The Astrophysical Journal Supplement Series, 270:15 (18pp), 2024 January https://doi.org/10.3847/1538-4365/ad0a6c

Predicting Solar Proton Events of Solar Cycles 22–24 Using GOES Proton and Soft-X-Ray Flux Features

Aatiya Ali[1], Viacheslav Sadykov[1], Alexander Kosovichev[2,3], Irina N. Kitiashvili[3], Vincent Oria[4], Gelu M. Nita[2], Egor Illarionov[5,6], Patrick M. O'Keefe[4], Fraila Francis[4], Chun-Jie Chong[4], Paul Kosovich[2], and Russell D. Marroquin[7]
[1] Physics & Astronomy Department, Georgia State University, Atlanta, GA 30303, USA; aali87@student.gsu.edu
[2] Physics Department, New Jersey Institute of Technology, Newark, NJ 07102, USA
[3] NASA Ames Research Center, Moffett Field, CA 94035, USA
[4] Computer Science Department, New Jersey Institute of Technology, Newark, NJ 07102, USA
[5] Department of Mechanics and Mathematics, Moscow State University, Moscow, 119991, Russia
[6] Moscow Center of Fundamental and Applied Mathematics, Moscow, 119234, Russia
[7] Department of Physics, University of California San Diego, La Jolla, CA 92093, USA
Received 2023 March 9; revised 2023 October 25; accepted 2023 November 4; published 2024 January 11

Abstract

Solar energetic particle (SEP) events and their major subclass, solar proton events (SPEs), can have unfavorable consequences on numerous aspects of life and technology, making them one of the most harmful effects of solar activity. Garnering knowledge preceding such events by studying operational data flows is essential for their forecasting. Considering only solar cycle (SC) 24 in our previous study, we found that it may be sufficient to only utilize proton and soft X-ray (SXR) parameters for SPE forecasts. Here, we report a catalog recording $\geqslant$10 MeV $\geqslant$10 particle flux unit SPEs with their properties, spanning SCs 22–24, using NOAA's Geostationary Operational Environmental Satellite flux data. We report an additional catalog of daily proton and SXR flux statistics for this period, employing it to test the application of machine learning (ML) on the prediction of SPEs using a support vector machine (SVM) and extreme gradient boosting (XGBoost). We explore the effects of training models with data from one *and* two SCs, evaluating how transferable a model might be across different time periods. XGBoost proved to be more accurate than SVMs for almost every test considered, while also outperforming operational SWPC NOAA predictions and a persistence forecast. Interestingly, training done with SC 24 produces weaker true skill statistic and Heidke skill scores_2, even when paired with SC 22 or SC 23, indicating transferability issues. This work contributes toward validating forecasts using long-spanning data—an understudied area in SEP research that should be considered to verify the cross cycle robustness of ML-driven forecasts.

Unified Astronomy Thesaurus concepts: Solar energetic particles (1491); Space weather (2037); Solar cycle (1487)

1. Introduction

Solar energetic particle (SEP) events are enhanced fluxes of high-energy particles ejected by the Sun. The occurrence rates of such events are greatest closer to the maxima of ~11 yr solar cycles (SCs). These events encompass a wide range of energies from keVs up to multiple GeVs (Anastasiadis et al. 2019), ejected into the heliosphere. Solar proton events (SPEs), a subclass of SEPs, are characterized as protons with energies $\geqslant$10 MeV exceeding a threshold of $\geqslant$10 particle flux units (pfus). Energetic protons can harm satellites, navigation and communication systems, technological grids, and other equipment. High-energy charged particles in the magnetosphere can manipulate the output signals of electronic devices. This causes spacecraft calibration systems to malfunction, and when these charged particles strike a critical device, the instrument may fail entirely. Other examples of solar transient activity and its consequences include high-energy electrons that further complicate operations as they can penetrate shielding aboard satellites and spacecraft. They quickly pileup, and eventually discharge the accumulated energy mirroring a *lightning strike* (Bollavaram & Asmatulu 2016). Radio-wave-dependent communication systems are also vulnerable to such events. Extreme ultraviolet (EUV) and X-ray radiation from the Sun can ionize the Earth's ionosphere, elevating electron density in the medium radio waves travel through. This delays transmission time from satellites to ground-based global positioning systems, ultimately causing the misalignment of positions by a few meters. While seemingly insignificant, this poses issues for aviation, robotics, military, transportation, and other industries' operations.[8] SEPs and cosmic rays are similarly capable of ionizing and altering the ionosphere.

The limited understanding of solar processes leading to the generation of SPEs motivates research in heliophysics and astrobiology to advance. By enhancing the radiation levels in interplanetary space,[9] SPEs may be responsible for the atrophy of astronaut health. Onorato et al. (2020) expand on this, highlighting the risks of astronauts developing cancer, experiencing central nervous system decrements, and even exhibiting degenerative tissue effects. More recently, such concerns have caught the attention of administrations working with commercial airlines and space tourism. Beck et al. (2005) conclude that the combined effects of magnetic field disturbances and solar particle fluence due to solar storms can be responsible for up to ~70% variation in radiation exposure at

[8] http://www.space.com/solar-storms-destroy-satellites
[9] https://srag.jsc.nasa.gov/spaceradiation/what/what.cfm

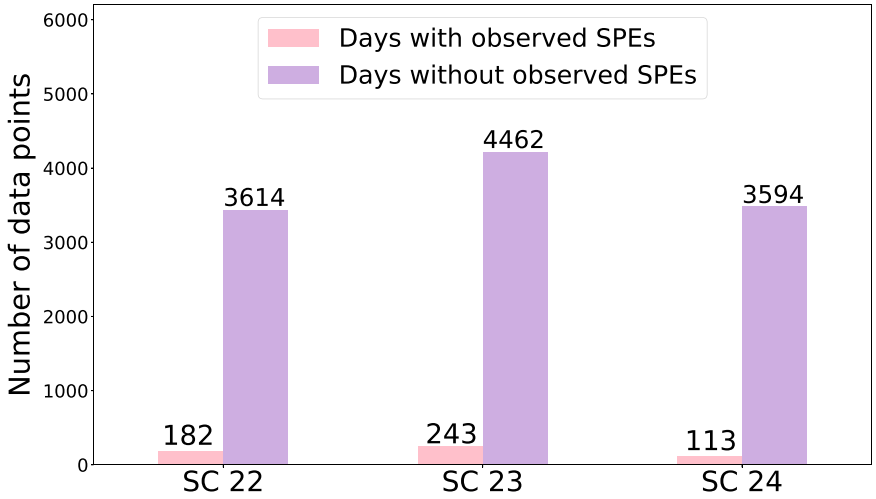


Figure 1. The drastic imbalance in the number of days where SPEs are observed per SC, compared to days they are not observed. *Note that SC 22 is dated from* 1986 September to 1996 August, SC 23 from 1996 August to 2008 December, and SC 24 from 2008 December to 2019 December.

typical flight altitudes. Collins (2006) also notes higher exposure to cosmic radiation as the main safety concern during space tourism between lunar and orbital travel. This would only be enhanced during the propagation of SEPs. Further, Naito et al. (2020) conclude that some composite materials (i.e., carbon fiber-reinforced plastic and silicon-carbon plastic, Bollavaram & Asmatulu 2016) show promise in shielding spacecraft from enhanced radiation. Still, none have been confirmed to withstand the expected radiative variations during SPEs. Given the many hazards presented by these events, it is critical to develop reliable forecasts to provide sufficient time for astronauts and equipment to be safely relocated. External SEP detectors coupled with an appropriate predictive algorithm may be mounted onto spacecraft to act as warning and defense systems in this effort. Our overarching goal is to investigate the capabilities of different machine-learning (ML) models with respect to the prediction of SPEs. We employ our models with only proton and soft X-ray (SXR) flux data, and assess performance variations when considering timescales longer than a single SC during the training phase.

1.1. The Problem of SPE Prediction

Approaches to building predictive models for SPEs become arduous given their rare nature and reliance on indirect measures of volatile SC activity. If using measures like forecasting accuracy to quantify the success of a predictive model, the rarity of SPEs permits an algorithm to be able to miss most, if not all, incoming SPEs while sustaining high accuracy in prediction scores (as we see in Section 3.3). The need for balanced data sets and assessment metrics to reflect accurate scores is emphasized by Martens & Angryk (2017). Specifically, the class imbalance in our data set shows 11,946 negative cases (days with no SPEs), in vast contrast to only 538 positive cases (days with SPEs). This imbalance overshadows positive cases by the prevalence of quiet (days with no observed SPEs) periods, and is the overarching problem encountered in this work, and the disparity is illustrated in Figure 1. Variance in the Sun's global magnetic activity and consequential changing event frequency brings forward the question of how transferable an algorithm built using previous SC data may be when considering future SCs of unknown activity levels.

1.2. Current and Previous Prediction Efforts and Limitations

In recent years, there has been a plethora of research projects contributing to the effort of predicting SPEs in an attempt to mitigate their detrimental effects. In our previous study (Sadykov et al. 2021) we used SXR wavelength ranges (long (0.1–0.8 nm) and short (0.05–0.4 nm)), along with $\geqslant$10 MeV proton flux data observed by the Geostationary Operational Environmental Satellite (GOES) series. From the various products obtained by GOES, we retrieve and use SXR flux data with a 1 minute cadence and $\geqslant$10 MeV proton flux data with a 5 minute cadence. The data have been made publicly available by the National Oceanic & Atmospheric Association (NOAA) National Center for Environmental Information (NCEI[10]). The promising results of utilizing derivatives of these data products alone motivate us to continue exploring these parameters in-depth in this work. Sadykov et al. (2021) also discuss the lack of performance loss when excluding characteristics of active regions (ARs) and type II, III, and IV radio bursts when generating predictive algorithms, although acknowledging the brevity of the considered data set (SC 24 alone). Predictive scores resulting from using proton flux alone were compared to those with the addition of SXR data, which proved to enhance prediction accuracy. Therefore, in this work we study operational proton and SXR flux features in detail, exploring the potential to develop a reliable SPE-predictive model using these features.

However, in addition to these parameters, different approaches to SPE prediction include physics-based and empirical models that take into account parameters of solar magnetograms, optical imaging, EUV imaging, coronal mass ejections from single or multiple vantage points, in situ energetic proton and electron observations, particle acceleration, and transport, and measurements of solar wind density, temperature, and magnetic fields (Whitman et al. 2023). Whitman et al. (2023) also elaborate on current model validation diagnostics. Most physics-based models (i.e., computing SEP acceleration and transport from first principles) are computationally expensive, limiting their integration into current workflows to make *real-time* predictions. Statistics-based or ML-driven models can capture empirical, yet

[10] www.ncei.noaa.gov/data/goes-space-environment-monitor/access/avg/

nonlinear dependencies between observational data and SPE processes given sufficient data presented for training these models. We supply short-term (data from a single SC) *and* long-term (data from two SCs) fluxes during our models' learning phases, exploring proton, short, and long SXR variations leading to SPEs. We use data acquired using only NOAA's GOES series for consistency.

Studies comparing results between ML-driven models with those used as daily operational forecasts by NOAA's Space Weather Prediction Center (SWPC) for SPEs may identify observed parameters preceding SPEs that could improve such forecasts (Sadykov et al. 2021). To achieve this, we aggregate a catalog of statistical proton, short, and long SXR flux parameters (discussed in Section 2.3), which are minimized by meticulously selecting features most relevant to predicting these SEP events during unpredictable levels of solar activity. In parallel, fundamental parameters required by the model may be poorly characterized without a complete understanding of the underlying solar mechanisms associated with SEP ejection and acceleration. SEP modeling has thus been motivated to explore the physical processes related to SEPs and operational forecasting needs. These current complex models show promise in modeling time-dependent distributions of SEP events. ML approaches are still being investigated to yield a new class of SEP models to produce fast, reliable forecasts (Whitman et al. 2023). Our work here presents a different approach to the problem.

1.3. Scope of Our Work

The presented work assesses ML-driven models for the prediction of SPEs using data from three previous SCs (22–24). From this, we build an understanding of the cross cycle transferability of these pre-established models. Starting with detecting SPEs from these SCs using GOES series data, event parameters are stored in the first catalog built in this effort. Using the continuous flux records we have of these SCs, we form an additional catalog of daily statistical features of SXR and $\geqslant$10 MeV proton fluxes, supplying the input data sets for our learning models.

Our first model, support vector machines (SVMs), has become a standard classifier in space weather prediction studies. Recent works by Bobra & Couvidat (2015), Ahmadzadeh et al. (2021), Kasapis et al. (2022), Asaly et al. (2021), Bobra & Ilonidis (2016), and others, employ these classifiers and acknowledge them being advantageous, robust, and fast when making predictions. Bobra & Couvidat (2015) also show that SVMs are successful classifiers when applied to large data sets. Although developed more recently than SVMs, extreme gradient boosting (XGBoost; our second model) has already gained recognition in capturing complex patterns in the data. Similar to SVMs, it is well suited for analyzing large data sets and is currently used in many areas of research (finance, healthcare, environmental sciences, etc.). Rotti & Martens (2023), Li et al. (2022), Bailey et al. (2021), and McGuire et al. (2019) highlight the success of using XGBoost to analyze SEP events, predict flares, and explore other space weather events. Within these works, XGBoost often outperforms more complex models (random forest, logistic regression, etc.) when regarding different model performance metrics.

We begin our work here by constructing SPE flux data catalogs, one of which is used as the input for our ML models (discussed in Sections 2.2 and 2.3). To combat the imbalance in the positive and negative classes, we use model-inherent class weight balancing techniques, as well as data generation using standard (duplication of positive cases), and synthetic oversampling. We reduce the number of features considered for forecasting using Gini importance, Fisher scoring, and XGBoost, exploring which supplied statistical features are most important when working toward making predictions. Finally, we define data from each SC as different training/testing data sets (discussed in Section 2.5) to use with SVM and XGBoost algorithms. Predictive capabilities are then measured using true skill statistics (TSS), Heidke skill scores (HSS_2, developed by Mason & Hoeksema 2010), and recall metrics. We also apply *k*-fold cross validation (CV) to optimize our models with respect to TSS. We also consider the effects of different training timescales when producing forecasts. Analyzing different parameters associated with SVM and XGBoost models, we strive to generalize our algorithms while retaining the highest scores achievable from our data set. Modifications applied to each model are further discussed in Section 3.

2. GOES Data Preparation and Products

2.1. Querying Flux Data

Our prediction algorithms are built using $\geqslant$10 MeV proton and SXR flux data queried from NOAA's National Center for Environmental Information (NCEI) through 1986–2019 to encompass SCs 22–24. The data are obtained by different GOES launched by NOAA as a series from GOES-05 to GOES-15 during the period of interest. The main objective of GOES is to aid forecasting operations by supplying real-time access to X-ray and proton flux measurements (Aminalragia-Giamini et al. 2021) from the geostationary orbit (an altitude of ~36,000 km above Earth's equator); making this satellite series a clear choice for several SC-long data collection for our work. In 1974, NOAA compiled a primary and secondary scheme[11] identifying GOES data to utilize during instances when, at each time, multiple satellites in the GOES series provided real-time data, assigning one to be the primary and others as secondary. Rotti et al. (2022) also do this in an effort to explore integral proton flux intensity profiles for space weather predictions. Influenced by these routines, we manually choose a primary instrument for every month across SCs 22–24. By our definition, a primary instrument reflects a higher peak proton flux count when two or more satellites or detectors are simultaneously capturing data. Other works prioritize GOES data differently, e.g., Aminalragia-Giamini et al. (2021) retain measurements from the highest-numbered GOES satellite instead.

GOES underwent a change from a mounted Energetic Particle Sensor (EPS) to an Energetic Proton, Electron, and Alpha Detector (EPEAD) in 2011 starting with GOES-13. Equipped with a single detector, the EPS was capable of distinguishing between SEPs and Galactic cosmic rays, measuring fluxes from different energy channels. Upgrading this instrument, EPEAD collects data from two detectors, one surveying east, at 75° W and the other west, at 135° W (with respect to the prime meridian); capturing slightly different populations of protons. He & Rodriguez (2018) further discuss this, concluding that differences in proton flux measurements between the detectors are due to the effect of magnetic field variation with geomagnetic longitude. This highlights that simply taking the average of these detectors'

[11] www.ngdc.noaa.gov/stp/satellite/goes/doc/GOES_XRS_readme.pdf

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1986	G6	G6	G6	G6	G6	G6	G6	G6	G6	G6	G6	G6
1987	G6	G6	G7	G7	G7	G6	G6	G6	G6	G6	G7	G7
1988	G7	G6	G7	G6	G6	G7	G7	G6	G7	G7	G7	G7
1989	G7	G6	G6	G7	G7	G7	G6	G7	G6	G7	G7	G7
1990	G6	G6	G7	G7	G7	G7	G7	G7	G7	G7	G7	G6
1991	G7	G7	G7	G7	G7	G7	G7	G7	G7	G7	G7	G7
1992	G7	G7	G7	G7	G7	G7	G7	G7	G7	G7	G7	G6
1993	G6	G7	G7	G7	G7	G7	G7	G7	G6	G7	G7	G7
1994	G7	G7	G7	G7	G7	G7	G7	G7	G6	G7	G7	G7
1995	G8	G8	G8	G7	G7	G7	G7	G7	G7	G8	G8	G7
1996	G8	G8	G8	G8	G8	G8	G8	G8	G8	G8	G8	G8
1997	G8	G8	G8	G8	G8	G8	G8	G8	G8	G8	G8	G8
1998	G8	G8	G8	G8	G8	G8	G8	G10	G10	G8	G8	G10
1999	G8	G8	G8	G8	G8	G8	G8	G8	G8	G8	G8	G8
2000	G8	G8	G8	G8	G8	G8	G8	G8	G8	G8	G8	G8
2001	G8	G8	G8	G8	G8	G8	G10	G10	G10	G10	G10	G10
2002	G10	G10	G10	G8	G8	G8	G10	G10	G10	G10	G10	G10
2003	G10	G10	G10	G10	G10	G10	G11	G11	G11	G11	G11	G11
2004	G11	G11	G11	G11	G11	G11	G11	G11	G11	G11	G11	G11
2005	G11	G11	G11	G11	G11	G11	G11	G11	G11	G11	G11	G11
2006	G11	G11	G11	G11	G11	G11	G11	G11	G11	G11	G11	G11
2007	G11	G11	G11	G11	G11	G11	G11	G11	G11	G11	G11	G11
2008	G11	G11	G11	G11	G11	G11	G11	G11	G11	G11	G11	G11
2009	G11	G11	G11	G11	G11	G11	G11	G11	G11	G11	G11	G11
2010	G11	G11	G11	G11	G11	G11	G11	G11	G11	G11	G11	G11
2011	G11	G11	G15 E	G15 W	G15 W	G15 W	G15 W	G15 W	G13 E	G13 E	G15 E	G15 E
2012	G15 E	G15 E	G15 E	G13 W	G15 W	G15 W	G15 W	G15 W	G15 W	G13 E	G15 E	G15 E
2013	G15 E	G15 E	G15 E	G13 W	G15 W	G15 W	G15 W	G15 W	G13 W	G15 E	G13 E	G13 E
2014	G13 E	G13 E	G13 E	G15 W	G15 W	G15 W	G15 W	G15 W	G15 W	G15 E	G15 E	G15 E
2015	G13 E	G13 E	G15 E	G15 W	G15 W	G13 W	G13 W	G13 W	G13 W	G15 E	G13 E	G13 E
2016	G15 E	G15 E	G15 E	G15 W	G15 W	G15 W	G15 W	G15 W	G15 W	G15 E	G15 E	G15 E
2017	G15 E	G15 E	G15 E	G15 W	G15 W	G15 W	G13 W	G15 W	G15 W	G15 E	G15 E	G15 E
2018	G15 E	G15 E	G15 E	G15 W	G15 W	G15 W	G15 W	G15 W	G15 W	G15 E	G15 E	G15 E
2019	G15 E	G15 E	G15 E	G15 W	G15 W	G15 W	G15 W	G15 W	G15 W	G15 E	G15 E	G15 E

Figure 2. Timeline of primary GOES instruments and detectors (when applicable) used to streamline SCs 22–24 flux data. Here, *G* represents "GOES," succeeding numbers represent the satellite number of the GOES series, and E or W (when applicable) indicates which of the east or west satellite detectors had detected the higher proton flux.

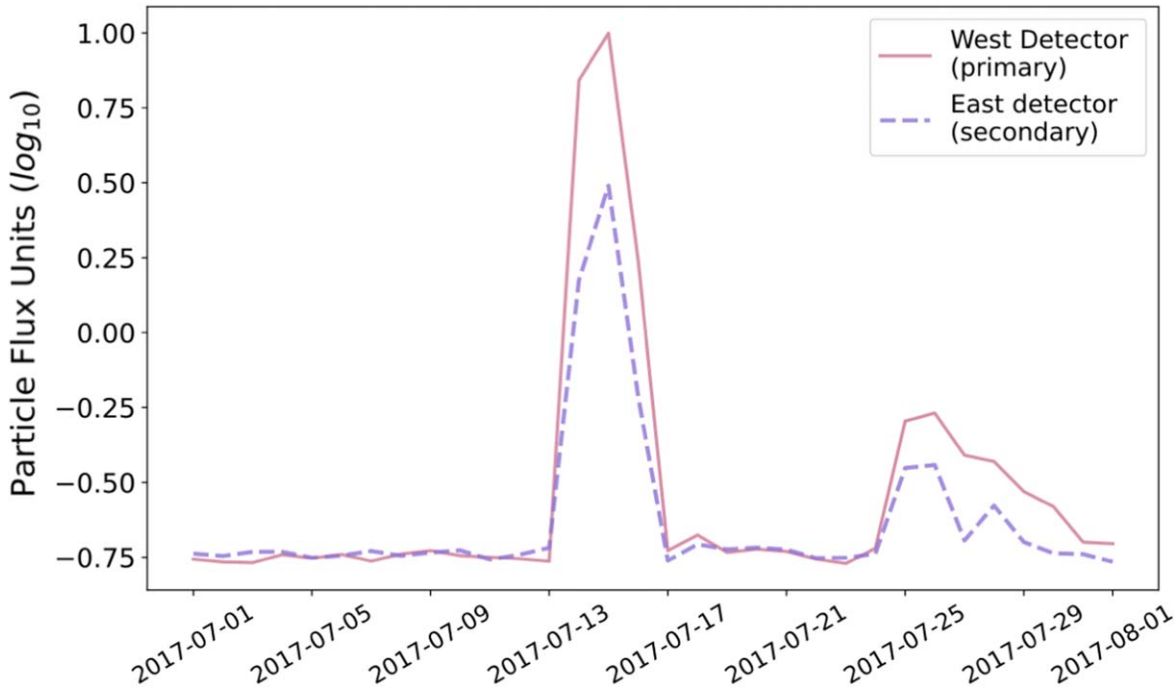


Figure 3. Example of different proton populations detected by east and west detectors of GOES-13 for 2017 July (shown are daily median fluxes). Given the consistently higher fluxes registered by the west detector compared to the east during the same observation period, we record it as the primary instrument/detector for the month.

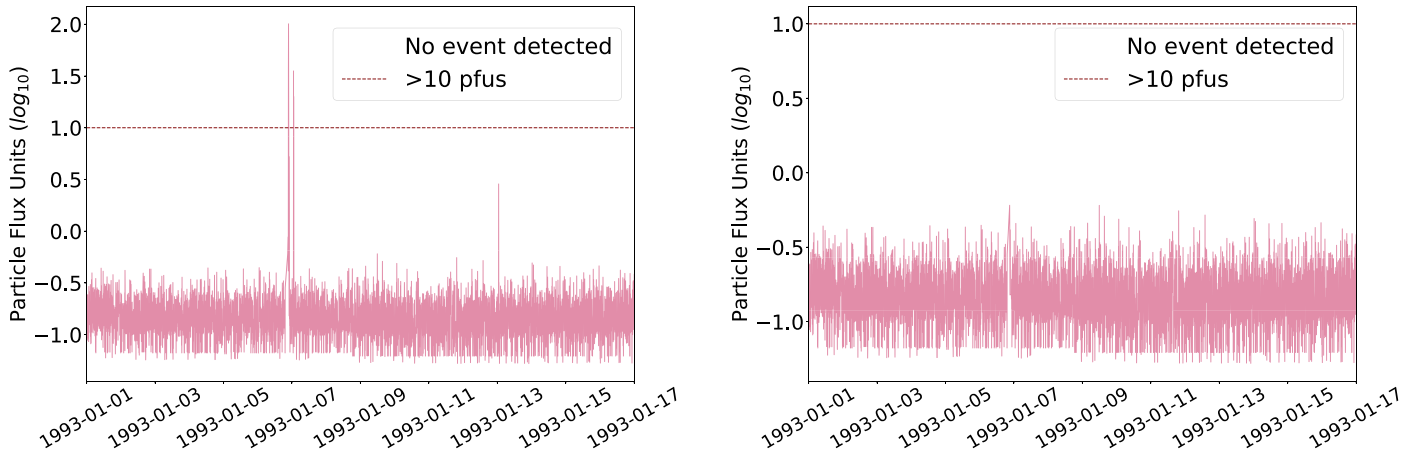


Figure 4. Examples of (left) instrumental effects producing a spike in the data, and (right) how a corrected spike is presented, mirroring a quiet period of the Sun.

fluxes will not accurately represent SEP propagation from the Sun to the magnetosphere. Therefore, when EPEAD satellite data are introduced into our data set, both primary instruments and primary detectors (labeled as east or west) are selected and recorded for use. The primary instrument selection used for SCs 22–24 is shown in Figure 2, and an illustration comparing proton populations captured by the primary versus secondary detectors is presented in Figure 3, reflecting the minimal—but still distinct differences in proton fluxes recorded by each detector.

It is also important to note changes in GOES directionality after launch for those interested in recording instruments/detectors with no data gaps. As specified by Rodriguez[12] and the National Geophysical Data Center, GOES-13–15 all undergo a yaw flip where the satellite rotates about its axis pointed toward the center of Earth, flipping detector orientations. During these flips, EPEAD telemetry channels labeled east are actually looking westward, and those labeled west are looking eastward. The inversions in directionality over time are as follows:

1. GOES-13 (2006–2018): only upright during its operational period between 2010 May and 2012 September.
2. GOES-14 (2009–2020): upright from its launch date in 2009 and inverted during an SPE in early 2012 September. The satellite has not corrected itself since.[13]
3. GOES-15 (2010–2018): experiences a flip twice a year at every equinox. This maneuver usually lasts under an hour, during which data are not recorded.

These inversions are considered for each satellite by their specific guidelines, and orientation labels are corrected for in the catalogs presented in Sections 2.2 and 2.3. During yaw flips and periods when proton and SXR flux data were not recorded (during orientation changes, instrumental corrections, etc.), we interpolate fluxes from the time before and after the data gap to attain continuous flux records for the timeline of interest. Streamlined proton and SXR flux data, our resulting catalogs, and time-series visuals are fetched onto a Solar Energetic Particle Prediction Portal[14] (SEP^3).

[12] https://www.ngdc.noaa.gov/stp/satellite/goes/doc/Note%20on%20GOES%2013-15%20Yaw%20Flip.pdf

[13] Powered off in 2020, this satellite can be called back into service if needed (https://www.nesdis.noaa.gov/news/noaa-readies-goes-15-and-goes-14-orbital-storage).

[14] https://Sun.njit.edu/SEP3/index.html

2.2. Catalog I: SPE Records

Considering the definition of an SPE as $\geqslant$10 pfu detections of protons $\geqslant$10 MeV, this is the threshold used throughout the process of generating our first catalog. NOAA defines the severity of solar storms using an S-scale[15] hierarchy of progressively damaging solar events spanning from S1 (minor) to S5 (extreme). An SPE is the baseline for an S1 event (the weakest-ranking solar storm in this hierarchy), primarily affecting high-frequency radio propagation in the polar regions. Higher scales (S2, S3, S4, and S5) are associated with events of much higher fluences (cumulative pfus detected during an event) (10^2, 10^3, 10^4, and 10^5, respectively), incrementally enhancing both the radiative environment and the damage they cause.

To build our catalog of SPE statistics, GOES data were transformed into a logarithmic form, and cleaned for instrumental effects prior to recording any event parameters—most of which present themselves as spikes in the data, an example of which is shown in the left panel of Figure 4. Specifically, we identify spikes where there is a heightened flux value, but the flux right before and after (5 minutes prior and after a spike, given that GOES proton flux data are provided with a 5 minute cadence) remains an order of magnitude lower. Given the drastic difference in fluxes where we see spikes without a gradual rise and fall, we are confident that they are instrumental effects, and do not reflect true SEP activity. We correct these by replacing the heightened flux value with the averaged data prior to and immediately after the spike is observed, as can be seen in the right panel of Figure 4. This allows for a more accurate reflection of the Sun's quiet period, while preventing the recording of an SPE when the artificially amplified flux value crosses the $\geqslant$10 pfu threshold. Further, given that the threshold can be repeatedly crossed during an event (as can be seen in the left panel of Figure 5), these oscillations are handled by checking if the time between an ongoing event's end and the start of the next is within 30 minutes of each other. Given that oscillations less than half an hour apart are presented in rapid succession and recorded as multiple events over a short period of time, they are instead stitched together as one event in the catalog. These rapid oscillations may represent true proton counts, or, and most likely, are instrumental side effects when detecting fluxes. Opposing these situations, a more prevalent

[15] www.swpc.noaa.gov/noaa-scales-explanation

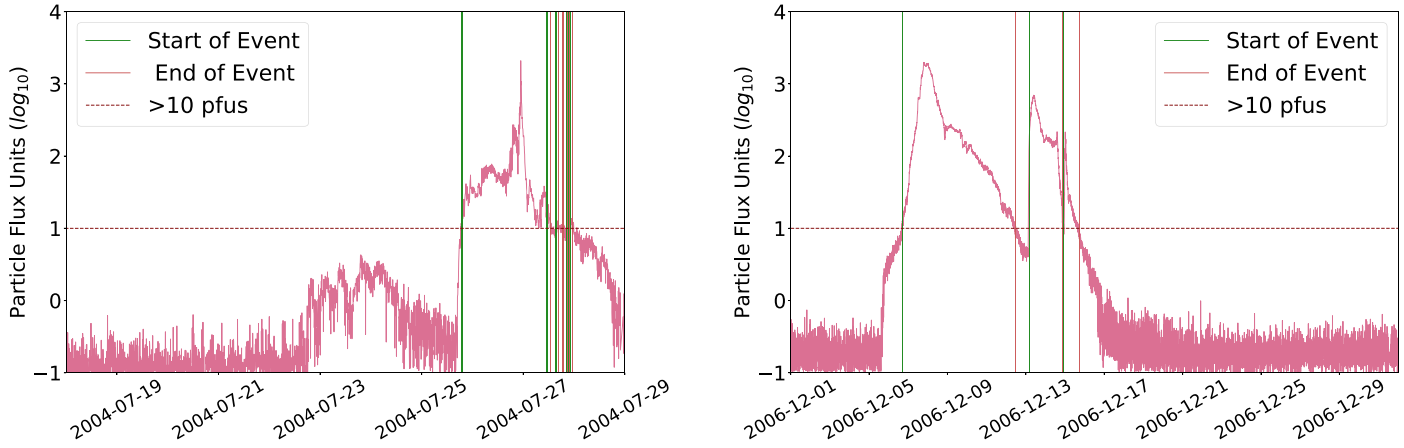


Figure 5. Example of (left) oscillations around the 10 pfu threshold during an event, and (right) an unambiguous representation of an SPE, as they are recorded in our catalog.

Averaged SPE Statistics	SC 22	SC 23	SC 24
Duration (~hh:mm)	11:19 ± 25:22	10:31 ± 23:50	10:47 ± 22:16
Peak Flux (~pfu)	432 ±3147	544 ±3245	156 ± 729
Fluence (~10^4 pfu)	3 ± 19	7 ± 42	3 ± 13
Days with Observed Events	182	243	113
Days with SPE > 24 hours	42	77	32
Previous day's flux ≥10 pfu	5	10	4

Figure 6. SPE statistics considering proton flux data. We summarize (and include the standard deviation when applicable), the average duration, peak flux, and fluence per event for each SC, among other characteristics. We can see here how much weaker SC 24 activity was compared to SCs 22 and 23.

example of an SPE usually has a more gradual rise and/or fall. The end product of this part of data processing is shown in the right panel of Figure 5, reflecting clear start and end dates of an SPE, barring any undesirable flux representations. At this point, our data set is cleaned and we continue with the remainder of the project.

In summary, this catalog includes:

1. date and times for the start of an event, an event's peak flux detection, end of an event
2. peak event fluxes detected in energy channels: 1, 5, 10, 30, 50, 60, and 100 MeV
3. fluence (calculated as the integral of detected fluxes) of SPEs in energy channels: 1, 5, 10, 30, 50, 60, and 100 MeV.

Examples of data products derived from this catalog are shown in Figure 6.

2.3. Catalog II: Daily Flux Feature Statistics

The second catalog produced during this project supplies the input data for our ML-driven forecasting models. This consists of numerous features of daily flux data acquired by GOES in both the proton and SXR channels. Before model training begins, it is common practice to convert input data feature vectors into a standardized range. An SVM's optimal hyperplane—the boundary between distinct classes—is influenced by the scale of input features, requiring data to be scaled prior to model training. The same is needed for XGBoost models, which are sensitive to the scale of features when trained using gradient-based methods.[16] Correspondingly, we apply logarithmic scaling and minima–maxima normalization to our training sets and scale this transformation to the corresponding test sets. This is primarily done to allow models to differentiate between various patterns and structures in the data without being influenced by each parameter's intrinsic physical units and dynamic ranges (Ahmadzadeh et al. 2021). Namely, the flux features generated in this catalog are (for each proton, SXR short wavelength, and SXR long-wavelength channels; features correspond to the current day's flux measurements unless otherwise stated):

1. Added for time-series records only, and not used in the forecast itself: instrumental data, dates of observation, GOES satellite used (with a primary detector when appropriate)
2. Daily aggregated flux data and statistics: mean, median, minimum, maximum, standard deviation, skewness, kurtosis, and the last measured flux of the previous day.

[16] https://forecastegy.com/posts/does-xgboost-need-feature-scaling-or-normalization#:~:text=If%20you%20are%20using%20XGBoost%20with%20linear%20models%20as%20base,can%20lead%20to%20better%20performance

In our previous work (Sadykov et al. 2021), we applied a 2 hr latency period between the time GOES data are released (22:00 UTC daily) and the next day (00:00 UTC). We handled this by applying a 2 hr offset for each day's records, i.e., we +2 hr to every time stamp, so that each day effectively *ends* when new GOES data are released. The new data supplies flux records for *only* the next day. In doing this, any predictions we make for the following day are technically done at 22:00 UTC. We did this to allow 1:1 comparisons between our forecasts and those made by the SWPC NOAA (they also make daily SPE predictions at 22:00 UTC, although using different features as input for probability-based predictions, Bain et al. 2021). We apply this offset and consider previous SWPC prediction probabilities as a baseline to assess our ML-driven model performance in Section 4.1.

2.4. Flux Feature Importance and Selection

As discussed in Bobra & Couvidat (2015) and Sadykov et al. (2021), including all available features in an ML model does not necessarily lead to an increase in predictive scores, and may even result in a notable decrease. In the presence of multiple irrelevant or redundant features, learning methods tend to overfit contributions and become less interpretable or produce entirely inadequate results. A common way to resolve this problem is by implementing feature selection, which works to reduce supplied data dimensionality by only selecting a subset of the input features (which in our case, is the catalog discussed in Section 2.3). We determine each flux feature's *importance* by using Gini importance, Fisher scoring, and XGBoost (uses an inherent feature importance scheme) to rank each feature's contributions toward reliable forecasts, retaining those scoring the highest. This also works to reduce associated computational costs and removes irrelevant features for problems with multi-dimensional data (Gu et al. 2012).

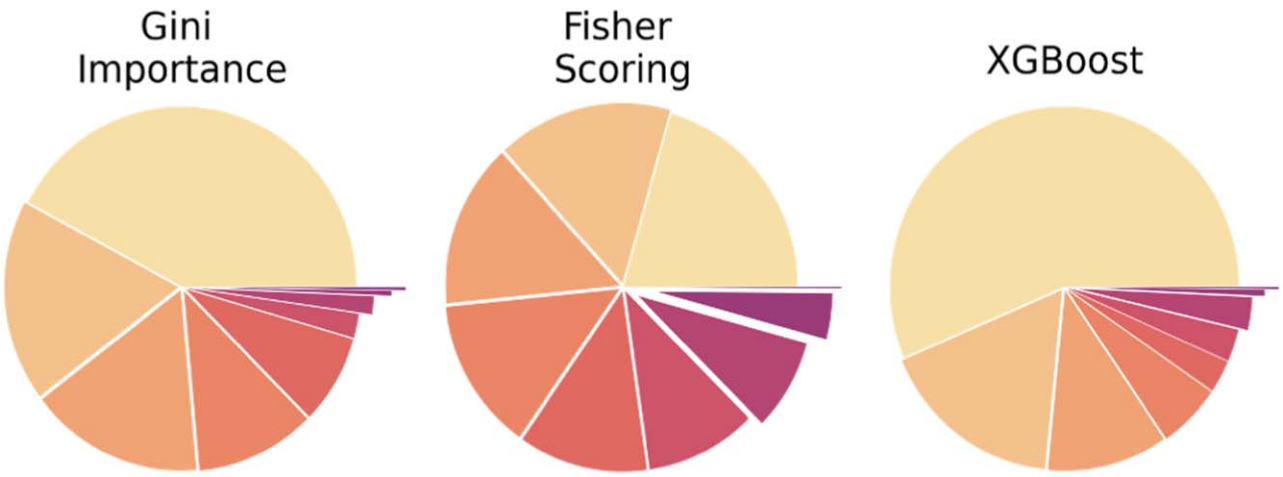
In order, Gini importance is computed using a random forest structure providing relative rankings of input features indicating how often specific features are selected for node splitting (deciding how to divide data into separate classes). In doing so, Gini importance quantifies different input features' contributions toward the improvement or decline of model performance (Menze et al. 2009). Aiming to reduce feature dimensionality like Gini importance, Fisher scoring is one of the most popular supervised univariate feature selection methods and is explained in detail by Gu et al. (2012). Concisely, Fisher scoring measures the intra-class variance between features in both positive and negative classes, identifying those that stand out from neighboring features (i.e., those defining the separation between the two classes). Lastly, the XGBoost algorithm has a built-in feature importance function. Given that XGBoost is decision tree–based and forms an ensemble of numerous submodels (further explained in Section 3.1.2), it determines feature importance in a slightly more complicated way compared to Gini and Fisher ranking. Recording how often a feature is used to split a node in the decision trees of the ensemble, the algorithm quantifies each feature's average contribution to the decision-making process. Summing up these quantities across all trees in the ensemble, XGBoost returns the order of each feature's contribution toward accurate predictions.

Comparing feature ranks across all methods, we retain only nine out of our original 24 features to develop our prediction models. The ranking of the parameters was completed separately for every SC, yet resulting in the same list of five top-ranking features (though not in the same order). Figure 7 highlights the substantial decline in scores between the leading feature—the previous day's last measured proton flux, compared to the last feature we retain for testing—long SXR flux standard deviation. The remaining features' importance scores become increasingly lower, and we do not consider them from this point on. While only the top five features were the same across the three methods, we selected four additional features by comparing scores appearing in at least two ranking methods (after the initial five matches), until scores dropped below 0.02. We deemed features below this insignificant toward the predictions of SPEs. Figure 7 presents these features and their rankings. Listed in descending order of their averaged ranking scores, the retained features are the previous day's last measured proton flux (measured at 22:00 UTC), proton flux maximum, proton flux standard deviation, short SXR flux mean, proton flux skewness, previous day's last measured long SXR flux, proton flux median, previous day's last measured short SXR flux, and long SXR flux standard deviation. It is clear from these retained features that the proton flux value from the previous day dominates in importance compared to the rest of the features in all feature selection measures. This is intuitive given that a rise of SEP intensity on the previous day can easily be used as a predictor that SEPs may cross (or continue crossing, if an SPE is in progress) the 10 pfu threshold the next day. Still, this may not always be the case as shown in the left panel of Figure 5 where an event may begin abruptly without any indication of rising proton intensity in the days leading up to it.

With five of the nine finalized features relating to proton flux (three of which are ranked as the first, second, *and* third most important), and two relating to each of the short and long SXR irradiance channels, we can confidently say that proton flux features are most important toward SPE prediction when using our data set. This upholds our conclusions from Sadykov et al. (2021) that using SXR features *in addition* to proton fluxes improves performance scores, but that higher scores depend largely on proton flux data.

2.5. Devising Training and Testing Data Sets by SC

For the application of ML models, data sets are divided into *training* and *testing* sets. We do so by splitting our data into the different SCs of interest, i.e., SC 22: 1986–1996, SC 23: 1996–2008, and SC 24: 2008–2019. During the training phase, models examine provided data from specific time intervals to learn patterns from and generate predictions for a different time interval (Bishop 2006). Part of the supplied data set is kept blind to the training data, forming the test set. After a model is initialized and parameterized, it is applied to the test set to generate predictions. These predictions are compared to the true target values to estimate how well a model may generalize to unforeseen data. For example, if considering training done using SC 23 data, model performance scores are generated based on how accurately the model is able to reproduce the observed events of the specified prediction (testing) window of either SC 22 or 24. All combinations of training and testing sets are used, allowing every SC to serve as the training and testing set at least once (e.g., training done on SC 22 and testing done using SC 23 data, training done on SC 22 and testing done using SC 24 data, and so on).



	Gini Importance	Fisher Score	XGBoost
Last proton flux	1	1	1
Proton flux maximum	0.43	0.79	0.30
Proton flux std	0.38	0.72	0.20
Short SXR flux mean	0.26	0.67	0.10
Proton flux skewness	0.19	0.58	0.05
Last long SXR flux	0.05	0.50	0.05
Proton flux median	0.04	0.40	0.05
Last short SXR flux	0.01	0.20	0.01
Long SXR flux std	6e-3	6e-3	3e-3

Figure 7. A representation of the top nine ranked features according to Gini importance, Fisher scoring, and XGBoost's built-in feature selection methods. Feature-ranking order being similar for all SCs of interest, scores are averaged across all cycles and are shown here. Given that the different methods scale on different ranges, we apply a minima-maxima normalization here *only* for visibility, and present these normalized ranking scores in the legend. Notably, the lowest ranked of these features (long SXR flux standard deviation) is barely visible in each chart.

Additionally, we stacked training data sets to explore if longer training time intervals allow more precise predictions (i.e., training SCs 23 and 24 together to test with SC 22 data, SCs 22 and 24 to test with SC 23 data, and finally, SCs 22 and 23 to test with SC 24 data). Once these different data sets are defined, we use a minima-maxima normalization with respect to *only* the training set and scale this transformation to the test set. It is important to directly normalize only the training set, otherwise, the model will be *exposed* to some of the test set's information and possibly learn from it, giving the model the advantage of having prior knowledge of its target output that it should not—and realistically will not have when applied elsewhere. Once the normalization is appropriately done, we employ SVM and XGBoost algorithms, discussed in Sections 3.1.1 and 3.1.2. Because our primary goal is not to parameterize a specific algorithm with minute detail, we use grid searches to modify a few model parameters relating *only* to optimizing classification. We then test the effects of over-sampling our positive classes using standard oversampling (positive-class duplication), synthetic minority oversampling technique (SMOTE), and adaptive synthetic (ADASYN) oversampling, discussed in Section 3.2. Model performance in both cases of single-cycle and double-cycle training using SVMs and XGBoost are discussed in Section 4.1.

It is important to note that the data are prepared equally between models to generalize results as much as possible. Concisely, data preparation and model evaluation follow the order: splitting our data set into three segments (each SC of interest) to use as training and testing sets, down-selecting input data to nine flux features contributing most to reliable predictions, normalizing only the training set and scaling this transformation to the testing set, using model-inherent class-balancing parameters or different oversampling methods, performing grid searches to establish optimal parameters to apply to our models, and finally, evaluating model performance considering evaluation metrics discussed in Section 3.3: TSS, Heidke Skill Scores$_2$, and recall.

3. ML Applications to SPE Forecasting

3.1. Learning Methods Considered

Selecting the appropriate model is vital when we consider resolving real-world challenges. Toward our effort of predicting SPEs, we compare the performance of supervised classification using an SVM, and a decision tree–based gradient boosting ensemble algorithm, XGBoost.

3.1.1. SVMs

Introduced in 1963, SVMs are supervised classifiers with roots in the theory of statistical learning with the ability to learn from nonlinear decision surfaces with the application of different kernels. Data are assigned to their respective classes depending on where they lie with respect to the hyperplane—features relevant and irrelevant toward the forecasting of SPEs (Bishop 2006). Black-box machines like SVMs aim to generate optimal hyperplanes with dimensions mirroring that of the number of features set as input data, determining feature space (the n-dimensions where input variables live) dimensionality (Bishop 2006). The arguments we apply to the SVM, supplied by the *scikit-learn*[17] library include a `radial basis function` (RBF) kernel, regularization parameter `C`, a kernel coefficient γ = `scale`, and `balanced` class weights.

For SVMs, a kernel refers to a method allowing the application of classifiers to nonlinear problems by mapping nonlinear data to higher-dimensional space, where data become more easily separable (either linearly, radially, or polynomially, depending on user input). Different kernels available to use with SVMs allow better data transformations depending on the input data set. A hyperplane is then built calculating the dot product between the transformed features. `Linear` kernels may be considered first given their low computation needs, allowing quick training and testing capabilities. Oftentimes, other kernels perform better, but it is simply important to note that this is not always the case. Over all kernels available, the `RBF` kernel proved to be the better choice when evaluated under parameter optimization techniques discussed in Section 3.2, and is the only kernel employed and discussed from this point on.

We apply a grid search to the regularization parameter `C`, which allows an SVM to build a hyperplane with both the largest minimum margin, and one separating as many instances as possible. The `C` parameter decides how to prioritize enhancing the latter. Given that the regularization parameter affects different testing data sets differently, there is no absolute of whether larger or smaller values lead to more appealing results. The γ = `scale` argument is the `RBF` kernel coefficient declaring how far the influence of a single training data point reaches. Automation by Buitinck et al. (2013) of scaling this parameter to the data alleviates the need to process it manually. Small γ values consider data points farther (the minority class) from where most clump together (the majority class) when defining the separation line. The automated scaling searches between higher and smaller values to achieve the best possible fit for the provided data set, working as almost a proxy to bridge the gap of our imbalance.

Lastly, we alter the assigned class weights, where the default value is `None` (all classes are assigned equal importance, i.e., weight = 1). As described in Bishop (2006), a loss function is a method of evaluating how well an algorithm models its supplied featured data set, i.e., an optimal algorithm minimizes its loss function. The `balanced` class weight parameter directly modifies an algorithm's loss function by varying penalties assigned to classes with different weights. Using this biases the model to favor predictions of the minority class by assigning larger weights to them.[18] To the same effect, we apply different oversampling techniques to an SVM, inflating positive-instance relevance in the training data set, discussed in Section 3.2.

3.1.2. Extreme Gradient Boosting: XGBoost

More recently, also exploring SEP prediction and considering TSS and HSS_2 as metrics to evaluate model performance, O'Keefe et al. (2023) show how ensembles lead to better and more robust (in terms of experiment-to-experiment variations) predictions. With increasing advances focusing on deep-learning techniques, Shwartz-Ziv & Armon (2021) found the open-source gradient boosting algorithm XGBoost developed in 2016 by Chen & Guestrin (2016) to be more effective for predictions compared to multiple more complicated deep-learning models. Therefore, in addition to an SVM, we test the performance of XGBoost when building SPE forecasts using our data set. The algorithm represents a boosting ensemble classifier and uses a *gradient descent* framework, generating new models from the output of preliminary models. As an iterative decision tree–based learning algorithm, one ends up with an *ensemble* of submodels working to optimize each new learner. This process terminates once the optimal model (as the loss function is minimized as far as possible) is reached. Building this ensemble allows accounting for multiple modeling results, leading to more stable and generalized results. We implement this method, again using the *scikit-learn* library, with the wrapper class `XGBClassifier`. Reiterating that our primary goal is not to parameterize a specific algorithm with minute detail, we specify only two default parameters: `booster: gbtree` and `scale positive` weights. Respectively, the `gbtree` specification indicates using tree-based models to incrementally build an ensemble. Other options include building an ensemble while dropping a submodel per iteration, or using linear functions instead. Lastly and similar to the SVM's `balanced` class weight, XGBoost has an intrinsic parameter to `scale positive` weights, balancing each data class instead of leaving them widely imbalanced. With the original imbalances $\sim \frac{\#\text{ of positive instances}}{\#\text{ of negative instances}} \ll 1$ for each SC of interest, `scale positive` weights work to alter weights applied to each data class and point to bring this closer to 1.

3.2. Considering Oversampling Techniques and Grid Search Optimization

In lieu of our original imbalance (Figure 1), oversampling calls for synthetically repeated observations of the minority class (days with SPEs), until its frequency in the data set is comparable to that of the majority class (days without SPEs). Doing this brings equal representation of both classes, as well as an artificially extended data set. Following the classical definition of oversampling, we use standard techniques to inflate the positive cases of days with SPEs observed on the multiplicity of reaching an imbalance ratio $\frac{\#\text{ of positive instances}}{\#\text{ of negative instances}} \sim 1$. While this seems analogous to the `balanced and scale positive` weight parameters inherent to SVM and XGBoost classifiers, oversampling alters the training data directly instead of adjusting class weights, providing different results due to their intrinsic methodologies. Doing this allows us to evaluate how SVM and XGBoost models perform when considering a data set with two classes of approximately the same weights instead of one drastically imbalanced (original imbalance ratios

[17] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[18] https://www.tensorflow.org/

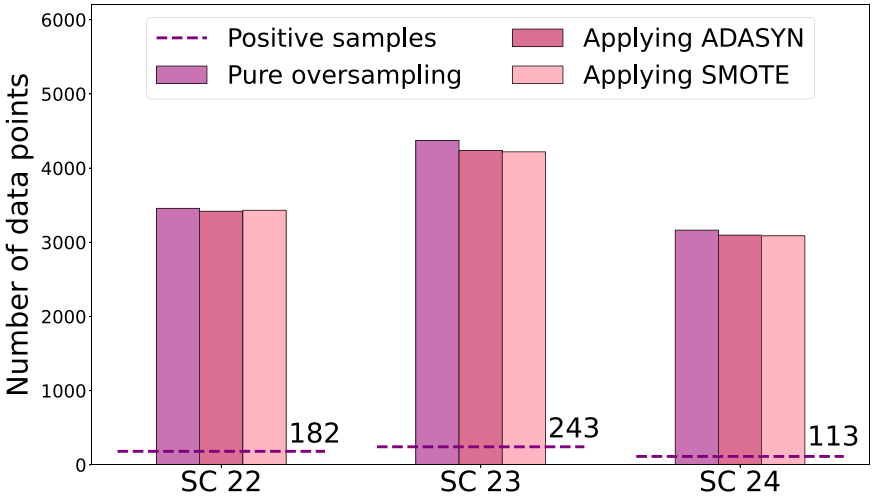


Figure 8. Differences between the number of data points after oversampling using standard (positive-class duplication), SMOTE, and ADASYN are shown here. Albeit on the order of tens, it is worth noting that the different methods intrinsically handle data differently.

follow: $\frac{\#\text{ of positive instances}}{\#\text{ of negative instances}} \sim 0.053$, 0.058, and 0.032 for SCs 22, 23, and 24 respectively).

The recent work of Stumpo et al. (2021) utilizes SMOTE to account for the maximum-likelihood estimation (MLE) for SPE prediction. They include SXR, radio fluence, and flare helio-longitudes as input features to take into account particle propagation from the Sun. Paired with a synthetically over-sampled data set, they found improved probabilities of detection compared to basic MLE and weighted MLE, with predictive scores increasing from 0.76 and 0.75 (respectively) to 0.80. Because of this, we explore how this technique may enhance our models. SMOTE works via linear interpolation, replicating minority instances between pre-existing positive data points to increase their presence in the data set. SMOTE uses an intrinsic k-nearest neighbor method (where we define $k = 5$) to select points for interpolation. Another oversampling technique we use, ADASYN, works similarly to SMOTE, except it focuses on generating data points in regions where the class imbalance is most prominent, giving more importance to positive instances harder to *reach* in feature space and learn from, reducing the risk of overfitting. ADASYN uses the k-nearest neighbor method on *each* data point so that each minority class's data points are associated with different neighborhoods. This allows for a more complicated data set to be generated, making the learning phase more complex (and realistic). Both SMOTE and ADASYN oversampling ensue until the ratio $\frac{\#\text{ of positive instances}}{\#\text{ of negative instances}} \simeq 1$.

As a final imbalance handling technique, we test the effects of standard oversampling on predictive models. Here, we simply duplicate the positive cases in our target training set as many times as needed until these cases appear just as frequently as the negative cases in our input catalog. In doing so, the training set is inflated to have as many positive cases as negative cases, but we apply none of the intricacies associated with SMOTE and ADASYN. The (small) differences in the number of positive cases after oversampling using these techniques are shown in Figure 8. Applying each technique to the input for an SVM and XGBoost algorithm, the synthesized data replaces the original training set, and the models continue working toward making predictions.

Our final modification to each model is using a parameter optimization technique to alter the most basic classification parameters of our models. Grid search is a process capable of automatically parsing through a specified range of numbers or strings appropriate for model arguments to find optimal values for evaluation (and we use the module provided by Buitinck et al. 2013). This pinpoints parameter values leading to the most accurate predictions. The grid search was performed when initializing the prediction models during the training phase, optimizing parameters to reach the maximum possible TSS scores (see Section 3.3).

We contain our parameter grid search on only two parameters for an SVM, kernels between `linear, polynomial, and radial basis functions`, finding `RBF` to be the optimal choice, and the regularization parameter `C`. Interestingly, changes in the `C` parameter did not lead to significant changes considering the CV TSS of each model when parsing through a wide range of 2^1–2^{10}, which Hsu et al. (2016) determined to be satisfactory.

We only apply one parameter of XGBoost to a grid search; the `learning rate`. The default value for this, defined by DMLC[19] is 0.3 (in a range of 0–1) to help prevent overfitting. When building an ensemble, the learning rate inherently decreases as the weights of each feature change at every *boosting* step. This allows the process to not *learn* too much from previous steps, which would otherwise incrementally build upon the initial feature weights to falsely make the first-built ensemble the most optimal. We see more variation in model accuracy here compared to how changes in `C` altered SVM performance.

Altering minimal hyperparameters, we are not fine-tuning the SVM or XGBoost algorithms to fit our specific training and testing configurations. Our results using grid searches when compared to default parameter model evaluations are just slightly better (on the order of $\sim 10^{-2}$), and therefore, are the only results considered from this point on (e.g., when discussing the default SVM model, we regard the SVM model with its default parameters *with* a grid search applied on the model-appropriate arguments mentioned above). We use grid searches on each type of model evaluation: each model (SVM and XGBoost) with its default parameters, each model with imbalance handling weights applied (`balanced` or `scale`

[19] https://xgboost.readthedocs.io/en/stable/parameter.html

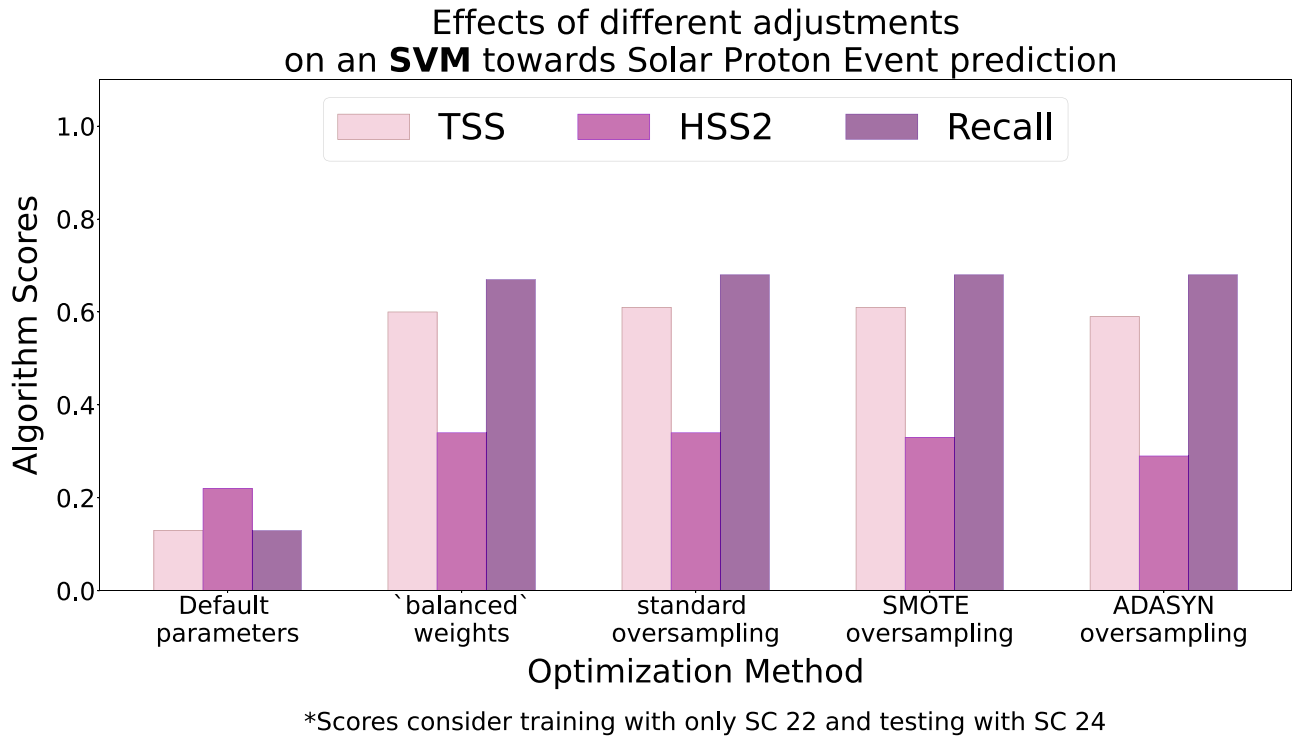


Figure 9. Resulting TSS, HSS_2, and recall scores of various adjustments made to single-cycle training data on an SVM.

`positive`), and the model applying standard/SMOTE/ADASYN oversampling (individually) on the training set. We assess each model's performance when making predictions based on short (trained on a single SC) and long (trained on two SCs) timescales.

3.3. Model Evaluation Metrics

Predictive scores and output may vary each time an algorithm is run, calling for methods to provide an average performance assessment over a large number of model iterations. *k*-fold CV is one of the most popular methods to do so. The term *k*-fold refers to how the entire training data set is partitioned into multiple subsets of equal sizes, or folds. $k = 10$ (10 folds) is commonly used (e.g., Leirvik & Yuan 2021; Bizzarri et al. 2022; Wang et al. 2023), and Olsen[20] discusses how on average, a model trained on 10 folds can be considered closest to that most effectively reducing prediction errors. After specifying the number of folds (which we leave as the default $k = 10$) to partition the data into, the model is trained on $(k - 1)$ folds, leaving the last for testing (each fold is used as a test set only once). The final performance score assigned to the model is aggregated from performance metrics across all training–testing splits, to provide a more comprehensive evaluation of predictive power. CV optimizes model performance with respect to a specified scoring scheme, for which we use TSS. As opposed to the default *k*-fold technique, we implement *scikit-learn's* modification of this, the *stratified* *k*-fold method. The difference here is that each fold preserves the number of positive instances in the data set, allowing positive instances to appear as many times as possible during the training phase to enhance model performance. Examples of these cross-validated results using single SC training are shown in Figures 9 and 10. We can see how consistent XGBoost scores remain throughout different training and testing combinations and oversampling methods, while SVMs show more variation *along with* reduced performance scores.

Evaluation metrics TSS and HSS_2 are widely used in space weather forecasting (O'Keefe et al. 2023). Following SWPC's formulation,[21] these are defined as

$$\mathrm{TSS} = \frac{(\mathrm{TP})}{(\mathrm{TP}) + (\mathrm{FN})} - \frac{(\mathrm{FP})}{(\mathrm{FP}) + (\mathrm{TN})}, \tag{1}$$

$$\mathrm{HSS}_2 = \frac{2 \cdot (\mathrm{TP} \times \mathrm{TN} - \mathrm{FN} \times \mathrm{FP})}{(\mathrm{TP} + \mathrm{FN}) \times (\mathrm{FN} + \mathrm{TN}) + (\mathrm{TP} + \mathrm{FP}) \times (\mathrm{TN} + \mathrm{FP})} \tag{2}$$

where TP = true positive, FN = false negative, FP = false positive, and TN = true negative forecasts. Mason & Hoeksema (2010) discuss how HSS_2 measures the performance of the forecasting model concerning random chance forecasts. TSS ranges from -1 to $+1$, where $+1$ indicates predictions made in perfect agreement with the testing set. Any misclassification then reduces this score accordingly. Values $\leqslant 0$ indicate model performance no better than a purely random forecast (Ahmadzadeh et al. 2021). HSS_2 ranges from -1 to $+1$ as well, where an algorithm of complete accuracy obtains a score of $+1$, an algorithm forecasting no events obtains a score of 0,[21] and an algorithm no better than random guesses obtains a negative score. An advantage of using TSS to validate our algorithm is its neutrality toward the class-imbalance ratio—the score itself does not depend on the inequality in trials. Bloomfield et al. (2012) and Manzato (2005) echo this in their

[20] https://cran.r-project.org/web/packages/cvms/vignettes/picking_the_number_of_folds_for_cross-validation.html

[21] https://www.swpc.noaa.gov/sites/default/files/images/u30/Forecast%20Verification%20Glossary.pdf

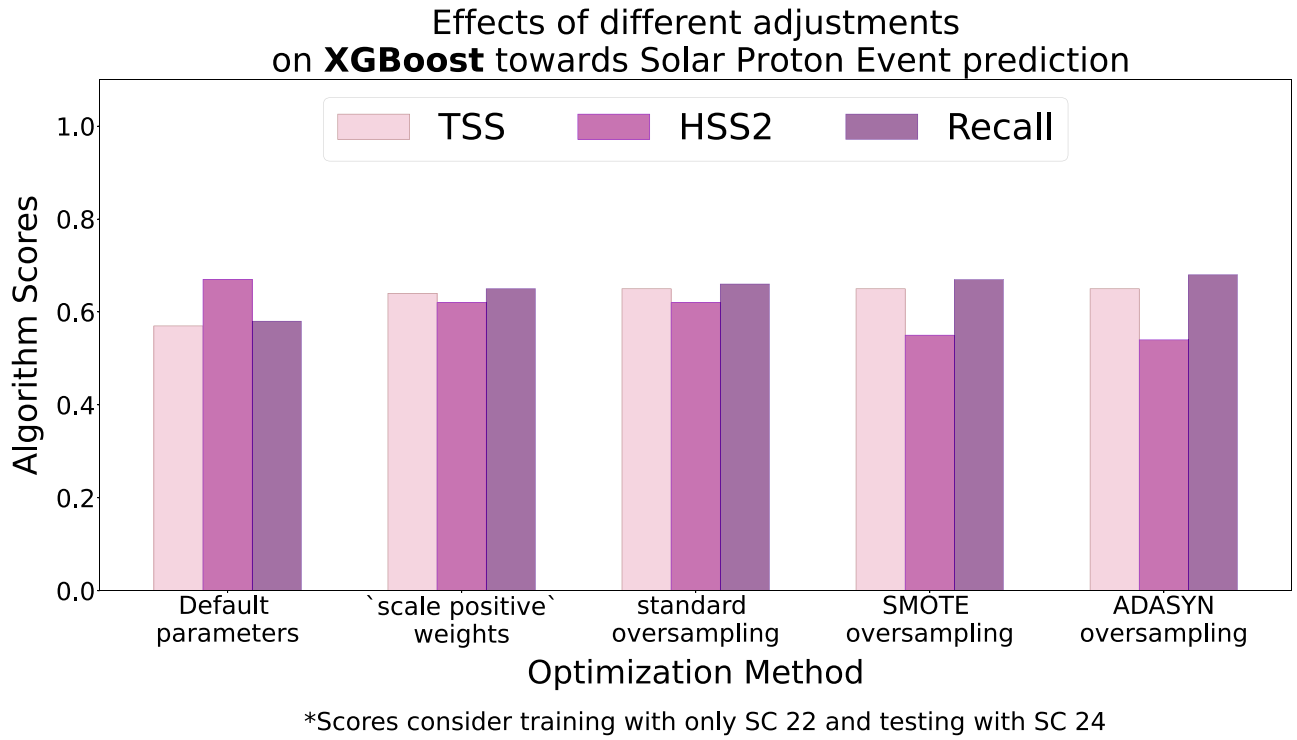


Figure 10. Resulting TSS, HSS_2, and recall scores of various adjustments made to single-cycle training data on an XGBoost model.

respective works, agreeing that TSS is an adequate measure of the overall classifier quality and should be the standard to use in comparisons of the performance of various classifiers for flare, weather, and rare event forecasting (Ahmadzadeh et al. 2021). Therefore in this work, we optimize model performance using CV with respect to TSS.

After the models undergo the process of tenfold CV, different performance measures are generated. A caveat here is that given the imbalance, CV scores falsely show model accuracy to be ∼97% even when failing to predict numerous SPEs. Accuracy is therefore not a reliable measure of predictive power, and we use *recall* to describe model capabilities instead. Recall is the calculation of a model's ability to predict/observe all positive instances supplied by the test set. Working directly on the ratio of predicted SPEs to the total number of observed SPEs, it is an unambiguous metric to evaluate how well the model has been trained to reproduce observations. Recall scores lie in a 0–1 range, where 0 reflects the model's inability to identify *any* positive instances, and 1 means that the model correctly identified *all* positive instances of the test set.

4. SPE Prediction Results and Discussion

Understanding the relations between physical parameters of solar radiation and energetic particle fluxes across different SCs is not an extensively studied problem for the prediction of SPEs. Following our conclusions in Sadykov et al. (2021) and using exclusively operational proton and SXR GOES flux data, our statistical catalog (model input data set; Section 2.3) records a total of 24 flux features. We reduce this to the *top nine ranked* features based on results of Gini importance, Fisher scoring, and XGBoost feature selection. By training SVM and XGBoost models using only these features along with the aforementioned grid search applied parameters and oversampling methods, we obtain the performance metrics (TSS, HSS_2, and recall) shown in Figure 11.

4.1. Resulting Forecast Accuracy

We find that the SVM model performance varies for single SC training and testing runs across the three SCs, as well as across the implemented oversampling strategies, reflected by the large spread of scores: TSS ranging between 0.12 and 0.68, HSS_2 between 0.19 and 0.50, and recall between 0.12 and 0.88. The lowest of these scores are associated with SVMs used with default parameters and with no class-imbalance treatment strategy applied. This reveals that without balancing the training data set, the classifier demonstrates inadequate results. Interestingly and in contrast, the XGBoost algorithm varies only about half as much, with TSS ranging between 0.56 and 0.75, and recall between 0.56 and 0.90, but a somewhat larger difference with HSS_2 between 0.22 and 0.75. For XGBoost, class-imbalance treatments typically result in higher TSS and recall scores, but simultaneously reduce HSS_2. Compared to SVMs used with default parameters, XGBoost used with default parameters performs significantly better, with performance metrics being ∼3× higher than those obtained with a default SVM. The shaded cells in Figure 11 indicate training–testing cases, where an SVM performed comparably to, or better than XGBoost in terms of the different metrics. As one can see, there are only a few shaded cells (only 23 out of 135), demonstrating the overall enhanced performance of XGBoost.

Further analyzing these scores, we summarize the highest prediction scores and identify median TSS scores in Figure 12 for every class-imbalance treatment technique (weight adjustments or oversampling). We again observe that XGBoost consistently sees improved median TSS scores compared to SVMs by ∼0.04–0.11 (columns 5 and 8). The peak TSS scores (columns 4 and 7) demonstrate approximately the same range

Case	Training Set	Testing Set	Input data adjustment	SVM			XGBoost		
				TSS	HSS_2	Recall	TSS	HSS_2	Recall
#1	SC 22	SC 23	default parameters	0.20	0.31	0.20	0.64	0.74	0.64
			class-weight balancing*	0.58	0.39	0.65	0.70	0.68	0.72
			Standard oversampling	0.55	0.38	0.63	0.72	0.68	0.74
			SMOTE oversampling	0.58	0.38	0.66	0.75	0.60	0.80
			ADASYN oversampling	0.59	0.36	0.69	0.75	0.53	0.81
#2	SC 22	SC 24	default parameters	0.13	0.22	0.13	0.57	0.67	0.58
			class-weight balancing	0.60	0.34	0.67	0.64	0.62	0.65
			Standard oversampling	0.61	0.34	0.68	0.65	0.62	0.66
			SMOTE oversampling	0.61	0.33	0.68	0.65	0.55	0.67
			ADASYN oversampling	0.59	0.29	0.68	0.65	0.54	0.68
#3	SC 23	SC 22	default parameters	0.25	0.36	0.26	0.62	0.69	0.63
			class-weight balancing	0.64	0.26	0.82	0.65	0.33	0.77
			Standard oversampling	0.63	0.28	0.79	0.66	0.35	0.77
			SMOTE oversampling	0.63	0.26	0.81	0.72	0.53	0.77
			ADASYN oversampling	0.64	0.22	0.86	0.70	0.40	0.80
#4	SC 23	SC 24	default parameters	0.14	0.23	0.14	0.56	0.68	0.56
			class-weight balancing	0.64	0.37	0.70	0.65	0.56	0.67
			Standard oversampling	0.58	0.38	0.64	0.65	0.60	0.67
			SMOTE oversampling	0.62	0.36	0.68	0.66	0.59	0.68
			ADASYN oversampling	0.64	0.33	0.73	0.67	0.53	0.70
#5	SC 24	SC 22	default parameters	0.35	0.37	0.38	0.64	0.69	0.65
			class-weight balancing	0.67	0.25	0.86	0.69	0.37	0.80
			Standard oversampling	0.68	0.30	0.82	0.69	0.40	0.78
			SMOTE oversampling	0.68	0.28	0.85	0.68	0.26	0.86
			ADASYN oversampling	0.66	0.23	0.88	0.67	0.22	0.90
#6	SC 24	SC 23	default parameters	0.36	0.50	0.36	0.67	0.74	0.67
			class-weight balancing	0.64	0.41	0.72	0.72	0.53	0.78
			Standard oversampling	0.59	0.42	0.65	0.70	0.50	0.76
			SMOTE oversampling	0.62	0.42	0.70	0.71	0.32	0.86
			ADASYN oversampling	0.67	0.37	0.78	0.70	0.27	0.89
#7	SC 23 & 24	SC 22	default parameters	0.25	0.36	0.26	0.61	0.68	0.62
			class-weight balancing	0.63	0.24	0.83	0.67	0.35	0.78
			Standard oversampling	0.64	0.25	0.82	0.66	0.34	0.77
			SMOTE oversampling	0.64	0.25	0.83	0.72	0.42	0.81
			ADASYN oversampling	0.62	0.19	0.87	0.70	0.34	0.82
#8	SC 22 & 24	SC 23	default parameters	0.24	0.38	0.24	0.65	0.75	0.65
			class-weight balancing	0.53	0.42	0.58	0.71	0.60	0.75
			Standard oversampling	0.53	0.42	0.59	0.71	0.63	0.74
			SMOTE oversampling	0.53	0.41	0.59	0.75	0.52	0.81
			ADASYN oversampling	0.57	0.42	0.64	0.74	0.39	0.86
#9	SC 22 & 23	SC 24	default parameters	0.12	0.21	0.12	0.58	0.69	0.58
			class-weight balancing	0.59	0.36	0.65	0.65	0.61	0.66
			Standard oversampling	0.61	0.37	0.67	0.64	0.58	0.65
			SMOTE oversampling	0.61	0.36	0.67	0.64	0.56	0.66
			ADASYN oversampling	0.62	0.35	0.69	0.68	0.54	0.71

Figure 11. SVM and XGBoost performance scores for each evaluated training–testing configuration. The shaded cells are the *only* instances where an SVM produces better (or comparable) results than XGBoost. Column 1 labels the different training–testing configurations referred to in Figures 12 and 13. Also note that each *model adjustment* here includes the grid search optimization discussed in Section 3.2. *Inherent class weight balancing adjustments are the `balanced` and `scale positive` weight parameters inherent to SVM and XGBoost respectively.

of improved scores. This allows us to conclude that XGBoost is performing statistically better for the considered problem compared to an SVM. Figure 13 presents the highest TSS, HSS_2, and recall scores found for single-SC and double-SC training timescales, and the corresponding oversampling techniques. While the oversampling strategy differs, the strongest HSS_2 and recall scores are again higher for the XGBoost model, except for the experiment presented in the last row. Interestingly, this figure shows that ADASYN more often leads to the maxima scores in comparison with other techniques (for both SVM and XGBoost).

Figures 12 and 13 help understand which class-imbalance treatment (i.e., various oversampling methods or class weight adjustments) leads to the best model performance. Columns 5 and 8 in Figure 12 show that all these techniques generate scores very close to each other in terms of their median TSS scores. The only significant difference occurs for XGBoost-based models trained on double-SC time intervals, where standard oversampling and class weight adjustment perform notably worse ($TSS = 0.66 \pm 0.03$ and $TSS = 0.67 \pm 0.03$ respectively) when compared to SMOTE ($TSS = 0.72 \pm 0.02$) and ADASYN ($TSS = 0.70 \pm 0.02$). Overall, we can generalize the above findings by stating that the employment of *any* balancing technique considered in this work improves predictions, with no clear preference for a single technique.

The data flows utilized for SPE prediction in this work (proton and SXR fluxes) are available in real time. Therefore, it is possible and meaningful to compare the effort in this paper with historical daily operational predictions of SPEs. Here, we examine the performance of developed ML models considering

Training timescale	Class-imbalance treatment	Train-test case	(a) Max. TSS	(a) Median TSS	Train-test case	(b) Max. TSS	(b) Median TSS
Single SC	Weight adjustment	#5	0.67	0.64 ± 0.02	#6	0.72	0.67 ± 0.03
	Standard	#5	0.68	0.60 ± 0.02	#1	0.72	0.68 ± 0.02
	SMOTE	#5	0.68	0.62 ± 0.01	#1	0.75	0.69 ± 0.03
	ADASYN	#6	0.67	0.64 ± 0.03	#1	0.75	0.68 ± 0.02
Double SC	Weight adjustment	#7	0.63	0.59 ± 0.05	#8	0.71	0.67 ± 0.03
	Standard	#7	0.64	0.61 ± 0.03	#8	0.71	0.66 ± 0.03
	SMOTE	#7	0.64	0.61 ± 0.03	#8	0.75	0.72 ± 0.02
	ADASYN	#7	0.62	0.62 ± 0.00	#8	0.74	0.70 ± 0.02

Figure 12. Oversampling techniques used with both (a) an SVM and (b) XGBoost, showing the maximum TSS obtained from each method across different timescales. Median absolute deviations are also shown.

previous SC (SCs 22–24) operational SWPC forecasts of SPEs. The SWPC NOAA forecasts considered in this work are daily probabilistic forecasts (in contrast with binary predictions made by ML models), with probabilities ranging between 1 and 99 on the possibility of an SPE occurring the next day. Bain et al. (2021) described how these forecasts relied on corrections made manually based on forecaster experience. The forecasts made by SWPC consider predictions made for three consecutive days, but we only use next-day predictions for a direct comparison between these forecasts with SVM and XGBoost models. To convert the probabilistic forecast to binary forecasts, we find the probability thresholds (i.e., the minimum probability starting from which a positive prediction is issued) that lead to the highest TSS or HSS_2 on the training data set, and apply it to the test data set. In addition, we analyze a persistence model as a baseline to assess our prediction scores. This model is straightforward and not ML-based—it does not train and test on data to make predictions. The persistence model uses the previous day's input of whether or not there was an SPE observed. If there was, the model makes a positive prediction for the next day; if not, the model predicts no events the next day. We show these scores in Figure 14, and compare them with the *highest* SVM and XGBoost performances (in terms of TSS) obtained from double-SC trained predictions.

There are several observations to mention based on Figure 14. First, the resulting metrics from the persistence model are very stable, ranging between 0.65 and 0.70 considering all three metrics. This model also demonstrates, on average, the highest HSS_2 of 0.65–0.68 across the considered models. Here we want to mention that XGBoost with default parameters (the scores are presented in Figure 13) leads to higher HSS_2 (0.74 and 0.75 for single-SC and double-SC training intervals respectively) than that found with the persistence model. This default model also led to acceptable TSS scores of 0.64 and 0.65 for the same experiments. Interestingly, the probabilistic SWPC NOAA results show significant variations from SC to SC. While the scores for SC 22 events were TSS = 0.49 and $HSS_2 = 0.20$—significantly lower with respect to persistence forecasts, scores increased to TSS = 0.69 and $HSS_2 = 0.59$ for the next two cycles, slightly outperforming the persistence forecast in terms of TSS scores.

It is also clear from Figure 14 that XGBoost typically performs better when considering TSS and recall metrics, albeit with a slightly lower TSS = 0.68 compared to SWPC NOAA forecasts during SC 24 (TSS = 0.69). In comparison, SVM and SWPC probabilistic forecasts typically show weaker performance in any training–testing case—together, these models account for the highest variance across all metrics, as well as the lowest measured TSS and HSS_2. Lastly, the persistence model showed the lowest variance in every metric, the highest resulting HSS_2, and the cost of the typically lower recall. It is worth noting that the higher HSS_2 are associated with the non-ML-driven models—this remains true even when the ML models are optimized with respect to HSS_2. As discussed in Section 3.3, increased HSS_2 indicates that the evaluated model performs significantly better than random guessing. We can account for these higher scores reflecting *less randomness* in SWPC forecasts given that it receives external input (from an experienced forecaster) that would only work to make predictions more accurate (Bain et al. 2021). Including relevant features in the model in this way could therefore enhance predictive abilities. Human input would also be very beneficial in cases where data needs to be re-assessed to continue making predictions because a machine alone may not know when/how to correct retrieved data. With persistence models, we already have knowledge of the occurrence of an SPE, and the model only continues its last prediction. With SCs dominated by non-eruptive periods, the relatively stable environment is reflected in the data and we see no random changes other than changes in labels from "0" to "1" and vice versa. This model's input can be very similar to the *previous day's last measured proton flux* feature (Section 2.4). Given the small model input, predictions align well with the observed outcomes, and we see improved HSS_2 compared to the ML models. However, in the ML-based models we use, we have eight additional flux features contributing to the accuracy of future forecasts. These other features allow the machines to learn new data and update forecasts when needed, building patterns between data leading up to SPEs rather than simply checking SEP counts from the

Training timescale	Scoring metric	Train-test case	(a) Max. score	(a) Oversampling method	Train-test case	(b) Max. score	(b) Oversampling method
single-SC	TSS	#5	0.68	SMOTE	#1	0.75	ADASYN
	HSS_2	#6	0.50	n/a	#1	0.74	n/a
	Recall	#5	0.88	ADASYN	#5	0.90	ADASYN
double-SC	TSS	#7	0.64	SMOTE	#8	0.75	SMOTE
	HSS_2	#8	0.42	ADASYN	#8	0.75	n/a
	Recall	#7	0.87	ADASYN	#8	0.86	ADASYN

Figure 13. Maximum TSS, HSS_2, and recall obtained using (a) an SVM and (b) XGBoost across different training timescales. Columns 5 and 8 show the oversampling technique used on the training data to achieve these maximum scores. Note: n/a applies to default model parameters, with no weight or positive-class adjustments.

day prior. Overall, it is worth noting that the ML-driven models, if tuned properly, outperform both the persistence model and SWPC NOAA operational forecasts across all three SCs, while being based on operational data flows.

4.2. Assessing SPE-predictive Model Cross Transferability

One of the interesting questions we analyze in this work is how model performance depends on the SCs on which they are trained/tested. SPEs remain relatively rare, leading to slightly different statistical properties and number of days with events during different SCs (see Figure 6). Interestingly, the differences in the forecast performances are not so drastic. Figure 12 illustrates that the median TSS scores for single-SC and double-SC training are generally comparable, with the median absolute deviations not exceeding 0.03. Yet, the individual experiments may demonstrate significant variations. For example, let us consider SMOTE oversampling for the XGBoost classifier in Figure 11. Below is a summary of TSS scores, with the training data intervals indicated in parentheses:

1. Predictions for SC 22: TSS = 0.72 (SC 23), TSS = 0.68 (SC 24), TSS = 0.72 (SCs 23 and 24);
2. Predictions for SC 23: TSS = 0.75 (SC 22), TSS = 0.71 (SC 24), TSS = 0.75 (SCs 22 and 24);
3. Predictions for SC 24: TSS = 0.65 (SC 22), TSS = 0.66 (SC 23), TSS = 0.64 (SCs 22 and 23).

We show the same for HSS_2:

1. Predictions for SC 22: $HSS_2 = 0.53$ (SC 23), $HSS_2 = 0.26$ (SC 24), $HSS_2 = 0.42$ (SCs 23 and 24);
2. Predictions for SC 23: $HSS_2 = 0.60$ (SC 22), $HSS_2 = 0.32$ (SC 24), $HSS_2 = 0.52$ (SCs 22 and 24);
3. Predictions for SC 24: $HSS_2 = 0.55$ (SC 22), $HSS_2 = 0.59$ (SC 23), $HSS_2 = 0.56$ (SCs 22 and 23).

We can make several observations following these scores. First, the TSS results were typically smaller for SC 24, irrelevant to the training interval used (SC 22 or 23). For example, median TSS scores across all the considered training intervals were TSS = 0.72 for SC 22, TSS = 0.75 for SC 23, and TSS = 0.65 for SC 24. Second, single SC training with SC 24 typically led to *less accurate* forecasting models (in terms of TSS and HSS_2) when training with SC 22 or 23. Interestingly, for double-SC training, the inclusion of SC 24 does not lead to an increase in TSS scores and even leads to a decrease in HSS_2. Further, according to Figure 6, the median properties of SPEs during SC 24 were comparable to the properties of SPEs for SC 22, so we cannot claim that the population of SPEs for SC 24 was statistically different from that of SC 22. We also note that this effect was not observed in SVMs trained on SC 24; the TSS and HSS_2 were comparable to the alternatives, yet still quantitatively smaller than in XGBoost experiments.

One fundamental difference between SCs 22, 23, and 24 is in the number of SPE events (or days with enhanced proton flux). Figure 6 points out that there were just 113 days during SC 24 when proton fluxes were enhanced, whereas SCs 22 and 23 saw 182 and 243 of such days, respectively. Overfitting is a known and expected side effect given the nature of learning models (Candice et al. 2019), and the problem of how rare SPEs are comes into play here, as the most common suggestion to reduce overfitting is by collecting more samples to provide models more data points to learn and generate complex patterns from. It is also documented that synthetic oversampling methods contribute to model overfitting (Li & Hu 2019), while providing no remedy against this effect. Figure 15 presents XGBoost's predictions for all training–testing cases across each class-imbalance handling technique. While most scores are acceptable, results also show certain training–testing configurations leading to inflated counts of false positives, or false alarms. This is most prominent when SC 24 serves as the training set (alone or in combination with another SC), and when training is done with SC 23 and tested using SC 22. Although the number of false positives is high, the model still may have potential in *all-clear* prediction efforts, given its high recall values/low false negative values.

Additionally, we would like to discuss what methods can potentially be implemented (none tested in this paper) to avoid overfitting. Yildirim[22] and Brownlee[23] discuss the most popular methods to reduce overfitting in SVMs: by adjusting C and γ parameters, using resampling to estimate model accuracy, and refraining interactions with a validation data set.

[22] https://towardsdatascience.com/hyperparameter-tuning-for-support-vector-machines-c-and-gamma-parameters-6a5097416167
[23] https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/

Scoring metric	Train-test case	Testing cycle	SWPC probability	Persistence model score	SVM score	XGBoost score
TSS	#7	SC 22	0.49	0.66	0.64	0.72
HSS_2			0.20	0.65	0.25	0.42
Recall			-	0.67	0.83	0.81
TSS	#8	SC 23	0.51	0.68	0.57	0.75
HSS_2			0.54	0.68	0.42	0.52
Recall			-	0.70	0.64	0.81
TSS	#9	SC 24	0.69	0.65	0.62	0.68
HSS_2			0.59	0.65	0.35	0.54
Recall			-	0.66	0.69	0.71

Figure 14. Comparing SVM and XGBoost performance to NOAA SWPC's previously predicted SPE probabilities and a persistence model. For SVM and XGBoost, we show scores obtained when training each model on the remaining two SCs when considering each testing cycle. Note that the persistence model does not follow a training–testing framework.

We apply all of these to our model through grid searches, specifying $\gamma =$ `scale`, using a k-fold CV to optimize the model by training and testing on different data subsets, and always specifying an SC as our test set. However, as noted above, the SVM does not particularly seem to experience this problem. On the other hand, the overfitting we see using XGBoost may come from many different factors. Confining our grid search here to only the model's `learning rate`, and specifying gradient boosting via trees, we do not augment multiple parameters associated with the algorithm. Jain[24] discusses how in terms of generalizing new data, tuning complexity *and* regularization parameters may show improvements. Important parameters to consider tuning in XGBoost in addition to the ones adjusted may be `max_depth` (controls tree depth), `subsample` (specifies the number of observations to consider in each tree), `colsample_bytree` (specifies the fraction of features to consider per tree), and setting L1 and L2 regularization terms using parameters `lambda` and `alpha`, respectively. Tuning these parameters was not evaluated in this work.

5. Summary and Conclusions

In this work, we have investigated the problem of predicting SPEs using ML-driven algorithms, and the cross cycle transferability of the developed models. We conclude that the XGBoost algorithm produces a finer predictive model compared to SVMs across all evaluations considered: each model with its default parameters, each model with imbalance handling class weights applied, and each model using data oversampled by standard, SMOTE, and ADASYN separately. We also find that these class-imbalance treatment techniques lead to approximately the same results with a slight preference toward oversampling done with synthetic data generation, especially in the case of XGBoost. A summary of our key results is as follows:

1. We built two catalogs during this effort using GOES data from SCs 22–24. The first catalog records the proton flux features of all SPEs detected during this time. The second catalog contains daily statistical flux features of these high-energy ($\geqslant$10 MeV) protons exceeding $\geqslant$10 pfus, and that of SXRs, which serve as the input data set for ML-based SVM and XGBoost algorithms.
2. We use Gini importance, Fisher scoring, and XGBoost's inherent feature-ranking method to determine which flux features are most important when building our prediction models. We find the top five ranked features to be the same across all methods for every SC considered, though not in the same order. The last proton flux count from the previous day is the most important, as can be expected. Of all flux features retained, those of protons were more relevant than those relating to both short- and long-wavelength SXRs.
3. We compare model-inherent class imbalance handling techniques (SVM: `balanced` class weights, XGBoost: `scale positive` weights) to oversampled versions of our data set using standard duplication, SMOTE, and ADASYN oversampling. For both models, we see

[24] https://medium.com/@rithpansanga/the-main-parameters-in-xgboost-and-their-effects-on-model-performance-4f9833cac7c

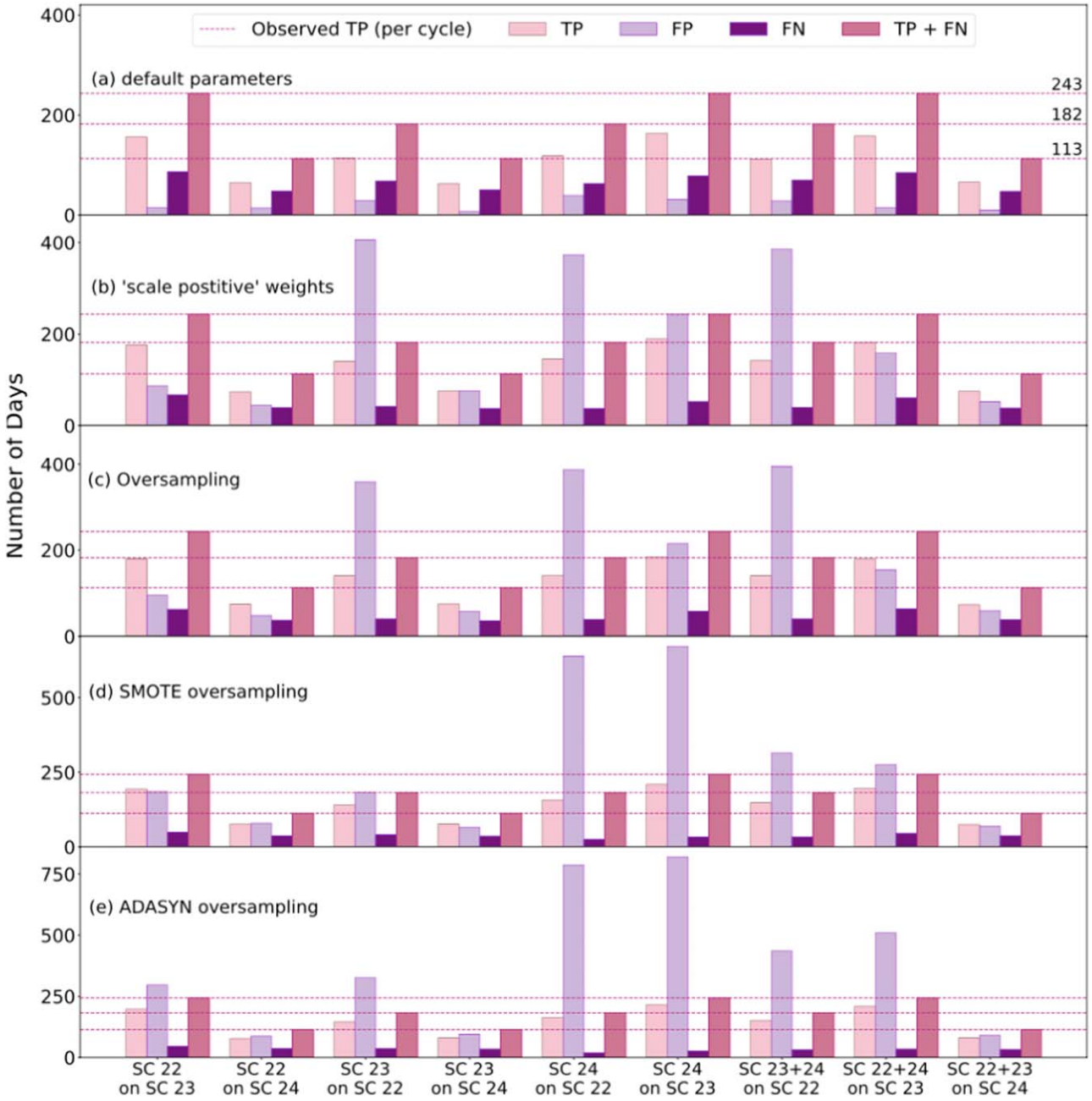


Figure 15. Predicted TP, TN, FP, and FNs made using an XGBoost-based model for different training and testing configurations (the *x*-axis shows which SC is the training set, then which SC it is tested on). The horizontal dotted lines are the target TPs as observed per cycle (SC 22: 182, SC 23: 243, SC 24: 113).

maximum scores obtained when ADASYN oversampling is applied (see Figure 13).

4. Figure 11 shows TSS, HSS_2, *and* recall scores from using XGBoost being statistically higher with respect to SVM scores for most of the considered experiments. On average, XGBoost predictions generate TSS $\sim +0.10$, $HSS_2 \sim +0.20$, *and* recall $\sim +0.10$ compared to those obtained by an SVM.
5. We assess our models considering both long (using two SCs for training) and short (using a single SC for training) timescales. We find TSS and HSS_2 to be comparable in both cases, for both models, for each tested SC.
6. We compare our results to SWPC daily probabilistic forecasts and a persistence model, shown in Figure 14, and find that XGBoost (optimized with respect to TSS) outperforms these baseline models concerning TSS and recall. While the HSS_2 is higher for the persistence model (0.65–0.68), we note that some experiments for XGBoost without oversampling still demonstrated higher scores (0.74–0.75; see Figure 13) with comparable TSS scores (0.64–0.65) for these cases. This indicates that ML-driven models based on operational data can outperform the current operational forecasts.
7. Training done with SC 24 produces weaker TSS and HSS_2, even when paired with SC 22 or SC 23. Inadequate performance of single SC training based on SC 24 may potentially indicate XGBoost's issues with overfitting, given poorer statistics of SPEs during this cycle. We also

observe that TSS for SC 24 are typically lower with respect to those obtained for other cycles.

8. Even with the highest predicted TSS scores, XGBoost produces a significant number of FPs in several training–testing configurations (Figure 15), including all experiments involving training on SC 24. Still, the model may have potential in all-clear prediction efforts, given its high recall value.

From these results, we conclude that XGBoost ensemble-based models combined with class-imbalance treatment (with no clear preference for any tested treatment) show the most potential compared to SVMs when building a model, in both the single- and double-SC trained cases. We find these ML-based models to outperform both the persistence and SWPC NOAA forecasts in all training–testing experiments considering only SCs 22–24. We also claim that the expected performance of the models may differ depending on the properties of the individual SC, specifically indicating that training with events during SC 24 may lead to poor model performance.

While TSS and HSS_2 have shown slight increases across SCs when changing the aforementioned input of proton and SXR flux data, there remains a large disparity when aiming for a reliable SPE forecasting algorithm. We hypothesize further improvement in predictions by complementing our work with (1) further hyperparameter tuning or regularization (e.g., using L1/4 regularization methods (Kolluri et al. 2020), using Bayesian regularization (Cawley & Talbot 2007), etc.) with respect to specific metrics, (2) alternative approaches with ML algorithms such as using neural networks (van der Sande et al. 2023), logistical regression, random forests (Sinha et al. 2022), etc. and (3) using proton and SXR fluxes' time series directly instead of utilizing their statistical moments. We also hypothesize significant contributions to better predictive scores by considering additional input data such as properties and topologies of source ARs (Marroquin et al. 2023), parameters of coronal mass ejections (Torres et al. 2022), or various dynamic features of the solar corona (Gibson 2015).

Acknowledgments

We thank the anonymous referee for the valuable suggestions that significantly improved the quality of the manuscript. The research was supported by NASA Early Stage Innovation program grant 80NSSC20K0302, NASA LWS grant 80NSSC19K0068, NSF EarthCube grant 1639683, and NSF grant 1835958. V.M.S. acknowledges the NSF FDSS grant 1936361 and NSF grant 1835958. E.I. acknowledges the RSF grant 20-72-00106.

ORCID iDs

Aatiya Ali https://orcid.org/0000-0003-3196-3822
Viacheslav Sadykov https://orcid.org/0000-0002-4001-1295
Alexander Kosovichev https://orcid.org/0000-0003-0364-4883
Irina N. Kitiashvili https://orcid.org/0000-0003-4144-2270
Gelu M. Nita https://orcid.org/0000-0003-2846-2453
Egor Illarionov https://orcid.org/0000-0002-2858-9625

References

Ahmadzadeh, A., Aydin, B., Georgoulis, M. K., et al. 2021, ApJS, 254, 23
Aminalragia-Giamini, S., Raptis, S., Anastasiadis, A., et al. 2021, JSWSC, 11, 59
Anastasiadis, A., Lario, D., Papaioannou, A., Kouloumvakos, A., & Vourlidas, A. 2019, RSPTA, 377, 20180100
Asaly, S., Gottlieb, L.-A., & Reuveni, Y. 2021, IJSTA, 14, 1469
Bailey, R. L., Reiss, M. A., Arge, C. N., et al. 2021, SpWea, 19, e02673
Bain, H. M., Steenburgh, R. A., Onsager, T. G., & Stitely, E. M. 2021, SpWea, 19, e2020SW002670
Beck, P., Latocha, M., Rollet, S., & Stehno, G. 2005, AdSpR, 36, 1627
Bishop, C. M. 2006, Pattern Recognition and Machine Learning (Berlin: Springer)
Bizzarri, I., Barghini, D., Mancuso, S., et al. 2022, MNRAS, 515, 5062
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
Bobra, M. G., & Ilonidis, S. 2016, ApJ, 821, 127
Bollavaram, P., & Asmatulu, R. 2016, Master's thesis, Wichita State Univ. http://hdl.handle.net/10057/12862
Buitinck, L., Louppe, G., Blondel, M., et al. 2013, arXiv:1309.0238
Candice, B., Csorgo, A., & Martinez-Munos, G. 2019, arXiv:1911.01914
Cawley, G. C., & Talbot, N. L. C. 2007, JMLR, 8, 841
Chen, T., & Guestrin, C. 2016, arXiv:1603.02754
Collins, P. 2006, AdSpR, 37, 116
Gibson, S. 2015, in IAU Symp. 305, Polarimetry: From the Sun to Stars and Stellar Environments, ed. K. N. Nagendra et al. (Cambridge: Cambridge Univ. Press), 245
Gu, Q., Li, Z., & Han, J. 2012, arXiv:1202.3725
He, J., & Rodriguez, J. V. 2018, SpWea, 16, 245
Hsu, C.-W., Chang, C.-C., & Lin, C.-J. 2016, A Practical Guide to Support Vector Classification, National Taiwan University, https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf
Kasapis, S., Zhao, L., Chen, Y., et al. 2022, SpWea, 20, e2021SW002842
Kolluri, J., Kotte, V. K., PhridviRaj, M. S. B., & Razia, S. 2020, in 2020 4th Int. Conf. on Trends in Electronics and Informatics (ICOEI) (Piscataway, NJ: IEEE), 934
Leirvik, T., & Yuan, M. 2021, E&SS, 8, e01527
Li, K., & Hu, Y. 2019, JPhCS, 1303, 012095
Li, X., Ma, L., Chen, P., et al. 2022, Energy Rep., 8, 1087
Manzato, A. 2005, WtFor, 20, 918
Marroquin, R. D., Sadykov, V., Kosovichev, A., et al. 2023, ApJ, 952, 97
Martens, P. C., & Angryk, R. A. 2017, in IAU Symp. 335, Space Weather of the Heliosphere: Processes and Forecasts, ed. C. Foullon & O. E. Malandraki (Cambridge Univ. Press: Cambridge), 344
Mason, J. P., & Hoeksema, J. T. 2010, ApJ, 723, 634
McGuire, D., Sauteraud, R., & Midya, V. 2019, in 2019 IEEE Int. Conf. on Big Data (Piscataway, NJ: IEEE), 5836
Menze, B., Kelm, B., Masuch, R., et al. 2009, BMC Bioinform., 10, 213
Naito, M., Kodaira, S., Ogawara, R., et al. 2020, LSSR, 26, 69
O'Keefe, P. M., Sadykov, V., Kosovichev, A., et al. 2023, arXiv:2303.08092
Onorato, G., Di Schiavi, E., & Di Cunto, F. 2020, FrP, 8, 362
Rotti, S., Aydin, B., Georgoulis, M. K., & Martens, P. C. 2022, ApJS, 262, 29
Rotti, S. A., & Martens, P. C. 2023, ApJS, 267, 40
Sadykov, V., Kosovichev, A., Kitiashvili, I., et al. 2021, arXiv:2107.03911
Shwartz-Ziv, R., & Armon, A. 2021, arXiv:2106.03253
Sinha, S., Gupta, O., Singh, V., et al. 2022, ApJ, 935, 45
Stumpo, M., Benella, S., Laurenza, M., et al. 2021, SpWea, 19, e2021SW002794
Torres, J., Zhao, L., Chan, P. K., & Zhang, M. 2022, SpWea, 20, e2021SW002797
van der Sande, K., Muñoz-Jaramillo, A., & Chatterjee, S. 2023, ApJ, 955, 148
Wang, J., Guo, D., Xiang, Z., et al. 2023, AdSpR, 71, 275
Whitman, K., Egeland, R., Richardson, I., et al. 2023, AdSpR, 72, 5161