COBER INSIGHTS™ – DATA ANALYSIS PROJECT

Jana Ciric, Jermaine Clarke, Sneha Dubey, Ligia Galvão, Vinod Katuwa Nanji, Swapna Salvagi

MAY 14, 2020

Table of Contents

Purpose of Project

All systems manufactured by Cober include precision power control, process monitoring, remote diagnostics, and support capability because all Cober equipment collects data remotely through the internet. The abovementioned features allow the tracking of all machines located in Cober clients without the need to provide an employee for this job. Also, this type of monitoring decreases the human error rate at the moment of collecting data and enables the company to get a diverse kind of information per minute. Collected data includes time, temperature, alerts, operational history, product information, and detailed electrical information relevant to the health of the equipment. This remote service is called Internet Diagnostics Online (iDOL) and is installed as a standard part of all Cober oven products.

With iDOL, Cober becomes more than a manufacturing company and starts to add value for its clients with solutions in service and consultations. The full set of services offered by the company is extensive. It includes training, support of Cober and non-Cober systems, remote support, system calibrations, preventative maintenance, and system upgrades. Recently, Cober developed a new data collection, analysis, and alerting system called Cober Insights™. With this new system, the company provides a custom report with historical data, charting, and relevant alerts about the Cober machinery for its customers. Cober Insights might optimize Cober's customer operations allowing the company to deliver a more strategic level of value.

The challenge Cober is facing is delivering this strategic level of value on top of the Cober Insights platform. For this purpose, Cober needs to develop a structure with a strong analytical background and skills in data science to expand the capabilities mentioned. By using knowledge of data science methods including cleaning, anomaly detection, modeling, analytics, aggregates, and A/B testing, the team can attempt to develop a dashboard transforming data into insights aligned with the business.

Goal

Using available data from the collection device eWON-Flexy stored in Microsoft Azure, the team will attempt to develop a predictive data model that will preemptively assist customers and users with failures and breakages. We will also utilize the given information to observe a pattern of device usage that leads to misuse and faster deterioration, thereby helping clients prolong the lifespan of their appliances. Aside from formal models and reports on the data analysis and modeling, we will visualize the obtained results and attempt to, if time permitting, create a model that tracks standard statistics such as average use of the device, the number of alerts, most common alert types, and others.

Resources Available

The team had access to Cober Inc.'s unstructured data in Microsoft Azure Blob Storage. The available records in the abovementioned storage are separated by year, month, day, and hour. We were able to access files that go back to October of 2019. The data from 2019 holds 1,354 blobs. There are 2,750 blob records from 2020, and these include data from January, February, and April. Each blob in the storage consists of a JSON payload with a JSON timestamp on each line. The data available comes from the eWON-Flexy data collection device inside Cober Inc.

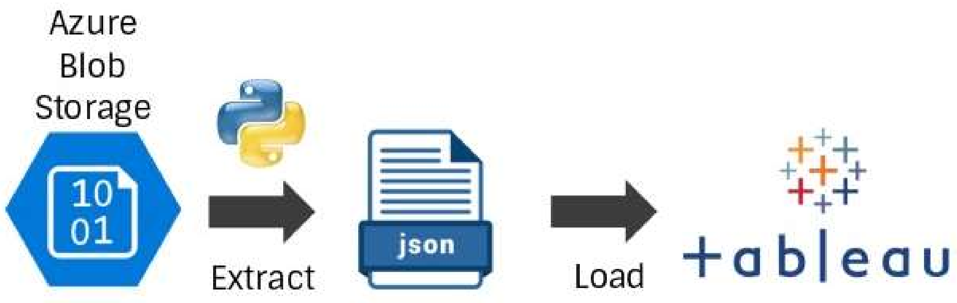
The data is collected every two seconds and includes information such as:

- General information including the
  - Timestamp
  - Owner/Location of the machine, and
  - Unique identifier for the machine.
- Overall state of the system which describes the
  - Machine runtime,
- Number of generators present in the system – e.g., necessary information such as the status of each generator, alarms, and faults, but also more in-depth details like
  - Electrical data: arc detector voltage, magnet current, anode current, anode voltage,
  - Water Temperature,
  - Power information – e.g., forward power, reflected power, set forward power,
  - Emergency stop, and
  - Flow Meter.
- Finally, the data collection device also keeps track of alarm-related information, such as alarm time, description, and status,
- As well as controls, which indicate what the control event is and the time when the control was made.

Tools

We utilized Python to retrieve the data programmatically. Tableau was the designated data analysis software. We conducted brief research on existing data tools and decided that Tableau best fits our needs. Pace University students have free licenses for this software, and Tableau Public, which is free for everyone and easy to download, allows users to view dashboards and visualizations easily. Applications such as WhatsApp, Slack, and Outlook, were used for communication.

Workflow Diagram

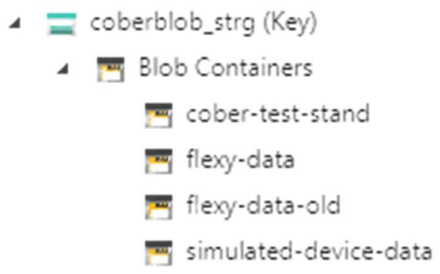


The diagram here illustrates the workflow of our project. The team accesses Cober's Azure Blob Storage, extracts the data programmatically while simultaneously converting it to JSON arrays. Afterward, we load the JSON files into Tableau for visualization and analysis. Each of these steps will be explained in much greater detail below.

Data

Sampling

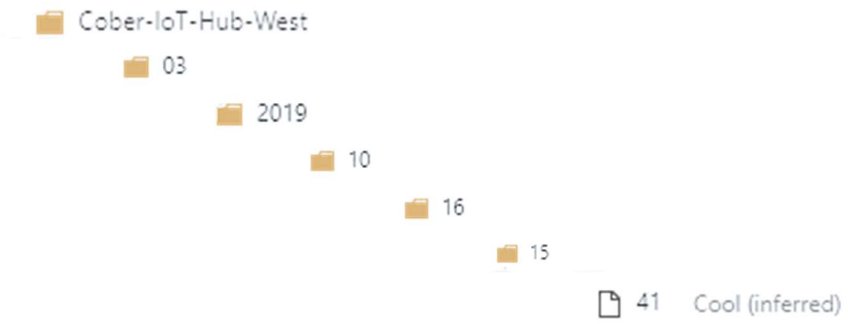
The first step in our data analysis project is to sample the data, as indicated by the workflow diagram. As previously mentioned, Cober's sensor data is stored in Azure. The sensor data is in the form of blob storage, which is Azure's service for storing large amounts of unstructured object data (such as text or binary data). Here is a snapshot from Azure of the data organization:



What is beneficial about Azure Blob storage is that it is accessible from anywhere in the world via HTTP/HTTPS. Objects in the blob storage can be accessed via the REST API, Azure PowerShell, or Azure CLI (command-line interpreter). We can also access Cober's sensor data via the Azure Storage Explorer. The Azure Storage Explorer is a free tool that allows clients to manage their Azure cloud storage resources from any location easily. The storage explorer enables us to connect to Cober's blob storage using a connection string which authorizes us to access this data. Here is a snapshot of the connection string:

```
DefaultEndpointsProtocol=https;AccountName=coberblobstorage;AccountKey=K0TglrFOh8eobGnFx7l7YTQRQjGj2WWgqJwJUO9YuSCGx
XsuhqzV0wkJJqpkTeWs8ROeXLJ1Ki+d4V/zr13FKg==;EndpointSuffix=core.windows.net
```

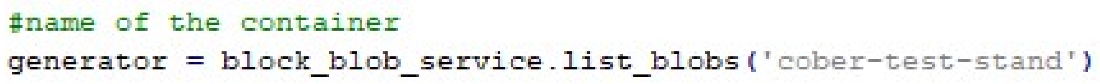
The Azure blob containers hold the sensor data that is organized by year, month, day, hour, and minute. The blobs are grouped in folders exhibiting a structure similar to the one below:



After sampling the data, we proceed with the ETL (extract, transform, and load) phase.

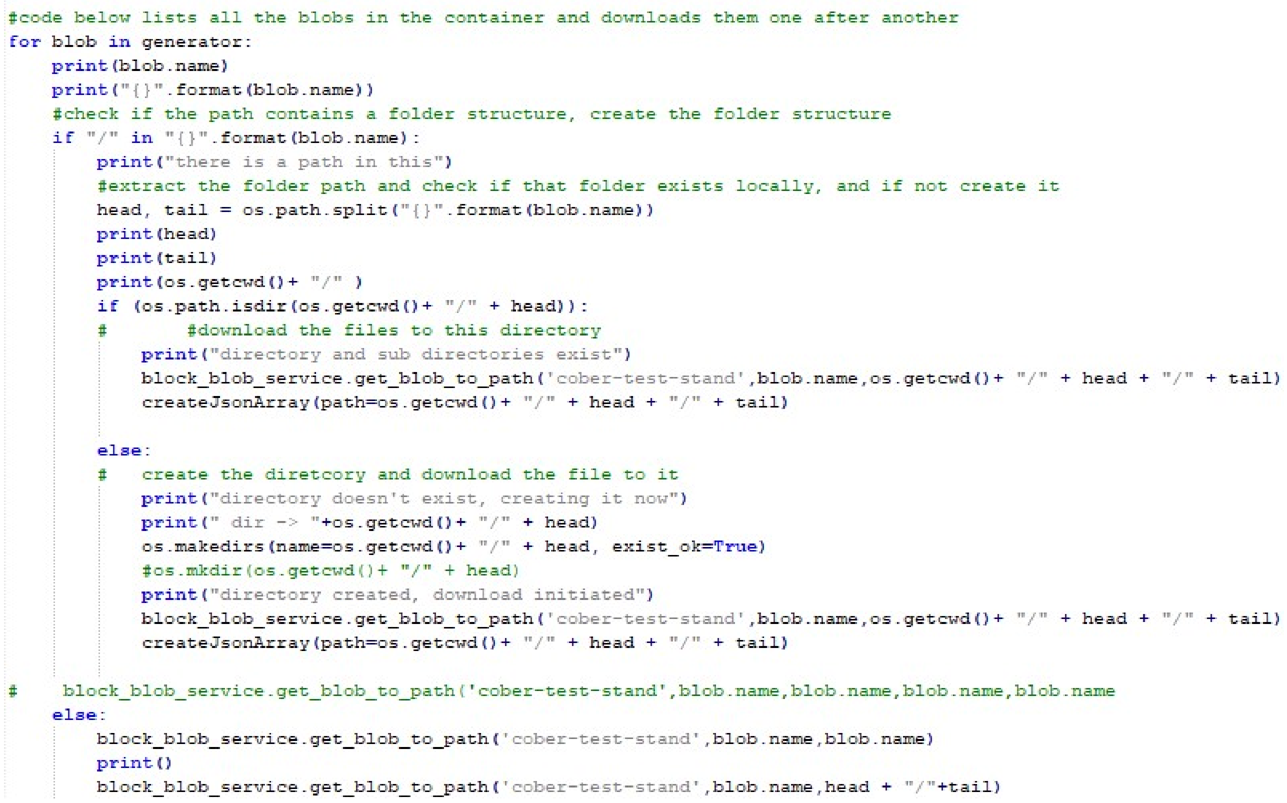
Extraction

The data from Azure is retrieved on to a team member's local machine using a Python script. Cober-test-stand is the particular blob container from which we are downloading the data, which is shown in the snippet from the Python script below:



```
#name of the container
generator = block_blob_service.list_blobs('cober-test-stand')
```

The following code fragment represents how the team is downloading all of the blobs from the container, one after another. The folder structure that is present in Cober's Azure storage is preserved during the download.

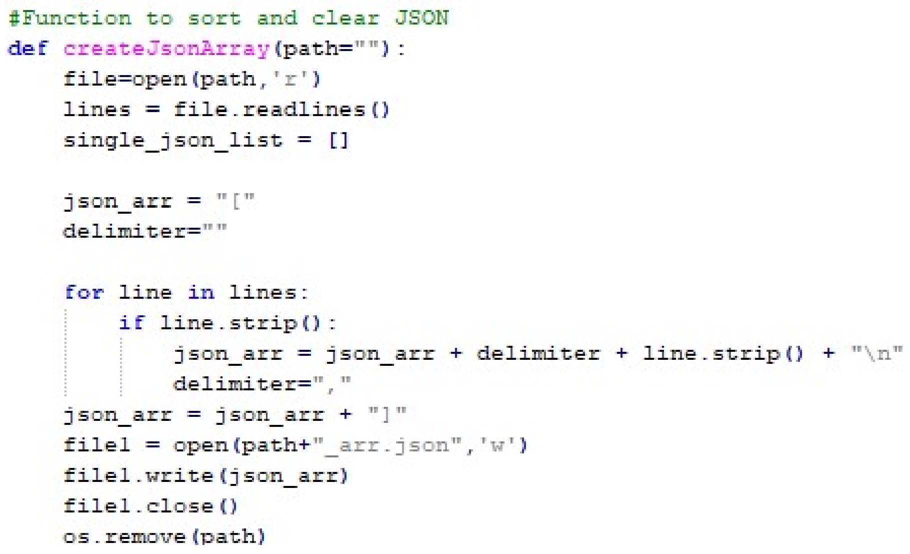


```
#code below lists all the blobs in the container and downloads them one after another
for blob in generator:
    print(blob.name)
    print("{}".format(blob.name))
    #check if the path contains a folder structure, create the folder structure
    if "/" in "{}".format(blob.name):
        print("there is a path in this")
        #extract the folder path and check if that folder exists locally, and if not create it
        head, tail = os.path.split("{}".format(blob.name))
        print(head)
        print(tail)
        print(os.getcwd()+ "/" )
        if (os.path.isdir(os.getcwd()+ "/" + head)):
        #       #download the files to this directory
            print("directory and sub directories exist")
            block_blob_service.get_blob_to_path('cober-test-stand',blob.name,os.getcwd()+ "/" + head + "/" + tail)
            createJsonArray(path=os.getcwd()+ "/" + head + "/" + tail)

        else:
        #   create the diretcory and download the file to it
            print("directory doesn't exist, creating it now")
            print(" dir -> "+os.getcwd()+ "/" + head)
            os.makedirs(name=os.getcwd()+ "/" + head, exist_ok=True)
            #os.mkdir(os.getcwd()+ "/" + head)
            print("directory created, download initiated")
            block_blob_service.get_blob_to_path('cober-test-stand',blob.name,os.getcwd()+ "/" + head + "/" + tail)
            createJsonArray(path=os.getcwd()+ "/" + head + "/" + tail)

#    block_blob_service.get_blob_to_path('cober-test-stand',blob.name,blob.name,blob.name,blob.name
    else:
        block_blob_service.get_blob_to_path('cober-test-stand',blob.name,blob.name)
        print()
        block_blob_service.get_blob_to_path('cober-test-stand',blob.name,head + "/"+tail)
```

Blobs, since they contain JSON messages, are simultaneously converted into JSON arrays using a helper function called createJsonArray.



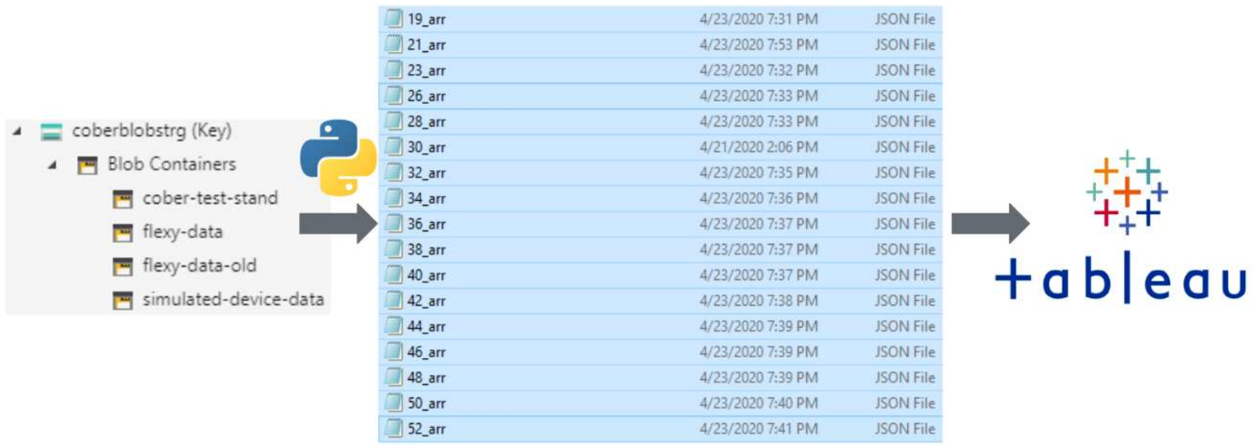
```
#Function to sort and clear JSON
def createJsonArray(path=""):
    file=open(path,'r')
    lines = file.readlines()
    single_json_list = []

    json_arr = "["
    delimiter=""

    for line in lines:
        if line.strip():
            json_arr = json_arr + delimiter + line.strip() + "\n"
            delimiter=","
    json_arr = json_arr + "]"
    file1 = open(path+"_arr.json",'w')
    file1.write(json_arr)
    file1.close()
    os.remove(path)
```

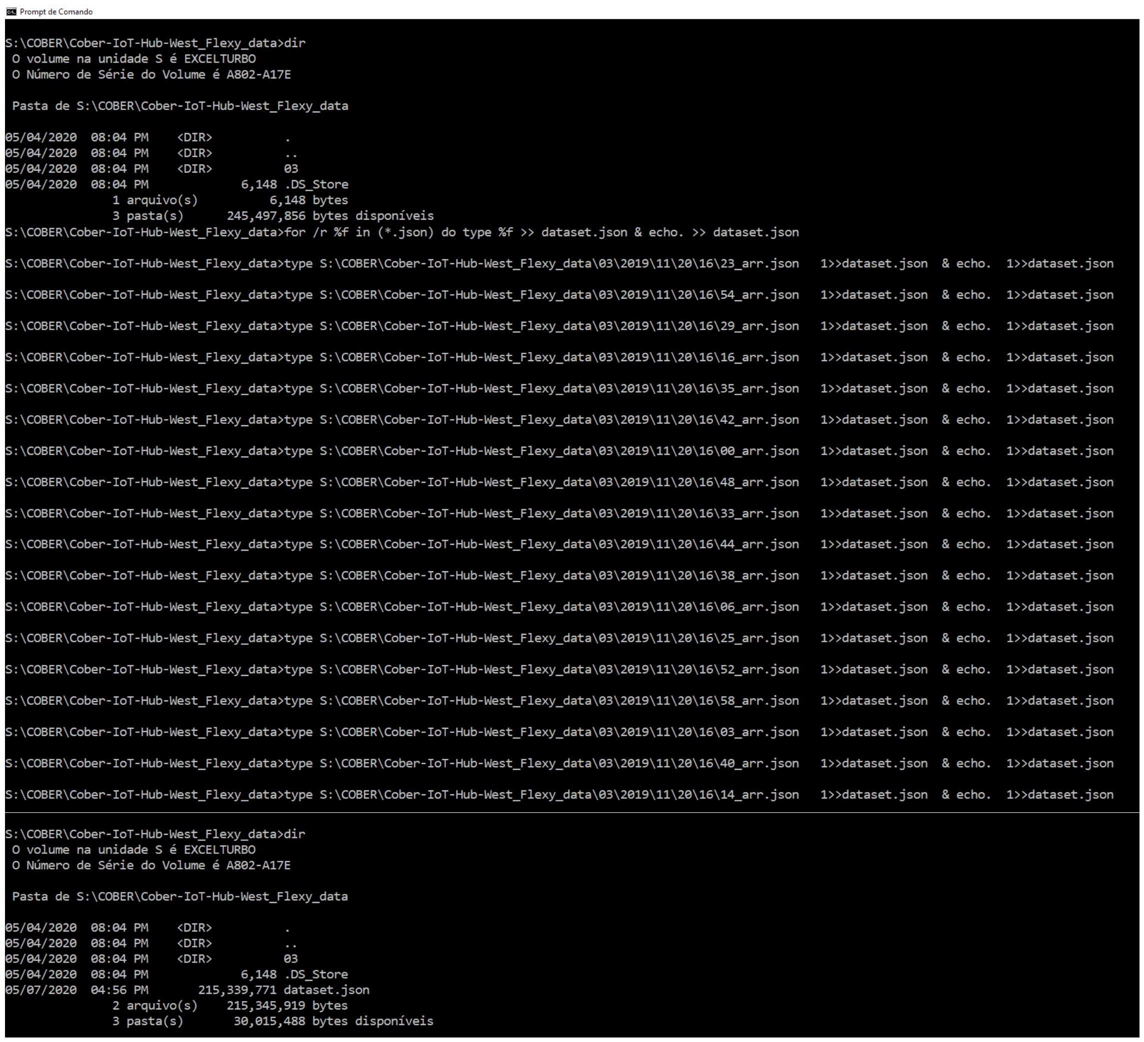
Preparation

Data preparation is the process of cleaning and transforming raw data before processing and analysis. It is an important step and often involves reformatting data, making corrections to data, and the combining of data sets to enrich data.



19_arr	4/23/2020 7:31 PM	JSON File
21_arr	4/23/2020 7:53 PM	JSON File
23_arr	4/23/2020 7:32 PM	JSON File
26_arr	4/23/2020 7:33 PM	JSON File
28_arr	4/23/2020 7:33 PM	JSON File
30_arr	4/21/2020 2:06 PM	JSON File
32_arr	4/23/2020 7:35 PM	JSON File
34_arr	4/23/2020 7:36 PM	JSON File
36_arr	4/23/2020 7:37 PM	JSON File
38_arr	4/23/2020 7:37 PM	JSON File
40_arr	4/23/2020 7:37 PM	JSON File
42_arr	4/23/2020 7:38 PM	JSON File
44_arr	4/23/2020 7:39 PM	JSON File
46_arr	4/23/2020 7:39 PM	JSON File
48_arr	4/23/2020 7:39 PM	JSON File
50_arr	4/23/2020 7:40 PM	JSON File
52_arr	4/23/2020 7:41 PM	JSON File

The graph above illustrates the process of data preparation performed by our team. As previously mentioned, we reformat the data from blob to JSON using Python. Afterward, we clean up the data, which will be described in the section below. Finally, before loading the data up in Tableau, we merge the JSON files into one unique folder. Tableau does not support loading thousands of files spread across many folders simultaneously. To overcome this issue, we run a command prompt script to aggregate all data in a single folder. The figure below illustrates this.



```
Prompt de Comando

S:\COBER\Cober-IoT-Hub-West_Flexy_data>dir
 O volume na unidade S é EXCELTURBO
 O Número de Série do Volume é A802-A17E

 Pasta de S:\COBER\Cober-IoT-Hub-West_Flexy_data

05/04/2020  08:04 PM    <DIR>          .
05/04/2020  08:04 PM    <DIR>          ..
05/04/2020  08:04 PM    <DIR>          03
05/04/2020  08:04 PM             6,148 .DS_Store
               1 arquivo(s)          6,148 bytes
               3 pasta(s)      245,497,856 bytes disponíveis
S:\COBER\Cober-IoT-Hub-West_Flexy_data>for /r %f in (*.json) do type %f >> dataset.json & echo. >> dataset.json

S:\COBER\Cober-IoT-Hub-West_Flexy_data>type S:\COBER\Cober-IoT-Hub-West_Flexy_data\03\2019\11\20\16\23_arr.json   1>>dataset.json  & echo.  1>>dataset.json

S:\COBER\Cober-IoT-Hub-West_Flexy_data>type S:\COBER\Cober-IoT-Hub-West_Flexy_data\03\2019\11\20\16\54_arr.json   1>>dataset.json  & echo.  1>>dataset.json

S:\COBER\Cober-IoT-Hub-West_Flexy_data>type S:\COBER\Cober-IoT-Hub-West_Flexy_data\03\2019\11\20\16\29_arr.json   1>>dataset.json  & echo.  1>>dataset.json

S:\COBER\Cober-IoT-Hub-West_Flexy_data>type S:\COBER\Cober-IoT-Hub-West_Flexy_data\03\2019\11\20\16\16_arr.json   1>>dataset.json  & echo.  1>>dataset.json

S:\COBER\Cober-IoT-Hub-West_Flexy_data>type S:\COBER\Cober-IoT-Hub-West_Flexy_data\03\2019\11\20\16\35_arr.json   1>>dataset.json  & echo.  1>>dataset.json

S:\COBER\Cober-IoT-Hub-West_Flexy_data>type S:\COBER\Cober-IoT-Hub-West_Flexy_data\03\2019\11\20\16\42_arr.json   1>>dataset.json  & echo.  1>>dataset.json

S:\COBER\Cober-IoT-Hub-West_Flexy_data>type S:\COBER\Cober-IoT-Hub-West_Flexy_data\03\2019\11\20\16\00_arr.json   1>>dataset.json  & echo.  1>>dataset.json

S:\COBER\Cober-IoT-Hub-West_Flexy_data>type S:\COBER\Cober-IoT-Hub-West_Flexy_data\03\2019\11\20\16\48_arr.json   1>>dataset.json  & echo.  1>>dataset.json

S:\COBER\Cober-IoT-Hub-West_Flexy_data>type S:\COBER\Cober-IoT-Hub-West_Flexy_data\03\2019\11\20\16\33_arr.json   1>>dataset.json  & echo.  1>>dataset.json

S:\COBER\Cober-IoT-Hub-West_Flexy_data>type S:\COBER\Cober-IoT-Hub-West_Flexy_data\03\2019\11\20\16\44_arr.json   1>>dataset.json  & echo.  1>>dataset.json

S:\COBER\Cober-IoT-Hub-West_Flexy_data>type S:\COBER\Cober-IoT-Hub-West_Flexy_data\03\2019\11\20\16\38_arr.json   1>>dataset.json  & echo.  1>>dataset.json

S:\COBER\Cober-IoT-Hub-West_Flexy_data>type S:\COBER\Cober-IoT-Hub-West_Flexy_data\03\2019\11\20\16\06_arr.json   1>>dataset.json  & echo.  1>>dataset.json

S:\COBER\Cober-IoT-Hub-West_Flexy_data>type S:\COBER\Cober-IoT-Hub-West_Flexy_data\03\2019\11\20\16\25_arr.json   1>>dataset.json  & echo.  1>>dataset.json

S:\COBER\Cober-IoT-Hub-West_Flexy_data>type S:\COBER\Cober-IoT-Hub-West_Flexy_data\03\2019\11\20\16\52_arr.json   1>>dataset.json  & echo.  1>>dataset.json

S:\COBER\Cober-IoT-Hub-West_Flexy_data>type S:\COBER\Cober-IoT-Hub-West_Flexy_data\03\2019\11\20\16\58_arr.json   1>>dataset.json  & echo.  1>>dataset.json

S:\COBER\Cober-IoT-Hub-West_Flexy_data>type S:\COBER\Cober-IoT-Hub-West_Flexy_data\03\2019\11\20\16\03_arr.json   1>>dataset.json  & echo.  1>>dataset.json

S:\COBER\Cober-IoT-Hub-West_Flexy_data>type S:\COBER\Cober-IoT-Hub-West_Flexy_data\03\2019\11\20\16\40_arr.json   1>>dataset.json  & echo.  1>>dataset.json

S:\COBER\Cober-IoT-Hub-West_Flexy_data>type S:\COBER\Cober-IoT-Hub-West_Flexy_data\03\2019\11\20\16\14_arr.json   1>>dataset.json  & echo.  1>>dataset.json

S:\COBER\Cober-IoT-Hub-West_Flexy_data>dir
 O volume na unidade S é EXCELTURBO
 O Número de Série do Volume é A802-A17E

 Pasta de S:\COBER\Cober-IoT-Hub-West_Flexy_data

05/04/2020  08:04 PM    <DIR>          .
05/04/2020  08:04 PM    <DIR>          ..
05/04/2020  08:04 PM    <DIR>          03
05/04/2020  08:04 PM             6,148 .DS_Store
05/07/2020  04:56 PM       215,339,771 dataset.json
               2 arquivo(s)    215,345,919 bytes
               3 pasta(s)       30,015,488 bytes disponíveis
```

It is incredibly helpful that we have constant and direct access to Cober's Azure Storage. The script that retrieves the data can easily be modified to automatically overwrite and download new files when they get updated in Blob Storage. That way, we can always have the most accurate and recent data available.

Cleaning

Data cleaning is the process of detecting and correcting (or removing) corrupt, irrelevant, or inaccurate records from a record-set, table, or dataset. Initially, we filtered Cober's sensor data based on the values of Set_Forward_Power, Forward_Power, and Reflected_Power. However, after consultations with the company, we have entirely filtered down the data to just focus on three key dates that exhibit greater variance in comparison to other data points – October 24, 2019, December 12, 2019, and January 16, 2020.

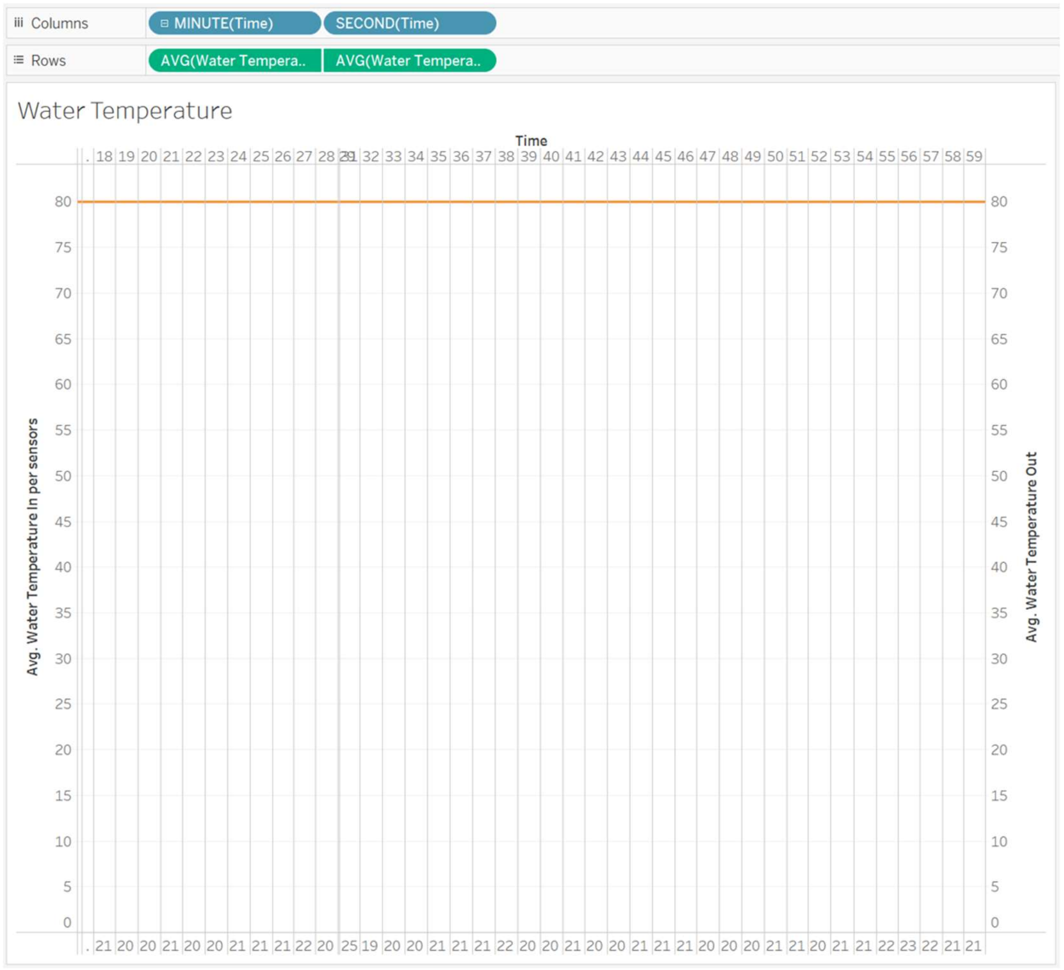
Manipulation

After the data is downloaded locally and the files are flattened into one folder, it is loaded up in Tableau for visualization.

Tableau

Tableau is a data software that makes it easier to create interactive visual analytics in the form of dashboards. Tableau effectively simplifies raw data in various formats into understandable representations. Data can easily be manipulated during visualizations, such as when creating groups, trends, models, and reference lines. Data analysis is very fast with Tableau, and the visualizations created are in the form of worksheets and dashboards. These visualizations help professionals understand the data and make better decisions.

Tableau supports the ability to connect to a wide variety of data types and storage locations. It allows us to connect to different (and many) JSON files, upload them, and create a single worksheet with all of the information combined. When we loaded the data initially, we kept getting results like the following snapshot:

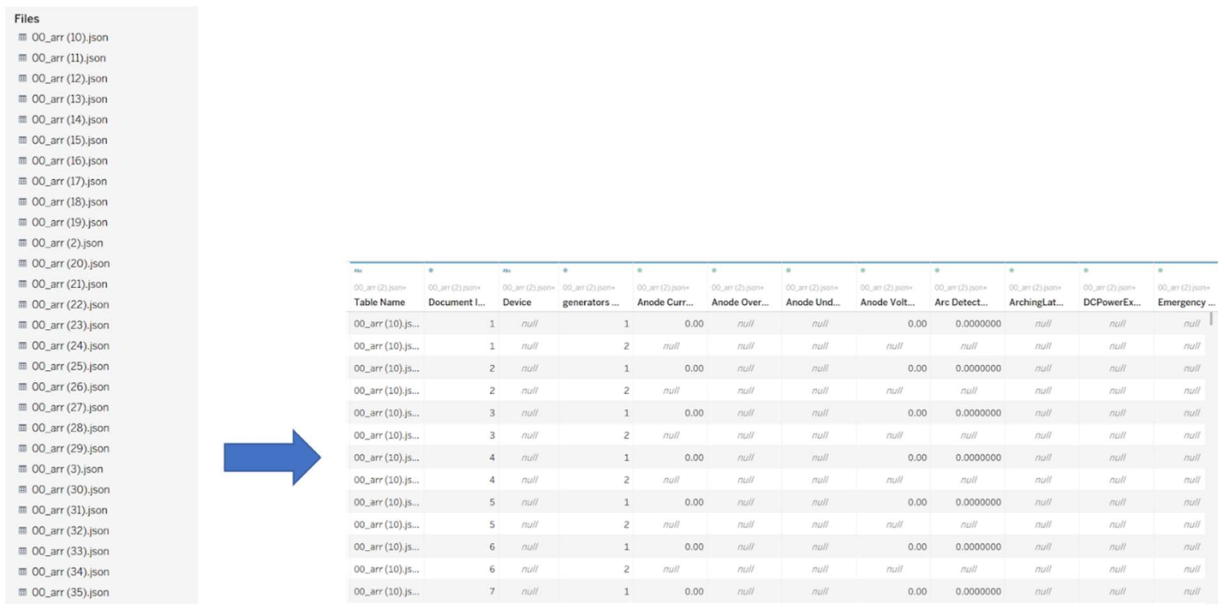


Limitations

As the flat-line indicates, the data that we were able to retrieve is very uniform, and we were unable to observe any extreme values or even any significant changes to the existing ones (as the graph above indicates). It seems as if everything is operating the way it is supposed to be. Lack of variability is significantly impacting our ability to predict potential faults, issues, and warnings. In order to overcome these limitations, we shifted our attention to data points that have more significant variability, such as the ones from October 24, 2019, December 12, 2019, and January 16, 2020.

Visualization, Analysis, and Results

Like previously mentioned, we used Tableau for data visualization and analysis. Tableau enabled our team to upload many distinct JSON files and then to create a single worksheet with all of the information condensed in one place. We have uploaded more than 4,000 JSON files into Tableau and combined them into a unique, single spreadsheet in under 2 minutes. The snapshot below illustrates this:



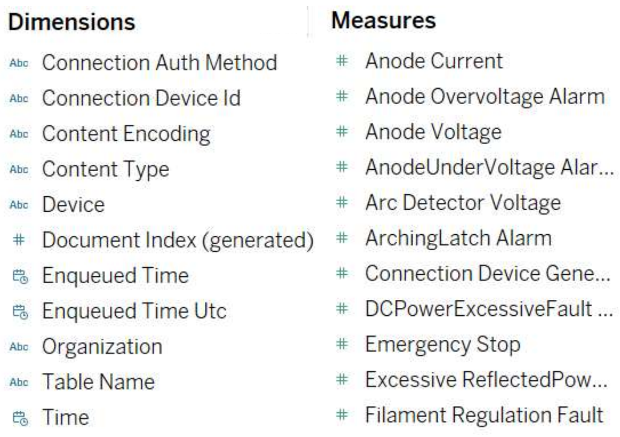
Files
- 00_arr (10).json
- 00_arr (11).json
- 00_arr (12).json
- 00_arr (13).json
- 00_arr (14).json
- 00_arr (15).json
- 00_arr (16).json
- 00_arr (17).json
- 00_arr (18).json
- 00_arr (19).json
- 00_arr (2).json
- 00_arr (20).json
- 00_arr (21).json
- 00_arr (22).json
- 00_arr (23).json
- 00_arr (24).json
- 00_arr (25).json
- 00_arr (26).json
- 00_arr (27).json
- 00_arr (28).json
- 00_arr (29).json
- 00_arr (3).json
- 00_arr (30).json
- 00_arr (31).json
- 00_arr (32).json
- 00_arr (33).json
- 00_arr (34).json
- 00_arr (35).json

00_arr (2).json+	00_arr (2).json+	00_arr (2).json+	00_arr (2).json+	00_arr (2).json+	00_arr (2).json+	00_arr (2).json+	00_arr (2).json+	00_arr (2).json+	00_arr (2).json+	00_arr (2).json+	00_arr (2).json+
Table Name	Document I...	Device	generators ...	Anode Curr...	Anode Over...	Anode Und...	Anode Volt...	Arc Detect...	ArchingLat...	DCPowerEx...	Emergency ...
00_arr (10).js...	1	null	1	0.00	null	null	0.00	0.0000000	null	null	null
00_arr (10).js...	1	null	2	null	null	null	null	null	null	null	null
00_arr (10).js...	2	null	1	0.00	null	null	0.00	0.0000000	null	null	null
00_arr (10).js...	2	null	2	null	null	null	null	null	null	null	null
00_arr (10).js...	3	null	1	0.00	null	null	0.00	0.0000000	null	null	null
00_arr (10).js...	3	null	2	null	null	null	null	null	null	null	null
00_arr (10).js...	4	null	1	0.00	null	null	0.00	0.0000000	null	null	null
00_arr (10).js...	4	null	2	null	null	null	null	null	null	null	null
00_arr (10).js...	5	null	1	0.00	null	null	0.00	0.0000000	null	null	null
00_arr (10).js...	5	null	2	null	null	null	null	null	null	null	null
00_arr (10).js...	6	null	1	0.00	null	null	0.00	0.0000000	null	null	null
00_arr (10).js...	6	null	2	null	null	null	null	null	null	null	null
00_arr (10).js...	7	null	1	0.00	null	null	0.00	0.0000000	null	null	null

Tableau reads all data and splits it into two main categories. The two categories are:

- Dimensions - usually those fields that cannot be aggregated and are used for row or column headings, and
- Measures - fields that can be measured, aggregated, or used for mathematical operations. Measures are often used for plotting or giving values to the sizes of markers.

When a new data source connects to Tableau, fields in the data source are automatically assigned as dimensions or measures in the data pane, depending on the type of information the field contains. These two fields are used to build views of the data. Dimensions include qualitative values such as names, dates, or geographical data. We can use dimensions to categorize, segment, and reveal the details in the data. Dimensions affect the level of detail in the view. Measures contain numeric, quantitative values that we can measure. Two columns below display a portion of dimensions and measures for the Cober data in Tableau.



Dimensions	Measures
Connection Auth Method	Anode Current
Connection Device Id	Anode Overvoltage Alarm
Content Encoding	Anode Voltage
Content Type	AnodeUnderVoltage Alar...
Device	Arc Detector Voltage
Document Index (generated)	ArchingLatch Alarm
Enqueued Time	Connection Device Gene...
Enqueued Time Utc	DCPowerExcessiveFault ...
Organization	Emergency Stop
Table Name	Excessive ReflectedPow...
Time	Filament Regulation Fault

In order to create visualizations, dimensions and measures need to be dragged to the rows and columns tabs in the view pane. The rows and columns are interchangeable and do not impact the original data source. Once a field is dragged to either rows or columns, it can be changed to a measure just by clicking the field and choosing Measure. This neat feature was used to convert Enqueued_Time field from a Dimension to a Measure. This conversion was done because it allowed us to visualize the data in a chronological order (based on date) as a continuous axis, rather than a table or a bar chart (since dimensions, which are discrete, are usually shown as tables). The example beneath illustrates this difference:

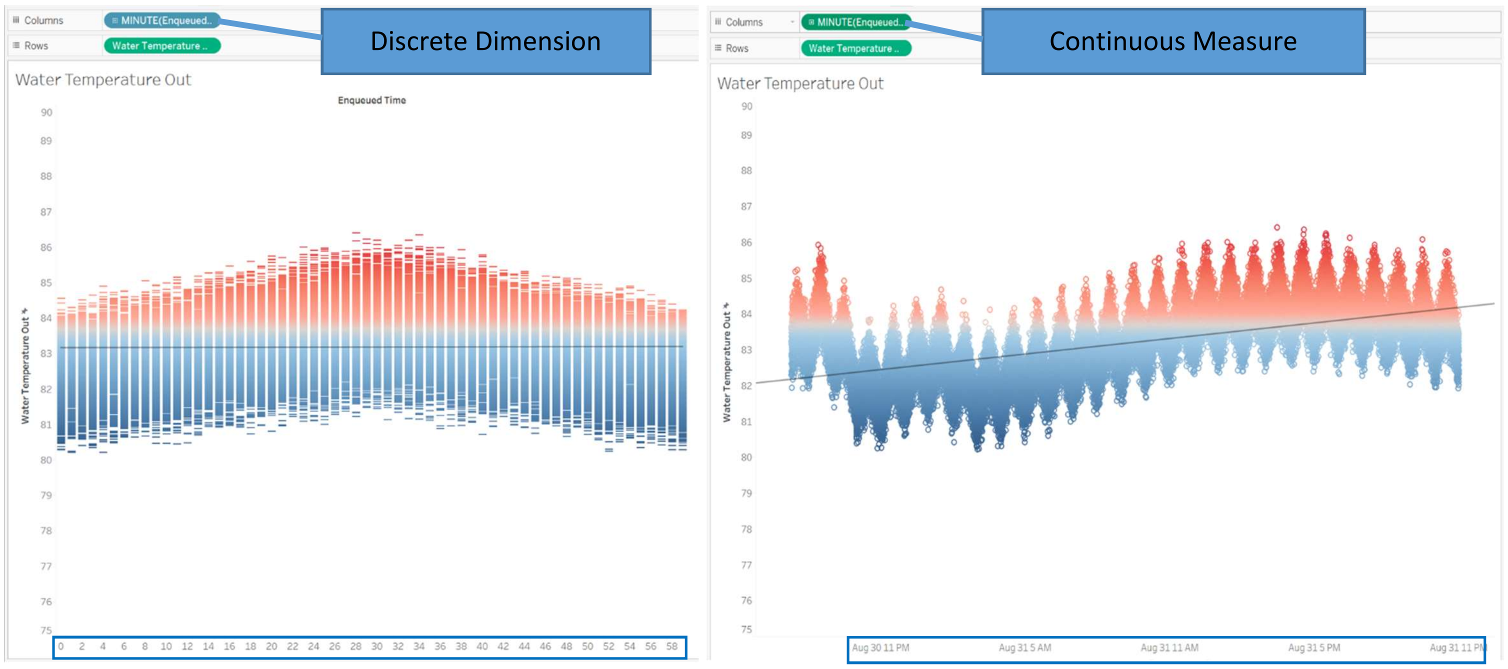
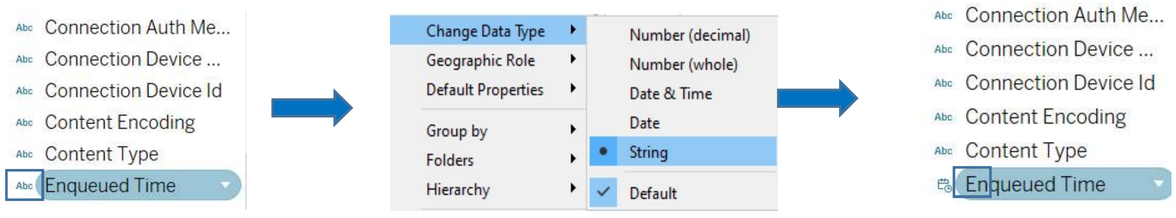
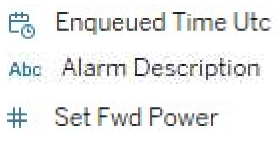


Tableau automatically picks up the type of the field from the data source, but it also allows its clients and users to switch the type by accessing the menu options. For instance, the Enqueued_Time field was initially picked up as a string; however, thanks to this feature, we were able to change this field to a date & time type.

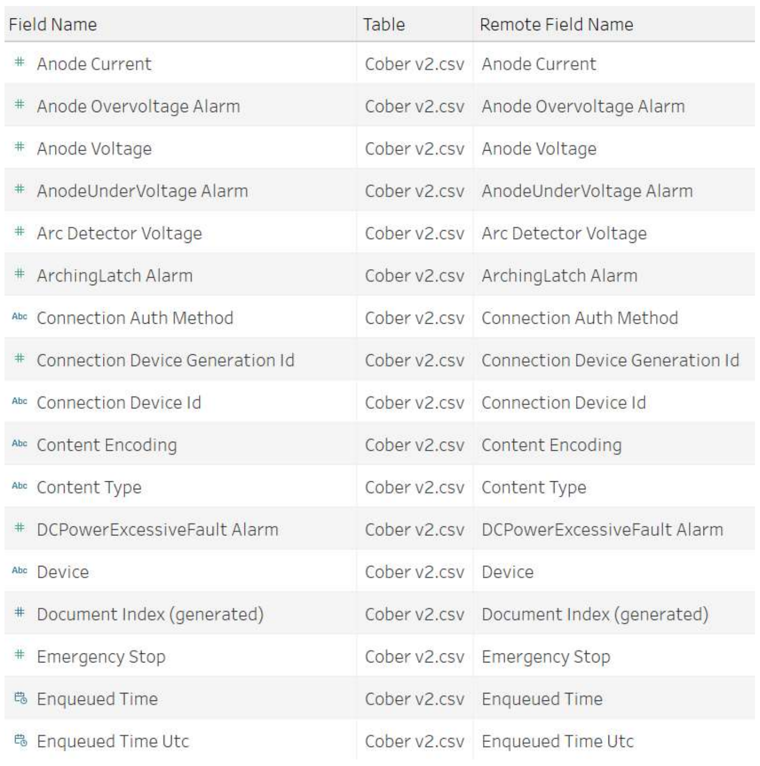


Cober's dataset contains plenty of real-time information about its systems. All of the information relevant to the health of the equipment is collected every two seconds. This data includes water temperature, water flow, emergency stop, electrical data, and power details, among others. All of the information is stored in separate fields, and the fields are of different types and formats, such as date & time, number, and string. We observe this in Tableau as:



Enqueued Time Utc
Abc Alarm Description
Set Fwd Power

A more in-depth overview of the types of these fields can be seen below:

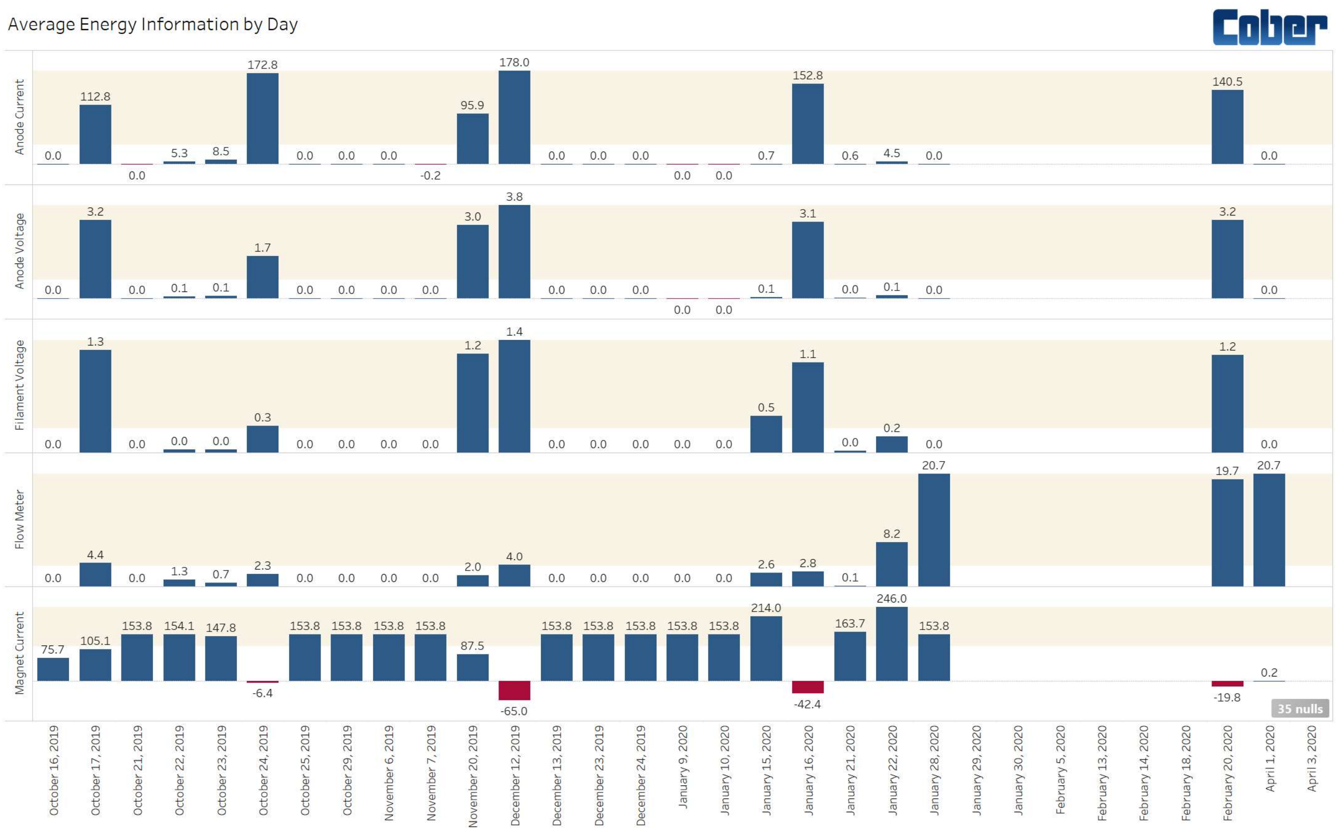


Field Name	Table	Remote Field Name
# Anode Current	Cober v2.csv	Anode Current
# Anode Overvoltage Alarm	Cober v2.csv	Anode Overvoltage Alarm
# Anode Voltage	Cober v2.csv	Anode Voltage
# AnodeUnderVoltage Alarm	Cober v2.csv	AnodeUnderVoltage Alarm
# Arc Detector Voltage	Cober v2.csv	Arc Detector Voltage
# ArchingLatch Alarm	Cober v2.csv	ArchingLatch Alarm
Abc Connection Auth Method	Cober v2.csv	Connection Auth Method
# Connection Device Generation Id	Cober v2.csv	Connection Device Generation Id
Abc Connection Device Id	Cober v2.csv	Connection Device Id
Abc Content Encoding	Cober v2.csv	Content Encoding
Abc Content Type	Cober v2.csv	Content Type
# DCPowerExcessiveFault Alarm	Cober v2.csv	DCPowerExcessiveFault Alarm
Abc Device	Cober v2.csv	Device
# Document Index (generated)	Cober v2.csv	Document Index (generated)
# Emergency Stop	Cober v2.csv	Emergency Stop
Enqueued Time	Cober v2.csv	Enqueued Time
Enqueued Time Utc	Cober v2.csv	Enqueued Time Utc

Results

Energy Information

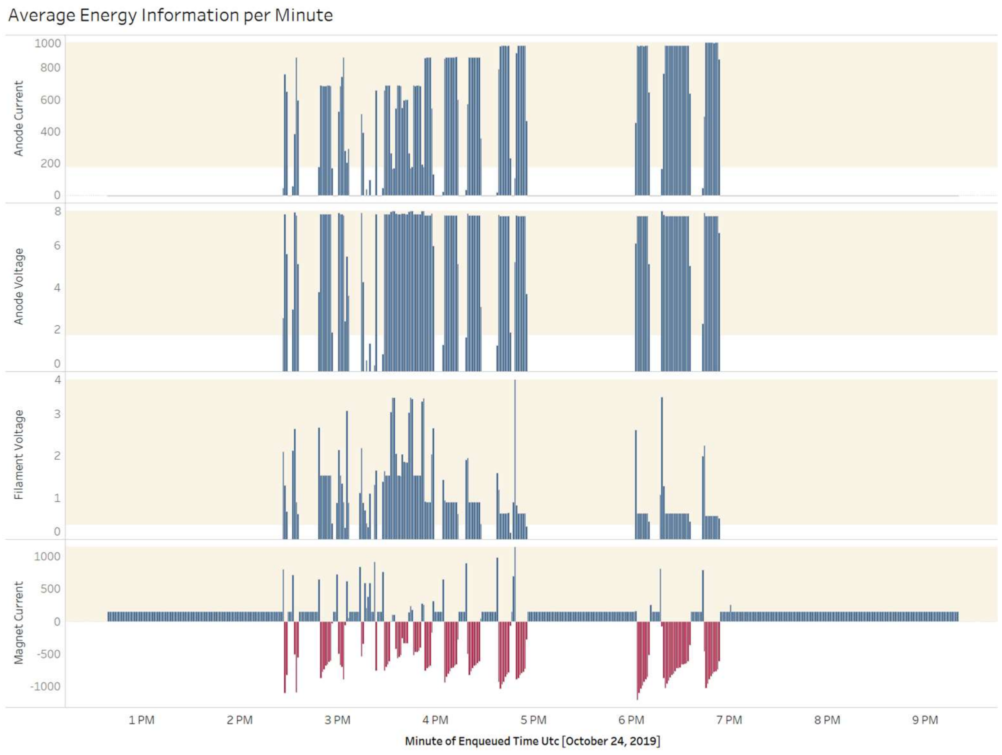
A portion of our focus was shifted towards the values of the anode and the cathode since we know they play an essential role in the operation of microwave ovens. Namely, microwave ovens operate by producing electromagnetic waves, which are generated by ejecting electrons from a heated conductive filament. These electrons are accelerated between a cathode and an anode. The graph below shows the relationship between the anode and the cathode in Cober's test machine.



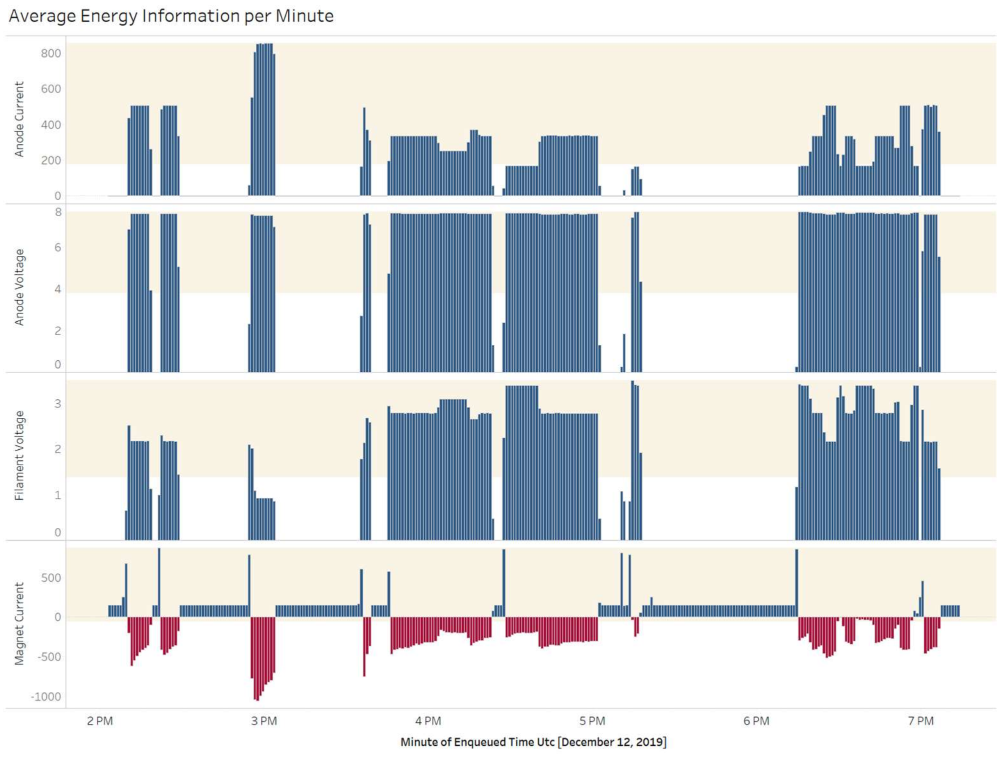
A negative result can be observed in the data from the following dates: October 24, 2019, December 12, 2019, January 16, 2020, and February 20, 2020. On these days, the energy values for Anode Voltage, Anode Current, and Filament Voltage are more significant than average.

A deep-dive analysis of the data for these three dates per minute shows that magnet current has negative values while these three parameters are peaking.

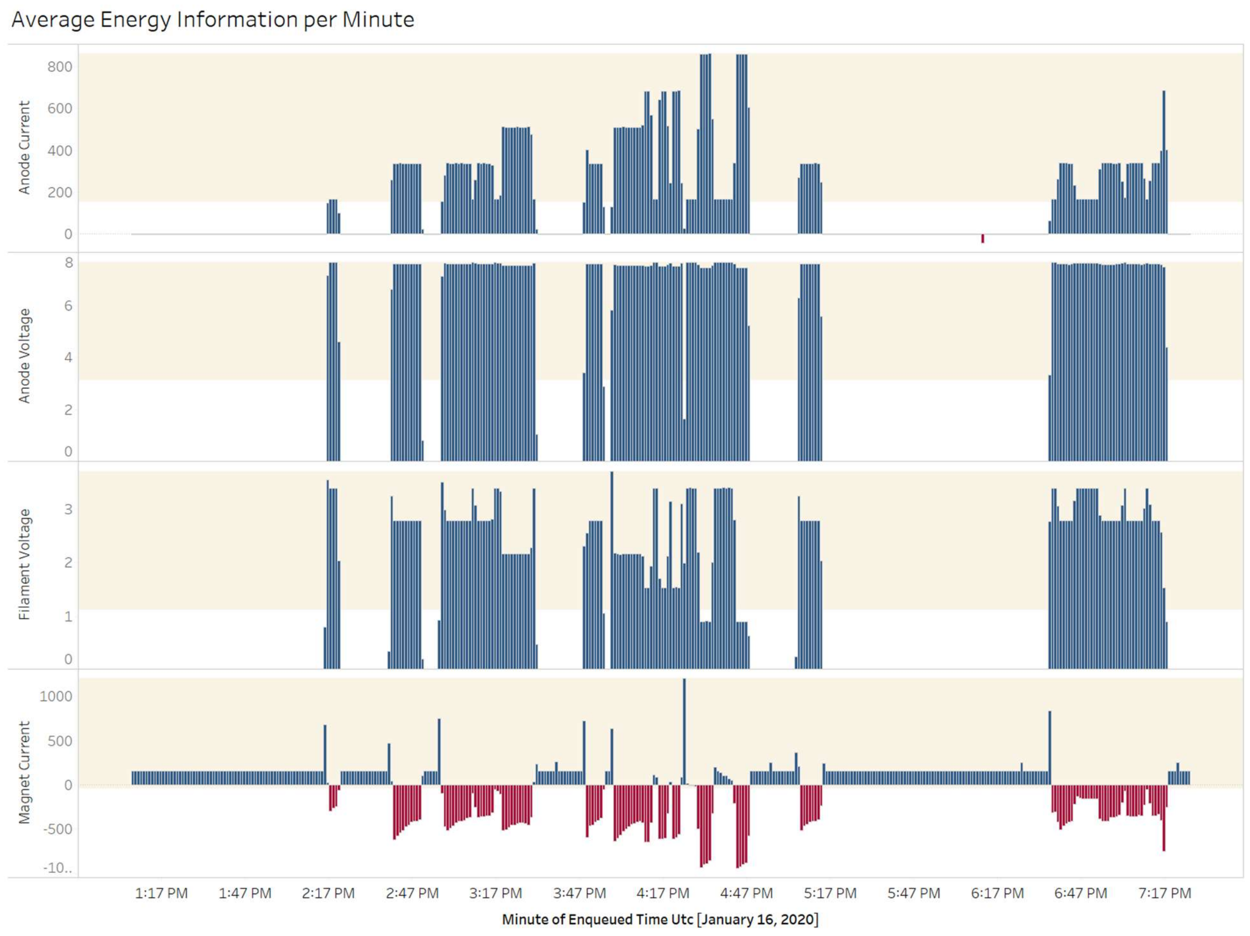
October 24, 2019



December 12, 2019



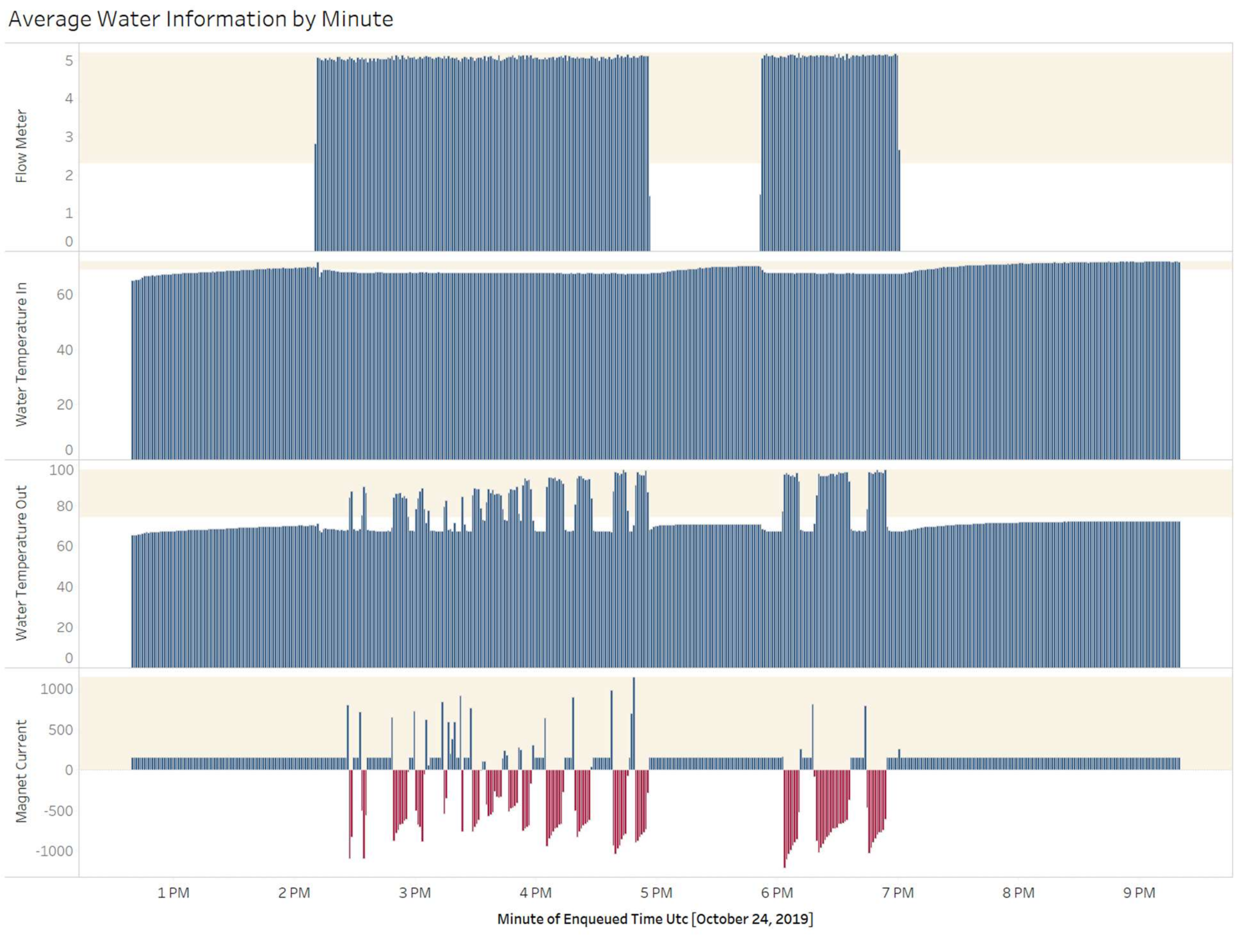
January 16, 2020



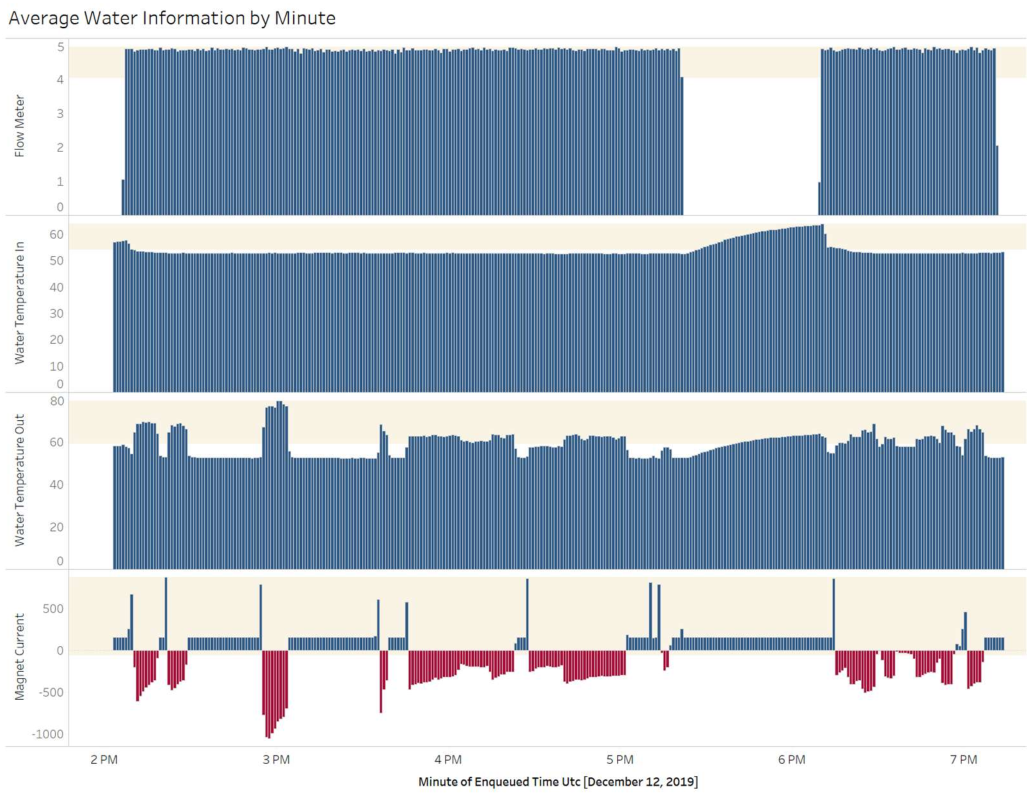
Water Information

By observing the water data, we can see the correlation between water temperature and the flow meter. Both Water_Temperature_In and Water_Temperature_Out sharply increase when the flow meter is equal to zero, indicating a strong relationship between those factors**.** Another observation is that Water_Temperature_Out always increases when the magnet current becomes negative. The same correlation cannot be seen for Water_Temperature_In. This parameter is mainly flat, showing minimal changes, primarily when the flow meter shifts to zero.

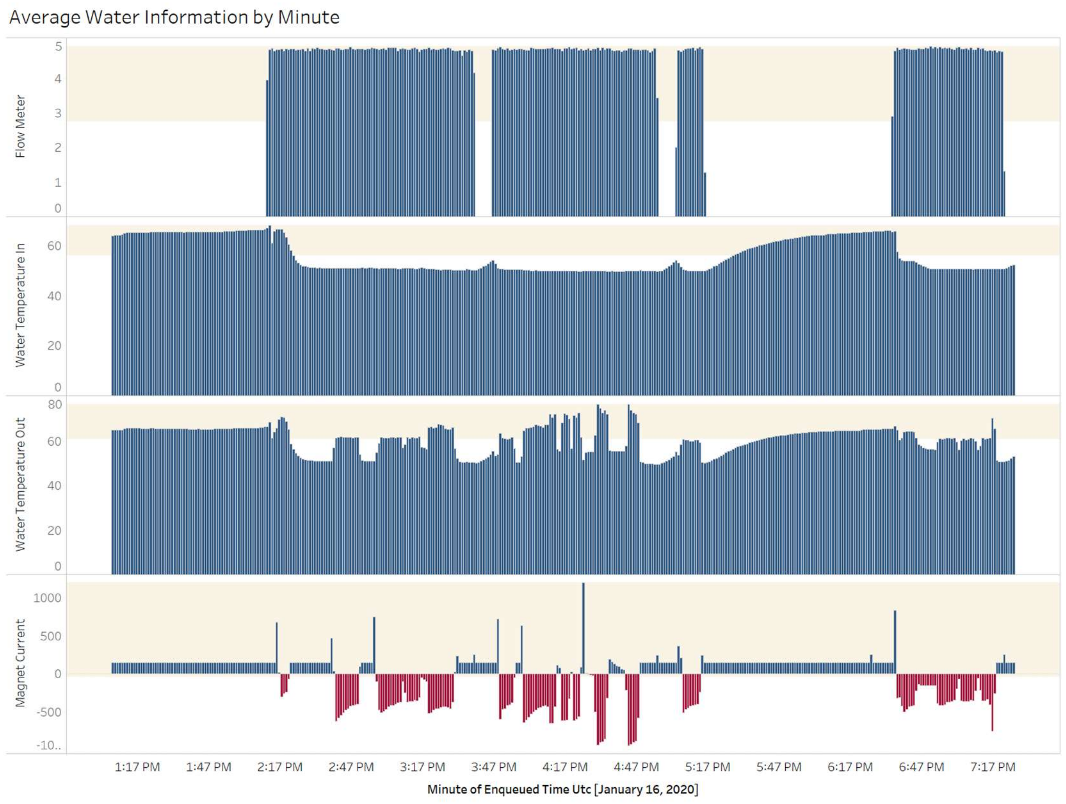
October 24, 2019



December 12, 2019

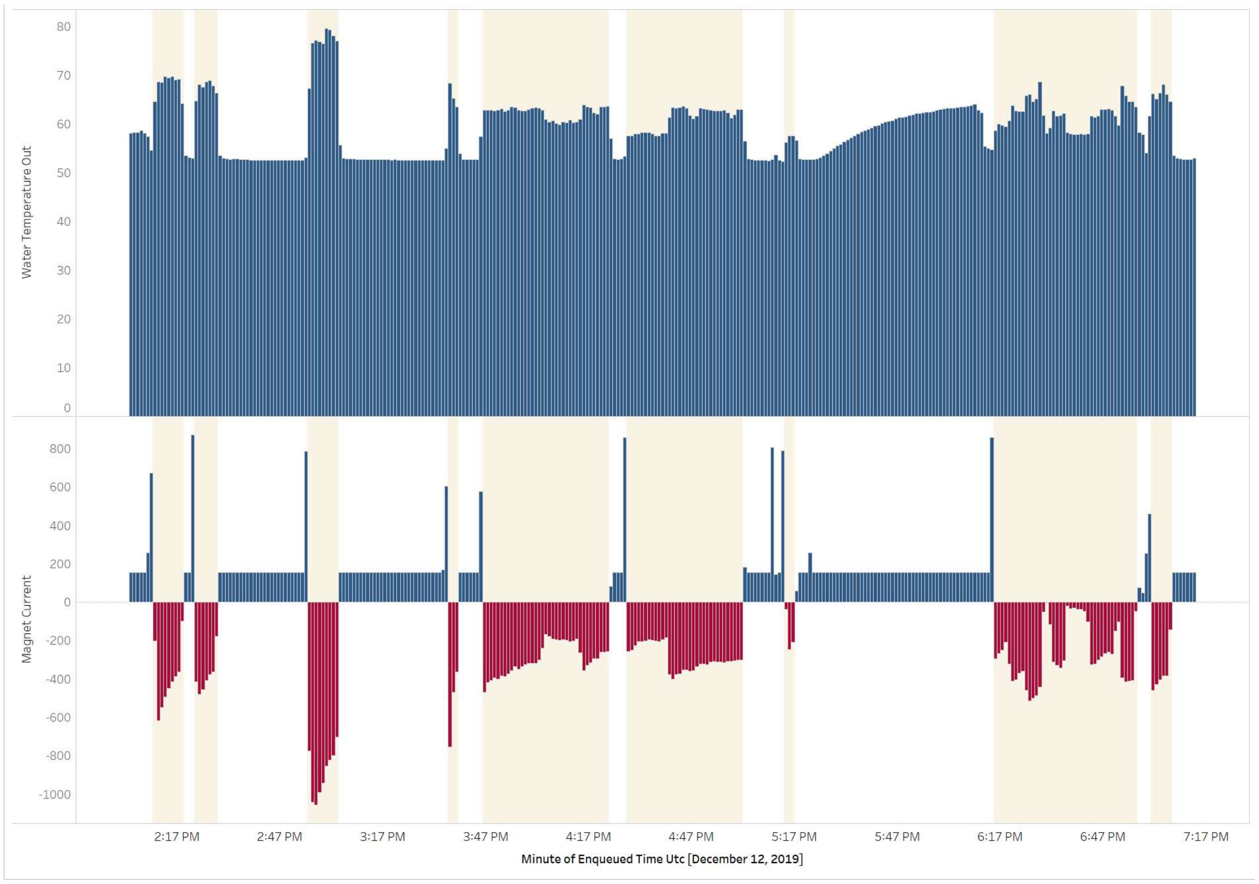


January 16, 2020



In order to further explore the relationship of Water_Temperature_Out with the magnet current, we applied linear regression on the data points from the three key dates – October 24, 2019, December 12, 2019, and January 16, 2020. Using this, we can see that the Water_Temperature_Out increases when the magnet current becomes negative, which is shown in the graph below.

Linear regression analysis generates an equation that describes the statistical relationship between one or more predictor variables (Water_Temperature_Out) and the response variable (magnet current). Linear Regression finds the line that best represents the input variable(s) with the output variable. Based on the data, the dispersion diagram is constructed; it must exhibit a linear trend to use linear regression.



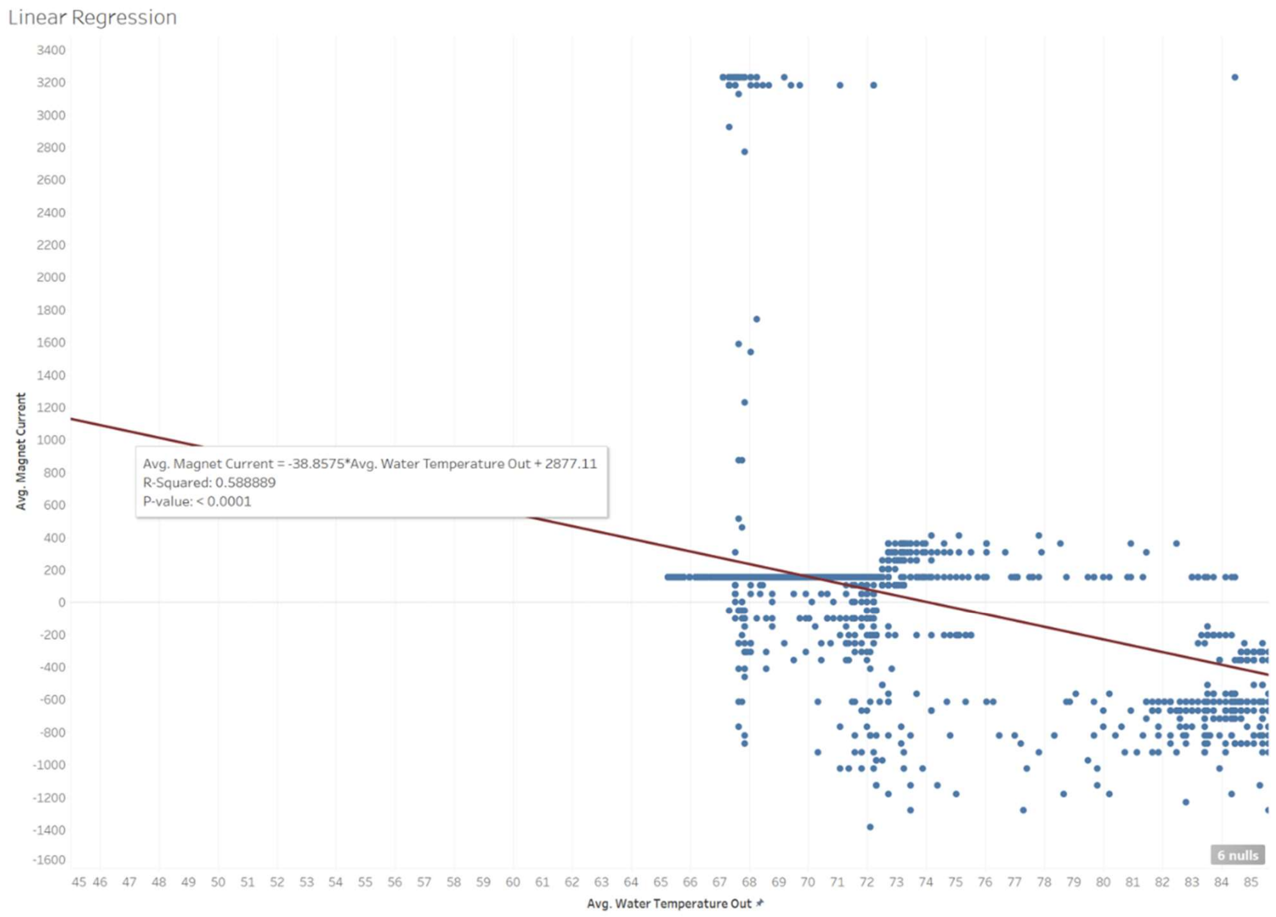
The dispersion diagram allows us to decide empirically:

- Whether a linear relationship between variables X (Water_Temperature_Out) and Y (magnet current) should be assumed.
- Whether the degree of the linear relationship between the variables is strong or weak, depending on how the points are located around an imaginary line that passes through the swarm of points.

To measure the relationship between Water_Temperature_Out and the magnet current, we will calculate the coefficient of determination (R^2), which is available in Tableau. This measure is used to analyze how a difference in a second variable can explain differences in one variable. More specifically, R-squared gives us the percentage variation in y explained by x-variables. The range of R^2 is 0 to 1, meaning the x-variables can explain from 0 % to 100% the change in y. The coefficient of determination, R^2, is similar to the correlation coefficient, R. The correlation coefficient formula will tell us how strong of a linear relationship there is between two variables. R^2 is the square of the correlation coefficient.

The linear regression results of Cober's data for the three key dates look like the following:

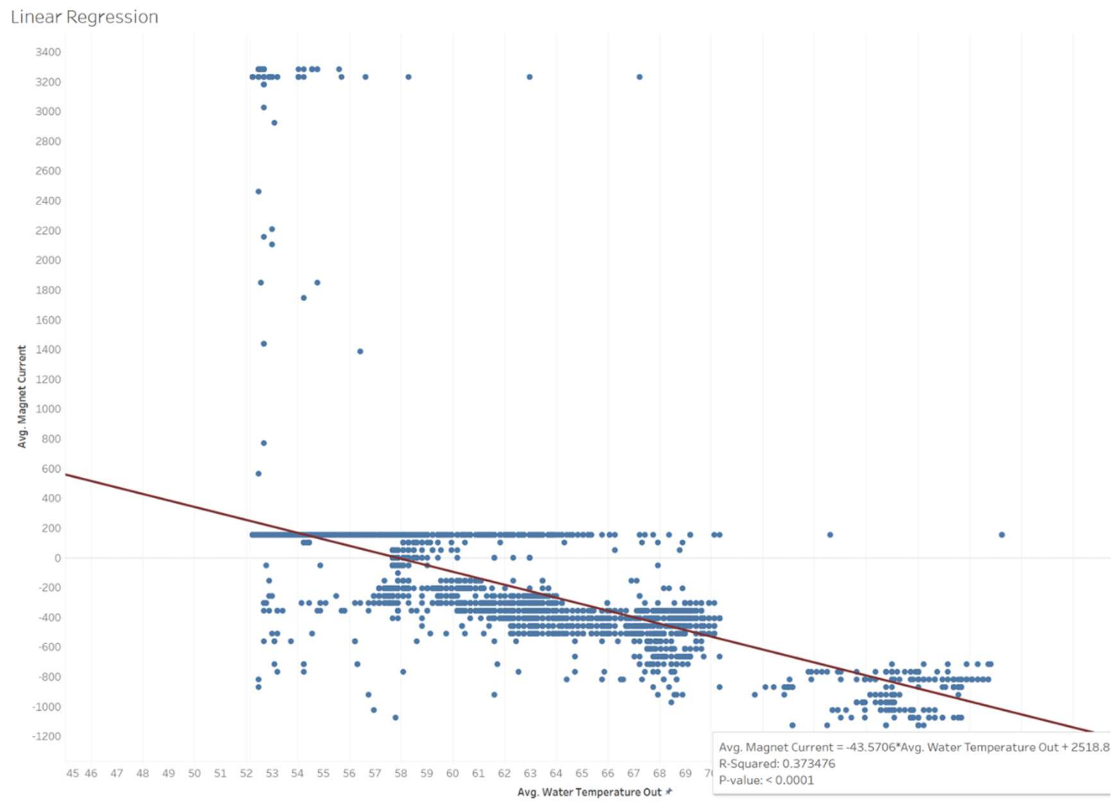
October 24, 2019



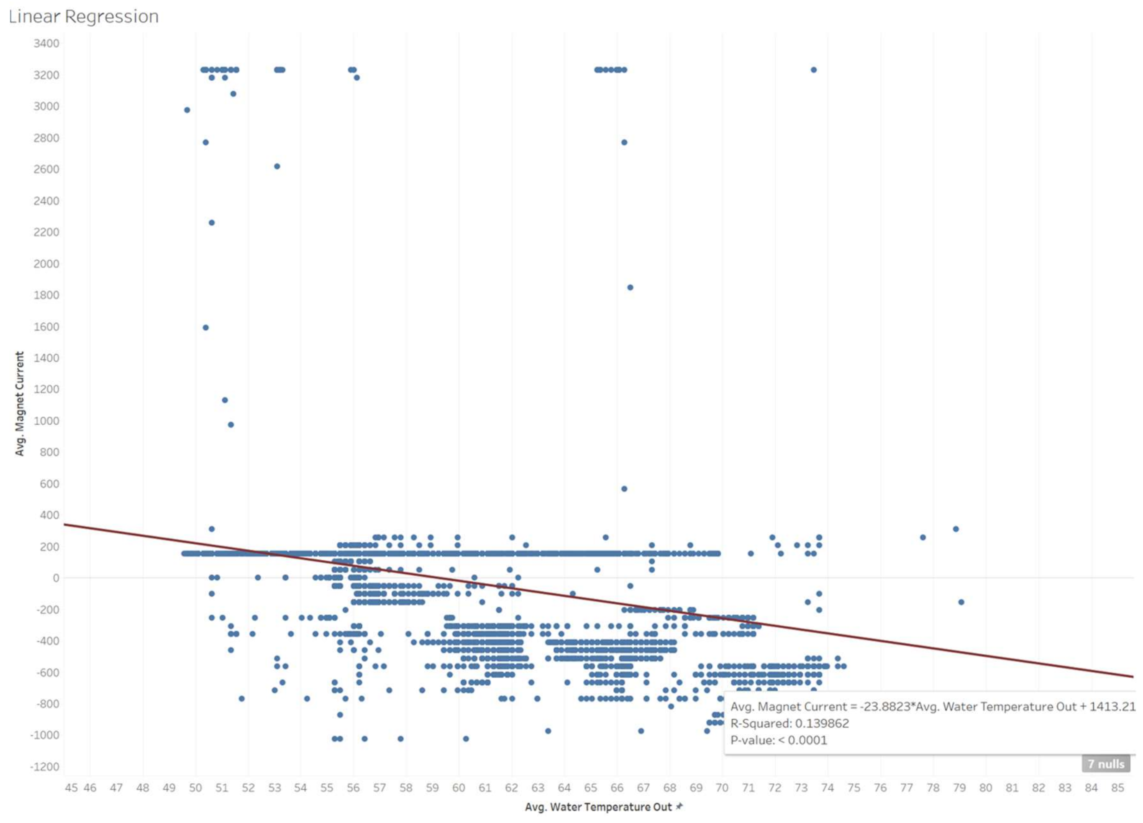
Average magnet current = -38.8575 * Average Water_Temperature_Out + 2877.11
$R^2 = 0.588889$
$p < 0.0001$

The p-value reports the probability that the produced linear equation was a result of chance. The smaller the p-value, the more significant the model is. A p-value of 0.05 or less is often considered sufficient. In our case, we have a p-value of less than 0.0001.

December 12, 2019



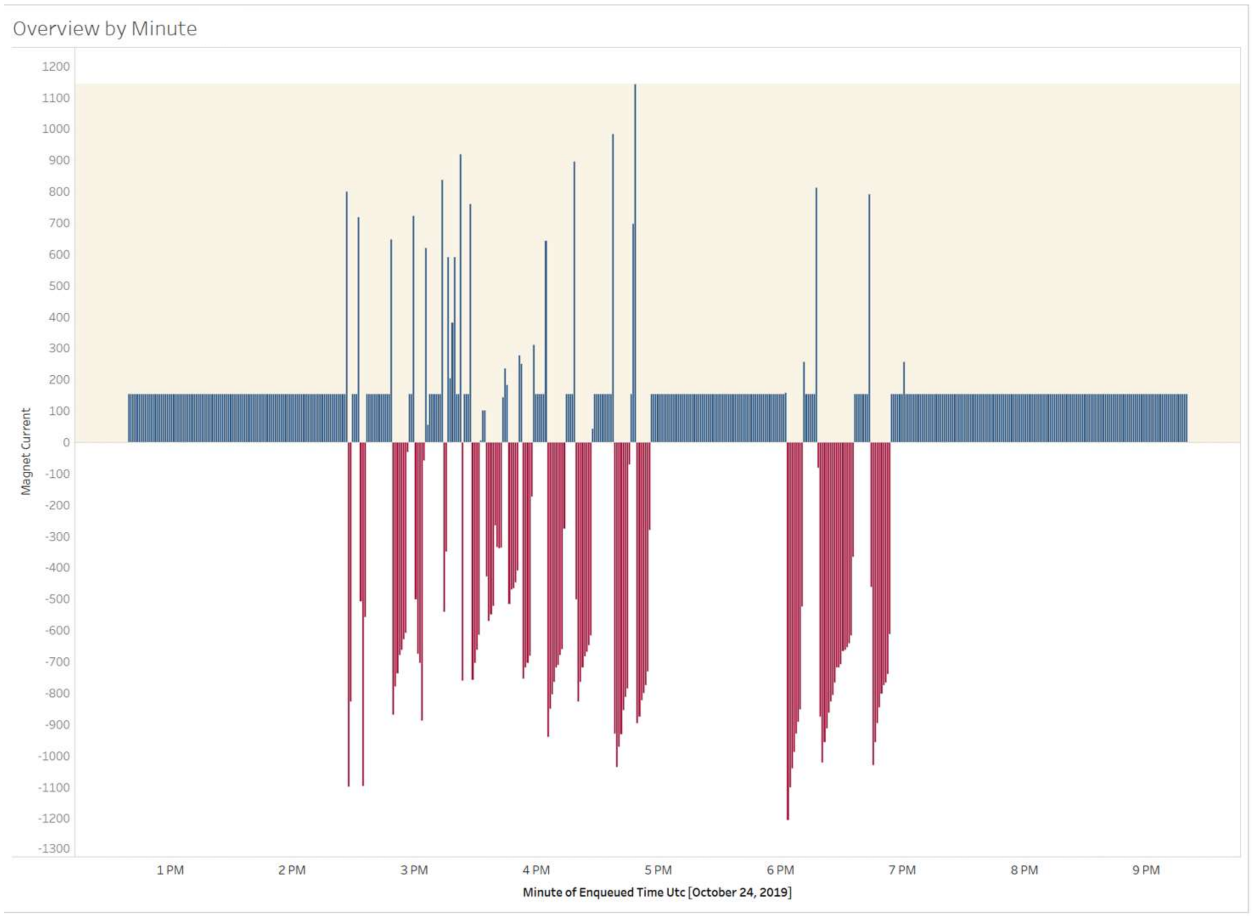
January 16, 2020



When R^2 is greater than zero, it implies a positive linear relationship between two variables. In Cober's case, the three significant dates we analyzed showed a positive linear relationship between two parameters – Water_Temperature_Out and magnet current since all R^2 are above zero. In other words, we were able to confirm that when the magnet current is negative, the average of Water_Temperature_Out increases.

Additional Observations

By observing the magnet current values per minute, we were able to see that magnet current has negative values, which are in red, immediately after a positive peak (in blue). The graph below illustrates this:



Future Work

The available data shows a lack of variation. There is only one device from which the information is collected; therefore, we suggest that Cober collects more sensor data from other devices. Secondly, the team had significant issues structuring the blobs. The JSON format presented some difficulty for the team and is also not the best file type to perform an analysis. Cober might want to revisit using JSON. The issues we had with the retrieval of the data, as well as the format, might have been due to the lack of our programming expertise. When assembling a data science team, Cober should look into hiring specialists and those with exceptional programming and analytical backgrounds.

The analysis in this report demonstrates the relationship between the two parameters – Water_Temperature_Out and magnetic current for three dates: October 24, 2019, December 12, 2019, and January 16, 2020. We recommend that Cober:

1. Conducts studies for other dates and compare the results
2. Explore potential relationships between other parameters, such as calculating the correlation coefficient (R) or the coefficient of determination (R^2) between Water_Temperature_In and the magnet current
3. More research is needed to define better how magnet current influences the water temperature. Several theories have been discussed, but gaining a better understanding of the mechanisms will result in a more efficient predictive analysis.