The Astrophysical Journal Supplement Series, 275:6 (31pp), 2024 November https://doi.org/10.3847/1538-4365/ad7c4a

Impacts of Data Preprocessing and Sampling Techniques on Solar Flare Prediction from Multivariate Time Series Data of Photospheric Magnetic Field Parameters

MohammadReza EskandariNasab[1], Shah Muhammad Hamdi, and Soukaina Filali Boubrahimi
Computer Science Department, Utah State University, Logan, UT 84322, USA; reza.eskandarinasab@usu.edu, s.hamdi@usu.edu, soukaina.boubrahimi@usu.edu
Received 2024 March 17; revised 2024 September 3; accepted 2024 September 7; published 2024 October 25

Abstract

The accurate prediction of solar flares is crucial due to their risks to astronauts, space equipment, and satellite communication systems. Our research enhances solar flare prediction by employing sophisticated data preprocessing and sampling techniques for the Space Weather Analytics for Solar Flares (SWAN-SF) data set, a rich source of multivariate time series data of solar active regions. Our study adopts a multifaceted approach encompassing four key methodologies. Initially, we address over 10 million missing values in the SWAN-SF data set through our innovative imputation technique called fast Pearson correlation-based k-nearest neighbors imputation. Subsequently, we propose a precise normalization technique, called LSBZM normalization, tailored for time series data, merging various strategies (log, square root, Box–Cox, Z-score, and min–max) to uniformly scale the data set's 24 attributes (photospheric magnetic field parameters), addressing issues such as skewness. We also explore the "near decision boundary sample removal" technique to enhance the classification performance of the data set by effectively resolving the challenge of class overlap. Finally, a pivotal aspect of our research is a thorough evaluation of diverse oversampling and undersampling methods, including SMOTE, ADASYN, Gaussian noise injection, TimeGAN, Tomek links, and random undersampling, to counter the severe imbalance in the SWAN-SF data set, notably a 60:1 ratio of major (X and M) to minor (C, B, and FQ) flaring events in binary classification. To demonstrate the effectiveness of our methods, we use eight classification algorithms, including advanced deep-learning-based architectures. Our analysis shows significant true skill statistic scores, underscoring the importance of data preprocessing and sampling in time-series-based solar flare prediction.

Unified Astronomy Thesaurus concepts: Solar flares (1496); Solar active regions (1974); Solar active region magnetic fields (1975); Time series analysis (1916); Classification (1907); Space weather (2037); Neural networks (1933)

1. Introduction

Solar flares pose a significant threat to humans and equipment in space due to their intense radiation (J. J. Curto 2020). These events can cause sudden and substantial increases in radiation, including extreme ultraviolet, X-rays, and gamma rays, across the electromagnetic spectrum. Since 1974, the National Oceanic and Atmospheric Administration's (NOAA) Geostationary Operational Environmental Satellites (GOES; W. P. Menzel & J. F. W. Purdom 1994) have been detecting and classifying solar flares within the 1–8 Å wavelength spectrum. The classification of these flares is logarithmically based on their peak soft X-ray flux, with categories labeled as A, B, C, M, and X, in ascending order of intensity, starting from a flux of $10^{-8}\,\mathrm{W\,m^{-2}}$. Therefore, an X-class flare's peak flux is typically 10 times more intense than that of an M-class flare and 100 times more intense than a C-class flare.

Recent research on flare prediction has focused on data science-based approaches, particularly using the spatiotemporal magnetic field data provided by the Helioseismic Magnetic Imager (HMI; P. H. Scherrer et al. 2012; J. Schou et al. 2012; J. T. Hoeksema et al. 2014) on board the Solar Dynamics Observatory (SDO; W. D. Pesnell et al. 2012; A. Ahmadzadeh et al. 2021). This data is transformed into multivariate time series (MVTS) instances for temporal window-based flare prediction (R. A. Angryk et al. 2020). In these instances, 24 photospheric magnetic field parameters, as outlined in Table 1 (M. G. Bobra & S. Couvidat 2015), are presented as time series, based on two key time windows: the prediction window (the period before the flare occurs) and the observation window (the period during which the active region (AR) parameter values are measured). The MVTS instances are then labeled with one of five classes, ranging from flare-quiet (FQ) instances (including both FQ and A class) to flare classes of increasing intensity (B, C, M, X). This classification approach has shown enhanced accuracy in predicting flares compared to models that use single time stamp–based magnetic field vector classification. A pivotal resource in this area of research is the Space Weather Analytics for Solar Flares (SWAN-SF) data set (R. A. Angryk et al. 2020), derived from solar photospheric vector magnetograms by the Space weather HMI Active Region Patch (SHARP) series (M. G. Bobra et al. 2014). This data set is notable for its accurate reflection of the class imbalance, which is inherent in solar flare prediction. It features a 60:1 imbalance ratio for GOES M- and X-class flares (major flaring) compared to B-, C-, and FQ-class flares (minor flaring), and an even more pronounced 800:1 imbalance ratio for X-class flare instances against FQ instances.

Data collected to address real-world problems is seldom clean or ready for immediate use, regardless of the thoroughness of the screening process (A. Behfar et al. 2023). Such data sets often inherit challenges related to the nature of the subject under study or the data collection strategy. These challenges, which include missing values, multiscaled attributes, skewness, class overlap, and class imbalance, are prevalent in many nonlinear dynamical

[1] Corresponding author.

Table 1
AR Magnetic Field Parameters in the SWAN-SF Data Set

Abbreviation	References	Description	Formula
ABSNJZH	K. D. Leka & G. Barnes (2003)	Absolute value of the net current helicity	$H_{c_{\rm abs}} \propto \lvert\sum B_z \cdot J_z\rvert$
EPSX	G. H. Fisher et al. (2012)	Sum of x-components of normalized Lorentz force	$\delta F_x \propto \frac{\sum B_x B_z}{\sum B^2}$
EPSY	G. H. Fisher et al. (2012)	Sum of y-components of normalized Lorentz force	$\delta F_y \propto \frac{-\sum B_y B_z}{\sum B^2}$
EPSZ	G. H. Fisher et al. (2012)	Sum of z-components of normalized Lorentz force	$\delta F_z \propto \frac{\sum (B_x^2 + B_y^2 - B_z^2)}{\sum B^2}$
MEANALP	K. D. Leka & A. Skumanich (1999)	Mean characteristic twist parameter, α	$\alpha_{\rm total} \propto \frac{\sum J_z B_z}{\sum B_z^2}$
MEANGAM	K. D. Leka & G. Barnes (2003)	Mean angle of field from radial	$\overline{\gamma} = \frac{1}{N}\sum \arctan\left(\frac{B_h}{B_z}\right)$
MEANGBH	K. D. Leka & G. Barnes (2003)	Mean gradient of the 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	K. D. Leka & G. Barnes (2003)	Mean gradient of the total field	$\overline{\lvert\nabla B_{\rm tot}\rvert} = \frac{1}{N}\sum \sqrt{\left(\frac{\partial B}{\partial x}\right)^2 + \left(\frac{\partial B}{\partial y}\right)^2}$
MEANGBZ	K. D. Leka & G. Barnes (2003)	Mean gradient of the 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	K. D. Leka & G. Barnes (2003)	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	K. D. Leka & G. Barnes (2003)	Mean current helicity (B_z contribution)	$\overline{H_c} \propto \frac{1}{N}\sum B_z \cdot J_z$
MEANPOT	J. Wang et al. (1996)	Mean photospheric magnetic free energy	$\bar{\rho} \propto \frac{1}{N}\sum (\boldsymbol{B}^{\rm Obs} - \boldsymbol{B}^{\rm Pot})^2$
MEANSHR	J. Wang et al. (1996)	Mean shear angle	$\bar{\Gamma} = \frac{1}{N}\sum \arccos\left(\frac{\boldsymbol{B}^{\rm Obs}\cdot\boldsymbol{B}^{\rm Pot}}{\lvert B^{\rm Obs}\rvert\,\lvert B^{\rm Pot}\rvert}\right)$
R_VALUE	C. J. Schrijver (2007)	Sum of flux near polarity inversion line	$\Phi = \Sigma\lvert B_{\rm LoS}\rvert\ dA$ (within R mask)
SAVNCPP	K. D. Leka & G. Barnes (2003)	Sum of the modulus of the net current per polarity	$J_{z_{\rm sum}} \propto \lvert\sum^{B_z^+} J_z dA\rvert + \lvert\sum^{B_z^-} J_z dA\rvert$
SHRGT45	K. D. Leka & G. Barnes (2003)	Fraction of area with shear >45°	Area with shear >45°/total area
TOTBSQ	G. H. Fisher et al. (2012)	Total magnitude of Lorentz force	$F \propto \sum B^2$
TOTFX	G. H. Fisher et al. (2012)	Sum of x-components of Lorentz force	$F_x \propto -\sum B_x B_z dA$
TOTFY	G. H. Fisher et al. (2012)	Sum of y-components of Lorentz force	$F_y \propto \sum B_y B_z dA$
TOTFZ	G. H. Fisher et al. (2012)	Sum of z-components of Lorentz force	$F_z \propto \sum (B_x^2 + B_y^2 - B_z^2) dA$
TOTPOT	K. D. Leka & G. Barnes (2003)	Total photospheric magnetic free energy density	$\rho_{\rm tot} \propto \sum (\boldsymbol{B}^{\rm Obs} - \boldsymbol{B}^{\rm Pot})^2 dA$
TOTUSJH	K. D. Leka & G. Barnes (2003)	Total unsigned current helicity	$H_{c_{\rm total}} \propto \sum B_z \cdot J_z$
TOTUSJZ	K. D. Leka & G. Barnes (2003)	Total unsigned vertical current	$J_{z_{\rm total}} = \sum \lvert J_z\rvert\ dA$
USFLUX	K. D. Leka & G. Barnes (2003)	Total unsigned flux	$\Phi = \sum \lvert B_z\rvert\ dA$

systems such as solar flare prediction (R. A. Angryk et al. 2020), and auditory attention detection (M. EskandariNasab et al. 2024c). For instance, the distribution of peak X-ray fluxes from solar flares follows a strong power law, covering a wide range across multiple orders of magnitude. This distribution is often explained by viewing flares as random events that exhibit self-organized criticality (M. J. Aschwanden et al. 2016). This distribution points to a significant class imbalance in flare occurrences. For example, data from solar cycle 23 (1996–2008) indicates that around half of the ARs (S. K. Dhakal & J. Zhang 2023) produced at least one C-class flare, but less than 2% produced an X-class flare (M. K. Georgoulis 2012). Moreover, solar cycle 24 (2009 to present) showed a similar trend in major flare occurrences despite having a similar number of ARs as cycle 23, emphasizing the criticality of addressing the class imbalance in flare prediction (A. Ahmadzadeh et al. 2021). Such significant challenges emphasize the necessity of developing sophisticated machine learning-based approaches for solar flare prediction.

Previous studies on solar flare prediction have concentrated on the development and optimization of machine learning algorithms to improve prediction accuracy (N. Nishizuka et al. 2017, 2018; K. D. Leka et al. 2019; S. Sinha et al. 2022). Among photospheric vector magnetogram-based methods, M. G. Bobra & S. Couvidat (2015) utilized preflare instantaneous values of AR magnetic field parameters to

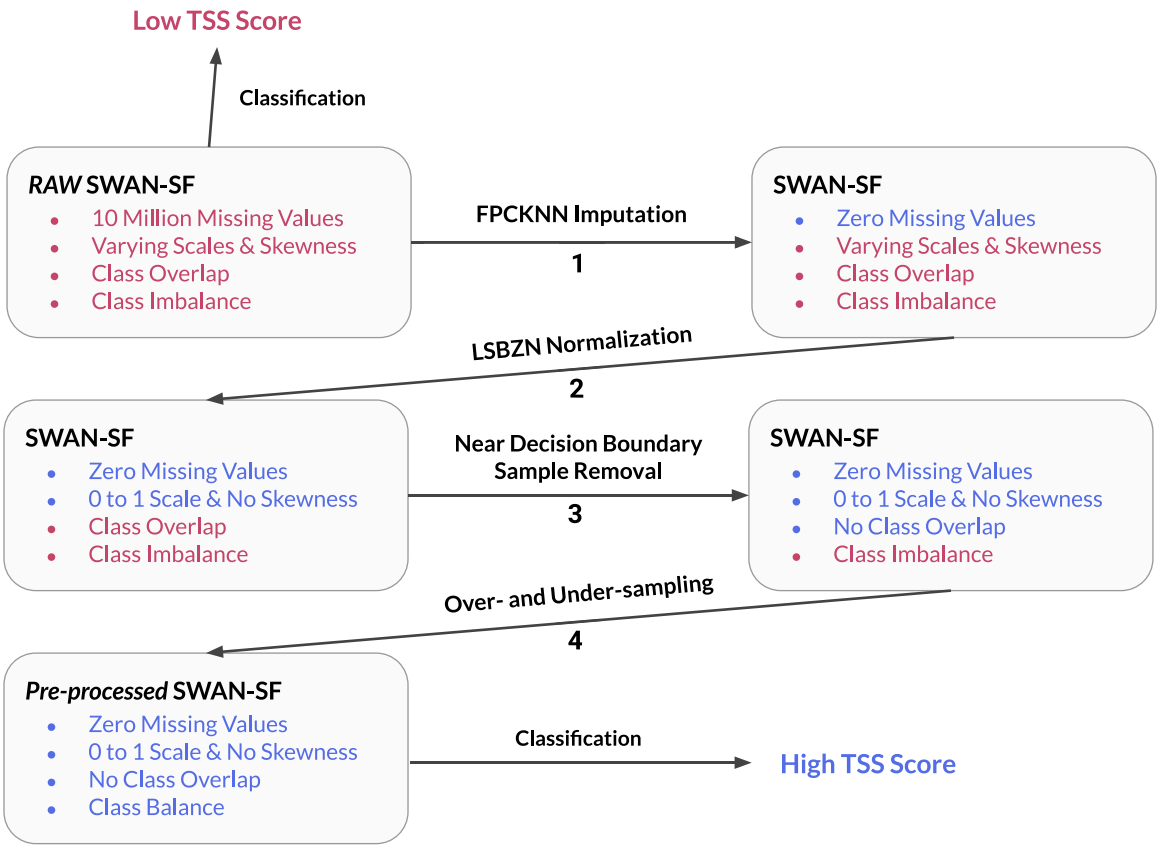

Figure 1. Our four essential data preprocessing and sampling techniques to enhance classification performance on the SWAN-SF data set.

forecast solar flares with a support vector machine (SVM) classifier. S. M. Hamdi et al. (2017) proposed a flare prediction method by extracting time series samples of AR parameters and employing k-nearest neighbors (k-NN) classification on the univariate time series. A. Ahmadzadeh et al. (2021) addressed specific challenges in solar flare forecasting, such as class imbalance and temporal coherence. They discussed strategies like undersampling and oversampling to manage class imbalance in the SWAN-SF data set and emphasized the importance of proper data splitting and validation techniques to ensure model robustness against temporal coherence. A. A. M. Muzaheed et al. (2021) utilized long short-term memory (LSTM) networks for effective end-to-end classification of MVTS in solar flare prediction, outperforming traditional models and underscoring the potential of deep learning. Similarly, S. M. Hamdi et al. (2022) developed a novel approach that combines graph convolutional networks with LSTM networks, effectively capturing both spatial and temporal relationships in solar flare prediction and surpassing other baseline methods. K. Alshammari et al. (2022) addressed the forecasting of magnetic field parameters related to flaring events using a deep sequence-to-sequence learning model with batch normalization and LSTM networks.

Unlike previous studies on solar flare prediction that primarily concentrated on enhancing classification methodologies, our study focuses on novel and comprehensive data preprocessing and sampling techniques for improving the data set quality. It addresses challenges within the SWAN-SF data set, including over 10 million missing values, significant class overlap and imbalance, and heterogeneity in attribute magnitudes across its photospheric magnetic field parameters. Techniques such as missing value imputation, which is crucial for creating a complete data set, and normalization, essential for harmonizing the magnitudes of heterogeneous attributes, play a key role in our methodologies. These techniques assist in resolving skewness, facilitating faster learning, and ensuring numerical stability. Moreover, addressing class overlap and employing sampling techniques such as oversampling and undersampling are essential for managing class imbalance, enhancing minority-class representation, and reducing majority-class dominance. Our contributions are as follows:

1. Introducing an advanced imputation technique called fast Pearson correlation-based k-NN (FPCKNN) imputation to effectively address missing values, ensuring high data quality and realism.
2. Introducing a comprehensive normalization technique called LSBZM (log, square root, Box–Cox, Z-score, and min–max) normalization, specifically tailored for time series data. This technique merges various strategies to uniformly scale the data set's 24 attributes (photospheric magnetic field parameters), effectively addressing skewness and ensuring data consistency.
3. Applying "near decision boundary sample removal (NDBSR)" techniques to enhance classification performance by resolving class overlap.
4. Evaluating of diverse oversampling and undersampling methods to counter the significant imbalance in the SWAN-SF data set.

In Figure 1, we summarize all the proposed preprocessing techniques and their respective order of importance to enhance the classification performance on the SWAN-SF data set.

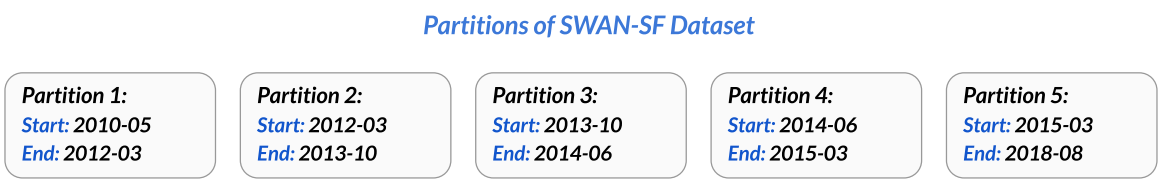

Figure 2. This figure presents a timeline indicating the occurrence of ARs within each partition of SWAN-SF. The first number represents the year, while the second number denotes the month.

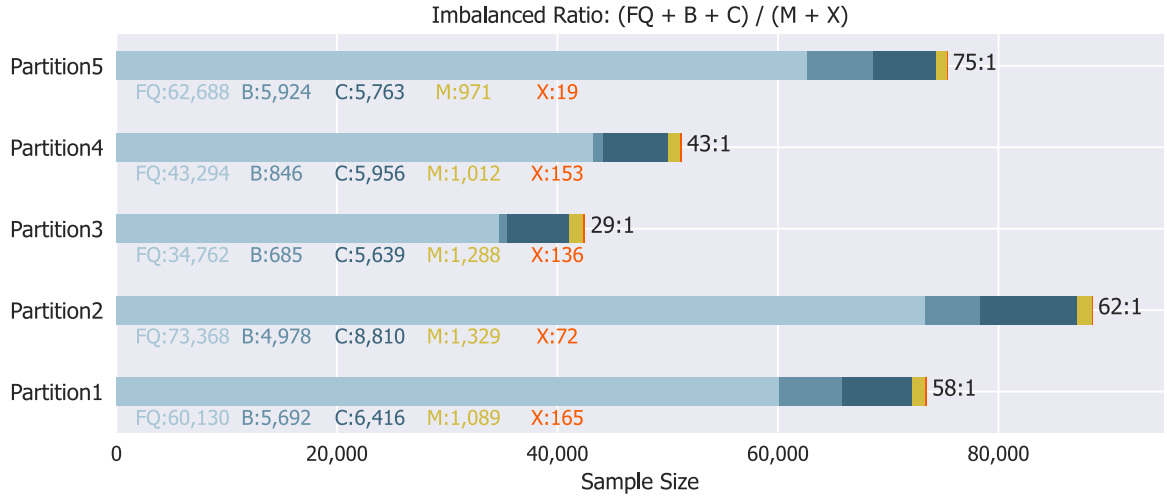

Figure 3. A stacked bar chart representing the population distribution of various flare classes in each partition of the SWAN-SF benchmark data set over time. Flare classes X, M, C, and B are derived and confirmed using the GOES classification, whereas FQ represents instances of FQ and those classified as A class by GOES. This visualization is based on the current methodology of time series slicing used in SWAN-SF, which involves steps of 1 hr, an observation period of 12 hr, and a prediction span of 24 hr. Each slice of the MVTS is categorized according to the most intense flare reported within its prediction window.

This paper is further organized as follows: Section 2 details the SWAN-SF data set and its challenges. Section 3 delves into our comprehensive methodologies. Specifically, Section 3.1 presents our innovative approach for imputation and normalization, while Sections 3.2 and 3.3 provide an in-depth discussion of our advanced sampling techniques. Sections 4 and 5 present the results of our data preprocessing and sampling methods and discuss the classification algorithms used for MVTS classification. Finally, Section 6 concludes the paper with a summary and a look toward future work. For hyperparameters, notations, and additional figures, refer to Appendices A, B, and C.

2. SWAN-SF Data Set

2.1. SWAN-SF Benchmark Data Set

In our study, the SWAN-SF data set (R. A. Angryk et al. 2020) is employed as a foundational data set for solar flare prediction, leveraging MVTS of photospheric magnetic field parameters. It classifies solar flares into five categories: GOES X, M, C, B, and FQ, where class FQ covers both FQ and GOES A-class events. The data set is segmented into five partitions, each part maintaining a roughly equal distribution of X- and M-class flares and aligning with a specific AR time frame. These partitions are organized in a sequential manner, as shown in Figure 2. Figure 3 displays the class distribution across these partitions. The data set includes time series data from solar photospheric vector magnetograms and NOAA's flare history, sourced from the SHARP. Each MVTS entry in SWAN-SF comprises 24 time series of magnetic field parameters from the ARs, as listed in Table 1. These series are recorded at 12 minute intervals over 12 hr, totaling 60 time steps. In this paper, $T = 60$ denotes the time steps, and $N = 24$ the magnetic field parameters. Our experiment involves binary classification within MVTS data to distinguish between major-flaring (classes X and M) and minor-flaring (classes C, B, and FQ) ARs.

2.2. Missing Values in SWAN-SF

In the realm of time series classification, particularly in the context of solar flare prediction using SWAN-SF MVTS data, the significance of missing value imputation is paramount. This process is vital as it directly influences the accuracy and reliability of subsequent analyses. Missing values, often resulting from sensor malfunctions, data transmission errors, or human oversight, pose a considerable challenge, especially in data sets such as SWAN-SF, which contains over 10 million missing values across its five partitions. As outlined in Table 2, the distribution and volume of missing values vary for different attributes, with some, such as the R-value, exhibiting a significantly higher number of not a number (NaN) values. In machine learning, missing values can severely compromise model accuracy, distorting the data distribution and structure, and leading to flawed pattern recognition and biased training (T. Emmanuel et al. 2021). This issue is exacerbated in time series classification, where sequences are categorized based on time-dependent patterns, necessitating data continuity and completeness. Classification algorithms, such as SVM and neural networks, generally require complete data sets; missing values, if not adequately addressed, can skew model outcomes, resulting in inaccurate classifications (T. Emmanuel et al. 2021).

Table 2
Missing Value Distribution for AR Magnetic Field Parameters in the SWAN-SF Data Set

Abbreviation	Partition 1	Partition 2	Partition 3	Partition 4	Partition 5
R_VALUE	2,399,220	2,934,918	1,361,095	1,748,394	2,755,911
SHRGT45	81,406	134,585	84,113	103,943	185,217
TOTBSQ	652	93,300	2718	4844	4964
TOTFX	652	93,300	2718	4844	4964
TOTFY	652	93,300	2718	4844	4964
TOTFZ	652	93,300	2718	4844	4964
TOTPOT	652	93,300	2718	4844	4964
TOTUSJH	652	93,300	2718	4844	4964
TOTUSJZ	652	93,300	2718	4844	4964
USFLUX	652	93,300	2718	4844	4964
ABSNJZH	652	93,300	2718	4844	4964
SAVNCPP	652	93,300	2,725	4844	4964

Note. For the attributes that are not listed in this table, there are zero missing values.

The choice between imputation methods should be guided by the specific characteristics of a data set and the objectives of the analysis. For instance, in time series data, where temporal patterns and correlations are significant, k-NN might be more appropriate. However, mean imputation might suffice in cases where data is randomly missing and lacks intricate patterns (D. P. Anil Jadhav & K. Ramanathan 2019). In dealing with imbalanced data sets such as SWAN-SF, where the major-flaring class may have limited samples, the handling of missing values becomes a critical task. The potential loss of valuable information represents a significant concern, particularly given that there may only be approximately 1000 samples for the major-flaring class per partition. Ignoring or deleting these samples with missing values could lead to a substantial reduction in the already scarce data, adversely affecting the model's ability to learn and predict accurately. Therefore, it is crucial to employ effective imputation techniques that can preserve and utilize every possible instance of the data set. This approach not only maintains the integrity of the data set but also ensures that the predictive model is trained on the most comprehensive data available, enhancing its performance and reliability in forecasting solar flare events.

2.3. Multiscaled Features in SWAN-SF

Normalization is a critical preprocessing step in machine learning and data analysis, crucial for managing data sets that feature attributes of different scales and distributions. This technique adjusts the data features to a uniform scale, minimizing the influence of outliers and skewed data. It guarantees that each attribute equally influences the analysis, boosting the accuracy and efficiency of models, especially those sensitive to data scale. Figure 4 shows the minimum and maximum values of each attribute across all the partitions of the SWAN-SF data set. It demonstrates the heterogeneous scales of these values and the complexity involved in normalizing them. Although the concept seems simple, its importance is often overlooked. There are various normalization methods, including linear, nonlinear, and data-specific transformations, which can be applied either globally or locally. Global normalization uses the data set's overall statistics by adjusting based on the minimum and maximum values of each feature. In contrast, local normalization focuses on adjusting for local extremes within subsets of data, treating each part distinctly. Beyond just scaling, normalization tackles issues such as skewness in data distribution, which can introduce bias into machine learning models. Skewed data can lead to inaccurate predictions by overemphasizing certain parts of the data. Normalization helps in differentiating attribute values across different classes, simplifying classification tasks for models. Relying on a single method, such as min–max or Z-score normalization, may not suffice for complex data types such as MVTS, where each time series has unique characteristics. Table 3 identifies the attributes of the SWAN-SF data set that are not ideal for normalization using the Z-score technique. These attributes are also unsuitable for min–max normalization, as neither method effectively addresses left or right skewness. Advanced techniques that are tailored to the specific attributes of the data can normalize more effectively. Effective normalization is essential, as it enables a machine learning model to accurately interpret the inherent relationships among variables rather than being skewed by their disparate scales. This critical procedure substantially impacts the performance of machine learning models in data analysis tasks.

2.4. Class Overlap in SWAN-SF

Class overlap in data sets occurs when different categories or groups within the data exhibit similar or identical characteristics, making it challenging to distinguish them accurately. This issue is common in real-world data where the boundaries between classes are not always clear. The impact of class overlap on classification performance is significant. When classes significantly overlap, machine learning models have difficulty classifying new instances accurately, leading to a decrease in both accuracy and stability. This happens because the model struggles to find an effective decision boundary that can separate the classes. Consequently, the model may either

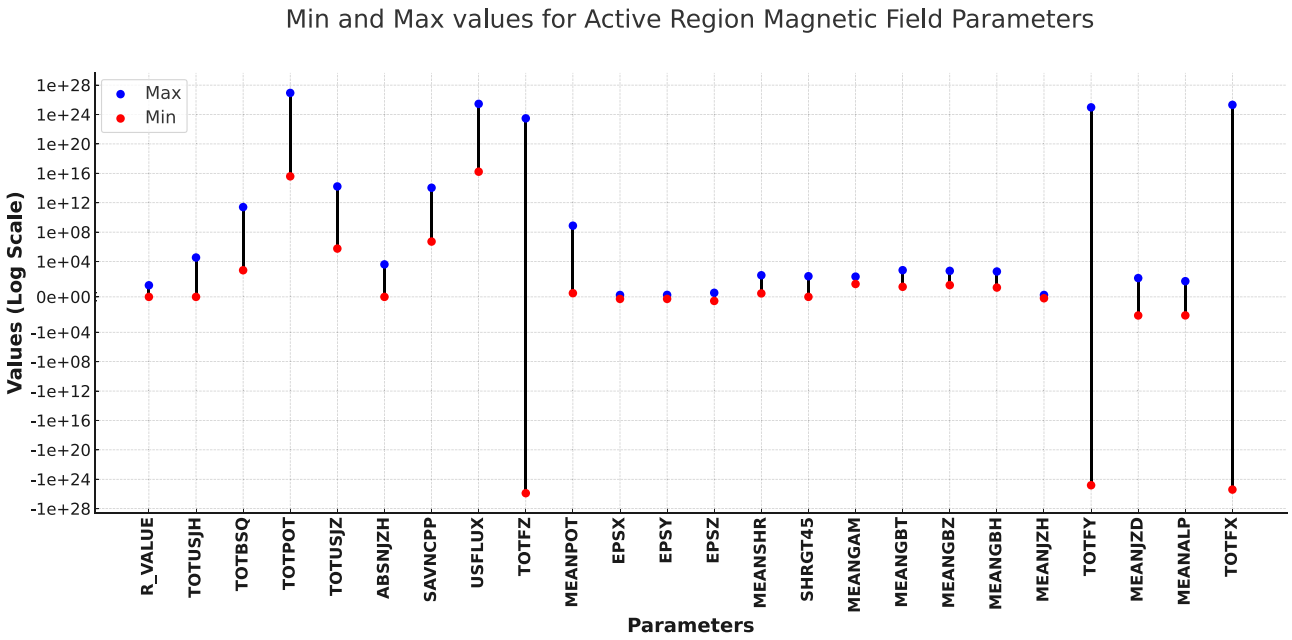

Figure 4. This plot demonstrates the significant variation in minimum and maximum values across different attributes of the SWAN-SF data set. The diverse scales underscore the need for an advanced normalization technique to effectively analyze and process the data.

overfit to the training data, excessively adjust to the intricacies of the overlap, or exhibit suboptimal performance due to insufficient generalization capacity, otherwise known as underfitting. To mitigate the effects of class overlap, methods such as NDBSR can be employed to improve the model's ability to distinguish between overlapping classes. Figure 5 illustrates the decision boundary before and after the removal of class overlap. The elimination of class overlap leads to a more precise decision boundary, thereby enhancing classification performance on test data.

Since the mid-1970s, the detection and categorization of X-ray flares have been conducted by NOAA's GOES within the 1–8 Å wavelength spectrum. Flares are assessed on a logarithmic scale from A to X, indicating ascending strength, with the minimum threshold value set at 10^{-8} W m^{-2}. An X-class flare exceeds an M-class flare in peak flux by an order of magnitude and is a hundredfold more intense than a C-class flare. Each class includes a subdivision ranging from 1.0 to 9.9, providing a finer granularity of classification. Based on this categorization, the M and C classes exhibit a high degree of similarity, with many samples from these classes being nearly indistinguishable. This significant overlap between the two classes poses a substantial challenge for any classifier tasked with differentiating between them. This overlap significantly contributes to the difficulty in surpassing a high true skill statistic (TSS) score in the binary classification of flares in the SWAN-SF data set.

2.5. Class Imbalance in SWAN-SF

We identify a data set as class imbalance when one or more of its class populations are substantially smaller than those of the majority classes. These smaller groups are termed minority instances, or positive instances, in forecasting contexts. On the other hand, the larger class groups are known as majority or negative instances, especially when the forecasting is focused on the relative importance of minority instances. An example of this is seen in the SWAN-SF benchmark data set, where GOES M- and X-class flares are used as the minority class, as shown in Figure 3. However, its significance is sometimes not fully recognized in complex tasks of forecasting, leading to inconsistencies in the performance of models. The problem of class imbalance affects classification models that aim to minimize the overall misclassification. Due to the prevalence of the majority class, an optimal classification boundary, such as that created by an SVM classifier, is often biased. This bias can lead to misclassifying many instances of the minority class, resulting in a disproportionate number of false negatives (FNs) for the minority class and a reduced number of true positives (TPs). This indicates a tendency for models in imbalanced scenarios to favor the majority class, which is a significant concern in fields focusing on minority instances, such as flare forecasting. The issue also impacts the selection and effectiveness of performance metrics such as accuracy, precision (excluding recall), and the F1-score. A model that erroneously classifies all instances as the majority class might appear highly accurate but offers little insight into the minority class.

3. Data Preprocessing and Sampling Methodologies

3.1. Missing Value Treatment and Feature Scaling

The treatment of missing values and the scaling of features are critical elements in the preprocessing stages of time series classification, ensuring that the data is adequately prepared for analysis. It is essential to recognize that most machine learning models require a complete data set and inherently lack the capability to process missing data (T. Emmanuel et al. 2021). The choice of technique for dealing with missing values, whether it be imputation or deletion, is crucial and depends on the nature of the data and the pattern of missing values. Effective handling of missing values ensures the model's robustness and its applicability in real-world scenarios where data imperfections are common (T. Emmanuel et al. 2021). On the other hand, feature scaling (normalization) brings all variables to a common scale, which is crucial since different features often exist on varying scales and units. This uniformity

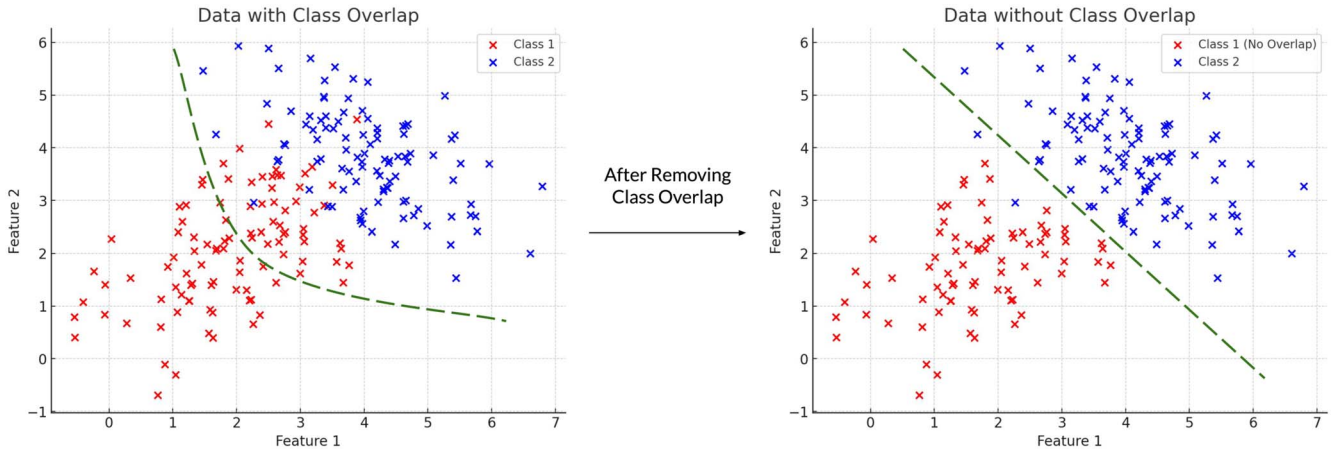

Figure 5. These scatter plots illustrate the distribution of two classes and their overlap. The dashed line highlights a decision boundary built by a classifier. It demonstrates the impact of class overlap on the complexity and effectiveness of the decision boundary and therefore the potential for higher classification performance on test data.

Table 3
Optimal Normalization Techniques for SWAN-SF Data Set's Attributes, the Best Method for Each, and an Overview of Their Characteristics

Attribute	Best Method	Right Skewness	Left Skewness
R_VALUE	Z-score	⋯	⋯
TOTUSJH	Box–Cox	Yes	⋯
TOTBSQ	Log	Yes	⋯
TOTPOT	Log	Yes	⋯
TOTUSJZ	Log	Yes	⋯
ABSNJZH	Box–Cox	Yes	⋯
SAVNCPP	Log	Yes	⋯
USFLUX	Log	Yes	⋯
TOTFZ	Sqrt	⋯	Yes
MEANPOT	Log	Yes	⋯
EPSX	Z-score	⋯	⋯
EPSY	Z-score	⋯	⋯
EPSZ	Z-score	⋯	⋯
MEANSHR	Box–Cox	Yes	⋯
SHRGT45	Box–Cox	Yes	⋯
MEANGAM	Box–Cox	Yes	⋯
MEANGBT	Z-score	⋯	⋯
MEANGBZ	Z-score	⋯	⋯
MEANGBH	Z-score	⋯	⋯
MEANJZH	Z-score	⋯	⋯
TOTFY	Sqrt	Yes	Yes
MEANJZD	Box–Cox	Yes	⋯
MEANALP	Z-score	⋯	⋯
TOTFX	Z-score	⋯	⋯

Note. These include right and left skewness.

is vital for algorithms, especially those relying on distance calculations, such as SVM, k-NN, or even neural networks, as it accelerates convergence. Without scaling, certain features with larger ranges can dominate the model, leading to a skewed influence and potentially inaccurate predictions (D. Singh & B. Singh 2020). Moreover, for algorithms utilizing gradient descent optimization, such as linear and logistic regression and neural networks, feature scaling can significantly expedite the convergence process, making model training more efficient and effective.

In the realm of MVTS classification, especially in predicting solar flares, the importance of these processes is amplified. MVTS data typically consists of various features with distinct scales. Feature scaling is essential in this context to ensure that these varying scales do not negatively impact the model's capability to identify underlying patterns. Likewise, time series data may contain gaps resulting from issues such as sensor malfunctions. Addressing these gaps appropriately is vital for preserving the temporal continuity and precision of the analysis. Therefore, both feature scaling and missing value treatment are crucial in preparing data for machine learning models. They help in standardizing the data, ensuring algorithms work efficiently and accurately, and making models robust enough to handle real-world data with its inherent imperfections and complexities.

3.1.1. Missing Value Imputation

To address missing values in time series data, we typically have two main options: either removing the samples with missing values or imputing them (T. Emmanuel et al. 2021). In specific contexts such as solar flare prediction, where data sets often encounter issues such as class imbalance, retaining samples is recommended. This is because doing so may lead to the loss of critical data, potentially from the minority class. Furthermore, every sample represents valuable real-world data, and its removal could diminish the data set's integrity. Imputation, therefore, emerges as a preferable strategy, aiming to create a more comprehensive data set for the benefit of future research. When it comes to imputing missing values in an MVTS, the choice of technique is crucial, yet there is no one-size-fits-all solution. For instance, mean imputation might be a convenient approach for filling one or two consecutive missing

time stamps in a univariate time series attribute. However, its applicability reduces significantly in scenarios where multiple consecutive time stamps are missing. In such cases, mean imputation might fail to generate realistic values for these gaps. Therefore, a more nuanced strategy involves employing a primary missing value imputation technique for general use, supplemented by alternative methods tailored for specific, less common scenarios. This approach acknowledges the complexity and variability of time series data. The selection of the primary imputation technique is critical, as it significantly influences the overall quality and reliability of the imputed data. Making an informed choice requires a careful consideration of the data set's unique characteristics and the specific challenges posed by its missing values. Determining the superiority of one missing value imputation technique over another for time series data requires a holistic evaluation of several factors.

The chosen method must preserve the inherent characteristics of the time series, such as seasonality, trend, and autocorrelation. This preservation is crucial for maintaining the integrity and interpretability of the time series data. Moreover, the scalability and computational efficiency of the method play a significant role, especially in handling large data sets. Techniques that offer a balance between imputation quality and computational demand are often preferred in practical applications. In the case of MVTS, the ability to handle interdependencies and correlations between different variables is paramount. Techniques that can effectively leverage information from related variables tend to produce more accurate imputation. Additionally, the method's performance should not be overly sensitive to the proportion of missing data. A robust technique maintains its effectiveness even with a high percentage of missing data. Practical considerations, such as ease of implementation and integration into existing data processing pipelines, also influence the choice of technique. In many cases, the simplicity and usability of a method can be as important as its statistical performance. Furthermore, the flexibility and customizability of the method to adjust to specific data set characteristics can offer a significant advantage, allowing for more tailored and effective imputation. In summary, the selection of the best imputation technique for a given time series data set is a multifaceted decision, balancing statistical properties, data set characteristics, and practical considerations. It often involves a trade-off between various factors, including accuracy, robustness, efficiency, and usability.

3.1.1.1. FPCKNN Imputation

In the SWAN-SF data set, there are over 10 million missing values in total. Table 2 displays the count of missing values for each attribute and for each partition of the data set. k-NN imputation effectively handles a variety of missing value patterns, including sequences of consecutive missing values. It imputes values based on the most similar samples, referred to as the nearest neighbors (O. Troyanskaya et al. 2001). However, to fully address the issue of missing values, augmenting k-NN imputation with additional concepts is crucial in formulating an accurate algorithm. In FPCKNN imputation, the Pearson correlation coefficient (PCC) is used for identifying the nearest neighbors, specifically finding the two most similar samples for imputation. The missing values are then imputed using data from these nearest neighbors. The algorithm typically calculates the mean of the values from these "k" neighbors for numerical variables, and this calculated mean is used to replace the missing data. In scenarios where consecutive missing values occur in time series data, the k-NN imputation method can sequentially impute these values. Each missing value is replaced by the mean of the corresponding time stamps from the k-NN. This approach ensures the preservation of the time series' temporal dynamics.

In an MVTS data set such as SWAN-SF, there are two main types of missing values. The first type occurs when all the values of an attribute within an MVTS instance are missing. In such instances, we simply replicate all values from the corresponding attribute of the most similar sample that does not have missing values. The second type arises when only some values of an attribute in an MVTS are missing. In these cases, we identify the most similar MVTS samples to the ones with missing values. In other words, we perform one nearest neighbor imputation to accurately impute the best values for the missing data. For the similarity measure, we initially substitute the NaN values with zero values and then employ the PCC to calculate the similarity between two MVTS. The PCC is a robust method for this purpose, owing to its efficacy in capturing linear relationships between variables. It quantifies the degree of correlation between variables, offering valuable insights, particularly in time series analysis where understanding these dynamics is crucial. A major advantage of the PCC is its ability to normalize correlation values, ensuring they always fall between -1 and 1. This scaling enables an intuitive interpretation and comparison of the strength of relationships, irrespective of the scale of the variables involved. Additionally, the mathematical simplicity and computational efficiency of PCC are advantageous, especially in handling large data sets common in time series analysis. One of the critical attributes of PCC is its insensitivity to the scale and location changes in the time series data, ensuring its consistent applicability across various data sets. The computation of the PCC between two MVTS X and Y is detailed in Algorithm 1, providing a systematic approach to this analysis.

To reduce the computational load in identifying the most similar sample for imputation, we employ a heuristic based on the classification system used for solar flares. This method classifies solar flares based on their peak soft X-ray flux into five categories: A, B, C, M, and X, each indicating increasingly stronger flares. An X-class flare, for instance, is about 10 times more intense in peak flux than an M-class flare and 100 times more than a C-class flare. Additionally, within each category, flares are further ranked on a scale from 1.0 to 9.9, which specifies the intensity of the X-ray flux. For example, an X9.9 flare represents the maximum intensity within the X class, whereas X1.0 is considered the minimum, followed closely by M9.9 as the next most intense category. Since X1.0 flares and M9.9 flares are comparable in terms of their X-ray flux, this classification system effectively narrows the focus for identifying the most comparable samples. The heuristic is outlined as follows:

1. Initiate the imputation procedure with the most intense flare sample, X9.9, and advance sequentially to X1.0. Subsequently, proceed from M9.0 to the least intense flare samples, finishing with FQ 1.0.
2. If missing values are encountered, identify the most similar sample by calculating the PCC with the 50 previous samples. Since there is a logical order and

samples close to each other are more likely to be similar in terms of category and X-ray flux, the highest PCC score among these 50 samples will likely identify the most similar sample. These 50 samples are either imputed or free of missing values.
3. If there are fewer than 50 previously imputed samples available, calculate the PCC with all available imputed samples.
4. If there are fewer than 10 previous samples, calculate the PCC using the next 10 samples (instead of the previous ones), even if they contain missing values. In such cases, NaN values should be substituted with zeros solely for the purpose of calculation.

By applying this heuristic, we limit PCC calculations to the 50 previous samples, rather than all 60,000 samples in the first partition of the SWAN-SF data set. This approach significantly reduces computational effort. In our FPCKNN imputation, we address two types of missing values that can occur in an attribute of an MVTS sample. k-NN imputation, our primary technique, is capable of imputing multiple consecutive missing values effectively. For rare cases where all values of an attribute are NaN, we employ the "replication" method. Additionally, we use the PCC technique to identify similar samples, which is both fast and robust. Lastly, we implement a heuristic to narrow down the search space for finding similar samples, further optimizing the process. In Algorithm 2, the FPCKNN imputation is described.

3.1.2. Data Normalization

Data normalization is an essential step in data preprocessing, particularly in the context of machine learning and data analysis. Its importance arises from the fact that data sets often contain attributes with varying scales and units. When different attributes are measured on different scales, it can lead to challenges in analysis, as some algorithms might incorrectly interpret the significance of certain features based on their scale. This can adversely affect the model's performance, as larger-scale attributes might dominate the outcome, while smaller-scale attributes might not contribute significantly, regardless of their actual importance in the data set (D. Singh & B. Singh 2020). Skewness, on the other hand, refers to the degree of asymmetry observed in the data distribution of a single attribute. It is an entirely separate aspect of data characteristics and is not a direct consequence of varying scales and units across different attributes. Skewness can affect the performance of many machine learning algorithms, especially those that assume a normal distribution of the input data (D. Singh & B. Singh 2020).

Algorithm 1. PCC Calculation for Two MVTS

Require: two MVTS X and Y (X, $Y \in \mathbb{R}^{(T\times N)}$)
Ensure: PCC between X and Y
$FlattenedX \leftarrow$ Flatten(Transpose(X))
$FlattenedY \leftarrow$ Flatten(Transpose(Y))
$meanX \leftarrow$ mean($FlattenedX$)
$meanY \leftarrow$ mean($FlattenedY$)
$stdX \leftarrow$ standard_deviation($FlattenedX$)
$stdY \leftarrow$ standard_deviation($FlattenedY$)
$covariance \leftarrow$ mean(($FlattenedX - meanX$) $\times$ ($FlattenedY - meanY$))
$CorrCoef \leftarrow \frac{covariance}{stdX \times stdY}$
return $CorrCoef$

In the context of the SWAN-SF data set, which is an MVTS data set with diverse attributes, normalization becomes even more crucial. Each attribute might require a different normalization approach to ensure that all the attributes contribute equally to the analysis and that the model's performance is not biased toward certain features simply because of their scale. This tailored approach to normalization helps in better capturing the underlying patterns in the data, leading to improved classification performance. The following section will explore various normalization techniques that are particularly beneficial for handling data sets, such as SWAN-SF. These techniques address the challenges posed by different scales and distributions in the data, ensuring that each attribute is processed in a way that optimizes its contribution to the overall analysis and model performance. Additionally, some of these techniques are specifically designed to address various types of skewness in the data.

1. *Log normalization*. Primarily used for data with right skewness (C. Feng et al. 2014). It transforms data into a more normally distributed data set,

$$x' = \log_b(x + c). \tag{1}$$

Here, x' is the transformed value, x is the original value, b is the logarithm base (typically 10, e, or 2), and c is a constant added to ensure all values are positive, necessary for data sets with zero or negative values.

2. *Square-root normalization*. Effective for reducing left skewness in data (P. Muhammad Ali & R. Faraj 2014).

$$x' = \sqrt{x + c}. \tag{2}$$

In this formula, x' is the transformed value, and x is the original value. A constant c is added to each value in the data set to ensure positivity, especially if the data set includes negative values.

3. *Box–Cox normalization (R. M. Sakia 1992)*. A more versatile technique, suitable for both right- and left-skewed data, designed to stabilize variance and make data more closely resemble a Gaussian distribution.

$$x'(\lambda) = \begin{cases} \dfrac{(x + c)^{\lambda} - 1}{\lambda} & \text{if } \lambda \neq 0, \\ \log(x + c) & \text{if } \lambda = 0. \end{cases} \tag{3}$$

In this formula, $x'(\lambda)$ represents the transformed value, x is the original value, and c is a constant added to ensure that $x+c$ is positive and nonzero. The parameter λ is known as the transformation parameter, which dictates the nature of the transformation applied to the data. The choice of λ is crucial for the effectiveness of the transformation:

1. When $\lambda = 0$, the transformation becomes a logarithmic transformation.
2. As λ approaches 1, the transformation becomes closer to the identity transformation, where no change is applied to the data.
3. Different values of λ can be used to address varying degrees and directions of skewness in the data.

Selecting the optimal value of λ usually involves finding the value that maximizes the log-likelihood of the transformed data fitting a normal distribution, often facilitated by statistical software.

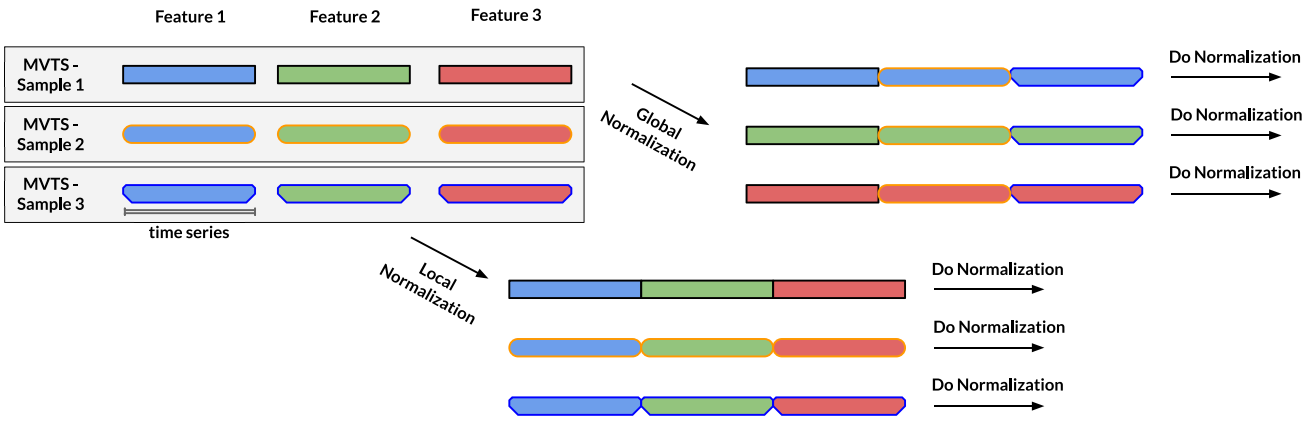

Figure 6. This figure demonstrates the application of both global and local normalization techniques on an MVTS data set. Global normalization is employed to preserve the relative magnitudes of time stamps across different time series, ensuring that the values are comparable between samples. This is particularly crucial in the SWAN-SF data set, where the magnitudes of values within each time series hold significant importance. Conversely, local normalization is used when the temporal dynamics or the structural patterns of a time series are more critical than the actual magnitudes of the values. This method emphasizes the importance of the time series' shape and trends over their absolute values.

Algorithm 2. FPCKNN Imputation

Require: data set *data*, number of partitions *numParts*, number of attributes *numAttrs*, number of time stamps *numTimes*, comparison limit *kMax*
Ensure: imputed *data*
for *partIdx* = 1 **to** *numParts* **do**
  partition ← *data*[*partIdx*] {The shape of partition is (*numTimes*, *numAttrs*, numSamples)}
  for *sampleIdx* = 1 **to** shape(*partition*)[2] **do**
    currSample ← *partition*[:,:,*sampleIdx*]
    Fill in all the missing values in *currSample* with NaN
    Initialize list *missingIdx*
    if NaN exists in *currSample* **then**
      compCount ← min(*sampleIdx*, *kMax*)
      Initialize array *corrCoeff* of size *compCount* with -2.0
      sampleX ← replace NaN with 0 in *currSample*
      for *compIdx* = 1 **to** *compCount* **do**
        sampleY ← *partition*[:,:,*sampleIdx* − *compIdx*]
        corrCoeff [*compIdx*]← calcPearson(*sampleX*[:,:], *sampleY*[:,:])
      end for
      for *attrIdx* = 1 **to** *numAttrs* **do**
        if all values in *currSample*[:,*attrIdx*] are NaN **then**
          Duplicate the entire values of the attribute *attrIdx* to *currSample*[:,*attrIdx*] from the sample that has the highest PCC score in *corrCoeff*
        else
          Perform k-NN imputation for *currSample*[:,*attrIdx*] based on the samples that have the highest PCC scores in *corrCoeff*
        end if
      end for
    end if
    partition[:,:,*sampleIdx*] ← *currSample*
  end for
  data[*partIdx*] ← *partition*
end for
return *data*

4. *Z-score normalization* (*standardization*). Used for normalizing the distribution of values in a data set, particularly effective for data without skewness.

$$z = \frac{x - \mu}{\sigma}, \tag{4}$$

where z is the standardized value, x is the original value, μ is the mean, and σ is the standard deviation. This method is suitable for any range of values and is often used to bring different variables to the same scale.

5. *Min–max normalization (min–max scaling)*. A simple method to rescale features to a standard range, usually between 0 and 1.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}, \tag{5}$$

where x' is the normalized value, x is the original value, $\min(x)$ and $\max(x)$ are the minimum and maximum values, respectively. This method is sensitive to outliers but is popular due to its simplicity.

Each technique has its specific application, chosen based on data characteristics and analysis requirements.

3.1.2.1. LSBZM Normalization

The SWAN-SF data set, being an MVTS data set, presents a diverse array of attributes, each exhibiting distinct characteristics. A notable aspect of these attributes is their varying scales; some attributes are confined within a small range, fluctuating between −1 and 1, while others exceed 100,000, displaying tendencies of either right or left skewness. The skewness and magnitude of these values necessitate a tailored approach for each attribute, achievable through conditional programming techniques. To effectively normalize the data, it is imperative to uniformly apply the normalization process across the corresponding attributes of all samples. Figure 6 clarifies our methodology for normalizing an MVTS data set, a critical step especially when calculating the mean or standard deviation of the values, such as the implementation of the Z-score normalization technique.

In our LSBZM normalization, the criteria for selecting an appropriate normalization technique for each attribute include its magnitude and skewness. For the primary normalization technique, our choices are varied and depend on the data's characteristics. For attributes with right skewness, the logarithmic transformation is applied. In contrast, for left-skewed data, the square root transformation is preferable. The Box–Cox transformation, effective for both right and left skewness, is most suitable when dealing with values that are not extremely small or large. In cases where skewness is absent, the Z-score

normalization is utilized. Following the primary normalization, we employ the min–max normalization technique to ensure all attribute values range between 0 and 1. This scaling is particularly crucial for certain machine learning techniques, including neural networks. It is crucial to note that min–max normalization does not alter the data distribution; it merely adjusts the range of the values. Consequently, this preserves the integrity and impact of the primary normalization technique employed. Algorithm 3 in this paper provides a detailed description of these conditions and the corresponding normalization technique.

Using the LSBZM normalization technique, we have gathered essential information on the data set's 24 attributes (photospheric magnetic field parameters). This process helped us identify the most effective normalization method for each attribute, as detailed in Table 3. After applying Algorithm 3 to the SWAN-SF data set, we obtained the information presented in Table 3. To assess the skewness of the time series data, we calculate the skewness value using statistical methods. Skewness, mathematically, measures the asymmetry of the probability distribution of a real-valued random variable around its mean. As indicated in line eight of Algorithm 3, which presents the formula for skewness calculation, if an attribute's skewness value is between -1 and 1, it signifies minimal skewness. In such cases, we use Z-score normalization. For attributes with very small or very large values, we apply either Log normalization or square root normalization. Specifically, if the skewness indicates right skewness (value greater than 1), we use Log normalization; for left skewness (value less than -1), we use square root normalization. If an attribute that is neither excessively small nor large exhibits skewness, we use Box–Cox normalization. Consequently, Table 3 can be used to determine the optimal normalization approach for different attributes in the SWAN-SF data set without directly implementing the LSBZM technique. Additionally, the table identifies which attributes exhibit right skewness (skewness values greater than 1) or left skewness (skewness values less than -1).

3.2. NDBSR

In the realm of machine learning, particularly in classification tasks, the distinctiveness of classes plays a crucial role. The essence of a classification task lies in the algorithm's ability to accurately discern between different classes. When these classes exhibit significant similarities or overlap, it becomes challenging for machine learning techniques to identify and establish an effective decision boundary (J. Zhang et al. 2019). This boundary is fundamental, as it defines the criteria by which data points are categorized into one class or another. A lack of sufficient differentiation between classes often results in suboptimal classification performance. This is because the algorithm struggles to develop a clear and robust criterion for class separation, leading to increased instances of misclassification. In such scenarios, the decision boundary, rather than being a clear demarcation, tends to become ambiguous and less effective in its purpose, as shown in Figure 5.

To address this issue, techniques such as NDBSR are employed. These techniques focus on enhancing the distinctiveness of classes. NDBSR, in particular, involves the identification and removal of data points that are situated too close to the decision boundary (J. Zhang et al. 2019). These data points are typically those that the model finds most challenging to classify and are often the source of uncertainty. By removing or otherwise handling these near-boundary samples, NDBSR helps to create a more defined and discernible separation between classes. The SWAN-SF data set presents significant challenges for classification due to extensive overlap between classes, affecting both binary and multiclass classification. This complexity stems from the data set's structure, which includes five distinct classes. Each class is defined by a numerical value that represents the intensity of the flare, ranging from 1.0 to 9.9. As a result, some classes share notable similarities; for example, the X1.0 samples are quite similar to the M9.9 samples. However, this overlap is more complex than it appears at first glance. For instance, X1.0 samples might resemble samples ranging from M9.9 down to M2.0. Similarly, class-M samples can appear similar to many samples from classes B and C, leading to difficulties for a machine learning model in distinguishing these similarities. Therefore, due to the presence of similar samples from two classes regarding major-flaring (M and X) and minor-flaring (B, C, and FQ) instances, classifiers struggle to learn an accurate decision boundary. This results in lower TSS scores, even after applying proper imputation and normalization techniques.

In the realm of binary classification of SWAN-SF, a significant overlap is observed between the C- and M-class samples. Addressing this issue necessitates the exploration of multiple solutions. A primary strategy involves the selective removal of samples from the minor-flaring categories (namely, C, B, and FQ classes), as opposed to those from the major-flaring class (M. G. Bobra & S. Couvidat 2015). This approach is justified by the relatively scarce availability of samples in the major-flaring class, underscoring the importance of preserving these data points. There are two straightforward strategies to reduce class overlap in the binary classification of the SWAN-SF data set. First, we can eliminate class-C samples during training. This action helps resolve many similarities between classes M and C, making it easier for the classifier to successfully identify major-flaring samples (M and X) and distinguish between major-flaring and minor-flaring samples. Second, we can remove both the class-B and -C samples during training. This is because there might still be overlap between classes M and B, and removing only class C may not fully help the classifier to clearly separate the two classes. As demonstrated in Figure 7, this objective can be achieved by removing the minor-flaring class samples postimputation and prior to any further preprocessing steps. As a result, by keeping all major-flaring samples and achieving a high recall score while reducing the false positive rate (FPR), we can improve the TSS scores.

3.3. Sampling Techniques

To address the issue of data imbalance, it is essential to generate additional synthetic samples for the underrepresented minority class. In our binary classification task using the SWAN-SF data set, the minority class comprises X and M flares, whereas the majority class includes C, B, and FQ classes. Generating more synthetic samples will help achieve a more balanced data set. When selecting an oversampling technique for our time series data, two critical factors must be considered. First, the technique must maintain the temporal dynamics inherent in the samples. Second, it should generate samples across the entire distribution range of the minority class. Techniques that are genuinely generative, capable of creating new samples, are preferable. These are more beneficial

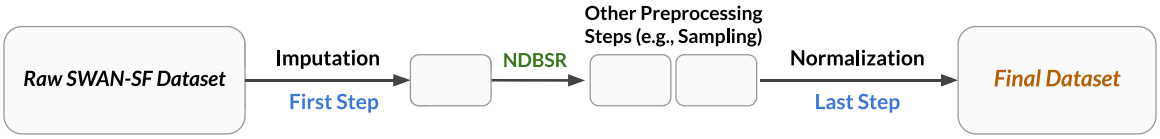

Figure 7. The diagram illustrates the proper placement of NDBSR within a data preprocessing pipeline.

compared to methods that merely perform random sampling from the original data without the generative aspect (J. Yoon et al. 2019).

We propose two distinct strategies to address the issue of imbalanced classification. The initial approach concentrates exclusively on utilizing oversampling techniques to generate additional synthetic samples for the minority class, targeting a 1:1 imbalance ratio, where there are approximately equal numbers of samples for each class. However, having over 100,000 samples for a binary classification task could lead to extended training time. To mitigate this, our second approach combines both oversampling and undersampling techniques. In this approach, we reduce the number of samples from the majority class while also generating fewer synthetic samples for the minority class than in the first approach. This balanced strategy seeks to mitigate the challenges associated with a large number of samples while effectively tackling the issue of imbalance in the data set.

3.3.1. Oversampling Techniques

In our oversampling approach, we utilize techniques such as the Synthetic Minority Oversampling Technique (SMOTE; N. V. Chawla et al. 2002), adaptive synthetic sampling (ADASYN; H. He et al. 2008), and Gaussian noise injection (GNI). These methods are perturbation based, meaning they do not generate new data in the true sense but create synthetic samples through random variations of the original samples. According to P. Hosseinzadeh et al. (2024), these techniques are effective for time series generation. However, we also explore the time series generative adversarial networks (TimeGAN; J. Yoon et al. 2019), an advanced deep learning-based data augmentation technique for MVTS data. Distinct from the earlier mentioned techniques, TimeGAN has the capability to generate completely new data, offering a novel and potentially more effective solution for our needs. Additionally, it uniquely preserves the temporal dynamics of the attributes.

3.3.1.1. SMOTE

SMOTE (N. V. Chawla et al. 2002) is a statistical technique used to increase the number of instances in a data set in a balanced way. It is primarily used in the context of imbalanced data sets, where the number of instances for one class (often the minority class) is much less than those for other classes.

The core idea of SMOTE is to create synthetic samples from the minority class instead of creating copies. This is done as follows:

1. Randomly pick a sample from the minority class. Compute the k-NN for this sample. The neighbors are chosen from the minority class only.
2. Randomly select one of these neighbors and compute the vector difference between this neighbor and the chosen sample.
3. Multiply this difference by a random number between 0 and 1, and add it to the chosen sample to create a new sample.

Algorithm 3. LSBZM Normalization

Require: data set *data*, number of partitions *numParts*, number of attributes *numAttrs*, number of time stamps *numTimes*
Ensure: normalized *data*
for *partIdx* = 1 **to** *numParts* **do**
  partition ← *data*[*partIdx*] {The shape of partition is (*numTimes*, *numAttrs*, numSamples)}
  for *attrIdx* = 1 **to** *numAttrs* **do**
    Initialize *attributeVector*, *normalizedVector* as empty lists
    theAttribute ← *partition*[:,*attrIdx*, :]
    attributeVector ← *Flatten*(*theAttribute*)
    Calculate min and max of *attributeVector*
    Calculate skewness of *attributeVector* using the formula: skewness $= \frac{N\sum_{i=1}^{N}(X_i-\bar{X})^3}{(N-1)(N-2)(\sum_{i=1}^{N}(X_i-\bar{X})^2)^{3/2}}$, where N is the number of samples, X_i is each individual sample, and $\bar{X}$ is the mean of *attributeVector*
    if (max−min>100,000) **then**
      if skewness > 1 **then**
        attributeVector ← Log(*attributeVector*)
      else if skewness < −1 **then**
        attributeVector ← Sqrt(*attributeVector*)
      else
        attributeVector ← Zscore(*attributeVector*)
      end if
      normalizedVector ← MinMaxScale(*attributeVector*)
    else if (max<1 and min>−1) **then**
      Apply a similar normalization logic based on the previously mentioned conditions.
    else
      if skewness > 1 or skewness < −1 **then**
        attributeVector ← BoxCox(*attributeVector*)
      else
        attributeVector ← Zscore(*attributeVector*)
      end if
      normalizedVector ← MinMaxScale(*attributeVector*)
    end if
    for *sampleIdx* = 1 **to** shape(*partition*)[2] **do**
      Update the values of the attribute *attrIdx* for all the samples in *partition* using the *normalizedVector* values
    end for
  end for
  data[*partIdx*] ← *partition*
end for
return *data*

This process can be expressed in the following formula, where x_i is a randomly chosen minority-class sample, x_{zi} is one of its k-NN, λ is a random number between 0 and 1, and x_i' is the new sample:

$$x_i' = x_i + \lambda \times (x_{zi} - x_i). \quad (6)$$

This method effectively forces the decision region of the minority class to become more general. It helps in overcoming the problem of overfitting, which is common when using simple random oversampling. SMOTE does not change the number of majority-class instances but augments the data set by adding more examples from the minority class, thus balancing the class distribution.

3.3.1.2. ADASYN

ADASYN (H. He et al. 2008) represents a sophisticated oversampling methodology designed to address imbalanced data sets analogous to SMOTE. The technique focuses on generating synthetic data by using a weighted distribution that prioritizes minority-class examples based on their learning difficulty. Consequently, it produces more synthetic data for the minority-class examples that are more challenging to learn.

ADASYN works as follows:

1. *Calculate the class imbalance ratio*. Determine the degree of imbalance by calculating the ratio of the number of instances in the minority class to the majority class.
2. *Compute the Euclidean distance and k-NN*. For each sample in the minority class, compute the Euclidean distance to find its k-NN. The neighbors are chosen from the entire training set, not just the minority class.
3. *Determine the number of synthetic samples to generate for each minority sample*. This is based on the density of majority-class examples around each minority-class example. More synthetic examples are generated for minority examples that have more majority-class neighbors.
4. *Generate synthetic samples*. Similar to SMOTE, for each minority-class sample, synthetic samples are generated by linearly interpolating between the minority-class sample and its nearest neighbors. The number of samples generated is proportionate to the level of difficulty in learning that particular minority-class example.

The key formula in ADASYN is the calculation of the number of synthetic samples to be generated for each minority sample, represented by G_i, where G is the total number of synthetic samples to generate, and Δ_i is the ratio of majority-class neighbors among the k-NN of the ith minority-class sample:

$$G_i = G \times \frac{\Delta_i}{\sum_i \Delta_i}. \tag{7}$$

This technique adapts to the intrinsic distribution of the minority class and focuses on the samples that are more difficult to classify, thereby improving the learning behavior of the classifier. Unlike SMOTE, which treats all minority-class samples equally, ADASYN shifts the focus toward the regions where the class imbalance is more pronounced.

3.3.1.3. GNI

GNI is an oversampling technique used for addressing imbalanced data sets. It involves augmenting the minority class by adding small variations, or noise, to existing samples. This method relies on injecting random noise, following a Gaussian distribution, into the data.

The process can be described as follows:

1. *Determine the standard deviation* σ. Calculate the standard deviation of the data set (or a subset of features), denoted as σ. This measures the variation or dispersion from the average value.
2. *Set a noise proportion* α. Define α, a user-determined parameter that specifies the proportion of noise to be added. It is a small fraction used to scale the standard deviation.
3. *Calculate the noise level* λ. The noise level, λ, is obtained by multiplying the standard deviation σ with the noise proportion α. This determines the scale of the Gaussian noise to be added, i.e., $\lambda = \sigma \times \alpha$.
4. *Generate Gaussian noise*. Generate Gaussian noise, η, with a mean of 0 and a standard deviation equal to the noise level λ, for each sample in the data set. Thus, $\eta \sim \mathcal{N}(0, \lambda^2)$.
5. *Create new samples*. The generated noise η is added to the existing samples x to create new, slightly altered samples x'.

The key formula for GNI is

$$x' = x + \eta. \tag{8}$$

This technique effectively creates more diverse training data, preventing overfitting by providing variations of the training samples and enabling the model to generalize better. GNI is favored for its simplicity and effectiveness, particularly in scenarios where more complex oversampling techniques may not be necessary.

3.3.1.4. TimeGAN

TimeGAN (J. Yoon et al. 2019) is a sophisticated method for generating synthetic time series data. It aims to capture the complex temporal dynamics inherent in time series data, making it particularly useful for dealing with imbalanced time series data sets.

The process of TimeGAN can be mathematically described as follows:

1. *Embedding function (e)*. This function maps the original time series data x to a latent space Z. The embedding function is given by $e: X \to Z$.
2. *Recovery function (r)*. This is the inverse of the embedding function, mapping the latent representation back to the data space, represented by $r: Z \to X$.
3. *Sequence generator (G)*. The generator creates synthetic data in the latent space from a noise vector ϵ, given by G: $\epsilon \to \tilde{Z}$.
4. *Sequence discriminator (D)*. The discriminator differentiates between original and synthetic data, represented by a function D: $Z \cup \tilde{Z} \to \{0, 1\}$, where 0 indicates synthetic data and 1 indicates original data.
5. *Joint training scheme*. TimeGAN employs both supervised and unsupervised loss functions for training. The supervised loss ensures the model accurately captures the conditional distributions of the data, while the unsupervised loss (typical of GANs) aims to model the overall data distribution.

The key equations governing these components are:

1. *For the embedding function*. $h_t = e(x_t, h_{t-1})$, where h_t represents the latent representation at time t, and x_t is the original data at time t.
2. *For the recovery function*. $\tilde{x}_t = r(h_t)$ where $\tilde{x}_t$ is the recovered data at time t from the latent representation.
3. *For the generator*. $h_t^g = g(z_t, h_{t-1}^g)$ where h_t^g represents the generator's latent state at time t, and z_t is the input noise vector at time t.
4. *For the discriminator*. $D(h_t, h_t^g) = \begin{cases} 0 & \text{if sample is synthetic} \\ 1 & \text{if sample is original} \end{cases}$.

TimeGAN's uniqueness lies in its ability to effectively learn and replicate the temporal dynamics of time series data, which

is crucial for generating realistic synthetic sequences. This characteristic makes it highly suitable for augmenting time series instances, particularly in scenarios where additional data is required for effective model training.

3.3.2. Combination of Oversampling and Undersampling Techniques

Using only oversampling to achieve a balanced ratio of 1:1 might lead to an excessively large data set. This is particularly true for the SWAN-SF data set, where the majority class could exceed 60,000 samples, potentially resulting in a longer training duration. To mitigate this, we combine oversampling and undersampling techniques. Initially, we reduce the size of the majority class by employing random undersampling (RUS), which involves randomly removing samples. Given that the SWAN-SF data set is prone to class overlap, it is advisable to employ a class overlap removal method, such as Tomek links (TLs). Thus, following RUS, we apply TLs to eliminate samples near the decision boundary from the majority class. After these two stages of undersampling, we use one of the four discussed oversampling techniques to synthetically augment the minority class, aiming for a balanced 1:1 ratio. This approach results in a smaller overall data set, substantially reducing the likelihood of overfitting.

3.3.2.1. RUS

RUS is a technique used for handling imbalanced data sets, particularly focusing on the undersampling of the majority class. Unlike oversampling techniques that increase the size of the minority class, undersampling reduces the size of the majority class to balance the data set. The key idea of RUS is to randomly eliminate instances from the majority class to prevent its dominance when training a machine learning model. This method is straightforward and can be very effective, especially when the data set is sufficiently large, and the reduction does not lead to a significant loss of information. However, one must be cautious as undersampling can lead to the loss of potentially important information from the majority class.

3.3.2.2. TLs

TL (I. Tomek 1976) is a more nuanced undersampling technique used in the context of imbalanced data sets, specifically focusing on the elimination of instances from the majority class. Unlike RUS, which removes instances randomly, TL identifies and removes certain specific samples that contribute to class overlap, thus improving the class separability.

The process can be described as follows:

1. *Identify TLs*. A TL is defined between two instances x_i and x_j from different classes if they are each other's nearest neighbor. Formally, a pair of instances (x_i, x_j) form a TL if x_i is the nearest neighbor of x_j and vice versa, and they belong to different classes.
2. *Remove instances*. The common approach in using TLs for undersampling involves removing the majority-class instances that are part of TLs. This is based on the idea that in a TL, one of the instances is likely noise or borderline, and removing it can help in making the decision boundary more distinct.
3. *Refine the data set*. After the removal of these instances, the data set typically exhibits a clearer separation between classes.

The key idea behind TL is to enhance class separability rather than directly achieving class balance. This method is particularly effective in reducing class overlap, aiding classifiers in better distinguishing between classes. However, it may not significantly alter the class distribution balance, as its primary focus is on improving the decision boundary clarity.

Figure 8 illustrates the implementation of two sampling approaches on the SWAN-SF data set.

4. Experiments and Prediction Methods

4.1. Cleaned SWAN-SF Data Set and Code Repository of the Study

We have released the cleaned SWAN-SF data set across all five partitions. The training set in this improved version uses our FPCKNN imputation technique, removes samples from Classes B and C to reduce class overlap, and includes both oversampling and undersampling methods such as RUS, TL, and TimeGAN. Additionally, LSBZM normalization has been applied. The test set includes only the FPCKNN imputation technique and LSBZM normalization. This optimized data set allows researchers to concentrate on developing more accurate classifiers rather than preprocessing, thus saving time in space weather analysis. The "Cleaned SWAN-SF" can be accessed on GitHub[2] and is also archived on Zenodo at doi:10.5281/zenodo.11566472 (M. EskandariNasab et al. 2024a).

All Jupyter notebooks related to imputation, normalization, NDBSR, sampling algorithms, and classification algorithms in this study are meticulously documented and available for comprehensive review and application. The "SWAN-SF Data Preprocessing and Sampling Notebooks" are accessible on GitHub[3] and are also archived on Zenodo at doi:10.5281/zenodo.11564789 (M. EskandariNasab et al. 2024b). This repository not only contains the key algorithms but also offers precise specifications of the hyperparameters used in the analyses. The primary aim of making this resource available is to enhance transparency, ensure reproducibility, and foster further exploration into the methodologies utilized within the confines of our research. The hyperparameters of the classifiers and the oversampling techniques are also explained in Appendix A.

4.2. Train and Test Sets

The SWAN-SF data set consists of five distinct partitions, each representing a specific timeline of ARs. For our experiments, we utilize four unique train–test combinations, as illustrated in Figure 9. Given the temporal ordering of the partitions, it was optimal to select combinations that are consecutive. This approach ensures that the training set precedes the test set in terms of the temporal sequence, which is a crucial factor in our analysis.

[2] The "Cleaned SWAN-SF" is available on GitHub at https://github.com/samresume/Cleaned-SWANSF-Dataset.

[3] The "SWAN-SF Data Preprocessing and Sampling Notebooks" are available on GitHub at https://github.com/samresume/SWANSF-Data Preprocessing-Sampling-Notebooks.

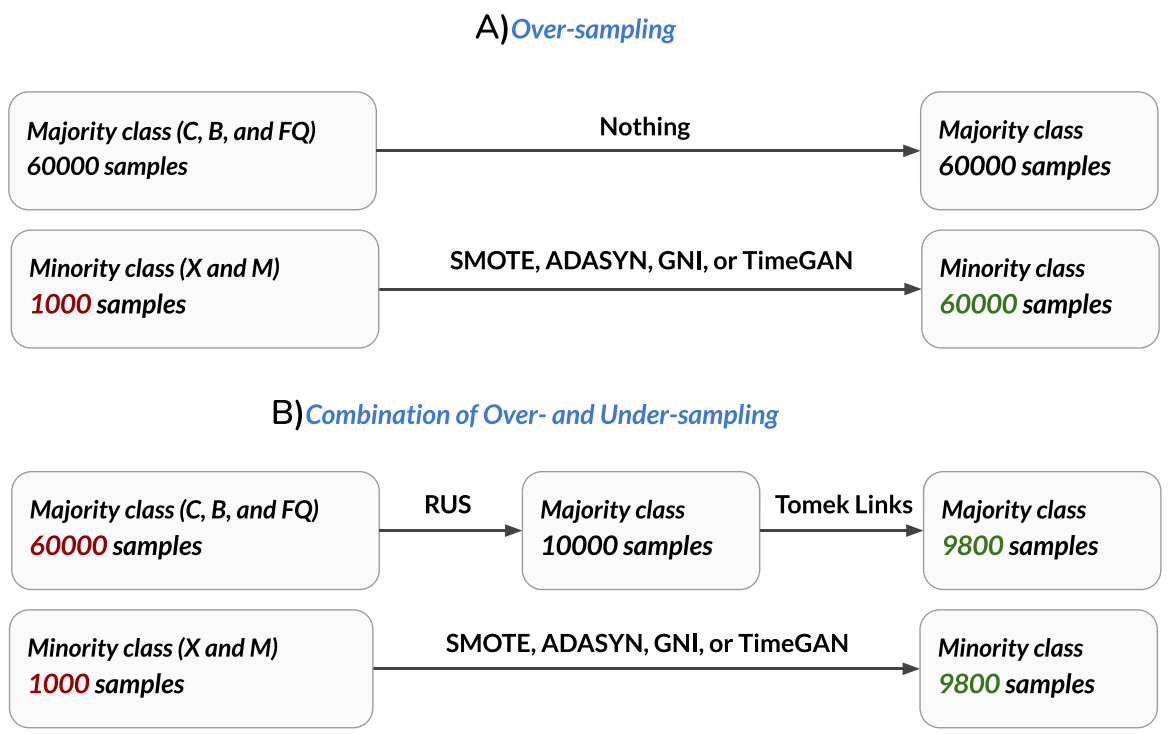

Figure 8. Illustration of two sampling approaches for SWAN-SF data set: the top panel (A) shows the original data set with a majority class having 60,000 samples and a minority class with 1000 samples. It demonstrates oversampling techniques (SMOTE, ADASYN, GNI, TimeGAN) applied to the minority class to match the majority class's 60,000 samples. The bottom panel (B) depicts a combination of oversampling and undersampling, where the majority class is reduced using RUS, and TL to 9800 samples, and the minority class is augmented to 9800 samples using the same oversampling techniques.

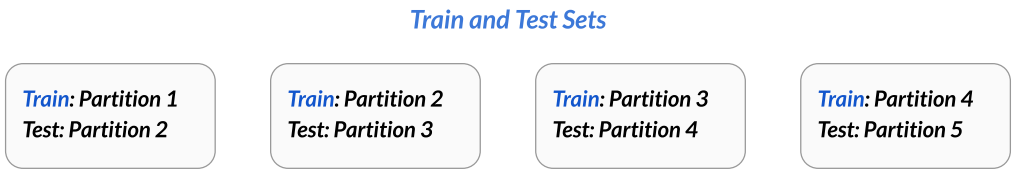

Figure 9. This figure showcases four distinct train–test sets employed in each classification experiment. This approach ensures a more comprehensive and accurate assessment of algorithms across all data set partitions.

4.3. Classification Algorithms

To evaluate the impact of preprocessing and sampling techniques on the binary classification of flares in the SWAN-SF data set, we employed various classification algorithms to thoroughly assess these techniques' effects on classification performance. Testing with diverse classifiers is essential to confirm their efficacy in enhancing classification outcomes. First, we converted the MVTS data into vector data through hand-engineered feature extraction. We then compared the performance of each step in our preprocessing and sampling pipeline against baseline techniques using four widely recognized classifiers, namely, SVM, multilayer perceptron (MLP), k-NN, and random forest (RF). Second, we implemented all four preprocessing algorithms, including FPCKNN imputation, LSBZM normalization, NDBSR, and sampling on the data set. The performance of the thoroughly preprocessed SWAN-SF data set was subsequently compared to that of the unprocessed SWAN-SF data set, utilizing the actual MVTS data on advanced time-series-based classifiers, including LSTM, recurrent neural network (RNN), gated recurrent unit (GRU), and 1D convolutional neural network (1D-CNN).

As the evaluation metric, we employ the TSS (M. G. Bobra & S. Couvidat 2015). The TSS is a valuable metric for evaluating imbalanced binary classification models, especially in solar flare prediction. It effectively balances the model's recall (TP rate) and its ability to limit the FPR, thus providing a comprehensive measure of model performance. The TSS is recalculated as follows:

$$\mathrm{TSS} = \mathrm{recall} - \mathrm{FPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} - \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}. \quad (9)$$

In this equation, TP represents the number of true positives, FN is the number of false negatives, FP is the number of false positives, and TN is the number of true negatives. The TSS value, ranging from -1 to $+1$, indicates perfect skill at $+1$, no skill (similar to random chance) at 0, and inverse skill at -1. This metric's independence from conditions such as the proportion of actual positives or negatives makes it particularly useful for evaluating models on imbalanced data sets. By encompassing both the ability of the model to correctly identify positive cases (recall) and its effectiveness in avoiding FP, TSS provides a thorough insight into the classifier's overall performance.

Based on the formula for calculating the TSS, achieving a high TSS score necessitates obtaining the highest possible recall. For example, if the recall is only 0.9, considering that TSS also requires minimizing the FPR as part of its calculation, the maximum TSS score attainable would be limited to 0.9. Therefore, our objective should be to achieve the highest possible recall, ideally 1, while concurrently striving to reduce the FPR as much as possible to optimize the TSS score.

4.3.1. Feature Extraction-based Classification

In the field of machine learning and data science, traditional classifiers such as SVM (C. Cortes & V. Vapnik 1995), MLP (M. W. Gardner & S. R. Dorling 1998), k-NN (L. E. Peterson 2009), and RF (L. Breiman 2001) are not specifically designed for handling time series data. However, with proper feature extraction techniques, these conventional models can be as effective as the latest deep-learning algorithms in classifying time series data.

Therefore, a crucial step is the strategic extraction of important features from each attribute of the MVTS data set, which typically consists of univariate time series. By transforming the MVTS data set into a set of statistical features, we can accelerate the training process and enhance the model's performance, leading to a more efficient and effective approach to time series classification. For example, consider the SWAN-SF data set, which includes 24 attributes, each recorded at 60 time stamps. This results in a total of 1440 data points (60×24), a volume that surpasses the processing capacity of most conventional classifiers. Many of these data points may be redundant or have minimal impact on distinguishing between classes.

To achieve this, we employ a methodical strategy for feature extraction by focusing on nine statistical properties. These properties collectively define the key characteristics of each univariate time series in an MVTS data set. These nine statistical features are shown below. In these formulas, t refers to the number of time stamps of a feature in an MVTS data set, which is set to 60 for the SWAN-SF data set.

1. *First value.* X_1
2. *Last value.* X_t
3. *Mean.* $\mu = \frac{1}{t}\sum_{i=1}^{t} X_i$
4. *Median.* $\text{Median}(X)$
5. *Weighted average.* $\bar{X}_w = \frac{\sum_{i=1}^{t} w_i X_i}{\sum_{i=1}^{t} w_i}$
6. *Standard deviation.* $\sigma = \sqrt{\frac{1}{t-1}\sum_{i=1}^{t}(X_i - \mu)^2}$
7. *Skewness.* $\frac{t}{(t-1)(t-2)}\sum_{i=1}^{t}\left(\frac{X_i - \mu}{\sigma}\right)^3$
8. *Kurtosis.* $\frac{t(t+1)}{(t-1)(t-2)(t-3)}\sum_{i}^{t} = 1\left(\frac{X_i - \mu}{\sigma}\right)^4 - \frac{3(t-1)^2}{(t-2)(t-3)}$
9. *Slope.* $a = \frac{t(\sum xy) - (\sum x)(\sum y)}{t(\sum x^2) - (\sum x)^2}$.

By concentrating on these statistical properties, we can greatly reduce the dimensionality of the data set without losing essential information necessary for accurate classification. This not only makes training the model more straightforward but also enhances the classifiers' ability to differentiate between various classes in the data set more precisely and efficiently. Through careful feature extraction and selection, traditional machine learning classifiers can be successfully adapted to the specifics of time series data, allowing them to compete with more complex deep-learning models in certain scenarios.

4.3.2. Time-series-based Classification

In the domain of time series data analysis, advanced deep learning-based models enable us to perform classification using the actual MVTS data set without feature extraction (J. Chen et al. 2022; Z. Sun et al. 2022). Models such as LSTM (S. Hochreiter & J. Schmidhuber 1997), RNN (A. Sherstinsky 2020), GRU (J. Chung et al. 2014), and 1D-CNN (Y. Lecun et al. 1998) are particularly adept at enhancing classification accuracy in MVTS data. Therefore, there is no need for any feature extraction techniques such as those in Section 4.3.1. This approach is especially effective in scenarios where understanding complex temporal patterns and structural details in the time series data is difficult to achieve through mere statistical feature extraction. Our study rigorously evaluates these advanced classifiers to gain a deeper understanding of how preprocessing and sampling strategies impact the performance of classification on the SWAN-SF data set.

In the context of MVTS data analysis, it is vital to recognize the inherently three-dimensional nature of this data. Each sample in an MVTS data set can be viewed as a two-dimensional matrix, where each attribute represents a one-dimensional time series. Deep-learning models such as LSTM, RNN, GRU, and 1D-CNN are inherently designed to handle three-dimensional data structures. This capability eliminates the need for transforming each two-dimensional data point into a one-dimensional vector (concatenation). Additionally, these neural network-based classifiers come with an embedded feature learning mechanism within their architecture. This feature significantly reduces the need for handcrafted feature engineering, thereby streamlining the model training process. The incorporation of this automatic feature extraction is a substantial benefit, as it saves a considerable amount of time and resources that would otherwise be spent on feature engineering, and it also enhances the model's ability to detect and learn complex patterns in the data.

4.4. Visualization Techniques

We utilized various visualization techniques with distinct properties to facilitate the understanding of the experiment's results. The main techniques employed were bar charts, scatter plots, and box plots. Here is an explanation of these techniques.

Bar chart. A bar chart, as shown in Figures 11, 13, 16, and 20, visually represents data using rectangular bars, with the length of each bar proportional to its corresponding value. Bar charts are frequently used to compare different categories of data, where each bar stands for a category, and its height or length reflects the magnitude of the value it represents. They can be displayed vertically or horizontally. In these figures, each bar displays two values: the first value from left to right indicates the mean TSS, and the second value represents the max TSS.

Scatter plot. A scatter plot, as shown in Figures 12 and 19, depicts the results of an experiment more intuitively compared to the bar chart we have used. The scatter plots have TSS on the x-axis and recall on the y-axis, illustrating the importance of both values in imbalance classification tasks. To achieve a high TSS, a high recall is essential. In these plots, the size of each shape indicates the mean TSS, while the placement of the shape is determined by the max TSS and max recall. This visualization technique helps in identifying patterns and

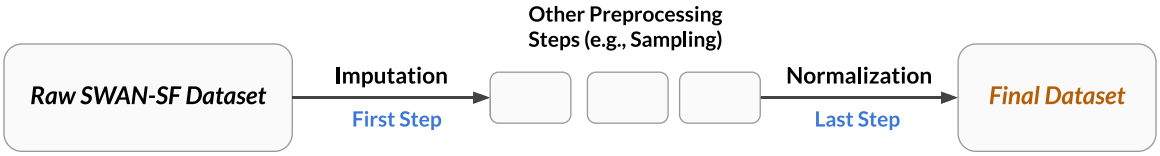

Figure 10. This figure illustrates the sequence of imputation and normalization tasks in a data preprocessing pipeline.

potential outliers in the data, making it easier to analyze and interpret the results of imbalance classification tasks.

Box plot. A box plot, as shown in Figures 17, 18, 21, and 22, visually represents the distribution of data in an experiment. It displays the median, quartiles, and potential outliers. The box indicates the interquartile range (IQR), which contains the middle 50% of the data, with the line inside the box representing the median. The "whiskers" extend to the smallest and largest values within 1.5 times the IQR from the quartiles. An outlier is a data point that significantly deviates from other observations. It lies far outside the expected range, typically more than 1.5 times the IQR above the third quartile or below the first quartile. Outliers can result from data variability or measurement errors and are often marked with diamond symbols in box plots to highlight these extreme values.

5. Prediction Results

5.1. Impact of FPCKNN Imputation and LSBZM Normalization

In the preprocessing pipeline for the SWAN-SF data set, FPCKNN imputation is employed as the initial step. This is crucial for addressing missing values before any subsequent preprocessing techniques are implemented. Following this, the final step in the preprocessing sequence involves LSBZM normalization. It is essential that other preprocessing operations, such as sampling, are executed postimputation but prenormalization. The justification behind this sequencing is that sampling methods, along with other preprocessing techniques, can potentially alter the data set by either omitting or introducing additional samples. These changes consequently affect the data set's mean and standard deviation, which in turn impacts the efficacy of normalization procedures. If feature extraction-based classification is our choice, we need to extract the statistical features and create the new data set before normalization. Figures 10 and 14 visually represent the sequential order of these preprocessing stages.

To analyze the impact of FPCKNN imputation and LSBZM normalization on our data, we first implemented the FPCKNN imputation technique, proceeded with the extraction of statistical features, and then applied the LSBZM normalization method. Both the training and test data sets undergo these techniques before proceeding to the model training phase. Subsequently, we establish baseline methodologies for the comparative analysis of our results. For baseline comparisons, we have selected two distinct imputation strategies: mean imputation and next value imputation. Additionally, we have chosen two normalization techniques: Z-score normalization and min–max scaling. Our analysis focuses on comparing the outcomes of combining three imputation methods (FPCKNN, next value, and mean) with three normalization techniques (LSBZM, Z-score, and min–max) to identify the highest-performing combination.

In Section 3.1.2, we explored the methodologies behind Z-score normalization and min–max scaling. The following section delves into the mechanisms of mean imputation and next value imputation, two established techniques in data preprocessing. Mean imputation replaces missing values in a data set with the mean value of the entire feature column. While straightforward, this method can lead to an underestimation of the true variability in the data, as it artificially reduces variance by replacing missing values with the mean, affecting statistical analyses and model performance. On the contrary, the next value imputation substitutes a missing value with the next available (nonmissing) value in the data sequence. This method relies on the temporal or sequential structure of the data, making it well suited for time series data sets where the missing value can be reasonably estimated by the subsequent observed value (S. I. Khan & A. S. M. L. Hoque 2020).

5.1.1. Classification Results

Based on Figures 11 and 12, as well as Table 4, it is evident that our combined imputation and normalization technique (FPCKNN-I + LSBZM-N) significantly enhances the mean TSS scores compared to the baseline methods used for comparison, as discussed in Section 5.1. Specifically, as shown in Figure 11, the SVM classifier achieved a mean TSS score of 0.154 with the best baseline approach (next value-I + min–max-N) and a mean TSS score of 0.2 by utilizing our normalization technique with a baseline imputation (next value-I + LSBZM-N). In contrast, our combined imputation and normalization technique (FPCKNN-I + LSBZM-N) improved the mean TSS score to 0.255. Similarly, for the MLP classifier, our combined imputation and normalization method alone resulted in a mean TSS score of 0.321, without the need for additional preprocessing steps such as sampling.

Figure 12 simply illustrates the significance of the two pivotal evaluation metrics for achieving optimal classification performance in the SWAN-SF imbalanced data set. Following the TSS formula, maximizing the recall value is paramount for attaining the highest TSS score in the binary classification of flares within the SWAN-SF data set. Consequently, the recall value holds a crucial position within the TSS score, making it imperative to display both metrics in a 2D space for a more intuitive visualization. As shown in the figure, the green markers, representing our combined imputation and normalization methods, achieve the highest TSS and recall scores compared to the baseline techniques, indicated by the red and blue markers.

5.2. Impact of NDBSR

To demonstrate the effectiveness of eliminating class overlap in enhancing classification performance, we have dedicated a separate section to this topic in our paper, even though it essentially involves a sampling technique. The NDBSR method should be applied after imputation in the preprocessing pipeline. It is important to perform the NDBSR technique, or potentially additional sampling strategies, before feature

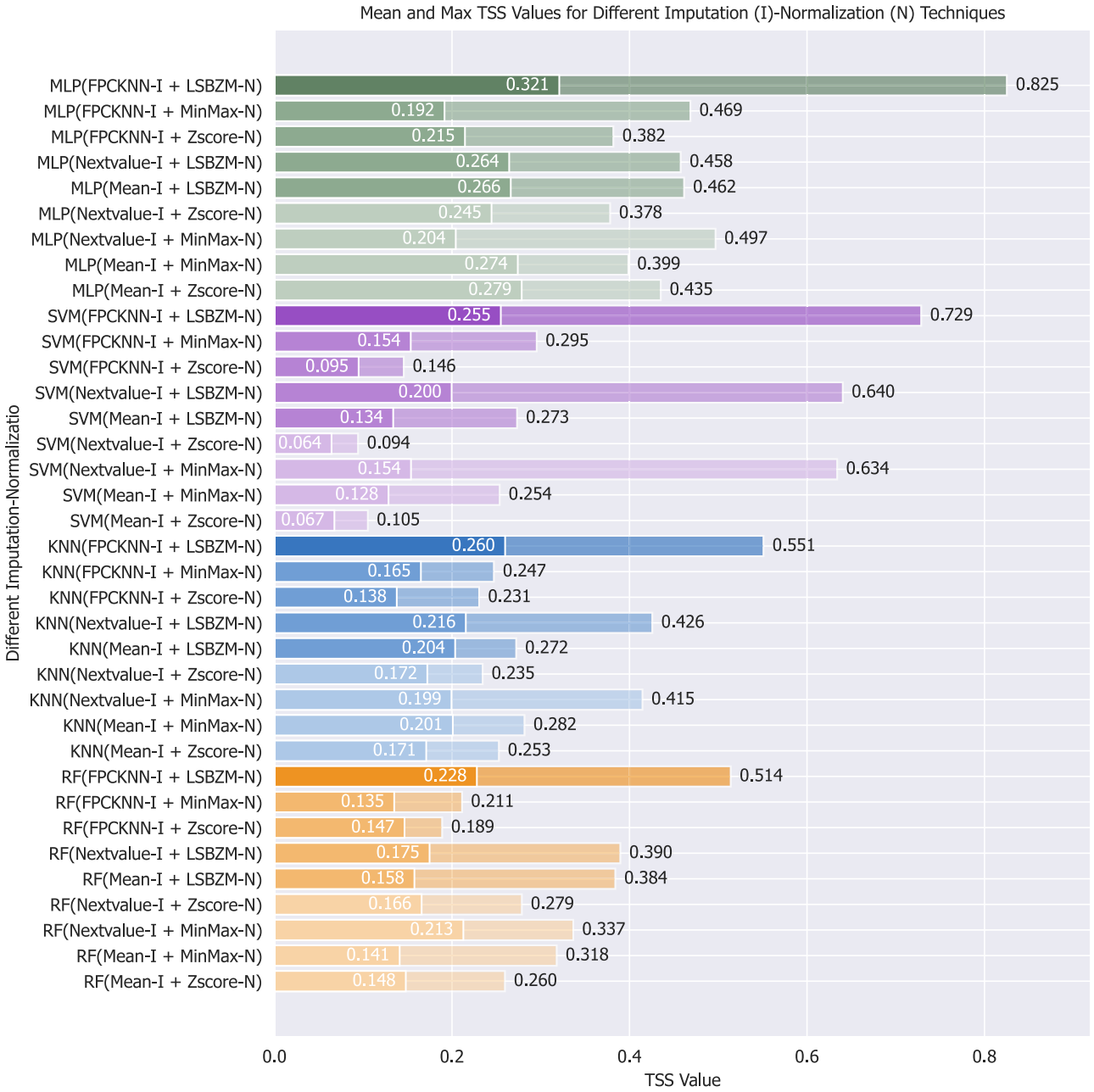

Figure 11. Comparative analysis of the mean and max TSS values across nine combinations of imputation (I) and normalization (N) techniques. Efficacy evaluated via four classifiers: SVM, MLP, k-NN, and RF. For each bar, the first value indicates the mean TSS, and the second value indicates the max TSS of four train–test combinations.

extraction. Normalization is invariably the final step, executed after feature extraction. Our baseline method for this section is the data set with FPCKNN imputation and LSBZM normalization but without class overlap removal. Based on Figure 13 and Table 5, our findings suggest that removing both classes B and C is more effective, resulting in fewer FPs, and consequently, a higher TSS score. For comparative analysis and to assess the impact of these methods, we first apply our imputation technique, followed by the removal of class C, or classes B and C, then feature extraction, and finally, our normalization technique. The order and integration of these steps in the preprocessing pipeline are clearly illustrated in Figure 14. It is important to note that for the test sets, we do not remove classes B and C or any instances; instead, we perform the remaining steps. Implementing these methods is straightforward: researchers simply need to exclude classes B and C entirely or only class C. This results in a minor-flaring class that contains only FQ or FQ and B samples.

5.2.1. Classification Results

Based on Figure 13 and Table 5, removing classes B and C together is more effective than removing only class C. This approach leads to higher mean TSS scores for SVM, MLP, k-NN, and RF classifiers. Both methods significantly enhance TSS results compared to not addressing class overlap. Specifically, for the MLP classifier, the mean TSS improved from 0.321 to 0.671 by eliminating classes B and C from the training data set. Among the classifiers tested, MLP and SVM were the most effective. These results underscore the

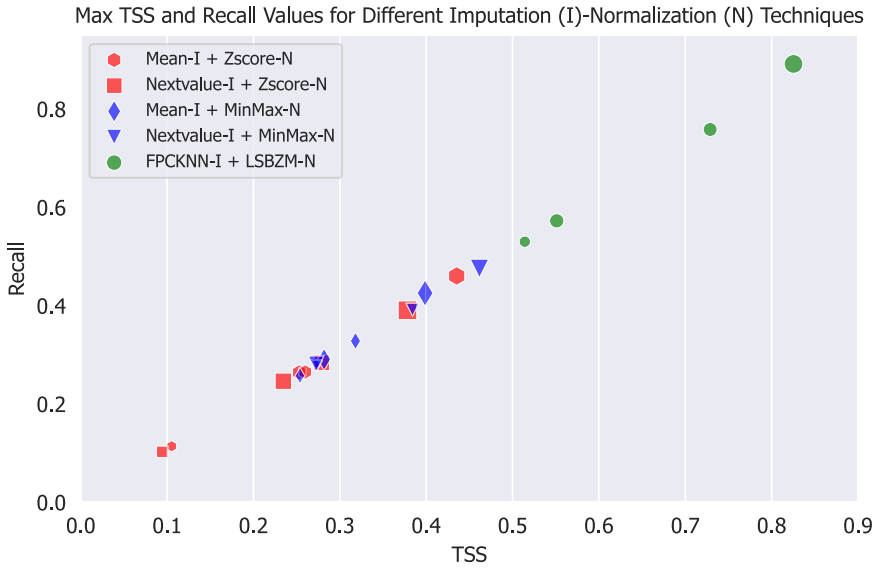

Figure 12. The comparative analysis of max TSS and max recall across different imputation (I) and normalization (N) techniques is illustrated. The efficacy of these algorithms is assessed using SVM, MLP, k-NN, and RF classifiers. In the plot, each shape represents the result of classification using the four train–test combinations with one of the mentioned classifiers. The size of each shape indicates the mean TSS, while the position of the shape is determined by the max TSS and max recall values.

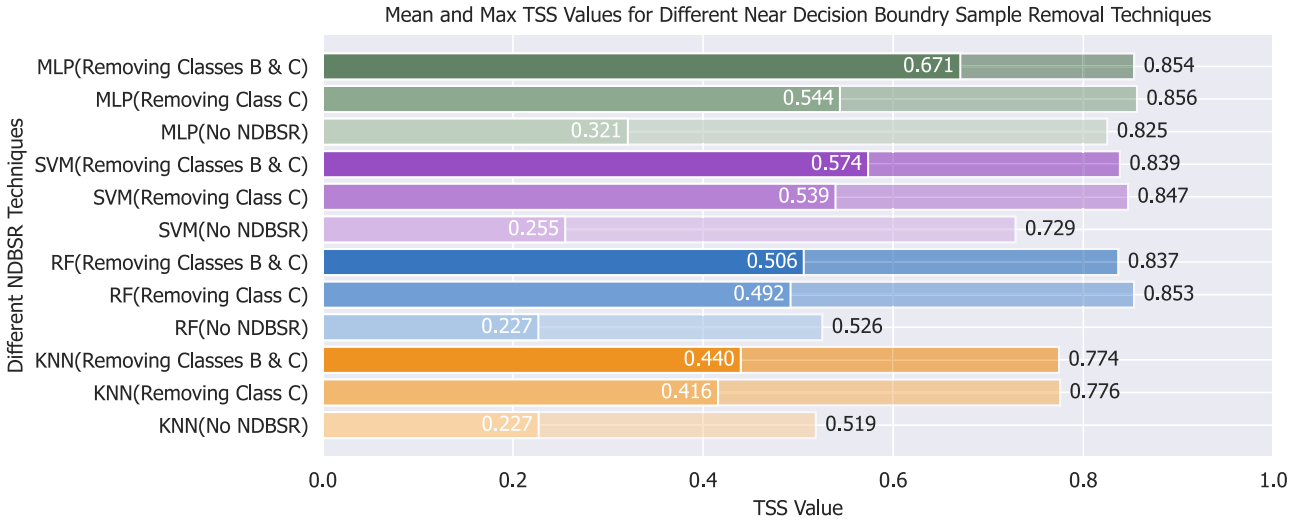

Figure 13. This figure presents a comparison of the mean and max TSS scores achieved by various NDBSR methods when applied to different classifiers. These classifiers include MLP, RF, SVM, and k-NN. For each bar, the first value indicates the mean TSS, and the second value indicates the max TSS of four train–test combinations.

Table 4
Mean TSS Values for Nine Combinations of Imputation (I) and Normalization (N) Techniques

Technique	k-NN	SVM	RF	MLP
FPCKNN-I and LSBZM-N	**0.260** ± 0.171	**0.255** ± 0.274	**0.228** ± 0.173	**0.321** ± 0.299
FPCKNN-I and min–max-N	0.165 ± 0.059	0.154 ± 0.092	0.135 ± 0.062	0.192 ± 0.162
FPCKNN-I and Z-score-N	0.138 ± 0.068	0.095 ± 0.036	0.147 ± 0.046	0.215 ± 0.153
Next value-I and LSBZM-N	0.216 ± 0.128	0.223 ± 0.141	0.175 ± 0.124	0.264 ± 0.176
Mean-I and LSBZM-N	0.199 ± 0.126	0.154 ± 0.145	0.213 ± 0.114	0.204 ± 0.192
Next value-I and min–max-N	0.203 ± 0.072	0.133 ± 0.096	0.157 ± 0.138	0.266 ± 0.152
Next value-I and Z-score-N	0.172 ± 0.055	0.064 ± 0.030	0.165 ± 0.098	0.244 ± 0.115
Mean-I and min–max-N	0.200 ± 0.071	0.128 ± 0.087	0.141 ± 0.118	0.274 ± 0.141
Mean-I and Z-score-N	0.170 ± 0.063	0.067 ± 0.030	0.147 ± 0.076	0.278 ± 0.146

Note. The method in bold achieved the top scores.

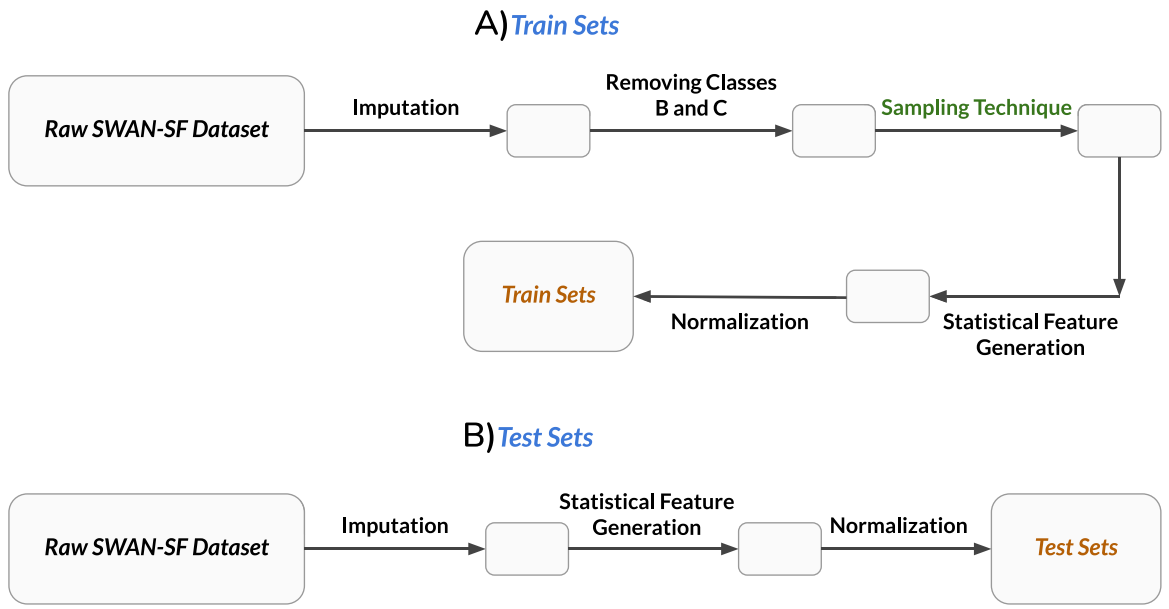

Figure 14. This figure effectively showcases the preprocessing pipelines, detailing the specific steps and methods employed for both training (panel (A)) and test sets (panel (B)). To ensure an unbiased comparison of the sampling results, it is crucial that the test sets remain unaltered, with no samples being either removed or added.

Table 5
Mean TSS Values for Different NDBSR Techniques

NDBSR Technique	k-NN	SVM	RF	MLP
No NDBSR	0.227 ± 0.171	0.255 ± 0.274	0.227 ± 0.174	0.321 ± 0.299
Removing class C	0.416 ± 0.210	0.539 ± 0.182	0.492 ± 0.212	0.544 ± 0.201
Removing classes B and C	**0.440** ± 0.195	**0.574** ± 0.160	**0.506** ± 0.192	**0.671** ± 0.169

Note. The method in bold was the best-performing NDBSR technique.

importance of these techniques for data sets such as SWAN-SF, which show significant class overlap in both binary and multiclass classification scenarios.

5.3. Impact of Sampling Techniques

As illustrated in Figure 14, our initial step involved applying our imputation technique to all training samples. We then purposefully removed classes B and C due to their significant overlap with class M. This decision was aimed at improving the classification results by excluding all classes B and C samples from the majority class. Following this exclusion, we applied sampling techniques to the training sets. Additionally, we generated nine statistical features for each attribute within the data set, culminating the process with normalization. In contrast, the test sets underwent a different treatment: we did not apply sampling and chose to retain classes B and C samples. The processing for these test sets was limited to imputation, the generation of statistical features, and normalization.

5.3.1. t-Distributed Stochastic Neighbor Embedding Visualizations

Following the generation of synthetic samples for the minority class, we conduct a balanced selection, choosing an equal number of synthetic samples as there are in the original minority class (approximately 1000). Subsequently, we utilize t-distributed stochastic neighbor embedding (t-SNE; L. van der Maaten & G. Hinton 2008) for the visualization of both original and synthetic samples. This technique facilitates the assessment of similarity in distribution between the synthetic and original samples by representing them in a two-dimensional space, thereby offering a visual comparison of their respective distributions.

As illustrated in Figure 15, which shows the t-SNE visualizations between original and synthetic samples, the distributions of synthetic samples created by SMOTE and GNI are most effective, while ADASYN struggles to replicate the entire distribution of original samples. This limitation stems from ADASYN's strategy of concentrating on generating synthetic samples for instances that classifiers find more difficult to learn. TimeGAN also produces synthetic data that closely mirrors the distribution of original data. However, the synthetic data generated by TimeGAN holds greater value as it represents entirely new and unique data, rather than perturbing the original samples. As a result, TimeGAN's data is less prone to overfitting since the synthetic samples are not mere replications of the original samples. Overall, samples generated by TimeGAN are highly valuable, as they not only represent new and unique data but also effectively encompass the full distribution of the original samples. For the principal component analysis (PCA) visualizations, please refer to Appendix C.

5.3.2. Discriminative Score

For a quantitative measure of similarity, we train an SVM classifier to distinguish between synthetic and original samples.

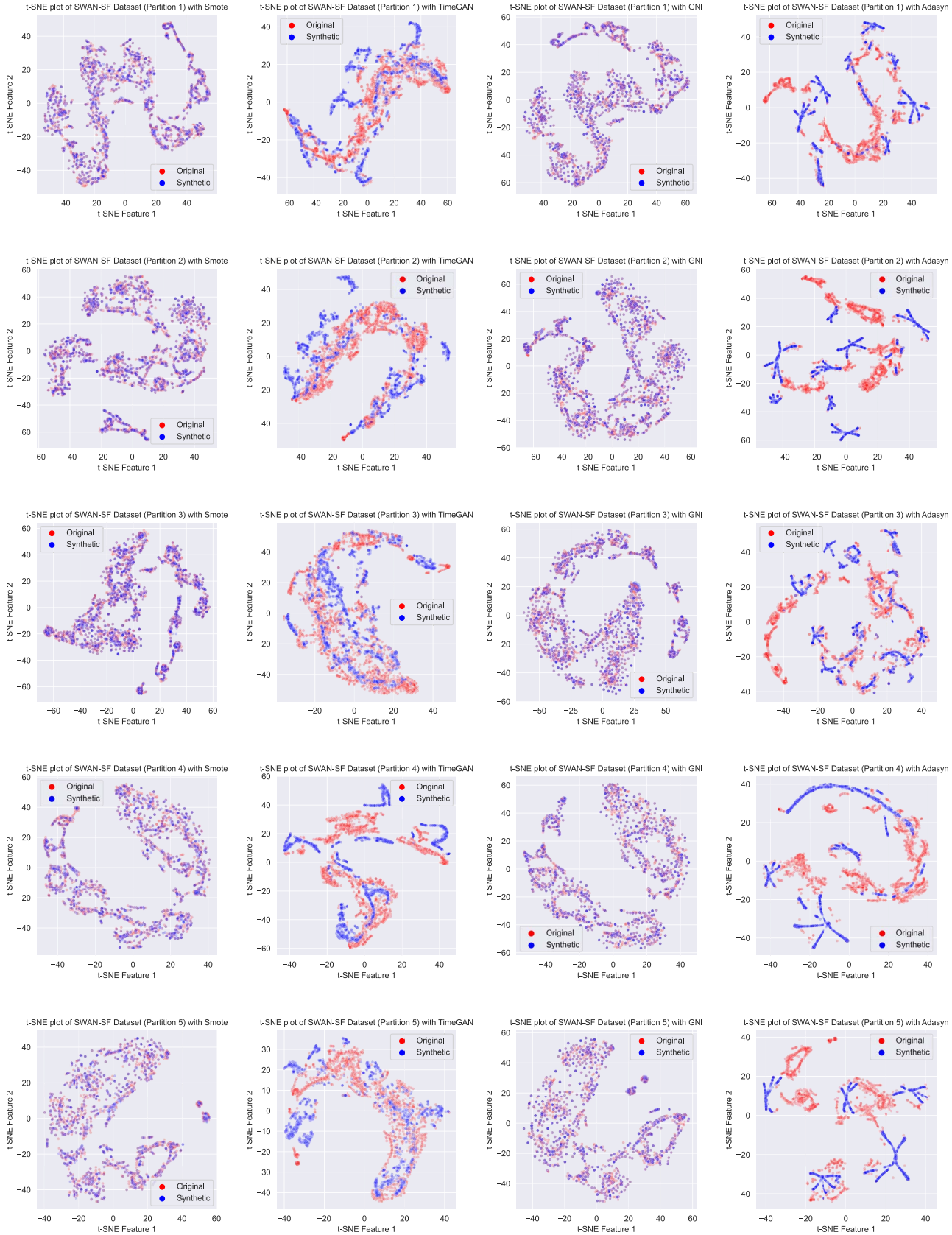

Figure 15. t-SNE visualizations demonstrating the distributional alignment of original and synthetic data samples for each oversampling technique across data set partitions. These visualizations highlight the efficacy of each oversampling method in replicating the comprehensive distribution characteristics of the original data samples.

Table 6
Comparison of Discriminative Scores for the Oversampling Techniques across the Data Set Partitions

Technique	Partition 1	Partition 2	Partition 3	Partition 4	Partition 5
SMOTE	**0.000664**	0.037456	0.021638	**0.016452**	0.023569
TimeGAN	0.238379	0.300237	0.259064	0.389842	0.287878
GNI	0.023241	**0.008918**	**0.020468**	0.019314	**0.001683**
ADASYN	0.213147	0.347800	0.185380	0.252503	0.306397

Note. The bold scores correspond to the lowest discriminative scores for each partition, indicating the best results, as lower discriminative scores reflect better performance.

Table 7
Mean TSS Values for Different Sampling Techniques

Sampling Technique	k-NN	SVM	RF	MLP
No Sampling	0.227 ± 0.171	0.255 ± 0.274	0.227 ± 0.174	0.321 ± 0.299
SMOTE	0.685 ± 0.067	0.704 ± 0.140	0.568 ± 0.122	0.757 ± 0.051
ADASYN	0.667 ± 0.051	0.656 ± 0.137	0.513 ± 0.118	0.662 ± 0.107
GNI	0.627 ± 0.111	0.692 ± 0.106	0.705 ± 0.083	0.792 ± 0.056
TimeGAN	0.576 ± 0.106	0.718 ± 0.077	0.716 ± 0.080	0.700 ± 0.059
RUS-TL-SMOTE	**0.739** ± 0.014	0.726 ± 0.126	0.681 ± 0.098	**0.808** ± 0.044
RUS-TL-ADASYN	0.715 ± 0.036	0.704 ± 0.143	0.651 ± 0.071	0.776 ± 0.044
RUS-TL-GNI	0.655 ± 0.056	**0.783** ± 0.068	**0.792** ± 0.065	0.778 ± 0.055
RUS-TL-TimeGAN	0.649 ± 0.097	0.742 ± 0.107	0.789 ± 0.049	0.794 ± 0.038

Note. The bold scores represent the highest scores achieved for each classifier.

Initially, each original sample is labeled as "real," and each synthetic sample as "not real." Then, the training of the SVM is conducted to categorize these two distinct classes in a typical supervised learning framework. We then obtain the classification accuracy on a held-out test set, which constitutes 30% of the data. Afterward, we compute the error using the formula

$$e = a - 0.5, \tag{10}$$

where e represents the error and a the accuracy. Finally, we report this error to provide a quantitative assessment. Lower scores indicate better performance, with the ideal score being 0.

Based on Table 6, SMOTE and GNI exhibit the lowest error values, signifying their effectiveness in producing high-quality synthetic data for the minority class of the SWAN-SF data set. However, the data they generate lacks ingenuity and uniqueness. The discriminative score results for ADASYN are unsatisfactory, indicating its inability to produce synthetic data resembling the original. Additionally, while the results for TimeGAN are also unsatisfactory, the data it generates holds greater value due to its originality and uniqueness. A comparison of the classification results is essential to make a final decision.

5.3.3. Classification Performance

Based on Figures 16, 17, 18, and 19, as well as Table 7, sampling significantly enhances classification performance when compared to the absence of sampling. For example, as illustrated in Figure 16, when sampling techniques are not employed (No Sampling), the MLP classifier achieves a mean TSS score of 0.321, while the RF classifier attains a mean TSS of 0.227. However, after applying a combination of oversampling and undersampling techniques, the performance significantly improves. The mean TSS score for the MLP classifier increases to a range of 0.776 (RUS-TL-ADASYN) to 0.808 (RUS-TL-SMOTE), and for the RF classifier, it rises to between 0.651 (RUS-TL-ADASYN) and 0.792 (RUS-TL-GNI), indicating a substantial enhancement. Additionally, as illustrated in Figures 17 and 18, the combination of oversampling and undersampling techniques produces better results compared to oversampling alone in most cases. Therefore, it is recommended to perform undersampling alongside oversampling to reduce the number of majority-class samples and prevent overfitting. Based on Figure 16, all four techniques, including SMOTE, ADASYN, GNI, and TimeGAN, achieve a high mean TSS score depending on the classifier used. Therefore, any of these techniques can be used to achieve satisfying results, with the choice depending on the specific classifier. However, given that TimeGAN is a generative model capable of producing unique synthetic samples that closely align with the original data distribution, and considering the t-SNE visualization results shown in Figure 15, it can be concluded that TimeGAN is the optimal oversampling technique for the SWAN-SF data set.

In situations where researchers cannot use TimeGAN for oversampling, SMOTE and GNI stand out as the second and third recommended techniques for the SWAN-SF data set due to their high TSS scores. Additionally, the synthetic data generated by these two methods closely resemble the distribution of the original data. However, it is important to note that the synthetic data, while similar, are not new or genuine.

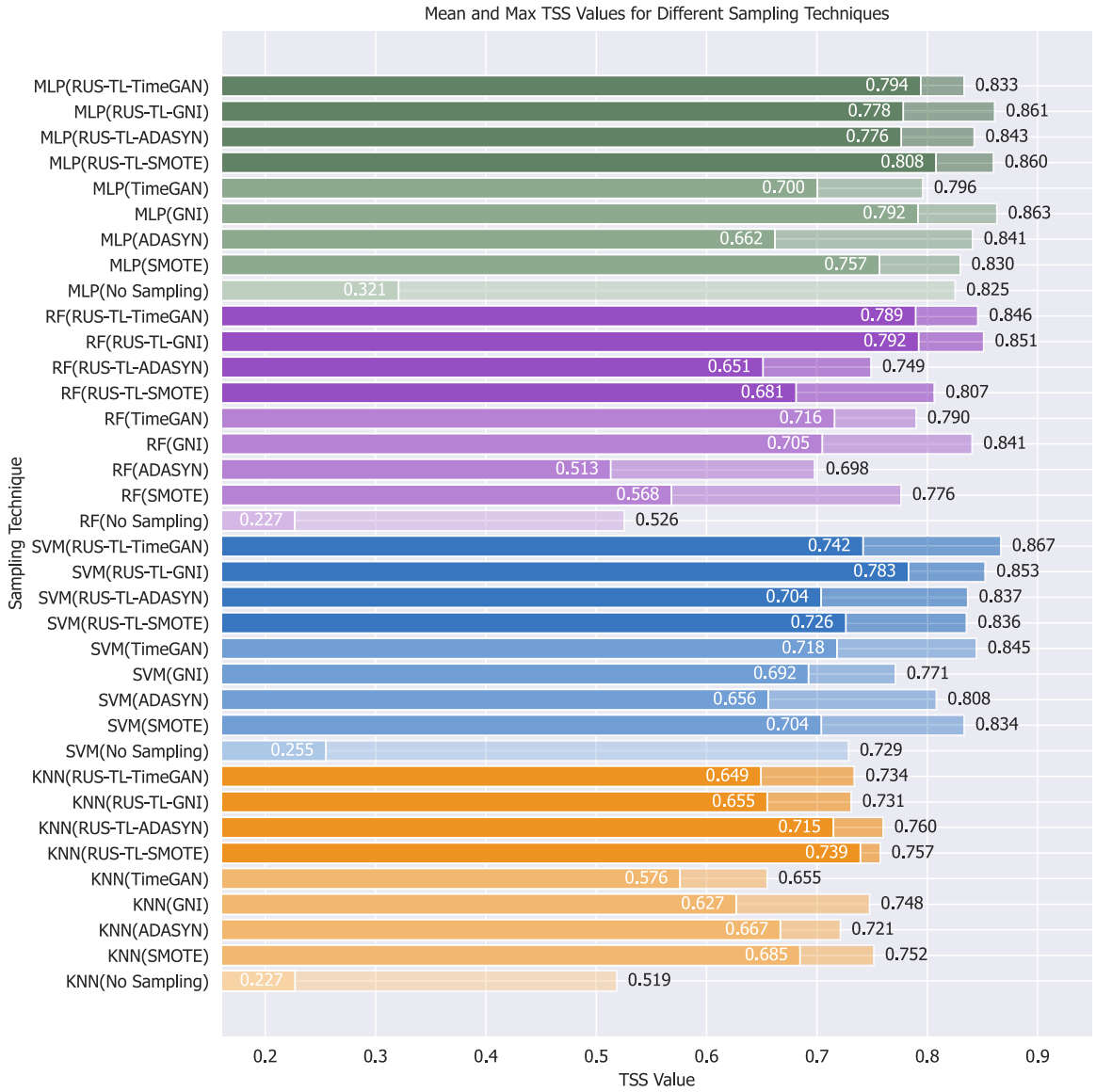

Figure 16. Comparative analysis of mean and max TSS across various sampling techniques and classifiers: This figure illustrates the performance of different sampling methods, including exclusive oversampling techniques (SMOTE, ADASYN, GNI, TimeGAN) and a combination of oversampling and undersampling approaches (RUS and TL with one of the oversampling methods). The efficacy of these techniques is evaluated using four distinct classifiers: SVM, MLP, k-NN, and RF. For each bar, the first value indicates the mean TSS, and the second value indicates the max TSS of four train–test combinations.

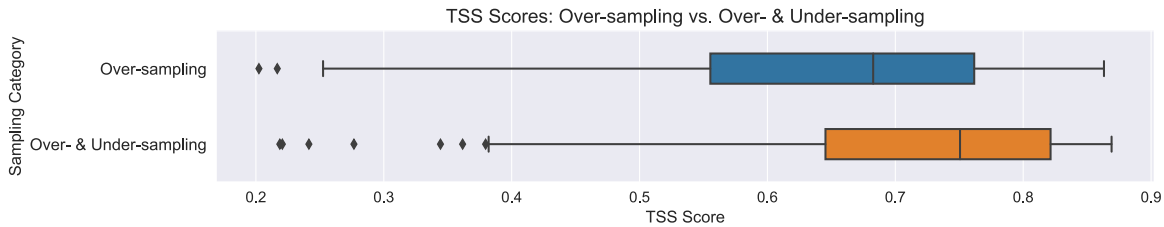

Figure 17. This analysis evaluates the performance of oversampling techniques vs. combined oversampling and undersampling methods across eight classifiers and four unique training and testing combinations. The top box displays the results of four oversampling techniques applied to the eight classifiers, each with four different train–test combinations, resulting in a total of 128 experiments. The bottom box presents the same set of experiments but uses four combined oversampling and undersampling techniques. In the plot, diamond-shaped symbols indicate outliers.

5.4. Comparison Using Time-series-based Classifiers

In the final comparison, we first apply our imputation technique, followed by the removal of classes B and C. Subsequently, we conduct both solely oversampling and a combination of oversampling and undersampling on the data, and then proceed with our normalization process. For the test sets, we do not remove classes B and C, nor do we perform sampling, but we do apply our imputation and normalization techniques. In this comparison, we omit feature extraction and instead focus on time-series-based classification using

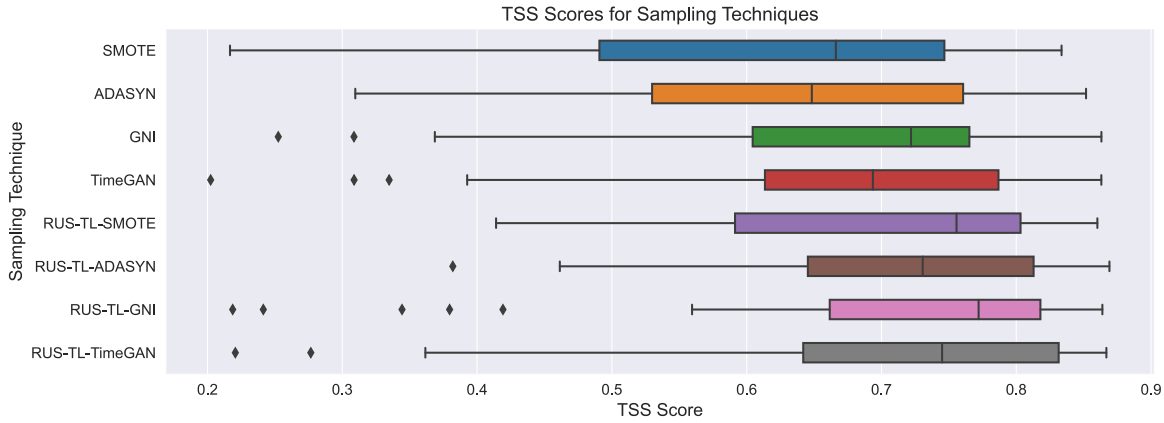

Figure 18. Assessing TSS scores of the eight sampling techniques across our eight classifiers and four unique training and test combinations. Each box shows the result of a specific sampling technique on eight classifiers and four train–test combinations, totaling 32 experiments. In the plot, diamond-shaped symbols indicate outliers.

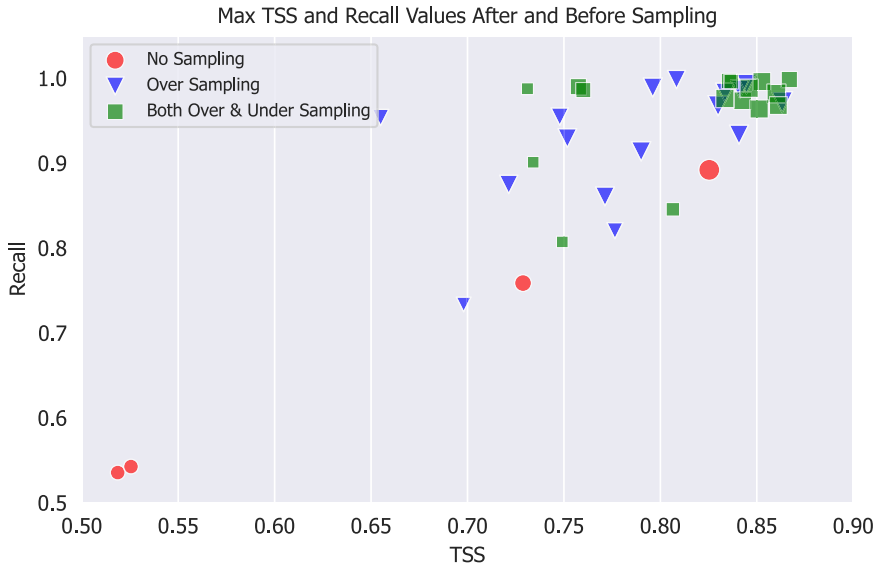

Figure 19. In-depth comparison of max TSS and max recall across different sampling techniques and classifiers: This figure illustrates the performance of various sampling methods, including exclusive oversampling techniques and combinations of oversampling and undersampling techniques. The efficacy of these methods is evaluated using four distinct classifiers: SVM, MLP, k-NN, and RF. Each shape in the plot represents the result of classification using four train–test combinations, resulting in four values. Generally, each shape in the plot shows the result of one of the three categories on one of the four classifiers across four train–test combinations, totaling four experiments. The size of each shape indicates the mean TSS, while the placement of the shape is determined by the max TSS and max recall.

state-of-the-art classifiers, including LSTM, RNN, GRU, and 1D-CNN.

Leveraging insights from Figure 20 and Table 8, which clearly demonstrate the efficacy of deep learning-based architectures in our time series classification task, it becomes evident that the GRU model outperforms others on the SWAN-SF data set. Furthermore, the integration of both oversampling and undersampling strategies enhances the TSS score more effectively than relying on oversampling alone. This approach not only elevates the TSS but also markedly reduces training duration by minimizing the generation of minority-class instances and eliminating a substantial number of majority-class samples. The LSTM network and the 1D-CNN are identified as the second and third top-performing deep-learning models, respectively, on the SWAN-SF data set. Except for the RNN classifier, the combined application of RUS, TL, and ADASYN, as well as RUS, TL, and SMOTE, has yielded the highest mean TSS outcomes for the SWAN-SF data set. Nonetheless, the absence of any preprocessing steps on the SWAN-SF data set predominantly results in a TSS score of zero, underscoring the critical role of a comprehensive preprocessing pipeline. The TSS of zero is primarily due to the large and differing scales of attributes and the low number of major-flaring samples, leading to non-convergence. Consequently, the model outputs a recall value of 0, resulting in a TSS of 0. Specifically, our approach increased the TSS from 0.0 (no preprocessing) to a mean of 0.835 (I-N-NDBSR-RUS-TL-ADASYN) for the GRU model. Additionally, for the LSTM model, the mean TSS improved from 0.302 (no preprocessing) to 0.736 (I-N-NDBSR-RUS-TL-ADASYN), demonstrating the effectiveness of our preprocessing methodology in enhancing solar flare prediction.

5.5. Comparison of Classifiers and Partitions

Based on Figure 21, the MLP and SVM classifiers demonstrate the highest effectiveness for feature extraction-based binary classification of major and minor-flaring events within the SWAN-SF data set. Additionally, the GRU and LSTM classifiers

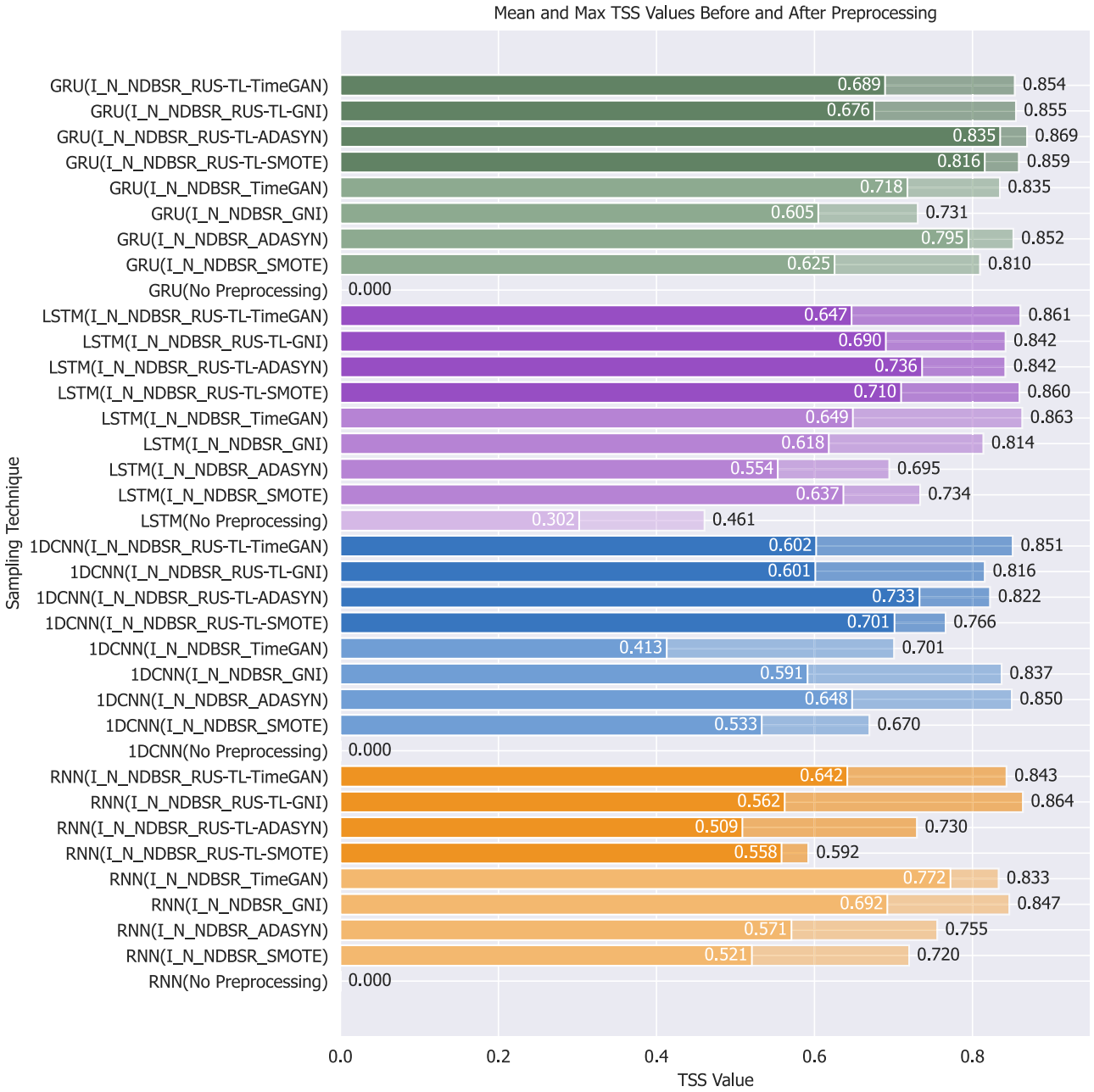

Figure 20. This figure presents the outcomes of employing no preprocessing on the data set vs. the application of our imputation (I), our normalization (N), NDBSR, and various sampling techniques in combination. The efficacy of these techniques is evaluated using four time-series-based classifiers: LSTM, GRU, RNN, and 1D-CNN. For each bar, the first value indicates the mean TSS, and the second value indicates the max TSS of four train–test combinations.

are identified as the optimal time-series-based classifiers for this data set. As illustrated in Figure 22, the selection of Partition 3 as the training set and Partition 4 as the test set yields the highest TSS scores.

5.6. Comparison with Flare Prediction Baselines

In the final section, we aim to compare our findings and results with similar flare prediction baselines conducted by A. Ahmadzadeh et al. (2021). In this comparison, we evaluate the max TSS achieved by A. Ahmadzadeh et al. (2021) against our own max TSS scores for each train–test combination of the SWAN-SF data set. Based on Figure 23, our results significantly surpass the baseline results, achieving a max TSS of over 0.8 in many cases and reaching a max TSS of 0.87. This highlights the impact of thorough preprocessing and sampling before classification. The experiments were conducted on four train–test combinations, as explained earlier. Our best result, with a max TSS ranging from 0.8 to 0.87, was achieved by employing a GRU as the classifier. This was followed by the implementation of our FPCKNN imputation technique, along with LSBZM normalization, and employing the NDBSR technique to remove classes B and C. Additionally, we utilized a combination of oversampling and undersampling techniques, including RUS, TL, and ADASYN. Unlike feature extraction-based classifiers such as SVM, ADASYN performs well with deep learning-based classifiers.

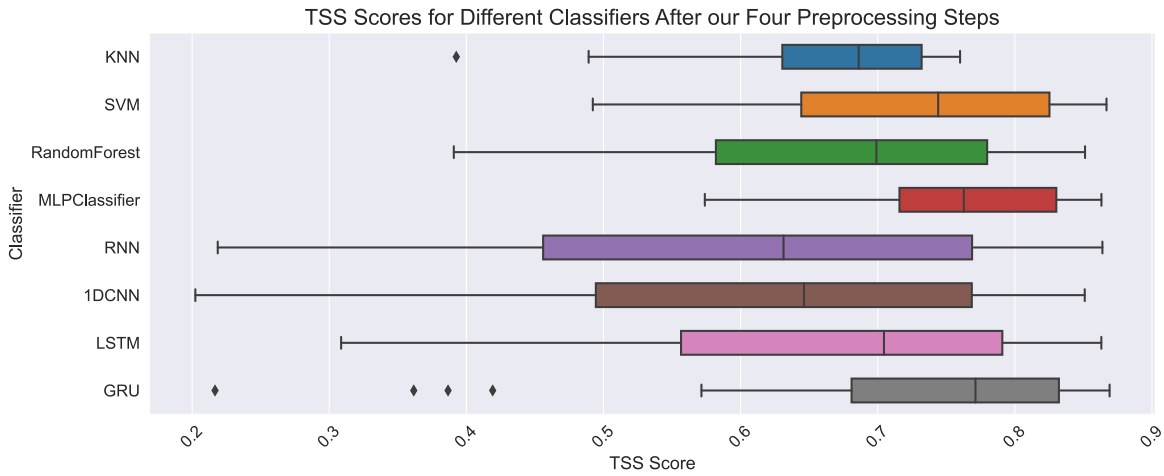

Figure 21. This plot depicts a comparative analysis of TSS scores obtained by different classifiers on the SWAN-SF data set. Each box represents the results of a specific classifier on a fully preprocessed version of the SWAN-SF data set, which includes FPCKNN imputation, LSBZM normalization, and the removal of classes B and C. The analysis also incorporates four combinations of oversampling and undersampling techniques applied to four train–test combinations, resulting in each box representing the outcome of 16 experiments. In the plot, diamond-shaped symbols indicate outliers.

Figure 22. This plot presents a comparative analysis of the performance across various training and testing combinations applied to the SWAN-SF data set. "P" signifies partition. Each box represents the results of our fully preprocessed version of the SWAN-SF data set, which includes FPCKNN imputation, LSBZM normalization, and the removal of classes B and C. The analysis also incorporates four combinations of oversampling and undersampling techniques applied to eight classifiers on a specific train–test combination, resulting in each box showing the outcome of 32 experiments. In the plot, diamond-shaped symbols indicate outliers.

Table 8
Mean TSS Values before and after Preprocessing, including Our Imputation (I), Our Normalization (N), NDBSR, and Sampling Techniques

Technique	RNN	1D-CNN	LSTM	GRU
No preprocessing	0.0 ± 0.0	0.0 ± 0.0	0.302 ± 0.180	0.0 ± 0.0
I and N and NDBSR and SMOTE	0.521 ± 0.116	0.533 ± 0.126	0.637 ± 0.109	0.625 ± 0.238
I and N and NDBSR and ADASYN	0.571 ± 0.151	0.648 ± 0.198	0.554 ± 0.149	0.795 ± 0.070
I and N and NDBSR and GNI	0.692 ± 0.189	0.591 ± 0.212	0.618 ± 0.188	0.605 ± 0.140
I and N and NDBSR and TimeGAN	**0.772** ± 0.041	0.413 ± 0.183	0.649 ± 0.224	0.718 ± 0.111
I and N and NDBSR and RUS-TL-SMOTE	0.558 ± 0.034	0.701 ± 0.104	0.710 ± 0.174	0.816 ± 0.033
I and N and NDBSR and RUS-TL-ADASYN	0.509 ± 0.132	**0.733** ± 0.086	**0.736** ± 0.074	**0.835** ± 0.031
I and N and NDBSR and RUS-TL-GNI	0.562 ± 0.271	0.601 ± 0.230	0.690 ± 0.201	0.676 ± 0.160
I and N and NDBSR and RUS-TL-TimeGAN	0.642 ± 0.252	0.602 ± 0.218	0.647 ± 0.153	0.689 ± 0.192

Note. The bold scores highlight the best results.

6. Conclusions

Through extensive experiments incorporating our FPCKNN imputation, LSBZM normalization, NDBSR, and eight distinct sampling techniques across eight classifiers, we have demonstrated a remarkable improvement in the TSS score following each preprocessing step. A mean TSS score of 0.835 was achieved using GRU as the classifier, supported by FPCKNN and LSBZM for imputation and normalization, with the exclusion of classes B and C as NDBSR and the application of RUS, TL, and ADASYN as sampling techniques, underscoring the pivotal importance of a robust preprocessing pipeline. This is particularly relevant in the field of solar flare prediction and when dealing with challenging

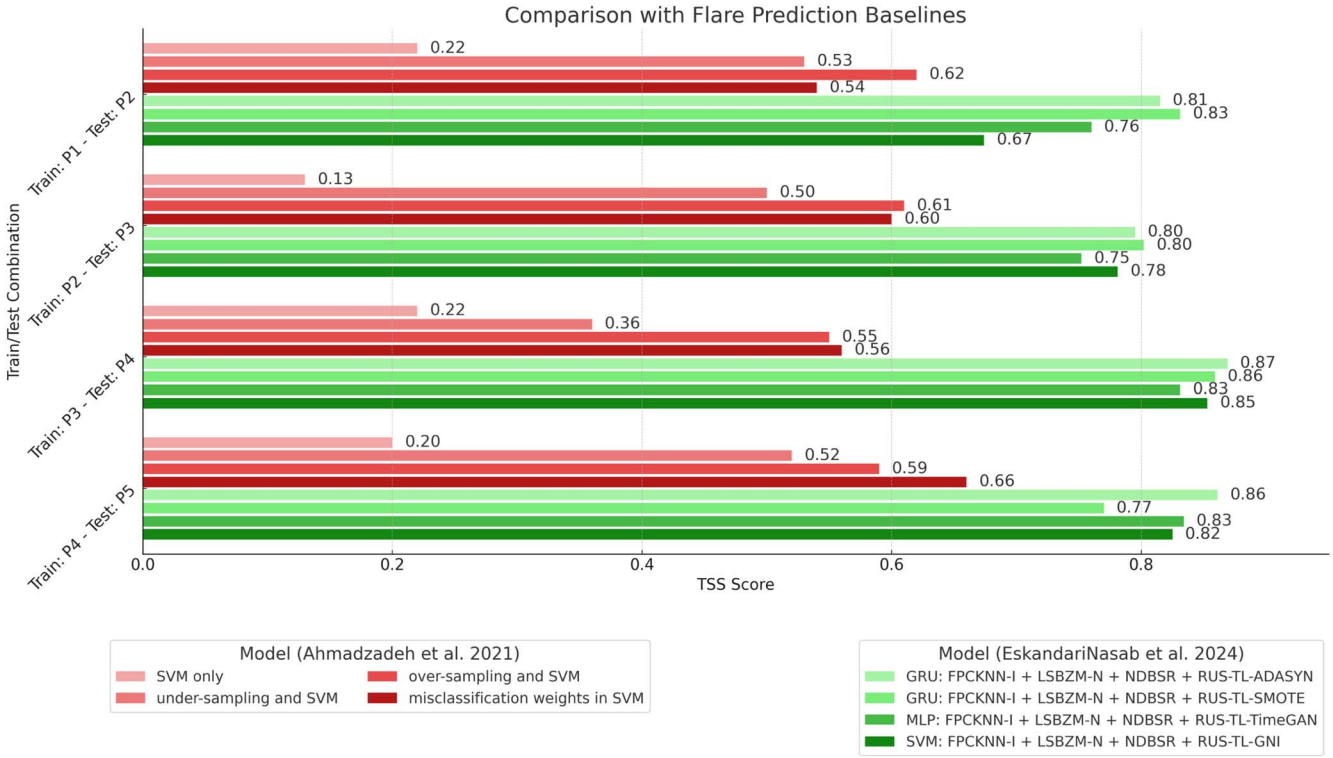

Figure 23. This figure displays a comparison of our best results, depicted in shades of green, against the results from A. Ahmadzadeh et al. (2021), shown in shades of red. "I" stands for imputation, "N" for normalization, and "P" for partition of the SWAN-SF data set. Each plot shows the best TSS achieved for both A. Ahmadzadeh et al. (2021) and our technique.

data sets such as SWAN-SF, which pose distinct preprocessing challenges. The combination of FPCKNN imputation and LSBZM normalization outperformed the baseline imputation and normalization techniques, demonstrating the efficacy of our imputation and normalization methods in improving classification performance. As an NDBSR technique, removing classes B and C from the minor-flaring category (FQ, B, and C) improved the TSS score compared to only removing class C or not addressing the class overlap issue at all. The combination of oversampling and undersampling techniques, specifically RUS, TL, and GNI (or SMOTE), yielded the most favorable outcomes, significantly enhancing TSS scores compared to the sole use of oversampling or the complete absence of sampling. When evaluating solely oversampling techniques, TimeGAN and SMOTE emerged as the top performers, outperforming scenarios where sampling was not utilized. In deep learning-based classifiers, the integration of RUS, TL, and ADASYN proved most effective. ADASYN's capacity to generate samples of the minority class, which are challenging to learn due to their scarcity, stood out. For the SWAN-SF data set, the MLP, SVM, GRU, and LSTM models were identified as the most suitable classifiers. For future work, we aim to integrate TimeGAN with adversarial autoencoders, striving to create a more accurate oversampling technique tailored for MVTS data. Furthermore, we aspire to devise a sophisticated deep learning-based method to diminish the class overlap within the SWAN-SF data set. Lastly, we plan to develop a deep learning-based imputation technique specifically for MVTS data, utilizing autoencoders.

Acknowledgments

This project received support from the Division of Atmospheric and Geospace Sciences within the Directorate for Geosciences, under NSF award Nos. 2301397, 2204363, and 2240022, as well as from the Office of Advanced Cyberinfrastructure within the Directorate for Computer and Information Science and Engineering, under NSF award No. 2305781.

Appendix A
Hyperparameters

Hyperparameters are a crucial aspect of any machine learning task. These parameters, adjusted before training, significantly influence the model's performance. By carefully tuning hyperparameters, such as learning rate and batch size, one can optimize the model to achieve maximum performance. Effective hyperparameter tuning leads to more accurate predictions and improved model stability, making it an essential step in developing high-performing machine learning models. Explaining hyperparameters is essential to ensure reproducibility in machine learning experiments. Therefore, in this section, we provide a comprehensive list of all the hyperparameters used in our sampling techniques and classifiers. By detailing these parameters, we aim to facilitate the replication of our work and contribute to the transparency and reliability of our research.

A.1. Sampling Techniques

Among our oversampling and undersampling techniques, only GNI and TimeGAN require hyperparameter tuning. For the other methods, such as SMOTE, ADASYN, RUS, and TL, no hyperparameter adjustments are necessary. In addition, for the oversampling techniques, we augment sufficient data for the minority class to match the amount of the majority class. However, when combining oversampling and undersampling

techniques, we reduce the majority-class samples to approximately 10,000, and augment the minority-class data to also reach approximately 10,000 samples.

1. *GNI.* We set the noise proportion (α) to 0.05 for the GNI algorithm.
2. *TimeGAN.* For the TimeGAN framework, the module is set to GRU, with a hidden dimension of 24. The framework utilizes three layers and runs for 8000 iterations. Additionally, the batch size is configured to 128.

A.2. Classifiers

We utilize eight classifiers to demonstrate the effectiveness of our preprocessing and sampling techniques. The hyperparameters for each classifier are explained in detail below.

1. *MLP.* The model consists of an input layer with 216 features (nine features for 24 attributes), followed by hidden layers with 64, 32, 16, and 8 units, respectively, each using ReLU activation. The output layer consists of a single unit with sigmoid activation for binary classification. The model is compiled with the Adam optimizer, binary cross-entropy loss, and precision at a recall of 0.95 as the evaluation metric. We train the model for 15 epochs with a batch size of 32. Predictions on the test data are then converted to binary outcomes using a threshold of 0.5.
2. *SVM.* For the SVM classifier, we use an RBF kernel with the regularization parameter *C* set to 1.0.
3. *k-NN.* For the k-NN classifier, we set the number of neighbors to 5.
4. *RF.* For the RF classifier, we use 64 estimators, set the random state to 42, and use entropy as the criterion.
5. *RNN.* The model consists of a simple RNN layer with 120 units and ReLU activation, with an input shape of (60, 24). This is followed by a dropout layer with a rate of 0.3. The subsequent layers include a dense layer with 120 units and ReLU activation, a dense layer with two units and ReLU activation, and a final dense layer with one unit and sigmoid activation for binary classification. The model is compiled with the Adam optimizer, binary cross-entropy loss, and specificity at a sensitivity of 0.95 as the evaluation metric.
6. *LSTM.* The model consists of an LSTM layer with 120 units and an input shape of (60, 24), followed by a dropout layer with a rate of 0.3. The subsequent layers include a dense layer with 120 units and ReLU activation, a dense layer with two units and ReLU activation, and a final dense layer with one unit and sigmoid activation for binary classification. The model is compiled with the Adam optimizer, binary cross-entropy loss, and specificity at a sensitivity of 0.95 as the evaluation metric.
7. *GRU.* The model consists of a GRU layer with 120 units and ReLU activation, with an input shape of (60, 24). This is followed by a dropout layer with a rate of 0.3. The subsequent layers include a dense layer with 120 units and ReLU activation, a dense layer with two units and ReLU activation, and a final dense layer with one unit and sigmoid activation for binary classification. The model is compiled with the Adam optimizer, binary cross-entropy loss, and specificity at a sensitivity of 0.95 as the evaluation metric.
8. *1D-CNN.* The model consists of a Conv1D layer with 64 filters, a kernel size of 5, and ReLU activation, with an input shape of (60, 24). This is followed by a Dropout layer with a rate of 0.2 and a MaxPooling1D layer with a pool size of 2. The next layer is another Conv1D layer with 128 filters, a kernel size of 5, and ReLU activation, followed by a dropout layer with a rate of 0.3 and another MaxPooling1D layer with a pool size of 2. The model then includes a flatten layer, a dense layer with 357 units and ReLU activation, a dense layer with 128 units and ReLU activation, a dense layer with two units and ReLU activation, and a final dense layer with one unit and sigmoid activation for binary classification. The model is compiled with the Adam optimizer, binary cross-entropy loss, and specificity at a sensitivity of 0.95 as the evaluation metric.

For the LSTM, GRU, RNN, and 1D-CNN models, we configure the parameters with the number of epochs set to 15 and the batch size to 32. Additionally, we evaluate predictions on the test data across various thresholds to determine the optimal one based on the TSS. This is done by iterating from 0.01 to 0.99 in increments of 0.01 and calculating the TSS for each threshold. The threshold yielding the highest TSS is selected as the optimal threshold.

Appendix B
Notations

Table 9 provides a detailed compilation of the notations employed throughout the document, aimed at ensuring concise writing and facilitating the efficient design of figures. This table serves as a reference point, enabling readers to quickly understand the shorthand terms used in the text and in the graphical representations, thereby enhancing readability and comprehension.

Table 9
List of Notations

Abbreviation	Complete Name	Abbreviation	Complete Name
SWAN-SF	Space Weather Analytics for Solar Flares	AR	Active region
NOAA	National Oceanic and Atmospheric Administration	HSS	Heidke skill score
GOES	Geostationary Operational Environmental Satellites	TSS	True skill statistic
HMI	Helioseismic Magnetic Imager	I	Imputation
SHARP	Spaceweather HMI Active Region Patch	N	Normalization
SDO	Solar Dynamics Observatory	FQ	Flare-quiet
FPCKNN	Fast Pearson correlation-based k-nearest neighbors	SVM	Support vector machine
LSBZM	Log, square root, Box–Cox, Z-score, min–max	k-NN	k-nearest neighbors
MLP	Multilayer perceptron	RF	Random forest
RNN	Recurrent neural networks	GRU	Gated recurrent unit
1D-CNN	1D convolutional neural network	LSTM	Long short-term memory
GCN	Graph convolution network	TP	True positive
NDBSR	Near decision boundary sample removal	TN	True negative
TimeGAN	Time series generative adversarial network	FP	False positive
ADASYN	Adaptive synthetic sampling	FN	False negative
PCA	Principal component analysis	FPR	False positive rate
SMOTE	Synthetic Minority Oversampling Technique	TL	Tomek links
GNI	Gaussian noise injection	ML	Machine learning
RUS	Random undersampling	NaN	Not a number
t-SNE	t-distributed stochastic neighbor embedding	MVTS	Multivariate time series
T	Number of time stamps	SQRT	Square root
PCC	Pearson correlation coefficient		
N	Number of magnetic field parameters (attributes)		

Appendix C
PCA Visualizations

After generating synthetic samples to augment the minority class, we proceed with a balanced selection strategy. This involves selecting an equal number of synthetic samples to match the count of the original minority class, roughly 1000 in number. Following this, we employ PCA to visualize both the original and synthetic samples (Figure 24). PCA is instrumental in mapping these samples onto a two-dimensional space, which allows for an intuitive comparison of their distributions. This visual representation is key to evaluating how closely the synthetic samples mimic the distribution of the original ones, providing insight into the effectiveness of the synthetic sample generation process in maintaining the underlying data characteristics.

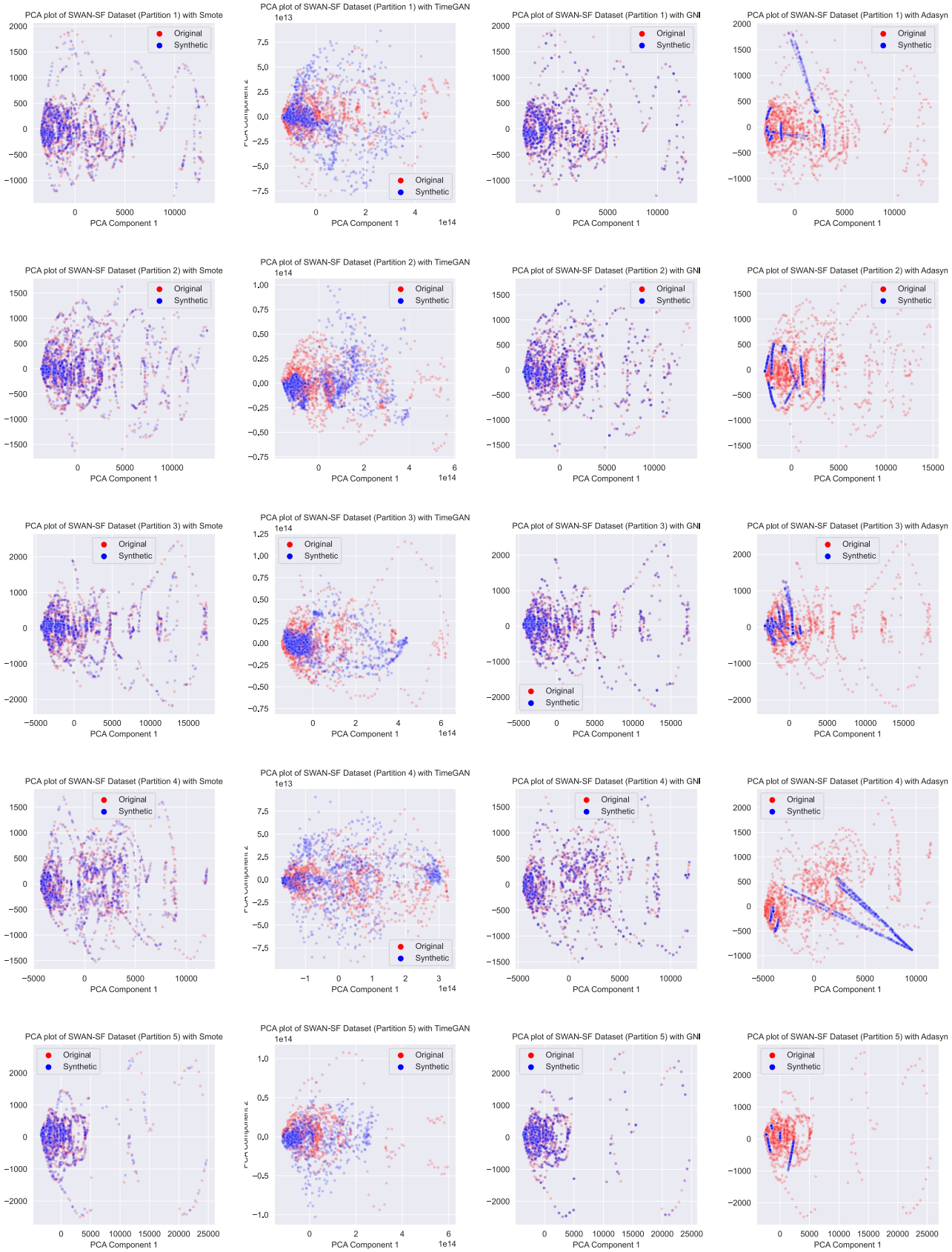

Figure 24. PCA visualizations demonstrating the distributional alignment of original and synthetic data samples for each oversampling technique across data set partitions. These visualizations highlight the efficacy of each oversampling method in replicating the comprehensive distribution characteristics of the original data samples.

ORCID iDs

MohammadReza EskandariNasab https://orcid.org/0009-0004-0697-3716
Shah Muhammad Hamdi https://orcid.org/0000-0002-9303-7835
Soukaina Filali Boubrahimi https://orcid.org/0000-0001-5693-6383

References

Ahmadzadeh, A., Aydin, B., Georgoulis, M. K., et al. 2021, ApJS, 254, 23
Alshammari, K., Hamdi, S. M., Muzaheed, A. A., & Filali Boubrahimi, S. 2022, CEUR Workshop Proc. 3375, Workshop on Applied Machine Learning Methods for Time Series Forecasting (AMLTS 2022), ed. W. Liu & L. Pang, (CEUR Workshop Proceedings), https://ceur-ws.org/Vol-3375/paper4.pdf
Angryk, R. A., Martens, P. C., Aydin, B., et al. 2020, NatSD, 7, 227
Anil Jadhav, D. P., & Ramanathan, K. 2019, Appl. Artif. Intell., 33, 913
Aschwanden, M. J., Crosby, N. B., Dimitropoulou, M., et al. 2016, SSRv, 198, 47
Behfar, A., Atashpanjeh, H., & Al-Ameen, M. N. 2023, in CSCW '23 Companion Publication of the 2023 Conf. on Computer Supported Cooperative Work and Social Computing, ed. C. Fiesler (New York: Association for Computing Machinery), 164
Bobra, M. G., & Couvidat, S. 2015, ApJ, 798, 135
Bobra, M. G., Sun, X., Hoeksema, J. T., et al. 2014, SoPh, 289, 3549
Breiman, L. 2001, Mach. Learn., 45, 5
Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. 2002, J. Artif. Intell. Res., 16, 321
Chen, J., Li, W., Li, S., et al. 2022, SpScT, 2022, 9761567
Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. 2014, arXiv:1412.3555
Cortes, C., & Vapnik, V. 1995, Mach. Learn., 20, 273
Curto, J. J. 2020, JSWSC, 10, 27
Dhakal, S. K., & Zhang, J. 2023, ApJ, 960, 36
Emmanuel, T., Maupong, T., Mpoeleng, D., et al. 2021, J. Big Data, 8, 140
EskandariNasab, M., Hamdi, S. M., & Boubrahimi, S. F. 2024a, Cleaned SWANSF Dataset, v1.0.0, Zenodo, doi:10.5281/zenodo.11566472
EskandariNasab, M., Hamdi, S. M., & Boubrahimi, S. F. 2024b, SWAN-SF Data Preprocessing and Sampling Notebooks, v1.0.0, Zenodo, doi:10.5281/zenodo.11564789
EskandariNasab, M., Raeisi, Z., Lashaki, R. A., & Najafi, H. 2024c, NatSR, 14, 8861
Feng, C., Wang, H., Lu, N., et al. 2014, Shanghai Arch. Psychiatry, 26, 105
Fisher, G. H., Bercik, D. J., Welsch, B. T., & Hudson, H. S. 2012, Solar Flare Magnetic Fields and Plasmas (New York: Springer), 59
Gardner, M. W., & Dorling, S. R. 1998, AtmEn, 32, 2627
Georgoulis, M. K. 2012, SoPh, 276, 161
Hamdi, S. M., Ahmad, A. F., & Boubrahimi, S. F. 2022, CEUR Workshop Proc. 3375, Workshop on Applied Machine Learning Methods for Time Series Forecasting (AMLTS 2022), ed. W. Liu & L. Pang, (CEUR Workshop Proceedings), https://ceur-ws.org/Vol-3375/paper3.pdf
Hamdi, S. M., Kempton, D., Ma, R., Boubrahimi, S. F., & Angryk, R. A. 2017, in Proc. of the IEEE Int. Conf. on Big Data (BigData) (Piscataway, NJ: IEEE), 2543
He, H., Bai, Y., Garcia, E. A., & Li, S. 2008, in 2008 IEEE Int. Joint Conf. on Neural Networks (IEEE World Congress on Computational Intelligence) (Piscataway, NJ: IEEE), 1322
Hochreiter, S., & Schmidhuber, J. 1997, Neural Comput., 9, 1735
Hoeksema, J. T., Liu, Y., Hayashi, K., et al. 2014, SoPh, 289, 3483
Hosseinzadeh, P., Filali Boubrahimi, S., & Hamdi, S. M. 2024, ApJS, 270, 31
Khan, S. I., & Hoque, A. S. M. L. 2020, J. Big Data, 7, 37
Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. 1998, IEEEP, 86, 2278
Leka, K. D., & Barnes, G. 2003, ApJ, 595, 1296
Leka, K. D., Park, S.-H., Kusano, K., et al. 2019, ApJS, 243, 36
Leka, K. D., & Skumanich, A. 1999, SoPh, 188, 3
Menzel, W. P., & Purdom, J. F. W. 1994, BAMS, 75, 757
Muhammad Ali, P., & Faraj, R. 2014, Data Normalization and Standardization: A Technical Report, Machine Learning Technical Reports doi:10.13140/RG.2.2.28948.04489
Muzaheed, A. A. M., Hamdi, S. M., & Boubrahimi, S. F. 2021, in Proc. of the 20th IEEE Int. Conf. on Machine Learning and Applications (ICMLA) (Piscataway, NJ: IEEE), 435
Nishizuka, N., Sugiura, K., Kubo, Y., Den, M., & Ishii, M. 2018, ApJ, 858, 113
Nishizuka, N., Sugiura, K., Kubo, Y., et al. 2017, ApJ, 835, 156
Pesnell, W. D., Thompson, B. J., & Chamberlin, P. C. 2012, SoPh, 275, 3
Peterson, L. E. 2009, SchpJ, 4, 1883
Sakia, R. M. 1992, J. R. Stat. Soc. D, 41, 169
Scherrer, P. H., Schou, J., Bush, R. I., et al. 2012, SoPh, 275, 207
Schou, J., Scherrer, P. H., Bush, R. I., et al. 2012, SoPh, 275, 229
Schrijver, C. J. 2007, ApJL, 655, L117
Sherstinsky, A. 2020, PhyD, 404, 132306
Singh, D., & Singh, B. 2020, Appl. Soft Comput., 97, 105524
Sinha, S., Gupta, O., Singh, V., et al. 2022, ApJ, 935, 45
Sun, Z., Bobra, M. G., Wang, X., et al. 2022, ApJ, 931, 163
Tomek, I. 1976, ITSMC, SMC-6, 769
Troyanskaya, O., Cantor, M., Sherlock, G., et al. 2001, Bioin, 17, 520
van der Maaten, L., & Hinton, G. 2008, JMLR, 9, 2579, https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf
Wang, J., Shi, Z., Wang, H., & Lue, Y. 1996, ApJ, 456, 861
Yoon, J., Jarrett, D., & van der Schaar, M. 2019, Advances in Neural Information Processing Systems 32 (NeurIPS 2019), ed. H. Wallach et al. (NeurIPS), https://papers.nips.cc/paper_files/paper/2019/hash/c9efe5f26cd17ba6216bbe2a7d26d490-Abstract.html
Zhang, J., Wang, T., Ng, W. W. Y., Zhang, S., & Nugent, C. D. 2019, in 2019 Int. Conf. on Machine Learning and Cybernetics (ICMLC) (Piscataway, NJ: IEEE)