This is a repository copy of *CME arrival modeling with machine learning*.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/210137/

Version: Published Version

Article:

Chierichini, S. orcid.org/0009-0005-6746-2917, Liu , J orcid.org/0000-0003-2569-1840, Korsós, M.B. orcid.org/0000-0002-0049-4798 et al. (2 more authors) (2024) CME arrival modeling with machine learning. The Astrophysical Journal, 963 (2). 121. ISSN 0004-637X

https://doi.org/10.3847/1538-4357/ad1cee

Reuse

This article is distributed under the terms of the Creative Commons Attribution (CC BY) licence. This licence allows you to distribute, remix, tweak, and build upon the work, even commercially, as long as you credit the authors for the original work. More information and the full terms of the licence here: https://creativecommons.org/licenses/

Takedown

If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk/

The Astrophysical Journal, 963:121 (12pp), 2024 March 10 https://doi.org/10.3847/1538-4357/ad1cee

© 2024. The Author(s). Published by the American Astronomical Society.
OPEN ACCESS

CME Arrival Modeling with Machine Learning

Simone Chierichini[1,2], Jiajia Liu (刘佳佳)[3,4], Marianna B. Korsós[5,6,7], Dario Del Moro[2], and Robertus Erdélyi[1,6,7]

[1] Solar Physics and Space Plasma Research Centre (SP2RC), School of Mathematics & Statistics, The University of Sheffield Sheffield S3 7RH, UK
[2] Department of Physics, University of Rome "Tor Vergata," Rome, Italy
[3] Deep Space Exploration Lab/School of Earth and Space Sciences University of Science and Technology of China, Hefei, 230026, People's Republic of China
[4] CAS Key Laboratory of Geospace Environment/CAS Center for Excellence in Comparative Planetology/Mengcheng National Geophysical Observatory, University of Science and Technology of China, Hefei, 230026, People's Republic of China
[5] Dipartimento di Fisica e Astronomia "Ettore Majorana," Università di Catania, Via S. Sofia 78, I-95123 Catania, Italy
[6] Department of Astronomy, Eötvös Loránd University, Pázmány Péter sétány 1/A, H-1112 Budapest, Hungary
[7] Gyula Bay Zoltán Solar Observatory (GSO), Hungarian Solar Physics Foundation (HSPF), Petőfi tér 3, H-5700 Gyula, Hungary

Received 2023 November 3; accepted 2024 January 7; published 2024 March 6

Abstract

Space weather phenomena have long captured the attention of the scientific community, and along with recent technological developments, the awareness that such phenomena can interfere with human activities on Earth has grown considerably. Coronal mass ejections (CMEs) are among the main drivers of space weather. Therefore, developing tools to provide information on their arrival at Earth's nearby space has become increasingly important. Liu et al. developed a tool, called CME Arrival Time Prediction Using Machine Learning Algorithms (CAT-PUMA), to obtain fast and accurate predictions of CME transit time. This present work aims at the expansion of the CAT-PUMA concept, employing supervised learning to obtain vital information about the arrival of CMEs at Earth. In this study, we report the results of our work following the implementation of supervised regression and classification models in the CAT-PUMA framework. We conducted a comparison of various machine learning models in the context of predicting the transit time of CMEs and classifying CMEs as either Earth impacting or non-impacting. In this way, we are able to provide information on the possibility of a CME reaching Earth relying on CME features and solar wind parameters measured at take-off. This application thus provides quantitative indications about the geoeffectiveness of these space weather events. While machine-learning models can demonstrate fairly strong performance in regression and classification tasks, it is not always straightforward to extrapolate their practical potential and real-world applicability. To address this challenge, we employed model interpretation techniques, specifically Shap values, to gain quantitative insights into the limitations that affect these models.

Unified Astronomy Thesaurus concepts: Solar coronal mass ejections (310); Space weather (2037)

1. Introduction

The subject of space weather is becoming increasingly important in today's high-tech-driven society as well as for the scientific community. A continuously increasing number of studies have shown the significance of the impact of space weather on Earth (Lanzerotti 2001; Schwenn 2006; Pulkkinen 2007; Temmer 2021; Cliver et al. 2022). The threats posed by space weather to our technosphere are varied, ranging from the interference of satellite communication to the disruption of power grids resulting in possible prolonged blackouts (Daglis 2001; Schrijver & Siscoe 2010; Riley et al. 2018; Pilipenko 2021).

Coronal mass ejections (CMEs) are the largest eruptive phenomena in the solar system, consisting of gigantic bubbles of plasma and magnetic field shot into interplanetary space at very high speeds (Low 2001; Chen 2011; Webb & Howard 2012; Gopalswamy 2016; Kilpua et al. 2017). CMEs are considered to be the major drivers of space weather. This status is attributed not only to their potential to impact the Earth's magnetosphere upon arrival but also because they are recognized for triggering secondary space weather phenomena, including solar energetic particles. (Temmer 2021; Whitman et al. 2022). For these reasons, the scientific community is significantly increasing its efforts to develop predictive tools for CMEs (Camporeale 2019; Vourlidas et al. 2019).

The study of the CME arrival at Earth primarily focuses on the prediction of the time of arrival (ToA) and speed of arrival (SoA) of CMEs. However, it is equally critical to have the capability to assess whether a CME is likely to reach Earth. All these aspects are for mitigating their impact on our technosphere.

The most common forecast methods are based on empirical, semi-empirical models of CME geometry with simplified physics (Cargill 2004; Vršnak et al. 2013) or are full MHD models (e.g., ENLIL or EUHFORIA Odstrcil 2003; Pomoell & Poedts 2018). Thanks to the major technological advances made in the last few decades, artificial intelligence, and in particular machine learning, have led to remarkable results in space weather research, particularly for the prediction of CMEs (Camporeale et al. 2018; Camporeale 2019). Liu et al. (2018) developed a model, called CME Arrival Time Prediction Using Machine Learning Algorithms (CAT-PUMA), which exploits the popular supervised learning model support vector machines (SVMs), to obtain accurate and fast predictions of the transit time of CMEs from Sun to Earth. CAT-PUMA inputs physical quantities of (i) CMEs at launch time (such as velocity, mass, angular width) and information on the state of the interplanetary medium, and (ii) the solar wind to obtain quick predictions of their arrival time at Earth. This model is able to

Original content from this work may be used under the terms of the Creative Commons Attribution 4.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.

generate accurate predictions with a mean absolute error (MAE) of 5.9 hr on a test set composed of 37 CME events. The use of deep learning has also proved feasible in predicting the arrival of CMEs at Earth. Wang et al. (2019) proposed a regression model based on a convolutional neural network (CNN) that takes white-light observations of CMEs as its only input. The main advantage of this model is that it is time-effective in feature collection, allowing predictions to be made from raw observations. Forecasts from CNN models show an MAE of about 12.4 hr, measured by relying on a *k*-fold cross-validation (CV) approach. The potential of deep learning was further explored by, e.g., Fu et al. (2021); they proposed a deep neural network architecture to obtain both geoeffectiveness and transit time predictions of the CME events. Briefly, Fu et al. used a deep residual network with attention layers to extract feature maps directly from white-light and EUV observations. In Fu et al. (2021), the term *geoeffective* refers to a CME that reaches Earth causing geomagnetic disturbances with an associated Disturbance Storm Time index of less than −30 nT. This model therefore first provides information on the hazard posed by the CME, and then provides a forecast of its arrival time (with an MAE of 5.8 hr). However, despite the above promising results in terms of performance, they are evaluated only on limited test samples. In a more recent study, Alobaid et al. (2022) conducted research where they compared various machine-learning models for the prediction of CME transit times. Their findings demonstrated the potential for enhanced prediction performance by employing an ensemble of different models. Notably, their results are noteworthy as the optimal combination of models yielded an MAE of less than 10 hr. It is important to emphasize that there are several factors to consider when evaluating the performance of a machine-learning model, and in particular, data is a key part of the learning process of such models. Furthermore, performance tends to decrease when test samples are larger. Vourlidas et al. (2019) compared the most recent forecasting methods and concluded that several observational and physical factors limit the potential of all approaches.

One of the main challenges in assessing the goodness of a machine-learning model is that it is not straightforward to compare its performance with that of other models. The reason is that the data sets used to train the models are often tailor-made to address a specific task, or to fit the needs of different models. This makes it difficult to generalize the results. Furthermore, there is relatively little data available on Earth-impacting CMEs to obtain reliable statistics, and test samples of dozens of CME events could be not sufficiently representative of the real-world scenario, consequently introducing bias due to the size of the data sets.

The idea that inspired this work is to investigate further the potential of a machine-learning approach to study the arrival of CMEs at Earth and also explore its limitations. To do so, we focus our attention on the model developed by Liu et al. (2018). A data-driven model such as CAT-PUMA offers several promising practical applications. It exploits only CME and solar wind features at the time of CME launch to obtain transit time predictions, without requiring large amounts of computing power, resulting in fast forecasts. At the same time, it bases its decisions on physical information, which makes the model easier to interpret than deep-learning models. In this work, we therefore pursue more than one objective.

First, there is the need to compile a comprehensive data set, including CME events up to 2022. Second, we aim to investigate the CAT-PUMA concept further, discussing its potential and limitations in more detail by having more data available for training. In addition, we develop a variant of the model that is able to address the problem of Earth-impacting CMEs and thus deliver predictions about their arrival at Earth. Finally, we apply an interpretation approach to the proposed models, in an attempt to extract as much information as possible from this study regarding the capabilities of machine learning in predicting the arrival of CMEs at Earth.

The structure of this paper is as follows. Section 2 describes the data set and the preprocessing procedure. In Section 3, we introduce the models and address briefly the different aspects of the training process in general. Finally, in Sections 4 and 5 we summarize, discuss, and assess the results.

2. Data Analysis

2.1. Data Set

In this section, we describe the procedure of data collection. The procedure we employed for compiling the Earth-impacting CMEs data set follows closely the one introduced in Liu et al. (2018). In essence, the data mining process is divided into two phases. First, we identify all the observed geoeffective CME events from 1996–2022, and then we associate each event with the features that will make up the input space of the machine-learning models. We used four different CME lists to collect events from the beginning of the Solar and Heliospheric Observatory (SOHO) era:

1. The Richardson and Cane list (Richardson & Cane 2010),
2. The full halo CMEs list provided by the University of Science and Technology of China (Shen et al. 2013),
3. The George Mason University CME/ICME list (Hess & Zhang 2017),
4. The CME Scoreboard (by NASA).[8]

Among other information, these catalogs report the time of first appearance of the CMEs in the field of view of the SOHO LASCO Coronagraph (onset time), and the starting of the geomagnetic storm associated with their interaction with the magnetosphere (arrival time).

With the aim of obtaining a clean data set, it is important to consider only those events that can be associated with a precise onset and arrival time; therefore, all ambiguous CME events (e.g., multiple CMEs) and those without an associated shock are excluded. In addition, duplicated events in the four source lists are identified and removed. Once the list of CME events that reached Earth is obtained, the next step is to associate their features and targets for training.

The input space is constructed using the CME features collected in the SOHO LASCO CME catalog, which contains estimates of physical quantities about the CMEs such as average velocity, velocity at 20 $R_\odot$, mass, angular width, and the measurement position angle. The features are associated with the collected events by matching the take-off time given here with the one recorded in the source lists. In a similar way, solar wind features are associated with the CMEs via OMNIWeb Plus data.

[8] The CME scoreboard can be found at https://kauai.ccmc.gsfc.nasa.gov/CMEscoreboard

The idea is to include solar wind plasma density (SW proton density), flow latitude (SW plasma flow latitude angle), flow longitude (SW plasma flow longitude angle), plasma-beta (SW plasma-beta), plasma pressure (SW pressure), speed, and plasma temperature (SW plasma speed and temperature), and the solar wind magnetic field components in Geocentric Solar Ecliptic system coordinates (B_x, B_y, B_z) to the input space allowing the model to encode solar wind state information. We also added the number of sunspots present on the solar surface (R) at the time of take-off, to also include information on the state of the solar cycle.

The solar wind quantities are taken on an average of 6 hr after the take-off time, since Liu et al. (2018) found this as the most suitable window for the forecast. Additionally, we discard CME events characterized by an angular width below 90°. Such CMEs are generally considered to be less likely to reach Earth and to produce disturbances on Earth. Once the data mining process is completed, the data set consists of 295 CME events each described by 17 features, five of them represent the CME physical characteristics while 12 encode solar wind state at the time of take-off.

In the supervised training framework, it is also necessary to provide the model with a training target. In this study, we employ supervised learning models to address two main topics:

1. Forecast transit time: This is a regression task, so each CME event is associated with the value of the *transit time* from the Sun to Earth, calculated as the difference between the take-off time and the arrival time. This information is already contained in the source lists mentioned above.
2. Forecast CME arrival at Earth: In the context of a machine-learning approach, this is a *supervised binary classification* task, where we will refer to Earth-impacting CMEs as *positive events* (i.e., belonging to the *positive* class), while non-Earth-impacting CMEs are *negative events* (i.e., *negative* class).

To train classification models in a machine-learning context we need both positive and negative events. The Earth-impacting CMEs serve as positive examples. To compose the list of negative events, i.e., CMEs that did not reach Earth, there is no standard method. The LASCO catalog contains thousands of observed events and only a small percentage end up impacting the Earth's magnetosphere. One way to build the negative class could be to consider events not included in the list of Earth-impacting CMEs as negative events.

However, this approach is too simplistic and would lead to a strong imbalance between the two classes, making classification very difficult. Naturally, the number of negative events is much higher than the number of positive ones, which is also reflected in our data set. To get around this problem, we apply filters to limit class imbalance and obtain a cleaner data set. First, we discard all events that show notes indicating them as *poor events*. This helps to keep the data set representative of the problem. Second, consistent with what we did for the positive events, we discard all events with an angular width inferior to 90°. The final version of the data set thus contains 295 labeled Earth-impacting CMEs and 3453 non-Earth-impacting CMEs.

2.2. Missing Values Problem

Before proceeding with the analysis, it is necessary to pay some extra attention to the handling of missing values. Indeed, among all the events that make up our data set, some have missing values in certain features. Not all supervised learning models can handle data with missing values, so it is essential to determine a way to impute them. In a machine-learning context, the handling of missing values is widely discussed, and there is more than one way to tackle the problem of missing data (Saar-Tsechansky & Provost 2007; García et al. 2015).

We, initially, preferred to eliminate all events with missing values from the data set, to avoid the introduction of biased data. This choice further reduced the number of Earth-impacting CMEs. Subsequently, to improve the size of the data, we decided to conduct a parallel analysis by generating a second version of the data set, including the previously discarded events appropriately imputed. We opted for unsupervised learning techniques, such as clustering, to replace missing values with values similar to neighboring feature space points. This procedure is applied by exploiting a well-known clustering algorithm, the *k*-nearest neighbors (KNN), using the `sci-kit learn` KNNImputer library.

The KNN is a supervised learning classifier that employs proximity to produce classifications or predictions about the grouping of a single data point. Although it may be applied to classification or regression issues, it is commonly employed as a classification method since it relies on the idea that comparable points lie close in the feature space (Bishop & Nasrabadi 2007; Kuhn & Johnson 2013, p.159–161, p. 124–7). The underlying concept of utilizing KNN for missing values imputation is that an input instance may be roughly estimated by the values of the points that are nearest to it, leveraging non-missing features. We thus obtained two versions of the data set:

1. *Data set v1*: This data set consists of 209 Earth-impacting CMEs and 2968 non-Earth-impacting CMEs.
2. *Data set v2*: The second data set (that we call the augmented version) consists of 295 Earth-impacting CMEs and 3453 non-Earth-impacting CMEs.

There are 17 features characterizing CME events in our data set, which makes the input space highly dimensional; however, they generally have different importance concerning the target to be predicted. In the next section, we take a closer look at the feature space.

2.3. Feature Selection

Before proceeding with the training of the models, it is crucial to analyze the feature space closely. Feature selection is essential because the input space is quite high-dimensional, and irrelevant features could adversely affect the performance of the models. The relevance of features is inspected using a function of the `sci-kit` learn Python package SelectKBest, which is equipped with several feature selection tools. We leverage the ANOVA (analysis of variance) F-Score and the so-called mutual information score between the features and the target in both cases, i.e., regression (f_regression,mutual_info_regression) and classification (f_classif, mutual_info_classif), respectively.

The F-score function ranks the features according to the linear separability of the class distributions in terms of variance, returning the ANOVA *F*-value for the features. The mutual information function, on the other hand, calculates the mutual information between the input variables and the target variable, thus capturing the nonlinear relationships between the

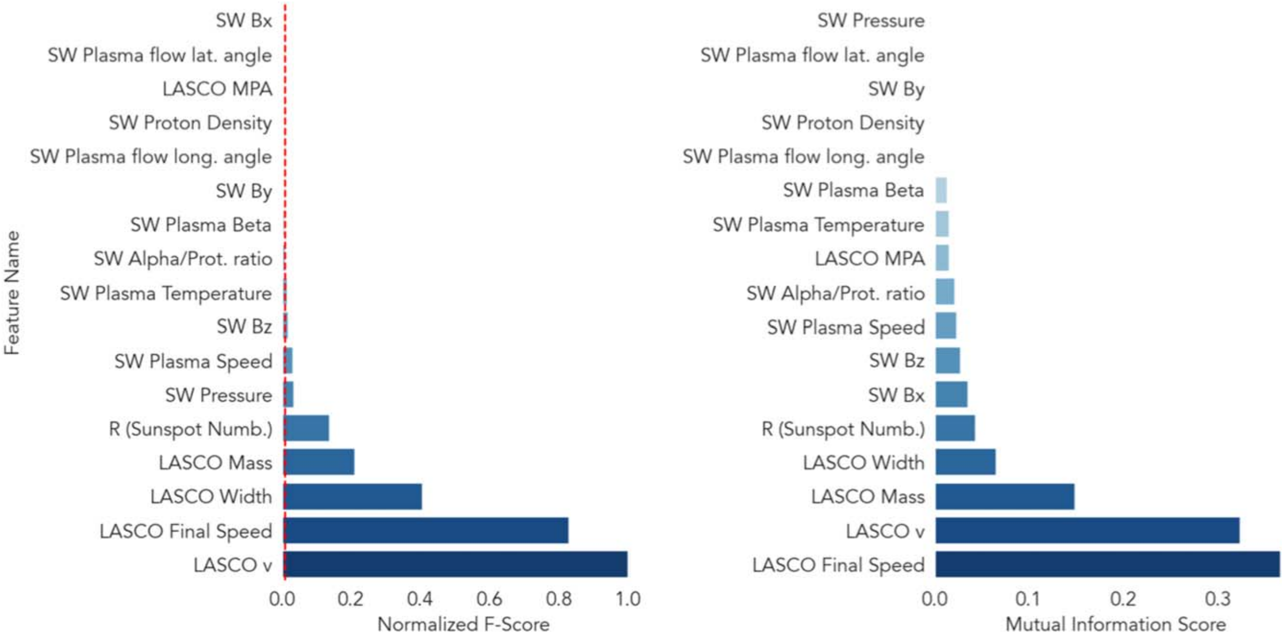

Figure 1. Bar plots of F-score (right), and mutual information score (left) of the features related to the Regression target, i.e., CME transit time.

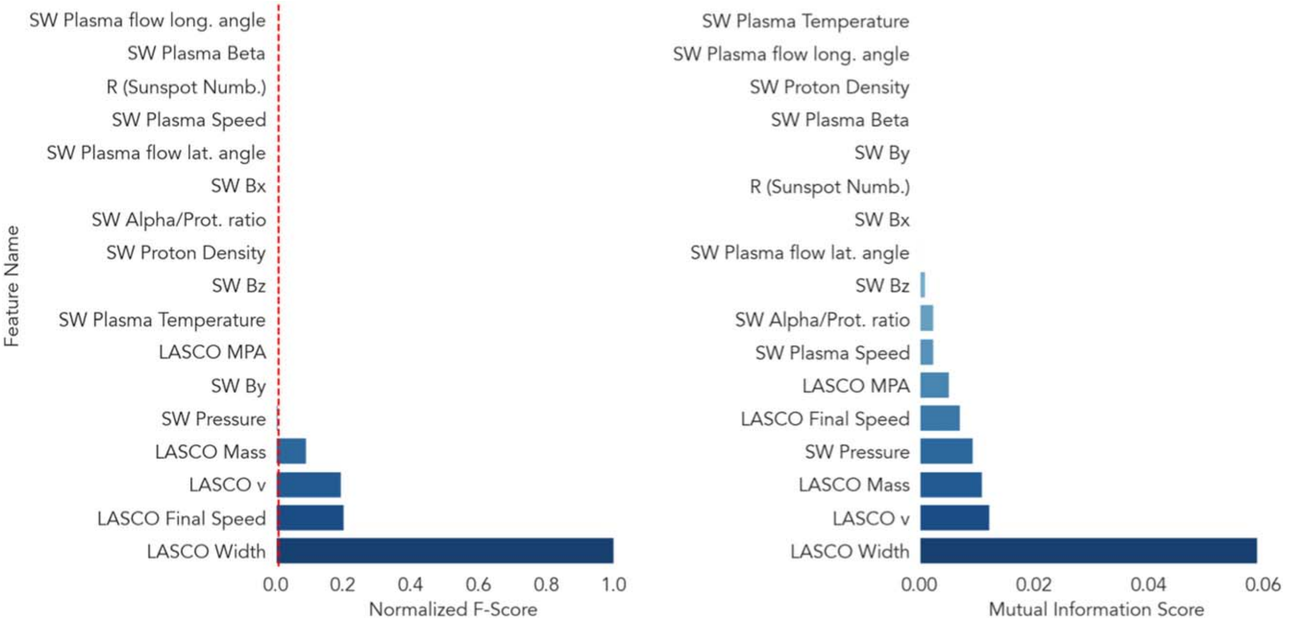

Figure 2. Bar plots of F-score (right), and mutual information score (left) of the features related to the classification label.

features and the target. Figures 1 and 2 show the F-score and mutual information ranking of the features, for the regression and classification cases, respectively. The bar plots in the figure show two different scenarios.

Figure 1 illustrates that the features with the highest degree of correlation with CME transit time are those related to CME velocity: CME average velocity (LASCO v) and final velocity, followed by the width and mass of the CMEs. Some features related to solar wind are also relevant, although to a lesser extent. Mutual information values are relatively low, indicating weak nonlinear relationships between the data and the target.

The classification data set shows similar results (Figure 2), but in this case, there are only four relevant features, all relating to the state of the CME at launch. We therefore adopt a consistent approach in both cases for feature selection that is in line with that used for CAT-PUMA, i.e., eliminate features with a normalized F-score value below 0.01. Feature selection allows the input space to be resized; the input space for the regression now consists of eight features, namely, both CME speed features, CME width, and mass, SW B_z, SW plasma temperature, SW plasma speed, SW pressure, and sunspot number R. On the other hand, the feature space for the classification only consists of CME average speed, CME final speed, and CME width and mass. The ranking of the features for the augmented version of the data set is very similar, the only difference being that the SW alpha/proton ratio takes the place of the B_z in the input space. Once feature selection is complete, the data set is ready for the model training phase, which will be described in detail in the next section.

3. Methods

In this work, we aim to study the application of the CAT-PUMA concept in two different machine-learning tasks to characterize the arrival of CMEs at Earth.

First, we are interested in obtaining an update on the regression model developed by Liu et al. (2018) that can output predictions on the CME transit time. A regression problem consists of the task of finding a map between a set of D-dimensional input vectors $\boldsymbol{x}$ into one or more continuous target variables y (Camporeale 2019). In our case, the aim is to find a function that is able to associate a CME event (encoded as a vector whose components are the features obtained in the mining process) with the appropriate transit time.

The second objective is to obtain a model that can predict the arrival of a CME at Earth. In a machine-learning context, this translates into a binary classification task. Classifiers answer the question: What class does an event belong to?

There are a variety of methods for solving both regression and classification problems that differ mainly in the assumptions one makes regarding the nonlinear function for mapping inputs to outputs. The CAT-PUMA model uses a well-known machine-learning algorithm, SVMs. In a nutshell, the SVM tool solves the regression linearly in a higher dimensional space than the one in which the problem is defined, using a kernel trick method.

Alongside SVMs, in this work, we also wish to study the problem with algorithms that combine the predictive capabilities of multiple models, often referred to as *ensemble models*. By the term ensemble model, we can refer to various techniques that exploit the combination of simple models (often called weak learners) to generate a more sophisticated model with better performance and generalization capability.

Bootstrap aggregating technique (bagging) makes use of the same training algorithm for every predictor, where each one is trained on a different random subset (chosen by sampling with replacement) of the training set. In bagging, the models that make up the ensemble operate in parallel, i.e., once all learners are trained, each on a slightly different version of the training set, the ensemble can make a prediction for a new instance by simply aggregating the predictions of all learners. In the case of a regression problem, the final output is typically the mean of all the outputs. This technique helps to reduce variance and minimize overfitting.

Another efficient way of constructing an ensemble is to combine models sequentially. Each weak learner is trained to correct the prediction errors made by the previous one. This technique is called *boosting*.

We then implemented two well-known and widely used ensemble model algorithms, the random forest and the extreme gradient boosting (XGBoost), which leverage the bagging and boosting techniques, respectively. Both of these algorithms use decision trees as weak learners. Decision trees are versatile machine-learning algorithms that can perform both classification and regression tasks and even multi-output tasks. Decision trees break down complex decision-making processes into a list of more straightforward decisions.

Much like a real tree, a decision tree consists of a root node (the first node in the decision tree), decision nodes (splitting points), and leaf nodes (endpoints with no further splitting). These models are constructed by partitioning the data set recursively into heterogeneous subsets on the basis of a function that takes into account the information carried by each attribute (a more detailed description of such machine-learning methods can be found in Bishop & Nasrabadi 2007; Yadav et al. 2020).

We used the models mentioned above, SVMs, random forest, and XGBoost, for regression and classification to explore their capabilities and better understand their limitations. The training process of a machine-learning model involves several steps, and for clarity, we will break down the discussion into different sections.

3.1. Training

The training process is crucial in the design of a machine-learning model and involves three types of variables.

First, it is crucial to collect good *input data*, i.e., a collection of instances characterized by the features necessary for the machine-learning problem under study. Input data are not directly part of the training, but they are crucial for configuring the model to make predictions on similar data.

Second, there are the parameters that the model uses to adjust the predictions to the data. These are typically called model *parameters*.

When we speak of training a supervised learning model, we typically refer to an optimization process that aims to minimize a cost function that quantifies the model's ability to generate predictions close to actual values. A model's parameters are the core of the model and are set during training. The last type of variables are those that configure the architecture of the model and govern the training process itself, called *hyperparameters*. The hyperparameters are usually fixed during the training process, unlike the model parameters, which are modified to adjust to the data. Choosing the optimal set of hyperparameters for the machine-learning problem under study (be it regression or classification) is essential to maximize the model's performance.

3.1.1. Validation

There are several methods to establish the capability of a machine-learning model to accomplish a given task. The simplest method is to divide the available data into two subgroups, the *training set* with which to train the model and the *test set* to measure its performance. In this way, the test set represents a sample of events that the model has *never seen* and thus a test bed for assessing learning ability. The train/test splitting is typically done randomly, to limit bias.

CAT-PUMA does use such a validation method, however, with an extra addition. Once the SVM model has been optimized, it selects the train/test split that returns the best performance score among 10^6 random splits. This choice follows the idea of selecting the split that ensures the highest representations of the training data in the test set. We will refer to this method as best-split validation (BSV). However, BSV may be affected by bias due to limited data availability. In fact, the resulting best split could lead to an overly optimistic evaluation. Therefore, we chose to flank this evaluation method with a more conservative and less optimistic one widely used in machine learning, i.e., the *k-fold cross-validation method* (k-fold CV; Refaeilzadeh et al. 2009; Yadav & Shukla 2016).

Instead of performing a single train/test split, the k-fold CV consists of dividing the data set into k subsets and keeping one out of the training as a validation set. The model is trained on the remaining $k-1$, and the validation set is used to evaluate performance. The procedure is then repeated by iterative changing the validation set. More in detail, the k-fold CV method involves the following steps:

1. A value is selected for the set of hyperparameters.
2. The training set is divided into k subsets, $k-1$ of which are used to train a model, and the remaining one (the validation set) is used to evaluate performance. The performance score average is stored.
3. The previous step is repeated using all k subsets as validation sets. This allows the model to be trained with the same hyperparameters set and then evaluate the performance on k different subsets.
4. A new value for the set of hyperparameters is selected and the process is repeated from step 1.

This method provides the advantage of training the model on k different training sets and evaluating its performance on as many test sets, resulting in more robust results.

3.1.2. Hyperparameter Tuning

The tuning process consists of exploring the space of hyperparameters in search of those that return the best performance. Nevertheless, there are several ways to explore the space of hyperparameters, such as grid search or random search (Bergstra & Bengio 2012).

Here, optimization was carried out using a sequential model-based optimization approach, in particular the tree-structured Parzen estimator. This approach was implemented using the Optuna optimization library (Akiba et al. 2019), and allows the hyperparameter space to be inspected based on the process history, thus focusing the search on the areas where performance is at its highest (more information on this approach can be found in Bergstra et al. 2011). This method helps to significantly shorten the optimization process by minimizing the time spent sampling suboptimal areas of the hyperparameter space.

The tuning process is carried out by means of a fivefold CV for both regression and classification tasks. In particular for the latter, in order to keep the training consistent with the unbalanced data, the fivefold CV is performed in a *stratified* manner, so that the validation and training subsets reflect the proportions of the overall data set.

4. Results

In this section, we describe the results obtained for the present work. We set out to train three different machine-learning models and use them for two distinct tasks: regression and classification.

1. The regression models provide an answer to the question: How long do CMEs take to reach Earth?
2. The classification models generate predictions as to whether a CME will reach Earth or not.

We studied various models systematically for both problems in this analysis. Each model is optimized to address the relevant machine-learning problem at hand and then we analyze the performance by comparing different evaluation metrics. For clarity, we will first focus on the regression problem, followed by addressing the classification problem.

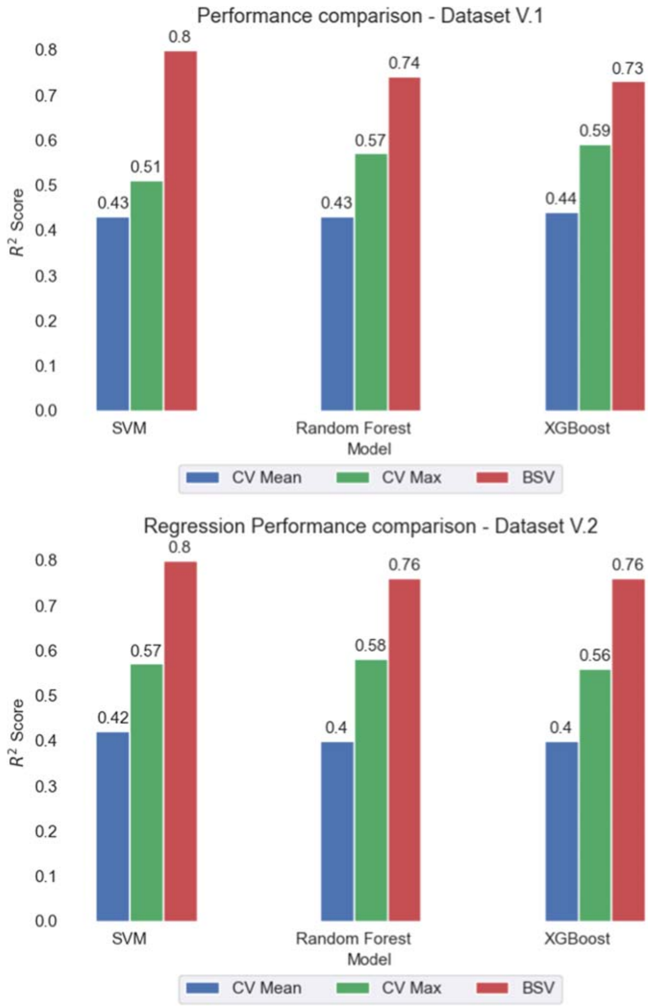

Figure 3. Performance scores for regression models. Performance comparison by means of CV mean, CV max, and best-split scores for data set v1 (top) and data set v2 (bottom), respectively.

4.1. Regression

The training follows the same steps for all three models:

1. After the feature selection procedure (described in Section 2.3) we extract eight relevant features for predicting the transit time; four CME features and four SW state features.
2. Each model, SVM, random forest, and XGBoost are optimized by means of k-fold CV, with $k=5$ (Section 3.1.1).
3. Once the optimized models are obtained, we evaluate their performance through CV and BSV (Section 3.1.1), using the R^2 score as a reference metric.

This procedure is applied to both versions of the data set; the first consists of 209 Earth-impacting CMEs, while the second version contains 295 CMEs, 86 properly imputed as described in Section 2.2.

Figure 3 summarizes the results obtained using different validation methods. We report the average value (blue) and the maximum value (green) obtained by a fivefold CV, and also the BSV score (red).

CV is a more conservative method than BSV, as mentioned in Section 3.1.1, which puts the spotlight on the best train/test split. This is evident in the figure; the best-split score is the highest for any model-data set combination.

Furthermore, it is essential to point out that for ensemble techniques, BSV is less effective, returning a lower value than SVM. The reason is probably to be found in the architecture of the models. Ensemble models can better generalize predictions and not fit too closely to the specific training set used for training. This makes it more difficult to find a training and test pair that performs dramatically better than a random split.

Nevertheless, the ensemble models also achieve fairly high performance, with a BSV score ranging from 0.73–0.76.

The results show a significant difference between the performance according to the BSV score and the CV score. The CV mean scores are similar for all models but are still considerably low compared to the CV max values. To understand it better, this means that of the five different random train/test splits for CV, the most optimistic one returns a considerably higher R^2 score than the average. This is true for both versions of the data set, underlining the difficulty in characterizing a model capable of generalizing the regression problem well. We obtain the best performance from the SVM; the BS validation technique achieves an R^2 score of 0.80, and the related MAE is 7.6 ± 5.2 hr. Although the MAE is higher than in the original version of CAT-PUMA, this result is still reasonably good, considering that the test set includes more events. However, for CV the MAE remains above 10 hr.

It is important to stress the concept; although one can obtain a very high-performing model through BSV, it does not necessarily maintain such high performance on new samples.

4.2. Classification

Next, let us devote the study to the point of whether machine-learning models are capable of predicting whether a CME will reach Earth.

The feature selection process (Section 2.3) reveals that within a classification framework, only four features exhibit the highest correlation with the target variable. These features are LASCO width, final speed, average velocity, and the CME's mass at the time of launch. Again, we opted to test different models on two versions of the data set. Data set v1 includes 2968 CME events, of which 209 are positive (i.e., Earth-impacting CMEs). The augmented version, on the other hand, consists of 3543 CME events, of which 295 are Earth-impacting. For the classification problem, we adopted a more standard validation method. Before training, we divided the data set into training (80% of the total) and test (20% of the total); we optimized and trained the models on the training set and then evaluated the performance on the test.

Given the highly unbalanced nature of the problem, it is even more challenging to determine whether and how well a classifier succeeds in solving the problem under analysis. For this reason, we decided to compare several performance evaluation metrics to extrapolate a wider spectrum of information about models' capabilities.

Table 1 summarizes the results, comparing the values of some relevant metrics to assess the goodness of the classification. There is much information to extrapolate from the results obtained. First, it is important to emphasize that the performance of the different models is comparable and the score values are generally better for the augmented data set version (data set v2).

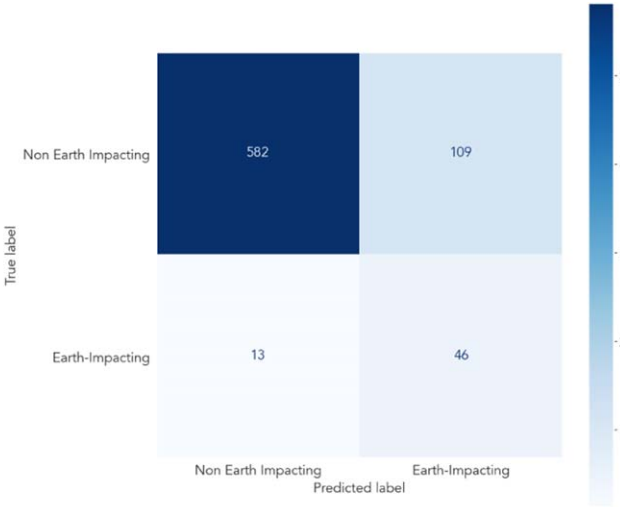

Figure 4. Confusion matrix for the test set for the random forest model, trained on the augmented data set version (data set v2). Matrix entries are TP (bottom right), TN (top left), FP (top left), and FN (bottom left).

Table 1
Table Presenting the Metric Scores for All the Models Employed in the Work for Both Versions of the Data Set (Data Sets v1 and v2)

Metric	SVM	Random Forest	XGBoost
Accuracy	0.76|0.77	0.82|**0.84**	0.73|0.78
Precision	0.19|0.24	0.24|**0.30**	0.18|0.24
Recall	0.71|**0.85**	0.67|0.78	0.79|0.81
Balanced accuracy	0.58|0.61	0.60|**0.64**	0.58|0.61
False alarm ratio	0.81|0.76	0.76|**0.70**	0.81|0.76
		Data set v1|data set v2	

Note. Bold text highlights the best performance scores.

Accuracy is higher than 70% in all scenarios. However, the accuracy value is not an optimal indicator of the model's goodness because it is affected by the unbalance of the classes. The balanced accuracy value gives a more realistic interpretation of the classifiers' ability to assign the correct class to each instance, never exceeding a value of 65%.

In general, the models show an excellent ability to recognize events in the majority class while lacking precision for the minority class, resulting in a high false alarm ratio. Precision is generally very low, reaching a maximum value of 30% for random forests. Nevertheless, the recall is generally fairly high, indicating the ability of the models to obtain reliable forecasts for non-Earth-impacting CMEs.

It is now essential to go into detail on this topic because there is usually a tendency to confuse model performance, which inevitably depends heavily on the type of validation chosen, with the actual capabilities of the model.

For the sake of clarity, we provide the confusion matrix for the random forest in Figure 4.

The precision score encodes the following information; among 155 events predicted as Earth impacting, only 46 are correctly classified. Low precision directly implies a high false alarm ratio. Despite this, the model still shows potential for operational application because of the high Recall. In fact, of 686 events labeled as Earth impacting, only 13 are predicted incorrectly.

4.3. Interpretation of Results

One of the main challenges leveled at prediction tools based on machine-learning algorithms is that it is difficult to judge their actual capabilities and limitations because there is often no way of getting a sense of the process that drives the models to produce a specific prediction.

In addition, hard-to-interpret models such as deep neural networks and gradient-boosting machines are increasingly efficient and now outperform in most cases linear models that are typically easier to interpret. The main consequence of the lack of interpretation is the distrust in the model.

The subject of model interpretation has been widely discussed in recent years and various methods have emerged to try to better understand the results obtained by artificial intelligence. Local explanation methods aim to assess the influence of input variables/features on a specific prediction/output.

In this paper, we employ one of these tools, called *Shapley values* (Lundberg & Lee 2017), to gain more insights into model decisions. Shapley values are model-agnostic local explanation markers originated in the field of game theory to determine the payouts of players depending on their contribution to the total payout (Aas et al. 2021). In an artificial intelligence explanation setting, this method is used to calculate the contribution of each feature to the final output. In particular, this technique allows us to decompose the output of a model $f(\bar{\boldsymbol{x}})$, where $\bar{\boldsymbol{x}}$ is a specific feature vector, into the sum of the contributions ϕ of each feature:

$$f(\bar{\boldsymbol{x}}) = \phi_0 + \sum_{i=1}^{F} \phi_i. \tag{1}$$

Considering a set of F features and a subset $S \subseteq F = \{1,\ldots,F\}$ consisting of $|S|$ features; the Shapley value related to feature j can be expressed as

$$\begin{aligned} \phi_j(v) = \phi_j = \\ = \sum_{S \subseteq F} \frac{|S|!(F - |S| - 1)!}{F!}(c(S \cup \{j\}) - c(S)) \\ j = 1,\ldots,F, \end{aligned}$$

where $c(S)$ is the contribution function that maps subsets of features to the contribution they have on the prediction. Such function is typically the expected output of the model, conditional on the feature vector $\mathbf{x}_S$:

$$c(S) = E[f(\boldsymbol{x})|\boldsymbol{x}_S = \bar{\boldsymbol{x}}]. \tag{2}$$

In essence, the Shapley values determine the difference in the contribution that features j bring to the prediction if included in a specific subset S, and average this over every possible combination of possible subsets S of features in terms of the contribution function: c (subset S including feature j) $-$ c (subset S without feature j).

In this work, we used the python package https://shap.readthedocs.io/en/latest/SHAP (SHapley Additive exPlanations) to apply the theory of Shapley values to the predictions made for CMEs and try to obtain some more information on the feature space of the CAT-PUMA framework. Since we tested different machine-learning models, we opted to deal in more detail with the cases where performance is highest, to determine whether there are patterns that characterize the best-performing models. Let us now start by treating the regression case and then discuss the classification task, as it was done with the description of the results.

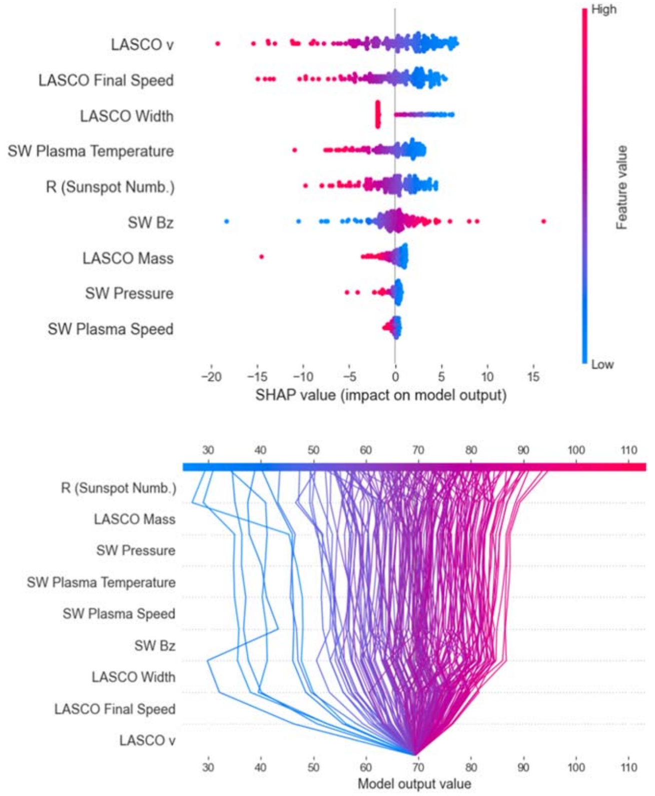

Figure 5. ((a), top) SHAP summary plot for the training set. The *y*-axis ranks the features sorted from the most (top) to least (bottom) important. The *x*-axis depicts the SHAP value. Each point refers to a specific instance of the training set, pointing out the related SHAP value associated with a value of a certain feature. The color bar displays whether the feature value is high (pink) or low (blue). ((b), bottom) SHAP decision plot for the training set. This plot shows the decision path for each instance in the training set. Each line shows each feature's contribution (*y*-axis) to the final output of the model (*x*-axis). The color depends on the magnitude of the output and ranges from blue for lower output values to red for higher ones.

4.3.1. Regression

For the regression case, we considered the SVM model trained on the data set v1. One of the main tools offered by the SHAP algorithm is the summary plot in Figure 5(a), which shows for each feature the SHAP values of all instances in the training set. This plot contains a wealth of information about the predictions made by the model. Let us break down the main ones.

First of all, the features on the *y*-axis are ordered in ascending order (from bottom to top) according to the average contribution they have on the predictions. This means that, according to SHAP, the feature with the greatest influence on the predictor output is the average speed of CMEs followed by angular width, final speed, and sunspot number R, while the least influential features are the SW plasma speed and pressure.

In addition, SHAP values are typically higher for low feature values and lower (negative range) for high feature values; this is true for all features, especially speed features, except for SW B_z. In practice, very high feature values tend to push the model predictions toward lower ToA. Trivially, if the speed of the CME is very high, the model will tend to opt for low ToA estimates.

Another convenient way of obtaining information on the model's decision-making process is the decision plot (Figure 5b). This visualization helps to understand the decision path that the model takes for each instance.

In the training set, the graph shows the contribution a feature has on the final output for each instance. The paths are clustered by similarity, this allows similar decision patterns to be identified. Two different macro-patterns can be distinguished; the first relates to most instances and mainly involves output values higher than 50 hr, while the second refers to low ToA predictions.

For all instances, the LASCO speed and width-related features influence the prediction the most. All other features have a lower impact, and at the top of the cascade, the sunspot number R tends the push toward the final output of the model. The instances associated with lower predictions (ToA <50 hr) appear to be largely conditioned by the velocity value of the CMEs at launch time; this suggests that if the initial velocity of the CMEs is very high, the model is likely to generate lower ToA predictions.

Furthermore, the decision pattern for low ToA predictions appears less stable, there are a couple of cases where B_z and LASCO mass values drive the predictions considerably toward higher or lower output, respectively. This is interesting because, in fact, there are relatively few examples of CMEs associated with a very low ToA (<40 hr); this might suggest that due to the few examples available, the model appears to rely more on speed features to make decisions about lower outputs. This is because the correlation between the ToA and the speed of the CMEs is higher, and it is therefore easier to establish a relationship with the few examples available.

Moreover, SHAP being a local technique, is valuable to inspect decisions on individual instances. Waterfall plots of the instances with the highest and lowest prediction error are shown in Figure 6. Such plots can clearly and compactly display the relative contributions of the different features in order of importance. The least performing instance has a recorded arrival time of 108 hr.

In Figure 6 (top), we see how almost all the features overwhelm the output toward very high ToA values, though fail to reach the actual value, which is still very high compared to the average value. This effect is probably still due to a poor representation of rare events in the training set, as we do not have many examples of such slow CMEs in our data set. In contrast, the best-performing CME is associated with a ToA much closer to the mean value. Figure 6 (bottom) shows that almost all features hold the prediction value close to the base value. The sunspot number R has the most significant contribution by pushing the prediction very close to the actual value, resulting in an error of only 0.5 hr.

4.3.2. Classification

In this section, we delve into the decision-making process leading to the predictions in the classification task; in particular, it is interesting to exploit the SHAP values to find insights as to why the false alarm ratio (FAR) remains so high. For this purpose, we analyze the predictions made on the test set by the best-performing model, the random forest trained on the v2 data set. The model outputs are values between 0 and 1, and instances are associated with the positive or negative class by identifying a threshold value, usually 0.5; thus, samples with an output greater than the threshold value are associated with the positive class (i.e., Earth-impacting CMEs). Otherwise, the prediction is negative (i.e., not Earth impacting). The output score also indicates how confident the model is in making decisions. The closer the output value is to the threshold value, the more uncertain the decision possibly is.

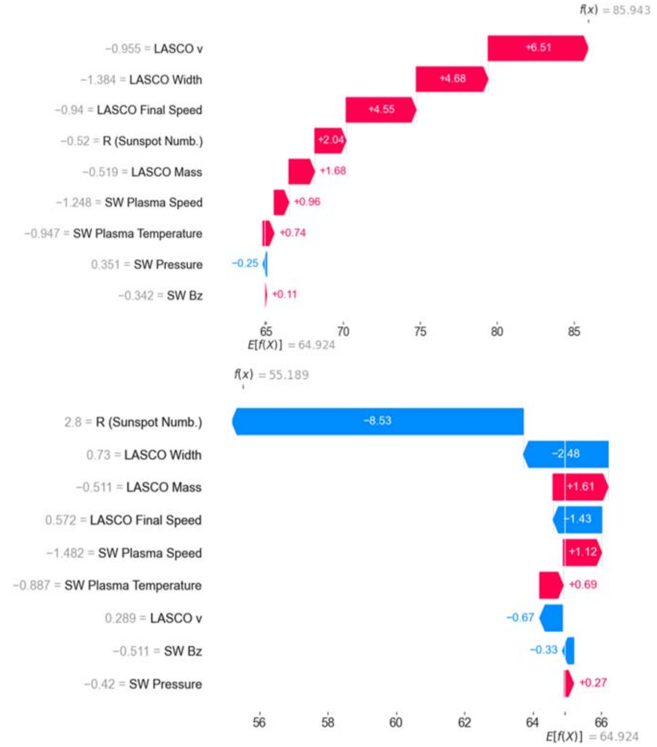

Figure 6. Waterfall plot related to the best ((a), top) and worst ((b), bottom) performing CMEs.The plot shows the relative contribution of each feature to the model's prediction $f(x)$, starting from the base value $E[f(x)]$. The y-axis shows the features and their value (scaled for training), while the x-axis represents the ToA. The arrows display the SHAP value associated with each feature, colored red if positive and blue if negative.

Figure 7(a) (top) shows the classification confidence for the CMEs in the test set; the histogram suggests that for most of the misclassified CMEs, the model decision was made with confidence of less than 0.7; in contrast, correctly classified CMEs typically have very high confidence, in most cases greater than 0.8. This suggests that despite its high FAR, the model is relatively confident when making a correct decision, while it is generally less secure when it makes incorrect predictions. This result is reassuring because it suggests that the model learns the difference between Earth-impacting and non-Earth-impacting CMEs. The SHAP method allows the decision-making process of the model to be analyzed instance by instance.

Figure 7(b) (bottom) shows the decision plot for the misclassified test set events. The decision plot highlights some interesting aspects. First of all, we notice two main decision patterns; in blue are the CMEs assigned to the negative class and in red those assigned to the positive class. There are also

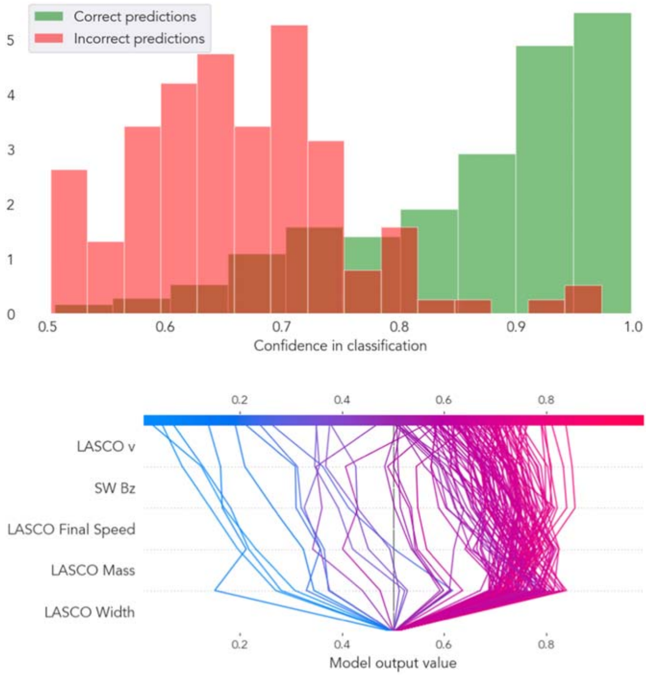

Figure 7. Interpretability plots for the classification task. The graphs refer to the test set for the best-performing classifier (random forest trained on data set v2). ((a), top) Histogram of the classification confidence distribution. Red highlights the misclassified instances, while green highlights the correct predictions. ((b), bottom) Decision plot for the misclassified instances. This panel shows the decision patterns; the color bar indicates the magnitude of the output; in blue are highlighted those instances for which the model returns values close to zero (assigned to the negative class). Red marks the positive class. Examples related to values close to the base value (i.e., the threshold value) are purple.

some instances in which the model associates an output value very close to the base value, i.e., close to the threshold value. The latter shows a more uncertain decision pattern, with some features pushing them toward higher values while others lowering their output value; the result is an output that settles close to the threshold value. Instances with an output greater than the threshold value, thus assigned to the positive class, contribute to the high FAR. For those instances, the graph shows that the feature that most influences the decision is the LASCO width, which pushes the prediction toward high values. However, the other features tend to lower the output value by pushing back the output, making the decision of the model less secure. This is interesting because it suggests that, in these cases, the model is principally *confused* by the value of the width of the CMEs; most of the misclassified events are halo CMEs (LASCO width = 360°). Considering the instances incorrectly assigned to the negative class, even though their count is quite low, the decision plot indicates that the model typically exhibits a high level of confidence in its decisions. This is evident as most features tend to steer the model's output toward values close to zero, despite the limited number of misclassified instances in this scenario.

5. Discussion and Conclusions

Machine learning has promising applications in space weather research. In this work, we aimed to study the CAT-PUMA concept in more depth, by exploring the potential of machine learning in obtaining predictions about the arrival of CMEs on Earth.

In addition to its potential, this work also highlights some of its limitations. One of the most critical issues is the amount of data available and its quality. The limited number of examples of Earth-impacting CMEs makes it challenging to characterize the problem.

As far as the regression problem is concerned, this limitation results in a weak ability of the models to generalize the predictive power; the results show that there are cases in which the training set represents well the test set, and this leads to fairly high performance. On the other hand, CV scores emphasize that in general, this is not always the case.

Moreover, depending solely on BSV for model selection carries the potential risk of creating a tool where the training and test data are carefully selected to minimize errors. This could result in the model's inability to achieve predictions with the same level of reliability when applied to new, unseen data. Certainly, the quality of the data undoubtedly has a significant impact on the performance of the models. The data used to represent CMEs in the input space are frequently rough approximations and may not provide a comprehensive representation of the problem we aim to address.

The results from the classifiers highlight the challenge posed by the imbalanced distribution of Earth-impacting examples, which makes it difficult to achieve reliable predictions of CMEs reaching Earth. Although the models exhibit strong predictive performance for the majority class, making only a few errors in determining when a CME will not impact Earth, the FAR for Earth-impacting CMEs does not drop below 70%, even in the best-case scenarios. Fu et al. (2021) highlight the same problem, even though they achieve excellent transit time prediction performance in a deep-learning framework, the FAR related to the arrival of CMEs on Earth remains very high (81%).

The problem of high FAR is also shared by other models that do not base their forecasts on machine-learning techniques. Vourlidas et al. (2019) compared, among other things, the success ratio and false alarm ratio of different models using MHD-based approaches to approximate the state of the solar wind, such as the WSA-ENLIL+Cone model. The random forest shows results in line with those reported in Vourlidas et al. (2019), although performance remains lower than those shown for data sets of comparable size to ours. Nevertheless, there is room for improvement.

It is essential to remember that the CME events considered for the classification task are only described by their speed, mass, and angular width. The number of factors influencing the interplanetary travel of CMEs to Earth is high, and it is challenging to find a model that can take this into account because of the limited means we have of characterizing the state of the solar wind in interplanetary space. However, it is clear that this is crucial for improving forecasts.

CAT-PUMA attempts to encode this information by equipping the feature space with indicators that attempt to encode the solar wind state at the time of CME lift-off. However, the solar wind data are calculated at the L1 position on a 6 hr average before the CMEs take off. This approximation is probably too strong to characterize the interplanetary medium well. Feature selection and SHAP values indicate that such features have little influence on the decision-making process and must probably be more accurate to better address the classification problem. The study on the interpretability of model decisions revealed interesting insights into the possible factors limiting this approach.

Overall, the results obtained with the SHAP values suggest that models typically tend to rely more on the most relevant features (for both cases studied) to produce predictions. Models learn to associate, for example, very fast CMEs with short transit times, and vice versa. Thus, in the case where a CME is recorded with very high speeds, the rest of the feature space fails to influence the decision sufficiently to obtain longer transit time predictions. The same is true for binary classification. SHAP values show that decisions are largely influenced by the angular width of CMEs, and since the data set is characterized by geoeffective events that are mostly halo CMEs, most of the errors are due to cases where halo CMEs do not reach Earth. An approach to improve the models might involve incorporating features that more effectively encapsulate key elements of interplanetary travel, including factors like the propagation direction and additional details about the characteristics of CMEs, beyond just their angular width. Furthermore, as demonstrated by Guastavino et al. (2023) in their study, the integration of physical information into the model's architecture has been proven to enhance predictive performance and overall robustness. The next step envisaged in research may therefore attempt to expand the input space of CMEs. One feasible way could be to combine derived features, such as those used in CAT-PUMA, with models capable of extracting information directly from white-light or EUV images. The work conducted by Alobaid et al. (2022) has, in a sense, initiated this journey by introducing an ensemble model comprising machine-learning models designed for tabular data processing and a CNN equipped to handle image data. Their findings demonstrate that combining models and various types of input significantly contributes to enhancing prediction performance. Deep-learning models may have the capability to reduce the preprocessing time required to fit the data to the feature space of the models by using images directly as input (Wang et al. 2019; Fu et al. 2021). Nevertheless, it is important to note that deep-learning models may face challenges if an insufficient amount of data is available to effectively characterize the problem. Exploring the use of pre-trained models could be a compelling approach to mitigate data-related challenges. Leveraging pre-trained models allows for harnessing the capabilities of established feature extraction methods, enhancing the power of physical feature extraction in the absence of extensive data. Furthermore, it could be intriguing to employ interpretation methods to probe into the model's decision process and determine the most critical regions within the coronagraph images that influence the model's decisions.

Acknowledgments

This research work is part of the Space Weather Awareness Training NETwork (SWATNet) project. SWATNet has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant Agreement No 955620. This research was also carried out in the framework of the CAESAR project, supported by the Italian Space Agency and the National Institute of Astrophysics through the ASI-INAF n.2020-35-HH.0 agreement for the development of the ASPIS prototype of the scientific data center for Space Weather. This research has received financial support from the European Union's Horizon 2020 research and innovation program under grant agreement No. 824135 (SOLARNET). M.B.K. acknowledges support by the Università degli Studi di Catania (PIA.CE.RI. 2020-2022 Linea 2), by the Italian MIUR-PRIN grant 2017APKP7T, CAESAR ASI-INAF n. 2020-35-HH.0 project and ÚNKP-23-4-II-ELTE-107, ELTE Hungary. J.L. acknowledges support from the National Natural Science Foundation of China. R.E. is grateful to STFC (UK, grant No. ST/M000826/1), NKFIH OTKA (Hungary, grant No. K142987), and the ISSI-Beijing program "Step forward in solar flare and coronal mass ejection (CME) forecasting" for the support received. D. D.M. is grateful to the Italian Space Weather Community (SWICo).

ORCID iDs

Simone Chierichini https://orcid.org/0009-0005-6746-2917
Jiajia Liu (刘佳佳) https://orcid.org/0000-0003-2569-1840
Marianna B. Korsós https://orcid.org/0000-0002-0049-4798
Dario Del Moro https://orcid.org/0000-0003-2500-5054
Robertus Erdélyi https://orcid.org/0000-0003-3439-4127

References

Aas, K., Jullum, M., & Løland, A. 2021, J. Artif. Intell., 298, 103502
Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. 2019, in Proc. 25th ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining (New York: ACM), 2623
Alobaid, K. A., Abduallah, Y., Wang, J. T., et al. 2022, FrASS, 9, 1013345
Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. 2011, in Proc. 24th Int. Conf. on Neural Information Processing Systems, 24 (Red Hook: Curran Associates, Inc.), 2546
Bergstra, J., & Bengio, Y. 2012, J. Mach. Learn. Res., 13, 281
Bishop, C. M., & Nasrabadi, N. M. 2007, JEI, 16, 049901
Camporeale, E. 2019, SpWea, 17, 1166
Camporeale, E., Wing, S., & Johnson, J. 2018, Machine Learning Techniques for Space Weather (Amsterdam: Elsevier)
Cargill, P. J. 2004, SoPh, 221, 135
Chen, P. 2011, LRSP, 8, 1
Cliver, E. W., Schrijver, C. J., Shibata, K., & Usoskin, I. G. 2022, LRSP, 19, 2
Daglis, I. A. 2001, in Proc. NATO Advanced Study Institute on Space Storms and Space Weather Hazards, ed. I. A. Daglis (New York: Kluwer), 1
Fu, H., Zheng, Y., Ye, Y., et al. 2021, RemS, 13, 1738
García, S., Luengo, J., Herrera, F., et al. 2015, Data Preprocessing in Data Mining, 72 (Berlin: Springer), 59
Gopalswamy, N. 2016, GSL, 3, 1
Guastavino, S., Candiani, V., Bemporad, A., et al. 2023, ApJ, 954, 151
Hess, P., & Zhang, J. 2017, SoPh, 292, 80
Kilpua, E., Koskinen, H. E., & Pulkkinen, T. I. 2017, LRSP, 14, 1
Kuhn, M., & Johnson, K. 2013, Applied Predictive Modeling (Berlin: Springer)
Lanzerotti, L. J. 2001, GMS, 125, 11
Liu, J., Ye, Y., Shen, C., Wang, Y., & Erdélyi, R. 2018, ApJ, 855, 109
Low, B. 2001, JGR, 106, 25141
Lundberg, S. M., & Lee, S.-I. 2017, in Proc. 31st International Conference on Neural Information Processing Systems (New York: ACM), 4768
Odstrcil, D. 2003, AdSpR, 32, 497
Pilipenko, V. 2021, STP, 7, 68
Pomoell, J., & Poedts, S. 2018, JSWSC, 8, A35
Pulkkinen, T. 2007, LRSP, 4, 1
Refaeilzadeh, P., Tang, L., & Liu, H. 2009, in Cross-Validation, ed. L. Liu & M. T. Özsu (Boston, MA: Springer), 532
Richardson, I. G., & Cane, H. V. 2010, SoPh, 264, 189
Riley, P., Baker, D., Liu, Y. D., et al. 2018, SSRv, 214, 1
Saar-Tsechansky, M., & Provost, F. 2007, J. Mach. Learn. Res., 8, 1625
Schrijver, C. J., & Siscoe, G. L. 2010, Heliophysics: Space Storms and Radiation: Causes and Effects (Cambridge: Cambridge Univ. Press)
Schwenn, R. 2006, LRSP, 3, 1
Shen, C., Wang, Y., Pan, Z., et al. 2013, JGRA, 118, 6858
Temmer, M. 2021, LRSP, 18, 4
Vourlidas, A., Patsourakos, S., & Savani, N. 2019, RSPTA, 377, 20180096

Vršnak, B., Žic, T., Vrbanec, D., et al. 2013, SoPh, 285, 295
Wang, Y., Liu, J., Jiang, Y., & Erdélyi, R. 2019, ApJ, 881, 15
Webb, D. F., & Howard, T. A. 2012, LRSP, 9, 1
Whitman, K., Egeland, R., Richardson, I. G., et al. 2022, AdSpR , 72, 5161
Yadav, S., & Shukla, S. 2016, in 2016 IEEE 6th Int. Conf. on Advanced Computing (IACC) (Piscataway, NJ: IEEE), 78
Yadav, S. P., Mahato, D. P., & Linh, N. T. D. 2020, Distributed Artificial Intelligence: A Modern Approach (Boca Raton, FL: CRC Press)