Solar Flare Prediction Using Two-tier Ensemble with Deep Learning and Gradient Boosting Machine

Chau Pham
Data Scientist
R&D Lab, Zalo Group
Ho Chi Minh, Vietnam
chaupm.cs@gmail.com

Vung Pham
Computer Science Department
Texas Tech University
Lubbock, TX 79409, USA
vung.pham@ttu.edu

Tommy Dang
Computer Science Department
Texas Tech University
Lubbock, TX 79409, USA
tommy.dang@ttu.edu

***Abstract*—This paper describes a machine learning approach to the solar flare prediction competition, a track in IEEE Big Data 2019 Big Data Cup. The competition task is to predict whether or not there is a solar flare event basing on a given time series of solar magnetic field parameters. Our method involves exploring and constructing data-driven machine learning models for the classification task of two imbalanced class labels from time series. Specifically, the investigated models include boosting, logistic regression, multilayer perceptron neural network, and long short-term memory neural network. These models have been successfully deployed and combined in an ensemble framework with two tiers in our final proposed solution for this competition. Our proposed approach ranked at the second place in the competition (the first on the private board and the eleventh on the public board).**

***Index Terms*—Gradient boosting decision tree, Logistic regression, Multilayer perceptron neural network, Long Short-term memory neural network, Ensemble framework, Solar flare prediction, multivariate time series.**

I. Introduction

Solar flares and Coronal Mass Ejections (CMEs) [2] may affect technology-dependent society, such as having negative impacts on space equipment, power grids, and high-frequency radio communication. Many of these fields are critical to security and economic vitality [32]. Therefore, many ongoing research and development efforts are striving to predict solar eruptive activities and then to perform timely actions to mitigate their negative impacts. Recent advancements in the machine learning field suggest that it has strong potential in resolving complex tasks using a data-driven approach. This paper proposes a novel data-driven method in response to the *Solar Flare Prediction from Time Series of Solar Magnetic Field Parameters*, a track in the IEEE Big Data Cup 2019 [12].

In this paper, we approach this challenge by first exploring relevant machine learning models for this specific task. We then combine the investigated models into an ensemble framework with two tiers. Specifically, the first tier contains four explored machine learning models, namely boosting, logistic regression, multilayer perceptron neural network, and long short-term memory neural network. Stacked on top of the first tier, the second tier collects the outputs from the models in the first tier, analyzes, and produces the final prediction. More information about the tiers is provided in Section V. Also, the given data is divided into three sets as training, validation, and submission sets. The training and validation sets are used to explore the models and the final configurations. The submission set is used to generate the prediction results for this IEEE Big Data 2019 competition.

The rest of this paper is organized as follows. Section II briefly describes the data and evaluation metric of this competition. This section is followed by Section III, which discusses the current, data-driven approaches to this specific task in the literature. Next, Section IV uses visualization techniques to explore and analyze the given data. After getting a good understanding of the input data, Section V details our approach concerning how we explore and combine the strengths of machine learning models in our final framework. This section also reviews the results of individual models, as well as the combined solution. Finally, we give conclusions and future directions in Section VII.

II. Data and evaluation metric

The dataset for this competition stemmed from Spaceweather HMI (Helioseismic and Magnetic Imager) Active Region Patch (SHARP) [4]. Data Mining Lab at Georgia State University processed it considerably and made the resulted dataset available for this competition [13]. The main task of the competition is to predict whether or not there would be an M-class or an X-class flare event [14] within the next twenty-four hours given observations as a multivariate time series. Each time series contains 25 space-weather variables over 60 steps observed at 12-minute intervals. A critical aspect of this dataset making it realistic and challenging is the imbalance of the two predicting classes (1 as flaring and 0 otherwise). The data can be briefly described as the following equations:

$$Input : (x_1, x_2, ..., x_N)_1, ..., (x_1, x_2, ..., x_N)_T$$

$$Output : \begin{cases} 0, & \text{if non-flaring} \\ 1, & \text{if flaring} \end{cases}$$

where $N = 25$ is the number of space-weather variables, and $T = 60$ is the number of observed time steps prior to the predicted output. One single set of inputs with a corresponding output class is called an *object*. The object has the format as follows:

- **id** - an identifier for the object,
- **classNum** - class identifier for the object,
- **values** - an array time series of shape 60×25.

The dataset is a set of partially overlapping sliced samples pulled from larger multi-variate time series. It is then separated into some smaller, temporally disjoint parts to mitigate overfitting and sampling bias. Specifically, the competition data is divided into three folds sampled from May 2010 to March 2012 (*fold 1*), March 2012 to October 2013 (*fold 2*), and October 2013 to March 2014 (*fold 3*) correspondingly. The sample sizes and ratios of class 1 to total data for these three partitions are as follows:

- *fold 1*: 76,773 objects; *class 1* accounting for 16.39%
- *fold 2*: 92,481 objects; *class 1* accounting for 15.10%
- *fold 3*: 27,006 objects; *class 1* accounting for 17.66%

There is one additional fold for the submission sampled from March 2014 up to August 2018 [11]. This submission fold is further divided into a public fold (for ranking on the public board) and a private fold (for ranking on the private board). The two sampling periods for these two folds are non-overlapping and are from March 2014 up to March 2015 and from March 2015 until August 2018. However, by the time of this competition, it was unknown which fold is public or private. Furthermore, the submission fold contains 173,512 objects of unknown class labels (0 or 1). However, it is observable from the above three folds and is confirmed by the competition data description details [13] that it is safe to assume that percentages of *class 1* are relatively similar across all folds.

Due to imbalanced binary classification nature, the evaluation metric for this competition is the F_1 score [17] of the predicted results. The F_1 score is the harmonic mean of precision and recall, thus helps to balance these two quantities. It is defined as:

$$F_1 = \frac{2}{recall^{-1} + precision^{-1}} = 2 \times \frac{precision \times recall}{precision + recall}$$

III. Literature review

Most of the recent approaches to this solar flare prediction problem are data-driven [18]. These approaches can be grouped into two main categories. They are linear statistical and non-linear statistical approaches. The typical works of the first approach include correlation-based methods, which study the relationship between the observed data and the flare events [8], [20] and linear discriminant analysis (LDA) for classification [26]. The latter, non-linear methods, is based on logistic regressions [44], relevance vector machine [1], support vector machine (SVM) [5], [35], [41], cascade-correlation neural networks (CCNN) [41], k-nearest neighbors (k-NN) [35], or extremely randomized tree (ERT) [35].

A recent research [18] suggested to reduce the large number of dimensions in this dataset by six statistical summaries (min, max, standard deviation, skew, mean, and median) to present each time series before applying the k-NN classification method for the prediction task. These statistical values are used to consider the impact of the time component in the dataset. This work also suggests that TOTUSJH is a useful variable as it achieves maximum mean True Skill Statistic (TSS) [3]. Similarly, there are works in the literature that exploit the significance of individual variables in the classification process. For instance, Ma et al. [29] suggests that USFLUX and TOTUSJZ are worthy variables for this task. On the other hand, based on domain knowledge, maximum R_VALUE can be used as a filter on non-flaring regions [43]. Understanding the technical descriptions of these variables is not required to follow the data-driven approaches to this problem described in this section. However, interested readers can refer to the competition site [13] for the names and technical details of the solar magnetic field parameters.

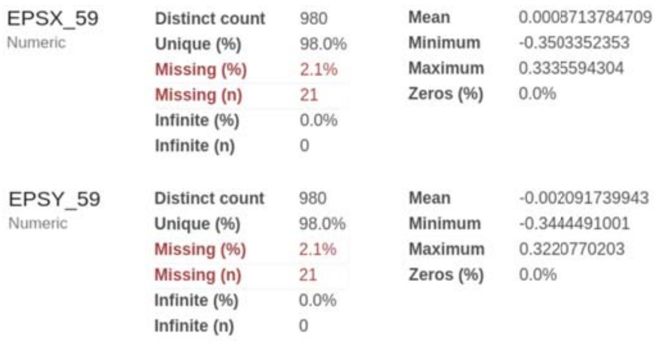
Multivariate time series data is growing in terms size and application domains [10] - cybersecurity [36], environmental monitoring [39], abnormality detection [37], social media topic evolution [9], [33], [38] to name but a few. These real-world applications generate multiple variables across time to produce high-quality, reliable, and statistically sound information. The datasets for this competition also involve multivariate time series data. Thus, our solution also takes time and sequences into account.

Long Short-Term Memory (LSTM) [19] neural networks produce current successes in various application domains, which involve multivariate time series data, from predicting the saturated thickness of aquifers [34] to efficient manufacturing and system operations [25]. There are also recent works [6] in the literature using LSTM to predict solar flare events. Experimental results from these works show that LSTM neural network outperforms related machine-learning methods in the solar flare prediction problem [27]. Therefore, the first tier in our ensemble framework also includes the LSTM model to analyze the temporal information from the datasets.

As discussed, there are different approaches to this solar flare prediction problem with different models and results. Thus, it is natural that there were many efforts in the literature, which follow the hybrid and ensemble approaches to deal with this research problem. In general, these techniques combine a set of predictions from different models to improve the prediction accuracy. The predicted results could merely be taking the average of multiple models, or using more sophisticated techniques for data assimilation. Interested readers can refer to [31] for a useful and thorough review of related works in this direction to space weather forecasting tasks. We also adopt this research approach in our method. To the best of our knowledge, this is the first method that ensembles the LSTM, logistic regression, multilayer perceptron (MLP) neural network [16], and boosting (LightGBM [21]) techniques in a two-tier structure. We discuss the details of our approach in Section V. We first provide an oveview of the data for this *Solar Flare Prediction from Time Series of Solar Magnetic Field Parameters* challenge.

IV. Data exploration

The first step toward tackling this challenge is to get a sense of how the data looks like. Thus, we use *pandas profiling*[1] for data exploratory process. This library helps to generate a comprehensive statistical report for the data set, including basic data type information, descriptive statistics (e.g., min, max, mean, average), quantile statistics, and other useful statistics. Figure 1 shows some worth noting statistics for two sample variables of the dataset (the 59^{th} time step of EPSX and EPSY variables). It's worth noting from this quick profiling step that there are missing values, and different variables have different distribution shapes and/or domain ranges.



EPSX_59				
Numeric	Distinct count	980	Mean	0.0008713784709
	Unique (%)	98.0%	Minimum	-0.3503352353
	Missing (%)	2.1%	Maximum	0.3335594304
	Missing (n)	21	Zeros (%)	0.0%
	Infinite (%)	0.0%		
	Infinite (n)	0		

EPSY_59				
Numeric	Distinct count	980	Mean	-0.002091739943
	Unique (%)	98.0%	Minimum	-0.3444491001
	Missing (%)	2.1%	Maximum	0.3220770203
	Missing (n)	21	Zeros (%)	0.0%
	Infinite (%)	0.0%		
	Infinite (n)	0		

Fig. 1. Sample overview of two variables (59^{th} time step of EPSX and EPSY) from the dataset using descriptive statistics.

Missing values are another critical aspect of the data exploration step. Missing values hurt the accuracy of our prediction models if we do not handle these with care. Hence, we also explore the percentages of missing values for the 25 variables from the competition dataset. The boxplot in Figure 2 depicts the distribution of the portions of missing values for these 25 magnetic field parameters. In particular, most of the variables contain about 0.6% of missing values. The maximum of the missing percentages are approximately 2.2%. These significant numbers suggest the need to devise strategies for handling missing values in our solution.

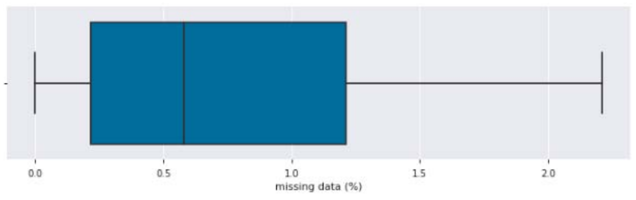


Fig. 2. Distribution of the percentages of missing values for the 25 variables in the dataset. Most of the variables have about 0.6% of missing values, and the maximum percentage of missing values for an individual variable is approximately 2.2%.

Furthermore, there are also suggestions from domain experts and related work in the literature that some raw features (provided by the competition data) or generated features (derived from the provided data, e.g., using statistical formulas) are more important than the others. For instance, Figure 3 shows that the maximum value of the variable R_VALUE, *MAX(data.R_VALUE)*, and the maximum value of USFLUX variable, *MAX(data.USFLUX)*, from each time series help in separating the two class labels (i.e., 0 vs. 1) reasonably well. This observation suggests us to analyze and explore the significance of different raw and generated features in our solution.

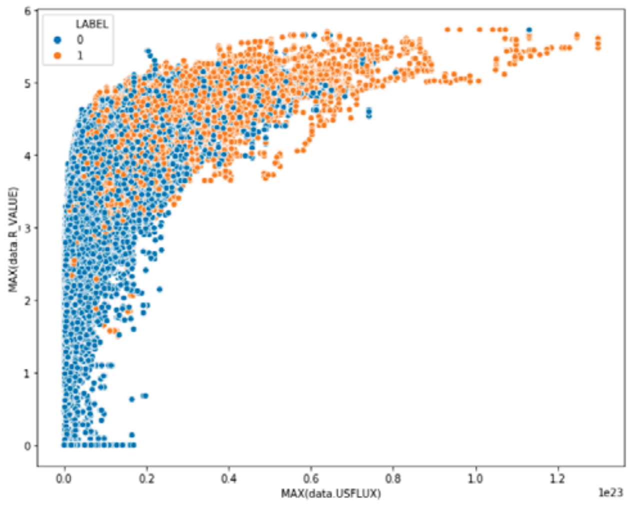


Fig. 3. Maximum of R_VALUE (*MAX(data.R_VALUE)*), and maximum of USFLUX (*MAX(data.USFLUX)*) are two promising variables in classifying solar flare events: orange for flaring and blue for non-flaring.

From the previous exploration, it is tempting to visualize and examine the differences of the solar magnetic field parameters over time for class-0 objects versus those for class-1 objects. The differences in values of an individual parameter of class 1 versus class 0 indicate its potential uses in classifying the two classes. However, this task is challenging due to a large number of objects to be visualized. Therefore, we sample *1,000* objects of class 0 and another *1,000* objects of class 1 from the training set. The sample size (i.e., *1,000*) is large enough to assure that the sampled objects are representative of the two classes.

After having the samples, the next step is to select a visualization solution that enbles us to visualize such a large amount of data. Line-graph is often used to visualize time series data because it can represent the value differences well [39]. However, for a large number of time-series objects, line-graph representation induces visual cluttering issues. Heat-map is one potential solution to solve this issue. It represents data in the form of rows and columns. The rows represent the variables, and the columns represent time steps. The cell color represents the value of a variable of a specific data entry at a specified time-step. However, in this case, the large numbers of samples (*1,000*) and time steps (*60*) requires a large amount of cells (*60,000*) to be visualized. These extreme number

[1]https://pandas-profiling.github.io/pandas-profiling/docs/

of visual elements make it impractical to use the traditional heatmap to represent the data.

Therefore, we devise a type of heatmap called *Continuous Heatmap* for this task. It is an approach for abstracting a large number of time series data in the form of continuous heatmaps, which significantly reduce the rendering time compared to the conventional heatmap. *Continuous Heatmap* first orders the time-series by their similarities. It then groups proximity cells with similar values into blobs. Instead of visualizing every individual cell, *Continuous Heatmap* visualizes the data by a smaller number of blobs. This approach helps in rendering a large amount of time series, thus enables us to explore their overall patterns. Besides allowing us to render a large amount of time series, our continuous visualization approach also provides a high level of generalization in visual encoding. Therefore, it allows the user to see the overview of the data rather than looking at individual details. The ability to provide quick and general views is helpful at the data exploration stage.

Figure 4 shows the *Continuous Heatmap* visualizations for *1,000* class-0 objects (on the left) and the same number of class-1 objects (on the right) for four example variables (i.e., TOTUSJH, TOTUSJZ, MEANJZH, and EPSX). The time series from each of the variable across two classes are combined and scaled linearly to the range [0, 1] before splitting back to their corresponding set of object types. This scaling step helps to show the value differences of an individual variable across the two classes and relative changes among variables. The color scale is shown at the top-left corner: darker green for low values and darker red for high values. Notice that the vertical orders of objects are different for different *Continuous Heatmaps*. That is, the vertical orders of objects are to optimize the continuous areas in the individual heatmap to reduce the number of blobs and hence to minimize the rendering time.

It is observable that there are apparent differences in value distributions of TOTUSJH and TOTUSJZ between class-0 and class-1 objects. Also, the differences are relatively consistent among objects. Specifically, most of them are lower in class 1 compared to class 0. On the other hand, the differences between the MEANJZH parameters for the two object labels are not significant. Furthermore, there are also changes in values of EPSX in the two classes, but they are not consistent among objects (some are higher, while the others are lower). These also confirm that some variables (e.g., TOTUSJH and TOTUSJZ) are more utilizable in the classification task [29] compared to the others (e.g., MEANJZH and EPSX) as discussed in our Related work section.

One additional observation which is shared among all *Continuous Heatmap* charts is that there are not many changes in values of an individual parameter across time (i.e., there are not many changes from left to right). The potential implication of this consistent values over time is that the time and sequences information might not be as useful in this case. This is why we decide to select the latest timestamps prior to the prediction of the other 23 variables (i.e., the $60th$ value in each of the 23 time series) in the Gradient boosting decision tree (GBDT) [15], will be discussed in Section V-A.

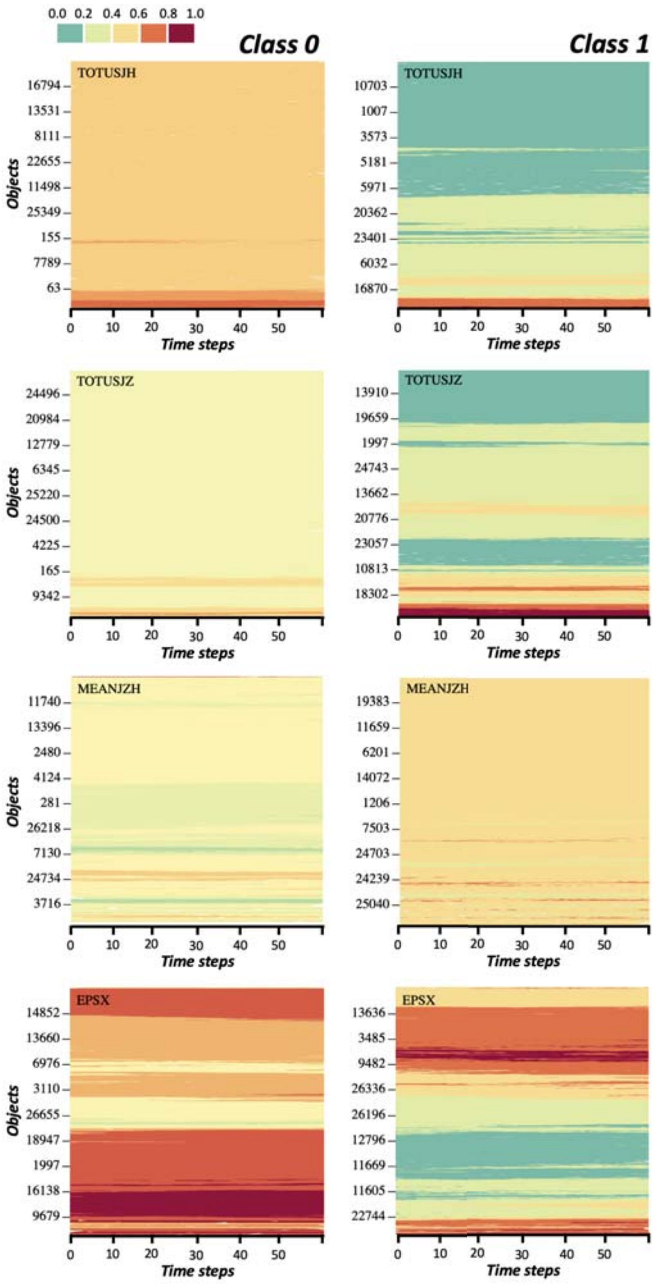


Fig. 4. *Continuous Heatmap* visualizations for *1,000* class-0 objects (on the left) and the same number of class-1 objects (on the right) for four sample variables (i.e., TOTUSJH, TOTUSJZ, MEANJZH, and EPSX). X-axis represent time steps (0 to 60), y-axis represent 1,000 objects (due to space limitation, only a few object ids are displayed). In the heatmap, darker green cells represent low values and darker red cells represent for high values.

V. Method and Result

Figure 5 shows a schematic overview of our ensemble approach by adopting an LSTM neural network, an MLP neural network, a logistic regression model, and a LightGBM model into one tier (*tier 1*). This tier then feed a second one (*tier 2*), which consists of another logistic regression model to produce the final output of our predictive model. Additional investigations into different ensemble configurations (e.g., different numbers of tiers and different numbers of models or model types per tier) are worth exploring in the future.

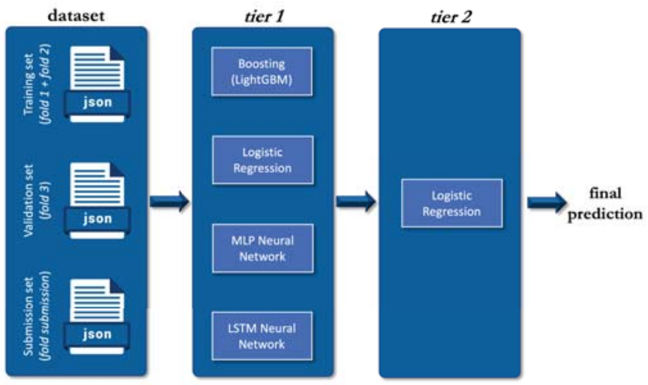


Fig. 5. General schematic diagram for our method: The first tier includes four models (i.e., they are a LightGBM model, a logistic regression model, an MLP neural network model, and an LSTM neural network model). The second tier contains another logistic regression model that assimilates the results from the previous tier and produces the final prediction.

The competition datasets are separated into training set (consisting of *fold 1* and *fold 2*), validation set (*fold 3*) and the submission set (*fold submission*). The training set is used to train models from tier 1, and the validation set is used for early stopping during the learning process (except for the logistic regression models). The learned models from tier 1 are then used to make probability predictions on this training dataset. The resulted probability predictions are then fed into tier 2 to train a meta-data model. The learned models from both tiers are then used to make probability predictions on the submission set.

The following sections discuss the detailed configurations of each model used in the two tiers of our solution, their corresponding individual result, and also the final result of our ensemble framework.

A. Gradient boosting decision tree

Gradient boosting decision tree (GBDT) [15] is a widely-used machine learning algorithm due to its efficiency, accuracy, and interpretability. GBDT achieves state-of-the-art performances in many machine learning tasks. For instance, among 29 challenges published on Kaggle [2] in 2015, 17 winning solutions used GBDT [7]. There are many implementations of GBDT, such as XGboost [7], Catboost [40], InfiniteBoost [42], and LightGBM [22]. In this paper, we adopt LightGBM as one of the four models in the tier due to its accuracy and speed (thanks to its parallel training support). It is also confirmed by Ma et al. [30] that recent experimental results support the statement “LightGBM is more efficient and accurate than other existing boosting tools”.

[2]https://www.kaggle.com/

The input data fed to the LightGBM model contains 318 features. These features include all temoral values of the two important variables TOTUSJH and TOTUSJZ (i.e. 2×60 variables), the latest timestamps prior to the prediction of the other 23 variables (i.e. the $60th$ value in each of the 23 time series), and seven statistical summaries (i.e. max, mean, median, min, skew, standard deviation, and sum) for each of the 25 raw variables (175 statistical summaries in total). The reason to select all temporal values of TOTUSJH and TOTUSJZ as input variables for the LightGBM model is that they are useful variables as explored from the literature review (discussed in Section III). Another decision worth discussing is to select the latest known points before the predictions but the other time steps of the other 23 variables. This decision was made with the assumption that the most recent known values before the event would reflect better knowledge about the event itself.

Also comparing to the previous work, besides the other six statistical features, we use ‘sum’ as an additional one. Though ‘sum’ and ‘mean’ in this competition dataset are highly correlated, they are slightly different because ‘sum’ takes missing values into account whereas ‘mean’ does not. Thus, ‘sum’ still brings some marginally useful information for the training process and prediction results. Specifically, without using the ‘sum’ feature in this model, the F_1 score on the validation set is 0.71637 with respect to (*w.r.t*, hereafter) a logarithmic loss (*log loss*, hereafter) of 0.38389. This F_1 score is slightly lower than the results (reported later in this section) utilizing ‘sum’.

To work with LightGBM, the analysts need to tune on several hyper-parameters. These parameters include the number of leaves per tree, learning rate, maximum learning depth, the minimum number of data in each leaf, feature fraction, and bagging fraction. In this case, we use *Hyperopt* [24] for optimizing of these values. It is a Bayesian model-based, distributed, asynchronous hyper-parameter optimization over awkward search spaces. The idea of *Hyperopt* is to choose the next hyper-parameter values to evaluate based on the past results. This strategy helps to concentrate the search on more promising values. Besides, we use a class weight ($class\_weight$) of 2.5 for *class 1* to tackle the class imbalance issue. Equally important, we set small max depth ($max\_depth = 3$), feature fraction ($feature\_fraction = 0.5$), and bagging fraction ($bagging\_fraction = 0.5$) to avoid overfitting.

The loss function used in this solution is *log* loss. It is the standard loss functions used in the literature on a probabilistic prediction from a data sequence [45]. Also, since F_1 score is the evaluation metric for the submission set of this competition, it is natural that we use it as an early stopping metric for this model. In other words, the early stopping criteria is the F_1 score of the predicted results over the validation set.

Figure 6 shows the training process of the LightGBM model completes after 1,815 boosting iterations, with F_1 score on the validation set of 0.7174 (w.r.t a log loss of 0.2656). Figure 7 shows the confusion matrix of the classification results when

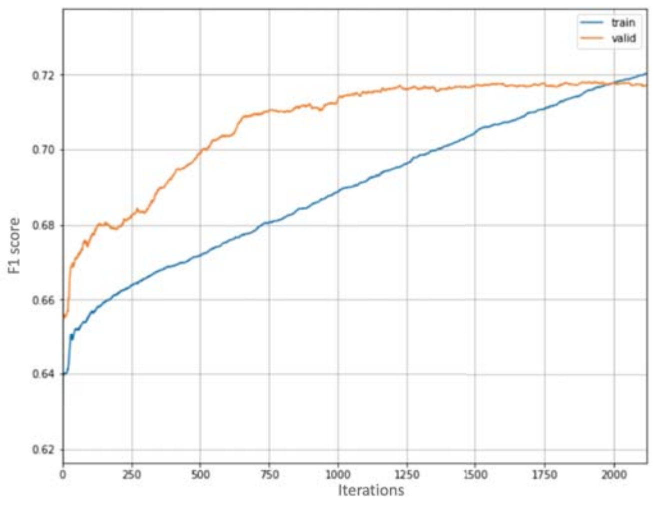


Fig. 6. F_1 scores of training and validation datasets during LightGBM model training. The LightGBM model training completes at about 1,815 boosting iterations (i.e., after this point, the training F_1 score continues to increase, but that score of the validation set starts to decrease).

apply the validation dataset to the learned LightGBM model.

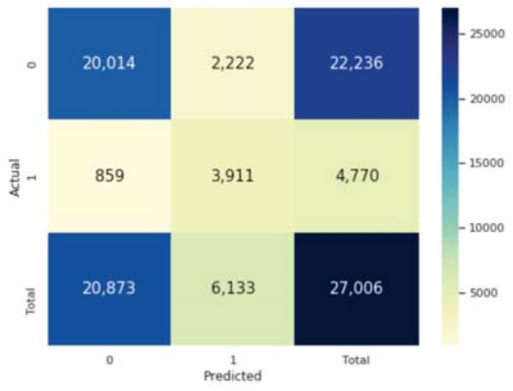


Fig. 7. Confusion matrix of the classification results applying only LightGBM model to the validation set.

Furthermore, it is also important to explore the importance of features in contribution to the accuracy of the results using the LightGBM model. After training, the importance of each feature of the model is calculated using SHAP (SHapley Additive exPlanations). SHAP [28] is a unified approach to explain the output of any machine learning model by game theory and local explanations. To be more specific, the importance represents the mean absolute value of the SHAP values for each feature.

Figure 8 shows top 15 useful features in this LightGBM model. Based on the figure, we can see that the standard deviation of ABSNJZH variable, *STD(data.ABSNJZH)*, is the most important feature. Besides, the 'sum' of R_VALUE, *SUM(data.R_VALUE)*, is also in the top list, this explains why we added this statistical feature in training the LightGBM model as discussed previously. Please also note that these are the levels of significance for the 318 individual features with corresponding values described above (i.e., they are not the raw variables as the whole time series in the competition dataset).

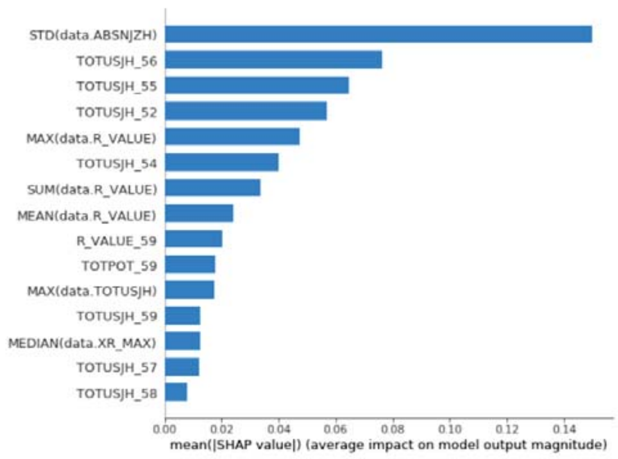
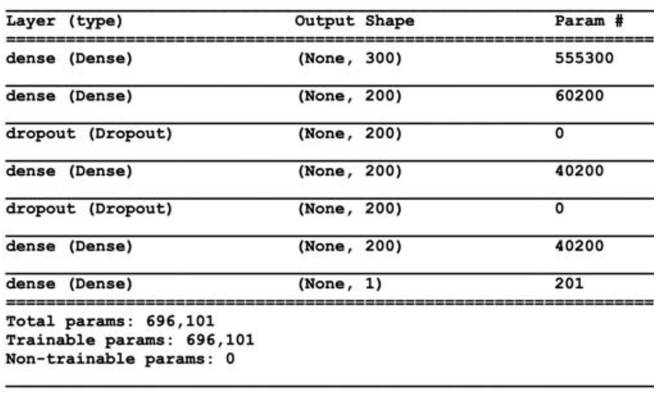


Fig. 8. Top 15 important features out of 318 individual features (see Section V-A) as inputs for the learned LightGBM model based on SHAP. Note that these are not the raw variables as the whole time series in the competition dataset.

B. MLP neural network



```
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 300)               555300
dense (Dense)                (None, 200)               60200
dropout (Dropout)            (None, 200)               0
dense (Dense)                (None, 200)               40200
dropout (Dropout)            (None, 200)               0
dense (Dense)                (None, 200)               40200
dense (Dense)                (None, 1)                 201
=================================================================
Total params: 696,101
Trainable params: 696,101
Non-trainable params: 0
```

Fig. 9. Our MLP neural network architecture. It comprises one output layer, one dense layer with 300 hidden units, and three other dense layers with 200 hidden units each. It also includes two dropout layers to tackle the overfitting issue.

As discussed previously, we also use an MLP neural network in tier 1 of our solution. The total number of input features for training this model is 1,850. These include all raw features ($60 \times 25 = 1,500$), 175 statistical features (as discussed in Section V-A), and 175 numerical ranks of these features. Also, different features would have different measurement units, thus value ranges. Therefore, we scale all these 1,850 features to [0,1] range using min-max normalization[3]. In other words, the values across each column are

[3]https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

linearly scaled to the range [0, 1]. Specifically, training set (*fold1* and *fold2*), validation set (*fold3*), and submission set (*fold submission*) are scaled individually and separately. This scaling step helps to avoid one, or some features dominating the others in contribution to the final classification model. Thus, it reduces overfitting issues and also makes the training process faster.

As explored in Section IV, the competition datasets contain missing values. There are several strategies to deal with missing values such as filling-forward, filling-backward, mean-value, or merely filling the missing values with zeros (or some other values based on domain knowledge). The filling-forward strategy means taking the most recent known value in the past to fill the missing value. Similarly, the filling-backward approach involves taking the closest known point after the point as the value for the missing one. On the other hand, the mean-value strategy takes the average value of the variable to fill the unknown places.

In this work, we fill the missing values as zeros for two main reasons. First, there are not many missing values in these datasets (see Section IV). Second, we add the 'sum' statistic (as discussed in Section V-A), which already considers the impacts of the missing values. Please also note that our approach to deal with missing value and the two supporting reasons do not rule out the possibilities to investigate the impacts of the other strategies. They should be exciting future directions to explore.

Figure 9 shows the summaries of our MLP neural network consolidated after several experiments on our dataset. Besides the output layer, it consists of one Dense layer with 300 hidden units and other three Dense layers with 200 hidden units each. Two Dropout layers are also used to tackle the overfitting issue. Also, with the same reasons discussed in Section V-A, we use F_1 score on the validation set for early stopping purpose and the log loss as the loss function. Specifically, the best F_1 score on the validation set is 0.6971 (w.r.t a log loss of 0.2412). In addition, Figure 10 shows the confusion matrix of the results applying only the generated MLP neural network into the validation set.

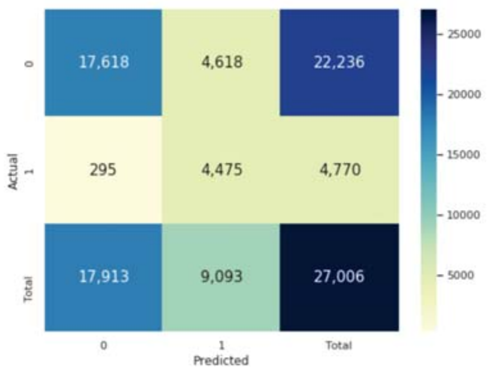


Fig. 10. Confusion matrix of the classification results applying only MLP neural network model to the validation set.

C. Logistic regression

Following a previous approach to this task [44], in tier 1 of our solution, we also adopt an ordinal logistic regression model for this problem. Mathematically, this ordinal model does not forecast the class labels but predicts the probability of having or not a flaring event given a time series of the observed solar magnetic field parameters. Though it is conceptually simple and algorithmically fast, experimental results confirm that it is compelling in evaluating the probability of flaring events.

We use all the raw features (25 variables over 60 time-steps) plus the other 175 statistical features (see Section V-A) to train this model. Before training, we also replace missing values with 0s (with the reasons discussed in Section V-B) and scale the values of the variables to [-1, 1] range (to avoid the case that one or some features dominate the others in our prediction result). The F_1 score on validation set for this model is 0.69448 (w.r.t a log loss of 0.25788). Specifically, Figure 11 shows the confusion matrix of the classification results applying this learned logistic model to the validation set.

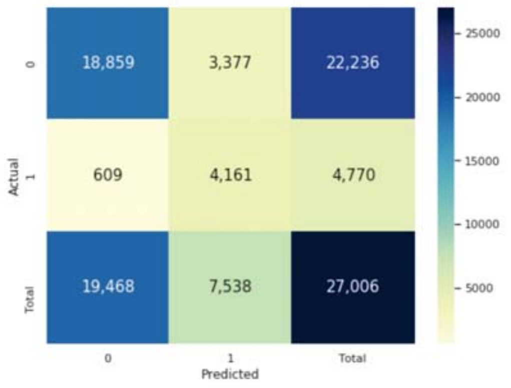


Fig. 11. Confusion matrix of the classification results applying only logistic regression model to the validation set.

D. LSTM neural network

LSTM is a type of neural network which allows the learning model to remember history information when processing each step in the sequential data. Therefore, it is considered as a natural fit for this problem. In this case, the input data for our LSTM model is a 3D array with the shape as the number of objects (IDs), time series steps, and the number of features. Specifically, *fold 1* and *fold 2* are used as training set, and *fold 3* is treated as validation set (for early stopping purpose). Similar to training MLP neural network, missing values are replaced by 0s, and the raw data are scaled to [-1, 1] range before putting into the networks.

In learning any neural network model, finding a suitable architecture is the key part. Liu et al., [27] suggest an interesting architecture for this type of problem. The proposed model comprises of an LSTM layer, an attention layer, two fully connected layers, and an output layer. We tried a variety of architectures, and from experimental results, we found a simpler architecture that seems to work well in this particular problem. As depicted in Figure 12, besides the output layer, this model consists of one LSTM layer (with 64 hidden units), and one Dense layer (with eight hidden units). It also includes a Dropout layer to tackle the overfitting issue.

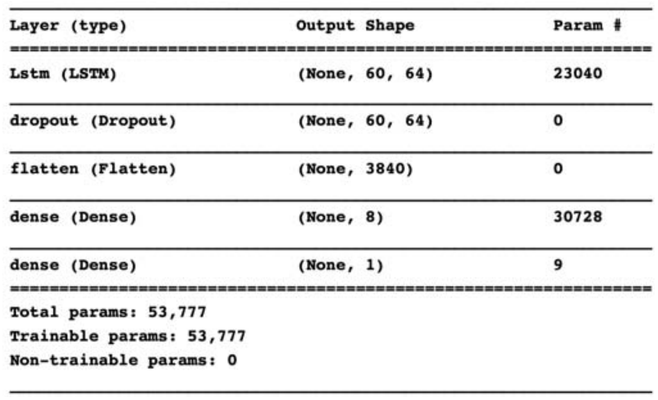
We also use the *log* loss as the objective function, and to void overfitting, we set early stop (*early_stop*) as 20 and validation metric is the F_1 score. In other words, after 20 epochs, if the F_1 score on the validation set does not increase, the training stops. After stopping, it returns the model at the epoch with the best validation F_1 score.

Similar to the other tested models, we also validate this LSTM model with the validation dataset. The best F_1 score is 0.6993 (w.r.t a log loss of 0.2823). Also, Figure 13 shows the confusion matrix of the classification results applying only LSTM neural network model to the validation set.

LSTM neural network by itself does not seem to be the best fit for our particular problem. Specifically, the result is not as good comparing to Boosting or MLP neural network models described previously for this case. One possible implication of this result is that the past information is not as useful as we think, or we may need another way to incorporate time and sequences information. In recent work, Cai et al., [6] also stated that past information is not as useful in this case.

E. Tier 2

As discussed, we explored relevant algorithms for this task, from simple one as logistic regression to sophisticated ones such as LSTM and LightGBM. We also reviewed the classification outcomes from individual best models selected from each of the algorithms. It was tempting to devise an ensemble to combine these models. Consequently, we added a simple logistic regression model as another tier stacked on the first one to connect the results from individual models in tier 1. Specifically, we take four sets of probability outputs from the four models in the first tier and generate the other four corresponding label output sets (class labels as 0s or 1s instead of probabilities for those values). These probability and



```
Layer (type)                 Output Shape              Param #
=================================================================
Lstm (LSTM)                  (None, 60, 64)            23040
_________________________________________________________________
dropout (Dropout)            (None, 60, 64)            0
_________________________________________________________________
flatten (Flatten)            (None, 3840)              0
_________________________________________________________________
dense (Dense)                (None, 8)                 30728
_________________________________________________________________
dense (Dense)                (None, 1)                 9
=================================================================
Total params: 53,777
Trainable params: 53,777
Non-trainable params: 0
_________________________________________________________________
```

Fig. 12. Our LSTM architecture comprises one LSTM layer (64 hidden units), two dense layers (one with eight hidden units, and another one for the output). It also includes a dropout layer to tackle the overfitting issue.

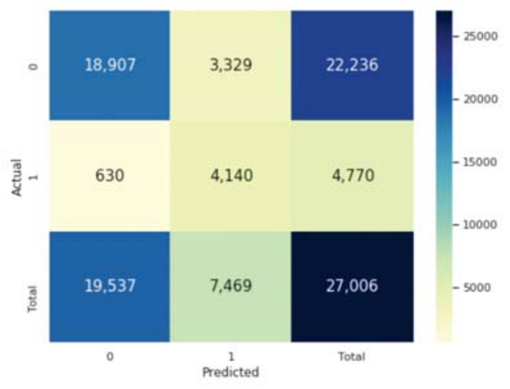


Fig. 13. Confusion matrix of the classification results applying only LSTM neural network model to the validation set.

label output sets add up to eight input features for training the logistic regression model in the second tier.

In detail, we normalize probability outputs from an individual model using percentile ranking [4] to make sure that the output distributions from the validation set and submission set are similar. It harms our results if we do not have similar distributions for these two output sets. For instance, Figure 14 shows the sample output probability distributions of the validation set (on the left) and the submission set (on the right) applied to the MLP neural network at tier 1. We can see the thicker tail toward 1.0 from the probability distributions of the outputs from the validation set comparing to that of the submission set. On the other hand, Figure 15 shows similar distributions for the probabilities outputs from these two sets after the normalization step.

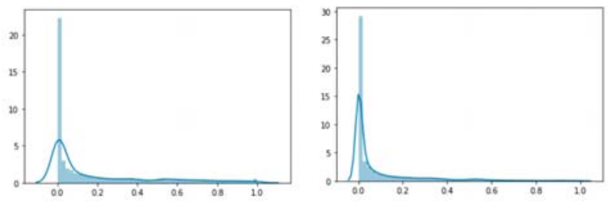


Fig. 14. Distribution of the output probabilities applying to the learned MLP neural network model on validation set (left) and submission set (right). The distribution on the left has thicker tail toward 1.0.

Regarding producing labels from probabilities, we tune a threshold for each model to produce the label predictions from the predicted probabilities. We assign class label 1 for an output probability, which is higher than this threshold and 0 otherwise. Table I summarizes the four thresholds and the corresponding $F_1 scores$ and log losses for the four models in tier 1. These thresholds are tuned in such a way that maximizes the F_1 score applied to the validation set and at the same time satisfying a second condition about similar percentages of class 1. The latter requirement means that the percentage of

[4]https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.rank.html

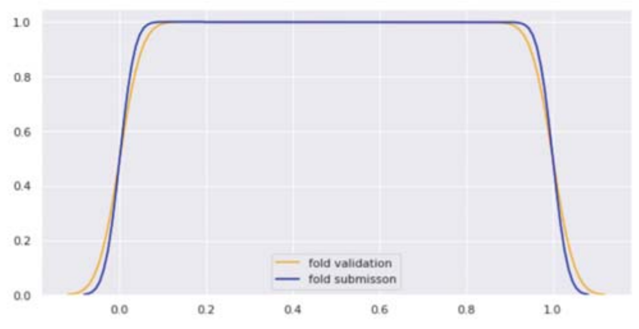


Fig. 15. Distributions of the output probabilities applying to the learned MLP neural network model on validation set (orange) and submission set (blue) after applying percentile ranking.

TABLE I
OUTPUT PROBABILITY TO CLASS LABEL THRESHOLDS AND CORRESPONDING F_1 SCORES AND LOG LOSSES.

	LightGBM	**MLP NN**	**Logistic**	**LSTM NN**
Threshold	0.16	0.37	0.5	0.28
F_1 score	0.7174	0.6971	0.6945	0.6993
Log loss	0.2656	0.2412	0.2579	0.2823

class 1 in predicted results for the submission fold is similar to that of the other known folds. The reason is that even though the class 1 percentage in the submission fold is unknown, it is safe to assume that this percentage is relatively similar to those for training and validation sets, as discussed in Section II.

After training the logistic model in the second tier, we also use a similar strategy discussed above to tune the prediction label threshold. Meaning we optimize the best F_1 score on the validation set, but at the same time having the percentage of the predicted class 1s in the submission set as similar to that in the validation set as possible. From experimental results, the best threshold for converting probability outputs from this logistic regression model (in the second tier) to labels is 0.42. Also, the resulted F_1 score is 0.7276 (w.r.t a log loss of 0.2664) on the validation set. We can see that this score on the validation set is higher than that of any individual model at tier 1, meaning that the ensemble performs well in combining the built models.

Finally, our solution has F_1 score of 0.65833 for the submission fold in the public board and F_1 score of 0.66933 for the submission fold on the private board.

VI. IMPLEMENTATION

The solution is implemented as Jupyter Notebooks [23] using several Python machine learning and data visualization packages such as *lightgbm*, *sklearn*, *keras*, *matplotlib*, and *seaborn* to name but a few. The source codes, and experiemental results are hosted on the Github page of the project: https://github.com/iDataVisualizationLab/C/tree/master/solar_flare_bigdatacup.

VII. CONCLUSION

In this paper, we propose an ensemble approach to solar eruptive activity prediction problem with five models organized into two tiers. The types of models themselves are the results of our study concerning relevant approaches to solar flare event prediction problem and the recent advancements in the machine learning field. Specifically, there are four models in the first tier (a LightGBM model, an MLP neural network model, an LSTM neural network model, and a logistic regression model), and another one logistic regression model in the second tier to assimilate the previous models. The architecture details and hyper-parameters of the models are learned from experimental results using the given data. In particular, we divided the competition data into three sets, namely training set (*fold 1* and *fold 2*), validation set (*fold 3*), and submission set (*fold submission*). The training set and validation set are used to explore and train the models in our solution before applying it to the submission set for the competition.

In the future, there are several directions worth exploring, such as using a convolution neural network (CNN) to preprocess data before feeding into an LSTM neural network to learn the model or to explore more models such as random forest or SVM models, or even adding more tiers. Additional worth trying directions would be testing other loss functions, optimizing other scores such as TSS and HSS (Heidke Skill Score), utilizing different techniques to handle the imbalance of classes in this problem.

REFERENCES

[1] A. Al-Ghraibah, L. Boucheron, and R. McAteer. An automated classification approach to ranking photospheric proxies of magnetic energy build-up. *Astronomy & Astrophysics*, 579:A64, 2015.
[2] A. O. Benz. Flare observations. *Living Reviews in Solar Physics*, 5(1):1, Feb 2008. doi: 10.12942/lrsp-2008-1
[3] D. S. Bloomfield, P. A. Higgins, R. J. McAteer, and P. T. Gallagher. Toward reliable benchmarking of solar flare forecasting methods. *The Astrophysical Journal Letters*, 747(2):L41, 2012.
[4] M. Bobra and HMI Vector Field Team. Spaceweather HMI Active Region Patch (SHARP). http://jsoc.stanford.edu/doc/data/hmi/sharp/sharp.htm. Accessed: 2019-09-27.
[5] M. G. Bobra and S. Couvidat. Solar flare prediction using sdo/hmi vector magnetic field data with a machine-learning algorithm. *The Astrophysical Journal*, 798(2):135, 2015.
[6] J. Cai, W. Carande, J. Craft, M. Hartnett, A. Jones, K. Kokkonen, T. Morland, and L. Sandoval. Solar flare forecasting: A novel deep learning approach. In *AGU Fall Meeting Abstracts*, 2018.
[7] T. Chen and C. Guestrin. XGBoost. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, vol. 19, pp. 785–794. ACM Press, New York, New York, USA, 2016. doi: 10.1145/2939672.2939785
[8] Y. Cui, R. Li, L. Zhang, Y. He, and H. Wang. Correlation between solar flare productivity and photospheric magnetic field properties. *Solar Physics*, 237(1):45–59, 2006.
[9] T. Dang, H. N. Nguyen, and V. Pham. WordStream: Interactive Visualization for Topic Evolution. In J. Johansson, F. Sadlo, and G. E. Marai, eds., *EuroVis 2019 - Short Papers*. The Eurographics Association, 2019. doi: 10.2312/evs.20191178
[10] T. N. Dang, A. Anand, and L. Wilkinson. Timeseer: Scagnostics for high-dimensional time series. *IEEE Transactions on Visualization and Computer Graphics*, 19(3):470–483, March 2013. doi: 10.1109/TVCG.2012.128

[11] Data Mining Lab at Georgia State University. Multivariate time series dataset for space weather data analytics. http://dmlab.cs.gsu.edu/?page_id=706. Accessed: 2019-09-27.

[12] Data Mining Lab at Georgia State University. Solar flare prediction from time series of solar magnetic field parameters. http://dmlab.cs.gsu.edu/bigdata19/flare-comp/index.html. Accessed: 2019-09-27.

[13] Data Mining Lab at Georgia State University. Solar flare prediction from time series of solar magnetic field parameters: Description of the Data. http://dmlab.cs.gsu.edu/bigdata19/flare-comp/Data.html. Accessed: 2019-09-27.

[14] K. Fox. Solar Flares: What Does It Take to Be X-Class? https://www.nasa.gov/mission_pages/sunearth/news/X-class-flares.html. Accessed: 2019-09-27.

[15] J. H. Friedman. Greedy Function Approximation: A gradient Boosting Machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[16] M. W. Gardner and S. R. Dorling. Artificial neural networks (the multilayer perceptron) - a review of applications in the atmospheric sciences. *Atmospheric Environment*, 32(14-15):2627–2636, 1998. doi: 10.1016/S1352-2310(97)00447-0

[17] C. Goutte and E. Gaussier. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In *European Conference on Information Retrieval*, pp. 345–359. Springer, 2005.

[18] S. M. Hamdi, D. Kempton, R. Ma, S. F. Boubrahimi, and R. A. Angryk. A time series classification-based approach for solar flare prediction. In *2017 IEEE International Conference on Big Data (Big Data)*, pp. 2543–2551. IEEE, 2017.

[19] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[20] J. Jing, H. Song, V. Abramenko, C. Tan, and H. Wang. The statistical relationship between the photospheric magnetic parameters and the flare productivity of active regions. *The Astrophysical Journal*, 644(2):1273, 2006.

[21] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pp. 3146–3154, 2017.

[22] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. Y. Liu. LightGBM: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, 2017-December(Nips):3147–3155, 2017.

[23] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. E. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. B. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, C. Willing, and et al. Jupyter notebooks - a publishing format for reproducible computational workflows. In *ELPUB*, 2016.

[24] B. Komer, J. Bergstra, and C. Eliasmith. Hyperopt-Sklearn: Automatic Hyperparameter Configuration for Scikit-Learn. *Proceedings of the 13th Python in Science Conference*, pp. 32–37, 2014. doi: 10.25080/majora-14bd3278-006

[25] D. D. Le, V. Pham, H. N. Nguyen, and T. Dang. Visualization and Explainable Machine Learning for Efficient Manufacturing and System Operations. *Smart and Sustainable Manufacturing Systems*, 3(2):20190029, feb 2019. doi: 10.1520/SSMS20190029

[26] K. Leka and G. Barnes. Photospheric magnetic field properties of flaring versus flare-quiet active regions. ii. discriminant analysis. *The Astrophysical Journal*, 595(2):1296, 2003.

[27] H. Liu, C. Liu, J. T. Wang, and H. Wang. Predicting solar flares using a long short-term memory network. *The Astrophysical Journal*, 877(2):121, 2019.

[28] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pp. 4765–4774, 2017.

[29] R. Ma, S. F. Boubrahimi, S. M. Hamdi, and R. A. Angryk. Solar flare prediction using multivariate time series decision trees. In *2017 IEEE International Conference on Big Data (Big Data)*, pp. 2569–2578. IEEE, 2017.

[30] X. Ma, J. Sha, D. Wang, Y. Yu, Q. Yang, and X. Niu. Study on a prediction of P2P network loan default based on the machine learning LightGBM and XGboost algorithms according to different high dimensional data cleaning. *Electronic Commerce Research and Applications*, 31(August):24–39, 2018. doi: 10.1016/j.elerap.2018.08.002

[31] S. A. Murray. The importance of ensemble techniques for operational space weather forecasting. *Space Weather*, 16(7):777–783, July 2018. doi: 10.1029/2018SW001861

[32] National Science and Technology Council. National space weather action plan. 2015.

[33] N. V. T. Nguyen, V. T. Nguyen, V. Pham, and T. Dang. Finanviz: Visualizing emerging topics in financial news. In *2018 IEEE International Conference on Big Data (Big Data)*, pp. 4698–4704, Dec 2018. doi: 10.1109/BigData.2018.8622097

[34] V. T. Nguyen, T. Dang, and F. Jin. Predict Saturated Thickness using TensorBoard Visualization. In K. Rink, D. Zeckzer, R. Bujack, and S. Jnicke, eds., *Workshop on Visualisation in Environmental Sciences (EnvirVis)*. The Eurographics Association, 2018. doi: 10.2312/envirvis.20181135

[35] N. Nishizuka, K. Sugiura, Y. Kubo, M. Den, S. Watari, and M. Ishii. Solar flare prediction model with three machine-learning algorithms using ultraviolet brightening and vector magnetograms. *The Astrophysical Journal*, 835(2):156, 2017.

[36] V. Pham and T. Dang. Cvexplorer: Multidimensional visualization for common vulnerabilities and exposures. In *2018 IEEE International Conference on Big Data (Big Data)*, pp. 1296–1301, Dec 2018. doi: 10.1109/BigData.2018.8622092

[37] V. Pham and T. Dang. Outliagnostics: Visualizing temporal discrepancy in outlying signatures of data entries, 2019.

[38] V. Pham, V. T. Nguyen, and T. Dang. Iotviz: Visualizing emerging topics in the internet of things. In *2018 IEEE International Conference on Big Data (Big Data)*, pp. 4569–4576, Dec 2018. doi: 10.1109/BigData.2018.8622375

[39] V. V. Pham and T. Dang. Mtdes: Multi-dimensional temporal data exploration system; strong support for exploratory analysis award in vast 2018, mini-challenge 2. In *2018 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 100–101, Oct 2018. doi: 10.1109/VAST.2018.8802440

[40] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. Catboost: Unbiased boosting with categorical features. *Advances in Neural Information Processing Systems*, 2018-December(Section 4):6638–6648, 2018.

[41] R. Qahwaji and T. Colak. Automatic short-term solar flare prediction using machine learning and sunspot associations. *Solar Physics*, 241(1):195–211, 2007.

[42] A. Rogozhnikov and T. Likhomanenko. InfiniteBoost: building infinite ensembles with gradient descent. pp. 1–7, 2017.

[43] C. J. Schrijver. A characteristic magnetic field pattern associated with all major solar flares and its use in flare forecasting. *The Astrophysical Journal Letters*, 655(2):L117, 2007.

[44] H. Song, C. Tan, J. Jing, H. Wang, V. Yurchyshyn, and V. Abramenko. Statistical assessment of photospheric magnetic features in imminent solar flare predictions. *Solar Physics*, 254(1):101–125, 2009.

[45] V. Vovk. The Fundamental Nature of the Log Loss Function. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9300, pp. 307–318. Springer, 2015. doi: 10.1007/978-3-319-23534-9_20