Solar Disturbance Storm Time Project Report ADS2002

Aiden, Batheendra, Ben, Marco, Ruchir, Shesh, Youjing

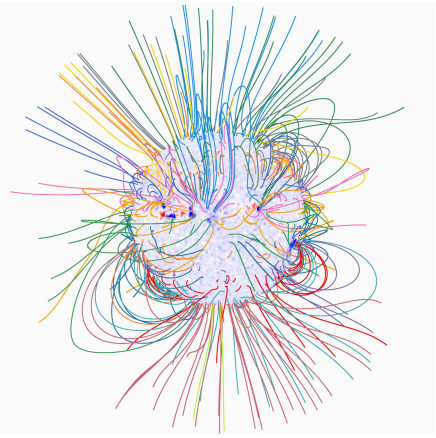
Table of Contents

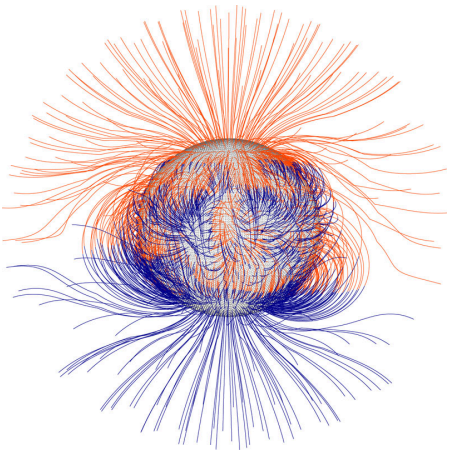
Executive Summary

Background Information and Context

The sun's **magnetic field** (see figure 1) works very differently than Earth's (see figure 2), its complex **multipolar** nature can cause significant trouble for Earth and other planets in the solar system, trouble mainly in the form of **Solar Storm**s! These storms begin with the formation of highly **concentrated magnetic fields** (sunspots) on the Sun's surface. The **magnetic field lines** between these sunspots become **highly stressed**, which causes them to snap, releasing enormous amounts of energy, radiation, and superheated material into space in the form of **Coronal Mass Ejections** (CMEs). These CMEs hurl **charged particles** into space, forming **solar wind.** When the solar wind reaches Earth, it **compresses our magnetosphere**, weakening it. This phenomenon is called a **geomagnetic storm**. We measure the strength of these storms using the **Disturbance Storm Time** (DST) index; it is an aggregated geomagnetic index/value measured from various **magnetometer stations** worldwide and is quantified in **nanoteslas** (nT). There are many different types and classes of storms but most can be **expressed in terms of severity,** the lower the DST index value, the worse the storm.

[Left figure 1: Helio-Magnetic Field (2021, "Metis Solar Orbiter")]
[Right figure 2: Geo-Magnetic Field(2020, Sciences)]





Accurate predictions can give stakeholders critical **time to prepare** for possible storms and mitigate important risks. Solar storms can disrupt power grids and electronics on Earth, and interfere with our orbital satellites creating problems for mobile phone coverage, GPS services, TV, radio, and internet connection. **The Victoria State Emergency Service** (VICSES) and the **Bureau of Meteorology** work together to warn the community about storms, and the military must be aware of solar flares to prevent **radio communication** disruptions. Governments also need to be prepared for **blackouts** and **potential internet outages** and space agencies can ensure their satellites won't fall out of the sky or have their astronauts and solar panels damaged by the **intense radiation**. Ultimately, predicting strong

geomagnetic storms is important to help **protect global communication, navigation, and power systems.**

For our project, the data utilized in this 12-week process was obtained from the **World Data Center for Geomagnetism in Kyoto** and the **Australian Bureau of Meteorology**. It encompasses form **DST index** records from **1957-2024** and **monthly visible sunspot count** for those years as well. The project aims to support future efforts in geomagnetic storm formation and improve global understanding of the impact of solar wind on Earth's magnetic fields, by using and comparing **recurrent neural network models** and **convolutional neural network models** and their capacity to be applied to **geomagnetic storm prediction.**

Approach and Results

Our project is focused on forecasting the DST (Disturbance Storm Time) index 1 to 6 hours in advance. Through the use of advanced machine learning models, the project intends to improve the prediction accuracy of forecasting geomagnetic storms. The main objective is to assess the prediction capabilities of these models on the DST time series data, evaluate their predictions, and finally analyse the accuracy by comparing the performance of these models with one another. Ultimately, aiming to identify the most effective approach to forecasting

The obtained historical DST index data is from the World Data Center for Geomagnetism, spanning from January 1 1957 to August 7, 2024 (see Figure 2). The columns provided are essential in understanding geomagnetic storms. Covering 592583 data points, the column DATE contains the timestamps for each observation. Formatted to represent the specific date and time when the measurement swerve was recorded. The TIME columns reflect hour intervals of geomagnetic data collection for possible temporal analysis. The DAY column indicates the day of the year offering clear references to seasons and daily patterns in geomagnetic activity. The DST index is measured in nanoteslas (NT). Quantifies the strength of the geomagnetic field disturbance, where negative values indicate geomagnetic storms while positive values suggest quiet conditions. The raw DST dataset presented several challenges that needed to be addressed before further analysis. For instance, DST encountered erroneous values like 99999.99, which distorted results. To clean the data merging of DATE and TIme columns into a single timestamp to ensure consistency and the removal of all these erroneous values. (see Figure 3)

Geomagnetic storms were classified based on DST index thresholds. Values below 50 nT indicate storm activity, while anything higher suggests calm conditions. The data was split into multiple categories (cal, storm forming, and ongoing storm), ultimately aiding model development. Through exploratory data analysis, we identified key patterns in the data, that the average DST index value is approximately -14.74 nT, indicating a predominance of negative disturbances within Earth's magnetosphere. The dataset exhibits significant negative skewness, revealing that mild disturbances are far more common than severe storms which are relatively rare. The most notable event was recorded on March 14, 1989,

when the DST index plummeted to a staggering -589 nT, corresponding to a powerful geomagnetic storm caused by a coronal mass (CME). Additionally, the introduction and exploration of sunspots, reflect the well-known solar cycle of approximately 11 years. Higher sunspot activity correlates with increased storm occurrences, suggesting that it does not directly influence the white severity of individuals' geomagnetic disturbances. Evident by the fluctuating Dst values independent of sins pot activity. Therefore to prepare for modelling we undertook a meticulous preprocessing and feature engineering process, which included creating a comprehensive data frame that consisted of crucial features such as timestamps, sunspot counts, and storm labels. These enhancements aim to improve model performance by placing greater emphasis on significant geomagnetic events.

In the modelling efforts to predict the DST index, we began with the basic architecture like LSTM, CNN, and GRU but faced the echo effect, where models forecasted the next DST value as a repeat of the last observation. This limitation prompted the investigation to compare the LSTM model, against a naive baseline, DST_Echo, with shifted DST values one hour ahead. The LSTM's RME ranged from 0.18 to 0.30, while DST_Echo ranged from 0.20 to 0.34, indicating no significant improvement in performance. To enhance predictions, we considered strategies like developing a classifier for storm phases and increasing the prediction window to focus on significant fluctuations. Despite time constraints preventing the full implementation of these strategies, we acknowledged the potential of advanced methods like XGBoost for improving accuracy in forecasting geomagnetic storms.

	DATE	TIME	DAY	DST
0	1/01/1957	00:00:00	1	11.00
1	1/01/1957	01:00:00	1	13.00
2	1/01/1957	02:00:00	1	12.00
3	1/01/1957	03:00:00	1	12.00
4	1/01/1957	04:00:00	1	9.00
...	...	...	...	...
593155	31/08/2024	19:00:00	244	99999.99
593156	31/08/2024	20:00:00	244	99999.99
593157	31/08/2024	21:00:00	244	99999.99
593158	31/08/2024	22:00:00	244	99999.99
593159	31/08/2024	23:00:00	244	99999.99

Figure 2 Original dataset

	DATE	TIME	DAY	DST	DATETIME
0	1/01/1957	00:00:00	1	11.0	1957-01-01 00:00:00
1	1/01/1957	01:00:00	1	13.0	1957-01-01 01:00:00
2	1/01/1957	02:00:00	1	12.0	1957-01-01 02:00:00
3	1/01/1957	03:00:00	1	12.0	1957-01-01 03:00:00
4	1/01/1957	04:00:00	1	9.0	1957-01-01 04:00:00
...	...	...	...	...	...
592578	7/08/2024	18:00:00	220	-1.0	2024-08-07 18:00:00
592579	7/08/2024	19:00:00	220	2.0	2024-08-07 19:00:00
592580	7/08/2024	20:00:00	220	1.0	2024-08-07 20:00:00
592581	7/08/2024	21:00:00	220	-4.0	2024-08-07 21:00:00
592582	7/08/2024	22:00:00	220	-12.0	2024-08-07 22:00:00

Figure 3 Cleaned dataset

Introduction

Powerful space weather phenomena called solar storms are brought on by the Sun's strong magnetic activity. On the surface of the Sun, sunspot areas with intense magnetic fields generate the precursors of these storms. When they experience extreme stress, they break, ejecting massive amounts of energy known as coronal mass ejections (CMEs). When these CMEs strike the planet, they interact with the magnetic field to produce geomagnetic storms. These CMEs launch charged particles into space, creating the solar wind. The Disturbance Storm Time (DST) index, a geomagnetic index/value that quantifies the severity of a geomagnetic storm measured in nanoteslas (nT), is how we gauge the power of these storms. Ultimately, If sunspots are active, more solar flares will result in a drop in DST, creating a decrease in geomagnetic storm activity for Earth. Because solar storms have the potential to seriously destroy electronics on Earth, interfere with power systems, and ruin satellites, which negatively impact our daily lives, it is crucial to anticipate geomagnetic storms. Precise forecasts can provide people with vital time to be ready for potential storms and reduce these hazards. For instance, the military has to be aware of solar flares to avoid interfering with radio communications, and the Victoria State Emergency Service (VICSES) and the Bureau of Meteorology collaborate to alert the public about storms. Additionally, governments must be ready for any internet disruptions and blackouts. In the end, forecasting powerful geomagnetic storms helps improve space weather forecasting and readiness, safeguarding international power, communication, and navigation infrastructure. The World Data Centre for Geomagnetism in Kyoto provided the data for our project, which took place over 12 weeks. From 1957 to 2024, form DST index records are included. In the end, the project seeks to forecast DST and predict storms using various machine learning models, comparing them to each other to determine which models were the most successful. It also seeks to assist future efforts in geomagnetic storm formation and understand the impact of solar winds on Earth's magnetic fields. An asset that needs to be considered is stakeholder interest, as these storms can damage electronics, disrupt power grids, and affect satellites, impacting daily life. Accurate predictions provide stakeholders with vital lead time to prepare for potential disruptions. For example, military operations must be aware of solar flares to prevent communication interruptions, while governments need to prepare for possible power outages and internet disruptions. Additionally, space agencies rely on storm forecasts to protect sensitive equipment in orbit, ensuring the safety of astronauts and preventing costly satellite damage. Tourists seeking to witness the northern or southern lights can also benefit from these forecasts, allowing them to plan their travels to experience these stunning natural phenomena. Ultimately effective geomagnetic storm forecasting enhances safety and supports critical infrastructure while enriching individual experiences with our planet's atmospheric wonders. This report will delve into our 12-week process, highlighting the data pre-processing, cleaning, explanation and modelling of the project.

Past Studies

Past studies have delved into various approaches for predicting geomagnetic storms, utilising a range of variables to enhance the accuracy and reliability of their models. One notable trend in these studies is the incorporation of multiple solar and interplanetary parameters, such as vector magnetic field, solar-wind velocity, proton temperature, proton density and interplanetary electric fields. Research indicated that these variables are critical for understanding the complex interaction that could lead to geomagnetic storms. For instance, studies have employed integrated gradients as a technique to interpret the outputs of convoluted neural network modes, as it allows researchers to pinpoint which variables are the most valuable and create the most impact when modelling for DST prediction. Additionally, past studies have highlighted the importance of utilised approaches, where models are trained and their outputs are combined to generate consensus prediction. Yet this strategy not only mitigates the risks associated with overfitting, it also enhances the robustness of predictions. Moreover, Satellite observations and ground-based measurements have proven beneficial in refining the inputs used for these predictive models, critical in allowing a more holistic view of the solar-terrestrial interaction and the factors contributing to geomagnetic storms. Ultimately the evolution of geomagnetic storm prediction has been marked by a shift towards more muti-variable models, incorporating things like AI and the aggregation go storm event data. These advancements have significantly enhanced the understanding of the processes leading to geomagnetic storms and have improved the predictive accuracy of models. Therefore these insights will be instrumental in developing more effective forecasting strategies. In the research, we intend to look into predictive capabilities of geomagnetic storm forecasting by utilising machine learning techie, providing stakeholders with more reliable tools for anticipating solar storm impacts. This is developed on the past research as it contributes to more informed decision-making across various sectors, like telecommunication, power management and emergency response.

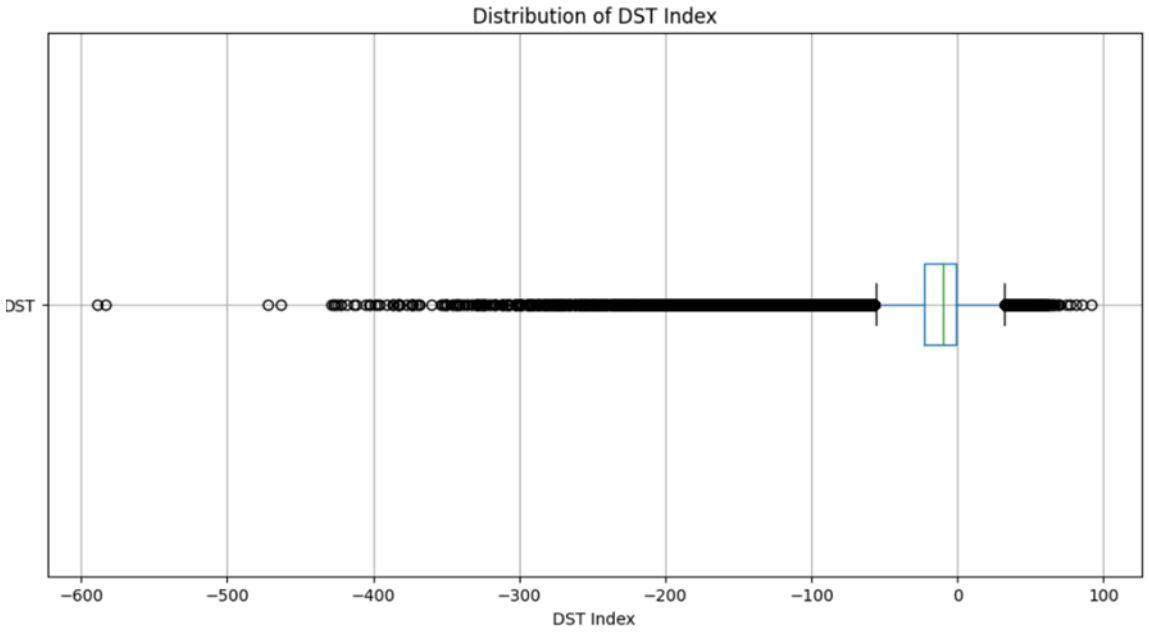
Data Quality and Cleaning

The data quality was fairly complete, with more recent timestamps not having data associated with them coming up as 999999.99 values. In the database read. me file, it claimed that the 9999.99 values were effectively Nan values, so we removed them. This made the entire dataset free from Nan values, and free from unexplainable outliers. This data is the same data used by big companies and is used in important research so the quality of the data is fairly reliable

Exploratory Data Analysis

The dst dataset contains **592,560** hourly measurements of the **Dst** (Disturbance Storm Time) index from **1957 to 2024**. To understand the storm indexes underlying patterns, we conducted exploratory data analysis with the hope of finding key features of the dataset and investigating variable significance before starting model training.

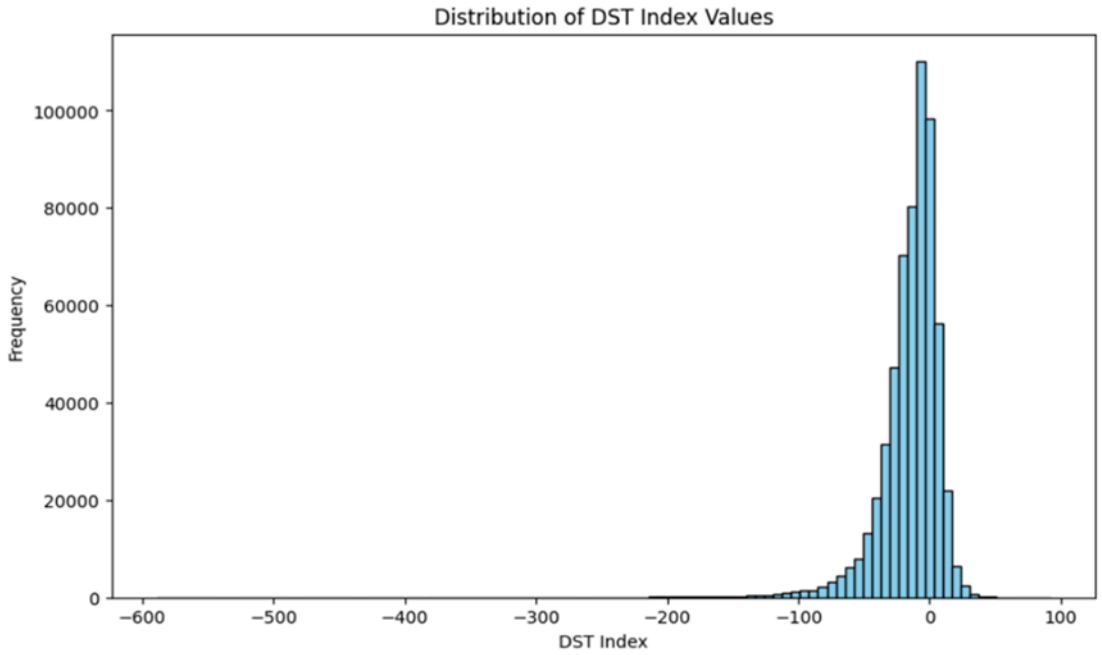
Statistical Summary of the Dst Index



- **Mean**: The average Dst index value is approximately **-14.74 nT**, indicating that Earth's magnetosphere typically experiences slight negative disturbances.
- **Standard Deviation**: A value of **23.15 nT** reflects considerable variability in geomagnetic activity, encompassing both mild and extreme disturbances.
- **Range**: The Dst index ranges from a **minimum of -589 nT** (representing a severe geomagnetic storm) to a **maximum of 92 nT** (corresponding to calm or positive geomagnetic conditions).
- **Percentiles**:
    - **25th Percentile (Q1)**: **-23 nT**—25% of the data indicates moderate geomagnetic disturbances. With **83.55%** of the data above **-25nT**
    - **Median (50th Percentile)**: **-10 nT**—half of the observations are mild to moderate disturbances.
    - **75th Percentile (Q3)**: **-1 nT**—75% of the values are below this level, highlighting that negative values dominate the dataset.

These statistics reveal that the dataset is skewed towards negative values, indicating frequent geomagnetic disturbances rather than calm conditions.

Distribution of Dst Index Values

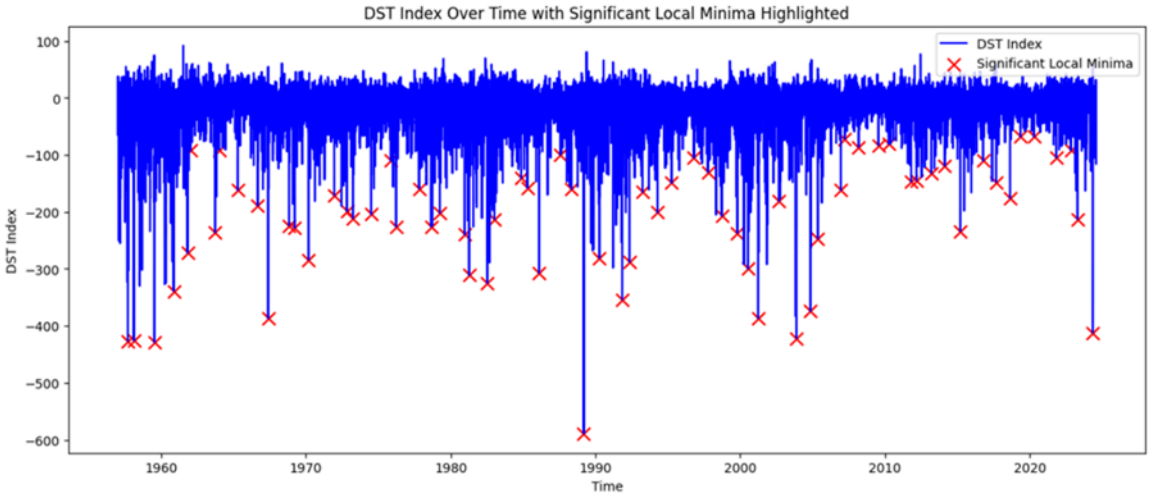


- **Negatively Skewed**: The data is **heavily skewed** to the left (negative skew), with most values clustered around close to 0 negative numbers.
- **Concentration of Values**: The highest concentration **73%** lies between **-50 nT and 0 nT**, suggesting that mild geomagnetic disturbances are the most common.
- **Extreme Values**: The long left tail extends to almost **-600 nT**, representing severe but less frequent geomagnetic storms. With only **5.75%** of the data being below **-50 nT** (Which is considered high storm activity)
- **Positive Values**: A small number of positive DST values appear on the right, indicating that periods of positive DST or exceptionally calm conditions are relatively rare.

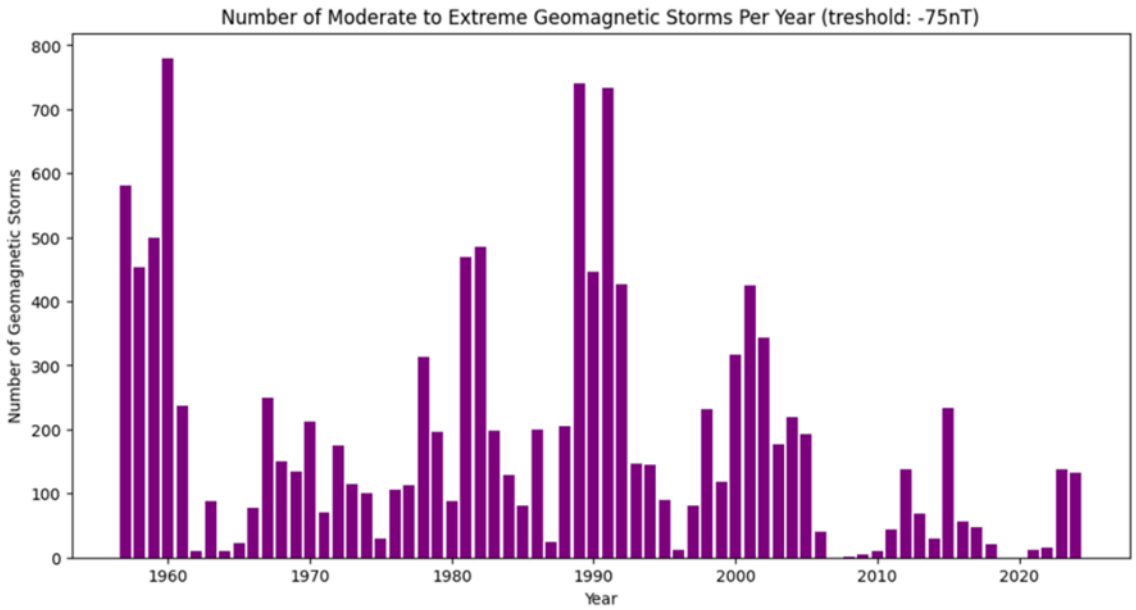
This distribution underscores the prevalence of mild disturbances and the rarity of major geomagnetic events in our dataset.

Temporal Trends in the Dst Index

The time series analysis of the Dst index over the years reveals significant patterns and events:



- **Frequent Fluctuations**: Regular variations in the Dst index indicate ongoing geomagnetic activity.
- **Significant Dips**: Notable drops correspond to periods of geomagnetic storms, with local minima highlighting the most severe events each year. Highlighted with a red X
- **Historical Event—March 14, 1989**:
    - The most significant drop in the Dst index reached **-589 nT**.
    - This event corresponds to a well-documented geomagnetic storm caused by a coronal mass ejection (CME).
    - Impacts included a major power blackout in Quebec, Canada, affecting **six million people** for about **nine hours**.
    - This storm highlighted the vulnerability of technological infrastructure to geomagnetic activity.
- **Clusters of Activity**:



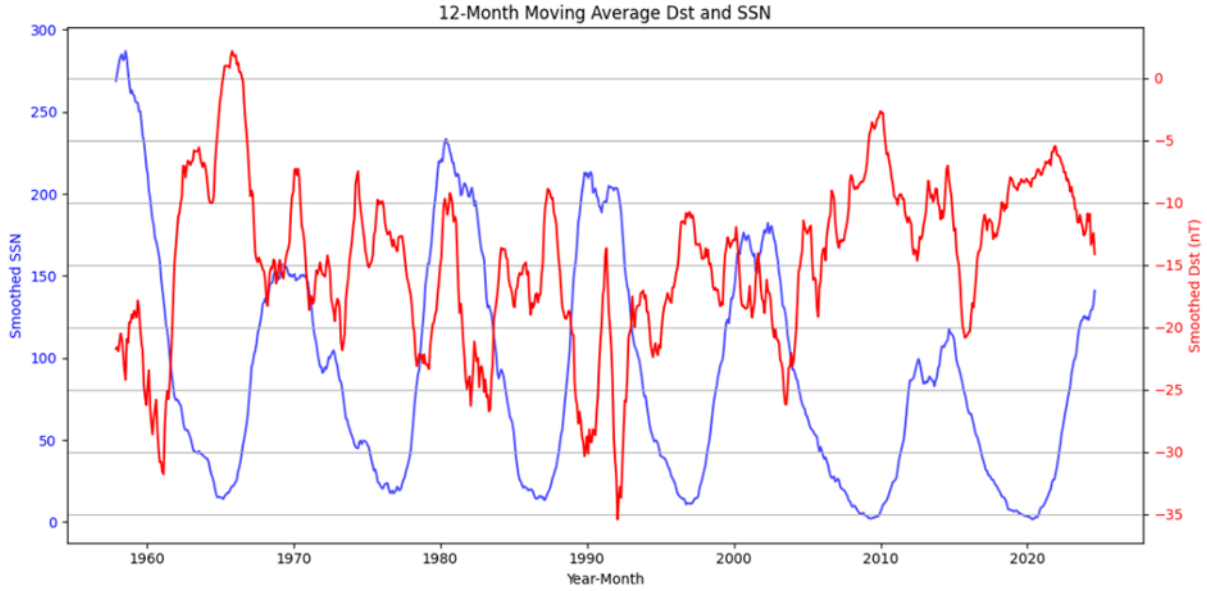
    - **Late 1950s, Early 1980s, and 1990s**: Periods with higher concentrations of severe storms.

- **Recent Years**: A noticeable decline in the frequency and severity of geomagnetic storms. Visible in the bar chart below

Relationship Between Sunspot Counts and Dst Index

To explore potential correlations between solar activity and geomagnetic disturbances, we incorporated monthly sunspot counts into our analysis:

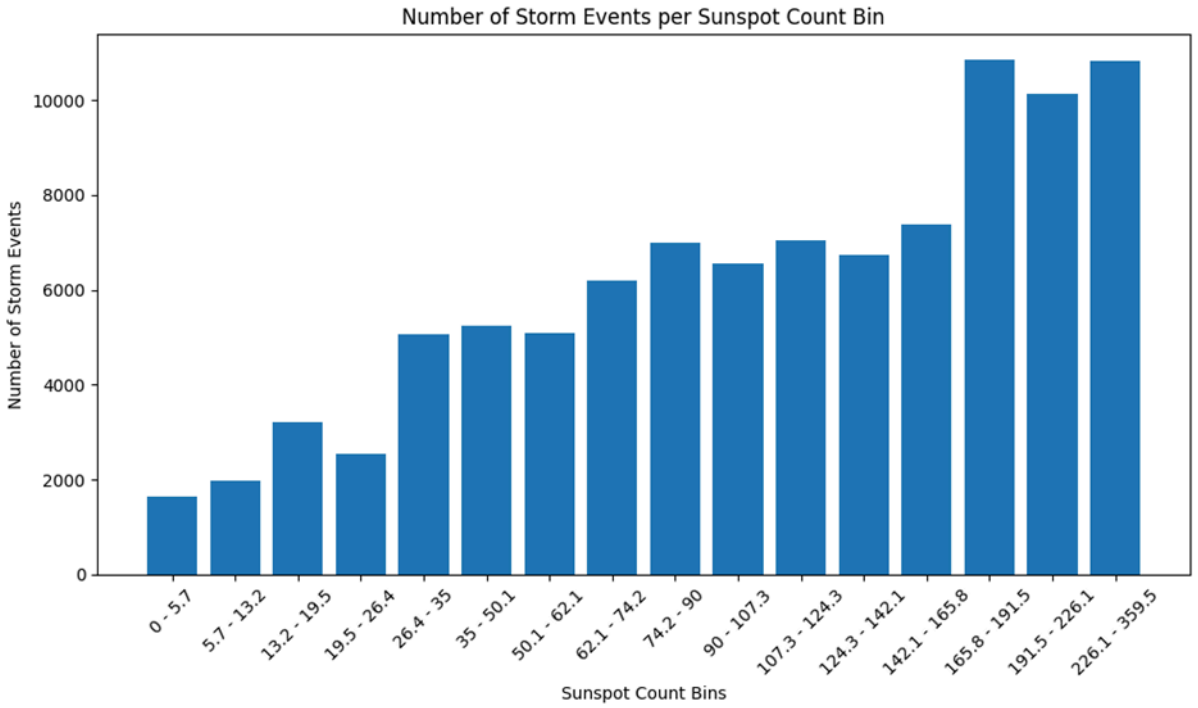
- **Sunspot Data Integration**:
  - Sunspots are indicators of **solar cycle** phases, affecting solar wind and CMEs.
  - We merged sunspot counts with the Dst index data for a comprehensive view.
- **Cyclic Patterns**:
  - Both sunspot counts and the **frequency** of geomagnetic storms exhibit cyclic behaviour, reflecting the approximate **11-year solar cycle**.
  - **Figure 4** illustrates the cyclic nature of both sunspots and Dst index values. **Note: High** sunspot count periods seem to align with **more severe lower** DST values and **lower** periods of sunspot counts align with **higher**, **less severe** values of DST as expected by the theoretical effect of sunspot counts.



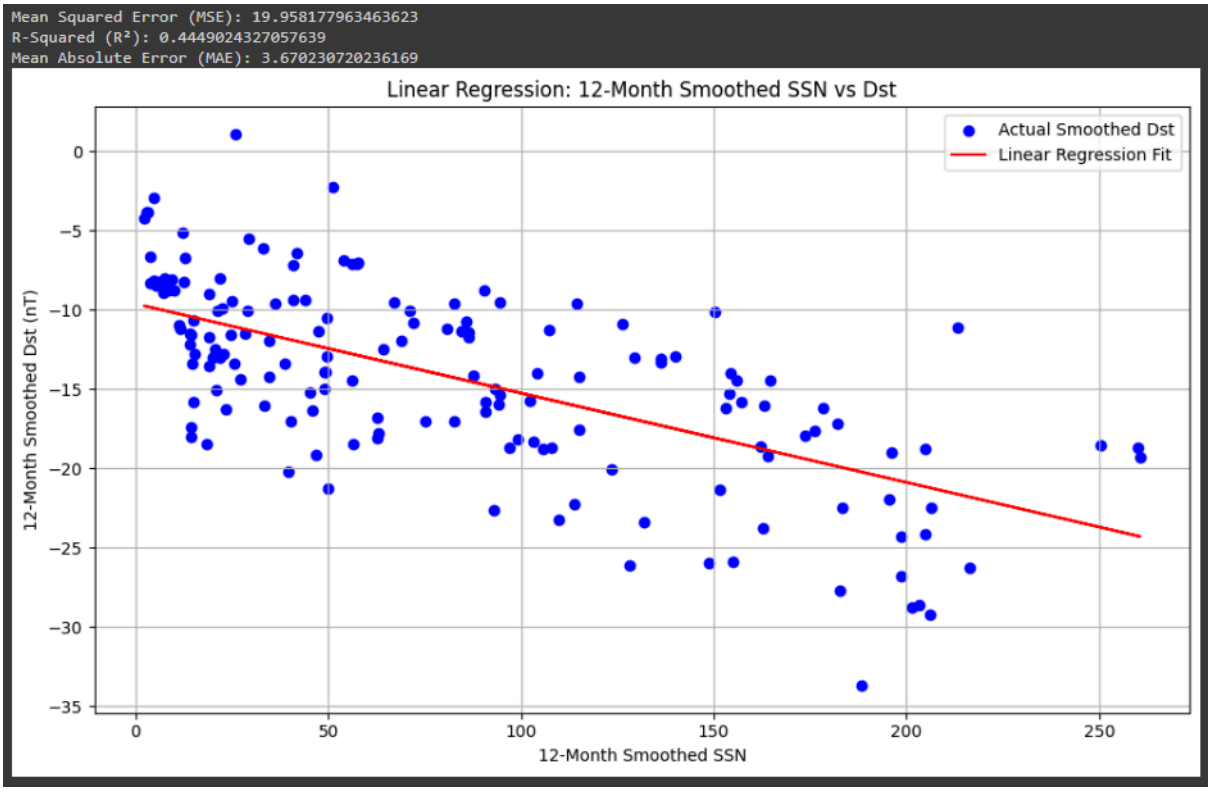
Correlation Findings:

- **Storm Frequency**:
  - During periods of high sunspot counts, storm frequency increased up to **fivefold**.
  - **Figure 5** shows the clear correlation between increased sunspot activity and frequency of storm occurrences. **Note:** Sunspot count was

placed into 16 **equally distributed** bins. And since sunspot count stays with very similar values over a single year, each sunspot bin should encapsulate a similar number of years.



- **Dst Index Values**:
  - Despite the increased frequency, there is **no clear evidence** that the magnitude of the Dst index is directly influenced by sunspot counts.
  - Dst values fluctuate independently of sunspot activity, as shown in **Figure 6**. **Note:** While the seems to be a negative correlation, the relatively high Mean Squared Error Indicates a **low correlation**.



These findings suggest that while solar activity influences the frequency of geomagnetic storms, it does not necessarily allow for the prediction of direct DST values.

Conclusion of EDA

Our exploratory data analysis highlights:

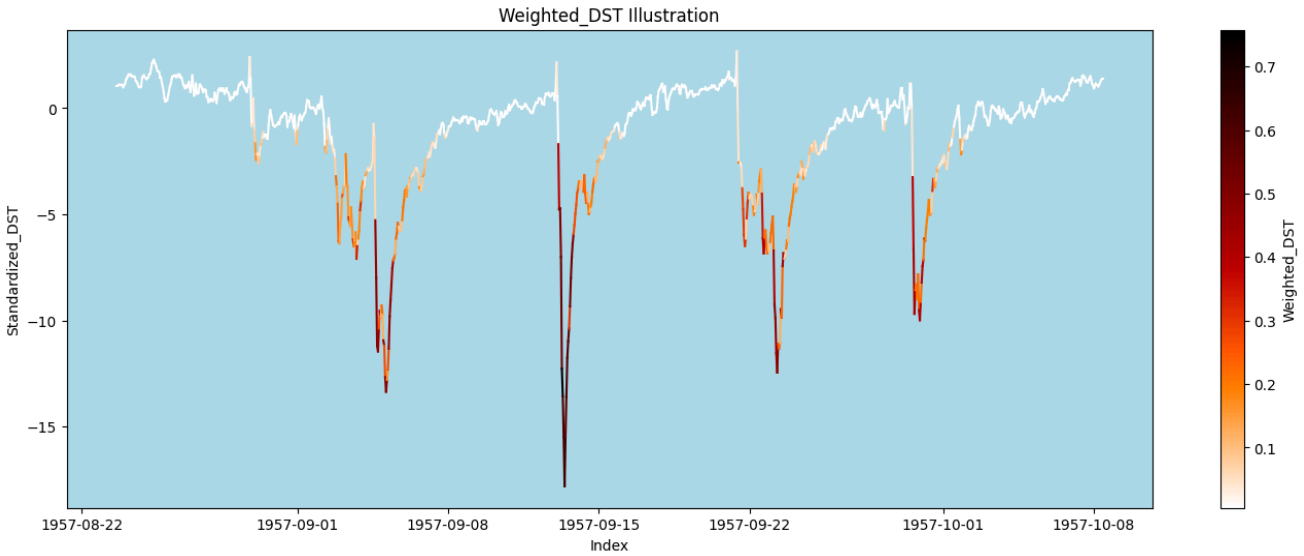
- The dataset is dominated by negative Dst index values, indicating frequent mild to moderate geomagnetic disturbances.
- Severe geomagnetic storms are rare but have significant impacts on Earth's magnetosphere and technological infrastructure.
- Temporal trends reveal periods of heightened geomagnetic activity, aligning with known solar cycles.
- While increased sunspot activity correlates with a higher frequency of storms, it does not directly influence the severity of individual geomagnetic disturbances.

Data Preprocessing and Feature Engineering

- **Creating the Data Frame**

To prepare our data for modelling, we constructed a comprehensive data frame with the following features:

- **Date**: Timestamp for each hourly observation. Useful for merging predicted datasets into larger databases by keeping a consistent **primary key**
- **Dst Index**: Hourly measurements of geomagnetic activity. The key **target variable** used in all our models
- **Sunspot Count**: Monthly values binned into 16 categories between 0 and 1 to capture **seasonal solar cycle information**. Theoretically useful for slightly boosting the prediction for the likelihood of the formation of a storm
- **Climax History**: During an ongoing storm (storm label = 2), this feature records the **highest Dst value** since the storm started. This aids in understanding storm intensity over time and ensures the model is aware of the contextual information behind the storm.
- **Gradient Index**: The difference between the latest observed point and the average of the **last three points**, indicating **sudden changes** in geomagnetic activity, included to capture the dynamics of storm development and recovery phases.
- **Storm Label**: Categorical variable indicating storm status:
  - **0**: No storm
  - **1**: Storm forming
  - **2**: Storm ongoing
  - *Note*: Not used for training but useful for evaluation and analysis.
- **Weighted Dst**: A custom variable to **emphasise important events** during training, giving higher weights to severe storms, proximity to storms, sudden changes in DST and Gradient indexes, active storm phases, lower DST values, and periods with high sunspot counts. Helps the model focus on significant geomagnetic events, potentially **improving prediction accuracy during critical periods**. Useful for getting the model to target more important rarer parts of the dataset, and to place **higher importance on storms** without worrying about over or under-sampling.



Model Development and Evaluation

CNN

CNN Model Description:

Convolutional Neural Networks (CNN) are deep learning algorithms that are typically used for processing image data. CNNs are set apart from other neural networks by their implementation of a convolutional layer, this layer applies filters to the inputs detecting patterns, which in the context of an image, generally include edges, textures, shape and colour. This neural network also traditionally applies a pooling layer after a convolutional layer, to reduce the dimensions of the data, simplifying the computational processes whilst preserving the relevance of important features.

In the context of this project, the primary data that would have to be fed into the CNN would be a time series, a 1-dimensional linear sequence, unlike the typical 2D input for a CNN. Where an image would be typically subdivided into smaller composite images, the DST times series for this project would have to be split into smaller sequences denoting a certain window of time in hours.

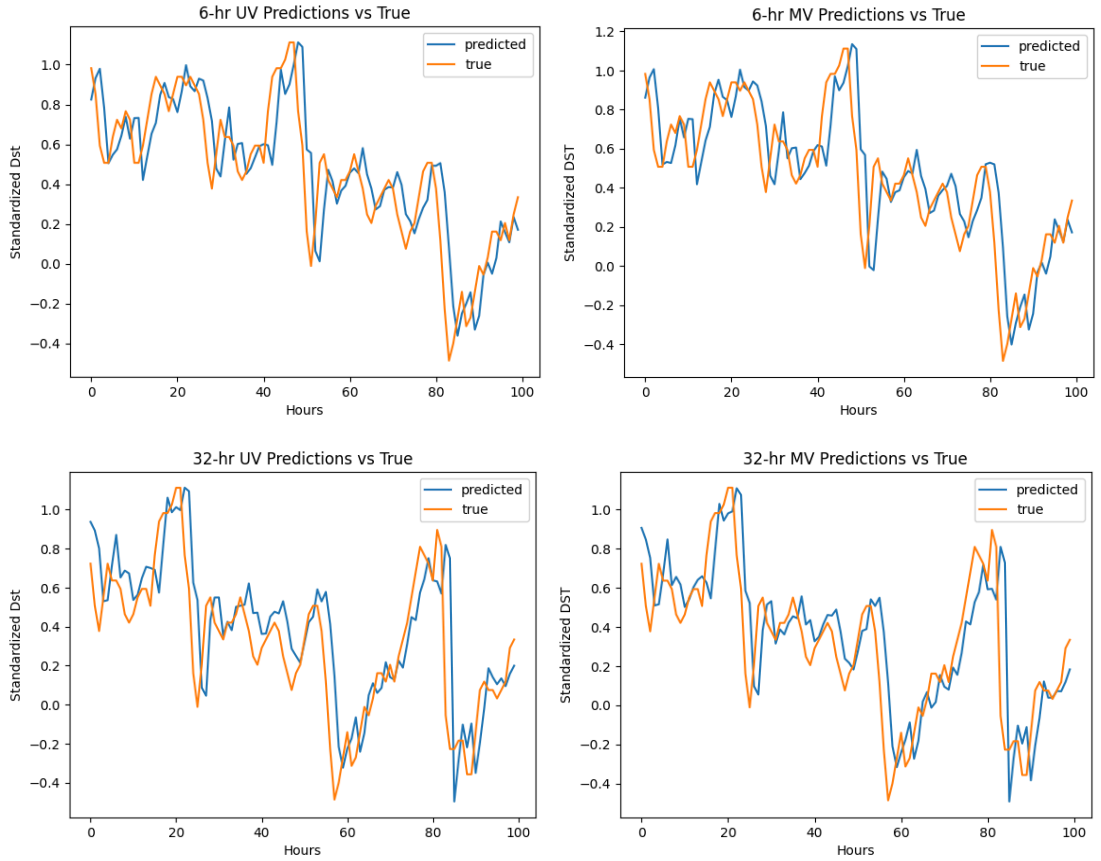
CNN Construction:

- **Preparing the Data:**
    - **Univariate:** 'Standardised_DST' was split to obtain 2 arrays from the time series, an array of sequences of DST indexes that are n hours long (the x value), and an array of the corresponding target DST value at n+1 (y value). The corresponding arrays of arrays were split into train, validation and test sets, using a 70:10:20 split. The arrays of size n were then reshaped to fit the standard input requirements of a CNN, where x.shape = (total number of sub-arrays, n, number of features).
    - **Multivariate:** Similar to the univariate process, a similar algorithm was used to split 'Standardized_DST' into arrays of x and y variables, however, this time the 'sunspot_bin' variable was also split, with only the array of x values being used. Following the same training, testing and validation split as in the univariate process, and both x variables were reshaped. Given that both x variables were of the same shape we were able to concatenate the two into a single input array for the model to use.
- **Defining the Model:**
    - **Conv1D:** The convolutional layer, required defining what shape the individual data points took, including how many features were contained in each.

    - **MaxPooling1D:** The pooling layer, with a pool size of 2, extracting the most important or maximal features from the 2 samples.

- **Flatten:** Converts the shape of the data into a 2-dimensional vector, to make the transition from feature extraction to regression or classification through Dense more simple.
- **Dense:** Applied twice, first to 50 neurons, then finally to 1 neuron to extract the target predicted

CNN Evaluation:



As seen across the 4 plots above, despite exploring a varying hourly window size as well as univariate and multivariate models we can see the performance of these 4 models are all extremely similar in predicting the n+1th hour DST index.

- **6-hr Univariate:**
  - **RMSE:** 0.2234
  - **Total Parameters:** 20,081 parameters
- **32-hr Univariate:**
  - **RMSE:** 0.2220
  - **Total Parameters:** 144,881 parameters
- **6-hr Multivariate:**
  - **RMSE: 0.2270**
  - **Total Parameters:** 20,081 parameters
- **32-hr Multivariate:**

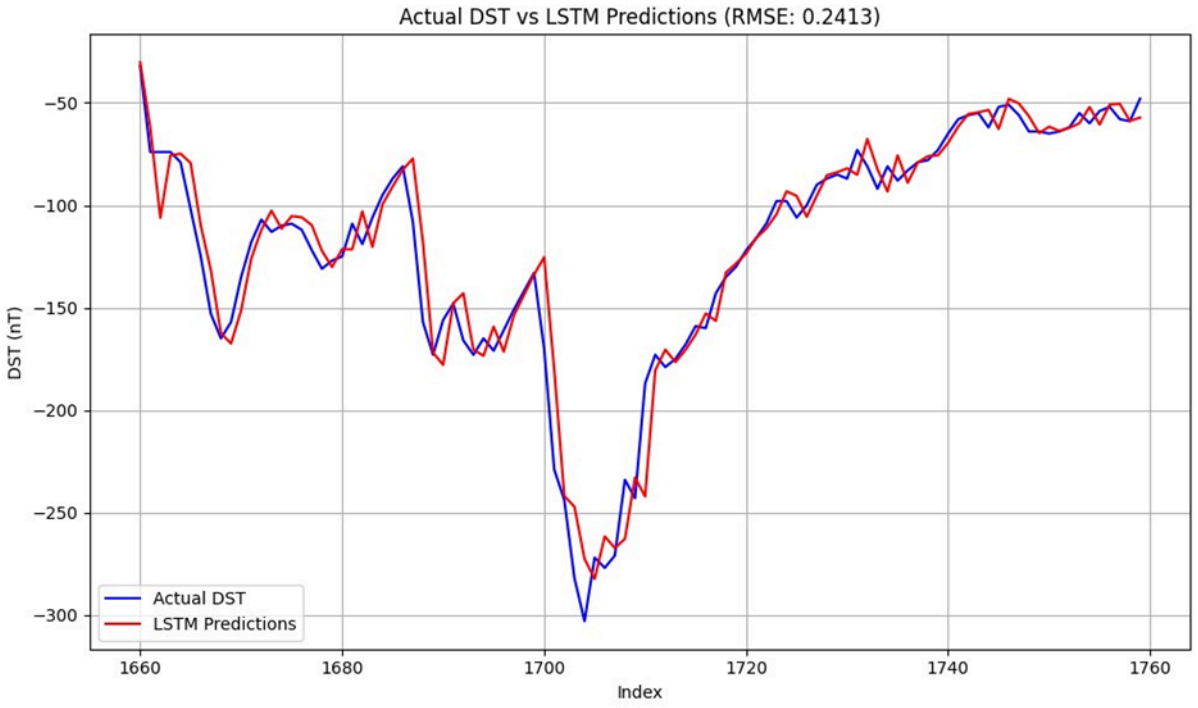
- **RMSE: 0.2216**
- **Total Parameters:** 144,881 parameters

Both 32 hour models seem to have a slightly better RMSE score than their 6 hour counterparts, however both have a total number of parameters that are more than 7 times that of their counterparts. Being both computationally expensive and indicating a high level of bias, both 32 hour models should not be considered useful for predicting DST. Considering RMSE values this means that the best performing CNN model of the 4 presented here is the 6 hour univariate model. However, we must recognise that plotting the predictions against the actuals shows that there is a slight latency which make the model look like it is echoing the last data point common across all models.

LSTM

LSTM Model Description:

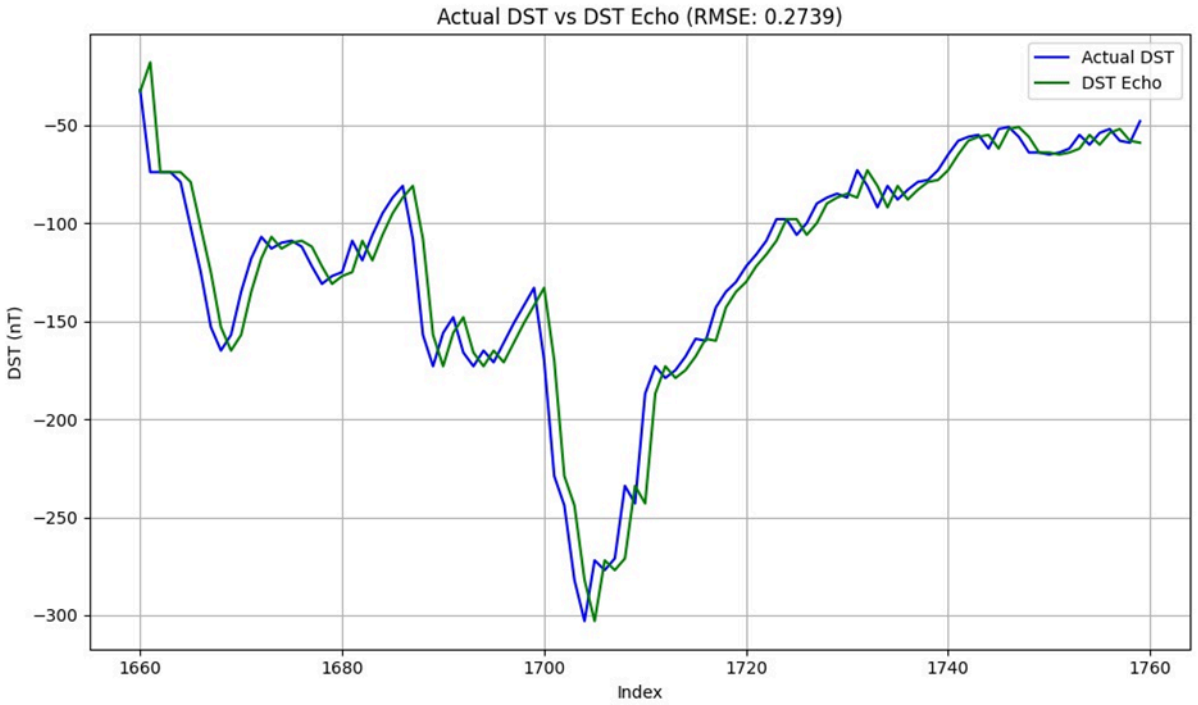
Long Short-Term Memory (LSTM) is a type of recurrent neural network(RNN), that is traditionally used to process time-series and sequential data, recognising long-term dependencies within it. Unlike other RNNs, the LSTM can handle long-term dependencies by using references to past information to define the current states of the latest inputs, making this neural network a better alternative when building a model on a larger sequence.



- **Echo Effect**: The models tended to predict the next Dst value as the same as the last value in the input sequence, effectively creating a lag or "echo" rather than a true forecast.

- **Evaluation**:
  - We compared the LSTM model's performance to a naive baseline model, **Dst_Echo**, which simply shifted the Dst values by one hour ahead. Likely caused by the model being **overly rewarded** for predicting **unchanging data** that it doesn't place more resources into figuring out larger patterns in the data and simply predicts that the data will remain as it has.
  - The LSTM model's RMSE had a 95% confidence interval of **0.18 to 0.30**, while the Dst_Echo model's RMSE was **0.20 to 0.34**.
  - The overlapping intervals indicated that the LSTM did not significantly outperform the naive model.

[ECHO "MODEL"]



- **Insights and Adjustments**
  - Recognizing that our models were not capturing the underlying patterns effectively, we considered additional strategies:
    - **Classifier Model**: Developing a classifier to predict storm phases, which could inform the forecasting model when a storm is likely to occur, to focus the model. When developed, this would act as our warning system. Telling the main model to be aware and get ready to change "tactics" by **warning** that the next 10 20 or 50 points are going to be a storm. This model would aim to simply get a gauge for when conditions are right for a storm and what type of storm it's going to be.
    - **Increasing Prediction Window**: By increasing the prediction window, the model is forced to attempt to make predictions about the overall direction of the data. This could lead to a new problem where instead of predicting the next few values to be the ones previously observed, it

could simply use the tangent of the gradient of the data at the last sequence of data points observed (Unfortunately we did not have time to verify our final model did not simply do this but would have been very useful in our final model evaluation)
        - **Less weight toward unchanging data**: With most of our data consisting of relatively unchanging, similar, calm data, making the model more sensitive to dramatically changing data (Higher Gradient Index) could cause it to lose “points” for predicting that the DST index will do exactly what it’s done in the past.
    - **Time Constraints**: Due to time limitations, we focused on establishing a working model before fully integrating these enhancements.

Challenges and Solutions

Throughout our project, we faced multiple modelling challenges that required us to come up with various proposed solutions to improve our geomagnetic storm forecasting models. About 67 different models were tested or simulated (by running limited epochs to save time) with many different basic parameters:

```
BATCH_SIZE = 64
LOOKBACK_WINDOW = 32
PREDICTION_WINDOW = 6
EPOCHS = 3
LEARNING_RATE = 0.01
```

Parameters experimented with such as learning rate, batch size (from 8, 16, 32, 64 and 128, all with similar returns), patience going from 2 up to 8, lookback window being experimented with from 12 to 64, and different diminishing learning rate specs. Many other variables were played with to the point where it would be impractical to go through all of them in this report especially changes and updates to weighted DST optimisation, one example is:

```
def adjust_weighted_dst(df):
        # Iterate over each storm group
    # Label consecutive storm periods
    storm_periods = (df['Standardized_DST'] < -1).astype(int)
    storm_labels = (storm_periods.diff(1) != 0).cumsum() * storm_periods
    storm_labels = storm_labels.fillna(0)

    for storm_label in storm_labels.unique():
        if storm_label == 0:  # Skip non-storm periods
            continue

        # Filter to the current storm group
        storm_data = df[storm_labels == storm_label]

        # Find the minimum Standardized_DST value within the storm
        min_storm_dst = storm_data['Standardized_DST'].min()

        # Calculate the multiplier based on the min storm DST
        multiplier = 1 + 3*abs(min_storm_dst / 40)

        # Apply this multiplier to the Weighted_DST values in the storm
        df.loc[storm_labels == storm_label, 'Weighted_DST'] *= multiplier

        # Find the index of the minimum Standardized_DST value (i.e., the peak)
        peak_index = storm_data['Standardized_DST'].idxmin()
        peak_value = storm_data.loc[peak_index, 'Standardized_DST']

        # Iterate over the storm data to apply multiplier based on distance from the peak
        for idx, row in storm_data.iterrows():
            hours_from_peak = abs((idx - peak_index).total_seconds() / 3600)
            if hours_from_peak <= abs(peak_value * 4):
                additional_multiplier = 2.5 - (hours_from_peak / abs(peak_value * 4))
                df.at[idx, 'Weighted_DST'] *= additional_multiplier


    # Apply transformations based on Standardized_DST values
    df.loc[df['Standardized_DST'] < -3, 'Weighted_DST'] *= 2
    df.loc[(df['Standardized_DST'] < -1.5) & (df['Standardized_DST'] >= -3), 'Weighted_DST'] *= 1.5
    df.loc[(df['Standardized_DST'] < -0.95) & (df['Standardized_DST'] >= -1.5), 'Weighted_DST'] *= 1.25

    # Apply transformations based on gradient index
    df.loc[df['gradient_index'] > 0.75, 'Weighted_DST'] *= 3
    df.loc[(df['gradient_index'] > 0.5) & (df['gradient_index'] <= 0.75), 'Weighted_DST'] *= 2.5
    df.loc[(df['gradient_index'] > 0.3) & (df['gradient_index'] <= 0.5), 'Weighted_DST'] *= 1.5

    # Apply transformations based on Weighted_DST values
    df.loc[(df['Weighted_DST'] >= 0.4) & (df['Weighted_DST'] <= 0.5), 'Weighted_DST'] += 0.1
    df.loc[df['Weighted_DST'] > 0.5, 'Weighted_DST'] *= 1.25

    # Apply multiplication based on climax_history value
    df.loc[(df['Weighted_DST'] < 0.5) & (df['climax_history'] != 0), 'Weighted_DST'] *= 2

    # Apply multiplication based on climax_history, Standardized_DST, and gradient_index values
    df.loc[(df['Standardized_DST'] > -0.5) & (df['gradient_index'] < 0.2), 'Weighted_DST'] *= 0.4

    df.loc[df['Weighted_DST'] > 1, 'Weighted_DST'] *= max(df['Weighted_DST'])

    # Normalise
    df['Weighted_DST'] = (df['Weighted_DST'] - df['Weighted_DST'].min()) / (df['Weighted_DST'].max() -
df['Weighted_DST'].min())

    # Apply transformations based on Weighted_DST values
    #df.loc[df['Weighted_DST'] > 0.6, 'Weighted_DST'] *= 1.05
    #df.loc[(df['Weighted_DST'] >= 0.5) & (df['Weighted_DST'] <= 0.6), 'Weighted_DST'] *= 1.2
    #df.loc[(df['Weighted_DST'] >= 0.2) & (df['Weighted_DST'] <= 0.5), 'Weighted_DST'] *= 1.35
    # Cap Weighted_DST values at 0.98
    df.loc[df['Weighted_DST'] > 2, 'Weighted_DST'] = 2
    # Spread values between 0.00001 and 0.1 to be between 0.01 and 0.1
    mask = (df['Weighted_DST'] >= 0.00001) & (df['Weighted_DST'] <= 0.1)
    df.loc[mask, 'Weighted_DST'] = 0.03 + (df.loc[mask, 'Weighted_DST'] - df['Weighted_DST'].min()) * (0.1 -
0.03) / (0.1 - 0.00001)
    return df
```

1. **Poor Model usefulness and predictability**

    - **Challenge:** Our single-hour models were only reactive, responding to data and not attempting to pre-empt the decrease of DST.

    - **Solution:** By using a **6-hour predictive window**, we hoped that the model would focus on more overlying trends within the DST. Unfortunately, even with this expanded prediction window, models still seemed to be slightly reactive and possibly overfitted. The models also, expectedly, had a significant decrease in MSE score, as the model was making **larger mistakes more frequently** as it's got more unseen data it's trying to predict towards. To best optimise this, if we had more time, we would experiment with a larger and longer range of predictive windows to see where we get the **best MSE loss** for **predictive capacity**.

2. **Short Training Epochs and Early Convergence**

    - **Challenge:** Our larger models were converging too quickly, often after only 10 epochs, without achieving satisfactory validation losses. This early convergence suggested that the models were not learning as effectively from the data as they could.

    - **Solution:** We increased the model's **patience** parameter in the early stopping function, allowing the training process to continue for more epochs before stopping hopefully **forcibly breaking past plateaus**. Additionally, we tried both a **decreased learning rate**, and a **slowly diminishing learning rate,** enabling the optimizer to make finer adjustments to the model weights. We even experimented with increasing and decreasing our lookback window during model development. We ended up deciding on a **36-hour lookback window** as it seemed to best capture the majority of important storms within them (unless it was a freak storm but we wanted to minimise GPU usage and speed up training times). We also changed the epoch selection loss function from validation loss to weighted validation loss, one that takes the best epoch that supported the weighting system we created rather than the general MSE loss. These changes lead to small decreases in validation losses, however with even small validation losses, for our big dataset, is extremely valuable.

3. **Overfitting Due to High Parameter Count**

    - **Challenge:** The initial models had a high number of parameters relative to the dataset size (e.g., 344,961 parameters for 508,415 data points), which increased the risk of overfitting. Overfitting occurs when a model learns the training data too well, including its noise and outliers, or when the main loss decreases much faster each epoch than the validation loss, thus performing poorly on new, unseen data.

    - **Solution:** We redesigned the model architecture by experimenting with stacking smaller LSTM layers instead of using a single large LSTM layer or even using a smaller single layer but decreasing and diminishing the learning rate to allow for longer training on smaller parameters. This approach reduced the total number of parameters while still capturing both short-term and long-term dependencies in the data. We found that the stacked model only slightly outperformed the smaller single-layer model, and we wished to confirm that stacking was better (as supported by our previous research {see (Rabby et al.) and (Van Houdt et al.)}) however due to time constraints were unable to appropriately experiment to confirm which was better.

4. **Ineffective Weighting for Important Data Points**

    - **Challenge:** When we implemented our weighted dst function to help train the model better, we did not initially see significant improvements in severe storm predictions. The models did not perform noticeably better on higher-weighted Dst values, which are crucial for accurate storm predictions.
    - **Solution:** We adjusted the weighting scheme to be more extreme, increasing the disparity between the weights in the lower end and the higher end, we also introduced more factors into the weighted DST function (all discussed above) to further polarise the data. By amplifying the weights for the most critical data points, we forced the models to further prioritize learning from these rare but important events. This approach improved the models' performance on higher-weighted Dst values, enhancing their ability to predict severe geomagnetic storms.

5. **Technical Limitations and Computational Constraints**

    - **Challenge:** We faced significant resource limitations, such as Google Colab sessions timing out and the STUDEL MASSIVE computing environment lacking the latest version of Keras needed for optimization and preventing GPU overload. This problem was faced by every other group we discussed this problem with, from both catheter group members and stock group members.

    - **Solution:** We implemented a checkpointing system by saving the model weights after each training epoch to a shared drive. This allowed us to resume training from the last saved state in case of unexpected shutdowns and even test model performance whilst it was still training. By continuously uploading the latest model weights, we ensured that our progress was preserved, and we could effectively utilize the available computational resources thus helping us sleep at night not worried about waking up to a failed model and a dead kernel.

Modelling Results

Despite the above challenges, we developed several models to forecast the Dst index 6 hours ahead, evaluating their performance across different weighted Dst ranges, particularly focusing on storm periods. Due to coding issues and syntax errors, we were unable to evaluate our models using the storm phase as we planned, however, because we know that the weighted dst values go up for stormier more important data points we were able to evaluate our models based on how well they performed with different weight ranges. Despite the basic lstm model having an overall lower test loss, it performed much worse when considering more important data points as it focused on the majority data rather than the minority that is important to us. We wish we could have used our earlier metrics to measure specific storm periods but this is all we have. Due to time constraints (each model took longer than 6 hours and often did not work in the end anyway) the time we had to experiment with different features and different modelling systems was severely limited. Only the best parameters were used for each model after many different parameters were tested.

1. **Echo Model**
    - **Description:** A fake, algorithmic model that simply predicts the next 6 Dst values as the same as the last observed values.
    - **Performance:**
        - **Overall MSE:** 0.0878
        - **RMSE by Weight Range:**
            - Range 0.00 - 1.00: RMSE = 0.2962
            - Range 1.00 - 2.50: RMSE = 0.3807
            - Range 2.50 - 5.00: RMSE = 0.4537
            - Range 5.00 - 7.50: RMSE = 0.8784
            - Range 7.50 - 10.00: RMSE = 1.5962
            - Range 10.00 - 1000.00: RMSE = 2.9248
    - **Analysis:** The Echo model performs reasonably well during calm periods but poorly during storm events, as it was a purely reactive model rather than proactive.

2. **Basic LSTM Model**

BATCH_SIZE = 8

LOOKBACK_WINDOW = 32

PREDICTION_WINDOW = 6

EPOCHS = 150

LEARNING_RATE = 0.0064

    - **Description:** An LSTM model without any advanced features or weighting schemes.

- **Test Loss:** 0.1327
- **Performance:**
  - **Range 0.00 - 1.00:** MSE = 0.0592, RMSE = 0.2433
  - **Range 10.00 - 1000.00:** MSE = 2.3273, RMSE = 1.5256
- **Analysis:** The Basic LSTM outperforms the Echo model overall but still underperforms during storm periods due to the dominance of calm data influencing the model's training.

3. **Advanced Weighted LSTM Model**

BATCH_SIZE = 8

LOOKBACK_WINDOW = 32

PREDICTION_WINDOW = 6

EPOCHS = 150

LEARNING_RATE = 0.0064

- **Description:** An LSTM model incorporating a weighted loss function to emphasise storm periods.
- **Test Loss:** 0.1237
- **Performance:**
  - **Range 0.00 - 1.00:** MSE = 0.0637, RMSE = 0.2524
  - **Range 10.00 - 1000.00:** MSE = 1.7526, RMSE = 1.3239
- **Analysis:** While the overall test loss is slightly higher than the Basic LSTM, the Advanced Weighted LSTM performs better on storm data, demonstrating the effectiveness of the weighting scheme.

4. **Weighted LSTM with Diminishing Learning Rate**

BATCH_SIZE = 8

LOOKBACK_WINDOW = 32

PREDICTION_WINDOW = 6

EPOCHS = 150

LEARNING_RATE = 0.0064

Diminish_LR = 0.95

    - **Description:** Similar to the Advanced Weighted LSTM but with a learning rate that diminishes over time.
    - **Test Loss:** 0.1237
    - **Performance:**
        - **Range 0.00 - 1.00:** MSE = 0.0604, RMSE = 0.2457
        - **Range 10.00 - 1000.00:** MSE = 1.8544, RMSE = 1.3618
    - **Analysis:** The diminishing learning rate helped stabilize training but did not outperform the Advanced Weighted LSTM.

Key Observations:

- **Trade-off Between Calm and Storm Periods:** Models focusing on reducing overall loss tend to perform better during calm periods but worse during storms. Conversely, models emphasizing storm data improve predictions during storms at the expense of slightly higher errors during calm periods. This is expected and not concerning, as predicting loss in calm periods is not very useful in the real world, however predicting loss in stormy periods is.

- **Importance of Weighting:** Implementing a weighted loss function was crucial in enhancing the model's ability to predict rare but significant storm events.

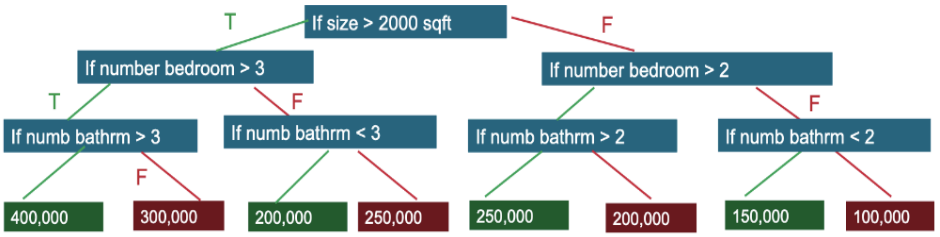
- **Limitations Due to Resource Constraints:** The inability to implement storm phase-specific evaluation metrics limited our analysis. Computational constraints also restricted the extent of experimentation with different model architectures and hyperparameters.

    -

XG BOOST:

Introduction:

XGBoost (eXtreme Gradient Boosting) is a high-performance machine learning algorithm designed for both classification and regression tasks. It is an implementation of gradient-boosted decision trees, a representation of an xgboost decision tree can be seen below along with the math that it takes to make gradient boosting possible which is beyond the scope of this report due to its complex nature but can be quite helpful to understand XGBoost at a basic level.



p_i : *Predictd value*

O_v : *Output value from tree*

λ : *Regualrization parameter*

$$Ov = \frac{Sum\ of\ residuals}{No.\ of\ residual + \lambda}$$

Reasons for using XGBoost for DST Prediction:

1. XGBoost's properties include the ability to learn from historical patterns and apply decision tree-based rules and thus while it's an unconventional candidate for forecasting it can be quite useful for DST index values since it can classify both numeric and categorical data values in decision trees.
2. Gradient Boosting Strength: XGBoost iteratively improves its predictions by focusing on errors made by previous models. This results in a powerful ensemble of models that can capture subtle patterns in the DST data.
3. Efficiency and Scalability: Another quality of XGBoost is that it is optimised for both memory and computation efficiency, which is crucial for our dataset as we have seen in the EDA that it is quite large and expansive due to several predictors aligned with DST values.

The preprocessing stage as we will see below, focused on transforming Disturbance Storm Time (DST) data from a single input variable to multivariable input into the XGBoost machine learning model to improve the predictions by increasing the number of predictors. The goal of the preprocessing stage was to leverage the XGBoost model's ability to predict geomagnetic storms, particularly weak to moderate storms where the behaviour is more predictable, since some of the data features such as sunspots were used in earlier models we will not be using them here and the focus will solely be on forecasting, thus we will be using DST and time-based variables of the dataset.

Preprocessing:

The dataset for the XGBoost model consisted of time-series data representing the DST index. The key attributes were the timestamps and corresponding DST index values. We also added additional features to improve the model's performance:

```
# Short-term test split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=5)

# Long-term test split
X_train_long, X_test_long, y_train_long, y_test_long =
train_test_split(X, y, test_size=100)

model = XGBRegressor()
model.fit(X_train, y_train)
```

- **Lag features:** These represent the DST index from prior time steps, such as Lag1_DST (1 time step behind), Lag2_DST (2 steps behind), etc. To do this, pandas' .shift() function was utilized, you can see this in Figure 4 below.
- **Rolling means:** A moving average calculated over a sliding window, such as 3, 5, or 10-time steps using the .rolling() function, you can see this in Figure 4 below.

```
data['Lag1_DST'] = data['DST'].shift(1)
data['RollingMean5'] = data['DST'].rolling(window=5).mean()
```

Figure 4

- **Time-based features:** These features extracted the hour, day of the week, and month from the timestamps to account for periodic trends, as you can see in Figure 5 below.

```
data['Hour'] = data['Timestamp'].dt.hour
data['Month'] = data['Timestamp'].dt.month
```

Figure 5

- **Train-Test Splitting:** We used many different short-term and long-term test splits to assess XGBoost's performance on various years of the dataset you can see an example of this below in Figure 6.

Figure 6

Reasons for preprocessing:

- **Lag Features** were added to help the model learn from historical patterns in the DST data. For example, if a rapid decline in DST typically follows a sharp increase, lag features help the model capture this.
- **Rolling Mean Features** were used to reduce noise in the data, smoothing out short-term fluctuations and helping the model focus on broader trends in the DST index. This is especially useful for weak storms, where small fluctuations may not be as significant as the overall trend.
- **Time-based features** were introduced to capture any potential periodicity in the data, such as storms being more likely during specific times of the day, week, or month. This would allow XGBoost to better account for cyclical patterns in geomagnetic activity.

Error Handling:

There were very minor errors since the missing values were a key issue (such as the 99999.99 values that were eliminated in the earlier models). When creating lag and rolling mean features, the first few rows in the dataset would lack these values. To handle this, we used forward-filling techniques to propagate the most recent valid observation forward. Any extreme outliers in the DST data were also capped or smoothed out using a median replacement approach. This ensured that the model was not misled by erroneous spikes.

Model analysis of results and figures

1. Short-Term DST Prediction:

In the first graph, the model attempts to predict DST values for the last 5 time steps. Here, the model does a reasonably good job of predicting the general drift but struggles with volatility.

Observations

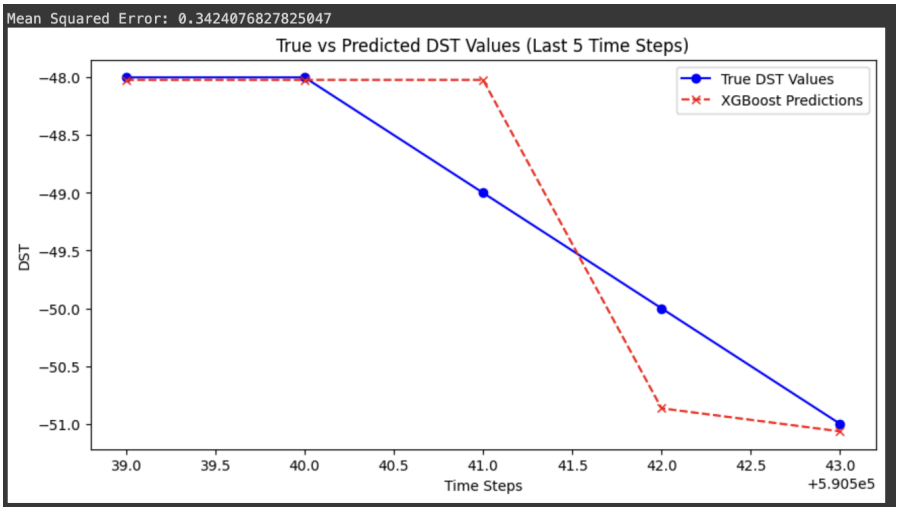
At time step 40, the model perfectly matches the true DST value. Between time steps 40 and 41, there is a sharp decline in the true DST value that the XGBoost model fails to perceive. The prediction line stays flat and only catches up with the actual change at around 43-time steps.

The Mean Squared Error (MSE) for this prediction is **0.34.** This is relatively low and shows that the model is reasonably accurate in this short-term scenario, except for its delay in responding to sudden changes.

Analysis

XGBoost effectively captures smooth trends and minor noise. However, this figure illustrates its limitations with volatility as discussed before. The model

fails to predict it accurately and responsively, which is likely due to the nature of gradient-boosting methods that optimise for incremental improvements rather than high volatility.



2. Medium-Term DST Prediction:

In the second graph, the model is tested over a longer period (100 time steps), showing its prediction performance over a moderate period.

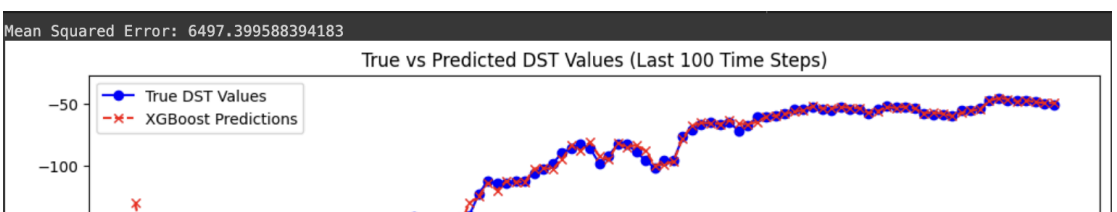
Observations

Initially, the model struggles to follow the true general drift because of its weaknesses in accuracy about sudden, dramatic changes. The model's prediction values stay relatively unchanged in contrast to the true DST values. However (from time step ≈590470 onwards), the model mirrors the true values quite closely as the DST values form a noticeable trend and deviate minimally.

The MSE is 6497.4. This is considerably higher than the MSE over the short term. This tells us the model struggles with accuracy over longer periods, particularly when heavy fluctuations are present.

Analysis

While XGBoost is reasonably successful for short-term trends, its shortcomings are evident in this figure. The flat predictions by the model - while the true DST values are undergoing sharp changes - confirm that the model's canonicity is effective. However, the figure also displays the model's strength in effectively predicting smooth, gradual trends.



3. Weak Storms Short-Term Prediction:

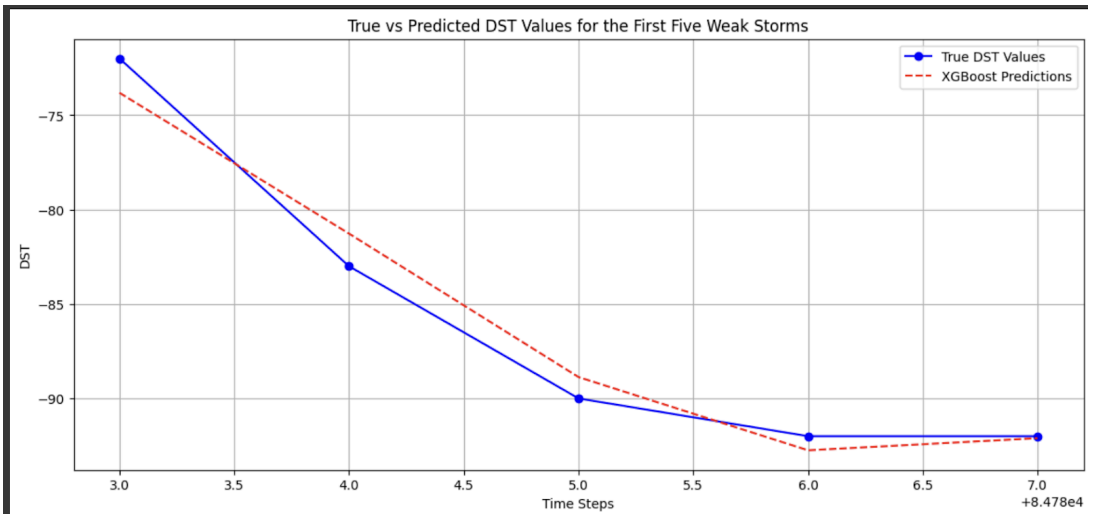
In this graph, the model's ability to predict DST values during weak geomagnetic storms is analysed.

Observations

The model roughly follows the true values consistently throughout the time steps. Overall, the predictions display the same general trend as the true values.

Analysis

The XGBoost model performs very well for weak storms because these events are characterized by much lower volatility in DST. This works well with the model's strengths in capturing slow-moving trends.



4. Final model On DST:

The final graph represents the model's performance over an extended period (several thousand-time steps), capturing a wide range of geomagnetic disturbances.

Observations:

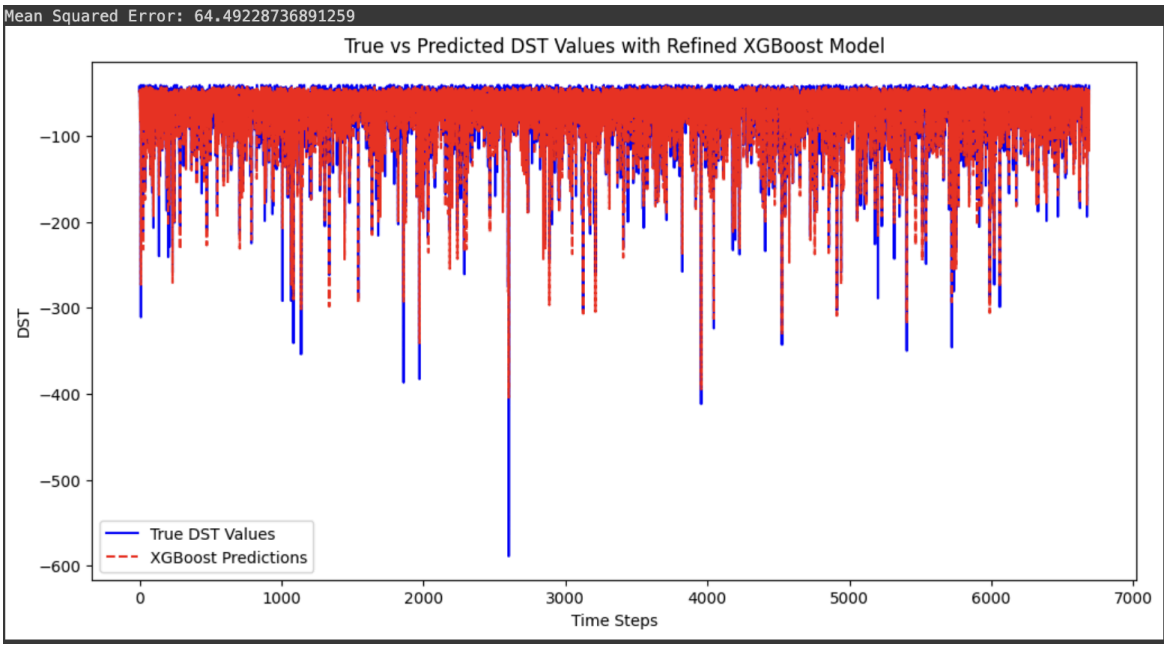
The model generally underestimates the magnitude of sharp drops in DST values.

The MSE for this graph is 64.49 is significantly lower than the Medium-Term DST prediction, indicating an effective accuracy about the true DST values.

Despite the slightly high MSE, the model follows the general trend during periods of mild or moderate geomagnetic activity but usually fails to capture the severity of sharp disturbances.

Analysis:

Over a long time horizon, the model's limitations did not become more pronounced in comparison to the Medium-Term DST prediction. While the model can gauge overall drift during periods of low volatility, its inability to accurately predict sharp dips leads to a higher error. This confirms that XGBoost is efficacious for small trends but is not well-suited for handling highly volatile, rapid changes.



XG Boost Conclusion:

The XGBoost model performs well for short-term predictions and weak to moderate geomagnetic disturbances, where DST changes are gradual and trends are easier to capture. This is likely because XGBoost is a gradient boosting model and it tries to improve on the error of the gradient by fitting improved decision trees which try to minimize this error. Since we have used rolling statistics in the preprocessing, the gradient of DST has been smoothed, and since rolling was a predictor of high importance the decision trees have given it a high bias thus causing weak and moderate storms to be oversampled as these are overrepresented by the rolling statistics. However, the same factor causes it to struggle with sharp, sudden changes in the DST index, particularly during severe geomagnetic storms. Over

longer periods, this limitation becomes more apparent as we have seen in the figures, leading to higher prediction errors and a higher MSE.

In short-term predictions such as those in the sample figures which went over 5-hour intervals, the model performs particularly well, making it a viable candidate for early warning systems. This capability could be leveraged in real-world applications where timely alerts of geomagnetic disturbances are crucial, such as for satellite operators or power grid managers. By providing accurate short-term forecasts, this model could help mitigate the effects of weak storms or provide early notice before larger disturbances, for storms that have the property of weak and moderate storms being accompanied by severe storms. Overall we can conclude that this model is the best for most of the quiet, weak, and moderate storm periods as it is capable of predicting those types of storms for all intervals..

Plans for Future Work

Given more time and resources, we would attempt to implement the following features:

2. **Hybrid Model Architectures**
    - **CNN-LSTM and GRU Integration:** Combining Convolutional Neural Networks (CNNs) with LSTM or Gated Recurrent Unit (GRU) layers to capture both spatial and temporal features in the data, potentially improving the model's ability to detect complex patterns associated with geomagnetic storms. (This would be best optimised with more variables such as solar wind data, magnetosphere image data and the auroral electrojet indices)

3. **Storm Phase Classifier Integration**
    - **Separate Storm Classification Model:** Developing a classifier to predict storm phases (calm, growth, ongoing) and integrating its outputs into the forecasting model. This would help the model to adjust its predictions based on the current or upcoming storm phase. Furthermore, continuously integrating model outputs into our more advanced models could potentially boost the predictive power by allowing the model to learn from the mistakes of other models. Almost like an aggregate of many models.

4. **Dynamic Input Window**
    - **Adaptive Sequence Lengths:** Implementing a dynamic window that adjusts the input sequence length based on the severity or duration of storms. This would provide the model with more contextual information during critical periods. Unfortunately, however, we are unsure if this is compatible with LSTM, CNN GRU keras or pytorch model architecture.

5. **Explainable AI Techniques**
    - **LIME (Local Interpretable Model-agnostic Explanations):** Applying LIME or other explainable AI systems to help us understand the model's decision-making processes, especially during local storm predictions. LIME would hopefully pick up on which variables were most important, including gradient index, sunspot data, climax history and more (best implementation would include solar wind, auroral electrojet and magnetosphere data). This could reveal which features are most influential, aiding in model refinement and feature selection.

6. **Incorporation of Additional Data Sources**
    - **Solar Wind and Auroral Electrojet Data:** Integrating real-time solar wind measurements and auroral electrojet indices to enhance the model's predictive capabilities.
    - **Magnetograph Images:** Utilising solar magnetogram images to capture sunspot activity and solar flares, providing richer input data. This could be computationally expensive, so could potentially be used in conjunction with the storm classifier model, allowing the model to only check the state of the magnetograph images if there is a higher likelihood of a storm forming, to help it “prepare” for the storm.

7. **Addressing Class Imbalance**
    - **Split models:** To help the model focus on storms, we wanted to implement a split model, that, utilising statefulness could work two models on one dataset at a time.
    - **Advanced Loss Functions:** Alternatively, experiment with different loss functions (as there are plenty out there) that may focus a little more on unbalanced data like ours.

8. **Enhanced Computational Resources**
    - **High-Performance Computing:** Leveraging more powerful GPUs or distributed computing resources to train larger, more complex models and to conduct extensive hyperparameter tuning. Previous studies have even utilized quantum computing in their model-making.

Conclusion

Our project contributed to the discussion around geomagnetic storm forecasting by attempting to predict the Disturbance Storm Time (Dst) index 1 to 6 hours in advance using advanced machine learning models **LSTM, CNN, GRU networks and XGBoost**. Our exploratory data analysis revealed a heavily imbalanced dataset dominated by calm conditions, with severe storms constituting only about **5.75% of the data.** This **class imbalance** poses significant challenges for all DST researchers attempting to build predictive models for this complex space phenomenon, as **traditional models tended to minimise overall error while failing to accurately predict critical storm events.**

The challenges we faced are outlined below:

Early Convergence and Overfitting: Our first few models converged too quickly for unknown reasons and seemed to be overfitting the training data as there were excessive parameters relative to the dataset size. We adjusted training parameters to decrease convergence sensitivity and made our models simpler but with more layers to reduce overall parameters.

Ineffective Storm Predictions: Early models performed worse than a simple algorithmic process of simply echoing the data, this “echo” effect then became our baseline for model performance, a model needed to beat this “echo” to count as a semi-insightful model.

Limitations of MSE as an Evaluation Metric: Attempting to identify the “top” or “best” model, proved futile in the end, as many models performed different tasks to varying degrees of success, simpler models may predict calmer periods better, whereas more complex models may predict more stormy weather better. With MSE being an overview of the whole dataset, we found it was an insufficient evaluation metric for how “good” our solar storm predictive models are.

To address these issues, we implemented weighted loss functions to prioritize learning from storm events, engineered additional features for better context, and explored alternative models like XGBoost. Despite improvements in predicting calm and moderate conditions, accurately forecasting severe storms remained challenging, underscoring the need for more tailored evaluation metrics and modelling strategies.

Future Directions:

Develop Specialised Models: Create models designed to handle imbalanced, volatile time series data.

Integrate Additional Data Sources: Incorporate real-time solar wind measurements and other relevant features to improve predictive capabilities.

Advance Evaluation Frameworks: Use metrics that specifically assess accuracy during storm periods, possibly employing cost-sensitive learning.

Explore Hybrid Models: Combine different algorithms, such as integrating XGBoost with LSTM networks, to capture both gradual trends and sudden changes.

Accurately predicting geomagnetic storms is a complex challenge and won't be simply solved by any one of our models. Our findings contribute to the discussion around DST prediction and highlight the importance of appropriate evaluation metrics and specialised modelling approaches not just to DST prediction but to all forecast modelling. Deep, comprehensive research and interdisciplinary collaboration are invaluable when it comes to developing valuable, useful and accurate forecast models.

References

- Admajorjora. (2023, October). The Simple Maths Behind XGBoost - Aditya rajora - Medium. Medium. https://medium.com/@57fdaditya/a-simple-maths-behind-xgboost-d13aac14e578
- Brownlee, Jason.(2020, Aug 28) “How to Develop Convolutional Neural Network Models for Time Series Forecasting” *Machine Learning Mastery.* https://machinelearningmastery.com/how-to-develop-convolutional-neural-network-models-for-time-series-forecasting/
- Craig, Lev. “convolutional neural network (CNN)” *TechTarget.* https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network
- Introduction to Boosted Trees — xgboost 1.5.1 documentation. (n.d.). Xgboost.readthedocs.io. https://xgboost.readthedocs.io/en/stable/tutorials/model.html
- “Metis Solar Orbiter.” *Inaf.it*, 2022, metis.oato.inaf.it/. Accessed 19 Oct. 2024
- Nvidia. (2024). What is XGBoost? NVIDIA Data Science Glossary. https://www.nvidia.com/en-us/glossary/xgboost/
- Rabby, Md Fazle, et al. “Stacked LSTM Based Deep Recurrent Neural Network with Kalman Smoothing for Blood Glucose Prediction.” *BMC Medical Informatics and Decision Making*, vol. 21, no. 1, 16 Mar. 2021, https://doi.org/10.1186/s12911-021-01462-5.
- Reddit - Dive into anything. (2024). Reddit.com. https://www.reddit.com/r/MachineLearning/comments/peyyfx/xgboost_vs_lstm_d/
- Sciences, Chinese Academy of. “Global Maps of the Solar Corona Magnetic Field Created for the First Time.” *SciTechDaily*, 24 Sept. 2020, scitechdaily.com/global-maps-of-the-solar-corona-magnetic-field-created-for-the-first-time/.
- Van Houdt, Greg, et al. “A Review on the Long Short-Term Memory Model.” *Artificial Intelligence Review*, vol. 53, no. 8, 13 May 2020, https://doi.org/10.1007/s10462-020-09838-1.