**SUMMARY REPORT**

Table of Contents

[**1 METHODS** 2](#_Toc72707715)

[**1.1** **DATA GENERATION** 2](#_Toc72707716)

[**1.2 MODEL DEVELOPMENT** 4](#_Toc72707717)

[**1.2.1 K-MEANS** 4](#_Toc72707718)

[**1.2.2 AUTOENCODER** 4](#_Toc72707719)

[**1.2.3 ISOLATION FOREST** 5](#_Toc72707720)

[**1.2.4 IMPLEMENTATION DETAILS** 5](#_Toc72707721)

[**1.3 POST-MODEL ANALYSIS** 6](#_Toc72707722)

[**2 RESULTS** 8](#_Toc72707723)

[**2.1** **DESCRIPTIVE STATISTICS** 8](#_Toc72707724)

[**2.2 DISTRIBUTION PLOT ANALYSIS** 8](#_Toc72707725)

[**2.3 SCATTERPLOT VISUALIZATION OF ANOMALIES** 9](#_Toc72707726)

[**2.4 PATTERN OF FAILURE ANALYSIS** 10](#_Toc72707727)

[**3 DISCUSSION** 12](#_Toc72707728)

[**4 LIMITATIONS** 12](#_Toc72707729)

[**TECHNICAL REQUIREMENTS** 13](#_Toc72707730)

# **1 METHODS**

This machine learning project is conducted using a proof of concept approach. There are 3 key steps involved in this project. The first step involves the generation of synthetic data to simulate the values of assets. The second step involves the development of the anomaly detection machine learning models. The third step involves the post-model analysis of the anomalies. Details of each step are elaborated below.

## **DATA GENERATION**

The data generation process was necessary for this machine learning project due to the lack of real-world data. Without data, the machine learning models could not be developed. The entire data generation process was automated using Python 3 scripts. Specifically, 2 scripts were written to create 10 datasets each for “Current (Ampere)” and “Temperature (Celsius)”. Both scripts are largely similar except for the range of values taken on by current and temperature.

3 helper functions and 1 main function are created for each script. The helper functions facilitate code reusability and modularity by addressing small chunks of problems separately. The main function indicates the starting point for the script where the code will be executed sequentially. The helper functions will be executed within the main function. The table below details the helper functions and their description.

**Table 1 Description of Helper Functions**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Current (Ampere) Dataset** | **Temperature (Celsius) Dataset** | **Description** |
| **Helper Function 1** | shift\_value\_generator | shift\_value\_generator | Generate values for a shift such as day or night shift |
| **Helper Function 2** | min\_zero\_repeat | min\_num\_repeat | Ensure “0” or relevant number is repeated 6-12 times (30-60 mins) to simulate "off" asset |
| **Helper Function 3** | shift\_df\_generator | shift\_df\_generator | Generate day and night dataframe |

To simulate real-world conditions, there will be different values generated for day and night shifts respectively. The day shift consists of the working hours from 8 am to 8 pm while the night shift will consist of the remaining period. The majority of the assets’ usage happens during the day shift and experiences a higher workload as well. It is assumed that anomalous values only appear during the day shift where assets are heavily utilized and values might exceed the threshold. During the night shift, the assets either experience a lighter workload or are turned off. The table below summarizes the distribution of values for the datasets.

**Table 2 Distribution of Values for the Datasets**

|  |  |  |
| --- | --- | --- |
|  | **Current (Ampere) Dataset** | **Temperature (Celsius) Dataset** |
| **Day Shift** | | |
| **Dataset 1** | 44, 45, 47, 49, 50, 51, 54, 58, 67, 70, 73 | 44, 48, 51, 54, 57, 61, 65, 70, 79, 81, 84 |
| **Dataset 2** | 44, 45, 46, 48, 50, 51, 54, 58, 67, 69, 71 | 44, 48, 51, 53, 57, 60, 64, 69, 79, 80, 82 |
| **Dataset 3** | 43, 44, 46, 48, 49, 51, 53, 57, 65, 68, 69 | 43, 47, 49, 53, 57, 61, 65, 70, 78, 80, 82 |
| **Dataset 4** | 43, 44, 45, 47, 49, 51, 53, 57, 65, 67, 69 | 43, 47, 49, 52, 56, 60, 64, 69, 78, 79, 81 |
| **Dataset 5** | 42, 43, 45, 47, 48, 51, 53, 56, 63, 65, 67 | 42, 46, 49, 53, 56, 60, 63, 67, 77, 79, 80 |
| **Dataset 6** | 42, 43, 44, 46, 48, 50, 52, 56, 63, 64, 66 | 42, 46, 49, 51, 55, 59, 63, 68, 77, 78, 80 |
| **Dataset 7** | 41, 42, 44, 46, 47, 50, 52, 55, 61, 64, 65 | 41, 45, 48, 51, 54, 58, 62, 67, 76, 78, 80 |
| **Dataset 8** | 41, 42, 43, 45, 47, 50, 52, 55, 61, 63, 65 | 41, 45, 48, 52, 53, 57, 61, 67, 76, 77, 79 |
| **Dataset 9** | 40, 41, 43, 45, 46, 50, 51, 54, 60, 62, 63 | 40, 44, 48, 51, 55, 59, 63, 68, 75, 77, 78 |
| **Dataset 10** | 40, 41, 42, 44, 46, 50, 51, 53, 60, 61, 64 | 40, 44, 48, 51, 54, 58, 62, 67, 75, 76, 78 |
| **Weights** | 8, 8, 15, 16, 16, 16, 8, 5, 3, 3, 2 | 8, 8, 15, 16, 16, 16, 8, 5, 3, 3, 2 |
| **Night Shift** | | |
| **All datasets** | 0, 30, 33, 35, 38, 40, 43, 45, 48 | 26, 31, 35, 37, 40, 45, 49, 51, 53 |
| **Weights** | 5, 9, 9, 16, 16, 16, 16, 8, 5 | 5, 9, 9, 16, 16, 16, 16, 8, 5 |

The day shift values are varied for each dataset while the night shift values are kept constant for both current and temperature. To simplify the data generation process, all values are discrete instead of continuous. The weights sum to 100 and represent the frequency at which each respective value appears for the corresponding weight value. For example, in dataset 1 for the current value, the value 44 will appear approximately 8% (8/100) of the time.

For the night shift values in the current and temperature datasets, 0A and 26-degree celsius are used to represent the assets’ “off” condition where they are not being utilized at all by the operators.

## **1.2 MODEL DEVELOPMENT**

The initial phase consists of defining the business problem at hand to accurately decide on the types of machine learning algorithms that can be utilized for the anomaly detection task. The main task was to identify the anomalous values for both current and temperature to prevent damage to the assets. However, there were no labels available to decide whether a data point was considered “normal” or “anomalous”. Therefore, an unsupervised machine learning approach has to be formulated.

Several existing algorithms for unsupervised anomaly detection are available. Some examples of such algorithms are K-means, autoencoder, principal component analysis, and isolation forest. A brief introduction of a few algorithms will be provided below.

### **1.2.1 K-MEANS**

K-means analyzes all data points and clusters those that are similar based on some distance metrics such as Euclidean distance. Human input is required to predefine the hyperparameter K, the number of clusters desired. The center point of the cluster is called the centroid. The user can define a threshold value such that if the distance between a data point and its nearest centroid exceeds it, then it is classified as an anomaly. This method suffers from input bias due to the requirement of predefining K. Most systems are dynamic and the ideal K value differs from time to time. The centroid positions are randomly initialized which results in non-deterministic solutions on different runs of the algorithm. Multiple runs are encouraged to obtain reliable results. Furthermore, the k-means algorithm has a quadratic time complexity which prevents it from scaling well with a large amount of data.

### **1.2.2 AUTOENCODER**

Autoencoder belongs to the artificial neural network family where the input is the same as the output. It utilizes a bottleneck architecture where the network is forced to learn a compressed representation of the input data and attempt to reproduce the original input from the compressed version. In the application for anomaly detection, normal data points are fed to the model where it attempts to reproduce them with as little loss as possible. Subsequently, when an anomalous data point is fed to the model, the autoencoder will face difficulty reproducing the normal input from the learned weights and result in a high reconstruction loss. Above a certain predefined threshold, the data point will be labeled as anomalous. The drawbacks of autoencoder are similar to most neural networks. It is data-hungry and requires sufficient inputs to prevent overfitting. Intensive hyperparameter tuning is required to find the best model architecture.

### **1.2.3 ISOLATION FOREST**

Isolation forest differs from most model-based approaches. It does not attempt to learn from normal data points first and later identify anomalous data points that do not fit the “normal” profile. Instead, it explicitly isolates anomalous data points based on the concept of anomalies being “few and different”. The base model of an isolation forest is a decision tree. The common usage of a decision tree is to split the data points until all terminal nodes are approximately pure or contain all points belonging to the same class. This concept forms the basis of an isolation tree. Since anomalies are relatively different from normal data points, the tree can isolate, or obtain a pure terminal node, using a lower number of splits. Normal points require more splits, or features, to be isolated from other normal points. An isolation forest is simply an ensemble of isolation trees to reduce the variance from using only a single tree. This provides stability and ensures that the results are confirmed by the consensus of multiple trees. There are a few advantages to this method. The isolation forest does not require building the entire tree as most anomalies can be isolated close to the root. It also does not require any distance computation which is a computational bottleneck for most anomaly detection algorithms like K-means and principal component analysis. It has a linear time complexity with a low memory requirement. This means that it can scale to a large amount of data.

After careful consideration of the pros and cons of each anomaly detection algorithm, the isolation forest is chosen for this project. It provides superior time complexity and memory usage.

### **1.2.4 IMPLEMENTATION DETAILS**

The isolation forest implemented in Python requires predefining multiple hyperparameters. In machine learning, a hyperparameter is a parameter whose value is used to control the learning process and cannot be learned during training like other parameters. Details of the hyperparameters used and their values are provided in the table below.

**Table 3 Hyperparameters Used and Chosen Values**

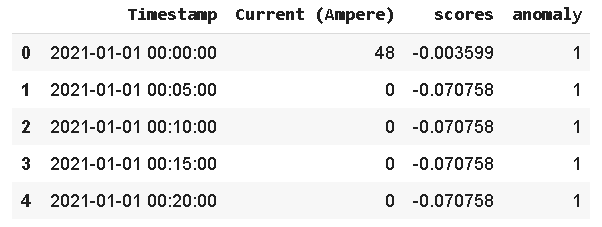
|  |  |  |  |
| --- | --- | --- | --- |
| **Hyperparameter** | **Data Type** | **Description** | **Value** |
| **n\_estimators** | int | The number of base estimators in the ensemble. | 100 |
| **max\_samples** | int | The number of samples to draw from X to train each base estimator. | 'auto' |
| **contamination** | float | The proportion of outliers in the data set. | float(0.05) |
| **max\_features** | int or float | The number of features to draw from X to train each base estimator.  If int, then draw max\_features features.  If float, then draw max\_features \* X.shape[1] features. | 1.0 |
| **bootstrap** | boolean | If True, individual trees are fit on random subsets of the training data sampled with replacement.  If False, sampling without replacement is performed. | False |
| **n\_jobs** | int or None | The number of jobs to run in parallel for both fit and predict.  None means 1 unless in a joblib.parallel\_backend context.  -1 means using all processors. | -1 |
| **random\_state** | int | The seed used by the random number generator. | 42 |
| **verbose** | int | Controls the verbosity of the tree building process. | 0 |

After defining the hyperparameters, the isolation forest can be fitted to each dataset. One model is developed for each dataset. A total of 20 models are created as a result. After the model is defined and fitted, the “scores” and “anomaly” columns are created. The values of both columns can be determined by calling decision\_function() of the trained model and passing the target column (current or temperature) as a parameter. A more negative score value and a value of “-1” for the anomaly column indicate the presence of an anomaly. A less negative score value and a value of “1” for the anomaly column represent a normal data point.

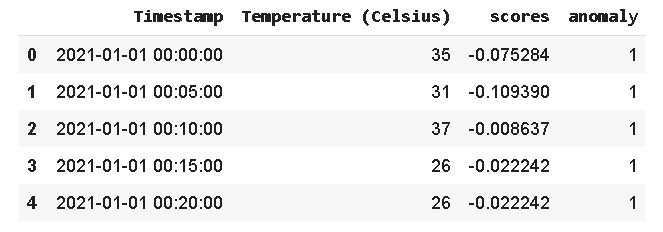
## **1.3 POST-MODEL ANALYSIS**

After building the isolation forest models, the post-model analysis was performed in Google Colab notebook using Python. A notebook allows a better visualization experience as compared to an integrated development environment (IDE). A separate notebook was created each for both current and temperature analysis. The analysis work performed was largely similar for both notebooks except for the values taken on by the different datasets.

The datasets containing the new columns “scores” and “anomaly” after training the isolation forest models were loaded in the Colab environment. Each dataset has 8928 rows and 4 columns. The following 2 figures show the first 5 rows of dataset 1 for current and temperature respectively.

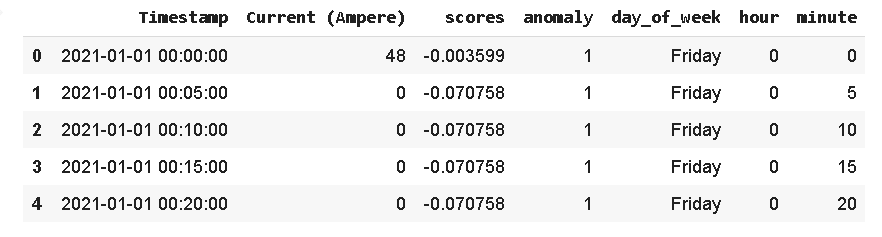


**Figure 1 First 5 Rows of Current (Ampere) Dataset 1**



**Figure 2 First 5 Rows of Temperature (Celsius) Dataset 1**

Next, the timestamp column was converted to datetime format (YYYYMMDD HH:mm:ss). With a standard format that Python can recognize, relevant features can be extracted from the timestamp column. Specifically, 3 new columns are engineered. They are “day\_of\_week”, “hour”, and “minute” columns. The “day\_of\_week” column contains the 7 days of the week: “Monday”, “Tuesday”, “Wednesday”, “Thursday”, “Friday”, “Saturday”, “Sunday”. The “hour” column contains integers from 0-23 representing the hour of the day under the 24-hour clock system. The “minute” column contains integers from 0-55 in intervals of 5 minutes. The figure below shows the updated columns.



**Figure 3 First 5 Rows of Current (Ampere) Dataset 1**

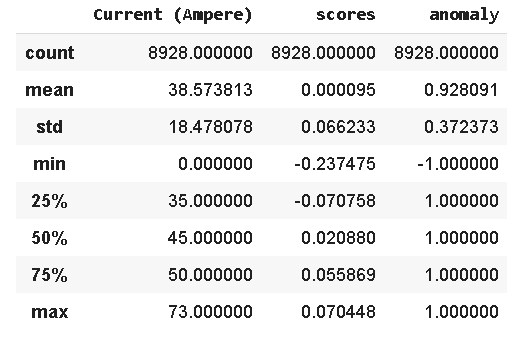
After the pre-processing steps are performed, proper analysis can be conducted. The following section will present the results of the analysis.

# **2 RESULTS**

The results section presents 4 key analyses to understand the data and model results in more detail. They are descriptive statistics, distribution plot analysis, scatterplot visualization of anomalies, and patterns of failure analysis.

## **DESCRIPTIVE STATISTICS**

Descriptive statistics provides a summary of statistics about the DataFrame columns. It is only applicable for columns of numeric data types. Some useful statistics computed are mean, standard deviation, and interquartile range values. The figure below shows an example of the results of the descriptive statistics.

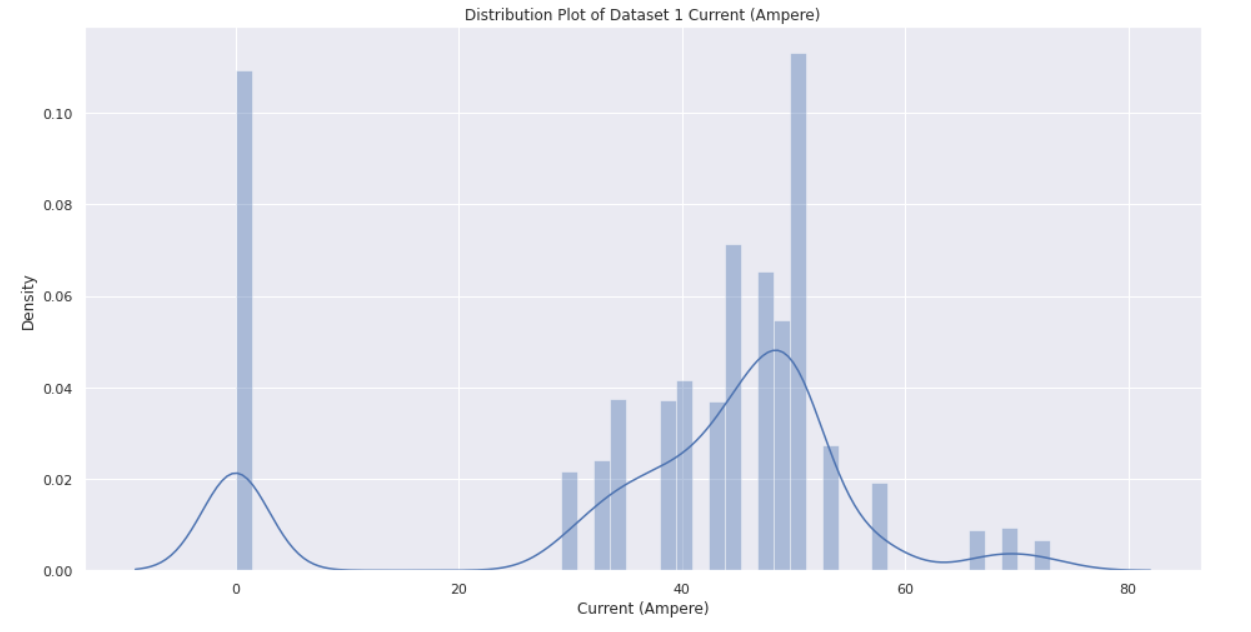


**Figure 4 Descriptive Statistics for Current (Ampere) Dataset 1**

With the information from the descriptive statistics computation, a quick understanding of the data can be achieved. For example, the mean value of the current is 38.57A with 50% of the values lying between 35A to 50A. The anomaly column indicates that most of the data points belong to the “normal” category as the mean is 0.93 which is close to 1, an indication of normality.

## **2.2 DISTRIBUTION PLOT ANALYSIS**

The distribution plot flexibly plots a univariate distribution of observations. It combines the histogram function with the gaussian kernel density estimate function. The figure below shows an example of a distribution plot.

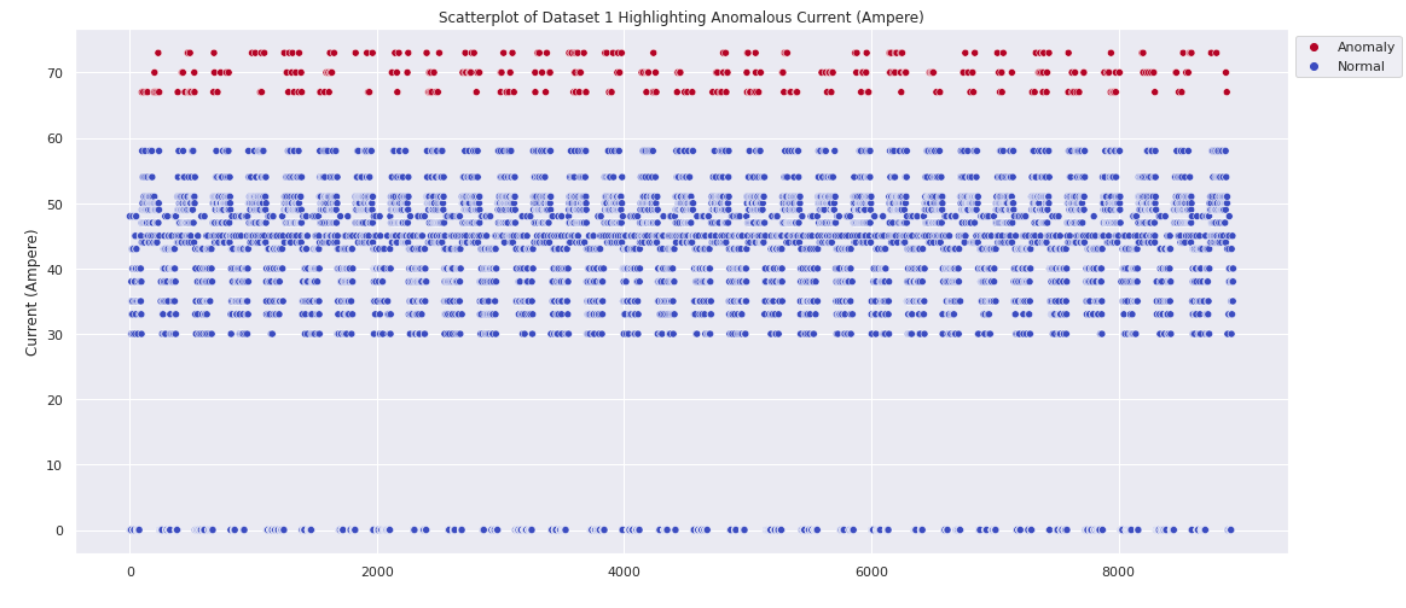


**Figure 5 Distribution Plot of Current (Ampere) Dataset 1**

It can be observed from the distribution plot that the data has a bimodal distribution with 2 distinct peaks at the 0 and 50+ range. The 0 values indicate the asset being turned off and transmits no current. A few outlier values can be observed at the right tail beyond the value 60. These are the values of interest as there is a high chance they rightfully belong to the “anomaly” category.

## **2.3 SCATTERPLOT VISUALIZATION OF ANOMALIES**

A scatterplot is used to display the relationship between 2 numerical variables. Every data point is represented by a dot corresponding to its value on the horizontal and vertical axis. The scatterplot shown below uses the DataFrame row index as the X-axis and the current values as the Y-axis. The row index is used instead of timestamp for brevity. They are essentially the same as the values are still ordered by time since the rows are not shuffled. The red dots represent anomalous data points while the blue dots represent normal data points.

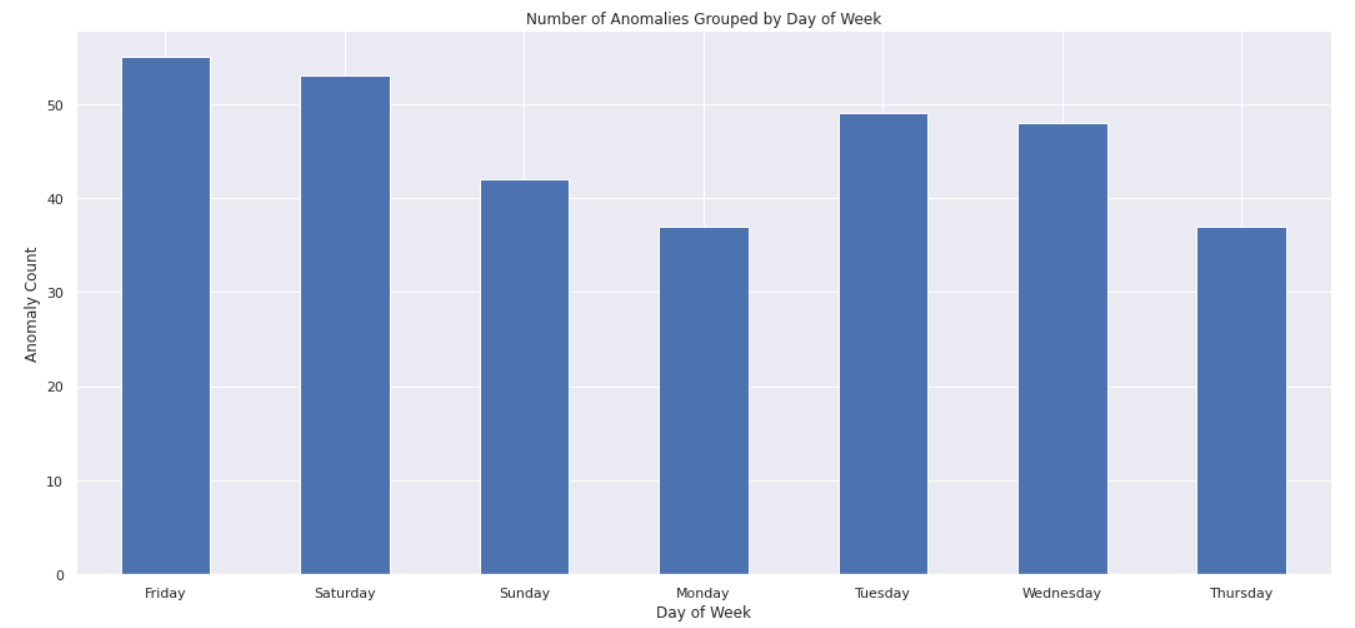


**Figure 6 Scatterplot of Current (Ampere) Dataset 1**

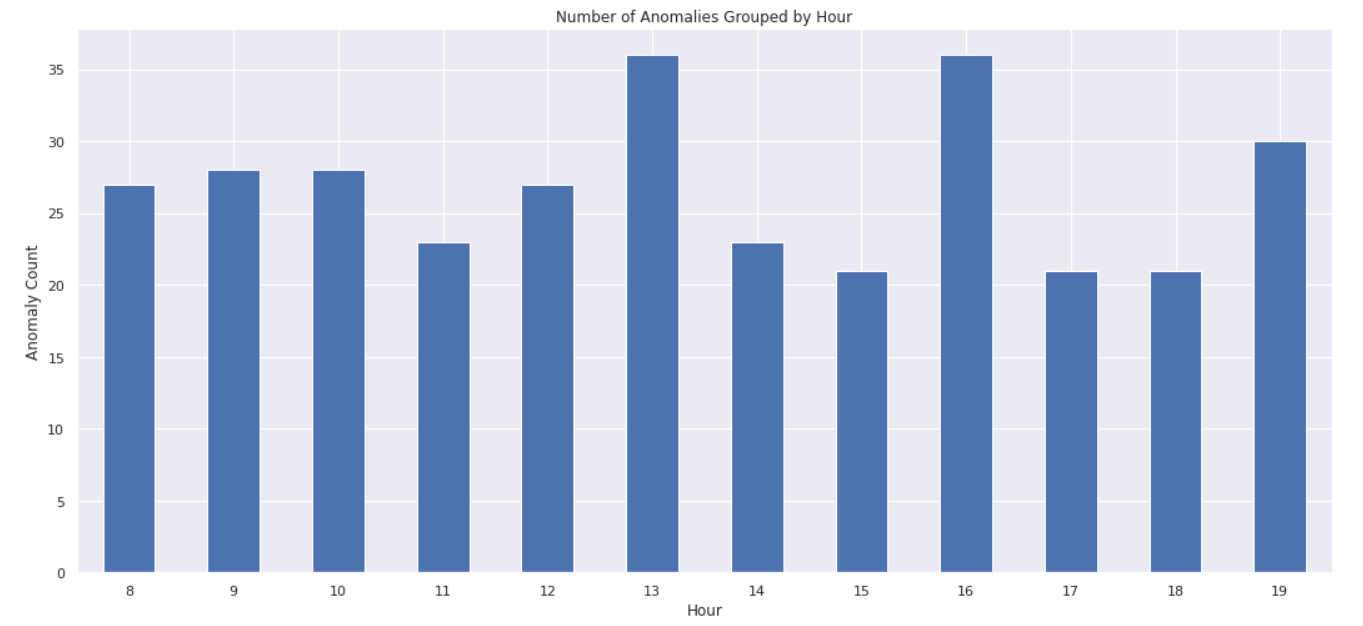
As can be seen from the scatterplot, the anomalous (red) data points are accurately detected by the isolation forest model. They are the values exceeding 60A which are usually considered to be the threshold. The 0 values, though far away from the bulk of the data points, are not mistakenly identified as anomalies as they appear rather consistently. The hyperparameter “contamination” has also been aptly set at 0.05 which indicates to the model that approximately 5% of the data will be anomalous on average. This might not work at times due to the changing environment which might affect the asset’s workload. However, as more data is fed into the model, the model stability should improve and reduce the variance of the predictions. Whenever necessary, the model can be retrained as well.

## **2.4 PATTERN OF FAILURE ANALYSIS**

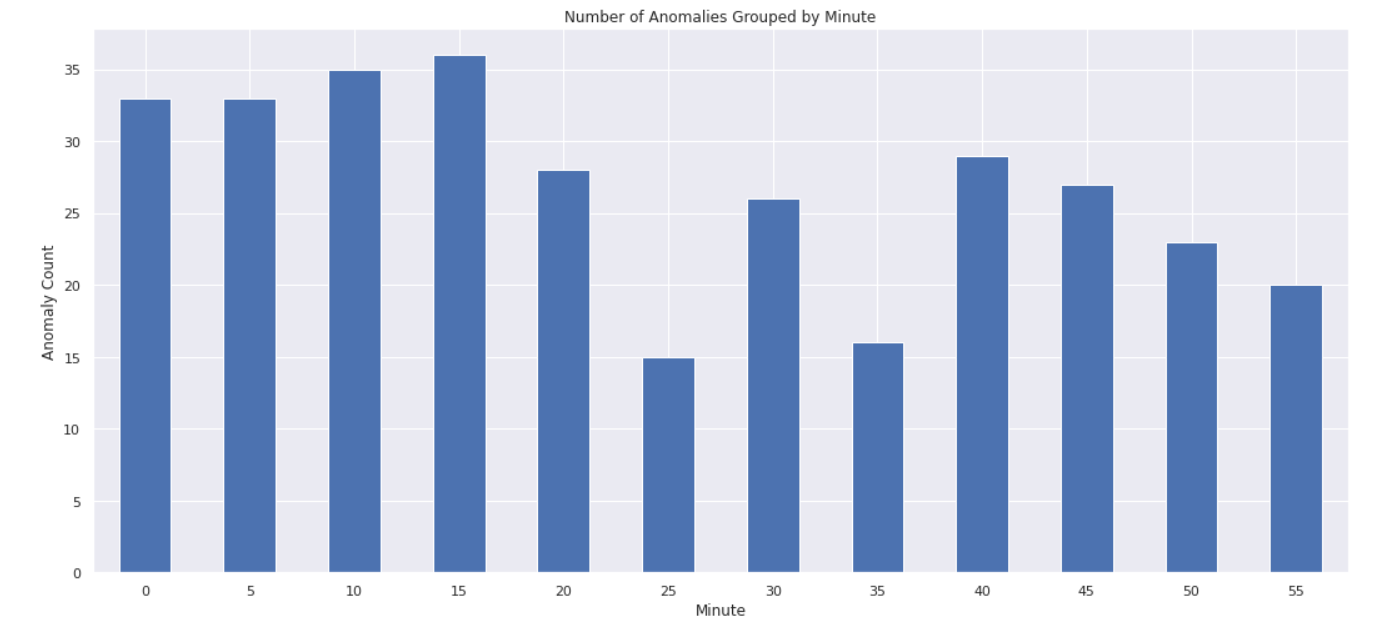
The pattern of failure analysis aims to determine when, where, and how the failure has occurred. Given the limited data available, a temporal analysis can be performed to correlate time with failure patterns. The timestamp column and its extracted features will be useful in this case. The following figures show the number of anomalies grouped by day of week, hour, and minute respectively.



**Figure 7 Number of Anomalies Grouped by Day of Week for Current (Ampere) Dataset 1**



**Figure 8 Number of Anomalies Grouped by Hour for Current (Ampere) Dataset 1**



**Figure 9 Number of Anomalies Grouped by Minute for Current (Ampere) Dataset 1**

# **3 DISCUSSION**

The isolation forest algorithm was used to develop the machine learning models for anomaly detection. Due to the lack of real-world data, synthetic data was generated. A total of 20 datasets was created using Python scripts, 10 each for current and temperature values. The results were satisfactory with all the anomalous data points being identified correctly by the isolation forest models.

The post-model analysis dived deeper into the data and the model results. Relevant features such as “day\_of\_week\_”, “hour”, “minute” were extracted from the “Timestamp” column. Descriptive statistics were computed to obtain a high-level overview of the datasets. The distribution plots provided a quick visual overview of the distribution of the values for both current and temperature. The scatterplot was chosen in place of the line plot as it was able to display the anomalies with little cluttering. The pattern of failure analysis segmented the data temporally to understand when failure was more likely to occur.

# **4 LIMITATIONS**

One of the main limitations is the lack of real-world data. Synthetic data tends to be much more simplistic and clean unlike data generated by physical assets. Therefore, the isolation forest models performed extremely well in detecting the anomalies. However, the performance is expected to be reduced when deployed in real-time settings. Periodic retraining of the models might be necessary as the behavior of the assets change over time and the models’ weights might be suboptimal.

The isolation forest algorithm is also not flawless. It inherits some of the weaknesses from its base model, aka the decision tree. The decision boundary of the isolation forest is limited to the horizontal and vertical axis. This can lead to overfitting as the tree splits more times to isolate data points as compared to if the decision boundary has no direction limitations.

# **TECHNICAL REQUIREMENTS**

**Operating system(s):** Linux, Windows, macOS

**Programming language:** Python

**IDE:** Thonny, Google Colab

**Other requirements:** Python (version >= 3.7.4), Microsoft Excel (View Excel/CSV files)