The Astrophysical Journal Supplement Series, 271:39 (17pp), 2024 April https://doi.org/10.3847/1538-4365/ad21e4

Identifying Flare-indicative Photospheric Magnetic Field Parameters from Multivariate Time-series Data of Solar Active Regions

Khaznah Alshammari[1], Shah Muhammad Hamdi[2], and Soukaina Filali Boubrahimi[2]
[1] New Mexico State University, Las Cruces, NM, USA
[2] Utah State University, Logan, UT, USA
Received 2023 July 4; revised 2024 January 19; accepted 2024 January 21; published 2024 March 19

Abstract

Photospheric magnetic field parameters are frequently used to analyze and predict solar events. Observation of these parameters over time, i.e., representing solar events by multivariate time-series (MVTS) data, can determine relationships between magnetic field states in active regions and extreme solar events, e.g., solar flares. We can improve our understanding of these events by selecting the most relevant parameters that give the highest predictive performance. In this study, we propose a two-step incremental feature selection method for MVTS data using a deep-learning model based on long short-term memory (LSTM) networks. First, each MVTS feature (magnetic field parameter) is evaluated individually by a univariate sequence classifier utilizing an LSTM network. Then, the top performing features are combined to produce input for an LSTM-based multivariate sequence classifier. Finally, we tested the discrimination ability of the selected features by training downstream classifiers, e.g., Minimally Random Convolutional Kernel Transform and support vector machine. We performed our experiments using a benchmark data set for flare prediction known as Space Weather Analytics for Solar Flares. We compared our proposed method with three other baseline feature selection methods and demonstrated that our method selects more discriminatory features compared to other methods. Due to the imbalanced nature of the data, primarily caused by the rarity of minority flare classes (e.g., the X and M classes), we used the true skill statistic as the evaluation metric. Finally, we reported the set of photospheric magnetic field parameters that give the highest discrimination performance in predicting flare classes.

Unified Astronomy Thesaurus concepts: Solar flares (1496); Space weather (2037); Time series analysis (1916); Solar active region magnetic fields (1975); Multivariate analysis (1913); Solar physics (1476)

1. Introduction

Photospheric magnetic field parameter values such as helicity, flux, and Lorentz force are used to identify flaring events of the Sun (Bobra & Couvidat 2015; Angryk et al. 2020). Solar flares are associated with the release of magnetic energy stored in the Sun's atmosphere, particularly around active regions (ARs) where magnetic fields are strong and complex. Radiation resulting from extreme-ultraviolet, X-ray, and gamma-ray emissions during major flaring events can have harmful effects on life and infrastructure in both space and on the ground. From the radiation-exposure-based health risks of astronauts to multiscale damage in our technology-dependent society (e.g., malfunctioning of numerous electronic devices, disruption in GPS and radio communication, etc.), the economic loss of the modern world due to an extreme solar event can rise up to trillions of dollars (Eastwood et al. 2017). While the complete characterization of the energy release mechanisms in solar flares remains unknown, it is evident that they are primarily of a magnetic nature. Therefore, a crucial aspect of comprehending and ultimately forecasting solar flares involves examination of the magnetic field configuration in solar ARs (Kazachenko 2023). A theoretical relationship between magnetic field influx and the occurrence of flaring activities in solar ARs has not yet been established; therefore, space weather researchers frequently use data-science-based approaches for predicting solar events. The primary data source used in these efforts is the images captured by the Helioseismic Magnetic Imager (HMI) located on the Solar Dynamics Observatory (SDO). HMI images (captured in near-continuous time) contain spatiotemporal magnetic field data of solar ARs. For performing temporal-window-based flare prediction of an AR instance, the magnetic field data of that region are mapped into a multivariate time-series (MVTS) instance (Angryk et al. 2020). MVTS instances, collected with a uniform sampling rate throughout a preset observation period, are labeled with multiple flare classes (e.g., flare-quiet, A, B, C, M, and X) based on the peak X-ray flux observed by GOES, and machine-learning-based classifiers are trained with labeled MVTS instances to predict the occurrences of the events within a preset prediction window.

Space Weather Analytics for Solar Flares (SWAN-SF) is a comprehensive, MVTS data set extracted from solar photospheric vector magnetograms by the Space-weather HMI Active Region Patch (SHARP) series. We used the SWAN-SF data set in our study because of its realistic class imbalance properties (Tafazoli & Keogh 2023). SWAN-SF is attributed with a 60:1 imbalance ratio for GOES M- and X-class flares and an 800:1 imbalance ratio for X-class flares against flare-quiet instances. Multiple research efforts (Hamdi et al. 2017; Ma et al. 2017; Ahmadzadeh et al. 2021; Muzaheed et al. 2021) addressed MVTS-based solar flare prediction, and their work centered around the classification of magnetic field time series of ARs of different flare classes. Hamdi et al. (2017) and Ahmadzadeh et al. (2021) used statistical summarization of individual time series for computing low-dimensional representations of MVTS instances and trained downstream

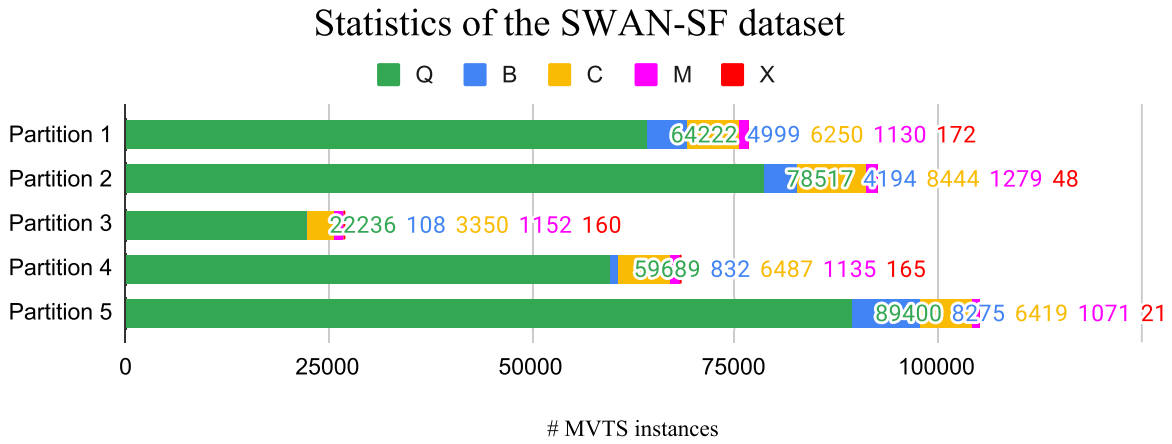

Figure 1. Stacked bar plot of MVTS populations for each of the five flare classes across different partitions of the SWAN-SF benchmark data set. Flare classes X, M, C, and B are taken and validated from GOES classification, while Q denotes flare-quiet and GOES A-class instances. The annotated populations stem from the current time-series slicing strategy of the SWAN-SF with a 12 hr observation window and a 24 hr prediction window. The sliced multivariate time series are labeled with the class of the strongest reported flare, if any, within the respective prediction window (Ahmadzadeh et al. 2021).

classifiers, e.g., support vector machine (SVM) for binary flare prediction. Muzaheed et al. (2021) used an end-to-end long short-term memory (LSTM)–based deep sequence model for classifying MVTS instances of different flare classes. Ma et al. (2017) proposed an MVTS decision tree model for binary flare prediction. From the regression perspective, Alshammari et al. (2022b) addressed forecasting future values of magnetic field parameters, given past values in the MVTS representations.

In this work, we propose a two-step deep-learning framework for feature selection from MVTS-represented solar flare data. In the first step, a 1D LSTM-based univariate sequence classification is performed to compute the true skill statistic (TSS; individual importance score) corresponding to each individual magnetic field parameter. In the second step, the parameters' individual importance scores are sorted in descending order, and parameters with the highest individual importance scores are combined to perform multivariate sequence classification through multiple LSTM networks (that handle 1D, 2D, .., ND inputs, where N is the number of magnetic field parameters). The second step provides us with an optimal set of parameters (features) that can be used for testing the discrimination ability of the selected features with downstream classifiers (e.g., SVM, decision tree, MINImally RandOm Convolutional KErnels Transform (MINIROCKET), etc.). Finally, we have identified and reported the most relevant set of magnetic field parameters for the prediction of major/minor flare classes from multiple partitions of the SWAN-SF data set. Selecting the most relevant and informative parameters can help reduce prediction errors by avoiding overfitting (Kamalov et al. 2022). Feature selection is also an effective measure for data compression and visualization (Filali Boubrahimi & Hamdi 2022). We summarize the contributions of this paper as follows.

1. We propose a novel machine-learning algorithm for feature selection from MVTS data using LSTM-based univariate and multivariate sequence learning.
2. We experiment the proposed MVTS feature selection algorithm on five partitions of the benchmark flare prediction data set SWAN-SF, where each partition is featured with an extreme class imbalance property.
3. Finally, we report the most important solar magnetic field parameters after partition-wise experiments on the SWAN-SF data set.

This paper is organized as follows. In Section 2, we introduce the SWAN-SF benchmark data set. In Section 3, we discuss the related work. Section 4 presents the LSTM-based feature selection model from MVTS data. In Section 5, we demonstrate our experimental findings. Section 5.5 discusses our insights found from the experiments. In Section 6, we conclude our study and outline our future research directions.

2. SWAN-SF Benchmark Data Set

As the benchmark data set of our experiments, we use the MVTS-based solar flare prediction data set SWAN-SF published by Angryk et al. (2020). The SWAN-SF benchmark data set is a collection of MVTS data instances that facilitate unbiased flare forecasting. The MVTS instances of the SWAN-SF benchmark data set are labeled by five different flare classes, namely, GOES-based X, M, C, and B, and a nonflaring class denoted by Q. Class Q includes flare-quiet events and GOES A-class events. The data set comprises five temporally segmented partitions and is designed in a way that each partition includes approximately the same number of X- and M-class flares. Figure 1 shows the partition-wise label statistics of the SWAN-SF data set. The data set contains various time-series parameters derived from solar photospheric magnetograms along with NOAA's flare history of ARs. The magnetograms and their metadata are obtained from the SHARP data product. The magnetic field parameters are physics based and were recalculated and enhanced for validation purposes. Each MVTS instance in the data set is made up of 24 time series of AR magnetic field parameters (the full list can be found in Table 1). The references in Table 1 indicate the initial use of these parameters for flare prediction using machine-learning algorithms. The time-series instances are recorded at 12 minute intervals for a total duration of 12 hr (which results in 60 time steps). In this paper, $T = 60$ is used to denote the number of observation time steps, and $N = 24$ is used to denote the number of magnetic field parameters. In our experiments of feature selection from MVTS data, we conduct the binary classification between flaring and nonflaring MVTS instances, where we consider major flaring events (classes X and M) to be in a positive class and nonflaring events (class Q) to be in the negative class during the training of the classifiers. We remove the B- and C-class events during training (where

Table 1
List of Active Region Magnetic Field Parameters

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

the B- and C-class events were preserved in the test data as Q class), since the opposite event classes (X + M versus Q) help in contrastive learning. The removal of the B- and C-class flares for maximizing flare prediction performance was also suggested by the experimental findings of multiple previous studies (Bobra & Couvidat 2015; Hamdi et al. 2017).

3. Related Work

In space weather research, studies of solar events enhance our understanding of their impact on Earth's environment, particularly focusing on how these phenomena can influence the geomagnetic fields (Johnson et al. 2022). Extreme solar events, e.g., major solar flares, can affect the radiation environment outside the Earth's magnetosphere, which is explored by many space missions. These events can cause extreme radiation-exposure-based health risks to astronauts on space missions (Sadykov et al. 2017; Rotti et al. 2020). Recent research efforts on solar flare prediction mostly are based on data science, e.g., machine learning. Data-driven extreme solar event prediction models stem from linear and nonlinear statistics. Data sets used in the statistical models were collected from line-of-sight magnetogram and vector magnetogram data. A line-of-sight magnetogram contains only the line-of-sight component of the magnetic field, while a vector magnetogram contains the full-disk magnetic field data (Boubrahimi et al. 2016). NASA launched the SDO in 2010. Since then, SDO's

HMI has been mapping the full-disk vector magnetic field every 12 minutes (Bobra & Couvidat 2015). Magnetic field parameters (e.g., helicity, flux, etc.) were developed to find relationships between the photospheric magnetic field states and the following solar activities. Most of the recent prediction models use the near-continuous stream of vector magnetogram data produced by SDO (Mason & Hoeksema 2010). Ahmadzadeh et al. (2021) proposed a machine-learning-based method to train a flare prediction model for predicting solar flares. The paper focuses on addressing the issue of imbalanced data in the training data set, where there are significantly fewer positive examples (flares) than negative examples (nonflares). Their work highlights the importance of addressing the issue of imbalanced data in training predictive models. Up until now, the focus of data-science efforts has primarily been on predicting flares through magnetic field parameter-based MVTS classification. However, the identification of the most significant magnetic field parameters through feature selection of MVTS data has yet to be addressed.

Feature selection has proven to be an effective technique in data preprocessing, which enhances the performance of machine-learning models, particularly when dealing with data sets that have a high number of dimensions (Hamdi et al. 2019). For solar flare data, each instance of a flaring or nonflaring AR is represented by an MVTS representation of magnetic field parameters. High dimensionality in this MVTS data set may result in low performances for the classifiers. Dimensionality reduction by selecting the most discriminatory features can increase the classification performance by reducing the overfitting tendency (Peng et al. 2005; Gu et al. 2012).

Feature selection methods can be categorized into three categories: filter based, wrapper based, and embedded (Alshammari et al. 2022a). Filter-based methods involve ranking the features as a preprocessing step before applying the learning algorithm, where they incrementally select features with the top discrimination scores. Wrapper-based methods, on the other hand, score the features using the specific learning algorithm that will be used in the final model. Embedded methods combine feature selection with the learning algorithm itself, where they are closely tied to a specific learning algorithm that limits their applicability to other algorithms (Guyon & Elisseeff 2003). For this study, we focus on filter-based methods for supervised feature selection. The Fisher score (FS), mutual information (MI), and minimal redundancy maximal relevance (mRMR) are commonly used filter-based methods for feature selection algorithms that aim to identify relevant and informative features for classification and regression tasks. The FS is a filter-based feature selection criterion that evaluates the discriminative power of each feature by comparing the between-class and within-class variability. It assigns higher scores to features that show significant differences between different classes or categories in the data (Gu et al. 2012). MI is a measure of statistical dependency between two random variables. In the context of feature selection, MI measures the relevance or dependency of each feature with respect to the target variable. Features with high MI scores are considered more informative and are selected as relevant features (Batina et al. 2011; Li et al. 2022). mRMR is a feature selection algorithm that combines both relevance and redundancy considerations. It aims to select a subset of features that have high relevance to the target variable while minimizing redundancy among the selected features. mRMR evaluates both the MI between each feature and the target variable (maximal relevance) and the MI between features themselves (minimal redundancy) to find an optimal subset of features (Peng et al. 2005).

Deep-learning-based sequence models such as recurrent neural networks (RNNs) learn representations from multidimensional sequential data, e.g., univariate/MVTS data (Mikolov et al. 2010; Boubrahimi et al. 2020). They have been widely used in numerous sequence learning tasks such as machine translation (Bahdanau et al. 2014) and text summarization (Cascalheira et al. 2022). They process sequential data by maintaining an internal state or memory, allowing them to retain information about past inputs and make context-based decisions (Bahri et al. 2022; Hosseinzadeh et al. 2023). LSTM is a type of RNN architecture that captures and retains long-term dependencies in sequential data by incorporating specialized memory cells (Hochreiter & Schmidhuber 1997; Yu et al. 2019).

In our study, we propose a model-agnostic approach, which is a novel two-step deep-learning framework for feature selection from MVTS data. In the first step, we apply a univariate sequence learning method using LSTM for computing individual discrimination scores of the magnetic field parameters. In the second step, we apply LSTM-based multivariate sequence learning to identify the set of magnetic field parameters that jointly distinguish flare classes with maximum performance. We can use any downstream classifier to test the discrimination ability of the selected features in the second step.

4. LSTM-based Feature Selection from MVTS Data

4.1. Notations

Solar event instance i is represented by MVTS instance mvts_i. MVTS instance $\text{mvts}_i \in \mathbb{R}^{T\times N}$ is a collection of individual time series of N magnetic field parameters, where each time series contains periodic observation values of the corresponding parameter for observation period T. In MVTS instance $\text{mvts}_i = \{v_{t_1}, v_{t_2}, .,.,.,., v_{t_T}\}$, $v_{t_i} \in \mathbb{R}^N$ represents a time stamp vector.

4.2. Long Short-term Memory

LSTM networks (Hochreiter & Schmidhuber 1997) are deep-learning models that are commonly used in learning from sequences (e.g., text, speech signals, and time series). They are designed to deal with the problem of vanishing gradients that can occur in RNNs, which makes them unable to learn long-range dependencies in sequential data effectively. The architecture consists of memory cells that can store information for long periods and a series of gates that control the flow of information into and out of the cells. The gates consist of sigmoid activation functions that allow the network to learn which information to keep or forget. The input gate controls the flow of information into the cell, the forget gate determines which information to discard, and the output gate controls the flow of information out of the cell. LSTM has been successfully applied in various tasks, such as speech recognition (Oruh et al. 2022), image captioning (Wang et al. 2016), and weather forecasting (Srivastava & S 2022). The

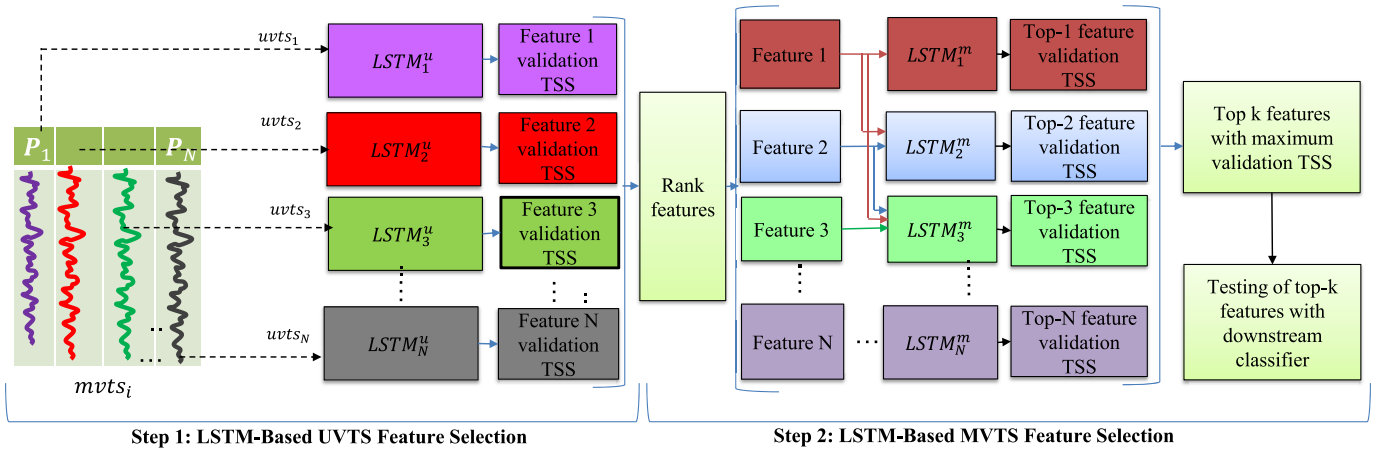

Figure 2. Deep-learning-based MVTS feature selection framework.

mathematical definition of LSTM is as the following:

$$
\begin{aligned}
f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f),\\
i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i),\\
\tilde{C}_t &= \tanh(W_C x_t + U_C h_{t-1} + b_C),\\
C_t &= f_t \odot C_{t-1} + i_t \odot \tilde{C}_t,\\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o),\\
h_t &= o_t \odot \tanh(C_t).
\end{aligned}
$$

Here, x_t is the input at time step t, h_{t-1} is the hidden state at time step $t-1$, f_t is the forget gate that controls how much information is discarded from the previous cell state, i_t is the input gate that controls how much information is added to the current cell state, $\tilde{C}_t$ is the candidate cell state at time step t, C_t is the cell state at time step t, o_t is the output gate that controls how much information is extracted from the cell state, h_t is the hidden state at time step t, σ is the sigmoid function, $\odot$ is the element-wise multiplication operator, and W_f, W_i, W_C, W_o, U_f, U_i, U_C, U_o, b_f, b_i, b_C, and b_o are the learnable parameters that are trained through the gradient-descent-based backpropagation algorithm.

The application of LSTM units in our feature selection model is shown in Figure 2. We can easily integrate RNN and gated recurrent unit (GRU; Chung et al. 2014) cells into our model by substituting the LSTM cells with either RNN or GRU cells. However, we have chosen to employ LSTM cells due to their superior effectiveness in managing long-term dependencies in time-series data. The study conducted by Muzaheed et al. (2021) demonstrated that the LSTM-based classification model outperforms both GRU and RNN models in accurately classifying solar flare data.

4.3. Data Preprocessing and Normalization

Since the magnetic field parameter values are recorded at different scales, we perform z-score normalization of each individual time series of each MVTS instance. An mvts_i, parameter-based individual time series is denoted by P_1, P_2,...,P_N. For each individual time series P_j, we perform z-normalization as follows:

$$x_k^{(j)} = \frac{x_k^{(j)} - \mu^{(j)}}{\sigma^{(j)}}.$$

Here, $x_k^{(j)}$ is the kth value of time series P_j, where $1 \leqslant k \leqslant T$, $\mu^{(j)}$ is the mean of time series P_j, and $\sigma^{(j)}$ is the standard deviation of time series P_j. We apply the z-normalization for each partition individually. When partition i is used for training and partition j is used for testing, we perform the above z-normalization independently for the MVTS instances of partition i and j.

4.4. LSTM-based MVTS Feature Selection Framework

First, we select two partitions from the data set: one for training and validation, and the other for testing. Then we apply a two-step deep-learning-based feature selection approach (Figure 2). The first step uses sequence classification from the univariate time series (UVTS) of the individual parameters. After training N univariate LSTM classifiers (LSTM^u) with each parameter individually, we calculate the validation TSS of each parameter to get an importance score for each parameter. In the second step, we rank the parameters according to their individual validation TSS (importance score) from highest to lowest. Then, we incrementally select the top K-performing parameters with the high validation TSS and train the LSTM-based multivariate sequence classifiers (LSTM^m) with the multiple selected parameters. The parameter set that produced the maximum validation TSS is selected for testing on the test set using a downstream classifier (e.g., SVM, decision tree, MINIROCKET, etc.).

Algorithm 1. LSTM-based UVTS Feature Selection

Input: Training set $X\_train \in \mathbb{R}^{n_{\text{train}} \times N \times T}$, training labels $Y\_train \in \mathbb{R}^{n_{\text{train}} \times C}$, validation set $X\_val \in \mathbb{R}^{n_{\text{val}} \times N \times T}$, validation labels $Y\_val \in \mathbb{R}^{n_{\text{val}} \times C}$, and number of training epochs n_{epochs}. Here, C is the number of classes, n_{train} and n_{val} are the number of examples in the training and validation sets, respectively, and the labels are in one-hot representation.
Output: Validation TSS score of each individual parameter.
1: **for** parameters $j = 1$ to N **do**
2: Initialize weight and bias of LSTM_j^u
3: **for** number of training epochs $e = 1$ to n_{epochs} **do**
4: **for** MVTS instance $i = 1$ to n_{train} **do**
5: $train\_mvts = X\_train[i, :, :]$
6: $train\_uvts = train\_mvts[j, :]$
7: $target = Y\_train[i]$
8: $scores = LSTM_j^u(train\_uvts)$
9: $loss = negative\_log\_likelihood(scores, target)$
10: $loss.backward()$
//backpropagation-based weight update
11: **end for**
12: **end for**

(Continued)

13: **for** MVTS instance $i = 1$ to n_{val} **do**
14: $val\_mvts = X\_val[i, :,:]$
15: $val\_uvts = val\_mvts[j, :]$
16: $val\_label = Y\_val[i]$
17: $val\_class\_scores = LSTM_j^u(val\_uvts)$
18: $Y\_val\_pred[i] = argmax(val\_class\_scores)$
19: **end for**
20: TP_val, TN_val, FP_val, FN_val= $confusion\_matrix(Y\_val\_pred, Y\_val)$
21: $val\_TSS = TP\_val/(TP\_val + FN\_val) - FP\_val/(FP\_val + TN\_val)$
22: $univariate\_importance[j] = val\_TSS$
23: **end for**
24:**return** *univariate_importance*

Algorithms 1 and 2 explain our two-step MVTS feature selection method for MVTS data. Algorithm 1 describes the process of feature selection using an LSTM-based UVTS learning approach. The algorithm takes as input the training set *X_train* (a 3D array representing the input time-series data), training labels *Y_train*, validation set *X_val*, validation labels *Y_val*, and the number of training epochs *n_epochs*. It aims to compute the validation TSS score for each individual parameter. The first loop iterates over the parameters *j* from 1 to *N*, where *N* represents the total number of parameters/features. For each parameter *j*, the weights and biases of the LSTM unit LSTM_j^u are initialized. The inner loop iterates *n_epochs* times, representing the number of training epochs. Within the nested loops, the algorithm performs training on the training set. It iterates over each MVTS instance *i* in the training set. The MVTS instance *train_mvts* is extracted from *X_train*, and UVTS *train_uvts* is obtained by selecting the *j*th parameter from *train_mvts*. The target label *target* is obtained from *Y_train* for the current MVTS instance. The LSTM unit LSTM_j^u is fed with *train_uvts*, and the output scores are predicted. The negative log-likelihood loss is calculated based on the predicted scores and the target label. The backpropagation is performed by calling *loss.backward*() to update the weights and biases of LSTM_j^u using gradient information. After training, the algorithm proceeds to evaluate the trained LSTM unit on the validation set. Similar to the training phase, the algorithm iterates over each MVTS instance *i* in the validation set. The MVTS instance *val_mvts* is extracted from *X_val*, and UVTS *val_uvts* is obtained by selecting the *j*th parameter from *val_mvts*. The true label *val_label* is obtained from *Y_val* for the current MVTS instance. The LSTM unit LSTM_j^u is fed with *val_uvts* to obtain class scores. The predicted label for the MVTS instance is determined by taking the `argmax` of the class scores. The true positive (*TP_val*), true negative (*TN_val*), false positive (*FP_val*), and false negative (*FN_val*) values are computed by comparing the predicted labels with the true labels using a confusion matrix (truth table). The validation TSS (*val_TSS*) is calculated based on the TP, TN, FP, and FN values. TSS is a measure that combines sensitivity and specificity for binary classification tasks. The computed *val_TSS* is stored in the *univariate_importance* array at the *j*th index, representing the importance of the *j*th parameter. The outer loop continues until all parameters have been processed. Finally, the algorithm returns the *univariate_importance* array containing the TSS scores for each individual parameter, representing their relative importance in the classification task. In sum, the algorithm iteratively trains an LSTM unit for each parameter and evaluates its performance using the validation TSS score, allowing for the selection of important features based on their individual contributions to the classification task.

Algorithm 2. LSTM-based MVTS Feature Selection

Input: $univariate\_importance \in \mathbb{R}^N$, X_{train}, Y_{train}, X_{val}, Y_{val}, and n_{epochs}
Output: Feature set producing the best validation TSS.
1: Sort the parameters in descending order according to *univariate_importance*, and initialize *topK_features* as an empty list of lists
2: **for** sorted parameters $j = 1$ to N **do**
3: $top\ K = list(1{:}\ j)$
4: Initialize weight and bias of LSTM_j^m
5: **for** number of training epochs $e = 1$ to n_{epochs} **do**
6: **for** MVTS instance $i = 1$ to n_{train} **do**
7: $train\_mvts = X\_train[i, :,:]$
8: $train\_mvts\_topK = train\_mvts[topK, :]$
9: $target = Y\_train[i]$
10 $scores = LSTM_j^m(train\_mvts\_topK)$
11 $loss = negative\_log\_likelihood(scores, target)$
12 $loss.backward()$
13: **end for**
14: **end for**
15: **for** MVTS instance $i = 1$ to n_{val} **do**
16: $val\_mvts = X\_val[i, :,:]$
17: $val\_mvts\_topK = val\_mvts[topK, :]$
18: $val\_label = Y\_val[i]$
19: $val\_class\_scores = LSTM_j^m(val\_mvts\_topK)$
20: $Y\_val\_pred[i] = argmax(val\_class\_scores)$
21: **end for**
22: $TP\ val$, $TN\ val$, $FP\ val$, $FN\ val$= $confusion\_matrix(Y\_val\_pred, Y\_val)$
23: $val\_TSS = TP\_val/(TP\_val + FN\_val) - FP\_val/(FP\_val + TN\_val)$
24: $multivariate\_importance[j] = val\_TSS$
25: $topK\_features[j] = topK$
26: **end for**
27: **return** $topK\_features[argmax(multivariate\_importance)]$

Algorithm 2 takes the same inputs as Algorithm 1 along with a list of *univariate_importance*, which stores the TSS values representing the individual importance of each parameter found in Algorithm 1. The algorithm aims to select a feature set by sorting the parameters in descending order according to their *univariate_importance*. For each sorted parameter *j* from 1 to *N*, it initializes an empty list of lists called *topK_features*. Then, it sets *topK* as a sublist containing the first *j* parameters. After that, it initializes the weight and bias of an LSTM model denoted as LSTM_j^m. It performs training for n_{epochs} on the training data set. For each MVTS instance *i* in the training data set it retrieves the MVTS instance *train_mvts*. The top *j* features from *train_mvts* are selected and assigned to *train_mvts_topK*. Then, the target label *target* is retrieved. It passes *train_mvts_topK* through LSTM_j^m to obtain *scores*. The negative log-likelihood loss is calculated between *scores* and *target*. After that, it performs backpropagation to update the weights and biases of LSTM_j^m. After training, it evaluates the performance on the validation data set for each MVTS instance *i* in the validation data set. It retrieves the MVTS instance *val_mvts*, selects the top *j* features from *val_mvts*, and assigns it to *val_mvts_topK*. It retrieves the target label *val_label*. The algorithm passes *val_mvts_topK* through LSTM_j^m to obtain *val_class_scores*. Then, the predicted label for *val_mvts* is stored as *Y_val_pred*[*i*]. It calculates the true positives (*TP_val*), true negatives (*TN_val*), false positives (*FP_val*),

and false negatives (*FN_val*) based on the confusion matrix of *Y_val_pred* and *Y_val*. Then, the validation TSS is calculated. The *val_TSS* is stored in the *multivariate_importance* array at index *j* and the current *topK* feature set is stored in the *topK_features* array at index *j*. Finally, the algorithm returns the feature set *topK_features* that produces the best validation TSS. It selects the feature set corresponding to the index with the maximum value in the *multivariate_importance* array. In summary, Algorithm 2 performs feature selection for an MVTS using an LSTM model. It iteratively selects features based on their univariate importance and evaluates the performance of the selected feature set using the validation TSS metric. The feature set that maximizes the TSS is returned as the output.

5. Experiments

In this section, we demonstrate our experimental methods and results. In the first part, we present the prediction performance on mean-reduced data, where we compare our LSTM-based feature selection model with three baseline methods (FS, MI, and mRMR) for selecting the best discriminatory features in the MVTS-based flare prediction data set. In this part, first we extract important features from the training and validation partition using each baseline method (where we apply mean reduction to the data set), and then we use the selected features for classifying the test instances using a downstream classifier (e.g., SVM, decision tree, *k*-nearest neighbors (*k*NN), random forest, and XGBoost). We report the best-selected features found by each method and the corresponding test TSS of the downstream classifier. In the second part, we apply our parameter selection method to the full time-series data. Then, we use three MVTS classifiers as downstream models (LSTM, MiniRocket, and the transformer model) to classify MVTS instances of the selected parameters.

5.1. Evaluation Metrics

We use TSS as a performance measure for our experiments. The TSS takes into account both the hits and false alarms in the prediction. It is calculated as:

$$\mathrm{TSS} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} - \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}.$$

Here, TP is the number of true positives (correct predictions of flares), FN is the number of false negatives (missed predictions of flares), FP is the number of false positives (incorrect predictions of flares), and TN is the number of true negatives (correctly predicted nonflares). TSS spans from -1 to $+1$, demonstrating equitability: 0 for random or constant forecasts, 1 for perfect forecasts, and -1 for consistently incorrect forecasts (Bloomfield et al. 2012). A TSS score of -1 is just as useful as one that is always right, and the absolute value of the TSS is relevant. Another advantageous aspect of the TSS is its impartiality concerning the class imbalance ratio. It means that it can accurately measure the model's ability to distinguish between the classes, regardless of how common or rare they are.

5.2. Training, Validation, and Testing Sets Splitting Method

The SWAN-SF data set has a temporal coherence property that measures how stable and consistent the magnetic field structures of a solar AR are over time. It poses a challenge for forecasting rare events such as solar flares using time-series data. It requires that the predictions for a given time point are in agreement with past and future predictions. If temporal coherence is ignored, the model performance may be artificially inflated. This problem stems from the data collection method and affects the data splitting into training, validation, and testing sets. To address the issue of temporal coherence, we use different time-segmented partitions of the data set for training and testing. This is the reason why the SWAN-SF data set has multiple nonoverlapping partitions. By using different partitions of SWAN-SF for training and testing, we avoid testing the model on a time series that is partly identical to those used for training (Ahmadzadeh et al. 2021). We use one partition for training and validation and another partition for testing. Specifically, we use partition i for training and validation, and partition j for testing, where $i \neq j$ and $i, j = \{1, 2, 3, 4, 5\}$. In Figure 3, we show this training/testing partition strategy using a complete graph with bidirectional edges. In each training partition, we apply random sampling for training/validation splitting, using the stratified splitting method (80% for training and 20% for validation) to ensure nearly equal class ratios in the partitions. We report the mean and standard deviation of the TSS for the test partitions. For example, when partition 1 is used for training and validation, we test with partitions 2, 3, 4, and 5 separately. This results in 20 different training/testing settings as shown in Figure 3. We use eight downstream classifiers on the data (five for classifying mean-reduced MVTS instances: SVM, *k*NN, decision tree, random forest, and XGBoost; and three for classifying full-length MVTS instances: LSTM, MINIROCKET, and transformer), which leads to 160 different experiments (100 for vector classification and 60 for MVTS classification).

5.3. Prediction Performance on Mean-reduced Data

In this experiment, we use mean reduction, where for each MVTS instance mvts_i we calculate the mean of each time series (representing the feature or magnetic field parameter) to get a vector-based data set. Then, we use the mean-reduced data as input for each classifier. The purpose of testing the effectiveness of the selected features on vector-represented data is that most flare prediction models (e.g., Hamdi et al. 2017; Ahmadzadeh et al. 2021) perform MVTS to vector transformation when applying traditional classifiers. In this task, first, we use mean-reduced data for both feature selection (with three baseline feature selection methods) and downstream classification. Second, we use the full time-series data for feature selection using our proposed method, and perform downstream classification on the mean-reduced data of selected features.

5.3.1. Baselines Feature Selection Methods

As baseline feature selection methods, we use FS, MI, and mRMR.

1. *Fisher score*. FS is a feature selection criterion that aims to find a subset of features such that in the data space spanned by the selected features, the distances between data points of different classes are as large as possible, while the distances between data points in the same class are as small as possible (Gu et al. 2012). It applies importance scores to each feature independently in accordance with their class labels. The FS formula is used to evaluate the importance of each feature in a given

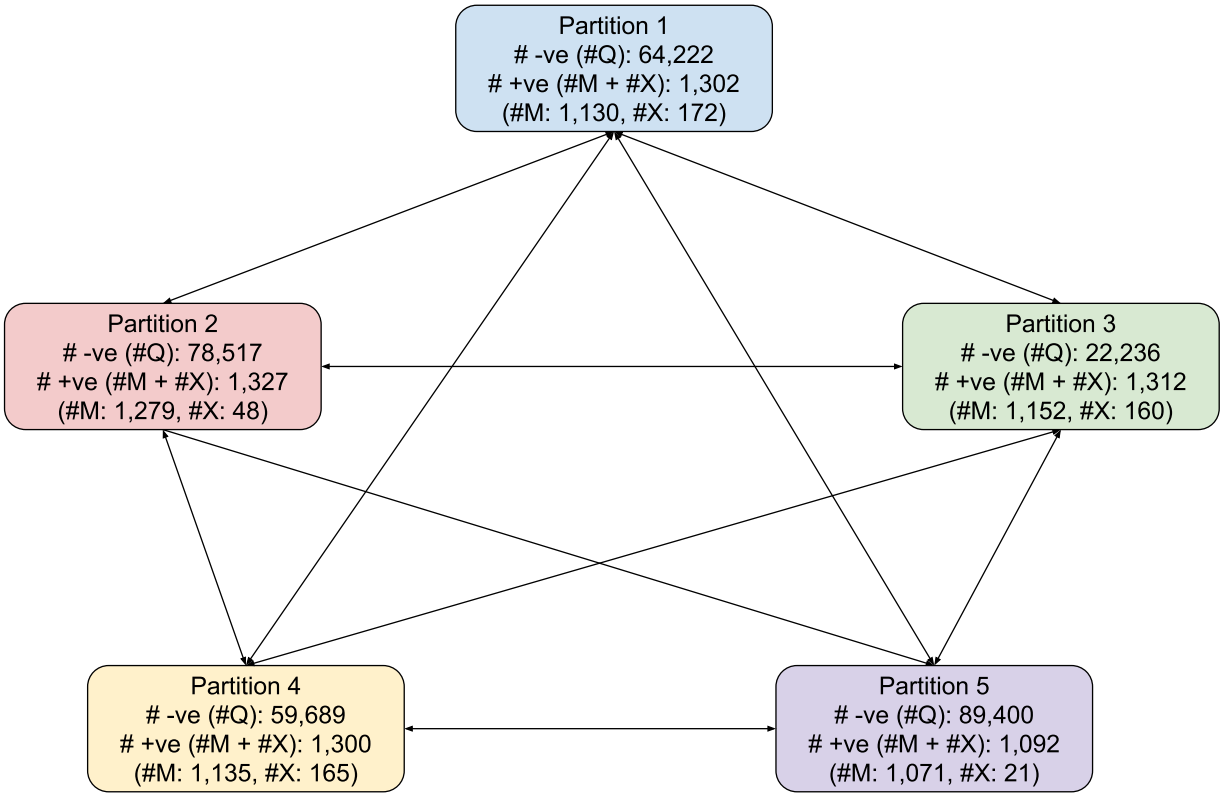

Figure 3. Relationships between the training and testing partitions, and label description of each partition during training. In this directed graph, each edge/link (i, j) represents partition i is used for training (testing) while partition j is used for testing (training).

data set. It is calculated as the following:

$$F(x_j) = \sum_{k=1}^{c} n_k \left(\bar{j}_k - \bar{j}\right)^2 / (\sigma_j)^2, \tag{1}$$

where n_k is the number of samples in class k, $\bar{j}$ is the mean of feature j, σ_j is the variance of feature j, and $\bar{j}_k$ is the mean of feature j in class k (Gu et al. 2012).

2. *Mutual information*. MI is a measure of the mutual dependence between two variables and quantifies the entropy obtained for one random variable (features) by observing the other random variable (labels). It is a measure of the amount of information that two random variables share. It is defined as the reduction in uncertainty of one variable given knowledge of another variable (Batina et al. 2011). The formula for the MI between two discrete random variables X and Y is given by:

$$I(X; Y) = \sum_{y \in Y} \sum_{x \in X} p(x, y) \log \frac{p(x, y)}{p(x)p(y)}. \tag{2}$$

3. *Minimal redundancy maximal relevance*. The mRMR algorithm finds the most relevant and least redundant features (Peng et al. 2005). mRMR is a feature scoring method, where in each iteration it selects the features that have the maximum relevance with the target variable and minimum redundancy with the features that have been selected in previous iterations. It is a filter-based feature selection method that calculates the redundancy between features and the correlation between features and class labels based on MI $I(x; y)$. The mRMR formula is as follows:

$$\max_{F \subseteq \mathcal{X}} I(Y; F) - \frac{1}{k} \sum_{j=1}^{k} I(X_{S_j}; F). \tag{3}$$

Here, $\mathcal{X} = X_1, X_2, ..., X_n$ is a set of n features, Y is the output variable of interest, $F \subseteq \mathcal{X}$ is a subset of features, and $S_1, S_2, ..., S_k$ are disjoint subsets of 1, 2,...,n such that $S_1 \cup S_2 \cup ... \cup S_k = 1, 2, ..., n$. X_{S_j} is the subset of features denoted by S_j. $I(Y; F)$ is the MI between the output variable Y and the feature set F. $I(X_{S_j}; F)$ is the MI between the feature set F and the subset of features X_{S_j}. k is the number of disjoint subsets of features. The goal of the mRMR formula is to find the subset of features F that maximizes the MI between the output variable Y and the selected feature set F, while minimizing the redundancy between the features in F and the subsets of features X_{S_j} (Peng et al. 2005).

For applying the baseline methods to the MVTS data set, we use mean reduction, where for each MVTS instance mvts_i we calculate the mean of each time series to get a vector-based data set. Then we perform feature scoring, where we compute Fisher, MI, and mRMR scores for each parameter. Finally, we apply incremental feature selection based on those scores on the validation data set to find the best candidate parameters for testing them with a downstream classifier. We use SVM (Cortes & Vapnik 1995), kNN (Fix & Hodges 1951), decision tree (Safavian & Landgrebe 1991), random forest (Pal 2005), and XGBoost (Chen & Guestrin 2016) as downstream classifiers.

In our experiments, we use the following hyperparameters for each downstream classifier. The hyperparameters are tuned

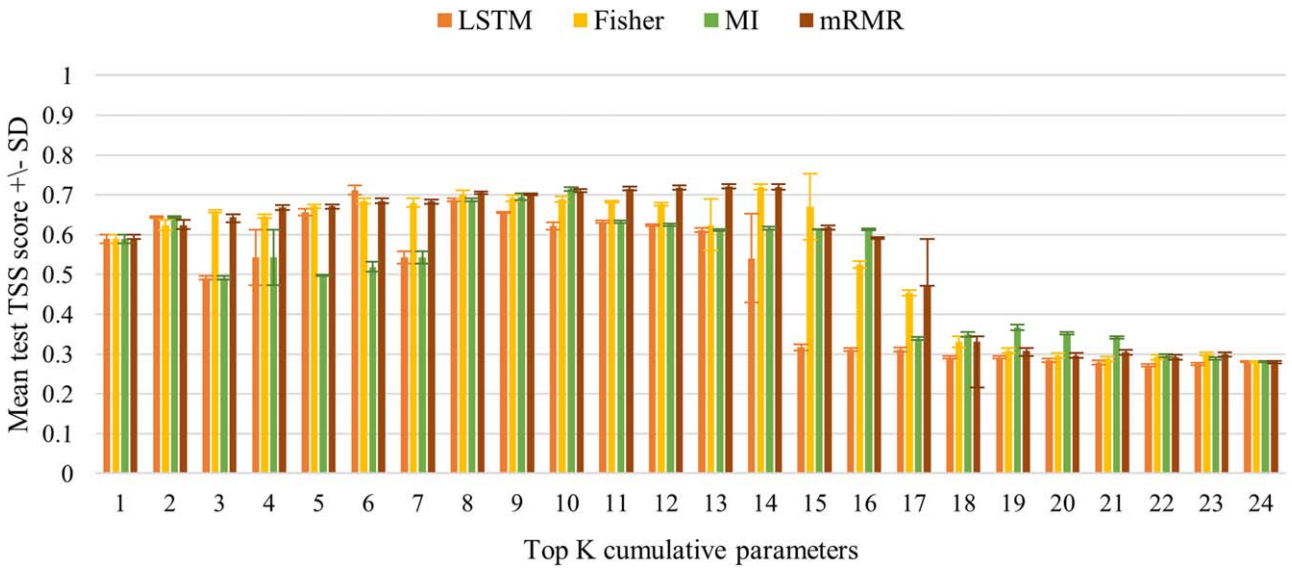

Figure 4. Mean ± standard deviation test TSS score where partition 5 is used for training and validation, partition 3 for testing, and SVM as the downstream classifier.

using a random search strategy (Bergstra & Bengio 2012) according to the classifier's performance for the validation set. For LSTM networks, we use 128 hidden layer units, 20 training epochs, and a learning rate in stochastic gradient descent of 0.01. For the SVM downstream classifier, we set the regularization parameter C to 1.0 and the kernel to a radial basis function. In *k*NN, we use $k = 5$ and uniform weights, where all points in each neighborhood are weighted equally. For the decision tree classifier, the attribute selection criterion Gini is used. For the random forest classifier, the same attribute selection criterion is used, and the number of estimators is set to 100, which is the number of decision trees in the forest.

5.3.2. Comparison of LSTM-based MVTS Feature Selection with Other Baseline Feature Selection Algorithms

For the training and validation partitions, we apply LSTM-based MVTS feature selection to the full time-series data, and the baseline methods FS, MI, and mRMR to the mean-reduced data. After feature selection, we test the selected features of each method with five downstream classifiers on the test partitions consisted of mean-reduced instances. Our experimental results show that the LSTM-based MVTS feature selection method outperforms the baseline approaches, and it gives us the most important set of parameters. We find out that our method gives a TSS of 0.72 with the top six parameters (total unsigned current helicity, total magnitude of the Lorentz force, sum of the modulus of the net current per polarity, total photospheric magnetic free energy density, absolute value of the net current helicity, and total unsigned vertical current), when we use SVM as a downstream classifier, partition 5 for training and validation, and partition 3 for testing, as Figure 4 shows. It also gives a TSS of 0.70 with the top eight parameters (sum of the modulus of the net current per polarity, sum of the flux near the polarity inversion line, absolute value of the net current helicity, total unsigned current helicity, total magnitude of the Lorentz force, total photospheric magnetic free energy density, total unsigned vertical current, and total unsigned flux), when we use *k*NN as a downstream classifier, partition 1 for training and validation, and partition 4 for testing as in Figure 5. We achieve a TSS of 0.74 when we use decision tree as a downstream classifier, with partition 5 for training and validation and partition 4 for testing with 10 parameters (total unsigned current helicity, total magnitude of the Lorentz force, sum of the modulus of the net current per polarity, total photospheric magnetic free energy density, absolute value of the net current helicity, total unsigned vertical current, total unsigned flux, sum of the x-component of Lorentz the force, sum of the y-component of the Lorentz force, and sum of the z-component of the Lorentz force), as in Figure 6. It gives a TSS of 0.71 with the top four parameters (sum of the modulus of the net current per polarity, total unsigned current helicity, absolute value of the net current helicity, and total unsigned vertical current) with random forest as a downstream classifier, where partition 2 is used for training and validation and partition 5 for testing, as Figure 7 shows. The highest test TSS with the smallest amount of parameters achieved is 0.74 with XGBoost as the downstream classifier, where we use partition 1 for training and validation and partition 4 for testing, with the top eight parameters (sum of the modulus of the net current per polarity, sum of the flux near the polarity inversion line, absolute value of the net current helicity, total unsigned current helicity, total magnitude of the Lorentz force, total photospheric magnetic free energy density, total unsigned vertical current, and total unsigned flux), as in Figure 8.

When we apply our two-step LSTM-based MVTS feature selection model to the SWAN-SF data set, we perform the following steps. First, we use each partition data set as the training/validation data set, where we apply the UVTS feature selection to get the importance score for each magnetic field parameter individually. Second, we apply the multivariate feature selection algorithm, which selects features based on their univariate importance and evaluates the performance of the selected feature set using the TSS metric on the validation set. The feature set that maximizes the validation TSS score is returned as the output. After that, we test the top performing parameters (i.e., the feature set that maximizes the validation TSS score) starting from 1 to 24 using the downstream classifier (SVM, *k*NN, decision tree, random forest, and XGBoost) on all the test partitions. For example, when the SVM classifier is applied as the downstream classifier, we have

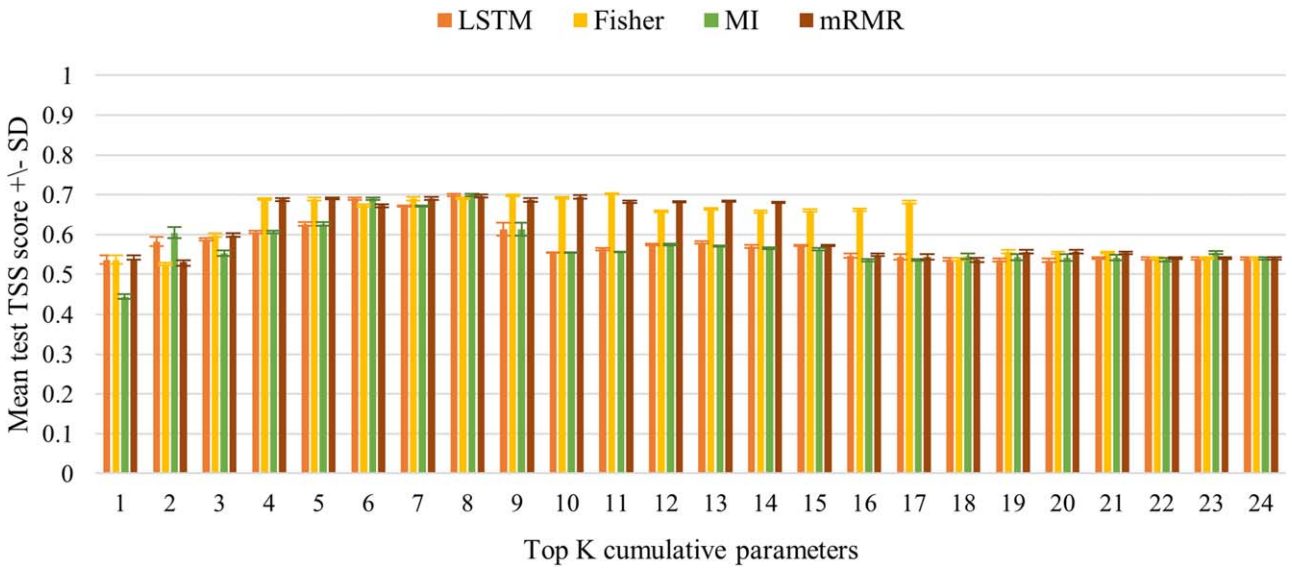

Figure 5. Mean ± standard deviation test TSS score where partition 1 is used for training and validation, partition 4 for testing, and *k*NN as the downstream classifier.

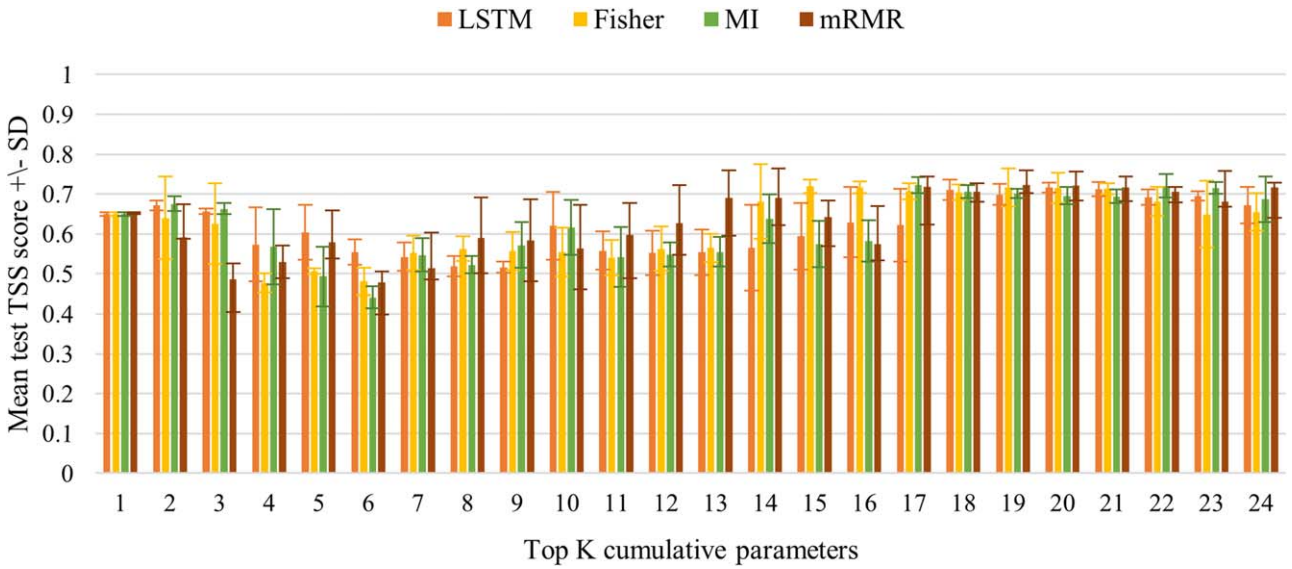

Figure 6. Mean ± standard deviation test TSS score where partition 5 is used for training and validation, partition 4 for testing, and decision tree as the downstream classifier.

20 training plus validation/testing partitions, as Figure 3 shows.

In each partition-based experiment, we extract the top K parameters by our feature selection algorithm. Suppose, parameter p appears m times in the top K list in those 20 trials. Then, parameter p's frequency is $m/20$, which is shown in Table 2. Figures 9 and 10 show the mean frequency and majority vote (winning percentage) for each parameter. The term frequency refers to the following:

$$\text{frequency}_i = n_i/n_{\text{exp}}, \tag{4}$$

where n_i is the number of times that parameter i appears in each classifier experiment, and $n_{\text{exp}} = 20$ is the total number of experiments conducted for each downstream classifier. The term majority vote (winning percentage) refers to:

$$\text{majority\_vote}_i = c_i/(n_{\text{exp}} \times n_{\text{classifiers}}), \tag{5}$$

where c_i is the number of times that parameter i appears across the experiments of all the downstream classifiers (SVM, *k*NN, DT, RF, and XGBoost), and $n_{\text{classifiers}} = 5$ is the number of classifiers.

Table 2 shows the top performing parameters after applying our incremental feature selection method with each downstream classifier, each parameter frequency when applying the SVM, *k*NN, DT, RF, and XGBoost classifiers, and the majority vote for each parameter across all 100 experiments.

5.4. Prediction Performance of the MVTS Classifiers

In this experiment, we use the full MVTS data for downstream classification. Where the downstream classifiers discussed in the previous section are trained and tested on mean-reduced data, the three classifiers considered in this experiment are trained and tested on MVTS data.

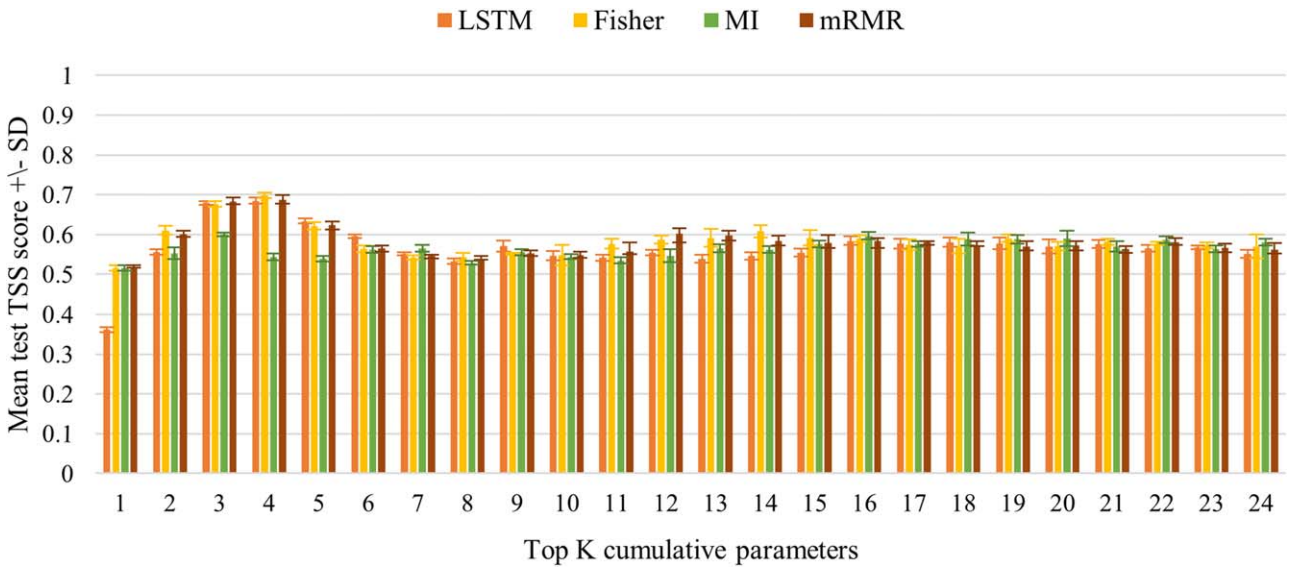

Figure 7. Mean ± standard deviation test TSS score where partition 2 is used for training and validation, partition 5 for testing, and random forest as the downstream classifier.

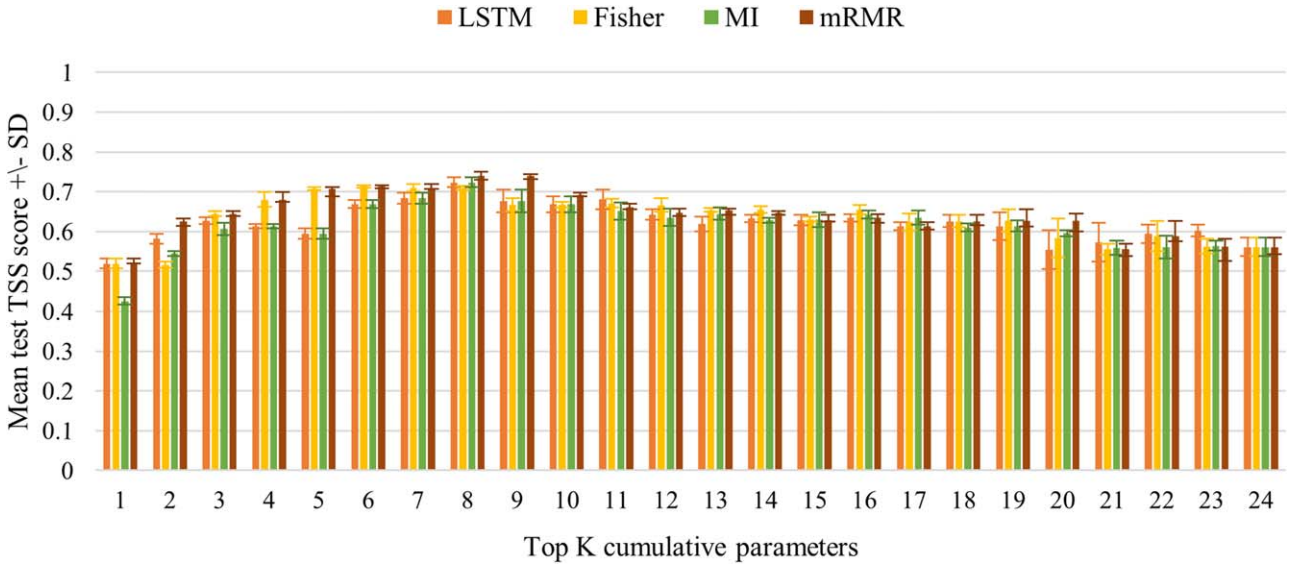

Figure 8. Mean ± standard deviation test TSS score where partition 1 is used for training and validation, partition 4 for testing, and XGBoost as the downstream classifier.

5.4.1. Performance of the Three MVTS Classifiers on the Features Selected by the LSTM-based Feature Selection Method

In this experiment, three MVTS classifiers (LSTM, MINIROCKET, and transformer) are employed for downstream classification. To compare, we train and test these classifiers both before and after feature selection. For each training/testing partition pair, first we train and test the downstream MVTS classifier with all 24 parameters, and finally, we train and test those classifiers with features selected by our proposed model (which is applied to the training/validation partition). Below, we briefly outline the MVTS classifiers.

1. *Long short-term memory.* In this experiment, we use LSTM-based MVTS classification as described in Muzaheed et al. (2021). First, we use all magnetic field parameters in each training/testing partition to report the TSS values that are found without feature selection. Second, we apply our proposed LSTM-based feature selection method in the training/validation partition, and we use the selected features to train and test the downstream LSTM classifier. Figure 11 shows that the selected features improve the TSS values in all partition pairs.
2. *Minimally Random Convolutional Kernels Transform.* MINIROCKET is a fast and accurate algorithm for time-series classification. It is a (nearly) deterministic reformulation of the ROCKET algorithm, which is a state-of-the-art algorithm for time-series classification (Dempster et al. 2021). It transforms the input time series using a small, fixed set of convolutional kernels and passes the features to a linear classifier. The convolutional kernels are designed to capture different temporal patterns in the time-series data. MINIROCKET is more robust

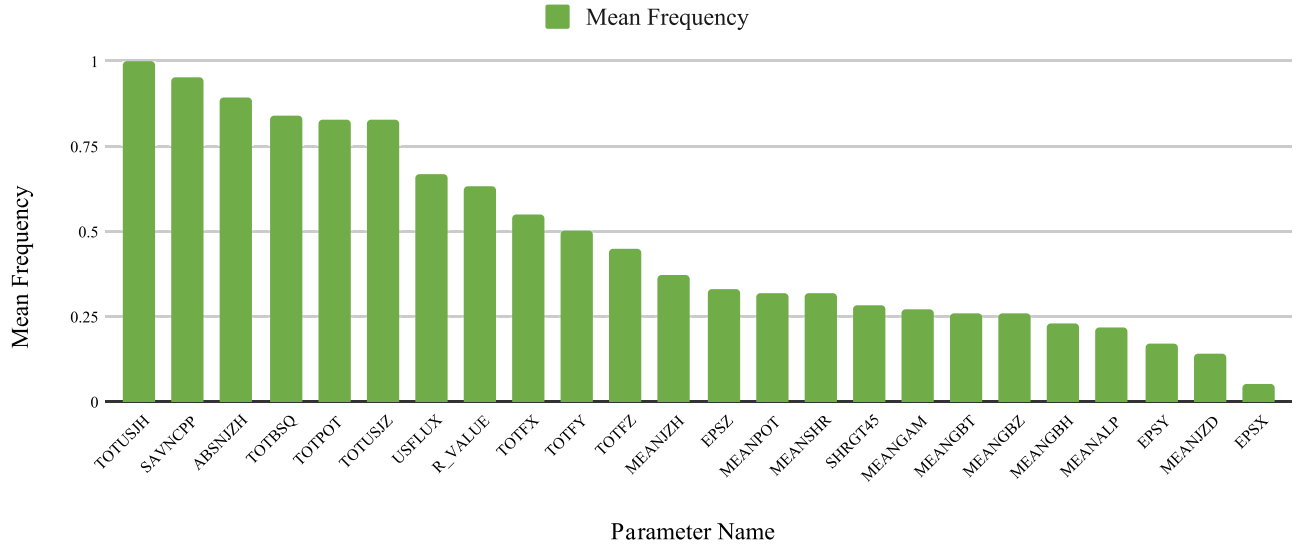

Figure 9. Magnetic field parameters: mean frequency.

Table 2
Parameter Importance Ranking Based on the Experiments of Five Downstream Vector Classifiers

Parameter Name	SVM Frequency	*k*NN Frequency	DT Frequency	RF Frequency	XGBoost Frequency	Majority Vote
TOTUSJH	1	1	1	1	1	1
SAVNCPP	1	0.85	0.95	0.95	1	0.95
ABSNJZH	1	0.65	0.9	0.95	0.95	0.89
TOTBSQ	0.95	0.6	0.85	0.85	0.95	0.84
TOTPOT	0.85	0.65	0.8	0.85	1	0.83
TOTUSJZ	0.95	0.55	0.8	0.9	0.95	0.83
USFLUX	0.8	0.25	0.65	0.75	0.9	0.67
R_VALUE	0.6	0.35	0.55	0.8	0.85	0.63
TOTFX	0.6	0.15	0.5	0.75	0.75	0.55
TOTFY	0.45	0.15	0.45	0.75	0.7	0.5
TOTFZ	0.35	0.15	0.35	0.75	0.65	0.45
MEANJZH	0.2	0.15	0.2	0.75	0.55	0.37
EPSZ	0.05	0.1	0.25	0.65	0.6	0.33
MEANPOT	0.1	0.1	0.25	0.55	0.6	0.32
MEANSHR	0.05	0.1	0.2	0.65	0.6	0.32
SHRGT45	0.05	0.1	0.2	0.6	0.45	0.28
MEANGAM	0.05	0.1	0.15	0.6	0.45	0.27
MEANGBT	0.05	0.1	0.15	0.55	0.45	0.26
MEANGBZ	0.05	0.1	0.15	0.55	0.45	0.26
MEANGBH	0.05	0.1	0.1	0.55	0.35	0.23
MEANALP	0.05	0	0.15	0.45	0.45	0.22
EPSY	0	0	0.5	0.15	0.2	0.17
MEANJZD	0	0	0.1	0.35	0.25	0.14
EPSX	0	0	0.05	0.05	0.15	0.05

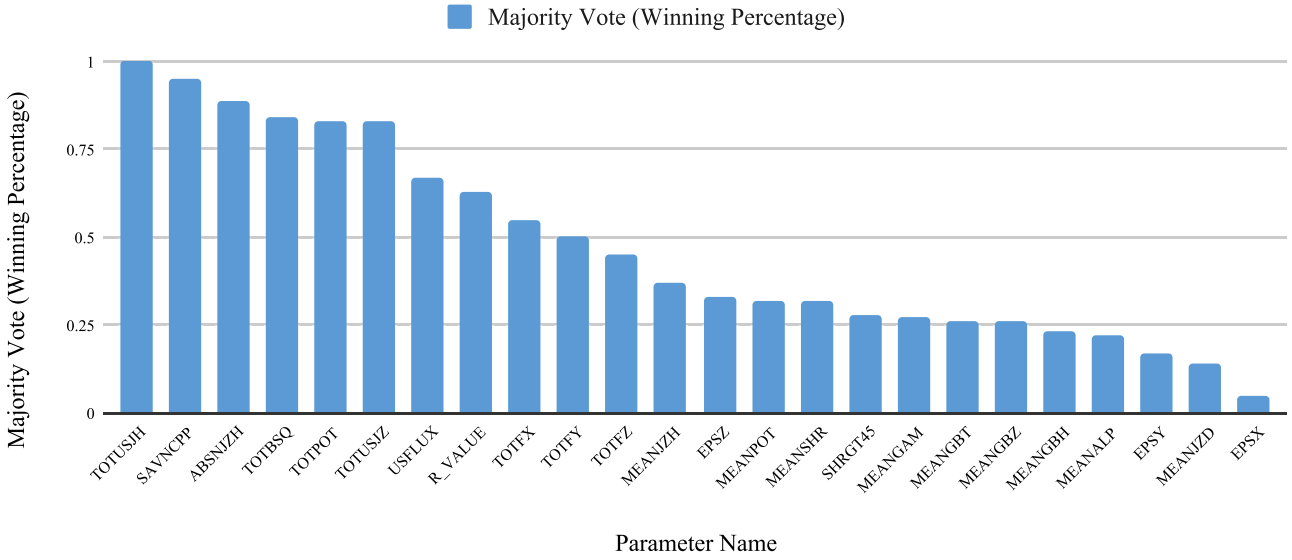

Figure 10. Magnetic field parameters: majority vote (winning percentage).

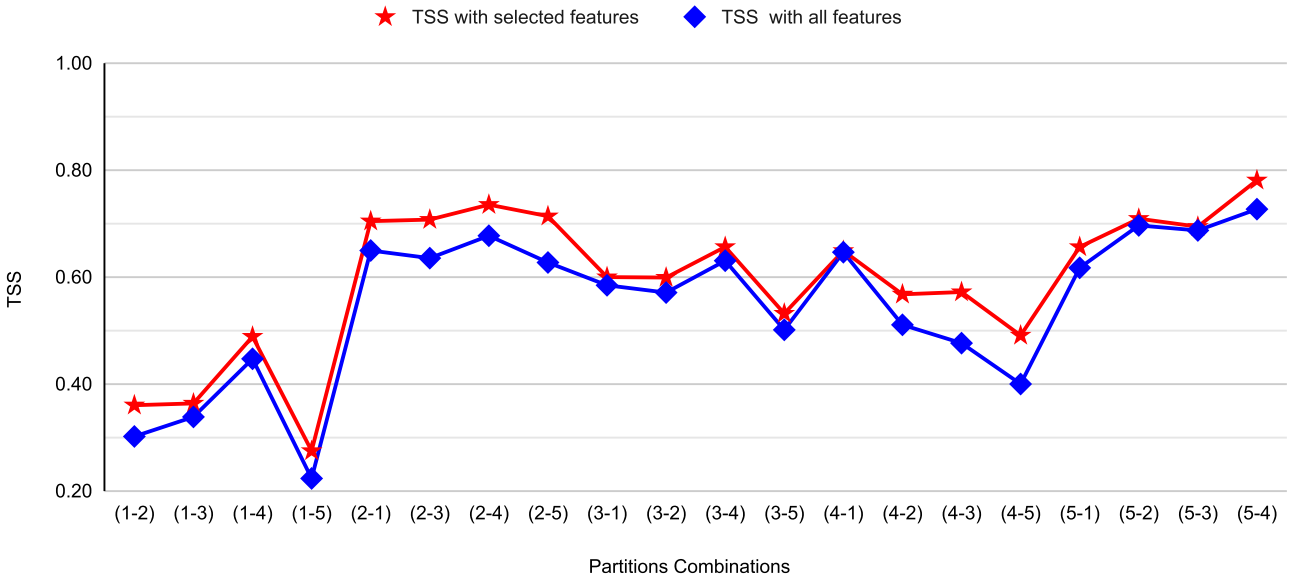

Figure 11. LSTM is used as the downstream classifier. Shown is the test TSS score when the selected number of features is used compared to when all features are used.

against overfitting because it uses a fixed set of kernels and deterministic parameter initialization. This makes MINIROCKET a good choice for large data sets and near-real-time applications. In our experiment, we use MINIROCKET as a downstream classifier to classify the MVTS instances before and after applying our LSTM feature selection algorithm. The results of this experiment as shown in Figure 12 show the performance gains after selecting features.

3. *Transformer*. model as the downstream MVTS classifier. Transformer models for sequence classification have gained prominence for their ability to capture long-range dependencies and contextual information within input sequences effectively (Vaswani et al. 2017; Alshammari et al. 2023). Unlike traditional recurrent architectures, transformers leverage attention mechanisms to process the input sequences in parallel, making them particularly well suited for tasks such as natural language processing and time-series analysis. The results in Figure 13 demonstrate the effectiveness of the selected features in improving the prediction performance of transformer model.

5.4.2. Comparison of LSTM Classifier Performance with a Baseline Flare Prediction Model

Ahmadzadeh et al. (2021) trained and tested an SVM classifier with vector-represented MVTS instances over all possible SWAN-SF partition pairs. The performance of their model is compared to our model (LSTM-based feature selection followed by LSTM classification), as shown in Figure 14. Both the SVM and LSTM models are trained without any special data preprocessing (e.g., undersampling or oversampling). Figure 14 shows that the LSTM classifier significantly outperforms the baseline.

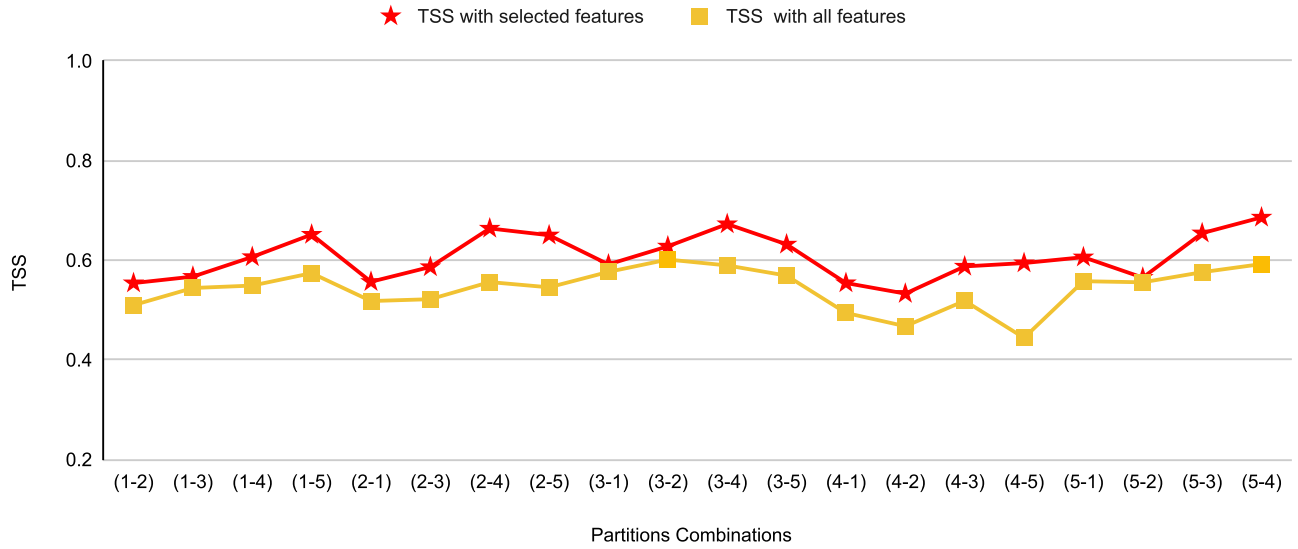

Figure 12. MINIROCKET is used as the downstream classifier. Shown is the test TSS score when the selected number of features is used compared to when all features are used.

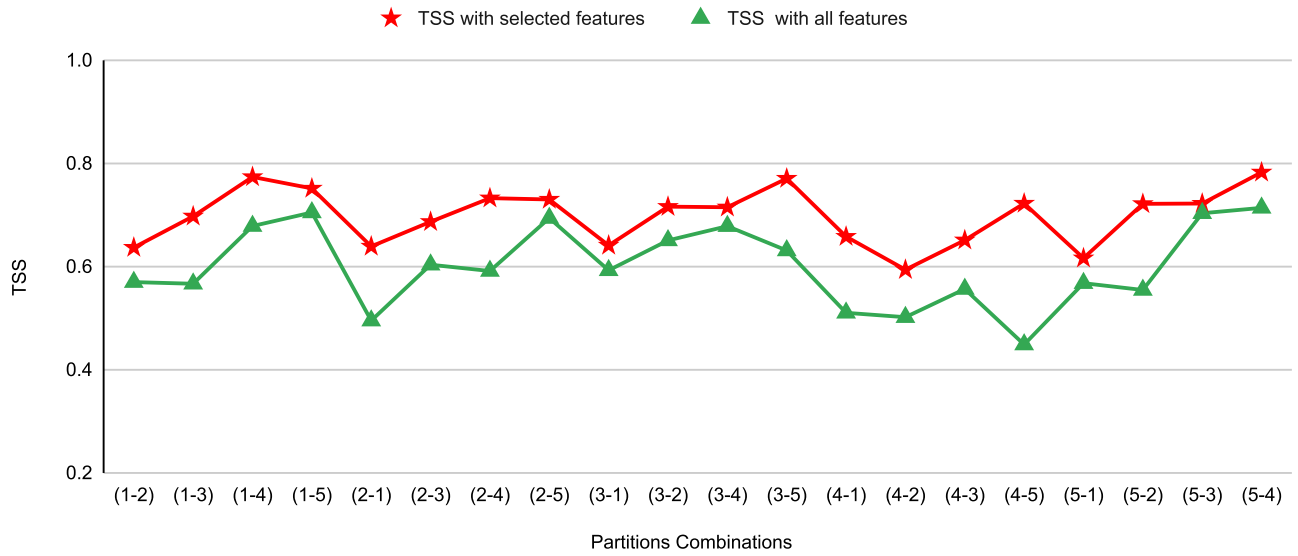

Figure 13. Transformer model is used as the downstream classifier. Shown is the test TSS score when the selected number of features is used compared to when all features are used.

5.5. Extraction of Classifier-wise Discriminative Feature Sets

Some major outcomes of this study are presented in Tables 2–4, which quantify the importance of individual magnetic field parameters in solar flare prediction. These tables provide insights into the selection of parameters based on different downstream classifiers. By referring to Table 2, researchers can gain a practical understanding of these parameters in the context of various classifiers. For this, they can use a predefined threshold for the frequency or majority voting. Suppose, the frequency threshold is 0.5; this means that when utilizing the SVM classifier, the top nine parameters are identified as the most effective magnetic field parameters (sum of the modulus of the net current per polarity, sum of the flux near the polarity inversion line, absolute value of the net current helicity, total unsigned current helicity, total magnitude of the Lorentz force, total photospheric magnetic free energy density, total unsigned vertical current, total unsigned flux, and sum of the x-component of the Lorentz force). Similarly, when employing the kNN classifier, the top six parameters are considered optimal (sum of the modulus of the net current per polarity, absolute value of the net current helicity, total unsigned current helicity, total magnitude of the Lorentz force, total photospheric magnetic free energy density, and total unsigned vertical current). The decision tree classifier indicates the top nine parameters as the most significant (sum of the modulus of the net current per polarity, absolute value of the net current helicity, total unsigned current helicity, total magnitude of the Lorentz force, total photospheric magnetic free energy density, total unsigned vertical current, total unsigned flux, sum of the x-component of the Lorentz force, and sum of the z-component of the Lorentz force). It is important to note that different thresholds can be applied to select the best-performing parameters for a specific downstream classifier. Researchers addressing solar flare prediction from MVTS data can refer to this table to gain insights into the 24 photospheric magnetic field parameters and their relevance

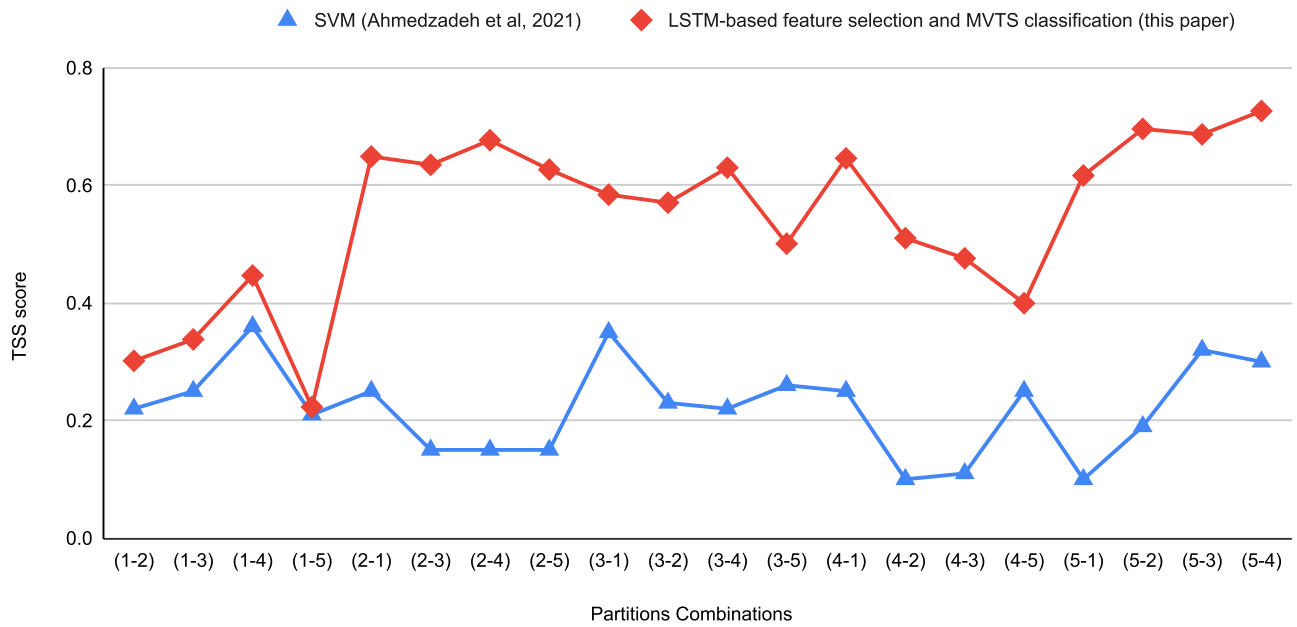

Figure 14. Test TSS score comparison between our LSTM classifier model and the baseline results of Ahmadzadeh et al. (2021).

Table 3
Classifier-wise Best TSS and Selected Features

Parameter	SVM	*k*NN	DT	RF	XGBoost
TOTUSJH	✓	✓	✓	✓	✓
SAVNCPP	✓	✓	✓	✓	✓
ABSNJZH	✓	✓	✓	✓	✓
TOTBSQ	✓	✓	✓		✓
TOTPOT	✓	✓	✓		✓
TOTUSJZ	✓	✓	✓	✓	✓
USFLUX		✓	✓		✓
R_VALUE		✓			✓
TOTFX			✓		
TOTFY			✓		
TOTFZ			✓		
Other 13 parameters	X	X	X	X	X
Best test TSS after 20 training/testing partition experiments	0.72	0.70	0.74	0.71	0.74
Training/testing partitions	P5/P3	P1/P4	P5/P4	P2/P5	P1/P4
Confusion matrix [TP, FP, FN, TN]	[1007, 2465, 270, 34,070]	[659, 1578, 231, 41,117]	[695, 1806, 195, 40,889]	[659, 1636, 235, 63,973]	[696, 1790, 194, 40,905]

Table 4
Multivariate Time Series Classifier-wise Best TSS and Selected Features

Parameter	LSTM	MINIROCKET	Transformer
TOTUSJH	✓	✓	✓
SAVNCPP	✓		✓
ABSNJZH	✓	✓	✓
TOTBSQ	✓		✓
TOTPOT	✓		✓
TOTUSJZ	✓	✓	✓
USFLUX	✓		✓
R_VALUE	✓	✓	✓
TOTFX			
TOTFY			
TOTFZ			✓
Other 13 parameters	X	X	X
Best test TSS after 20 training/testing partition experiments	0.73	0.65	0.77
Training/testing partitions	P2/P4	P1/P5	P1/P4
Confusion matrix [TP, FP, FN, TN]	[689, 1806, 201, 40,889]	[597, 1066, 297, 64,543]	[725, 1806, 165, 40,889]

within the context of the downstream classifiers used in this study.

Tables 3 and 4 show the parameters that contribute to each downstream classifier's highest test TSS score, the corresponding (training + validation)/testing partitions, and the four elements of the confusion matrix (true positive (TP), false positive (FP), false negative (FN), and true negative (TN)) for each classifier. We highlight the experiments where we train and validate on events (partitions) that happened before the test period such as P1/P4 and P2/P5 in Table 3 and P2/P4, P1/P5, and P1/P4 in Table 4. A key insight from Tables 3 and 4 is the

small number of false negatives (FN), which in the context of flare prediction, indicates a smaller number of misclassifications of true flare events as nonflare events.

6. Conclusion

In this work, we present a two-step deep-learning-based framework for feature selection from MVTS data and apply the method to a benchmark solar flare prediction data set for finding discriminatory magnetic field parameters. In the first step, the feature importance score of each individual parameter is approximated by the application of an LSTM-based univariate sequence classifier. Finally, the best feature set that can jointly discriminate examples is extracted by applying an LSTM-based multivariate sequence classifier. The discrimination ability of the selected feature set is assessed by testing with multiple downstream classifiers to get the set of most important parameters. For the solar flare prediction data set, our two-step LSTM-based feature selection model followed by multiple downstream classifiers outperforms the baseline feature selection approaches by giving us the most important magnetic field parameter set.

In the future, we plan to use attention and transformer mechanisms to get better performance for solar flare classification problems. Also, we aim to work on graph-based feature selection from MVTS data. MVTS data represented by graphs, aka networks, can encode higher-order relationships between the variables (Li et al. 2021; Hamdi et al. 2022). We are interested in the application of graph neural networks to MVTS graphs to extract graph-level discriminatory feature sets represented by subgraphs, paths, cycles, and so on (Hamdi & Angryk 2019). Graph-based discriminatory features can help us understand the higher-order relationships of magnetic field parameters in extreme solar events.

Acknowledgments

This project has been supported in part by funding from the Division of Atmospheric and Geospace Sciences within the Directorate for Geosciences, under NSF awards #2301397, #2204363, and #2240022, and by funding from the Office of Advanced Cyberinfrastructure within the Directorate for Computer and Information Science and Engineering, under NSF award #2305781. The authors also acknowledge all those involved with the GOES missions as well as the SDO mission.

ORCID iDs

Khaznah Alshammari https://orcid.org/0009-0005-4435-9642
Shah Muhammad Hamdi https://orcid.org/0000-0002-9303-7835
Soukaina Filali Boubrahimi https://orcid.org/0000-0001-5693-6383

References

Ahmadzadeh, A., Aydin, B., Georgoulis, M. K., et al. 2021, ApJS, 254, 23
Alshammari, K., Hamdi, S. M., & Boubrahimi, S. F. 2022a, in 2022 IEEE Int. Conf. on Big Data (Big Data), ed. S. Tsumoto et al. (Piscataway, NJ: IEEE), 4796
Alshammari, K., Hamdi, S. M., Muzaheed, A. A. M., & Boubrahimi, S. F. 2022b, CIKM Workshop for Applied Machine Learning Methods for Time Series Forecasting (AMLTS'22) https://amlts.github.io/amlts2022/submissions/AMLTS22_paper_2269.pdf
Alshammari, K., Saini, K., Hamdi, S. M., & Boubrahimi, S. F. 2023, in 22nd IEEE Int. Conf. on Machine Learning and Applications (ICMLA), ed. M. Arif Wani (Piscataway, NJ: IEEE), 558
Angryk, R. A., Martens, P. C., Aydin, B., et al. 2020, NatSD, 7, 227
Bahdanau, D., Cho, K., & Bengio, Y. 2014, arXiv:1409.0473
Bahri, O., Boubrahimi, S. F., & Hamdi, S. M. 2022, arXiv:2208.10462
Batina, L., Gierlichs, B., Prouff, E., et al. 2011, J. Cryptol., 24, 269
Bergstra, J., & Bengio, Y. 2012, JMLR, 13, 281
Bloomfield, D. S., Higgins, P. A., McAteer, R. T. J., & Gallagher, P. T. 2012, ApJL, 747, L41
Bobra, M. G., & Couvidat, S. 2015, ApJ, 798, 135
Boubrahimi, S. F., Aydin, B., Kempton, D., & Angryk, R. 2016, in 2016 IEEE Int. Conf. on Big Data (Big Data), ed. J. Joshi et al. (Piscataway, NJ: IEEE), 3149
Boubrahimi, S. F., Hamdi, S. M., Ma, R., & Angryk, R. 2020, in 2020 IEEE Int. Conf. on Big Data (Big Data), ed. X. Wu et al. (Piscataway, NJ: IEEE), 493
Cascalheira, C. J., Hamdi, S. M., Scheer, J. R., et al. 2022, in Proc. 16th Int. AAAI Conf. on Web and Social Media (ICWSM), ed. C. Budak, M. Cha, & D. Quercia (Palo Alto, CA: AAAI), 1373
Chen, T., & Guestrin, C. 2016, in Proc. 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, ed. B. Krishnapuram (New York: ACM), 785
Chung, J., Gülçehre, Ç., Cho, K., & Bengio, Y. 2014, arXiv:1412.3555
Cortes, C., & Vapnik, V. 1995, Mach. Learn., 20, 273
Dempster, A., Schmidt, D. F., & Webb, G. I. 2021, in Proc. 27th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining, ed. F, Zhu et al. (New York: ACM), 248
Eastwood, J. P., Biffis, E., Hapgood, M. A., et al. 2017, RiskA, 37, 206
Filali Boubrahimi, S., & Hamdi, S. M. 2022, in Proc. 31st ACM Int. Conf. on Information & Knowledge Management, ed. M. Al Hasan & L. Xiong (New York: ACM), 3943
Kamalov, F., Thabtah, F., & Hon Leung, H. 2022, Ann. Data Sci., 10, 1527
Fisher, G. H., Bercik, D. J., Welsch, B. T., & Hudson, H. S. 2012, SoPh, 277, 59
Fix, E., & Hodges, J., Jr. 1951, Discriminatory Analysis, Nonparametric Discrimination: Consistency Properties, Technical Report 21-49-004, US Air Force, School of Aviation Medicine, https://apps.dtic.mil/sti/citations/ADA800276
Gu, Q., Li, Z., & Han, J. 2012, arXiv:1202.3725
Guyon, I., & Elisseeff, A. 2003, JMLR, 3, 1157
Hamdi, S. M., Ahmad, A. F., & Boubrahimi, S. F. 2022, CIKM Workshop for Applied Machine Learning Methods for Time Series Forecasting (AMLTS'22) https://amlts.github.io/amlts2022/submissions/AMLTS22_paper_7855.pdf
Hamdi, S. M., & Angryk, R. 2019, in 2019 IEEE Int. Conf. on Data Mining (ICDM), ed. J. Wang, K. Shim, & X. Wu (Piscataway, NJ: IEEE), 270
Hamdi, S. M., Kempton, D., Ma, R., Boubrahimi, S. F., & Angryk, R. A. 2017, in 2017 IEEE Int. Conf. on Big Data (Big Data), ed. J.-Y. Nie et al. (Piscataway, NJ: IEEE), 2543
Hamdi, S. M., Wu, Y., Angryk, R., Krishnamurthy, L. C., & Morris, R. 2019, Int. J. Semantic Comput., 13, 25
Hochreiter, S., & Schmidhuber, J. 1997, Neural Comput., 9, 1735
Hosseinzadeh, P., Nassar, A., Boubrahimi, S. F., & Hamdi, S. M. 2023, Hydro, 10, 29
Johnson, R., Boubrahimi, S. F., Bahri, O., & Hamdi, S. M. 2022, in Pattern Recognition, Computer Vision, and Image Processing. ICPR 2022 Int. Workshops and Challenges, ed. J. J. Rousseau & B. Kapralos (Cham: Springer), 273
Kazachenko, M. D. 2023, ApJ, 958, 104
Leka, K., & Barnes, G. 2003, ApJ, 595, 1296
Leka, K., & Skumanich, A. 1999, SoPh, 188, 3
Li, P., Bahri, O., Boubrahimi, S. F., & Hamdi, S. M. 2022, in 21st IEEE Int. Conf. on Machine Learning and Applications (ICMLA), ed. M. Arif Wani et al. (Piscataway, NJ: IEEE), 1238
Li, P., Boubrahimi, S. F., & Hamdi, S. M. 2021, in 2021 IEEE Int. Conf. on Big Data (Big Data), ed. Y. Chen et al. (Piscataway, NJ: IEEE), 4464
Ma, R., Boubrahimi, S. F., Hamdi, S. M., & Angryk, R. A. 2017, in 2017 IEEE Int. Conf. on Big Data (Big Data), ed. J. Y. Nie et al. (Piscataway, NJ: IEEE), 2569
Mason, J. P., & Hoeksema, J. 2010, ApJ, 723, 634
Mikolov, T., Karafiát, M., Burget, L., Černocký, J., & Khudanpur, S. 2010, Proc. Interspeech 2010, ed. K. Hirose, 1045
Muzaheed, A. A. M., Hamdi, S. M., & Boubrahimi, S. F. 2021, in 20th IEEE Int. Conf. on Machine Learning and Applications (ICMLA), ed. M. Arif Wani et al. (Piscataway, NJ: IEEE), 435
Oruh, J., Viriri, S., & Adegun, A. 2022, IEEEA, 10, 30069

Pal, M. 2005, IJRS, 26, 217
Peng, H., Long, F., & Ding, C. 2005, ITPAM, 27, 1226
Rotti, S. A., Martens, P. C., & Aydin, B. 2020, ApJS, 249, 20
Sadykov, V. M., Kosovichev, A. G., Oria, V., & Nita, G. M. 2017, ApJS, 231, 6
Safavian, S. R., & Landgrebe, D. 1991, ITSMC, 21, 660
Schrijver, C. J. 2007, ApJL, 655, L117
Srivastava, A., & S, A. 2022, in 7th Int. Conf. for Convergence in Technology (I2CT) (Piscataway, NJ: IEEE)
Tafazoli, S., & Keogh, E. 2023, in Proc. 2023 SIAM Int. Conf. on Data Mining (SDM), ed. S. Shekhar et al. (Philadelphia, PA: SIAM), 685
Vaswani, A., Shazeer, N., Parmar, N., et al. 2017, Advances in Neural Information Processing Systems 30, ed. I. Guyon et al. (NeurIPS), https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html
Wang, C., Yang, H., Bartz, C., & Meinel, C. 2016, arXiv:1604.00790
Wang, J., Shi, Z., Wang, H., & Lue, Y. 1996, ApJ, 456, 861
Yu, Y., Si, X., Hu, C., & Zhang, J. 2019, Neural Comput., 31, 1235