UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Tecnologia

Tiago Cinto

Solar flare forecasting: a methodology to automate the design of classifiers for events of diverse classes

Predição de explosões solares: uma metodologia para automatizar o projeto de classificadores para eventos de classes diversas

Limeira
2020

Tiago Cinto

Solar flare forecasting: a methodology to automate the design of classifiers for events of diverse classes

Predição de explosões solares: uma metodologia para automatizar o projeto de classificadores para eventos de classes diversas

Tese apresentada à Faculdade de Tecnologia da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Doutor em Tecnologia, na área de Sistemas de Informação e Comunicação.

Thesis presented to the School of Technology of the University of Campinas in partial fulfillment of the requirements for the degree of Doctor in Technology in Computer Science, in the area of Sistemas de Informação e Comunicação.

Supervisor/Orientador: Prof. Dr. André Leon Sampaio Gradvohl
Co-supervisor/Coorientador: Prof. Dr. Guilherme Palermo Coelho

Este exemplar corresponde à versão final da Tese defendida por Tiago Cinto e orientada pelo Prof. Dr. André Leon Sampaio Gradvohl.

Limeira
2020

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Faculdade de Tecnologia
Felipe de Souza Bueno - CRB 8/8577

C493s

Cinto, Tiago, 1990-
Solar flare forecasting : a methodology to automate the design of classifiers for events of diverse classes / Tiago Cinto. – Limeira, SP : [s.n.], 2020.

Orientador: André Leon Sampaio Gradvohl.
Coorientador: Guilherme Palermo Coelho.
Tese (doutorado) – Universidade Estadual de Campinas, Faculdade de Tecnologia.

1. Astronomia. 2. Astrofísica. 3. Erupções solares. 4. Aprendizado de máquina. 5. Mineração de dados (Computação). I. Gradvohl, André Leon Sampaio, 1973-. II. Coelho, Guilherme Palermo, 1980-. III. Universidade Estadual de Campinas. Faculdade de Tecnologia. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: Predição de explosões solares : uma metodologia para automatizar o projeto de classificadores para eventos de classes diversas
Palavras-chave em inglês:
Astronomy
Astrophysics
Solar flares
Machine learning
Data mining
Área de concentração: Sistemas de Informação e Comunicação
Titulação: Doutor em Tecnologia
Banca examinadora:
André Leon Sampaio Gradvohl [Orientador]
José Roberto Cecatto
Marcela Xavier Ribeiro
Paulo José de Aguiar Simões
João Roberto Bertini Júnior
Data de defesa: 10-11-2020
Programa de Pós-Graduação: Tecnologia

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0003-0724-780X
- Currículo Lattes do autor: http://lattes.cnpq.br/3206039376378317

FOLHA DE APROVAÇÃO

Abaixo se apresentam os membros da comissão julgadora da sessão pública de defesa de dissertação para o Título de Doutor em Tecnologia na área de concentração Sistemas de Informação e Comunicação, a que se submeteu o aluno Tiago Cinto, em 10 de novembro de 2020 na Faculdade de Tecnologia – FT/UNICAMP, em Limeira/SP.

Prof. Dr. André Leon Sampaio Gradvohl
Presidente da Comissão Julgadora

Dr. José Roberto Cecatto
DAS/INPE

Profa. Dra. Marcela Xavier Ribeiro
DC/UFSCar

Prof. Dr. Paulo José de Aguiar Simões
CRAAM/Mackenzie

Prof. Dr. João Roberto Bertini Junior
FT/UNICAMP

Ata da defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria de Pós Graduação da FT.

Acknowledgements

Writing this Ph.D. thesis without the support of some kind people could not have been possible. I would like to express a mention here to some of them.

First and foremost, I would like to thank my daughter Melissa and wife Marina for the unceasing motivational, emotional, and personal support. My parents, father Sérgio and mother Kelly, and family, grandmother Zilda and aunt Zilda Maria, for the unequivocal wisdom.

This thesis could not have been possible without the help and support of my principal supervisor, Prof. Dr. André L. S. Gradvohl, not to mention his knowledge and advice on space weather. Besides, the advice and support on machine-learning of my co-advisor, Prof. Dr. Guilherme P. Coelho, have been rather valuable, for which I am grateful. Not to mention, the advice also on machine-learning of Profa. Dra. Ana Estela A. da Silva and her support since undergraduation, for which I am grateful too.

I would also like to thank Federal Institute of Education, Science, and Technology of Rio Grande do Sul (IFRS) – Campus Feliz, for the financial support and cooperation with this research.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

Resumo

O Sol possui uma atmosfera ativa, apresentando diversos fenômenos que afetam diretamente todos os corpos do sistema solar. O clima espacial refere-se aos eventos do Sol, incluindo o vento solar, o espaço próximo à Terra e a atmosfera terrestre. Perturbações no clima espacial podem prejudicar diversas áreas, incluindo aviação e espaço aéreo, satélites, indústrias de óleo e gás e sistemas elétricos, levando a perdas econômicas. Explosões solares – um dos eventos mais significantes – compreendem liberações repentinas de radiação que podem afetar a atmosfera terrestre em poucos minutos após a ocorrência. Deste modo, é de extrema importância a criação de sistemas para predizê-las. Embora diversas abordagens de predição *ad hoc* foram propostas na literatura nos últimos anos, poucos autores focaram-se em criar um processo de *design* automatizado para projetar tais sistemas com flexibilidade e desempenho otimizado. Assim, nesta tese, é proposta uma metodologia automatizada para projetar classificadores de explosões solares usando o framework Scikit-learn, baseado em Python. Esta metodologia objetiva oferecer um arranjo compreensivo de processos de aprendizagem de máquina baseados em boas práticas para o clima espacial. Basicamente, tal metodologia compreende: (i) divisão apropriada de dados; (ii) seleção de modelos; (iii) seleção de atributos; (iv) otimização de parâmetros; (v) análise da função de custo dos classificadores; (vi) reamostragem de dados; (vii) ajuste do ponto de corte dos classificadores; (viii) e avaliação sobre conjuntos de dados de validação/teste. Para validar esta metodologia, foram propostos alguns estudos de caso objetivando projetar classificadores de explosões de classe ≥ C e ≥ M (eventos acima ou iguais à classe C e M, respectivamente) para até 3 dias a frente. Resultados destes classificadores mostraram-se estar de acordo com as abordagens de predição da literatura de maiores desempenhos. Assim, para eventos ≥ C, os modelos otimizados pontuaram escores de *true skill statistics* (TSS) de 0.69 (próximas 24 h), 0.70 (próximas 48 h) e 0.73 (próximas 72 h), em consonância com seus altos escores de *true positive rates* (TPR) de 0.86 (próximas 24 h) and 0.89 (próximas 48 and 72 h). Por sua vez, para eventos ≥ M, os modelos pontuaram TSS = 0.53 (próximas 24 e 72 h) e 0.55 (próximas 48 h), também em consonância com seus altos TPRs de 0.83 (próximas 24 h), 0.85 (próximas 48 h) e 0.80 (próximas 72 h). Além disso, resultados da área sobre a curva (AUC) dos modelos confirmaram a potencial utilidade das predições positivas: AUC = 0.93 (próximas 24 h) e 0.94 (próximas 48 e 72 h), para eventos ≥ C, e AUC = 0.84 (próximas 24 h) e 0.85 (próximas 48 e 72 h), para eventos ≥ M. Aprendizagem de máquina automatizada pode ser útil para oferecer métodos do estado-da-arte e processos de *design* para pesquisadores interessados em aplicar conceitos ao invés de conhecer algoritmos detalhadamente. Isso permite projetar automaticamente modelos de predição com desempenho otimizado ao mesmo tempo em que poupa tempo e dinheiro, pois especialistas em aprendizagem de máquina podem ser caros ou difíceis de encontrar. Acredita-se que, especificamente para o clima espacial, automatizar a aprendizagem de máquina pode ser consideravelmente útil, uma vez que nem todos os físicos são especialistas no domínio da inteligência artificial.

Abstract

The Sun has a highly active atmosphere, featuring several events that directly affect all solar system bodies. The space weather refers to any conditions and events in the Sun, including the solar wind, impacting on the space near Earth, and the Earth's atmosphere. Disturbances in space weather can damage several fields, including aviation and aerospace, satellites, oil and gas industries, and electrical systems, leading to economic and commercial losses. Solar flares – one of the most significant events – comprehend sudden releases of radiation and particles affecting the Earth's atmosphere in a few minutes after occurrence. As such, it is imperative to create systems to forecast them. Although many *ad hoc* forecast approaches have been proposed in the literature in recent years, few authors have focused their research on establishing an automated design process to develop such predictors with flexibility and optimized performance. Accordingly, in this thesis, we propose an automated novel methodology for designing solar flare classifiers under Python's Scikit-learn framework. Our methodology intends to offer a comprehensive pipeline of machine learning processes relying on good practices for space weather forecasting. Overall, such methodology comprehends: (i) proper data splitting; (ii) model selection; (iii) feature selection; (iv) hyperparameter optimization; (v) analysis of classifiers' cost function; (vi) data resampling; (vii) adjustment of cut-off point of classifiers; and (viii) evaluation of validation and test data. We investigated some distinct case studies to design classifiers for forecasting ≥ C- and ≥ M-class flare events (higher than or equal to C- and M-class, respectively) up to three days ahead to validate our methodology. The designed classifiers' results agreed with the best performing forecast approaches in the literature. Accordingly, for ≥ C events, our models scored true skill statistics (TSS) scores of 0.69 (next 24 h), 0.70 (next 48 h), and 0.73 (next 72 h), agreeing with their high corresponding true positive rates (TPR) of 0.86 (next 24 h) and 0.89 (next 48 and 72 h). On the other hand, for ≥ M events, our models scored TSS = 0.53 (next 24 and 72 h) and 0.55 (next 48 h), also agreeing with their high corresponding TPRs of 0.83 (next 24 h), 0.85 (next 48 h), and 0.80 (next 72 h). Besides, results from our models' area under the curve (AUC) scores have confirmed the usefulness of the positive forecasts: AUC = 0.93 (next 24 h) and 0.94 (next 48 and 72 h), for ≥ C flares, and AUC = 0.84 (next 24 h) and 0.85 (next 48 and 72 h). Automated machine learning efforts can be useful to provide state-of-the-art learning methods and design processes to researchers interested in applying concepts rather than knowing the technologies in their in-depth details. That allows automatically designing forecast models with improved performance while saving a considerable amount of time and money as experts in machine learning can sometimes be expensive or hard to find. We believe that automating machine learning, specifically in space weather research, can be rather valuable as not all solar physicists are experts in the artificial intelligence domain.

List of Figures

List of Tables

List of Abbreviations and Acronyms

ACC	Accuracy
AI	Artificial Intelligence
ApSS	Appleman Skill Score
ASAP	Automated Solar Activity Prediction
AR	Active Region
BoM	Australia Bureau of Meteorology
BSS	Brier Skill Score
AUC	Area Under the Receiver Operating Characteristic Curve
CART	Classification and Regression Decision Trees
CME	Coronal Mass Ejection
CNN	Convolutional Neural Network
CSI	Critical Success Index
DAFFS	Discriminant Analysis Flare Forecasting System
DW	Data Warehousing
DSD	Daily Solar Data
ERT	Extremely Randomized Tree
ETL	Extract, Transform, and Load
FAR	False Alarm Ratio
FLARECAST	Flare Likelihood and Region Eruption foreCASTing
FORSPEF	The Forecasting Solar Particle Events and Flares Tool
FPR	False Positive Rate
GOES	Geostationary Operational Environmental Satellite
GPS	Global Positioning System
HMI	Helioseismic and Magnetic Imager
HSS	Heidke Skill Score
JSOC	Joint Science Operations Center
LASSO	The Least Absolute Shrinkage and Selection Operator
LSTM	Long-Short Term Memory Neural Network
k-NN	k-Nearest Neighbors
KDD	Knowledge Discovery in Databases
KMA	Korean Meteorological Administration
MAG4	Magnetogram Forecast Forecasting Tool

MOSWC	The Met Office Space Weather Operations Centre
MDI	Michelson Doppler Imager
NCEI	National Centers for Environment Information
NGDC	National Geophysical Data Center
NICT	Japan National Institute of Information and Communications Technology
NOAA	National Oceanic and Atmospheric Administration
NPV	Negative Predictive Value
OLAP	Online Analytical Processing
PC	Percent Correct
POD	Probability of Detection
POFD	Probability of a False Detection
PPV	Positive Predictive Value
PCA	Principal Components Analysis
QS	Quadratic Score
ROC	Receiver Operating Characteristic
ROB	Royal Observatory of Belgium
RVM	Relevance-Vector Machine
SDO	Solar Dynamic Observatory
SEP	Solar Energetic Particle
SEPsFLAREs	Solar Events Prediction System For Space Launch Risk Estimation
SMART	Solar Monitor Active Region Tracker
SIDC	Solar Influences Data Center
SRS	Sunspot Region Summary
SOHO	Solar and Heliospheric Observatory
SPRINTS	Space Radiation Intelligence System
SVM	Support-Vector Machine
SVR	Support-Vector Machine Regressor
SWPC	Space Weather Prediction Center
TPOT	Tree-Based Pipeline Optimization Tool
TNR	True Negative Rate
TPR	True Positive Rate
TSS	True Skill Statistics
UFCORIN	Universal Forecast Constructor by Optimized Regression of Inputs
UKMO	UK Meteorology Office
WEKA	Waikato Environment for Knowledge Analysis
WMFR	Weighted Mean Flare Rate

Contents

Chapter 1

Introduction

The Sun is located in the solar system center and weighs 300,000 times the Earth's mass. Its core mostly comprehends fusions of hydrogen and helium (MOLDWIN, 2008). The Sun has a very active atmosphere, mostly related to its magnetic fields' dynamic associated with its atmosphere's plasma. Accordingly, it features several events that affect other solar system bodies and the space weather, such as solar flares, solar energetic particles (SEP), and coronal mass ejections (CME) (MESSEROTTI et al., 2009).

Space weather involves the study of events that occur in the Sun, Earth's atmosphere, and the near-Earth space. Accordingly, it corresponds to an emerging field of space science aimed at researching how the Sun influences the Earth, including the technological and social interaction effects between both celestial bodies (MOLDWIN, 2008).

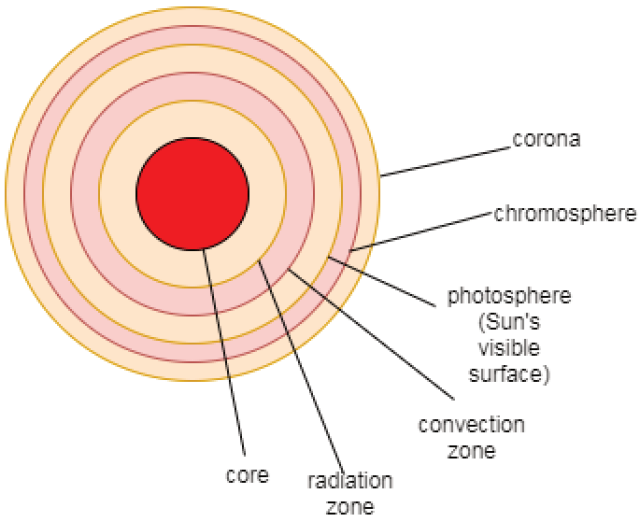

Figure 1.1: Sun's layers. Adapted from Moldwin (2008).

We can divide the Sun into distinct layers: the core, radiation zone, convection zone, photosphere, chromosphere, and the corona (Figure 1.1). Its most internal part – the core – is responsible for generating and transporting light energy towards space. The chromosphere and corona represent the Sun's atmosphere, whereas the photosphere is the surface. The Sun's atmosphere produces the events affecting the space weather (MOLDWIN, 2008).

In addition to the events mentioned earlier, the Sun also produces some structures often influencing the space weather, that is, the sunspots (MOLDWIN, 2008) and active regions (AR) (CANFIELD, 2001). ARs have their bases in the photosphere – in the form of sunspots and groups of sunspots –, but spread along the solar atmosphere through arcs and configurations of magnetic fields connecting distinct sunspots' and ARs' magnetic polarities from the photosphere. Those phenomena often create conditions for solar flare releases (CANFIELD, 2001).

Solar flares comprehend sudden releases of electromagnetic radiation. Depending on their intensities, they can associate with CMEs and SEPs eventually. Their electromagnetic radiation has a broad spectra, including X- and γ-rays. The energy associated with flares ranges on $[10^{24}, 10^{32}]$ erg. Flares can last for dozens of seconds to a few hours, depending on their intensities and characteristics.

There are many initiatives to monitor the Sun's events. Taking flares as an example, they are constantly recorded by the Geostationary Operational Environmental Satellites' (GOES) X-ray instruments, linked to the Space Weather Prediction Center (SWPC) of the National Oceanic and Atmospheric Administration (NOAA)[1], in the United States of America.

The X-ray instruments aboard the GOES satellites measure solar flare's fluxes (W/m^2) into two bands, namely the 0.5–4 Ångström (Å) and 1–8 Å. The latter band is used to classify solar flares' intensity into five distinct classes, as described next (CANFIELD, 2001).

Solar physicists classify solar flares through a labeled scale ranging between A- (the smallest events), B- (as tiny as A, usually called subflares), C- (cause few noticeable consequences on Earth), M- (medium-sized events often accompanying CMEs), and X-class (the largest events, causing major radio blackouts and long-lasting radiation storms in the Earth's atmosphere). Taking the strongest flare type (X-class) as an example, composed of X-rays releases > $10^{-4} W/m^2$, each class has its X-ray peak flux ten times higher than its predecessor.

Overall, disturbances in space weather can negatively affect several fields, including aviation and aerospace, satellites, Global Positioning System (GPS), oil and gas industries, and electrical

[1] http://www.swpc.noaa.gov

systems, leading to economic and commercial losses (NRC, 2009). Because of the several known solar flare effects, it is imperative to employ efforts to forecast them. Accordingly, this thesis investigates a novel automated methodology to design classifiers for such events.

In this sense, forecasting – accurately – solar flare events allows a series of mitigation actions, including warning astronauts in deep space that they are likely to be hit by radiation and powering off electronic equipments to avoid their risk of burning out. On the other hand, the consequences of mispredicting solar flares can be severe, depending on the associated event class (RABOONIK et al., 2016). We are facing a hot topic which became recurrent in research agendas (LEKA; BARNES, 2018).

Several SWPCs exist in an attempt to mitigate solar flare effects (CROWN, 2012; DEVOS; VERBEECK; ROBBRECHT, 2014; MURRAY et al., 2017; KUBO; DEN; ISHII, 2017; INPE, 2020). Such centers often employ human-based hybrid forecasts, such as in NOAA/SWPC (CROWN, 2012), which uses an expert system outputting cadenced forecasts, further confirmed or adjusted by solar physicists.

On the other hand, many researchers have also been employing efforts on system-based forecast approaches. Whereas some authors propose using photospheric magnetic data to investigate ARs and their relationship with solar activity (MCATEER; GALLAGHER, P. T.; CONLON, 2010), others focus on the research of the photospheric features (MCINTOSH, 1990) and magnetic topologies (HALE et al., 1919) of ARs concerning their association with solar flare productivity. Regardless of such data nature, researchers have been proposing many methods to forecast solar flare events. To name some, we can cite those listed in Table 1.1.

Table 1.1: Solar flare forecasting efforts.

Method	*Authorship*
AdaBoost	Lan et al. (2012)
Bayesian statistics	Díscola Jr. et al. (2018a,b), X. Zhang, J. Liu, and Q. Wang (2011), Yu, Huang, H. Wang, Cui, et al. (2010) and Wheatland (2005)
DecisionTrees	Díscola Jr. et al. (2019, 2018a,b), Huang and H.-N. Wang (2013), X. Zhang, J. Liu, and Q. Wang (2011), Huang, Yu, et al. (2010), Yu, Huang, Q. Hu, et al. (2010) and Yu, Huang, H. Wang, and Cui (2009)

Ensembles	Lim et al. (2019), Díscola Jr. et al. (2019), Guerra, Pulkkinen, and Uritsky (2015) and Huang, Yu, et al. (2010)
Expert systems	McIntosh (1990) and Miller (1988)
Extremely randomized trees (ERT)	Nishizuka, Sugiura, Kubo, Den, Watari, et al. (2017)
Image-case-based prediction	J.-F. Liu, F. Li, H.-P. Zhang, et al. (2017)
k-Nearest Neighbors (k-NN)	Díscola Jr. et al. (2019, 2018a,b), Nishizuka, Sugiura, Kubo, Den, Watari, et al. (2017), Winter and Balasubramaniam (2015), Huang and H.-N. Wang (2013) and R. Li, Cui, et al. (2008)
Linear classifiers	Jonas et al. (2018)
Linear discriminant analysis	Leka, Barnes, and Wagner (2018)
Learning vector quantization	R. Li and Zhu (2013) and Yu, Huang, H. Wang, and Cui (2009)
Long-short term memory network (LSTM)	Jiao et al. (2020), X. Wang et al. (2020) and H. Liu et al. (2019)
Multi-model prediction	J.-F. Liu, F. Li, Wan, et al. (2017)
Multiple linear regression	Shin et al. (2016)
OneR	Díscola Jr. et al. (2018a,b)

Neural networks	X. Li et al. (2020), Domijan, Bloomfield, and Pitié (2019), Zheng, X. Li, and X. Wang (2019), Nishizuka, Sugiura, Kubo, Den, and Ishii (2018), Inceoglu et al. (2018), Florios et al. (2018), Huang, H. Wang, Xu, et al. (2018), E. Park et al. (2018), Hada-Muranushi et al. (2016), Shin et al. (2016), Ahmed et al. (2013), R. Li and Zhu (2013), Colak and Qahwaji (2009), H.N. Wang et al. (2008), Qahwaji and Colak (2007), Colak and Qahwaji (2007) and Qahwaji and Colak (2006)
Poisson statistics	McCloskey, P. T. Gallagher, and Bloomfield (2018), D. A. Falconer et al. (2014), Bloomfield et al. (2012), D. Falconer et al. (2011) and P.T. Gallagher, Moon, and H. Wang (2002)
Radial basis functions	Colak and Qahwaji (2007) and Qahwaji and Colak (2007, 2006)
RandomForests	Domijan, Bloomfield, and Pitié (2019), C. Liu et al. (2017) and Florios et al. (2018)
Regression models	Anastasiadis et al. (2017), Muranushi et al. (2015), Yuan et al. (2010), H. Song et al. (2009) and J.-Y. Lee et al. (2007)
Relevance-vector machine (RVM)	Al-Ghraibah, Boucheron, and McAteer (2015)
The least absolute shrinkage and selection operator (LASSO)	Benvenuto et al. (2018) and Jonas et al. (2018)
Superposed epoch analysis	Mason and Hoeksema (2010)

Support-vector machines (SVM)	Domijan, Bloomfield, and Pitié (2019), Alipour, Mohammadi, and Safari (2019), Díscola Jr. et al. (2019, 2018a,b), Florios et al. (2018), Inceoglu et al. (2018), Sadykov and Kosovichev (2017), Nishizuka, Sugiura, Kubo, Den, Watari, et al. (2017), Raboonik et al. (2016), Bobra and Couvidat (2015), Muranushi et al. (2015), Yang et al. (2013), Yuan et al. (2010), R. Li, Cui, et al. (2008) and Qahwaji and Colak (2007)
Support-vector machine regressor (SVR)	Boucheron, Al-Ghraibah, and McAteer (2015)
Unsupervised fuzzy clustering	Benvenuto et al. (2018)
Unsupervised learning vector quantization	R. Li, H. Wang, et al. (2011)

Usually, the approaches mentioned earlier aimed to improve some reference performance score, find new promising features associated with flare occurrence or increase how far in advance solar flares could be forecast. Despite most times succeeding in achieving those goals, most of them share a common aspect: using statistical or machine learning techniques to design their classifiers (CAMPOREALE, 2019). Machine learning is a recurrent research branch of Computer Science whose techniques we use to learn from historical data and forecast future observations (HAN; KAMBER, 2006).

1.1 Motivation

In the past decade, a plethora of machine learning methods has emerged along with their corresponding applications. However, a notable barrier for new users comprehends the performance of those methods as they are very sensitive to design decisions (HUTTER; KOTTHOF; VANSCHOREN, 2019).

Efficient decision-making is paramount in the machine learning domain, where engineers urge to design correct algorithms' architectures, training procedures, input features, and

hyperparameters to leverage their expected forecast performance. Accordingly, the design of classifiers comprehends a repeated process, often leading experts to be stuck into tedious trial-and-error episodes until they reach a reasonable set of choices for a particular dataset (HUTTER; KOTTHOF; VANSCHOREN, 2019).

Within this context, a new research branch of machine learning has emerged to support the decisions mentioned earlier, that is, a domain in which algorithms make decisions through an automated, data-driven, and objective way. As such, users can simply provide their input data, and the automated machine learning process fully determines the best performing forecast approach for that particular case. Besides, automated machine learning provides state-of-the-art learning methods to researchers interested in applying concepts rather than knowing the technologies in their details (HUTTER; KOTTHOF; VANSCHOREN, 2019).

Accordingly, automated machine learning approaches aim to automatically design classifiers with improved performance while saving a considerable amount of time and money, as experts in machine learning can sometimes be expensive or hard to find (HUTTER; KOTTHOF; VANSCHOREN, 2019). Overall, various automated machine learning frameworks have been proposed. The most remarkable examples include: Auto-WEKA (KOTTHOFF et al., 2019), Hyperopt-Sklearn (KOMER; BERGSTRA; ELIASMITH, 2019), Auto-sklearn (FEURER; KLEIN, et al., 2019), Auto-net (Auto-PyTorch) (MENDONZA et al., 2019; ZIMMER; LINDAUER; HUTTER, 2020), Tree-Based Pipeline Optimization Tool (TPOT) (OLSON; MOORE, 2019), Auto-keras (JIN; SONG, Q.; HU, X., 2018), RoBO (KLEIN et al., 2017), H2O AutoML (LEDELL; POIRIER, 2020), and AutoGluon-Tabular (ERICKSON et al., 2020). At this point, it is worth noting that they did not hold a design process adhering to the most recurrent and worth considering aspects authors deal with when designing classifiers for solar flare forecasting (we shall discuss all the aspects constituting an effective space weather forecasting project in Chapter 3).

However, the space weather field also needs the automation provided by such tools. The solar flare forecasting efforts mentioned earlier often had tailor-made methodologies, that is, authors proposed them for their particular *ad hoc* scenarios. *Ad hoc* decisions and processes prevent the creation of generic design pipelines to produce flare forecasting classifiers with flexibility and optimized performance – and general space weather approaches for events other than solar flares (i. e., CMEs and solar wind).

In fact, to date, some proposals attempted to create tools to automate the design of classifiers for space weather events. To illustrate, we can mention the researches by Muranushi et al. (2015), Leka, Barnes, and Wagner (2018), Massone and Piana (2018), Engell et al. (2017), Garciá-Rigo et al. (2016), and Anastasiadis et al. (2017). However, once more, it is worth noting that they could not fulfill all the premises we envisioned for an optimized design process, and presented several design restrictions (as we shall further discuss in Chapter 3). By an optimized design, we meant to design classifiers not incurring in most negative aspects authors of *ad hoc* proposals usually encompassed.

1.2 Aims and scope

Aware of the benefits of automating the design of machine learning classifiers and the space weather needs, we proposed in this thesis a comprehensive pipeline of machine learning processes focused on the design optimization of solar flare classifiers. Accordingly, we proposed a novel general methodology (framework) to design, train, and evaluate flare forecasting systems with flexibility and optimized performance. We aimed to create a flexible pipeline of processes relying on good practices for space weather. Besides, to ease the classifiers' design, we fully automated this methodology under Python's Scikit-learn framework.

To validate the proposed methodology, we employed it in three case studies focused on solar flare forecasting. We defined the processes comprehending it after an in-depth look at the literature state-of-the-art, that is, what issues researchers often encompassed when designing solar flare predictors – for instance, leveraging the performance while reducing false alarms, dealing with imbalanced data, discarding useless features, adjusting hyperparameters to avoid over-fitting, among others. Besides, we also deployed the classifiers from those case studies into a real-time forecasting environment, namely the Guaraci system.

1.2.1 Target audience

Noteworthily, space weather data comprehend a compendium of multimission and multidisciplinary data sources to monitor the heliosphere. That turns the Sun's regions and the near-Earth space of particular interest to space weather forecasters.

In this sense, a needed milestone is the capability to supply data acquisition systems at strategic points of near-Earth regions to provide real-time data to evaluate the space weather.

Accordingly, we have several instruments continuously monitoring the space weather on one hand – such as the GOES satellites with tools for X-ray imaging and recording. On the other hand, recorded big data analysis requires complex techniques and algorithms – not rarely, machine learning-based ones.

Provided that the community can better understand those algorithms, it would certainly produce better solutions for solar physics. Such an audience – and especially solar physicists working with solar flare forecasting – is the target audience of this research.

In fact, the advances in machine learning achieved in the last decade – and particularly the ability to train very deep and complex neural networks more recently – can greatly improve the performance when forecasting solar flares (NISHIZUKA; SUGIURA; KUBO; DEN; ISHII, 2018). However, one must bear in mind that not all solar physicists can naturally transit between the artificial intelligence domain and physics. In this sense, we believe that providing a self-contained tool for physicists to design forecast systems with performance and flexibility would shorten the gap between them and the desired artificial intelligence knowledge. That is what we intend with this thesis.

1.3 Thesis organization

We divided this thesis into seven regular chapters, and three appendices. In Chapter 2, we provide a theoretical reference of concepts needed for better understanding the content discussed in this thesis. In Chapter 3, we further detail the literature state-of-the-art of solar flare forecasting efforts. In Chapter 4, we explain the proposed methodology, providing details of each stage, and employed techniques. Chapter 5 discusses the datasets employed in our case studies, as well as our flare catalogs and features. In Chapter 6, we underlie the results of case studies I, II, and III, and show how the methodology has improved the performance of forecasting systems, comparing results to the specialized literature. Finally, in Chapter 7, we underlie the conclusions of this research and comment on future work. Regarding the supporting content of this thesis, Appendix A further discusses the scores employed for assessing the classifiers' performance. In turn, appendixes B and C present the detailed tables of our case studies' results and the Guaraci system, respectively.

Chapter 2

Theoretical reference

This chapter discusses some supporting concepts for this thesis. First and foremost, we shall introduce the machine learning domain under a data mining perspective, underlying the concepts comprehending how learning algorithms learn from and represent historical data. Within such a perspective, we shall consider the classification learning task specifically. We will then move our focus to further explanations of the most recurrent Sun's events affecting the space weather, namely the sunspots, ARs, and solar flares.

2.1 Machine learning and data classification

As stated by Fayyad, Piatetsky-Shapiro, and Smyth (1996), the Knowledge Discovery in Databases (KDD) domain comprehends a non-trivial process to extract implicit, previously unknown, and potentially useful information from data stored in a wide range of data sources. The KDD process encompasses a well-defined pipeline of actions (HAN; KAMBER, 2006):

1. Data cleaning: noise and inconsistency removal;
2. Data integration: integration of data from multiple sources;
3. Data selection: a selection of only relevant data for analysis;
4. Data transformation: data transformation into proper formats;
5. Data mining: mining data to discover hidden patterns (use of learning algorithms);
6. Pattern evaluation: discover useful patterns representing the desired knowledge;

7. Knowledge representation: use of techniques to present and represent the extracted knowledge.

During data mining, the idea is to build computer programs that automatically analyze databases and seek regularities or patterns. Whether the algorithms find strong patterns, they will likely learn better and shall be able to make better generalizations (accurate predictions) on future data (WITTEN; FRANK; HALL, 2011). To carry out data mining, KDD often borrows techniques and methods intersecting between distinct domains, including (ZAKI; MEIRA JR., 2013):

- Machine learning: supervised, unsupervised, and by reinforcement methods;
- Relational databases: Data Warehousing (DW) and Online Analytical Processing (OLAP);
- Statistics;
- Knowledge engineering;
- Data visualization.

Not rarely, data mining will find issues in data. Many patterns can be banal or not interesting. In turn, others can be spurious or contingent on accidental coincidences within a particular dataset. Moreover, some parts of the data can be garbled or even missing, as well as exceptions to the generalization rules can be encountered. As such, algorithms must be robust enough to cope with such imperfect data (WITTEN; FRANK; HALL, 2011).

Because of its broad applicability, machine learning often provides alone the technical basis for data mining. Usually, the kind of descriptions found in data mining can be used to predict future events (i. e., the classification task – employing supervised machine learning algorithms) or explain/understand historical data (i. e., the clustering task – employing unsupervised machine learning algorithms) (WITTEN; FRANK; HALL, 2011). This thesis focused on the classification task, mostly using supervised techniques.

The classification task assign items in a collection to their corresponding classes (categories). As such, its goal relies on accurately predicting the target class of new cases of data. For instance, a typical classification problem would be to fit an algorithm to classify the credit risk of new loan applicants (i. e., low, medium, or high) (HAN; KAMBER, 2006).

The most straightforward classification problem is binary. In binary classification, the target assumes two possible classes (i. e., in case of flares, "yes" or "no" for occurring events). On the other hand, multi-class targets have distinct categories (i. e., in the case of flares, the class of each event) (HAN; KAMBER, 2006).

During the design process (training), a classification algorithm uses data to find relationships between the input features and the target values. Machine learning algorithms differ in their techniques when seeking such relationships. After training, the algorithm summarizes the relationships it found into a forecast model, which can be employed later into a different dataset, that is, a new set of records in which we do not know the class assignments (ZAKI; MEIRA JR., 2013).

We test the trained classification model by comparing the predicted values to the test data's known values. We usually divide the historical data from a classification project into two parts: one for training and the remaining for testing (ZAKI; MEIRA JR., 2013).

To design machine learning based prediction models, we must first provide them with training tuples and their corresponding classes (labels) representing the existence of some specific event. We define a tuple X as a n-dimensional array, $X = [x_1, x_2, \dots, x_n]$, referring to n measures from n attributes $A_1, A_2, \dots, A_n$ (features). Each tuple belongs to a predefined target class, which assigns meaning to the measures from n attributes. During testing, the trained model acts as a function $y = f(X)$, thus forecasting the class y of a given test tuple X (ZAKI; MEIRA JR., 2013).

The output of most classification models comprehends the class assignments and, additionally, the probabilities for each class. For instance, the model forecasting the occurrence of flares as "yes" or "no" would also predict the probability of each assignment. Not rarely, outputting the class probabilities allows the thresholding of answers. By default, we calculate the class assignments with a threshold of 0.5 (i. e., probabilities higher than or equal to 0.5 belong to the "yes" class). However, such a default threshold might not yield the best performance regarding the number of false alarms, which turns the threshold correcting an activity worth performing (i. e., to adjust the precision-recall trade-off) (ZAKI; MEIRA JR., 2013).

2.2 Solar events and structures

The Sun's events and structures often affecting the space weather include sunspots (MOLDWIN, 2008), active regions (ARs) (CANFIELD, 2001), and solar flares (MESSEROTTI et al., 2009). We discuss such elements supporting the forecast models analyzed in this thesis next.

2.2.1 Sunspots

Sunspots are dark phenomena related to intense – and sometimes complex – magnetic fields in the photosphere. Not rarely, they can spread along the Sun's atmosphere (chromosphere and corona). Their colors refer to their lower temperatures compared to the photosphere (MOLDWIN, 2008).

Sunspots can last for a few hours to months. Regarding their forms, they can vary a lot, ranging from small dots to complex and giant representations. Their diameters can extend over a range from 300 to 100,000 km (almost eight times the Earth's diameter) (MOLDWIN, 2008).

Surrounded by a lighter region – the penumbra –, the umbra of a sunspot – its center – is a dark-colored area (MOLDWIN, 2008). Figure 2.1 highlights the umbra and penumbra of a sunspot in an intensity diagram taken by the Michelson Doppler Imager (MDI) instrument aboard the Solar and Heliospheric Observatory (SOHO) (SCHERRER et al., 1995).

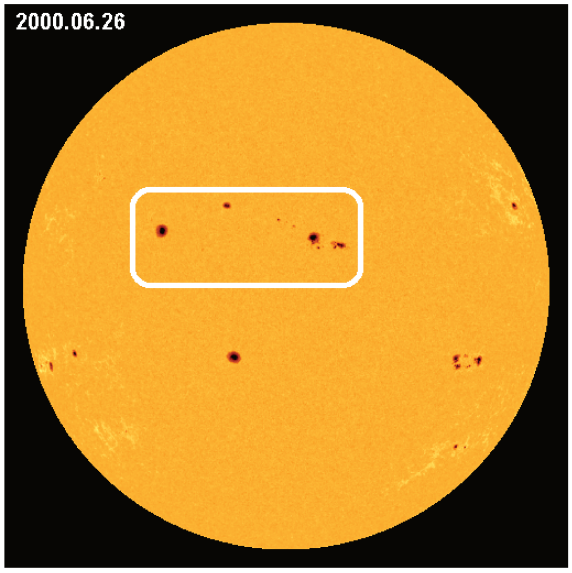

Figure 2.1: MDI intensity diagram showing sunspots. Intensity diagrams represent the Sun's brightness through light colors, with the sunspots' umbras and penumbras in dark and orange tones, respectively.

2.2.2 Active regions

In the Sun's atmosphere, active regions are extensions to sunspots' and groups of sunspots' magnetic fields observed in the photosphere (CANFIELD, 2001). ARs increase their activity levels as the number of sunspots in their composition increases. Figure 2.2 highlights an AR through a MDI magnetogram. MDI magnetograms represent the Sun into three distinct colors: dark (negatively polarized areas), white (positively polarized areas), and grey (areas with neutral polarity).

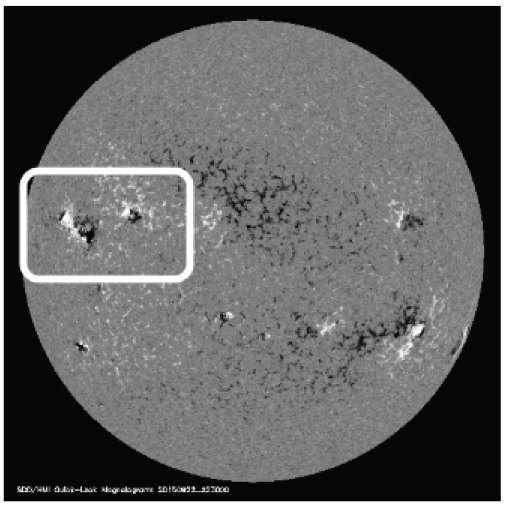

Figure 2.2: MDI magnetogram showing some ARs along with the Sun's magnetic fields. MDI magnetograms show the Sun's surface into the dark (areas with negative polarity), white (areas with positive polarity), and grey colors (areas with neutral polarity).

To classify and distinguish the magnetic configuration of ARs, solar physicists employ two distinct taxonomies: McIntosh (1990) and Mt. Wilson (HALE et al., 1919). Whereas the former classifies ARs regarding the forms of their composing umbras and penumbras, the latter further details the magnetic fields inside them:

- *Alpha*: a group of unipolar sunspots;
- *Beta*: a group of bipolar sunspots that has a simple division line between positive and negative polarities;
- *Gamma*: a group of sunspots that has positive and negative polarities irregularly distributed, in such a way it is not possible to define the group as bipolar;
- *Delta:* a complex group comprehending umbras with opposite polarities inside the same penumbra;

- *Beta-gamma*: a group of bipolar spots; however, complex enough in such a way it is not possible to draw a dividing line to separate the spots with opposite polarities;
- *Gamma-delta*: a group of spots mostly represented by the gamma class; however, there is also at least one delta class spot;
- *Beta-delta*: a group of spots mostly represented by the beta class; however, with at least one delta class spot;
- *Beta-gamma-delta*: a group of spots mostly represented by the beta-gamma class; however, with at least one delta class spot.

On the other hand, the McIntosh taxonomy describes ARs' forms through three distinct components (*Zpc*) (MCINTOSH, 1990):

- Also known as the modified Zurich class, the first component (Z) describes whether a penumbra is present, how it is distributed, and the length of the group. It draws values from the following scale:
  - *A*: unipolar group containing no penumbra (it represents the final or formative stage of evolution within a group);
  - *B*: bipolar group holding no penumbra on any spot;
  - *C*: bipolar group with penumbra on one end of the group;
  - *D*: bipolar group with length ≤ 10°, containing penumbra on spots at both ends of the group;
  - *E*: bipolar group with 10° < length ≤ 15°, containing penumbra on spots at both ends of the group;
  - *F*: bipolar group with length > 15°, containing penumbra on spots at both ends of the group;
  - *H*: unipolar group holding penumbra.
- Also known as the penumbral class, the second component (p) describes the type of the largest spot in a group along with the type, size, symmetry, and umbra of its penumbra. It draws values from the following scale:
  - *x*: the penumbra does not exist;

  - *r*: a rudimentary (incomplete, granular, and brighter) penumbra partially surrounds the largest spot;
  - *s*: the largest spot has a mature, dark, and filamentary penumbra with no irregularities on its borders, which have elliptical or circular shape – its North-South diameter is ≤ 2.5°;
  - *a*: the penumbra of the largest spot is irregular in outline and its North-South diameter is ≤ 2.5°;
  - *h*: the penumbra has the same structure as type *s*, but its North-South diameter is > 2.5°;
  - *k*: the penumbra has the same structure as type *a*, but its North-South diameter is > 2.5°.

- Also known as the compactness class, the last component (c) describes the interior spot distribution inside a sunspot group. It draws values from the following scale:
  - *x*: undefined for unipolar groups;
  - *o*: there are only few spots between the leader and follower (the interior of spots are of very small size);
  - *i*: there are numerous spots lying between the leading and following portions of the group (none of them has a mature penumbra);
  - *c*: many strong spots populate the area between the leading and following ends of the spot group (at least one spot possesses a mature penumbra).

2.2.3 Solar flares

Mostly related to ARs, solar flares comprehend sudden releases of electromagnetic radiation – including X-rays in the 1–8 Å wavelength, whose intensities we represent in W/m^2. Such events can reach temperatures between 10 – 20 million Kelvin. They can affect the Earth's atmosphere in a few minutes/hours and are considered one of the most powerful phenomena in the solar system (CANFIELD, 2001). Figure 2.3 shows a solar flare in an extreme ultraviolet MDI image.

The frequency of solar flares can vary over time, which means that several events can occur in a short period, or weeks to months can pass without their occurrence (MOLDWIN,

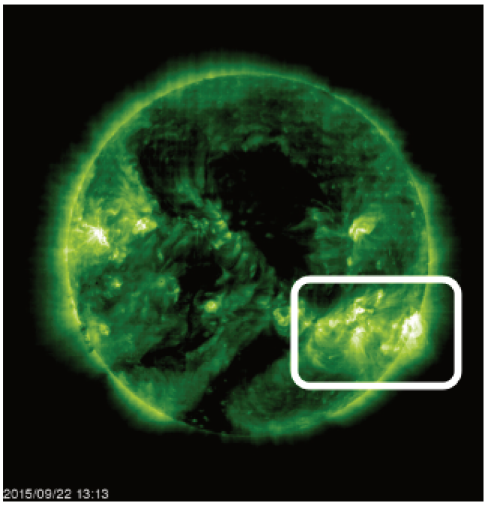

Figure 2.3: Extreme ultraviolet MDI image showing a solar flare.

2008). The Sun emits radio energy with varying intensity. By originating from atmospheric layers high in the Sun's chromosphere and low in the corona, this flux changes gradually from day-to-day – as a direct response to the number of spot groups on the disk. As an indicator of solar activity, such radio flux can be used to show the solar cycle, i. e., the peaks of solar activity, in which more flares are released (Figure 2.4).

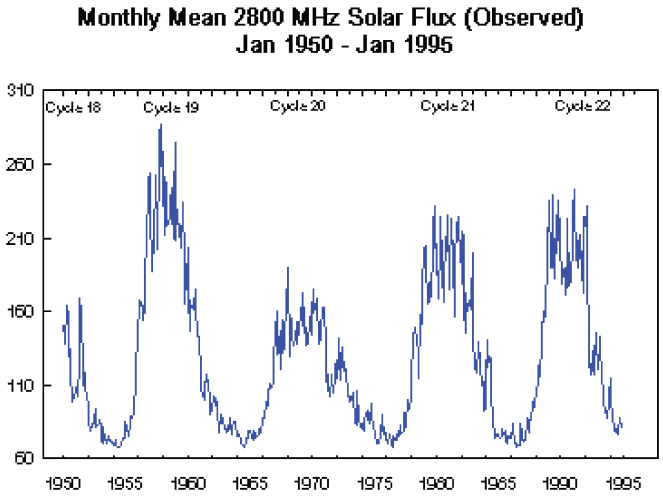

Figure 2.4: Solar cycle (Monthly Mean 2800 Mhz Solar Flux). Source: `https://www.ngdc.noaa.gov/stp/solar/flux.html`

To represent the magnitude of flare events, solar physicists classify them through a labeled scale ranging between A-, B-, C-, M-, and X-class (indicated in Table 2.1). X-class flares ($> 10^{-4}$ W/m^2) are the strongest events, whereas A-class ones ($< 10^{-7}$ W/m^2) are the weakest. Each class of flare has its X-ray peak flux ten times higher than its predecessor.

Table 2.1: Solar flare classes.

Class	*Peak flux 1-8 Ångstrom (W/m^2)*
A	$< 10^{-7}$
B	10^{-7} to 10^{-6}
C	10^{-6} to 10^{-5}
M	10^{-5} to 10^{-4}
X	$> 10^{-4}$

Besides, each flare class also ranges over the linear scale [1 , 9] representing the event's intensity. Hence, to report flares, we multiply their class peak values by their corresponding intensity factors, i. e., X2, M5, and X6. To illustrate the occurrence of those events, Figure 2.5[1] shows a time-series with three days of GOES X-ray data (temporal resolution of five minutes) and some highlighted events (X2, M5, and X6). The red curve corresponds to the 1–8 Å wavelength.

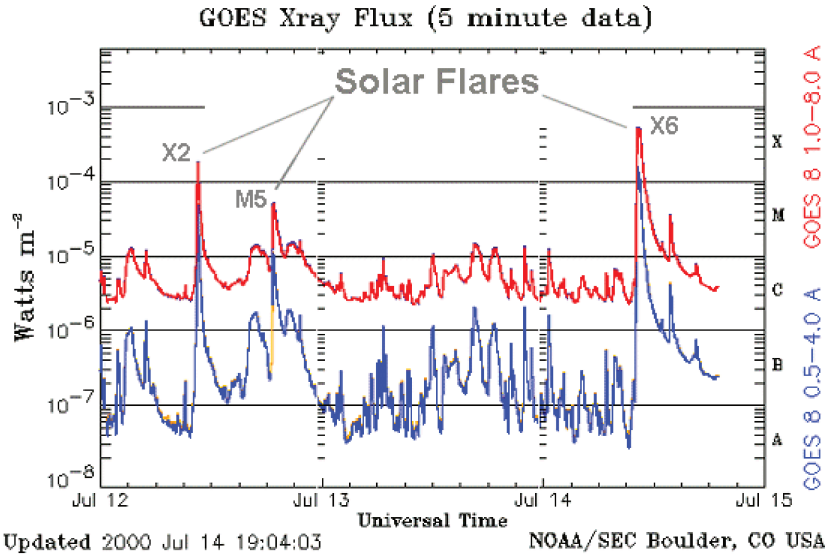

Figure 2.5: GOES X-ray five minute data indicating three solar flares, namely X2, M5, and X6. Source: `http://spaceweather.com/glossary/flareclasses.html`.

2.3 Concluding remarks

This chapter presented some supporting concepts for this thesis, namely the machine learning research domain and its data classification perspective, as well as the most recurring solar events. We believe such discussed concepts should provide a clearer understanding of our

[1] Such a figure represents many X-ray peaks (solar flares). However, there are only three highlighted events.

literature survey in Chapter 3, when we introduce and analyze state-of-the-art solar flare forecasting systems, their underlying characteristics, and behavioral aspects.

Chapter 3

Literature survey

This chapter shall give an in-depth look at the literature state-of-the-art solar flare forecasting systems, as well as some automated machine learning approaches. In the beginning, this chapter introduces a comprehensive number of forecast models for space weather whose design processes have not focused on constant reproducing, i. e., *ad hoc* proposals. Our focus then moves to the presentation of several systems proposed to automate and ease the design of general machine learning forecast models, not forgetting to include some approaches specifically created for space weather's needs.

3.1 Solar flare forecasting efforts

To detail the forecast models relying on *ad hoc* methodologies, we shall describe how researchers designed them, emphasizing how they assembled their datasets and features, and estimated their prediction errors.

To report systems' performance, we shall use some well-known machine learning scores borrowed from binary deterministic-based forecast scenarios (i. e., forecasting whether flares shall happen or not within some given period), such as the True Skill Statistics (TSS)[1]. The TSS ranks the performance of learning models over a scale lying on $[-1\ ,\ 1]$: values next to -1 mean a majority of incorrect predictions, whereas near 1, a majority of correct predictions.

Most scores could not represent two outcome scores simultaneously. However, TSS successfully holds in its calculation two components for individually assessing the forecast success rates of the positive (True Positive Rate (TPR); Han and Kamber (2006)) and negative

[1]For the mathematical formulation, see Appendix A.10 and the article from Jolliffe and Stephenson (2003).

(True Negative Rate (TNR); Han and Kamber (2006)) outcome classes at once (see appendixes A.4 and A.5 for further details involving TPR and TNR, respectively).

Roughly speaking, TPR accounts for the number of positive samples correctly forecast, whereas TNR is the number of negative ones (HAN; KAMBER, 2006). As such, we shall also include both individual hit rates while analyzing literature. Besides, we will detail the Accuracy (ACC) (see Appendix A.3; Han and Kamber (2006)) of systems and ratios of predicted false alarms (False Alarm Ratio (FAR), Appendix A.6) (JOLLIFFE; STEPHENSON, 2003).

As argued in Barnes and Leka (2008)'s, Barnes, Leka, et al. (2016)'s, Leka, S.-H. Park, et al. (2019b)'s, and Leka, S.-H. Park, et al. (2019a)'s researches, unless the datasets of two distinct forecasting efforts are identical, directly comparing their metrics is meaningless. As such, characteristics preventing direct comparisons involve distinct splitting approaches for training, validating, and testing data, how researchers designed their target features (forecasts), the type of prediction (for instance, whether systems analyze the Sun's full-disk or provide forecasts for each observed AR at some period in time, that is, AR-by-AR), among others.

The characteristics, as mentioned earlier, introduce uncertainty when directly comparing scores from distinct learning models. In fact, it is not clear whether comparing differences occur in response to the methods used (i. e., performance merit) or to the underlying differences of datasets (i. e., sets of data easier to classify) (BARNES; LEKA, 2008; BARNES; LEKA, et al., 2016). Accordingly, results commented herein should not by any means be considered direct comparisons of scores between distinct forecasting approaches.

Overall, for scope definition purposes, we included in our literature research learning models relying on the following criteria:

- Systems able to forecast ≥ C- (events of C-class and above) and ≥ M-class flares (events of M-class and above).
- Systems designed to forecast events in the next 24 h, 24 h – 48 h, and 48 h – 72 h. Besides, we also sought overlapping forecasting horizons, that is, forecasting events in the next 48 and 72 h.
- Systems relying on features of mixed origin (i. e., photospheric magnetic data, ARs' photospheric features, among others).
- Systems relying on algorithms of mixed nature (i. e., SVMs, ensembles, DecisionTrees, statistical predictors, among others).

- Systems outputting AR-by-AR or full-disk forecasts.
- Literature focusing on the period between 2010 and 2020 July. Our aim is not to disregard the literature of longer periods. However, our focus is an in-depth look at the state-of-the-art, instead. Nevertheless, we may also include some seminal articles.
- In addition to system forecasts, we shall also include human-based ones from prediction centers.

Besides, while analyzing the literature, we managed to distinguish systems between two distinct groups: systems posing biased results or not. Biased results link to systems holding some bias in their reported performance.

In fact, biased results are not wrong. However, they represent systems evaluated in scenarios that would mask, to some extent, their generalization skills in real operational settings. Henceforth, let us acknowledge the following four criteria to distinguish between biased and unbiased results:

1. *Evaluation with truly unseen data*: as posited in Ahmed et al. (2013)'s research, removing samples from the training process gives a true estimate of systems' performance while forecasting real test data, i. e., unseen samples. They proposed to yearly split their dataset into training and testing data, thus reserving some years for designing models and others to validate them as if they were forecast in a real operational environment. Accordingly, articles not explicitly using unseen data to report their results shall be classified as biased.
2. *Availability of enough data for model design*: some authors do not include enough examples to fit their models, thus incurring in under-fitting. Learning models relying on less representative datasets can have their generalization skills over unseen samples negatively affected. Within this context, increasing datasets' size allows finer discrimination between input features, positively leveraging their generalization skills (PYLE, 1999). As such, articles lacking enough data shall be classified as biased.
3. *Use of ARs' data near the disk center only*: many researchers discard magnetograms with ARs far from the disk center when designing models, that is, beyond a defined radius. However, as posited in Nishizuka, Sugiura, Kubo, Den, Watari, et al. (2017)'s research, their reported scores raise uncertainty and weaken results interpretation for operational purposes since, in real operational environments, their systems would have to behave

with ARs at any location in the disk. Therefore, those proposals shall also be classified as biased.

4. *Use of ARs' data linked to ≥ C1.0 flares only when forecasting ≥ M-class events*: many researchers only handle magnetograms with ARs producing ≥ C1.0 flares and distinguish samples linked to ≥ M1.0 events as the positive class. However, as posited in Bloomfield et al. (2012)'s research, their reported scores also raise uncertainty and weaken results interpretation for operational purposes, since in real operational environments, their systems would have to behave with ARs not linked to any type of flare. Hence, we shall classify those proposals as biased.

3.1.1 ≥ C-class flare forecasting

Table 3.1 shows our literature survey involving ≥ C-class flares forecasting approaches. We divided this table into three parts: the upper-hand part represents biased models, the middle-hand has the unbiased ones, and the lower-hand includes human-based forecasts from space weather prediction centers.

Biased results

By sampling Helioseismic and Magnetic Imager (HMI) magnetograms (BOBRA; SUN, et al., 2014) with a one-hour cadence from the period between 2011 and 2012, Muranushi et al. (2015) processed the Sun images to create some scalar time series using wavelet analysis. They then attached labels to their data samples for predicting ≥ C-, ≥ M-, and X-class flares in the next 24 h based on GOES soft x-ray measures.

Concerning ≥ C flare forecasting, the linear regressor from Muranushi et al. (2015) scored a TSS of 0.63 (FAR = 0.07). Despite the positive results (both TPR and TNR trespassed 0.80), the authors only chose samples of magnetograms with ARs within 69° of the solar disk center, and thus we classified their results as biased.

For only choosing magnetograms with ARs within 30° of the solar disk center, we also classified Huang, H. Wang, Xu, et al. (2018)'s study as biased. They aimed to propose a novel forecasting approach based on a proxy dataset combining data from MDI magnetograms (SCHERRER et al., 1995) and HMI images. Accordingly, they designed a convolutional neural

Table 3.1: ≥ C-class flare forecasting models.

Forecasting time	*Authorship*	*Grouping*	*ACC*	*TPR*	*TNR*	*TSS*	*FAR*
next 24 h	Muranushi et al. (2015)[a]	biased results	0.81	0.80	0.83	0.63	0.07
next 24 h	Huang, H. Wang, Xu, et al. (2018)[b]	biased results	0.76	0.73	0.76	0.49	0.65
next 48 h	Huang, H. Wang, Xu, et al. (2018)[b]	biased results	0.81	0.81	0.81	0.62	0.84
next 24 h	Ahmed et al. (2013)[c]	biased results	0.96	0.52	0.98	0.53	0.25
next 24 h	Yang et al. (2013)[d]	biased results	0.76	0.61	0.84	0.47	-
next 24 h	Domijan, Bloomfield, and Pitié (2019)[e]	biased results	0.89	0.95	0.89	0.84	-
next 24 h	Domijan, Bloomfield, and Pitié (2019)[f]	biased results	0.81	0.87	0.80	0.67	-
next 24 h	H. Liu et al. (2019)[g]	biased results	0.82	0.76	0.84	0.60	-
next 24 h	Florios et al. (2018)[h]	biased results	0.84	-	-	0.60	-
next 24 h	X. Wang et al. (2020)[i]	biased results	0.88	0.63	0.93	0.56	0.36
next 24 h	X. Li et al. (2020)[j]	biased results	0.86	0.88	0.79	0.67	0.09
next 24 h	Anastasiadis et al. (2017)[k]	biased results	-	-	-	0.25	-
next 24 h	Nishizuka, Sugiura, Kubo, Den, and Ishii (2018)[l]	unbiased results	0.82	0.81	0.82	0.63	0.47
next 24 h	Hada-Muranushi et al. (2016)[m]	unbiased results	0.66	0.72	0.57	0.30	0.30
next 24 h	Colak and Qahwaji (2009)[n]	unbiased results	0.81	0.81	-	-	0.30
next 24 h	Leka, Barnes, and Wagner (2018)[o]	unbiased results	0.75	0.69	0.82	0.51	0.17
24 h – 48 h	Leka, Barnes, and Wagner (2018)[o]	unbiased results	0.77	0.71	0.83	0.55	0.16
48 h – 72 h	Leka, Barnes, and Wagner (2018)[o]	unbiased results	0.71	0.60	0.85	0.45	0.17
next 24 h	Leka, Barnes, and Wagner (2018)[p]	unbiased results	0.92	0.30	0.99	0.29	0.30
24 h – 48 h	Leka, Barnes, and Wagner (2018)[p]	unbiased results	0.93	0.27	0.99	0.26	0.25
48 h – 72 h	Leka, Barnes, and Wagner (2018)[p]	unbiased results	0.94	0.27	0.99	0.26	0.30
next 24 h	Bloomfield et al. (2012)[q]	unbiased results	0.71	0.75	0.70	0.45	0.64
next 24 h	E. Park et al. (2018)[r]	unbiased results	0.82	0.85	0.78	0.63	0.17
next 24 h	Benvenuto et al. (2018)[s]	unbiased results	0.83	0.53	0.89	0.43	0.47
next 24 h	Díscola Jr. et al. (2018a)[t]	unbiased results	0.72	0.70	0.79	0.49	-
next 24 h	Díscola Jr. et al. (2018b)[u]	unbiased results	0.91	0.94	0.86	0.80	-
next 24 h	Crown (2012)[v]	human-based forecasts	0.90	0.63	0.94	0.57	0.40

[a] Scores calculated over the confusion matrix of Table 5. Although Muranushi et al. (2015) proposed an automated space weather design approach (as we shall discuss in Section 3.2.1, such scores correspond to a reported case study.
[b] Scores calculated over the confusion matrix of Table 4 (HUANG; WANG, H.; XU, et al., 2018).
[c] Scores collected from Table 6 (training and testing columns: operational set up). TSS calculated over TPR and TNR (AHMED et al., 2013).
[d] Scores collected from Table 4 (YANG et al., 2013).
[e] Scores collected from Table 4 (DOMIJAN; BLOOMFIELD; PITIÉ, 2019) ($p = 0.05$).
[f] Scores collected from Table 7 (DOMIJAN; BLOOMFIELD; PITIÉ, 2019) ($p = 0.20$).
[g] Scores collected from Table 3. Results refer to the LSTM entry. TNR calculated over TPR and TSS (LIU, H. et al., 2019).
[h] Scores collected from the conclusions of page 27 (FLORIOS et al., 2018).
[i] Results computed over the confusion matrix of Table 5. Results refer to the LSTM-15_18 model (WANG, X. et al., 2020).
[j] Results collected from Table 2. TNR computer over TPR and TSS (LI, X. et al., 2020).
[k] Approximated TSS and TPR values from the graph of Figure 8 (TSS peak at a threshold of 0.40). Although Anastasiadis et al. (2017) proposed an automated space weather design approach (as we shall discuss in Section 3.2.1, such TSS and TPR correspond to a reported case study.
[l] Scores calculated over the confusion matrix of Figure 5 (NISHIZUKA; SUGIURA; KUBO; DEN; ISHII, 2018).
[m] Scores calculated over the confusion matrix of Table 5 (HADA-MURANUSHI et al., 2016).
[n] Scores collected from Table 3 (COLAK; QAHWAJI, 2009).
[o] Scores calculated over the confusion matrix of Fig. 13 (full-disk forecasts). Although Leka, Barnes, and Wagner (2018) proposed an automated space weather design approach (as we shall discuss in Section 3.2.1, such scores correspond to a reported case study.
[p] Scores calculated over the confusion matrix of Fig. 13 (active-region-based forecasts). Although Leka, Barnes, and Wagner (2018) proposed an automated space weather design approach (as we shall discuss in Section 3.2.1, such scores correspond to a reported case study.
[q] Scores collected from Table 4. TNR calculated over TPR and TSS (BLOOMFIELD et al., 2012).
[r] Scores collected from Table 3 (Model 3). TNR calculated over TPR and TSS (PARK, E. et al., 2018).
[s] Scores calculated over the confusion matrix of Table 1 (Hybrid entry) (BENVENUTO et al., 2018).
[t] TSS calculated over TPR and TNR. Results collected from Fig. 60.2 and 60.3 (DÍSCOLA JR. et al., 2018a).
[u] TSS calculated over TPR and TNR. Scores collected from page 13 (DÍSCOLA JR. et al., 2018b).
[v] Scores calculated over the confusion matrix of Table 4 (CROWN, 2012).

network (CNN) over MDI data from 1996 to 2010 and used HMI data from 2010 to 2015 for testing.

Although Huang, H. Wang, Xu, et al. (2018) have proposed several forecasting horizons (the next 6, 12, 24, and 48 h) and flare importance thresholds (C-, M-, and X-class events), we only considered the scenarios adhering to this study, namely predicting ≥ C flares in the next 24 and 48 h. For those cases, their model scored TSS = 0.49 (FAR = 0.65) and 0.62 (FAR = 0.84), respectively for 24 and 48 h.

Another research whose authors also included only MDI samples of magnetograms within a defined radius (45°) is the one by Ahmed et al. (2013). They aimed to assess the flare-prediction potential of what they defined as MF properties, i. e., sets of magnetic features automatically tracked and extracted from MDI images by their Solar Monitor Active Region Tracker (SMART).

Ahmed et al. (2013) proposed two strategies for defining positive events for their cascade correlation neural network: segmented and operational. They classified segmented samples as flaring if they produced at least one C-, M-, or X-class flare in the following 24 h, and non-flaring if they did not produce any of those events in the following 48 h. On the other hand, they defined operational samples – the scenario we are referencing in Table 3.1 – as positive exactly in the same way as segmented ones. However, the non-flaring flag was attached to MF properties not producing any flare in the following 24 h.

To evaluate their model, Ahmed et al. (2013) divided their samples into years of training (1996 April to 2000 December, and 2003 January to 2008 December) and testing data (2001 January to 2002 December, and 2009 January to 2010 December). Despite the identified bias of filtering ARs far from the disk center, they scored TSS = 0.53 at the expense of forecasting false alarms one-fourth of the time.

Based on data from the Solar Magnetic Field Telescope located at the Huairou Solar Observing Station in China, Yang et al. (2013) chose several photospheric magnetic features, including the mean planar magnetic shear angle, the vector magnetic field mean shear angle and the mean absolute vertical current density. They then proposed a support vector classifier with tenfold cross-validation to assess their forecasting performance. Since in this research the authors only used ARs samples locating within 30° from the disk center, we treated their results as biased, despite their TSS = 0.47.

More recently, Florios et al. (2018) presented an approach for investigating the forecast performance with ≥ C-class events using multi-layer perceptrons, support vector machines, and random forests. As such, they based their predictive features on near-real-time HMI data from 2012 to 2016, with all calendar days within this period included in the sample. The cadence of

predictors in the chosen days was three hours. For each AR record, they verified whether it flared within the next 24 h.

Their whole set of data comprised more than 23,100 observations. To investigate their algorithms, they repeated split at random this set into training (50%) and test (50%) samples and applied the same subsets for them. Besides, they assessed the effects of varying the prediction threshold of algorithms within 200 evaluations. They chose their best model – random forest – considering the algorithm peaking its TSS at some particular threshold. As they used test data for decision-making, we considered their approach biased, despite scoring TSS = 0.60.

Aiming a direct comparison with Ahmed et al. (2013)'s research, Domijan, Bloomfield, and Pitié (2019) used the same data period of MF properties as the former, including the year segments for training and testing. Hence, from 1996 April to 2010 December, they assembled two datasets: a full set of MF properties and a filtered one, comprehending only MF properties linked to NOAA ARs (SMART did not associate every MF detection with a valid NOAA AR number; Ahmed et al. (2013)).

Despite Domijan, Bloomfield, and Pitié (2019) had two datasets, both of them only included MF properties tracked within 45° of the disk center. In addition, they treated positive events exactly in the same way as in Ahmed et al. (2013)'s article, i. e., they deemed each MF property as positive whether it did produce C-class events or above within the following 24 h after observation.

Besides two datasets, Domijan, Bloomfield, and Pitié (2019) also carried some feature selection studies with distinct prediction algorithms, namely linear regression, random forest, and SVM. They defined the high-gradient neutral-line length, maximum gradient along polarity inversion line, and neutral-line length as the top three features for both full (TSS = 0.84) and filtered datasets (TSS = 0.67).

In turn, H. Liu et al. (2019) employed custom long-short term memory networks (LSTM) to forecast whether ARs would flare in the next 24 h. The essence of their research was to model AR records as time series and use such networks to capture their temporal information. Each AR sample held 40 features, including the 25 HMI parameters proposed by Bobra and Couvidat (2015) and 15 features to represent the flare history of ARs.

Within this context, H. Liu et al. (2019) surveyed flare events occurring from 2010 May to 2018 May in the GOES X-ray flare catalogs provided by the NCEI, selecting events linked to their ARs samples – that is, the ones representing ≥ M- and X-class events. To avoid the

bias of mixing data during the design of their models, they used disjoint periods: data samples collected from 2010 to 2013 comprehended the training set, the year of 2014 composed their validation set, and the period between 2015 – 2018 was their test set. However, they included ARs samples only within 70° to the central meridian, as suggested by Bobra and Couvidat (2015). Nevertheless, for ≥ C-class forecasting, their biased TSS equaled 0.60.

More recently and also using disjoint periods for designing their classifier – a LSTM – to forecast ≥ C- and M-class events within the next 24 h, X. Wang et al. (2020) have split their HMI AR data from 2011 to 2014 and from 2015 to 2018 for training and test, respectively. As of H. Liu et al. (2019) mostly based their features on Bobra and Couvidat (2015)'s set, so did X. Wang et al. (2020). However, despite their TSS = 0.56, they discarded ARs with records beyond 68° from the disk center.

The remainder of articles for ≥ C flare forecasting including only AR samples near the disk center comprehends the research by X. Li et al. (2020) (45°, TSS = 0.67) and Anastasiadis et al. (2017) (70°, TSS = 0.25).

Unbiased results

Conversely to the previous articles, from 300,000 HMI images between 2010 and 2015 taken by the Solar Dynamic Observatory (SDO), Nishizuka, Sugiura, Kubo, Den, and Ishii (2018) calculated 79 features for each sunspot and attached labels of X-, M-, and C-class flares based on GOES X-ray measures.

For each sunspot, Nishizuka, Sugiura, Kubo, Den, and Ishii (2018) included features proposed by Bobra and Couvidat (2015) and Nishizuka, Sugiura, Kubo, Den, Watari, et al. (2017), along with the coronal hot brightening at 131 Å and the X-ray intensity data linked to 1 and 2 h before images. To evaluate their convolutional neural network, they split their database into years of training and testing data. They used data between 2010 and 2014 for training, and 2015 for testing.

Nishizuka, Sugiura, Kubo, Den, and Ishii (2018) aimed to calculate the flare probabilities of each region involving the occurrence of ≥ C and ≥ M events in the next 24 h after images. For ≥ C flare forecasting, their model scored TSS = 0.63 (FAR = 0.47).

Although we previously classified Muranushi et al. (2015)'s results as biased for only choosing samples of magnetograms with ARs near the disk center, recently, there has been a proposal to deploy an operational system designed by their tool to design forecast models

(HADA-MURANUSHI et al., 2016). Instead of a regression classifier, authors designed a LSTM based on wavelet features calculated from HMI images (TSS = 0.30).

As Nishizuka, Sugiura, Kubo, Den, and Ishii (2018), Hada-Muranushi et al. (2016) also aimed to calculate the probabilities of ≥ C-, ≥ M-, and X-class events occurring in the next 24 h. For predicting ≥ C events, their LSTM scored 0.30 for both TSS and FAR.

Colak and Qahwaji (2009) introduced an operational system for flare forecasting named Automated Solar Activity Prediction (ASAP), which integrated machine learning, solar physics, and image processing to make forecasts based on descriptive characteristics of sunspots. They composed their system of two modules, namely a neural network to predict the flaring probability of ≥ C events and another neural network to specify the class of events when needed.

For designing their models, Colak and Qahwaji (2009) used MDI magnetograms from 1982 to 2006. Within this period, they reserved the years between 1999 and 2002 for testing. Accordingly, their models processed such images and provided automated tracking and classification of sunspots according to their McIntosh (1990)'s classes prior to making forecasts, which led them to score TPR = 0.81 and FAR = 0.30.

In turn, Leka, Barnes, and Wagner (2018) designed some linear discriminant models to forecast ≥ C flares within the next 24 h, 24 h – 48 h, and 48 h – 72 h based on full-disk and AR-by-AR analysis. Then, they deployed those models into a real forecasting environment. Concerning full-disk forecasts, they scored TSS values of 0.51, 0.55, and 0.45 for the next 24 h, 24 h – 48 h, and 48 h – 72 h, respectively. Not only they achieved positive TSS results, but also low FAR measures, noticeably 0.17 (next 24 h and 48 h – 72 h) and 0.16 (24 h – 48 h).

On the other hand, for the AR-by-AR strategy, Leka, Barnes, and Wagner (2018) did score more limited results, notably TSS = 0.29 (next 24 h and 24 h – 48 h) and 0.26 (48 h – 72 h). Conversely, unlike the full-disk scenario, their FAR almost doubled: 0.30 (next 24 h and 48 h – 72 h) and 0.25 (24 h – 48 h).

Finally, by using SRS data from NOAA/SWPC in their study, Bloomfield et al. (2012) handled Poisson statistics to calculate the flaring probabilities of each McIntosh (1990)'s class. To evaluate their model, they split their dataset into years of training (1988 – 1996) and test (1996 – 2010) data. For ≥ C1.0 flare forecasting in the next 24 h, their model scored TSS = 0.45 and FAR = 0.64.

E. Park et al. (2018) proposed another strategy to yearly segmenting their data to evaluate classifiers implemented as convolutional neural networks. They picked MDI magnetograms from 1996 May to 2010 December for every 00:00 UTC and HMI images from 2011 January to 2017 June also at this time. On the one hand, they processed images to calculate their pixel-based input features. On the other hand, their target feature comprehended a binary forecast from 00:00 UTC to 24:00 UTC based on GOES X-ray flux data: "no-flare" (weaker than C1.0) and "flare" (stronger than or equal to C1.0).

To evaluate their models, E. Park et al. (2018) chronologically separated their data into training and test sets. For training, they used images spanning from 1996 to 2008 (the solar cycle 23). For tests, they employed images from 2009 to 2017 (the solar cycle 24 partially represented). In this sense, their best model scored TSS = 0.63.

In addition to Bloomfield et al. (2012) and E. Park et al. (2018), who have split their data by years, Benvenuto et al. (2018) and Díscola Jr. et al. (2018a) also used such strategy. Following, the former authors assembled a dataset based on AR records tracked in the SRS repository. By considering the categorical classes of McIntosh (1990)'s components (Z, p, and c) and the magnetic classes of Mt. Wilson, authors computed the occurrence frequencies with which variables occurred or not linked to flares – ≥ C and ≥ M. On the other hand, the latter authors designed a Naïve Bayes-based classifier on X-ray time series data of 2014 and tested it on 2015 (TSS = 0.49).

To evaluate the performance of their model, namely a hybrid algorithm composed of a regularization method for regression and fuzzy C-means, they used the time range between 1996 August and 2010 December as the test set, whereas employing data collected between 1988 December and 1996 June as the training set. Consequently, they scored TSS = 0.43 for C-class events and above at the cost of somehow leveraging their false alarms (FAR = 0.47).

Human-based forecasts

Finally, we also included human-based forecasts from space weather prediction centers in Table 3.1. Here, we considered one proposal we have found, namely the one by Crown (2012). Their TSS was 0.57 for predicting C-class flares and higher in the next 24 h at NOAA/SWPC in the period between 1996 May and 2008 December (the solar cycle 23). On the other hand, they scored TPR = 0.63 and TNR = 0.94 (FAR = 0.40).

3.1.2 ≥ M-class flare forecasting

Conversely, tables 3.2 and 3.3 show our literature research involving ≥ M-class flare forecasting approaches. For better readability, the former contains biased results, whereas the latter includes unbiased and forecasts from prediction centers.

Biased results

In their research, Yang et al. (2013) employed ARs' vector magnetic data from the Solar Magnetic Field Telescope located at the Huairou Solar Observing Station in China. Their input features included the mean planar magnetic shear angle, vector magnetic field mean shear angle, and the mean absolute vertical current density. To forecast flares, Yang et al. (2013) designed a support-vector classifier relying on tenfold cross-validation.

For only including magnetograms with ARs within 30° from the disk center, we did classify Yang et al. (2013)'s research as biased. Nonetheless, they scored TSS = 0.48 and 0.53, and TPR = 0.41 and 0.43 respectively for predicting events within the next 24 and 48 h.

In turn, J.-F. Liu, F. Li, Wan, et al. (2017) proposed another forecasting approach only including ARs within 30° from the center. By using a mixed scenario of input features, the authors included several magnetic field data from MDI magnetograms on one hand, such as the singular points number, neutral line length, and maximum horizontal gradient. On the other hand, they analyzed the McIntosh (1990)'s classes of ARs in the catalog by the National Geophysical Data Center (NGDC).

J.-F. Liu, F. Li, Wan, et al. (2017) then designed a multi-model integrated learner by fitting and integrating neural networks, naïve classifiers, and SVMs as base learners. They merged their outputs with the linear sum, adjusting weights by a genetic algorithm. As of Yang et al. (2013), J.-F. Liu, F. Li, Wan, et al. (2017) also employed tenfold cross-validation to assess their model performance (TSS = 0.47).

In comparison to J.-F. Liu, F. Li, Wan, et al. (2017)'s methodology, J.-F. Liu, F. Li, H.-P. Zhang, et al. (2017) employed the same filter to select magnetogram samples, that is, they only reserved ARs locating 30° near the disk center. Instead of a multi-model learner, to forecast events within 48 h, they used image-case-based reasoning (TSS = 0.5).

In turn, C. Liu et al. (2017)'s article focused on evaluating the performance of a random forest model to forecast M- and X-class events over HMI magnetograms recorded at midnight with 24 h in advance. As they encountered an imbalanced set of data (only 24% of positive

Table 3.2: ≥ M-class flare forecasting models: biased results.

Forecasting Time	*Authorship*	*Grouping*	*ACC*	*TPR*	*TNR*	*TSS*	*FAR*
next 24 h	Yang et al. (2013)[a]	biased results	0.90	0.41	0.96	0.48	-
next 48 h	Yang et al. (2013)[a]	biased results	0.86	0.43	0.95	0.53	-
next 48 h	J.-F. Liu, F. Li, Wan, et al. (2017)[b]	biased results	-	0.64	0.83	0.47	-
next 48 h	J.-F. Liu, F. Li, H.-P. Zhang, et al. (2017)[c]	biased results	0.75	0.76	0.74	0.50	-
next 24 h	C. Liu et al. (2017)[d]	biased results	0.76	0.74	0.78	0.53	-
next 48 h	R. Li and Zhu (2013)[e]	biased results	0.82	0.69	0.83	0.52	-
next 48 h	R. Li, H. Wang, et al. (2011)[f]	biased results	0.74	0.69	0.75	0.44	-
next 48 h	Huang and H.-N. Wang (2013)[g]	biased results	0.72	0.72	0.71	0.71	0.70
next 48 h	X. Zhang, J. Liu, and Q. Wang (2011)[h]	biased results	-	0.75	-	-	-
next 48 h	Yu, Huang, H. Wang, and Cui (2009)[i]	biased results	-	0.82	0.84	0.66	-
next 48 h	Yu, Huang, Q. Hu, et al. (2010)[j]	biased results	0.92	0.94	0.91	0.86	0.28
next 48 h	Yu, Huang, H. Wang, Cui, et al. (2010)[k]	biased results	-	0.85	0.87	0.72	0.28
exact 24 h	Bobra and Couvidat (2015)[l]	biased results	0.92	0.83	0.92	0.76	-
exact 48 h	Bobra and Couvidat (2015)[l]	biased results	0.94	0.86	0.94	0.81	-
next 48 h	Raboonik et al. (2016)[m]	biased results	0.94	0.97	0.88	0.85	0.05
next 24 h	Muranushi et al. (2015)[n]	biased results	0.7	0.85	0.67	0.52	0.35
next 24 h	Jonas et al. (2018)[o]	biased results	-	-	-	0.81	-
next 48 h	Huang, Yu, et al. (2010)[p]	biased results	-	0.91	0.87	0.78	-
next 24 h	Huang, H. Wang, Xu, et al. (2018)[q]	biased results	0.81	0.85	0.81	0.66	0.90
next 48 h	Huang, H. Wang, Xu, et al. (2018)[q]	biased results	0.81	0.81	0.81	0.62	0.84
next 24 h	Sadykov and Kosovichev (2017)[r]	biased results	0.87	0.89	0.86	0.76	0.77
next 24 h	Nishizuka, Sugiura, Kubo, Den, Watari, et al. (2017)[s]	biased results	0.99	0.90	0.99	0.90	0.07
next 24 h	H. Liu et al. (2019)[t]	biased results	0.90	0.88	0.91	0.79	-
next 24 h	Florios et al. (2018)[u]	biased results	0.93	-	-	0.74	-
next 24 h	Alipour, Mohammadi, and Safari (2019)[v]	biased results	-	-	-	0.95	-
next 24 h	X. Wang et al. (2020)[w]	biased results	0.94	0.73	0.95	0.68	0.72
next 24 h	X. Li et al. (2020)[y]	biased results	0.89	0.81	0.93	0.74	0.11
next 24 h	Anastasiadis et al. (2017)[x]	biased results	-	-	-	0.30	-
next 24 h	Jiao et al. (2020)[z]	biased results	-	0.54	0.97	0.51	-

[a] Results collected from Table 4. TPR treated as frequency of hits (FOH). TNR defined as frequency of correct nulls forecasts (FOCN) (YANG et al., 2013).
[b] Results collected from Tables 3 and 4. Authors did not inform the ACC. TSS calculated over TPR and TNR (LIU, J.-F.; LI, F.; WAN, et al., 2017).
[c] Scores collected from Table 6. TSS calculated over TPR and TNR (LIU, J.-F.; LI, F.; ZHANG, H.-P., et al., 2017).
[d] Results collected from Table 4 (LIU, C. et al., 2017).
[e] Results collected from Table 2. ACC is treated as CORR. Results refer to the w = 0 column. TSS calculated over TPR and TNR (LI, R.; ZHU, 2013).
[f] Results collected from Table 3. ACC is treated as correctness. Results refer to the KM-LVQ column. TSS calculated over TPR and TNR (LI, R.; WANG, H., et al., 2011).
[g] Scores computed over the confusion matrix of Table 3 (HUANG; WANG, H.-N., 2013).
[h] Score gathered from Table III. Authors did not inform the TSS, TNR, and ACC. Results refer to the C4.5 column (ZHANG, X.; LIU, J.; WANG, Q., 2011).
[i] Scores gathered from Table 3. TSS computed over TPR and TNR. Results refer to the LVQ (w = 45) column (YU; HUANG; WANG, H.; CUI, 2009).
[j] Scores computed over the confusion matrix of Table 4. Results refer the MODWT_DB2_Red model (YU; HUANG; HU, Q., et al., 2010).
[k] Scores computed over the confusion matrix of Table 4. Results refer to the BN_F column (YU; HUANG; WANG, H.; CUI, et al., 2010).
[l] Results collected from Table 3 (BOBRA; COUVIDAT, 2015).
[m] Results collected from Table 3 (RABOONIK et al., 2016).
[n] Scores calculated over the confusion matrix of Figure 5. Although Muranushi et al. (2015) proposed an automated space weather design approach (as we shall discuss in Section 3.2.1, such scores correspond to a reported case study.
[o] The TSS refers to the highest score of Figure 14 (24 h prediction task, features included: HMI and flare hist). Authors did not inform ACC, TPR, and TNR (JONAS et al., 2018).
[p] Approximated scores from Figure 5 (in the graph, refer to the number of base prediction models equals 11). TSS calculated over TPR and TNR. Authors did not provide ACC (HUANG; YU, et al., 2010).
[q] Scores calculated over the confusion matrix of Table 4 (HUANG; WANG, H.; XU, et al., 2018).
[r] Scores computed from the TP, TN, FP, and FN values available at page 7 (SADYKOV; KOSOVICHEV, 2017).
[s] Results computed over the confusion matrix of Table 3. Results refer to the k-NN model (NISHIZUKA; SUGIURA; KUBO; DEN; WATARI, et al., 2017).
[t] Scores collected from Table 3. Results refer to the LSTM entry. TNR calculated over TPR and TSS (LIU, H. et al., 2019).
[u] Scores collected from the conclusions of page 27 (FLORIOS et al., 2018).
[v] Scores collected from the conclusions of page 12 (ALIPOUR; MOHAMMADI; SAFARI, 2019).
[w] Results computed over the confusion matrix of Table 4. Results refer to the LSTM-15_18 model (WANG, X. et al., 2020).
[y] Results collected from Table 2 of X. Li et al. (2020). TNR computed over TSS and TPR.
[x] Approximated TSS and TPR values from the graph of Figure 8 (TSS peak at a threshold of 0.15). Although Anastasiadis et al. (2017) proposed an automated space weather design approach (as we shall discuss in Section 3.2.1, such scores correspond to a reported case study.
[z] Results collected from Table B3 of Jiao et al. (2020) (metrics entry: 12-06). TNR computed over TSS and TPR.

Table 3.3: ≥ M-class flare forecasting models: unbiased and human-based results (continued).

Forecasting Time	*Authorship*	*Grouping*	*ACC*	*TPR*	*TNR*	*TSS*	*FAR*
next 24 h	Bloomfield et al. (2012)[a]	unbiased results	0.83	0.70	0.83	0.53	0.85
next 24 h	Shin et al. (2016)[b]	unbiased results	-	0.61	0.76	0.37	0.78
next 24 h	Leka, Barnes, and Wagner (2018)[c]	unbiased results	0.89	0.20	0.99	0.19	0.21
24 h – 48 h	Leka, Barnes, and Wagner (2018)[c]	unbiased results	0.87	0.03	1.00	0.03	0.20
48 h – 72 h	Leka, Barnes, and Wagner (2018)[c]	unbiased results	0.87	0.06	1.00	0.05	0.13
next 24 h	Nishizuka, Sugiura, Kubo, Den, and Ishii (2018)[d]	unbiased results	0.86	0.95	0.86	0.80	0.82
next 24 h	Hada-Muranushi et al. (2016)[e]	unbiased results	0.82	0.39	0.88	0.27	0.68
next 24 h	McCloskey, P. T. Gallagher, and Bloomfield (2018)[f]	unbiased results	-	-	-	0.47	-
next 24 h	D. Falconer et al. (2011) and D. A. Falconer et al. (2014)[g]	unbiased results	0.95	0.31	-	0.47	0.50
next 24 h	D. A. Falconer et al. (2014)[h]	unbiased results	0.95	0.38	-	0.49	0.48
next 24 h	Benvenuto et al. (2018)[i]	unbiased results	0.91	0.53	0.92	0.45	0.77
next 24 h	Kubo, Den, and Ishii (2017)[j]	human-based forecasts	0.84	0.60	0.90	0.50	-
next 48 h	Devos, Verbeeck, and Robbrecht (2014)[k]	human-based forecasts	0.88	0.37	0.97	0.34	-
next 24 h	Crown (2012)[l]	human-based forecasts	0.97	0.56	0.98	0.53	-

[a] Scores collected from Table 4. TNR computed over TSS and TPR. Results refer to the optimum TSS entry (BLOOMFIELD et al., 2012).
[b] Scores collected from Tables 6 and 10. Results refer to the MLR1 model. TNR calculated over TPR and TSS (SHIN et al., 2016).
[c] In Figure 13, those results refer to the full-disk performance. Although Leka, Barnes, and Wagner (2018) proposed an automated space weather design approach (as we shall discuss in Section 3.2.1, such results correspond to a case study reported.
[d] Scores calculated over the confusion matrix of Figure 5 (NISHIZUKA; SUGIURA; KUBO; DEN; ISHII, 2018).
[e] Scores calculated over the confusion matrix of Table 5 (HADA-MURANUSHI et al., 2016).
[f] TSS collected from the graph of Figure A.1 (p = 0.08 in the evolution line). Author did not inform TPR, TNR, ACC, and FAR (MCCLOSKEY; GALLAGHER, P. T.; BLOOMFIELD, 2018).
[g] Scores collected from the Table 2 of D. A. Falconer et al. (2014) (Present MAG4 entry).
[h] Scores collected from the Table 2 of D. A. Falconer et al. (2014) (Next MAG4 entry).
[i] Scores calculated over the confusion matrix of Table 2 (Hybrid entry) (BENVENUTO et al., 2018).
[j] Results collected from Table 4. TNR calculated over TSS and TPR (KUBO; DEN; ISHII, 2017).
[k] Results collected from Table 3. TNR calculated over TSS and TPR (DEVOS; VERBEECK; ROBBRECHT, 2014).
[l] Scores calculated over the confusion matrix of Table 4 (CROWN, 2012).

samples), the authors downsampled a hundred times their majority data part. They then combined it with the minority samples, thus creating 100 distinct subsets.

Accordingly, C. Liu et al. (2017) repeated tenfold cross-validation in each downsampled subset. This approach output a mean TNR = 0.78 and TPR = 0.74 (TSS = 0.53). Not only did they not properly reserve test sets, but they also only employed AR samples near the Sun's central meridian (70°), which made us classify their article as biased.

Similarly to J.-F. Liu, F. Li, Wan, et al. (2017)'s research, R. Li and Zhu (2013) also used tenfold cross-validation to evaluate the performance of their multi-layer perceptron model. For forecasting within the next 48 h, their dataset held samples spreading from 1996 April to 2008 December. By designing their features through a sliding time window schema, they included descriptive features like the sunspots area, ARs' magnetic classes, and X-ray fluxes.

However, due to their long records period, R. Li and Zhu (2013)'s dataset naturally held imbalanced class ratios. To cope with this nature, the authors carried out a k-means-based undersample – further explained by R. Li, H. Wang, et al. (2011). In fact, they performed undersampling before reserving the tenfold testing data segments, which made us treat their results as biased – despite their TPR = 0.69 and TSS = 0.52.

R. Li, H. Wang, et al. (2011)'s article detailed the under-sampling algorithm borrowed by R. Li and Zhu (2013). R. Li, H. Wang, et al. (2011)'s also aimed to design a flare forecasting model, that is, a learning vector quantization classifier (TSS = 0.44). To under-sample their data, the authors divided the dataset into negative and positive samples and inputted the negative set into a k-means algorithm – they set the k-means' k value to be the number of the flaring samples.

Then, R. Li, H. Wang, et al. (2011) reserved the clusters' centroids and appended them in the positive samples set (before performing their model's tenfold cross-validation). As R. Li and Zhu (2013) held some bias in results for not properly reserving test data in the beginning, so did R. Li, H. Wang, et al. (2011).

Besides using only magnetograms with ARs near the Sun's central meridian, the articles by J.-F. Liu, F. Li, H.-P. Zhang, et al. (2017), J.-F. Liu, F. Li, Wan, et al. (2017), R. Li, H. Wang, et al. (2011), and R. Li and Zhu (2013) also matched the fourth criterion of biased models. They designed their data to have only ARs linked to C-class events comprehending negative samples, thus excluding A- and B-class flares and their non-existence – in their case, ARs linked to ≥ M flares accounted for the positive class.

Analogously, Huang and H.-N. Wang (2013) also matched the third and fourth criteria for predicting ≥ M1.0 flares with a single decision tree in the next 48 h (TSS = 0.71). Not only Huang and H.-N. Wang (2013), but also X. Zhang, J. Liu, and Q. Wang (2011) have matched them with their C4.5 decision tree while analyzing features such as the magnetic field and texture distribution, and the largest sunspot group fractal dimensional to forecast events 48 h ahead (TPR = 0.75). In comparison, Huang, Yu, et al. (2010) pointed out to use the same data selection criteria as X. Zhang, J. Liu, and Q. Wang (2011) and Huang and H.-N. Wang (2013) for picking ARs samples for their ensemble and predictor teams (TSS = 0.78).

On the other hand, Yu, Huang, Q. Hu, et al. (2010), Yu, Huang, H. Wang, Cui, et al. (2010), and Yu, Huang, H. Wang, and Cui (2009) presented other forecasting approaches only including ARs linked to ≥ C1.0 flares in their datasets and discarded samples beyond the central radius. Accordingly, the authors calculated the maximum horizontal gradient, the neutral line length, and the singular points number over ARs from MDI magnetograms and used them as features for all articles.

Besides, the articles mentioned earlier employed sliding time window schemas to represent their evolutionary data streams. However, they diverged about the algorithms: Yu, Huang,

H. Wang, and Cui (2009) used a learning vector quantization classifier (TSS = 0.66), Yu, Huang, Q. Hu, et al. (2010) employed a C4.5 tree (TSS = 0.86), and in Yu, Huang, H. Wang, Cui, et al. (2010), the model was a Bayesian network (TSS = 0.72).

Conversely, Bobra and Couvidat (2015)'s article coped with positive and negative classes differently from J.-F. Liu, F. Li, Wan, et al. (2017), J.-F. Liu, F. Li, H.-P. Zhang, et al. (2017), Yu, Huang, H. Wang, and Cui (2009), Yu, Huang, Q. Hu, et al. (2010), R. Li and Zhu (2013), R. Li, H. Wang, et al. (2011), Huang and H.-N. Wang (2013), and Yu, Huang, H. Wang, Cui, et al. (2010). They employed ≥ M1.0 X-ray measures to flag positive AR samples exactly 24 (TSS = 0.76) and 48 h (TSS = 0.81) after a given instant and defined both C-class flares and the non-existence of events as the negative class. As such, their dataset held about 300 positive samples and 5,000 negative AR cases selected at random from HMI magnetograms, notably comprehending imbalanced class ratios.

By using the cost function of an SVM classifier, Bobra and Couvidat (2015) coped with their class ratio imbalanced nature. Accordingly, they sought ideal weights for both of their classes in an attempt to guarantee that the classifier would not have given much emphasis to majority samples. As the validation strategy, they employed repeated subsampling (hold-out), randomly splitting their data into training (70%) and testing (30%) sets at each iteration.

As a matter of fact, Bobra and Couvidat (2015) investigated how their TSS varied over test sets as weights of positive and negative classes changed. In this context, they picked optimal weight values to optimize the TSS in the test sets.

However, Bobra and Couvidat (2015)'s TSS adjustment strategy incurred in some bias for outputting tailored results concerning sampled test sets, that is, they did not use true unseen data to evaluate their final model. Their results could have underestimated their real test error substantially (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). In their case, they also risked statistical flukes, i. e., test sets too easy to forecast. Besides, because their dataset only held about 300 positive examples – within 68° from the central meridian –, this could affect their model generalization skill in real operational settings, thus also incurring in some bias for lacking enough data.

By using the same HMI data period as in Bobra and Couvidat (2015)'s research, Jonas et al. (2018) designed some linear classifiers upon time series to predict M- and X-class flares in the next 24 h (TSS = 0.81). To evaluate classifiers, the authors investigated distinct combinations of features also using repeated random subsampling (training (80%) vs. test (20%) data).

Jonas et al. (2018) aimed to report the best subset of features, i. e., the one that best increased TSS upon test sets. Since the authors employed test data during decision-making instead of assessing generalization error over real unseen data, we classified their results as biased. In fact, they treated test samples as validation data (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). Another forecast approach for M-class events and above employing test data within a validation scenario is the one by Florios et al. (2018) (TSS = 0.74), for varying the prediction threshold within several hold-out iterations and choosing the threshold that peaked some model's TSS.

Raboonik et al. (2016) proposed another classifier lacking enough data during training. Despite scoring high results with their SVM for predicting flares in the next 48 h (TSS = 0.85), their dataset only held 85 positive and 208 negative class samples. As previously mentioned, reduced datasets can weaken results interpretation involving how systems would behave in real operational environments, that is, models probably would not generalize well because of the small number of analyzed samples (under-fitting) (PYLE, 1999).

In turn, Nishizuka, Sugiura, Kubo, Den, Watari, et al. (2017) were the first researchers analyzing ARs beyond the limb with their models (beyond any defined radius from the disk center). Accordingly, they calculated about 60 features from HMI magnetograms from 2010 to 2015, including those proposed by Bobra and Couvidat (2015) and some others related to UV brightening and flare history. Then, to find the best algorithm regarding its mean TSS, they carried out a comparison between some distinct models, such as SVM, extremely randomized trees, and k-NN, through repeated random subsampling (training (70%) versus testing (30%) data).

However, Nishizuka, Sugiura, Kubo, Den, Watari, et al. (2017)'s rationale for picking the best model was similar to Bobra, Sun, et al. (2014)'s and Jonas et al. (2018)'s, who employed test sets for decision-making. Hence, we also treated their results as biased for not reserving real unseen data and keeping them aside while fitting their models. Despite that, their best model – the k-NN – scored TSS = 0.90.

The remainder of articles for ≥ M flare forecasting including only AR samples near the disk center comprehends the researches by Muranushi et al. (2015) (69°, TSS = 0.52), Huang, H. Wang, Xu, et al. (2018) (30°, TSS = 0.66 and 0.62 respectively for 24 and 48 h), X. Wang et al. (2020) (68°, TSS = 0.68), X. Li et al. (2020) (45°, TSS = 0.74), Sadykov and Kosovichev (2017) (68°, TSS = 0.76), H. Liu et al. (2019) (70°, TSS = 0.79), Alipour, Mohammadi, and Safari (2019) (60°, TSS = 0.95), Anastasiadis et al. (2017) (70°, TSS = 0.30), and Jiao et al. (2020) (0.68°, TSS = 0.51).

Unbiased results

Conversely to the articles mentioned earlier, Bloomfield et al. (2012)'s approach presented a methodology that employs data from the summaries of sunspots from NOAA/SWPC and calculates the flaring probabilities of McIntosh (1990)'s classes using Poisson statistics. Accordingly, they distinguished their dataset by years during their model evaluation: whereas they used the period between 1988 and 1996 for training, data from 1996 to 2010 accounted for the test samples. To forecast ≥ M1.0 events in the next 24 h, they scored TSS = 0.53 (FAR = 0.85).

Analogously to Bloomfield et al. (2012), McCloskey, P. T. Gallagher, and Bloomfield (2018) employed Poisson statistics with McIntosh (1990)'s classes too (TSS = 0.47). However, they analyzed how such elements evolve within sunspot groups instead of dealing with static observations leading to flares. Their training period comprehended 1988 and 1996, whereas their test data accounted for 1996 to 2008.

In turn, Shin et al. (2016) designed a prediction schema with a focus on each AR employing multiple linear regression. In comparison to the articles mentioned earlier, from the sunspot summaries by NOAA/SWPC, they calculated the Weighted Mean Flare Rate (WMFR) of McIntosh (1990)'s classes and magnetic configurations, and the weighted total flare flux of previous flaring days. They sampled at random records of ≥ M-class events from 1996 January to 2004 December to train their model. On the other hand, they employed the years between 2005 January to 2013 November for testing, scoring TSS = 0.37 (FAR = 0.78).

Besides ≥ C-class flare forecasting, Leka, Barnes, and Wagner (2018) also designed some linear discriminant models for ≥ M events. As such, within the next 24 h, 24 h – 48 h, and 48 h – 72 h, they respectively scored TSS = 0.19 (FAR = 0.21), 0.03 (FAR = 0.20), and 0.05 (FAR = 0.13). Those results refer to a full-disk scenario of prediction.

Even though we previously considered Muranushi et al. (2015)'s results for ≥ M events as biased for only using ARs near the disk center during training, more recently, there was an attempt to deploy an operational system by their time series design engine (HADA-MURANUSHI et al., 2016). As of Muranushi et al. (2015)'s results for ≥ C events, the authors adjusted their engine to design an LSTM learning model upon wavelet features calculated over HMI images. For forecasting ≥ M-class events within the 24 h, they scored TSS = 0.27 (FAR = 0.68).

On the other hand, D. Falconer et al. (2011) proposed another approach already deployed into a real operational forecast environment, namely the Magnetogram Forecast Forecasting Tool (MAG4). Their algorithm aimed to monitor and predict astronauts' radiation exposure levels by forecasting the occurrence of ≥ M-class flares, solar energetic particles, and coronal mass ejections (TSS = 0.47). As such, they designed MAG4 upon a set of data comprehending 40,000 MDI magnetograms from 1,300 ARs.

Recently, by employing the same data as in D. Falconer et al. (2011)'s research, D. A. Falconer et al. (2014) tried to improve MAG4 performance using modeling of previous flare history with features characterizing free-energy in ARs. This new forecast approach scored TSS = 0.49 (FAR = 0.48).

Although both D. Falconer et al. (2011)'s and D. A. Falconer et al. (2014)'s researches have used MAG4 within a deployment environment, their dataset only included ARs within 30° of the solar disk center. For predicting with ARs beyond this limit, the tool warned reduced performance, as pointed out by their authors. This scenario raises uncertainty about the reported scores, despite MAG4 did forecast operationally.

By employing the same period of data and features from Nishizuka, Sugiura, Kubo, Den, Watari, et al. (2017), Nishizuka, Sugiura, Kubo, Den, and Ishii (2018) reserved the period between 2010 to 2014 for training and 2015 for testing. Through yearly splitting data instead of repeatedly subsampling, their convolutional neural network scored TSS = 0.80 (FAR = 0.82).

As of Nishizuka, Sugiura, Kubo, Den, Watari, et al. (2017) and Nishizuka, Sugiura, Kubo, Den, and Ishii (2018), which have used disjoint periods for evaluating their classifiers, so did Benvenuto et al. (2018). The period between 1996 August and 2010 December was the test set, whereas data collected between 1988 December and 1996 June comprehended the training set. Following, their hybrid model scored TSS = 0.45.

Human-based forecasts

Finally, we also included in Table 3.3 articles comprehending human-based forecasts from space weather prediction centers. Accordingly, we quoted the prediction centers by Crown (2012), Kubo, Den, and Ishii (2017), and Devos, Verbeeck, and Robbrecht (2014). At NOAA, they scored TSS = 0.49 (FAR = 0.57) for predicting ≥ M events within the next 24 h in the solar cycle 23 (1996 May to 2008 December), as pointed out by Crown (2012). In turn, their ACC, TPR, and TNR respectively equaled 0.97, 0.56, and 0.98.

On the other hand, at the SWPC of the Japan National Institute of Information and Communications Technology (NICT), human-based forecasters scored TSS = 0.50 (FAR = 0.42) for a time horizon of 24 h in the period between 2000 and 2015. Their ACC and TPR, in turn, respectively equalled 0.84 and 0.60 (KUBO; DEN; ISHII, 2017).

Finally, at the prediction center of the Solar Influences Data Center (SIDC) of the Royal Observatory of Belgium (ROB), they scored TSS = 0.34 (FAR = 0.35) for forecasting in the next 48 h during the period between 2004 June to 2012 December. Regarding both positive and negative hit rates, they reached TPR = 0.37 and TNR = 0.97 (DEVOS; VERBEECK; ROBBRECHT, 2014).

3.2 Automated machine learning efforts

As we previously commented, the classifiers mentioned earlier in this chapter held *ad hoc* design methodologies, thus not focusing on constant reproducing. Aware of the benefits of automated design pipelines, which mostly comprehend methodologies that make decisions through an automatic, data-driven, and objective way to determine the best performing forecast model fully, it is worth further discussing such automated solutions' capabilities.

To date, several tools have been proposed to automate the design of general basis classifiers. As initial examples, we can cite the Auto-WEKA (KOTTHOFF et al., 2019), Hyperopt-Sklearn (KOMER; BERGSTRA; ELIASMITH, 2019), Auto-sklearn (FEURER; KLEIN, et al., 2019), Auto-net (Auto-PyTorch) (MENDONZA et al., 2019; ZIMMER; LINDAUER; HUTTER, 2020), TPOT (OLSON; MOORE, 2019), Auto-keras (JIN; SONG, Q.; HU, X., 2018), RoBO (KLEIN et al., 2017), H2O AutoML (LEDELL; POIRIER, 2020), and AutoGluon-Tabular (ERICKSON et al., 2020).

Based on the Waikato Environment for Knowledge Analysis (WEKA) machine learning toolkit (WITTEN; FRANK; HALL, 2011), Auto-WEKA comprehends a system for automatically searching over the available options of WEKA for model selection, feature selection, and hyperparameter optimization. As such, Auto-WEKA can investigate 10 meta-methods (algorithms using or combining multiple algorithms), 28 base learners (machine learning algorithms), and the hyperparameters of each learner in an attempt to output a learning model with improved performance (KOTTHOFF et al., 2019).

On the other hand, Komer, Bergstra, and Eliasmith (2019) designed Hyperopt-Sklearn to automate the design of classifiers in Python's Scikit-learn framework (PEDREGOSA et al., 2011).

In an attempt to fit custom classifiers, Hyperopt-Sklearn allows users to configure some distinct design aspects, such as the search domain, objective function, optimization algorithm, and learning algorithm (i. e., a standard algorithm of the Scikit-learn, such as the RandomForest, SVM, k-NN, or Principal Component Analysis – PCA). The tool then uses those elements in a search process, seeking the best performing model along with its hyperparameters (KOMER; BERGSTRA; ELIASMITH, 2019).

Also designed upon the Scikit-learn environment, Auto-sklearn relies on similar optimization techniques as in Auto-WEKA's research. However, it holds some improvements compared to the latter, such as meta-learning, automatic ensembling (automatic design of ensembles from optimized models), and feature preprocessing (FEURER; KLEIN, et al., 2019). Authors proposed not to discard all models trained during the automated investigation but rather to store and combine them to use within an efficient post-processing method, that is, to design an ensemble with them.

In turn, Mendonza et al. (2019) and Zimmer, Lindauer, and Hutter (2020) focused their automated design processes – Auto-net (Auto-PyTorch) – upon some specific feed-forward neural networks, that is, they proposed a system for automated deep learning aimed to select both the architecture and hyperparameters of deep neural networks. The rationale for restricting Auto-net to feed-forward neural networks relied on their applicability to a wide range of different datasets.

Noteworthily, another tool for automated deep learning allowing automatic search for deep neural networks' architectures and hyperparameters is Auto-Keras (JIN; SONG, Q.; HU, X., 2018). However, instead of relying on Python's PyTorch deep learning toolkit – as Auto-net (Auto-PyTorch) does –, Auto-keras was designed under Python's Keras deep learning toolkit. As such, Auto-keras employs network morphisms guided by Bayesian optimization.

More recently, Olson and Moore (2019) presented the TPOT approach to automatically build and optimize tree-based machine learning algorithms (i. e., DecisionTree, RandomForest, ExtremeGradientBoosting – XGBoost, among others). Conversely to the approaches mentioned earlier, TPOT explicitly includes automated feature selection. Analogously to Hyperopt and Auto-sklearn, their authors conceived it upon the Scikit-learn environment.

Also written in Python, RoBO (KLEIN et al., 2017) is a framework for Bayesian optimization aimed to create custom forecast models. Its core comprehends modular components designed to add and exchange components of Bayesian optimization easily.

However, due to its nature, RoBO restricts itself to regression models naturally, such as Gaussian processes, RandomForests, and Bayesian neural networks. Specifically, it aims to optimize different acquisition functions, such as expected improvement, probability of improvement, lower confidence bound, or information gain.

In turn, LeDell and Poirier (2020) proposed the H2O AutoML tool as a fully automated supervised algorithm for H2O, the open source, scalable, and distributed machine learning framework. Such a tool mostly relies on the automatic training and optimization – through random search – of a stacked ensemble from a set of defined algorithms (i. e., XGBoost, GradientTreeBoosting, RandomForest, and deep neural networks). For preprocessing input data, H2O AutoML provides some distinct methods, such as automatic imputation, normalization, and one-hot encoding for XGBoost models. On the other hand, H2O AutoML's tree-based models support group-splits on categorical variables.

Also relying on automatic ensembling, AutoGluon-Tabular (ERICKSON et al., 2020) is another automated design tool for Python based on the Scikit-learn. Similar to H2O AutoML, AutoGluon-Tabular includes several capabilities for input data preprocessing. In fact, Erickson et al. (2020) highlighted data preprocessing as a differential of AutoGluon-Tabular, as a few frameworks can robustly process raw data and deliver high-quality predictions without user interventions. After preparing data, AutoGluon-Tabular then fits some distinct models from a defined set (i. e., neural networks, XGBoost, RandomForests, ExtremelyRandomizedTrees, and k-NN), and ensembles them in a novel way in the end (it stacks them into multiple layers, which are trained through a layer-wise manner).

Other automated machine learning platforms worth mentioning include: auto-xgboost (THOMAS; COORS; BISCHL, 2018), AutoFolio (LINDAUER et al., 2015), Python's auto_ml package (PARRY, 2016), and tuneRanger (PROBST; WRIGHT; BOULESTEIX, 2018).

Despite their notable flexible aspects, we observed those general basis systems do not hold a comprehensive pipeline to cope with solar flare forecasting, as we propose in this thesis. To exemplify, we can cite some encountered drawbacks:

- Hyperopt-Sklearn, Auto-sklearn, and Auto-keras lack an automated feature selection process for selecting only relevant input data for their classifiers;
- Most of them restrict their design processes to some learning algorithms, thus missing a generic algorithm selection process: Auto-sklearn (ensembles), Auto-net (neural networks), TPOT (tree-based learners), H2O AutoML (tree-based algorithms, deep

neural networks, and stacked ensembles), AutoGluon-Tabular (tree-based algorithms, deep neural networks, k-NN, and stacked ensembles), RoBO (regression models), tuneRanger (RandomForests), and auto-xgboost (XGBoost);

- Some tools restrict their hyperparameter optimization methods to predefined algorithms: Auto-WEKA and Auto-sklearn (Bayesian optimization), H2O AutoML (random search), and TPOT (genetic programming);
- Neither of those approaches report how they could deal with imbalanced class ratios, thus not coping with data resampling or cost-sensitive learning;
- Neither of those approaches report how they could adjust the cut-off points (prediction thresholds) of their classifiers to leverage the expected performance while reducing the number of false alarms.

In fact, because of their broader nature, we could not expect such general basis approaches to fulfill all space weather needs satisfactorily. This naturally led us to seek proposals focused on space weather in the specialized literature, which we shall discuss next.

3.2.1 Automated space weather forecast

As we previously mentioned in Chapter 1, the automated design processes for space weather mostly comprehended one capability of fully integrated services focusing on the automatic computation of big data. Such proposals included the researches by Muranushi et al. (2015), Leka, Barnes, and Wagner (2018), Massone and Piana (2018), Engell et al. (2017), Garciá-Rigo et al. (2016), and Anastasiadis et al. (2017).

For emphasizing big data computation, such approaches mostly plugged their design pipelines into some previously defined data sources. To name the sources, we can cite those providing HMI images (LEKA; BARNES; WAGNER, 2018; MASSONE; PIANA, 2018; ENGELL et al., 2017; GARCIÁ-RIGO et al., 2016), MDI magnetograms (LEKA; BARNES; WAGNER, 2018), summaries of ARs from NOAA (MASSONE; PIANA, 2018; ENGELL et al., 2017; GARCIÁ-RIGO et al., 2016), the catalog of events also from NOAA (LEKA; BARNES; WAGNER, 2018), and users feeding data into the platform (i. e., crowd-sourcing) (ENGELL et al., 2017). They then allowed users to train custom classifiers to some extent, comprehending the adjustment of some flexible forecast aspects, as described next.

By employing the machine learning's regression task, Muranushi et al. (2015) proposed the Universal Forecast Constructor by Optimized Regression of Inputs (UFCORIN). Using linear classifiers, their system comprehended a generic time series predictor, which could be set to forecast generic time series features from arbitrary input data sets.

Hence, UFCORIN provided users with the flexibility to easily change the input and output target features when new data became available, or new models were needed. In fact, UFCORIN was a general automated space weather approach since it allowed the prediction of events not restricted to flares (i. e., mass and speed of CMEs, and total flux and power indices of SEPs).

Muranushi et al. (2015) designed UFCORIN to accept a fixed number of time series data and output another time series. All the input and output data had to be features globally defined at any given moment of the Sun (i. e., supporting full-disk forecasts), thus excluding the possibility to process features defined per sunspot or active regions.

In turn, Leka, Barnes, and Wagner (2018) proposed the Discriminant Analysis Flare Forecasting System (DAFFS). DAFFS comprehended a fully integrated solar flare prediction system for analyzing the Sun's magnetic fields with an embedded automated design process. When designing new models, DAFFS always used as much data as possible (i. e., it used to employ the period between 2012 and the most recent month at the time of its execution). Besides, DAFFS provided some custom options for their models, including adjusting the type of prediction (i. e., whether it would consider a full-disk or AR-by-AR forecast scenario), threshold for the size of events (i. e., ≥ C-, M-, and X-class events), forecasting horizon, latency of forecasts, and the performance optimization strategy against either of the two error types (i. e., false alarms or missed events).

On the other hand, Massone and Piana (2018) presented another fully integrated technological service allowing automatic computation of a big data amount of observations, namely the Flare Likelihood and Region Eruption foreCASTing (FLARECAST). As pointed out by their authors, FLARECAST was a project aiming to develop an advanced solar flare prediction system based on automatically extracted physical properties of ARs, along with state-of-the-art prediction algorithms – basically, linear classifiers or neural networks.

Besides, FLARECAST explicitly employed its data input into a feature selection process. Despite providing flexibility to some extent, Massone and Piana (2018) acknowledged that FLARECAST restricted the design of their classifiers to a predefined set of algorithms and

data sources – in fact, they intended to allow users to plug in their classifiers (or already preprocessed features) into the FLARECAST's pipeline in the future.

Engell et al. (2017) introduced the Space Radiation Intelligence System (SPRINTS), which also comprehended a fully integrated service for automatic computation and big data forecast. SPRINTS has improved the MAG4 tool for solar flare forecasting by D. A. Falconer et al. (2014) with SEP forecasting. It also provided components to ingest data from external data sources using the Hadoop tool for big data (WHITE, 2009).

Hence, SPRINTS used processes to Extract, Transform, and Load (ETL) data to pull contents into its transactional database. For data arriving every minute, the tool supported near real-time loading. With a particular emphasis on the interpretability of forecast models, SPRINTS focused their models on tree-based algorithms, or the k-NN.

Similarly to the SPRINTS, the platform by Garciá-Rigo et al. (2016) – Solar Events Prediction System for Space Launch Risk Estimation (SEPsFLAREs) – also has been created by integrating two distinct tools, namely the ASAP by Colak and Qahwaji (2009), for flare forecasting, and the UMASEP by Nunez (2011), for SEP forecasting. Regarding the former, it was a neural network based system developed to automate the extraction of sunspot data from solar images (i. e., McIntosh (1990)'s class and areas) directly and provide flare forecasts in near real-time.

Noteworthily, ASAP was initially developed to operate on MDI solar data, but then their authors redesigned it to process HMI records to operate along with UMASEP. Besides, SEPsFLAREs provided a set of defined forecasting horizons to customize (i. e., users could change between forecasts within the next 6, 12, 24, and 48 h) (GARCIÁ-RIGO et al., 2016).

In addition to flare and SEP forecasting, Anastasiadis et al. (2017) also provided capabilities for anticipating CMEs with their tool, namely The Forecasting Solar Particle Events and Flares (FORSPEF). FORSPEF provided two modes of operation: forecasting and nowcasting. The former worked only on the analysis of ARs before the occurrence of some event. In turn, the latter corresponded to the analysis of flares and CMEs that happened and their relation with future SEPs. Another approach considering such modes of operation was Garciá-Rigo et al. (2016)'s. Concerning flare forecasting, FORSPEF restricted its data processing cadence to a 3 h interval, as well as provided custom forecasts for ≥ C-, M-, and X-class events (always within the next 24 h).

It is worth mentioning that we have also found a similar approach to flexibilize the design of solar flare classifiers (despite its authors did not define it as an automated process), i. e.,

the sequence miner (SeMiner) by Díscola Jr. et al. (2018b). SeMiner's authors acknowledged that the specialist's knowledge is determinant to the success of forecasting methods based on data mining. In this sense, to help physicists, they envisioned a design process focused on the preprocessing of X-ray time series input data, that is, the adjustment of some custom aspects of such series, such as the selection of the most relevant features, the most relevant sub-series, the characteristics of the time series evolution most significantly affecting the forecasting results, periods for training and test subsets, and both sliding time window's and forecasting horizon's length.

Accordingly, in the beginning, SeMiner processes solar X-ray time series and transforms them into sequences. It then maps the sequence's records to their corresponding occurred events. At this point, the specialist can adjust the custom aspects mentioned earlier. Then, SeMiner submits their sequences to a feature selection method (which can be the Starminer or relief attribute evaluation) to determine the features most affecting the forecast performance. Finally, it employs the prepared sequences to train a classifier (such as a decision tree, k-NN, Naïve Bayes, OneR, or SVM) and use test data to evaluate it.

3.3 Optimized design for solar flare classifiers

Regardless of the flexible aspects discussed earlier, empowering an optimized design (automated or not) for solar flare classifiers may not comprehend a straightforward process at all. By an optimized design, we meant training forecast models not incurring in most negative issues we observed authors usually encompassed in their design processes (automated or not).

Within this context, it was not clear how the space weather automated approaches could deal with some design aspects worth improving in their output models (i. e., Leka, Barnes, and Wagner (2018) produced classifiers that did not equally deal with both positive and negative classes, and Garciá-Rigo et al. (2016) produced classifiers forecasting a great number of false alarms, sometimes trespassing the 90% level). Besides, due to their nature, such systems did not seem to easily expand their functionalities to accommodate more data sources or algorithms.

Not only dealing with those issues in the classifiers or design processes but also accommodating more custom aspects to satisfy users' needs in constant change is worth considering. For instance, some authors reported a fixed cadence for analyzing data in their

tools, namely Anastasiadis et al. (2017), for processing data always at a 3 h interval (for flares) and 6 h (for CMEs).

Another example would be allowing the forecast of events other than solar flares, that is, Garciá-Rigo et al. (2016) restricted their tools to SEP and flare forecasting, whereas Massone and Piana (2018), and Leka, Barnes, and Wagner (2018) only coped with flares. Alternatively, Anastasiadis et al. (2017) provided flare, SEP, and CME forecasting. Those examples may not yield enough flexibility for general space weather forecasting.

Bearing such restrictions in mind, Muranushi et al. (2015) were the only authors reporting to have proposed a system adhering to any space weather event. As such, they remarkably managed to provide a generic data input, thus not restricting to only one or a few data sources.

However, Muranushi et al. (2015)'s drawback concerned their engine's data input nature: they only accepted data designed upon a time-series format, and that was globally defined at any given moment of the Sun, excluding features that were defined per sunspots, or ARs. Their approach suggested an automated pipeline to design space weather models – as the tools for general machine learning – instead of a fully integrated system mostly focused on big data processing from default sources.

3.3.1 Good practices for space weather design

Noteworthily, the issues mentioned earlier would naturally lead us to the following question: what aspects do constitute a positive – yet effective – design process for solar flare classifiers? Camporeale (2019) summarized eight distinct premises one must consider when applying machine learning to space weather effectively: algorithm and data selection, tuning hyperparameters, proper data splitting, cross-validation, testing data, specialized metrics, and bias-variance decomposition (we shall discuss his motivation soon).

Camporeale (2019) did not aim to provide a strict and rigid set of rules, but a general pipeline for good practices. Despite presenting some remarkable aspects, we believed that his list was not sufficiently comprehensive, as it lacked some other equally important aspects.

However, we considered Camporeale (2019)'s list as a starting point for proposing a novel comprehensive pipeline, that is, a new one extending the original proposal and including other aspects we observed authors often need when designing flare classifiers. This new proposal naturally led us to include the following elements into their his list: data resampling, cost-

sensitive learning, and cut-off point adjustment. We comment on such an extended collection of premises as follows.

Algorithm selection

A straightforward taxonomy to classify supervised algorithms distinguishes them between parametric and non-parametric approaches. Whereas the former are usually faster to train and handle larger data sets (i. e., linear discriminant analysis, logistic regression, and neural networks) – sometimes making unappropriated assumptions due to their input bias –, the latter often make milder assumptions on data but are computationally expensive for larger datasets (i. e., SVMs, k-NNs, and DecisionTrees) (CAMPOREALE, 2019).

Choosing a promising algorithm for solar flare forecasting – and also other domains – may depend on several factors, such as performance, training time and complexity. Other factors may include whether algorithms need to be retrained when new data income, how fast they make their predictions, and their scalability when increasing the data size (KUHN; JOHNSON, 2013).

Machine learning employed in space weather must rely on processes following the selection criteria mentioned earlier to select between distinct algorithms. The set of available algorithms in the automated space weather tools comprehended SVMs, and linear regression (MURANUSHI et al., 2015); penalized logistic regressors, SVMs, and multi-layer perceptrons (MASSONE; PIANA, 2018); DecisionTrees, RandomForests, and k-NN (ENGELL et al., 2017); linear discriminant analysis (LEKA; BARNES; WAGNER, 2018); neural networks (GARCIÁ-RIGO et al., 2016); and statistical models (ANASTASIADIS et al., 2017).

Data selection

The quality and subset of input data certainly affect the performance of machine learning algorithms. Hence, discarding useless or redundant features is an imperative task to perform. Not only selecting relevant features but also providing means to preprocess data is rather useful (HAN; KAMBER, 2006).

Within this context, discarding input features must adhere to some measure of dependence between them. For instance, the entropy's well-known concept can be used to describe mutual information shared between input data, and infer their casual dependence concerning a target feature. This approach allows to rank features in terms of their importance and defines the

extent which the causal relationship between inputs and outputs is significant. Not only entropy, but several other metrics can also be used to measure mutual information, such as some correlation-based or univariate metric (CAMPOREALE, 2019).

The techniques available to select input data in the automated space weather tools comprehended L1-logit and LASSO (MASSONE; PIANA, 2018); and computing the skill scores for all possible 2-variable combinations (LEKA; BARNES; WAGNER, 2018). Besides, Leka, Barnes, and Wagner (2018) also provided automated treatment for missing data. In turn, the remainder articles did not report whether either of such capabilities were available.

Tuning hyperparameters

Machine learning algorithms often provide parameters that are free to choose to control how they behave (i. e., in tree-based classifiers, how deep the inner nodes can be). Not rarely, hyperparameters can also be used in an attempt to control over-fitting – provided that the algorithm employs parameters for regularization (i. e., in tree-based ensembles, controlling the early stopping aspect when training the inner base learners) (CAMPOREALE, 2019).

Appropriately tuning hyperparameters may incur in a non-negligible impact on the performance and computational cost of forecast models (FEURER; HUTTER, 2019). Muranushi et al. (2015) reported to cope with automated hyperparameter optimization in their design pipeline directly. Once more, the remainder of authors did not report whether such capability was available.

Proper data splitting

Besides selecting algorithms and appropriate data for them, as well as adjusting hyperparameters, it is worth caring about how to split data to train machine learning algorithms. Using the entire set of data when training algorithms could be the first choice sometimes, but not the best. There are various strategies for splitting data, but not all of them may incur in results assessed without bias (CAMPOREALE, 2019).

Overall, it is a good practice to divide records into three subsets, namely for training (for fitting models), validating (for decision-making), and testing (for predicting real unseen data) models (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). Examples of splitting techniques employed in the automated space weather tools comprehended repeatedly splitting data into training and test sets (MURANUSHI et al., 2015); yearly splitting data into training and test

sets (GARCIÁ-RIGO et al., 2016); employing the Monte Carlo strategy[2] (MASSONE; PIANA, 2018; ANASTASIADIS et al., 2017); and using the training/test sets of an ordinary k-fold cross-validation (LEKA; BARNES; WAGNER, 2018). Noteworthily, neither of those tools used validation sets for decision-making in their processes as suggested by Hastie, Tibshirani, and Friedman (2009).

Cross-validation

Finding ways to avoid sets of data outputting positive performance due to good luck (risking statistical flukes) is worth considering (CAMPOREALE, 2019). This aspect involves cross-validation procedures, mostly known for dividing original data into k disjoint subsets and using $k - 1$ of them for training and the remaining one for test. Accordingly, the permutation between training and test sets produces k different models, whose results are draw by their average performance (JAMES et al., 2013).

Examples of techniques based on cross-validation in the automated space weather tools included iteratively splitting data into training/test sets (MURANUSHI et al., 2015); and empowering an ordinary k-fold cross-validation method (LEKA; BARNES; WAGNER, 2018). On the other hand, as both Massone and Piana (2018) and Anastasiadis et al. (2017) repeatedly employed the Monte Carlo approach, they could not guarantee that each data sample would participate in the test set at least once. The other automated proposals did not provide enough details for allowing us to identify how they managed to avoid such flukes.

Testing data

Ideally, test data should play the role of "fresh" unseen evidences. The test concept gives a true estimate of models' performance when forecasting as if they were executed in a real deployment environment (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). However, some researchers may be tempted to use information gained during the evaluation of models over the test sets to improve or correct design decisions or even assess which model performed better, clearly representing a bias of tailoring results for the test sets, i. e., Bobra and Couvidat (2015) and Nishizuka, Sugiura, Kubo, Den, Watari, et al. (2017), and Jonas et al. (2018).

Besides, one must also consider some pitfalls while reserving data for space weather sets, such as performing data reservation at random or temporally – whether the aim is to focus on

[2]Such a strategy relies on repeated random subsampling.

the periodicity of the time series (i. e., the solar cycle dependence) (CAMPOREALE, 2019). An automated space weather tool clearly employing test data truly unseen was Garciá-Rigo et al. (2016)'s, for empowering data from 1981 December to 2013 December for designing the forecast models, and testing their performance on a dataset from 2014 January to 2015 December.

Specialized metrics

The performance of learning algorithms is represented in terms of metrics. As classifiers may have different aspects in which they output their results, it is a good practice to assess how they perform with individual scores regardless of the nature of metrics, which can comprehend deterministic (i. e., class-specific recalls, precision, quality skill scores, among others), or probabilistic (i. e., area under the curve) indexes (BARNES; LEKA, et al., 2016).

Not only assessing various scores but also employing them in comparisons with baseline forecasts is worth carrying out. Baseline references allow a broad view of the achieved improvements (if any) (CAMPOREALE, 2019). However, not all space weather platforms empowered a broader view of performance for their classifiers. Some of them focused only on deterministic scores, such as:

- TSS (MURANUSHI et al., 2015);
- ACC, Probability of Detection (POD), FAR, Heidke Skill Score (HSS), and TSS (MASSONE; PIANA, 2018);
- Critical Success Index (CSI), Percent Correct (PC), ACC, POD, FAR, HSS, and TSS (ANASTASIADIS et al., 2017);
- POD, FAR, and HSS (ENGELL et al., 2017).

On the other hand, the remaining proposals managed to use both deterministic and probabilistic scores:

- Brier Skill Score (BSS), The Receiver Operating Characteristic (ROC) curve, and Appleman Skill Score (ApSS) (LEKA; BARNES; WAGNER, 2018);
- Reliability plots, Quadratic Score (QS), POD, FAR, HSS, and TSS (GARCIÁ-RIGO et al., 2016).

Bias-variance decomposition

The bias of learning algorithms represents the extent to which the average prediction over all data sets differs from the desired outcome. In turn, the variance measures the extent to which the solutions for individual data vary around their average. Both concepts usually encompass a trade-off that must be adjusted during design (JAMES et al., 2013).

Many techniques exist to deal with the assumptions from bias and variance, so one must ideally include them during the design process. For instance, removing irrelevant features with feature selection (LEKA; BARNES; WAGNER, 2018; MASSONE; PIANA, 2018) naturally contributes to reducing the input bias. On the other hand, repeatedly carrying out the k-fold cross validation (MURANUSHI et al., 2015) gives an estimate of how sensitive a model is to a particular choice of data (standard deviation).

Data resampling

Real-world datasets often have imbalanced class ratios, and so does space weather. Positive events are rarer, thus often incurring in over-fitted performance whether the class ratios are not properly corrected (BATISTA; PRATI; MONARD, 2004). To deal with imbalanced class ratios, Camporeale (2019) suggested using data augmentation within the practice of data selection.

However, as data augmentation usually restricts to image processing – i. e., increasing image samples under different perspectives (rotating, scaling, cropping, etc.) (SHORTEN; KHOSHGOFTAAR, 2019) –, we believe that the imbalance issue must encompass a broader view, thus comprehending the resampling of data (which can lead samples to be over- and under-sampled, or a mixture of both) (BATISTA; PRATI; MONARD, 2004). The reason is that not all forecast models rely on image processing, and their design also often involves the processing of physical features from images or other sources. Examples of *ad hoc* forecast models using data resampling included, but not restricted to, the researches by J.-F. Liu, F. Li, H.-P. Zhang, et al. (2017), C. Liu et al. (2017), Yu, Huang, Q. Hu, et al. (2010), Yu, Huang, H. Wang, Cui, et al. (2010), Zheng, X. Li, and X. Wang (2019) and R. Li, H. Wang, et al. (2011), and R. Li and Zhu (2013).

Cost-sensitive learning

Whether the design issue once more comprehends the imbalanced class ratios mentioned earlier, but now the size of data significantly matters – a rather common scenario for space weather

big data –, employing cost-sensitive learning may reduce the computational cost (provided that the chosen algorithm supports such learning scheme) (ELKAN, 2001; HUANG; WANG, H.; DAI, 2012). The only automated tool supporting cost-sensitive learning was Muranushi et al. (2015)'s (i. e., by allowing the automated adjustment of the penalty parameters of their SVM). On the other hand, several *ad hoc* forecast models have used such penalized learning, such as those by Bobra and Couvidat (2015) and Florios et al. (2018), and H. Liu et al. (2019).

Cut-off point adjustment

The default threshold (0.5) for probabilistic classifiers may not yield the desired performance concerning the two possible error types (false alarms and missed events). Adjusting the prediction threshold of flare classifiers strictly depends on the users' needs. They can set the error type to be minimized (i. e., notably the Leka, Barnes, and Wagner (2018)'s approach), or some desired skill score to optimize. We observed that most automated space weather tools provided custom threshold adjusting.

3.4 Concluding remarks

This chapter gave an in-depth look at the literature state-of-the-art of solar flare forecasting systems, as well as some automated machine learning approaches. It introduced a comprehensive number of forecast models for space weather whose design processes have not been thought focusing on reproducibility, i. e., *ad hoc* proposals. Besides describing several aspects of those approaches (i. e., distinct algorithms, types of prediction, input features, and time horizons), we also managed to distinguish between biased and unbiased results following some predefined criteria. The articles discussed at this point shall be used again during the results and discussion of our case studies (Chapter 6), when they shall serve as a referenced performance for our designed models.

Our focus then moved to the presentation of several systems proposed to automate the design of general machine learning forecast models. Besides, we also further discussed the automated approaches specifically created for space weather's needs, detailing most of their positive aspects and drawbacks. Aware of the importance of creating an optimized and automated design process for solar flare classifiers, we managed to envision a comprehensive theoretical collection of good practices for supporting space weather's needs. Those practices

shall serve as the basis for the novel automated space weather methodology we propose in Chapter 4.

Chapter 4

A methodology to automate the design of solar flare classifiers

Space weather data comprehend a compendium of multimission and multidisciplinary data sources to monitor the heliosphere. As such, regions comprehending the Sun and the near-Earth space are of particular interest to space weather forecasters. The NOAA/SWPC and other international partners work under the premises of monitoring and forecasting solar events to provide space weather products, services, and alerts, such as the Australia Bureau of Meteorology (BoM) (BALA; REIFF, 2018), Korean Meteorological Administration (KMA) (BALA; REIFF, 2018), UK Meteorology Office (UKMO) (BALA; REIFF, 2018), and the Programa de Estudo e Monitoramento Brasileiro de Clima Espacial (EMBRACE) (INPE, 2020).

A needed milestone comprehends the capability to supply data acquisition systems at strategic points of near-Earth regions to provide real-time data to evaluate the space weather. If, on one hand, we may have several instruments continuously monitoring the space weather – such as the GOES satellites holding tools for X-ray imaging, recording, and monitoring extreme ultraviolet –, on the other hand, such capabilities must have a tolerance to space's potential radiation hazards. Hence, it is necessary to provide short-term hazardous event forecasting to satisfy the needs of such satellites and various other users (i. e., power grid operators and crewed space flights) (BALA; REIFF, 2018).

Solar flare forecasting comprehends one of the most active research in space weather due to the associated technological consequences. As argued by Massone and Piana (2018), the research of solar flare forecasting must rely on three fundamental pillars:

i. At an experimental level, the observation of ARs and their capability to host flare events. Such observation includes, but is not restricted to, ARs properties and magnetic fields information. Such pillar mainly refers to the compendium of multidisciplinary data.

ii. At a computational level, the observation mentioned in the first pillar must encapsulate in parameters, further used as input data for learning algorithms realizing binary (yes/no) or probabilistic ([0 , 1]) forecasts.

iii. At a technological level, one must provide modern software platforms realizing services for input and output big data, and automatic and user-friendly design of forecasting models. This pillar includes the platforms by Muranushi et al. (2015) (UFCORIN), Leka, Barnes, and Wagner (2018) (DAFFS), Massone and Piana (2018) (FLARECAST), Anastasiadis et al. (2017) (FORSPEF), and Engell et al. (2017) (SPRINTS).

However, at a first look, as the pillars mentioned earlier suggest, the lack of care with the methodology when designing the forecasting models can incur in several drawbacks with the reported performance. Accordingly, also at a computational level, the methodology can be thought of as a fourth pillar. Not employing a well-defined and well-thought methodology when designing forecasting models can certainly lead to several issues in classifiers, as those further commented in our literature survey in Chapter 3 (i. e., bias, under-fitting, over-fitting, classifiers over-performing with general quality scores but lacking performance with several others, and so forth).

Typical examples of the fourth pillar would include the majority of authors employing computational approaches belonging to the family of machine learning in the literature survey – especially using supervised methods (MASSONE; PIANA, 2018). Regardless of their learning algorithms (i. e., SVMs, neural networks, linear regressors, tree-based learners, among others), most of them have employed *ad hoc* methodologies to tailor the processes for their needs.

Within this context, the collection of good practices from Chapter 3 represented theoretical aspects worth considering in an optimized design process for flare classifiers. In this chapter, we shall present how we conceived our automated methodology based on such a collection. Noteworthily, we envisioned several requirements for the collection's items, mostly focused on guiding their implementation into the automated methodology:

- **Algorithm selection**: provide an investigation process for automatically searching learning algorithms from a predefined custom and flexible set.

- **Feature selection**: provide means for automating the selection of relevant features, thus avoiding some undesirable issues such as redundancy and co-linearity. We do not intend to create another rigid process for big data processing, but instead to concentrate on the automation of a generic feature selection process adhering to any type of data (and not to data only from sources plugged by default).
- **Hyperparameter optimization**: define a process in which algorithms can automatically adjust their behaviors, thus minimizing excessive complexity or over-fitting.
- **Proper data splitting**: while designing classifiers, provide data for training, validating, and testing, thus properly evaluating them without bias in results.
- **Cross-validation**: not only properly splitting data but also accommodating evaluations not incurring in statistical flukes (due to chance) shall be persecuted.
- **Test data**: empower true unseen samples as test data to assess classifiers' performance as if they forecast operationally, thus not incurring in the use of test records for decision-making.
- **Specialized metrics**: empower metrics for analyzing space weather classifiers in both points of view, deterministic and probabilistic.
- **Bias-variance decomposition**: provide techniques for minimizing the variance on data while reducing – to some extent – the input bias of algorithms, so that the bias-variance trade-off can be adjusted.
- **Data resampling**: employ unbiased resampling of data when class ratios do not match.
- **Cost-sensitive learning**: whether the size of data must be taken into account, provide adjustment of the cost function of classifiers (if possible).
- **Cut-off point adjustment**: as flare classifiers hold distinct misprediction costs mostly depending on the users' needs, provide adjustments for classifiers' prediction threshold.

4.1 Methodology overview

The proposed methodology follows, by default, the pipeline of processes presented in the scheme of Figure 4.1. The design of a classifier for flare forecasting under such methodology's assumptions starts with an initial splitting of data into training, validation, and test subsets. It then proceeds with the model and feature selection processes, hyperparameter optimization, data resampling, cost function analysis, cut-off point adjustment, and evaluation of validation and test sets.

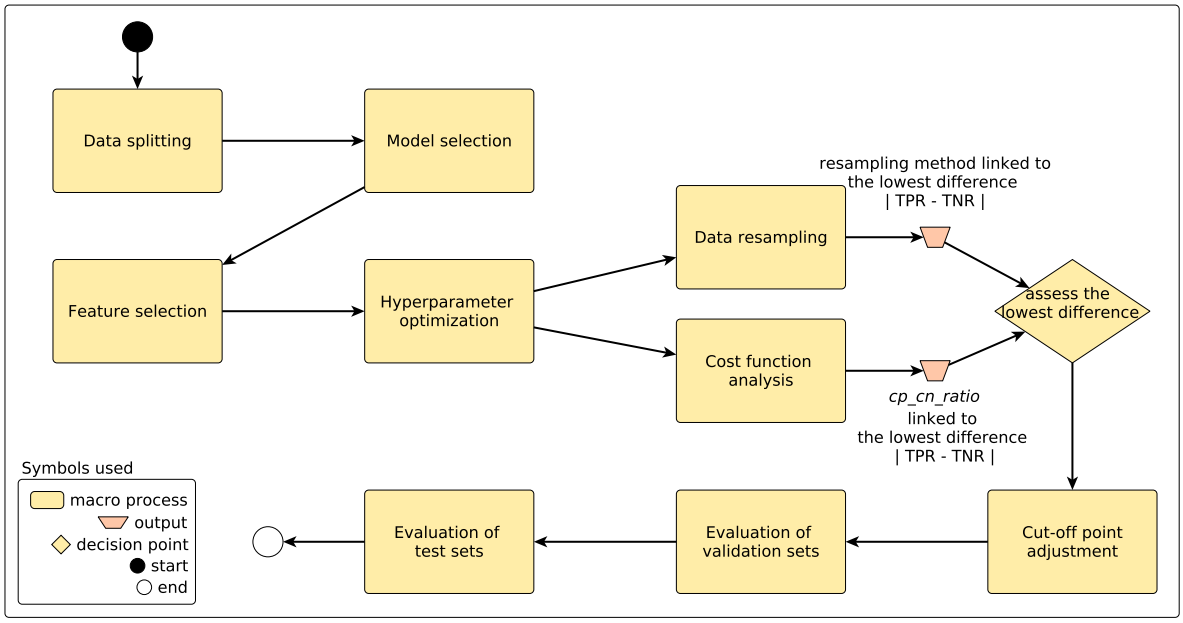

Figure 4.1: Methodology overview.

Besides the initial data splitting, we summarize the processes comprehending our methodology as follows:

- **Model selection**[1]: the focus here is to evaluate a generic set of custom learning algorithms and choose the one that minimizes the training error of some particular performance score without major adjustments or refinements (i. e., keep that algorithm that most increased such a score). Noteworthily, the methodology does not restrict the model selection to a specific performance score, which means that the user must choose the score satisfying his needs. Despite that, we used the TSS for our case studies (we will present the reason for this choice in Section 4.3).

[1]For the "model" word, we meant a classification model created by a machine learning algorithm over some appropriate set of data. Henceforth, such a term should not be confused with a physics model of solar flares.

- **Feature selection**: such a process intends to discard irrelevant or redundant features, thus selecting a subset that can maximize some particular performance score. As of model selection, the methodology does not restrict the feature selection to a specific performance score. Once more, the user must ideally opt for the desired metric. Nevertheless, we also opted for the TSS for the case studies discussed in Chapter 6.

- **Hyperparameter optimization**: not only selecting an algorithm and its input data, but also adjusting its hyperparameters for a better system behavior is needed – the aim of hyperparameter optimization. Here, the methodology seeks the set of parameters most increasing a particular performance score – in our case studies, the Area Under the ROC Curve (AUC) (we will present the reason for this choice in Section 4.5).

- **Data resampling**: real-world datasets often suffer from imbalanced class ratios, and so do those related to space weather. Positive samples can be less common than negative ones and vice versa. Aiming at not outputting over-fitted classifiers in favor of the majority class, data resampling works with training samples for correcting skewed class ratios (we shall describe how we employed resampling techniques in Section 4.6.1).

- **Cost function analysis**: as data resampling, cost function analysis tries to correct imbalanced class ratios. However, the classifiers' cost functions are not always available, which strictly depends on the chosen learning algorithm. Both data resampling and cost function analysis start concurrently (in case of both availability) and the methodology relies on the process outputting the lowest difference $|\text{TPR} - \text{TNR}|$[2] to continue its pipeline (we will detail the reason for this choice in Section 4.7).

- **Cut-off point adjustment**: most learning algorithms output posterior probabilities as their forecasts. For binary problems, classifiers convert those probabilities into yes/no forecasts using a predefined threshold t (cut-off point). This threshold is often assumed as $t = 0.5$. However, such default value for t might not always yield the best performance for predicting positive events concerning the expected number of false alarms – indeed, it is necessary to adjust t so that a balance point for false alarms can be found (the aim of this process). The criterion here is to choose the custom cut-off point outputting the lowest difference $|\text{TPR} - \text{PPV}|$ (we will explain the reason for this choice in Section 4.8).

[2] To keep both hit rates simultaneously at a close level, and consequently reduce the class skew.

- **Evaluation of validation sets**: at this point, the methodology trains some models of the same type in the training sets, that is, the chosen algorithm, features, hyperparameters, resampling method (if any), and cut-off point. It then uses such models to forecast their corresponding validation sets. Besides, it also trains some baseline models over the training sets and forecasts their corresponding validation sets. For baseline models, we mean the set-ups found in model selection. We use both baseline and methodology-outputted models to verify the pipeline's effectiveness before assessing the generalization error with test data[3]

- **Evaluation of test sets**: finally, provided that the methodology confirmed improvements in the classifier during the evaluation of validation sets, it can now assess its generalization error over unseen data – the aim with the evaluation of test tests.

At this point, it is worth mentioning that we have conceived the complete pipeline mentioned earlier by combining two initial design processes for automating flare forecasting already published in the literature, that is, the articles by Cinto et al. (2020a) and Cinto et al. (2020b). Accordingly, whereas the former accounted for a pipeline comprehending model selection, feature selection, hyperparameter optimization, data resampling, and evaluation of test sets, the latter encompassed feature selection, hyperparameter optimization, cost function analysis, cut-off point adjustment, and evaluation of validation and test sets. We discussed case studies designed to validate those two initial processes in Cinto et al. (2020a) and Cinto et al. (2020b).

4.1.1 Pipeline arrangement

Noteworthily, we have empirically defined the pipeline order of Figure 4.1. As such, we are aware that the output model may not comprehend the optimal classifier because our methodology does not carry out an exhaustive search over all choices available (i. e., the Cartesian product of possible choices). For instance, changing the order of feature selection, hyperparameter optimization, and data resampling might somehow alter the output model. However, the proposed arrangement leads to a high-quality classifier (as observed in the case studies).

[3]The methodology reserves samples of data for distinct purposes in the beginning. It uses training samples to fit the learning algorithm. In turn, it employs validation samples for decision-making. Finally, it uses test samples for performance assessment over real unseen data. Section 4.2 presents further details of such concepts.

Provided that we try all possible pipeline arrangements, obtaining the optimal classifier could be possible. However, this search's nature would be exhaustive and probably unfeasible in practice depending on the kind and size of data.

In this sense, the rationale explaining data resampling in the middle refers to the techniques we have chosen for data resampling, which create a lot of synthetic samples and this significantly increases the dataset size. If we have considered resampling in the beginning, other processes could become significantly slower.

Besides, as we treat our processes as blocks with well-defined inputs and outputs, changing the proposed pipeline order to other desired arrangements could be done with little effort (i. e., that corresponds to a custom option within the methodology implementation). To justify this rationale, we can cite the research by C. Zhang, Bi, and Soda (2017), which investigated what happens when machine-learning processes are inverted.

C. Zhang, Bi, and Soda (2017) observed the effects of using a resampling approach – in their case, under- and over-sampling – before feature selection and vice versa. They designed experiments comprehending nine feature selection methods, six resampling approaches, three classifiers, and 35 datasets.

Then C. Zhang, Bi, and Soda (2017) compared the performance of models regarding their overall and balanced accuracies and F-measures. Overall, they found that there was not any winner between both orders. They pointed out that researchers may try both to investigate the best classifier (sometimes an earlier usage of feature selection outperforms a later data resampling and vice versa). This conclusion corroborates our argument for suggesting other pipeline arrangements if needed.

4.2 Data splitting

In the beginning, our methodology splits a piece of provided data into macro-training and five test segments. It then performs a new split with the macro-training portion, dividing its samples into validation and training sets, as we show in Figure 4.2. Henceforth, let us explain the concepts of training, validation, and test data as follows.

At first, our methodology reserves through random subsampling without replacement about 365 samples for each test segment (this number of samples refer to our case studies). It

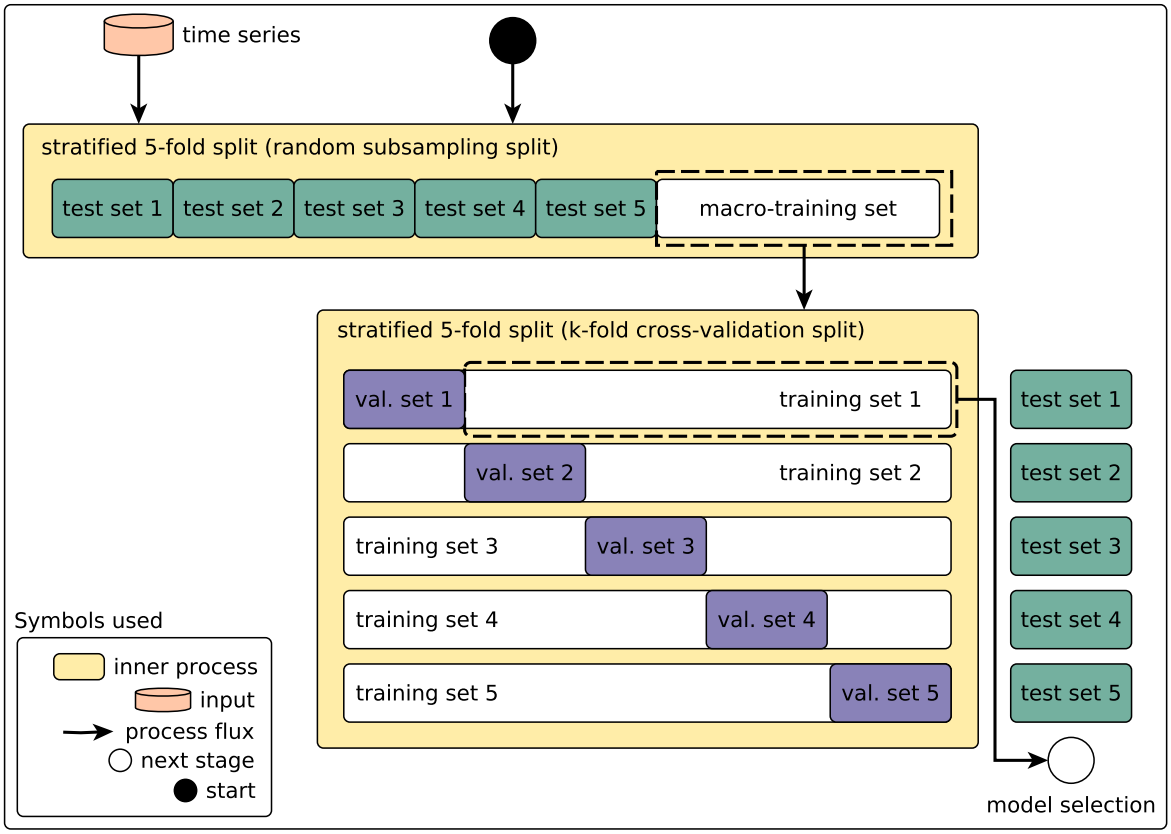

Figure 4.2: Data splitting scheme.

performs this splitting in a stratified fashioned way, which guarantees an equally distributed ratio of positive and negative classes within the macro-training and test sets.

The method then keeps test sets aside until the end, when it assesses the output model prediction error, that is, the model's generalization error over real unseen data (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). Accordingly, the methodology uses test data to forecast with its output model in a "simulated" operational environment.

After retaining test sets, the methodology resplits the macro-training data portion into fivefolds through a stratified cross-validated manner, which separates five training sets from their corresponding validation portions. It employs training data during model designing for fitting and optimizing the learning algorithm.

Noteworthily, the rationale for proposing fivefold-based splittings within our methodology is solely related to our case studies. Depending on the number of available data samples or the expected impact of variance on features, one may consider another n-fold scheme. Despite, splitting data in a fivefold cross-validated manner is rather common in literature since it

helps to reduce the model's variance, thus leveraging its performance (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

Similarly to test sets, which the pipeline reserves and only brings them back at the end of the process, the methodology also keeps validation sets and prevents their use during model design. However, the purpose of validation data differs from that of test data: whereas test samples support generalization error assessment, the methodology uses the latter for the final decision-making process, i. e., for verifying if the output model is ready to forecast the test samples. Despite that, their samples are suitably pure, i. e., we do not treat nor modify them in any form during model designing.

Alternatively, our method also supports time segmented data, such as reserving the oldest period for training, the intermediate for validating, and the most recent for testing (or some other criteria based on periods as pointed out in Ahmed et al. (2013)'s and Colak and Qahwaji (2009)'s researches). However, for brevity, we only included in this section the description to illustrate the k-fold-based splitting.

At the end of those data splittings, our methodology performs several well-defined machine learning-based processes over each training set, namely: model selection, feature selection, hyperparameter optimization, data resampling, cost function analysis, and cut-off point adjustment. Starting with model selection, as we show in Figure 4.2, we shall further discuss all of them in the next sections.

To validate our pipeline, we assembled data for our case studies I and II from two distinct NOAA/SWPC's repositories, namely the Daily Solar Data (DSD) and Sunspot Region Summary (SRS). Whereas the former comprises the Sun's behaviors as daily aggregated records, the latter further details the ARs of DSD's records, i. e., it provides their magnetic types, locations, corresponding areas etc. Briefly, we assembled data from the following features (for a detailed reference of them, see Chapter 5):

- *Sunspot area* (NOAA/SWPC, 2011);
- *Radio flux* (NOAA/SWPC, 2011);
- *Sunspot number* (NOAA/SWPC, 2011);
- *X-ray background flux* (NOAA/SWPC, 2011);
- *Daily Weighted Mean Flare Rate (WMFR) of magnetic classes* (SHIN et al., 2016);

- *Daily WMFRs of Zpc components from McIntosh (1990)'s classes.*

On the other hand, for Case Study III, we borrowed data from C. Liu et al. (2017)'s research. We did not modify or assemble any feature. Instead, we only processed their records through our pipeline to train the classifiers. The features involved HMI magnetograms magnetic data proposed in Bobra and Couvidat (2015)'s article at first, such as the total unsigned current helicity, total unsigned flux, total magnitude of Lorentz force, area of strong field pixels in the AR, among others (for a complete reference of features, refer to Chapter 5).

4.3 Model selection

The focus of model selection relies on choosing a model that best fits data and minimizes the training error without major adjustments or refinements. Accordingly, the proposed methodology evaluates some provided machine learning algorithms – one at a time – over each training set, as we show in Figure 4.3.

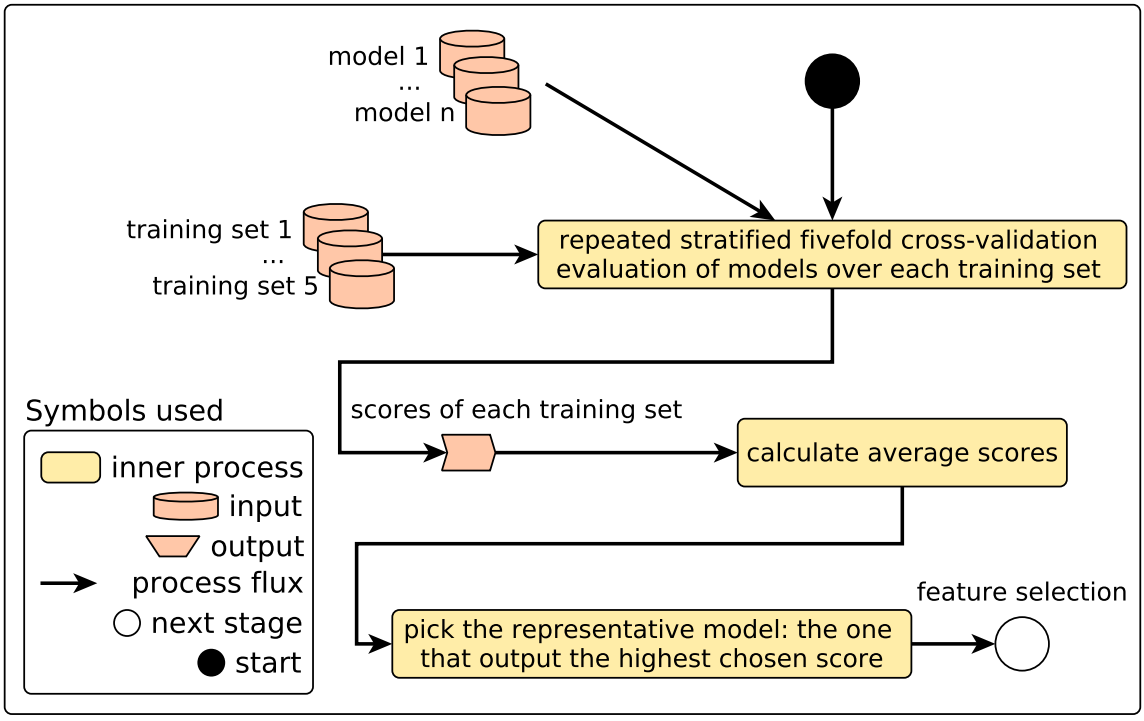

Figure 4.3: Model selection scheme.

The method carries out the performance assessment of learning algorithms through repeated randomized, stratified fivefold cross-validation evaluations. It repeats the k-fold cross-validation evaluations to output more reliable estimates of performance, thus attempting to minimize

data variance (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). The criterion for choosing the representative model relies on keeping the algorithm that maximized some custom performance score among all training sets – in our case, the TSS (see Appendix A.10). By representative model, we meant picking the best performing model concerning the chosen score and discarding others. In our case studies, the rationale for a decision-making process based on TSS refers to its nature, particularly its skill to represent both class-specific hit rates. However, choosing a score to optimize during model selection must consider the requirements of the expected output forecast model.

Most scores could not represent two outcome scores simultaneously. However, TSS can overcome this restriction by including in its calculation two components for individually assessing the success rates of positive and negative outcome classes at once (see appendices A.4 and A.5 for further details involving TPR and TNR, respectively).

Accordingly, employing efforts to boost TSS naturally leads to higher TPRs and TNRs at once. Examples of researchers whose focus relied on optimizing their TSSs include Nishizuka, Sugiura, Kubo, Den, Watari, et al. (2017), Bobra and Couvidat (2015), and Bloomfield et al. (2012).

Noteworthily, because of TSS's categorical nature, the proposed methodology keeps models' prediction thresholds[4] at the default level (0.5) during model selection to calculate confusion matrices and assess scores mentioned earlier. However, in later stages, the pipeline shall adjust those thresholds, as described in Section 4.8.

4.3.1 DecisionTree ensembles

We used DecisionTree ensembles in model selection as our case studies, that is, algorithms that merge the output of several models (base learners) to boost their overall predictive performance. Depending on the forecast scenario, some learning algorithms might outperform others. Therefore, it is rather important to establish some sort of cooperation between them in an attempt to reduce the noise of weak models (WITTEN; FRANK; HALL, 2011).

The reason for employing DecisionTrees instead of more complex algorithms like SVMs (ZAKI; MEIRA JR., 2013) or neural networks (WITTEN; FRANK; HALL, 2011) is solely related to our case studies. In fact, DecisionTrees are naturally able to handle outlier data and features

[4]Probabilistic classifiers use prediction thresholds to make their forecasts: predictions higher than or equal to the thresholds output positive answers, whereas the rest, negative.

with mixed nature (HASTIE; TIBSHIRANI; FRIEDMAN, 2009), as observed with our case studies' records, further commented on in Chapter 5.

Besides, whereas the predictive performance of trees may be slightly worse than other techniques, we manage to overcome this disadvantage using ensembles. However, the proposed methodology is not restricted to such type of algorithm and surely can handle other learning alternatives provided that Python's Scikit-learn supports them.

Ensembles help to reduce data variance when combined with the bagging strategy, which represents a complement to the k-fold cross-validation in mitigating variance (JAMES et al., 2013). As a sampling approach for training multiple inner models within an ensemble, the bagging strategy creates several base learners for such an ensemble by training each of them on a different sampling of data (with replacement).

Accordingly, we assessed the raw performance (without any adjust) of three DecisionTree ensembles using bagging for sampling data while training: RandomForest (JAMES et al., 2013), AdaBoost (ZAKI; MEIRA JR., 2013), and GradientTreeBoosting (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). The base learners used – i. e., each component of the ensembles – comprehended Classification and Regression DecisionTrees (CART) models (BREIMAN et al., 1984).

A RandomForest classifier resembles an algorithm fitting several DecisionTree models (base learners) on distinct subsets drew from training data. It uses soft voting[5] to aggregate their outputs (JAMES et al., 2013). Other authors efficiently using the RandomForest for flare forecasting include C. Liu et al. (2017).

Unlike the RandomForest that independently fits several models and aggregates their results in the end without emphasizing any base learner, boosting is a strategy by which each model drives the samples next models will focus on. The AdaBoost algorithm introduced this strategy (ZAKI; MEIRA JR., 2013).

AdaBoost is an algorithm that begins by fitting a single learner on the original dataset. It then sequentially fits more models focusing on the mispredicted samples from previous models. Accordingly, AdaBoost highlights the most challenging samples in its design process (ZAKI; MEIRA JR., 2013). Other authors efficiently using AdaBoost to forecast flares include Lan et al. (2012).

[5]Soft voting means to average the probabilities of base learners within an ensemble (HAN; KAMBER, 2006).

The GradientTreeBoosting algorithm, in turn, also relies on the boosting strategy while training several base learners gradually and sequentially. However, conversely to the AdaBoost, which identifies the weakness of base learners by adjusting weights of samples hard to predict, GradientTreeBoosting employs efforts to boost its loss function (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

4.4 Feature selection

When designing learning systems, one must take extreme care of input data. The existence of noise or useless features in the input of classifiers can certainly incur in poor system performance (HAN; KAMBER, 2006).

Not only noise but also high data input dimensionality can become an undesirable issue for classifiers. By dimensionality, we mean classifiers' input sets leading to redundancy between elements or negatively affecting training time.

In this context, feature selection (HAN; KAMBER, 2006) involves a powerful machine learning method for letting us distinguish between suitable and useless features. Aware of the benefits, we managed to provide feature selection within our methodology.

Accordingly, our method seeks an effective set of features during the feature selection stage to use along with the previously chosen model. Hence, it evaluates the randomly initialized baseline algorithm of model selection with the full set of features over each training set using repeated randomized, stratified fivefold cross-validation. Also, it distinguishes between useful and useless features in each training segment.

Our pipeline performs two distinct methods, filtering and wrapper-based schemes, to rank the usefulness of features and support their discarding (GUYON; ELISSEEFF, 2003). Whereas the former ranks each feature using some proxy metric – such as Pearson correlation analysis – to provide an ordered list of their importance in the end, the latter uses an ordinary learning model to evaluate several predefined feature subsets.

Discarding features only with a filter method solely depends on the proxy's rank, that is, we discard features with lowest scores. On the other hand, to discard features using a wrapper technique, we assess the learning model performance among the predefined feature subsets and pick the set that best increased some desired performance score.

Within this research, we employed both filter and wrapper methods. As the filter, we designed an univariate feature selection method – provided with the F-score proxy measure (BOBRA; COUVIDAT, 2015) – combined to the model selection chosen algorithm (wrapper).

Univariate feature selection methods do not consider the correlation between features. Alternatively, they assume features are independent and evaluate their complementary nature, if any (GUYON; ELISSEEFF, 2003).

Within this context, arbitrary features may incur in poor performance when predicting alone a target. However, they may turn out to be good predictors when combined, without necessarily possessing a high correlation coefficient (GUYON; ELISSEEFF, 2003).

Accordingly, the flare forecasting literature had already pointed out several benefits of employing univariate methods linked to the F-score measure for feature selection (BOBRA; COUVIDAT, 2015). In Equation 4.1, we show how to calculate the F-score (CHANG; LIN, 2008):

$$F(i) = \frac{(\bar{x}_i^+ - \bar{x}_i)^2 + (\bar{x}_i^- - \bar{x}_i)^2}{\frac{1}{n^+-1}\sum_{k=1}^{n^+}(x_{k,i}^+ - \bar{x}_i)^2 + \frac{1}{n^--1}\sum_{k=1}^{n^-}(x_{k,i}^- - \bar{x}_i)^2}, \tag{4.1}$$

where the denominator comprehends the variance sum within each class; the numerator comprehends the inter-class variance; $\bar{x}_i^+$ and $\bar{x}_i^-$ correspond to the average values of positive and negative samples, respectively; n^+ and n^- represent the number of positive and negative samples, respectively; and $\bar{x}_i$ accounts for the average of the feature.

Over each training segment – one at a time –, our methodology calculates the F-score of all elements in the full features set, outputting a ranked features list in descendant order (Figure 4.4). Then, it picks the two best features and assesses their performance when used as the input for the previously chosen ensemble through repeated randomized, stratified fivefold cross-validation.

Subsequently, we increase the number of chosen best features by one and re-evaluates the model with them. The method continues this pipeline from the two best features to their total number, always recording the desired custom score of each iteration – in our case, TSS.

By using this approach, in the end, our methodology maintains only the five best feature sets, that is, those linked to the highest TSS training scores (one for each training set). Then, it keeps only the set associated with the highest TSS and re-evaluates the ensemble using it in all training sets.

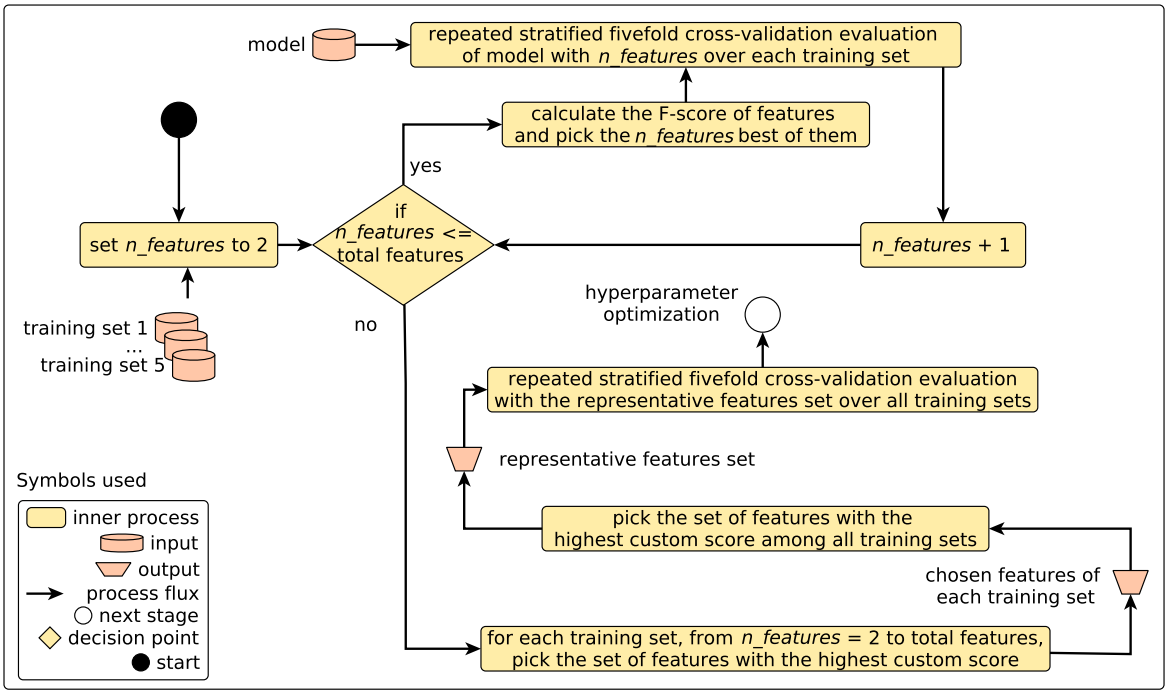

Figure 4.4: Feature selection scheme.

4.5 Hyperparameter optimization

While optimizing the hyperparameters of a given learning algorithm, the aim relies on fitting a model ($\mathcal{M}$) and reducing its loss function ($\mathcal{L}(X^{(val)}; \mathcal{M})$) in validation data samples ($X^{(val)}$). As such, $\mathcal{M}$ is fit by an algorithm ($\mathcal{A}$) in training data samples $X^{(tr)}$ (CLAESEN; DE MOOR, 2015).

Within this context, $\mathcal{A}$ usually possesses a hyperparameters set (λ, $\mathcal{M} = \mathcal{A}(X^{(tr)}; \lambda)$), which must be optimized (adjusted) to reduce $\mathcal{L}$. Besides, $\mathcal{M}$ requires an affordable level of complexity to avoid poorly generalizing unseen data, which can lead it to overfit. Thus, $\mathcal{A}$ provides λ to allow the adjustment of its complexity (CLAESEN; DE MOOR, 2015).

Consequently, hyperparameters are employed within a search process to look for a set ($\lambda^\star$) yielding an optimal model $\mathcal{M}$, as Equation 4.2 defines (CLAESEN; DE MOOR, 2015):

$$\lambda^\star = \lambda \arg\min \mathcal{L}(X^{(val)}; \mathcal{A}(X^{(tr)}; \lambda)) = \lambda \arg\min \mathcal{F}(\lambda; \mathcal{A}, X^{(tr)}, X^{(val)}, \mathcal{L}), \tag{4.2}$$

where the function $\mathcal{F}$ uses a set of hyperparameters and outputs the loss value provided that $X^{(tr)}$, $X^{(val)}$ and $\mathcal{L}$ are given.

Aware of the hyperparameter optimization theory and its benefits, we designed a process in the methodology comprehending it. As such, the aim is to adjust how the ensemble behaves and better leverages its generalization skills.

4.5.1 Randomized search for hyperparameters

The two most common techniques usually employed for hyperparameter optimization refer to grid search and random search. In the former, the user provides a finite set of values for each hyperparameter and the algorithm evaluates the Cartesian product between them. Inevitably, this can lead to a significant computational burden since the number of required evaluations can grow exponentially according to the input space's size (FEURER; HUTTER, 2019).

On the other hand, a random search is also fed with the set of values mentioned earlier. However, the algorithm randomly samples combinations until a stop criterion is met. This approach can be as effective as the grid-based, especially when not all parameters are equally important to adjust (FEURER; HUTTER, 2019).

Besides, we believe that random search is a useful method to start the search process for hyperparameters. It evaluates almost the entire input space and finds parameters settings with reasonable results. Those results are arbitrarily close to the optimum ones found with the exhaustive grid search (FEURER; HUTTER, 2019).

However, when choosing between the grid and random approach in practical scenarios, it is worth taking care of the trade-off between the performance of models and resource consumption (FEURER; HUTTER, 2019). As our methodology needs to fit most of the data scenarios, hyperparameters, and grids, we judged random search a more suitable approach – to avoid evaluations growing exponentially –, thus providing it within the proposed pipeline. Our case studies included the following grid of hyperparameters to be used along with our random search implementation:

- the number of base learners;
- the maximum depth of base learners;
- the number of samples used to fit each base learner (for GradientTreeBoosting);
- the rate driving how much each base learner contributes to the loss gradient (for GradientTreeBoosting and AdaBoost);
- the number of features used to fit each base learner;
- the number of samples needed to split an internal node;
- the number of samples needed for a leaf node;

- the threshold used to stop the growth of base learners – each inner node continues to grow whether its impurity remains above this threshold; otherwise, it is turned into a leaf (for RandomForest).

As we show in Figure 4.5, our method performs a randomized search in each training set to seek the hyperparameters set up that most increases some chosen custom score – in our case, the AUC (WITTEN; FRANK; HALL, 2011) (see Appendix A.8 for a concise reference for AUC). Furthermore, after those searches, the methodology picks the representative set of parameters, that is, the one increasing the most the AUC among all training segments.

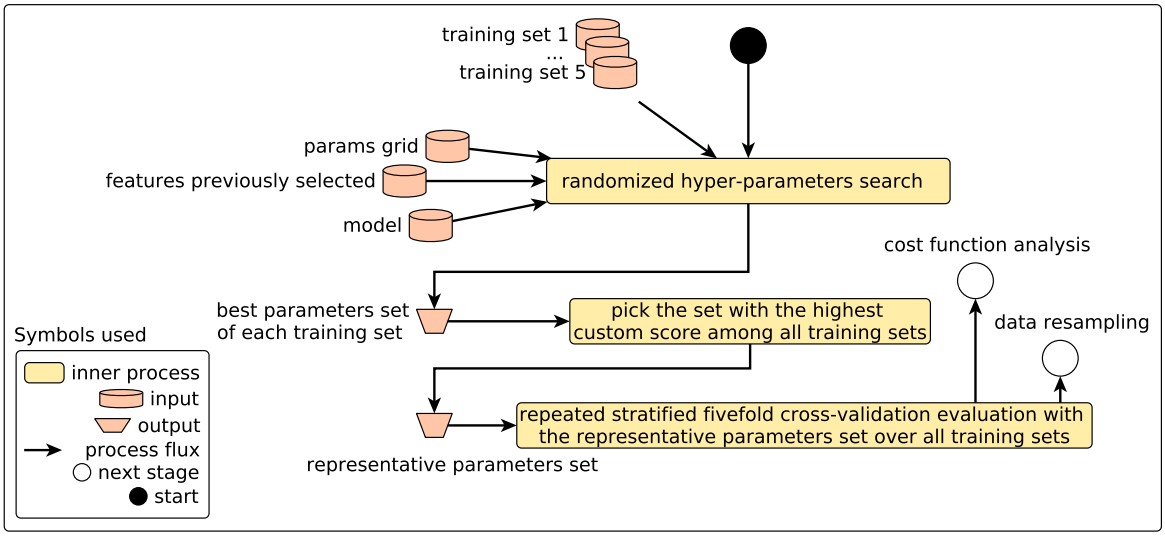

Figure 4.5: Hyperparameter optimization scheme.

Binary classifiers – as those of our case studies – output posterior probabilities for distinguishing between their classes. Often, those classifiers rely on some custom prediction threshold t to classify their predictions: all samples above t are from the positive class, whereas the remainder belongs to the negative class. However, choosing arbitrary t values might not yield the best performance for classifiers.

Within this context, the ROC (WITTEN; FRANK; HALL, 2011) analysis assesses the classifiers' performance at increasing prediction thresholds. It plots the TPRs versus the probabilities of false detections (False Positive Rate (FPR); see the Appendix A.8) concerning the range of increasing thresholds (i. e., $t = 0.1, 0.2$, and so on).

Besides, the ROC analysis creates a curve by interpolating between the points TPRs versus FPRs as mentioned earlier. The area underneath this curve is the AUC. The closer a classifier

scores its TPR from 1 and FPR from 0, the better the classifier (AUC = 1, the ideal case). This classifier has a increased probability of detecting a random positive sample over a negative one.

Accordingly, the AUC score represents a measure of potential usefulness for classifiers. We proposed AUC here instead of the earlier TSS since we want to guarantee that our model will have increased probabilities for detecting random positive samples over negative ones at different prediction thresholds. As such, we seek a parameters set up that score high TPRs at increasing prediction thresholds without incurring in high FPRs, thus leading to high AUCs. Not only AUC but also other scores may be optimized in this process (that depends on the requirements of the expected output forecast model).

4.6 Data resampling

Frequently, real-world datasets suffer from imbalanced class ratios. Positive samples can be less common than negative ones and vice versa. Fitting models on imbalanced data can be costly and end up outputting over-fitted classifiers concerning the majority class, that is, classifiers poorly generalizing minority samples (CHAWLA et al., 2002).

A straightforward – yet effective – way to cope with imbalanced data relies on changing how classifiers report their performance. Accordingly, reporting only the biased overall accuracy may not be enough to measure the classifier's effectiveness. Instead, one can use individual metrics to verify performance when forecasting individual classes (for instance, both class hit rates) (BATISTA; PRATI; MONARD, 2004).

However, provided that the aim is also to increase classifiers' performance, only reporting individual scores may not be enough. In this sense, the literature also addresses the class imbalance issue in some other ways, such as by using cost-sensitive learning (BOBRA; COUVIDAT, 2015) and resampling the dataset (CHAWLA et al., 2002) – both approaches provided in our pipeline.

4.6.1 Methods for data resampling

Our methodology employs data resampling and cost-sensitive learning concurrently (i. e., it carries them out at the same time). Depending on their results, the pipeline shall proceed using the former or the latter approach afterwards.

For data resampling, the methodology evaluates at least three distinct techniques to resample data before fitting the learning model and evaluating its performance: SMOTE (CHAWLA et al., 2002), SMOTE-Tomek (BATISTA; PRATI; MONARD, 2004), and SMOTE-ENN (BATISTA; PRATI; MONARD, 2004). The rationale for employing some distinct resampling techniques is to have diversity in results, i. e., some approaches can outperform others depending on the data's design.

SMOTE is an over-sampling technique that creates synthetic minority samples by interpolating between original minority samples close to each other (CHAWLA et al., 2002). Such an interpolation incurs in the spread of the minority class decision boundary over the majority class space, reducing the risk of over-fitting with minority samples duplicated at random with replacement, that is, an ordinary random over-sampling (BATISTA; PRATI; MONARD, 2004).

On the other hand, as of combined methods, both SMOTE-ENN and SMOTE-Tomek over-samples minority class data with SMOTE at first. They then employ two distinct undersampling techniques over the majority data. Whereas ENN removes samples from the training set that are misclassified by its three-nearest-neighbors algorithm, Tomek removes examples comprehending Tomek-links[6] (BATISTA; PRATI; MONARD, 2004).

The use of SMOTE, SMOTE-ENN, and SMOTE-Tomek relates to our case studies. As we have more skewed classes in the shortest forecasting horizons (see Chapter 5 for a complete view of our data), we chose over-sampling techniques to reduce the potential loss of data compared to pure under-sampling approaches.

Concerning our methodology, as Figure 4.6 shows, it assesses the model's performance over each training set through resampled repeated stratified, fivefold cross-validation. Hence, repeatedly, it splits each training set into fivefolds and resamples only the subsets used for fitting the model (the remainder validation subset remains unaffected).

The process mentioned earlier continues until the pipeline uses all training samples of each training segment for validation purposes. Finally, the method chooses the representative resampling method and discard others, that is, the one scoring the lowest difference |TPR – TNR| to keep both hit rates simultaneously at a close level, and consequently reduce the class skew.

[6] We define Tomek-links as follows: given two examples E_i and E_j belonging to different classes, and $d(E_i, E_j)$ is the distance between the two examples. A (E_i, E_j) pair represents a Tomek-link if there is not any example E_l, such that $d(E_i, E_l) < d(E_i, E_j)$, or $d(E_j, E_l) < d(E_i, E_j)$. If two examples form a Tomek-link, then they can be considered noise (BATISTA; PRATI; MONARD, 2004).

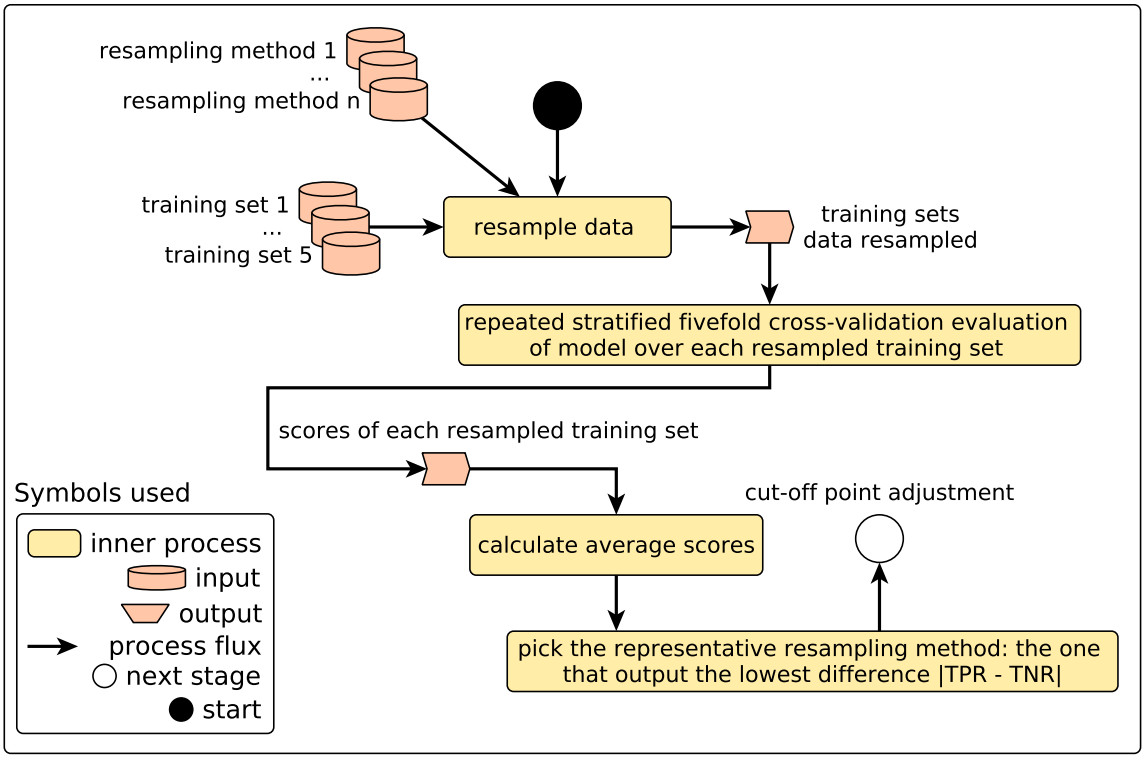

Figure 4.6: Data resampling scheme.

4.7 Cost function analysis

Approaches other than resampling and changing how classifiers report their performance involve the use of cost-sensitive learning. Instead of creating minority synthetic samples at some defined criterion, this approach assigns a misclassification cost to each predicted sample. Then, the classifier must try to reduce the total cost (ELKAN, 2001).

Whereas both data resampling and cost-sensitive learning approaches may reach similar results, the latter usually outperforms the former regarding resource consumption. That is, since cost-sensitive learning does not explicitly create synthetic samples, it can represent a better choice for bigger sets of data. Other authors effectively using cost-sensitive learning include Bobra and Couvidat (2015). Besides, as argued by Huang, H. Wang, and Dai (2012), cost-sensitive learning must be used to adjust classifiers' performance when authors use custom prediction thresholds, as our case (discussed in Section 4.8).

Usually, learning algorithms have a feature offering a practical implementation of cost-sensitive learning[7]. Those algorithms provide two cost functions (functionalities) referring to

[7]Regarding our case studies' models, except for the GradientTreeBoosting – here, our method only copes with data resampling – , RandomForest and AdaBoost provide cost-sensitive learning.

weighted multiplications to samples of positive and negative class. For majority negative data, the minority class weight often must assume 1 as its cost on one hand.

On the other hand, the majority class weight must employ the ratio of positive examples over the negative ones as a rule of thumb, that is, C_p/C_n – where C_p and C_n are the numbers of positive and negative class samples, respectively. Analogously, for majority positive data, the negative weight is now expected to be 1 and the positive one assumes C_n/C_p.

However, the ratio C_p/C_n might not yield the perfect balance for classifiers' data when it comes to keeping both TPR and TNR at a close level. By perfect balance, we mean ratios that output the lowest difference |TPR – TNR|. Hence, as the ratio C_p/C_n lies around [0 , 1], it is worth seeking in such an entire interval the value incurring in the perfect balancing performance[8].

Accordingly, as we show in Figure 4.7, the methodology seeks the ratio C_p/C_n of each training set (one at a time) linked to the lowest difference |TPR – TNR| to keep both hit rates simultaneously at a close level. Besides, as we show in Figure 4.8[9] for a hypothetical training segment, the methodology investigates at a 0.1 step-based increment how TPR and TNR change as the ratio C_p/C_n varies along its interval, that is, from C_p/C_n = 0.1 to 1. In the end, the pipeline picks the ratio C_p/C_n that output the lowest difference between hit rates as the representative one and re-evaluate the model through all training sets.

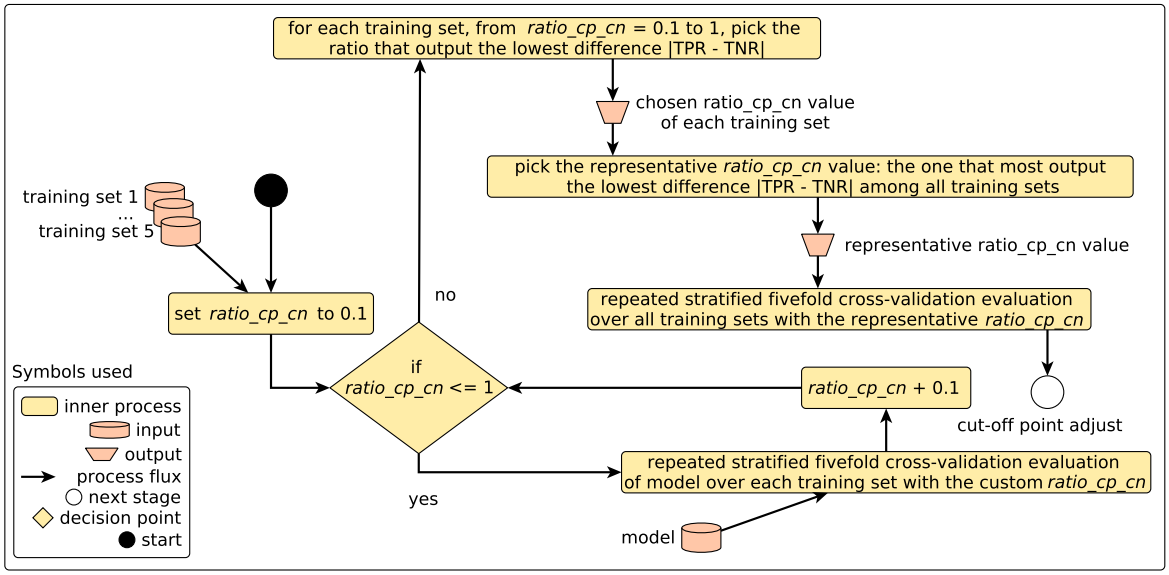

Figure 4.7: Cost function analysis scheme.

[8]Analogously, for majority positive data, the ratio C_n/C_p also lies around [0 , 1] and a desired point must be sought.

[9]We interpolated points in this graph to ease visualization.

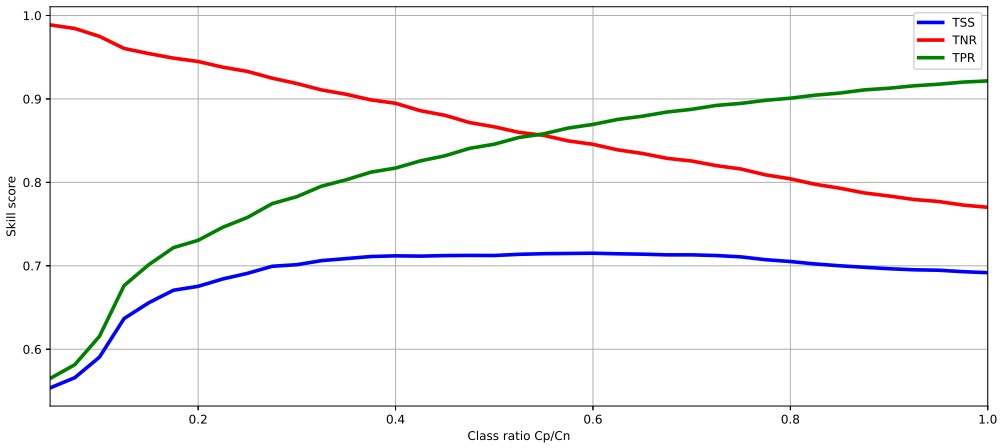

Figure 4.8: Class ratio graphical plot for a hypothetical training segment.

Depending on the majority data nature, cost function analysis may somehow end up decreasing TNR (for majority negative data) or TPR (for majority positive data). However, despite those decreases, their complementary scores shall consequently increase. Given that both hit rates directly influence TSS (the higher the class-specific scores, the higher the TSS is), choosing the ratio C_p/C_n (or C_n/C_p) that makes the latter scores to come close indirectly increases the former.

4.8 Cut-off point adjustment

Most learning algorithms output posterior probabilities as their forecasts – and so do the models of our case studies. For binary problems, classifiers convert those probabilities into yes or no forecasts using predefined thresholds t.

Overall, usually $t = 0.5$ – that is, provided that classifiers' probabilities are greater or equal 0.5, they output positive answers as their forecasts. Also known as the cut-off point of classifiers, $t = 0.5$ might not yield the best performance for predicting positive events concerning the expected number of false alarms.

Directly related to the TPR (recall), the Positive Predictive Value (PPV) (precision, see Appendix A.1) represents the accuracy with which we predict positive events (ZAKI; MEIRA JR., 2013). The former, in turn, accounts for the number of positive events correctly predicted. This rationale is analogous to the TNR and Negative Predictive Value (NPV).

The relation between TPR and PPV – or TNR and NPV – is directly linked to the classifiers' cut-off points. As such, there are two trade-offs between the class-specific recall and precision scores. For instance, it would be rather easy to score TPR = 1, as we predicted that all testing samples were positive. However, positive precision would be rather low (ZAKI; MEIRA JR., 2013). Consequently, the number of false alarms would increase a lot.

On the other hand, the positive precision would be highly increased provided that we predict only a few testing samples as positive (for instance, the samples in which our classifier is the most confident). Conversely, now the positive recall would be rather low. As a consequence, the number of false alarms would decrease a lot.

An optimal cut-off point must be encountered in such a way that both recall and precision are high at once (ZAKI; MEIRA JR., 2013) – the aim of the last design process in the pipeline. As such, for each training set (one at a time), from $t = 0.1$ to $t = 0.9$ at a 0.1 step-based increment, we seek the t value linked to the lowest difference $|TPR - PPV|$ (Figure 4.9[10]). As an example, in Figure 4.10, we show how TPR and PPV vary in relation to t in a hypothetical training segment.

Subsequently, our method keeps the t value that maintains TPR closer to PPV over all training sets as the representative one. By using the rationale mentioned earlier, we can reach an optimal balance point between recall and precision at the same time we take care of the number of false alarms.

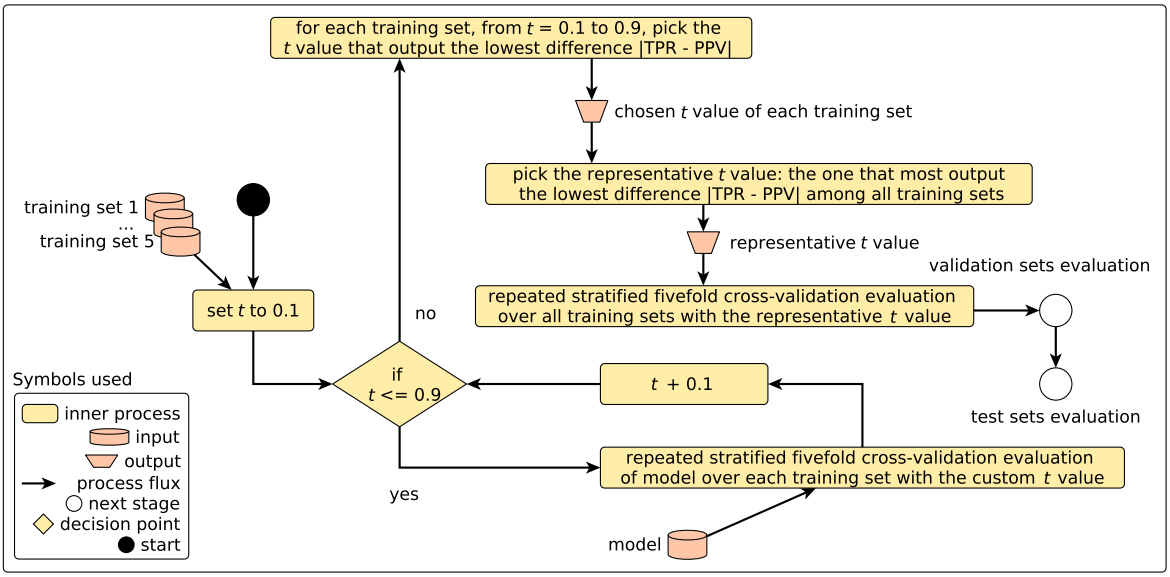

Figure 4.9: Adjustment of the cut-off point scheme.

[10]We interpolated points in this graph to ease visualization.

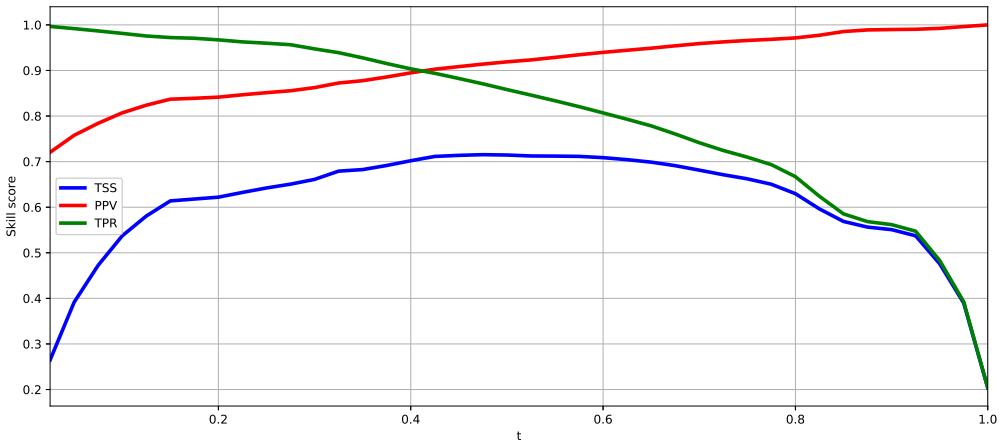

Figure 4.10: Decision threshold graphical plot for a hypothetical training segment.

4.9 Evaluation of validation sets

During the evaluation of validation sets, the pipeline trains five optimized models of the same type in each training set, that is, the same chosen algorithm, selected features, optimized hyperparameters, custom C_p/C_n ratio/data resampling method, and adjusted cut-off point. It then uses those models to forecast their corresponding validation sets.

Besides forecasting the validation sets with optimized models, the method also fits five baseline models over each training set and forecast their corresponding validation sets. As of the optimized models, baseline models also have the same configuration. However, this configuration comprehends the algorithm of model selection, noticeably a model designed with parameters initialized at random and using the full set of features.

The methodology uses both classes of models – baseline and optimized – to assess the improvements before assessing the generalization error. This means that it verifies whether some custom score increases from baseline to optimized results and, if so, it proceeds with the evaluation of test sets. In our case studies, the methodology checks the TSS performance.

4.10 Evaluation of test sets

Finally, provided that the methodology confirmed improvements in the forecast performance during Section 4.9, it can now assess the optimized models' generalization errors over unseen data. Accordingly, it uses the previously trained baseline and optimized models to forecast their corresponding test sets.

4.11 Concluding remarks

This chapter presented the proposed methodology[11] envisioning an optimized automated design process for solar flare classifiers. We based this methodology in the collection of good practices specifically designed for space weather's needs, further discussed in Chapter 3.

Overall, such methodology has encompassed the automation of the following machine learning based pipeline of processes: (i) data splitting; (ii) model selection; (iii) feature selection; (iv) hyperparameter optimization; (v) data resampling; (vi) cost function analysis; (vii) cut-off point adjustment; and (viii) evaluation of validation, and (ix) test sets. We shall present the classifiers designed under this methodology's premises in Chapter 6, further commenting on how the pipeline affected their performance, gains, and drawbacks.

Although the methodology allows an "operational" assessment of performance in a portion of data, we have also deployed the experimental forecast models designed for case studies I and II into a real forecast environment[12]. This implementation is further detailed on the Appendix C and shall serve as one basis for the future work discussed in Chapter 7.

[11] The full source code comprehending the methodology automation is available at: `https://github.com/tiagocinto/guaraci-toolkit`.

[12] The full source code for the deployment of the forecast models is available at: `https://github.com/tiagocinto/guaraci-forecast`.

Chapter 5

Datasets

This chapter shall present the datasets used to evaluate the methodology presented in Chapter 4. Overall, we employed two major datasets, one comprehending full-disk forecasts – used in case studies I and II –, and other supporting AR-by-AR-based processing – for Case Study III. Case Study I aimed to forecast ≥ C flares, while case studies II and III employed ≥ M flare forecasting.

5.1 Dataset for Case Study I and II

The dataset employed in case studies I and II used data from NOAA/SWPC[1]. That prediction center provides real-time monitoring of solar events affecting navigation, telecommunications, and satellites. As the official source for space weather alerts in the USA, NOAA/SWPC freely provides their data for study and research purposes[2].

Data from NOAA/SWPC feed distinct repositories, such as the DSD and SRS. The former comprises the Sun's behaviors as daily aggregated records (i. e., X-ray background measures, the total number of sunspots etc.), the latter further details the ARs of DSD's records (i. e., it provides their magnetic types, locations, corresponding areas, etc).

NOAA/SWPC regularly issues data from DSD at 02:30 AM, 08:30 AM, 02:30 PM, and 08:30 PM UTC (NOAA/SWPC, 2011). On the other hand, the SRS repository is daily updated always at 00:30 UTC (NOAA/SWPC, 2008).

Since data issued in SRS at 00:30 refer to previous days, it was worth using this time as a reference for assembling our dataset (with data from DSD previously issued at 20:30). To link between DSD and SRS data, we used their available compilation dates (month, day, and year).

[1] `http://www.swpc.noaa.gov`
[2] `ftp://ftp.swpc.noaa.gov/pub/warehouse/`

Accordingly, we assembled data from the period comprehending 1997 January 01 to 2017 January 15, that is, records involving almost two entire solar cycles (23 and 24). Each sample of our dataset corresponds to one record of DSD, combined with its corresponding SRS data. Thus, we managed to compile 7,320 records comprising data from several distinct features:

- *C-class flares*: daily observed C-class events (to design the target variable) (NOAA/SWPC, 2011).
- *M-class flares*: daily observed M-class events (to design the target variable) (NOAA/SWPC, 2011).
- *X-class flares*: daily observed X-class events (to design the target variable) (NOAA/SWPC, 2011).
- *Sunspot area*: the daily sum of all sunspot areas (measured in millionth units of the solar hemisphere) (NOAA/SWPC, 2011).
- *Radio flux*: also known as the F10.7 index, the solar radio flux at 10.7 cm is an index of daily solar activity reported by the Dominion Radio Astrophysical Observatory at Penticton, Canada (NOAA/SWPC, 2011).
- *Sunspot number*: the daily aggregated number of sunspots. Also known as Wolf's sunspot number, it is calculated by $R = k(10g + s)$, where k is a variable scaling factor that represents the observation conditions, g is the number of active regions, and s is the number of sunspots inside active regions (NOAA/SWPC, 2011).
- *X-ray background flux*: the daily average background X-ray flux. To achieve this value, NOAA/SWPC primary GOES satellite sensors hourly records 24 X-ray measures for each day and split them into three groups of 8 hours. The lowest flux measure of each group is kept, and the average between the first and third minimal measures is also calculated. Then, NOAA/SWPC compares this average measure with the second group minimal flux, and reports the lowest value as the X-ray background flux (NOAA/SWPC, 2011).
- *Daily WMFRs of magnetic classes*: The WMFR calculates the weighted mean rate that each Mt. Wilson class occurs along with C-, M-, and X-class flares, defined as in Equation 5.1 (SHIN et al., 2016):

$$\mathrm{wmfr}_{\mathrm{daily}} = \sum_{i=1}^{y} \frac{n_c + 10n_m + 100n_x}{n}, \tag{5.1}$$

where n refers to how many classes were observed regardless of being linked to any sort of flare and n_c, n_m, and n_x correspond to the number of classes linked to C-, M-, and X-class events, respectively. The constants 10 and 100, in turn, are the weights for flare importance. Noteworthily, as our data refer to solar behaviors aggregated daily, several sunspots with different WMFRs are often recorded every day. We treat the daily WMFR as the sum of all observed WMFRs, i. e., y refers to the daily number of sunspots and i accounts for the WMFRs of their magnetic classes[3]

- *Daily WMFRs of Zpc components from McIntosh classes*: Some authors argue that *Zpc* components can be interpreted as proxies to represent magnetic fluxes emergence or decay inside sunspot groups (KILCIK et al., 2018; EREN et al., 2017; MCCLOSKEY; GALLAGHER, P. T.; BLOOMFIELD, 2016; LEE, K. et al., 2012). Within this context, some cases of flux emergence may incur in higher flare amounts, that is, they can represent correlations between individual *Zpc* components and flares. Accordingly, we calculate the daily WMFR of McIntosh classes as mentioned earlier in Equation 5.1. However, we treat each *Zpc* component independently, thus creating three daily WMFRs for each sunspot.

5.1.1 Data preprocessing

Data from this dataset needed two distinct preprocessing techniques: data standardization and missing data imputation (HAN; KAMBER, 2006). Examples of missing data issues included: DSD records without their corresponding SRS data, sunspot area and number with absent values, and features with noisy data (for instance, X-ray background flux with zeroed measures).

To input missing data, we used a k-NN based imputation (HAN; KAMBER, 2006). Usually, missing data are treated with the deletion at random of the entire record, leading to data loss. However, k-NN provides a more suitable alternative, that is, it inputs missing data considering the similarity between data records. Hence, to input missing feature values, the k-NN uses the most predominant value among the k closest neighbors to the record with missing data.

Accordingly, the k-NN defines its similarity in terms of a general distance metric – in this research, the ordinary Euclidean score. Equation 5.2 defines the Euclidean distance between two records $X_1 = (x_{11}, x_{12}, \dots, x_{1n})$ and $X_2 = (x_{21}, x_{22}, \dots, x_{2n})$ (ZAKI; MEIRA JR., 2013).

[3]We included the WMFRs of the most frequent magnetic classes based on Jaeggli and Norton (2016)'s. findings: they researched the years between 1992 and 2015 highlighting the most frequent cases (almost the entire period we are using).

$$\text{dist}(X_1, X_2) = \sqrt{\sum_{i=1}^{n}(x_{1i} - x_{2i})^2}, \tag{5.2}$$

where n is the number of features; x_{1i} is the i^{th} feature of the first record; and x_{2i} is the i^{th} feature of the second record.

Some solar physicists suggest not using any type of data imputation to reduce the probability of distorting data behavior in the solar cycle. However, we believe that inputting values considering the similarities of the k nearest tuples avoid such distortions. As such, simply inputting values by the feature mean value or through feature constants does not necessarily reflect the solar cycle, which we avoided. Nevertheless, by using the k-NN, we tried to approximate the tuples with missing data to their most similar days, which would naturally reduce the effects of inputting wrong values.

Besides missing data, we also encountered dissimilar data ranges among features. For instance, whereas the sunspot number and area, and radio flux respectively ranged between [0 , 401], [0 , 5690], and [65 , 298], the X-ray background flux lied in a shorter interval, namely $[10^{-7}\ ,\ 2 \times 10^{-5}]$. Those dissimilar data ranges are known to highly affect the performance of machine learning predictors. In this sense, we used the z-score to normalize features (HAN; KAMBER, 2006).

The z-score – also known as the standard score – is an algorithm for reducing data ranges to $\mu = 0$ and $\sigma = 1$. Z-score is highly recommended for solar flare forecasting models since it positively affects the classifiers' predictive performance (NISHIZUKA; SUGIURA; KUBO; DEN; WATARI, et al., 2017). In Equation 5.3, we show how to calculate the z-score.

$$z = \frac{x - \mu}{\sigma}, \tag{5.3}$$

where μ is the feature mean; σ is the feature standard deviation; x is an arbitrary feature value; and z is the standardized value of x.

5.1.2 Sliding time window

Forecasting models that cope with the evolution of data over some period and predict events in a supervised and sequential fashioned way are called short-term predictors (YU; HUANG; WANG, H.; CUI, 2009). To properly provide data input for them, one must design data through

the sliding time window scheme. This scheme represents the evolution of data n days before the existence of some event. Many authors argue in favor of using this time window combined with solar flare forecastings, such as Yu, Huang, H. Wang, and Cui (2009) and Huang, Yu, et al. (2010).

As such, within a sliding time window, solar data are observed at the t instant – corresponding to an arbitrary day – and some days before t, that is, $[t - \Delta t]$. This interval between t and $[t - \Delta t]$ corresponds to the window.

When designing data through a sliding time window, one must carefully choose a reasonable window length. Long windows may lead to redundancy with data. On the other hand, data may not be enough when someone designs extremely short windows.

By trying to include only enough data, we defined our window length based on the ARs' lifetime. As argued by Canfield (2001), ARs are born when magnetic flux strands become visible into the photosphere from the solar interior. Not rarely, this flux lasts for at least five days, that is, the period in which ARs grow to larger sizes before quickly stops emerging until vanishing. Hence, our data stream comprehended five days of evolutionary data features, i. e., we aimed to cover ARs during their entire life cycles:

- *xray_background_flux* $\left[t - 4d, t - 3d, t - 2d, t - 1d, t\right]$;
- *radio_flux* $\left[t - 4d, t - 3d, t - 2d, t - 1d, t\right]$;
- *sunspot_area* $\left[t - 4d, t - 3d, t - 2d, t - 1d, t\right]$;
- *sunspot_number* $\left[t - 4d, t - 3d, t - 2d, t - 1d, t\right]$;
- *daily_magnetic_class_wmfr* $\left[t - 4d, t - 3d, t - 2d, t - 1d, t\right]$;
- *daily_z_component_wmfr* $\left[t - 4d, t - 3d, t - 2d, t - 1d, t\right]$;
- *daily_p_component_wmfr* $\left[t - 4d, t - 3d, t - 2d, t - 1d, t\right]$;
- *daily_c_component_wmfr* $\left[t - 4d, t - 3d, t - 2d, t - 1d, t\right]$.

At this point, it is worth noting that some solar physicists suggest not employing k-fold cross-validation with solar time series data. Otherwise, sampled folds may distort the behavior of data in the solar cycle. However, we carried out the splitting after designing the sliding time window, which means we managed to guarantee that each observed event always succeeded its corresponding five days of evolutionary data, thus not causing distortions due to sampling.

Nevertheless, our methodology also supports time segmented data, such as reserving the oldest period for training, the intermediate for validating, and the most recent for testing (or some other criteria based on periods as pointed out on Ahmed et al. (2013)'s and Colak and Qahwaji (2009)'s researches). However, for brevity, we only included in this thesis the case study to illustrate the k-fold-based splitting.

5.1.3 Event definition and forecasting horizons

To design our forecasting horizons, we used the counts found in DSD's records for each flare class. As such, we transformed those counts into binary flags for representing the occurrence of ≥ C (Case Study I), and ≥ M (Case Study II) flare events within 24, 48, and 72 h ahead of the t instant.

After designing the target features, our datasets presented some skewed distributions for class ratios. For ≥ C-class events forecasting, we observed the class ratios as follows – in all the cases, we could note imbalanced classes:

- the next 24 h: 4,225 (58%, positive samples) versus 3,095 (42%, negative samples);
- the next 48 h: 4,783 (65%, positive samples) versus 2,531 (35%, negative samples);
- the next 72 h: 5,122 (70%, positive samples) versus 2,191 (30%, negative samples).

On the other hand, for ≥ M-class events forecasting, the class ratios had been distributed by – for Case Study II, we could also notice imbalanced classes:

- the next 24 h: 1,279 (18%, positive samples) versus 6,033 (82%, negative samples);
- the next 48 h: 1,894 (26%, positive samples) versus 4,418 (74%, negative samples);
- the next 72 h: 2,322 (32%, positive samples) versus 4,990 (68%, negative samples).

Conversely to The Met Office Space Weather Operations Centre (MOSWOC) in the United Kingdom (MURRAY et al., 2017) and NOAA/SWPC (CROWN, 2012), which both employed a hybrid human-based approach for making full-disk forecasts in the next 24 h, 24 h – 48 h, and 48 h – 72 h ahead of a given t instant, we designed overlapping time horizons, that is, flare forecasting in the next 24, 48, and 72 h. Other researchers effectively using overlapping time horizons included Colak and Qahwaji (2009), Yu, Huang, H. Wang, and Cui (2009) and Huang, H. Wang, Xu, et al. (2018).

MOSWOC/SWPC provided forecasts for specific classes of flares. In contrast, NOAA/SWPC made forecasts for events higher than or equal to some magnitude thresholds, namely ≥ C and ≥ M. In this sense, we adhered to NOAA's approach for predicting above a threshold while modeling our target features.

The reason for assembling the dataset as such is merely related to our case study. We do not restrict the methodology proposed herein to those types of input and target features. Hence, one can use the method along with other features with only a few or no adjustments.

5.2 Dataset for Case Study III

Conversely to the case studies I and II, in which we authored the dataset, we used data assembled in C. Liu et al. (2017)'s research in Case Study III. As described in Chapter 3, directly comparing scores from studies employing different datasets can be meaningless due to the several distinct underlying data characteristics, that is, reliable forecast approach comparisons comprehend those made upon the same set of data (BARNES; LEKA, et al., 2016). Bearing this in mind, we used C. Liu et al. (2017)'s dataset to design a classifier with our method in Case Study III and compared its output with the classifier designed by them[4].

In C. Liu et al. (2017)'s article, the authors used HMI vector magnetic data to forecast the magnitude of flares in terms of their specific GOES classes – B-, C-, M-, and X-class – occurring in a given AR in the next 24 h. In addition, they also designed a binary scenario for forecasting, namely by grouping B- with C-class events (negative class), and M- with X-class flares (positive class). For Case Study III, we only employed data from the binary scenario, that is, we only designed experiments for ≥ M-class AR-by-AR forecasting, thus not considering the prediction of specific classes of flares. We designed Case Study III to allow a direct comparison of results with other literature approaches.[5]

To assemble their data, C. Liu et al. (2017) surveyed flares – and also ARs in which they occurred – in the X-ray flare catalogs from the National Centers for Environment Information (NCEI)/NGDC. The period investigated comprehended 2010 May and 2016 December (the main peak of solar cycle 24).

[4]For reproducibility, the authors provided their complete dataset and source code of experiments in `https://web.njit.edu/cl45/Fpredict/`.

[5]Although we used data from other authors, we prepared their samples differently. As such, to fit our classifier, we prepared samples through our methodology pipeline instead of the method C. Liu et al. (2017) defined.

To select flaring AR samples, C. Liu et al. (2017) assured that: (i) they tracked ARs within 70° from the disk center; (ii) flaring AR samples had valid values for their 13 HMI magnetic parameters at the beginning of the flaring days; (iii) they counted only one event when a flaring AR produced multiple events of the same type on a given day; (iv) they counted multiple events for flaring ARs producing multiple flares on different days.

Overall, C. Liu et al. (2017) selected 845 AR samples, comprehending 23 X-class, 142 M-class, 552 C-class, and 128 B-class ARs. Then, to integrate with vector magnetic data to detail the complexity of magnetic fields, they surveyed the Joint Science Operations Center (JSOC)/HMI data products – namely *hmi.sharp* and *cgem.Lorentz* – for the magnetic parameters[6] proposed by Bobra and Couvidat (2015) (they processed such magnetic parameters always at 00:12 AM UTC and verified the existence of flares 24 h ahead of this time):

- Total unsigned current helicity (TOTUSJH);
- Total magnitude of Lorentz force (TOTBSQ);
- Total photospheric magnetic free energy density (TOTPOT);
- Total unsigned vertical current (TOTUSJZ);
- Absolute value of the net current helicity (ABSNJZH);
- Sum of the modulus of the net current per polarity (SAVNCPP);
- Total unsigned flux (USFLUX);
- Area of strong field pixels in the active region (AREA_ACR);
- Sum of z-component of Lorentz force (TOTFZ);
- Mean photospheric magnetic free energy (MEANPOT);
- Sum of flux near the polarity inversion line (R_VALUE);
- Sum of the z-component of normalized Lorentz force (EPSZ);
- Fraction of area with shear >45° (SHRGT45).

[6]For the complete mathematical formulae of the parameters, refer to Bobra and Couvidat (2015)'s and C. Liu et al. (2017)'s articles.

5.3 Concluding remarks

This chapter presented the datasets employed in the case studies discussed in Chapter 6. Not only we assembled our own datasets but also used data from third parties to allow direct and more reliable comparisons of performance. As such, we shall employ those datasets to design classifiers under the concepts of the methodology proposed in Chapter 4.

Chapter 6

Results and discussion

This chapter shall discuss the results of carrying out the methodology as previously defined in Chapter 4 and agreeing with data from Chapter 5. Hence, it presents the performance of models designed to forecast ≥ C- (Case Study I) and ≥ M-class events (case studies II and III) up to three days ahead.

To avoid mixing results from different case studies, we shall group them into three distinct main sections, namely 6.1, 6.2, and 6.3 (one for each scenario). Within each section, we will roughly organize the content into two main parts (subsections), one for discussing results from the methodology, and the remainder for comparing such results to the literature.

6.1 Case Study I: ≥ C-class flare forecasting

This section contains the analysis for Case Study I. Specifically in Case Study I, we shall dive a bit deeper into the results analysis and present the detailed discussion of the methodology's inner results for a particular forecasting horizon, that is, the model designed to forecast in the next 24 h[1]. In this sense, Section 6.1.1 shall discuss how the performance has changed between the methodology's inner processes for the shortest horizon.

[1]For presenting the detailed methodology's inner results of longer forecasting horizons within Case Study I and other cases, refer to Appendix B.

6.1.1 Forecasts within the next 24 h

To verify the significance of the herein reported increases/decreases, we will use a paired two-tailed Wilcoxon Signed-Rank Test (WILCOXON, 1945)[2]. For instance, at model selection, when directly comparing the highest TSS to the second highest, the observed difference must be statistically significant, i. e., it shall pass such test.

The Wilcoxon Signed-Rank Test compares whether the difference between two paired dependent observations equals zero. This comparison comprehends its $H0$ hypothesis and checks against some predefined confidence intervals (i. e., the α values), outputting a *p-value*. Whether the test affirms the observed effect significantly differs from zero – that is, its $p < \alpha$ –, the methodology can then proceed within its pipeline.

Noteworthily, the methodology checks the observed differences against the Wilcoxon Signed-Rank Test using two distinct confidence intervals: $\alpha = 0.05$ and $\alpha = 0.1$. Passing either of those allows the pipeline to proceed. For instance, some score difference outputting $p < 0.1$ means a small chance of Type-1 statistical error (i.e., rejecting a correct $H0$), notably 10%. The same reasoning is valid for $p < 0.05$, but the chance of a Type-1 error decreases to 5%.

Model selection results

Table 6.1 shows the results for ≥ C-class flare forecasting in the next 24 h within Case Study I for each training set. The intermediate lines – denoted by the "*avg*" notation – refer to the averaged score values among all training sets. On the other hand, the remainder lines refer to the learning algorithms cross-validated over all training sets (refer to the [a] note in Table 6.1).

Overall, we can see fair scores with all models. For instance, their TPR and TNR results ranged on [0.81 , 0.88] and [0.75 , 0.80], respectively, which consequently increased their TSSs ([0.59 , 0.64]). Not only TPRs, TNRs, and TSSs had positive results, but also FARs ([0.15 , 0.18]) and AUCs ([0.85 , 0.90]). In addition, PPVs ([0.82 , 0.85]) and NPVs ([0.76 , 0.82]) also reached high levels.

However, to forecast in the next 24 h, the methodology maintained the RandomForest model and discarded the others. It has achieved the highest TSS concerning the second highest, GradientTreeBoosting (refer to the [b] note in Table 6.1): $p < 0.05$ – and indirectly $p < 0.1$.

[2]The literature suggests the use of the Wilcoxon Signed-Rank Test instead of the Student's T test as we have a few values to test due to the 5-fold split schema from case studies (i. e., for reliability purposes).

Table 6.1: Case Study I, ≥ C flares, the next 24 h: model selection results.

Model/Train.Set[a]	ACC^{acc}	TPR^{tpr}	TNR^{tnr}	PPV^{ppv}	NPV^{npv}	FAR^{far}	TSS^{tss}	HSS^{hss}	AUC^{auc}
AdaBoost/1	0.802	0.836	0.755	0.824	0.771	0.176	0.591	0.592	0.882
AdaBoost/2	0.803	0.838	0.755	0.825	0.773	0.175	0.593	0.595	0.882
AdaBoost/3	0.804	0.842	0.752	0.823	0.777	0.177	0.594	0.596	0.882
AdaBoost/4	0.808	0.846	0.755	0.826	0.783	0.174	0.601	0.604	0.884
AdaBoost/5	0.800	0.838	0.749	0.821	0.772	0.179	0.587	0.589	0.877
avg(AdaBoost)	0.80	0.84	0.75	0.82	0.78	0.18	0.59	0.60	0.88
RandomForest/1	0.825	0.879	0.750	0.829	0.820	0.171	0.630	0.637	0.907
RandomForest/2	0.828	0.883	0.754	0.831	0.825	0.169	0.636	0.644	0.911
RandomForest/3	0.828	0.885	0.748	0.828	0.827	0.172	0.634	0.642	0.907
RandomForest/4	0.829	0.881	0.758	0.833	0.823	0.167	0.638	0.645	0.908
RandomForest/5	0.834	0.883	0.766	0.838	0.828	0.162	0.649	0.656	0.909
avg(RandomForest)	0.83	0.88	0.76	0.83	0.82	0.17	0.64[b]	0.64	0.91
GradientTreeBoosting/1	0.804	0.803	0.804	0.851	0.752	0.149	0.607	0.601	0.855
GradientTreeBoosting/2	0.801	0.807	0.793	0.844	0.753	0.156	0.600	0.595	0.854
GradientTreeBoosting/3	0.804	0.808	0.799	0.849	0.756	0.151	0.607	0.602	0.853
GradientTreeBoosting/4	0.808	0.826	0.784	0.842	0.771	0.158	0.610	0.608	0.857
GradientTreeBoosting/5	0.802	0.804	0.799	0.848	0.753	0.152	0.603	0.597	0.853
avg(GradientTreeBoosting)[c]	0.80	0.81	0.80	0.85	0.76	0.15	0.61[b]	0.60	0.85

[a] The *Model/Train.Set* notation means a particular learning algorithm cross-validated over some specific training set.

[b] $TSS_{RandomForest} > TSS_{GradientTreeBoosting}$ ($p < 0.05$). As the RandomForest outputs the highest TSS, the methodology chose it to proceed in the pipeline. Although such a difference may suggest a small improvement (+0.03), we managed to statistically confirm it with 95% of confidence – i. e., through the Wilcoxon-signed rank test described earlier.

[c] Although the GradientTreeBoosting may suggest a reasonable classifier (i. e., reduced class skew and FAR), the criteria used in our case studies's model selection processes only considered the TSS.

[acc] ACC lies around [0 , 1]: the higher the score, the better the classifier (HAN; KAMBER, 2006).

[tpr] TPR lies around [0 , 1], where higher values represent better classifiers (HAN; KAMBER, 2006).

[tnr] We measure and analyze TNR results the way as that of TPR (HAN; KAMBER, 2006).

[ppv] PPV ranges on [0 , 1]: the higher the values, the better the classifier (ZAKI; MEIRA JR., 2013).

[npv] NPV ranges on [0 , 1]: the higher the values, the better the classifier (ZAKI; MEIRA JR., 2013).

[far] FAR scores on [0 , 1]: the lower the score, the better the classifier (JOLLIFFE; STEPHENSON, 2003).

[tss] The TSS ranks a model over a scale lying on [−1 , 1]. Results close to −1 mean all predictions incorrect (positive and negative), and those close to 1 mean all predictions correct – zeroed results refer to no-skilled classifiers (JOLLIFFE; STEPHENSON, 2003).

[hss] HSS ranks classifiers over [−1 , 1], where results close to -1 mean all predictions incorrect (positive and negative) and close to 1 mean all predictions correct – zeroed results refer to classifiers' forecast skills equal to random chance (JOLLIFFE; STEPHENSON, 2003).

[auc] The AUC is always positive and should be greater than 0.5 ideally. Best classifiers score AUCs next to 1 (WITTEN; FRANK; HALL, 2011).

Since the *p-value* mentioned earlier was less than both of our defined confidence intervals, we could successfully reject our Wilcoxon Signed-Rank Test $H0$ hypothesis (the difference between two paired dependent observations equals zero). It means that the chance of a Type-1 statistical error (i. e., rejecting a correct $H0$) is small (5% and 10% for $\alpha = 0.05$ and 0.1, respectively). Hence, we could affirm that the difference between the averaged TSSs of RandomForest and GradientTreeBoosting was statistically significant.

Regardless of the positive results previously commented, there was still room for further improvements. For instance, taking the chosen RandomForest model as a reference, we could note a scenario of slightly imbalanced class ratios. Bearing in mind its ACC = 0.83, whereas the TPR reached 0.88, the TNR only scored 0.76.

In fact, the RandomForest model was not able to correctly forecast the negative class as well as it performed with the positive one. Not only coping with the imbalance issue, but the methodology shall also persecute increases in other scores next.

Feature selection results

Table 6.2 shows the detailed results[3] from the Case Study I's feature selection for each training set (for forecasts within the next 24 h). These results refer to the representative set of features selected through univariate analysis re-evaluated over all training sets. Such representative set comprehended the features as listed below:

- *radio_flux_10.7cm_t2* (F-score = 2599.78);
- *radio_flux_10.7cm_t1* (F-score = 2770.28);
- *radio_flux_10.7cm_t* (F-score = 2974.9);
- *sesc_sunspot_number_t* (F-score = 3130.42);
- *c_component_wmfr_t* (F-score = 3308.50).

Regardless of those features being linked to flare occurrence (as suggested by the univariate analysis), their selection also agreed with the specialized literature. For instance, the WMFR for the McIntosh (1990)'s *c* component agreed with the findings of Eren et al. (2017) and Kilcik et al. (2018), who suggested that it could be used to represent magnetic flux emergence inside

[3]From now on, we shall use the $\pm n$ notation of the averaged results in Table 6.2 to denote how scores changed by n concerning their previous – confirmed – results.

ARs leading to higher flare amounts. In turn, the radio flux derived features agreed with NOAA (2020)'s observations of the F10.7 index – i. e., by analyzing the NOAA (2020)'s graph, we could observe a radio flux movement roughly following the solar cycle, whose majority of flares occurred in the peaks.

Table 6.2: Case Study I, ≥ C flares, the next 24 h: feature selection results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
RandomForest/1	0.844	0.881	0.792	0.854	0.831	0.147	0.674	0.678	0.902
RandomForest/2	0.848	0.882	0.802	0.860	0.833	0.140	0.684	0.687	0.900
RandomForest/3	0.845	0.876	0.803	0.860	0.826	0.140	0.679	0.682	0.904
RandomForest/4	0.843	0.881	0.789	0.852	0.830	0.148	0.671	0.675	0.905
RandomForest/5	0.846	0.882	0.798	0.857	0.832	0.143	0.680	0.683	0.904
avg(RandomForest)	**0.85**[a,acc]	0.88[b]	**0.80**[tnr]	**0.86**[ppv]	**0.83**[npv]	**0.14**[far]	**0.68**[tss]	**0.68**[hss]	0.90[c,auc]

[a] Scores marked in bold represent positive results (increases/decreases) concerning the previous process, confirmed with $p < 0.05$ or $p < 0.1$.
[b] Scores not marked with any different style mean they remained unaffected concerning the previous process, confirmed with $p > 0.05$ and $p > 0.1$. Such scores can also be interpreted as positive results, since they did not increase nor even decrease.
[c] Underlined scores represent negative results (increases/decreases) concerning the previous process, confirmed with $p < 0.05$ or $p < 0.1$.
[acc] +0.02 ($p < 0.05$).
[tnr] +0.04 ($p < 0.05$).
[ppv] +0.03 ($p < 0.05$).
[npv] +0.01 ($p < 0.1$).
[far] -0.03 ($p < 0.05$).
[tss] +0.04 ($p < 0.05$).
[hss] +0.04 ($p < 0.05$).
[auc] -0.01 ($p < 0.05$).

Noteworthily, we are aware that some features mentioned earlier may suggest similarity to each other – i. e., the correlated nature of the radio flux sliding time window –, which can somehow directly lead to redundancy in the input of RandomForest (i. e., features co-linearity). However, we are using a method to test the complementary nature of features instead of their correlation – that is, we aimed at identifying independent features that led to good predictors when combined, regardless of their correlation.

Compared to the model selection results shown in Table 6.1, we could note a single negative result, namely with AUC, which decreased by 0.01 ($p < 0.05$). In turn, other scores positively

increased (ACC, TNR, PPV, NPV, TSS, and HSS) or decreased (FAR)[4]. Despite the AUC decrease, the methodology shall manage to increase it again with the hyperparameter optimization.

Concerning positive results, ACC, TNR, PPV, NPV, FAR, TSS, and HSS changed by 0.02 ($p < 0.05$), 0.04 ($p < 0.05$), 0.03 ($p < 0.05$), 0.01 ($p < 0.05$), -0.03 ($p < 0.05$), 0.04 ($p < 0.05$), and 0.04 ($p < 0.05$), respectively. In turn, the TPR remained the same as of model selection (confirmed with a *p-value* higher than our confidence intervals), which can also be interpreted as a positive result, since the score did not increase nor even decrease.

Hyperparameter optimization results

Table 6.3 shows the detailed results from the hyperparameter optimization of Case Study I for each training set (for forecasts within the next 24 h). These results refer to the parameter set up of RandomForest that most increased AUC in some particular training set re-evaluated over all training sets. Such set comprehended the following parameters:

- number of base learners: 300;
- maximum depth for base learners: 15;
- number of features used to fit each base learner: 4;
- number of samples needed to split an internal node: 100;
- number of samples needed for a leaf node: 40;
- threshold to stop the growth of base learners: 0.001.

Compared to the feature selection results shown in Table 6.2, we could notice two negative results, namely with TPR and NPV, which decreased by 0.01 ($p < 0.05$ – compared to model selection) and 0.01 ($p < 0.05$), respectively. In turn, other scores positively increased (TNR, PPV, HSS, and AUC) or remained at the same level as of feature selection (ACC, FAR, TSS).

Although small, we expected the decreases in TPR and NPV. Those changes were directly related to the precision-recall trade-off (ZAKI; MEIRA JR., 2013). This trade-off states that increasing recall scores – positive or negative – naturally decreases the corresponding precision scores and vice versa. In fact, it suggests that we may have found a parameter set up that

[4]Refer to the [a,b,c] notes in Table 6.2 for further information on styling and highlighting table results.

Table 6.3: Case Study I, ≥ C flares, the next 24 h: hyperparameter optimization results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
RandomForest/1	0.848	0.870	0.818	0.868	0.822	0.132	0.688	0.688	0.930
RandomForest/2	0.850	0.872	0.819	0.869	0.824	0.131	0.691	0.691	0.929
RandomForest/3	0.846	0.871	0.812	0.864	0.822	0.136	0.683	0.684	0.928
RandomForest/4	0.842	0.872	0.802	0.858	0.821	0.142	0.674	0.676	0.929
RandomForest/5	0.848	0.873	0.814	0.865	0.824	0.135	0.686	0.687	0.929
avg(RandomForest)	0.85	0.87[tpr]	**0.81**[tnr]	**0.87**[ppv]	0.82[npv]	0.14	0.68	**0.69**[hss]	**0.93**[auc]

[tpr] -0.01 ($p < 0.05$) – compared to model selection.
[tnr] +0.01 ($p < 0.05$).
[ppv] +0.01 ($p < 0.05$).
[npv] -0.01 ($p < 0.05$).
[hss] +0.01 ($p < 0.05$).
[auc] +0.03 ($p < 0.05$).

caused minor changes to the prediction threshold (cut-off point) of the RandomForest – directly related to controlling TPR vs. PPV and TNR vs. NPV.

Concerning positive results, TNR, PPV, and HSS increased by 0.01 ($p < 0.05$), as well as AUC, by 0.03 ($p < 0.05$). In turn, ACC, FAR, and TSS remained the same as feature selection, which we can interpret as positive results since they did not increase or decrease.

Data resampling results

Both data resampling and the cost function analysis are independent processes carried out concurrently after the hyperparameter optimization. Whether both are available[5], the methodology chooses only one to proceed in the pipeline – the one outputting the lowest difference |TPR – TNR|. To certify which process has been chosen, refer to their corresponding table of results (tables 6.4 and 6.5).

Table 6.4 shows the detailed results after the data resampling of Case Study I for forecasts in the next 24 h. These results refer to the RandomForest model re-evaluated over the resampled training sets with fine-tuned parameters, and features previously selected.

For each training set, the methodology assessed the performance of three distinct resampling techniques used before the RandomForest's re-evaluations, namely SMOTE, SMOTE-ENN, and SMOTE-Tomek. Noteworthily, the methodology only opts for some data resampling method

[5]Not all learning algorithms support cost-sensitive learning.

provided that it scored the lowest difference |TPR – TNR| (i. e., the [a,b,c] notes in Table 6.4) compared to the cost function analysis.

Table 6.4: Case Study I, ≥ C flares, the next 24 h: data resampling results.

Method/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
SMOTE/1	0.820	0.751	0.915	0.924	0.729	0.076	0.666	0.643	0.927
SMOTE/2	0.824	0.761	0.910	0.921	0.736	0.079	0.671	0.650	0.927
SMOTE/3	0.819	0.754	0.907	0.918	0.730	0.083	0.661	0.640	0.925
SMOTE/4	0.830	0.772	0.909	0.921	0.744	0.079	0.681	0.661	0.928
SMOTE/5	0.824	0.753	0.921	0.929	0.731	0.071	0.674	0.651	0.928
avg(SMOTE)	0.82	0.76[a]	0.91[a]	0.92	0.73	0.08	0.67	0.65	0.93
SMOTE-ENN/1	0.835	0.801	0.881	0.902	0.764	0.098	0.681	0.668	0.922
SMOTE-ENN/2	0.833	0.807	0.869	0.895	0.768	0.106	0.676	0.664	0.918
SMOTE-ENN/3	0.840	0.822	0.864	0.893	0.781	0.107	0.686	0.676	0.920
SMOTE-ENN/4	0.837	0.800	0.888	0.908	0.765	0.092	0.689	0.674	0.921
SMOTE-ENN/5	0.843	0.829	0.863	0.893	0.787	0.108	0.692	0.683	0.922
avg(SMOTE-ENN)	0.84	0.81[b]	0.87[b]	0.90	0.77	0.10	0.68	0.67	0.92
SMOTE-Tomek/1	0.841	0.833	0.852	0.885	0.789	0.115	0.685	0.677	0.928
SMOTE-Tomek/2	0.844	0.836	0.855	0.888	0.793	0.112	0.691	0.684	0.929
SMOTE-Tomek/3	0.843	0.837	0.851	0.885	0.793	0.115	0.688	0.681	0.927
SMOTE-Tomek/4	0.846	0.837	0.859	0.891	0.794	0.109	0.696	0.689	0.929
SMOTE-Tomek/5	0.850	0.843	0.859	0.891	0.801	0.109	0.702	0.695	0.929
avg(SMOTE-Tomek)	0.84	0.84[c]	0.86[c]	0.89	0.79	0.11	0.69	0.69	0.93

[a] |TPR – TNR| = |0.76 – 0.91| = 0.15.
[b] |TPR – TNR| = |0.81 – 0.87| = 0.06.
[c] |TPR – TNR| = |0.84 – 0.86| = 0.02.

Overall, except for the SMOTE, which inverted the trend TPR/TNR and was not able to make them reach close levels, and SMOTE-ENN, which only inverted the TPR and TNR, SMOTE-Tomek succeeded when making the positive recall come close to the negative one: |TPR – TNR| = 0.02. Bearing this difference in mind, let us assess the performance of the cost function analysis next.

Cost function analysis results

Table 6.5 shows the results from the Case Study I's cost function analysis for forecasts in the next 24 h. These results refer to the C_p/C_n ratio that most led the TPR come close to the TNR in the majority of training sets re-evaluated over all training sets.

Table 6.5: Case Study I, ≥ C flares, the next 24 h: cost function analysis results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
RandomForest/1	0.845	0.844	0.847	0.883	0.799	0.117	0.691	0.685	0.929
RandomForest/2	0.842	0.839	0.846	0.882	0.794	0.118	0.686	0.680	0.929
RandomForest/3	0.847	0.840	0.855	0.888	0.797	0.112	0.695	0.689	0.930
RandomForest/4	0.844	0.843	0.846	0.882	0.798	0.118	0.689	0.683	0.928
RandomForest/5	0.851	0.848	0.856	0.890	0.804	0.111	0.703	0.697	0.930
avg(RandomForest)	0.85	0.84[a,tpr]	**0.85**[a,tnr]	**0.89**[ppv]	0.80[npv]	**0.11**[far]	0.69[tss]	0.69	0.93

[a] |TPR − TNR| = |0.84 − 0.85| = 0.01. As the cost function scored the lowest difference, the pipeline shall proceed with it instead of data resampling.
[tpr] -0.03 ($p < 0.05$).
[tnr] +0.04 ($p < 0.05$).
[ppv] +0.02 ($p < 0.05$).
[npv] -0.02 ($p < 0.05$).
[far] -0.03 ($p < 0.05$) − compared to feature selection.
[tss] +0.01 ($p > 0.05$ and 0.1) − compared to feature selection.

Compared to the SMOTE-Tomek from Table 6.4, the cost function analysis scored the lowest difference |TPR − TNR|, noticeably, 0.01. Henceforth, the methodology shall use cost-sensitive learning instead of data resampling to cope with the imbalanced class ratios. Accordingly, it will employ $C_p/C_n = 0.78$ since this custom ratio made the TPR come close to the TNR in two training sets.

Overall, we had two negative results during the cost function analysis, namely with the TPR and NPV, which decreased by 0.03 ($p < 0.05$) and 0.02 ($p < 0.05$), respectively. In turn, other scores positively changed (TNR, PPV, and FAR) or remained the same as of hyperparameter optimization or feature selection (HSS, AUC, and ACC).

Regarding positive results, the TNR, PPV, and FAR changed by 0.04 ($p < 0.05$), 0.02 ($p < 0.05$), and -0.03 ($p < 0.05$ − compared to feature selection), respectively. Since we had increases with the TNR and PPV scores, we suggest the decreasing TPR and NPV as directly related to the precision-recall trade-off mentioned earlier. Besides, by maintaining the HSS, AUC, and ACC at the high levels of the hyperparameter optimization (HSS and AUC) or feature selection (ACC), the methodology managed to keep fair results here.

Last but not least, concerning the TSS, although this process increased it by about 0.01 − compared to feature selection −, we had to reject the significance of this change ($p > 0.05$ and $p > 0.1$). Nevertheless, the TSS = 0.68 yet remained at the high level of feature selection.

Cut-off point adjustment results

Table 6.6 shows the results from the cut-off point adjustment of Case Study I for forecasts in the next 24 h. These results refer to the prediction threshold t that most led the TPR come close to the PPV in the majority of training sets re-evaluated over all training sets – in this case, $t = 0.45$ (it made the TPR come close to the PPV in all training sets).

Table 6.6: Case Study I, ≥ C flares, the next 24 h: cut-off point adjustment results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
RandomForest/1	0.847	0.869	0.818	0.867	0.820	0.133	0.687	0.687	0.929
RandomForest/2	0.848	0.866	0.824	0.871	0.818	0.129	0.690	0.689	0.930
RandomForest/3	0.846	0.868	0.816	0.866	0.819	0.134	0.684	0.684	0.928
RandomForest/4	0.843	0.867	0.810	0.862	0.817	0.138	0.677	0.678	0.929
RandomForest/5	0.850	0.867	0.826	0.872	0.820	0.128	0.693	0.692	0.930
avg(RandomForest)	0.85	**0.87**$^{\text{tpr}}$	0.82$^{\text{tnr}}$	0.87$^{\text{ppv}}$	**0.82**$^{\text{npv}}$	0.13$^{\text{far}}$	0.69	0.69	0.93

$^{\text{tpr}}$ +0.03 ($p < 0.05$).
$^{\text{tnr}}$ -0.03 ($p < 0.05$).
$^{\text{ppv}}$ -0.02 ($p < 0.05$).
$^{\text{npv}}$ +0.02 ($p < 0.05$).
$^{\text{far}}$ +0.02 ($p < 0.05$).

Compared to the cost function results shown in Table 6.5, we could notice three negative results, namely with the TNR, PPV, and FAR. The TNR and PPV decreased by 0.03 ($p < 0.05$) and 0.02 ($p < 0.05$), respectively. In turn, the FAR negatively increased by 0.02 ($p < 0.05$). On the other hand, other scores positively increased (TPR and NPV) or remained at the same level as previously observed (ACC, TSS, HSS, and AUC).

Since we aimed to adjust the RandomForest's cut-off point to find a balanced t, i. e., a prediction threshold that would boost the TPR to higher levels while controlling false alarms, we expected the decreases of TNR and NPV, as well as the increases of TPR and NPV. Once more, we argue that those changes refer to the precision-recall trade-off. Nevertheless, the TPR and NPV positively increased by 0.03 ($p < 0.05$) and 0.02 ($p < 0.05$), respectively.

Evaluation of validation sets

Provided with the set of selected features, optimized set of parameters, custom C_p/C_n ratio, and adjusted prediction threshold, we could then fit five RandomForest models over training

sets and forecast their corresponding validation sets. Besides, we also fit five baseline models[6] over training sets and also forecast their corresponding validation sets. Table 6.7 shows the results of the baseline (BaselineRandomForest entries in the table) and methodology-supported predictions (OptimizedRandomForest entries in the table).

Table 6.7: Case Study I, ≥ C flares, the next 24 h: evaluation of validation sets.

Pred.Type/Val.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
BaselineRandomForest/1	0.829	0.887	0.751	0.830	0.829	0.170	0.638	0.646	0.906
BaselineRandomForest/2	0.827	0.865	0.774	0.840	0.807	0.160	0.639	0.643	0.910
BaselineRandomForest/3	0.847	0.914	0.755	0.836	0.865	0.164	0.669	0.680	0.921
BaselineRandomForest/4	0.819	0.867	0.753	0.828	0.805	0.172	0.619	0.625	0.902
BaselineRandomForest/5	0.830	0.855	0.796	0.851	0.801	0.149	0.651	0.652	0.912
avg(BaselineRandomForest)	0.83	0.88	0.77	0.84	0.82	0.16	0.64	0.65	0.91
OptimizedRandomForest/1	0.859	0.878	0.834	0.879	0.833	0.121	0.712	0.712	0.931
OptimizedRandomForest/2	0.844	0.848	0.839	0.878	0.801	0.122	0.686	0.682	0.929
OptimizedRandomForest/3	0.852	0.884	0.809	0.864	0.836	0.137	0.692	0.696	0.936
OptimizedRandomForest/4	0.830	0.854	0.798	0.853	0.800	0.147	0.652	0.652	0.926
OptimizedRandomForest/5	0.843	0.844	0.841	0.879	0.798	0.121	0.685	0.680	0.928
avg(OptimizedRandomForest)	**0.85**$^{\text{acc}}$	<u>0.86</u>$^{\text{tpr}}$	**0.82**$^{\text{tnr}}$	**0.87**$^{\text{ppv}}$	0.81$^{\text{npv}}$	**0.13**$^{\text{far}}$	**0.69**$^{\text{tss}}$	**0.68**$^{\text{hss}}$	**0.93**$^{\text{auc}}$

$^{\text{acc}}$ +0.02 ($p < 0.05$).
$^{\text{tpr}}$ -0.02 ($p < 0.05$).
$^{\text{tnr}}$ +0.05 ($p < 0.05$).
$^{\text{ppv}}$ +0.03 ($p < 0.05$).
$^{\text{npv}}$ -0.01 ($p > 0.05$ and 0.1).
$^{\text{far}}$ -0.03 ($p < 0.05$).
$^{\text{tss}}$ +0.05 ($p < 0.05$).
$^{\text{auc}}$ +0.02 ($p < 0.05$).

Overall, except for the TPR, which decreased by 0.02 ($p < 0.05$) concerning the baseline models, other scores have been positively affected (ACC, TNR, PPV, FAR, TSS, HSS, and AUC). Regarding NPV, although the methodology decreased it by 0.01, we could not confirm the significance of this change ($p > 0.05$ and 0.1). Concerning ACC, TNR, PPV, FAR, TSS, HSS, and AUC, the methodology made them improve by 0.02 ($p < 0.05$), 0.05 ($p < 0.05$), 0.03 ($p < 0.05$), -0.03 ($p < 0.05$), 0.05 ($p < 0.05$), 0.03 ($p < 0.05$), and 0.02 ($p < 0.05$), respectively. This scenario of positive effects over almost all scores – and specially the TSS increase – made the pipeline proceed with the evaluation over test sets.

[6]Baseline models refer to the RandomForest algorithm provided with a randomly initialized set of values for their parameters used in feature selection and the full set of features. Also, we did not use any custom C_p/C_n ratio, data resampling method, or adjusted prediction threshold.

Evaluation of test sets

Provided that the methodology confirmed positive forecasts during the evaluation of validation sets, it could then assess the generalization error of RandomForest over unseen data. Table 6.8 shows the results of baseline and method-supported predictions over test sets.

Table 6.8: Case Study I, ≥ C flares, the next 24 h: evaluation of test sets.

Pred.Type/Test Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
BaselineRandomForest/1	0.809	0.863	0.736	0.816	0.797	0.184	0.598	0.604	0.899
BaselineRandomForest/2	0.800	0.834	0.753	0.822	0.768	0.178	0.587	0.589	0.883
BaselineRandomForest/3	0.809	0.880	0.712	0.807	0.813	0.193	0.593	0.603	0.912
BaselineRandomForest/4	0.866	0.870	0.861	0.896	0.828	0.105	0.731	0.727	0.920
BaselineRandomForest/5	0.830	0.862	0.785	0.845	0.807	0.155	0.647	0.650	0.901
avg(BaselineRandomForest)	0.82	0.86	0.77	0.84	0.80	0.16	0.63	0.63	0.90
OptimizedRandomForest/1	0.831	0.877	0.768	0.837	0.821	0.163	0.645	0.650	0.928
OptimizedRandomForest/2	0.825	0.834	0.812	0.859	0.781	0.142	0.646	0.642	0.916
OptimizedRandomForest/3	0.851	0.857	0.843	0.882	0.811	0.118	0.700	0.696	0.935
OptimizedRandomForest/4	0.874	0.855	0.901	0.922	0.819	0.078	0.756	0.746	0.946
OptimizedRandomForest/5	0.872	0.882	0.859	0.895	0.842	0.105	0.741	0.739	0.933
avg(OptimizedRandomForest)	**0.85**^{acc}	0.86	**0.84**^{tnr}	**0.88**^{ppv}	0.81^{npv}	**0.12**^{far}	**0.70**^{tss}	**0.69**^{hss}	**0.93**^{auc}

^{acc} +0.03 ($p < 0.05$).
^{tnr} +0.07 ($p < 0.05$).
^{ppv} +0.04 ($p < 0.05$).
^{npv} +0.01 ($p > 0.05$ and 0.1).
^{far} -0.04 ($p < 0.05$).
^{tss} +0.07 ($p < 0.05$).
^{auc} +0.03 ($p < 0.05$).

As we could observe a majority of positive results during the evaluation of validation sets, so did we during the evaluation of test sets. Despite the unchanged TPR and NPV, which increased by 0.01 but we could not confirm the significance, other scores have been positively affected. In this sense, ACC, TNR, PPV, FAR, TSS, HSS, and AUC have been changed by 0.03 ($p < 0.05$), 0.07 ($p < 0.05$), 0.04 ($p < 0.05$), -0.04 ($p < 0.05$), 0.07 ($p < 0.05$), 0.06 ($p < 0.05$), and 0.03 ($p < 0.05$), respectively.

6.1.2 Forecasts in longer horizons

The results of inner processes for longer forecasting horizons within Case Study I followed the trend of changes observed during the next 24 h. Overall, we could observe that some longer horizons have achieved some scores slightly better than their predecessors (i. e., the

optimized test results for TPR and FAR concerning forecasts in the next 48 and 72 h). As pointed out by Colak and Qahwaji (2009), this happens when working with overlapping forecasting horizons since, in longer horizons, the amount of positive events tends to positively increase, consequently minimizing the associated imbalance issue with data.

Within this context, for brevity and in an attempt to avoid discussing redundant changes in results, we only included in this chapter the detailed discussion for forecasts in the next 24 h for Case Study I. However, we provided detailed tables as mentioned earlier in this chapter for the remainder forecasts of Case Study I and other case studies in Appendix B, including the statistical analysis for significance. Henceforth, we shall comment on the summary of results presented in Table 6.9[7] for the remainder forecasting horizons of Case Study I.

Table 6.9: Case Study I, ≥ C flares, the next 48 and 72 h: summary of results.

Process	*Forecasting Time*	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*AUC*	*TSS*	*HSS*
Model selection	next 48 h	0.85	0.94	0.69	0.85	0.85	0.15	0.90	0.62	0.65
	next 72 h	0.86	0.92	0.72	0.88	0.79	0.12	0.89	0.63	0.65
Feature selection	next 48 h	0.87	0.92	0.76	0.88	0.84	0.12	0.89	0.68	0.69
	next 72 h	0.88	0.93	0.77	0.90	0.83	0.10	0.93	0.70	0.72
Hyperparameter optimization	next 48 h	0.87	0.92	0.77	0.88	0.83	0.12	0.94	0.69	0.70
	next 72 h	0.89	0.94	0.76	0.90	0.84	0.10	0.94	0.70	0.72
Cost function analysis	next 48 h	0.86	0.85	0.86	0.92	0.76	0.08	0.94	0.72	0.69
	next 72 h	0.87	0.87	0.87	0.94	0.74	0.06	0.94	0.74	0.70
Cut-off point adjustment	next 48 h	0.87	0.90	0.81	0.90	0.81	0.10	0.94	0.70	0.70
	next 72 h	0.88	0.90	0.83	0.93	0.78	0.07	0.94	0.73	0.71
Evaluation of validation sets (baseline)	next 48 h	0.85	0.94	0.69	0.85	0.86	0.15	0.91	0.63	0.66
	next 72 h	0.85	0.93	0.68	0.87	0.81	0.13	0.89	0.61	0.64
Evaluation of validation sets (optimized)	next 48 h	0.87	0.90	0.81	0.90	0.81	0.10	0.94	0.70	0.70
	next 72 h	0.88	0.90	0.83	0.93	0.78	0.07	0.94	0.73	0.72
Evaluation of test sets (baseline)	next 48 h	0.84	0.93	0.66	0.84	0.84	0.16	0.90	0.59	0.62
	next 72 h	0.85	0.92	0.68	0.87	0.79	0.13	0.90	0.60	0.62
Evaluation of test sets (optimized)	next 48 h	0.86	0.89	0.80	0.90	0.80	0.10	0.94	0.70	0.69
	next 72 h	0.86	0.89	0.80	0.91	0.75	0.09	0.94	0.69	0.68

For forecasting the next 48 and 72 h, our method has designed models relying on the RandomForest and AdaBoost algorithms, respectively. Overall, from the model selection to cut-off point adjustments of such models, we could observe TSS increases in both horizons, notably +0.08 and +0.10 for the next 48 and 72 h, respectively. Those increases in the TSS scores agreed with the improvements for HSS results: +0.05 and +0.06 respectively for the next 48 and 72 h.

[7]For the statistical tests of significance, refer to Appendix B.

Besides, we could observe the HSSs scoring higher results than TSSs in the earlier processes (i. e., model selection). Those behaviors have been inverted by the time the class skew was corrected (i. e., cut-off point adjustment).

Bloomfield et al. (2012) argued that the HSS has a biased nature when data are imbalanced – as our class ratios in earlier processes –, thus tending to output higher results than TSS. Results observed in the TPRs of model selection (0.94 and 0.92 respectively for the next 48 and 72 h) and TNRs (0.69 and 0.71 respectively for the next 48 and 72 h) confirmed that models somehow coped with skewed class ratios.

Not only the TSSs and HSSs have incurred in benefits from the model selection to cut-off point adjustments, but also AUCs and FARs. Whereas the former scores have been improved by +0.04 (next 48 h) and +0.05 (next 72 h), the latter positively improved by -0.05 in both horizons.

It is worth emphasizing that models scoring AUCs ≥ 0.9 possess increased probabilities for forecasting random samples as positive instead of negative (ZAKI; MEIRA JR., 2013). The low FAR ratios have confirmed our positive forecasts' potential usefulness suggested by the high AUCs in both horizons.

Concerning our imbalanced class ratios, by the cut-off point adjustments, the differences |TPR – TNR| have been positively decreased by -0.16 (next 48 h) and -0.14 (next 72 h) – in comparison to model selection. Following, the TPRs have been adjusted by -0.04 (next 48 h) and -0.02 (next 72 h), as well as the TNRs by +0.12 in both horizons.

Despite those TPR decreases, the TSSs have been positively affected, as mentioned earlier. In response to the changes observed on the TPRs and TNRs, the PPVs and NPVs have been affected to some extent by the precision-recall trade-off (ZAKI; MEIRA JR., 2013): the positive precision scores have been affected by +0.05 in both horizons, as well as the negative ones by -0.04 (next 48 h) and -0.01 (next 72 h).

As the final decision-making performed in Case Study I, our methodology assessed the improvements (if any) in both models' forecast performance – using their cut-off point adjustment designs – while forecasting the validation sets. Noticeably, it confirmed some positive effects, such as the increasing TSSs (+0.07 and +0.12 respectively for the next 48 and 72 h) and HSSs (+0.04 and +0.08 respectively for the next 48 and 72 h)[8].

[8]For the complete analysis of improved/harmed validation scores of Case Study I, refer to Appendix B.

Since the pipeline identified increases in the validation TSSs, it then proceeded with the generalization error assessment of both models over the test sets. Overall, we could notice these remarkable effects in the test results of all forecasting horizons of Case Study I:

1. The model's test AUCs have been increased to 0.93 (the next 24 h) and 0.94 (the remainder horizons), that is, next to the AUC's ideal case (AUC = 1). The AUC refers essentially to the probability that a classifier shall rank a positive sample higher than a negative one (ZAKI; MEIRA JR., 2013). Classifiers scoring their AUCs next to the maximum value have their TPRs and FPRs, respectively, next to 1 and 0 in the ROC plot. Accordingly, such classifiers do not forecast many false negatives and effectively forecast the true positives (consequently, they also correctly forecast almost all samples of the negative class).

2. Results observed for the test FARs (0.12, 0.10, and 0.09, for the next 24, 48, and 72 h, respectively), TPRs (0.86, for the next 24 h, and 0.89, for the next 48 and 72 h), and TNRs (0.84, for the next 24 h, and 0.80, for the next 48 and 72 h) corroborate what we have just affirmed for the AUC: both TPR and TNR are simultaneously high, as well as the FAR reached a low level.

3. Both test skill scores (TSS and HSS) comprehending the quality of forecasts – positive and negative – are high, equaling about two-thirds of their ideal cases (TSS = 1 and HSS = 1) for the next 24 and 48 h (TSS = 0.70 and HSS = 0.69), and 72 h (TSS = 0.69 and HSS = 0.68). Besides, the achieved HSSs suggest that our models are far from random guessing (i. e., HSS = 0) or outputting wrong forecasts (i. e., HSS < 0) (JOLLIFFE; STEPHENSON, 2003). Accordingly, because of our high TPR and TNR results in all horizons, our TSSs reached reasonably high levels, namely TSSs ≥ 0.69.

4. By observing the differences between the test TPRs and TNRs over all horizons compared to the baseline, we could notice the methodology successfully corrected their class skew. Noticeably, our method scored the absolute differences of recalls by 0.02 (-0.07, for the next 24 h) and 0.09 (-0.18, for the next 48 h; and -0.24, for the next 72 h).

6.1.3 Literature comparison

Table 6.10 shows the literature results along with our designed models. As previously commented in Chapter 3, the models differ a lot concerning their data segmentation strategies

(training and test sets), target features, type of prediction, and data sources, even though their main goal is to predict ≥ C-class events. Unless the datasets are identical, there is not enough meaning in comparing metrics from different methods, as argued by Barnes and Leka (2008) and Barnes, Leka, et al. (2016). It is not clear whether the observed differences are merit or because of the different datasets.

Hence, the scores mentioned in Table 6.10 should not be by any means directly judged one against the other, that is, results commented herein should not be considered direct comparisons of scores between different systems. Instead, we shall analyze systems according to some identified interplay effects involving accuracy, recall, precision, and the number of false alarms, seeking over-fitted systems. Besides, for better visualization and understanding of score levels, we shall group systems according to their TSSs, which roughly serve as a benchmark for distinct forecast approaches (BLOOMFIELD et al., 2012).

Remarks from the literature comparison of biased results

Overall, we could not analyze all results posing some bias in Table 6.10 in a similar manner. We found no similar article comprehending our longest time horizon. On the other hand, some articles lacked enough data for analysis, such as the ones by Florios et al. (2018) and Anastasiadis et al. (2017). For the rest of the approaches, we managed to distinguish them between forecasts in the next 24 and 48 h.

Regarding flare forecasting within the next 24 h, the first group of articles scored $0.65 \leq \text{TSS} < 0.85$, namely the researches by Domijan, Bloomfield, and Pitié (2019) (TSS = 0.67 and 0.84), X. Li et al. (2020) (TSS = 0.67), and our model (TSS = 0.69). By achieving TSS scores in this interval, they were able to keep their TPR and TNR scores at close levels. Noticeably, their TPRs lied on [0.86 , 0.95], whereas their TNRs, [0.79 , 0.89]. As they scored ACC results ([0.81 , 0.89]) close to the TPRs and TNRs, we could not suggest over-fitted systems here (i. e., increased ACCs and TNRs along with decreased TPRs).

However, the remainder of models for forecasting within the next 24 h scored $0.45 \leq \text{TSS} < 0.65$, that is, the researches by Yang et al. (2013) (TSS = 0.47), Huang, H. Wang, Xu, et al. (2018) (TSS = 0.49), Ahmed et al. (2013) (TSS = 0.53), X. Wang et al. (2020) (TSS = 0.56), H. Liu et al. (2019) (TSS = 0.60), and Muranushi et al. (2015) (TSS = 0.63). Despite scoring TSS results at lower levels, they scored TPRs and TNRs most of the time without a specific class

Table 6.10: Comparison of models for ≥ C-class flare forecasting.

Forecasting time	*Authorship*	*Grouping*	*ACC*	*TPR*	*TNR*	*TSS*	*FAR*
next 24 h	Our RandomForest model[a]	biased results	0.85	0.86	0.82	0.69	0.16
next 48 h	Our RandomForest model	biased results	0.87	0.90	0.81	0.70	0.10
next 72 h	Our AdaBoost model	biased results	0.88	0.90	0.83	0.73	0.07
next 24 h	Muranushi et al. (2015)	biased results	0.81	0.80	0.83	0.63	0.07
next 24 h	Huang, H. Wang, Xu, et al. (2018)	biased results	0.76	0.73	0.76	0.49	0.65
next 48 h	Huang, H. Wang, Xu, et al. (2018)	biased results	0.81	0.81	0.81	0.62	0.84
next 24 h	Ahmed et al. (2013)	biased results	0.96	0.52	0.98	0.53	0.25
next 24 h	Yang et al. (2013)	biased results	0.76	0.61	0.84	0.47	-
next 24 h	Domijan, Bloomfield, and Pitié (2019)	biased results	0.89	0.95	0.89	0.84	-
next 24 h	Domijan, Bloomfield, and Pitié (2019)	biased results	0.81	0.87	0.80	0.67	-
next 24 h	H. Liu et al. (2019)	biased results	0.82	0.76	0.84	0.60	-
next 24 h	Florios et al. (2018)	biased results	0.84	-	-	0.60	-
next 24 h	X. Wang et al. (2020)	biased results	0.88	0.63	0.93	0.56	0.36
next 24 h	X. Li et al. (2020)	biased results	0.86	0.88	0.79	0.67	0.09
next 24 h	Anastasiadis et al. (2017)	biased results	-	-	-	0.25	-
next 24 h	Our RandomForest model[b]	unbiased results	0.85	0.86	0.84	0.70	0.12
next 48 h	Our RandomForest model	unbiased results	0.86	0.89	0.80	0.70	0.10
next 72 h	Our AdaBoost model	unbiased results	0.85	0.89	0.80	0.69	0.09
next 24 h	Nishizuka, Sugiura, Kubo, Den, and Ishii (2018)	unbiased results	0.82	0.81	0.82	0.63	0.47
next 24 h	Hada-Muranushi et al. (2016)	unbiased results	0.66	0.72	0.57	0.30	0.30
next 24 h	Colak and Qahwaji (2009)	unbiased results	0.81	0.81	-	-	0.30
next 24 h	Leka, Barnes, and Wagner (2018)	unbiased results	0.75	0.69	0.82	0.51	0.17
24 h – 48 h	Leka, Barnes, and Wagner (2018)	unbiased results	0.77	0.71	0.83	0.55	0.16
48 h – 72 h	Leka, Barnes, and Wagner (2018)	unbiased results	0.71	0.60	0.85	0.45	0.17
next 24 h	Leka, Barnes, and Wagner (2018)	unbiased results	0.92	0.30	0.99	0.29	0.30
24 h – 48 h	Leka, Barnes, and Wagner (2018)	unbiased results	0.93	0.27	0.99	0.26	0.25
48 h – 72 h	Leka, Barnes, and Wagner (2018)	unbiased results	0.94	0.27	0.99	0.26	0.30
next 24 h	Bloomfield et al. (2012)	unbiased results	0.71	0.75	0.70	0.45	0.64
next 24 h	E. Park et al. (2018)	unbiased results	0.82	0.85	0.78	0.63	0.17
next 24 h	Benvenuto et al. (2018)	unbiased results	0.83	0.53	0.89	0.43	0.47
next 24 h	Díscola Jr. et al. (2018a)	unbiased results	0.72	0.70	0.79	0.49	-
next 24 h	Díscola Jr. et al. (2018b)	unbiased results	0.91	0.94	0.86	0.80	-

[a] Biased results of our model refer to those from the evaluation of validation sets.
[b] Unbiased results of our model refer to those from the evaluation of test sets.

preference. Whereas their positive hit rates ranged on [0.52 , 0.80], the negative ones lied on [0.76 , 0.98].

By most of the time in the last paragraph, we meant we could identify Ahmed et al. (2013)'s and X. Wang et al. (2020)'s models as potential over-fitted systems in favor of their negative classes. As such, Ahmed et al. (2013) scored a high ACC (0.96) and TNR (0.98), but their TPR only equaled 0.52, that is, their model was not able to generalize well positive samples. In turn, X. Wang et al. (2020) scored ACC = 0.88, TPR = 0.63, and TNR = 0.93.

Besides, except for Yang et al. (2013), which did not inform their false alarm ratio or positive precision, Huang, H. Wang, Xu, et al. (2018) somehow output a high FAR (0.65). We argue here

that by trying to improve their TPRs, those authors may have harmed the precision of their systems (i. e., the precision-recall trade-off), thus increasing their FAR. Huang, H. Wang, Xu, et al. (2018)'s precision corroborate this statement, for it only equaled FAR = 0.35.

Other articles suffering to some extent from false alarms are those by H. Liu et al. (2019) and X. Wang et al. (2020). Although H. Liu et al. (2019) did not provide their FAR, their PPV equaled 0.54. On the other hand, X. Wang et al. (2020) scored FAR = 0.36 – which corroborated their PPV = 0.63.

On the other hand, in the group of forecasting models within the next 48 h, there exists only our model (TSS = 0.70) and Huang, H. Wang, Xu, et al. (2018)'s (TSS = 0.62). As those approaches scored high ACCs (0.87 and 0.81, respectively), TPRs (0.90 and 0.81, respectively), and TNRs (both equaled 0.81), none of them incurred in over-fitting. However, as of Huang, H. Wang, Xu, et al. (2018)'s model for the next 24 h, their forecasting approach in the next 48 h may also have had the precision harmed by the precision-recall trade-off, that is, their FAR equaled 0.84 here.

Remarks from the literature comparison of unbiased results

In turn, while posing unbiased results when forecasting within the next 24 h, the first group of articles scored $0.50 \leq \text{TSS} \leq 0.80$ and comprehended the researches by Leka, Barnes, and Wagner (2018) (TSS = 0.51), Nishizuka, Sugiura, Kubo, Den, and Ishii (2018) (TSS = 0.63), E. Park et al. (2018) (TSS = 0.63), our model (TSS = 0.70), and Díscola Jr. et al. (2018b) (TSS = 0.80). Their TPRs ranged over [0.69 , 0.94] and TNRs [0.78 , 0.86], with ACCs varying over [0.75 , 0.91]. Hence, we could not suggest over-fitted systems.

However, we could observe some harmed precision scores, which somehow leveraged the number of false alarms, such as with Nishizuka, Sugiura, Kubo, Den, and Ishii (2018) (FAR = 0.47). Their PPV corroborates the FAR number, for having equaled only PPV = 0.53.

In turn, the remainder of models for forecasting within the next 24 h scored $0.25 \leq \text{TSS} < 0.50$, that is, the researches by Leka, Barnes, and Wagner (2018) (TSS = 0.29), Hada-Muranushi et al. (2016) (TSS = 0.30), Benvenuto et al. (2018) (TSS = 0.43), Bloomfield et al. (2012) (TSS = 0.45), and Díscola Jr. et al. (2018a) (TSS = 0.49). For this group, TPRs varied over [0.30 , 0.75] and TNRs in [0.57 , 0.99]. As their ACCs lied around higher levels, [0.66 , 0.92], we could infer over-fitted systems in Leka, Barnes, and Wagner (2018)'s and Benvenuto et al.

(2018)'s approaches, i. e., they had high TNRs (0.99 and 0.89, respectively) and ACCs (0.92 and 0.83, respectively), along with low TPRs (0.30 and 0.53, respectively).

Besides, we could also observe a high false alarm ratio with Bloomfield et al. (2012)'s approach (FAR = 0.64). Although we could not identify their PPV, we may also suggest this high ratio as directly related to the precision-recall trade-off, i. e., their decision threshold might have been shifted to a position where both TPR and FAR increased at once.

6.2 Case Study II: ≥ M-class flare forecasting

This section introduces and discusses Case Study II's results, namely the one aimed at forecasting ≥ M flares up to three days ahead. Following the content organization introduced in Section 6.1, we will summarize results in Section 6.2.1[9] and compare them to the specialized literature in Section 6.2.2.

6.2.1 Results analysis

As in Case Study I's horizons, those from Case Study II followed similar trends in results, as the summary of results in Table 6.11[10] shows. Accordingly, some longer horizons have achieved some scores slightly better than their predecessors (i. e., the TSS and AUC of both model and feature selection for the next 24, 48, and 72 h). Once more, this may have happened because of the overlapping forecasting horizons (COLAK; QAHWAJI, 2009): the number of positive events tended to positively increase while increasing the length of our horizons for ≥ M flares, thus consequently minimizing the associated data skew.

For all forecasting horizons, our method has designed models relying on the GradientTreeBoosting algorithm. We could observe TSS increases over all horizons from the model selection to cut-off point adjustments of such models, notably by +0.22 (for the next 24 h) and +0.13 (for the next 48 and 72 h). Not only the TSSs, but also the HSSs have been improved: +0.10, +0.07, and +0.11 respectively for the next 24, 48, and 72 h.

Results observed in the TPRs of model selection (0.53, 0.64, and 0.67 respectively for the next 24, 48, and 72 h) and their corresponding TNRs (0.79, 0.75, and 0.75) suggest that models somehow coped with skewed class ratios in the beginning. However, by the resampling of data,

[9] Tables containing the detailed results for the validation/test sets of Case Study II are in Appendix B, including their statistical significance analysis.

[10] For the statistical tests for significance, refer to Appendix B.

Table 6.11: Case Study II, ≥ M flares, the next 24, 48 and 72 h: summary of results.

Process	*Forecasting Time*	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*AUC*	*TSS*	*HSS*
Model selection	next 24 h	0.74	0.53	0.79	0.36	0.89	0.64	0.67	0.32	0.27
	next 48 h	0.72	0.64	0.75	0.48	0.86	0.52	0.72	0.39	0.35
	next 72 h	0.72	0.67	0.75	0.56	0.83	0.44	0.73	0.42	0.39
Feature selection	next 24 h	0.74	0.56	0.78	0.36	0.89	0.64	0.69	0.34	0.27
	next 48 h	0.73	0.66	0.75	0.48	0.87	0.52	0.73	0.41	0.36
	next 72 h	0.73	0.71	0.74	0.56	0.85	0.44	0.75	0.45	0.42
Hyperparameter optimization	next 24 h	0.85	0.27	0.97	0.66	0.86	0.34	0.85	0.24	0.32
	next 48 h	0.81	0.52	0.91	0.67	0.84	0.33	0.85	0.43	0.46
	next 72 h	0.80	0.61	0.88	0.71	0.83	0.29	0.86	0.50	0.51
Data resampling	next 24 h	0.77	0.75	0.78	0.42	0.94	0.58	0.85	0.53	0.40
	next 48 h	0.76	0.73	0.78	0.53	0.89	0.47	0.84	0.50	0.45
	next 72 h	0.78	0.76	0.79	0.62	0.88	0.38	0.86	0.55	0.52
Cut-off point adjust	next 24 h	0.74	0.82	0.72	0.38	0.95	0.62	0.85	0.54	0.37
	next 48 h	0.73	0.83	0.69	0.48	0.92	0.52	0.84	0.52	0.42
	next 72 h	0.77	0.80	0.75	0.60	0.89	0.40	0.86	0.55	0.50
Evaluation of validation sets (baseline)	next 24 h	0.74	0.51	0.79	0.35	0.89	0.65	0.64	0.30	0.25
	next 48 h	0.75	0.60	0.80	0.51	0.86	0.49	0.73	0.40	0.38
	next 72 h	0.75	0.69	0.78	0.59	0.84	0.41	0.78	0.47	0.45
Evaluation of validation sets (optimized)	next 24 h	0.74	0.82	0.72	0.39	0.95	0.61	0.85	0.55	0.38
	next 48 h	0.73	0.83	0.69	0.48	0.92	0.52	0.85	0.52	0.42
	next 72 h	0.77	0.81	0.75	0.60	0.89	0.40	0.86	0.56	0.51
Evaluation of test sets (baseline)	next 24 h	0.75	0.54	0.80	0.38	0.89	0.62	0.66	0.34	0.29
	next 48 h	0.75	0.59	0.81	0.52	0.85	0.48	0.85	0.55	0.38
	next 72 h	0.73	0.70	0.75	0.56	0.84	0.44	0.78	0.44	0.42
Evaluation of test sets (optimized)	next 24 h	0.72	0.83	0.70	0.37	0.95	0.63	0.84	0.53	0.36
	next 48 h	0.74	0.85	0.70	0.50	0.93	0.50	0.85	0.55	0.45
	next 72 h	0.75	0.80	0.73	0.58	0.89	0.42	0.85	0.53	0.48

the absolute differences between both recalls have been positively affected, thus scoring 0.03 (-0.23, next 24 h), 0.05 (-0.06, next 48 h), and 0.03 (-0.05, next 72 h).

In agreement with the reduced absolute differences mentioned earlier, our methodology scored TPRs ≥ 0.80 over all horizons by their cut-off point adjustments, which comprehended improvements in the positive recall by +0.29 (TPR = 0.82, next 24 h), +0.19 (TPR = 0.83, next 48 h), and +0.13 (TPR = 0.80, next 72 h). As a measure of potential usefulness for the positive forecasts (ZAKI; MEIRA JR., 2013), the AUCs have increased by +0.18 (AUC= 0.85, next 24 h), +0.12 (AUC = 0.84, next 48 h), and +0.13 (AUC = 0.86, next 72 h).

Conversely to Case Study I, when we had reasonable low FAR levels by the cut-off point adjustment of classifiers, our methodology could not decrease such score in the same manner when forecasting ≥ M classes. Whereas the ratios for false alarms lay on [0.44 , 0.64] in the beginning, they slightly decreased to [0.40 , 0.62] during the search for prediction thresholds.

It is worth noting that, despite not decreasing FAR significantly, our methodology could maintain it lying close to the level of its beginning by the time of models had their thresholds adjusted. For instance, for forecasting in the next 24 h, from the model selection to cut-off point adjustment, both TPR (0.82) and TSS (0.53) significantly increased by +0.22 and +0.29, respectively, whereas the FAR remained with minor effects (FAR = 0.62, -0.02). The remainder horizons followed those change trends with the TSS, TPR, and FAR.

To complete the design processes of Case Study II, the methodology assessed the improvements in all models' forecast performance while forecasting the validation sets (using their cut-off point adjustment designs). Remarkably, it confirmed some positive effects, such as with the increasing TSSs (+0.25, +0.12, and +0.09 respectively for the next 24, 48, and 72 h) and HSSs (+0.13 and +0.06 respectively for the next 24 and 72 h)[11].

As the pipeline confirmed increases with the validation TSSs, it then proceeded with the generalization error assessment of models over the test sets. Overall, we could notice these remarkable effects in the test results of all forecasting horizons of Case Study II:

1. The methodology has minimized the class skew successfully in most horizons, thus reducing the absolute differences between both recalls: |TPR − TNR| = 0.07 (-0.19, next 24 h) and 0.05 (-0.17, next 48 h). The difference for the next 72 h was already short in the test baseline.

2. Our method managed to score test TSSs ≥ 0.53, that is, competing to the literature, thus consequently keeping both TPR (0.83, 0.85, and 0.80 respectively for the next 24, 48, and 72 h) and TNR (0.70, for the next 24 and 48 h, and 0.73, for the next 72 h) at high levels. As a reference, the highest results in the literature for unbiased ≥ M flare forecasts in the next 24 h scored 0.53 ≤ TSS ≤ 0.83, 0.70 ≤ TPR ≤ 0.95, and 0.70 ≤ TNR ≤ 0.86 (we shall focus on the comparison to the literature in Section 6.2.2).

3. All test AUCs scored [0.84 , 0.85] − close to the score's optimum level (AUC = 1), thus increasing the probability of forecasting a random positive sample over a negative one and suggesting potential useful positive forecasts.

Conversely to ≥ C forecasts, in which models scored TSSs ≥ 0.69, the TSSs for ≥ M-class flares scored ≥ 0.53, results about 0.2 less than the former. Noteworthily, forecasting M-class

[11]For the complete analysis of improved/harmed/unaffected validation scores of Case Study II, refer to Appendix B.

flares is a more challenging task than C events. As such, the decision boundaries of positive and negative ≥ M instances might have been overlapped to some extent, which increased the difficulty to classify them. Analogously, the decision boundaries of ≥ C events might have been more well-defined, which decreased the classification difficulty level, thus consequently increasing the performance of classifiers.

6.2.2 Literature comparison

Table 6.12 shows the indirect comparison of results for ≥ M-class forecasting between our models and the specialized literature. As for ≥ C-class events, we included research posing some bias in their results on the upper hand part and used our validation results as a reference. On the other hand, we included unbiased results in the lower hand table part aside with our test scores.

Remarks from the literature comparison of biased results

Overall, we could not analyze all results posing some bias in Table 6.12 in a similar manner. There were cases of insufficient data, such as the articles by X. Zhang, J. Liu, and Q. Wang (2011), Jonas et al. (2018), Florios et al. (2018), Anastasiadis et al. (2017) and Alipour, Mohammadi, and Safari (2019) – which turned the analysis unavailable –, or with less frequent time horizons. For instance, our model for the next 72 h and Bobra and Couvidat (2015)'s approaches for forecasting 24 and 48 h exactly after a t instant. For the rest of articles, we managed to distinguish them between forecasts for the next 24 and 48 h.

Regarding events forecasting within the next 24 h, the first group of articles scored $0.40 < \text{TSS} \leq 0.65$, namely the researches by Yang et al. (2013) (TSS = 0.48), Muranushi et al. (2015) (TSS = 0.52), C. Liu et al. (2017) (TSS = 0.53), Jiao et al. (2020) (TSS = 0.51), and our model (TSS = 0.55). By achieving TSS scores in this interval, they were able to keep their TPR and TNR scores most of the time at close levels (without preference for a specific class). Noticeably, their TPRs lied on [0.41 , 0.85], whereas their TNRs varied over [0.67 , 0.97].

By most of the time in the last paragraph, we meant we could identify Yang et al. (2013)'s model as a potential over-fitted system in favor of the negative class. As such, they scored a high ACC = 0.90 and TNR = 0.96, but their TPR only equaled 0.41, that is, their model was not able to generalize well positive samples.

Table 6.12: Comparison of models for ≥ M-class flare forecasting.

Forecasting Time	*Authorship*	*Grouping*	*ACC*	*TPR*	*TNR*	*TSS*	*FAR*
next 24 h	Our GradientTreeBoosting model	biased results	0.74	0.82	0.79	0.55	0.65
next 48 h	Our GradientTreeBoosting model	biased results	0.73	0.83	0.69	0.52	0.52
next 72 h	Our GradientTreeBoosting model	biased results	0.77	0.81	0.78	0.56	0.41
next 24 h	Yang et al. (2013)	biased results	0.90	0.41	0.96	0.48	-
next 48 h	Yang et al. (2013)	biased results	0.86	0.43	0.95	0.53	-
next 48 h	J.-F. Liu, F. Li, Wan, et al. (2017)	biased results	-	0.64	0.83	0.47	-
next 48 h	J.-F. Liu, F. Li, H.-P. Zhang, et al. (2017)	biased results	0.75	0.76	0.74	0.50	-
next 24 h	C. Liu et al. (2017)	biased results	0.76	0.74	0.78	0.53	-
next 48 h	R. Li and Zhu (2013)	biased results	0.82	0.69	0.83	0.52	-
next 48 h	R. Li, H. Wang, et al. (2011)	biased results	0.74	0.69	0.75	0.44	-
next 48 h	Huang and H.-N. Wang (2013)	biased results	0.72	0.72	0.71	0.71	0.70
next 48 h	X. Zhang, J. Liu, and Q. Wang (2011)	biased results	-	0.75	-	-	-
next 48 h	Yu, Huang, H. Wang, and Cui (2009)	biased results	-	0.82	0.84	0.66	-
next 48 h	Yu, Huang, Q. Hu, et al. (2010)	biased results	0.92	0.94	0.91	0.86	0.28
next 48 h	Yu, Huang, H. Wang, Cui, et al. (2010)	biased results	-	0.85	0.87	0.72	0.28
exact 24 h	Bobra and Couvidat (2015)	biased results	0.92	0.83	0.92	0.76	-
exact 48 h	Bobra and Couvidat (2015)	biased results	0.94	0.86	0.94	0.81	-
next 48 h	Raboonik et al. (2016)	biased results	0.94	0.97	0.88	0.85	0.05
next 24 h	Muranushi et al. (2015)	biased results	0.7	0.85	0.67	0.52	0.35
next 24 h	Jonas et al. (2018)	biased results	-	-	-	0.81	-
next 48 h	Huang, Yu, et al. (2010)	biased results	-	0.91	0.87	0.78	-
next 24 h	Huang, H. Wang, Xu, et al. (2018)	biased results	0.81	0.85	0.81	0.66	0.90
next 48 h	Huang, H. Wang, Xu, et al. (2018)	biased results	0.81	0.81	0.81	0.62	0.84
next 24 h	Sadykov and Kosovichev (2017)	biased results	0.87	0.89	0.86	0.76	0.77
next 24 h	Nishizuka, Sugiura, Kubo, Den, Watari, et al. (2017)	biased results	0.99	0.90	0.99	0.90	0.07
next 24 h	H. Liu et al. (2019)	biased results	0.90	0.88	0.91	0.79	-
next 24 h	Florios et al. (2018)	biased results	0.93	-	-	0.74	-
next 24 h	Alipour, Mohammadi, and Safari (2019)	biased results	-	-	-	0.95	-
next 24 h	X. Wang et al. (2020)	biased results	0.94	0.73	0.95	0.68	0.72
next 24 h	X. Li et al. (2020)	biased results	0.89	0.81	0.93	0.74	0.11
next 24 h	Anastasiadis et al. (2017)	biased results	-	-	-	0.30	-
next 24 h	Jiao et al. (2020)	biased results	-	0.54	0.97	0.51	-
next 24 h	Our GradientTreeBoosting model	unbiased results	0.75	0.83	0.70	0.53	0.63
next 48 h	Our GradientTreeBoosting model	unbiased results	0.75	0.85	0.70	0.55	0.48
next 72 h	Our GradientTreeBoosting model	unbiased results	0.73	0.80	0.75	0.53	0.44
next 24 h	Bloomfield et al. (2012)	unbiased results	0.83	0.70	0.83	0.53	0.85
next 24 h	Shin et al. (2016)	unbiased results	-	0.61	0.76	0.37	0.78
next 24 h	Leka, Barnes, and Wagner (2018)	unbiased results	0.89	0.20	0.99	0.19	0.21
24 h – 48 h	Leka, Barnes, and Wagner (2018)	unbiased results	0.87	0.03	1.00	0.03	0.20
48 h – 72 h	Leka, Barnes, and Wagner (2018)	unbiased results	0.87	0.06	1.00	0.05	0.13
next 24 h	Nishizuka, Sugiura, Kubo, Den, and Ishii (2018)	unbiased results	0.86	0.95	0.86	0.80	0.82
next 24 h	Hada-Muranushi et al. (2016)	unbiased results	0.82	0.39	0.88	0.27	0.68
next 24 h	McCloskey, P. T. Gallagher, and Bloomfield (2018)	unbiased results	-	-	-	0.47	-
next 24 h	D. Falconer et al. (2011) and D. A. Falconer et al. (2014)	unbiased results	0.95	0.31	-	0.47	0.50
next 24 h	D. A. Falconer et al. (2014)	unbiased results	0.95	0.38	-	0.49	0.48
next 24 h	Benvenuto et al. (2018)	unbiased results	0.91	0.53	0.92	0.45	0.77

However, the remainder of models for forecasting within the next 24 h scored $0.65 < \text{TSS} \leq 0.90$, that is, the researches by Huang, H. Wang, Xu, et al. (2018) (TSS = 0.66), X. Wang et al. (2020) (TSS = 0.68), X. Li et al. (2020) (TSS = 0.74), Sadykov and Kosovichev (2017) (TSS = 0.76), H. Liu et al. (2019) (TSS = 0.79), and Nishizuka, Sugiura, Kubo, Den, Watari, et al. (2017) (TSS = 0.90). Besides scoring TSSs at higher levels, they also managed to reach

close levels with their TPRs ([0.81 , 0.90]) and TNRs ([0.81 , 0.99]). As they scored increased ACCs ([0.81 , 0.99]), we could not suggest over-fitted systems here.

Noteworthily, within the next 24 h, some models suffered from the precision-recall trade-off, thus incurring in the forecasting of increased numbers of false alarms: the proposals by Huang, H. Wang, Xu, et al. (2018) (FAR = 0.90), Sadykov and Kosovichev (2017) (FAR = 0.77), and X. Wang et al. (2020) (FAR = 0.72). In comparison, our model in this time horizon also increased this ratio. However, this issue became less evident than the previous articles. Nevertheless, FAR comprehends a worth caring aspect for future work in our dataset: leveraging the precision to decrease the occurrence of false alarms consequently.

Not only with Huang, H. Wang, Xu, et al. (2018)'s, Sadykov and Kosovichev (2017)'s, and X. Wang et al. (2020)'s researches, but also with Bobra and Couvidat (2015)'s the trade-off mentioned earlier was evident. Despite not providing the model's FAR for their exact 24 h time horizon, we could suggest an increased ratio here, as they under-performed with the positive precision (PPV = 0.41). Besides, other articles under-performing with their precision scores were H. Liu et al. (2019)'s (PPV = 0.22) and X. Wang et al. (2020) (PPV = 0.27).

On the other hand, regarding flare forecasting within the next 48 h, the first group of articles also scored $0.40 < TSS \leq 0.65$: R. Li, H. Wang, et al. (2011) (TSS = 0.44), J.-F. Liu, F. Li, Wan, et al. (2017) (TSS = 0.47), J.-F. Liu, F. Li, H.-P. Zhang, et al. (2017) (TSS = 0.50), R. Li and Zhu (2013) (TSS = 0.52), our model (TSS = 0.52), Yang et al. (2013) (TSS = 0.53), Huang, H. Wang, Xu, et al. (2018) (TSS = 0.62), and X. Wang et al. (2020) (TSS = 0.68). By achieving TSS scores at a fair interval, they were able to keep their TPR and TNR scores most of the time at close levels, which means without preference for a specific class. Noticeably, their TPRs lied on [0.43 , 0.83], whereas their TNRs [0.69 , 0.95].

Once more, by most of the time in the last paragraph, we meant we could identify Yang et al. (2013)'s model as a potential over-fitted system in favor of the negative class. As such, they scored a high ACC = 0.86 and TNR = 0.95, but their TPR only equaled 0.43, that is, their model was not able to generalize well positive samples. Besides, R. Li and Zhu (2013) also seemed to present a slightly over-fitted classifier – yet less evident than Yang et al. (2013)'s approach. In this sense, their model scored ACC = 0.82, TPR = 0.69, and TNR = 0.83.

The remainder of models for forecasting within the next 48 h scored $0.65 < TSS \leq 0.90$: Yu, Huang, H. Wang, and Cui (2009) (TSS = 0.66), Huang and H.-N. Wang (2013) (TSS = 0.71), Yu, Huang, H. Wang, Cui, et al. (2010) (TSS = 0.72), Huang, Yu, et al. (2010) (TSS = 0.78), Raboonik

et al. (2016) (TSS = 0.85), and Yu, Huang, Q. Hu, et al. (2010) (TSS = 0.86). Besides scoring TSSs at higher levels, those approaches also managed to reach close levels with their TPRs ([0.72 , 0.97]) and TNRs ([0.71 , 0.91]). However, only Raboonik et al. (2016) (ACC = 0.94), Huang and H.-N. Wang (2013) (ACC = 0.72), and Yu, Huang, Q. Hu, et al. (2010) (ACC = 0.92) provided their accuracies, which showed classifiers not over-fitting to any class.

Noteworthily, within the next 48 h, the model by Huang and H.-N. Wang (2013) suffered from the precision-recall trade-off, which incurred in the forecasting of an increased number of false alarms (FAR = 0.70). In comparison, our model in this time horizon also increased this ratio (FAR = 0.52). However, once more, this issue became less evident than the former. Besides, this issue was also less evident than for the next 24 h. Nevertheless, FAR comprehends a worth caring aspect for future work in our dataset, i. e., leveraging the precision to decrease the occurrence of false alarms consequently.

Remarks from the literature comparison of unbiased results

In turn, while posing unbiased results in Table 6.12, articles mainly focused on the next 24 h time horizon. However, there were also proposals lacking enough data (MCCLOSKEY; GALLAGHER, P. T.; BLOOMFIELD, 2018) or with less frequent time horizons, such as the researches by Leka, Barnes, and Wagner (2018) (24 h – 48 h and 48 h – 72 h) and our models designed for the next 48 and 72 h.

Among Leka, Barnes, and Wagner (2018)'s results, we could identify reasonably high imbalanced class ratio scenarios for both 24 h – 48 h and 48 h – 72 h. As such, they have only reached hit rates of 0.03 (24 h – 48 h) and 0.06 (48 h – 72 h), whereas scoring perfect TNRs of a hundred percent in both time horizons. As such, ACC reached 0.87 in both horizons.

On the other hand, regarding ≥ M-class events forecasting within the next 24 h, papers mostly managed to score TSSs < 0.5 (seven approaches) and also ≥ 0.5 (three approaches). The latter case included our model (TSS = 0.53) and the papers by Bloomfield et al. (2012) (TSS = 0.53) and Nishizuka, Sugiura, Kubo, Den, and Ishii (2018) (TSS = 0.80).

The approaches scoring TSSs ≥ 0.5 varied their TPRs over [0.70 , 0.95] and TNRs [0.70 , 0.86]. For having achieved ACCs lying on [0.75 , 0.86], we could not suggest over-fit proposals here. However, all of them suffered from false alarms in their positive forecasts – probably because of prediction thresholds worth better adjusting that harmed their precision indexes –, especially

Nishizuka, Sugiura, Kubo, Den, and Ishii (2018) (FAR = 0.82) and Bloomfield et al. (2012) (FAR = 0.85).

In turn, concerning the remainder of models – the ones scoring TSS < 0.5 –, except for the proposal lacking enough data to analyze (MCCLOSKEY; GALLAGHER, P. T.; BLOOMFIELD, 2018), the TPRs and TNRs varied on [0.20 , 0.61] and [0.76 , 0.99], respectively (SHIN et al., 2016; LEKA; BARNES; WAGNER, 2018; HADA-MURANUSHI et al., 2016; FALCONER, D. et al., 2011; FALCONER, D. A. et al., 2014; BENVENUTO et al., 2018). As their ACCs have reached high levels ([0.82 , 0.95]), we could suggest over-fitted systems here because of the discrepancy between score ranges.

Noticeably, the articles by Leka, Barnes, and Wagner (2018), Shin et al. (2016), Hada-Muranushi et al. (2016), D. Falconer et al. (2011), D. A. Falconer et al. (2014) and Benvenuto et al. (2018) over-performed with the negative class, thus poorly generalizing with the positive score. Besides, as of models scoring TSS ≥ 0.5, almost all approaches also suffered from prediction thresholds worth better adjusting, thus outputting decreased precision scores: Hada-Muranushi et al. (2016) (FAR = 0.68), D. Falconer et al. (2011) (FAR = 0.50), D. A. Falconer et al. (2014) (FAR = 0.48), Benvenuto et al. (2018) (FAR = 0.77), and Shin et al. (2016) (FAR = 0.78).

6.3 Case Study III: ≥ M-class flare forecasting with C. Liu et al. (2017)'s dataset

This section will present the results from Case Study III, namely the one designed for ≥ M flare forecasting in the next 24 h using the dataset proposed by C. Liu et al. (2017).

6.3.1 Results analysis

Table 6.13 shows the summary of results for the methodology inner processes of Case Study III. Tables containing the detailed results from the validation/test sets of such case study are in Appendix B.

As in the Case Study I for the next 24 and 48 h, the model optimized in Case Study III was based on the RandomForest algorithm – the original research from C. Liu et al. (2017) also comprehended such algorithm. Concerning its input data, although we initialized the feature selection with the full set borrowed from C. Liu et al. (2017) – that is, TOTUSJH, TOTBSQ, TOTPOT, TOTUSJZ, ABSNJZH, SAVNCPP, USFLUX, AREA_ACR, TOTFZ, MEANPOT,

Table 6.13: Case Study III, ≥ M flares, the next 24 h: summary of results.

Process	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*AUC*	*TSS*	*HSS*
Model selection	0.84	0.29	0.97	0.75	0.85	0.25	0.79	0.26	0.33
Feature selection	0.84	0.35	0.96	0.70	0.86	0.30	0.78	0.31	0.38
Hyperparameter optimization	0.84	0.35	0.96	0.69	0.86	0.31	0.83	0.31	0.38
Cost function analysis	0.76	0.76	0.76	0.44	0.93	0.56	0.83	0.52	0.40
Cut-off point adjustment	0.80	0.66	0.83	0.50	0.91	0.50	0.83	0.50	0.44
Evaluation of validation sets (baseline)	0.84	0.26	0.98	0.75	0.84	0.25	0.77	0.23	0.31
Evaluation of validation sets (optimized)	0.79	0.69	0.82	0.49	0.91	0.51	0.82	0.51	0.44
Evaluation of test sets (baseline)	0.87	0.35	0.99	0.76	0.87	0.04	0.86	0.34	0.43
Evaluation of test sets (optimized)	0.81	0.78	0.81	0.50	0.94	0.50	0.87	0.59	0.49

R_VALUE, EPSZ, and SHRGT45 – our method discarded most of them, thus maintaining only data from the TOTUSJH, ABSNJZH, SAVNCPP, and R_VALUE attributes.

The differing sets of features mentioned earlier may be linked to the nature of the feature selection methods: whereas C. Liu et al. (2017) have analyzed their feature importance through the Gini impurity score, we tested them for a complementary nature incurring in flare occurrence. Not surprisingly, Bobra and Couvidat (2015) also selected the same set of C. Liu et al. (2017) after working over a broader collection of attributes. However, their article employed the C. Liu et al. (2017)'s features on a different dataset.

Overall, we could observe HSSs higher than TSSs in both model and feature selection of Table 6.13. Those higher HSSs suggest the existence of imbalanced class ratios, which we confirmed by observing the ACCs, TPRs, and TNRs of both processes (i. e., ACC = 0.84, TPR = 0.29, and TNR = 0.97 and ACC = 0.84, TPR = 0.35, and TNR = 0.96, for model and feature selection, respectively).

Bloomfield et al. (2012) argued that the drawback with the HSS comprehends its biased output toward an increased score when the classifier is over-fitting – as the RandomForest by both model and feature selection. This is another reason why we used both HSS and TSS as the skill scores and not only HSS in addition to the rationale described in Chapter 4.

In hyperparameter optimization, we observed that the methodology could effectively boost the RandomForest's AUC to 0.83 (+0.05). As a measure of potential usefulness for positive forecasts, the AUC value is essentially the probability that classifiers shall rank a random positive sample higher than a random negative one (ZAKI; MEIRA JR., 2013).

Hence, classifiers outputting their AUCs next to 1 have an increased probability of ranking a random sample as positive – equal to 0.5 is equivalent to random guessing and below 0.5 is

the anti-learning (worst than random guessing) (ZAKI; MEIRA JR., 2013). In response to the adjusted AUC, other scores remained at the same level as in feature selection or incurred in minor effects (i. e., FAR and PPV).

Besides, it is worth noting that the AUC was insensitive to class skew: from the model selection to cut-off point adjustment, the AUC varied over [0.78 , 0.83], which intuitively can be interpreted as a high interval. That happened because both TPR – understood as the probability of predicting a positive sample as positive – and FPR – understood as the probability of predicting a negative sample as positive – do not depend on the class ratio size of positive and negative samples (ZAKI; MEIRA JR., 2013).

The AUC's insensitivity to the class skew is a desirable property in imbalanced domains since such score shall essentially vary on close levels whether the classes are balanced (i. e., cost function analysis/data resampling and cut-off point adjustment) or skewed (i. e., model and feature selection). That is another reason for using AUC during the hyperparameter optimization in addition to the rationale described in Chapter 4.

To correct the imbalanced class ratios of Case Study III in Table 6.13, our method has used the RandomForest's cost function, which remarkably zeroed the difference between both positive and negative recalls compared to the data resampling methods. At the cost function analysis, the TPR and NPV increased, as well as the TNR and PPV decreased, probably as a direct response to the precision-recall trade-off.

As a consequence of the PPV decrease mentioned earlier, the FAR increased by about a quarter compared to model selection. However, although we observed an increasing FAR (the methodology shall attempt to minimize it next), the TSS increased by exactly a quarter of its original value in model selection.

Within the solar flare forecasting domain, the costs with false alarms (false positives) are lower than with missed events (false negatives) (BOBRA; COUVIDAT, 2015; RABOONIK et al., 2016). However, one must employ efforts to reduce the false alarms ratio so this metric can come close to its optimum level, i. e., FAR = 0 (JOLLIFFE; STEPHENSON, 2003), which our method attempted to perform when adjusting the RandomForest's prediction threshold.

During the adjustment of the cut-off point in Table 6.13, in response to the shifting of both TPR and PPV scores so they could come to close levels, the FAR positively decreased by -0.06, as well as the PPV consequently increased (+0.06). Noteworthily, the TSS decreased by only -0.02 at this point.

As the final decision-making performed in Case Study III, our methodology assessed the improvements in the forecast performance of RandomForest while forecasting the validation sets (if any). Noticeably, we confirmed some positive effects, such as with the increasing TPR (+0.43) and TSS (+0.28)[12]. Since the pipeline identified an increase with the TSS, it then proceeded with the generalization error assessment over the test sets, whose observed improvements we summarized below:

1. Our method has shortened the distance between the test TPR and TNR successfully, thus consequently minimizing the over-fitted baseline test performance. Accordingly, the difference |TPR – TNR| has been positively improved by -0.61, decreasing from 0.64 to 0.03.

2. Our method managed to score both test TSS (0.59) and HSS (0.49) quality skill scores at reasonable levels (compared to the literature), thus consequently keeping both test TPR = 0.78 and TNR = 0.81 at high levels. As a reference, the results in the literature for biased ≥ M-class flare forecasts[13] in the next 24 h scored $0.40 < \text{TSS} \leq 0.65$, $0.41 \leq \text{TPR} \leq 0.85$, and $0.67 \leq \text{TNR} \leq 0.97$.

3. The RandomForest scored a test AUC = 0.87, that is, next to the score's optimum level (AUC = 1). As the AUC is a measure of potential usefulness for the positive forecasts, we can affirm the positive forecasts were effective.

Finally, it is worth commenting on the observed decrease of the test PPV leading to an increase of FAR, as they are related one to each other. From the baseline test performance to the optimized one, the PPV has been decreased by about a quarter of its original value. This certainly made our RandomForest forecast a higher number of false alarms (i. e., FAR +0.46).

As a reference, C. Liu et al. (2017) scored a PPV (0.77) close to our baseline test precision (0.76). In this sense, we can assume they also produced false alarms to some extent (regardless of not providing this ratio). However, whereas C. Liu et al. (2017) scored its PPV over the full set of records, our method was fed with a decreased number of available samples while designing its RandomForest (because of the test sets reserved in the beginning).

Within this context, we suggest that the decrease in our test PPV may be related to the decreased number of samples used during our training. This scenario may somehow have

[12]For a complete analysis of improved/harmed/unaffected scores refer to Appendix B.

[13]Although we are discussing test scores, for conciseness, we referenced the literature's biased forecast performance because C. Liu et al. (2017) only included AR records near the Sun's central meridian.

harmed our RandomForest's generalization skill since it did not have as many available samples as C. Liu et al. (2017) allowing a finer discriminating between features and events.

Classifiers designed with lower numbers of samples do not generalize as well as those using higher numbers (PYLE, 1999). In fact, C. Liu et al. (2017) used 845 samples while designing their model, whereas our method employed only 634 samples for design and the remainder 210 for the test (about 42 – 5% – for each test set). Nevertheless, we believe that employing only 845 samples represents a small and restricted dataset, that is, one should consider increasing the dataset size whether the aim is to reach better performance levels in Case Study III.

6.3.2 Literature comparison

In C. Liu et al. (2017)'s dataset, we could notice that there were more C-class events (n = 552) in comparison to M (n = 142), X (n = 23), and the absence of them (n = 128). They binarized the events to design their target feature, thus grouping non-flare samples with C-class ones (negative class) and M-class events with X ones (positive class).

In an attempt to minimize their imbalanced class ratios, they repeatedly and randomly selected 100 times 165 negative samples to couple with the M/X-class ones, thus creating 100 distinct downsampled subsets. To evaluate the performance of their RandomForest model, C. Liu et al. (2017) carried out a tenfold cross-validation strategy.

For each downsampled subset, they performed stratified tenfold partitioning, that is, they divided their samples into ten groups of nearly equal sizes and with a balanced distribution of AR classes. They then trained their model using nine folds and used the fold left out for tests. The average of the 100 cross-validation iterations for each of the 100 downsampled data sets yielded their final results, as presented in Table 6.14:

Table 6.14: Results comparison with C. Liu et al. (2017)'s model.

Authorship	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*TSS*
C. Liu et al. (2017)	0.76	0.74	0.78	0.77	0.75	0.53
our RandomForest model	0.81	0.78	0.81	0.50	0.94	0.59

Our methodology produced a RandomForest model outperforming C. Liu et al. (2017)'s in the following aspects: ACC (+0.05), TPR (+0.04), TNR (+0.03), NPV (+0.19), and TSS (+0.06). On the other hand, despite not outperforming the remainder PPV, our model scored about a quarter less than the reference.

Noteworthily, our method managed to score most of the time the highest results using only four features – out of the 13 total – lasting from our feature selection: the total unsigned current helicity, the absolute value of the net current helicity, the sum of the modulus of the net current per polarity, and the sum of the flux near the polarity inversion line. In comparison, C. Liu et al. (2017) scored their results with the full set of them.

Finally, it is worth saying that C. Liu et al. (2017) did not explicitly reserve test sets in the beginning. Instead, by undersampling before the cross-validations, they created synthetic subsets not fully representing their samples' original composition (i. e., due to the discarding nature at random of their undersampling approach, they may have missed important records of the original dataset in the subsets used for evaluations). In comparison, our method achieved the results mentioned earlier while forecasting samples never saw during its model's design, not to say also holding a representative nature concerning the original dataset (i. e., through the stratified random reservation of test sets).

6.4 Concluding remarks

This chapter discussed the performance of our optimized models while forecasting ≥ C (Case Study I) and ≥ M (Case Study II and III) flare events up to three days ahead. Overall, designed classifiers' results showed agreement with the best performing forecast approaches in the literature.

For instance, for ≥ C events in the next 24 h, whereas the literature ranged on $0.50 < \text{TSS} \leq 0.80$, our methodology output a model scoring a test TSS = 0.69. On the other hand, for ≥ M events – also in the next 24 h –, whereas the literature ranged on $0.53 \leq \text{TSS} \leq 0.83$, our model scored a test TSS = 0.53.

Besides, results from our models' test AUCs have confirmed the usefulness of their positive forecasts – TPR = 0.86 and 0.83 respectively for ≥ C and ≥ M flares – as they reached 0.93 and 0.84 respectively for ≥ C and ≥ M flares. When directly compared to the literature of C. Liu et al. (2017), by using fewer resources and notably forecasting on a more challenging scenario, our optimized model outperformed the reference performance in the following aspects: ACC (+0.05), TPR (+0.04), TNR (+0.03), NPV (+0.19), and TSS (+0.06).

Chapter 7

Conclusions

Aiming at proposing some standardization and flexibility to support the design of flare forecasting systems, we proposed a novel methodology to cope with most of the aspects with which the literature is concerned while developing such models. To validate our methodology, we assembled a dataset based on daily aggregated solar data provided by the NOAA/SWPC, including the measures for the radio flux, x-ray background flux, sunspot number and area, and the WMFR of magnetic and McIntosh (1990)'s classes. We employed such data in two case studies for ≥ C- and ≥ M-class events up to 3 days ahead, namely forecasting them in the next 24, 48, and 72 h. Besides, we designed a third case study with C. Liu et al. (2017)'s dataset for ≥ M flares within 24 h, focusing on a direct comparison to other literature approaches.

One of the themes that emerged from this research involved the aspects constituting an optimized design process for flare classifiers. Anchored in some good practices already discussed in the literature and a reasonable number of observations from related proposals in the last ten years, we proposed a comprehensive pipeline of machine learning methods to use in such an optimized process, that is, the one not incurring in the negative design issues authors usually encompassed with their systems. Our pipeline comprehended:

i. Searching for distinct learning algorithms from a custom and flexible set (model selection).

ii. A selection of only relevant features (feature selection).

iii. Adjusting the behavior of algorithms (hyperparameter optimization).

iv. Providing proper unbiased data splitting (training, validation, and test sets).

v. Accommodating evaluations not incurring in statistical flukes or due to chance (repeated and stratified cross-validations).

vi. Empowering a broad specialized performance score analysis (deterministic and probabilistic metrics).

vii. Carrying out operational evaluations of models (true unseen records).

viii. Providing methods for minimizing the variance on data while reducing the input bias of algorithms (adjust the bias-variance trade-off).

ix. Employing unbiased resampling of data (imbalanced class ratios treatment).

x. Adjusting the cost function of classifiers.

xi. Adjusting the prediction threshold (configuring the costs from error types).

Aware of the benefits of automating machine learning classifiers' design and the absence of such tools for flare forecasting, we automated the methodology mentioned earlier under Python's Scikit-learn framework. The findings with our case studies showed that our methodology was able to produce models with scores consistent with previous research.

For instance, for forecasting ≥ C-class events within the next 24 h, our model scored a test TSS = 0.69, as high as the approaches reaching the highest results of this metric ($0.50 < \text{TSS} \leq 0.80$). By reaching this score level, other related metrics, such as the TPR = 0.86 and TNR = 0.82, also consequently increased in line with the literature (i. e., $0.78 \leq \text{TNR} \leq 0.86$ and $0.69 \leq \text{TPR} \leq 0.94$). Besides, as our model scored a high ACC = 0.85 – also agreeing with the best performing models from literature ($0.75 \leq \text{ACC} \leq 0.91$) – we did not notice any over-fitting. For longer forecasting horizons, we found no similar unbiased approach as a reference. However, our methodology produced models with scores as high as those for the next 24 h: TSS = 0.70 and 0.69 for the next 48 and 72 h, respectively.

Our results also agreed with the literature for ≥ M flares in the next 24 h. Accordingly, our test TSS = 0.53 has pushed its corresponding TPR = 0.83 and TNR = 0.70 to higher levels – the best performing models in literature scored $0.53 \leq \text{TSS} \leq 0.83$, $0.70 \leq \text{TPR} \leq 0.95$, and $0.70 \leq \text{TNR} \leq 0.86$, thus avoiding over-fitting with $0.75 \leq \text{ACC} \leq 0.86$. However, all those models – including ours – did somehow forecast false alarms, namely scoring FARs ≥ 0.63. For longer forecasting horizons, the methodology optimized models with scores as high as for

the next 24 h: TSS = 0.55 and 0.53 for the next 48 and 72 h, respectively (we found no similar unbiased approach as a reference).

The summarized referenced performance mentioned earlier has been taken as indirect comparisons as we could not surely assume whether the observed underlying differences in performance could be associate with classifiers' merit. On the other hand, when directly compared to the literature (i. e., by using the same dataset of a reference), our optimized model outperformed the reference performance (i. e., the TSS, TPR, and TNR of C. Liu et al. (2017)) using fewer resources (i. e., a reduced set of features) and notably forecasting on a more challenging scenario (that is, true unseen data).

Although the overview of results mentioned earlier was generally compatible with the literature, we argue that the findings with the performance of our case studies should not be taken as absolute references for future optimized models. Accordingly, one should not expect the same magnitude of increases/decreases when using the proposed methodology within other forecast scenarios. The effects on scores may depend on distinct aspects, such as the type of learning algorithm, data input features, hyperparameter set, optimized scores, among others.

Noteworthily, this research has coded its proposed automated methodology on the Scikit-learn framework. It means that if, on the one hand, the methodology can only expand itself under the premises of such platform (i. e., it restricts to what the platform offers), on the other hand, the Scikit-learn framework has a very comprehensive collection of capabilities[1].

For instance, besides the tree-based ensembles, the Scitkit-learn framework currently offers a broad range of learning algorithms, such as SVMs, multi-layer perceptron neural networks, linear classifiers (i. e., logistic regressors, LASSO, multi-task LASSO, elastic-nets etc.), linear discriminant analysis, k-NN, naïve classifiers (for instance, Gaussian Naïve Bayes, Bernoulli Naïve Bayes etc.), general basis ensembles (e. g., stacking the available learning algorithms through hard- or soft-voting), among others. If needed, one can include all those learning algorithms during the model selection.

Not only the learning algorithms but also several other aspects can be customized within our methodology (once more, some of them are directly related to the Scikit-learn's related capabilities). The list comprehending the methodology custom aspects includes:

i. Event type or magnitude: we set the event definition to represent ≥ C- and ≥ M-class events in our case studies. However, depending on the input data design, one can design

[1] https://scikit-learn.org/stable/user_guide.html.

the target feature to accommodate multi-class flare forecasts or other flare thresholds. This aspect grants to the methodology a broader application in solar weather research.

ii. Feature selection method: although we based our feature selection method in the case studies on a univariate schema provided with the F-score, one can add other algorithms in the methodology pipeline, such as the Pearson correlation analysis, variance thresholding, recursive feature elimination, and tree-based feature selection using impurity-based feature importance. The Scikit-learn framework currently offers all those methods.

iii. Inner validation scores: the TSS and AUC should not be considered the only available options for optimizing through methodology inner processes. The cross-cutting decisions between inner processes can comprise the boosting of several other performance scores, such as these directly offered in the Scikit-learn framework: accuracy, balanced accuracy, F1-score, recall and precision, among others. Whether the available scores are not enough, one can provide the calculation of its performance score within the methodology pipeline as it provides a custom capability for processing user-defined metrics[2].

iv. Hyperparameter optimization method: the random-based approach for tuning the hyperparameters opted in the case studies can be changed to the grid-based method. The Scikit-learn framework offers both methods.

v. Resampling method: the methodology does not restrict itself to those three SMOTE variations of our case studies for resampling data. As such, it currently borrows the capabilities of Python's imbalanced-learn toolkit[3] for correcting data skew. Such a library currently offers 11 distinct under-sampling methods (e. g., TomekLinks, OneSidedSelection, InstanceHardnessThreshold, RandomUnderSampler, among others), seven over-sampling approaches (for instance, Adasyn, RandomOverSampler, KMeans-SMOTE, among others), two combining methods for under- and over-sampling (i. e., both SMOTE-ENN and SMOTE-Tomek used in the case studies), and six ensemble resampling techniques (e. g., BalancedBaggingClassifier, BalancedRandomForest, among others) for correcting skewed class ratios. One can use all of them within the methodology pipeline.

[2]The Scikit-learn framework did not code TSS and HSS directly. However, we provided them as user-defined metrics in the pipeline.

[3]`https://imbalanced-learn.readthedocs.io/`.

vi. Cut-off point adjustment criterion: minimizing the absolute difference between recall and precision to adjust the classifiers' cut-off point solely related to our case studies. In fact, the methodology supports user-defined criteria to drive the prediction threshold adjustment, such as choosing the point with which some performance score peaked (BOBRA; COUVIDAT, 2015; NISHIZUKA; SUGIURA; KUBO; DEN; WATARI, et al., 2017).

vii. Cost function analysis decision criterion: the same reasoning from the previous item is valid here. Accordingly, minimizing the absolute difference between both recalls in the cost function analysis solely related to our case studies. At this point, the methodology also supports user-defined criteria, such as choosing the ratio with which some performance scores peaked.

It is worth mentioning that we believe our optimized models could support hybrid prediction schemes, such as the approach by the NOAA/SWPC, according to which an expert system estimates the predictions at first, and then human experts adjust them (CROWN, 2012). As Murray et al. (2017) argues, hybrid forecasts are helpful to reduce some climatology effects that certainly affect predictions made over longer forecasting horizons (for instance, which ARs may leave or return to the solar disk within the next few days or how the ARs evolve while crossing the disk).

However, before employing the optimized models into a hybrid forecast approach, one must think of assessing their real operational forecast performance, that is, how they would perform as they forecast NOAA/SWPC's daily assembled data in a real-time sense. In this context, we deployed the optimized experimental models from Case Study I and II into a real forecast environment providing daily forecasts at a fixed-cadence for up to three days ahead, namely the Guaraci system[4] (refer to Appendix C for further explanations on the Guaraci system).

Guaraci regularly connects to the NOAA/SWPC's database – always at 00:30 AM UTC – and dynamically calculates the features described in Chapter 5. Guaraci then uses such input to forecast ≥ C- and ≥ M-class events in the next 24, 48, and 72 h. As potential future research, we plan to study and assess our optimized models' true real-time forecast performance.

Besides assessing models' true operational performance, we can also think of other potential future research proposals. To date, our methodology lacks a comprehensive pipeline to tackle data preprocessing. Although there is an automated feature selection process, we did not envision other input data preprocessing methods, such as missing data treatment or data

[4]The Guaraci system is currently available at: `https://highpids.ft.unicamp.br/guaraci`.

standardization[5]. Automating a broad range of design processes grants the methodology to expand its capabilities, such as in Leka, Barnes, and Wagner (2018)'s research, whose authors reported automated missing data treatment in their design process.

In addition, we suggest adjusting our methodology to design interpretable models as future research. To date, our forecasts are designed through simple thresholded probabilities, as shown with Guaraci (Appendix C). However, as argued by Molnar (2020), interpretability in machine learning means the degree to which a human can understand the cause of decisions.

Accordingly, the higher the interpretability of a machine learning model, the easier it is for users to comprehend its decisions (i. e., why certain predictions have been made). This would be rather useful to cope with our methodology to improve the usability of designed models, especially for our end users, which are not always experts in machine learning.

For instance, as argued by Molnar (2020), machine learning predictions must employ contrastive explanations, that is, why positive forecasts have been made instead of the negative ones and vice versa (i. e., which features or phenomenon has been observed that most contributed/affected the forecasts). Other rules of thumb for designing interpretable models shall be sought in future research, such as including short sentences explaining the forecasts or the object of interest (MOLNAR, 2020).

Other interesting proposal for future work would be to design meta-learning (a research branch of automated machine learning) capabilities in our methodology. Vanschoren (2019) states that the concept of meta-learning comprehends the science of observing how different machine learning approaches perform on a wide range of learning tasks. Meta-learning algorithms then learn from this experience (meta-data) and suggest learning algorithms to use.

Accordingly, Vanschoren (2019) comments on several meta-features usually employed for meta-learning, such as the number of classes, missing values, features, and outliers. Depending on the values observed from those features, decision rules can be inferred to suggest which learning algorithms might outperform others in certain datasets. We believe that meta-learning would contribute to the ease of use of our methodology, as users would be released to define a set of learning algorithms manually.

Noteworthily, the domain of automated machine learning has emerged in the last years to support the design decisions through an automated, data-driven, and objective way. As we

[5]Despite treating missing feature values and standardizing our case studies' data, we processed our data set's records through *ad hoc* processes outside the methodology pipeline, which made data be inputted as is in the methodology.

could notice, users could simply provide their input data, and the automated machine learning process would fully determine the best performing forecast approach for that particular case.

Also, automated machine learning can be envisioned to provide state-of-the-art learning methods (i. e., deep learning) and design processes to researchers interested in applying concepts rather than knowing the technologies in their details. This scenario aims to automatically design forecast models with improved performance while saving a considerable amount of time and money, as experts in machine learning can sometimes be expensive or hard to find. Specifically in space weather research, automating machine learning can be rather valuable as not all solar physicists are experts in the artificial intelligence domain.

To contribute to this research's reproducibility, we made the methodology source code fully available to the community at GitHub[6]. We believe that opening our source code will let other researchers improve and use it with their forecast projects.

[6] `https://github.com/tiagocinto/guaraci-toolkit`.

References

AHMED, O. W. et al. Solar Flare Prediction Using Advanced Feature Extraction, Machine Learning, and Feature Selection. **Solar Physics**, v. 283, n. 1, p. 157–175, 2013. DOI: 10.1007/s11207-011-9896-1.

ALIPOUR, N.; MOHAMMADI, F.; SAFARI, H. Prediction of Flares within 10 Days before They Occur on the Sun. **The Astrophysical Journal Supplement Series**, v. 243, n. 20, 12pp, 2019. DOI: 10.3847/1538-4365/ab289b.

ANASTASIADIS, A. et al. Predicting Flares and Solar Energetic Particle Events: The FORSPEF Tool. **Solar Physics**, Springer Science+Business Media B.V., v. 292, n. 134, 21pp, 2017. DOI: 10.1007/s11207-017-1163-7.

BALA, R.; REIFF, P. Data Availability and Forecast Products for Space Weather. In: CAMPOREALE, E.; WING, S.; JOHNSON, J. (Eds.). **Machine Learning Techniques for Space Weather**. 1. ed. [S.l.]: Elsevier Inc., 2018. chap. 2, p. 27–41. DOI: 10.1016/B978-0-12-811788-0.00002-0.

BARNES, G.; LEKA, K. D. Evaluating the Performance of Solar Flare Forecasting Methods. **The Astrophysical Journal**, v. 688, n. 2, p. l107–l110, 2008. DOI: 10.1086/595550.

BARNES, G.; LEKA, K. D., et al. A Comparison of Flare Forecasting Methods. I. Results from the All-Clear Workshop. **The Astrophysical Journal**, v. 829, n. 89, 32pp, 2016. DOI: 10.3847/0004-637X/829/2/89.

BATISTA, G. E. A. P. A.; PRATI, R. C.; MONARD, M. C. A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. **Sigkdd Explorations**, v. 6, n. 1, p. 20–29, 2004. DOI: 10.1145/1007730.1007735.

BENVENUTO, F.; PIANA, M.; CAMPI, C.; MASSONE, A. M. A Hybrid Supervised/Unsupervised Machine Learning Approach to Solar Flare Prediction. **The Astrophysical Journal**, v. 853, n. 90, 9pp, 2018. DOI: 10.3847/1538-4357/aaa23c.

BLOOMFIELD, D. S.; HIGGINS, P. A.; MCATEER, R. T. J.; GALLAGHER, P. T. Toward Reliable Benchmarking of Solar Flare Forecasting Methods. **The Astrophysical Journal**, v. 747, p. l41, 2012. DOI: 10.1088/2041-8205/747/2/L41.

BOBRA, M. G.; COUVIDAT, S. Solar Flare Prediction Using SDO/HMI Vector Magnetic Field Data With a Machine-Learning Algorithm. **The Astrophysical Journal**, v. 798, n. 2, p. 135, 2015. DOI: 10.1088/0004-637X/798/2/135.

BOBRA, M. G.; SUN, X., et al. The Helioseismic and Magnetic Imager (HMI) Vector Magnetic Field Pipeline: SHARPs – Space-Weather HMI Active Region Patches. **Solar Physics**, v. 289, n. 9, p. 3549–3578, 2014. DOI: 10.1007/s11207-014-0529-3.

BOUCHERON, L. E.; AL-GHRAIBAH, A.; MCATEER, R. T. J. Prediction of Solar Flare Size and Time-to-Flare Using Support Vector Machine Regression. **The Astrophysical Journal**, v. 812, n. 51, 11pp, 2015. DOI: 10.1088/0004-637X/812/1/51.

BREIMAN, L.; FRIEDMAN, J.; OLSHEN, R.; STONE, C. **Classification and Regression Trees**. 1st. Monterey, CA: Wadsworth, 1984.

CAMPOREALE, E. The Challenge of Machine Learning in Space Weather: Nowcasting and Forecasting. **Space Weather**, v. 17, p. 1166–1207, 2019. DOI: 10.1029/2018SW002061.

CANFIELD, R. C. Solar Active Regions. In: MURDIN, P. (Ed.). **Encyclopedia of the history of astronomy and astrophysics**. Bristol, UK: Institute of Physics Publishing, 2001. p. 1–6.

CHANG, Y.-W.; LIN, C.-J. Feature Ranking Using Linear SVM. In: JMLR: Workshop and Conference Proceedings. WCCI2008 Workshop on Causality. Hong Kong: PMLR, 2008. v. 3, p. 53–64.

CHAWLA, N. V.; BOWYER, K. W.; HALL, L. O.; KEGELMEYER, W. P. SMOTE: Synthetic Minority Over-Sampling Technique. **Journal of Artificial Intelligence Research**, v. 16, p. 321–357, 2002. DOI: 10.1613/jair.953.

CINTO, T.; GRADVOHL, A. L. S.; COELHO, G. P.; SILVA, A. E. A. da. A Framework for Designing and Evaluating Solar Flare Forecasting Systems. **Monthly Notices of the Royal Astronomical Society**, Oxford University Press, v. 495, p. 3332–3349, 2020. DOI: 10.1093/mnras/staa1257.

CINTO, T.; GRADVOHL, A. L. S.; COELHO, G. P.; SILVA, A. E. A. da. Solar Flare Forecasting Using Time Series and Extreme Gradient Boosting Ensembles. **Solar Physics**, Springer Nature B.V., v. 295, n. 93, 30pp, 2020. DOI: 10.1007/s11207-020-01661-9.

CLAESEN, M.; DE MOOR, B. Hyperparameter Search in Machine Learning. **CoRR**, abs/1502.0, 2015. arXiv: 1502.02127. Available from: <http://arxiv.org/abs/1502.02127>.

COLAK, T.; QAHWAJI, R. Automated Prediction of Solar Flares Using Neural Networks and Sunspots Associations. In: SOFT Computing in Industrial Applications. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. v. 39. p. 316–324. DOI: 10.1007/978-3-540-70706-6_29.

COLAK, T.; QAHWAJI, R. Automated Solar Activity Prediction: A Hybrid Computer Platform Using Machine Learning and Solar Imaging for Automated Prediction of Solar Flares. **Space Weather**, v. 7, p. 1–12, 2009. DOI: 10.1029/2008SW000401.

CROWN, M. D. Validation of the NOAA Space Weather Prediction Center's Solar Flare Forecasting Look-up Table and Forecaster-issued Probabilities. **Space Weather**, v. 10, S06006, 4pp, 2012. DOI: 10.1029/2011SW000760.

DEVOS, A.; VERBEECK, C.; ROBBRECHT, E. Verification of Space Weather Forecasting at the Regional Warning Center in Belgium. **Journal of Space Weather and Space Climate**, v. 4, a29, 2014. DOI: 10.1051/swsc/2014025.

DÍSCOLA JR., S. L.; CECATTO, J. R.; FERNANDES, M. M.; RIBEIRO, M. X. An Optimized Data Mining Method to Support Solar Flare Forecast. In: LATIFI, S. (Ed.). **Information Technology – New Generations, Advances in Intelligent Systems and Computing**. Switzerland: Springer International Publishing, 2018. v. 558, p. 467–474. DOI: 10.1007/978-3-319-54978-1_60.

DÍSCOLA JR., S. L.; CECATTO, J. R.; FERNANDES, M. M.; RIBEIRO, M. X. Handling Imbalanced Time Series Through Ensemble of Classifiers: A Multi-class Approach for Solar Flare Forecasting. In: LATIFI, S. (Ed.). **16th International Conference on Information Technology-New Generations**. Switzerland: Springer Nature, 2019. DOI: 10.1007/978-3-030-14070-0_29.

DÍSCOLA JR., S. L.; CECATTO, J. R.; FERNANDES, M. M.; RIBEIRO, M. X. SeMiner: A flexible sequence miner method to forecast solar time series. **Information**, v. 9, n. 8, 23pp, 2018. ISSN 20782489. DOI: 10.3390/info9010008.

DOMIJAN, K.; BLOOMFIELD, D. S.; PITIÉ, F. Solar Flare Forecasting from Magnetic Feature Properties Generated by the Solar Monitor Active Region Tracker. **Solar Physics**, v. 294, n. 6, 2019. DOI: 10.1007/s11207-018-1392-4.

ELKAN, C. The Foundations of Cost-Sensitive Learning. In: PROCEEDINGS of the 17th International Joint Conference on Artificial Intelligence. San Francisco: Morgan Kaufmann Publishers Inc., 2001. p. 973–978.

ENGELL, A. J. et al. SPRINTS: A Framework for Solar-Driven Event Forecasting and Research. **Space Weather**, v. 15, n. 10, p. 1321–1346, 2017. ISSN 15427390. DOI: 10.1002/2017SW001660.

EREN, S. et al. Flare-Production Potential Associated with Different Sunspot Groups. **Monthly Notices of the Royal Astronomical Society**, v. 465, n. 1, p. 68–75, 2017. DOI: 10.1093/mnras/stw2742.

ERICKSON, N. et al. AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data. **arXiv (stat)**, 28pp, 2020. arXiv: 2003.06505. Available from: <http://arxiv.org/abs/2003.06505>.

FALCONER, D.; BARGHOUTY, A. F.; KHAZANOV, I.; MOORE, R. A Tool for Empirical Forecasting of Major Flares, Coronal Mass Ejections, and Solar Particle Events from a Proxy of Active-Region Free Magnetic Energy. **Space Weather**, v. 9, S04003, 12pp, 2011. DOI: 10.1029/2009SW000537.

FALCONER, D. A.; MOORE, R. L.; BARGHOUTY, A. F.; KHAZANOV, I. MAG4 Versus Alternative Techniques for Forecasting Active Region Flare Productivity. **Space Weather**, v. 12, n. 5, p. 306–317, 2014. DOI: 10.1002/2013SW001024.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. From Data Mining to Knowledge Discovery in Databases. **AI Magazine**, v. 17, n. 3, p. 37–54, 1996. DOI: 10.1609/aimag.v17i3.1230.

FEURER, M.; HUTTER, F. Hyperparameter Optimization. In: HUTTER, F.; KOTTHOF, L.; VANSCHOREN, J. (Eds.). **Automated Machine Learning: Methods, Systems, Challenges**. 1st. Cham, Switzerland: The Springer Series on Challenges in Machine Learning, Springer, 2019. chap. 1, p. 3–33. DOI: 10.1007/978-3-030-05318-5_1.

FEURER, M.; KLEIN, A., et al. Auto-sklearn: Efficient and Robust Automated Machine Learning. In: HUTTER, F.; KOTTHOF, L.; VANSCHOREN, J. (Eds.). **Automated Machine Learning: Methods, Systems, Challenges**. 1st. Cham, Switzerland: The Springer Series on Challenges in Machine Learning, Springer, 2019. chap. 6, p. 113–134. DOI: 10.1007/978-3-030-05318-5_6.

FLORIOS, K. et al. Forecasting Solar Flares Using Magnetogram-based Predictors and Machine Learning. **Solar Physics**, v. 293, n. 28, 42pp, 2018. DOI: 10.1007/s11207-018-1250-4.

GALLAGHER, P.; MOON, Y.-J.; WANG, H. Active Region Monitoring and Flare Forecast. **Solar Physics**, v. 209, p. 171–183, 2002. DOI: 10.1023/A:1020950221179.

GARCIÁ-RIGO, A. et al. Prediction and Warning System of SEP Events and Solar Flares for Risk Estimation in Space Launch Operations. **Journal of Space Weather and Space Climate**, v. 6, a28, 2016. DOI: 10.1051/swsc/2016021.

AL-GHRAIBAH, A.; BOUCHERON, L. E.; MCATEER, R. T. J. An Automated Classification Approach to Ranking Photospheric Proxies of Magnetic Energy Build-Up. **Astronomy & Astrophysics**, v. 579, a64, 2015. DOI: 10.1051/0004-6361/201525978.

GUERRA, J. A.; PULKKINEN, A.; URITSKY, V. M. Ensemble Forecasting of Major Solar Flares: First Results. **Space Weather**, v. 13, p. 626–642, 2015. DOI: 10.1002/2015SW001195.

GUYON, I.; ELISSEEFF, A. An Introduction To Variable and Feauture Selection. **Journal of Machine Learning Research**, v. 3, p. 1157–1182, 2003. DOI: 10.5555/944919.944968.

HADA-MURANUSHI, Y. et al. A Deep-Learning Approach for Operation of an Automated Realtime Flare Forecast. abs/1606.0, 6pp, 2016. arXiv: 1606.01587. Available from: <http://arxiv.org/abs/1606.01587>.

HALE, G. E.; ELLERMAN, F.; NICHOLSON, S. B.; JOY, A. H. The Magnetic Polarity of Sun-Spots. **Astrophysical Journal**, v. 49, p. 153, 1919. DOI: 10.1086/142452.

HAN, J.; KAMBER, M. **Data Mining: Concepts and Techniques**. 2. ed. San Francisco, USA: Morgan Kaufmann Publishers, 2006.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The Elements of Statistical Learning - Data Mining, Inference, and Prediction**. 2nd. New York, NY, USA: Springer, 2009.

HUANG, X.; WANG, H.; DAI, X. Influences of Misprediction Costs on Solar Flare Prediction. **Science China: Physics, Mechanics and Astronomy**, v. 55, n. 10, p. 1956–1962, 2012. DOI: 10.1007/s11433-012-4878-3.

HUANG, X.; WANG, H.; XU, L., et al. Deep Learning Based Solar Flare Forecasting Model. I. Results for Line-of-Sight Magnetograms. **The Astrophysical Journal**, v. 856, n. 1, p. 7, 2018. DOI: 10.3847/1538-4357/aaae00.

HUANG, X.; WANG, H.-N. Solar Flare Prediction Using Highly Stressed Longitudinal Magnetic Field Parameters. **Research in Astronomy and Astrophysics**, v. 13, n. 3, p. 351–358, 2013. DOI: 10.1088/1674-4527/13/3/010.

HUANG, X.; YU, D., et al. Short-Term Solar Flare Prediction Using Predictor Teams. **Solar Physics**, v. 263, n. 1, p. 175–184, 2010. DOI: 10.1007/s11207-010-9542-3.

HUTTER, F.; KOTTHOF, L.; VANSCHOREN, J. Preface. In: HUTTER, F.; KOTTHOF, L.; VANSCHOREN, J. (Eds.). **Automated Machine Learning: Methods, Systems, Challenges**. 1st. Cham, Switzerland: The Springer Series on Challenges in Machine Learning, Springer, 2019. Preface. DOI: 10.1007/978-3-030-05318-5.

INCEOGLU, F. et al. Using Machine Learning Methods to Forecast if Solar Flares Will Be Associated with CMEs and SEPs. **The Astrophysical Journal**, v. 861, n. 128, 10pp, 2018. DOI: 10.3847/1538-4357/aac81e.

INPE. **Programa de Estudo e Monitoramento Brasileiro de Clima Espacial (EMBRACE)**. São José dos Campos, 2020.

JAEGGLI, S. A.; NORTON, A. A. The Magnetic Classification of Solar Active Regions 1992 - 2015. **The Astrophysical Journal Letters**, v. 820, n. L11, 4pp, 2016. DOI: 10.3847/2041-8205/820/1/L11.

JAMES, G.; WITTEN, D.; HASTIE, T.; TIBSHIRANI, R. **An Introduction to Statistical Learning**. 1st. New York, NY, USA: Springer, 2013.

JIAO, Z. et al. Solar Flare Intensity Prediction With Machine Learning Models. **Space Weather**, v. 18, 2020. ISSN 15427390. DOI: 10.1029/2020SW002440.

JIN, H.; SONG, Q.; HU, X. Auto-keras: An Efficient Neural Architecture Search System. In: PROCEEDINGS of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. Anchorage, AK, USA: ACM, 2018. p. 1946–1956. DOI: 10.1145/3292500.3330648.

JOLLIFFE, I. T.; STEPHENSON, D. B. **Forecast Verification - A Practitioner's Guide in Atmospheric Science**. 1st. Chichester, England: John Wiley & Sons, 2003.

JONAS, R. et al. Flare Prediction Using Photospheric and Coronal Image Data. **Solar Physics**, v. 293, p. 48, 2018. DOI: 10.1007/s11207-018-1258-9.

KILCIK, A. et al. The Evolution of Flaring and Non-Flaring Active Regions. **Monthly Notices of the Royal Astronomical Society**, v. 477, n. 1, p. 293–297, 2018. DOI: 10.1093/mnras/sty388.

KLEIN, A.; FALKNER, S.; MANSUR, N.; HUTTER, F. RoBO: A Flexible and Robust Bayesian Optimization Framework in Python. In: PROCEEDINGS of the 31st Conference on Neural Information Processing Systems (NIPS). Long Beach, CA, USA: [s.n.], 2017. p. 1–5.

KOMER, B.; BERGSTRA, J.; ELIASMITH, C. Hyperopt-Sklearn. In: HUTTER, F.; KOTTHOF, L.; VANSCHOREN, J. (Eds.). **Automated Machine Learning: Methods, Systems, Challenges**. 1st. Cham, Switzerland: The Springer Series on Challenges in Machine Learning, Springer, 2019. chap. 5, p. 97–111.

KOTTHOFF, L. et al. Auto-WEKA: Automatic Model Selection and Hyperparameter Optimization in WEKA. In: HUTTER, F.; KOTTHOF, L.; VANSCHOREN, J. (Eds.). **Automated Machine Learning: Methods, Systems, Challenges**. 1st. Cham, Switzerland: The Springer Series on Challenges in Machine Learning, Springer, 2019. chap. 4, p. 81–95. DOI: 10.1007/978-3-030-05318-5_4.

KUBO, Y.; DEN, M.; ISHII, M. Verification of Operational Solar Flare Forecast: Case of Regional Warning Center Japan. **Journal of Space Weather and Space Climate**, v. 7, a20, 2017. DOI: 10.1051/swsc/2017018.

KUHN, M.; JOHNSON, K. **Applied Predictive Modeling**. 1st. New York, NY, USA: Springer, 2013.

LAN, R.-S.; JIANG, Y.; DING, L.-G.; YANG, J.-W. Automated Flare Prediction Using the AdaBoost Algorithm. **Research in Astronomy and Astrophysics**, v. 12, n. 9, p. 1191–1196, 2012. DOI: 10.1088/1674-4527/12/9/002.

LEDELL, E.; POIRIER, S. H2O AutoML: Scalable Automatic Machine Learning. In: PROCEEDINGS of the 7th ICML Workshop on Automated Machine Learning. Vienna, Austria: [s.n.], 2020. p. 1–16.

LEE, J.-Y. et al. Prediction of Daily Maximum X-Ray Flux Using Multilinear Regression and Autoregressive Time-Series Methods. **Journal of The Korean Astronomical Society**, v. 40, p. 99–106, 2007. DOI: 10.5303/JKAS.2007.40.4.099.

LEE, K. et al. Solar Flare Occurrence Rate and Probability in Terms of the Sunspot Classification Supplemented with Sunspot Area and Its Changes. **Solar Physics**, v. 281, p. 639–650, 2012. DOI: 10.1007/s11207-012-0091-9.

LEKA, K. D.; BARNES, G. Solar Flare Forecasting: Present Methods and Challenges. In: NATALIA, B. (Ed.). **Extreme Events in Geospace: Origins, Predictability, and Consequences**. 1st. [S.l.]: Elsevier Inc., 2018. chap. 3, p. 65–98. DOI: 10.1016/B978-0-12-812700-1.00003-0.

LEKA, K. D.; BARNES, G.; WAGNER, E. The NWRA Classification Infrastructure: Description and Extension to the Discriminant Analysis Flare Forecasting System (DAFFS). **Journal of Space Weather and Space Climate**, v. 8, a25, 2018. DOI: 10.1051/swsc/2018004.

LEKA, K. D.; PARK, S.-H., et al. A Comparison of Flare Forecasting Methods. II. Benchmarks, Metrics, and Performance Results for Operational Solar Flare Forecasting Systems. **The Astrophysical Journal Supplement Series**, v. 243, n. 36, 2019. DOI: 10.3847/1538-4365/ab2e12.

LEKA, K. D.; PARK, S.-H., et al. A Comparison of Flare Forecasting Methods. III. Systematic Behaviors of Operational Solar Flare Forecasting Systems. **The Astrophysical Journal**, v. 881, n. 101, 2019. DOI: 10.3847/1538-4357/ab2e11.

LI, R.; CUI, Y.; HE, H.; WANG, H. Application of Support Vector Machine Combined with K-nearest Neighbors in Solar Flare and Solar Proton Events Forecasting. **Advances in Space Research**, v. 42, n. 9, p. 1469–1474, 2008. DOI: 10.1016/j.asr.2007.12.015.

LI, R.; WANG, H.; CUI, Y.; HUANG, X. Solar Flare Forecasting Using Learning Vector Quantity and Unsupervised Clustering Techniques. **Science China Physics, Mechanics and Astronomy**, v. 54, n. 8, p. 1546–1552, 2011. DOI: 10.1007/s11433-011-4391-0.

LI, R.; ZHU, J. Solar Flare Forecasting Based on Sequential Sunspot Data. **Research in Astronomy and Astrophysics**, v. 13, n. 9, p. 1118–1126, 2013. DOI: 10.1088/1674-4527/13/9/010.

LI, X.; ZHENG, Y.; WANG, X.; WANG, L. Predicting Solar Flares Using a Novel Deep Convolutional Neural Network. **The Astrophysical Journal**, v. 891, n. 10, 11pp, 2020. DOI: 10.3847/1538-4357/ab6d04.

LIM, D. et al. Ensemble Forecasting of Major Solar Flares with Short-, Mid-, and Long-term Active Region Properties. **The Astrophysical Journal**, v. 885, n. 35, 9pp, 2019. DOI: 10.3847/1538-4357/ab45e7.

LINDAUER, M.; HUTTER, F.; HOOS, H. H.; SCHAUB, T. AutoFolio: An Automatically Configured Algorithm Selector. **Journal of Artificial Intelligence Research**, v. 53, p. 745–778, 2015. DOI: 10.1613/jair.4726.

LIU, C.; DENG, N.; WANG, J. T. L.; WANG, H. Predicting Solar Flares Using SDO /HMI Vector Magnetic Data Products and the Random Forest Algorithm. **The Astrophysical Journal**, v. 843, n. 2, p. 104, 2017. DOI: 10.3847/1538-4357/aa789b.

LIU, H.; LIU, C.; WANG, J. T. L.; WANG, H. Predicting Solar Flares Using a Long Short-term Memory Network. **The Astrophysical Journal**, v. 877, n. 121, 14pp, 2019. DOI: 10.3847/1538-4357/ab1b3c.

LIU, J.-F.; LI, F.; WAN, J.; YU, D.-R. Short-Term Solar Flare Prediction Using Multi-Model Integration Method. **Research in Astronomy and Astrophysics**, v. 17, n. 4, p. 034, 2017. DOI: 10.1088/1674-4527/17/4/34.

LIU, J.-F.; LI, F.; ZHANG, H.-P.; YU, D.-R. Short-Term Solar Flare Prediction Using Image-Case-Based Reasoning. **Research in Astronomy and Astrophysics**, v. 17, n. 11, p. 116, 2017. DOI: 10.1088/1674-4527/17/11/116.

MASON, J. P.; HOEKSEMA, J. T. Testing Automated Solar Flare Forecasting With 13 Years of Michelson Doppler Imager Magnetograms. **The Astrophysical Journal**, v. 723, n. 1, p. 634–640, 2010. DOI: 10.1088/0004-637X/723/1/634.

MASSONE, A. M.; PIANA, M. Machine Learning for Flare Forecasting. In: CAMPOREALE, E.; WING, S.; JOHNSON, J. (Eds.). **Machine Learning Techniques for Space Weather**. 1. ed. [S.l.]: Elsevier Inc., 2018. chap. 14, p. 355–364. DOI: 10.1016/B978-0-12-811788-0.00014-7.

MCATEER, R. T. J.; GALLAGHER, P. T.; CONLON, P. A. Turbulence, Complexity, and Solar Flares. **Advances in Space Research**, v. 45, p. 1067–1074, 2010. DOI: 10.1016/j.asr.2009.08.026.

MCCLOSKEY, A. E.; GALLAGHER, P. T.; BLOOMFIELD, D. S. Flare Forecasting Using the Evolution of McIntosh Sunspot Classifications. **Journal of Space Weather and Space Climate**, v. 8, a34, 2018. DOI: 10.1051/swsc/2018022.

MCCLOSKEY, A. E.; GALLAGHER, P. T.; BLOOMFIELD, D. S. Flaring Rates and the Evolution of Sunspot Group McIntosh Classifications. **Solar Physics**, Springer Science+Business Media Dordrecht, v. 291, p. 1711–1738, 2016. DOI: 10.1007/s11207-016-0933-y.

MCINTOSH, P. S. The Classification of Sunspot Groups. **Solar Physics**, v. 125, p. 251–267, 1990. DOI: 10.1007/BF00158405.

MENDONZA, H. et al. Towards Automatically-Tuned Deep Neural Networks. In: HUTTER, F.; KOTTHOF, L.; VANSCHOREN, J. (Eds.). **Automated Machine Learning: Methods, Systems, Challenges**. 1st. Cham, Switzerland: The Springer Series on Challenges in Machine Learning, Springer, 2019. chap. 7, p. 135–149. DOI: 10.1007/978-3-030-05318-5_7.

MESSEROTTI, M. et al. Solar Weather Event Modelling and Prediction. **Space Science Reviews**, v. 147, p. 121–185, 2009. DOI: 10.1007/s11214-009-9574-x.

MILLER, R. Wolf - A Computer Expert System for Sunspot Classification and Solar-Flare Prediction. **The Royal Astronomical Society of Canada**, v. 82, n. 4, p. 191–203, 1988.

MOLDWIN, M. **An Introduction to Space Weather**. New York: Cambridge University Press, 2008.

MOLNAR, C. **Interpretable Machine Learning: A Guide for Making Black Box Models Explainable.** [S.l.: s.n.], 2020. Available from: <https://christophm.github.io/interpretable-ml-book/index.html>.

MURANUSHI, T. et al. UFCORIN: A Fully Automated Predictor of Solar Flares in GOES X-ray Flux. **Space Weather**, v. 13, n. 11, p. 778–796, 2015. DOI: 10.1002/2015SW001257.

MURRAY, S. A.; BINGHAM, S.; SHARPE, M.; JACKSON, D. R. Flare Forecasting at the Met Office Space Weather Operations Centre. **Space Weather**, v. 15, n. 4, p. 577–588, 2017. DOI: 10.1002/2016SW001579.

NISHIZUKA, N.; SUGIURA, K.; KUBO, Y.; DEN, M.; ISHII, M. Deep Flare Net (DeFN) Model for Solar Flare Prediction. **The Astrophysical Journal**, v. 858, n. 113, 8pp, 2018. DOI: 10.3847/1538-4357/aab9a7.

NISHIZUKA, N.; SUGIURA, K.; KUBO, Y.; DEN, M.; WATARI, S., et al. Solar Flare Prediction Model with Three Machine-Learning Algorithms Using Ultraviolet Brightening and Vector Magnetogram. **The Astrophysical Journal**, v. 835, n. 2, 156 (10pp), 2017. DOI: 10.3847/1538-4357/835/2/156.

NOAA. **Penticton/Ottawa 2800 MHz Solar Flux**. [S.l.: s.n.], 2020. Available from: <https://www.ngdc.noaa.gov/stp/solar/flux.html>.

NOAA/SWPC. **Daily Solar Indices Summaries**. [S.l.: s.n.], 2011. Available from: <ftp://ftp.swpc.noaa.gov/pub/indices/old_indices/README>.

NOAA/SWPC. **Sunspot Region Summary (SRS)**. [S.l.: s.n.], 2008. Available from: <ftp://ftp.swpc.noaa.gov/pub/forecasts/SRS/README>.

NRC, N. R. C. Severe Space Weather Events - Understanding Societal and Economic Impacts. In: COMMITTEE on the Societal and Economic Impacts of Severe Space Weather Events: A Workshop. Washington, DC: The National Academic Press, 2009. p. 1–131.

NUNEZ, M. Predicting Solar Energetic Proton Events (E > 10 MeV). **Space Weather**, v. 9, s07003 (28pp), 2011. DOI: 10.1029/2010SW000640.

OLSON, R. S.; MOORE, J. TPOT: A Tree-Based Pipeline Optimization Tool for Automating Machine Learning. In: HUTTER, F.; KOTTHOF, L.; VANSCHOREN, J. (Eds.). **Automated Machine Learning: Methods, Systems, Challenges**. 1st. Cham, Switzerland: The Springer Series on Challenges in Machine Learning, Springer, 2019. chap. 8, p. 151–160. DOI: 10.1007/978-3-030-05318-5_8.

PARK, E. et al. Application of the Deep Convolutional Neural Network to the Forecast of Solar Flare Occurrence Using Full-disk Solar Magnetograms. **The Astrophysical Journal**, v. 869, n. 91, 6pp, 2018. DOI: 10.3847/1538-4357/aaed40.

PARRY, P. **Auto_ml documentation**. [S.l.: s.n.], 2016. Available from: <https://auto-ml.readthedocs.io/en/latest/>.

PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011. DOI: 10.5555/1953048.2078195.

PROBST, P.; WRIGHT, M. N.; BOULESTEIX, A.-L. Hyperparameters and Tuning Strategies for Random Forest. **Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery**, e1301, p. 1–15, 2018. DOI: 10.1002/widm.1301.

PYLE, D. **Data Preparation for Data Mining**. San Francisco, USA: Morgan Kaufmann Publishers, 1999.

QAHWAJI, R.; COLAK, T. Automatic Short-Term Solar Flare Prediction Using Machine Learning and Sunspot Associations. **Solar Physics**, v. 241, p. 195–211, 2007. DOI: 10.1007/s11207-006-0272-5.

QAHWAJI, R.; COLAK, T. Neural Network-Based Prediction of Solar Activities. In: PROCEEDINGS of the 3rd International Conference on Cybernatics and Information Technologies. Florida, US: [s.n.], 2006.

RABOONIK, A.; SAFARI, H.; ALIPOUR, N.; WHEATLAND, M. S. Prediction of Solar Flares Using Unique Signatures of Magnetic Field Images. **The Astrophysical Journal**, v. 834, n. 1, 11 (8pp), 2016. DOI: 10.3847/1538-4357/834/1/11.

SADYKOV, V. M.; KOSOVICHEV, A. G. Relationships Between Characteristics of the Line-of-Sight Magnetic Field and Solar Flare Forecasts. **The Astrophysical Journal**, v. 849, n. 2, p. 148, 2017. DOI: 10.3847/1538-4357/aa9119.

SCHERRER, P. H. et al. The Solar Oscillations Investigation - Michelson Doppler Imager. **Solar Physics**, v. 162, n. 1-2, p. 129–188, 1995. DOI: 10.1007/BF00733429.

SHIN, S. et al. Development of Daily Maximum Flare-Flux Forecast Models for Strong Solar Flares. **Solar Physics**, v. 291, n. 3, p. 897–909, 2016. DOI: 10.1007/s11207-016-0869-2.

SHORTEN, C.; KHOSHGOFTAAR, T. M. A Survey on Image Data Augmentation for Deep Learning. **Journal of Big Data**, Springer International Publishing, v. 6, n. 60, 48pp, 2019. DOI: 10.1186/s40537-019-0197-0.

SONG, H. et al. Statistical Assessment of Photospheric Magnetic Features in Imminent Solar Flare Predictions. **Solar Physics**, v. 254, n. 1, p. 101–125, 2009. DOI: 10.1007/s11207-008-9288-3.

THOMAS, J.; COORS, S.; BISCHL, B. Automatic Gradient Boosting. **arXiv (stat)**, July 2018. arXiv: 1807.03873. Available from: <http://arxiv.org/abs/1807.03873>.

VANSCHOREN, J. Meta-Learning. In: HUTTER, F.; KOTTHOF, L.; VANSCHOREN, J. (Eds.). **Automated Machine Learning: Methods, Systems, Challenges**. 1st. Cham, Switzerland: The Springer Series on Challenges in Machine Learning, Springer, 2019. chap. 2, p. 39–68. DOI: 10.1007/978-3-030-05318-5_2.

WANG, H. et al. Solar Flare Forecasting Model Supported with Artificial Neural Network Techniques. **Advances in Space Research**, v. 42, p. 1464–1468, 2008. DOI: 10.1016/j.asr.2007.06.070.

WANG, X. et al. Predicting Solar Flares with Machine Learning: Investigating Solar Cycle Dependence. **The Astrophysical Journal**, v. 895, n. 3, 13pp, 2020. DOI: 10.3847/1538-4357/ab89ac.

WHEATLAND, M. S. A Statistical Solar Flare Forecast Method. **Space Weather**, v. 3, S07003, 2005. DOI: 10.1029/2004SW000131.

WHITE, T. **Hadoop: The Definitive Guide**. 1st. [S.l.]: O'Reilly Media, Inc., 2009. ISBN 0596521979.

WILCOXON, F. Individual Comparisons by Ranking Methods. **Biometrics Bulletins**, v. 1, n. 6, p. 80–83, 1945. DOI: 10.2307/3001968.

WINTER, L. M.; BALASUBRAMANIAM, K. Using the Maximum X-ray Flux Ratio and X-ray Background to Predict Solar Flare Class. **Space Weather**, v. 13, n. 5, p. 286–297, 2015. DOI: 10.1002/2015SW001170.

WITTEN, I. H.; FRANK, E.; HALL, M. A. **Data Mining: Practical Machine Learning Tools and Techniques**. 3. ed. Burlington: Morgan Kaufmann Publishers, 2011.

YANG, X.; LIN, G.; ZHANG, H.; MAO, X. Magnetc Nonpotentiality in Photospheric Active Regions as a Predictor of Solar Flares. **The Astrophysical Journal**, v. 774, n. L27, 6pp, 2013. ISSN 2041-8205. DOI: 10.1088/2041-8205/774/2/L27.

YOUDEN, W. J. Index for Rating Diagnostic Tests. **Cancer**, v. 3, n. 1, p. 32–35, 1950. ISSN 10970142. DOI: 10.1002/1097-0142(1950)3:1<32::AID-CNCR2820030106>3.0.CO;2-3.

YU, D.; HUANG, X.; HU, Q., et al. Short-Term Solar Flare Prediction Using Multiresolution Predictors. **The Astrophysical Journal**, v. 709, p. 321–326, 2010. DOI: 10.1088/0004-637X/709/1/321.

YU, D.; HUANG, X.; WANG, H.; CUI, Y. Short-Term Solar Flare Prediction Using a Sequential Supervised Learning Method. **Solar Physics**, v. 255, p. 91–105, 2009. DOI: 10.1007/s11207-009-9318-9.

YU, D.; HUANG, X.; WANG, H.; CUI, Y., et al. Short-Term Solar Flare Level Prediction Using a Bayesian Network Approach. **The Astrophysical Journal**, v. 710, n. 1, p. 869–877, 2010. DOI: 10.1088/0004-637X/710/1/869.

YUAN, Y.; SHIH, F. Y.; JING, J.; WANG, H.-M. Automated Flare Forecasting Using a Statistical Learning Technique. **Research in Astronomy and Astrophysics**, v. 10, n. 8, p. 785–796, 2010. DOI: 10.1088/1674-4527/10/8/008.

ZAKI, M. J.; MEIRA JR., W. **Data Mining and Analysis: Fundamental Concepts and Algorithms**. New York: Cambridge University Press, 2013.

ZHANG, C.; BI, J.; SODA, P. Feature Selection and Resampling in Class Imbalance Learning: Which Comes First? An Empirical Study in the Biological Domain. In: PROCEEDINGS of the 2017 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2017. Kansas, USA: IEEE, 2017. p. 933–938. DOI: 10.1109/BIBM.2017.8217782.

ZHANG, X.; LIU, J.; WANG, Q. Image Feature Extraction for Solar Flare Prediction. In: QIU, P. et al. (Eds.). **4th International Congress on Image and Signal Processing**. [S.l.: s.n.], 2011. p. 910–914.

ZHENG, Y.; LI, X.; WANG, X. Solar Flare Prediction with the Hybrid Deep Convolutional Neural Network. **The Astrophysical Journal**, v. 885, n. 73, 14pp, 2019. DOI: 10.3847/1538-4357/ab46bd.

ZIMMER, L.; LINDAUER, M.; HUTTER, F. Auto-PyTorch Tabular: Multi-Fidelity MetaLearning for Efficient and Robust AutoDL. **arXiv (cs)**, 15pp, 2020. arXiv: 2006.13799. Available from: <http://arxiv.org/abs/2006.13799>.

Appendix A

Performance assessment

In this thesis, we used some well-defined scores to guide the proposed framework during its decision-making process. This appendix shall describe all of them, namely the PPV (ZAKI; MEIRA JR., 2013), NPV (ZAKI; MEIRA JR., 2013), ACC (HAN; KAMBER, 2006), TPR (HAN; KAMBER, 2006), TNR (HAN; KAMBER, 2006), FAR (JOLLIFFE; STEPHENSON, 2003), AUC (WITTEN; FRANK; HALL, 2011), FPR (WITTEN; FRANK; HALL, 2011), HSS (JOLLIFFE; STEPHENSON, 2003), and TSS (JOLLIFFE; STEPHENSON, 2003; YOUDEN, 1950).

To calculate those scores, we employed confusion matrices. We use such representations to observe classifiers' performance regarding their outcomes (i. e., correct vs. incorrect forecasts). To n-class based problems, the confusion matrices have the form of $n \times n$ – in our case, several 2×2 matrices. Cells $[i, j]$ of a confusion matrix refer to samples of class i classified as that of class j (Table A.1). Accordingly, we define their cells as follows:

- TP: positive samples predicted as positive;
- TN: negative samples predicted as negative;
- FP: negative samples incorrectly classified;
- FN: positive samples incorrectly classified.

Table A.1: Confusion matrix.

		Predicted Class	
		$C_{positive}$	$C_{negative}$
Correct Class	$C_{positive}$	True Positives (TP)	False Negatives (FN)
	$C_{negative}$	False Positives (FP)	True Negatives (TN)

A.1 Positive predictive value

Also known as the classifier's class-specific accuracy scores, both PPV and NPV refer to the classifier precision scores regarding the individual outcome classes. The positive precision measures the fraction of true positives over all samples predicted as positive (true and false positives). We measure PPV on the scale [0 , 1], in which the higher the values, the better the classifier. In Equation A.1, we show how to calculate PPV (ZAKI; MEIRA JR., 2013):

$$\text{PPV} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{A.1}$$

where TP and FP are the cells of a confusion matrix.

A.2 Negative predictive value

The negative precision, in turn, measures the fraction of true negatives over all samples predicted as negative (true and false negatives). Similarly to the PPV, NPV also ranges on [0 , 1] (once more, the higher this score, the better the classifier). In Equation A.2, we show how to calculate NPV (ZAKI; MEIRA JR., 2013):

$$\text{NPV} = \frac{\text{TN}}{\text{TN} + \text{FN}}, \tag{A.2}$$

where TN and FN are the cells of a confusion matrix.

A.3 Accuracy

The overall accuracy is a global score for representing classifiers' performance concerning the fraction of true positives and negatives over all predictions made. In this sense, ACC also lies around [0 , 1] (the higher the score, the better the classifier). In fact, this rank corresponds to the weighted mean between the class-specific accuracy scores mentioned earlier, as we show in Equation A.3 (HAN; KAMBER, 2006):

$$\text{ACC} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{FP} + \text{TN}}, \tag{A.3}$$

where TP, TN, FN, and FP are the cells of a confusion matrix.

Noteworthily, in imbalanced class ratio scenarios, ACC cannot be used by itself, since it would certainly mask the real hit rates of individual classes. For instance, an elevated ACC may suggest a quite accurate classifier. However, this would certainly harm the interpretation of this score, whether only a few portions of training samples are from the positive class, i. e., we cannot distinguish how well our classifier recognizes the positive or negative class. Still, we decided to keep such rank in our study for completeness purposes, since most authors use it.

A.4 True positive rate

Also defined as the positive recall, TPR corresponds to the fraction of true positives over all samples of the positive class (true positives and false negatives). As the scores mentioned earlier, TPR also lies around [0 , 1], where higher values represent better classifiers. In Equation A.4, we show how to calculate TPR (HAN; KAMBER, 2006):

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{A.4}$$

where TP and FN are the cells of a confusion matrix.

A.5 True negative rate

On the other hand, also defined as the negative recall, TNR corresponds to the fraction of true negatives over all samples of the negative class (true negatives and false positives). We measure and analyze TNR results the way as that of TPR, as we show in Equation A.5 (HAN; KAMBER, 2006):

$$\text{TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}}, \tag{A.5}$$

where TN and FP are the cells of a confusion matrix.

A.6 False alarm ratio

In turn, the false alarm ratio (or false positive ratio) accounts for the proportion of positive forecasts that are not followed by a true occurrence. Given that it has a complementary nature, we must analyze it in conjunction with the positive recall of classifiers to allow a better

understanding involving the quality of their TPR results. As scores mentioned earlier, FAR also ranges around [0 , 1]. However, the lower the score, the better the classifier. In Equation A.6, we show how to calculate FAR (JOLLIFFE; STEPHENSON, 2003):

$$\text{FAR} = \frac{\text{FP}}{\text{TP} + \text{FP}}, \tag{A.6}$$

where FP and TP are the cells of a confusion matrix.

A.7 False positive rate

Also known as the probability of a false detection (POFD), the FPR calculates the rate of false alarms among negative predictions, as we show in Equation A.7 (WITTEN; FRANK; HALL, 2011):

$$\text{FPR} = \frac{\text{FP}}{\text{TN} + \text{FP}}, \tag{A.7}$$

where FP and TN are the cells of a confusion matrix.

A.8 Area under the curve

The AUC measures the two-dimensional area underneath the ROC curve. Accordingly, it graphically analyzes the TPR scores (y-axis) vs. the FPRs (x-axis) for a set of increasing probability thresholds to make the yes/no decisions, i. e., 0.1, 0.2, 0.3, and so on (refer to Figure A.1 for an example).

Best classifiers score the AUC next to the graph left-hand corner (FPR = 0 and TPR = 1). On the other hand, the worst classifiers score next to the graph bottom right-hand corner (FPR = 1 and TPR = 0). The AUC is always positive and, ideally, should be greater than 0.5.

A.9 Heidke skill score

Conversely to the scores mentioned earlier for assessing individual classifiers' behaviors (PPV, NPV, TPR, TNR, FAR, and FPR) or their probabilistic skill (AUC), both HSS and TSS refer to generalized quality skill scores for them. Regarding HSS, it measures the fraction of correct forecasts after eliminating those forecasts, which would be correct due to random chance. As

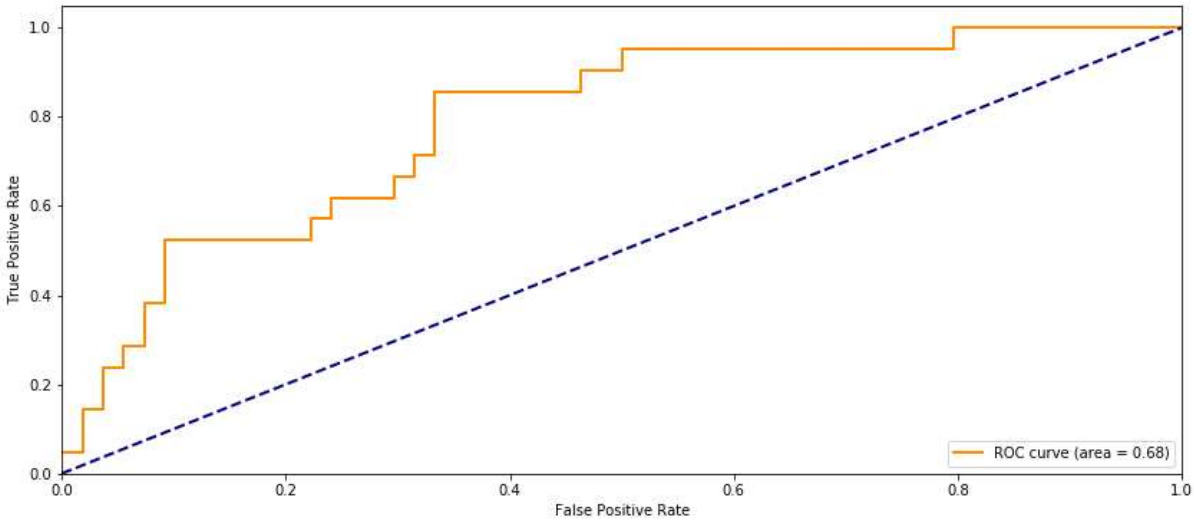

Figure A.1: Example of ROC curve graphical plot.

such, it assesses the improvement of forecasts over the random chance. For reporting purposes, HSS ranks classifiers over [−1 , 1], where results close to -1 mean all predictions incorrect (positive and negative) and close to 1 mean all predictions correct − zeroed results refer to classifiers' forecast skills equal to random chance. In Equation A.8, we show a simplified form to calculate HSS (JOLLIFFE; STEPHENSON, 2003):

$$\text{HSS} = \frac{2 \times [(\text{TP} \times \text{TN}) - (\text{FN} \times \text{FP})]}{(\text{TP} + \text{FN}) \times (\text{FN} + \text{TN}) + (\text{TP} + \text{FP}) \times (\text{FP} + \text{TN})}, \tag{A.8}$$

where TP, TN, FN, and FP are the cells of a confusion matrix.

A.10 True skill statistics

Finally, as a generalized quality skill score similar to HSS, the TSS score also ranks a model performance over a scale lying on [−1 , 1]. Hence, results close to −1 mean all incorrect predictions (positive and negative), and those close to 1 mean all correct predictions − zeroed results refer to no-skilled classifiers. Conversely to HSS, TSS is not affected by imbalanced class ratios (BLOOMFIELD et al., 2012). In Equation A.9, we show how to calculate TSS (JOLLIFFE; STEPHENSON, 2003; YOUDEN, 1950):

$$\text{TSS} = \text{TPR} + \text{TNR} - 1, \tag{A.9}$$

where TPR and TNR are the corresponding scores mentioned earlier.

Appendix B

Detailed case study results

Chapter 6 discussed the results of carrying out the methodology as defined in Chapter 4 , thus presenting the performance of models designed to forecast ≥ C- (Case Study I) and M-class flares (case studies II and III) up to three days ahead.

Specifically for Case Study I, Chapter 6 dove deeper into the analysis and presented the detailed discussion of methodology's inner results for flare forecasting in the next 24 h, excluding this detailed text for longer forecasting horizons in such case and all horizons of others.

This appendix is a complement for Chapter 6 since it presents the tables of detailed results showing how the performance has changed between the methodology inner processes of the remainder forecasting horizons of Case Study I and other cases[1].

[1]The notation used in tables shown herein is further explained in Chapter 6, specifically in Page 105.

Table B.1: Case Study I, ≥ C flares, the next 48 h: model selection results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
AdaBoost/1	0.83	0.887	0.720	0.857	0.773	0.143	0.608	0.617	0.888
AdaBoost/2	0.829	0.885	0.723	0.858	0.769	0.142	0.607	0.616	0.891
AdaBoost/3	0.835	0.888	0.733	0.863	0.777	0.137	0.621	0.630	0.894
AdaBoost/4	0.829	0.884	0.723	0.858	0.768	0.142	0.608	0.616	0.892
AdaBoost/5	0.832	0.895	0.714	0.856	0.783	0.145	0.608	0.621	0.893
avg(AdaBoost)	0.83	0.89	0.72	0.86	0.77	0.14	0.61[a]	0.62	0.89
RandomForest/1	0.853	0.937	0.693	0.853	0.854	0.147	0.630	0.659	0.903
RandomForest/2	0.850	0.937	0.685	0.849	0.853	0.151	0.622	0.652	0.903
RandomForest/3	0.851	0.937	0.687	0.850	0.854	0.150	0.624	0.654	0.905
RandomForest/4	0.848	0.933	0.687	0.850	0.845	0.150	0.620	0.648	0.906
RandomForest/5	0.851	0.936	0.688	0.851	0.852	0.149	0.625	0.654	0.905
avg(RandomForest)	0.85	0.94	0.69	0.85	0.85	0.15	0.62[a]	0.65	0.90
GradientTreeBoosting/1	0.810	0.806	0.817	0.896	0.699	0.104	0.623	0.598	0.861
GradientTreeBoosting/2	0.811	0.837	0.762	0.872	0.718	0.128	0.599	0.589	0.838
GradientTreeBoosting/3	0.811	0.834	0.768	0.875	0.716	0.125	0.601	0.590	0.841
GradientTreeBoosting/4	0.822	0.850	0.769	0.876	0.735	0.124	0.619	0.611	0.850
GradientTreeBoosting/5	0.810	0.817	0.798	0.887	0.710	0.113	0.615	0.597	0.847
avg(GradientTreeBoosting)	0.81	0.83	0.78	0.88	0.72	0.12	0.61[a]	0.60	0.85

[a] $TSS_{RandomForest} > TSS_{AdaBoost}$ and $TSS_{GradientTreeBoosting}$ ($p < 0.05$). As the RandomForest output the highest TSS, the methodology has chosen it to proceed in the pipeline.

Table B.2: Case Study I, ≥ C flares, the next 48 h: feature selection results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
RandomForest/1	0.866	0.925	0.753	0.877	0.843	0.124	0.678	0.695	0.892
RandomForest/2	0.864	0.917	0.765	0.881	0.831	0.119	0.682	0.694	0.898
RandomForest/3	0.865	0.917	0.766	0.882	0.831	0.119	0.683	0.696	0.891
RandomForest/4	0.863	0.925	0.745	0.873	0.841	0.127	0.670	0.688	0.887
RandomForest/5	0.868	0.932	0.748	0.875	0.853	0.125	0.680	0.700	0.886
avg(RandomForest)	**0.87**[acc]	0.92[tpr]	**0.76**[tnr]	**0.88**[ppv]	0.84[npv]	**0.12**[far]	**0.68**[tss]	**0.69**[hss]	0.89[auc]

[acc] +0.02 ($p < 0.05$). [tnr] +0.07 ($p < 0.05$). [npv] -0.01 ($p < 0.1$). [tss] +0.06 ($p < 0.05$). [auc] -0.01 ($p < 0.05$).
[tpr] -0.02 ($p < 0.05$). [ppv] +0.03 ($p < 0.05$). [far] -0.03 ($p < 0.05$). [hss] +0.04 ($p < 0.05$).

Table B.3: Case Study I, ≥ C flares, the next 48 h: hyperparameter optimization results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
RandomForest/1	0.869	0.922	0.770	0.884	0.839	0.116	0.692	0.705	0.941
RandomForest/2	0.866	0.912	0.777	0.886	0.825	0.114	0.689	0.699	0.940
RandomForest/3	0.866	0.921	0.761	0.880	0.835	0.120	0.682	0.696	0.939
RandomForest/4	0.865	0.918	0.767	0.882	0.832	0.118	0.684	0.697	0.942
RandomForest/5	0.865	0.909	0.781	0.887	0.820	0.113	0.690	0.698	0.940
avg(RandomForest)	0.87	0.92	**0.77**[tnr]	0.88	0.83[npv]	0.12	**0.69**[tss]	0.70	**0.94**[auc]

[tnr] +0.01 ($p < 0.1$). [npv] -0.01 ($p > 0.05$ and 0.1). [tss] +0.01 ($p < 0.1$). [auc] +0.05 ($p < 0.05$).

Table B.4: Case Study I, ≥ C flares, the next 48 h: data resampling results.

Method/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
SMOTE/1	0.810	0.743	0.935	0.956	0.659	0.044	0.678	0.617	0.937
SMOTE/2	0.803	0.730	0.942	0.960	0.649	0.040	0.672	0.607	0.938
SMOTE/3	0.802	0.729	0.941	0.959	0.648	0.041	0.670	0.605	0.937
SMOTE/4	0.798	0.721	0.945	0.961	0.642	0.039	0.666	0.599	0.937
SMOTE/5	0.803	0.728	0.943	0.961	0.647	0.039	0.671	0.606	0.935
avg(SMOTE)	0.80	0.73[a]	0.94[a]	0.96	0.65	0.04	0.67	0.61	0.94
SMOTE-ENN/1	0.851	0.835	0.882	0.931	0.739	0.069	0.717	0.686	0.936
SMOTE-ENN/2	0.854	0.837	0.886	0.933	0.743	0.067	0.723	0.692	0.936
SMOTE-ENN/3	0.854	0.838	0.883	0.932	0.743	0.068	0.721	0.690	0.938
SMOTE-ENN/4	0.847	0.828	0.884	0.931	0.731	0.069	0.712	0.678	0.938
SMOTE-ENN/5	0.844	0.821	0.888	0.933	0.725	0.067	0.709	0.673	0.936
avg(SMOTE-ENN)	0.85	0.83[b]	0.88[b]	0.93	0.74	0.07	0.72	0.68	0.94
SMOTE-Tomek/1	0.858	0.849	0.875	0.928	0.755	0.072	0.724	0.698	0.940
SMOTE-Tomek/2	0.857	0.850	0.870	0.925	0.755	0.075	0.720	0.695	0.941
SMOTE-Tomek/3	0.857	0.855	0.861	0.921	0.759	0.079	0.716	0.694	0.939
SMOTE-Tomek/4	0.856	0.856	0.857	0.919	0.759	0.081	0.713	0.692	0.941
SMOTE-Tomek/5	0.849	0.841	0.864	0.922	0.742	0.079	0.705	0.678	0.938
avg(SMOTE-Tomek)	0.86	0.85[c]	0.87[c]	0.92	0.75	0.08	0.72	0.69	0.94

[a] $|TPR - TNR| = |0.73 - 0.94| = 0.21$. [b] $|TPR - TNR| = |0.83 - 0.88| = 0.05$. [c] $|TPR - TNR| = |0.85 - 0.87| = 0.02$.

Table B.5: Case Study I, ≥ C flares, the next 48 h: cost function analysis results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
RandomForest/1	0.859	0.852	0.874	0.927	0.758	0.073	0.725	0.700	0.940
RandomForest/2	0.850	0.843	0.862	0.921	0.744	0.079	0.705	0.680	0.939
RandomForest/3	0.858	0.858	0.856	0.919	0.762	0.081	0.715	0.694	0.941
RandomForest/4	0.857	0.856	0.861	0.921	0.759	0.079	0.716	0.694	0.940
RandomForest/5	0.856	0.848	0.870	0.925	0.752	0.075	0.718	0.692	0.941
avg(RandomForest)	0.86[acc]	0.85[a,tpr]	**0.86**[a,tnr]	**0.92**[ppv]	0.76[npv]	**0.08**[far]	**0.72**[tss]	0.69	0.94

[a] |TPR − TNR| = |0.85 − 0.86| = 0.01. As the cost function output the lowest difference, the methodology has chosen it to proceed in the pipeline.
[acc] -0.01 ($p < 0.05$) – compared to feature selection.
[tpr] -0.07 ($p < 0.05$) – compared to feature selection.
[tnr] +0.09 ($p < 0.05$).
[ppv] +0.04 ($p < 0.05$) – compared to feature selection.
[npv] -0.08 ($p < 0.05$) – compared to feature selection.
[far] -0.04 ($p < 0.05$) – compared to feature selection.
[tss] +0.03 ($p < 0.05$).

Table B.6: Case Study I, ≥ C flares, the next 48 h: cut-off point adjustment results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
RandomForest/1	0.864	0.895	0.806	0.897	0.803	0.103	0.701	0.700	0.941
RandomForest/2	0.863	0.902	0.789	0.890	0.811	0.110	0.692	0.696	0.939
RandomForest/3	0.866	0.894	0.815	0.901	0.802	0.099	0.709	0.706	0.940
RandomForest/4	0.867	0.904	0.798	0.895	0.814	0.105	0.702	0.705	0.941
RandomForest/5	0.865	0.888	0.820	0.903	0.796	0.097	0.708	0.703	0.940
avg(RandomForest)	**0.87**[acc]	**0.90**[tpr]	0.81[tnr]	0.90[ppv]	**0.81**[npv]	0.10[far]	0.70[tss]	**0.70**[hss]	0.94

[acc] +0.01 ($p < 0.05$).
[tpr] +0.05 ($p < 0.05$).
[tnr] -0.04 ($p < 0.05$).
[ppv] -0.02 ($p < 0.05$).
[npv] +0.05 ($p < 0.05$).
[far] +0.02 ($p < 0.05$).
[tss] -0.02 ($p < 0.05$).
[tss] +0.01 ($p < 0.05$).

Table B.7: Case Study I, ≥ C flares, the next 48 h: evaluation of validation sets.

Pred.Type/Val.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
BaselineRandomForest/1	0.850	0.935	0.690	0.851	0.848	0.149	0.625	0.654	0.887
BaselineRandomForest/2	0.848	0.931	0.690	0.850	0.840	0.150	0.621	0.648	0.922
BaselineRandomForest/3	0.868	0.958	0.698	0.857	0.899	0.143	0.657	0.693	0.927
BaselineRandomForest/4	0.848	0.925	0.701	0.854	0.832	0.146	0.626	0.650	0.897
BaselineRandomForest/5	0.847	0.947	0.659	0.840	0.869	0.160	0.606	0.643	0.912
avg(BaselineRandomForest)	0.85	0.94	0.69	0.85	0.86	0.15	0.63	0.66	0.91
OptimizedRandomForest/1	0.863	0.900	0.793	0.892	0.808	0.109	0.693	0.696	0.936
OptimizedRandomForest/2	0.866	0.896	0.808	0.899	0.804	0.102	0.704	0.704	0.940
OptimizedRandomForest/3	0.870	0.922	0.772	0.884	0.840	0.116	0.694	0.708	0.947
OptimizedRandomForest/4	0.858	0.864	0.845	0.914	0.767	0.087	0.709	0.693	0.942
OptimizedRandomForest/5	0.869	0.897	0.816	0.902	0.808	0.098	0.714	0.712	0.936
avg(OptimizedRandomForest)	**0.87**[acc]	0.90[tpr]	**0.81**[tnr]	**0.90**[ppv]	0.81[npv]	**0.10**[far]	**0.70**[tss]	**0.70**[hss]	**0.94**[auc]

[acc] +0.02 ($p < 0.05$).
[tpr] -0.04 ($p < 0.05$).
[tnr] +0.12 ($p < 0.05$).
[ppv] +0.05 ($p < 0.05$).
[npv] -0.05 ($p < 0.05$).
[far] -0.05 ($p < 0.05$).
[tss] +0.07 ($p < 0.05$).
[hss] +0.04 ($p < 0.05$).
[auc] +0.03 ($p < 0.05$).

Table B.8: Case Study I, ≥ C flares, the next 48 h: evaluation of test sets.

Pred.Type/Test Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
BaselineRandomForest/1	0.858	0.937	0.709	0.858	0.857	0.142	0.646	0.673	0.914
BaselineRandomForest/2	0.833	0.921	0.667	0.840	0.816	0.160	0.587	0.614	0.891
BaselineRandomForest/3	0.809	0.907	0.624	0.821	0.780	0.179	0.531	0.558	0.897
BaselineRandomForest/4	0.827	0.944	0.605	0.819	0.852	0.182	0.549	0.590	0.880
BaselineRandomForest/5	0.861	0.948	0.697	0.855	0.876	0.145	0.645	0.677	0.929
avg(BaselineRandomForest)	0.84	0.93	0.66	0.84	0.84	0.16	0.59	0.62	0.90
OptimizedRandomForest/1	0.888	0.916	0.835	0.913	0.841	0.088	0.751	0.752	0.949
OptimizedRandomForest/2	0.841	0.866	0.794	0.888	0.758	0.112	0.660	0.652	0.925
OptimizedRandomForest/3	0.837	0.878	0.760	0.874	0.766	0.126	0.638	0.639	0.918
OptimizedRandomForest/4	0.877	0.902	0.831	0.910	0.818	0.091	0.732	0.730	0.942
OptimizedRandomForest/5	0.864	0.896	0.803	0.896	0.803	0.104	0.699	0.699	0.954
avg(OptimizedRandomForest)	**0.86**	0.89	**0.80**	**0.90**	0.80	**0.10**	**0.70**	**0.69**	**0.94**

^{acc} +0.02 ($p < 0.05$). ^{tnr} +0.14 ($p < 0.05$). ^{npv} -0.04 ($p < 0.05$). ^{tss} +0.11 ($p < 0.05$). ^{auc} +0.04 ($p < 0.05$).
^{tpr} -0.04 ($p < 0.05$). ^{ppv} +0.06 ($p < 0.05$). ^{far} -0.06 ($p < 0.05$). ^{hss} +0.07 ($p < 0.05$).

Table B.9: Case Study I, ≥ C flares, the next 72 h: model selection results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
AdaBoost/1	0.854	0.924	0.691	0.875	0.796	0.125	0.615	0.639	0.896
AdaBoost/2	0.854	0.920	0.700	0.878	0.790	0.122	0.620	0.641	0.889
AdaBoost/3	0.857	0.918	0.714	0.883	0.789	0.118	0.632	0.649	0.893
AdaBoost/4	0.857	0.918	0.712	0.882	0.790	0.118	0.631	0.649	0.894
AdaBoost/5	0.858	0.921	0.713	0.883	0.794	0.118	0.634	0.652	0.894
avg(AdaBoost)	0.86	0.92	0.71	0.88	0.79	0.12	0.63[a]	0.65	0.89
RandomForest/1	0.863	0.950	0.660	0.867	0.850	0.133	0.610	0.651	0.901
RandomForest/2	0.865	0.949	0.669	0.870	0.849	0.130	0.618	0.658	0.899
RandomForest/3	0.864	0.947	0.670	0.870	0.844	0.130	0.617	0.655	0.905
RandomForest/4	0.863	0.948	0.664	0.868	0.845	0.132	0.612	0.652	0.900
RandomForest/5	0.863	0.949	0.662	0.868	0.847	0.132	0.610	0.651	0.898
avg(RandomForest)	0.86	0.95	0.66	0.87	0.85	0.13	0.61[a]	0.65	0.90
GradientTreeBoosting/1	0.830	0.885	0.703	0.877	0.723	0.123	0.588	0.589	0.820
GradientTreeBoosting/2	0.818	0.853	0.736	0.886	0.687	0.114	0.589	0.574	0.815
GradientTreeBoosting/3	0.825	0.841	0.787	0.905	0.689	0.096	0.628	0.602	0.840
GradientTreeBoosting/4	0.830	0.862	0.754	0.893	0.708	0.107	0.616	0.603	0.825
GradientTreeBoosting/5	0.815	0.855	0.721	0.882	0.687	0.118	0.576	0.565	0.806
avg(GradientTreeBoosting)	0.82	0.86	0.74	0.89	0.70	0.11	0.60	0.59	0.82

[a] $TSS_{AdaBoost} > TSS_{RandomForest}$ ($p < 0.05$). As the AdaBoost output the highest TSS, the methodology has chosen it to proceed in the pipeline.

Table B.10: Case Study I, ≥ C flares, the next 72 h: feature selection results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
AdaBoost/1	0.883	0.934	0.762	0.902	0.834	0.098	0.696	0.713	0.928
AdaBoost/2	0.885	0.929	0.784	0.910	0.827	0.090	0.712	0.723	0.934
AdaBoost/3	0.885	0.937	0.765	0.903	0.839	0.097	0.702	0.719	0.930
AdaBoost/4	0.882	0.927	0.776	0.907	0.822	0.093	0.703	0.714	0.932
AdaBoost/5	0.883	0.936	0.760	0.901	0.837	0.099	0.696	0.715	0.928
avg(AdaBoost)	**0.88**[acc]	**0.93**[tpr]	**0.77**[tnr]	**0.90**[ppv]	**0.83**[npv]	**0.10**[far]	**0.70**[tss]	**0.72**[hss]	**0.93**[auc]

[acc] +0.02 ($p < 0.05$). [tnr] +0.06 ($p < 0.05$). [npv] +0.04 ($p < 0.05$). [tss] +0.07 ($p < 0.05$). [auc] +0.04 ($p < 0.05$).
[tpr] +0.01 ($p < 0.05$). [ppv] +0.02 ($p < 0.05$). [far] -0.02 ($p < 0.05$). [hss] +0.07 ($p < 0.05$).

Table B.11: Case Study I, ≥ C flares, the next 72 h: hyperparameter optimization results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
AdaBoost/1	0.888	0.936	0.774	0.907	0.840	0.093	0.711	0.727	0.946
AdaBoost/2	0.883	0.941	0.747	0.897	0.846	0.103	0.688	0.712	0.943
AdaBoost/3	0.886	0.939	0.762	0.902	0.844	0.098	0.701	0.721	0.943
AdaBoost/4	0.885	0.942	0.753	0.899	0.847	0.101	0.695	0.717	0.945
AdaBoost/5	0.885	0.939	0.760	0.901	0.842	0.099	0.698	0.718	0.943
avg(AdaBoost)	0.89[acc]	**0.94**[tpr]	0.76[tnr]	0.90	**0.84**[npv]	0.10	0.70	0.72	**0.94**[auc]

[acc] +0.01 ($p > 0.05$ and 0.1).
[tpr] +0.01 ($p < 0.05$).
[tnr] -0.01 ($p > 0.05$ and 0.1).
[npv] +0.01 ($p < 0.05$).
[auc] +0.01 ($p < 0.05$).

Table B.12: Case Study I, ≥ C flares, the next 72 h: data resampling results.

Method/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
SMOTE/1	0.818	0.769	0.932	0.964	0.634	0.036	0.701	0.618	0.939
SMOTE/2	0.812	0.762	0.928	0.961	0.626	0.039	0.690	0.606	0.935
SMOTE/3	0.814	0.764	0.930	0.963	0.628	0.038	0.694	0.610	0.938
SMOTE/4	0.814	0.764	0.931	0.963	0.628	0.037	0.695	0.611	0.937
SMOTE/5	0.813	0.764	0.927	0.961	0.628	0.039	0.691	0.608	0.937
avg(SMOTE)	0.81	0.76[a]	0.93[a]	0.96	0.63	0.04	0.69	0.61	0.94
SMOTE-ENN/1	0.855	0.853	0.862	0.935	0.716	0.065	0.714	0.675	0.938
SMOTE-ENN/2	0.863	0.863	0.863	0.936	0.730	0.064	0.725	0.689	0.940
SMOTE-ENN/3	0.865	0.866	0.862	0.937	0.735	0.064	0.728	0.694	0.939
SMOTE-ENN/4	0.865	0.866	0.863	0.937	0.735	0.063	0.730	0.695	0.941
SMOTE-ENN/5	0.861	0.862	0.859	0.935	0.728	0.065	0.722	0.686	0.938
avg(SMOTE-ENN)	0.86	0.86[b]	0.86[b]	0.94	0.73	0.06	0.72	0.69	0.94
SMOTE-Tomek/1	0.863	0.863	0.864	0.937	0.730	0.063	0.727	0.691	0.942
SMOTE-Tomek/2	0.869	0.872	0.862	0.937	0.743	0.063	0.734	0.702	0.944
SMOTE-Tomek/3	0.871	0.874	0.863	0.937	0.746	0.063	0.737	0.705	0.945
SMOTE-Tomek/4	0.870	0.872	0.864	0.938	0.744	0.062	0.736	0.704	0.942
SMOTE-Tomek/5	0.866	0.868	0.861	0.936	0.737	0.064	0.729	0.695	0.942
avg(SMOTE-Tomek)	0.87	0.87[c]	0.86[c]	0.94	0.74	0.06	0.73	0.70	0.94

[a] |TPR − TNR| = |0.76 − 0.93| = 0.17. [b] |TPR − TNR| = |0.86 − 0.86| = 0. [c] |TPR − TNR| = |0.87 − 0.86| = 0.01.

Table B.13: Case Study I, ≥ C flares, the next 72 h: cost function analysis results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
AdaBoost/1	0.870	0.871	0.869	0.940	0.742	0.060	0.740	0.705	0.945
AdaBoost/2	0.864	0.862	0.869	0.939	0.729	0.061	0.731	0.693	0.943
AdaBoost/3	0.866	0.867	0.865	0.938	0.736	0.062	0.732	0.697	0.943
AdaBoost/4	0.868	0.869	0.867	0.938	0.739	0.062	0.735	0.700	0.945
AdaBoost/5	0.871	0.873	0.866	0.938	0.745	0.062	0.739	0.706	0.943
avg(AdaBoost)	0.87[acc]	0.87[a,tpr]	**0.87**[a,tnr]	**0.94**[ppv]	0.74[npv]	**0.06**[far]	**0.74**[tss]	0.70[hss]	0.94

[a] |0.87 − 0.87| = 0. Noteworthily, both SMOTE-ENN and the cost function zeroed the difference |TPR − TNR|. However, we maintained the latter for outputting the highest TSS (0.74, $p < 0.05$) – compared to feature selection.
[acc] -0.01 ($p < 0.05$) – compared to feature selection.
[tpr] -0.07 ($p < 0.05$).
[tnr] +0.10 ($p < 0.05$) – compared to feature selection.
[ppv] +0.04 ($p < 0.05$) – compared to feature selection.
[npv] -0.10 ($p < 0.05$).
[far] -0.04 ($p < 0.05$) – compared to feature selection.
[tss] +0.04 ($p < 0.05$) – compared to feature selection.
[hss] -0.02 ($p < 0.05$) – compared to feature selection.

Table B.14: Case Study I, ≥ C flares, the next 72 h: cut-off point adjustment results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
AdaBoost/1	0.882	0.905	0.829	0.926	0.790	0.075	0.734	0.724	0.945
AdaBoost/2	0.873	0.893	0.826	0.923	0.769	0.077	0.719	0.704	0.943
AdaBoost/3	0.875	0.897	0.825	0.923	0.775	0.077	0.722	0.709	0.943
AdaBoost/4	0.877	0.893	0.840	0.929	0.771	0.071	0.733	0.715	0.943
AdaBoost/5	0.881	0.901	0.834	0.927	0.784	0.073	0.735	0.722	0.945
avg(AdaBoost)	**0.88**$^{\text{acc}}$	**0.90**$^{\text{tpr}}$	0.83$^{\text{tnr}}$	0.93$^{\text{ppv}}$	**0.78**$^{\text{npv}}$	0.07$^{\text{far}}$	0.73$^{\text{tss}}$	**0.71**$^{\text{hss}}$	0.94

$^{\text{acc}}$ +0.01 ($p < 0.05$). $^{\text{tnr}}$ -0.04 ($p < 0.05$). $^{\text{npv}}$ +0.04 ($p < 0.05$). $^{\text{tss}}$ -0.01 ($p < 0.05$).
$^{\text{tpr}}$ +0.03 ($p < 0.1$). $^{\text{ppv}}$ -0.01 ($p < 0.05$). $^{\text{far}}$ +0.01 ($p < 0.05$). $^{\text{hss}}$ +0.01 ($p < 0.05$).

Table B.15: Case Study I, ≥ C flares, the next 72 h: evaluation of validation sets.

Pred.Type/Val.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
BaselineAdaBoost/1	0.845	0.934	0.638	0.857	0.805	0.143	0.571	0.608	0.888
BaselineAdaBoost/2	0.858	0.928	0.694	0.876	0.804	0.124	0.621	0.647	0.910
BaselineAdaBoost/3	0.852	0.917	0.700	0.877	0.783	0.123	0.617	0.636	0.867
BaselineAdaBoost/4	0.861	0.942	0.673	0.871	0.832	0.130	0.614	0.650	0.891
BaselineAdaBoost/5	0.856	0.933	0.676	0.871	0.811	0.130	0.608	0.639	0.893
avg(BaselineAdaBoost)	0.85	0.93	0.68	0.87	0.81	0.13	0.61	0.64	0.89
OptimizedAdaBoost/1	0.892	0.912	0.846	0.933	0.805	0.068	0.758	0.747	0.947
OptimizedAdaBoost/2	0.869	0.873	0.861	0.936	0.744	0.064	0.734	0.702	0.947
OptimizedAdaBoost/3	0.870	0.890	0.824	0.922	0.762	0.078	0.714	0.698	0.939
OptimizedAdaBoost/4	0.888	0.914	0.827	0.925	0.805	0.075	0.742	0.736	0.949
OptimizedAdaBoost/5	0.874	0.904	0.803	0.915	0.782	0.085	0.707	0.702	0.940
avg(OptimizedAdaBoost)	**0.88**$^{\text{acc}}$	0.90$^{\text{tpr}}$	**0.83**$^{\text{tnr}}$	**0.93**$^{\text{ppv}}$	0.78$^{\text{npv}}$	**0.07**$^{\text{far}}$	**0.73**$^{\text{tss}}$	**0.72**$^{\text{hss}}$	**0.94**$^{\text{auc}}$

$^{\text{acc}}$ +0.03 ($p < 0.05$). $^{\text{tnr}}$ +0.15 ($p < 0.05$). $^{\text{npv}}$ -0.03 ($p > 0.05$ and 0.1). $^{\text{tss}}$ +0.12 ($p < 0.05$). $^{\text{auc}}$ +0.05 ($p < 0.05$).
$^{\text{tpr}}$ -0.03 ($p < 0.1$). $^{\text{ppv}}$ +0.06 ($p < 0.05$). $^{\text{far}}$ -0.06 ($p < 0.05$). $^{\text{hss}}$ +0.08 ($p < 0.05$).

Table B.16: Case Study I, ≥ C flares, the next 72 h: evaluation of test sets.

Pred.Type/Test Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
BaselineAdaBoost/1	0.839	0.910	0.673	0.866	0.763	0.134	0.583	0.603	0.903
BaselineAdaBoost/2	0.858	0.910	0.734	0.889	0.777	0.111	0.644	0.654	0.914
BaselineAdaBoost/3	0.854	0.941	0.648	0.863	0.824	0.137	0.589	0.628	0.884
BaselineAdaBoost/4	0.838	0.920	0.645	0.859	0.775	0.141	0.565	0.594	0.874
BaselineAdaBoost/5	0.852	0.923	0.686	0.874	0.791	0.126	0.609	0.633	0.911
avg(BaselineAdaBoost)	0.85	0.92	0.68	0.87	0.79	0.13	0.60	0.62	0.90
OptimizedAdaBoost/1	0.866	0.891	0.809	0.916	0.761	0.084	0.700	0.687	0.945
OptimizedAdaBoost/2	0.833	0.836	0.826	0.919	0.682	0.082	0.662	0.624	0.934
OptimizedAdaBoost/3	0.881	0.906	0.824	0.924	0.788	0.076	0.730	0.720	0.951
OptimizedAdaBoost/4	0.869	0.885	0.832	0.925	0.754	0.075	0.716	0.696	0.946
OptimizedAdaBoost/5	0.858	0.911	0.733	0.889	0.778	0.111	0.644	0.655	0.946
avg(OptimizedAdaBoost)	0.86^{acc}	<u>0.89</u>^{tpr}	**0.80**^{tnr}	**0.91**^{ppv}	<u>0.75</u>^{npv}	**0.09**^{far}	**0.69**^{tss}	0.68^{hss}	**0.94**^{auc}

^{acc} +0.01 ($p > 0.05$ and 0.1). ^{tnr} +0.12 ($p < 0.05$). ^{npv} -0.04 ($p < 0.1$). ^{tss} +0.09 ($p < 0.05$). ^{auc} +0.04 ($p < 0.05$).
^{tpr} -0.03 ($p < 0.1$). ^{ppv} +0.04 ($p < 0.05$). ^{far} -0.04 ($p < 0.05$). ^{hss} +0.06 ($p > 0.05$ and 0.1).

Table B.17: Case Study II, ≥ M flares, the next 24 h: model selection results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
AdaBoost/1	0.823	0.207	0.953	0.483	0.850	0.517	0.160	0.205	0.766
AdaBoost/2	0.821	0.207	0.951	0.471	0.850	0.529	0.157	0.201	0.756
AdaBoost/3	0.821	0.195	0.954	0.471	0.849	0.529	0.149	0.193	0.760
AdaBoost/4	0.822	0.221	0.950	0.483	0.852	0.517	0.171	0.216	0.770
AdaBoost/5	0.824	0.229	0.950	0.492	0.853	0.508	0.179	0.224	0.765
avg(AdaBoost)	0.82	0.21	0.95	0.48	0.85	0.52	0.16[a]	0.21	0.76
RandomForest/1	0.834	0.146	0.980	0.620	0.844	0.380	0.126	0.180	0.787
RandomForest/2	0.835	0.156	0.979	0.617	0.846	0.383	0.135	0.190	0.786
RandomForest/3	0.837	0.146	0.984	0.659	0.845	0.341	0.130	0.186	0.797
RandomForest/4	0.837	0.153	0.982	0.649	0.846	0.352	0.135	0.192	0.787
RandomForest/5	0.836	0.149	0.982	0.647	0.845	0.353	0.131	0.188	0.792
avg(RandomForest)	0.84	0.15	0.98	0.64	0.85	0.36	0.13	0.19	0.79
GradientTreeBoosting/1	0.740	0.543	0.782	0.350	0.891	0.650	0.325	0.265	0.679
GradientTreeBoosting/2	0.747	0.525	0.794	0.358	0.889	0.642	0.319	0.266	0.665
GradientTreeBoosting/3	0.743	0.531	0.788	0.354	0.890	0.646	0.319	0.262	0.670
GradientTreeBoosting/4	0.751	0.542	0.795	0.368	0.893	0.633	0.337	0.280	0.687
GradientTreeBoosting/5	0.738	0.533	0.782	0.351	0.890	0.649	0.315	0.257	0.664
avg(GradientTreeBoosting)	0.74	0.53	0.79	0.36	0.89	0.64	0.32[a]	0.27	0.67

[a] $TSS_{GradientTreeBoosting} > TSS_{AdaBoost}$ ($p < 0.05$). As the GradientTreeBoosting output the highest TSS, the methodology has chosen it to proceed in the pipeline.

Table B.18: Case Study II, ≥ M flares, the next 24 h: feature selection results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
GradientTreeBoosting/1	0.740	0.570	0.776	0.363	0.898	0.637	0.346	0.280	0.695
GradientTreeBoosting/2	0.750	0.514	0.800	0.361	0.888	0.639	0.314	0.266	0.672
GradientTreeBoosting/3	0.734	0.570	0.769	0.351	0.896	0.649	0.339	0.269	0.695
GradientTreeBoosting/4	0.741	0.553	0.781	0.355	0.894	0.645	0.334	0.271	0.699
GradientTreeBoosting/5	0.741	0.573	0.777	0.359	0.897	0.642	0.350	0.281	0.702
avg(GradientTreeBoosting)	0.74	**0.56**[tpr]	0.78[tnr]	0.36	0.89	0.64	0.34[tss]	0.27	**0.69**[auc]

[tpr] +0.03 ($p < 0.1$).
[tnr] -0.01 ($p > 0.05$ and 0.1).
[tss] +0.02 ($p > 0.05$ and 0.1).
[auc] +0.02 ($p < 0.05$).

Table B.19: Case Study II, ≥ M flares, the next 24 h: hyperparameter optimization results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
GradientTreeBoosting/1	0.851	0.279	0.972	0.678	0.864	0.322	0.251	0.325	0.854
GradientTreeBoosting/2	0.851	0.288	0.970	0.671	0.866	0.329	0.258	0.332	0.861
GradientTreeBoosting/3	0.848	0.274	0.970	0.659	0.863	0.341	0.244	0.316	0.855
GradientTreeBoosting/4	0.847	0.270	0.969	0.651	0.863	0.349	0.239	0.310	0.853
GradientTreeBoosting/5	0.845	0.254	0.970	0.643	0.860	0.357	0.224	0.293	0.851
avg(GradientTreeBoosting)	**0.85**[acc]	0.27[tpr]	**0.97**[tnr]	**0.66**[ppv]	0.86[npv]	**0.34**[far]	0.24[tss]	**0.32**[hss]	**0.85**[auc]

[acc] +0.11 ($p < 0.05$) – compared to model selection.
[tpr] -0.29 ($p < 0.05$).
[tnr] +0.18 ($p < 0.05$) – compared to model selection.
[ppv] +0.30 ($p < 0.05$) – compared to model selection.
[npv] -0.03 ($p < 0.05$) – compared to model selection.
[far] -0.30 ($p < 0.05$) – compared to model selection.
[tss] -0.08 ($p < 0.05$) – compared to model selection.
[hss] +0.05 ($p < 0.05$) – compared to model selection.
[auc] +0.16 ($p < 0.05$).

Table B.20: Case Study II, ≥ M flares, the next 24 h: data resampling results.

Method/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
SMOTE/1	0.741	0.806	0.728	0.386	0.947	0.614	0.534	0.373	0.843
SMOTE/2	0.739	0.799	0.726	0.382	0.945	0.618	0.525	0.367	0.841
SMOTE/3	0.740	0.799	0.728	0.384	0.945	0.616	0.527	0.370	0.843
SMOTE/4	0.744	0.821	0.728	0.390	0.951	0.610	0.548	0.382	0.849
SMOTE/5	0.746	0.814	0.732	0.392	0.949	0.608	0.546	0.383	0.841
avg(SMOTE)	0.74	0.81[a]	0.73[a]	0.39	0.95	0.61	0.54	0.38	0.84
SMOTE-ENN/1	0.671	0.893	0.624	0.335	0.965	0.665	0.517	0.312	0.841
SMOTE-ENN/2	0.677	0.884	0.634	0.339	0.963	0.661	0.518	0.317	0.842
SMOTE-ENN/3	0.664	0.889	0.616	0.329	0.963	0.671	0.505	0.302	0.839
SMOTE-ENN/4	0.671	0.884	0.626	0.334	0.962	0.666	0.510	0.309	0.843
SMOTE-ENN/5	0.685	0.893	0.640	0.345	0.966	0.655	0.533	0.328	0.846
avg(SMOTE-ENN)	0.67	0.89[b]	0.63[b]	0.34	0.96	0.66	0.52	0.31	0.84
SMOTE-Tomek/1	0.771	0.749	0.776	0.415	0.936	0.585	0.525	0.399	0.851
SMOTE-Tomek/2	0.771	0.743	0.777	0.414	0.935	0.586	0.520	0.396	0.848
SMOTE-Tomek/3	0.771	0.744	0.777	0.415	0.935	0.586	0.521	0.397	0.850
SMOTE-Tomek/4	0.770	0.759	0.772	0.415	0.938	0.586	0.532	0.400	0.848
SMOTE-Tomek/5	0.774	0.764	0.776	0.419	0.940	0.581	0.540	0.408	0.857
avg(SMOTE-Tomek)	0.77[acc]	**0.75**[c,tpr]	0.78[c,tnr]	0.42[ppv]	**0.94**[npv]	0.58[far]	**0.53**[tss]	**0.40**[hss]	0.85[auc]

[a] |TPR − TNR| = |0.81 − 0.73| = 0.08.
[b] |TPR − TNR| = |0.89 − 0.63| = 0.26.
[c] |TPR − TNR| = |0.75 − 0.78| = 0.03. As the SMOTE-Tomek output the lowest difference, the methodology has chosen it to proceed in the pipeline (GradientTreeBoosting does not support cost-sensitive learning).
[acc] -0.08 ($p < 0.05$). [ppv] -0.24 ($p < 0.05$). [tss] +0.29 ($p < 0.05$).
[tpr] +0.48 ($p < 0.05$). [npv] +0.08 ($p < 0.05$). [hss] +0.08 ($p < 0.05$).
[tnr] -0.19 ($p < 0.05$). [far] +0.24 ($p < 0.05$). [auc] -0.01 ($p > 0.05$ and 0.1).

Table B.21: Case Study II, ≥ M flares, the next 24 h: cut-off point adjustment results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
GradientTreeBoosting/1	0.731	0.812	0.714	0.376	0.947	0.624	0.527	0.361	0.848
GradientTreeBoosting/2	0.737	0.810	0.722	0.382	0.947	0.618	0.532	0.369	0.850
GradientTreeBoosting/3	0.744	0.832	0.725	0.391	0.953	0.609	0.557	0.385	0.857
GradientTreeBoosting/4	0.739	0.813	0.723	0.384	0.948	0.616	0.536	0.372	0.852
GradientTreeBoosting/5	0.737	0.823	0.719	0.384	0.951	0.617	0.542	0.374	0.848
avg(GradientTreeBoosting)	0.74[acc]	**0.82**[tpr]	0.72[tnr]	0.38[ppv]	**0.95**[npv]	0.62[far]	**0.54**[tss]	0.37[hss]	0.85

[acc] -0.03 ($p < 0.05$). [tnr] -0.06 ($p < 0.05$). [npv] +0.01 ($p < 0.05$). [tss] +0.01 ($p < 0.05$).
[tpr] +0.07 ($p < 0.05$). [ppv] -0.04 ($p < 0.05$). [far] +0.04 ($p < 0.05$). [hss] -0.03 ($p < 0.05$).

Table B.22: Case Study II, ≥ M flares, the next 24 h: evaluation of validation sets.

Pred.Type/Val.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
BaselineGradientTreeBoosting/1	0.651	0.720	0.636	0.296	0.915	0.704	0.357	0.228	0.687
BaselineGradientTreeBoosting/2	0.796	0.482	0.863	0.427	0.887	0.573	0.344	0.328	0.631
BaselineGradientTreeBoosting/3	0.741	0.534	0.784	0.345	0.888	0.656	0.318	0.262	0.611
BaselineGradientTreeBoosting/4	0.782	0.318	0.880	0.359	0.859	0.641	0.198	0.207	0.559
BaselineGradientTreeBoosting/5	0.747	0.495	0.800	0.343	0.882	0.657	0.295	0.251	0.690
avg(BaselineGradientTreeBoosting)	0.74	0.51	0.79	0.35	0.89	0.65	0.30	0.25	0.64
OptimizedGradientTreeBoosting/1	0.732	0.788	0.720	0.374	0.941	0.627	0.507	0.353	0.833
OptimizedGradientTreeBoosting/2	0.739	0.855	0.714	0.388	0.959	0.612	0.569	0.386	0.852
OptimizedGradientTreeBoosting/3	0.747	0.839	0.727	0.395	0.955	0.605	0.567	0.393	0.866
OptimizedGradientTreeBoosting/4	0.738	0.823	0.720	0.383	0.951	0.617	0.542	0.373	0.854
OptimizedGradientTreeBoosting/5	0.748	0.818	0.733	0.393	0.950	0.608	0.550	0.386	0.845
avg(OptimizedGradientTreeBoosting)	0.74	**0.82**$^{\text{tpr}}$	0.72$^{\text{tnr}}$	0.39$^{\text{ppv}}$	**0.95**$^{\text{npv}}$	0.61$^{\text{far}}$	**0.55**$^{\text{tss}}$	**0.38**$^{\text{hss}}$	**0.85**$^{\text{auc}}$

$^{\text{tpr}}$ +0.31 ($p < 0.05$). $^{\text{ppv}}$ +0.04 ($p > 0.05$ and 0.1). $^{\text{far}}$ -0.04 ($p > 0.05$ and 0.1). $^{\text{hss}}$ +0.13 ($p < 0.05$).
$^{\text{tnr}}$ -0.09 ($p > 0.05$ and 0.1). $^{\text{npv}}$ +0.06 ($p < 0.05$). $^{\text{tss}}$ +0.25 ($p < 0.05$). $^{\text{auc}}$ +0.21 ($p < 0.05$).

Table B.23: Case Study II, ≥ M flares, the next 24 h: evaluation of test sets.

Pred.Type/Test Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
BaselineGradientTreeBoosting/1	0.664	0.719	0.652	0.305	0.916	0.695	0.371	0.242	0.701
BaselineGradientTreeBoosting/2	0.825	0.500	0.894	0.500	0.894	0.500	0.394	0.394	0.648
BaselineGradientTreeBoosting/3	0.743	0.603	0.773	0.359	0.902	0.642	0.376	0.296	0.647
BaselineGradientTreeBoosting/4	0.774	0.381	0.858	0.364	0.866	0.636	0.239	0.234	0.595
BaselineGradientTreeBoosting/5	0.767	0.484	0.828	0.375	0.882	0.625	0.312	0.280	0.691
avg(BaselineGradientTreeBoosting)	0.75	0.54	0.80	0.38	0.89	0.62	0.34	0.29	0.66
OptimizedGradientTreeBoosting/1	0.702	0.797	0.682	0.347	0.941	0.653	0.479	0.317	0.822
OptimizedGradientTreeBoosting/2	0.743	0.844	0.721	0.391	0.956	0.609	0.565	0.388	0.827
OptimizedGradientTreeBoosting/3	0.713	0.810	0.692	0.357	0.945	0.643	0.502	0.334	0.856
OptimizedGradientTreeBoosting/4	0.749	0.905	0.715	0.404	0.972	0.596	0.620	0.417	0.883
OptimizedGradientTreeBoosting/5	0.716	0.807	0.697	0.362	0.944	0.638	0.503	0.339	0.826
avg(OptimizedGradientTreeBoosting)	0.72$^{\text{acc}}$	**0.83**$^{\text{tpr}}$	0.70$^{\text{tnr}}$	0.37$^{\text{ppv}}$	**0.95**$^{\text{npv}}$	0.63$^{\text{far}}$	**0.53**$^{\text{tss}}$	**0.36**$^{\text{hss}}$	**0.84**$^{\text{auc}}$

$^{\text{acc}}$ -0.03 ($p > 0.05$ and 0.1). $^{\text{ppv}}$ -0.01 ($p > 0.05$ and 0.1). $^{\text{tss}}$ +0.19 ($p < 0.05$).
$^{\text{tpr}}$ +0.29 ($p < 0.05$). $^{\text{npv}}$ +0.06 ($p < 0.05$). $^{\text{hss}}$ +0.07 ($p < 0.1$).
$^{\text{tnr}}$ -0.10 ($p < 0.1$). $^{\text{far}}$ +0.01 ($p > 0.05$ and 0.1). $^{\text{auc}}$ +0.18 ($p < 0.05$).

Table B.24: Case Study II, ≥ M flares, the next 48 h: model selection results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
AdaBoost/1	0.773	0.409	0.900	0.589	0.814	0.411	0.309	0.343	0.786
AdaBoost/2	0.766	0.393	0.897	0.573	0.809	0.427	0.289	0.321	0.776
AdaBoost/3	0.763	0.378	0.897	0.565	0.805	0.435	0.275	0.307	0.774
AdaBoost/4	0.769	0.408	0.894	0.576	0.812	0.424	0.302	0.334	0.783
AdaBoost/5	0.763	0.377	0.897	0.562	0.805	0.438	0.274	0.306	0.775
avg(AdaBoost)	0.77	0.39	0.90	0.57	0.81	0.43	0.29[a]	0.32	0.78
RandomForest/1	0.784	0.319	0.946	0.677	0.799	0.324	0.265	0.319	0.801
RandomForest/2	0.789	0.344	0.945	0.687	0.805	0.313	0.289	0.345	0.804
RandomForest/3	0.783	0.333	0.941	0.664	0.802	0.336	0.274	0.326	0.801
RandomForest/4	0.786	0.336	0.943	0.675	0.803	0.325	0.279	0.333	0.801
RandomForest/5	0.781	0.324	0.941	0.659	0.799	0.341	0.264	0.316	0.796
avg(RandomForest)	0.78	0.33	0.94	0.67	0.80	0.33	0.27	0.33	0.80
GradientTreeBoosting/1	0.711	0.667	0.727	0.466	0.864	0.534	0.394	0.345	0.724
GradientTreeBoosting/2	0.718	0.643	0.745	0.473	0.859	0.527	0.388	0.346	0.723
GradientTreeBoosting/3	0.725	0.618	0.763	0.480	0.853	0.520	0.380	0.346	0.706
GradientTreeBoosting/4	0.730	0.624	0.767	0.489	0.857	0.511	0.391	0.356	0.724
GradientTreeBoosting/5	0.725	0.634	0.757	0.482	0.858	0.518	0.391	0.352	0.727
avg(GradientTreeBoosting)	0.72	0.64	0.75	0.48	0.86	0.52	0.39[a]	0.35	0.72

[a] $\text{TSS}_{\text{GradientTreeBoosting}} > \text{TSS}_{\text{AdaBoost}}$ ($p < 0.05$). As the GradientTreeBoosting output the highest TSS, the methodology has chosen it to proceed in the pipeline.

Table B.25: Case Study II, ≥ M flares, the next 48 h: feature selection results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
GradientTreeBoosting/1	0.727	0.699	0.737	0.484	0.877	0.516	0.436	0.379	0.746
GradientTreeBoosting/2	0.724	0.633	0.756	0.480	0.857	0.520	0.389	0.351	0.724
GradientTreeBoosting/3	0.732	0.690	0.747	0.490	0.875	0.510	0.437	0.385	0.736
GradientTreeBoosting/4	0.721	0.642	0.749	0.477	0.859	0.523	0.391	0.350	0.732
GradientTreeBoosting/5	0.722	0.645	0.749	0.478	0.860	0.522	0.395	0.353	0.735
avg(GradientTreeBoosting)	0.73[acc]	**0.66**[tpr]	0.75	0.48	0.87[npv]	0.52	**0.41**[tss]	0.36[hss]	**0.73**[auc]

[acc] +0.01 ($p > 0.05$ and 0.1). [npv] +0.01 ($p > 0.05$ and 0.1). [hss] +0.01 ($p > 0.05$ and 0.1).
[tpr] +0.02 ($p < 0.1$). [tss] +0.02 ($p < 0.1$). [auc] +0.01 ($p < 0.05$).

Table B.26: Case Study II, ≥ M flares, the next 48 h: hyperparameter optimization results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
GradientTreeBoosting/1	0.814	0.531	0.913	0.681	0.848	0.319	0.444	0.478	0.860
GradientTreeBoosting/2	0.802	0.503	0.906	0.652	0.839	0.348	0.409	0.442	0.849
GradientTreeBoosting/3	0.808	0.515	0.910	0.667	0.843	0.333	0.425	0.459	0.852
GradientTreeBoosting/4	0.808	0.519	0.909	0.666	0.844	0.334	0.428	0.461	0.854
GradientTreeBoosting/5	0.808	0.526	0.906	0.663	0.846	0.337	0.433	0.463	0.854
avg(GradientTreeBoosting)	**0.81**[acc]	0.52[tpr]	**0.91**[tnr]	**0.67**[ppv]	0.84[npv]	**0.33**[far]	0.43[tss]	**0.46**[hss]	**0.85**[auc]

[acc] +0.09 ($p < 0.05$) – compared to model selection.
[tpr] -0.14 ($p < 0.05$).
[tnr] +0.16 ($p < 0.05$) – compared to model selection.
[ppv] +0.19 ($p < 0.05$) – compared to model selection.
[npv] -0.02 ($p < 0.05$) – compared to model selection.
[far] -0.19 ($p < 0.05$) – compared to model selection.
[tss] +0.02 ($p > 0.05$ and 0.1).
[hss] +0.11 ($p < 0.05$) – compared to model selection.
[auc] +0.12 ($p < 0.05$).

Table B.27: Case Study II, ≥ M flares, the next 48 h: data resampling results.

Method/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
SMOTE-ENN/1	0.723	0.832	0.684	0.480	0.921	0.520	0.516	0.417	0.846
SMOTE-ENN/2	0.720	0.844	0.677	0.477	0.926	0.523	0.521	0.417	0.842
SMOTE-ENN/3	0.724	0.835	0.686	0.482	0.922	0.518	0.520	0.420	0.842
SMOTE-ENN/4	0.725	0.841	0.685	0.483	0.925	0.517	0.526	0.424	0.846
SMOTE-ENN/5	0.723	0.843	0.681	0.480	0.926	0.520	0.524	0.421	0.843
avg(SMOTE-ENN)	0.72	0.84[a]	0.68[a]	0.48	0.92	0.52	0.52	0.42	0.84
SMOTE-Tomek/1	0.789	0.683	0.826	0.579	0.882	0.421	0.509	0.481	0.854
SMOTE-Tomek/2	0.777	0.674	0.813	0.558	0.877	0.442	0.487	0.456	0.844
SMOTE-Tomek/3	0.780	0.677	0.816	0.563	0.879	0.437	0.493	0.462	0.849
SMOTE-Tomek/4	0.782	0.676	0.819	0.567	0.879	0.433	0.495	0.466	0.848
SMOTE-Tomek/5	0.780	0.675	0.817	0.563	0.878	0.437	0.492	0.462	0.848
avg(SMOTE-Tomek)	0.78	0.68[b]	0.82[b]	0.57	0.88	0.43	0.50	0.47	0.85
SMOTE/1	0.768	0.736	0.779	0.538	0.894	0.462	0.514	0.459	0.849
SMOTE/2	0.758	0.727	0.769	0.524	0.890	0.476	0.496	0.441	0.839
SMOTE/3	0.763	0.731	0.774	0.531	0.892	0.469	0.505	0.450	0.844
SMOTE/4	0.766	0.725	0.780	0.536	0.891	0.464	0.505	0.453	0.844
SMOTE/5	0.762	0.723	0.776	0.530	0.889	0.470	0.499	0.446	0.842
avg(SMOTE)	0.76[acc]	**0.73**[c,tpr]	0.78[c,tnr]	0.53[ppv]	**0.89**[npv]	0.47[far]	**0.50**[tss]	0.45[hss]	0.84[auc]

[a] |TPR – TNR| = |0.84 – 0.68| = 0.16.
[b] |TPR – TNR| = |0.68 – 0.82| = 0.14.
[c] |TPR – TNR| = |0.73 – 0.78| = 0.05. As the SMOTE output the lowest difference, the methodology has chosen it to proceed in the pipeline (GradientTreeBoosting does not support cost-sensitive learning).
[acc] -0.05 ($p < 0.05$).
[tpr] +0.21 ($p < 0.05$).
[tnr] -0.13 ($p < 0.05$).
[ppv] -0.14 ($p < 0.05$).
[npv] +0.05 ($p < 0.05$).
[far] +0.14 ($p < 0.05$).
[tss] +0.09 ($p < 0.05$) – compared to feature selection.
[hss] -0.01 ($p < 0.05$).
[auc] -0.01 ($p < 0.05$).

Table B.28: Case Study II, ≥ M flares, the next 48 h: cut-off point adjustment results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
GradientTreeBoosting/1	0.724	0.827	0.688	0.481	0.920	0.519	0.515	0.417	0.839
GradientTreeBoosting/2	0.729	0.823	0.696	0.486	0.918	0.514	0.518	0.423	0.843
GradientTreeBoosting/3	0.727	0.828	0.692	0.485	0.920	0.515	0.520	0.423	0.844
GradientTreeBoosting/4	0.724	0.826	0.689	0.481	0.919	0.519	0.514	0.417	0.843
GradientTreeBoosting/5	0.730	0.829	0.695	0.487	0.921	0.513	0.524	0.426	0.850
avg(GradientTreeBoosting)	0.73$^{\text{acc}}$	**0.83**$^{\text{tpr}}$	0.69$^{\text{tnr}}$	0.48$^{\text{ppv}}$	**0.92**$^{\text{npv}}$	0.52$^{\text{far}}$	**0.52**$^{\text{tss}}$	0.42$^{\text{hss}}$	0.84

$^{\text{acc}}$ -0.03 ($p < 0.05$). $^{\text{tnr}}$ -0.09 ($p < 0.05$). $^{\text{npv}}$ +0.03 ($p < 0.05$). $^{\text{tss}}$ +0.02 ($p < 0.05$).
$^{\text{tpr}}$ +0.10 ($p < 0.05$). $^{\text{ppv}}$ -0.05 ($p < 0.05$). $^{\text{far}}$ +0.05 ($p < 0.05$). $^{\text{hss}}$ -0.03 ($p < 0.05$).

Table B.29: Case Study II, ≥ M flares, the next 48 h: evaluation of validation sets.

Pred.Type/Val.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
BaselineGradientTreeBoosting/1	0.755	0.717	0.769	0.520	0.886	0.480	0.485	0.432	0.746
BaselineGradientTreeBoosting/2	0.790	0.709	0.818	0.576	0.890	0.425	0.526	0.490	0.823
BaselineGradientTreeBoosting/3	0.721	0.691	0.731	0.472	0.872	0.528	0.422	0.367	0.788
BaselineGradientTreeBoosting/4	0.737	0.442	0.840	0.490	0.812	0.510	0.282	0.291	0.636
BaselineGradientTreeBoosting/5	0.745	0.456	0.846	0.508	0.817	0.492	0.302	0.312	0.644
avg(BaselineGradientTreeBoosting)	0.75	0.60	0.80	0.51	0.86	0.49	0.40	0.38	0.73
OptimizedGradientTreeBoosting/1	0.730	0.839	0.692	0.488	0.925	0.512	0.531	0.430	0.856
OptimizedGradientTreeBoosting/2	0.738	0.811	0.712	0.496	0.915	0.504	0.523	0.433	0.850
OptimizedGradientTreeBoosting/3	0.727	0.846	0.685	0.484	0.927	0.516	0.531	0.427	0.851
OptimizedGradientTreeBoosting/4	0.715	0.842	0.670	0.472	0.924	0.529	0.512	0.408	0.839
OptimizedGradientTreeBoosting/5	0.727	0.804	0.700	0.483	0.911	0.517	0.503	0.414	0.848
avg(OptimizedGradientTreeBoosting)	0.73$^{\text{acc}}$	**0.83**$^{\text{tpr}}$	0.69$^{\text{tnr}}$	0.48$^{\text{ppv}}$	**0.92**$^{\text{npv}}$	0.52$^{\text{far}}$	**0.52**$^{\text{tss}}$	0.42$^{\text{hss}}$	**0.85**$^{\text{auc}}$

$^{\text{acc}}$ -0.02 ($p < 0.05$). $^{\text{tnr}}$ -0.11 ($p < 0.05$). $^{\text{npv}}$ +0.06 ($p < 0.05$). $^{\text{tss}}$ +0.12 ($p < 0.1$). $^{\text{auc}}$ +0.12 ($p < 0.05$).
$^{\text{tpr}}$ +0.23 ($p < 0.05$). $^{\text{ppv}}$ -0.03 ($p < 0.05$). $^{\text{far}}$ +0.03 ($p < 0.1$). $^{\text{hss}}$ +0.04 ($p > 0.05$ and 0.1).

Table B.30: Case Study II, ≥ M flares, the next 48 h: evaluation of test sets.

Pred.Type/Test Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
BaselineGradientTreeBoosting/1	0.762	0.642	0.804	0.535	0.865	0.465	0.447	0.419	0.671
BaselineGradientTreeBoosting/2	0.781	0.663	0.822	0.568	0.874	0.432	0.485	0.460	0.829
BaselineGradientTreeBoosting/3	0.724	0.660	0.746	0.477	0.862	0.523	0.406	0.361	0.790
BaselineGradientTreeBoosting/4	0.740	0.441	0.845	0.500	0.812	0.500	0.286	0.298	0.623
BaselineGradientTreeBoosting/5	0.753	0.539	0.828	0.521	0.837	0.479	0.366	0.362	0.697
avg(BaselineGradientTreeBoosting)	0.75	0.59	0.81	0.52	0.85	0.48	0.40	0.38	0.72
OptimizedGradientTreeBoosting/1	0.749	0.863	0.709	0.509	0.937	0.491	0.572	0.466	0.861
OptimizedGradientTreeBoosting/2	0.748	0.884	0.700	0.509	0.945	0.491	0.584	0.472	0.858
OptimizedGradientTreeBoosting/3	0.743	0.798	0.724	0.503	0.911	0.497	0.522	0.439	0.835
OptimizedGradientTreeBoosting/4	0.721	0.807	0.691	0.478	0.910	0.522	0.497	0.406	0.835
OptimizedGradientTreeBoosting/5	0.742	0.890	0.690	0.500	0.947	0.500	0.580	0.462	0.882
avg(OptimizedGradientTreeBoosting)	0.74$^{\text{acc}}$	**0.85**$^{\text{tpr}}$	0.70$^{\text{tnr}}$	0.50$^{\text{ppv}}$	**0.93**$^{\text{npv}}$	0.50$^{\text{far}}$	**0.55**$^{\text{tss}}$	**0.45**$^{\text{hss}}$	**0.85**$^{\text{auc}}$

$^{\text{acc}}$ -0.01 ($p > 0.05$ and 0.1). $^{\text{ppv}}$ -0.02 ($p > 0.05$ and 0.1). $^{\text{tss}}$ +0.15 ($p < 0.05$).
$^{\text{tpr}}$ +0.26 ($p < 0.05$). $^{\text{npv}}$ +0.08 ($p < 0.05$). $^{\text{hss}}$ +0.07 ($p < 0.05$).
$^{\text{tnr}}$ -0.11 ($p < 0.05$). $^{\text{far}}$ +0.02 ($p > 0.05$ and 0.1). $^{\text{auc}}$ +0.13 ($p < 0.05$).

Table B.31: Case Study II, ≥ M flares, the next 72 h: model selection results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
AdaBoost/1	0.747	0.544	0.842	0.616	0.799	0.384	0.386	0.398	0.791
AdaBoost/2	0.749	0.540	0.846	0.620	0.798	0.380	0.386	0.399	0.794
AdaBoost/3	0.754	0.572	0.838	0.623	0.808	0.378	0.410	0.419	0.794
AdaBoost/4	0.750	0.539	0.847	0.622	0.799	0.378	0.387	0.400	0.794
AdaBoost/5	0.746	0.527	0.848	0.617	0.794	0.383	0.374	0.389	0.791
avg(AdaBoost)	0.75	0.54	0.84	0.62	0.80	0.38	0.39	0.40	0.79
RandomForest/1	0.763	0.522	0.875	0.662	0.798	0.338	0.397	0.420	0.819
RandomForest/2	0.766	0.504	0.888	0.679	0.794	0.321	0.392	0.421	0.821
RandomForest/3	0.767	0.515	0.884	0.675	0.797	0.325	0.398	0.425	0.822
RandomForest/4	0.771	0.526	0.885	0.682	0.801	0.318	0.411	0.437	0.821
RandomForest/5	0.763	0.515	0.878	0.664	0.796	0.336	0.393	0.417	0.816
avg(RandomForest)	0.77	0.52	0.88	0.67	0.80	0.33	0.40[a]	0.42	0.82
GradientTreeBoosting/1	0.718	0.697	0.728	0.547	0.840	0.453	0.425	0.395	0.739
GradientTreeBoosting/2	0.719	0.661	0.746	0.552	0.829	0.448	0.407	0.385	0.734
GradientTreeBoosting/3	0.723	0.662	0.752	0.558	0.830	0.442	0.414	0.393	0.738
GradientTreeBoosting/4	0.724	0.670	0.748	0.556	0.833	0.444	0.419	0.395	0.734
GradientTreeBoosting/5	0.731	0.639	0.773	0.569	0.824	0.431	0.413	0.398	0.725
avg(GradientTreeBoosting)	0.72	0.67	0.75	0.56	0.83	0.44	0.42[a]	0.39	0.73

[a] $\text{TSS}_{\text{GradientTreeBoosting}} > \text{TSS}_{\text{RandomForest}}$ ($p < 0.05$). As the GradientTreeBoosting output the highest TSS, the methodology has chosen it to proceed in the pipeline.

Table B.32: Case Study II, ≥ M flares, the next 72 h: feature selection results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
GradientTreeBoosting/1	0.728	0.693	0.744	0.561	0.841	0.439	0.438	0.411	0.744
GradientTreeBoosting/2	0.730	0.703	0.742	0.563	0.846	0.437	0.445	0.416	0.750
GradientTreeBoosting/3	0.729	0.699	0.743	0.561	0.844	0.439	0.442	0.414	0.750
GradientTreeBoosting/4	0.721	0.738	0.713	0.549	0.856	0.451	0.451	0.413	0.761
GradientTreeBoosting/5	0.743	0.694	0.765	0.582	0.845	0.418	0.459	0.436	0.761
avg(GradientTreeBoosting)	**0.73**[acc]	**0.71**[tpr]	0.74[tnr]	0.56	**0.85**[npv]	0.44	**0.45**[tss]	**0.42**[hss]	**0.75**[auc]

[acc] +0.01 ($p < 0.1$).
[tpr] +0.04 ($p < 0.1$).
[tnr] -0.01 ($p > 0.05$ and 0.1).
[npv] +0.02 ($p < 0.05$).
[tss] +0.03 ($p < 0.05$).
[hss] +0.03 ($p < 0.05$).
[auc] +0.02 ($p < 0.05$).

Table B.33: Case Study II, ≥ M flares, the next 72 h: hyperparameter optimization results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
GradientTreeBoosting/1	0.797	0.614	0.883	0.710	0.831	0.290	0.497	0.515	0.863
GradientTreeBoosting/2	0.804	0.620	0.890	0.724	0.835	0.276	0.510	0.530	0.869
GradientTreeBoosting/3	0.795	0.612	0.880	0.705	0.830	0.295	0.492	0.510	0.861
GradientTreeBoosting/4	0.798	0.619	0.882	0.709	0.833	0.291	0.501	0.518	0.865
GradientTreeBoosting/5	0.791	0.609	0.876	0.695	0.828	0.305	0.485	0.501	0.861
avg(GradientTreeBoosting)	**0.80**[acc]	0.61[tpr]	**0.88**[tnr]	**0.71**[ppv]	0.83[npv]	**0.29**[far]	**0.50**[tss]	**0.51**[hss]	**0.86**[auc]

[acc] +0.07 ($p < 0.05$).
[tpr] -0.10 ($p < 0.05$).
[tnr] +0.13 ($p < 0.05$) – compared to model selection.
[ppv] +0.15 ($p < 0.05$) – compared to model selection.
[npv] -0.02 ($p < 0.05$).
[far] -0.15 ($p < 0.05$) – compared to model selection.
[tss] +0.05 ($p < 0.05$).
[hss] +0.09 ($p < 0.05$).
[auc] +0.11 ($p < 0.05$).

Table B.34: Case Study II, ≥ M flares, the next 72 h: data resampling results.

Method/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
SMOTE/1	0.757	0.819	0.728	0.584	0.897	0.416	0.547	0.494	0.857
SMOTE/2	0.760	0.813	0.736	0.589	0.894	0.411	0.548	0.498	0.861
SMOTE/3	0.753	0.817	0.723	0.579	0.895	0.421	0.540	0.487	0.855
SMOTE/4	0.761	0.814	0.737	0.591	0.895	0.410	0.551	0.500	0.863
SMOTE/5	0.756	0.818	0.727	0.582	0.896	0.418	0.545	0.492	0.856
avg(SMOTE)	0.76	0.82[a]	0.73[a]	0.58	0.90	0.42	0.55	0.49	0.86
SMOTE-ENN/1	0.737	0.862	0.679	0.556	0.914	0.444	0.541	0.471	0.852
SMOTE-ENN/2	0.730	0.859	0.671	0.548	0.911	0.452	0.530	0.460	0.846
SMOTE-ENN/3	0.739	0.860	0.683	0.558	0.913	0.442	0.543	0.474	0.854
SMOTE-ENN/4	0.738	0.849	0.687	0.558	0.907	0.442	0.536	0.470	0.851
SMOTE-ENN/5	0.733	0.869	0.669	0.550	0.917	0.450	0.538	0.466	0.849
avg(SMOTE-ENN)	0.74	0.86[b]	0.68[b]	0.55	0.91	0.45	0.54	0.47	0.85
SMOTE-Tomek/1	0.778	0.763	0.784	0.623	0.877	0.378	0.547	0.516	0.861
SMOTE-Tomek/2	0.775	0.763	0.781	0.619	0.876	0.381	0.544	0.512	0.859
SMOTE-Tomek/3	0.783	0.757	0.795	0.633	0.876	0.367	0.553	0.525	0.867
SMOTE-Tomek/4	0.779	0.754	0.791	0.627	0.874	0.373	0.545	0.517	0.864
SMOTE-Tomek/5	0.773	0.759	0.779	0.615	0.875	0.385	0.538	0.507	0.860
avg(SMOTE-Tomek)	0.78[acc]	**0.76**[c,tpr]	0.79[c,tnr]	0.62[ppv]	**0.88**[npv]	0.38[far]	**0.55**[tss]	0.52[hss]	0.86

[a] $|TPR - TNR| = |0.82 - 0.73| = 0.09$
[b] $|TPR - TNR| = |0.86 - 0.68| = 0.18$
[c] $|TPR - TNR| = |0.76 - 0.79| = 0.03$. As the SMOTE-Tomek output the lowest difference, the methodology has chosen it to proceed in the pipeline (GradientTreeBoosting does not support cost-sensitive learning).
[acc] -0.02 ($p < 0.05$).
[tpr] +0.15 ($p < 0.05$).
[tnr] -0.09 ($p < 0.05$).
[ppv] -0.09 ($p < 0.05$).
[npv] +0.05 ($p < 0.05$).
[far] +0.09 ($p < 0.05$).
[tss] +0.05 ($p < 0.05$).
[hss] +0.01 ($p > 0.05$ and 0.1).

Table B.35: Case Study II, ≥ M flares, the next 72 h: cut-off point adjustment results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
GradientTreeBoosting/1	0.773	0.799	0.760	0.608	0.891	0.392	0.559	0.516	0.867
GradientTreeBoosting/2	0.763	0.800	0.745	0.594	0.889	0.406	0.545	0.499	0.859
GradientTreeBoosting/3	0.762	0.802	0.743	0.592	0.890	0.408	0.545	0.498	0.860
GradientTreeBoosting/4	0.766	0.801	0.750	0.599	0.890	0.401	0.551	0.506	0.861
GradientTreeBoosting/5	0.768	0.794	0.755	0.602	0.888	0.398	0.550	0.507	0.864
avg(GradientTreeBoosting)	0.77^{acc}	**0.80**^{tpr}	0.75^{tnr}	0.60^{ppv}	**0.89**^{npv}	0.40^{far}	0.55	0.50^{hss}	0.86

^{acc} -0.01 ($p < 0.05$). ^{tnr} -0.04 ($p < 0.05$). ^{npv} +0.01 ($p < 0.05$). ^{hss} -0.02 ($p > 0.05$ and 0.1).
^{tpr} +0.04 ($p < 0.05$). ^{ppv} -0.02 ($p < 0.05$). ^{far} +0.02 ($p < 0.05$).

Table B.36: Case Study II, ≥ M flares, the next 72 h: evaluation of validation sets.

Pred.Type/Val.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
BaselineGradientTreeBoosting/1	0.769	0.711	0.795	0.618	0.856	0.382	0.507	0.487	0.809
BaselineGradientTreeBoosting/2	0.751	0.669	0.789	0.595	0.836	0.405	0.457	0.443	0.797
BaselineGradientTreeBoosting/3	0.768	0.669	0.814	0.626	0.841	0.374	0.482	0.474	0.805
BaselineGradientTreeBoosting/4	0.743	0.769	0.731	0.571	0.872	0.429	0.500	0.458	0.795
BaselineGradientTreeBoosting/5	0.716	0.619	0.761	0.546	0.811	0.455	0.380	0.366	0.708
avg(BaselineGradientTreeBoosting)	0.75	0.69	0.78	0.59	0.84	0.41	0.47	0.45	0.78
OptimizedGradientTreeBoosting/1	0.777	0.820	0.757	0.611	0.900	0.389	0.577	0.528	0.874
OptimizedGradientTreeBoosting/2	0.751	0.800	0.727	0.577	0.887	0.423	0.527	0.478	0.849
OptimizedGradientTreeBoosting/3	0.784	0.797	0.778	0.626	0.892	0.374	0.575	0.536	0.870
OptimizedGradientTreeBoosting/4	0.771	0.811	0.753	0.604	0.896	0.396	0.564	0.517	0.863
OptimizedGradientTreeBoosting/5	0.762	0.811	0.739	0.591	0.894	0.409	0.550	0.500	0.861
avg(OptimizedGradientTreeBoosting)	**0.77**^{acc}	**0.81**^{tpr}	0.75^{tnr}	0.60^{ppv}	**0.89**^{npv}	0.40^{far}	**0.56**^{tss}	**0.51**^{hss}	**0.86**^{auc}

^{acc} +0.02 ($p < 0.1$). ^{ppv} +0.01 ($p > 0.05$ and 0.1). ^{tss} +0.09 ($p < 0.05$).
^{tpr} +0.12 ($p < 0.05$). ^{npv} +0.05 ($p < 0.05$). ^{hss} +0.06 ($p < 0.05$).
^{tnr} -0.03 ($p > 0.05$ and 0.1). ^{far} -0.01 ($p < 0.05$). ^{auc} +0.08 ($p < 0.05$).

Table B.37: Case Study II, ≥ M flares, the next 72 h: evaluation of test sets.

Pred.Type/Test Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
BaselineGradientTreeBoosting/1	0.762	0.750	0.768	0.600	0.869	0.400	0.518	0.486	0.819
BaselineGradientTreeBoosting/2	0.764	0.750	0.771	0.604	0.869	0.396	0.521	0.490	0.833
BaselineGradientTreeBoosting/3	0.716	0.644	0.749	0.544	0.819	0.456	0.393	0.374	0.774
BaselineGradientTreeBoosting/4	0.732	0.719	0.738	0.562	0.849	0.438	0.457	0.425	0.790
BaselineGradientTreeBoosting/5	0.676	0.625	0.700	0.493	0.800	0.507	0.325	0.303	0.699
avg(BaselineGradientTreeBoosting)	0.73	0.70	0.75	0.56	0.84	0.44	0.44	0.42	0.78
OptimizedGradientTreeBoosting/1	0.801	0.871	0.768	0.635	0.928	0.365	0.639	0.581	0.889
OptimizedGradientTreeBoosting/2	0.770	0.828	0.743	0.600	0.902	0.400	0.571	0.518	0.870
OptimizedGradientTreeBoosting/3	0.702	0.765	0.672	0.521	0.860	0.479	0.437	0.389	0.829
OptimizedGradientTreeBoosting/4	0.757	0.772	0.750	0.591	0.876	0.409	0.522	0.483	0.852
OptimizedGradientTreeBoosting/5	0.739	0.768	0.725	0.566	0.870	0.434	0.493	0.450	0.830
avg(OptimizedGradientTreeBoosting)	0.75^{acc}	**0.80**^{tpr}	0.73^{tnr}	0.58^{ppv}	**0.89**^{npv}	0.42^{far}	**0.53**^{tss}	**0.48**^{hss}	**0.85**^{auc}

^{acc} +0.02 ($p > 0.05$ and 0.1). ^{ppv} +0.02 ($p > 0.05$ and 0.1). ^{tss} +0.09 ($p < 0.05$).
^{tpr} +0.10 ($p < 0.05$). ^{npv} +0.05 ($p < 0.05$). ^{hss} +0.06 ($p < 0.05$).
^{tnr} -0.02 ($p > 0.05$ and 0.1). ^{far} -0.02 ($p > 0.05$ and 0.1). ^{auc} +0.07 ($p < 0.05$).

Table B.38: Case Study III, ≥ M flares, the next 24 h: model selection results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
AdaBoost/1	0.811	0.397	0.913	0.546	0.861	0.454	0.310	0.34	0.735
AdaBoost/2	0.808	0.320	0.928	0.547	0.848	0.453	0.248	0.287	0.690
AdaBoost/3	0.819	0.361	0.932	0.579	0.857	0.421	0.293	0.335	0.712
AdaBoost/4	0.822	0.400	0.926	0.583	0.863	0.417	0.326	0.365	0.731
AdaBoost/5	0.810	0.369	0.918	0.537	0.856	0.463	0.286	0.320	0.710
avg(AdaBoost)	0.81	0.37	0.92	0.56	0.86	0.44	0.29[a]	0.33	0.72[b]
RandomForest/1	0.838	0.293	0.972	0.751	0.849	0.249	0.265	0.335	0.801
RandomForest/2	0.839	0.303	0.971	0.737	0.850	0.263	0.273	0.344	0.779
RandomForest/3	0.831	0.260	0.972	0.710	0.843	0.290	0.232	0.297	0.780
RandomForest/4	0.842	0.278	0.980	0.783	0.847	0.217	0.258	0.334	0.784
RandomForest/5	0.842	0.294	0.976	0.757	0.850	0.243	0.270	0.3445	0.793
avg(RandomForest)	0.84	0.29	0.97	0.75	0.85	0.25	0.26[a]	0.33	0.79[b]
GradientTreeBoosting/1	0.699	0.494	0.750	0.380	0.850	0.620	0.244	0.230	0.605
GradientTreeBoosting/2	0.697	0.443	0.760	0.351	0.842	0.639	0.203	0.194	0.583
GradientTreeBoosting/3	0.709	0.466	0.768	0.368	0.852	0.632	0.234	0.218	0.620
GradientTreeBoosting/4	0.701	0.444	0.764	0.356	0.847	0.644	0.208	0.197	0.579
GradientTreeBoosting/5	0.713	0.499	0.766	0.398	0.857	0.602	0.264	0.249	0.628
avg(GradientTreeBoosting)	0.70	0.47	0.76	0.37	0.85	0.63	0.23	0.22	0.60

[a] $TSS_{AdaBoost} > TSS_{RandomForest}$ ($p > 0.05$ and 0.1).
[b] $AUC_{RandomForest} > AUC_{AdaBoost}$ ($p < 0.05$). Noteworthily, as we could not confirm the significance of $TSS_{AdaBoost} > TSS_{RandomForest}$, we checked a second criteria, namely the AUCs. We then chose RandomForest to proceed in the pipeline.

Table B.39: Case Study III, ≥ M flares, the next 24 h: feature selection results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
RandomForest/1	0.841	0.355	0.960	0.705	0.859	0.295	0.315	0.382	0.762
RandomForest/2	0.846	0.386	0.959	0.708	0.865	0.292	0.345	0.411	0.806
RandomForest/3	0.844	0.383	0.957	0.708	0.864	0.292	0.340	0.404	0.795
RandomForest/4	0.842	0.329	0.967	0.724	0.855	0.276	0.296	0.367	0.782
RandomForest/5	0.828	0.319	0.952	0.645	0.851	0.355	0.271	0.330	0.772
avg(RandomForest)	0.84	**0.35**[tpr]	0.96[tnr]	0.70[ppv]	**0.86**[npv]	0.30[far]	**0.31**[tss]	**0.38**[hss]	0.78[auc]

[tpr] +0.06 ($p < 0.05$).
[tnr] -0.01 ($p < 0.05$).
[ppv] -0.05 ($p < 0.05$).
[npv] +0.01 ($p < 0.05$).
[far] +0.30 ($p < 0.05$).
[tss] +0.05 ($p < 0.05$).
[hss] +0.05 ($p < 0.1$).
[auc] -0.01 ($p > 0.05$ and 0.1).

Table B.40: Case Study III, ≥ M flares, the next 24 h: hyperparameter optimization results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
RandomForest/1	0.837	0.326	0.962	0.680	0.854	0.310	0.288	0.355	0.833
RandomForest/2	0.844	0.380	0.958	0.694	0.863	0.306	0.338	0.403	0.848
RandomForest/3	0.830	0.298	0.960	0.674	0.848	0.327	0.258	0.323	0.819
RandomForest/4	0.850	0.402	0.960	0.722	0.868	0.278	0.362	0.429	0.840
RandomForest/5	0.839	0.352	0.958	0.691	0.858	0.309	0.310	0.377	0.816
avg(RandomForest)	0.84	0.35	0.96	0.69[ppv]	0.86	0.31[far]	0.31	0.38	**0.83**[auc]

[ppv] -0.01 ($p > 0.05$ and 0.1).
[far] +0.01 ($p > 0.05$ and 0.1).
[auc] +0.04 ($p < 0.05$) – compared to model selection.

Table B.41: Case Study III, ≥ M flares, the next 24 h: data resampling results.

Method/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
SMOTE/1	0.745	0.776	0.737	0.425	0.931	0.575	0.513	0.391	0.833
SMOTE/2	0.730	0.736	0.729	0.405	0.919	0.595	0.465	0.355	0.808
SMOTE/3	0.744	0.760	0.740	0.422	0.927	0.578	0.500	0.384	0.822
SMOTE/4	0.739	0.773	0.730	0.419	0.930	0.581	0.503	0.382	0.833
SMOTE/5	0.725	0.756	0.717	0.402	0.924	0.598	0.473	0.355	0.814
avg(SMOTE)	0.74	0.76[a]	0.73[a]	0.41	0.93	0.59	0.49	0.37	0.82
SMOTE-ENN/1	0.757	0.783	0.751	0.440	0.935	0.560	0.533	0.412	0.837
SMOTE-ENN/2	0.733	0.774	0.723	0.411	0.929	0.589	0.497	0.373	0.811
SMOTE-ENN/3	0.746	0.759	0.743	0.424	0.927	0.576	0.502	0.386	0.829
SMOTE-ENN/4	0.727	0.799	0.710	0.407	0.936	0.594	0.508	0.372	0.822
SMOTE-ENN/5	0.727	0.770	0.717	0.404	0.929	0.596	0.487	0.362	0.811
avg(SMOTE-ENN)	0.74	0.78[b]	0.73[b]	0.42	0.93	0.58	0.51	0.38	0.82
SMOTE-Tomek/1	0.757	0.711	0.768	0.434	0.916	0.566	0.479	0.386	0.813
SMOTE-Tomek/2	0.783	0.747	0.792	0.474	0.928	0.526	0.539	0.442	0.845
SMOTE-Tomek/3	0.767	0.745	0.773	0.449	0.926	0.551	0.518	0.413	0.827
SMOTE-Tomek/4	0.770	0.725	0.781	0.453	0.922	0.547	0.506	0.411	0.838
SMOTE-Tomek/5	0.750	0.730	0.755	0.427	0.921	0.573	0.485	0.382	0.815
avg(SMOTE-Tomek)	0.77	0.73[c]	0.77[c]	0.45	0.92	0.55	0.51	0.41	0.83

[a] $|\text{TPR} - \text{TNR}| = |0.76 - 0.73| = 0.03$.
[b] $|\text{TPR} - \text{TNR}| = |0.78 - 0.73| = 0.05$.
[c] $|\text{TPR} - \text{TNR}| = |0.73 - 0.77| = 0.04$.

Table B.42: Case Study III, ≥ M flares, the next 24 h: cost function analysis results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
RandomForest/1	0.753	0.770	0.749	0.432	0.931	0.568	0.519	0.400	0.828
RandomForest/2	0.739	0.759	0.734	0.417	0.927	0.584	0.492	0.375	0.818
RandomForest/3	0.763	0.761	0.764	0.446	0.929	0.554	0.524	0.414	0.839
RandomForest/4	0.755	0.734	0.761	0.433	0.921	0.567	0.494	0.392	0.815
RandomForest/5	0.774	0.773	0.775	0.461	0.934	0.539	0.548	0.436	0.845
avg(RandomForest)	0.76[acc]	**0.76**[a,tpr]	0.76[a,tnr]	0.44[ppv]	**0.93**[npv]	0.56[far]	**0.52**[tss]	0.40[hss]	0.83

[a] |TPR − TNR| = |0.76 − 0.76| = 0. As the cost function output the lowest difference, the methodology has chosen it to proceed in the pipeline.
[acc] -0.08 ($p < 0.05$) – compared to model selection.
[tpr] +0.41 ($p < 0.05$) – compared to feature selection.
[tnr] -0.20 ($p < 0.05$) – compared to feature selection.
[ppv] -0.26 ($p < 0.05$) – compared to feature selection.
[npv] +0.07 ($p < 0.05$) – compared to feature selection.
[far] +0.26 ($p < 0.05$) – compared to feature selection.
[tss] +0.21 ($p < 0.05$) – compared to feature selection.
[hss] +0.02 ($p > 0.05$ and 0.1) – compared to feature selection.

Table B.43: Case Study III, ≥ M flares, the next 24 h: cut-off point adjustment results.

Model/Train.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
RandomForest/1	0.795	0.676	0.824	0.489	0.913	0.511	0.500	0.436	0.828
RandomForest/2	0.819	0.658	0.858	0.539	0.911	0.461	0.516	0.475	0.839
RandomForest/3	0.790	0.647	0.825	0.483	0.906	0.517	0.472	0.417	0.815
RandomForest/4	0.807	0.695	0.835	0.515	0.918	0.486	0.530	0.467	0.845
RandomForest/5	0.795	0.642	0.833	0.493	0.905	0.507	0.474	0.424	0.818
avg(RandomForest)	**0.80**[acc]	0.66[tpr]	**0.83**[tnr]	**0.50**[ppv]	0.91[npv]	**0.50**[far]	0.50[tss]	**0.44**[hss]	0.83

[acc] +0.04 ($p < 0.05$).
[tpr] -0.10 ($p < 0.05$).
[tnr] +0.07 ($p < 0.05$).
[ppv] +0.06 ($p < 0.05$).
[npv] -0.02 ($p < 0.05$).
[far] -0.06 ($p < 0.05$).
[tss] -0.02 ($p > 0.05$ and 0.1)
[hss] +0.06 ($p < 0.05$) – compared to feature selection.

Table B.44: Case Study III, ≥ M flares, the next 24 h: evaluation of validation sets.

Pred.Type/Val.Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
BaselineRandomForest/1	0.819	0.120	0.990	0.750	0.821	0.250	0.110	0.161	0.706
BaselineRandomForest/2	0.819	0.160	0.980	0.667	0.826	0.333	0.140	0.197	0.787
BaselineRandomForest/3	0.858	0.320	0.990	0.889	0.856	0.111	0.310	0.409	0.741
BaselineRandomForest/4	0.858	0.360	0.980	0.818	0.862	0.182	0.340	0.432	0.808
BaselineRandomForest/5	0.825	0.320	0.951	0.615	0.850	0.385	0.271	0.330	0.812
avg(BaselineRandomForest)	0.84	0.26	0.98	0.75	0.84	0.25	0.23	0.31	0.77
OptimizedRandomForest/1	0.756	0.640	0.784	0.421	0.899	0.579	0.424	0.355	0.769
OptimizedRandomForest/2	0.827	0.760	0.843	0.543	0.935	0.457	0.603	0.524	0.881
OptimizedRandomForest/3	0.748	0.600	0.784	0.405	0.889	0.595	0.384	0.325	0.782
OptimizedRandomForest/4	0.819	0.760	0.833	0.528	0.934	0.472	0.593	0.509	0.867
OptimizedRandomForest/5	0.818	0.680	0.852	0.531	0.915	0.469	0.532	0.481	0.814
avg(OptimizedRandomForest)	0.79[acc]	**0.69**[tpr]	0.82[tnr]	0.49[ppv]	**0.91**[npv]	0.51[far]	**0.51**[tss]	0.44[hss]	0.82

[acc] -0.05 ($p > 0.05$ and 0.1).
[tpr] +0.43 ($p < 0.05$).
[tnr] -0.16 ($p < 0.05$).
[ppv] -0.26 ($p < 0.05$).
[npv] +0.07 ($p < 0.05$).
[far] +0.26 ($p < 0.05$).
[tss] +0.28 ($p < 0.05$).
[hss] +0.13 ($p > 0.05$ and 0.1).

Table B.45: Case Study III, ≥ M flares, the next 24 h: evaluation of test sets.

Pred.Type/Test Set	*ACC*	*TPR*	*TNR*	*PPV*	*NPV*	*FAR*	*TSS*	*HSS*	*AUC*
BaselineRandomForest/1	0.907	0.500	1	1	0.897	0	0.500	0.620	0.836
BaselineRandomForest/2	0.814	0	1	0	0.814	0	0	0	0.884
BaselineRandomForest/3	0.881	0.375	1	1	0.872	0	0.375	0.493	0.809
BaselineRandomForest/4	0.881	0.500	0.970	0.800	0.892	0.200	0.471	0.549	0.925
BaselineRandomForest/5	0.878	0.375	1	1	0.868	0	0.375	0.491	0.839
avg(BaselineRandomForest)	0.87	0.35	0.99	0.76	0.87	0.04	0.34	0.43	0.86
OptimizedRandomForest/1	0.767	0.875	0.743	0.438	0.963	0.563	0.618	0.446	0.866
OptimizedRandomForest/2	0.883	0.875	0.886	0.636	0.969	0.364	0.761	0.665	0.911
OptimizedRandomForest/3	0.785	0.625	0.824	0.455	0.903	0.546	0.449	0.392	0.829
OptimizedRandomForest/4	0.881	0.875	0.882	0.636	0.968	0.364	0.757	0.662	0.965
OptimizedRandomForest/5	0.707	0.625	0.727	0.357	0.889	0.643	0.352	0.274	0.765
avg(OptimizedRandomForest)	0.81[acc]	**0.78**[tpr]	0.81[tnr]	0.50[ppv]	**0.94**[ppv]	0.50[far]	**0.59**[tss]	0.49[hss]	0.87[auc]

[acc] -0.06 (p > 0.05 and 0.1).
[tpr] +0.43 (p < 0.05).
[tnr] -0.18 (p < 0.05).
[ppv] -0.26 (p > 0.05 and 0.1).
[npv] +0.07 (p < 0.05).
[far] +0.49 (p < 0.05).
[tss] +0.25 (p < 0.1).
[hss] +0.06 (p > 0.05 and 0.1).
[auc] +0.01 (p > 0.05 and 0.1).

Appendix C

The Guaraci forecast system

We deployed the experimental models from case studies I and II into a real forecast environment providing daily forecasts at a fixed-cadence for up to three days ahead, namely the Guaraci[1] forecast system. We aim to evaluate such experimental models' performance in a true operational sense, as suggested in Chapter 7.

To provide forecasts, Guaraci regularly assembles data[2] from the repositories of NOAA/SWPC, namely the DSD and SRS data products. As NOAA/SWPC regularly issues data for DSD at 02:30 AM, 08:30 AM, 02:30 PM, and 08:30 PM UTC (NOAA/SWPC, 2011), and the SRS repository is daily updated always at 00:30 AM UTC (NOAA/SWPC, 2008), Guaraci runs daily at 01:00 AM UTC.

The system then gathers data from the SRS at 00:30 AM UTC – referring to the previous day – and from the DSD – previously issued at 08:30 PM UTC. To link between DSD and SRS data, Guaraci uses the compilation dates of records (month, day, and year).

Accordingly, we designed the Guaraci system under the layered architecture shown in Figure C.1. Overall, the Guaraci[3] comprehends an Ubuntu-based Amazon EC2 instance running a Python virtual machine prepared for machine learning, namely with the Scikit-learn[4], NumPy[5], Pandas[6], and imbalanced-learn[7] toolkits – the Artificial Intelligence (AI) layer.

[1] In the Guaraní mythology, Guaraci is the god of the Sun.
[2] Such data assembling involves the dynamic calculation of features as stated in Chapter 5.
[3] The Guaraci system is currently available at: `https://highpids.ft.unicamp.br/guaraci`.
[4] The Scikit-learn toolkit is available at: `https://scikit-learn.org/`.
[5] The NumPy toolkit is available at: `https://numpy.org/`.
[6] The Pandas toolkit is available at: `https://pandas.pydata.org/`.
[7] The imbalanced-learn toolkit is available at: `https://imbalanced-learn.readthedocs.io/en/stable/`.

The components of the AI layer process NOAA/SWPC data and feed the model layer prepared under an Apache Web Server – containing the data for system's forecasts, history, and graphics (i. e., XML and CSV data files). The system then renders such data in the view layer (i. e., HTML and CSS) supported by its PHP-based controller module.

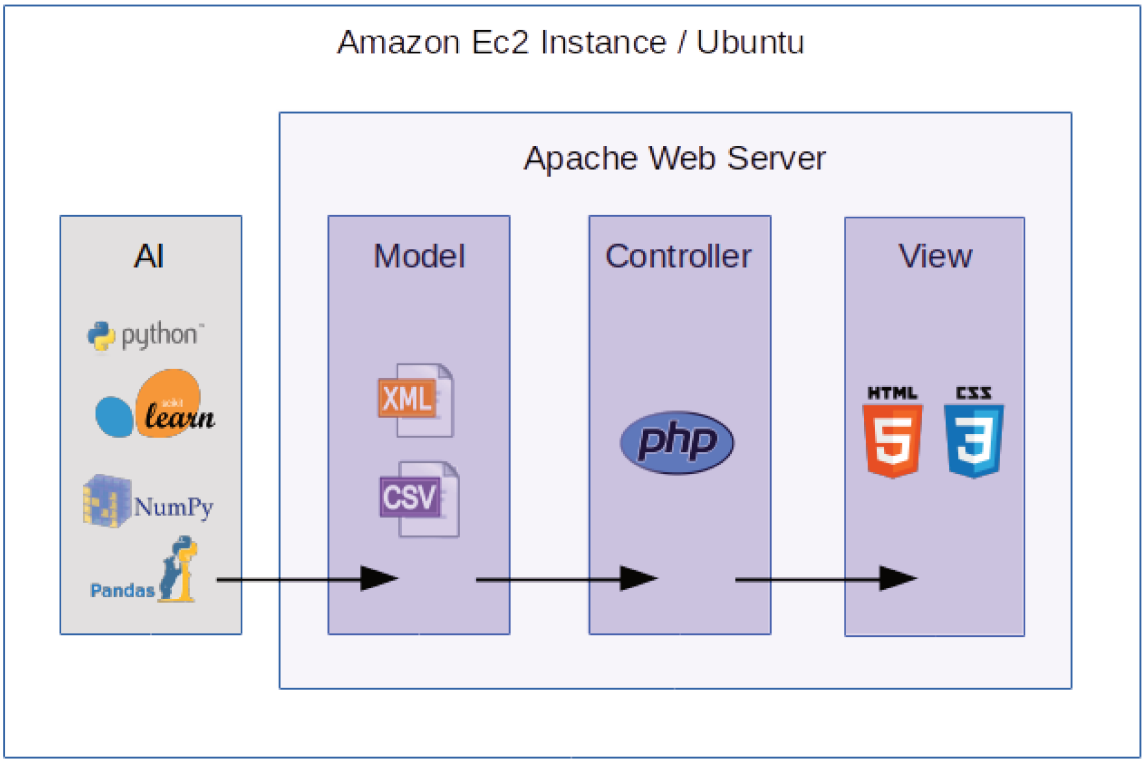

Figure C.1: Guaraci's architecture.

In agreement with the experimental models, Guaraci daily provides ≥ C- and M-class flare forecasts within the next 24, 48, and 72 h. To support the visualization of such forecasts, the system has a detailed interface holding the calculated probabilities for each horizon, as well as the suggested binary forecasts, drawn by the custom threshold adjusted during the methodology design. Figure C.2 shows this interface for flares higher than or equal to M-class.

Besides presenting detailed forecast reports daily, Guaraci also holds a feature for displaying the graphical probabilities of events in the last two weeks. However, in this graph, such probabilities are only displayed for the next 24 h along with a horizontal red line: the flare threshold. Accordingly, Figure C.3 shows this graph for flares higher than or equal to M-class. Once more, the tool also extends this feature to ≥ C-class events.

The last feature comprehending Guaraci is the full report of forecasts within 24 h not restricting to a defined period. This component refers to the table shown in Figure C.4. This

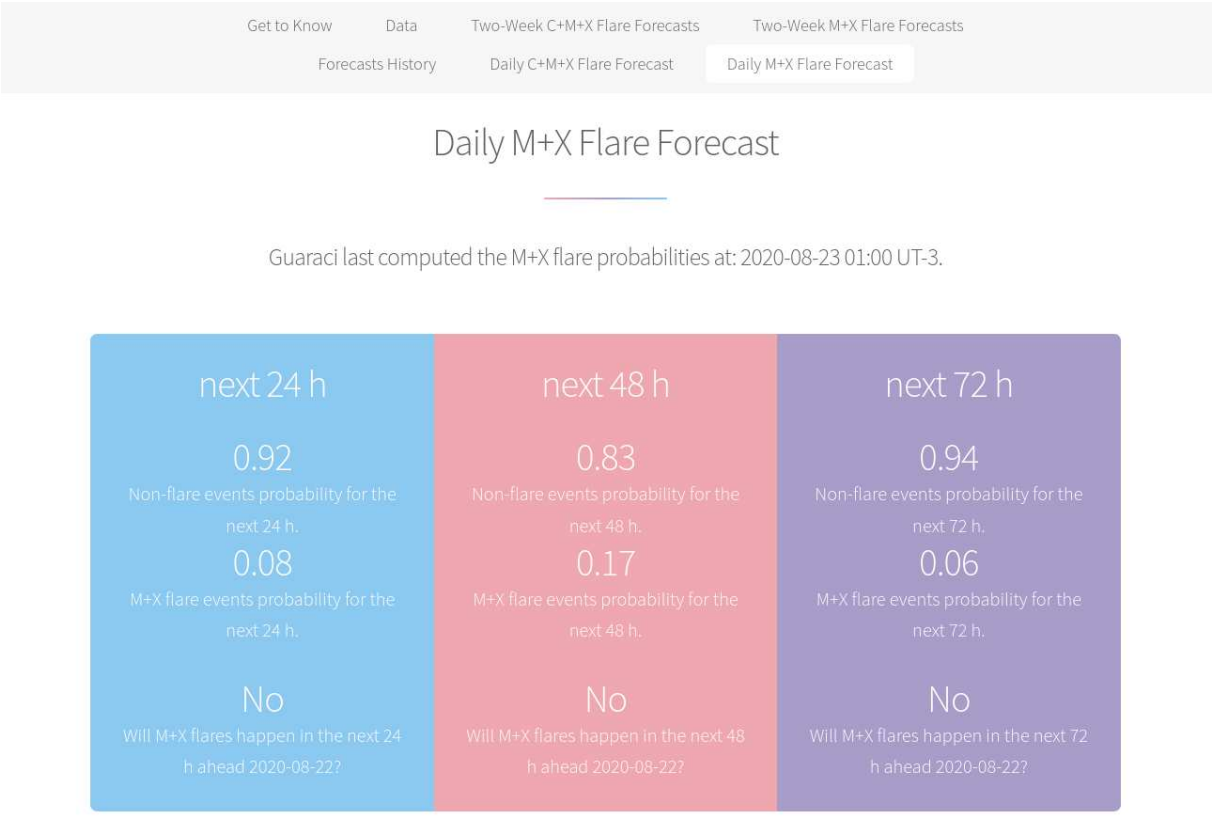

Figure C.2: Guaraci's daily M+X forecasts.

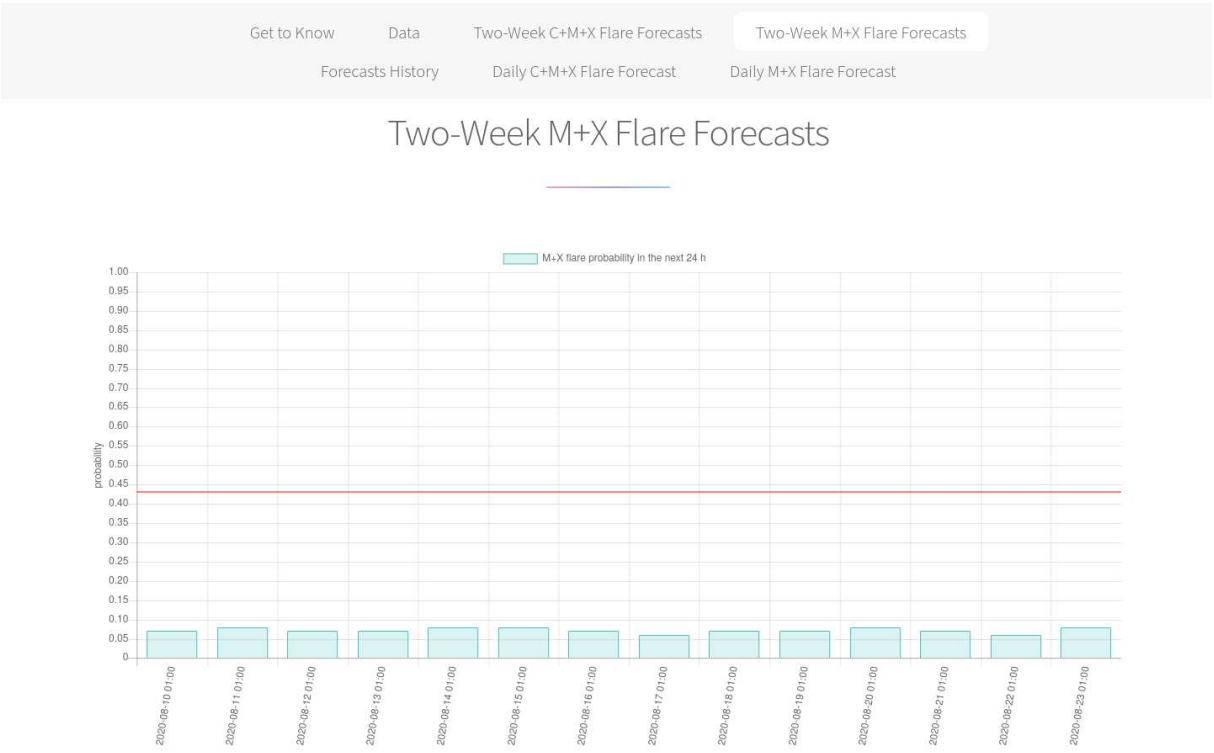

Figure C.3: Guaraci's two-week M+X flare forecasts.

table holds “True” and “Forecast” columns representing whether flares happened (NOAA/SWPC confirmations), and the probabilities of flares by Guaraci, respectively. Each new entry in this table always keeps the “True” column as “Not available” until the next 24 h period closes, when the system confirms with NOAA/SWPC whether some event occurred.

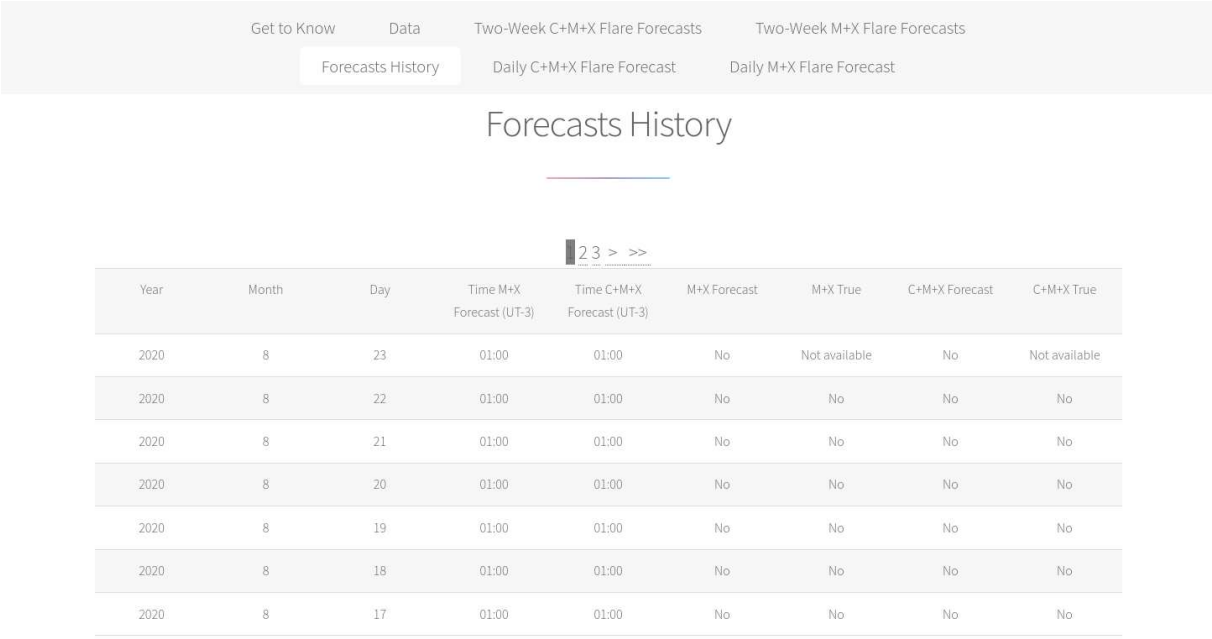

Figure C.4: Guaraci’s forecast history.

Finally, the full source code for implementing both forecasting models comprehending Guaraci is currently available at GitHub[8]. Not only Guaraci’s source code, but also the code for the automated methodology we turned available to the community in GitHub[9]. Opening both tool’s codes will let other researchers improve and use them with their own datasets and projects.

[8] https://github.com/tiagocinto/guaraci-forecast.
[9] https://github.com/tiagocinto/guaraci-toolkit.