remote sensing

MDPI

Article

Solar Flare Prediction Using Multivariate Time Series of Photospheric Magnetic Field Parameters: A Comparative Analysis of Vector, Time Series, and Graph Data Representations

Onur Vural *, Shah Muhammad Hamdi and Soukaina Filali Boubrahimi

Department of Computer Science, Utah State University, Logan, UT 84322, USA; s.hamdi@usu.edu (S.M.H.); soukaina.boubrahimi@usu.edu (S.F.B.)
* Correspondence: onur.vural@usu.edu

Abstract: The purpose of this study is to provide a comprehensive resource for the selection of data representations for machine learning-oriented models and components in solar flare prediction tasks. Major solar flares occurring in the solar corona and heliosphere can bring potential destructive consequences, posing significant risks to astronauts, space stations, electronics, communication systems, and numerous technological infrastructures. For this reason, the accurate detection of major flares is essential for mitigating these hazards and ensuring the safety of our technology-dependent society. In response, leveraging machine learning techniques for predicting solar flares has emerged as a significant application within the realm of data science, relying on sensor data collected from solar active region photospheric magnetic fields by space- and ground-based observatories. In this research, three distinct solar flare prediction strategies utilizing the photospheric magnetic field parameter-based multivariate time series dataset are evaluated, with a focus on data representation techniques. Specifically, we examine vector-based, time series-based, and graph-based approaches to identify the most effective data representation for capturing key characteristics of the dataset. The vector-based approach condenses multivariate time series into a compressed vector form, the time series representation leverages temporal patterns, and the graph-based method models interdependencies between magnetic field parameters. The results demonstrate that the vector representation approach exhibits exceptional robustness in predicting solar flares, consistently yielding strong and reliable classification outcomes by effectively encapsulating the intricate relationships within photospheric magnetic field data when coupled with appropriate downstream machine learning classifiers.

Keywords: solar flare prediction; multivariate time series classification; representation learning; graph machine learning; time series features; deep learning

check for updates

Academic Editor: Manuel Antón

Received: 30 January 2025
Revised: 9 March 2025
Accepted: 13 March 2025
Published: 18 March 2025

Citation: Vural, O.; Hamdi, S.M.; Boubrahimi, S.F. Solar Flare Prediction Using Multivariate Time Series of Photospheric Magnetic Field Parameters: A Comparative Analysis of Vector, Time Series, and Graph Data Representations. *Remote Sens.* **2025**, *17*, 1075. https://doi.org/10.3390/rs17061075

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

A solar flare is a sudden burst in magnetic flux emanating from the Sun's surface within the solar corona and heliosphere. In accordance with peak soft X-Ray flux in the 1–8 Å wavelength range, flares can be logarithmically categorized as A, B, C, M, and X, with increasing intensities where M and X category flares correspond to the most intense flare categories [1]. These major solar events, as in Figure 1, can produce emissions such as Gamma Ray, X-Ray, and Extreme Ultraviolet (EUV). As a result, astronauts operating in space stations can face radiation-based health risks, electronic devices such as GPS and radio may fail to function, and serious infrastructure problems may occur [2,3]. Furthermore, the infamous 1859 Carrington Event shows that a similar massive solar superstorm hitting our

contemporary technology-reliant society could lead to a prolonged nationwide blackout in the United States, lasting for months. The aftermath is projected to inflict economic havoc, with anticipated damages reaching USD trillions [4]. Currently, there is no well-established theoretical connection between the influx of magnetic fields and the occurrence of extreme events in solar active regions. Accordingly, in the heliophysics community, this draws emphasis on detailed data analysis and investigation of different methodologies to obtain robust flare prediction performance from magnetic field state data of solar active regions, with sensor observations from space-based and ground-based instruments playing a crucial role in data collection.

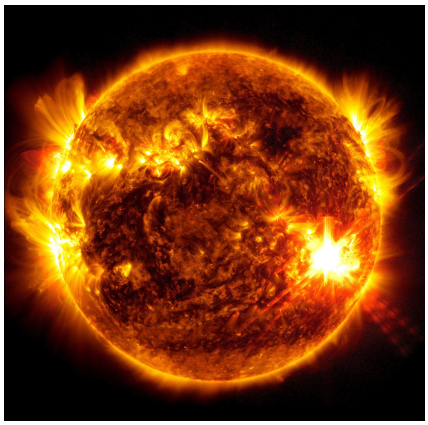


Figure 1. NASA's Solar Dynamics Observatory captured this image of an X5.8-category solar flare peaking at 9:23 p.m. EDT on 10 May 2024.

Recently (2020), Space Weather Analytics for Solar Flares (SWAN-SF) [5] has been proposed as an openly accessible, comprehensive dataset encapsulating the multivariate time series (MVTS) representation which is obtained from solar photospheric vector magnetograms in the Space-weather HMI Active Region Patch (SHARP) series. Including data from solar active regions that are collected from observations between May 2010 and December 2018, SWAN-SF contains features of 24 photospheric field parameters and integrates over 10,000 flare reports. In SWAN-SF, each MVTS instance captures magnetic field data over a predetermined observation period, with labeled flare classes occurring subsequent to a specified prediction time. In this respect, SWAN-SF allows the task of solar flare prediction to be formulated as an MVTS classification task. SWAN-SF enhances the accuracy of solar flare predictions and enables comprehensive investigations into elusive predictors and early indicators of flares. These improvements yield advantages in both practical operational applications and fundamental research [6].

To date, predicting solar flares through photospheric magnetic field data has been explored through various machine learning methodologies, naturally incorporating different data modalities, including the use of SWAN-SF for MVTS classification. However, in these solar flare prediction tasks, there are serious challenges that degrade flare prediction quality, which come about for three reasons: the extreme class imbalance between minor and major flare events, the intricate temporal dynamics, and the interdependencies of magnetic field parameters. To address these challenges and improve predictive performance, obtaining meaningful data representations is a crucial part of developing effective classification models [7]. By leveraging different representations, we can assess how the structure of the input data impacts model performance in downstream tasks, particularly under conditions of extreme class imbalance and noisy data. Accordingly, in this study, we compare the effectiveness of data representation methods for representing the MVTS data instances within the SWAN-SF dataset under conditions of extreme class imbalance, undersampling, and optimized preprocessing. Three distinct methodologies merit examination:

(1) a vector-based approach where MVTS instances of SWAN-SF are reduced to summary vectors, encapsulating key features of an MVTS instance; (2) the maintenance of the native MVTS structure of SWAN-SF, with each univariate time series capturing distinct magnetic field parameters over time; and (3) a graph-based representation, wherein each univariate time series, corresponding to a specific magnetic field parameter, is modeled as a node, and the interdependencies between these parameters are expressed through edges. A thorough comparison of these representations could yield deeper insights into the optimal approach for accurately predicting solar flare occurrences and guide future research through the selection of proper representations while designing state-of-the-art frameworks.

To obtain vector representations, we apply two distinct techniques. First, we extract the last timestamps of the MVTS instances, assuming that the most recent data point from each series encapsulates critical information. Second, we generate summary vectors by aggregating key statistical features across the entire time series to capture the overall temporal dynamics in a compact form. Both techniques reduce the dimensionality of the data while preserving essential characteristics for predictive modeling. For the time series representation, we maintain the full temporal resolution of the data. Each magnetic field parameter is treated as an individual univariate time series, preserving the original time dependencies between parameters. This approach allows us to leverage time series-specific classification models to exploit temporal patterns and capture long-range dependencies critical to solar flare prediction. For the graph representation, we transform the MVTS instances into graphs using two measures defining relationships between variables: correlation and distance similarity. After constructing the graphs, we employ three different techniques to obtain graph embeddings: (1) summarizing each graph using node degree information; (2) utilizing graph embedding algorithms; and (3) using graph neural networks. This allows us to capture both local and global dependencies between magnetic field parameters across the graph structure, offering a new dimension to model interactions effectively. The contributions of this paper are as follows:

- An in-depth exploration of the utilization of three distinct data representations of the SWAN-SF dataset, where each proposed method leverages its unique attributes to enhance predictive performance.
- The full introduction of graph-based machine learning into solar flare prediction tasks by obtaining graph representations, applying multiple graph embedding techniques, and creating a graph convolutional network module.
- The experimental evaluation of the strengths and shortcomings of each data representation method under different conditions of sampling and preprocessing in light of the appropriate selection of performance metrics that align with the nuanced characteristics of the benchmark dataset.

The rest of this paper is organized as follows. In Section 2, we cover related work. In Section 3, we discuss the SWAN-SF dataset and our proposed data representation methods. In Section 4, we present our experimental findings. Finally, we provide concluding remarks and discuss future work in Section 5.

2. Related Work

2.1. Solar Flare Prediction

The earliest attempt at solar flare prediction was THEO, an expert system that relied on human inputs. It was officially adopted for use by the Space Environment Center (SEC) of the National Oceanic and Atmospheric Administration (NOAA) in 1987 [8]. In the following years, with the help of an abundance of magnetic field data collected by different space-based and ground-based observatories, the task of solar prediction was often treated

as a data science task. Focusing on how the data are represented, we review three machine learning-driven techniques for solar flare prediction, categorizing them into vector-based, time series-based, and graph-based approaches.

Vector-based approaches have gained prominence in solar flare prediction, with line-of-sight magnetogram-based models characterizing solar active region photospheric magnetic field parameters by the line-of-sight component, and vector magnetogram-based models capturing solar active region parameters using the full-disk photospheric vector magnetic field [9]. Since 2010, the launch of NASA's Solar Dynamics Observatory (SDO), the Helioseismic and Magnetic Imager (HMI) instrument has continuously mapped the full-disk vector magnetic field every 12 min [10]. As a result, most of the current models utilize this stream of vector magnetogram data from SDO. Nonlinear statistical models, mainly machine learning classifiers, have been widely utilized for solar flare prediction, including logistic regression [11], C4.5 decision tree [12], fully connected neural network [13], support vector machine [3], and relevance vector machine [14].

In the time series-based approach, a significant advancement was made by extending single timestamp-based models to incorporate temporal window-based flare prediction, allowing for more accurate predictions over extended time periods [6]. This resulted in the creation of SWAN-SF [5], an MVTS dataset where each instance records magnetic field data over a preset observation time, providing labeled flare classes occurring after a specific prediction time window. From that point, various MVTS classification approaches were used, such as statistical summarization for k-nearest neighbors training [2], MVTS decision trees with clustering as a preprocessing step [15], deep sequence modeling for end-to-end flare classification with automated feature learning [16], and contrastive representation learning for addressing challenges of temporal dependencies and extreme class imbalance [7,17]. Additionally, there have been recent attempts to provide comparative analysis of deep learning models that utilize full-disk line-of-sight magnetograms and active region parameters, highlighting the strengths and weaknesses of some vector-based and time series-based models in capturing the dynamics of flare-imminent active regions [18]. In other studies, the focus was extended to the diversity of sequence models for time series of active region parameters, including long short-term memory (LSTM), LSTM with attention (LSTM-A), bidirectional LSTM (BLSTM), and BLSTM with attention (BLSTM-A), to predict flare classes [19]. However, there is still limited exploration of graph models in this area of research.

While graph-based representation learning has not been explored as extensively as vector-based and time series-based approaches, there have been some efforts to apply it to solar flare prediction. One such example is the use of functional network embedding together with sequence modeling to capture both temporal and spatial relationships of MVTS instances. The graph approach is evident in the functional network embedding stage where the inter-parameter dependencies are modeled as the edges in graphs [20]. Section 2.2 provides an overview of recent advancements in graph-based machine learning techniques to obtain fixed-dimension embeddings from graph data. Overall, these diverse methodologies showcase the evolution from traditional linear models to sophisticated machine learning techniques in predicting solar flare activities.

2.2. Graph Representation Learning

Graph representation learning is a fundamental part of graph-based machine learning research, a process aimed at embedding graph structures into a lower-dimensional space while preserving the underlying relationships and dependencies between nodes and edges that make up the graph [21]. By learning these embeddings, the intricate structures of graphs (e.g., node proximity, connectivity patterns, and community structures) can be

effectively captured and utilized in machine learning models [22,23]. This method allows complex graph data to be transformed into meaningful representations that can be used for a variety of downstream tasks such as node classification, link prediction, and graph classification. Among these representation learning strategies, graph embedding algorithms provide alternative ways to represent components of graph structures. One example is the matrix decomposition-based Laplacian Eigenmaps algorithm, which uses eigenvectors derived from the graph Laplacian matrix to cover the one-hop neighborhood of nodes [24]. GraRep learns one d-dimensional embedding for each k-th order proximity, integrating global structural information of the graph into the learning process [25]. Another example is tensor decomposition-based node embedding, which uses higher-order transition probability matrices to construct one or more third-order tensors [26]. DeepWalk and Node2Vec employ flexible, biased random walks that can balance between a local and global view of the network [27,28]. Moreover, neural networks adapted and modified specifically for graph tasks have increasingly gained popularity in graph representation learning. Graph neural networks (GNNs) extend traditional neural networks, enabling them to operate on graph-structured data to model complex relationships and dependencies. Following a local aggregation mechanism, the GNN processes the node embedding vector of a node by recursively aggregating the embedding vectors of local neighborhood nodes [29,30]. This novel deep learning-based approach has led to many variations. One example is the graph convolutional network (GCN), which provides a localized first-order approximation of spectral graph convolutions [31]. Another variation is Graph-SAGE, an inductive framework that leverages node feature information to efficiently generate embeddings for previously unseen data [32]. The graph attention network (GAT) introduces masked self-attentional layers to overcome the limitations of earlier methods based on graph convolutions or their approximations [33]. These advancements highlight significant achievements in graph representation learning.

3. Materials and Methods

3.1. Data Collection

Various studies aimed to utilize machine learning for the task of solar flare prediction; however, they were using point-in-time measurements [3]. As a result, it becomes difficult to ascertain whether variations in accuracy or skill score values across these studies stem from the inherent stochasticity in flare occurrence, the preprocessing and sampling approaches employed, or the execution of machine learning models [1]. The SWAN-SF dataset [5] was created to address the issues, being composed of MVTS, to enable impartial flare forecasting and contribute to significant advancements in forecasting that go beyond incremental improvements. SWAN-SF primarily relies on sensor data obtained from NASA's SDO, specifically the SHARPs from the Joint Science Operations Center (JSOC). These patches are derived from solar vector magnetograms captured by the HMI instrument that functions as a sensor onboard SDO, continuously observing the Sun and providing detailed magnetic field data from the solar photosphere. However, the SHARPs do not contain flare information, which is instead sourced from the Geostationary Operational Environmental Satellites (GOES) operated by NOAA. GOES, equipped with X-ray and particle detectors, has been detecting solar flares since 1975 and provides a catalog of flare events, including key details like flare time, classification, and spatial location on the solar disk, thereby providing the labeling information. The dataset creation process involves three key steps. First, solar flare reports from GOES are cleaned to resolve conflicting information. Second, these flare reports are matched with solar magnetic data from the HMI, either through matching NOAA AR numbers with HARP numbers in SHARPs or using a spatiotemporal overlap procedure to align flare occurrences with HMI active region

patches. Lastly, sampling biases are eliminated to ensure a balanced and accurate dataset for training machine learning models [6]. Figure 2 illustrates the data pipeline.

Data points within SWAN-SF belong to one of five distinct categories—FQ, B, C, M, and X—each representing events with varying intensity. FQ contains both flare-quiet and category A flare events. For the task of solar flare prediction, we categorize SWAN-SF instances into two main classes, nonflare (NF) and flare (F) superclasses, encompassing all mentioned categories. The NF superclass includes FQ, B, and C category instances that are either quiet or minor flaring events. Conversely, the F superclass contains M and X flare categories, which are major flare types that can lead to severe consequences, including radiation-related health risks for astronauts, damage to infrastructure, and failures of electronic equipment and signals [2–4].

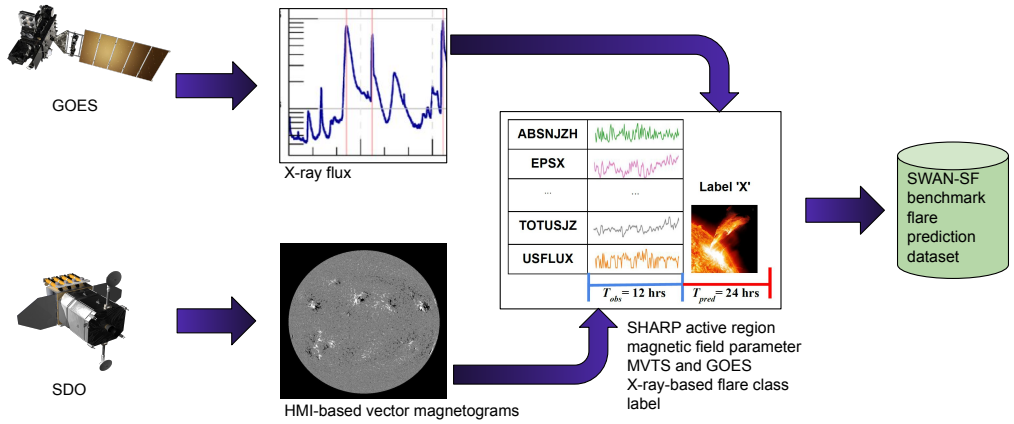


Figure 2. Photospheric magnetic field parameter-based MVTS data pipeline for flare prediction using GOES and SDO sensor observations.

Each instance in SWAN-SF corresponds to an MVTS slice of multiple magnetic field parameters extracted from solar active regions with a sliding window. Accordingly, for a flare having a unique ID, multiple equal-length MVTS data instances are obtained. This time frame is referred to as the observation window (T_{obs}). If s_i denotes the starting point of the i-th time series segment, then the subsequent $(i+1)$-th segment starts at $s_i + \tau$, where τ represents the step size in the sliding process. Each extracted MVTS is labeled based on the class of the most intense flare observed within a predefined temporal window following T_{obs}. This window is known as the prediction window (T_{pred}) and commences precisely at the end of T_{obs}. In the SWAN-SF context, T_{obs} and T_{pred} extend over 12 and 24 h, respectively, with τ set at 1 h [1,6]. Each MVTS data instance in SWAN-SF can be represented as $M^{(k)} \in \mathbb{R}^{\tau \times N}$, where $1 \leq k \leq K$ (i.e., K is the total number of MVTS instances), a collection of univariate time series corresponding to N magnetic field parameters, each having a step size length of τ. The N magnetic field parameters as univariate time series match the first 24 parameters derived in Bobra et al.'s work [3]. Table 1 provides a full list of these magnetic field parameters, including their corresponding mathematical formulas.

Table 1. Solar active region photospheric magnetic field parameters.

Abbreviation	Description	Formula
ABSNJZH	Absolute value of the net current helicity	$H_{c_{abs}} \propto \lvert \sum B_z \cdot J_z \rvert$
EPSX	Sum of x-component of normalized Lorentz force	$\delta F_x \propto \frac{\sum B_x B_z}{\sum B^2}$

Table 1. *Cont.*

Abbreviation	Description	Formula
EPSY	Sum of y-component of normalized Lorentz force	$\delta F_y \propto \frac{-\sum B_y B_z}{\sum B^2}$
EPSZ	Sum of z-component of normalized Lorentz force	$\delta F_z \propto \frac{\sum \left(B_x^2 + B_y^2 - B_z^2\right)}{\sum B^2}$
MEANALP	Mean characteristic twist parameter, α	$\alpha_{\text{total}} \propto \frac{\sum J_z \cdot B_z}{\sum B_z^2}$
MEANGAM	Mean angle of field from radial	$\overline{\gamma} = \frac{1}{N} \sum \arctan\left(\frac{B_h}{B_z}\right)$
MEANGBH	Mean gradient of horizontal field	$\overline{\lvert \nabla B_h \rvert} = \frac{1}{N} \sum \sqrt{\left(\frac{\partial B_h}{\partial x}\right)^2 + \left(\frac{\partial B_h}{\partial y}\right)^2}$
MEANGBT	Mean gradient of total field	$\overline{\lvert \nabla B_{tot} \rvert} = \frac{1}{N} \sum \sqrt{\left(\frac{\partial B}{\partial x}\right)^2 + \left(\frac{\partial B}{\partial y}\right)^2}$
MEANGBZ	Mean gradient of vertical field	$\overline{\lvert \nabla B_z \rvert} = \frac{1}{N} \sum \sqrt{\left(\frac{\partial B_z}{\partial x}\right)^2 + \left(\frac{\partial B_z}{\partial y}\right)^2}$
MEANJZD	Mean vertical current density	$\overline{J_z} \propto \frac{1}{N} \sum \left(\frac{\partial B_y}{\partial x} - \frac{\partial B_x}{\partial y}\right)$
MEANJZH	Mean current helicity (B_z contribution)	$\overline{H_c} \propto \frac{1}{N} \sum B_z \cdot J_z$
MEANPOT	Mean photospheric magnetic free energy	$\overline{\rho} \propto \frac{1}{N} \sum \left(\boldsymbol{B}^{Obs} - \boldsymbol{B}^{Pot}\right)^2$
MEANSHR	Mean shear angle	$\overline{\Gamma} = \frac{1}{N} \sum \arccos\left(\frac{\boldsymbol{B}^{Obs} \cdot \boldsymbol{B}^{\text{Pot}}}{\lvert B^{Obs} \rvert \lvert B^{Pot} \rvert}\right)$
R_VALUE	Sum of flux near polarity inversion line	$\Phi = \Sigma \lvert B_{\text{LoS}} \rvert dA$ (within R mask)
SAVNCPP	Sum of the modulus of the net current per polarity	$J_{z_{sum}} \propto \lvert \sum^{B_z^+} J_z dA \rvert + \lvert \sum^{B_z^-} J_z dA \rvert$
SHRGT45	Fraction of area with shear > 45°	Area with shear > 45°/total area
TOTBSQ	Total magnitude of Lorentz force	$F \propto \sum B^2$
TOTFX	Sum of x-component of Lorentz force	$F_x \propto -\sum B_x B_z dA$
TOTFY	Sum of y-component of Lorentz force	$F_y \propto \sum B_y B_z dA$
TOTFZ	Sum of z-component of Lorentz force	$F_z \propto \sum \left(B_x^2 + B_y^2 - B_z^2\right) dA$
TOTPOT	Total photospheric magnetic free energy density	$\rho_{tot} \propto \sum (\boldsymbol{B}^{Obs} - \boldsymbol{B}^{Pot})^2 dA$
TOTUSJH	Total unsigned current helicity	$H_{c_{\text{total}}} \propto \sum B_z \cdot J_z$
TOTUSJZ	Total unsigned vertical current	$J_{z_{total}} = \sum \lvert J_z \rvert dA$
USFLUX	Total unsigned flux	$\Phi = \sum \lvert B_z \rvert dA$

SWAN-SF is divided into five partitions (i.e., P1, P2, P3, P4, and P5), where each partition covers a different observation period. Due to the nature of the dataset, where major flare events occur infrequently, SWAN-SF experiences a significant class imbalance issue between NF and F examples. In our task of binary solar flare prediction, F-class examples represent positive classes, whereas NF-class examples represent negative classes. Because of the negative-class MVTS instances dominating the dataset, classification results tend to favor the majority class, resulting in low flare prediction performance. In analyzing the data for partitions 1 through 5, we observe notably low flare-to-nonflare (F:NF) ratios of 1:58, 1:62, 1:29, 1:43, and 1:75, respectively. Figure 3 demonstrates the class distributions in detail. This extreme class imbalance presents a significant challenge in predicting solar flares in flare-forecasting research. Accordingly, for a robust and goal-based evaluation of the classification results, proper evaluation metrics must be selected. These evaluation metrics are discussed under Section 4.1.

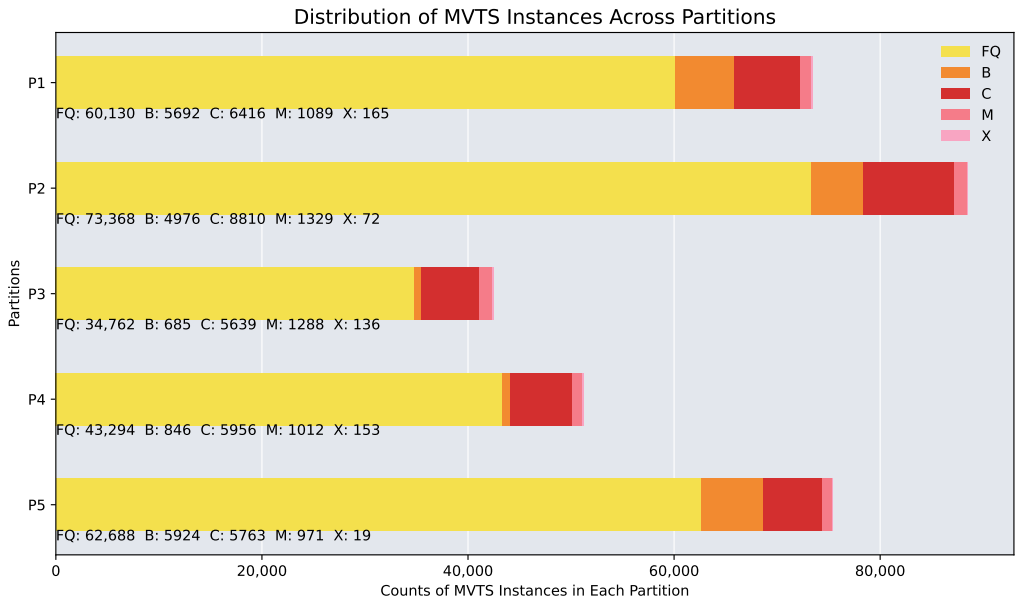


Figure 3. Class distributions for each partition are presented in a stacked bar plot format. The plot displays five flare classes for each partition, along with their corresponding values.

3.2. Overview: Data Representation Methods

For clarity and organization, we categorize data representations into three distinct methods and present multiple strategies within each, capturing different structural and temporal aspects of MVTS data. The vector representation method transforms MVTS instances into a fixed-length feature vector through two strategies: vector of last timestamp (VLT), which retains the final recorded state of each time series, and vector of statistical summary (VSTAT), which encapsulates key statistical properties. The time series representation method preserves temporal dependencies, treating the task as a time series classification (TSC) problem and utilizing specialized classifiers to learn from time-ordered information, capturing meaningful time-evolving patterns. The graph representation method models each univariate time series as a node within a graph, with edges encoding relationships between magnetic field parameters. This approach comprises three strategies: graph node degree (GND), which captures structural importance; graph node embedding (GNE), which learns latent representations; and graph neural networks (GNNs), which exploit complex graph-based interdependencies between features. An overview of these methods is provided in Table 2.

Table 2. Overview of data representation methods and the strategies within each.

Abbreviation	Name
Vector representation	
VLT	Vector of last timestamp
VSTAT	Vector of statistical summary
Time series representation	
TSC	Time series classification
Graph representation	
GND	Graph node degree
GNE	Graph node embedding
GNN	Graph neural network

3.3. Method 1: Vector Representation of SWAN-SF

In this paper, we use two strategies for vector-based representation. The aim is to summarize and simplify the features of MVTS instances such that they can be effectively utilized by the downstream classifiers to produce reliable solar flare prediction results.

3.3.1. Vector of Last Timestamp Representation

Inspired by the work presented in [3], we represent vector magnetogram data by utilizing only the last timestamp of the MVTS instances for all magnetic field parameters. The reason for this choice is that the last timestamp is temporally closest to the flaring event. Parallel to this, for each MVTS instance $M^{(k)} \in \mathbb{R}^{\tau \times N}$, we extract a one-dimensional feature vector $V_{VLT}^{(k)} \in \mathbb{R}^{N}$. After obtaining this representation vector for each MVTS instance, denoted as $V_{VLT} \in \mathbb{R}^{K \times N}$ where K is the total number of MVTS instances, we use V_{VLT} as input to train the downstream module in a supervised manner to provide the class prediction. Figure 4 gives an insight into the process of solar flare prediction using VLT representation.

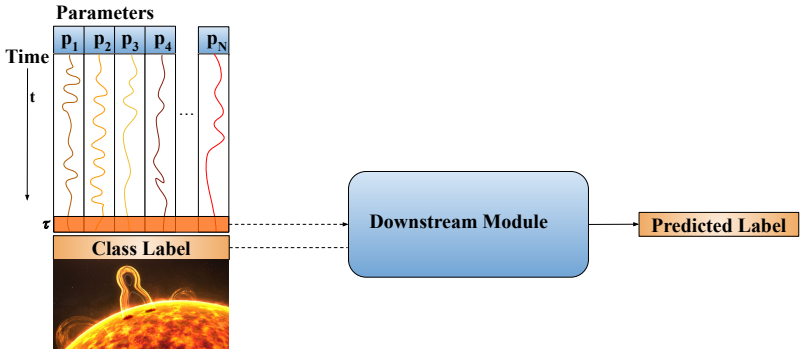


Figure 4. An overview of the vector-based solar flare prediction process by extracting the last timestamp from the MVTS instance.

3.3.2. Vector of Statistical Summary Representation

Inspired by the work presented in [1], we extract a limited set of descriptive statistics from each univariate time series within the MVTS instances, specifically the median, mean, standard deviation, skewness, and kurtosis. Unlike the original work, we also incorporate the point-in-time last-value feature in a manner similar to the VLT approach to have greater representation capabilities. For each MVTS instance $M^{(k)} \in \mathbb{R}^{\tau \times N}$, we extract a feature vector $V_{VSTAT}^{(k)} \in \mathbb{R}^{6N}$ by taking the mentioned six descriptive parameters. After obtaining this representation vector for each MVTS instance, denoted as $V_{VSTAT} \in \mathbb{R}^{K \times 6N}$ with K representing the total number of MVTS instances, we use V_{VSTAT} as input to train the downstream module in a supervised setting. Figure 5 gives an insight into the process of solar flare prediction using VSTAT representation.

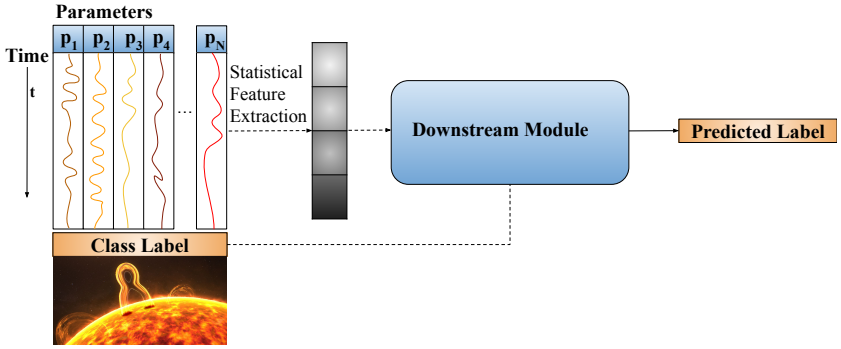


Figure 5. An overview of the vector-based solar flare prediction process by extracting the statistical features and last timestamp features from the MVTS instance.

3.4. Method 2: Time Series Representation of SWAN-SF

Considering the MVTS data instances in their inherent modality, we leverage TSC strategy for solar flare prediction in this method. Figure 6 gives an insight into the process of solar flare prediction using time series representation. In this respect, we employ five distinct models utilizing the MVTS representation, including time series classifiers and sequence models:

- Shapelet Transform (ST): ST aims to effectively capture multivariate features by focusing on shapelets. Shapelets are small, distinctive subsequences of a time series that are particularly effective for classification tasks. They are characterized by being phase-independent, meaning they can identify patterns regardless of their position in the time series. Shapelets are designed to capture unique and discriminative patterns that distinguish between different classes, making them a powerful tool for time series classification [34,35].
- Time Series Forest (TSF): TSF utilizes a blend of the entropy gain and a distance measure, termed the Entrance gain, to assess splits in the tree nodes. The algorithm randomly selects features at each tree node, exhibiting a computational complexity linear to the length of a time series [36].
- Random Convolutional Kernel Transform (ROCKET): ROCKET expands upon the recent achievements of convolutional neural networks in time series classification, employing randomly generated convolutional kernels to achieve state-of-the-art accuracy while being efficient in the use of computational resources [37].
- Recurrent Neural Network (RNN): As a neural network designed for sequential data processing, the RNN maintains hidden states to retain information from previous inputs, making it suitable for sequence modeling tasks, including time series analysis and natural language processing [38].
- Long Short-Term Memory (LSTM): As a specialized type of RNN that addresses the challenges of learning long-range dependencies of the input sequences, LSTM introduces memory cells and gating mechanisms to selectively store and retrieve information [39].

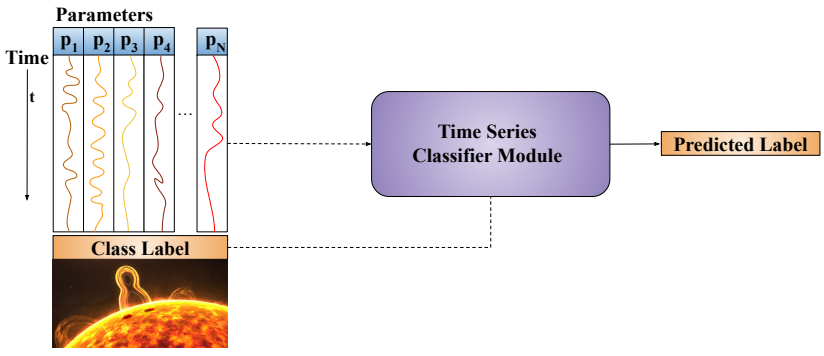


Figure 6. An overview of the time series-based solar flare prediction process by feeding each MVTS instance into the time series classification module as shown.

3.5. Method 3: Graph Representation of SWAN-SF

3.5.1. Functional Network-Based Graph Creation

Most of the existing MVTS classification methods concentrate on the temporal dependencies of the time series. Yet, in tasks of this nature, there might exist an intricate interdependence among MVTS variables. Capturing this relationship not only constitutes a noteworthy contribution to enhancing classification performance but also poses a significant challenge. Using graph-based representations is a strong candidate since complex pairwise dependencies among multivariate variables can be better described using ad-

vanced graph methods, where each variable is considered as a node in the graph, and their dependencies (e.g., positive correlations) are regarded as edges [21]. In this paper, we use two strategies to convert each MVTS instance, $M^{(k)} \in \mathbb{R}^{\tau \times N}$, into graphs $G = (V, E)$, where V denotes the nodes representing the individual magnetic field parameters and E represents the edges capturing the relationships between these parameters. We aim to utilize the graph-based approach to uncover spatial properties and discover interaction patterns between magnetic field parameters within MVTS instances.

- Correlation-Based Graph (COR) Creation: For each MVTS instance, $M^{(k)} \in \mathbb{R}^{\tau \times N}$, we first calculate the Pearson correlation matrix $C^{(k)} \in \mathbb{R}^{N \times N}$ such that $C^{(k)}_{i,j}$ corresponds to the Pearson correlation coefficient value c (where $-1 \leq c \leq 1$) between univariate time series $i^{(k)}$ and $j^{(k)}$ corresponding to the i-th and j-th column of $M^{(k)}$, respectively. We utilize this symmetric matrix $C^{(k)}$ as the adjacency matrix in the graph creation stage where each univariate time series is considered as a node in the graph. After applying a zero threshold to enforce the condition of only creating edges with positive weight, we create the correlation graph instances, $G^{(k)}_C = (V, E)$, from the correlation matrices of each MVTS instance.
- Distance Similarity-Based Graph (DS) Creation: For each MVTS instance, $M^{(k)} \in \mathbb{R}^{\tau \times N}$, we calculate the Euclidean distance matrix $E^{(k)} \in \mathbb{R}^{N \times N}$ such that $E^{(k)}_{i,j}$ corresponds to the Euclidean distance similarity measure e between univariate time series $i^{(k)}$ and $j^{(k)}$ corresponding to the i-th and j-th column of $M^{(k)}$, respectively. In the following step, we utilize the symmetric matrix $E^{(k)}$ as the adjacency matrix in the graph creation stage where each univariate time series is considered as a node in the graph. After applying a threshold t in creating edges, we create the Euclidian graph instances, $G^{(k)}_E = (V, E)$, from the Euclidian distance similarity matrices of each MVTS instance.

3.5.2. Graph Node Degree Representation

After obtaining graphs from each MVTS instance $M^{(k)}$ as $G^{(k)}_C$ and $G^{(k)}_E$, we vectorize the graph instances by extracting the degree information of nodes in the graph. The degree of a node can be described as the number of connections that the particular node forms with other nodes in the graph. Each graph instance yields a degree vector of $D^{(k)} \in \mathbb{R}^N$ containing the degrees of each node in the graph to compactly describe the graph structure. In this respect, for all graph instances in the training set, we obtain the degree matrix, $D \in \mathbb{R}^{K \times N}$, with K being the number of graphs. We use this information as input in the training stage for the downstream module. Figure 7 gives an insight into the process of solar flare prediction using degree representation.

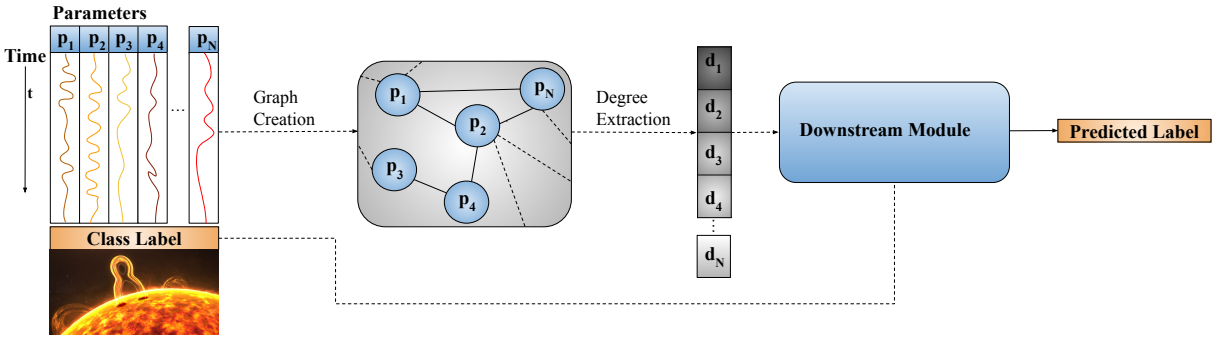


Figure 7. An overview of the graph-based solar flare prediction process by converting each MVTS instance into a graph and modeling inter-parameter relations as edges. Node degrees are extracted in the following step for training purposes.

3.5.3. Graph Node Embedding Representation

To investigate the effect of using node embeddings on solar flare prediction, we process the created graphs G_C and G_E by obtaining their node embeddings. Accordingly, we select two node embedding algorithms for this task. Figure 8 gives an insight into the process of solar flare prediction using the embeddings obtained from MVTS graphs.

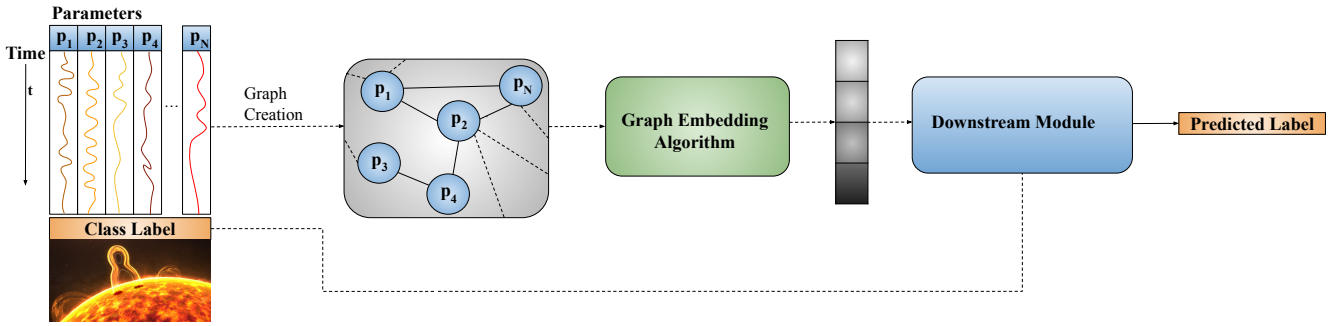


Figure 8. An overview of the graph-based solar flare prediction process where, after converting each MVTS instance into a graph, embeddings are learned to capture underlying patterns for prediction purposes.

- Laplacian Eigenmaps (LAP) [24]: In this method, for each created graph $G^{(k)}$, we obtain the corresponding Laplacian matrix as the first step. In the following step, we extract the eigenvectors and eigenvalues with eigendecomposition. After sorting the eigenvalues in descending order, we select the top-d eigenvectors representing each node in the graph by d dimensions. This results in an embedding matrix, $L^{(k)} \in \mathbb{R}^{N \times d}$, for each graph. After flattening this matrix, a feature vector of size Nd is used as the representation vector of the entire graph. In this respect, for all graph instances in the training set, we obtain the matrix $L \in \mathbb{R}^{K \times Nd}$, with K being the number of graphs. This information is fed into the downstream module for training purposes.
- Node2Vec (N2V) [28]: In this method, we utilize the Node2Vec algorithm to generate d-dimensional node embeddings for each node in the graph. Node2Vec is a graph embedding algorithm that leverages random walks to explore the structure of the graph. It performs biased random walks to capture both local and global structural information. Specifically, Node2Vec generates embeddings by performing random walks starting from each node, with the walk length w fixed. During these walks, the algorithm uses two parameters, p and q, to control the trade-off between breadth-first search (BFS) and depth-first search (DFS), allowing it to capture different types of structural relationships in the graph. The resulting node sequences from the random walks are then used to train a skip-gram model to learn node representations based on their co-occurrence in the walks. As a result, for each graph $G^{(k)}$ we obtain an embedding vector $NV^{(k)} \in \mathbb{R}^{N \times d}$. After flattening the matrix and obtaining the Nd-sized representation vector for each graph, $NV \in \mathbb{R}^{K \times Nd}$ where K represents the number of graphs, NV is used to train the downstream module.

3.5.4. Graph Neural Network Representation

To harness the power of GNNs on graph instances, we construct a two-layer GCN module. For each training instance, the GCN requires the graph as well as the node attributes. In this module architecture, for each layer l, the GCN convolutional layer considers the l-hop neighborhood of each node. The first layer has an input channel dimension size corresponding to the node attribute size. The GCN update process for a particular node v's representation in graph $G = (V, E)$, where V is the set of nodes and E is the set of edges in the graph, can be expressed as

$$h_v^{[0]} = x_v \tag{1}$$

$$h_v^{[l+1]} = \text{ReLU}\left(W_g^{[l]} \sum_{u \in N(v)} \frac{w_{uv} h_u^{[l]}}{|N(v)|} + B_g^{[l]} h_v^{[l]} \right), \quad \forall l \in \{0, 1, \ldots, L-1\} \tag{2}$$

$$z_v = h_v^l \tag{3}$$

$$z_G = \frac{1}{|V|} \sum_{v \in V} z_v \tag{4}$$

Let L be the total number of GCN layers, $x_v \in \mathbb{R}^\tau$ be node v's representation, $h_v^{[l]} \in \mathbb{R}^{d_g}$ be node v's representation in layer l, $W_g^{[l]} \in \mathbb{R}^{d_g \times d_g}$ be layer l's weight matrix, $B_g^{[l]} \in \mathbb{R}^{d_g}$ be the bias vector of the l^{th} layer, $N(v)$ be the set of neighbor nodes of a particular node v, w_{uv} be the weight for the edge between node v and neighbor node u, z_v be the final node representation of v at the end of L iterations of neighborhood aggregation, and z_G be the graph representation found by node-level representation averaging [20,31].

In our case, the input channel size is τ, since each node, each univariate time series, has τ values corresponding to their length. The first layer outputs n hidden channels, followed by the rectified linear unit (ReLU) activation function and dropout. ReLU introduces nonlinearity by zeroing out negative values, helping the model learn complex patterns. Dropout, on the other hand, randomly sets some of the activations to zero during training, reducing overfitting and improving generalization. The second layer has a matching input channel size of n and an output channel size of m. Followed by a node-wise average pooling, the GCN is followed by a fully connected layer in an end-to-end fashion to generate raw output values for the class labels. Then, a sigmoid function is applied to these raw outputs, converting them into probabilities suitable for a binary classification task, tailored to our solar flare prediction needs. Figure 9 gives an insight into the process of solar flare prediction using our GCN model.

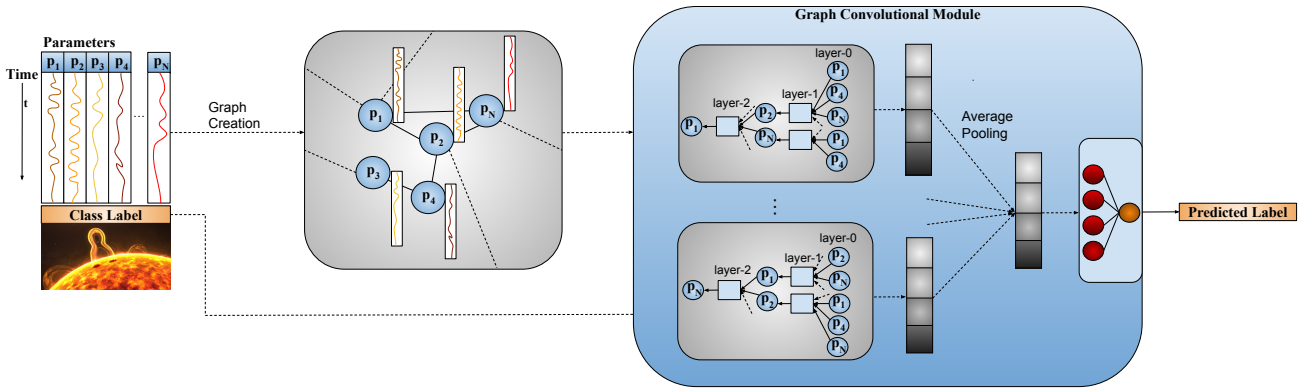


Figure 9. An overview of the graph-based solar flare prediction process is presented, wherein each MVTS instance is transformed into a graph consisting of nodes that represent magnetic field parameters, with their corresponding univariate time series serving as node features. Subsequently, each graph is processed through our graph convolutional module.

4. Results

In this section, we demonstrate our experimental findings. We conducted our experiments on a Windows machine having an AMD Ryzen 7 5800H processor, NVIDIA GeForce RTX 3060 GPU, and 16 GB memory, and we used Python 3.9.13, PyTorch 2.1.0, numpy 1.25.2, scikit-learn 1.2.0, and networkx 2.8.8 with CUDA 11.8 for imple-

menting our models. The source code, including all methods, experimentation phases, and experimentation dataset, is available in our GitHub repository (GitHub repository: https://github.com/OnurVural/Data_Representation_Analysis_SWAN-SF, accessed on 16 March 2025).

4.1. Evaluation Metrics

To evaluate the performance of the mentioned strategies in terms of their effectiveness for the binary prediction task, we selected six performance evaluation metrics: accuracy, F1 score, receiver operating characteristic area under the curve (ROC AUC), Heidke skill score (HSS2), Gilbert skill score (GS), and true skill statistic (TSS). As SWAN-SF has a significant class imbalance between the positive class of F examples and the negative class of NF examples, accuracy alone, which simply reports the number of correctly predicted classes among all examples, is not sufficient for proper evaluation of models. Because of that, metrics such as ROC AUC, HSS2, GS, and TSS are used as effective evaluation metrics in flare prediction [2,3,7,10,17,19,40,41]. ROC AUC measures a binary classifier's ability to distinguish between positive and negative classes by evaluating its performance across all possible thresholds, reflecting how well it ranks positive samples higher than negative ones, and offering a more reliable metric than accuracy, especially in imbalanced datasets. HSS2 assesses the improvement over a random prediction scenario, and GS considers the likelihood of obtaining true positives by chance. Among all metrics, TSS shows the most effectiveness against extreme class imbalance, as it shows the difference between the true-positive rate and false-positive rate by taking values in the range of -1 and 1, with a score of 1 indicating all correct predictions, a score of -1 indicating all wrong predictions, and a score of 0 indicating random predictions. The work of Bobra et al. [3] suggested that TSS is the most effective measure for solar flare prediction. For a classification result of true positives (*TP*) as correctly classified F examples, true negatives (*TN*) as correctly classified NF examples, false positives (*FP*) as incorrectly classified NF examples, and false negatives (*FN*) as incorrectly classified *F* examples, the performance evaluation can be expressed as

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

$$F1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{6}$$

$$\text{HSS}_2 = \frac{2 \times [(TP \times TN) - (FN \times FP)]}{[(TP + FN) \times (FN + TN)] + [(TN + FP) \times (TP + FP)]} \tag{7}$$

$$\text{GS} = \frac{TP - C}{TP + FP + FN - C} \tag{8}$$

$$\text{where } C = \frac{(TP + FP) \times (TP + FN)}{TP + TN + FP + FN} \tag{9}$$

$$\text{TSS} = \frac{TP}{TP + FN} - \frac{FP}{TN + FP} \tag{10}$$

4.2. Preprocessing

We employed two types of preprocessing techniques in our study. First, we applied a traditional approach commonly utilized in other flare prediction research, which we implemented in both imbalanced and oversampled settings. Additionally, we incorporated an optimized preprocessing approach to address the challenges associated with the dataset for evaluating the differences in performance and effectiveness.

4.2.1. Traditional Preprocessing Approach

For the traditional approach, while preprocessing the partitions that will be used in the training stage, among NF examples, we only selected the examples belonging to the FQ category for the train sets. The reason behind this decision is the two other categories among NF examples, namely, B and C categories, showing notable magnetic field similarities to M and X major flare types. These similarities emerge because of shared traits in the fundamental patterns and behaviors of solar activity [1]. As a result, in the training phase of classifiers to distinguish between F and NF examples, the presence of categories B and C among NF examples introduces a challenge. The classifier may face difficulties in terms of discerning between genuine flare instances and those that show a resemblance to major flare types within the NF class. As a subsequent step, to avoid skewing the data representations, we eliminated each MVTS instance containing null values in any of their entries corresponding to selected magnetic field parameters. Moving forward in the process, we performed Z-score normalization on each MVTS instance $M^{(k)}$, ensuring that the magnetic field parameter values were standardized to a common scale. The normalization can be expressed as

$$m_n^{<t>} = \frac{m_n^{<t>} - \mu_n}{\sigma_n}, \tag{11}$$

In Equation (11), $m_n^{<t>}$ is the t-th timestamp of the n-th magnetic field parameter time series of the k-th MVTS instance $M^{(k)} \in \mathbb{R}^{\tau \times N}$, where $1 \leq n \leq N$, N being the number of magnetic field parameters, and $1 \leq t \leq \tau$, τ being the time series length. μ_n is the mean value of time series n, and σ_n is the standard deviation of time series n of instance k.

4.2.2. Optimized Preprocessing Approach

For the optimized approach, we followed multiple steps to address the chronic challenges present in SWAN-SF such as class imbalance, class overlap, a high percentage of missing values, varying scales, and varying skewness. The preprocessing pipeline began with fast Pearson correlation-based k-nearest neighbors (FPCKNN) imputation [42], which handled missing data by estimating values based on the similarity of instances. Following this, a combination of log normalization, square root normalization, Box–Cox normalization, Z-score normalization, and min–max normalization (LSBZM) were applied to standardize the dataset, ensuring that all features have a zero mean and unit variance, thereby addressing varying scales and skewness [43]. To mitigate class overlap issues, the elimination of minor flare category samples allowed for emphasis on more distinct class characteristics between F and NF classes. Next, TimeGAN, a generative adversarial network, generated synthetic time series data that retain the temporal dynamics of the original dataset [44]. The Tomek links method was then applied to clean the dataset by removing ambiguous instances that lie between classes, thereby clarifying class boundaries. Finally, random undersampling was employed to address class imbalance by randomly removing instances from majority classes, resulting in a more equitable distribution of classes [45].

4.3. Analysis of Created Graphs

Here, we analyze the Pearson correlation and Euclidean distance graphs to explore the relationships among univariate time series instances, each representing solar magnetic field parameters. For the created Pearson correlation graphs, Figure 10 displays the inter-parameter connections to provide a focused view into the structures of graph classes. Our first observation is that the F and NF graphs express different connections for magnetic field parameters as expected. For example, in the selected F-class graph, the TOTZ parameter

forms significantly more connections, resulting in a higher node degree compared to the selected NF-class graph, where the TOTZ parameter has relatively fewer connections with other magnetic field parameters, leading to a lower node degree. Figure 11 shows various randomly selected graphs to provide an overview.

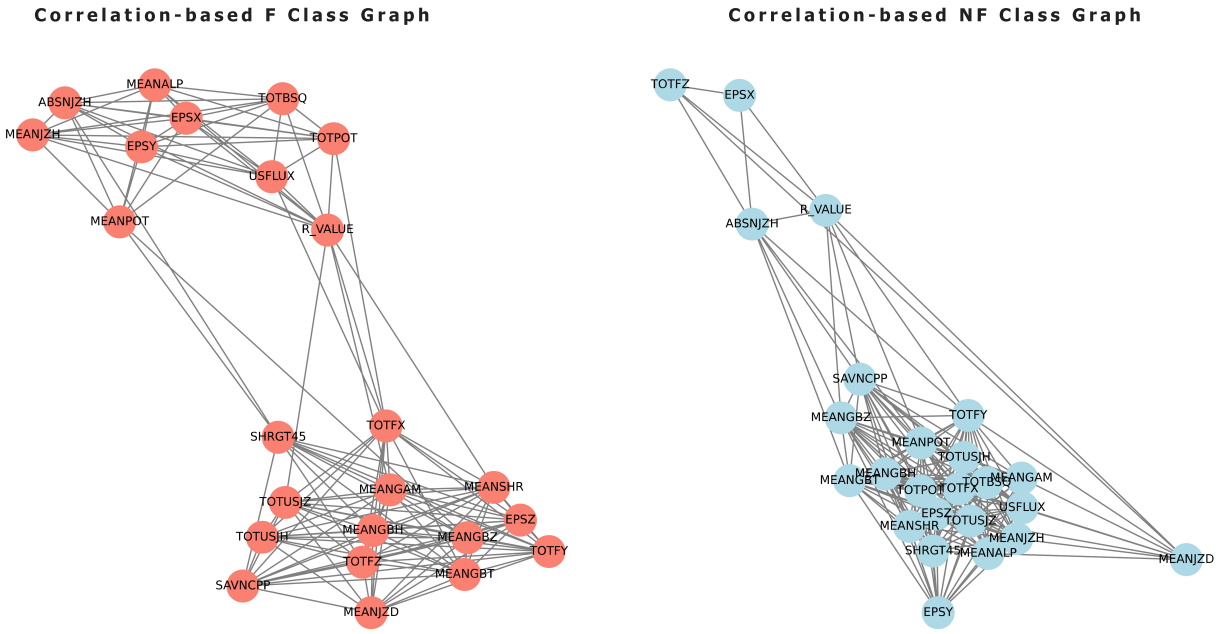


Figure 10. A view of the structure of randomly selected opposite-class correlation graphs created from Pearson correlation matrices, which illustrate the inter-parameter connections. The nodes have a one-to-one correspondence with the magnetic field parameters from Table 1. The F graph represents an X2.2 flare observed in NOAA active region 377, spanning 14 February 2011 from 02:00 to 13:48 UTC. The NF graph corresponds to an FQ event in NOAA active region 1038, with a timestamp from 9 November 2011 at 20:12 UTC to 10 November 2011 at 08:00 UTC.

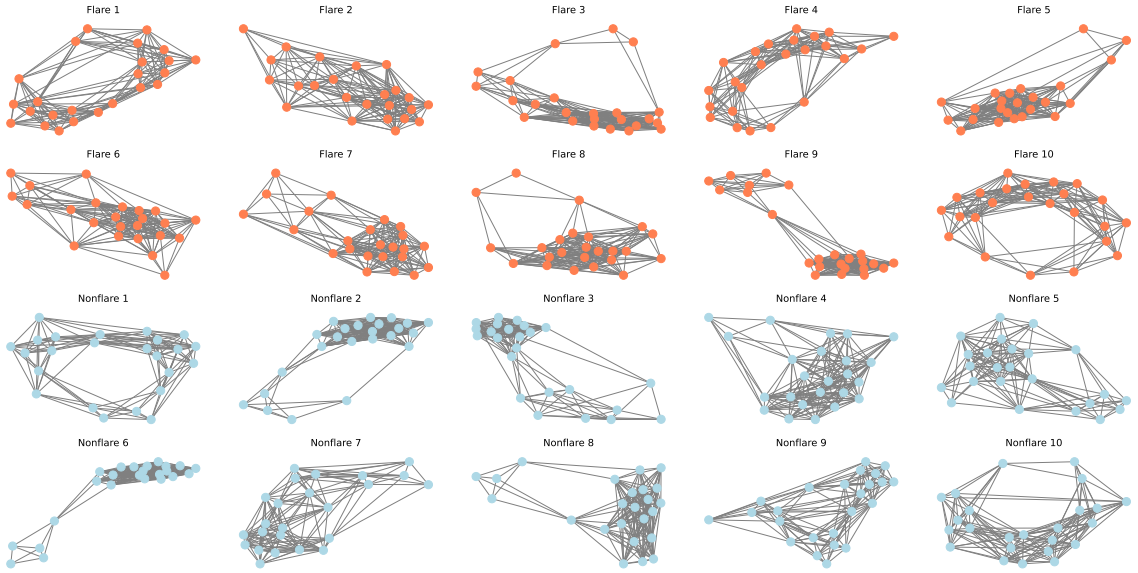


Figure 11. An overview of correlation-based F- and NF-class graphs randomly selected from P1 train set. The P1 train set covers events between 1 May 2010 and 13 March 2012. For F and NF graphs, different connection patterns are observed, highlighting varied inter-parameter relationships between events.

For Euclidean distance graphs, in the process of edge creation between nodes, we performed experiments to enforce that the created graph had the lowest possible edge density while still being connected (i.e., without having isolated nodes). The reason behind this choice was to enforce a high similarity condition between univariate time series instances as nodes get connected with edges. As seen in Figure 12, with the increase in threshold, the graph structure becomes highly connected where the average degree and graph density become high. This creates a challenge for the classifier models since with an increasing number of edges, the models will have difficulty distinguishing the most important interaction patterns. In contrast, with the decrease in threshold, the graph starts losing the fully connected property. Therefore, selecting the threshold enforces keeping the graph structure fully connected and keeping the edge density low to filter the most valuable structural information. Accordingly, the graph instances $G_E^{(k)}$ were created from the Euclidean distance matrices of each MVTS instance with a threshold t of 10 for imbalanced and undersampled settings and with a threshold t of 2 for optimized settings.

Figure 13 displays the difference in inter-parameter connections between F and NF graphs. The selected F-class graph exhibits a structure that contains two densely connected subgraphs bridged by a sparse connection. These two components represent clusters of parameters with high intra-cluster connectivity, reflecting strong correlations within each group. The presence of two such clusters suggests that the parameters within each group are strongly interdependent, while the interaction between the two clusters is relatively limited, leading to this topology. In contrast, the NF-class graph features a more uniformly connected structure, resembling a single connected component with fewer interactions overall. This suggests a more diffuse correlation pattern, where the parameters do not form distinct, highly connected groups but instead exhibit weaker, more distributed interactions across the entire graph. The reduced density of connections indicates that parameter interactions in the NF graph are more sparsely distributed, with fewer instances of strong correlations between specific parameters. Figure 14 shows various randomly selected graphs to provide an overview, which further confirms the presence of the mentioned structural patterns consistently observed in multiple F and NF graphs.

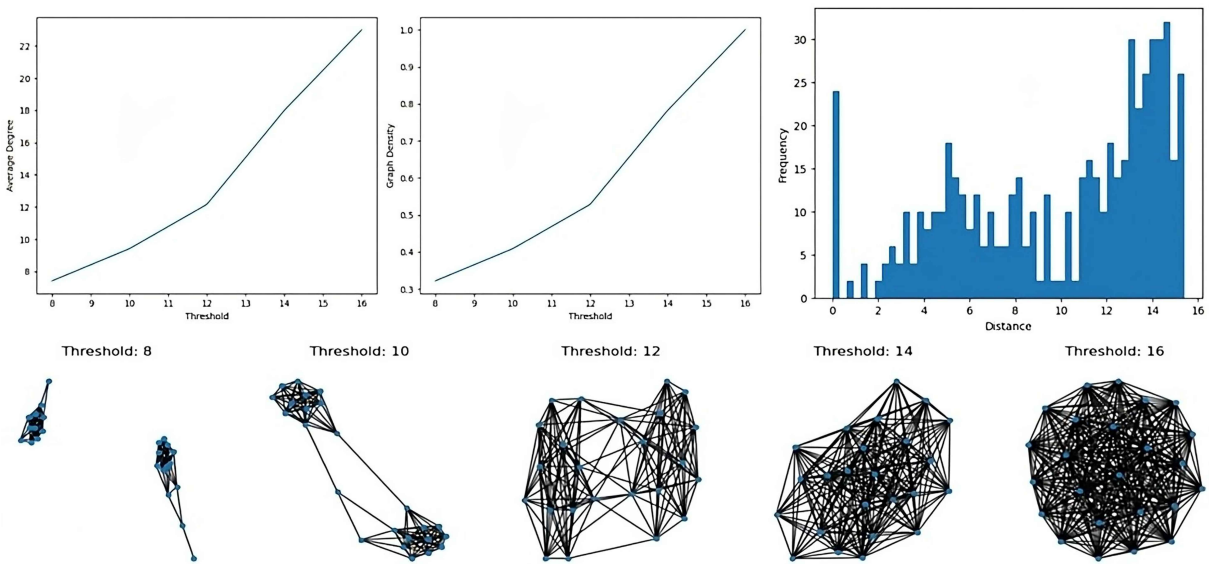


Figure 12. Experimentation of graph creation regarding different threshold values for Euclidian distance matrix randomly selected from P4 train set. Subfigures show the effect of changing the threshold on average degree, graph density, and graph structure. The figure includes the frequency distribution of the graph with respect to Euclidian distance values, providing a breakdown of the count of distance values that fall between certain intervals.

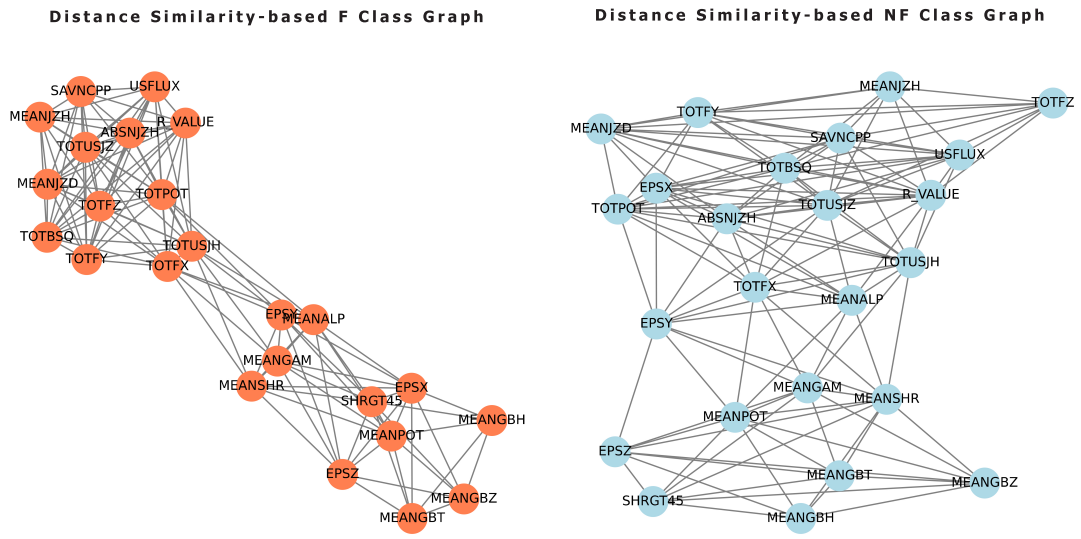


Figure 13. A view of the structure of randomly selected opposite-class distance similarity-based graphs created from thresholded Euclidian distance matrices, which illustrate the inter-parameter connections. The nodes have a one-to-one correspondence with the magnetic field parameters from Table 1. The F graph represents an X1.8 flare observed in NOAA active region 4920, spanning 19 December 2014 at 03:12 UTC to 19 December 2014 at 15:00 UTC. The NF graph corresponds to an FQ event in NOAA active region 4618, with a timestamp from 28 September 2014 at 00:48 UTC to 28 September 2014 at 12:36 UTC.

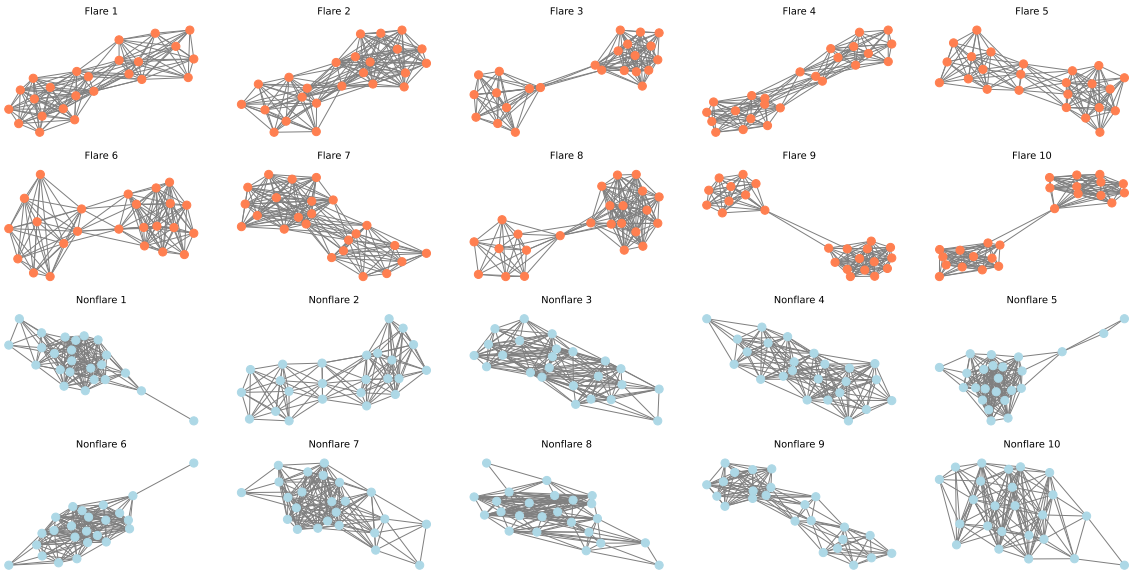


Figure 14. An overview of thresholded distance similarity-based F- and NF-class graphs randomly selected from P4 train set. The P4 train set covers events between 2 June 2014 and 18 March 2015. For F and NF graphs, different inter-parameter relationships are visible. The F-class graphs mostly show two tightly connected subgraphs with sparse connections between them, while the NF-class graphs have a more uniform, distributed connection pattern.

4.4. Training and Hyperparameter Settings

In this paper, we used MVTS instances that belong to consecutive partitions for training and testing purposes, respectively (e.g., training the models with MVTS instances of P1 and testing the models with MVTS instances of P2). We included all five partitions and thus had a total of four temporally consecutive train–test pairs (i.e., P1–P2, P2–P3, P3–P4,

and P4–P5). We used five selected downstream classifiers to train and test our vector-based models. The support vector machine (SVM) classifier was trained with a radial basis function (RBF) kernel, a regularization parameter C of 1.0, and a tolerance of 0.001 for the stopping criteria. The k-nearest neighbors (KNN) classifier was trained with k of five nearest neighbors, using uniform weights so that each neighbor contributed equally to the prediction and applying the Minkowski distance with p of 2, equivalent to Euclidean distance. The multilayer perceptron (MLP) classifier was trained with a hidden layer size of (100, 50) and a maximum iteration of 1000. This model used a ReLU activation function for the hidden layers and the Adam optimizer with a learning rate of 0.001 set to a constant schedule. The logistic regression (LR) classifier was trained with a penalty of l2 norm and maximum iterations of 100,000. Lastly, the decision tree (DT) classifier was trained using the Gini impurity criterion for splits with no maximum depth restriction, allowing it to grow until all leaves were pure or the minimum sample split of two was reached. For the VSTAT models, we repeated our experiments both without normalization (VSTAT) and with Z-score normalization (VSTATN) separately to generate more diverse results and assess the impact of normalization on model performance.

To train our time series-based models, we applied the following hyperparameter settings. The ST model was trained using a Rotation Forest [46] with three estimators, 100 shapelet samples, a maximum of 10 shapelets per class, and a batch size of 20. The TSF model was trained using 100 estimators and a minimum interval length of three. The ROCKET model was trained using 500 kernels. The RNN and LSTM models had an input size of 24 (corresponding to the number of time series features) and a hidden dimension of 64, followed by a fully connected layer with an output size of one for binary classification. They were trained for 10 epochs with binary cross-entropy loss, using the Adam optimizer with a learning rate of 10^{-3} (selected from a search space of [10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}]).

To train our graph-based models, we used the following settings. For the GND and GNE models, the same downstream classifiers—SVM, KNN, MLP, LR, and DT—were used with configurations identical to those in the vector-based method as alternative downstream module selections. In obtaining the embedding stage, the Laplacian embedding extraction process was configured with a dimension d of 14, which was chosen after our experiments for its ability to provide over 75% representation power for the graphs. Node2Vec embedding process was configured with a walk length of 10, p of 1, q of 1, and d of 32. For our GNN approach, the first layer had an input channel size of 60 (corresponding to MVTS timestamp length) and a hidden channel size of 64, the second layer had an input channel size of 64 and an output channel size of 32 followed by the fully connected layer. The module was trained for 20 epochs with binary cross-entropy loss, using the Adam optimizer with a learning rate of 10^{-2} (selected from a search space of [10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}]).

4.5. Solar Flare Prediction Performance

In the experimentation phase, we compare the solar flare prediction performance of our proposed strategies belonging to the three data representation methods discussed in Section 3 on the benchmark dataset under three different settings: under extreme class imbalance (i.e., as present in the dataset), under undersampling, and finally, under optimized preprocessing. The vector representation strategies include VLT and VSTAT, while the time series representation strategy is denoted as TSC, and graph representation consists of GND, GNE, and GNN. All models that belong to those strategies are trained and tested with data instances under consecutive partitions of the SWAN-SF dataset (i.e., P1–P2, P2–P3, P3–P4, and P4–P5), and their performance are evaluated with the performance evaluation metrics in Section 4.1. Each classifier model is designated by a codename structured as <ClassifierName>-<DataRepresentation>. The first component

of the codename corresponds to the classifier employed (e.g., SVM, KNN, TSF, LSTM, or GCN). The second component refers to the specific data representation utilized (e.g., VSTAT for statistical summary vector, CORLAP for correlation-based graph node embeddings via Laplacian Eigenmaps, or DSN2V for distance similarity-based graph node embeddings via Node2Vec). For convenient reference, a comprehensive list of classifiers and data representations is provided in Table 3.

Evaluation is presented in three views: (1) a summary of overall model performance, which provides a clear comparison across all metrics; (2) a visualization of performance distribution, showing which data representation method shows the most stable performance and less variability as a group; and (3) a detailed breakdown of results by partition, highlighting individual performance differences that are not captured in the overall summary.

Table 3. Overview of experimented classifiers and data representations.

Abbreviation	Full Name
Classifiers	
DT	Decision tree
GCN	Graph convolutional network
KNN	K-nearest neighbors
LR	Logistic regression
LSTM	Long short-term memory
MLP	Multilayer perceptron
RNN	Recurrent neural network
ROCKET	Random convolutional kernel transform
ST	Shapelet transform
SVM	Support vector machine
TSF	Time series forest
Data Representations	
COR	Correlation-based graph
CORGND	Correlation-based graph node degrees
CORLAP	Correlation-based graph node embeddings via Laplacian Eigenmaps
CORN2V	Correlation-based graph node embeddings via Node2Vec
DS	Distance similarity-based graph
DSGND	Distance similarity-based graph node degrees
DSLAP	Distance similarity-based graph node embeddings via Laplacian Eigenmaps
DSN2V	Distance similarity-based graph node embeddings via Node2Vec
TS	Time series
VLT	Vector of last timestamp
VSTAT	Vector of statistical summary
VSTATN	Vector of statistical summary normalized

4.5.1. Solar Flare Prediction Performance Under Extreme Class Imbalance

After the experiments under extreme class imbalance, the test results for all train and test partition pairs are averaged and presented in Table 4. Results documented in Table 4 show that, as expected, most methods tend to show high-accuracy results since class imbalance produces a tendency towards always predicting the negative class. The highest average accuracy performance is shared among various classifiers that are trained with different data representation methods, namely, SVM-CORLAP, LR-CORLAP, LR-DSLAP, LR-CORN2V, LR-DSN2V, LR-VLT, and LR-VSTAT. The high-accuracy results are not necessarily reflective of good performance. Therefore, in our experiments, we seek insight into other evaluation metrics that help to recognize the real performance of the classifiers trained with different data representations. *Accordingly, the results demonstrate that under crude conditions of extreme class imbalance and no application of sampling techniques, vector-based data representation is the most suitable method for solar flare prediction tasks.* The vector-based method achieves superior performance in four metrics, as SVM-VLT achieves

an average TSS of 0.6635 and an average ROC AUC score of 0.8318. SVM-VSTAT also shows a similarly strong performance in the same metrics by achieving an average TSS of 0.6621 and an average ROC AUC score of 0.831. LR-VSTATN is the best performer in the other two metrics, where it shows an average HSS2 of 0.3871 and an average F1 score of 0.3999. The highest average GS performance is produced by the sequence model time series representation method, namely, RNN, achieving a GS of 0.602.

After identifying the top performers across all selected performance metrics, we use Figure 15 to group the three methods and visualize the distribution of results to highlight the average performance across the methods. Upon examination of Figure 15, it is evident that the vector representation method consistently outperforms other methods overall, exhibiting higher median performance across TSS, HSS2, F1, GS, and ROC AUC metrics excluding accuracy. The time series representation method comes next in TSS, HSS2, F1, GS, and ROC AUC metrics and shows the highest median performance in accuracy, making the graph representation the least effective method in all five performance metrics, only being able to show the second highest median performance in accuracy. Observing the distribution, the graph representation method has the lowest overall variability, followed by time series and vector representation methods. These findings underscore the importance of considering both the central tendency and variability of performance metrics when evaluating the effectiveness of different data representation methods in solar flare prediction tasks.

Table 4. Performance results of classifier models for data representations of SWAN-SF across averaged train–test pairs with extreme class imbalance, where red fonts represent maximum values.

Models	Accuracy	TSS	HSS2	F1	GS	ROC AUC
VLT						
SVM-VLT	0.8776 ± 0.032	**0.6635 ± 0.0905**	0.1808 ± 0.0519	0.2098 ± 0.0591	0.01576 ± 0.0072	**0.8318 ± 0.0453**
KNN-VLT	0.9577 ± 0.0125	0.3347 ± 0.1041	0.2371 ± 0.062	0.2569 ± 0.0681	0.0076 ± 0.0045	0.6674 ± 0.052
MLP-VLT	0.9713 ± 0.0153	0.0818 ± 0.1368	0.0666 ± 0.0856	0.0745 ± 0.0949	0.0017 ± 0.0029	0.5409 ± 0.0684
LR-VLT	**0.9791 ± 0.009**	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.5 ± 0.0
DT-VLT	0.9529 ± 0.0154	0.5134 ± 0.1259	0.3012 ± 0.0607	0.3216 ± 0.0615	0.011 ± 0.0057	0.7567 ± 0.063
VSTAT						
SVM-VSTAT	0.891 ± 0.0269	0.6621 ± 0.0622	0.1977 ± 0.0561	0.2255 ± 0.0627	0.0155 ± 0.0071	0.831 ± 0.0311
KNN-VSTAT	0.9581 ± 0.0131	0.3258 ± 0.0844	0.2361 ± 0.0564	0.2558 ± 0.0626	0.0074 ± 0.0045	0.6629 ± 0.0422
MLP-VSTAT	0.9356 ± 0.0381	0.2779 ± 0.2138	0.1315 ± 0.0482	0.1536 ± 0.057	0.0061 ± 0.0049	0.639 ± 0.1069
LR-VSTAT	**0.9791 ± 0.009**	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.5 ± 0.0
DT-VSTAT	0.9537 ± 0.0139	0.5575 ± 0.0195	0.3241 ± 0.028	0.3438 ± 0.035	0.0121 ± 0.0055	0.7788 ± 0.0097
SVM-VSTATN	0.9654 ± 0.013	0.3102 ± 0.107	0.2603 ± 0.0377	0.276 ± 0.0373	0.0063 ± 0.0026	0.6551 ± 0.0535
KNN-VSTATN	0.9704 ± 0.0071	0.3942 ± 0.1495	0.3327 ± 0.0412	0.3465 ± 0.042	0.0077 ± 0.0028	0.697 ± 0.0748
MLP-VSTATN	0.9681 ± 0.0086	0.3903 ± 0.165	0.3127 ± 0.0335	0.327 ± 0.0365	0.0077 ± 0.003	0.6952 ± 0.0825
LR-VSTATN	0.9714 ± 0.0051	0.4927 ± 0.1821	**0.3871 ± 0.0753**	**0.3999 ± 0.0746**	0.0093 ± 0.0021	0.7464 ± 0.0911
DT-VSTATN	0.9536 ± 0.0144	0.4171 ± 0.1143	0.2603 ± 0.0597	0.2797 ± 0.0584	0.0083 ± 0.0017	0.7085 ± 0.0571
TSC						
ST-TS	0.9693 ± 0.0164	0.0246 ± 0.0226	0.028 ± 0.0237	0.0398 ± 0.0342	0.0006 ± 0.0007	0.5124 ± 0.0111
TSF-TS	0.9678 ± 0.0121	0.3718 ± 0.1215	0.3133 ± 0.0765	0.3293 ± 0.0814	0.0085 ± 0.0057	0.6859 ± 0.0608
ROCKET-TS	0.9782 ± 0.0091	0.0379 ± 0.0295	0.0654 ± 0.0491	0.0698 ± 0.0501	0.0009 ± 0.0007	0.5189 ± 0.0147
LSTM-TS	0.9772 ± 0.0091	0.0533 ± 0.0673	0.0794 ± 0.0921	0.0853 ± 0.0944	0.0011 ± 0.0014	0.5267 ± 0.0337
RNN-TS	0.9788 ± 0.0092	0.008 ± 0.0116	0.0147 ± 0.0209	0.016 ± 0.0228	**0.602 ± 0.8033**	0.504 ± 0.0058
GND						
SVM-CORGND	0.8429 ± 0.0263	0.2362 ± 0.0531	0.057 ± 0.0214	0.0909 ± 0.0328	0.0058 ± 0.0027	0.6181 ± 0.0266
KNN-CORGND	0.9726 ± 0.0086	0.0293 ± 0.0088	0.041 ± 0.0113	0.0515 ± 0.0133	0.0006 ± 0.0003	0.5147 ± 0.0044
MLP-CORGND	0.9737 ± 0.0077	0.0251 ± 0.0193	0.0328 ± 0.0229	0.0409 ± 0.0282	0.5741 ± 1.1468	0.5126 ± 0.0097
LR-CORGND	0.9688 ± 0.0212	0.0359 ± 0.0701	0.0227 ± 0.0421	0.0298 ± 0.0548	0.0008 ± 0.0015	0.518 ± 0.035
DT-CORGND	0.9459 ± 0.0121	0.0562 ± 0.0142	0.0392 ± 0.009	0.0636 ± 0.0148	0.0012 ± 0.0005	0.5281 ± 0.0071
SVM-DSGND	0.8286 ± 0.024	0.2959 ± 0.027	0.0646 ± 0.0204	0.0989 ± 0.0324	0.0074 ± 0.0031	0.648 ± 0.0135
KNN-DSGND	0.9708 ± 0.0089	0.0287 ± 0.0044	0.0385 ± 0.0105	0.0507 ± 0.0092	0.0006 ± 0.0003	0.5144 ± 0.0022
MLP-DSGND	0.9585 ± 0.0101	0.0621 ± 0.0231	0.0525 ± 0.006	0.0711 ± 0.0102	0.0012 ± 0.0004	0.5311 ± 0.0116
LR-DSGND	0.9781 ± 0.0089	0.0128 ± 0.0123	0.0229 ± 0.0214	0.0256 ± 0.0223	0.0002 ± 0.0003	0.5064 ± 0.0062
DT-DSGND	0.9464 ± 0.0083	0.0794 ± 0.0255	0.0538 ± 0.0138	0.0782 ± 0.0185	0.0017 ± 0.0007	0.5397 ± 0.0128
GNE						
SVM-CORLAP	**0.9791 ± 0.009**	0.0003 ± 0.0006	0.0006 ± 0.0011	0.0006 ± 0.0011	0.0 ± 0.0	0.5002 ± 0.0003
KNN-CORLAP	0.9781 ± 0.0089	0.0027 ± 0.0053	0.0046 ± 0.0089	0.0066 ± 0.0113	0.0001 ± 0.0001	0.5014 ± 0.0026

Table 4. *Cont.*

Models	Accuracy	TSS	HSS2	F1	GS	ROC AUC
MLP-CORLAP	0.9473 ± 0.0515	0.034 ± 0.0568	0.0134 ± 0.0116	0.0225 ± 0.0255	0.0045 ± 0.0075	0.517 ± 0.0284
LR-CORLAP	**0.9791 ± 0.009**	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.5 ± 0.0
DT-CORLAP	0.9549 ± 0.021	0.0458 ± 0.0332	0.0331 ± 0.0229	0.0518 ± 0.0367	0.001 ± 0.0008	0.5229 ± 0.0167
SVM-DSLAP	0.4206 ± 0.0209	0.4078 ± 0.0223	0.0278 ± 0.0113	0.0667 ± 0.0269	0.0204 ± 0.0086	0.7039 ± 0.0111
KNN-DSLAP	0.9783 ± 0.0091	0.0006 ± 0.0011	0.001 ± 0.002	0.0026 ± 0.0038	−1.2425 ± 2.485	0.5003 ± 0.0005
MLP-DSLAP	0.9516 ± 0.0412	0.0191 ± 0.0281	0.0101 ± 0.0056	0.0236 ± 0.0146	0.0003 ± 0.0005	0.5095 ± 0.0141
LR-DSLAP	**0.9791 ± 0.0091**	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.5 ± 0.0
DT-DSLAP	0.9425 ± 0.0142	0.0242 ± 0.0088	0.0154 ± 0.002	0.0409 ± 0.005	0.0005 ± 0.0001	0.5121 ± 0.0044
SVM-CORN2V	0.5856 ± 0.0415	0.0955 ± 0.0167	0.0105 ± 0.0033	0.0549 ± 0.0183	0.0037 ± 0.0007	0.5478 ± 0.0083
KNN-CORN2V	0.9789 ± 0.0089	−0.9586 ± 1.9167	−0.0004 ± 0.0005	0.0 ± 0.0	−0.0 ± 0.0	0.4999 ± 0.0002
MLP-CORN2V	0.9516 ± 0.0412	0.0191 ± 0.0281	0.0101 ± 0.0056	0.0236 ± 0.0146	0.0003 ± 0.0005	0.5095 ± 0.0141
LR-CORN2V	**0.9791 ± 0.009**	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.5 ± 0.0
DT-CORN2V	0.9554 ± 0.0214	0.0347 ± 0.0271	0.0246 ± 0.017	0.0434 ± 0.0295	−1.1866 ± 2.3752	0.5174 ± 0.0135
SVM-DSN2V	0.545 ± 0.0123	0.2909 ± 0.0274	0.0253 ± 0.0101	0.0639 ± 0.0254	0.0112 ± 0.0051	0.6454 ± 0.0137
KNN-DSN2V	0.9788 ± 0.009	0.0014 ± 0.0025	0.0026 ± 0.0046	0.0034 ± 0.0054	−0.4885 ± 0.9772	0.5007 ± 0.0012
MLP-DSN2V	0.9706 ± 0.0141	0.0012 ± 0.0024	0.0007 ± 0.0013	0.0055 ± 0.011	0.0 ± 0.0	0.5006 ± 0.0012
LR-DSN2V	**0.9791 ± 0.009**	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.5 ± 0.0
DT-DSN2V	0.9438 ± 0.0135	0.0053 ± 0.006	−2.4172 ± 4.8432	0.0284 ± 0.0083	−0.5559 ± 1.112	0.5027 ± 0.003
GNN						
GCN-COR	0.9749 ± 0.0154	0.0061 ± 0.014	0.0077 ± 0.0188	0.0144 ± 0.0287	0.0002 ± 0.0005	0.5032 ± 0.007
GCN-DS	0.9602 ± 0.0337	−0.0143 ± 0.0269	−0.0062 ± 0.0094	0.0018 ± 0.0035	−0.0003 ± 0.0005	0.4928 ± 0.0135

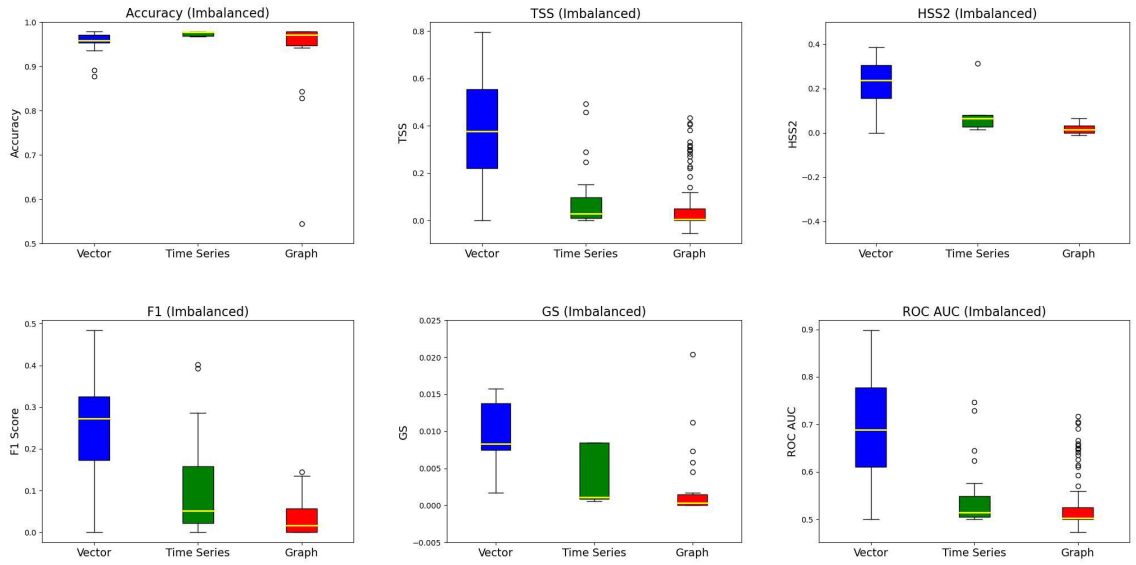


Figure 15. Demonstration of accuracy, TSS, HSS2, F1, GS, and ROC AUC performance results of vector (blue), time series (green), and graph (red) data representation methods in boxplot format under the extreme class imbalance setting. For each method, the classification results of all submethods are aggregated to illustrate the overall performance of the group.

After investigating the distributional characteristics of the results and showcasing the superior performance of the vector-based method, our analysis proceeds with a direct focus on the TSS, the most resilient metric against class imbalance. This examination aims to showcase the TSS performances of each model in each train–test partition pair. Accordingly, in Figure 16, when we concentrate on the best performers (Table 5), for the train–test pair of P1–P2, the top five TSS results are achieved by SVM-VLT, SVM-VSTAT, DT-VSTAT, DT-VLT, and LR-VSTATN, with TSS results of 0.6027, 0.5998, 0.5657, 0.5431, and 0.5199, respectively. For the train–test pair of P2–P3, the top five TSS results are achieved by SVM-VSTAT, SVM-VLT, DT-VSTAT, DT-VLT, and TSF-TS with values of 0.6661, 0.6488, 0.5671, 0.5534, and 0.4578, respectively. For the train–test pair of P3–P4, the top five TSS results are achieved by SVM-VLT, SVM-VSTAT, LR-VSTATN, DT-VSTAT, and KNN-VSTATN with values of

0.7957, 0.7461, 0.6037, 0.5688 and 0.5564, respectively. For the train–test pair of P4–P5, the top five TSS results are achieved by SVM-VSTAT, DT-VLT, LR-VSTATN, SVM-VLT, and DT-VSTAT with values of 0.6365, 0.6246, 0.6197, 0.6069, and 0.5268, respectively. Based on the findings, it can be inferred that SVM-VSTAT and SVM-VLT consistently compete to achieve top performance in attaining high TSS scores.

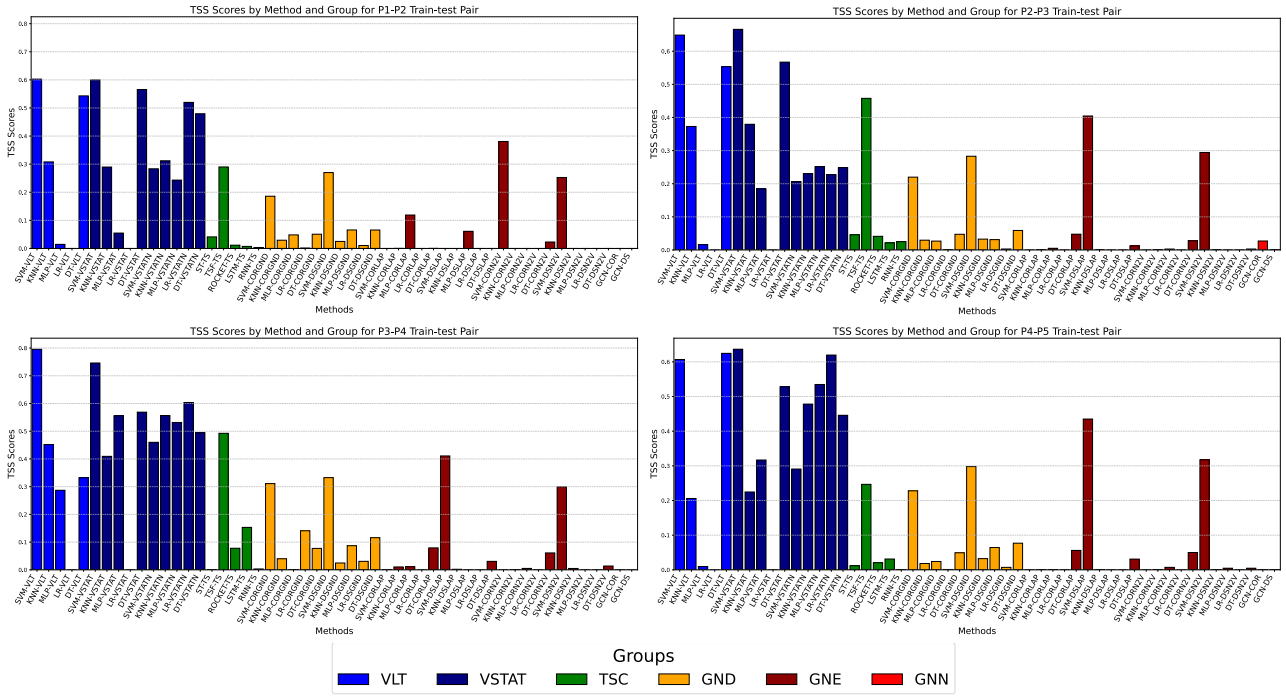


Figure 16. The TSS results for each model, based on the given data representation methods, are demonstrated using all train–test partition pairs under the imbalanced settings.

Table 5. Top 5 models and TSS comparison for different conditions and partitions, where red fonts represent maximum values.

Condition	P1–P2		P2–P3		P3–P4		P4–P5	
	Model	**TSS**	**Model**	**TSS**	**Model**	**TSS**	**Model**	**TSS**
Imbalanced	SVM-VLT	**0.6027**	SVM-VSTAT	**0.6661**	SVM-VLT	**0.7957**	SVM-VSTAT	**0.6365**
	SVM-VSTAT	0.5998	SVM-VLT	0.6488	SVM-VSTAT	0.7461	DT-VLT	0.6246
	DT-VSTAT	0.5657	DT-VSTAT	0.5671	LR-VSTATN	0.6037	LR-VSTATN	0.6197
	DT-VLT	0.5431	DT-VLT	0.5534	DT-VSTAT	0.5688	SVM-VLT	0.6069
	LR-VSTATN	0.5199	TSF-TS	0.4578	KNN-VSTATN	0.5564	DT-VSTAT	0.5268
Undersampled	TSF-TS	**0.6965**	DT-VSTAT	**0.7526**	SVM-VLT	**0.7885**	TSF-TS	**0.7891**
	ROCKET-TS	0.6154	SVM-VSTAT	0.7404	SVM-VSTAT	0.7826	SVM-VLT	0.7857
	SVM-VLT	0.6027	SVM-VLT	0.7375	TSF-TS	0.7564	SVM-VSTAT	0.7762
	SVM-VSTAT	0.5998	DT-VLT	0.7348	DT-VSTAT	0.7465	DT-VSTAT	0.7174
	DT-VSTAT	0.5657	KNN-VSTATN	0.7082	ROCKET-TS	0.6979	LR-VSTATN	0.6974
Optimized	RNN-TS	**0.8036**	TSF-TS	**0.768**	LR-VLT	**0.8094**	TSF-TS	**0.7645**
	LSTM-TS	0.8015	LR-VSTAT	0.7445	RNN-TS	0.7648	LR-VSTAT	0.7278
	LR-VSTAT	0.7819	RNN-TS	0.6899	LR-VSTAT	0.7117	RNN-TS	0.6873
	KNN-VLT	0.7585	KNN-VLT	0.6628	TSF-TS	0.6875	ST-TS	0.6464
	LR-VLT	0.7061	ST-TS	0.6551	LSTM-TS	0.6671	KNN-VLT	0.6434

In the culmination of our experiments, the vector-based representation method consistently emerges as the optimal choice for addressing the challenges posed by the extreme class imbalance inherent in the SWAN-SF dataset. Within this category, both VLT and VSTAT representations exhibit noteworthy performance. The SVM-VLT and SVM-VSTAT representations consis-

tently demonstrate robust representation capabilities, positioning them prominently within the dataset analysis. *These findings affirm not only the efficacy of the vector-based representation methods but also the prowess of SVM as the preferred downstream classifier in our experimental context*. This collective evidence emphasizes the resilience and generalizability of the selected representation methods across diverse train–test partition pairs. The comparison between SVM-VLT and SVM-VSTAT, spanning various partition pairs, underscores their versatility and effectiveness in capturing meaningful patterns within the dataset.

4.5.2. Solar Flare Prediction Performance Under Undersampling

In the next phase of our experimental evaluation, we utilize the balanced SWAN-SF dataset, where class imbalance issues are already addressed by undersampling the majority class instances to a 1:1 ratio. For each train partition, we randomly select MVTS instances within the NF group corresponding to the same number of instances as the F group. The sizes of the undersampled training sets are presented in Table 6. Table 7 presents the averaged test results of all train and test partition pairs after conducting the experiments under undersampling. The first observation from the table indicates that, overall, the implementation of undersampling techniques results in improved performance across the evaluation metrics. Moreover, TSC models demonstrated a significant leap in performance, with all of them performing markedly better than their counterparts. This suggests that by reducing the size of the majority class, we enhance the model's ability to focus on the underrepresented minority class. *This finding underscores the importance of addressing class imbalance in solar flare tasks, as undersampling can help mitigate the risks of bias and overfitting, ultimately contributing to more robust and reliable model predictions*. Regarding the results, the vector method attains the top performance in three metrics, the time series method attains the top performance in three metrics, and the graph method attains the top performance in three metrics. The highest average accuracy of 0.979 is achieved by LR-VLT and LR-VSTAT, clearly shown to be a result of overfitting. The highest average TSS score of 0.739 is achieved by TSF-TS, closely followed by SVM-VSTAT and SVM-VLT, which attained average TSS scores of 0.7286 and 0.7248, respectively. In both HSS2 and F1, DT-VSTAT stands out with the highest average performance, achieving scores of 0.2652 and 0.2899, respectively. In GS, SVM-CORN2V achieves the highest average performance with a score of 0.0221. Finally, in terms of ROC AUC, TSF-TS leads with a score of 0.8695, while SVM-VLT and SVM-VSTAT follow with scores of 0.8644 and 0.8624, respectively.

Table 6. Class sizes before and after undersampling for each training partition.

Partition	Before Undersampling		After Undersampling	
	F	NF	F	NF
P1	1180	56,319	1180	1180
P2	1285	65,364	1285	1285
P3	1277	30,766	1277	1277
P4	890	36,667	890	890

Table 7. Performance results of classifier models for data representations of SWAN-SF across averaged train–test pairs with undersampling, where red fonts represent maximum values.

Models	Accuracy	TSS	HSS2	F1	GS	ROC AUC
VLT						
SVM-VLT	0.8408 ± 0.0299	0.7286 ± 0.0871	0.1531 ± 0.0338	0.1843 ± 0.0444	0.0181 ± 0.0088	0.8644 ± 0.0436
KNN-VLT	0.8989 ± 0.0526	0.5712 ± 0.1759	0.1936 ± 0.0282	0.221 ± 0.034	0.0138 ± 0.0087	0.7856 ± 0.0879
MLP-VLT	0.7268 ± 0.2079	0.2906 ± 0.1529	0.065 ± 0.0379	0.0981 ± 0.0431	0.0099 ± 0.0083	0.6453 ± 0.0764
LR-VLT	**0.979 ± 0.009**	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.5 ± 0.0
DT-VLT	0.9172 ± 0.0385	0.6162 ± 0.0844	0.2406 ± 0.0528	0.2662 ± 0.0553	0.0146 ± 0.009	0.8081 ± 0.0422

Table 7. *Cont.*

Models	Accuracy	TSS	HSS2	F1	GS	ROC AUC
VSTAT						
SVM-VSTAT	0.8452 ± 0.0389	0.7248 ± 0.0853	0.1566 ± 0.028	0.1877 ± 0.0392	0.018 ± 0.009	0.8624 ± 0.0427
KNN-VSTAT	0.8995 ± 0.0529	0.5652 ± 0.185	0.1924 ± 0.0273	0.2198 ± 0.0325	0.0136 ± 0.0084	0.7826 ± 0.0925
MLP-VSTAT	0.7214 ± 0.1843	0.3985 ± 0.1131	0.0635 ± 0.0181	0.0994 ± 0.0217	0.0132 ± 0.0089	0.6993 ± 0.0565
LR-VSTAT	**0.979 ± 0.009**	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.5 ± 0.0
DT-VSTAT	0.9185 ± 0.0389	0.6956 ± 0.0879	**0.2652 ± 0.0353**	**0.2899 ± 0.0369**	0.0161 ± 0.0088	0.8478 ± 0.044
SVM-VSTATN	0.7527 ± 0.1512	0.3067 ± 0.1615	0.111 ± 0.1321	0.1431 ± 0.1229	0.0103 ± 0.0104	0.6534 ± 0.0808
KNN-VSTATN	0.761 ± 0.1469	0.5866 ± 0.1833	0.1489 ± 0.1266	0.1797 ± 0.1175	0.0177 ± 0.0116	0.7932 ± 0.0916
MLP-VSTATN	0.7666 ± 0.1427	0.5872 ± 0.2023	0.1463 ± 0.1177	0.177 ± 0.1083	0.0176 ± 0.0118	0.7936 ± 0.1011
LR-VSTATN	0.7491 ± 0.1539	0.6295 ± 0.0746	0.1675 ± 0.1769	0.199 ± 0.1653	0.0186 ± 0.0105	0.8148 ± 0.0373
DT-VSTATN	0.787 ± 0.1128	0.4175 ± 0.2439	0.0949 ± 0.0742	0.129 ± 0.075	0.0135 ± 0.0128	0.7087 ± 0.1219
TSC						
ST-TS	0.8585 ± 0.0244	0.3033 ± 0.0601	0.0786 ± 0.025	0.1114 ± 0.0356	0.0072 ± 0.0029	0.6517 ± 0.0301
TSF-TS	0.8912 ± 0.0269	**0.739 ± 0.0418**	0.2146 ± 0.0452	0.2423 ± 0.0509	0.017 ± 0.0071	**0.8695 ± 0.0209**
ROCKET-TS	0.7978 ± 0.0414	0.6182 ± 0.0566	0.1096 ± 0.0306	0.1432 ± 0.0413	0.0161 ± 0.0071	0.8091 ± 0.0283
LSTM-TS	0.6979 ± 0.0377	0.3204 ± 0.0498	0.0417 ± 0.0129	0.0786 ± 0.0265	0.0095 ± 0.004	0.6602 ± 0.0249
RNN-TS	0.726 ± 0.0974	0.3126 ± 0.1079	0.0647 ± 0.0424	0.1008 ± 0.0388	0.0091 ± 0.0045	0.6692 ± 0.0673
GND						
SVM-CORGND	0.708 ± 0.0194	0.3564 ± 0.0467	0.0385 ± 0.0102	0.0685 ± 0.0144	0.0084 ± 0.0026	0.6782 ± 0.0234
KNN-CORGND	0.695 ± 0.0189	0.2486 ± 0.0401	0.0261 ± 0.0076	0.0566 ± 0.0114	0.0059 ± 0.0015	0.6243 ± 0.0201
MLP-CORGND	0.7496 ± 0.0509	0.2618 ± 0.0425	0.035 ± 0.0158	0.0648 ± 0.0185	0.0057 ± 0.0013	0.6309 ± 0.0213
LR-CORGND	0.6507 ± 0.036	0.2804 ± 0.0618	0.0259 ± 0.0093	0.0566 ± 0.0136	0.0073 ± 0.0031	0.6402 ± 0.0309
DT-CORGND	0.6794 ± 0.0142	0.1845 ± 0.0479	0.0186 ± 0.0072	0.0495 ± 0.012	0.0046 ± 0.0022	0.5922 ± 0.024
SVM-DSGND	0.682 ± 0.0196	0.4282 ± 0.0275	0.042 ± 0.0081	0.0721 ± 0.0127	0.0104 ± 0.0027	0.7141 ± 0.0138
KNN-DSGND	0.6543 ± 0.0222	0.359 ± 0.0212	0.0326 ± 0.005	0.0631 ± 0.0093	0.009 ± 0.0014	0.6795 ± 0.0106
MLP-DSGND	0.7597 ± 0.0186	0.3324 ± 0.0146	0.0437 ± 0.0119	0.0732 ± 0.0159	0.0071 ± 0.0012	0.6662 ± 0.0073
LR-DSGND	0.6681 ± 0.0341	0.3865 ± 0.0587	0.0369 ± 0.0105	0.0672 ± 0.0148	0.0097 ± 0.0034	0.6932 ± 0.0294
DT-DSGND	0.7133 ± 0.0162	0.3357 ± 0.0264	0.0371 ± 0.0097	0.0672 ± 0.0141	0.0078 ± 0.0019	0.6678 ± 0.0132
GNE						
SVM-CORLAP	0.5379 ± 0.0157	0.0453 ± 0.0181	0.0043 ± 0.0029	0.0438 ± 0.0188	0.0018 ± 0.0012	0.5227 ± 0.009
KNN-CORLAP	0.436 ± 0.0104	0.1315 ± 0.0211	0.0097 ± 0.0052	0.0494 ± 0.0214	0.0067 ± 0.004	0.5658 ± 0.0106
MLP-CORLAP	0.536 ± 0.0109	0.0635 ± 0.0135	0.0059 ± 0.0038	0.0454 ± 0.0198	0.0026 ± 0.0017	0.5317 ± 0.0068
LR-CORLAP	0.4953 ± 0.0065	−0.0068 ± 0.0148	−0.0007 ± 0.001	0.0393 ± 0.0159	−0.0004 ± 0.0005	0.4966 ± 0.0074
DT-CORLAP	0.6701 ± 0.0271	0.3053 ± 0.0331	0.0358 ± 0.0127	0.0733 ± 0.0268	0.0093 ± 0.0037	0.6526 ± 0.0165
SVM-DSLAP	0.554 ± 0.0211	0.2668 ± 0.0287	0.024 ± 0.0106	0.0627 ± 0.0257	0.0101 ± 0.0044	0.6334 ± 0.0143
KNN-DSLAP	0.7028 ± 0.0172	0.264 ± 0.0161	0.0347 ± 0.0133	0.072 ± 0.0275	0.0078 ± 0.0036	0.632 ± 0.0081
MLP-DSLAP	0.6021 ± 0.0142	0.2306 ± 0.0397	0.0234 ± 0.0105	0.0618 ± 0.0251	0.0079 ± 0.0033	0.6153 ± 0.0199
LR-DSLAP	0.5014 ± 0.0054	0.0111 ± 0.002	0.0009 ± 0.0004	0.0404 ± 0.0159	−1.1157 ± 2.2321	0.5056 ± 0.001
DT-DSLAP	0.6532 ± 0.0086	0.299 ± 0.0156	0.0327 ± 0.0106	0.0709 ± 0.0261	0.0092 ± 0.0034	0.6495 ± 0.0078
SVM-CORN2V	0.4156 ± 0.0156	0.4018 ± 0.0143	0.0296 ± 0.0103	0.0717 ± 0.0234	**0.0221 ± 0.0073**	0.7009 ± 0.0071
KNN-CORN2V	0.5522 ± 0.0115	0.327 ± 0.0185	0.0309 ± 0.0085	0.0724 ± 0.0214	0.0131 ± 0.0035	0.6635 ± 0.0092
MLP-CORN2V	0.5989 ± 0.0577	0.3135 ± 0.0232	0.0301 ± 0.0097	0.0712 ± 0.0217	0.013 ± 0.0062	0.6567 ± 0.0116
LR-CORN2V	0.4238 ± 0.0064	0.3996 ± 0.0197	0.0298 ± 0.0102	0.0719 ± 0.0234	0.0216 ± 0.0076	0.6998 ± 0.0099
DT-CORN2V	0.6237 ± 0.0088	0.2602 ± 0.0023	0.0297 ± 0.01	0.0708 ± 0.0226	0.0093 ± 0.0029	0.6301 ± 0.0012
SVM-DSN2V	0.5174 ± 0.0017	0.2804 ± 0.0176	0.0249 ± 0.0078	0.0669 ± 0.0208	0.0122 ± 0.004	0.6402 ± 0.0088
KNN-DSN2V	0.5488 ± 0.0057	0.1326 ± 0.0131	0.0128 ± 0.0046	0.0551 ± 0.0177	0.0055 ± 0.0021	0.5663 ± 0.0066
MLP-DSN2V	0.4824 ± 0.0259	0.266 ± 0.0178	0.0221 ± 0.0072	0.0643 ± 0.0204	0.0127 ± 0.0052	0.633 ± 0.0089
LR-DSN2V	0.5835 ± 0.0042	0.27 ± 0.0029	0.0277 ± 0.0085	0.0692 ± 0.0213	0.0104 ± 0.0034	0.635 ± 0.0014
DT-DSN2V	0.5739 ± 0.0338	0.1624 ± 0.0677	0.0167 ± 0.0085	0.0548 ± 0.0185	0.0051 ± 0.0022	0.5624 ± 0.0073
GNN						
GCN-COR	0.6115 ± 0.2517	0.0878 ± 0.1267	0.0092 ± 0.0112	0.0445 ± 0.0204	0.0097 ± 0.0061	0.5197 ± 0.0226
GCN-DS	0.5347 ± 0.3828	0.0852 ± 0.1282	0.0219 ± 0.0184	0.0314 ± 0.0093	0.0052 ± 0.0073	0.5094 ± 0.0095

Figure 17 shows the result distribution after identifying the top performers to emphasize the average performance across the methods. Similar to the extreme class imbalance setting, the vector representation method continues to be the top performer, demonstrating higher median performance across all six metrics, followed by the time series representation method, while the graph representation method is identified as the least effective. Considering the variability in performance distribution, the time series representation method shows a broader performance range, with the best models making significant progress compared to previous settings. However, vector-based methods remain superior overall.

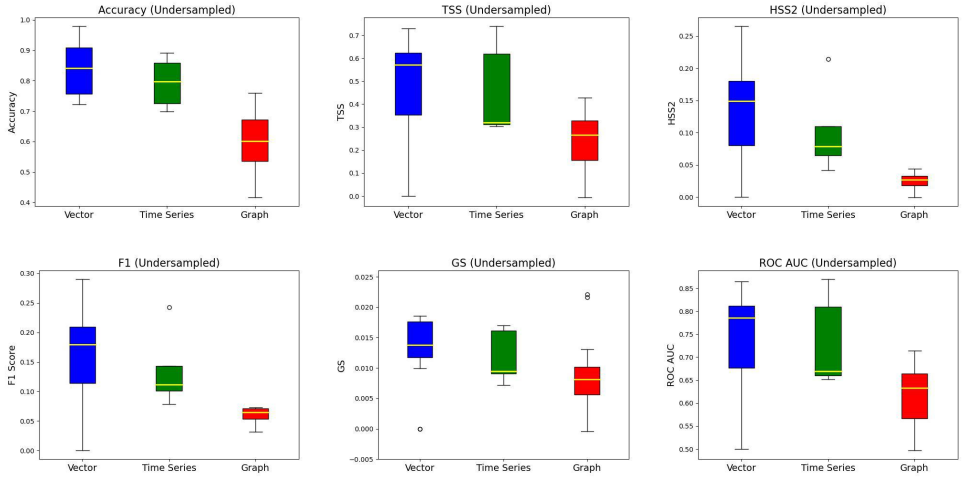


Figure 17. Demonstration of accuracy, TSS, HSS2, F1, GS, and ROC AUC performance results of vector (blue), time series (green), and graph (red) data representation methods in boxplot format under undersampling setting. For each method, the classification results of all submethods are aggregated to illustrate the overall performance of the group.

Figure 18 displays the results after comparing the TSS performances of each model in each train–test partition pair. Focusing on the best performers (Table 5), for the train–test pair of P1–P2, the top five TSS results are achieved by TSF-TS, ROCKET-TS, SVM-VLT, SVM-VSTAT, and DT-VSTAT, with TSS values of 0.6965, 0.6154, 0.6027, 0.5998, and 0.5657, respectively. For the train–test pair of P2–P3, the top five TSS results are achieved by DT-VSTAT, SVM-VSTAT, SVM-VLT, DT-VLT, and KNN-VSTATN with TSS values of 0.7526, 0.7404, 0.7375, 0.7348, and 0.7082, respectively. For the train–test pair of P3–P4, the top five TSS results are achieved by SVM-VLT, SVM-VSTAT, TSF-TS, DT-VSTAT, and ROCKET-TS with TSS values of 0.7885, 0.7826, 0.7564, 0.7465, and 0.6979, respectively. For the train–test pair of P4–P5, the top five TSS results are achieved by TSF-TS, SVM-VLT, SVM-VSTAT, DT-VSTAT, and LR-VSTATN with TSS values of 0.7891, 0.7857, 0.7762, 0.7174, and 0.6974, respectively.

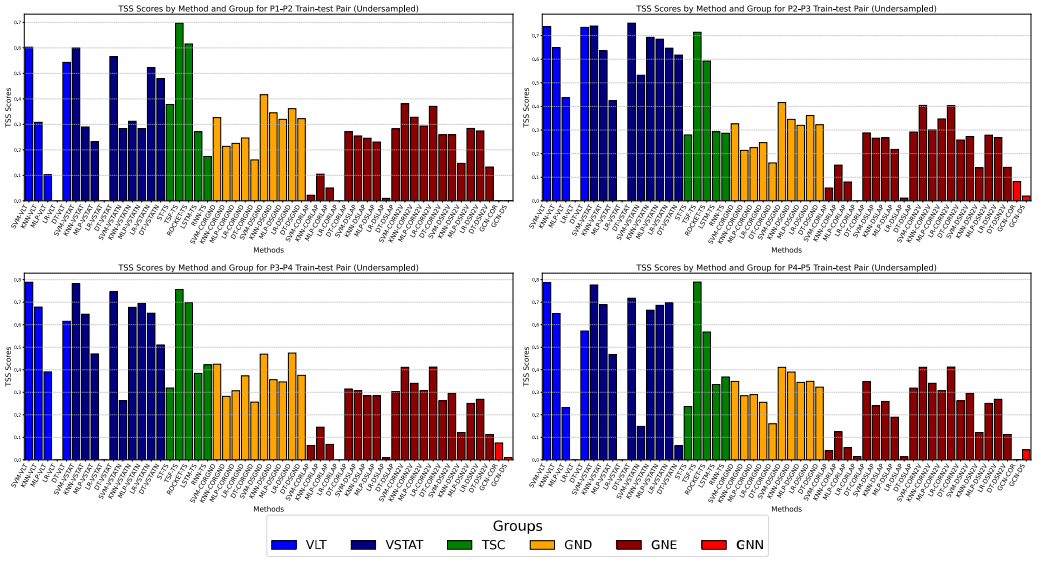


Figure 18. The TSS results for each model, based on the given data representation methods, are demonstrated using all train–test partition pairs under the undersampled settings.

In the culmination of our experiments under undersampled conditions, while the vector representation method demonstrates the most consistent performance across models, both the vector-based and time

series-based approaches exhibit noteworthy results. In particular, the best performers in previous imbalanced settings, the SVM-based vector models—SVM-VLT and SVM-VSTAT—continue to show promising solar flare prediction performance, with improvements across all metrics. Additionally, the vector-based DT-VSTAT model achieves the highest F1 and GS scores. On the other hand, the time series-based TSF-TS model outperforms in terms of TSS and ROC AUC scores. *Ultimately, the choice of the best model depends on the specific metric of interest and the task's requirements, as each method offers distinct strengths. The results consistently show that the graph-based method is the least effective across all measures*.

4.6. Solar Flare Prediction Performance Under Optimized Preprocessing

In the final stage of our experimental evaluation, we assess the solar flare prediction performance of our models utilizing the optimized SWAN-SF dataset, which has undergone a distinct set of preprocessing techniques, including advanced normalization and a combination of undersampling and oversampling methods. The sizes of the new training sets are presented in Table 8.

The averaged test results of all train and test partition pairs are presented in Table 9. Out of the six metrics evaluated, vector-based models demonstrate the highest performance in four metrics, while time series-based models excel in the remaining metrics. The SVM-VSTAT model achieves the highest accuracy at 0.9525. Regarding the GS and ROC AUC metrics, the RNN-TS model performs best, recording scores of 0.023 and 0.8764, respectively. The LR-VSTAT model excels in the TSS, HSS2, and F1 metrics, with scores of 0.7415, 0.2407, and 0.2718, thus demonstrating the most prominent performance in optimized preprocessing settings. *We can infer that among time series models, the sequence models LSTM-TS and RNN-TS exhibit a significant leap in performance, likely resulting from the optimized preprocessing that leverages the temporal characteristics of SWAN-SF*. Conversely, there is a noticeable decrease in previous top performers belonging to the vector representation, such as SVM and DT models, indicating a shift in effectiveness under these conditions. *Another noteworthy observation is that the GNN-based method demonstrates commendable performance for the first time in this setting, with GCN-DS using distance similarity-based graph inputs*. This finding will be a focal point for future research aimed at understanding the underlying reasons for this improvement and exploring strategies to enhance the resilience and robustness of graph-based methods.

Table 8. Class sizes before and after optimized preprocessing for each training partition.

Partition	Before Optimized Preprocessing		After Optimized Preprocessing	
	F	NF	F	NF
P1	1180	56,319	8778	9995
P2	1285	65,364	9807	10,000
P3	1277	30,766	9968	9997
P4	890	36,667	9320	10,000

Figure 19 shows the median performance of three data representation methods after we identify the top performers. In the optimized setting, the time series representation method demonstrates the best median performance with low variability, surpassing the vector-based methods in all six metrics, while the graph-based representation method consistently ranks the lowest.

We compare the TSS performances of each model in each train–test partition pair, and Figure 20 displays the results. Regarding the top performers (Table 5), for the train–test pair of P1–P2, the top five TSS results are achieved by RNN-TS, LSTM-TS, LR-VSTAT, KNN-VLT, and LR-VLT, with TSS values of 0.8036, 0.8015, 0.7819, 0.7585, and 0.7061, respectively. For

the train–test pair of P2–P3, the top five TSS results are achieved by TSF-TS, LR-VSTAT, RNN-TS, KNN-VLT, and ST-TS with TSS values of 0.768, 0.7445, 0.6899, 0.6628, and 0.6551, respectively. For the train–test pair of P3–P4, the top five TSS results are achieved by LR-VLT, RNN-TS, LR-VSTAT, TSF-TS, and LSTM-TS with TSS values of 0.8094, 0.7648, 0.7117, 0.6875, and 0.6671, respectively. For the train–test pair of P4–P5, the top five TSS results are achieved by TSF-TS, LR-VSTAT, RNN-TS, ST-TS, and KNN-VLT with TSS values of 0.7645, 0.7278, 0.6873, 0.6464, and 0.6434, respectively.

In the final analysis of our experiments under optimized preprocessing conditions, the time series representation method exhibits the most consistent performance, particularly with sequence models. While the vector representation models that previously performed best tend to underperform in this new setting, the vector representation still manages to secure the top results with the LR-VSTAT model. The findings indicate that the graph-based approach remains the least effective across all evaluation criteria in this context as well, rendering the graph representation a suboptimal choice for solar flare prediction and analysis tasks.

Table 9. Performance results of classifier models for data representations of SWAN-SF across averaged train–test pairs with optimized preprocessing, where red fonts represent maximum values.

Models	Accuracy	TSS	HSS2	F1	GS	ROC AUC
VLT						
SVM-VLT	0.6098 ± 0.4713	0.2208 ± 0.2609	0.074 ± 0.0744	0.1072 ± 0.0469	0.015 ± 0.0156	0.6104 ± 0.1304
KNN-VLT	0.8399 ± 0.1073	0.5562 ± 0.2717	0.1624 ± 0.0656	0.1942 ± 0.0624	0.0163 ± 0.0113	0.7781 ± 0.1359
MLP-VLT	0.6799 ± 0.3985	0.3392 ± 0.2878	0.1251 ± 0.1166	0.1563 ± 0.0884	0.0161 ± 0.0149	0.6679 ± 0.141
LR-VLT	0.6589 ± 0.2117	0.6378 ± 0.214	0.0952 ± 0.0634	0.1361 ± 0.055	0.0229 ± 0.0087	0.8189 ± 0.107
DT-VLT	0.4971 ± 0.2419	0.0027 ± 0.6265	0.0291 ± 0.0851	0.0605 ± 0.0983	0.0019 ± 0.0523	0.5014 ± 0.3133
VSTAT						
SVM-VSTAT	**0.9525 ± 0.0188**	0.1667 ± 0.2593	0.0727 ± 0.0887	0.0888 ± 0.0937	0.0029 ± 0.0043	0.5833 ± 0.1296
KNN-VSTAT	0.9013 ± 0.004	0.2856 ± 0.0088	0.1188 ± 0.0415	0.1514 ± 0.0493	0.0074 ± 0.0028	0.6428 ± 0.0044
MLP-VSTAT	0.8616 ± 0.1513	0.474 ± 0.2062	0.1889 ± 0.0731	0.219 ± 0.0543	0.015 ± 0.0133	0.7369 ± 0.1031
LR-VSTAT	0.8577 ± 0.0954	**0.7415 ± 0.0351**	**0.2407 ± 0.1461**	**0.2718 ± 0.1376**	0.0209 ± 0.0096	0.873 ± 0.0175
DT-VSTAT	0.5508 ± 0.0833	−0.1962 ± 0.3067	−0.0294 ± 0.0355	0.0172 ± 0.0199	−0.0105 ± 0.0132	0.4019 ± 0.1534
TSC						
ST-TS	0.8814 ± 0.1045	0.3915 ± 0.3428	0.1001 ± 0.081	0.126 ± 0.1019	0.0123 ± 0.0142	0.6958 ± 0.1714
TSF-TS	0.8855 ± 0.0935	0.4913 ± 0.4115	0.1452 ± 0.1078	0.1751 ± 0.122	0.0164 ± 0.0158	0.7457 ± 0.2058
ROCKET-TS	0.7279 ± 0.3083	0.2639 ± 0.1813	0.0586 ± 0.0343	0.0973 ± 0.0366	0.0144 ± 0.0162	0.6319 ± 0.0907
LSTM-TS	0.7363 ± 0.3172	0.6067 ± 0.2309	0.2089 ± 0.1979	0.2433 ± 0.1786	0.0207 ± 0.0101	0.8033 ± 0.1154
RNN-TS	0.7751 ± 0.0715	0.7364 ± 0.0578	0.1343 ± 0.0069	0.1722 ± 0.0109	**0.023 ± 0.0087**	**0.8764 ± 0.0289**
GND						
SVM-CORGND	0.8911 ± 0.0237	0.141 ± 0.0075	0.0608 ± 0.0336	0.0947 ± 0.0389	0.0037 ± 0.0011	0.5705 ± 0.0038
KNN-CORGND	0.8424 ± 0.0231	0.1143 ± 0.0106	0.0345 ± 0.0179	0.0731 ± 0.0281	0.0032 ± 0.0012	0.5572 ± 0.0053
MLP-CORGND	0.8741 ± 0.0304	0.144 ± 0.0138	0.053 ± 0.0233	0.0892 ± 0.0328	0.0039 ± 0.0015	0.572 ± 0.0069
LR-CORGND	0.7514 ± 0.0101	0.1611 ± 0.0177	0.0302 ± 0.0145	0.0721 ± 0.0279	0.0052 ± 0.0024	0.5805 ± 0.0088
DT-CORGND	0.8427 ± 0.0134	0.1052 ± 0.0185	0.0316 ± 0.017	0.0708 ± 0.0283	0.0031 ± 0.0016	0.5526 ± 0.0092
SVM-DSGND	0.8911 ± 0.0237	0.141 ± 0.0075	0.0608 ± 0.0336	0.0947 ± 0.0389	0.0037 ± 0.0011	0.5705 ± 0.0038
KNN-DSGND	0.8424 ± 0.0231	0.1143 ± 0.0106	0.0345 ± 0.0179	0.0731 ± 0.0281	0.0032 ± 0.0012	0.5572 ± 0.0053
MLP-DSGND	0.8759 ± 0.0167	0.1614 ± 0.023	0.0577 ± 0.0207	0.0938 ± 0.0293	0.0042 ± 0.0011	0.5807 ± 0.0115
LR-DSGND	0.7514 ± 0.0101	0.1611 ± 0.0177	0.0302 ± 0.0145	0.0721 ± 0.0279	0.0052 ± 0.0024	0.5805 ± 0.0088
DT-DSGND	0.8427 ± 0.0134	0.1052 ± 0.0185	0.0316 ± 0.017	0.0708 ± 0.0283	0.0031 ± 0.0016	0.5526 ± 0.0092
GNE						
SVM-CORLAP	0.7718 ± 0.0596	0.2244 ± 0.02	0.0438 ± 0.0109	0.085 ± 0.0236	0.0071 ± 0.0036	0.6122 ± 0.01
KNN-CORLAP	0.7491 ± 0.0257	0.1703 ± 0.0279	0.0317 ± 0.0151	0.0732 ± 0.0281	0.0054 ± 0.0022	0.5852 ± 0.014
MLP-CORLAP	0.7634 ± 0.013	0.1981 ± 0.0122	0.0375 ± 0.0136	0.0789 ± 0.027	0.0062 ± 0.0024	0.599 ± 0.0061
LR-CORLAP	0.5774 ± 0.1054	−0.0056 ± 0.0115	−0.0004 ± 0.0011	0.0447 ± 0.0168	−0.0002 ± 0.0006	0.4972 ± 0.0057
DT-CORLAP	0.7767 ± 0.0185	0.1655 ± 0.0061	0.034 ± 0.0151	0.0751 ± 0.0277	0.005 ± 0.0019	0.5827 ± 0.003
SVM-DSLAP	0.8405 ± 0.0525	0.5152 ± 0.1525	0.1375 ± 0.0828	0.1716 ± 0.0869	0.0138 ± 0.0066	0.7576 ± 0.0763
KNN-DSLAP	0.6636 ± 0.1059	0.4167 ± 0.1522	0.0628 ± 0.0527	0.1026 ± 0.0623	0.0142 ± 0.0072	0.7083 ± 0.0761
MLP-DSLAP	0.759 ± 0.0399	0.4047 ± 0.1047	0.069 ± 0.0227	0.1084 ± 0.0343	0.0125 ± 0.0071	0.7023 ± 0.0524
LR-DSLAP	0.5594 ± 0.0768	−0.0139 ± 0.0247	−0.001 ± 0.0023	0.0432 ± 0.0191	−0.0003 ± 0.0007	0.493 ± 0.0123
DT-DSLAP	0.7247 ± 0.0874	0.2625 ± 0.0753	0.0413 ± 0.0139	0.0824 ± 0.0294	0.0093 ± 0.0063	0.6312 ± 0.0377
SVM-CORN2V	0.5856 ± 0.0415	0.0955 ± 0.0167	0.0105 ± 0.0033	0.0549 ± 0.0183	0.0037 ± 0.0007	0.5478 ± 0.0083
KNN-CORN2V	0.4816 ± 0.0222	0.0384 ± 0.009	0.0036 ± 0.002	0.049 ± 0.018	0.002 ± 0.0011	0.5192 ± 0.0045
MLP-CORN2V	0.572 ± 0.1606	0.0605 ± 0.0097	0.008 ± 0.0058	0.0517 ± 0.0194	0.0025 ± 0.0005	0.5303 ± 0.0049

Table 9. *Cont.*

Models	Accuracy	TSS	HSS2	F1	GS	ROC AUC
LR-CORN2V	0.6374 ± 0.0505	0.0905 ± 0.0189	0.0109 ± 0.0005	0.0551 ± 0.0162	0.0032 ± 0.0008	0.5453 ± 0.0094
DT-CORN2V	0.5305 ± 0.0129	0.016 ± 0.0084	0.0016 ± 0.0009	0.0469 ± 0.0166	0.0007 ± 0.0004	0.508 ± 0.0042
SVM-DSN2V	0.4544 ± 0.1074	0.3046 ± 0.0477	0.0299 ± 0.0195	0.0765 ± 0.0303	0.017 ± 0.0032	0.6523 ± 0.0239
KNN-DSN2V	0.5044 ± 0.0722	0.1931 ± 0.0382	0.0207 ± 0.0132	0.0677 ± 0.0244	0.0097 ± 0.0032	0.5965 ± 0.0191
MLP-DSN2V	0.4014 ± 0.0556	0.2804 ± 0.0546	0.0241 ± 0.0137	0.0714 ± 0.0254	0.018 ± 0.006	0.6402 ± 0.0273
LR-DSN2V	0.4687 ± 0.0988	0.3106 ± 0.0477	0.031 ± 0.0197	0.0776 ± 0.0304	0.0168 ± 0.0035	0.6553 ± 0.0238
DT-DSN2V	0.4927 ± 0.0399	0.1435 ± 0.0391	0.0149 ± 0.0095	0.0623 ± 0.0213	0.0076 ± 0.0036	0.5717 ± 0.0196
GNN						
GCN-COR	0.658 ± 0.5498	−0.0016 ± 0.0035	−0.0003 ± 0.0006	0.0143 ± 0.0257	0.0072 ± 0.0128	0.5001 ± 0.0002
GCN-DS	0.5652 ± 0.0199	0.4258 ± 0.0793	0.0466 ± 0.0318	0.09 ± 0.0526	0.0189 ± 0.0124	0.7128 ± 0.0397

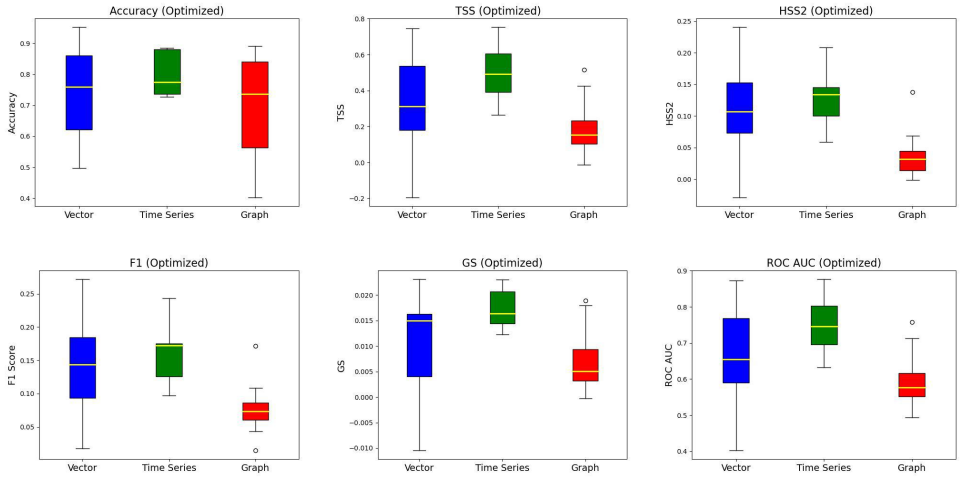


Figure 19. Demonstration of accuracy, TSS, HSS2, F1, GS, and ROC AUC performance results of vector (blue), time series (green), and graph (red) data representation methods in boxplot format under optimized preprocessing setting. For each method, the classification results of all submethods are aggregated to illustrate the overall performance of the group.

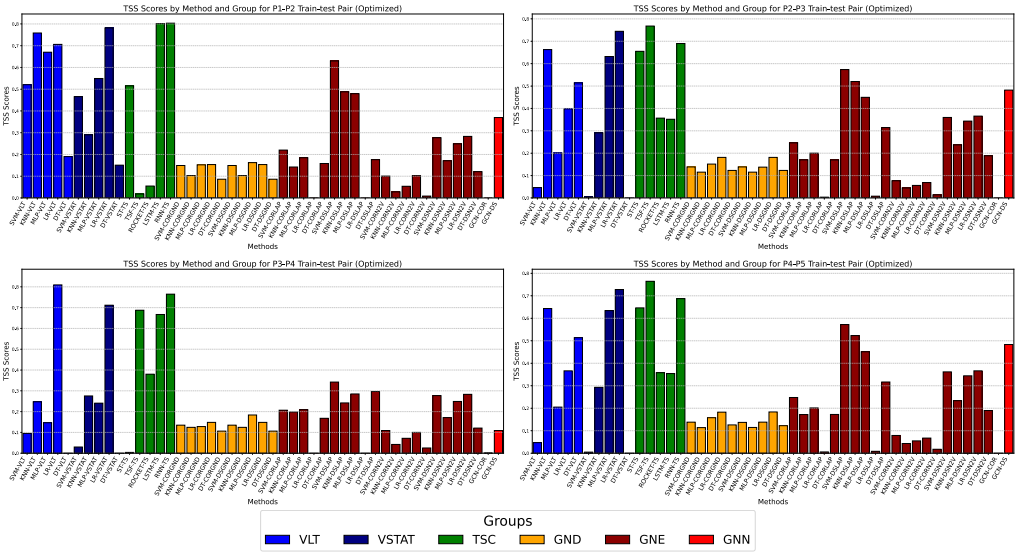


Figure 20. The TSS results for each model, based on the given data representation methods, are demonstrated using all train–test partition pairs under the optimized settings.

5. Discussion

In this work, we evaluated three solar flare prediction approaches regarding data representations of the SWAN-SF dataset: vector-based, time series-based, and graph-based

methods. The objective of this study was to provide a comprehensive guide for selecting machine learning-oriented models and components in the field of solar flare prediction research, focusing on various conditions under which different models and data representations are effectively chosen. In this context, we aimed to identify the data representation method that most effectively represents the characteristics of the solar flare dataset based on photospheric magnetic field parameters under three settings: imbalanced, undersampled, and custom-preprocessed. The vector representation intended to encapsulate the summarization of the multivariate time series, the time series representation focused on capturing temporal patterns, and the graph representation aimed to elucidate inter-parameter dependencies. The key findings of this study, as demonstrated by experimental validation, are as follows:

(1) Under extreme class imbalance conditions, superior performance was achieved by the vector-based approach, with SVM-VLT and SVM-VSTAT being the best performers in this method.
(2) Under undersampling conditions (i.e., when the training partition is balanced through undersampling), the best-performing vector and time series models demonstrated competitive performance. Specifically, the vector-based SVM-VLT and SVM-VSTAT models continued to show high performance, the DT-VSTAT model achieved the highest F1 and GS scores, and the time series-based TSF model excelled in TSS and ROC AUC scores. Ultimately, the utilization of these models depends on the specific metric of interest.
(3) Under optimized preprocessing conditions, the vector representation still secured leading outcomes with the LR-VSTAT model, while the time series representation method exhibited high consistency in performance, particularly with sequence models.
(4) In all settings, vector-based methods proved to be highly resilient in solar flare prediction tasks when coupled with an appropriate downstream classifier, demonstrating robust prediction results by effectively capturing complex relationships and summarizing multivariate data.

In future studies, we aim to expand the scope of data representations and investigate their effects under extended conditions, as this approach is deemed highly valuable and is expected to significantly influence model selections in subsequent research. Furthermore, we will experiment with different graph features, such as the clustering coefficient [47] and graphlet kernel [48,49], to yield better results. In addition, we have yet to explore the application of other normalization and preprocessing techniques or the utilization of node clustering methods to further investigate the relation between photospheric magnetic field parameters. Finally, we plan a specific study to improve the representation power of graph-based approaches in the SWAN-SF dataset by combining three types of features to capture temporal, spatial, and both higher-order and lower-order details. In this respect, MVTS data, the Pearson correlation matrix, and frequent subgraph similarity results will compose the feature space where we will select the most statistically significant features to train downstream classifiers.

Author Contributions: Conceptualization, O.V. and S.M.H.; methodology, O.V.; software, O.V.; validation, O.V.; formal analysis, O.V.; investigation, O.V.; resources, O.V., S.M.H., and S.F.B.; data curation, O.V.; writing—original draft preparation, O.V.; writing—review and editing, O.V. and S.M.H.; visualization, O.V.; supervision, S.M.H. and S.F.B.; project administration, O.V. and S.M.H.; funding acquisition, S.M.H. and S.F.B. All authors have read and agreed to the published version of the manuscript.

Funding: This project has been supported in part by funding from the Division of Atmospheric and Geospace Sciences within the Directorate for Geosciences, under NSF awards #2301397, #2204363, and

#2240022, and by funding from the Office of Advanced Cyberinfrastructure within the Directorate for Computer and Information Science and Engineering, under NSF award #2305781.

Data Availability Statement: The SWAN-SF dataset, utilized in this study's experiments, is publicly accessible via https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/EBCFKM (accessed on 30 January 2025).

Acknowledgments: The authors acknowledge all those involved with the GOES missions as well as the SDO mission.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Ahmadzadeh, A.; Aydin, B.; Georgoulis, M.K.; Kempton, D.J.; Mahajan, S.S.; Angryk, R.A. How to train your flare prediction model: Revisiting robust sampling of rare events. *Astrophys. J. Suppl. Ser.* **2021**, *254*, 23. [CrossRef]
2. Hamdi, S.M.; Kempton, D.; Ma, R.; Boubrahimi, S.F.; Angryk, R.A. A time series classification-based approach for solar flare prediction. In Proceedings of the 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, USA, 11–14 December 2017; pp. 2543–2551.
3. Bobra, M.G.; Couvidat, S. Solar flare prediction using SDO/HMI vector magnetic field data with a machine-learning algorithm. *Astrophys. J.* **2015**, *798*, 135. [CrossRef]
4. Eastwood, J.; Biffis, E.; Hapgood, M.; Green, L.; Bisi, M.; Bentley, R.; Wicks, R.; McKinnell, L.A.; Gibbs, M.; Burnett, C. The economic impact of space weather: Where do we stand? *Risk Anal.* **2017**, *37*, 206–218. [CrossRef] [PubMed]
5. Angryk, R.; Martens, P.; Aydin, B.; Kempton, D.; Mahajan, S.; Basodi, S.; Ahmadzadeh, A.; Cai, X.; Filali Boubrahimi, S.; Hamdi, S.M.; et al. *SWAN-SF*; Harvard Dataverse: Cambridge, MA, USA , 2020. [CrossRef]
6. Angryk, R.A.; Martens, P.C.; Aydin, B.; Kempton, D.; Mahajan, S.S.; Basodi, S.; Ahmadzadeh, A.; Cai, X.; Filali Boubrahimi, S.; Hamdi, S.M.; et al. Multivariate time series dataset for space weather data analytics. *Sci. Data* **2020**, *7*, 227. [CrossRef]
7. Vural, O.; Hamdi, S.M.; Filali Boubrahimi, S. EXCON: Extreme Instance-based Contrastive Representation Learning of Severely Imbalanced Multivariate Time Series for Solar Flare Prediction. In Proceedings of the 2024 IEEE International Conference on Big Data (BigData), Washington, DC, USA, 15–18 December 2024; pp. 1476–1483. [CrossRef]
8. McIntosh, P.S. The classification of sunspot groups. *Sol. Phys.* **1990**, *125*, 251–267. [CrossRef]
9. Boubrahimi, S.F.; Aydin, B.; Kempton, D.; Angryk, R. Spatio-temporal interpolation methods for solar events metadata. In Proceedings of the 2016 IEEE International Conference on Big Data (Big Data), Washington, DC, USA, 5–8 December 2016; pp. 3149–3157.
10. Mason, J.P.; Hoeksema, J. Testing automated solar flare forecasting with 13 years of Michelson Doppler Imager magnetograms. *Astrophys. J.* **2010**, *723*, 634. [CrossRef]
11. Song, H.; Tan, C.; Jing, J.; Wang, H.; Yurchyshyn, V.; Abramenko, V. Statistical assessment of photospheric magnetic features in imminent solar flare predictions. *Sol. Phys.* **2009**, *254*, 101–125. [CrossRef]
12. Yu, D.; Huang, X.; Wang, H.; Cui, Y. Short-term solar flare prediction using a sequential supervised learning method. *Sol. Phys.* **2009**, *255*, 91–105. [CrossRef]
13. Ahmed, O.W.; Qahwaji, R.; Colak, T.; Higgins, P.A.; Gallagher, P.T.; Bloomfield, D.S. Solar flare prediction using advanced feature extraction, machine learning, and feature selection. *Sol. Phys.* **2013**, *283*, 157–175. [CrossRef]
14. Al-Ghraibah, A.; Boucheron, L.; McAteer, R. An automated classification approach to ranking photospheric proxies of magnetic energy build-up. *Astron. Astrophys.* **2015**, *579*, A64. [CrossRef]
15. Ma, R.; Boubrahimi, S.F.; Hamdi, S.M.; Angryk, R.A. Solar flare prediction using multivariate time series decision trees. In Proceedings of the 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, USA, 11–14 December 2017; pp. 2569–2578.
16. Muzaheed, A.A.M.; Hamdi, S.M.; Boubrahimi, S.F. Sequence model-based end-to-end solar flare classification from multivariate time series data. In Proceedings of the 2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA), Boston, MA, USA, 11–14 December 2021; pp. 435–440.
17. Vural, O.; Hamdi, S.M.; Boubrahimi, S.F. Contrastive Representation Learning for Predicting Solar Flares from Extremely Imbalanced Multivariate Time Series Data. In Proceedings of the 2024 International Conference on Machine Learning and Applications (ICMLA), Miami, FL, USA, 18–20 December 2024; pp. 1077–1082. [CrossRef]
18. Sun, Z.; Bobra, M.G.; Wang, X.; Wang, Y.; Sun, H.; Gombosi, T.; Chen, Y.; Hero, A. Predicting solar flares using CNN and LSTM on two solar cycles of active region data. *Astrophys. J.* **2022**, *931*, 163. [CrossRef]
19. Zheng, Y.; Qin, W.; Li, X.; Ling, Y.; Huang, X.; Li, X.; Yan, P.; Yan, S.; Lou, H. Comparative analysis of machine learning models for solar flare prediction. *Astrophys. Space Sci.* **2023**, *368*, 53. [CrossRef]

20. Hamdi, S.M.; Ahmad, A.F.; Filali Boubrahimi, S. Multivariate time series-based solar flare prediction by functional network embedding and sequence modeling. In Proceedings of the CIKM workshop for Applied Machine Learning Methods for Time Series Forecasting (AMLTS 2022), Atlanta, GA, USA, 21 October 2022.
21. Duan, Z.; Xu, H.; Wang, Y.; Huang, Y.; Ren, A.; Xu, Z.; Sun, Y.; Wang, W. Multivariate time-series classification with hierarchical variational graph pooling. *Neural Netw.* **2022**, *154*, 481–490. [CrossRef] [PubMed]
22. Cai, H.; Zheng, V.W.; Chang, K.C.C. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Trans. Knowl. Data Eng.* **2018**, *30*, 1616–1637. [CrossRef]
23. Goyal, P.; Ferrara, E. Graph embedding techniques, applications, and performance: A survey. *Knowl.-Based Syst.* **2018**, *151*, 78–94. [CrossRef]
24. Ng, A.; Jordan, M.; Weiss, Y. On spectral clustering: Analysis and an algorithm. *Adv. Neural Inf. Process. Syst.* **2001**, *14*, 1–8.
25. Cao, S.; Lu, W.; Xu, Q. Grarep: Learning graph representations with global structural information. In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, Melbourne, Australia, 19–23 October 2015; pp. 891–900.
26. Hamdi, S.M.; Filali Boubrahimi, S.; Angryk, R. Tensor decomposition-based node embedding. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; pp. 2105–2108.
27. Perozzi, B.; Al-Rfou, R.; Skiena, S. Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014; pp. 701–710.
28. Grover, A.; Leskovec, J. node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 855–864.
29. Scarselli, F.; Gori, M.; Tsoi, A.C.; Hagenbuchner, M.; Monfardini, G. The graph neural network model. *IEEE Trans. Neural Netw.* **2008**, *20*, 61–80. [CrossRef]
30. Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; Sun, M. Graph neural networks: A review of methods and applications. *AI Open* **2020**, *1*, 57–81. [CrossRef]
31. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv* **2016**, arXiv:1609.02907.
32. Hamilton, W.; Ying, Z.; Leskovec, J. Inductive representation learning on large graphs. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 1–11.
33. Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. *Stat* **2017**, *1050*, 10-48550.
34. Lines, J.; Davis, L.M.; Hills, J.; Bagnall, A. A shapelet transform for time series classification. In Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Beijing, China, 12–16 August 2012; pp. 289–297.
35. Bostrom, A.; Bagnall, A. A shapelet transform for multivariate time series classification. *arXiv* **2017**, arXiv:1712.06428.
36. Deng, H.; Runger, G.; Tuv, E.; Vladimir, M. A time series forest for classification and feature extraction. *Inf. Sci.* **2013**, *239*, 142–153. [CrossRef]
37. Dempster, A.; Petitjean, F.; Webb, G.I. ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels. *Data Min. Knowl. Discov.* **2020**, *34*, 1454–1495. [CrossRef]
38. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. *Learning Internal Representations by Error Propagation*; Stanford University: Stanford, CA, USA, 1985.
39. Hochreiter, S.; Schmidhuber, J. Long Short-term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
40. Barnes, G.; Leka, K. Evaluating the performance of solar flare forecasting methods. *Astrophys. J.* **2008**, *688*, L107. [CrossRef]
41. Ji, A.; Aydin, B.; Georgoulis, M.K.; Angryk, R. All-clear flare prediction using interval-based time series classifiers. In Proceedings of the 2020 IEEE International Conference on Big Data (Big Data), Atlanta, GA, USA, 10–13 December 2020; pp. 4218–4225.
42. Batista, G.E.; Monard, M.C. A study of K-nearest neighbour as an imputation method. *His* **2002**, *87*, 48.
43. EskandariNasab, M.; Hamdi, S.M.; Boubrahimi, S.F. Impacts of data preprocessing and sampling techniques on solar flare prediction from multivariate time series data of photospheric magnetic field parameters. *Astrophys. J. Suppl. Ser.* **2024**, *275*, 6. [CrossRef]
44. Yoon, J.; Jarrett, D.; van der Schaar, M. Time-series Generative Adversarial Networks. In *Proceedings of the Advances in Neural Information Processing Systems*; Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2019; Volume 32.
45. Elhassan, T.; Aljurf, M. Classification of imbalance data using tomek link (t-link) combined with random under-sampling (rus) as a data reduction method. *Glob. J. Technol. Optim.* **2016**, *1*, 2016.
46. Rodriguez, J.; Kuncheva, L.; Alonso, C. Rotation Forest: A New Classifier Ensemble Method. *IEEE Trans. Pattern Anal. Mach. Intell.* **2006**, *28*, 1619–1630. [CrossRef]
47. Masuda, N.; Sakaki, M.; Ezaki, T.; Watanabe, T. Clustering coefficients for correlation networks. *Front. Neuroinform.* **2018**, *12*, 7. [CrossRef] [PubMed]

48. Shervashidze, N.; Vishwanathan, S.V.N.; Petri, T.; Mehlhorn, K.; Borgwardt, K.M. Efficient graphlet kernels for large graph comparison. In Proceedings of the International Conference on Artificial Intelligence and Statistics, Clearwater Beach, FL, USA, 16–18 April 2009.
49. Shervashidze, N.; Schweitzer, P.; van Leeuwen, E.J.; Mehlhorn, K.; Borgwardt, K.M. Weisfeiler-Lehman Graph Kernels. *J. Mach. Learn. Res.* **2011**, *12*, 2539–2561.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.