# Data Science in Discovery

## Machine Learning to Evaluate the Risk and Resilience of Societal Critical Infrastructure to Space Weather

Authors:

Sean Jung, Long Nguyen, Jiaqi Dong,

Jui-Wei Chang, Shiyun Zhou, Mazzy Chen, Tung Dang

University of Washington-Seattle

Department of Electrical and Computer Engineering

06 June 2024

Industry Sponsor: NASA Jet Propulsion Laboratory

## Table of Contents

# Introduction

**Industry Sponsor:** NASA Jet Propulsion Laboratory (JPL)

**Project Title:** Data Science for Discovery: Machine Learning to Evaluate the Risk and Resilience of Societal Critical Infrastructure to Space Weather

**Objectives:**

- Create a data framework for studying risk and resiliency of societal critical infrastructure from a space weather natural hazard.
- Integrate data from various domains to provide deeper insight into how solar events impact critical infrastructure.
- Utilize machine learning to predict outcomes in various space weather cases and identify subtle underlying connections.
- Explore the characteristics of space weather impacts on global critical infrastructure; primarily, electrical power grids and communications systems.

# Abstract

A Coronal Mass Ejection (CME) is a solar phenomenon that discharges a cloud of highly magnetized plasma into space, creating significant magnetic fluctuations on Earth's magnetosphere. This causes geomagnetically-induced currents (GIC) that damage critical, societal infrastructure, such as electrical transformers, oil pipelines, and long-range telecommunications. Accurately predicting and understanding the frequency, characteristics, and network effects of GICs requires deliberate data consolidation and analysis under a structured framework in order to enable deeper insights into the nature of GICs.

In collaboration with NASA Jet Propulsion Laboratory (JPL) and U.S Geological Survey (USGS), a comprehensive data pipeline was constructed to aggregate and analyze solar data from three major, public datasets (i.e SuperMAG, OMNIWeb, NERC). Combining and restructuring this data into several analysis-ready schema facilitated subsequent extreme value analysis, network analysis, and machine learning models for nowcasting and forecasting magnetic perturbation predictions.

Due to primary interests in high-value GICs, a Generalized Extreme Value (GEV) distribution was utilized to model the tail end of the GIC data. Quantile-quantile plots assisted in presenting comparisons between these GEV and Gaussian distributions, delineating distinct characteristics of each ground-based GIC region.

Network analysis was conducted using cross-correlations of wavelet decomposition coefficients, developing a response network to provide insight into subsequent effects of GICs on the physical U.S power grid. Furthermore, device-specific threshold optimization allowed a more representative, time-based spatial response network that assisted in identifying regions of vulnerability in U.S electrical infrastructure.

For magnetic perturbation and GIC predictions, three machine learning (ML) models were trained and tested, including a multi-linear regression (MLR), a random forest regression (RFR), and a long-short term memory model (LSTM). Kernel density plots with root means square metrics were used to compare and determine the models' accuracy and precision.

## Brief Results

Under extreme value analysis, the Generalized Extreme Value model provided a non-linear, inversely proportional trend between GIC values and predefined return periods (T). The GEV distribution also provided a superior fit for highly active GIC supernodal regions (i.e District of Columbia) when compared to a gaussian distribution, with quantile-quantile plots distinctly differentiating between supernodal and non-supernodal GIC characteristics.

Network analysis confirmed previous determinations of long range east-west connections in the response grid during heightened GIC activity (Orr 2021). Overlaying a physical U.S power grid to visualize the potential impact of GIC variations provided uninteresting conclusions, however, eigenvalue centrality measures of these nodal graphs highlighted previously neglected areas of interests, such as metropolitan areas of Seattle, WA, and Phoenix, AZ.

In machine learning, the LSTM model had the best performance in nowcasting magnetic perturbations with >95% accuracy given a 48-hour time frame of 5-min intervals. Trained on 50 epochs, batch size was set at 360, with smaller batch sizes inducing a less stable training process while larger batch sizes led to poorer performance.

# Teams Roles and Responsibilities

**Extreme Value Analysis**

- **Jui-Wei (Chris) Chang** **Data Analyst**
  - Analyzed the raw GIC data from North American Electrical Reliability Corporation (NERC) and SuperMAG datasets, extracting extreme values and other anomalies.
  - Processed and integrated the datasets into a single datasheet, and visualized values of interest on a geographical map.
  - Exported inputs for the machine learning team as crucial parameters and features.
  - Estimated 1-in-100 years estimates of future GIC events through Generalized Extreme Value (GEV) analysis to predict possible extreme values of GIC.
  - Applied GEV model with threshold exceedance method to processed NERC data and QDC continuous datasets.
- **Shiyun (Amy) Zhou** **Data Analyst**
  - Explored the theory and practical applications of extreme value analysis, analyzing NERC GIC datasets to extract extreme GIC values.
  - Utilized the pyextremes package to identify and analyze significant events.
  - Conducted comparisons between the data and various distributions, including the Generalized Extreme Value (GEV) distribution and the normal distribution, using quantile-quantile (QQ) plots.
  - Provided insights into the behavior of GICs in supernodes around Washington, D.C., and Michigan over multiple years.

**Network Analysis Team**

- **Sean Jung** **Team Leader**
  - Distributed tasks to team members based on project scope, aligning individual collaborative efforts for a comprehensive project completion.
  - Reorganized the wavelet decomposition by device ID and latitude and created an animation for visualizing wavelet coefficients after wavelet decomposition.

- Created overlays of U.S transmission lines and cross-correlations to visualize areas of interest.
- Created and deployed a Dash Plotly application for presenting research online and during the UW ENGINE 2024 Showcase.
- Primary liaison between industry sponsors, faculty mentors, and UW ENGINE team members.

- **Jiaqi (Doria) Dong** **Data Analyst**
  - Preprocessed GIC data to align on a concurrent time scale.
  - Performed discrete wavelet transform on GIC data and produced the heatmap of each ground-based device monitor
  - Performed maximal overlapping discrete wavelet transform (MOWDT) and found the relationship between each pair monitor in one geomagnetic disturbance by producing a time-varying cross-correlation heatmap and one-time cross-correlation network diagram.
  - Optimized specific threshold for each pair of monitors to decide the connections, preventing the network from being distorted.

**Machine Learning Team**

- **Long Thanh Nguyen** **Lead Technical Manager**
  - Consolidated OMNI raw dataset via API into Google Drive.
  - Preprocessed, resampled and harmonized raw data into clean data of 5-min cadence.
  - Created analysis-ready dataset (2010 - 2023) by joining SuperMAG and OMNI data based on date-time.
  - Evaluated Linear Regression and Random Forest models' performance with dataset range from 2010 - 2023, providing results for understanding the datasets.

- **Mazzy Chen** **Data Analyst**
  - Downloaded and transformed SuperMAG data to CSV format file, preprocessed and resampled data into 5-min cadence clean data for upload onto share drive.

- Setup the parameters for Long-Short Term Memory (LSTM) model, utilizing datetime OMNI data and geo-coordinates from SuperMAG as inputs.
- Trained LSTM model on 2010 - 2019 dataset while validating it in 2020 - 2023 dataset, successfully nowcasting magnetic perturbation over a 48-hour period.

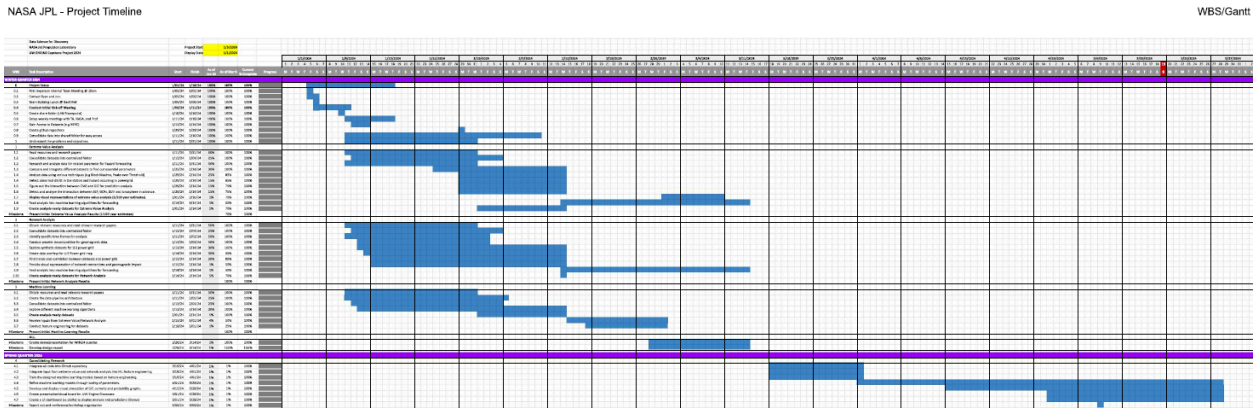
- **Tung Thanh Dang** **Data Engineer**
  - Downloaded and processed NERC GIC datasets as well as joining with the Supermag datasets to prepare for analyzing the change of magnetic perturbation (db/dt) effect on the Geomagnetic Induced Current(GIC).
  - Experimented 3 models including Linear Regression, Random Forest regression and Long Short Term Memory for the analysis.

**Industry and Faculty Mentors:**

- **NASA Jet Propulsion Laboratory (JPL)**
  - Ryan McGranaghan — Principal Data Scientist
  - Jonathan Sauder — Senior Mechatronics Engineer
- **University of Washington - Electrical & Computer Engineering Department**
  - Arindam Kumar Das — Faculty Mentor
  - Payman Arabshahi — UW Engine Program Manager
  - Kavya Balasubramanian — Teacher's Assistant

# Project Schedule

Creating our initial project schedule was heavily dependent on understanding the project scope from our industry sponsor. Due to the academic nature of this project, the initial schedule reflected an open-ended approach that relied on the progression of the project in the first quarter. The schedule became clearer as our understanding of each team member's research area improved through reading and discussing recent research papers suggested by our industry sponsors. Additionally, reiterating and discussing the industry sponsor's objectives and milestones for the project ensured the project was progressing in the right direction and on track..




*See attached excel spreadsheet for more details on Gantt Chart, Timeline, and WBS.*

## Key Milestones

| | |
|---|---|
| 1 | Collect background info and roles arrangement |
| 2 | Create analysis ready datasets |
| 3 | Development plan/approach |
| 4 | Extreme Value Analysis 1/100 year estimates |
| 5 | Network Analyses Initial Results |
| 6 | Machine Learning Initial Results |
| 7 | Report out and conference/workshop organization |

# Project Resources and Budget

This project incurred minimal expenses since the project scope was primarily software-based. Additionally, datasets used to conduct analysis and develop machine learning models were within the public domain and required no additional licensing, thus, no cost to access. The project's primary expense was a subscription to Google Collab Pro for computing our analysis and training machine learning models, individually expended and sought reimbursement through the University of Washington. The collective, actual costs was the following:

Colab Pro: 11.02 USD / month per person

Cost of Colab Pro x 4 months x 4 person = **176.32 USD**

*All receipts are place in "receipts" folder on team Google Drive.*

Our industry sponsor had no incurred expenses on our behalf for this project.

# Product Requirements / Deliverables

1) **Github Repository**
    a) Contains the software and data analysis of this research project..
    b) Contains the predictive models and relevant source code.
    c) Contains a description of machine learning algorithms and methods.
2) **Specifications**
    a) Shall be an intuitive and easy-to-build format for open-source contributors.
    b) Shall have an application that is simple to implement on local servers, if desired.
    c) Shall be able to extract current, up-to-date data from listed NASA and international sources (i.e NERC, SuperMAG, etc.).
3) **Gold Standard:**
    a) Develop accurate predictive models for geomagnetically induced currents (GICs) and/or magnetic perturbations (db/dt).

In addition to the Github Repository, a ~20 min outbrief to NASA JPL and USGS was required describing the progress made through this project research. (Delivered on 09 MAY 2024).

# Realistic Constraints

Due to privacy and security reasons, U.S electrical companies are often reluctant to release the blueprints and maintenance logs of their electrical infrastructure. Only when there exists a significant outage is there an incentive for electrical companies to release these events with greater detail to the public. Thus,this project relies on third-party or synthetic datasets that may reflect unknown biases and variances that may affect the accuracy of our analyses and machine learning models.

Additionally, real-time analysis for space weather predictions may require additional resources for practical applications that require additional compute costs. Since the primary goal of this project is to provide an analysis-ready dataset and framework for analysis and machine learning models, scaling these models for real-world applications is not the highest priority of this project at this time.

# System Requirements

Below states a list of detailed specifications required for the research. These system requirements were created in collaboration with our industry mentors and faulty advisors.

1. Research Requirements
    1.1. Extreme Value Analysis
        1.1.1. Read resources and research papers
            1.1.1.1. Shall have basic understanding of extreme value analysis, particularly referencing techniques and methods from Stuart Cole's "An Introduction to Statistical Modeling of Extreme Values".
        1.1.2. Consolidate datasets into centralized folder
            1.1.2.1. Shall have an intuitive folder structure, specified by date, source and the platform the data originates from.
        1.1.3. Visualize data distribution on a real-scale map of North America
            1.1.3.1. Shall integrate datasets from different sources into one map and operate as overlays.
        1.1.4. Research and analyze data for related parameters for hazard forecasting

1.1.4.1. Shall extract specific parameters that indicate likelihood of coronal mass ejections and/or other major solar events.

1.1.5. Compare and integrate different datasets to discover essential parameters

1.1.5.1. Shall identify the relevant parameters to feed into machine learning algorithms.

1.1.6. Analyze data using various techniques (e.g Block Maxima, Peaks over Threshold)

1.1.6.1. Shall create a distribution curve with associated probabilities for extreme value point, estimating 1/100 year events.

1.1.7. Detect abnormal dB/dt in the station and hazard occurring in powergrid.

1.1.7.1. Shall determine the interaction between CME and GIC for ML prediction.

1.1.7.2. Shall find and visualize correlation between GIC, CME and dB/dt.

1.1.8. Detect and analyze the interaction between SEP, GCRs, EUV and ionosphere in advance.

1.1.8.1. Shall identify timelines and correlations between SEP, GCRs, EUV and ionosphere.

1.1.9. Display visual representations of extreme value analysis (1/100 year estimates)

1.1.9.1. Shall provide graphics of future, possible GICs on earth given a future time reference.

1.1.10. Feed analysis into machine learning algorithms for forecasting

1.1.10.1. Shall provide heuristic analysis to assist in fine tuning machine learning algorithms.

1.2. Network Analysis

1.2.1. Obtain resources and read relevant research papers.

1.2.1.1. Shall understand the common and unique techniques of network analysis.

1.2.2. Consolidate datasets into centralized folder/storage locations.

1.2.2.1. Shall utilize efficient methods of extracting and storing information.

1.2.2.2. Shall have an intuitive folder structure, specified by date, source and the platform the data originates from.

1.2.3. Identify specific time frames for analysis.

1.2.3.1. Shall focus on significant solar events as well as low signal trends, omitted in eliminating noise and isolating trends.

1.2.4. Conduct wavelet decomposition for geomagnetic data.

1.2.4.1. Shall identify the specific time frames and locations where the GICs exhibit significant changes.

1.2.5. Explore synthetic datasets for the U.S power grid.

1.2.5.1. Shall focus on the time frames where geomagnetic disturbances occurred.

1.2.6. Create data overlays for the U.S power grid on a map.

1.2.6.1. Shall have overlays that can be easily toggled on/off.

1.2.6.2. Shall have color metrics to convey values of increased frequency

1.2.7. Find trends and correlation between datasets and power grid.

1.2.7.1. Shall identify the locations of those affected, vulnerable stations.

1.2.8. Provide visual representation of network connections and geomagnetic impact.

1.2.8.1. Shall generate an image depicting the power grid map, with each site marked to indicate the varying intensities of GIC.

1.2.9. Feed analysis into machine learning algorithms for forecasting

1.2.9.1. Shall provide insight in estimating the cascading impact of critical infrastructure.

1.3. Machine Learning

1.3.1. Obtain resources and read relevant research papers.

1.3.1.1. Shall understand the common and unique techniques of machine learning.

1.3.2. Create the data pipeline architecture.

1.3.2.1. Shall be efficient based on limiting using derived parameters which could lead to data redundancy and poor performance in ML models.

1.3.3. Consolidate datasets into centralized folder

1.3.3.1. Shall have an intuitive folder structure, specified by date, source and the platform the data originates from.

1.3.4. Explore different machine learning algorithms.

1.3.4.1. Shall utilize the most conducive algorithms for this machine learning application.

1.3.5. Create analysis-ready datasets.

1.3.5.1. Shall be in continuous datasets which are organized in continuous time series.

1.3.5.2. Shall datasets have missing values which should be applied to suitable imputing strategies, for example, closest values replacement (turning the 5-min cadence OMNI data into 1-min cadence OMNI data).

1.3.6. Receive inputs from Extreme Value/Network Analysis.

1.3.6.1. Shall be able to intake insights and incorporate in fine-tuning machine learning algorithms.

1.3.7. Conduct feature engineering for datasets

1.3.7.1. Shall utilize Multivariate linear regression and long-short term memory models to perform predictions of solar events and geomagnetically induced currents. Compare the performance of two models under the HSS measurement.

2. Software Requirements

2.1. Scripts must be written in Python 3.11 or greater due to compatibility with recent package updates.

2.2. Utilization of external packages (non-native to Python) is permitted.

2.3. Ensure import packages are non-redundant and unnecessary within and between Python scripts.

3. Data Storage & Format

3.1. Storage of data from public domains are permitted; however, may differ based on the dataset's licensing requirements

3.2. Utilization of cloud GPU processing is preferred due to the computational cost in training machine learning models

# Operating Hazards and Requirements

1. Safety and Security Requirements
   1.1. Refer to the licensing requirements of each dataset originator to comply with individual copyright and licensing requirements.
2. User Interface Requirements
   2.1. The user interface for website application must be intuitive, with HTML and CSS reflecting appropriate breakpoints to accommodate viewing for the clients' device.
   2.2. Interactive graphs must render smoothly and in a timely manner
3. Test and Validation Requirements
   3.1. Ensure reasonable training-validation-test split (i.e 80/10/10) for training machine learning models.
4. Precision and Accuracy Requirements
   4.1. Utilize industry standard precision and accuracy requirements (i.e F1 score, etc.)
5. Licensing Requirements
   5.1. Ensure all acknowledgements and references are properly cited.

# System Specifications

Since original system requirements were created in collaboration with industry mentors and faculty advisors, the system specifications directly reflect the system requirements.

Additional requirements of user interface were added due to most recent implementation of a website application for facilitating viewing during 2024 UW Electrical & Computer Engineering Showcase.

# Design Procedure/Methods

In data science, one of the most common challenges is creating a concise data pipeline that facilities the objectives of the project. Too much aggregation of data can become unnecessary and computationally expensive. Conversely, too little can be detrimental to developing meaningful outcomes, including analysis and machine learning models. Additionally, access and permission to use certain data can derail the project prior to its kick-off. Thus, this project focuses on extracting only magnetic and spatial data from three datasets on the public domain. After extraction, the data is loaded onto a common cloud storage where different schemas are created for enabling extreme value analysis, network analysis, and machine learning. Below is the block diagram of our project's data pipeline.

This data pipeline architecture contains the extraction, loading, transformation, and analysis processes for originating data from NERC, SuperMAG, OMNI, and other resources.

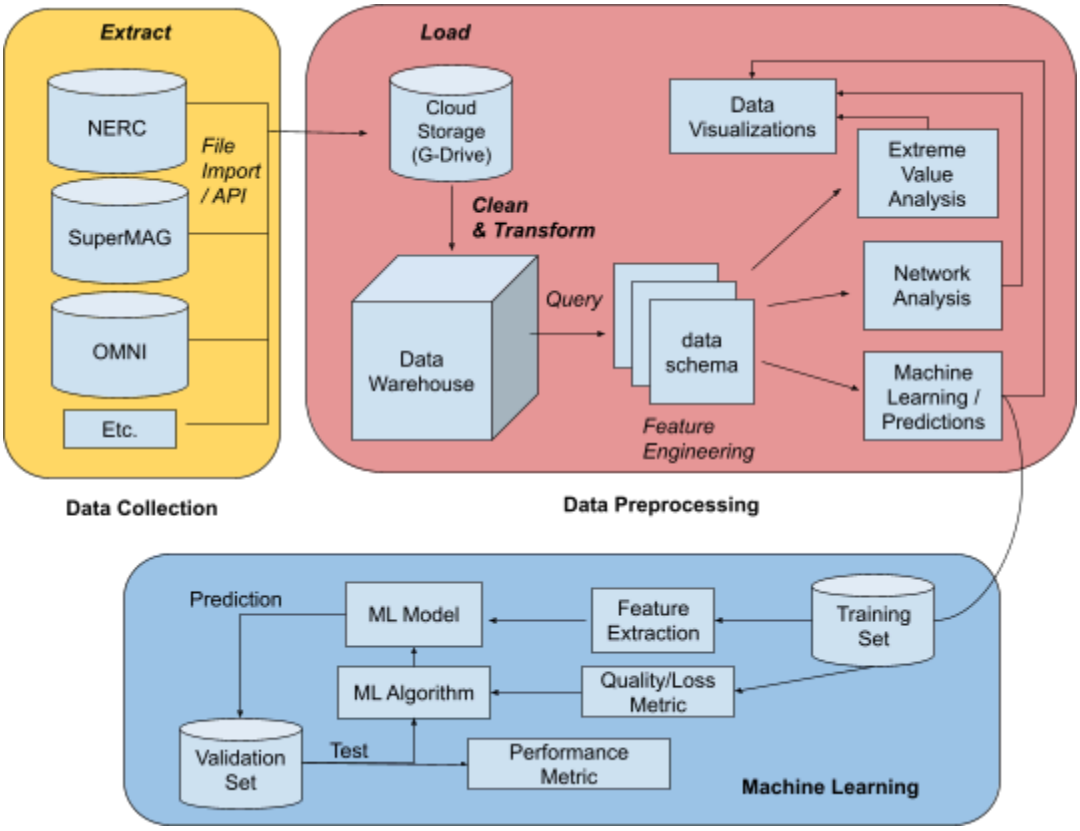



*Figure 1: Data Pipeline Architecture*

Overall Data Flow:

- Extract - GIC and spatial data are extracted from SuperMAG, OMNI and NERC datasets via scripts (API) or manually downloaded and uploaded to the common data storage, operating as a data warehouse.
- Transform - Data is transformed into useful and analysis-ready format by interpolating data gaps, removing redundant data, resampling for concurrent timeframes. (i.e Harmonize the SuperMAG 1-min cadence and OMNI 5-min cadence data.).
- Analysis - Data is analyzed for each research areas:
  - Extreme Value Analysis
  - Network Analysis
  - Machine Learning / Predictions
- Train and Test - (Machine Learning Models only)

Google Drive/Cloud Storage:

- Allows the different research areas of the project to experiment and create scripts for small subsets of data, prior to applying to greater scales.
- Raw data initially stored inside Google Drive as .csv files, and then, read out and transformed into necessary format and schema for analysis and machine learning models using python libraries (e.g pandas, numpy, etc.), and will be then loaded that data back to new .csv files.

Machine Learning Data Flow::

- This project trains and tests three machine learning model methods:
  - Multivariate Linear Regression (MLR)
  - Random Forest Regressor (RFR)
  - Long-short term memory (LSTM)
- Feature Engineering
  - Consists of feature selection and extraction for feeding into machine learning models for training.
- Training
  - Quantify the weights of neural networks based on the features selected.
  - Outputs predictions of solar events and future data.
- Testing
  - Tests predictions on accuracy based on a validation set using various performance metrics.
- Feedback/Tuning:
  - Fine-tune models to enhance performance.

# Software Implementation

The figures below display the code implementations for each research area. Due to different requirements amongst the research areas, the preprocessing of data is differentiated and tailored specifically to its application and visualization format.

## Extreme Value Analysis

```python
import os
import pandas as pd
# Base data directory containing event folders
base_data_path = '/content/risk-resiliency-spwx/data'
# List to collect max rows from all files in all events
max_row_list = []
# Loop over every event folder in the data directory
for event_folder in os.listdir(base_data_path):
    event_path = os.path.join(base_data_path, event_folder)

    # Check if it's a directory
    if os.path.isdir(event_path):
        # Construct path to the GIC subfolder within the event folder
        gic_path = os.path.join(event_path, 'GIC')
        # Check if the GIC subfolder exists
        if os.path.exists(gic_path) and os.path.isdir(gic_path):
            csv_files = [os.path.join(gic_path, f) for f in os.listdir(gic_path) if f.endswith('.csv')]
            csv_files.sort()  # Sort files by name
            # Exclude the last two CSV files
            csv_files_to_process = csv_files[:-2]
            # Process each CSV file
            for single_path in csv_files_to_process:
                df = pd.read_csv(single_path)
                df['GICMeasured_abs'] = df['GICMeasured'].abs()
                idx = df['GICMeasured_abs'].idxmax()
                max_row = df.loc[idx]

                # Add a new column with the event name
                max_row['Event'] = str(event_path[-8:])
                max_row['Year'] = str(event_path[-8:-4])
                # Add the event name as a new column to the max_row
                max_row_list.append(max_row)

# Convert the list to a DataFrame
max_rows_df = pd.DataFrame(max_row_list)
max_rows_df
```

***Figure 1**: Extracted the extreme values from every event and monitors of NERC GIC dataset and converted it to a useful dataframe for extreme value analysis.*

| | GICDeviceID | SampleDateTime | GICMeasured | GICMeasured_abs |
|---|---|---|---|---|
| 5444 | 10056 | 6/1/2013 6:07:20 AM | 19.30 | 19.30 |
| 7895 | 10075 | 6/1/2013 12:55:50 PM | -0.88 | 0.88 |
| 5501 | 10077 | 6/1/2013 6:16:50 AM | 9.40 | 9.40 |
| 1784 | 10083 | 5/31/2013 7:58:30 PM | -5.30 | 5.30 |
| 122 | 10112 | 6/1/2013 1:10:00 AM | -9.60 | 9.60 |
| ... | ... | ... | ... | ... |
| 7364 | 10437 | 6/1/2013 11:27:20 AM | -1.05 | 1.05 |
| 18507 | 10486 | 6/1/2013 11:33:48 AM | 5.50 | 5.50 |
| 971 | 10508 | 5/31/2013 4:04:44 PM | -11.87 | 11.87 |
| 9142 | 10555 | 6/1/2013 1:09:28 AM | -2.00 | 2.00 |
| 10326 | 10562 | 6/1/2013 2:28:24 AM | 25.00 | 25.00 |

***Figure 2:*** *Example of the Preprocessing of NERC GIC dataset*

```python
# @ Monitor Data
import os
import pandas as pd

# Base data directory containing event folders
base_data_path = '/content/risk-resiliency-spwx/data'

# List to collect data from all gic_monitor.csv files in all events
monitor_data = []

# Loop over every event folder in the data directory
for event_folder in os.listdir(base_data_path):
    event_path = os.path.join(base_data_path, event_folder)

    # Check if it's a directory
    if os.path.isdir(event_path):
        # Construct path to the GIC subfolder within the event folder
        gic_path = os.path.join(event_path, 'GIC')

        # Check if the GIC subfolder exists
        if os.path.exists(gic_path) and os.path.isdir(gic_path):
            # Construct the expected path to gic_monitor.csv
            monitor_file_path = os.path.join(gic_path, 'gic_monitors.csv')

            # Check if gic_monitor.csv exists
            if os.path.exists(monitor_file_path):
                # Read gic_monitor.csv and append its data to monitor_data
                df = pd.read_csv(monitor_file_path)
                monitor_data.append(df)

# Concatenate all DataFrames in the list to create a single DataFrame
monitor_df = pd.concat(monitor_data, ignore_index=True)

# Sort the dataframe file in ascending order of DeviceID
sorted_df = monitor_df.sort_values(by='Device ID', ascending=True)
sorted_df
```

***Figure 3****: Processed the monitor ID in order to integrate extreme values with coordinates.*

```python
# Select the useless columns
final_monitor_df = unique_df.drop(columns=[' Installation Type', ' Connection', ' Minimum Value in Measurement Range'])
# print the DataFrame that has only the selected columns
final_monitor_df # Final Monitor Table
```

| | Device ID | Latitude | Longitude |
|---|---|---|---|
| 1590 | 10051 | 44.4 | 96.5 |
| 603 | 10052 | 44.9 | 97.0 |
| 1843 | 10056 | 44.8 | 88.2 |
| 896 | 10063 | 33.4 | 112.5 |
| 977 | 10064 | 41.8 | 88.2 |
| ... | ... | ... | ... |
| 346 | 10578 | 40.2 | 82.8 |
| 936 | 10582 | 32.5 | 111.3 |
| 937 | 10584 | 40.0 | 104.0 |
| 894 | 10586 | 34.0 | 118.5 |
| 640 | 10591 | 35.8 | 106.1 |

305 rows × 3 columns

***Figure 4****: Built up a table of monitor device ID and coordinates*

```
# Rename the Device ID column in final_monitor_df to match max_rows_df for a consistent merge key
final_monitor_df_renamed = final_monitor_df.rename(columns={'Device ID': 'GICDeviceID'})

# Merge the DataFrames on the GICDeviceID column
merged_df = pd.merge(max_rows_df, final_monitor_df_renamed, on='GICDeviceID', how='inner')

# Sort the merged DataFrame by GICDeviceID
sorted_merged_df = merged_df.sort_values(by='Event')

# Print the sorted merged DataFrame
sorted_merged_df
```

```
print(sorted_merged_df.columns.to_list())
final_nerc = sorted_merged_df.drop('GICMeasured', axis=1)
final_nerc
```

```
['GICDeviceID', 'SampleDateTime', 'GICMeasured', 'GICMeasured_abs', 'Event', 'Year', ' Latitude', ' Longitude']
```

| | GICDeviceID | SampleDateTime | GICMeasured_abs | Event | Year | Latitude | Longitude |
|---|---|---|---|---|---|---|---|
| 1853 | 10562 | 6/1/2013 2:28:24 AM | 25.0 | 20130531 | 2013 | 40.7 | 81.4 |
| 1105 | 10401 | 6/1/2013 6:03:20 AM | 5.6 | 20130531 | 2013 | 39.1 | 78.3 |
| 1116 | 10402 | 6/1/2013 1:12:00 AM | 1.7 | 20130531 | 2013 | 39.5 | 78.9 |
| 1128 | 10403 | 6/1/2013 1:10:30 AM | 4.4 | 20130531 | 2013 | 39.5 | 78.9 |
| 101 | 10112 | 6/1/2013 1:10:00 AM | 9.6 | 20130531 | 2013 | 47.7 | 117.4 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 1742 | 10210 | 4/10/2022 4:39:05 AM | 0.3 | 20220409 | 2022 | 34.7 | 86.5 |
| 707 | 10299 | 4/10/2022 5:20:10 AM | 4.4 | 20220409 | 2022 | 46.6 | 87.4 |
| 1739 | 10209 | 4/10/2022 4:49:45 AM | 2.4 | 20220409 | 2022 | 34.8 | 85.7 |
| 1789 | 10314 | 4/10/2022 4:00:20 AM | 0.7 | 20220409 | 2022 | 44.5 | 89.6 |
| 637 | 10291 | 4/9/2022 12:00:00 AM | 0.0 | 20220409 | 2022 | 43.4 | 72.7 |

1854 rows × 7 columns

***Figure 5a & 5b**: Integrated with GIC extreme values and dropped the unrelated column into one CSV file*

| GICDeviceI | SampleDateTime | GICMeasure | Event | Year | Latitude | Longitud |
|---|---|---|---|---|---|---|
| 10397 | 6/1/2013 12:45:50 AM | 23.4 | 20130531 | 2013 | 39.1 | -78.3 |
| 10433 | 6/1/2013 6:04:20 AM | 4.76 | 20130531 | 2013 | 38.5 | -76.6 |
| 10433 | 6/1/2013 6:04:20 AM | 4.76 | 20130531 | 2013 | 38.5 | -76.6 |
| 10257 | 5/31/2013 7:07:00 PM | 1.22 | 20130531 | 2013 | 47.4 | -101.2 |
| 10218 | 6/1/2013 1:10:32 AM | 2 | 20130531 | 2013 | 42.6 | -96.3 |
| 10218 | 6/1/2013 1:10:32 AM | 2 | 20130531 | 2013 | 42.6 | -96.3 |
| 10257 | 5/31/2013 7:07:00 PM | 1.22 | 20130531 | 2013 | 47.4 | -101.2 |
| 10360 | 6/1/2013 3:37:00 AM | -1.3 | 20130531 | 2013 | 40.2 | -75.3 |
| 10197 | 6/1/2013 9:06:34 AM | 10.1 | 20130531 | 2013 | 36.4 | -82.2 |
| 10360 | 6/1/2013 3:37:00 AM | -1.3 | 20130531 | 2013 | 40.2 | -75.3 |
| 10181 | 6/1/2013 6:02:32 AM | 9 | 20130531 | 2013 | 41.9 | -74 |
| 10322 | 6/1/2013 6:05:10 AM | 2.6 | 20130531 | 2013 | 43.3 | -89.4 |

***Figure 6**: Format of processed NERC GIC data*

| Da | Mo | Year | Begin | End | Lat | Long | Device Type |
|---|---|---|---|---|---|---|---|
| 11 | 3 | 1989 | 7.27 | 999.99 | 39.18 | -76.53 | Powergrid |
| 11 | 3 | 1989 | 7.44 | 999.99 | 39.17 | -76.53 | Powergrid |
| 11 | 3 | 1989 | 14.04 | 999.99 | 39.33 | -76.83 | Powergrid |
| 11 | 3 | 1989 | 14.22 | 999.99 | 39.18 | -76.53 | Powergrid |
| 12 | 3 | 1989 | 20.29 | 999.99 | 49.99 | -97.42 | Powergrid |
| 12 | 3 | 1989 | 22.15 | 999.99 | 44.35 | -79.81 | Powergrid |
| 13 | 3 | 1989 | 0 | 1 | 38.43 | -76.44 | Powergrid |
| 13 | 3 | 1989 | 1.19 | 999.99 | 47.37 | -92.6 | Powergrid |
| 13 | 3 | 1989 | 1.19 | 999.99 | 49.99 | -97.42 | Powergrid |
| 13 | 3 | 1989 | 1.19 | 999.99 | 41.17 | -73.93 | Powergrid |

***Figure 7**: Format of Database of power-grid anomalies*

```
# Define a list to store the indices of unqualified rows
rows_to_drop = []

# Delete repeated time points
df = df.drop_duplicates(subset='SampleDateTime', keep='first')

# Delete station if all time points are empty
for index, row in df.iterrows():
    if row.drop('SampleDateTime').isnull().all():
        rows_to_drop.append(index)

# Find how many of the measurements differ from one time step to the next
for device_id in df.columns:
    if device_id != 'SampleDateTime':
        df[device_id] = pd.to_numeric(df[device_id], errors='coerce')
        diff_count = (df[device_id].diff() == 0).sum()
        if diff_count / len(df) > 0.6:
            rows_to_drop.extend(df.index)

# Remove rows with unacceptable bias
for device_id in df.columns:
    if device_id != 'SampleDateTime':
        med_tmp = np.nanmedian(df[device_id].values)
        if abs(med_tmp) >= (threshold / 2):
            rows_to_drop.extend(df.index)

# Drop the identified rows
df = df.drop(rows_to_drop)
```

***Figure 8**: Alternate method of extracting and transforming the NERC GIC data for extreme values for visualizing dataset.*

```
# List to store data
all_data = []

# Counter to keep track of the number of files read
file_count = 0

# Iterate over all files in the folder
for filename in os.listdir(folder_path):
    if filename.endswith('.csv'):
        file_path = os.path.join(folder_path, filename)
        # Read data from the current CSV file
        try:
            df = pd.read_csv(file_path)
            # Check if the required columns are present
            if 'SampleDateTime' in df.columns and 'GICMeasured' in df.columns:
                df['SampleDateTime'] = pd.to_datetime(df['SampleDateTime'])
                df.set_index('SampleDateTime', inplace=True)
                all_data.append((filename, df))  # Store filename and data frame tuple
            else:
                print(f"Warning: 'SampleDateTime' or 'GICMeasured' column not found in file: {filename}. Skipping this file.")
        except Exception as e:
            print(f"Error reading file {filename}: {str(e)}")

# Create a figure and axes
fig, ax = plt.subplots(figsize=(10, 6))
colors = plt.cm.tab10(np.linspace(0, 1, len(all_data)))

# Iterate through each data set and plot it
for i, (filename, data) in enumerate(all_data):
    #extremes = get_extremes(data['GICMeasured'], method='POT', threshold=0.2, r='6H')
    plot_extremes(ts=data['GICMeasured'], extremes=extremes, extremes_method='POT', extremes_type='high', threshold=0.5, r='6H', ax=ax)
    ax.plot(data.index, data['GICMeasured'], color=colors[i], linewidth=2) #label= file

plt.show()
```

***Figure 9**: Visualized extreme value within one event using pyextreme package*

# Network Analysis

```
# Extract the time and determine the common time point
for file_path, latitude in zip(file_paths, latitudes):
    df = pd.read_csv(file_path, usecols=[time_column, gic_column])
    df[time_column] = pd.to_datetime(df[time_column])

    # Set the datetime object as an index
    df.set_index(time_column, inplace=True)
    # Stores resampling data for each file
    # 'T' stands for every minute
    resampled_data.append(df.resample('T').mean().interpolate(method='spline', order=3))
```

***Figure 10**: Data preprocessing for Network Analysis*

*The date format was converted to a DateTime object in Pandas, and then the time column was set as an index, resampling the data to a frequency per minute. The re-sampled data is then averaged using the.mean() method for each new time interval. Finally, the spline method and third-order interpolation are used to smooth-fill any missing values that may occur after resampling.*

```
detail_coeffs_list = []
for i in range(len(file_paths)):
    data = resampled_data[i]
    coeffs = pywt.wavedec(data[gic_column], 'haar', level=2)
    detail_coeffs_list.append(coeffs[1])
```

```
modwt_results = {}
def determine_max_level(data_length, wavelet='db4'):
    return pywt.swt_max_level(data_length)
def modwt(data, wavelet, level):
    return pywt.swt(data, wavelet, level=level, start_level=0)

for file_name in file_names:
    file_path = os.path.join(folder_path, file_name)
    df = pd.read_csv(file_path)
    df[time_column] = pd.to_datetime(df[time_column])
    df = df.drop_duplicates(subset=time_column).set_index(time_column)
    df = df.reindex(pd.date_range(start=df.index.min(), end=df.index.max(), freq='T'), method='nearest')
    resampled_df = df.resample('T').mean().interpolate(method='spline', order=3)
    gic_signal = make_length_even(resampled_df[gic_column].values)
    level = determine_max_level(len(gic_signal))
    modwt_result = modwt(gic_signal, 'db4', level)
    modwt_results[file_name] = modwt_result
```

***Figure 11a & 11b**: Wavelet transform (DWT & MODWT)*

*Each file undergoes a wavelet decomposition that outputs a list of wavelet coefficients for each device ID.*

```
edges = []
def wavelet_cross_correlation(wt1, wt2):
    corrs = []
    for coeff1, coeff2 in zip(wt1, wt2):
        corr = np.corrcoef(coeff1[0], coeff2[0])[0, 1]
        corrs.append(corr)
    return corrs
for (file1, wt1), (file2, wt2) in itertools.combinations(filtered_modwt_results.items(), 2):
    cross_corrs = wavelet_cross_correlation(wt1, wt2)
    print(cross_corrs)
    threshold = 0.5
    for corr in cross_corrs:
        if np.abs(corr) > threshold:
            edges.append((file1, file2, corr))
```

```
sliding_correlations = {}
def sliding_window_cross_correlation(wt1, wt2, window_size=30):
    assert len(wt1) == len(wt2)
    sliding_corrs = []
    for start in range(len(wt1) - window_size + 1):
        end = start + window_size
        window_corr = np.corrcoef(wt1[start:end], wt2[start:end])[0, 1]
        sliding_corrs.append(window_corr)
    return sliding_corrs
for (file1, wt1), (file2, wt2) in itertools.combinations(filtered_modwt_results.items(), 2):
    corrs_over_time = sliding_window_cross_correlation(wt1[0][0], wt2[0][0])
    sliding_correlations[(file1, file2)] = corrs_over_time
```

***Figure 12a & 12b**: Correlation (time-varying & one-time) - Utilizing a sliding window method, the correlation between different devices are registered if each wavelet coefficient aligns and cumulatively reaches a certain threshold. This iteration of correlation has a threshold of 0.5*

# Machine Learning

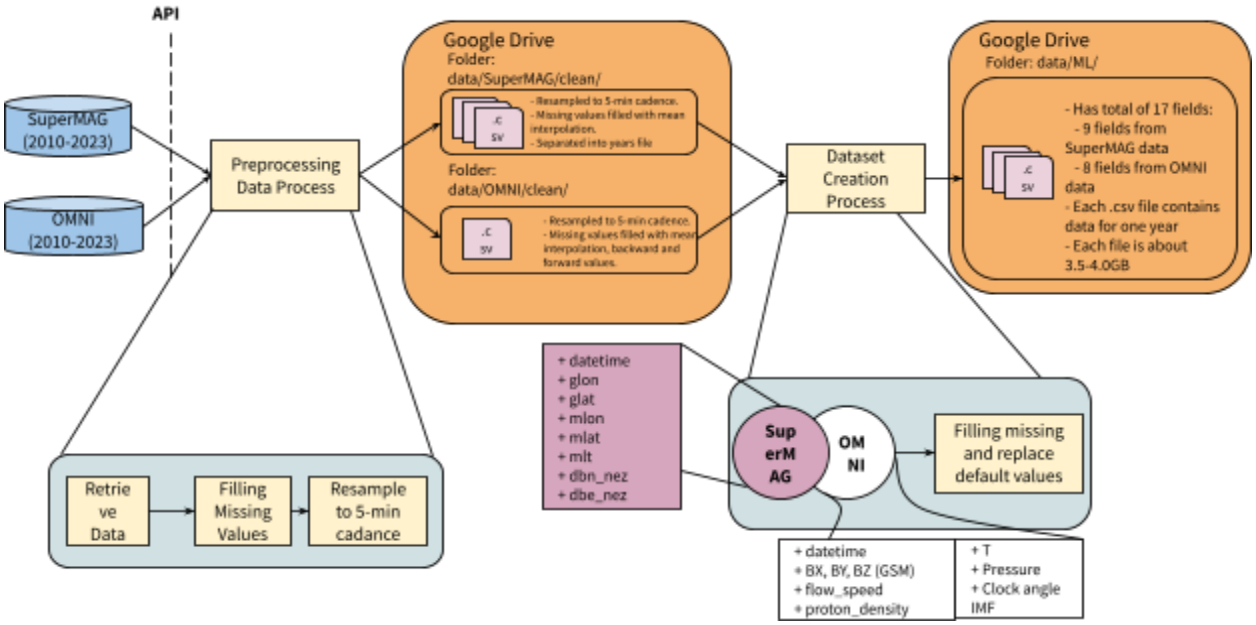



***Figure 13**: Detailed diagram of process of generating analysis-ready dataset for magnetic perturbation prediction model (2010 - 2023)*

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
```

```
Mounted at /content/drive
```

```
%cd /content/drive/MyDrive/'2024 UW ENGINE Capstone'
BASE_PATH = "/content/drive/MyDrive/2024 UW ENGINE Capstone/data/"
```

```
/content/drive/.shortcut-targets-by-id/1oyb6sI_Gh8b_o0AIjqSCq4UljphkuJTI/2024 UW ENGINE Capstone
```

```
import pandas as pd
import os
from tqdm import tqdm
import warnings
warnings.filterwarnings('ignore')
warnings.simplefilter('ignore')
```

```
def load_data(source, file_name, datatype="raw", sep=","):
    return pd.read_csv(f'{BASE_PATH}{source}/{datatype}/{file_name}', sep=sep)
```

```
def is_exist_nan_and_fill(df, name):
    nan_cols = df.isna().any()
    nan_cols = list(nan_cols[nan_cols == True].keys())

    if len(nan_cols) != 0:
        # print(f'There is NaN inside {name} data')
        df[nan_cols] = df[nan_cols].interpolate(method ='linear', axis=0, limit_direction='both')

    assert(df.isna().any().sum() == 0)

    return df
```

```python
def prepare_magnetic_pertubation_dataset(omni_data, sm_data, clockwise_angle_data, year):
    sm_data_features = sm_data[['datetime', 'glon', 'glat', 'mlon', 'mlat', 'mlt', 'dbn_nez', 'dbe_nez']]
    del sm_data
    omni_data_features = omni_data[['DateTime', 'BX_GSE', 'BY_GSM', 'BZ_GSM', 'flow_speed', 'proton_density', 'T', 'Pressure', 'SYM_H']]

    num_expected = len(sm_data_features)

    sm_data_features = is_exist_nan_and_fill(sm_data_features, 'superMAG')
    omni_data_features = is_exist_nan_and_fill(omni_data_features, 'OMNI')
    assert(len(omni_data_features) == 1472544)
    clockwise_angle_data = is_exist_nan_and_fill(clockwise_angle_data, 'CLOCKWISE_ANGLE')
    assert(len(clockwise_angle_data) >= 105120)

    sm_ljoin_omni = sm_data_features.merge(omni_data_features, left_on='datetime', right_on='DateTime', how='left')
    del sm_data_features
    dataset = sm_ljoin_omni.merge(clockwise_angle_data, left_on='datetime', right_on='DateTime', how='left')
    # del clockwise_angle_data
    del sm_ljoin_omni

    dataset.drop(columns=['DateTime_x', 'DateTime_y'], inplace=True)
    num_instances = len(dataset)

    assert(dataset.isna().any().sum() == 0)
    assert(num_expected == num_instances)

    output_file = BASE_PATH + f"ML/dataset_{year}.csv"
    dataset.to_csv(output_file, mode='a', header=not os.path.exists(output_file), index=False)
    # print(dataset)

    del dataset

    return num_instances
```

```python
def create_dataset(chunk_size=200000):
    # load OMNI data
    omni_data = load_data("OMNI", 'OMNI_2010_2023.csv', datatype='clean')

    for i in range(2010, 2024):
        print(f"Processing {i} data ...")
        # Load data file
        sm_data = load_data("SuperMAG & NERC", f'resample_{i}.csv')

        # n = 200000  #chunk row size
        sm_data_list = [sm_data[i:i+chunk_size] for i in range(0,sm_data.shape[0],chunk_size)]
        num_expected = len(sm_data)
        del sm_data

        clockwise_angle_data = load_data("OMNI", f'IMF_CLOCK_ANGLE_{i}.csv', datatype='clean')

        total_instance = 0

        for sm_data_chunk in tqdm(sm_data_list):
            total_instance += prepare_magnetic_pertubation_dataset(omni_data, sm_data_chunk, clockwise_angle_data, i)
            # del sm_data_chunk

        del clockwise_angle_data
        del sm_data_list

        assert(total_instance == num_expected)
```

```python
# del sm_data
# del clockwise_angle_data
create_dataset()
```

***Figure 14 a/b/c*** *: Code for generating ready for analysis dataset of magnetic perturbation data*

```python
BASE_PATH = "/Users/longnguyen/Documents/LongBm/UW_Engine_Capstone/"

def create_official_dataset(files, stations_glon_glat=None):
    result = pd.DataFrame()

    total_entries = 0

    for file in tqdm(files):
        dataset = pd.read_csv(BASE_PATH + file)
        total_entries += len(dataset)
        result = pd.concat([result, dataset], ignore_index=True)

    assert(total_entries == len(result))

    print('Num entries:', total_entries)

    return result
```

Python

```python
def generate_file_names(start, end):

    return [f'dataset_{i}.csv' for i in range(start, end + 1)]
```

Python

```python
train_files = generate_file_names(2010, 2019)
test_files = generate_file_names(2020, 2023)
```

Python

```python
from sklearn.datasets import make_regression
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error as mse
from sklearn.metrics import mean_absolute_error as mae
import pandas as pd
import datetime as dt
import numpy as np
from tqdm import tqdm
```

[8]

```python
def subsample_of_num_stations(df, stations_glon_glat=None):
    if stations_glon_glat == None:
        return df

    condition = False

    for glon, glat in stations_glon_glat:
        condition = condition | (df['glon'] == glon) & (df['glat'] == glat)

    return df[condition]
```

[9]

***Figure 15 a/b**: Code for experiment with Multivariate Linear Regression:*

```python
train_dataset = create_official_dataset(train_files, [(76.920, 43.250)]) # Concatenate and reset index
# result = result.reindex(range(8))
train_dataset
```

```
100%|██████████| 10/10 [22:23<00:00, 134.32s/it]
Num entries: 270731078
```

| | datetime | glon | glat | mlon | mlat | mlt | dbn_nez | dbe_nez | BX_GSE | BY_GSM | BZ_GSM | flow_speed | proton_density | T | Pressure | SY |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2010-01-01 00:00:00 | 2.930 | 36.85 | 77.884224 | 27.490238 | 14.398300 | -0.961335 | 2.232730 | 0.08 | 2.99 | 0.38 | 283.6 | 3.91 | 24946.0 | 0.63 | |
| 1 | 2010-01-01 00:05:00 | 2.930 | 36.85 | 77.884224 | 27.490238 | 0.082257 | -1.188596 | 2.232202 | 0.08 | 2.96 | 0.60 | 284.1 | 3.90 | 25642.0 | 0.63 | |
| 2 | 2010-01-01 00:10:00 | 2.930 | 36.85 | 77.884224 | 27.490238 | 0.166704 | -1.039363 | 2.223046 | 0.15 | 2.94 | 1.04 | 284.9 | 3.65 | 25423.0 | 0.59 | |
| 3 | 2010-01-01 00:15:00 | 2.930 | 36.85 | 77.884224 | 27.490238 | 0.251440 | -0.996011 | 2.120424 | 0.09 | 2.81 | 1.17 | 284.5 | 3.63 | 24643.0 | 0.58 | |
| 4 | 2010-01-01 00:20:00 | 2.930 | 36.85 | 77.884224 | 27.490238 | 0.336487 | -1.116696 | 1.921184 | -0.12 | 2.64 | 1.41 | 282.9 | 3.53 | 24771.0 | 0.57 | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 270731073 | 2019-12-31 23:35:00 | 245.518 | 62.48 | -56.854180 | 69.424225 | 14.601374 | -4.964983 | -1.253583 | 0.24 | 0.05 | -0.79 | 302.3 | 8.05 | 28546.0 | 1.47 | |
| 270731074 | 2019-12-31 23:40:00 | 245.518 | 62.48 | -56.854180 | 69.424225 | 14.684576 | -2.425964 | -0.768775 | -1.38 | 0.91 | -1.10 | 296.8 | 6.26 | 26910.0 | 1.10 | |
| 270731075 | 2019-12-31 23:45:00 | 245.518 | 62.48 | -56.854180 | 69.424225 | 14.767933 | -0.896954 | -0.782651 | -0.65 | 0.39 | -1.03 | 300.9 | 7.76 | 29161.0 | 1.42 | |
| 270731076 | 2019-12-31 23:50:00 | 245.518 | 62.48 | -56.854180 | 69.424225 | 14.851412 | -0.341447 | -1.462580 | -2.21 | 1.94 | -0.81 | 297.0 | 6.40 | 27451.0 | 1.14 | |
| 270731077 | 2019-12-31 23:55:00 | 245.518 | 62.48 | -56.854180 | 69.424225 | 14.934922 | 0.338462 | -0.611285 | -3.19 | 3.22 | -0.59 | 296.3 | 5.24 | 26135.0 | 0.92 | |

270731078 rows × 17 columns

```python
test_dataset = create_official_dataset(test_files, [(76.920, 43.250)]) # Concatenate and reset index
# result = result.reindex(range(8))
test_dataset
```

```
100%|██████████| 4/4 [03:15<00:00, 48.89s/it]
Num entries: 75897418
```

| | datetime | glon | glat | mlon | mlat | mlt | dbn_nez | dbe_nez | BX_GSE | BY_GSM | BZ_GSM | flow_speed | proton_density | T | Pressure |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2020-01-01 00:00:00 | 76.920 | 43.25 | 150.22480 | 38.829740 | 4.824365 | 3.734044 | 3.025891 | -2.93 | 3.02 | -0.51 | 295.60 | 5.310000 | 24010.000000 | 0.930000 |
| 1 | 2020-01-01 00:05:00 | 76.920 | 43.25 | 150.22480 | 38.829740 | 4.908339 | 3.690294 | 3.105935 | -2.58 | 2.88 | -0.47 | 296.65 | 6.470000 | 28973.000000 | 1.140000 |
| 2 | 2020-01-01 00:10:00 | 76.920 | 43.25 | 150.22480 | 38.829740 | 4.992806 | 3.848985 | 3.066320 | 2.03 | 1.07 | 0.63 | 297.70 | 7.630000 | 33936.000000 | 1.350000 |
| 3 | 2020-01-01 00:15:00 | 76.920 | 43.25 | 150.22480 | 38.829740 | 5.077563 | 4.146138 | 3.051936 | 0.81 | 1.11 | -0.03 | 296.30 | 8.610000 | 34855.000000 | 1.510000 |
| 4 | 2020-01-01 00:20:00 | 76.920 | 43.25 | 150.22480 | 38.829740 | 5.162632 | 4.603361 | 2.818436 | 0.11 | 0.12 | -0.24 | 295.20 | 8.930000 | 27156.000000 | 1.560000 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 75897413 | 2023-05-26 01:50:00 | 245.518 | 62.48 | -56.85418 | 69.424225 | 17.354526 | 82.314190 | 0.890782 | 2.39 | 0.10 | -2.55 | 512.00 | 1.550000 | 136874.000000 | 0.820000 |
| 75897414 | 2023-05-26 01:55:00 | 245.518 | 62.48 | -56.85418 | 69.424225 | 17.432081 | 70.770140 | 17.823336 | 2.91 | -0.10 | -2.20 | 490.90 | 1.580000 | 130945.000000 | 0.760000 |
| 75897415 | 2023-05-26 02:00:00 | 245.518 | 62.48 | -56.85418 | 69.424225 | 17.509791 | 79.602571 | 32.362805 | 1.76 | -1.19 | -3.51 | 503.90 | 1.560000 | 145751.000000 | 0.790000 |
| 75897416 | 2023-05-26 02:05:00 | 245.518 | 62.48 | -56.85418 | 69.424225 | 17.587575 | 85.544673 | 45.955596 | 1.87 | -0.58 | -3.32 | 500.10 | 1.586667 | 145173.333333 | 0.793333 |
| 75897417 | 2023-05-26 02:10:00 | 245.518 | 62.48 | -56.85418 | 69.424225 | 17.642046 | 92.803855 | 41.331915 | 1.58 | -1.12 | -3.40 | 496.30 | 1.613333 | 144595.666667 | 0.796667 |

75897418 rows × 17 columns

***Figure 15 c/d**: Code for experiment with Multivariate Linear Regression:*

```python
def num_of_days_in(year):
    """
    Check if the given year is a leap year or not.
    """
    if year % 4 == 0:
        if year % 100 == 0:
            if year % 400 == 0:
                return 366
            else:
                return 365
        else:
            return 366
    else:
        return 365

```

```python
import datetime as dt

def datetime_to_numeric(df):
    df['datetime'] = pd.to_datetime(df['datetime'])

    # Calculate day of the year
    day_of_year = df['datetime'].dt.dayofyear
    # Calculate universal time (UT) in hours
    ut_hours = df['datetime'].dt.hour + df['datetime'].dt.minute / 60

    # Convert day of year and UT to angles
    # print(len(day_of_year))
    # print(len(df['datetime'].dt.year.apply(num_of_days_in)))
    # return
    day_angle = 2 * np.pi * day_of_year / df['datetime'].dt.year.apply(num_of_days_in)
    ut_angle = 2 * np.pi * ut_hours / 24

    # Calculate sine and cosine of angles
    df['day_sin'] = np.sin(day_angle)
    df['day_cos'] = np.cos(day_angle)
    df['ut_sin'] = np.sin(ut_angle)
    df['ut_cos'] = np.cos(ut_angle)

    return df
```

***Figure 16***: *This block of code is to convert datetime into numeric values for training purpose:*

```python
y_columns = ["dbn_nez", "dbe_nez"]
x_columns = train_dataset_subsample.columns.difference(["dbn_nez", "dbe_nez", "datetime"])

# train, test = train_test_split(dataset_2023, test_size=0.2)

x_train = train_dataset_subsample[x_columns]
y_train = train_dataset_subsample[y_columns]

x_test = test_dataset_subsample[x_columns]
y_test = test_dataset_subsample[y_columns]
```

```python
# define model
model = LinearRegression()
# fit model
model.fit(x_train, y_train)
```

```
LinearRegression()
```

```python
y_predict = model.predict(x_test)
# summarize prediction
print(y_predict)
```

```
[[-1.26602354  1.92747607]
 [-1.26823838  1.93863425]
 [-1.44219262  2.03916193]
 ...
 [-5.31455098  3.43954702]
 [-5.8067625   3.45633991]
 [-6.57527299  3.65670953]]
```

***Figure 17**: This block of code is to train the Multivariate Linear Regression model and getting prediction on 2020 - 2023*

## Create LSTM ready dataset

```python
#Split into train and test set
df_train = dataset[dataset.index < datetime(2020, 1, 1)]
df_test = dataset[dataset.index >= datetime(2020, 1, 1)]
print(df_train.shape, df_test.shape)
```

Python

```
(1042065, 15) (353684, 15)
```

## Data Normalization

```python
def normalize_df(train_df, test_df):
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaler.fit(train_df)
    train_data = pd.DataFrame(scaler.transform(train_df), columns=train_df.columns, index=df_train.index)
    test_data = pd.DataFrame(scaler.transform(test_df), columns=test_df.columns, index=df_test.index)
    return train_data, test_data
```

Python

```python
data_train, data_test = normalize_df(df_train, df_test)
# print(data_train.head(5))
# print(data_test.tail(5))
```

Python

```python
def create_dataset(dataset, lookback):
    X, y = [], []
    dataset = pd.DataFrame(dataset)
    for i in range(len(dataset)-lookback):
        feature = dataset.iloc[i:i+lookback, :]
        target = dataset.iloc[i+lookback, -2:]
        X.append(feature)
        y.append(target)
    return np.asarray(X), np.asarray(y)
```

Python

```python
lookback = 1
train_X, train_y = create_dataset(data_train.values, lookback)
test_X, test_y = create_dataset(data_test, lookback)
```

Python

```python
# split into validation and test sets
val_size = int(test_X.shape[0] * 0.6)
# split into input and outputs
vld_X, vld_y = test_X[:val_size, :], test_y[:val_size, :]

print(train_X.shape, train_y.shape, vld_X.shape, vld_y.shape,test_X.shape, test_y.shape)
```

Python

```
(1042064, 1, 15) (1042064, 2) (212209, 1, 15) (212209, 2) (353683, 1, 15) (353683, 2)
```

***Figure 18 a/b**: This block of code is to create the LSTM model ready dataset. We normalized data and defined time-steps here.*

```python
# design network
# from keras.optimizers import Adam
model = Sequential()
model.add(LSTM(128, activation='relu', input_shape=(train_X.shape[1], train_X.shape[2])))
# model.add(Dense(64, activation='relu'))
# model.add(Dense(32, activation='relu'))
model.add(Dense(2))
model.compile(loss='mse', optimizer='adam')
# fit network
history = model.fit(train_X, train_y, epochs=50, batch_size=360, validation_data=(vld_X, vld_y), verbose=2, shuffle=False)
# plot history
pyplot.plot(history.history['loss'], label='train')
pyplot.plot(history.history['val_loss'], label='test')
pyplot.legend()
pyplot.savefig('loss_l.png')
pyplot.show()
```

Python

```
WARNING:tensorflow:Layer lstm_8 will not use cuDNN kernel since it doesn't meet the cuDNN kernel criteria. It will use generic GPU kernel as fallback when running on GPU
Epoch 1/50
2895/2895 - 6s - loss: 0.0015 - val_loss: 8.9620e-05
Epoch 2/50
2895/2895 - 6s - loss: 6.6578e-05 - val_loss: 1.3312e-05
Epoch 3/50
2895/2895 - 6s - loss: 3.8625e-05 - val_loss: 1.2015e-05
Epoch 4/50
2895/2895 - 6s - loss: 2.2844e-05 - val_loss: 1.3793e-05
Epoch 5/50
2895/2895 - 6s - loss: 3.1667e-05 - val_loss: 1.1724e-05
Epoch 6/50
2895/2895 - 6s - loss: 2.8381e-05 - val_loss: 1.1428e-05
Epoch 7/50
2895/2895 - 6s - loss: 3.2403e-05 - val_loss: 1.1531e-05
Epoch 8/50
2895/2895 - 6s - loss: 2.2578e-05 - val_loss: 1.1366e-05
Epoch 9/50
2895/2895 - 6s - loss: 3.0088e-05 - val_loss: 1.1589e-05
Epoch 10/50
2895/2895 - 6s - loss: 2.1456e-05 - val_loss: 1.1984e-05
Epoch 11/50
2895/2895 - 6s - loss: 2.8804e-05 - val_loss: 1.1429e-05
Epoch 12/50
2895/2895 - 5s - loss: 2.4221e-05 - val_loss: 1.1661e-05
...
Epoch 49/50
2895/2895 - 6s - loss: 2.3526e-05 - val_loss: 1.0845e-05
Epoch 50/50
2895/2895 - 5s - loss: 2.0079e-05 - val_loss: 1.0921e-05
```

## Data Prediction

```python
yhat = model.predict(test_X)

train_max0 = np.max(df_train['dbn_geo'])
train_min0 = np.min(df_train['dbn_geo'])
train_max1 = np.max(df_train['dbe_geo'])
train_min1 = np.min(df_train['dbe_geo'])

test_y[:,0] = (test_y[:,0]*(train_max0-train_min0)+train_min0)*-1
test_y[:,1] = (test_y[:,1]*(train_max1-train_min1)+train_min1)*-1
yhat[:,0] = (yhat[:,0]*(train_max0-train_min0)+train_min0)*-1
yhat[:,1] = (yhat[:,1]*(train_max1-train_min1)+train_min1)*-1

rmse_1 = sqrt(mean_squared_error(test_y[:,0], yhat[:,0]))
rmse_2 = sqrt(mean_squared_error(test_y[:,1], yhat[:,1]))
mae_1 = mean_absolute_error(test_y[:,0], yhat[:,0])
mae_2 = mean_absolute_error(test_y[:,1], yhat[:,1])
print('Test RMSE for dbn: %.3f' % rmse_1)
print('Test RMSE for dbe: %.3f' % rmse_2)
print('Test MAE for dbn: %.3f' % mae_1)
print('Test MAE for dbe: %.3f' % mae_2)
```

Python

```
Test RMSE for dbn: 1.603
Test RMSE for dbe: 1.655
Test MAE for dbn: 0.937
Test MAE for dbe: 0.863
```

***Figure 19 c/d**: This block of code is to set up and train the LSTM model. Then, make predictions and denormalize data for Loss calculation.*

# Test Procedure + Results and Analysis

## Extreme Value Analysis

In the initial phase of our analysis, we concentrated on processing on the North American Electric Reliability Corporation (NERC) raw dataset, spanning the years 2013 to 2022. The GIC (Geomagnetic Induced Current) values consisted of positive and negative data, thus, we converted all values into their absolute magnitudes and then extracted the maximum values from each GIC event from each GIC device across the United States. Subsequently, we visualized these values on a geographic map to access their distributions with other statistical measures. This approach facilitated our initial understanding of the spatial distribution of geomagnetic disturbances across different regions over a long timespan.

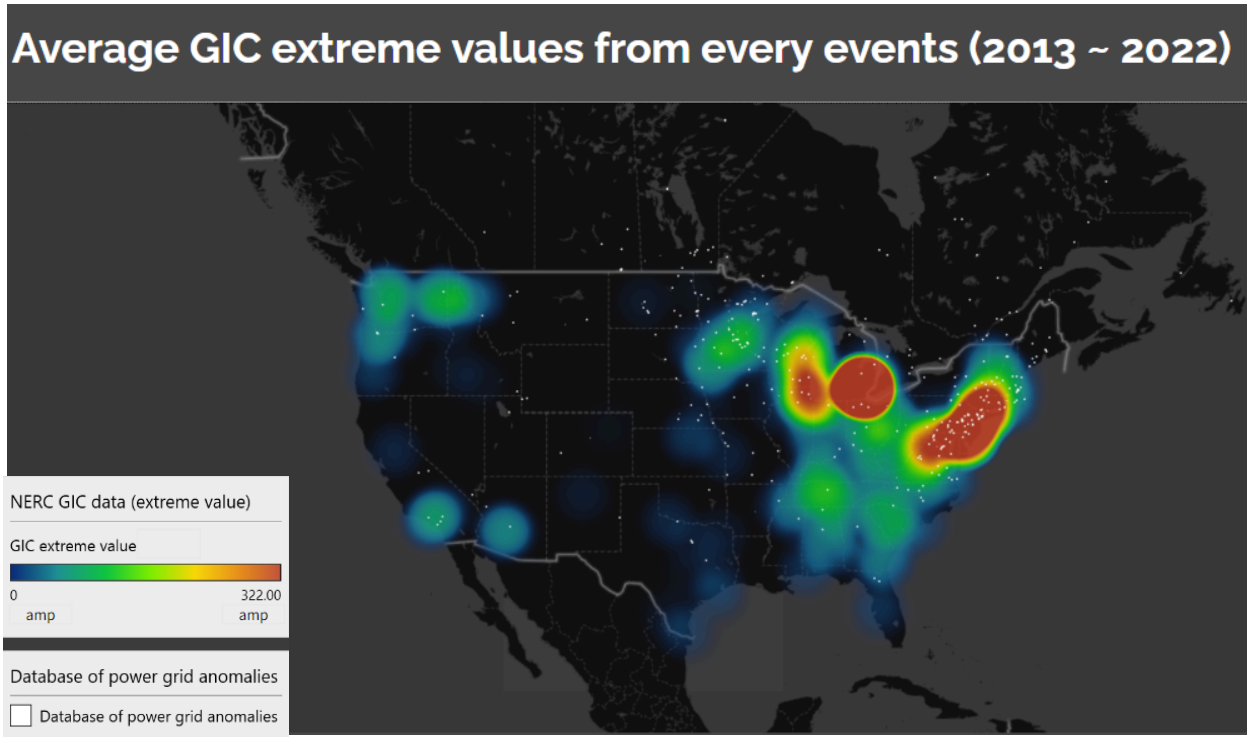



***Figure 20:*** *Visualization of GIC on USA map*

In the second phase of our analysis, we focused on estimating the hazard return levels for periods 1-in-100 years using the Generalized Extreme Value (GEV) method. This approach aimed to predict the worst-case scenarios for geomagnetically induced current (GIC) events based on our dataset. To implement this, we fit our extreme GIC data into the GEV model and specified a

return period (T), which the model uses to generate the corresponding return values. A critical aspect of this process involved setting an appropriate threshold for exceedance. We experimented with various thresholds and conducted additional estimation for 1-10 year return levels to validate the trend accuracy and refine our GEV model. In our examination, we acquired a return value of 700.2 amp when setting the thres=15 and T=100, while the max extreme value in our current dataset is 483 amp.

## Data and Model Setup

- Dataset: Extreme GIC measurements across the US, spanning from 2013 to 2022.
- Extreme Value of Interest: 483 amps
    - The highest GIC measured in the existing dataset.
- Threshold Setting for GEV: 15 amps
    - Data above this value are considered for the GEV fitting.
- Return Periods: We focus primarily on 100-year return level but also examine shorter periods(1-10 years) for comparative insights.

## GEV Model Fitting

1. Fit the GEV Model: Using GIC data (2013 to 2022) that exceeds the thresholds.
2. Calculated Return Levels:
    - For T=100 yrs, threshold =15 amps: The model predicted a return level of 700.2A
    - For T=10 yrs, threshold =42 amps: The model predicted a return level of 393.2A

    *NOTE: See Figures 21 and 22 to see the trend lines.*

## Threshold Sensitivity

- Lowering the threshold might include more data points, potentially smoothing out the return level curve but might also dilute the focus on truly extreme events. Conversely, a higher threshold could lead to data scarcity, increasing model uncertainty.

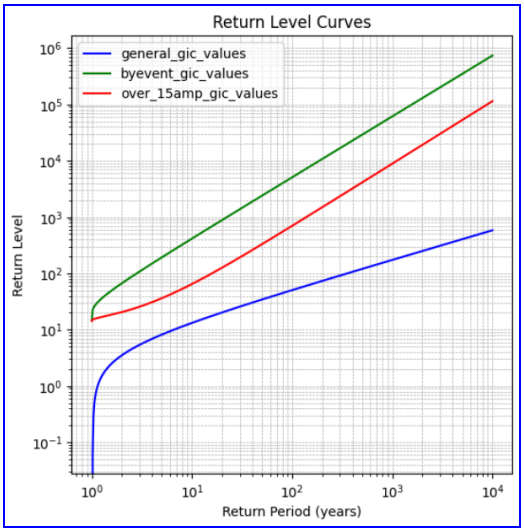



***Figure 21:*** *Return level plot for 3 kinds of data filter level*

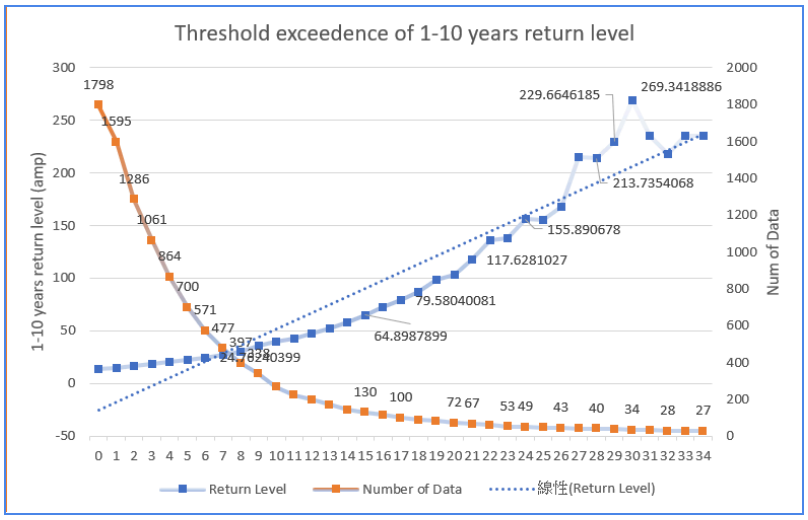



***Figure 22:*** *Threshold exceedance of 1-10 years GEV return*

$$z_T = \begin{cases} \mu - \frac{\sigma}{\xi}\left[1 - \{-\log(1 - 1/T)\}^{-\xi}\right] & \text{if} \quad \xi \neq 0 \\ \mu - \sigma \log\{-\log(1 - 1/T)\} & \text{if} \quad \xi = 0. \end{cases}$$

***Figure 23:*** $Z_T$: T-year return level equation;
This is the event value that happens once in every T years

## Supernodes

After identifying certain regions of the U.S (i.e Washington D.C) that have more activity than others, we focused on how these GICs differentiated based on regions and what their GIC characteristics were. At the start of our research, 18 significant events were identified between 2013 to 2022. We utilized the pyextremes package - a Python library aimed at performing univariate Extreme Value Analysis (EVA) - to examine these events by comparing GEV and Gaussian distributions of GICs to identify supernodal vs non-supernodal characteristics.

## Quantile-Quantile Plots

The quantile-quantile (QQ) plot, a model-independent method for comparing two empirically sampled statistical distributions (Braun & Murdoch, 2016), was employed to investigate changes in the distribution over its full range. This method does not require any prior assumptions of the

functional form of the underlying distribution. The QQ plot can be used to test how the functional form changes over time, or whether the samples are drawn from distributions with the same model but with time-dependent moments. Moreover, if the underlying distribution is multicomponent, the QQ plot will reveal the (potentially time-dependent) transitions between regions of parameter space that contain the distinct components or populations of observations.

## Supernode & Non-supernode Analysis

We conducted an analysis of the QQ plot for the supernodes, defined as the nodes within a 50-mile radius around Washington, D.C. (latitude 38.8951, longitude -77.0364) and Michigan (latitude 44.3148, longitude -85.6024). The analysis included comparisons between the data and the normal distribution as well as the Generalized Extreme Value (GEV) distribution for both supernodes and non-supernode regions.

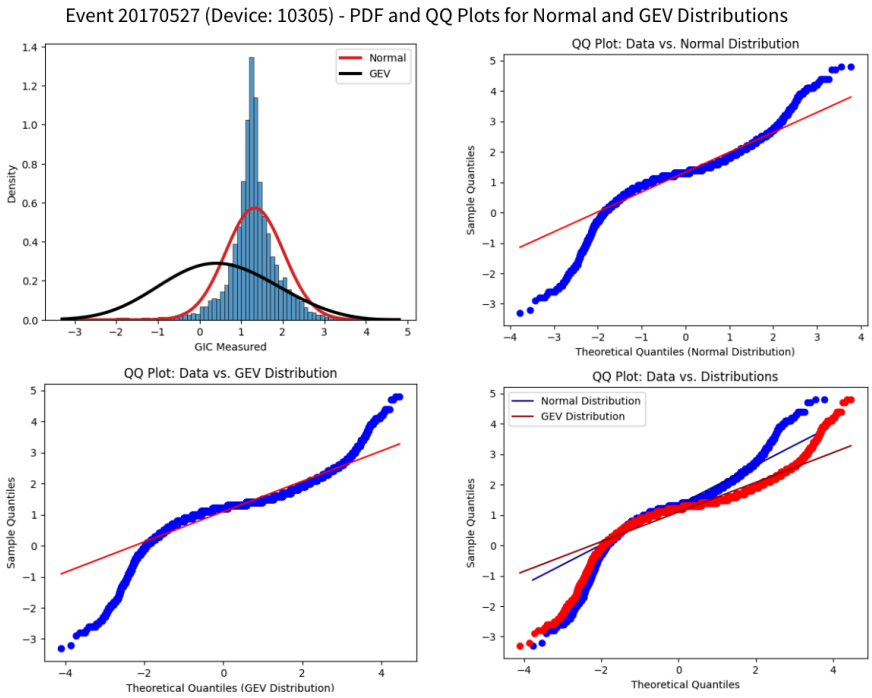



***Figure 24:*** Supernode 1 (Device 10305, Event 2017 05 27)

For Supernode 1 (Device 10305, Event 20170527) in Figure 1, we discovered significant deviation from the normal distribution, particularly in the upper tail. This deviation suggests the presence of extreme values that the normal distribution fails to capture. In contrast, the GEV

distribution showed better alignment in the upper tail, indicating that it models extreme values more effectively, which is typical for data with heavy tails.

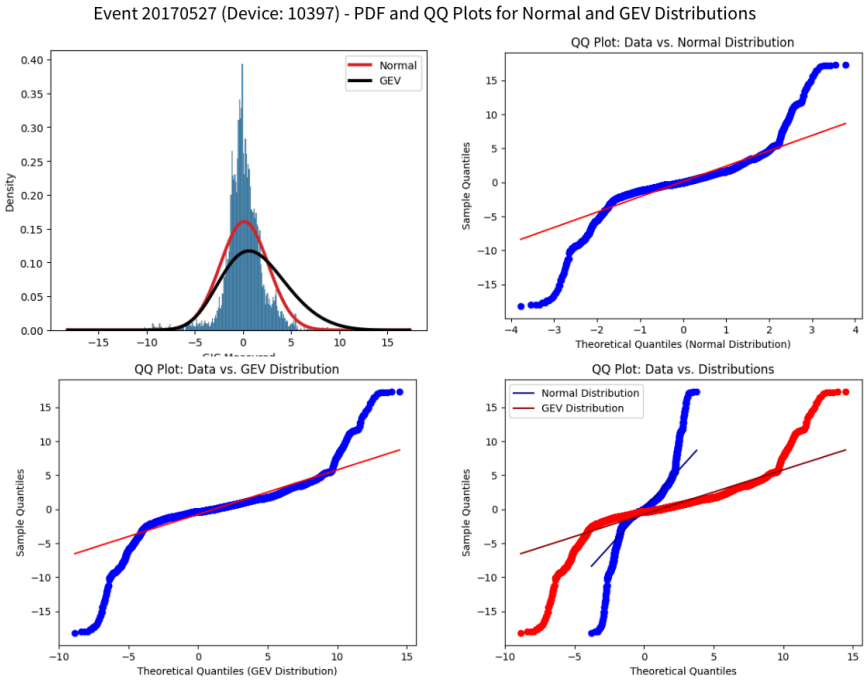



***Figure 25:*** Supernode 2 (Device 10397, Event 2017 05 27)

Similarly, in Supernode 2 (Device 10397, Event 20170527) in Figure 2, we observed significant deviations from the normal distribution. Consistent with the findings for the first supernode, the second supernode's data also exhibited a better fit with the GEV distribution, suggesting consistency in data behavior across different supernodes.

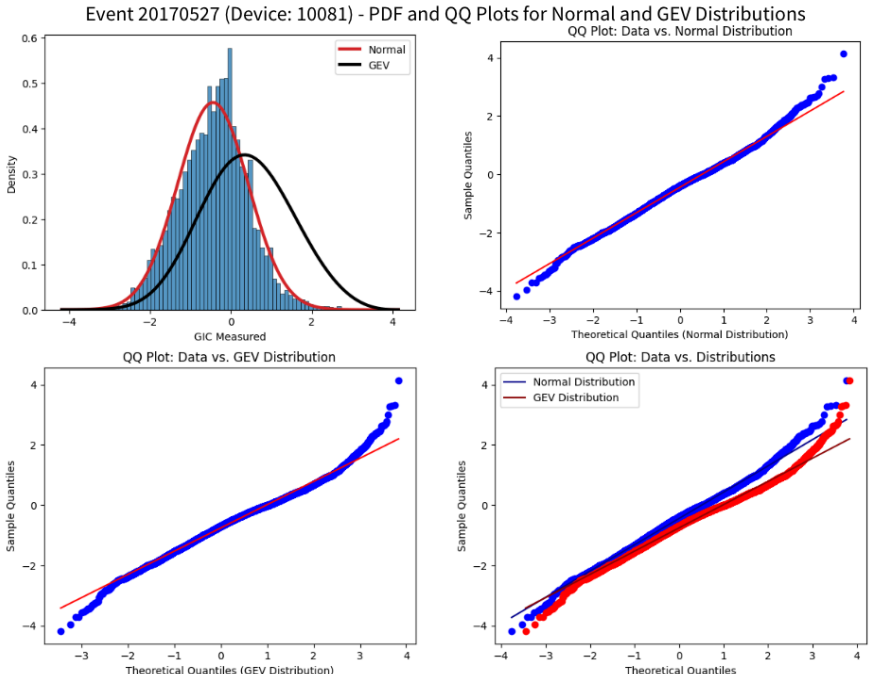



***Figure 26:*** *Non-supernode (Device 10081, Event 2017 05 27)*

For the non-supernode (Device 10081, Event 20170527) in Figure 3, the QQ plot revealed that the data align more closely with the normal distribution than with the GEV distribution. This pattern indicates fewer occurrences of extreme deviations and a more stable data pattern, typical of less critical infrastructure or differing operational influences.

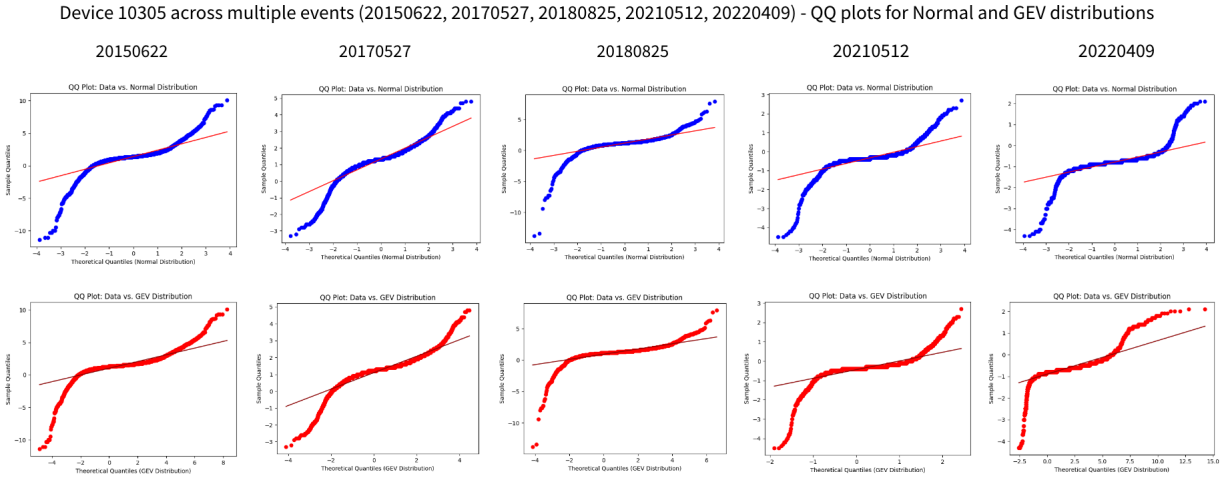



***Figure 27:*** *Comparison of non-supernodes*

## Results

When comparing QQ plots for Supernode 1 across multiple events (20150622, 20170527, 20180825, 20210512, 20220409) in Figure 4, we noted that the supernode data exhibited increasing deviations from the normal model over five years, particularly in the later years. This trend suggests a progression towards greater variability or an increase in operational anomalies as the system ages.

Conversely, the GEV model consistently provided a better fit, especially in the upper quantiles, across all years, indicating that the GEV distribution is more effective in capturing the extreme values within the dataset.

In most cases, more than 90% of the data fall into a single component, the “core” of the distribution. The remaining 10% of data form a distinct “tail” component, which itself splits into two parts in some cases. Generally, the core component in the distribution shows little sensitivity to the detailed differences across these years. However, the two-part tail dominates the plot on the linear scale due to the heavy-tailed nature of the distribution.

The Generalized Extreme Value (GEV) distribution, designed to model the tail behaviors of datasets, showed that when focusing on the largest sample quantile and observing its proximity and fit to the GEV line in the QQ plot, the data point for the largest sample quantile is closer to the theoretical line provided by the GEV distribution compared to the normal distribution. However, at values outside the 4-amps region, the QQ traces deviated from each other, highlighting the differences between the distributions.

The approximate linearity of each segment indicated that the functional form of the distribution remained similar throughout the years, exhibiting a consistent departure pattern.

# Network Analysis

Network Analysis contains data preprocessing, wavelet decomposition, and cross-correlation. Initially separated by events, measured GIC data from the North American Electric Reliability Corporation (NERC) was firstly preprocessed to prevent errors derived from data gaps and NaN values. Then, wavelet decomposition was applied to the resampled data, identifying strong changes in measured GIC values to infer fluctuations of GICs in the power grid. Lastly, cross-correlations between each Device ID were created using a global and sliding window analysis amongst different thresholds. Centrality measures are calculated to display the interconnectedness of the nodal graphs while overlaying this response network with the physical power grid over time to provide insight into the actual GIC fluctuation within the power grid.

## Software Design

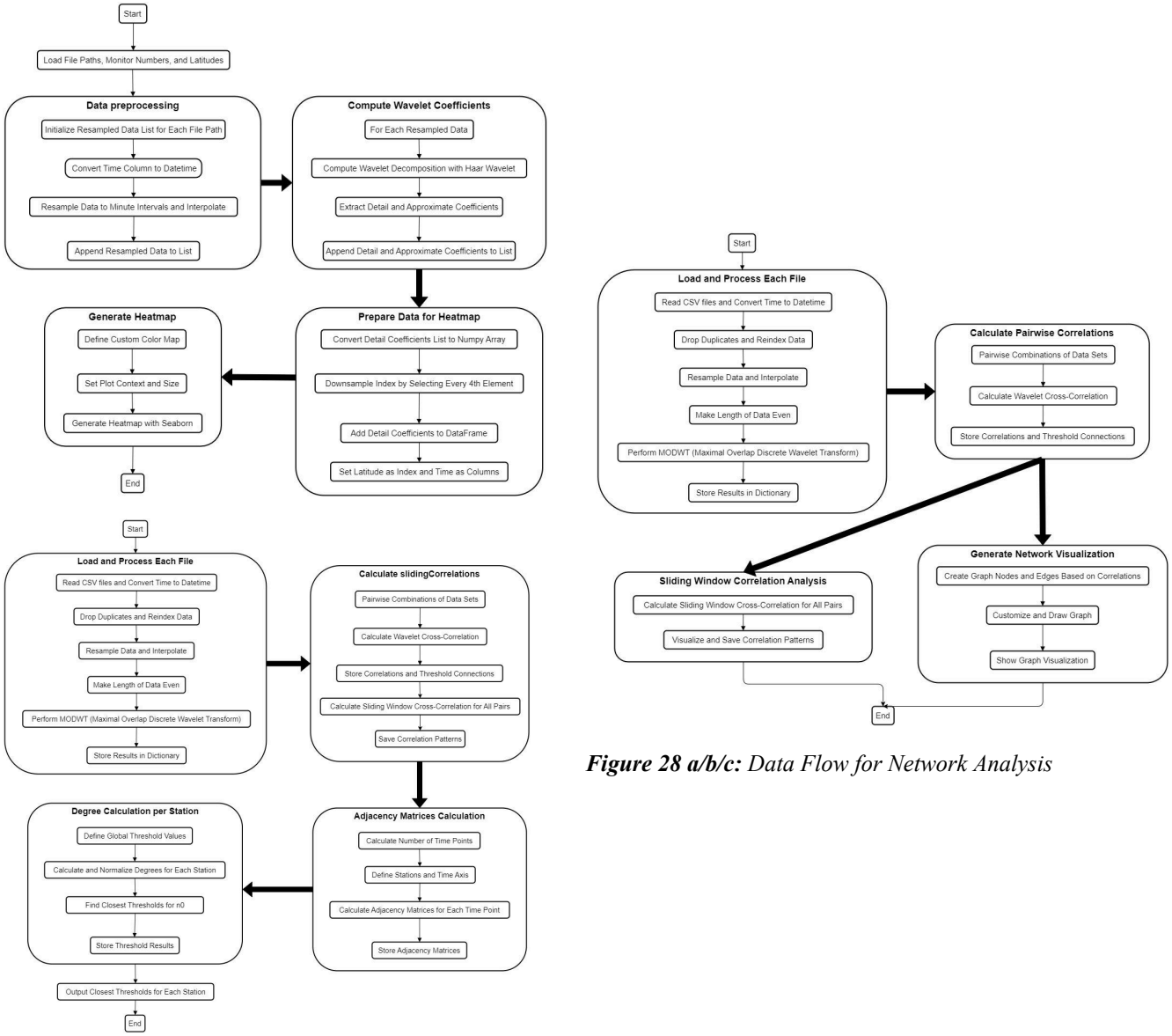



***Figure 28 a/b/c:*** *Data Flow for Network Analysis*

## Data Preprocessing

At the beginning of network analysis, data preprocessing was performed to ensure data quality and consistency. Geomagnetic-induced current (GIC) data was downloaded from the NERC. Firstly, because the sampling rate of the original data is different, to unify the analysis conditions, the original GIC data observed from different monitors of one event were resampled for every one minute to make the data consistent in time. The missing data points were filled by using a three-order spline interpolation method which is a mathematical method used to create a smooth curve through a discrete set of data points.

## Wavelet Transform

The next step is to perform a wavelet transform on the GIC data. Due to the nonlinear and non-stationary characteristics of GIC data, wavelet transform was chosen as the main analysis tool. Wavelet transform is different from Fourier transform in that it can provide localized information in the time-frequency domain. This means that the wavelet transform can simultaneously give the frequency characteristics of the signal at a certain point in time, which is particularly important for the analysis of geomagnetic disturbance which changes rapidly in time. In our project, the Haar wavelet was performed on the discrete wavelet transform for the 4-level decomposition, which extracted detail coefficients and approximate coefficients respectively. The first-level coefficients (approximate coefficients) of decomposition captured the major trend of the signal and the second-level coefficients (detail coefficients) captured some rapid changes representing noise or sudden fluctuations. Then the results of wavelet transform were displayed in the form of heat maps, with different colors representing the different values of coefficients, making the results intuitive and easy to understand.

## Correlation

To gain insight into the interactions and effects between the different monitors, we calculated two types of correlations. Firstly, the maximum overlapping discrete wavelet transform was utilized to extract the wavelet coefficients. Then the correlations were calculated.

1. Global correlation analysis: Correlation coefficients between every two monitors over one event were calculated to identify overall interdependencies.
2. Sliding-window cross-correlation: Sliding window cross-correlation of time series by setting different time Windows were calculated to capture dynamic changes more finely

## Threshold optimization

In previous data processing, we have been using the global threshold to determine the connection of two monitors, which was simple and efficient to use. But this approach works better when all monitors are the same and evenly distributed. If this method is applied to our program, it will lead to severely distorted results: those sites with high coefficients are always connected to the network, while those with lower coefficients are rarely connected. Therefore, using a long time series to determine individual station thresholds will account for our local conditions. Firstly, the calculated value of sliding cross-correlations between each pair of stations will be applied to generate an intermediate adjacency matrix $A^*_{ij}$. By setting the global threshold, the connection above the threshold is 1, and the connection below the threshold is 0. The equation for $A^*_{ij}$ is:

$$A^*_{ij}(C_T, t) = \Phi(|C_{ij}(t) - C_T|)$$

where $C_{ij}(t)$ is the correlation between monitor i and monitor j at time t, and $C_T$ is the global threshold. Then the average degree $n_i(C_T)$ at different global threshold $C_T$ values for each site need to be calculated:

$$n_i(C_T) = \left(\sum_t^T \sum_j^{N(t)} \frac{A^*_{ij}(C_T, t)}{N(t)-1}\right)/T$$

The purpose of this algorithm is to find a specific threshold $C_{Ti}$ for each monitor so that the calculated average degree of each monitor is equal to the normalization degree $n_0$. This requires iteratively adjusting $C_{Ti}$ for each monitor until the target average degree is reached. The next step is to determine two thresholds, $C_{Ti}$ and $C_{Tj}$ . If the lower threshold of the two sites is satisfied $C_{Tij}$

= min[$C_{Ti}$, $C_{Tj}$], it is considered that there is a connection between sites i and j. Finally, the specific threshold of each pair of monitors will be utilized to analyze the network.

## Results

After applying the Haar wavelet transform to the raw GIC data of even 2013_1002, two heat maps were obtained. Figure 29a shows the major trend of GIC data observed in 2013 Oct 2nd, where the color shows the magnitude of the first-level wavelet coefficients and latitudes represent different monitors. Figure 29b shows the rapid changes of GIC data observed in 2013 Oct 2nd, where the color shows the magnitude of the second-level wavelet coefficients, and the redder the color the stronger the change.

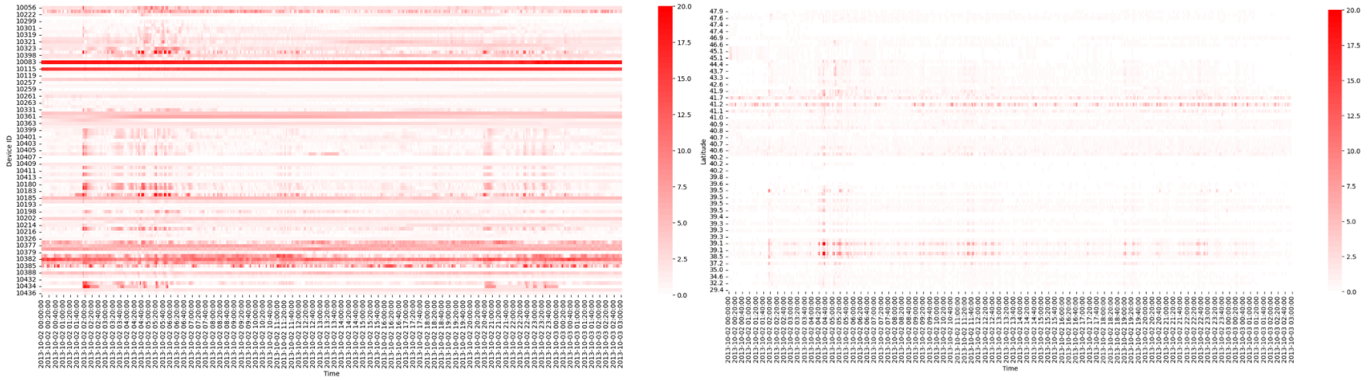


***Figure 29 a/b:*** *Wavelet heatmaps of first and second level coefficients for each device.*

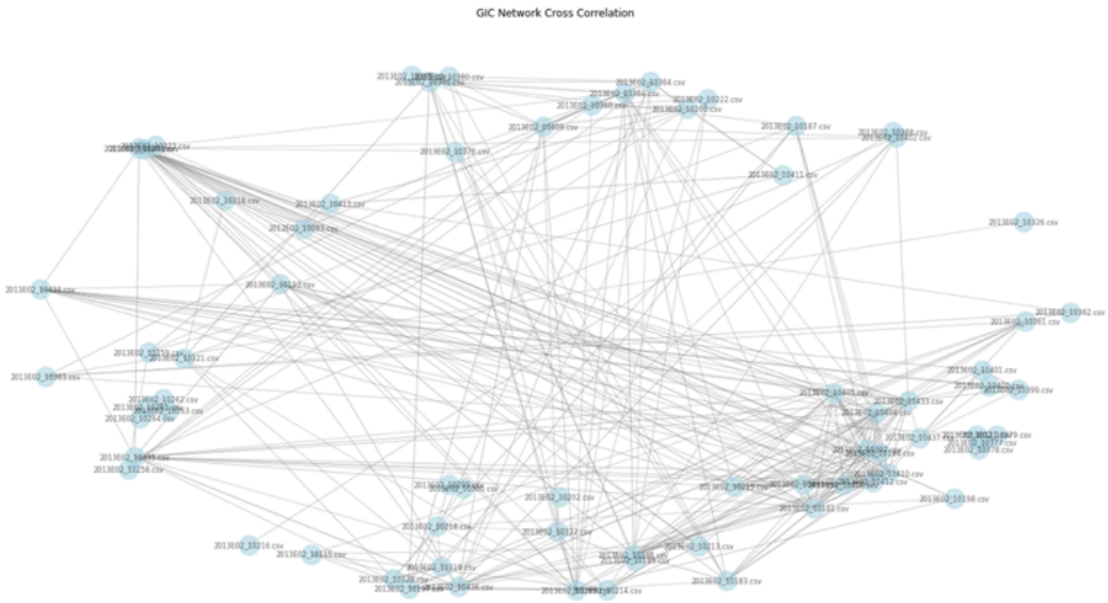



***Figure 30:*** *One-time cross correlation between coefficients.*

Figure 30 is to obtain a one-time cross-correlation on different scales of the wavelet transform. It shows a network diagram where the nodes represent different monitors and lines between nodes show correlations. The more nodes are wired, the higher the correlation with the other monitors.

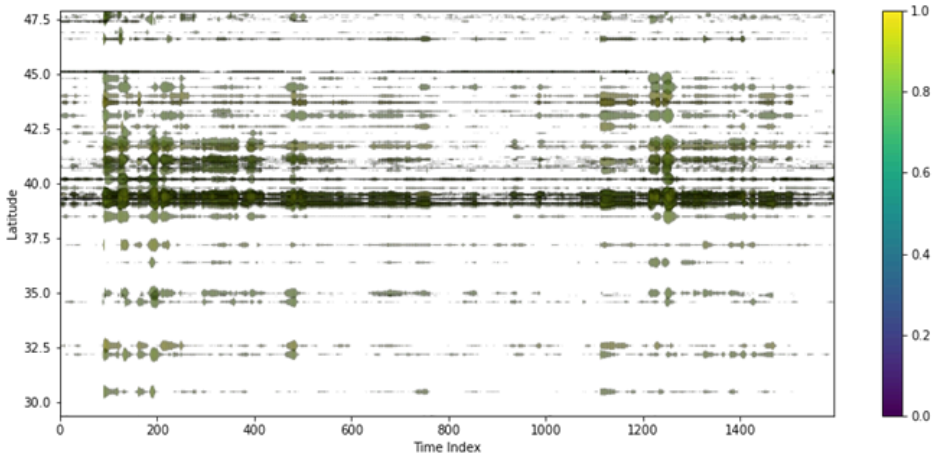



***Figure 31:*** *Time-based heatmap showing the relationship between GIC devices*

Figure 31 is a time-based heatmap showing the relationship between GIC devices. The size of the circles indicates the local degree of each node (number of connections to that node in the network at that time). Only cross-correlation greater than 0.8 will increase the node degree at this time point. Then the color plots the magnitude of the correlation, and it will be presented only when the correlation of the node must both (i) exceed 80% of all values obtained at that node and (ii) the wavelet cross-correlation between a pair of nodes must exceed 0.85 within a 30-minute leading edge window, otherwise it will show a circle with no color.

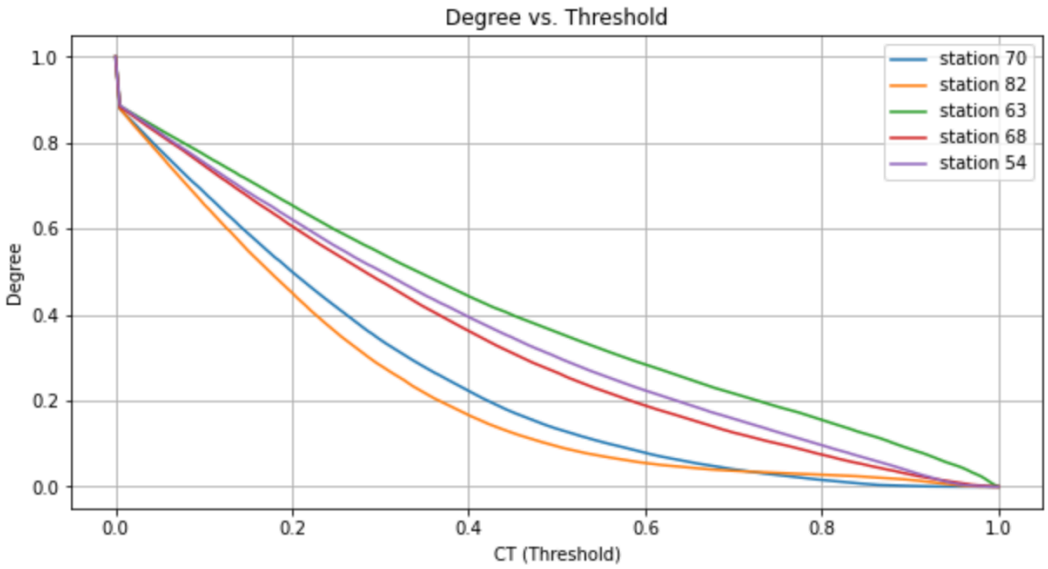



***Figure 32:*** *Inverse trend between between degree of iteration and device-specific correlation thresholds for 5 different GIC devices or stations*

Figure 32 shows the relationship between degree and threshold after applying the degree algorithm. Then the $n_0$ was set to equal 0.1, we could find different thresholds for each monitor. Finally, we could use it to find a new correlation between each monitor and make a clearer comparison with the physical power grid.

# Machine Learning

Our objective is to model and predict GIC events to mitigate the impact on the critical infrastructure. Thus, developing nowcasting and predictive models for magnetic perturbation prediction (dB/dt) and GICs required the most meaningful ML algorithms. Our approach was to divide efforts into two different directions for prediction: one is predicting magnetic perturbation (dB/dt) and other is predicting GIC values directly. The reason why we had 2 directions is that “since GIC data is proprietary, the time variability of the horizontal component of the magnetic field perturbation (dB/dt) is used as a proxy for GICs” (Upendran, V. et al., 2022).

## Preprocessing

The preprocessing dataset involved three raw datasets SuperMAG (2010 - 2023), OMNI (2010 - 2023) and NERC GIC (2013 - 2023). As we have predictive direction, we prepared two different type of dataset: one is for magnetic perturbation prediction (joined between SuperMAG and OMNI dataset) and other is for GIC values prediction (joined between SuperMAG, OMNI and NERC GIC dataset).

## Dataset for Magnetic Perturbation Prediction (dB/dt)

The dataset was created by joining the datetime between SuperMAG and OMNI dataset, both ranging from 2010 - 2023. The training dataset ranged from 2010 - 2019 and the testing dataset ranged from 2020 - 2023 (Upendran, V. et al., 2022). The diagram below shows the details of the data flow from collecting raw data to the complete data set files stored in Google Drive.

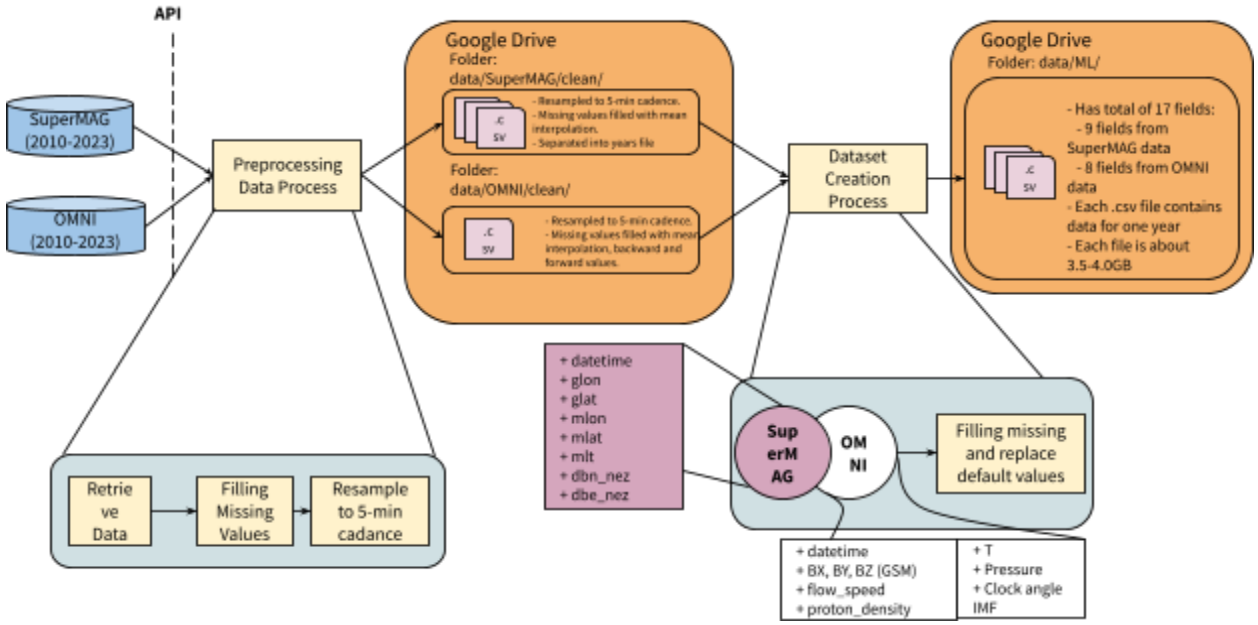



***Figure 33**: Diagram showing the process of creating ready for analysis*

## SuperMAG Dataset (dB/dt)

SuperMAG is a global network of around 300 magnetometer ground stations employed in measurement of geomagnetic perturbations. The available dataset that we use in this study provides the geomagnetic perturbations measurements of 1-min cadence from around 300 stations around the globe. From the SuperMAG data, we are using the following measurements: *datetime, glon, glat, mlon, mlat, mlt, dbn_nez* and *dbe_nez*. But since the provided data are in 1-min cadences and for the purpose of stability and synchronization with OMNI dataset, we resampled the provided into 5-min cadences. Our preprocessing data also includes filling missing values, remove outliers and replacing the default values in the provided data

## OMNI Dataset

OMNI dataset provided near-Earth solar wind magnetic field and plasma parameter data from several spacecraft in geocentric or L1 (Lagrange point) orbits. From OMNI data, we retrieved 5-min cadence data since we believe that 5-min cadence data can provide more stability. And indeed, the fields that we utilized from OMNI data are: *datetime, bx_gse, by_gsm, bz_gsm, flow_speed, proton_density, T, pressure* and *clock angle of the IMF.*

## Datasets NERC GIC Prediction

The dataset was created by joining the GIC event datetime and by all the devices sensors of GIC measured within the 500 miles radius of the station where Magnetic perturbation measured showed in figure , from the NERC GIC and Supermag datasets, both ranging from 2013 - 2023. The training dataset is ranged from 2013 - 2022 and the testing dataset 2023.

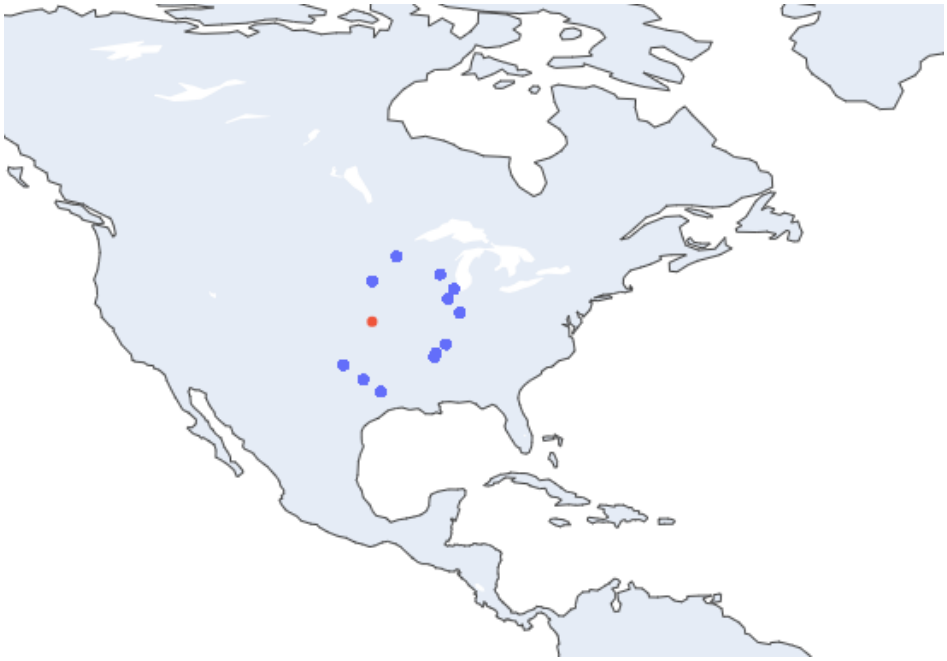


**Figure 34**: Stations and Devices sensor relationship where Station is marked Red and Station is blue

### NERC Geomagnetically Induced Current (GIC)

The North American Electric Reliability Corporation (NERC) has a program to collect data on geomagnetic disturbances (GMDs). These disturbances occur when the sun ejects charged particles that interact with the Earth's magnetic field and atmosphere. When this interaction occurs, it can cause changes in the Earth's magnetic field, which might disrupt or damage important infrastructure, like power systems. In rare but strong GMD events, these changes can create strong direct currents in the electric power grid. These currents can affect the stability of the system, interfere with protective devices, and potentially damage large power transformers. This is why NERC is collecting data to better understand and manage the risks associated with GMDs.

### SuperMAG Dataset (dB/dt)

As described above, we want to understand the geomagnetic perturbations that have an effect on the GIC sensor measurements. Similarly to our other prediction on the magnetic perturbation, we are using the following measurements: *datetime, glon, glat, mlon, mlat, mlt, dbn_nez* and *dbe_nez* from the SuperMag datasets to predict the Nerc GIC.

## Training & Testing Models (Results)

Since we have 2 predictions: Magnetic perturbations and GICs. The main 3 models we are using are Multivariate Linear Regression, Random Forest Regressor and Long Short Term Memory Models

### Magnetic Perturbation Models

#### Multivariate Linear Regression (MLR) Model

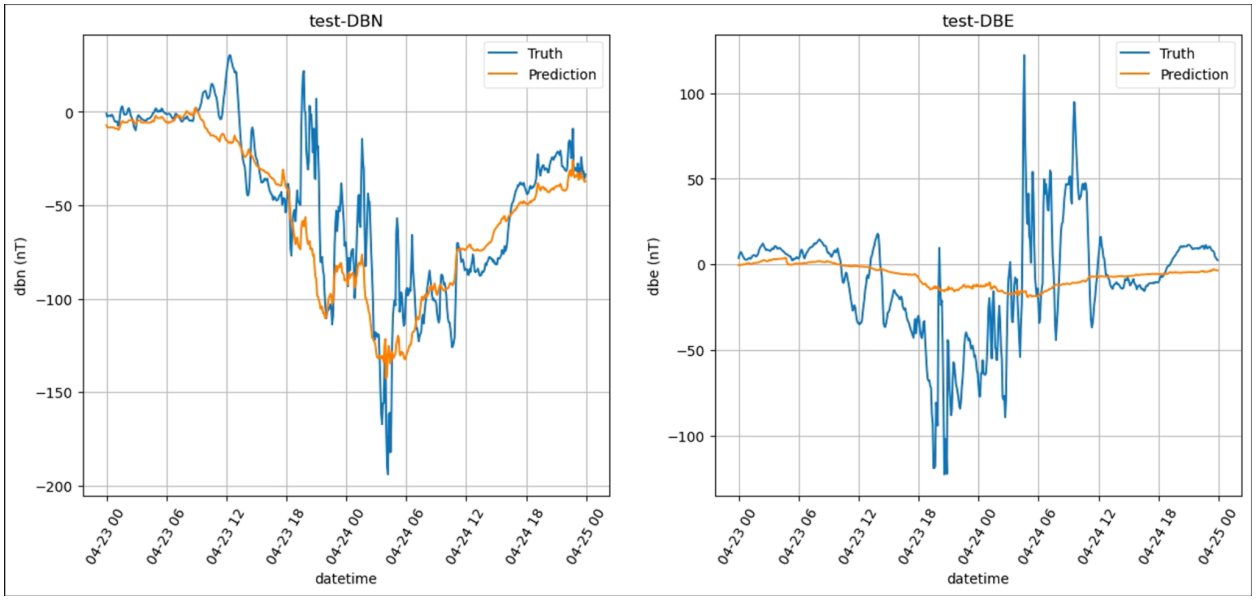
The Multivariate Linear Regression model is the first model that tried to understand the underlying data structure of our dataset. It also served as a baseline model for us to compare performance and figure out ways to improve performance of the models trained later on this report.

#### Results and Findings for Magnetic Perturbation (db/dt)

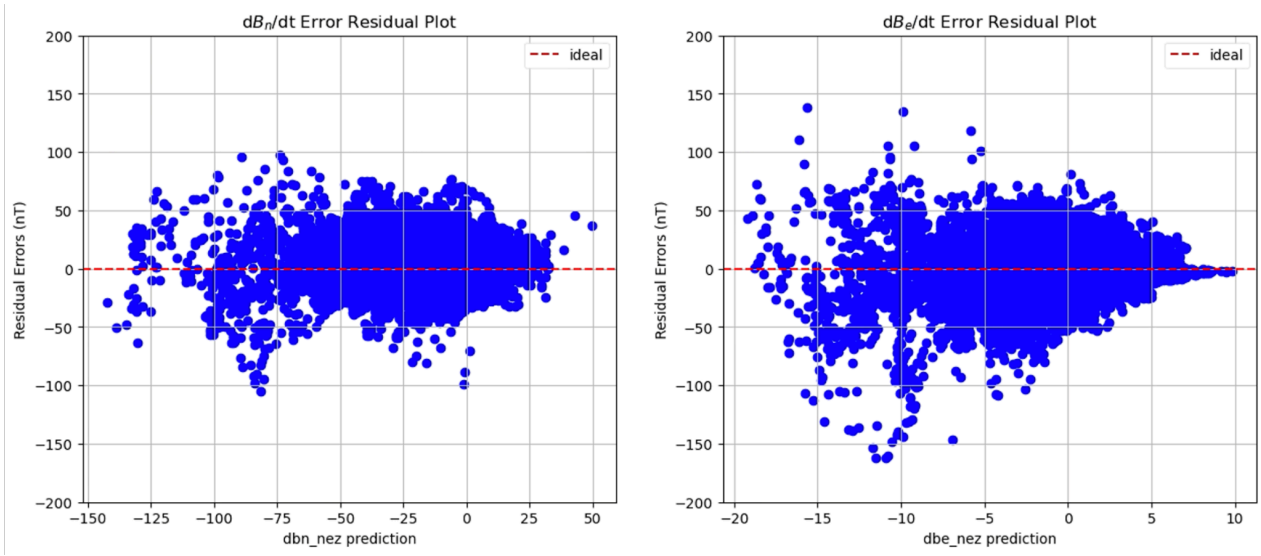
For Multivariate Linear Regression, we trained models for the dataset timeline from 2010 - 2019 and testing in 2020 - 2023. Below will be some graphs of our results:

*Table 1: RMSE and MAE value for training and testing dataset*

| | Train | | Test | |
|---|---|---|---|---|
| | dBn/dt | dBe/dt | dBn/dt | dBe/dt |
| RMSE | 7.357 | 7.932 | 7.530 | 7.702 |
| MAE | 5.095 | 4.954 | 5.298 | 4.905 |




***Plot 1**: Multivariate Linear Model - Plot of truth vs prediction zoom-in two days range of test dataset.*




***Plot 2**: Multivariate Linear Model - Residual Error Plot for dBn/dt and dBe/dt prediction*

Throughout the results, we recognize that multivariate linear regression is not really doing great in the dataset, especially for the dBe/dt prediction. From the metrics table, we can see that both values for RMSE and MAE of dBe/dt in training and testing dataset are both all higher than those values of dBn/dt. This tells us that the model performs worse on the dBe/dt prediction. Look into plot 2, the line plot of truth vs prediction, in this plot we have zoom into 2 days range data of prediction and we can see that the model is not able to capture the dynamics of dBe/dt really well, meanwhile, it's able to follow the trend that's happening in the dBn/dt. One of the

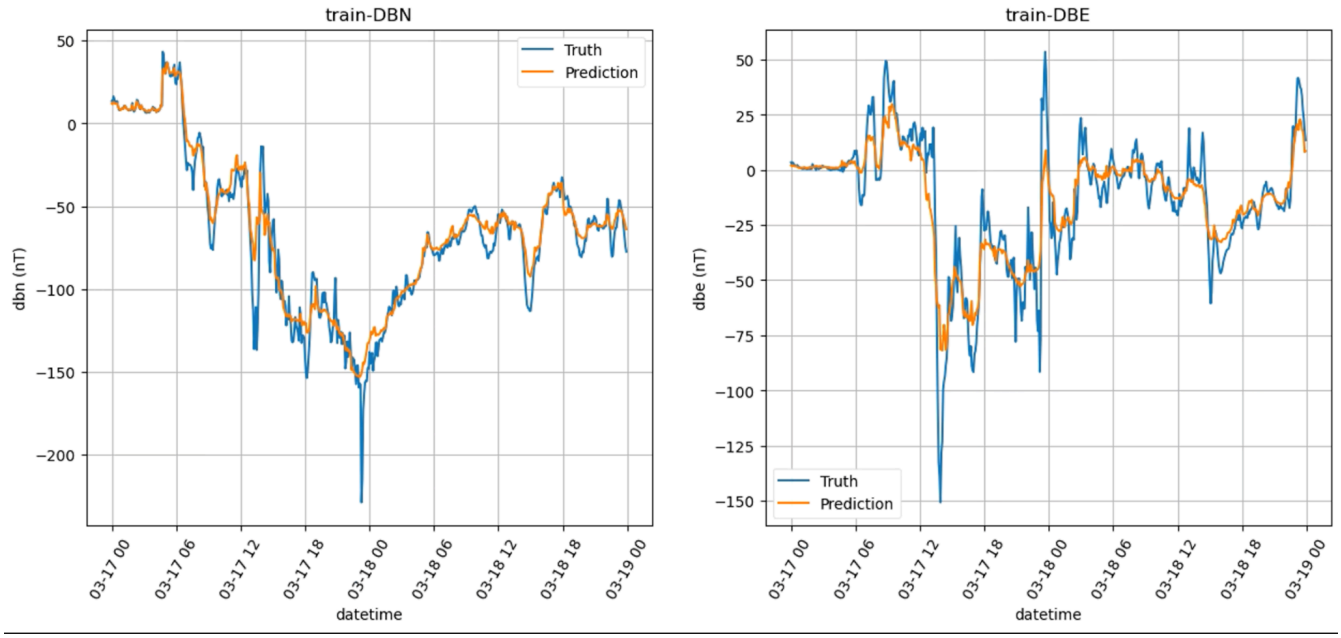
hypotheses that we have is that our multivariate linear regression is not able to capture the non-linear relationship between the features. Moreover, since we're working with the time series data, the order of the data point and historical information are also really important. Since that, we have experimented with Ranform Forest, which is known for capturing non-linear relationships between features, and Long Short Term Memory (LSTM), which is known for keeping the order of data points and maintaining information in memory for longer periods. We have also shown the results and analysis for those models in later sections below.

### Random Forest Regressor (RFR) Model

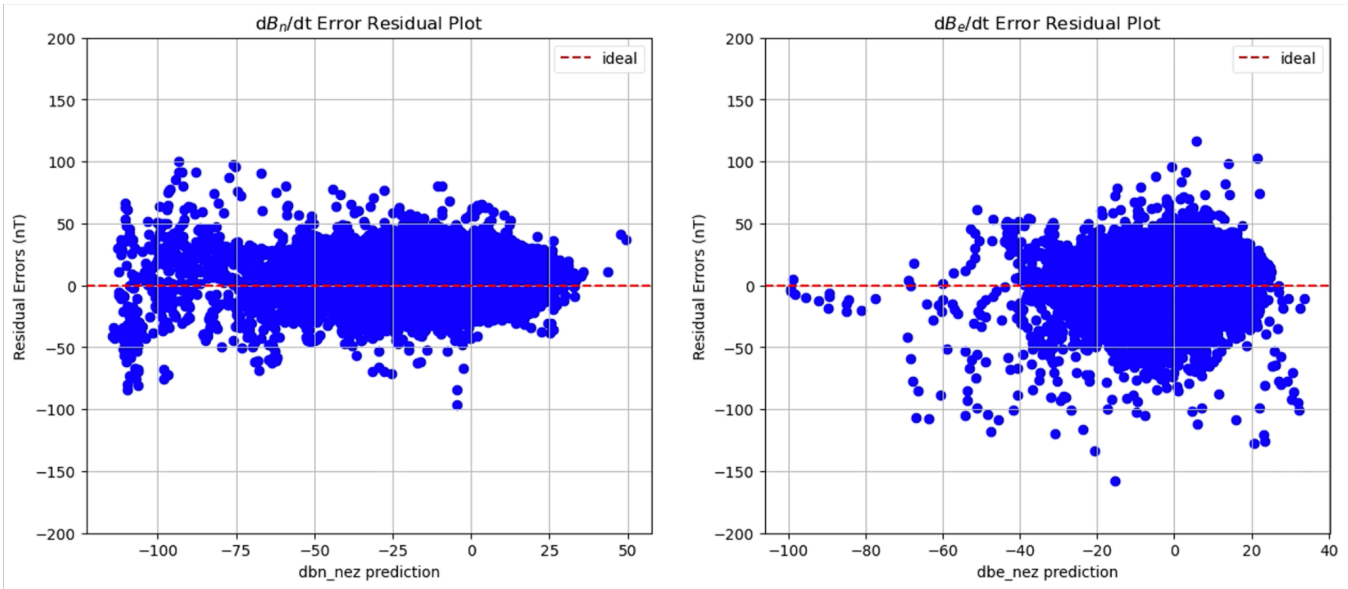
### Hyperparameters

- max_depth = 90
- max_features = 0.5
- min_samples_leaf = 8
- min_samples_split = 5
- n_estimators = 911
- Loss = RMSE

### Results and Findings for Magnetic Perturbation (db/dt)




***Plot 3**: Random Forest Model - Plot of truth vs prediction zoom-in two days range of test dataset.*




***Plot 4**: Random Forest Model - Residual Error Plot for dBn/dt and dBe/dt prediction*

RFRM resulted in mediocre performance in predicting magnetic perturbation values (nT) for the North and East axes.

However, the results are essential in understanding the dynamics and the characteristics of the data. The model captures low-magnitude and non-linear trends, however, fails to learn the high-magnitude dynamics of the data.

## Long Short Term Memory (LSTM) Model

LSTM networks are a type of recurrent neural network (RNN) particularly well-suited for time series forecasting, sequential data, and tasks where the order of data points matters. They were designed to overcome the limitations of standard RNNs. Compared to RNNs, they have a more complex architecture that allows them to maintain information in memory for longer periods.

The core components of LSTM are cell state and gates. Figure1 illustrates the mechanisms inside a LSTM unit, where i, f, g, and o denote the input gate, forget gate, cell candidate, and output gate, respectively.

| Component | Formula |
|---|---|
| Input gate | $i_t = \sigma_g(W_i \mathbf{x}_t + \mathrm{R}_i \mathbf{h}_{t-1} + b_i)$ |
| Forget gate | $f_t = \sigma_g(W_f \mathbf{x}_t + \mathrm{R}_f \mathbf{h}_{t-1} + b_f)$ |
| Cell candidate | $g_t = \sigma_c(W_g \mathbf{x}_t + \mathrm{R}_g \mathbf{h}_{t-1} + b_g)$ |
| Output gate | $o_t = \sigma_g(W_o \mathbf{x}_t + \mathrm{R}_o \mathbf{h}_{t-1} + b_o)$ |

By default, LSTM uses sigmoid activation to produce a filter for the information. Each gate produces different function:

- Forget Gate: Decides what information to discard from the cell state.
- Input Gate: Determines which new information to store in the cell state.
- Output Gate: Controls what information to output from the cell state to the next time step.
- Cell candidate: The cell state is updated with new information scaled by the input gate and old information scaled by the forget gate.

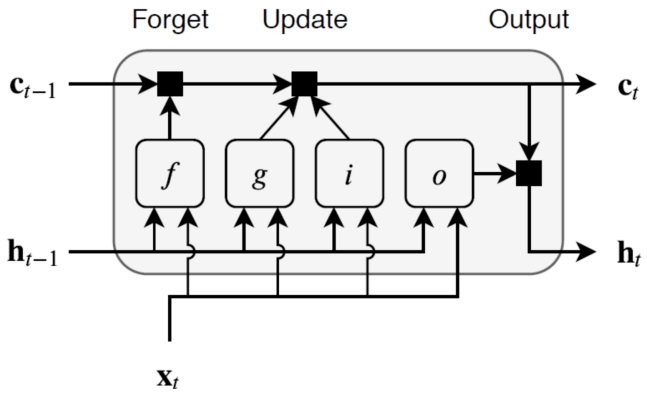



***Figure 35:*** *Standard LSTM unit*

Figure 36 illustrates the sequential data process inside a LSTM layer, through this unique architecture, LSTM comes out to be a powerful tool for capturing long-term dependencies and performing future prediction.

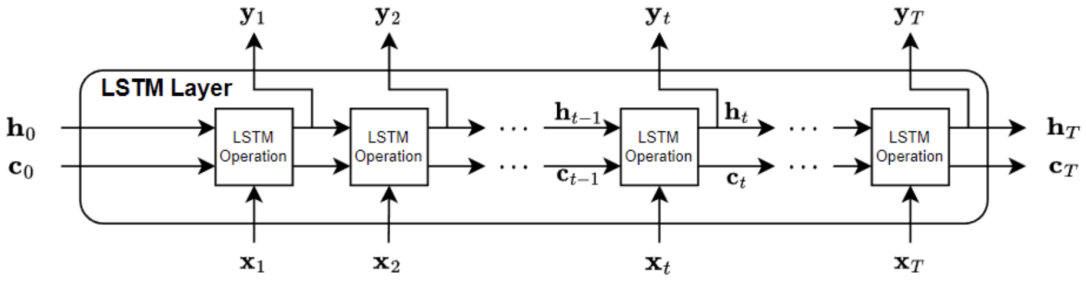
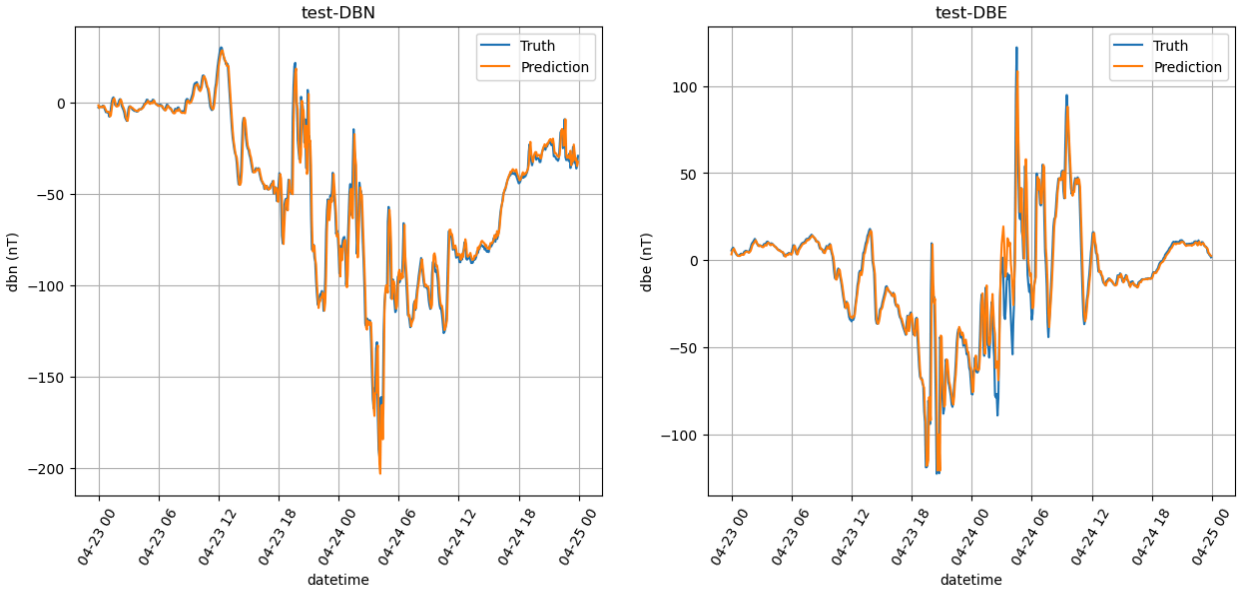
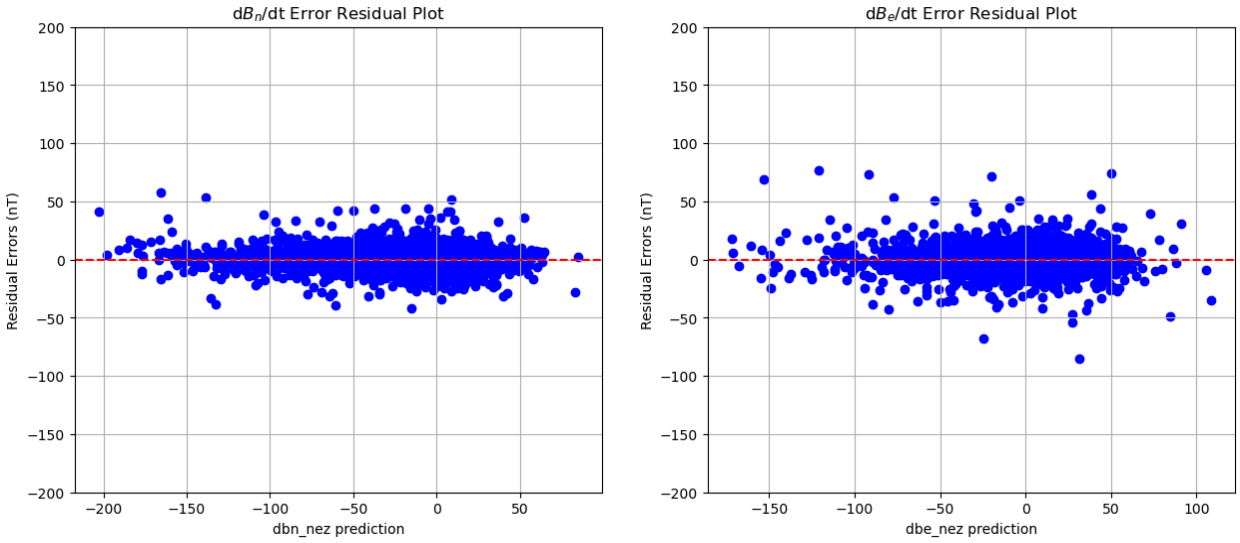



***Figure 36:*** *Standard LSTM Layer*

## Results and Findings for Magnetic Perturbation(Db/dt)




***Plot 5**: Long-Short Term Memory Model - Plot of truth vs prediction zoom-in two days range of test dataset.*




***Plot 6**: Long-Short Term Memory Model - Residual Error Plot for dBn/dt and dBe/dt prediction*

Our model was trained on the 2010-2019 dataset while validated on 2020 dataset. Our current configuration is: Batch_size = 360, Activation: ReLu, Optimization = Adam, Loss = MSE, Epochs = 50. Through the training process, we find out:

- Batch size significantly affects the performance and efficiency of our LSTM model. A smaller batch size induces a slower and less stable training process, while a larger batch size leads to poorer generalization.
- The default sigmoid activation did not work well on our dataset, instead, ReLu activation converged faster and achieved a lower training and validation loss.
- A single layer can achieve satisfying training performing on a single time step while multiple time steps require more complicated layers setting.

## NERC GIC models

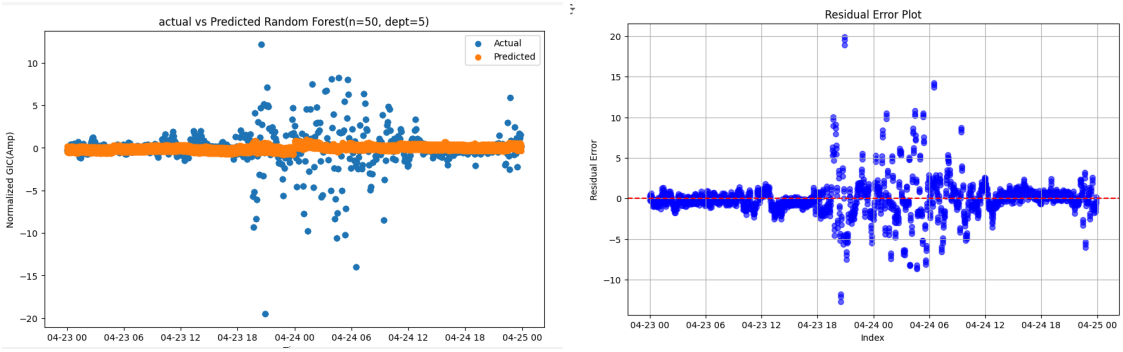
All the models are using a dataset of the station in Washington DC(39.0, 281.8) where the NERC GIC values are the most extreme. In this case, we can see how each of the models' performances on prediction compare to the actual values of the GIC sensor.

### Multivariate Linear Regression (MLR) Model

Hyperparameters

- Default Parameters from keras.

Results and Findings:




***Plot 7 & 8**: Multivariate Linear Regression Model - GIC Predictions + Residual Error Plot*

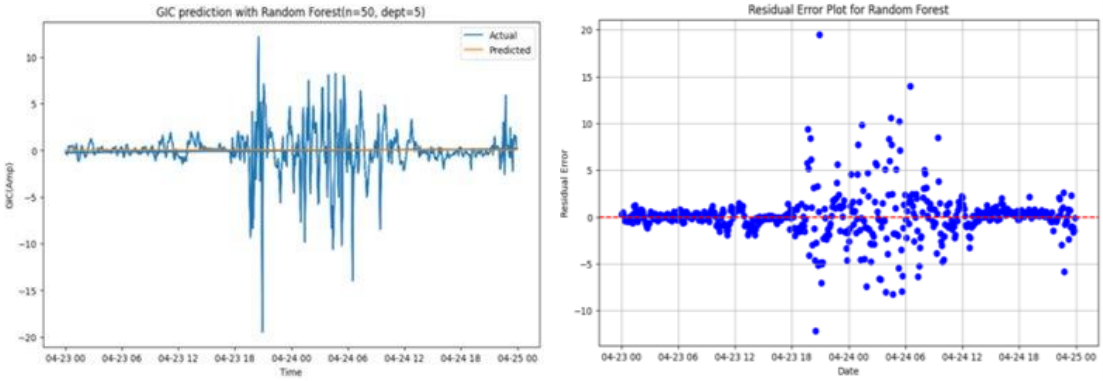
The model is able to pick up some trends of the data, which indicates we can definitely improve the models through hyperparameters.

## Random Forest Regressor (RFR) Model

### Hyperparameters:

- n_estimators=50
- max_depth=5

### Results and Findings




***Plot 9 & 10**: Random Forest Regressor Model - GIC Predictions + Residual Error Plot*

This model seems to be flat prediction at around 2 Amperes. Similar to the Linear Regression Model, my prediction is that because we are overpopulated the normal data compare to when the GIC events extreme happened, which cause the models to predict at 2 amperes which give us the most accurate overall but result incorrect.

## Long Short Term Memory (LSTM) Model

### Layers of Neural Network

- LSTM(128, input_shape=(1, X_train_normalized.shape[1]))
- Dense(64, activation='relu')
- Dense(32, activation='relu')
- Dense(1)

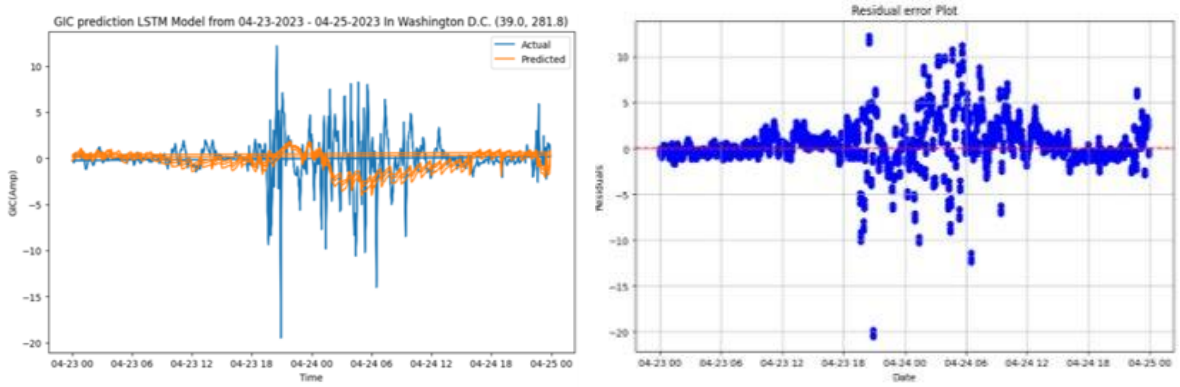
### Model evaluation:

- optimizer='adam'
- loss='mse'

### Model training:

- 5 epoch

### Results and Findings




***Plot 11 & 12**: Long-Short Term Memory Model - GIC Predictions + Residual Error Plot*

The LSTM model seems to be able to pick up the trend of the actual signal, which gives us an indication of these particular models performing better when it comes to our time series dataset prediction compared to random forest and linear regression.

# Project Success Criteria

Ability to accurately predict the next solar events and its consequential effects on the U.S power grid is, in itself, the ultimate goal of this project. To achieve this would require greater collaboration in form of feedback from various power companies within the United States as well as the inputs from the non-profit entities in charge of the SuperMAG, OMNIWeb, and NERC datasets. Additionally, it would require a significant solar event to have occurred that caused significant damage to report to the public. Thus, our success criteria for our project is to develop a framework of analysis based on the public datasets currently available, for other collaborators to imitate and/or derive methods of analysis for their specific situations. This would be in the form of a public github repository, that clearly outlines the procedures of the extraction, loading, transformation, and analysis. Additionally, this github repository will also outline the different diagrams and visualizations that may be useful for other applications.

**Goals:**

| | |
|---|---|
| Created a public github repository | (completed) |
| Create analysis ready datasets | (completed) |
| Create a data pipeline to efficiently extract data from various sources | (completed) |
| Conduct extreme value analysis to provide 1/100th estimates | (completed) |
| Provide Network Analysis within the United States Power Grid | (completed) |
| Machine Learning Models have +80% validation accuracy | (1 of 3 models) |

# Impact and Consequences

## Environmental

- **Positive Effects:**
    - We hope our project can highlight the different vulnerability points in our U.S power grid in order fortify this infrastructure with sustainability in mind. Despite the unlikelihood of a massive solar event causing a wide-scale blackout, the ability to take preemptive measures to minimize the cost if such an event does occur can equate to billions of dollars in savings with an unquantifiable amount of lives saved.
- **Negative Effects:**
    - With many machine learning pipelines, it should be acknowledged that there will likely be an increase in carbon emissions due to increased use of electricity from the intense computations needed for machine learning applications. Understanding how certain cloud services derive their electricity can help mitigate the carbon emission by allocating compute resources to specific geographical areas.

## Ethical

- **Commercialization**
    - Due to overlapping licensing and acknowledgement requirements, this path of commercialization can restrict the flow of information and prevent public use. Thus, it should be evident that this project shall remain open and free-of-use by indicating use under licensing.
- **Malicious Intent**
    - Assuming the accuracy of our machine learning models, the information and analysis provided can provide an understanding of the vulnerabilities of the U.S power grid and other infrastructure. Bad actors with malicious intent can seek to pinpoint weak areas, targeting them to disrupt commerce and sustainability. Disclaiming the use of our analysis for malicious intent can be a legal remedy in preventing this ill-intended use.

- **Validation of Efforts**
    - To legitimize our work, we may be enticed to promote the validity of our machine learning models. However, until the accuracy of machine learning model predictions are validated through data from catastrophic events, it is difficult to validate our models. Additionally, it may be unethical to hope for a catastrophic solar event to validate our analysis.

# Conclusions and Recommendations

We hope this research will serve as a framework for creating machine learning models in predicting effects of GICs and space weather. Although our highest performing machine learning model (i.e LSTM) is limited to a 48-hour nowcasting timeframe, we believe this performance can be extrapolated to greater timeframes given the additional resources to train and compute. Additionally, we believe traditional analyses (such as Extreme Value Analysis and Network Analysis) can help guide the feature engineering and inputs selections of training machine learning models. Translating qualitative results of these analyses into quantitative measures may present future work that may be more challenging, yet more effective in creating smarter and more accurate algorithms. We encourage free and open use of this research to anyone who is interested and willing to take the challenge.

This project successfully delivered a final presentation and github repository to NASA JPL and USGS on 09 MAY 2024. *See submitted video for the recording.*

Our Github repository can be found at:
https://github.com/jung2shinho/NASA_JPL_UW_ENGINE_24

# References


1. S. Coles. “An Introduction to Statistical Modeling of Extreme Values”. Department of Mathematics, University of Bristol, Bristol, UK. November 2013. doi 10.1007/978-1-4471-3675-0
2. Tindale, E., & Chapman, S. C. (2017). Solar wind plasma parameter variability across solar cycles 23 and 24: From turbulence to extremes. *Journal of Geophysical Research: Space Physics*, 122, 9824–9840. doi 10.1002/2017JA024412
3. A. L. Barabasi. “Network Science: The Power of Network Science, The Beauty of Network Visualization”. Cambridge University Press. 2015
4. C. M. Ngwira, A. A Pulkkinen. “An Introduction to Geomagnetically Induced Currents”. *American Geophysical Union*. *Geomagnetically Induced Currents from the Sun to the Power Grid*. September 2019. doi 10.1002/9781119434412.ch1
5. R. M. McGranaghan, J. Ziegler, T. Bloch, S. Hatch, E. Camporeale, K. Lynch, M. Owens, J. Gjerloev, B. Zhang, S. Skone. “Toward a Next Generation Particle Precipitation Model: Mesoscale Prediction Through Machine Learning (a Case Study and Framework for Progress)”. *Space Weather. Vol 19. Issue 6*. June 2021. doi 101029/2020SW002684
6. R. M. McGranaghan, A. J. Mannucci, B. Wilson, C. A. Mattmann, R. Chadwick. “New Capabilities for Prediction of High-Latitude Ionospheric Scintillation: A Novel Approach with Machine Learning”. *American Geophysical Union. Space Weather. Vol. 16. Issue 11*. doi 10.1029/2018SW002018
7. E. Camporeale. “The Challenge of Machine Learning in Space Weather: Nowcasting and Forecasting”. *American Geophysical Union. Space Weather. Vol 17. Issue 8. pp 1166-1207.* July 2019. doi 10.1029/2018SW002061
8. C. J. Schriver, K. Kauristie, A. D. Aylward, et al. “Understanding Space Weather to Shield Society: A Global Road Map for 2015-2025 Commissioned by COSPAR and ILWS”. *Science Direct. Advances in Space Research. Vol 55. Issue 12. pp. 2745-2807. June 2015. doi 10.1016/j.asr.2015.03.023*
9. L.Orr, S. C Chapman, C. D. Beggan, “Wavelet and Network Analysis of Magnetic Field Variation and Geomagnetically Induced Currents During Large Storms”. *American*

*Geophysical Union. Space Weather. Vol. 19. Issue 9.* August 2021. doi: 10.1029/2021SW002772

10. J. Dods, S. C. Chapman, J. W. Gjerloev, "Network Analysis of Geomagnetic Substorms Using The SuperMAG Database of Ground-based Magnetometer Stations". *American Geophysical Union*. *Journal of Geophysical Research: Space Physics.* Vol. 120. Issue 9. pp. 7774-7784. August 2015. doi 10.1002/2015JA021456
11. V. Upendran, P. Tigas, B. Ferdousi, T. Bloch, M. C. M. Cheung, S. Ganju, A. Bhatt, R. McGranaghan, Y. Gal. "Global Perturbation Forecasting Using Deep Learning". *American Geophysical Union. Space Weather*. Vol 20. Issue 6. May 2022. doi 10.1029/2022SW003045
12. M. Blandin, H. Connor, D. Ozturk, A. Keesee, V. Pinto, M. S. Mahmud, C. Ngwira, S. Priyadarshi. "Multi-variate LSTM Prediction of Alaska Magnetometer Chain Utilizing a Coupled Model Approach". *Frontiers in Astronomy and Space Sciences.* Vol 9. May 2022 doi 10.3389/fspas.2022.846291
13. J. Hughes, R. McGranaghan, A.C Kellerman, J. Bortnik, R.F Arrit, K. Venkataramani, C.H Perry, J. McCormick, C.M. Ngwira, M. Cohen. "Revealing Novel Connections Between Space Weather and the Power Grid: Network Analysis of Ground-Based Magnetometer and Geomagnetically Induced Currents (GIC) Measurements". *American Geophysical Union. Space Weather. Vol. 20. Issue 2.* December 2021. doi 10.1029/2021SW002727
14. J. Bergstra. Y. Bengio. "Random Search for Hyper-parameters Optimization" *Journal of Machine Learning Research 13 (2012) 281-305.* February 2012
15. A. C. Kellerman, R. McGranaghan, J. Bortnik, B. A. Carter, J. Hughes, R. F. Arritt, K. Venkataramani, C. H. Perry, J. McCormick, C. M. Ngwira, et. al. "Geomagnetically Induced Currents at Middle Latitudes: 1. Quiet-Time Variability" *American Geophysical Union. Space Weather. Vol. 20 Issue 2.* February 2022. doi 10.1029/2021SW002729
16. J. J. Love, B. S. Murphy, "North American Electricity Power-Grid and Communication-Network Anomalies for Several Magnetic Storms". U.S. Geological Survey. doi 10.5066/P9N4DVNT.

17. J. J. Love, S. F. Wilbur, “Voltages Measured on Long Grounded Electrically Conducting Lines in North American During Several Magnetic Storms”. 1891-1940: U.S Geological Survey Data Release. doi 10.5066/P9NEQGVB
18. M. W. Liemohn, J. P. McCollough, V. K. Jordanova, C. M. Ngwira, S. K. Morley, C. Cid, W. K. Tobiska, et. al. “Model Evaluation Guidelines for Geomagnetic Index Predictions”. *American Geophysical Union. Space Weather. Vol 16. Issue 12 pp 2079-2102*. doi 10.1029/2018SW002067
19. H. Winter, “Hydroinformatics for Hydrology: Extreme Value Modelling” *EDF Energy UK R&D Centre*. April 2015.

## Datasets

- SuperMAG (1970-present)
  - Super Magnetospheric Automated Geophysical Observatory (SuperMAG)
  - Provides global magnetic field observations from multiple international entities, organized by John Hopkins Applied Physics Laboratory.
  - Website: https://supermag.jhuapl.edu/
- OMNIWeb NASA (1963-present)
  - Observations and Measurements of the Interplanetary Medium (OMNI)
  - Provides interplanetary magnetic field (IMF) data, solar energetic particle (SEP) data, geomagnetic indices, and other field data from spacecraft in geocentric orbit detecting solar winds
  - Website: https://omniweb.gsfc.nasa.gov/
- NERC GIC (2013-present)
  - North American Electricity Reliability Corporation (NERC) Geomagnetic Disturbance Incidents (GMD)
  - Provides geomagnetically induced currents (GICs) on the U.S electrical power grid.
  - Website: https://www.nerc.com/pa/RAPA/GMD/Pages/GMDHome.aspx

# Acknowledgements

Special thanks to Ryan McGranaghan and Jonathan Sauder from NASA Jet Propulsion Laboratory (JPL), California Institute of Technology, and to Jeffrey Love from U.S Geological Survey (USGS) for their industry mentorship. Another special thanks to Arindam Kumar Das from the University of Washington and Office of Technology, Infusion and Strategy (OTIS) from JPL for sponsoring this project.

Lastly, we gratefully acknowledge the SuperMAG, OMNIWeb, and NERC for allowing access and availability of space weather data on the public domain.

# Appendices

- "OMNIWeb Data Explorer." NASA Goddard Space Flight Center. https://omniweb.gsfc.nasa.gov/
- North American Electric Reliability Corporation (NERC) GIC database: https://www.nerc.com/pa/RAPA/GMD/Pages/GMDHome.aspx
- Jack Eddy Symposium Risk Resiliency (Github Repository) https://github.com/jack-eddy-symposium/risk-resiliency-spwx