (Importance: 0.514159)**This feature contributed over half of the clustering decision influence, highlighting its significance in differentiating clusters.
2. **ETCHSUPPRESSORCURRENT (Importance: 0.462160)**The second most critical feature for defining clusters.
3. **FLOWCOOLPRESSURE (Importance: 0.014880)**While less influential than the first two, this feature also played a notable role.

The results show that the top features vary significantly in their importance levels. Most of the impact was dominated by a small subset of features, while others played a relatively minor role in determining clusters.

Comparative Analysis

**Comparative Analysis and Cluster Analysis:**

In this section we will go over the top 3 features that had the most impact on the clustering process, **ETCHPBNGASREADBACK, ETCHSUPPRESSORCURRENT** and **FLOWCOOLPRESSURE.**

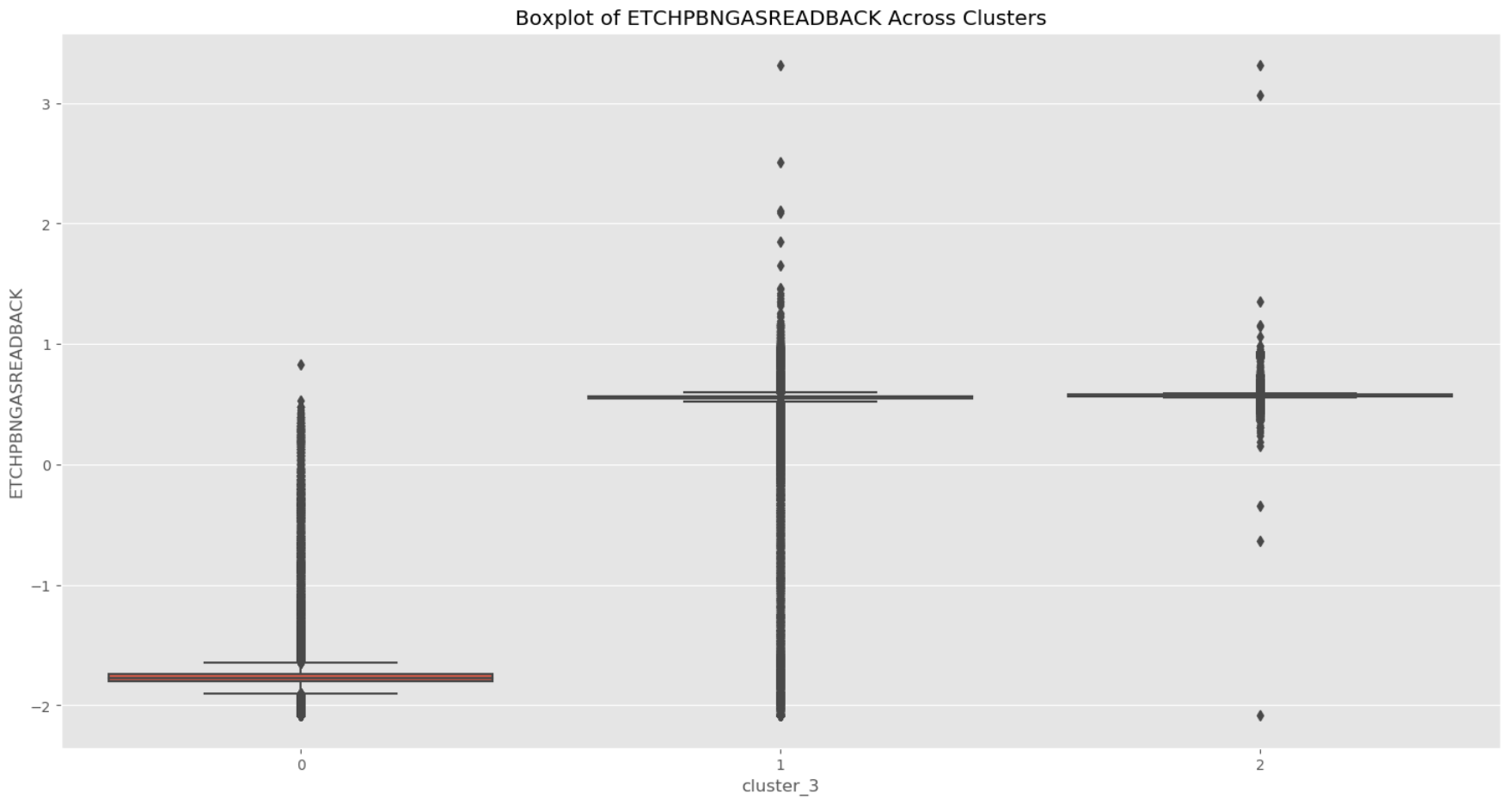
We will perform a comparative analysis of the features in the different clusters and learn how they differ from each other.

Furthermore we will analyze the clusters and their characteristics.

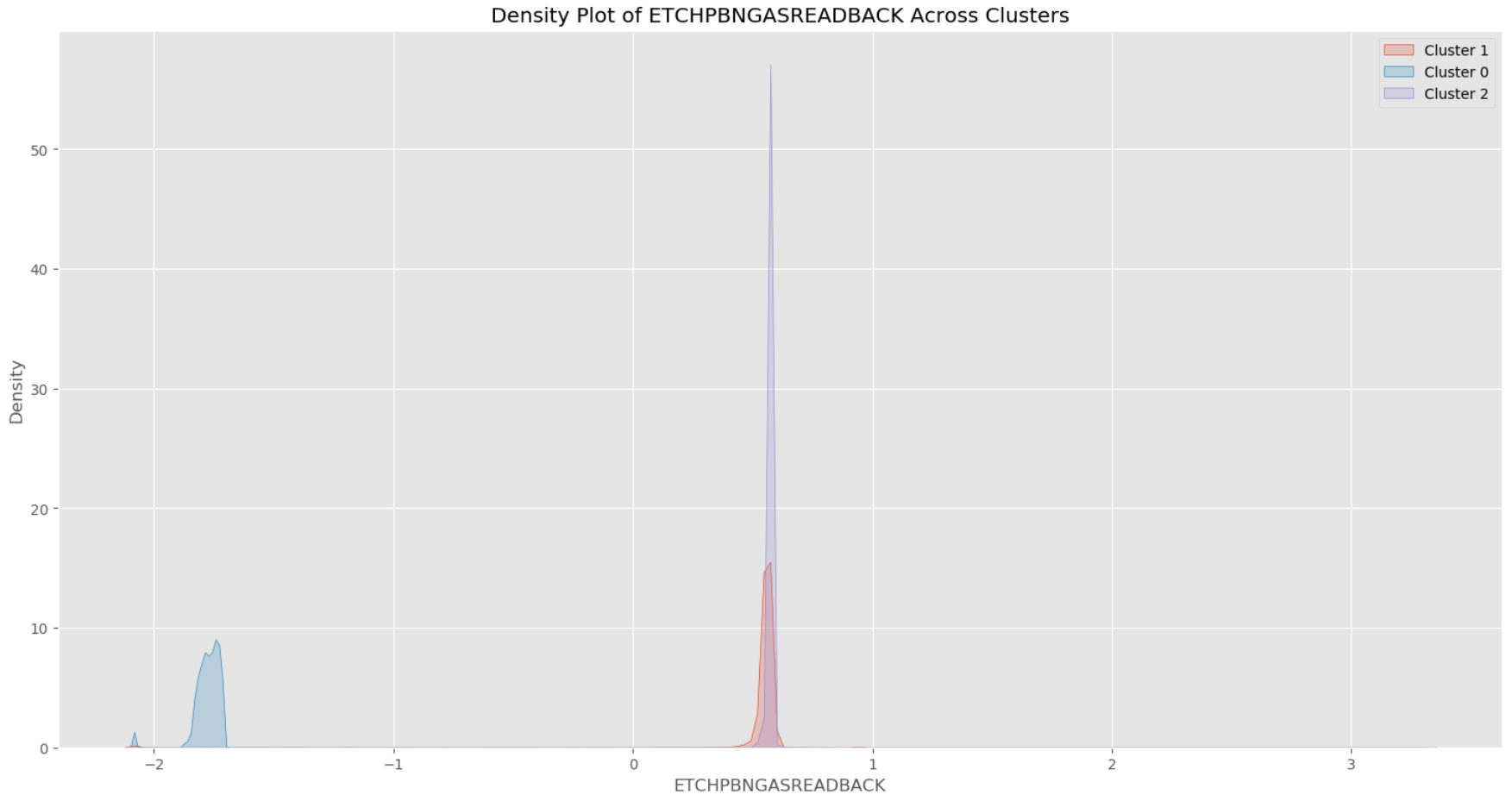
The comparisons are made through boxplots, density plots, barplots and graphs over time.

**ETCHPBNGASREADBACK** -

**Boxplot**:

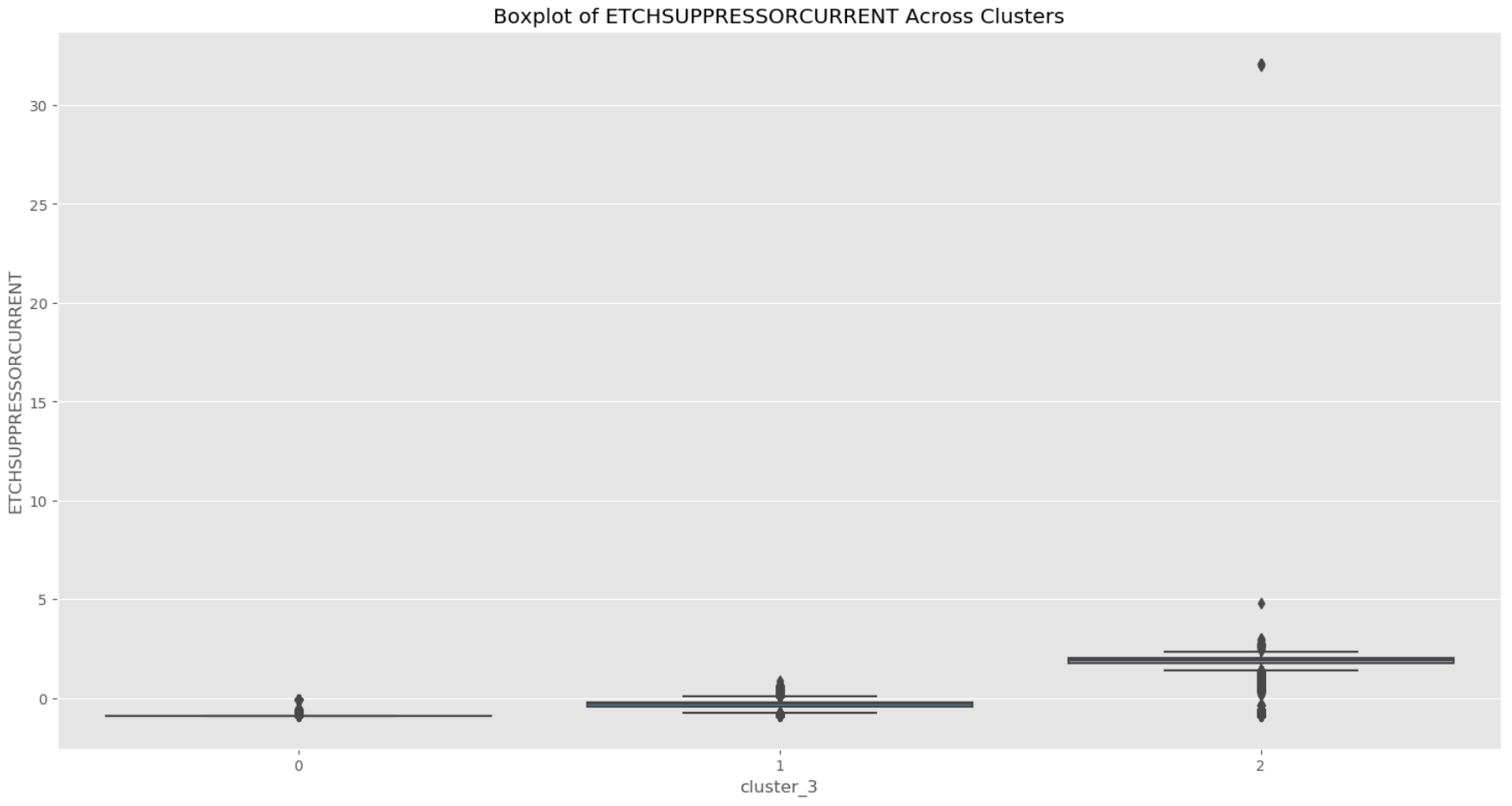


**Density plot:**

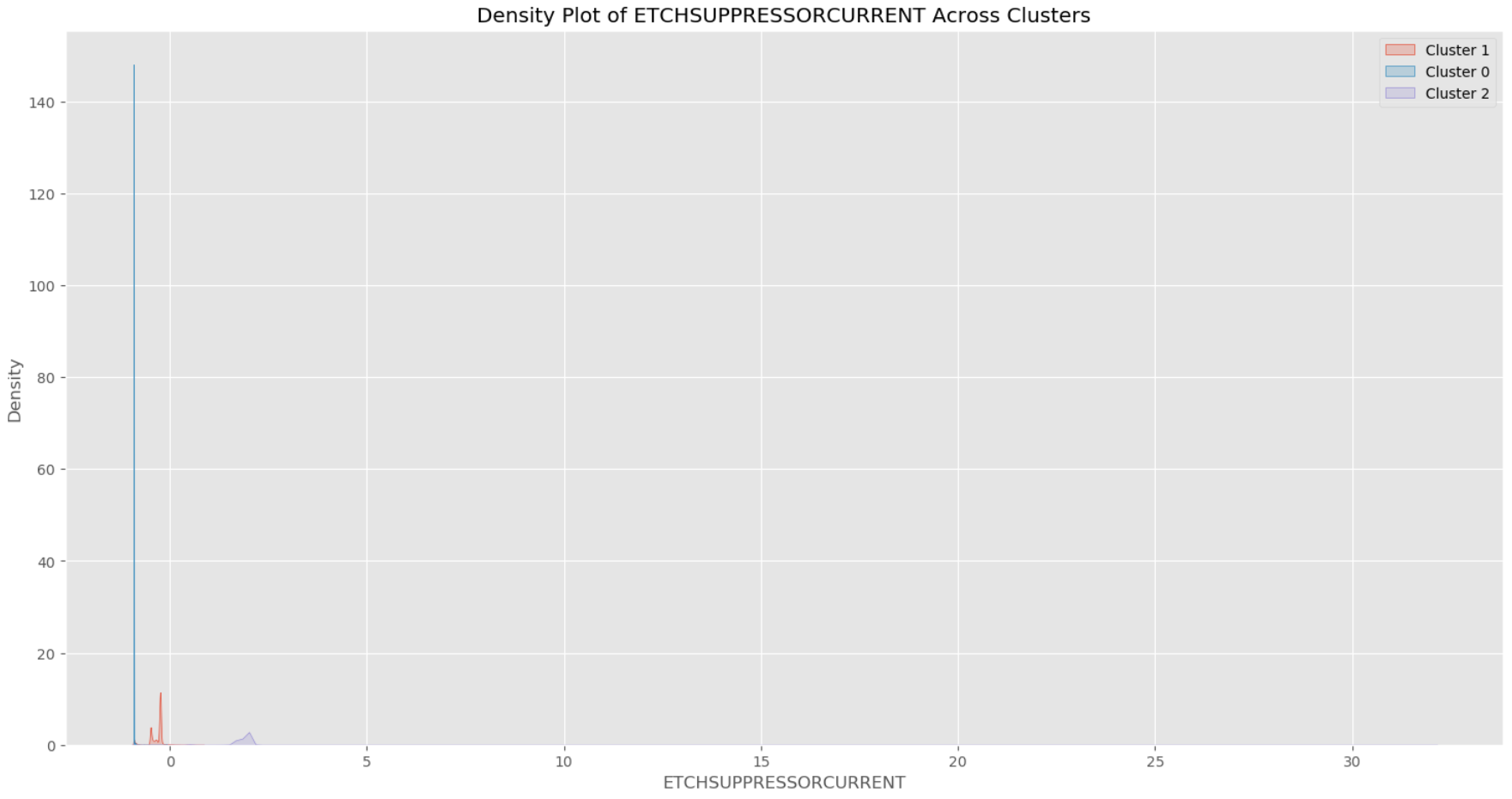


**ETCHSUPPRESSORCURRENT -**

**Boxplot:**

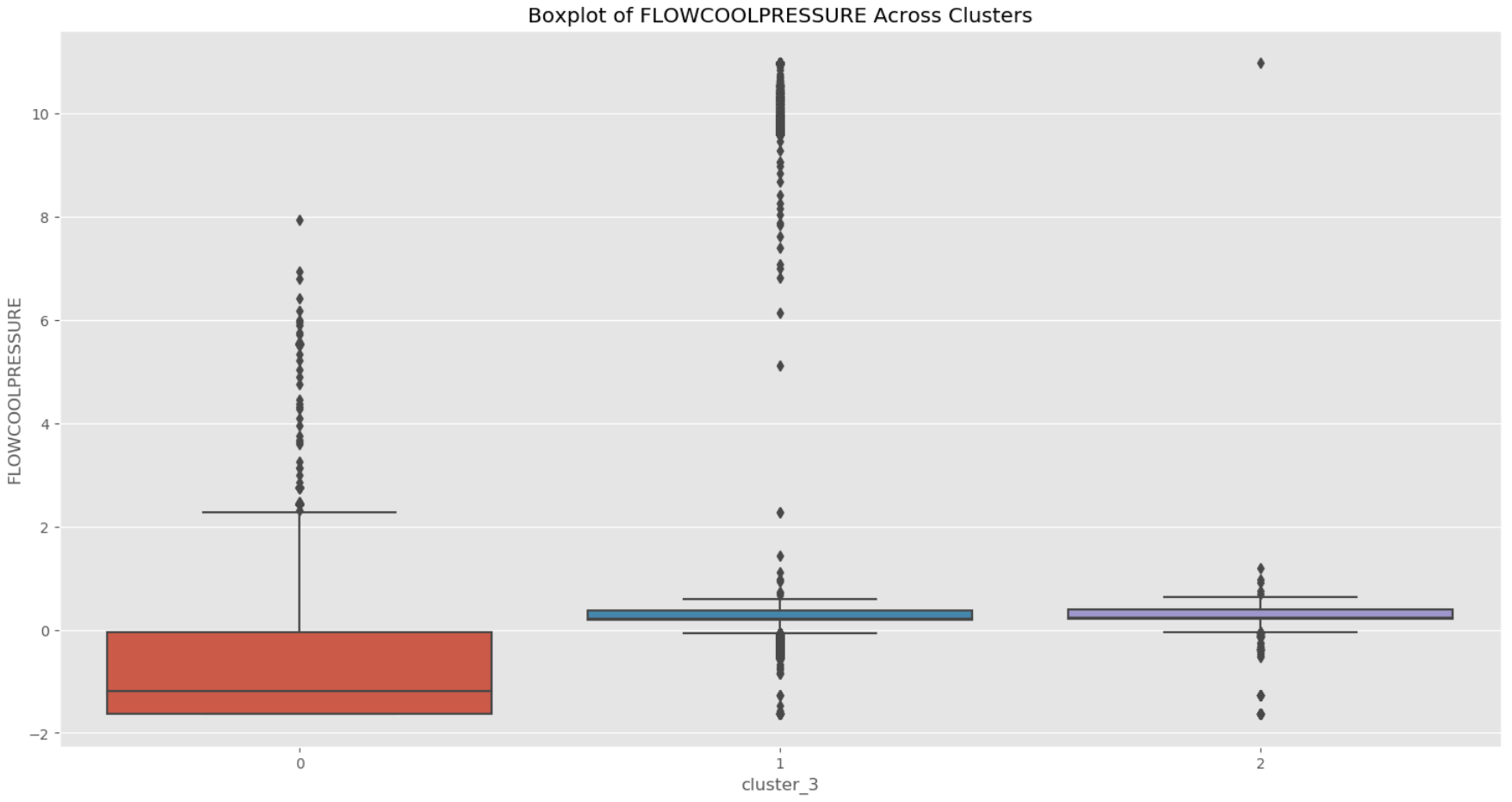
****

**Density plot:**

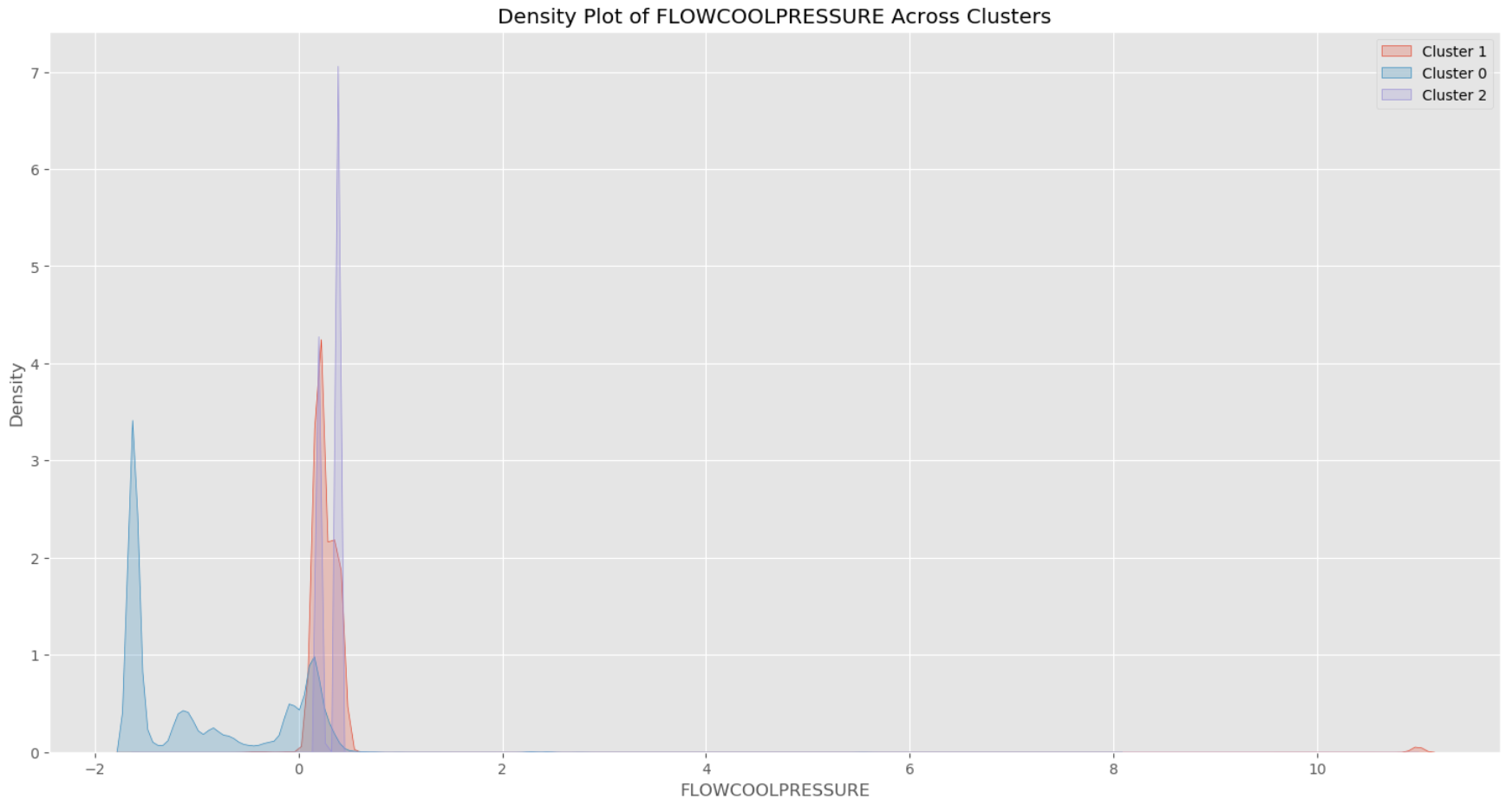
****

**FLOWCOOLPRESSURE -**

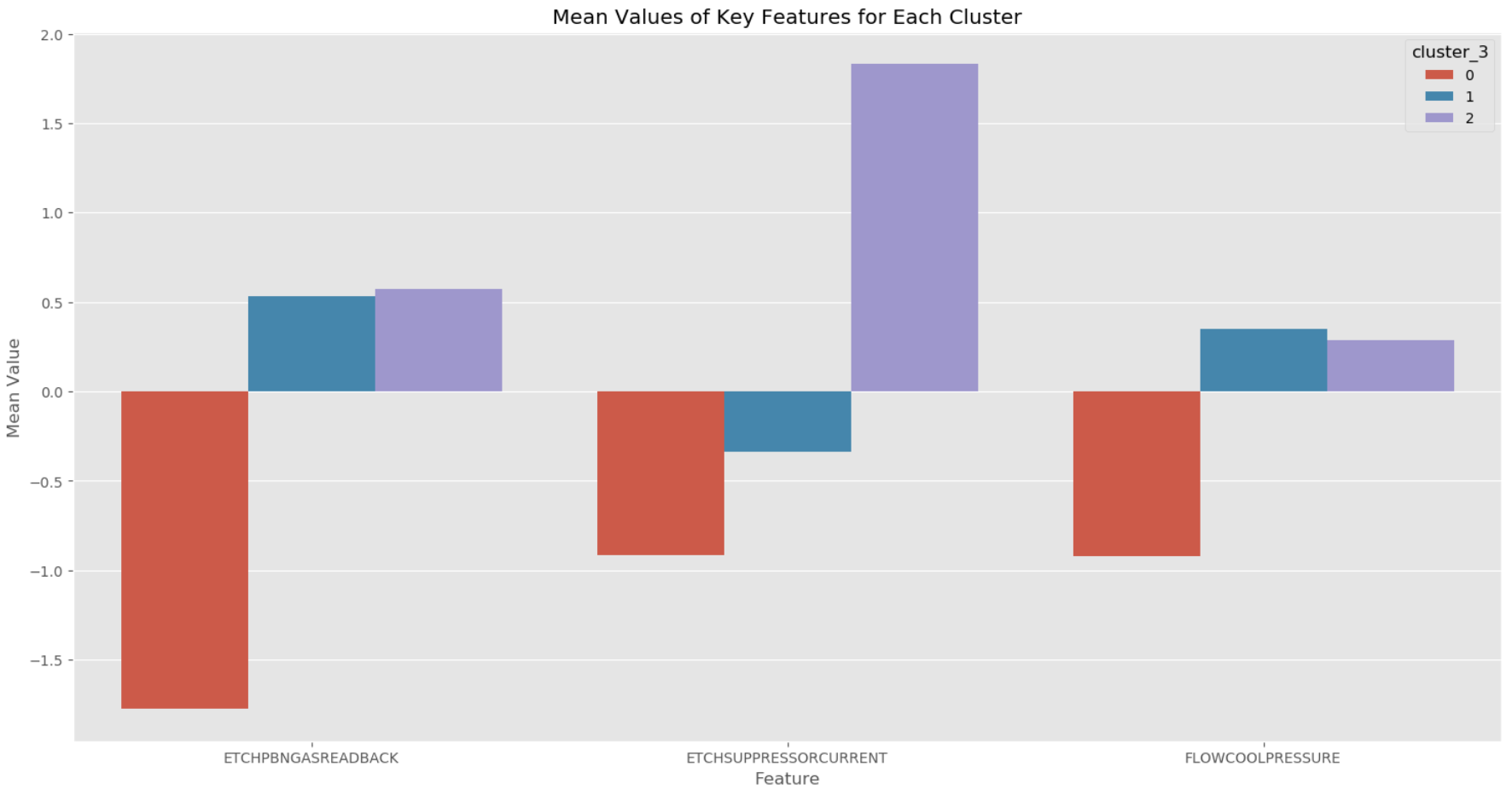
**Boxplot:**

****

**Density plot:**

****

Here’s a **barplot** of the Mean value of the top features in the different clusters:



**Analysis Summary:**

**Cluster 0:**

* **ETCHPBNGASREADBACK** has consistently low values. With most of the data points clustered around -2.
* **ETCHSUPPRESSORCURRENT** is tightly grouped around -0.9, showing low variability.
* **FLOWCOOLPRESSURE** has mostly negative values, indicating lower pressures for this cluster.

**Cluster 1:**

* **ETCHPBNGASREADBACK** shows a tighter grouping around positive values with some outliers.
* **ETCHSUPPRESSORCURRENT** shows a wider range of variability with values spread between -0.9 and 0.8.
* **FLOWCOOLPRESSURE** has a moderate range, with values generally close to zero but with significant outliers.

**Cluster 2:**

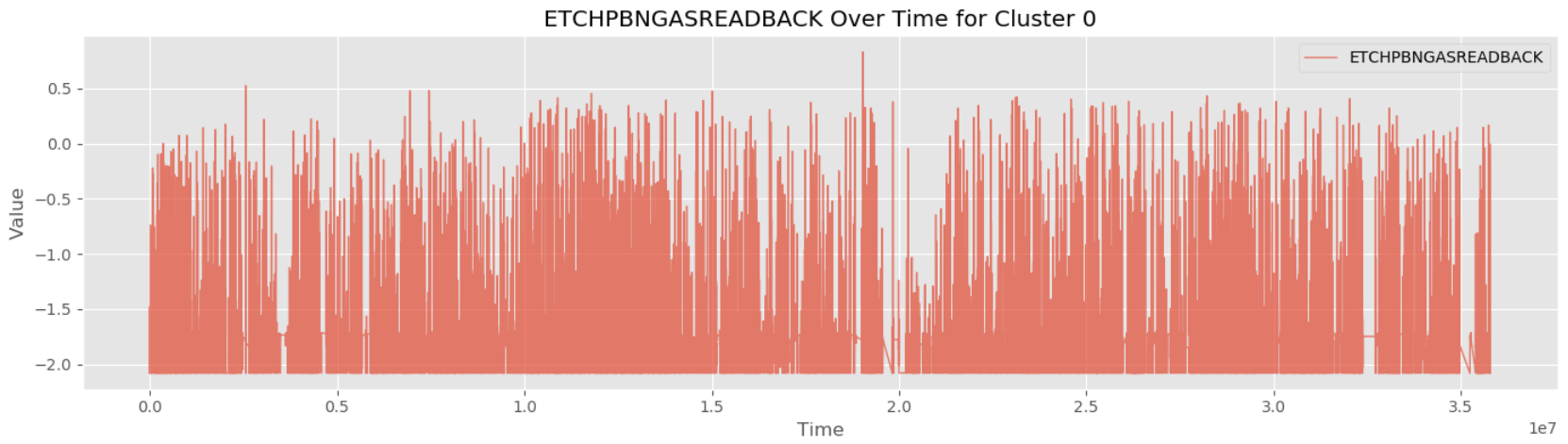
* **ETCHPBNGASREDBACK** shows a lot of extreme outliers reaching very high values indicating unstableness.
* **ETCHSUPPRESSORCURRENT** also shows a lot of extreme outliers with very high values.
* **FLOWCOOLPRESSURE** is generally clustered around positive values but also shows extreme outliers.

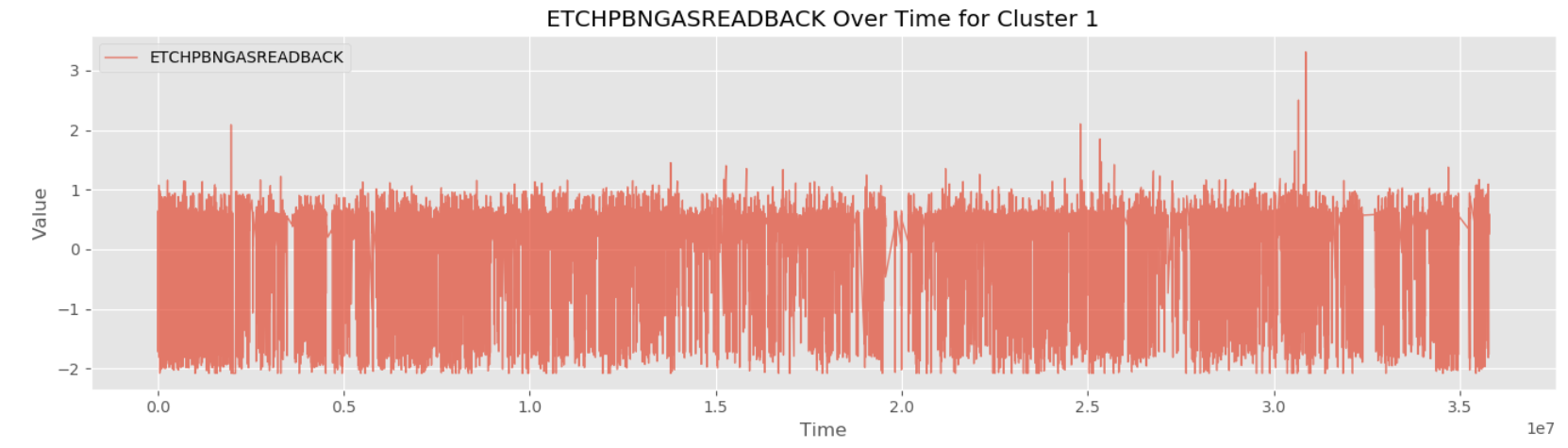
**Outliers:**

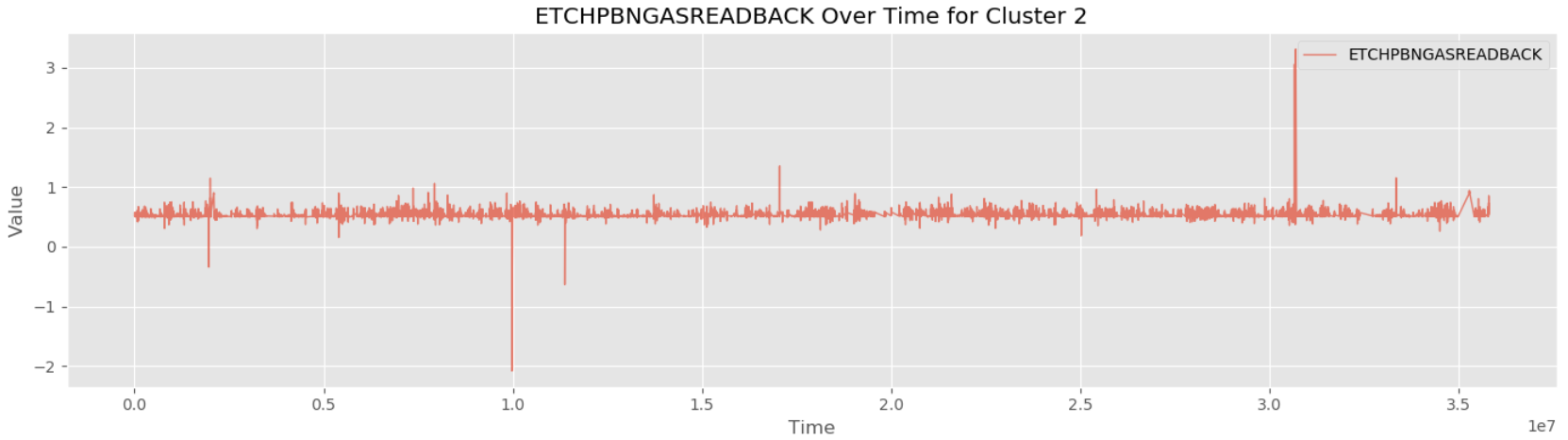
* Both Cluster 1 and Cluster 2 exhibit more **extreme** outliers, especially in **FLOWCOOLPRESSURE** and **ETCHSUPPRESSORCURRENT**.
* These outliers could indicate abnormal process conditions or operational anomalies.

**Time-Series Analysis:**

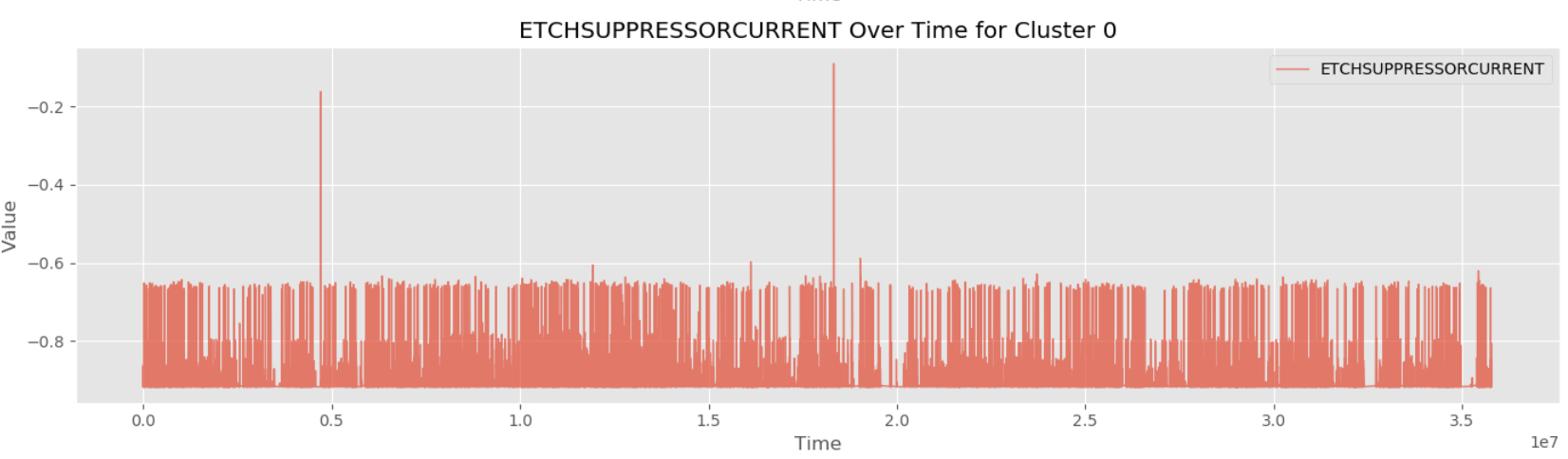
**ETCHPBNGASREADBACK -**

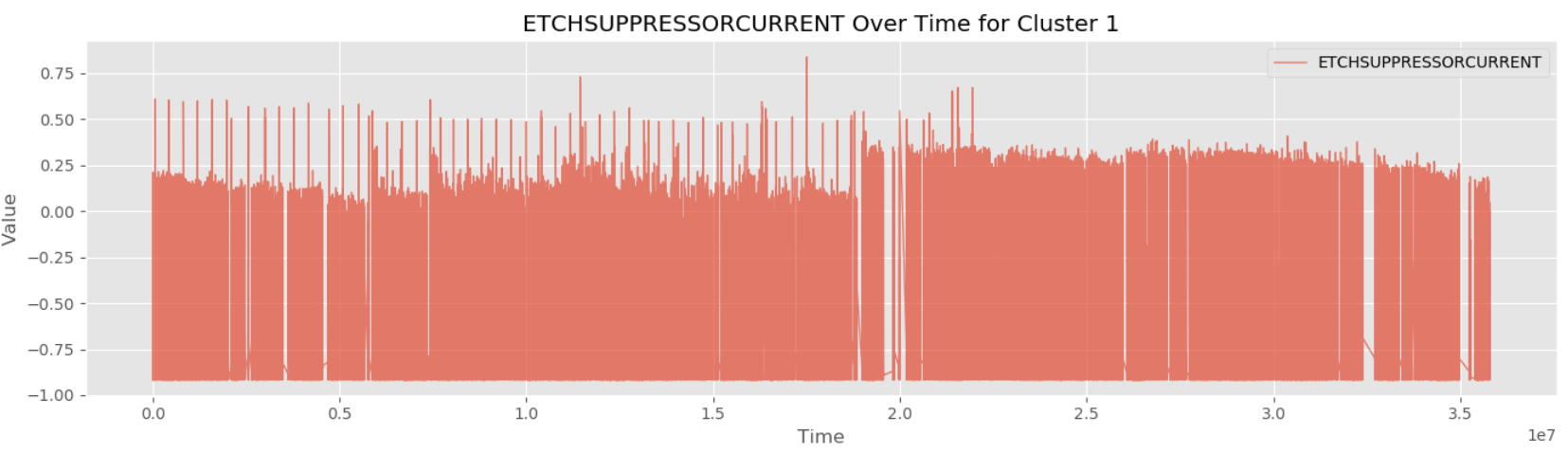
****

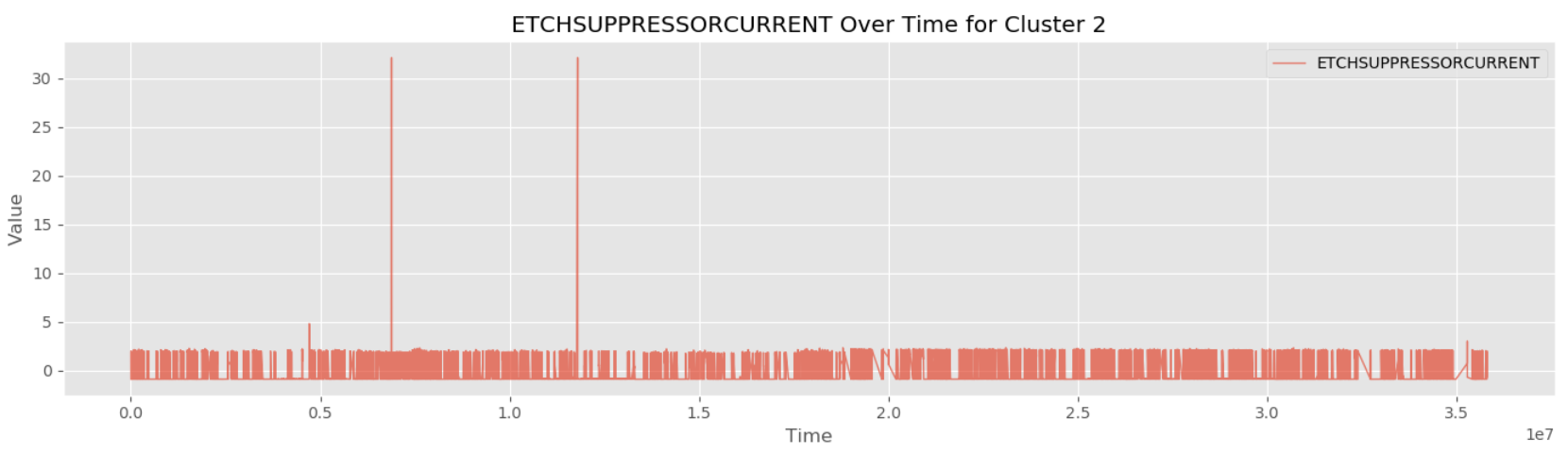
****

****

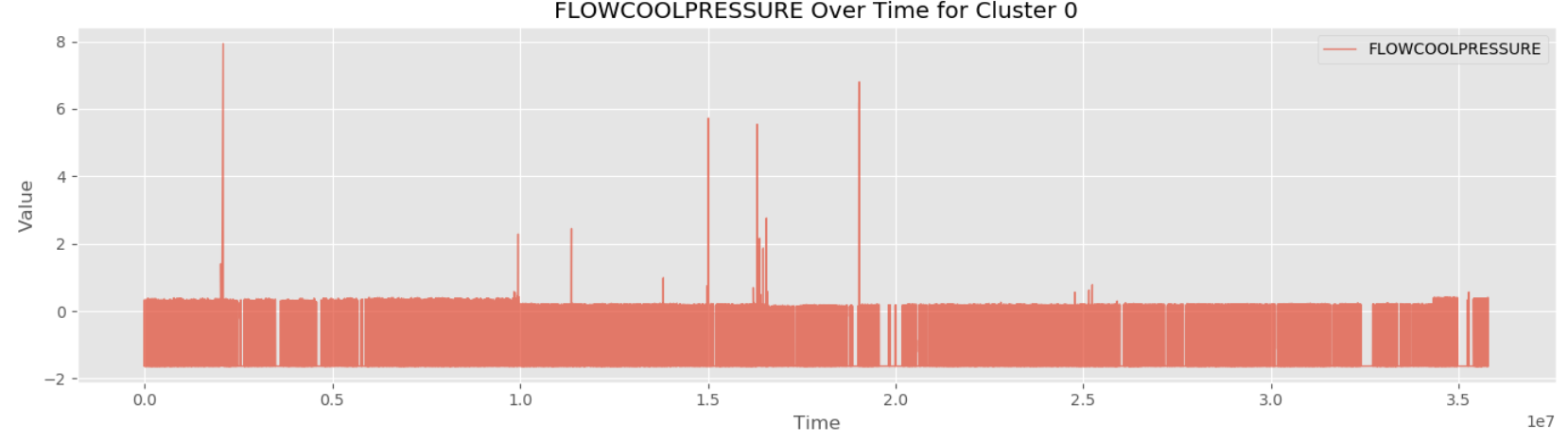
**ETCHSUPPRESSORCURRENT -**

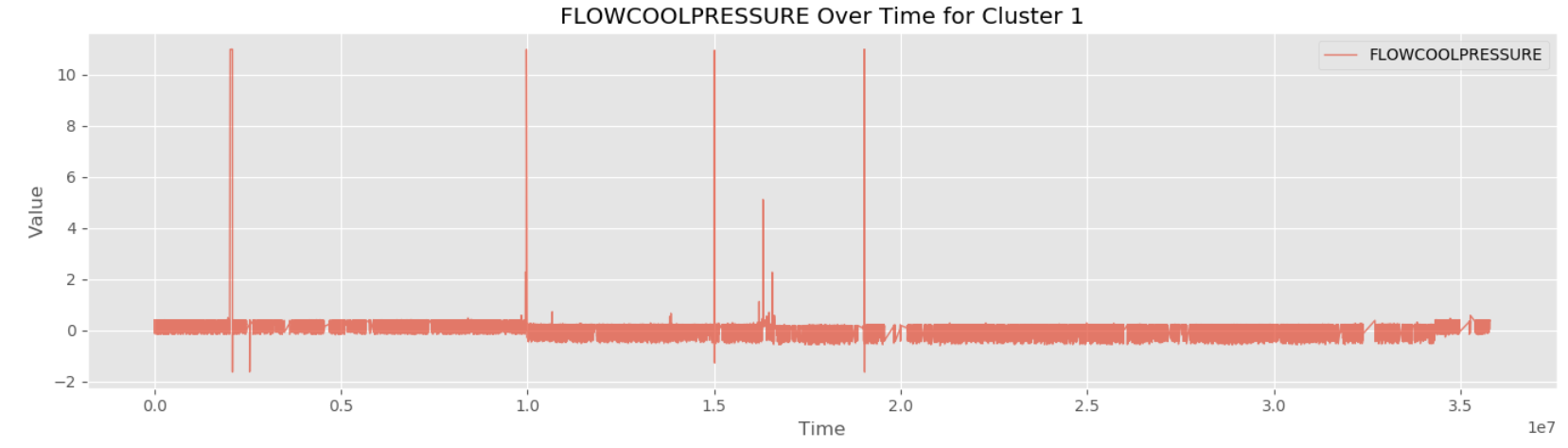
****

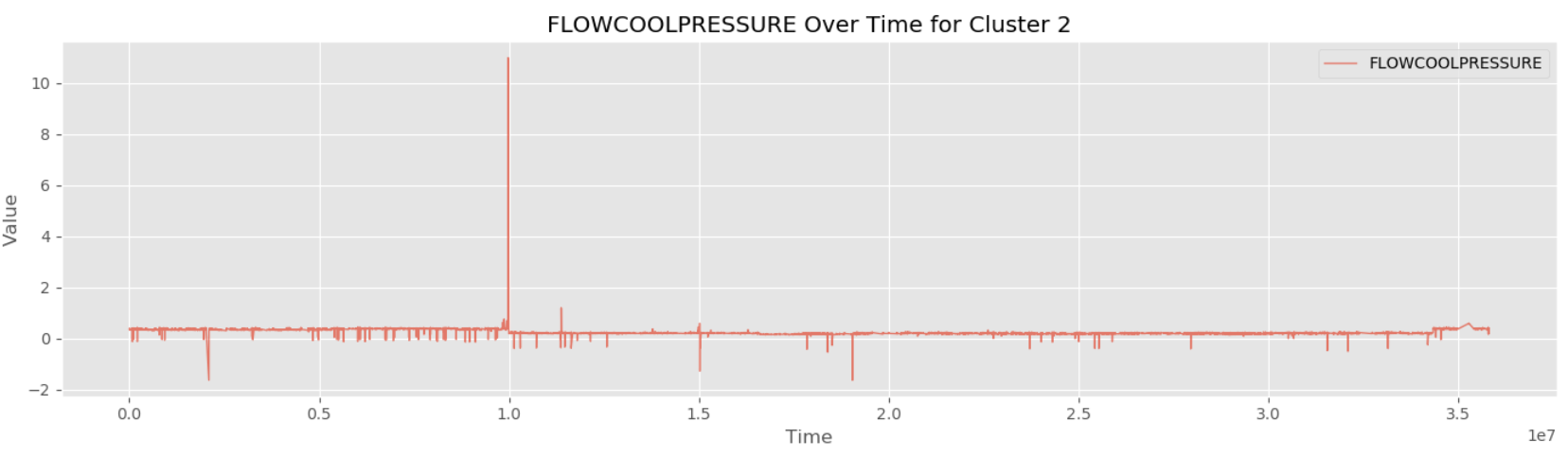
****

****

**FLOWCOOLPRESSURE -**

****

****

****

**Time-Series Analysis Summary:**

* **Cluster 0:** The Time series plots suggest consistent operation values over time, with fewer spikes or anomalies in the key features.
* **Cluster 1:** This cluster shows significant variability over time, with frequent spikes in **FLOWCOOLPRESSURE** and **ETCHSUPPRESSORCURRENT**, suggesting unstable behavior.
* **Cluster 2:** This cluster appears more stable overall but large spikes in **ETCHSUPPRESSORCURRENT** and **FLOWCOOLPRESSURE** could indicate critical operational issues during specific time windows.

**Conclusion:**

* **Cluster 0** represents stable operating conditions, with low variability and minimal outliers.
* **Cluster 1** displays high variability, particularly in **ETCHSUPPRESSORCURRENT** and **FLOWCOOLPRESSURE**, indicating unstable operations.
* **Cluster 2** shows mixed behavior, while mostly stable it has extreme outliers, especially in **ETCHSUPPRESSORCURRENT**, suggesting potential extreme conditions.

Unfortunately without further domain knowledge these findings come with limited understanding of the data.

🔍 Anomaly Detection

**Introduction to Anomaly Detection:**

The anomaly detection process aims to identify deviations from normal operational behavior, which could indicate faults or inefficiencies. While clustering provided insights into the dataset’s structure, it was determined that a separate **LSTM**-based approach would be more effective for detecting anomalies in this project.

**Why LSTM?**

**Long Short-Term Memory** (LSTM) is a type of recurrent neural network (RNN) architecture designed to learn from sequential data by capturing temporal dependencies over time. Unlike traditional RNNs, LSTMs address the problem of vanishing and exploding gradients, enabling them to learn long-term dependencies effectively.

In this project, the LSTM model analyzes sequences of operational data to classify them as normal or anomalous. The architecture includes memory cells and gating mechanisms that **allow the model to retain important historical patterns** while discarding irrelevant information, making it very effective for detecting anomalies in time-series datasets.

Methodology

**Methodology:**

1. **Data Preparation:**

Anomaly detection relies heavily on well-prepared data to ensure the LSTM model can effectively learn and generalize patterns. The steps for preparing the data are as follows:

* **Data Cleaning:**

Removing irrelevant features based on EDA and addressing missing values.

* **Normalization:**

All feature values of the train dataset were scaled between 0 and 1 using MinMaxScaler to ensure the LSTM model converges efficiently during training.

* **Sequence Generation**:

The time-series data was transformed into sequences of fixed length (30 timesteps per sequence). Each sequence captures temporal dependencies in the data.

* **Train and Test Split:**

The data was split into train and test datasets with a ratio of 80:20 respectively.

1. **Model Architecture:**

A Long Short-Term Memory (LSTM) neural network was implemented for anomaly detection. The architecture is designed as follows:

* **Input Layer:**

The input sequences consisted of 30 timesteps of the data.

* **LSTM Layers:**

The **First** layer consists of 128 LSTM units while returning the sequences in order to pass them to the next layer with a 20% dropout to reduce overfitting. The **Second** layer consists of 64 LSTM units without returning the sequences for conclusion, with 20% dropout for regularization.

* **Dense Output Layer:**

A single neuron with a **sigmoid activation function**, outputting probabilities between 0 and 1 to classify each sequence as normal or anomalous.

* **Loss Function:**

Binary cross-entropy was used to measure the model's error in predicting the anomaly labels.

* **Optimizer:**

Adam optimizer was chosen for efficient and adaptive learning.

1. **Training the Model:**

* **Training Configuration:**

The model was trained with a batch size of 64 and a limit of 30 epochs. Early stopping was used to stop training if the validation loss did not improve for 3 consecutive epochs, to reduce overfitting.

* **Class Imbalance Handling:**

Class weights were computed to address the imbalance between the data. This ensured the model did not disproportionately favor the majority class.

1. **Evaluation Metrics:**

* **Precision:**

The proportion of correctly identified anomalies out of all sequences predicted as anomalies.

* **Recall:**

The proportion of **true** anomalies identified by the model.

* **F1-Score**:

The harmonic mean of precision and recall.

* **Confusion Matrix:**

The visualization of the performance of the training model.

* **MSE (Mean Squared Error)**:

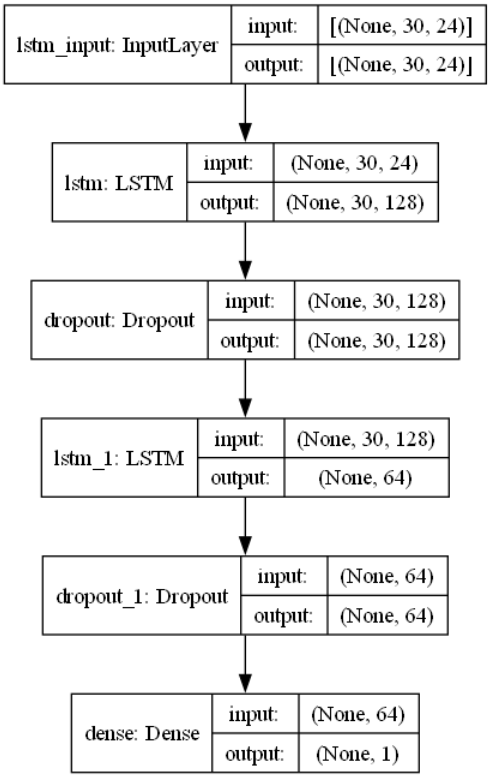
Used as an additional evaluation metric to assess prediction accuracy by comparing true and predicted probabilities. Mainly used to compare Train dataset and Test dataset later on.

1. **Deployment and Test Application:**

The model was applied to the TEST dataset in order to:

* Predict anomalies in an unseen environment.
* Evaluate the model’s performance on unlabeled data for further tuning and improvements.

*Architecture of the LSTM Model used for Anomaly Detection:*

**

Experimenting

**Experimenting with a Sample Dataset:**

The objective of this step was to validate the feasibility of using an LSTM model for anomaly detection by experimenting with a smaller, manageable subset of the dataset. This also helped in reduced runtimes and experimenting with different model architectures and hyperparameter tuning.

The experimentation was done starting from 200K data points and was scaled slowly to 1 Million data points for broader testing.

#### **Steps and Methodology:**

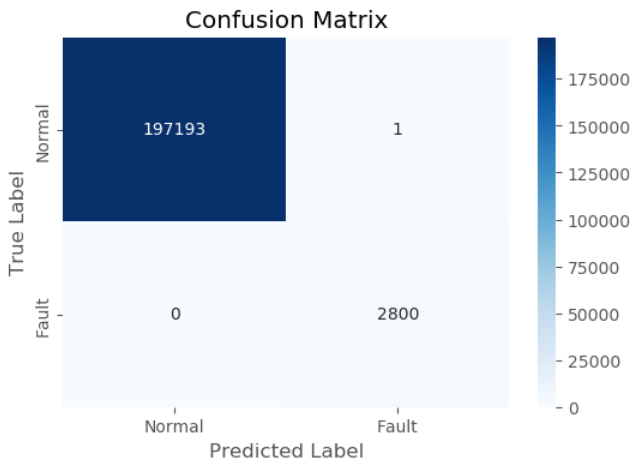
All of the steps in the previous Methodology section were also applied here.

1. **Data Preparation**:
   * A sample of 1 Million data points was extracted from the full dataset, ensuring that it included both "normal" and all of 13,639 "fault" labels to represent anomalies.
   * The dataset was preprocessed:
     + Features were normalized using the **MinMaxScaler** to bring them to a [0, 1] range.
     + Data was reshaped into a time-series format, with 30 timesteps and 24 features.
2. **Model Training**:
   * The LSTM model architecture described in the **Methodology** section was trained on the sample dataset.
   * A **binary cross-entropy loss function** was used for the binary classification task.
   * The **Adam optimizer** was applied.
   * To combat class imbalance, **class weights** were used.
3. **Validation**:
   * Validation data (20% of the sample dataset) was used to monitor the model's performance during training.
   * Early stopping was employed to prevent overfitting.

**Results:**

**Confusion Matrix**:

* A confusion matrix was plotted to assess how well the model identified "normal" and "fault" conditions in the validation dataset.



**Performance Metrics**:

| **Metric** | **Value** |
| --- | --- |
| Accuracy | 0.9999 |
| Precision | 0.9999 |
| Recall | 1.0 |
| F1 Score | 0.9998 |

#### **Key Observations:**

* The model achieved near-perfect results on the sample dataset, with high values for precision, recall, and F1-score.
* The confusion matrix revealed minimal false positives and false negatives, indicating strong predictive power on the sample dataset.
* These results validated the approach and allowed scaling the solution to the full dataset.

Even though the results might suggest overfitting, many steps were implemented in order to reduce overfitting as much as possible.

Scaling to Full Dataset

### **Scaling to the Full Dataset:**

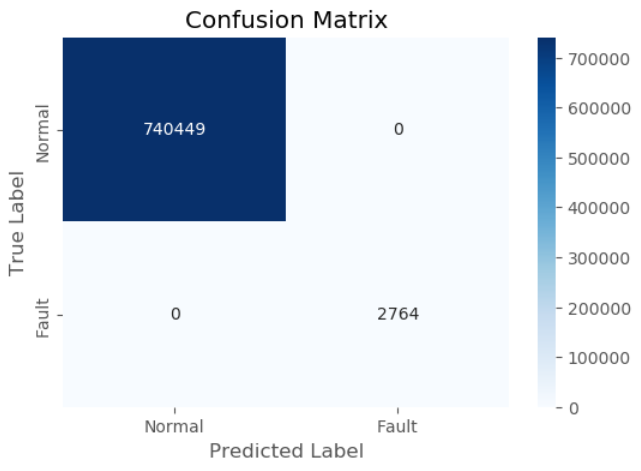
Following the success of the LSTM model on the sample dataset, the next step involved scaling the solution to the full dataset of approximately 3.7 million data points. This stage aimed to validate the model's ability to handle large-scale data while maintaining its performance for anomaly detection.

#### **Steps and Methodology:**

1. **Data Preparation**:
   * The entire dataset was preprocessed similarly to the sample dataset:
     + Features were normalized using the **MinMaxScaler**.
     + The time-series data was reshaped into a format with 30 timesteps and 24 features per sample.
   * A data generator was implemented to efficiently load batches of data during training, avoiding memory overflow.
2. **Model Training**:
   * The same LSTM model architecture was utilized.
   * The training setup included:
     + **Batch size**: 64.
     + **Early stopping**: To terminate training when the validation loss did not improve for three consecutive epochs.
     + **Class weights**: Used to address the imbalance between "normal" and "fault" samples.
   * Training on the full dataset required significantly more computational resources, but optimizations in data loading and processing ensured successful execution.
3. **Validation**:
   * Validation was performed on a subset (20%) of the full dataset to monitor the model’s performance during training.
   * Metrics such as loss, accuracy, precision, recall, and F1-score were evaluated to ensure consistent performance.
   * Confusion Matrix was also implemented in order to monitor model performance.
   * Early stopping was used to reduce overfitting.

**Results:**

**Confusion Matrix:**

****

**Performance Metrics:**

| **Metric** | **Value** |
| --- | --- |
| Accuracy | 1.0 |
| Precision | 1.0 |
| Recall | 1.0 |
| F1 Score | 1.0 |

#### **Key Observations:**

* The LSTM model scaled well to the full dataset, maintaining high accuracy and recall values for anomaly detection.
* Efficient data preprocessing and the use of a data generator were critical for handling the large dataset size without memory bottlenecks.

The results we managed to achieve here are perfect, too perfect actually. While overfitting is most likely present and is a concerning matter, from my understanding what matters most is how the model performs on the TEST data which is what’s coming next.

Applying TEST Data

**Applying the Model to the Test Dataset:**

The primary goal of this phase is to evaluate the trained LSTM model on the unseen test dataset. The test dataset was designed to simulate real-world operations and contained no fault labels. This presented the challenge of evaluating the model's performance without direct ground truth labels. Therefore there will be no Confusion Matrix or Metrics such as Precision, Recall and so on. Instead we will use **MSE** in order to compare the model’s performance on the Train Dataset and Test Dataset and derive insights from that.

The Test Dataset consists of 1,270,706 data points and 24 features.

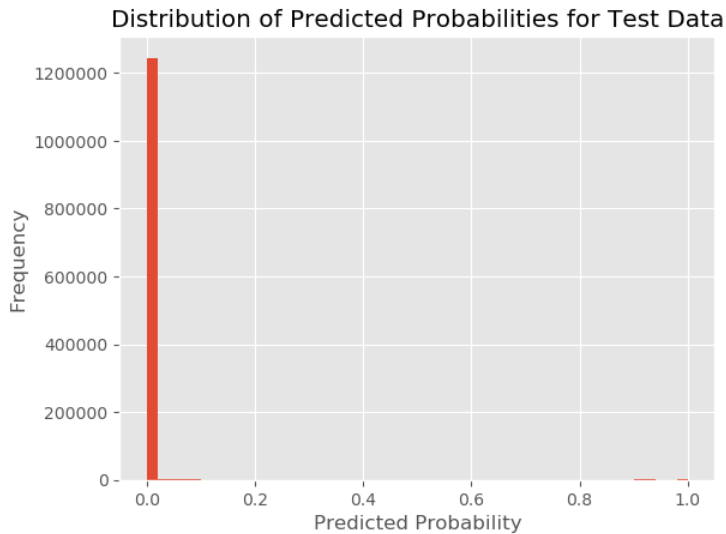
#### **Steps and Methodology:**

1. **Test Data Preparation**:
   * The test dataset was preprocessed to match the format used for training:
     + **Normalization**: The MinMaxScaler fitted on the training data was applied to scale the test dataset.
     + **Data Cleaning:** The data was cleaned and preprocessed just like the train dataset.
     + **Reshaping**: The data was transformed into the time-series format with 30 timesteps and 24 features per sample, consistent with the LSTM model's input requirements.
2. **Inference**:
   * Predictions were generated for the test dataset using the trained LSTM model.
   * The model returned **probability scores** indicating the likelihood of a data point being classified as an anomaly.
3. **Threshold-Based Classification**:
   * A probability threshold of 0.5, determined during validation on the full dataset, was applied to classify data points:
     + Data points with a probability greater than the threshold were labeled as anomalies.
     + Remaining points were labeled as normal operations.
4. **Post-Prediction Analysis**:
   * The predicted probabilities were analyzed to understand the distribution of anomalies in the test dataset.
   * Key statistics, such as the percentage of anomalies detected and MSE were calculated to provide insights.

**Results:**

**Class Distribution in Predictions:**

* The predicted labels were analyzed to determine the proportion of normal and anomalous data points in the test dataset.
  + Normal: 1,257,859 samples, roughly 98%.
  + Anomalies: 12,847 samples, roughly 2%.



If we refer back to the Training dataset, we saw that out of 740k validation data points only 2.8K were classified as anomalies.

The ratio of anomaly samples to normal samples in the test data falls in line with the results of the training dataset.

Furthermore as we can see from the distribution plot, almost all of the probability classifications were in the range of **0.8 - 1.0**, meaning there's a clear separation between normal and anomalous data.

**Summary:**

**Probability Distribution**:

* The model's predicted probabilities provided a clear separation between normal and anomalous samples.
* This separation was consistent with the model's performance on the training and validation datasets.

**Anomaly Detection**:

* The anomalies detected by the model more than likely corresponded to data points with unique patterns, suggesting the model's ability to generalize to unseen data.

**Evaluation Without Ground Truth**:

* In the absence of true labels, MSE and high-probability calculations were used to validate the model's predictions.

Train VS Test MSE

### **MSE Comparison Between Training and Test Datasets:**

First let’s take a look behind the purpose of MSE and why we are going to use it to evaluate the performance of our model.

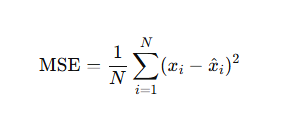
**MSE, a Brief Explanation:**

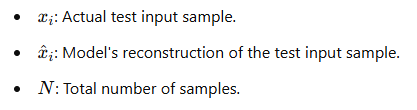
The purpose of Mean Squared Error (MSE) is to compute the reconstruction error for the test dataset and to analyze their distribution in order to detect potential anomalies.

The way it works is we take each test sample and calculate the squared difference between the actual input values and the predicted values from the model.

Then we average these squared differences to obtain the MSE for the dataset.

The formula is as follows:





**Leveraging MSE for Evaluation:**

The calculation of the MSE was applied to both the Training Data and the Testing Data with the following approach:

**Training Data MSE**:

* Calculate the MSE between the true training data values and the model's predicted values.

**Test Data MSE**:

* Similarly, compute the MSE for the test data.
* Since the test dataset lacks true labels, the MSE here is based on the reconstruction error from the trained LSTM model.

In the end we compared the two metrics against each other while looking for the **minimal** MSE value on the Test Data.

**Findings:**

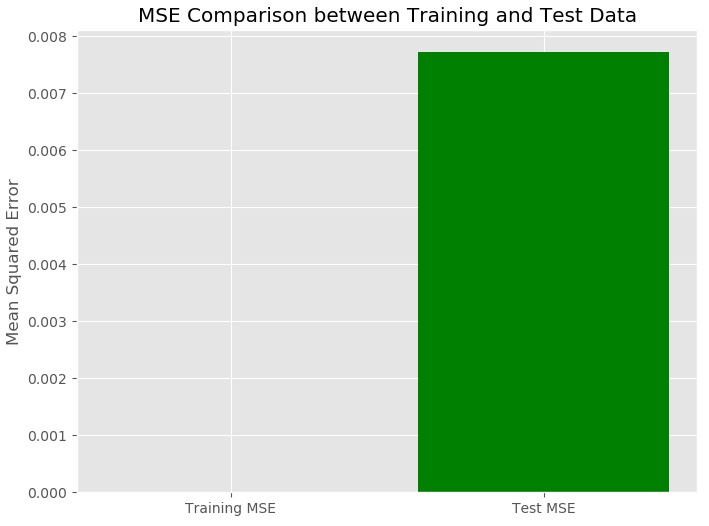
* **Training Data MSE**:
  + The MSE for the training dataset was **extremely low**, indicating the model has learned the patterns in the training data well.
* **Test Data MSE**:
  + The MSE for the test dataset was **higher than the training data MSE**, as expected.
  + However, the gap between the two metrics was **modest**, suggesting that the model has performed well on unseen data.

During the experimentation on the Test dataset we experimented with different results of the Training data.

Some results led to a **smaller gap** between the training MSE and testing MSE, **however the testing MSE was larger.**

Eventually we decided on **minimizing** the test MSE regardless of the training MSE since a smaller MSE value represents a better overall performance of the model.

The results are as follows:



Even though the gap seems big, the two values are extremely small numbers.

Here are the actual values of the MSE:

| **Dataset** | **MSE Value** |
| --- | --- |
| Training Data | 0.00000237 or 2.37e-06 |
| Test Data | 0.0077182333916425705 |

The results indicate that the **model exhibits good generalization**, with the test MSE slightly higher than the training MSE, which is expected when applying the model to unseen data.

❗ Limitations

**Limitations:**

While working on the project we came across some limitations.

#### **1. Lack of Ground Truth in Test Data:**

* Since the test dataset lacks true labels for anomalies, the evaluation relies entirely on reconstruction error (MSE) and probability distributions.
* This makes it difficult to validate the exact effectiveness of the model in a real-world scenario without a comparison to actual ground truth values.

#### **2. Imbalanced Dataset:**

* The dataset is highly imbalanced, with anomalies being rare compared to normal data. While methods like weighted loss and class-weights were used during training, these might not fully address the imbalance.
* The model could still have a bias toward normal data, missing subtle anomalies in the test data.

#### **3. Scalability and Computational Constraints:**

* Working with a large-scale dataset, especially for time-series data, posed challenges in terms of memory usage and computation time.
* These limitations could restrict the model's application to even larger datasets or real-time anomaly detection.

#### **4. Limited Interpretability:**

* While the LSTM-based approach performs well at detecting anomalies, it lacks transparency regarding *why* certain instances are classified as anomalous.

📕 Ending Notes

### **Ending Notes:**

The anomaly detection framework implemented in this project demonstrates the power and flexibility of LSTM-based models while using clustering for deeper exploration of the data.

#### **Key Takeaways:**

1. **High Generalization Capability**:
   * The model successfully scaled from a small sample dataset to the full dataset, as evidenced by a low value in Mean Squared Error (MSE) of the test dataset.
   * This demonstrates its ability to generalize and adapt to larger datasets without significant degradation in performance.
2. **Anomaly Identification**:
   * By leveraging probabilistic thresholds, the model effectively distinguished between normal and anomalous behaviors in the data.
3. **Scalability**:
   * The successful application of the model to a large industrial dataset confirms its scalability and effectiveness in handling complex, high-dimensional time-series data.
4. **Interpretability of Results**:
   * The clear distinction between anomaly and normal distributions (as visualized in the reconstruction error plots) provides confidence in the model’s decision-making process.

#### **Concluding Remarks:**

This project serves as a foundational step in leveraging deep learning for industrial anomaly detection. The scalability, flexibility, and effectiveness demonstrated by the model provide a strong basis for further research and practical applications.

With limited domain knowledge and tools there is only so much we can achieve. Hopefully the findings we found and the methods we used will prove useful in solving real-world problems with similar settings.