Draft version March 17, 2024
Typeset using LaTeX default style in AASTeX631

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

Anonymous

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 to the *Space Weather Analytics for Solar Flares (SWAN-SF)* dataset, a rich source of multivariate time series data of solar active regions. Our study adopts a multi-faceted approach encompassing four key methodologies. Initially, we address over 10 million missing values in the SWAN-SF dataset through our innovative imputation technique called Fast Pearson Correlation-based K-nearest neighbors imputation (FPCKNN imputation). Subsequently, we propose a precise normalization technique, called LSBZM normalization, tailored for time series data, merging various strategies (Log, Square Root, BoxCox, Z-score, and Min-Max) to uniformly scale the dataset'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 dataset by effectively resolving the challenge of class overlap. Finally, a pivotal aspect of our research is a thorough evaluation of diverse over-sampling and under-sampling methods, including SMOTE, ADASYN, Gaussian Noise Injection, TimeGAN, Tomek-links, and Random Under Sampling, to counter the severe imbalance in the SWAN-SF dataset, 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 outstanding True Skill Statistics (TSS) scores, underscoring the importance of data preprocessing and sampling in time series-based solar flare prediction.

Keywords: Data Preprocessing — Sampling — Data Augmentation — Normalization — Missing Value Imputation — Multivariate Time Series Classification — Solar Flare Prediction

1. INTRODUCTION

Since 1974, the National Oceanic and Atmospheric Administration's (NOAA) Geostationary Operational Environmental Satellites (GOES) have been crucial in the automated detection and classification of solar flares within the 1–8 Å wavelength spectrum. These events are sudden and significant increases in radiation, encompassing the entire electromagnetic spectrum, including its high-energy portions such as extreme-ultraviolet, X-rays, and gamma-rays. Occurring in localized areas on the Sun, solar flares represent a serious hazard to humans and equipment in space due to their intense radiation. The classification of these flares is logarithmically based on their peak soft X-ray flux within this wavelength range, with categories labeled as FQ, A, B, C, M, and X, in ascending order of intensity, starting from a flux of $10^{-8} W/m^2$. 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 than a C-class flare. Each class further includes a subclass designation ranging from 1.0 to 9.9, allowing for finer distinctions within each category. However, when the X-ray background level is high, typically during periods of elevated solar activity, detecting A- and B-class flares becomes challenging or even impossible due to their relatively lower intensity. 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 (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 active regions (AR) produced at least one C-class flare, but less than 2% produced an X-class flare (Georgoulis 2012). Moreover, solar cycle 24 (2009 to present) showed a similar trend in major flare occurrences despite having a similar

number of AR as cycle 23, emphasizing the criticality of addressing the class imbalance in flare prediction (Angryk et al. 2020). In response to these challenges, recent research has focused on data science-based approaches, particularly using the spatiotemporal magnetic field data provided by the Helioseismic Magnetic Imager (HMI) installed in the Solar Dynamics Observatory (SDO) (Ahmadzadeh et al. 2021). This data is transformed into multivariate time series (MVTS) instances for temporal window-based flare prediction (Angryk et al. 2020). In these instances, various solar magnetic field parameters are represented 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 AR parameter values are measured). The MVTS instances are then labeled with one of five classes, ranging from flare-quiet (FQ) regions (including both flare-quiet and A-class) to the highly intense X-class flares. This classification approach has shown enhanced accuracy in predicting flares compared to models that use single timestamp-based magnetic field vector classification. A pivotal resource in this area of research is the *Space Weather Analytics for Solar Flares (SWAN-SF)* dataset (Angryk et al. 2020), derived from solar photospheric vector magnetograms by the Spaceweather HMI Active Region Patch (SHARP) series. This dataset is highlighted by its accurate reflection of the class imbalance 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. Such significant class imbalances emphasize the infrequency of high-intensity flares and the necessity of developing sophisticated machine learning-based classifiers.

Data collected to address real-world problems is seldom clean or ready for immediate use, regardless of the thoroughness of the screening process. Such datasets often inherit challenges related to the nature of the subject under study or the data collection strategy. These challenges, which include missing values, multi-scaled attributes, skewness, class overlap, and class imbalance, are prevalent in many nonlinear dynamical systems such as solar energetic particle event prediction (Hosseinzadeh et al. 2024). This study revisits these challenges in the context of solar flare prediction, a natural manifestation of these issues. In this study, we concentrate on the effects of data preprocessing and sampling techniques on enhancing solar flare prediction performance. Traditional forecast metrics for performance evaluation in flare-forecasting models include the True Skill Statistic (TSS) and various forms of the Heidke Skill Score (HSS; HSS2) (Bobra & Couvidat 2015). Prior research in MVTS-based solar flare prediction primarily focused on classifying magnetic field time series of AR into different flare classes (Ahmadzadeh et al. 2021). Approaches ranged from statistical summarization of individual time series for low-dimensional representations of MVTS instances, employing classifiers such as SVM for binary flare prediction (Hamdi et al. 2017), to utilizing end-to-end LSTM-based deep sequence models for classifying different flare classes (Muzaheed et al. 2021). Some studies also forecasted future magnetic field parameter values based on past MVTS representations (Alshammari et al. 2022). However, this paper focuses on enhancing classification performance specifically in the context of solar flare prediction through novel, comprehensive data preprocessing and sampling techniques. It addresses challenges within the SWAN-SF dataset, selected for its known issues, including over 10 million missing values, significant class overlap and imbalance, and heterogeneity in attribute magnitudes across its photospheric magnetic field parameters (24 attributes), as detailed in Table 1. Techniques such as missing value imputation, crucial for creating a complete dataset, and normalization, essential for harmonizing the magnitudes of heterogeneous attributes, play a key role in our methodology. They assist in resolving skewness, facilitating faster learning, and ensuring numerical stability. Moreover, addressing class overlap and employing sampling techniques such as over-sampling and under-sampling are essential for managing class imbalance, enhancing minority-class representation, and reducing majority-class dominance. We evaluate our approaches by applying them to various classifiers, aiming to determine their effects across a broad range of models. Our contributions are as follows:

1. A Fast Pearson Correlation-based K-nearest neighbors imputation (FPCKNN imputation) technique for missing values, ensuring data quality and realism.
2. A comprehensive normalization technique tailored for time series data, merging various strategies to uniformly scale the dataset's 24 attributes (photospheric magnetic field parameters), addressing skewness.
3. Assessing 'Near Decision Boundary Sample Removal' techniques to enhance classification performance by resolving class overlap.
4. A thorough evaluation of diverse over-sampling and under-sampling methods to counter the significant imbalance in the SWAN-SF dataset.

In Figure 1, we summarize all the preprocessing techniques we propose, along with their order of importance, to improve the performance of classification on the SWAN-SF dataset.

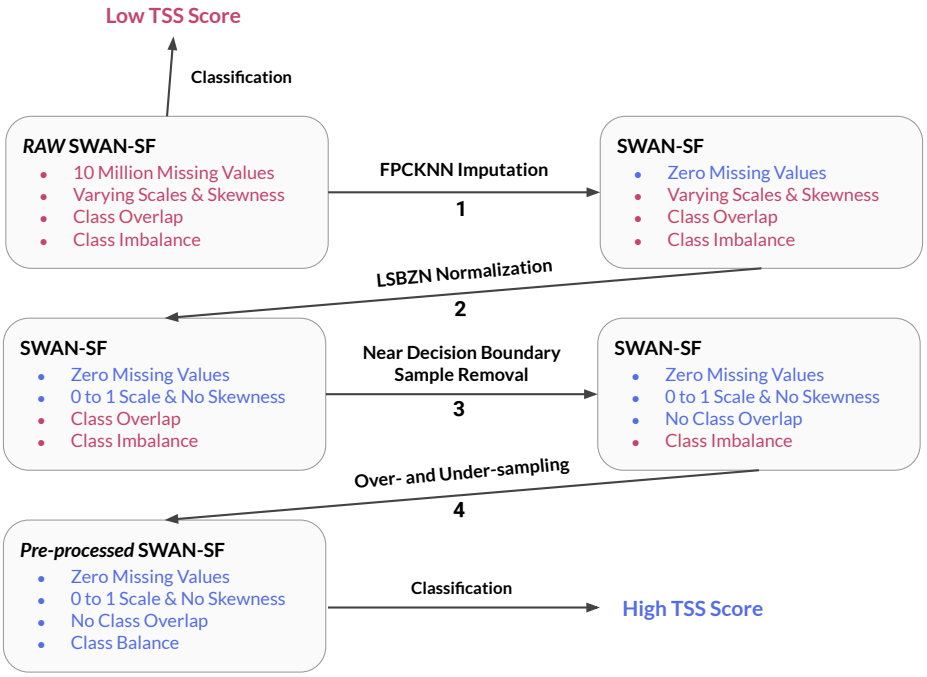

Figure 1. Our four essential data preprocessing and sampling techniques to enhance classification performance on the SWAN-SF dataset

We implemented two distinct approaches for classification: the first involves summarizing each time series by extracting predefined statistical features, which are then used to train traditional classifiers including Support Vector Machine (SVM) (Cortes & Vapnik 1995), Multilayer Perceptron (MLP) (Gardner & Dorling 1998), K-nearest Neighbors (k-NN) (Peterson 2009), and Random Forest (RF) (Breiman 2001). The second approach processes all attributes of the time series using advanced models such as Long Short-Term Memory (LSTM) network (Hochreiter & Schmidhuber 1997), Recurrent Neural Network (RNN) (Sherstinsky 2020), Gated Recurrent Unit (GRU) (Chung et al. 2014), and 1D Convolutional Neural Network (1D-CNN) (Lecun et al. 1998), eliminating the necessity for feature engineering. Our results, as indicated by competitive TSS scores, underscore the significance of data preprocessing and sampling in solar flare prediction. We also release a preprocessed version of the SWAN-SF dataset as a resource for further research. This paper is organized as follows: Section 2 reviews related studies and works. Section 3 goes into detail about the SWAN-SF dataset and its challenges. In Section 4, we introduce our method for imputation and normalization. Section 5 and 6 discusses our sampling techniques. Section 7 presents the results of our data preprocessing and sampling methods and discusses the classification algorithms used for MVTS classification. Finally, section 8 concludes our paper with a conclusion, and a look toward future work. For discussion, notations, and additional figures, refer to the sections A, B, and C.

2. RELATED WORK

Previous studies in solar flare prediction have primarily focused on the development and optimization of machine-learning algorithms to enhance prediction performance. For instance, (Bobra & Couvidat 2015) utilized photospheric vector magnetograms to forecast solar flares using an SVM classifier. They highlighted the importance of a robust set of features derived from vector magnetic field data, achieving significant prediction accuracy. Similarly, (Hamdi et al. 2017) presented a flare prediction method by extracting time series samples of Active Region parameters and employing a k-NN classification of the univariate time series. They discovered that for the classification task, a statistical summarization of the time series for a single Active Region parameter, specifically the *total unsigned current helicity*,

was more effective than using all Active Region parameters at a single instant of time. Furthermore, (Ahmadzadeh et al. 2021) addressed specific challenges in solar flare forecasting such as class imbalance and temporal coherence. They discussed strategies such as under-sampling and over-sampling to manage class imbalance in the SWAN-SF dataset and emphasized the necessity of proper data splitting and validation techniques to ensure model robustness against temporal coherence. These studies collectively enhance our understanding of solar flare forecasting, providing a foundation for further research in this domain. Moreover, (Muzaheed et al. 2021) leveraged Long Short Term Memory (LSTM) networks for effective end-to-end classification of multivariate time series in solar flare prediction, outperforming traditional models and highlighting the potential of deep learning. In a similar vein, (Hamdi et al. 2022) developed a novel approach combining Graph Convolution Networks (GCN) with LSTM networks, effectively capturing both spatial and temporal relationships in solar flare prediction and surpassing other baseline methods. (Alshammari et al. 2022) further expanded the field by focusing on forecasting magnetic field parameters related to solar events using a deep sequence-to-sequence learning model with batch normalization and LSTM networks, marking a significant advance over traditional sequence models.

The study by (Hosseinzadeh et al. 2024) explores improving Solar Energetic Particle (SEP) event prediction through data augmentation techniques applied to multivariate time series of proton flux data. It focuses on predicting 30, 60, and 100 MeV SEP events using machine learning-based prediction methods, notably the Time Series Forest (TSF) model. The research demonstrates that data augmentation, specifically the Synthetic Minority Over-sampling Technique (SMOTE), significantly increases the accuracy and F1-score of classifiers, with notable performance in the 100 MeV SEP prediction task. The use of multivariate time series data further enhanced prediction accuracy. A comprehensive hierarchical classification framework for SEP and non-SEP scenarios was developed, showcasing a marked improvement in prediction capabilities through synthetic data expansion and advanced classification models.

Unlike previous research on solar flare prediction that primarily concentrated on enhancing classification methodologies, our study adopts an innovative approach by emphasizing the improvement of dataset quality. Acknowledging the critical importance of data preprocessing for prediction accuracy, we explore advanced data preprocessing and sampling strategies to refine the SWAN-SF dataset. This approach is designed to achieve superior classification performance, representing a significant shift from the conventional focus on algorithmic improvements towards a foundational focus on dataset optimization. Furthermore, we intend to undertake a comprehensive investigation into advanced sampling techniques, including the utilization of state-of-the-art Generative Adversarial Networks, to address the imbalance of the SWAN-SF dataset and analyze their impact. Additionally, we also aim to tackle the issue of class imbalance by implementing under-sampling strategies to diminish the predominance of the minor-flaring class in the binary classification of solar flares.

3. DATASET AND PRELIMINARIES

3.1. *SWAN-SF Benchmark Dataset*

In our study, the *Space Weather Analytics for Solar Flares (SWAN-SF)* dataset (Angryk et al. 2020) is employed as a foundational dataset for solar flare prediction, leveraging multivariate time series (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 flare-quiet and GOES A-class events. The dataset is segmented into five partitions, each part maintaining a roughly equal distribution of X- and M-class flares and aligning with a specific active region (AR) timeframe. These partitions are organized in a sequential manner, as shown in Figure 2. Figure 3 displays the class distribution across these partitions. The dataset 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 AR, as listed in Table 1. These series are recorded at 12-minute intervals over 12 hours, 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) AR.

3.2. *Missing Values in SWAN-SF*

In the realm of time series classification, particularly in the context of solar flare prediction using SWAN-SF multivariate time series 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 datasets such as SWAN-SF,

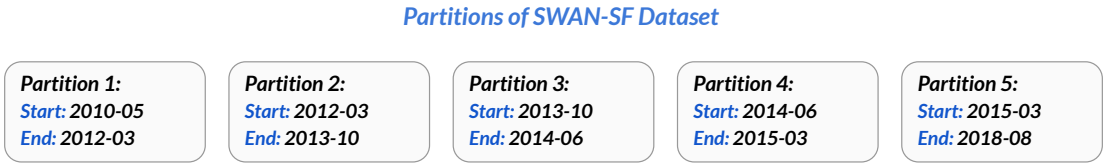

Figure 2. This figure presents a timeline indicating the occurrence of AR within each partition of SWAN-SF. The first number represents the year, while the second number denotes the month.

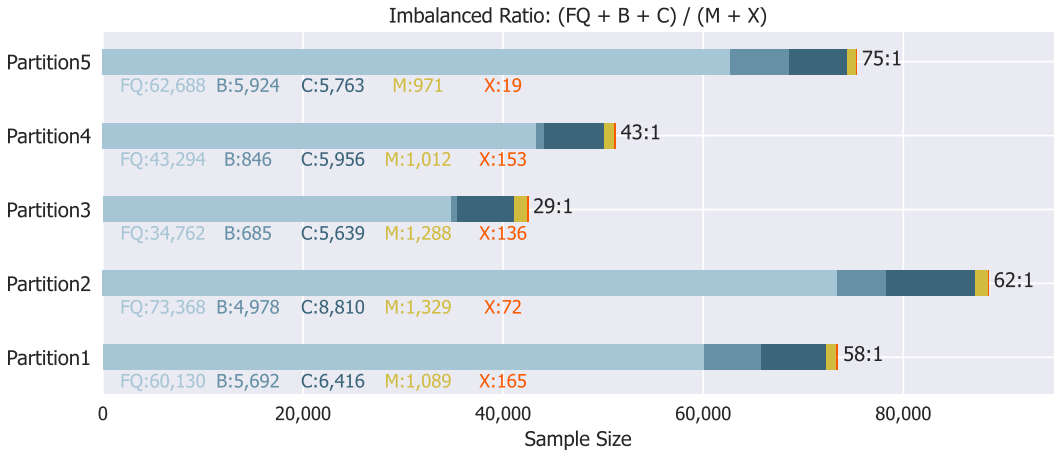

Figure 3. A stacked bar chart representing the population distribution of various flare classes in each partition of the SWAN-SF benchmark dataset, over time. Flare classes X, M, C, and B are derived and confirmed using the GOES classification, whereas FQ represents instances of flare-quiet and those 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 hour, an observation period of 12 hours, and a prediction span of 24 hours. Each slice of the multivariate time series is categorized according to the most intense flare reported within its prediction window.

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 NaN (Not a Number) 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 (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 datasets; missing values, if not adequately addressed, can skew model outcomes, resulting in inaccurate classifications (Emmanuel et al. 2021).

The choice between imputation methods should be guided by the specific characteristics of a dataset 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 (Anil Jadhav & Ramanathan 2019). In dealing with imbalanced datasets 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 1,000 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 dataset. This approach not only maintains the integrity of the dataset 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.

Table 1. Active region magnetic field parameters in SWAN-SF dataset.

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

Table 2. Missing value distribution for active region magnetic field parameters in SWAN-SF dataset. For the attributes that are not listed in this table, there are 0 missing values.

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	2,718	4,844	4,964
TOTFX	652	93,300	2,718	4,844	4,964
TOTFY	652	93,300	2,718	4,844	4,964
TOTFZ	652	93,300	2,718	4,844	4,964
TOTPOT	652	93,300	2,718	4,844	4,964
TOTUSJH	652	93,300	2,718	4,844	4,964
TOTUSJZ	652	93,300	2,718	4,844	4,964
USFLUX	652	93,300	2,718	4,844	4,964
ABSNJZH	652	93,300	2,718	4,844	4,964
SAVNCPP	652	93,300	2,725	4,844	4,964

3.3. *Multi-scaled Features in SWAN-SF*

Normalization is a critical preprocessing step in machine learning and data analysis, crucial for managing datasets 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 dataset. 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 dataset'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 multivariate time series, where each time series has unique characteristics. 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.

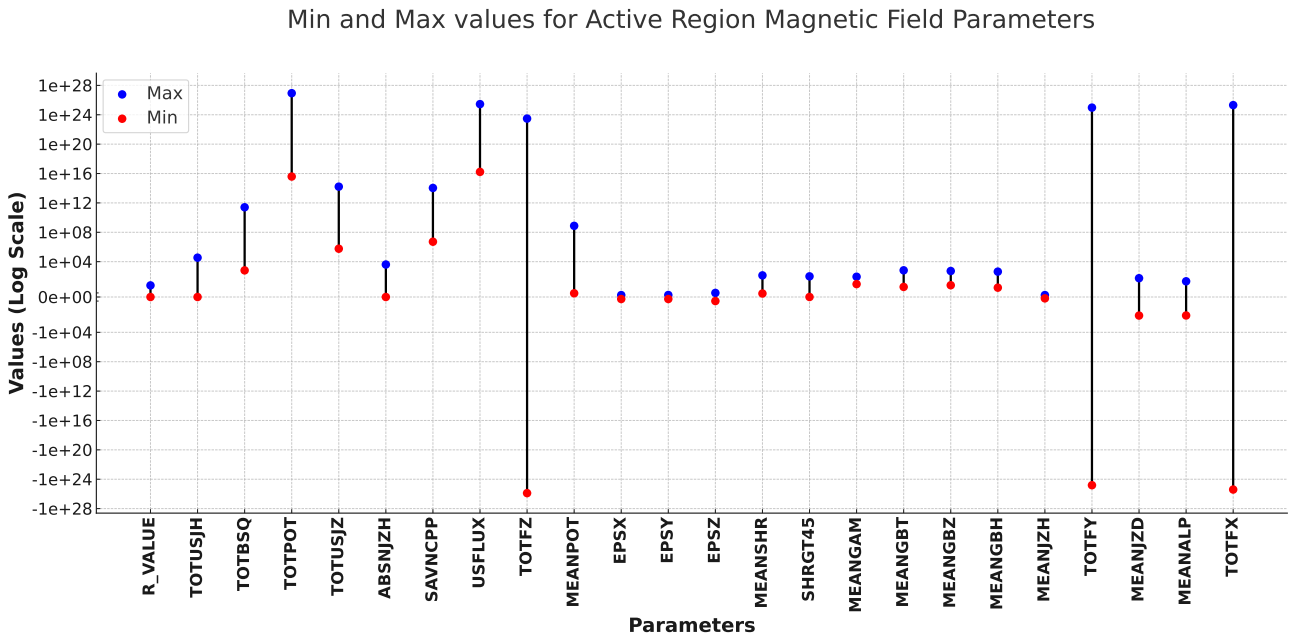

Figure 4. This plot demonstrates the significant variation in minimum and maximum values across different attributes of the SWAN-SF dataset. The diverse scales underscore the need for an advanced normalization technique to effectively analyze and process the data.

3.4. *Class Overlap in SWAN-SF*

Class overlap in datasets 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 overfit to the training data, excessively adjusting 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 'Near Decision Boundary Sample Removal' (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.

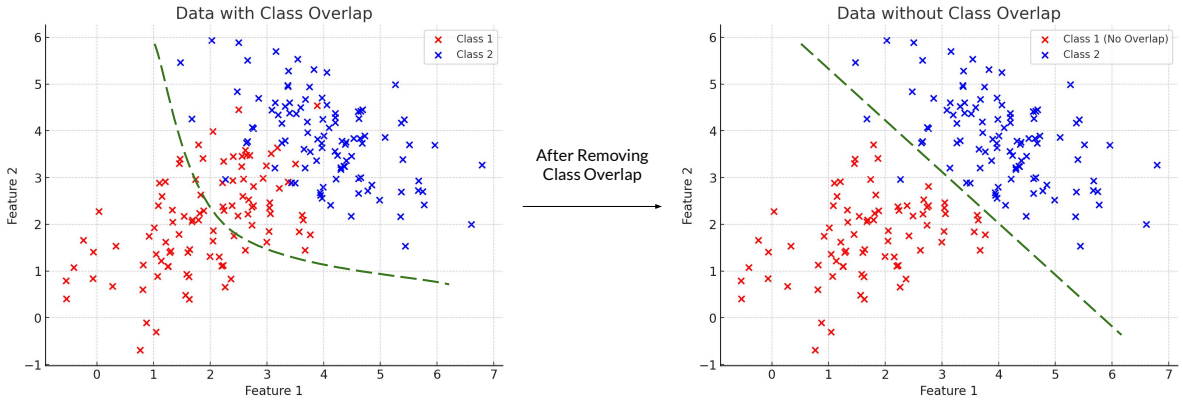

Figure 5. These scatter plots illustrate the distribution of two classes and their overlap. The dashed line highlights a decision boundary built by a classifier. It demonstrates the impact of class overlap on the complexity and effectiveness of the decision boundary and therefore the potential for higher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

Since the mid-1970s, the detection and categorization of X-ray flares have been conducted by the 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 $10^{-8}\,\mathrm{Wm}^{-2}$. An X-class flare exceeds an M-class 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. As a result of this class overlap, the TSS for binary classification is likely to be low. This overlap is a primary factor contributing to the difficulty in achieving a TSS higher than 0.9 in the binary classification of Flares in the SWAN-SF dataset.

3.5. *Class Imbalance in SWAN-SF*

We identify a dataset 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 dataset, 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 for the minority class and a reduced number of true positives. 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 minfority class.

4. 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 dataset and inherently lack the capability to process missing data (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 (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 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 (Singh & 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 multivariate time-series classification, especially in predicting solar flares, the importance of these processes is amplified. Multivariate time-series 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.

4.1. *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 (Emmanuel et al. 2021). In specific contexts such as solar flare prediction, where datasets 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 dataset's integrity. Imputation, therefore, emerges as a preferable strategy, aiming to create a more comprehensive dataset for the benefit of future research. When it comes to imputing missing values in a multivariate time series, the choice of technique is crucial, yet there's no one-size-fits-all solution. For instance, mean imputation might be a convenient approach for filling one or two consecutive missing timestamps in a univariate time series attribute. However, its applicability reduces significantly in scenarios where multiple consecutive timestamps 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 dataset'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 datasets. Techniques that offer a balance between imputation quality and computational demand are often preferred in practical applications. In the case of multivariate time series, 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 dataset

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 dataset is a multifaceted decision, balancing statistical properties, dataset characteristics, and practical considerations. It often involves a trade-off between various factors, including accuracy, robustness, efficiency, and usability.

4.1.1. *Fast Pearson Correlation-based K-nearest neighbors (FPCKNN) Imputation*

In the SWAN-SF dataset, 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 dataset. 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 (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 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 timestamps from the K nearest neighbors. This approach ensures the preservation of the time series' temporal dynamics.

In a multivariate time series dataset such as SWAN-SF, there are two main types of missing values. The first type occurs when all the values of an attribute within a multivariate time series 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 a multivariate time series are missing. In these cases, we identify the most similar multivariate time series samples to the one with missing values. In other words, we perform 1-nearest neighbors 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 Pearson Correlation Coefficient (PCC) to calculate the similarity between two multivariate time series. 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 datasets 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 datasets. The computation of the Pearson correlation coefficient between two multivariate time series 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 X 9.9 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 M 9.9 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, X 9.9, and advance sequentially to X 1.0. Subsequently, proceed from M 9.0 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 Pearson Correlation Coefficient (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 Pearson Correlation Coefficient (PCC) calculations to the 50 previous samples, rather than all 60,000 samples in the first partition of the SWAN-SF dataset. 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.

Algorithm 1 Pearson Correlation Coefficient calculation for two Multivariate Time Series

Require: two multivariate time series X and Y $(X, Y \in \mathbb{R}^{(T \times N)})$
Ensure: pearson correlation coefficient between X and Y
$FlattenedX \leftarrow \text{Flatten}(\text{Transpose}(X))$
$FlattenedY \leftarrow \text{Flatten}(\text{Transpose}(Y))$
$meanX \leftarrow \text{mean}(FlattenedX)$
$meanY \leftarrow \text{mean}(FlattenedY)$
$stdX \leftarrow \text{standard\_deviation}(FlattenedX)$
$stdY \leftarrow \text{standard\_deviation}(FlattenedY)$
$covariance \leftarrow \text{mean}((FlattenedX - meanX) \times (FlattenedY - meanY))$
$CorrCoef \leftarrow \frac{covariance}{stdX \times stdY}$
return $CorrCoef$

4.2. *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 datasets 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 dataset (Singh & 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 (Singh & Singh 2020).

In the context of the SWAN-SF dataset, which is a multivariate time series dataset 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 towards 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 datasets 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 (Feng et al. 2014). It transforms data into a more normally distributed dataset.

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

Algorithm 2 Fast Pearson Correlation-based K-nearest neighbors (FPCKNN) Imputation

Require: dataset *data*, number of partitions $numParts$, number of attributes $numAttrs$, number of timestamps $numTimes$, comparison limit $kMax$
Ensure: imputed *data*

for $partIdx = 1$ **to** $numParts$ **do**
  $partition \leftarrow data[partIdx]$ {The shape of partition is $(numTimes, numAttrs,$ numSamples)}
  for $sampleIdx = 1$ **to** shape($partition$)[2] **do**
    $currSample \leftarrow partition[:, :, sampleIdx]$
    Fill in all the missing values in $currSample$ with NaN
    Initialize list $missingIdx$
    if NaN exists in $currSample$ **then**
      $compCount \leftarrow \min(sampleIdx, kMax)$
      Initialize array $corrCoeff$ of size $compCount$ with -2.0
      $sampleX \leftarrow$ replace NaN with 0 in $currSample$
      for $compIdx = 1$ **to** $compCount$ **do**
        $sampleY \leftarrow partition[:, :, sampleIdx - compIdx]$
        $corrCoeff[compIdx] \leftarrow$ 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 Pearson Correlation Coefficient score in $corrCoeff$
        else
          Perform k-NN imputation for $currSample[:, attrIdx]$ based on the samples that have the highest Pearson Correlation Coefficient scores in $corrCoeff$
        end if
      end for
    end if
    $partition[:, :, sampleIdx] \leftarrow currSample$
  end for
  $data[partIdx] \leftarrow partition$
end for
return *data*

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 datasets with zero or negative values.

2. Square Root Normalization: Effective for reducing left skewness in data (Muhammad Ali & 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 dataset to ensure positivity, especially if the dataset includes negative values.

3. Box-Cox Normalization: (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.

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

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 non-zero. 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:

- When $\lambda = 0$, the transformation becomes a logarithmic transformation.
- As λ approaches 1, the transformation becomes closer to the identity transformation, where no change is applied to the data.
- 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.

4. Z-score Normalization (Standardization): Used for normalizing the distribution of values in a dataset, particularly effective for data without skewness.

$$z = \frac{x - \mu}{\sigma} \tag{3}$$

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.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{4}$$

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.

4.2.1. *LSBZM (Log, Square root, BoxCox, Z-score, and MinMax) Normalization*

The SWAN-SF dataset, being an MVTS dataset, 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 attribute of all samples. Figure 6 clarifies our methodology for normalizing a MVTS dataset, a critical step especially when calculating the mean or standard deviation of the values, such as the implementation of the Z-score normalization technique.

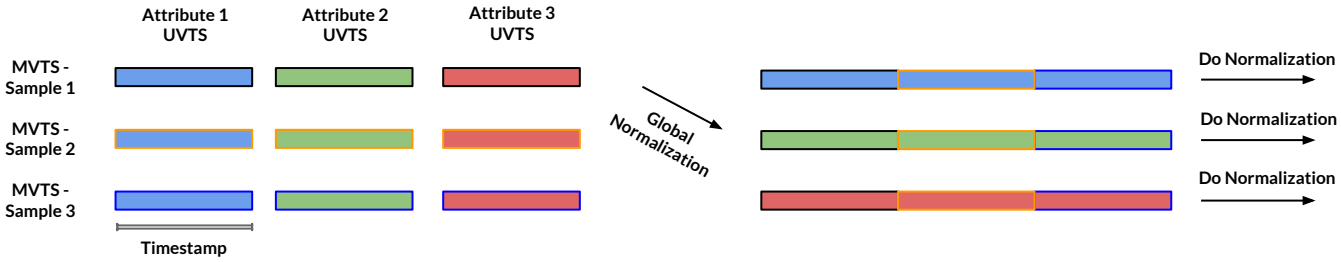

Figure 6. The figure illustrates the application of global normalization on the SWAN-SF dataset, a process designed to maintain the magnitude of each attribute's values in comparison to other samples. This approach is particularly crucial given that in the SWAN-SF dataset, the significance lies in the magnitude of the attribute values.

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 the paper provides a detailed description of these conditions and the corresponding normalization technique.

5. NEAR DECISION BOUNDARY SAMPLE REMOVAL (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 (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 sub-optimal 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 Near Decision Boundary Sample Removal (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 (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 dataset presents significant challenges for classification due to extensive overlap between classes, affecting both binary and multi-class classification. This complexity stems from the dataset'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, X 1.0 samples are quite similar to M 9.9 samples. However, this overlap is more complex than it appears at first glance. For instance, X 1.0 samples might resemble samples ranging from M 9.9 down to M 2.0. 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 (Bobra & 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 dataset. 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 classes 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 post-imputation 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, we can improve the TSS scores.

6. SAMPLING TECHNIQUES

To address the issue of data imbalance, it's essential to generate additional synthetic samples for the underrepresented minority class. In our binary classification task using the SWAN-SF dataset, 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 dataset. When selecting an over-sampling technique for our time series data, two critical factors must be considered. Firstly, the technique must maintain the temporal dynamics inherent in the samples. Secondly, 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 compared to methods that merely perform random sampling from the original data without the generative aspect (Yoon et al. 2019).

We propose two distinct strategies to address the issue of imbalanced classification. The initial approach concentrates exclusively on utilizing over-sampling 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,

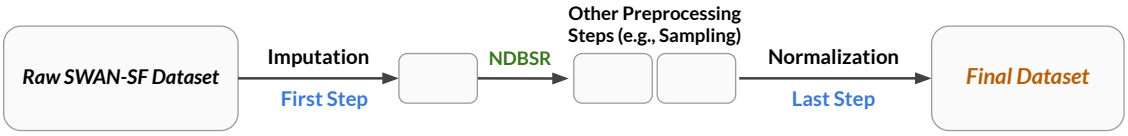

Figure 7. The diagram illustrates the proper placement of Near Decision Boundary Sample Removal (NDBSR) within a data preprocessing pipeline.

having over 100,000 samples for a binary classification task could lead to extended training time. To mitigate this, our second approach combines both over-sampling and under-sampling 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 dataset.

Algorithm 3 LSBZM Normalization

Require: dataset $data$, number of partitions $numParts$, number of attributes $numAttrs$, number of timestamps $numTimes$
Ensure: normalized $data$

for $partIdx = 1$ **to** $numParts$ **do**
  $partition \leftarrow data[partIdx]$ {The shape of partition is $(numTimes, numAttrs,$ numSamples$)$}
  for $attrIdx = 1$ **to** $numAttrs$ **do**
    Initialize $attributeVector$, $normalizedVector$ as empty lists
    $theAttribute \leftarrow partition[:, attrIdx, :]$
    $attributeVector \leftarrow Flatten(theAttribute)$
    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 $>$ 100000) **then**
      if skewness $>$ 1 **then**
        $attributeVector \leftarrow$ Log$(attributeVector)$
      else if skewness $<$ −1 **then**
        $attributeVector \leftarrow$ Sqrt$(attributeVector)$
      else
        $attributeVector \leftarrow$ Zscore$(attributeVector)$
      end if
      $normalizedVector \leftarrow$ 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**
        $attributeVector \leftarrow$ BoxCox$(attributeVector)$
      else
        $attributeVector \leftarrow$ Zscore$(attributeVector)$
      end if
      $normalizedVector \leftarrow$ 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
  $data[partIdx] \leftarrow partition$
end for
return $data$

6.1. *Over-sampling Techniques*

In our oversampling approach, we utilize techniques such as the Synthetic Minority Oversampling Technique (SMOTE) (Chawla et al. 2002), Adaptive Synthetic Sampling (ADASYN) (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 (Hosseinzadeh et al. 2024), these techniques are effective for time series generation. However, we also explore the Time-series Generative Adversarial Networks (TimeGAN) (Yoon et al. 2019), an advanced deep learning-based data augmentation technique for multivariate time series 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.

6.1.1. ***SMOTE***

Synthetic Minority Over Sampling Technique (SMOTE) (Chawla et al. 2002), is a statistical technique used to increase the number of instances in a dataset in a balanced way. It is primarily used in the context of imbalanced datasets, 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:

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

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-nearest neighbors, λ is a random number between 0 and 1, and x_i' is the new sample:

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

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 over-sampling. SMOTE does not change the number of majority class instances but augments the dataset by adding more examples from the minority class, thus balancing the class distribution.

6.1.2. ***ADASYN***

Adaptive Synthetic Sampling (ADASYN) (He et al. 2008) is an advanced over-sampling technique used in dealing with imbalanced datasets, similar to SMOTE. Its primary objective is to use a weighted distribution for different minority class examples according to their level of difficulty in learning, where more synthetic data is generated for minority class examples that are harder to learn.

ADASYN works as follows:

- 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.
- Compute the Euclidean Distance and K-nearest neighbors (k-NN): For each sample in the minority class, compute the Euclidean distance to find its K-nearest neighbors. The neighbors are chosen from the entire training set, not just the minority class.
- 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.

- 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-nearest neighbors of the ith minority class sample:

$$G_i = G \times \frac{\Delta_i}{\sum_i \Delta_i}$$

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 towards the regions where the class imbalance is more pronounced.

6.1.3. *Gaussian Noise Injection*

Gaussian Noise Injection (GNI) is an over-sampling technique used for addressing imbalanced datasets. 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:

- Determine the standard deviation σ: Calculate the standard deviation of the dataset (or a subset of features), denoted as σ. This measures the variation or dispersion from the average value.
- 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.
- 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., $\lambda = \sigma \times \alpha$.
- 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 dataset. Thus, $\eta \sim \mathcal{N}(0, \lambda^2)$.
- 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:

$$x' = x + \eta$$

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 over-sampling techniques may not be necessary.

6.1.4. *TimeGAN*

Time-series Generative Adversarial Networks (TimeGAN) (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 datasets.

The process of TimeGAN can be mathematically described as follows:

- Embedding Function (e): This function maps the original time-series data x to a latent space Z. The embedding function is given by $e : X \to Z$.
- Recovery Function (r): This is the inverse of the embedding function, mapping the latent representation back to the data space, represented by $r : Z \to X$.
- Sequence Generator (G): The generator creates synthetic data in the latent space from a noise vector ϵ, given by $G : \epsilon \to \tilde{Z}$.

- Sequence Discriminator (D): The discriminator differentiates between real and synthetic data, represented by a function $D : Z \cup \tilde{Z} \to \{0, 1\}$, where 0 indicates synthetic data and 1 indicates real data.
- 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:

- 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.
- 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.
- 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.
- For the Discriminator: $D(h_t, h_t^g) = \begin{cases} 0 & \text{if sample is synthetic} \\ 1 & \text{if sample is real} \end{cases}$.

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.

6.2. *Combination of Over and Under-sampling Techniques*

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

6.2.1. ***Random Under Sampling***

Random Under Sampling (RUS) is a technique used for handling imbalanced datasets, particularly focusing on the under-sampling of the majority class. Unlike over-sampling techniques that increase the size of the minority class, under-sampling reduces the size of the majority class to balance the dataset. 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 dataset is sufficiently large, and the reduction does not lead to a significant loss of information. However, one must be cautious as under-sampling can lead to the loss of potentially important information from the majority class.

6.2.2. ***Tomek-Links***

Tomek-Links (TL) (Tomek 1976) is a more nuanced under-sampling technique used in the context of imbalanced datasets, specifically focusing on the elimination of instances from the majority class. Unlike random under-sampling, 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:

- Identify Tomek Links: A Tomek Link 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 Tomek Link if x_i is the nearest neighbor of x_j and vice versa, and they belong to different classes.

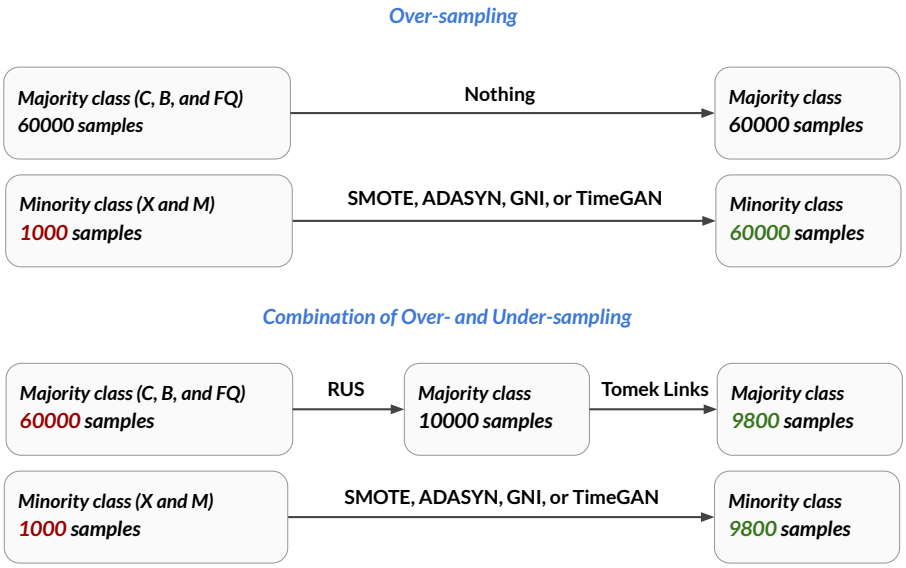

Figure 8. Illustration of two sampling approaches for SWAN-SF dataset: The top panel shows the original dataset with a majority class having 60,000 samples and a minority class with 1,000 samples. It demonstrates over-sampling techniques (SMOTE, ADASYN, GNI, TimeGAN) applied to the minority class to match the majority class's 60,000 samples. The bottom panel depicts a combination of over and under-sampling, where the majority class is reduced using RUS, and TL to 9,800 samples, and the minority class is augmented to 9,800 samples using the same over-sampling techniques.

- Remove Instances: The common approach in using Tomek Links for under-sampling involves removing the majority class instances that are part of Tomek Links. This is based on the idea that in a Tomek Link, one of the instances is likely noise or borderline, and removing it can help in making the decision boundary more distinct.
- Refine the dataset: After the removal of these instances, the dataset 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 dataset.

7. EXPERIMENTS AND RESULTS

7.1. *Preprocessed SWAN-SF Dataset and Code Repository of the Study*

We release of the preprocessed SWAS-SF dataset across all five partitions. The training set of this enhanced version incorporates our FPCKNN imputation technique, eliminates Classes B and C samples to address class overlap issues, and implements both over- and under-sampling strategies, including RUS, TL, and TimeGAN. Additionally, LSBZM normalization has been applied. The test set of this released version includes only the FPCKNN imputation technique and LSBZM normalization. This optimized dataset enables researchers to focus their efforts on developing more precise classifiers, rather than on preprocessing, thereby saving time for spaceweather analysis [1].

The entire codebase, which includes all preprocessing and sampling algorithms, as well as the classification algorithms detailed in this study, is thoroughly documented and has been made available for extensive review and application [2]. 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.

[1] The Preprocessed SWAN-SF is available here: https://anonymous.4open.science/r/Preprocessed-SWANSF-Anon

[2] The codebase of the study is available here: https://anonymous.4open.science/r/FlarePredict-DataPrep-Sampling-SWANSF-Anon

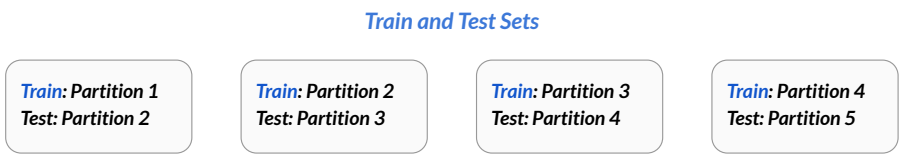

Figure 9. This figure showcases four distinct train-test sets employed in each classification experiment. This approach ensures a more comprehensive and accurate assessment of algorithms across all dataset partitions.

7.2. *Train and Test Sets*

The SWAN-SF dataset consists of five distinct partitions, each representing a specific timeline of active regions (AR). 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.

7.3. *Classification Algorithms*

To assess the impact of preprocessing and sampling techniques on the binary classification of the SWAN-SF dataset, we utilized various classification algorithms to demonstrate their effectiveness in enhancing classification performance. Testing these techniques with diverse classification algorithms is crucial to verify their benefits in improving classification outcomes. Initially, we evaluated the impact of each technique using four widely recognized classifiers: Support Vector Machine (SVM), Multilayer Perceptron (MLP), K-nearest Neighbors (k-NN), and Random Forest (RF). We then compared the performance of these classifiers to that achieved using standard baseline methods. Following this, we implemented a set of four preprocessing algorithms—FPCKNN imputation, LSBZM normalization, Near Decision Boundary Sample Removal, and sampling—on the dataset. The performance of the SWAN-SF dataset, once thoroughly preprocessed, was evaluated using advanced time series-based classifiers, including the Long Short-Term Memory (LSTM), Recurrent Neural Network (RNN), Gated Recurrent Unit (GRU), and 1D Convolutional Neural Network (1D-CNN). The results were then compared with those from the raw, unprocessed SWAN-SF dataset to highlight the critical role of preprocessing in achieving enhanced classification performance.

As the evaluation metric, we employ the True Skill Statistic (TSS) (Bobra & 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 (True Positive Rate) and its ability to limit the False Positive Rate, thus providing a comprehensive measure of model performance. The TSS is recalculated as follows:

$$\textbf{TSS} = \textbf{Recall} - \textbf{False Positive Rate} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} - \frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}} \tag{5}$$

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 datasets. By encompassing both the ability of the model to correctly identify positive cases (Recall) and its effectiveness in avoiding false positives (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 False Positive Rate (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 False Positive Rate as much as possible to optimize the TSS score.

7.3.1. *Feature Extraction-based Classification*

In the field of machine learning and data science, traditional classifiers such as Support Vector Machine (SVM) (Cortes & Vapnik 1995), Multilayer Perceptron (MLP) (Gardner & Dorling 1998), K-nearest Neighbors (k-NN) (Peterson 2009), and Random Forest (RF) (Breiman 2001) are not specifically designed for handling time series data. However, with proper preprocessing and feature extraction techniques, these conventional models can be as effective

as the latest deep learning algorithms in classifying time series data. The main issue is the complex nature and high dimensionality of time series datasets, especially MVTS, which makes the training process more complicated and time-consuming.

To overcome these challenges, an important step is the strategic extraction of important features from each attribute of the MVTS dataset, usually made up of univariate time series. By refining the dataset to its most significant components, we can speed up the training process and improve the model's performance, resulting in a more efficient and effective approach to time series classification. For illustration, consider the SWAN-SF dataset, encompassing 24 attributes, each marked by 60 timestamps. This equates to a cumulative total of 1440 attributes (60×24), a volume that exceeds the processing capacity of most conventional classifiers. A significant portion of these attributes may be redundant or contribute minimally to the differentiation between classes.

To address this, we employ a methodical strategy for feature extraction by focusing on nine statistical properties. These properties together define the key characteristics of each univariate time series in a multivariate time series (MVTS) dataset.

- **First Value:** X_1
- **Last Value:** X_n
- **Mean:** $\mu = \frac{1}{n}\sum_{i=1}^{n} X_i$
- **Median:** $\text{Median}(X)$
- **Weighted Average:** $\bar{X}_w = \frac{\sum_{i=1}^{n} w_i X_i}{\sum_{i=1}^{n} w_i}$
- **Standard Deviation:** $\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \mu)^2}$
- **Skewness:** $\frac{n}{(n-1)(n-2)}\sum_{i=1}^{n}\left(\frac{X_i-\mu}{\sigma}\right)^3$
- **Kurtosis:** $\frac{n(n+1)}{(n-1)(n-2)(n-3)}\sum_{i=1}^{n}\left(\frac{X_i-\mu}{\sigma}\right)^4 - \frac{3(n-1)^2}{(n-2)(n-3)}$
- **Slope:** $a = \frac{n(\sum xy)-(\sum x)(\sum y)}{n(\sum x^2)-(\sum x)^2}$

By concentrating on these statistical properties, we can greatly reduce the dimensionality of the dataset 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 dataset 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.

7.3.2. *Time Series-based Classification*

In the domain of time series data analysis, feature extraction and the use of traditional classifiers can speed up training and simplify the understanding of temporal dynamics. However, employing actual data sequences directly in advanced time series classifiers and deep learning models offers distinct advantages. Models such as Long Short-Term Memory (LSTM) network (Hochreiter & Schmidhuber 1997), Recurrent Neural Network (RNN) (Sherstinsky 2020), Gated Recurrent Unit (GRU) (Chung et al. 2014), and 1D Convolutional Neural Network (1D-CNN) (Lecun et al. 1998) are particularly adept at enhancing classification accuracy in MVTS data. 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 dataset.

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 dataset 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.

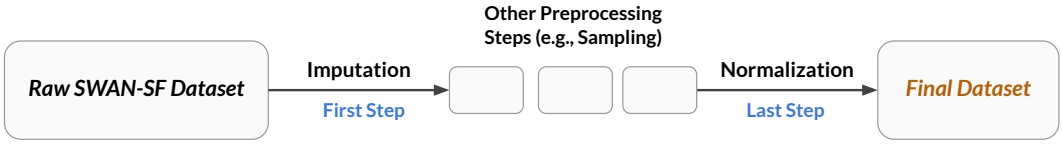

Figure 10. The figure illustrates the sequence of imputation and normalization tasks in a data preprocessing pipeline.

7.4. *Impact of FPCKNN Imputation and LSBZM Normalization*

In the preprocessing pipeline for the SWAN-SF dataset, 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's essential that other preprocessing operations, such as sampling, are executed post-imputation but pre-normalization. The justification behind this sequencing is that sampling methods, along with other preprocessing techniques, can potentially alter the dataset by either omitting or introducing additional samples. These changes consequently affect the dataset'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 dataset 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 datasets 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. By merging these methods, we generate four baseline models. Our analysis then focuses on comparing the results of merging FPCKNN imputation and LSBZM normalization against these four baseline methodologies.

In Section 4.2, 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 dataset 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 (non-missing) value in the data sequence. This method relies on the temporal or sequential structure of the data, making it well-suited for time-series datasets where the missing value can be reasonably estimated by the subsequent observed value (Khan & Hoque 2020).

7.4.1. *Insights on Normalization*

Based on the LSBZM normalization technique, we have extracted crucial information concerning the dataset's 24 attributes (photospheric magnetic field parameters). This encompasses identifying the most effective normalization method for each attribute, as detailed in Table 3. As a result, researchers can refer to Table 3 to determine the optimal normalization approach for different attributes in the SWAN-SF dataset, without the necessity of directly implementing the LSBZM technique. Furthermore, the table specifies which attributes demonstrate right or left skewness.

7.4.2. *Classification Results*

Based on Figures 11 and 12, as well as Table 4, it is evident that our combined imputation and normalization technique significantly enhances the Mean and Max TSS scores compared to the baseline methods. For example, using the SVM classifier, the highest TSS score achieved was 0.273 with the baseline approach. In contrast, our technique yielded a much improved Max TSS score of 0.729. Similarly, for the MLP classifier, an impressive Max TSS score of 0.825 was attained solely by applying our imputation and normalization method, without the need

Table 3. This table outlines optimal normalization techniques for SWAN-SF dataset's attributes, highlighting the best method for each, along with an overview of their characteristics. These include right and left skewness.

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

Table 4. Mean TSS values for different combination of imputation (I) and normalization (N) techniques.

Technique	SVM	Multilayer Perceptron	k-NN	Random Forest
FPCKNN-I and LSBZM-N	**0.254** $\pm$ 0.274	**0.320** $\pm$ 0.299	**0.227** $\pm$ 0.171	**0.226** $\pm$ 0.173
Nextvalue-I and MinMax-N	0.133 $\pm$ 0.096	0.266 $\pm$ 0.152	0.203 $\pm$ 0.072	0.157 $\pm$ 0.138
Nextvalue-I and Zscore-N	0.064 $\pm$ 0.030	0.244 $\pm$ 0.115	0.172 $\pm$ 0.055	0.165 $\pm$ 0.098
Mean-I and MinMax-N	0.128 $\pm$ 0.087	0.274 $\pm$ 0.141	0.200 $\pm$ 0.071	0.141 $\pm$ 0.118
Mean-I and Zscore-N	0.067 $\pm$ 0.030	0.278 $\pm$ 0.146	0.170 $\pm$ 0.063	0.147 $\pm$ 0.076

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 dataset. 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 dataset. 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.

7.5. *Impact of Near Decision Boundary Sample Removal*

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 Near Decision Boundary Sample Removal (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 extraction. Normalization is invariably the final step, executed after feature extraction. Our baseline method for this

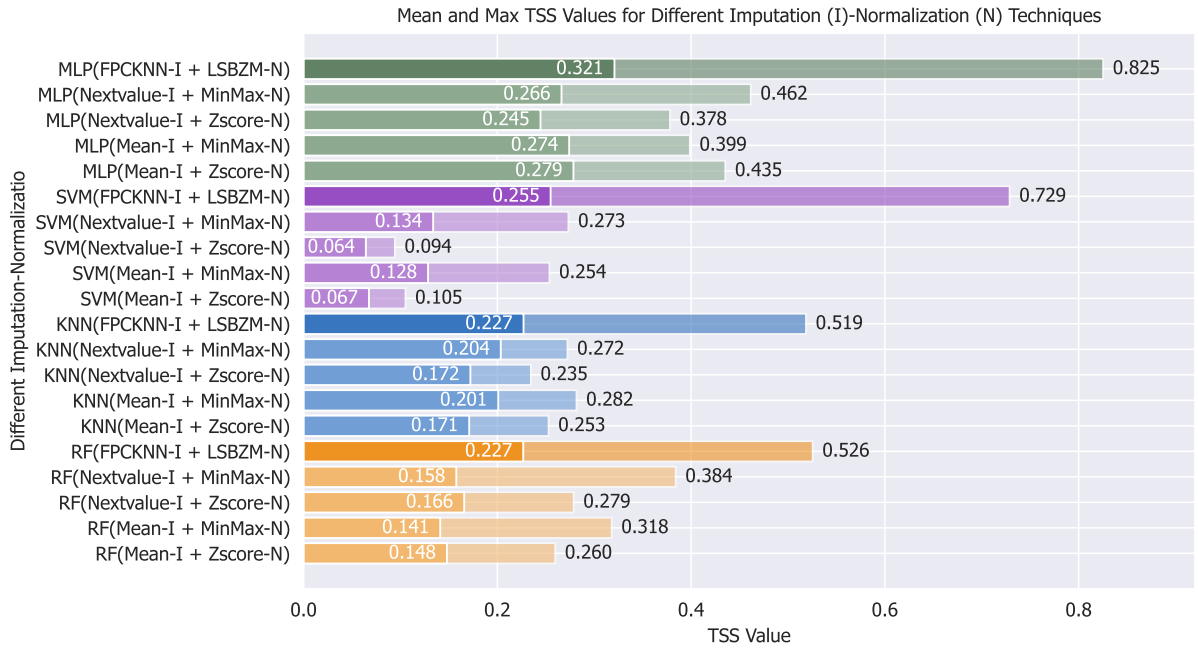

Figure 11. Comparative analysis of Mean and Max TSS across various imputation (I) and normalization (N) technique combinations. Efficacy evaluated via four classifiers: SVM, MLP, k-NN , and RF. For each bar, the first value indicates the Mean TSS, and the second value indicates the Max TSS of four train-test combinations.

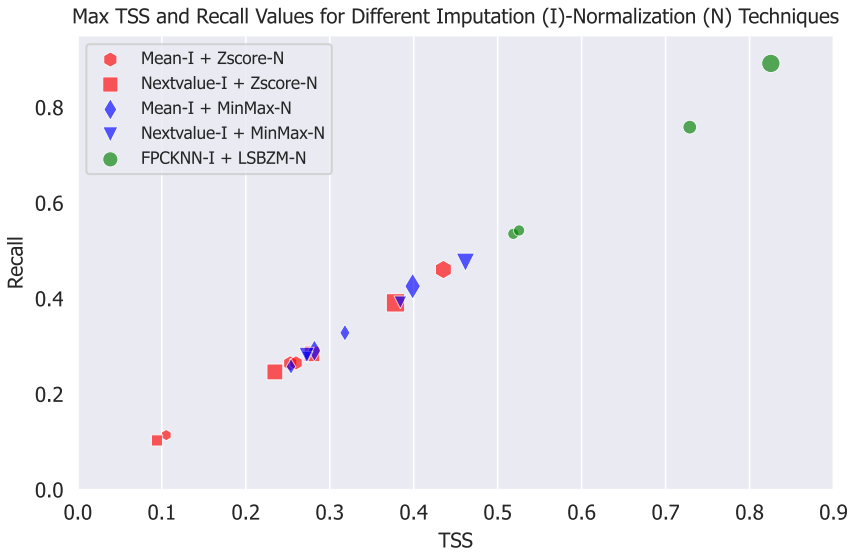

Figure 12. Comparative analysis of Max TSS and Recall across different imputation (I) and normalization (N) techniques and classifiers. Efficacy assessed with SVM, MLP, k-NN, and RF. Each shape in the plot represents the result of classification using four train-test combinations. The size of each shape indicates the Mean TSS, while the placement of the shape is determined by the Max TSS and Recall.

section is the dataset 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 false positives 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

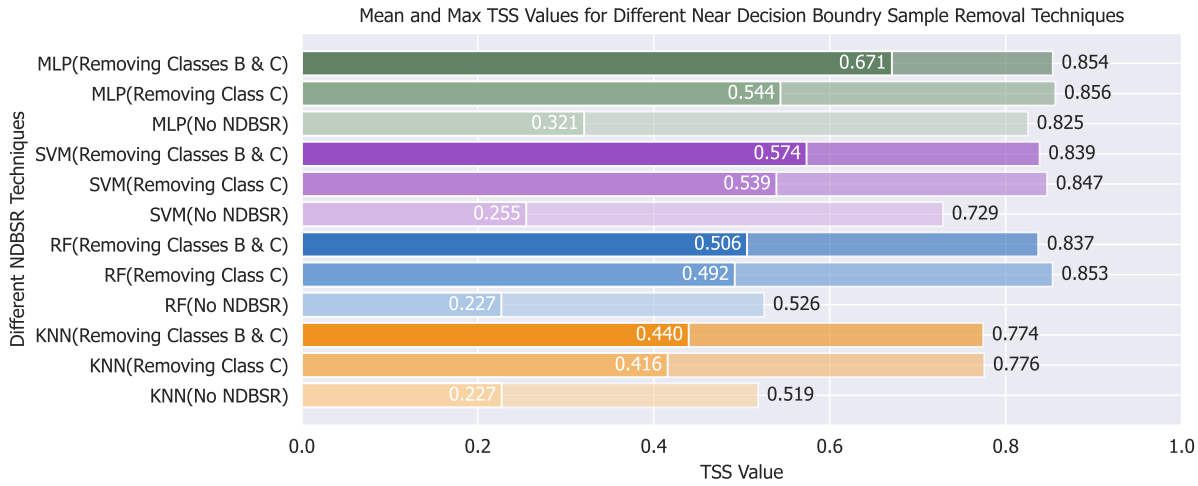

Figure 13. The figure presents a comparison of the Mean and Max TSS scores achieved by various Near Decision Boundary Sampling Removal (NDBSR) methods when applied to different classifiers. These classifiers include MLP, RF, SVM, and k-NN. For each bar, the first value indicates the Mean TSS, and the second value indicates the Max TSS of four train-test combinations.

Table 5. Mean TSS values for different NDBSR techniques

NDBSR Technique	**k-NN**	**SVM**	**Random Forest**	**Multilayer Perceptron**
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

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.

7.5.1. *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 dataset. Among the classifiers tested, MLP and SVM were the most effective. These results underscore the importance of these techniques for datasets such as SWAN-SF, which show significant class overlap in both binary and multi-class classification scenarios.

7.6. *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 dataset, 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.

7.6.1. *t-SNE Visualizations*

Following the generation of synthetic samples for the minority class, we conduct a balanced selection, choosing an

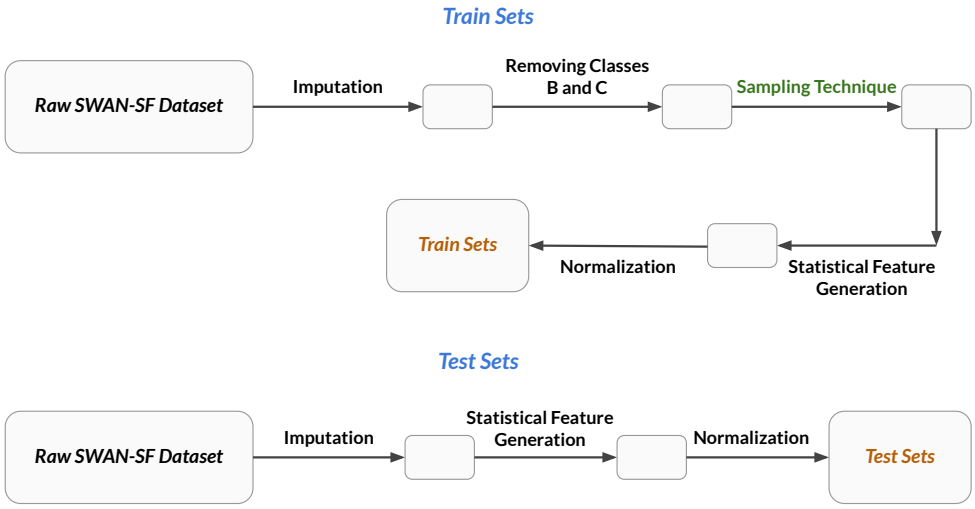

Figure 14. The figure effectively showcases the preprocessing pipelines, detailing the specific steps and methods employed for both training and test sets. To ensure an unbiased comparison of the sampling results, it is crucial that the test sets remain unaltered, with no samples being either removed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

equal number of synthetic samples as there are in the original minority class (approximately 1000). Subsequently, we utilize t-SNE (t-distributed Stochastic Neighbor Embedding) (van der Maaten & 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 2-dimensional space, thereby offering a visual comparison of their respective distributions.

As illustrated in Figure 24, 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.

7.6.2. *Discriminative Score*

For a quantitative measure of similarity, we train an SVM classifier to distinguish between synthetic and original samples. 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 percent of the data. Afterward, we compute the error using the formula:

$$\mathbf{e = a - 0.5}$$

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 dataset. 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.

7.6.3. *Classification Performance*

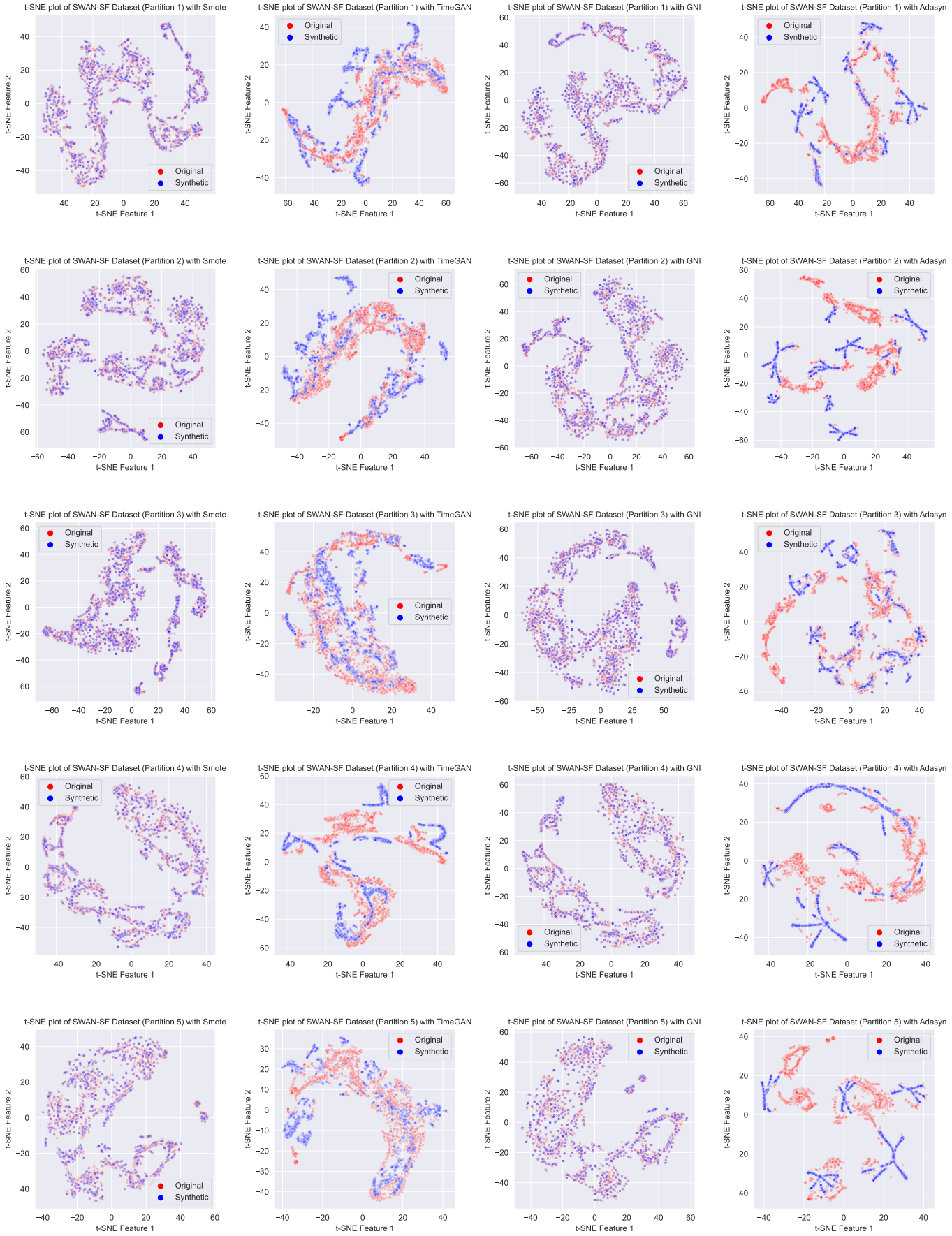

Figure 15. t-SNE visualizations demonstrating the distributional alignment of original and synthetic data samples for aach over-sampling technique across dataset partitions. These visualizations highlight the efficacy of each over-sampling method in replicating the comprehensive distribution characteristics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

Table 6. Comparison of discriminative scores for the over-sampling techniques across the dataset 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

Table 7. Mean TSS values for different sampling techniques

Sampling Technique	k-NN	SVM	Random Forest	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

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, the Mean TSS score without sampling ranges from 0.227 to 0.321, but after applying a combination of over- and under-sampling, this value rises to between 0.649 and 0.808, marking a remarkable improvement. Furthermore, the combination of over- and under-sampling techniques yields better results compared to solely over-sampling for both Max and Mean TSS scores. Therefore, in addition to over-sampling it is suggested to perform under-sampling as well to decrease the number of majority class samples which prevents overfitting. Overall, SMOTE and TimeGAN consistently show the highest Mean TSS for both the combination of over- and under-sampling and the solely over-sampling approaches. Given that TimeGAN is a generative model producing unique and original synthetic data, and considering the t-SNE visualization results, it is concluded that TimeGAN is the optimal over-sampling technique for the SWAN-SF dataset.

In situations where researchers cannot use TimeGAN for over-sampling, SMOTE and GNI stand out as the second and third recommended techniques for the SWAN-SF dataset 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.

7.7. *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 over-sampling and a combination of over-sampling and under-sampling 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 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 dataset. Furthermore, the integration of both over- and under-sampling strategies enhances the TSS score more effectively than relying on over-sampling 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 dataset. The combined application of RUS, TL, and ADASYN, as well as RUS, TL, and SMOTE, has yielded the most favorable Mean TSS outcomes for the SWAN-SF dataset. Additionally, the combination of RUS, TL, and TimeGAN consistently achieved a high Max TSS. Nonetheless, the

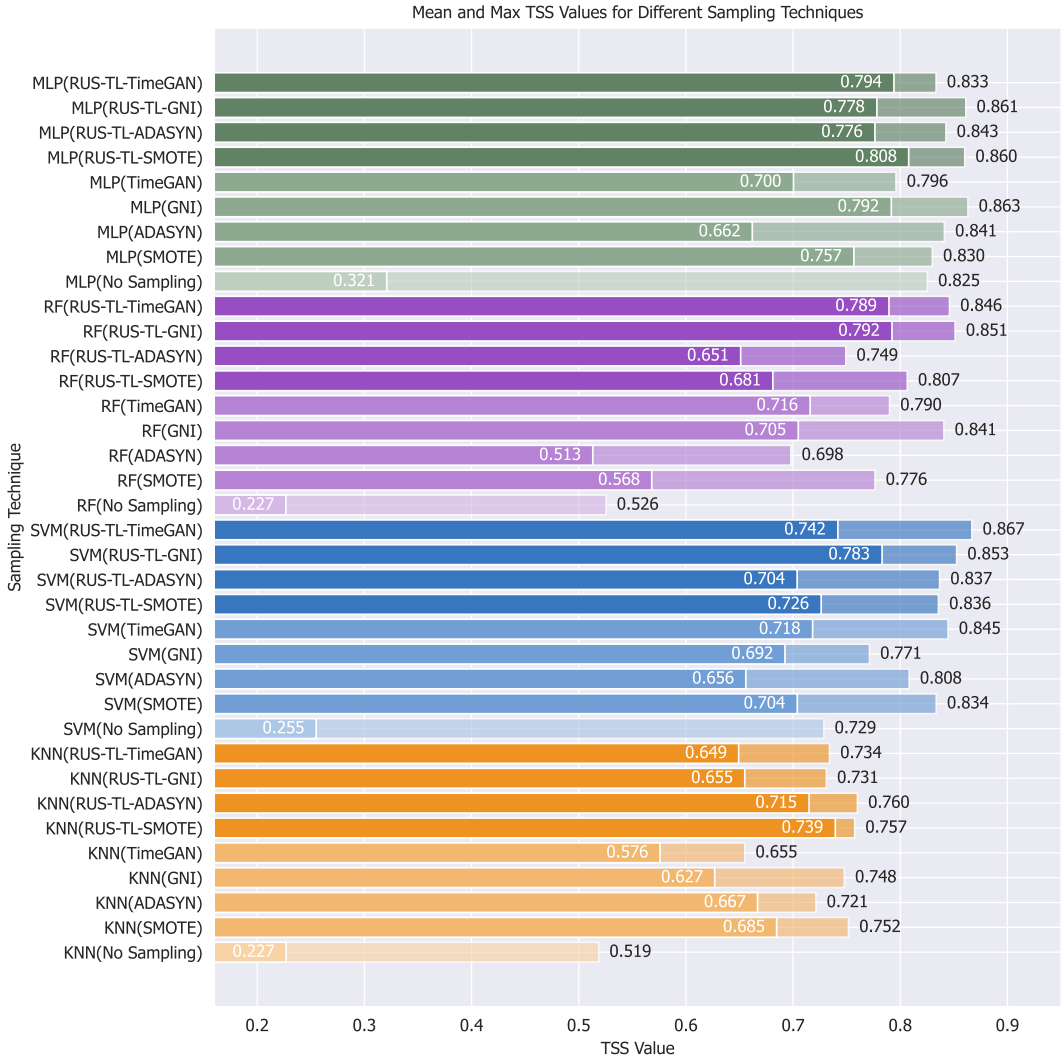

Figure 16. Comparative analysis of Mean and Max TSS across various sampling techniques and classifiers: This figure illustrates the performance of different sampling methods, including exclusive over-sampling techniques (SMOTE, ADASYN, GNI, TimeGAN) and combination of over and under-sampling approaches (RUS and TL with one of the over-sampling 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.

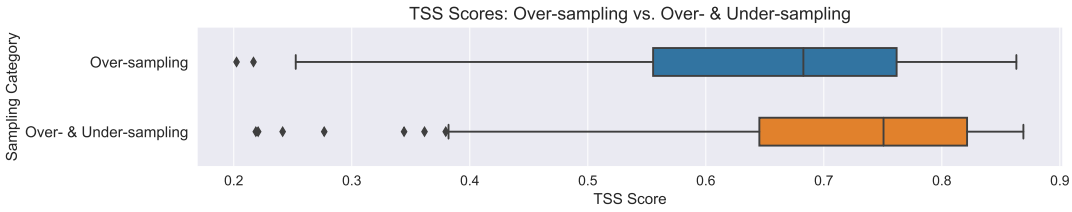

Figure 17. Comparative Analysis of TSS Scores: Assessing the Over-sampling techniques versus the combined over- and under-sampling methods across our eight classifiers and four unique training and testing combinations.

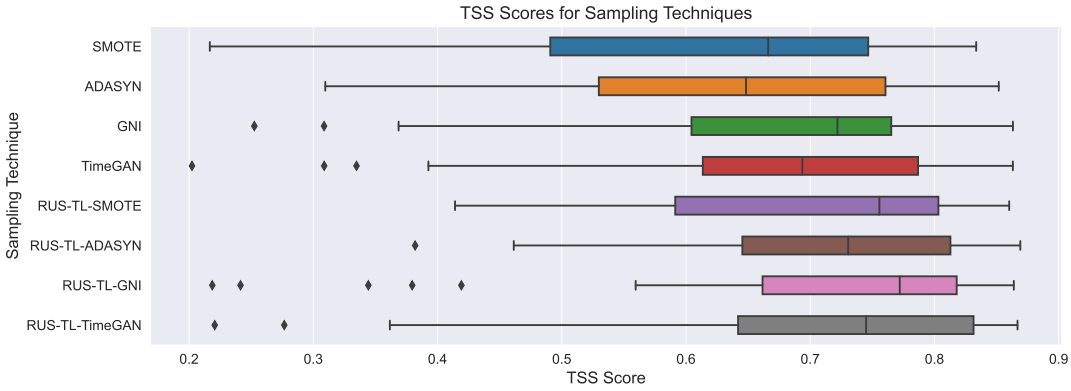

Figure 18. Assessing TSS scores of the eight sampling techniques across our eight classifiers and four unique training and test combinations.

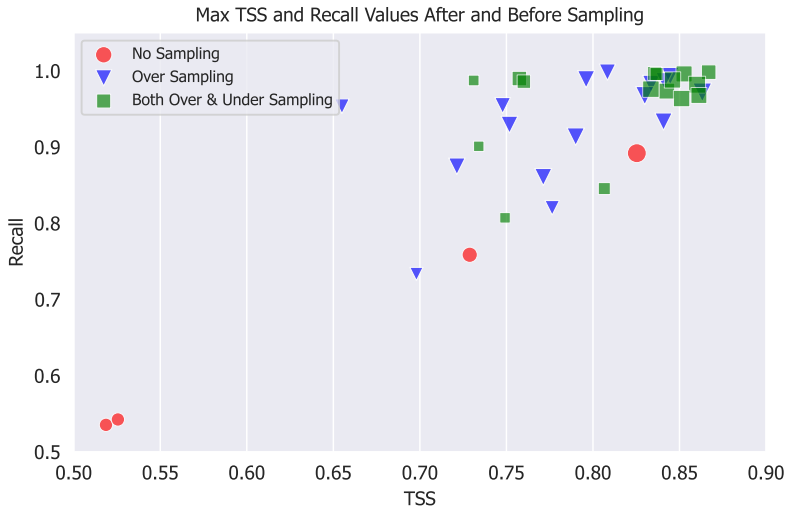

Figure 19. In-depth comparison of Max TSS and Recall across different sampling techniques and classifiers: This figure illustrates the performance of various sampling methods, encompassing exclusive over-sampling techniques as well as combinations of over- and under-sampling techniques. The efficacy 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. The size of each shape indicates the Mean TSS, while the placement of the shape is determined by the Max TSS and Recall.

absence of any preprocessing steps on the SWAN-SF dataset predominantly results in a TSS score of zero, underscoring the critical role of a comprehensive preprocessing pipeline. Specifically, our approach elevated the TSS from 0.0 to a peak of 0.869 for the GRU model. Furthermore, for the LSTM model, the Mean TSS was enhanced from 0.302 to 0.736, demonstrating the effectiveness of our preprocessing methodology in enhancing the performance of models.

7.8. *Comparison of Classifiers and Partitions*

Based on Figures 21 and 22, the Multilayer Perceptron (MLP) and Support Vector Machine (SVM) emerge as the most effective feature extraction-based classifiers for the binary classification of major and minor flaring events within the SWAN-SF dataset. Additionally, the Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM) classifiers are identified as the optimal time series-based classifiers for this dataset. The selection of Partition 3 as the training set and Partition 4 as the test set yields the highest TSS scores. Therefore, if researchers decide to focus on a limited number of training and testing combinations, the initial choice involves selecting partition 3 for training and partition 4 for testing. The subsequent option is to select partition 4 for training and partition 5 for testing, which resulted in the second-highest performance.

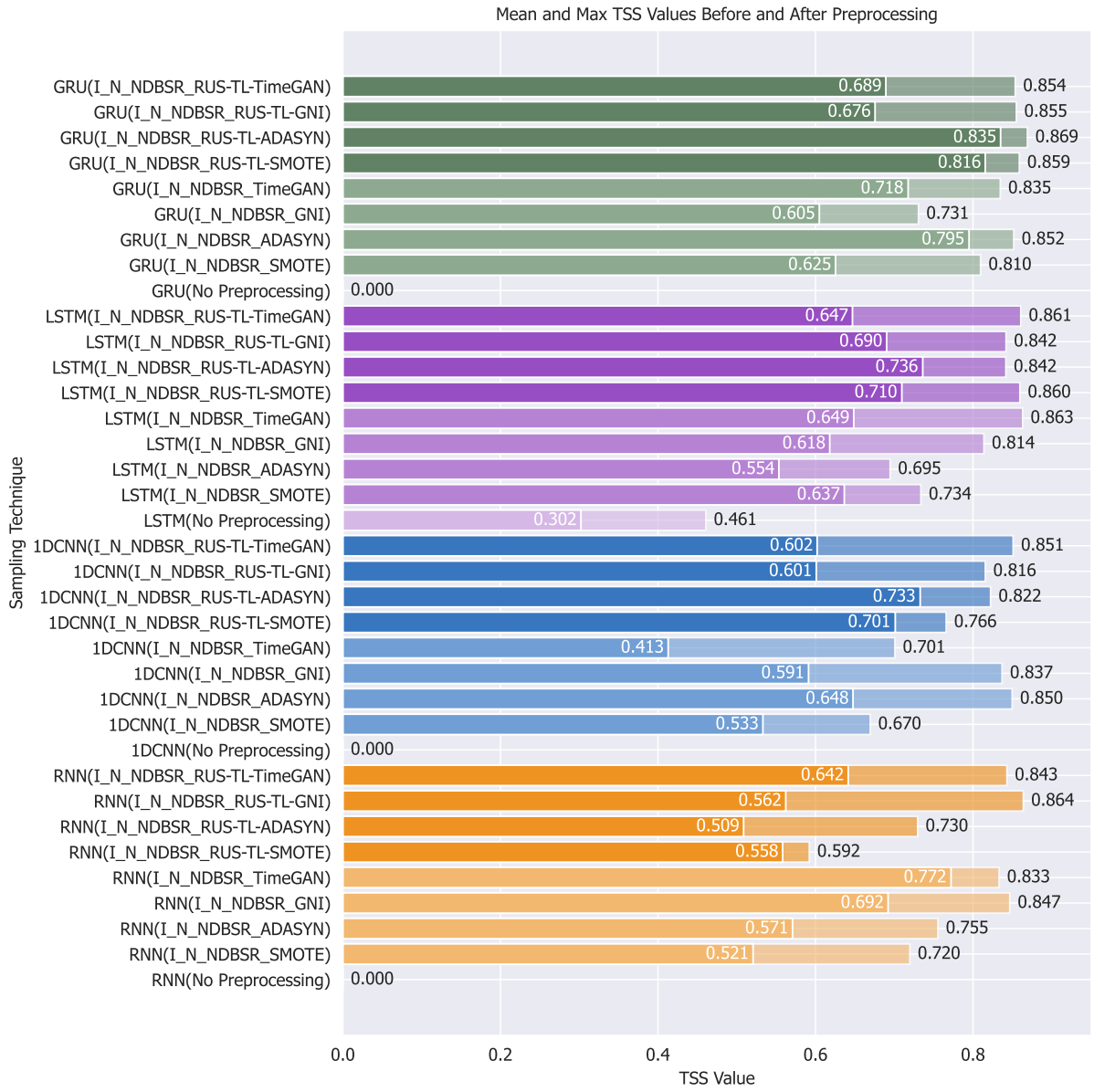

Figure 20. The figure presents the outcomes of employing no preprocessing on the dataset versus the application of our imputation (I), our normalization (N), Near Decision Boundary Sample Removal (NDBSR), and various sampling techniques in combination. The efficacy of these techniques is evaluated using 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.

7.9. *Comparison with Flare Prediction Baselines*

In the final section, we aim to compare our findings and results with similar flare prediction baselines conducted by (Ahmadzadeh et al. 2021). Based on Figure 23, our results significantly outperform the baseline results, achieving a True Skill Statistic (TSS) of more than 0.8 almost consistently and reaching a maximum 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 was achieved by utilizing a Gated Recurrent Unit (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 over- and under-sampling techniques, including Random Under Sampling (RUS), Tomek Links (TL), and Adaptive

Table 8. Mean TSS values before and after preprocessing, which includes our imputation (I), our normalization (N), Near Decision Boundary Sample Removal (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

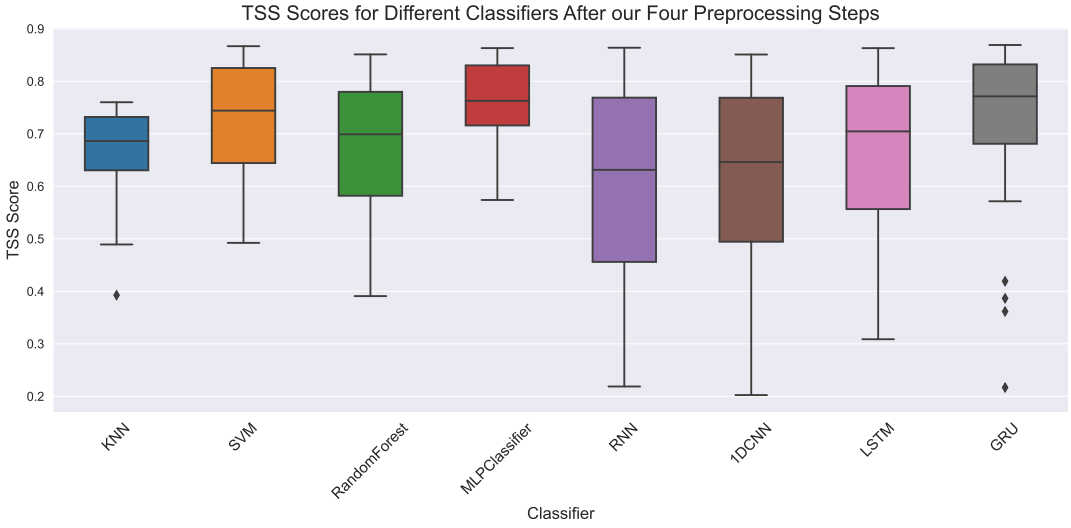

Figure 21. This plot depicts a comparative analysis of TSS scores obtained by different classifiers on the SWAN-SF dataset.

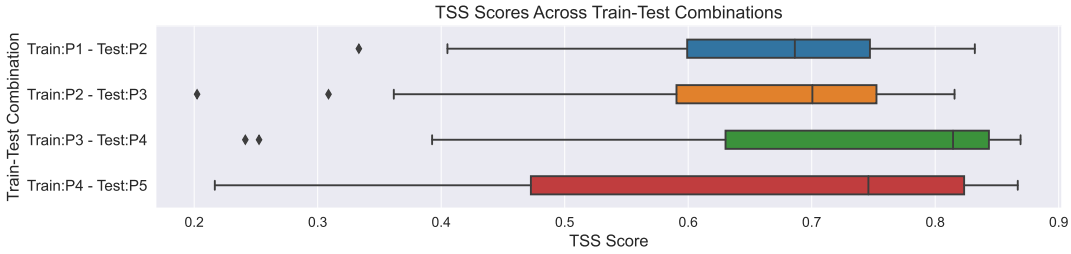

Figure 22. This plot presents a comparative analysis of the performance across various training and testing combinations applied to the SWAN-SF dataset. "P" signifies Partition.

Synthetic Sampling (ADASYN). Unlike feature extraction-based classifiers such as SVM, ADASYN performs well with deep learning-based classifiers.

8. CONCLUSION AND FUTURE WORK

Through extensive experiments incorporating our FPCKNN imputation, LSBZM normalization, Near Decision Boundary Sample Removal, and eight distinct sampling techniques across eight classifiers, we have demonstrated

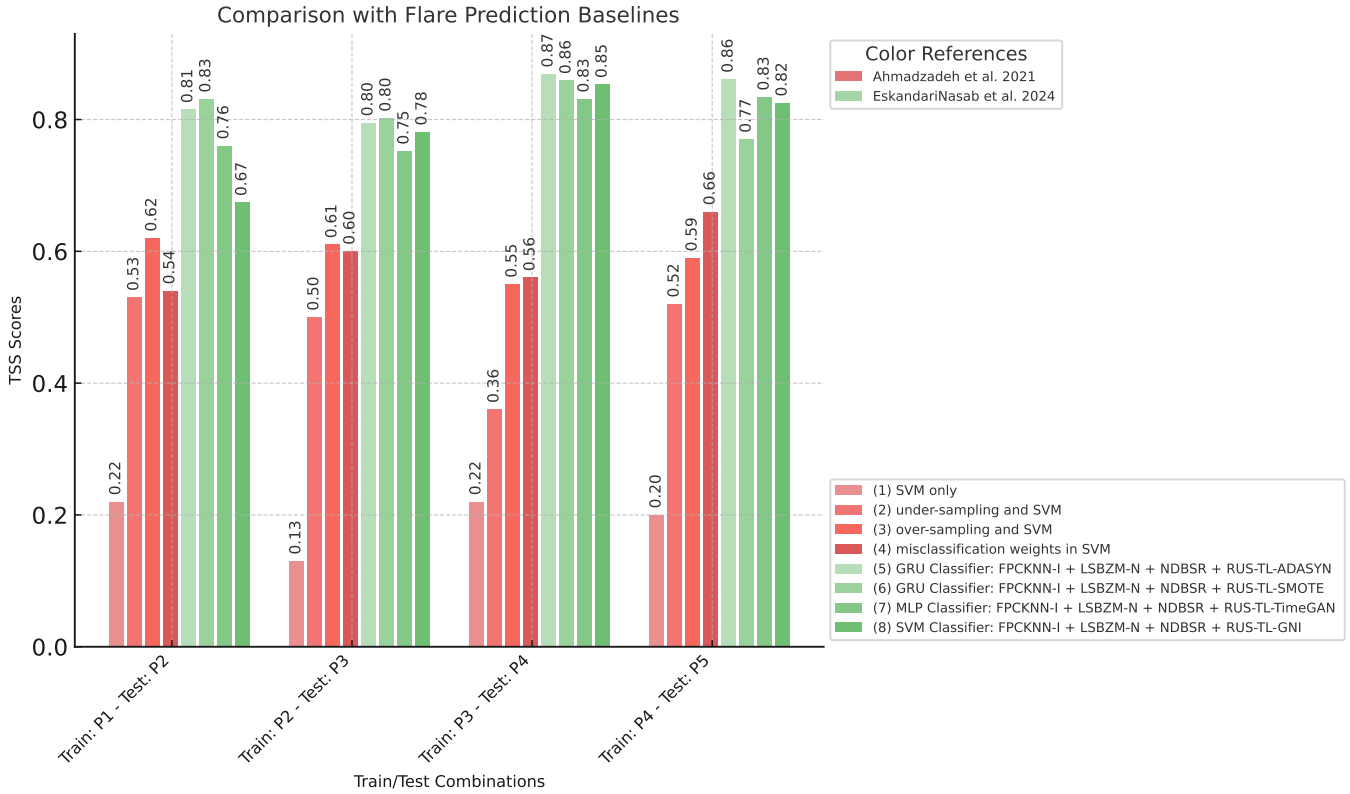

Figure 23. The figure displays a comparison of our best results, depicted in shades of green, against the results from (Ahmadzadeh et al. 2021), which are shown in shades of red. 'I' stands for imputation, and 'N' for normalization.

a remarkable improvement in True Skill Statistics (TSS) score following each preprocessing step. A TSS score exceeding 0.86 was achieved multiple times without the adoption of complex classification methodologies, highlighting the critical role of a precise preprocessing pipeline. This is particularly relevant in the field of solar flare prediction and when dealing with challenging datasets 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 a Near Decision Boundary Sample Removal (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 over- and under-sampling techniques, specifically Random Under Sampling (RUS), Tomek-links (TL), and TimeGAN (or SMOTE), yielded the most favorable outcomes, significantly enhancing TSS scores compared to the sole use of over-sampling or the complete absence of sampling. When evaluating solely over-sampling 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 dataset, the Multilayer Perceptron (MLP), Support Vector Machine (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 over-sampling technique tailored for multivariate time series data. Furthermore, we aspire to devise a sophisticated deep learning-based method to diminish the class overlap within the SWAN-SF dataset. Lastly, we plan to develop a deep learning-based imputation technique specifically for multivariate time series data, utilizing Autoencoders.

REFERENCES

Ahmadzadeh, A., Aydin, B., Georgoulis, M. K., et al. 2021, The Astrophysical Journal Supplement Series, 254, 23, doi: 10.3847/1538-4365/abec88

Alshammari, K., Hamdi, S. M., Muzaheed, A. A., & Filali Boubrahimi, S. 2022, CIKM workshop for Applied Machine Learning Methods for Time Series Forecasting (AMLTS) 2022. https://par.nsf.gov/biblio/10404768

Angryk, R. A., Martens, P. C., Aydin, B., et al. 2020, Scientific Data, 7, 227, doi: 10.1038/s41597-020-0548-x

Anil Jadhav, D. P., & Ramanathan, K. 2019, Applied Artificial Intelligence, 33, 913, doi: 10.1080/08839514.2019.1637138

Aschwanden, M. J., Crosby, N. B., Dimitropoulou, M., et al. 2016, Space Science Reviews, doi: 10.1007/s11214-014-0054-6

Bobra, M. G., & Couvidat, S. 2015, Astrophysical Journal, 798, 135, doi: 10.1088/0004-637X/798/2/135

Breiman, L. 2001, Machine Learning, 45, 5, doi: 10.1023/A:1010933404324

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. 2002, Journal of Artificial Intelligence Research, 16, 321–357, doi: 10.1613/jair.953

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. 2014, Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. https://arxiv.org/abs/1412.3555

Cortes, C., & Vapnik, V. 1995, Machine learning, 20, 273, doi: 10.1007/BF00994018

Emmanuel, T., Maupong, T., Mpoeleng, D., et al. 2021, Journal of Big Data, 8, 140, doi: 10.1186/s40537-021-00516-9

Feng, C., Wang, H., Lu, N., et al. 2014, Shanghai Arch Psychiatry, 26, 105, doi: 10.3969/j.issn.1002-0829.2014.02.009

Fisher, G. H., Bercik, D. J., Welsch, B. T., & Hudson, H. S. 2012, 59, doi: 10.1007/978-1-4614-3761-1_6

Gardner, M., & Dorling, S. 1998, Atmospheric Environment, 32, 2627, doi: https://doi.org/10.1016/S1352-2310(97)00447-0

Georgoulis, M. K. 2012, Solar Physics, 276, 161, doi: 10.1007/s11207-010-9705-2

Hamdi, S. M., Ahmad, A. F., & Boubrahimi, S. F. 2022, in Proceedings of the Workshop on Applied Machine Learning Methods for Time Series Forecasting (AMLTS 2022) co-located with 31st ACM International Conference on Information and Knowledge Management (CIKM 2022), Vol. 3375 (Atlanta, Georgia, USA: CEUR Workshop Proceedings, CEUR-WS.org). https://researchr.org/publication/HamdiAB22

Hamdi, S. M., Kempton, D., Ma, R., Boubrahimi, S. F., & Angryk, R. A. 2017, in Proceedings of the IEEE International Conference on Big Data (BigData), 2543–2551, doi: 10.1109/BigData.2017.8258213

He, H., Bai, Y., Garcia, E. A., & Li, S. 2008, in 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), 1322–1328, doi: 10.1109/IJCNN.2008.4633969

Hochreiter, S., & Schmidhuber, J. 1997, Neural Comput., 9, 1735–1780, doi: 10.1162/neco.1997.9.8.1735

Hosseinzadeh, P., Boubrahimi, S. F., & Hamdi, S. M. 2024, The Astrophysical Journal Supplement Series, 270, 31, doi: 10.3847/1538-4365/ad1de0

Khan, S. I., & Hoque, A. S. M. L. 2020, Journal of Big Data, 7, 37, doi: 10.1186/s40537-020-00313-w

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. 1998, Proceedings of the IEEE, 86, 2278, doi: 10.1109/5.726791

Leka, K., & Skumanich, A. 1999, Solar Physics, 188, 3, doi: 10.1023/A:1005108632671

Leka, K. D., & Barnes, G. 2003, ApJ, 595, 1296, doi: 10.1086/377512

Muhammad Ali, P., & Faraj, R. 2014, doi: 10.13140/RG.2.2.28948.04489

Muzaheed, A. A. M., Hamdi, S. M., & Boubrahimi, S. F. 2021, in Proceedings of the 20th IEEE International Conference on Machine Learning and Applications (ICMLA), 435–440, doi: 10.1109/ICMLA52953.2021.00074

Peterson, L. E. 2009, Scholarpedia, 4, 1883, doi: 10.4249/scholarpedia.1883

Sakia, R. M. 1992, Journal of the Royal Statistical Society. Series D (The Statistician), 41, 169, doi: 10.2307/2348250

Schrijver, C. J. 2007, The Astrophysical Journal, 655, L117, doi: 10.1086/511857

Sherstinsky, A. 2020, Physica D: Nonlinear Phenomena, 404, 132306, doi: 10.1016/j.physd.2019.132306

Singh, D., & Singh, B. 2020, Applied Soft Computing, 97, 105524, doi: https://doi.org/10.1016/j.asoc.2019.105524

Tomek, I. 1976, IEEE Transactions on Systems, Man, and Cybernetics, SMC-6, 769, doi: 10.1109/TSMC.1976.4309452

Troyanskaya, O., Cantor, M., Sherlock, G., et al. 2001, Bioinformatics, 17, 520, doi: 10.1093/bioinformatics/17.6.520

van der Maaten, L., & Hinton, G. 2008, Journal of Machine Learning Research, 9, 2579. http://jmlr.org/papers/v9/vandermaaten08a.html

Wang, J., Shi, Z., Wang, H., & Lue, Y. 1996, ApJ, 456, 861, doi: 10.1086/176703

Yoon, J., Jarrett, D., & van der Schaar, M. 2019, in Advances in Neural Information Processing Systems 32 (NeurIPS 2019), doi: https://openreview.net/forum?id=rJeZq4reLS

Zhang, J., Wang, T., Ng, W. W. Y., Zhang, S., & Nugent, C. D. 2019, in 2019 International Conference on Machine Learning and Cybernetics (ICMLC), 1–8, doi: 10.1109/ICMLC48188.2019.8949290

APPENDIX

A. DISCUSSION

The results of the experiments demonstrate significant and consistent improvements in classification performance when our preprocessing techniques are applied to the SWAN-SF dataset. The key findings from our research include:

- The implementation of FPCKNN imputation combined with LSBZM normalization techniques markedly outperforms all standard baseline methods. This is demonstrated by a substantial increase in the Maximum TSS, rising from 0.462 to 0.825, and an enhancement in the Mean TSS, improving from 0.279 to 0.321 when applied to the MLP classifier. Similarly, for the SVM classifier, there is a notable improvement, with the Maximum TSS escalating from 0.273 to 0.729 and the Mean TSS advancing from 0.134 to 0.255. These significant improvements in TSS scores underscore the vital importance of accurate imputation and effective normalization in achieving superior classification performance. If researchers opt not to implement our imputation and normalization technique, the integration of Nextvalue imputation with MinMax scaling emerges as the second most accurate approach for handling the SWAN-SF dataset.

- Eliminating classes B and C from the dataset and considering only class FQ as minor-flaring significantly enhances the TSS score. This approach proves more effective than either removing only class C or not addressing class overlap at all. Removing classes B and C results in a 100% increase in the Mean TSS score for the MLP classifier, moving from 0.321 to 0.671. This significant improvement underscores the importance of addressing the class overlap issue in the SWAN-SF dataset before proceeding with classification. The outcomes of removing both classes B and C, or just removing class C, are nearly identical, suggesting that the implementation of either approach depends on the classification's purpose and the choice of classifier.

- Both solely using over-sampling and the combination of over and under-sampling techniques significantly outperform the absence of any sampling. This improvement is evident in the increased Mean TSS scores, which rise from 0.321 to 0.808 for the MLP classifier and from 0.255 to 0.783 for the SVM classifier. The integration of both over- and under-sampling techniques surpasses the effectiveness of using over-sampling alone, chiefly due to a reduction in overfitting. This combined approach increases the Mean and Max TSS from 0.700 for the TimeGAN to 0.794 for the combination of RUS, TL, and TimeGAN using the MLP classifier, and from 0.716 to 0.789 for the RF classifier. Such improvements are observed when RUS, TL, and TimeGAN are implemented together, compared to the exclusive use of TimeGAN.

- TimeGAN is recognized as a highly effective over-sampling technique for the SWAN-SF dataset, offering distinct advantages in certain scenarios. While it might not universally outperform other methods such as SMOTE, ADASYN, and GNI in every case, its overall superiority is attributed to its ability to generate original and novel data. This unique capability of TimeGAN to produce new data, rather than merely interpolating or modifying existing samples, sets it apart and contributes to its effectiveness in enhancing dataset diversity and model performance. Furthermore, a hybrid approach that combines RUS and TL for under-sampling with TimeGAN for over-sampling, outperforms the exclusive use of TimeGAN. In cases where the originality of synthetic data is not a primary concern, SMOTE and GNI techniques serve as effective methods for generating additional synthetic data for the minority class of the SWAN-SF dataset, instead of using TimeGAN.

- Among traditional classifiers that use statistical feature extraction, the MLP, and SVM are the top performers in terms of TSS scores and stability for the SWAN-SF dataset, ranking first and second respectively.

- In deep learning-based architectures, GRU and LSTM rank as the top-performing models in terms of TSS score. The most effective sampling technique for these models involves a combination of RUS, TL, and ADASYN, leading to the highest Mean and Max TSS scores of 0.835 and 0.869, respectively, on the GRU model. Implementing our four-step preprocessing protocol has elevated the TSS from 0 to 0.869. Additionally, in deep learning models, the integration of over- and under-sampling techniques surpasses the performance achieved through the exclusive use of over-sampling.

Table 9. List of Notations

Abbreviation	Complete Name	Abbreviation	Complete Name
SWAN-SF	Space Weather Analytics for Solar Flares	AR	Active Regions
NOAA	National Oceanic and Atmospheric Administration	HSS	Heidke Skill Score
GOES	Geostationary Operational Environmental Satellites	TSS	True Skill Statistics
HMI	Helioseismic Magnetic Imager	I	Imputation
SHARP	Spaceweather HMI Active Region Patch	N	Normalization
SDO	Solar Dynamics Observatory	FQ	Flare-Quie
FPCKNN	Fast Pearson Correlation-based K-nearest neighbors Imputation	SVM	Support Vector Machine
LSBZM	Log, Square Root, BoxCox, Zscore, Min-Max Normalization	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 Over Sampling Technique	TL	Tomek-Links
GNI	Gaussian Noise Injection	Ml	Machine Learning
RUS	Random Under Sampling	NaN	Not a Number
t-SNE	t-distributed Stochastic Neighbor Embedding	MVTS	Multivariate Time Series
T	Number of Timestamps	SQRT	Square Root
PCC	Pearson Correlation Coefficient		
N	Number of Magnetic Field Parameters (attributes)		

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.

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 Principal Component Analysis (PCA) to visualize both the original and synthetic samples. 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.

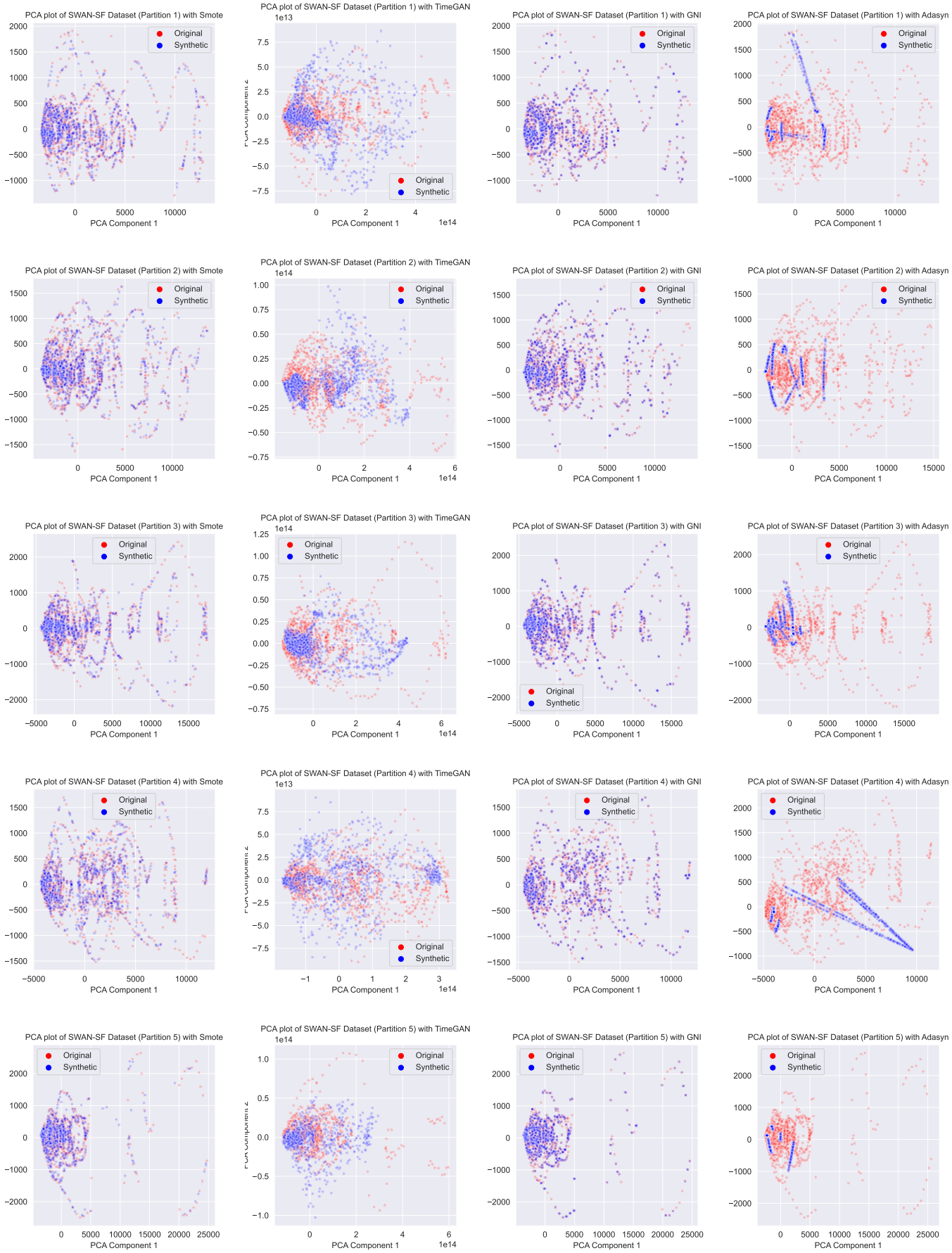

Figure 24. PCA Visualizations Demonstrating the Distributional Alignment of Original and Synthetic Data Samples for Each Over-sampling Technique Across dataset Partitions. These visualizations highlight the efficacy of each over-sampling method in replicating the comprehensive distribution characteristics of the original data samples.