



Article

TLW: A Real-Time Light Curve Classification Algorithm for Transients Based on Machine Learning

Mengci Li ^{1,2}, Chao Wu ^{3,*}, Zhe Kang ^{1,2}, Chengzhi Liu ^{1,2,4,*}, Shiyu Deng ^{1,3}  and Zhenwei Li ^{1,2} 

- ¹ Changchun Observatory, National Astronomical Observatories, Chinese Academy of Sciences, Changchun 130117, China; limengci329@163.com (M.L.); kangz@cho.ac.cn (Z.K.); dengsy@cho.ac.cn (S.D.); lizw@cho.ac.cn (Z.L.)
- ² University of Chinese Academy of Sciences, Beijing 100049, China
- ³ National Astronomical Observatories, Chinese Academy of Sciences, Beijing 100101, China
- ⁴ Key Laboratory of Space Object & Debris Observation, Purple Mountain Observatory, Chinese Academy of Sciences, Nanjing 210008, China
- * Correspondence: cwu@nao.cas.cn (C.W.); lcz@cho.ac.cn (C.L.)

Abstract: The real-time light curve classification of transients is helpful in searching for rare transients. We propose a new algorithm based on machine learning, namely the Temporary Convective Network and Light Gradient Boosting Machine Combined with Weight Module Algorithm (TLW). The TLW algorithm can classify the photometric simulation transients data in *g*, *r*, *i* bands provided via PLAsTiCC, typing Tidal Disruption Event (TDE), Kilonova (KN), Type Ia supernova (SNIa), and Type I Super-luminous supernova (SLSN-I). When comparing the real-time classification results of the TLW algorithm and six other algorithms, such as Rapid, we found that the TLW algorithm has the best comprehensive performance indexes and has the advantages of high precision and high efficiency. The average accuracy of TLW is 84.54%. The average implementation timings of the TLW algorithm for classifying four types of transients is 123.09 s, which is based on TensorFlow's architecture in windows and python. We use three indicators to prove that the TLW algorithm is superior to the classical Rapid algorithm, including Confusion Matrix, PR curve, and ROC curve. We also use the TLW algorithm to classify ZTF real transients. The real-time classification results for ZTF transients show that the accuracy of the TLW algorithm is higher than the other six algorithms.

Keywords: transient; light curve; photometry; real-time classification; machine learning



Citation: Li, M.; Wu, C.; Kang, Z.; Liu, C.; Deng, S.; Li, Z. TLW: A Real-Time Light Curve Classification Algorithm for Transients Based on Machine Learning. *Universe* **2024**, *10*, 31. <https://doi.org/10.3390/universe10010031>

Academic Editor: Jason McEwen

Received: 29 November 2023

Revised: 28 December 2023

Accepted: 9 January 2024

Published: 11 January 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

A transient is an extreme, short-lived astronomical phenomenon with durations of fractions of a second to weeks or years [1]. Transients are generally related to the destruction of the astrophysical object. Transients have great significance for studying the origin of the universe and physical phenomena in extreme environments [2]. In the era of multi-messenger astronomy, observation facilities have the characteristics of a wide field and high temporal sampling rate [1]. Therefore, many transients have been found, such as gravitational wave (GW) [3], supernova (SN) [4], Tidal Disruption Event (TDE) [5], etc. In particular, a counterpart of GW170817 [6,7] has been confirmed to be a Kilonova (KN) [8] and has emerged as a focal point in transient science research.

Some facilities, which can detect transients, are working or being built, such as the Zwicky Transient Facility (ZTF) [9], Large Synoptic Survey Telescope (LSST) [10], Dark Energy Survey (DES) [11], Panoramic Survey Telescope and Rapid Response System, (PanSTARRS) [12], Gravitational wave high-energy Electromagnetic Counterpart All-sky Monitor (GECAM) [13], China Space Station Telescope (CSST) [14], Space-based multi-band astronomical Variable Objects Monitor (SVOM) [15], etc. One of the important goals of the SVOM mission is to find Target of Opportunity (ToOs) [15], such as KN. Optical light curves of transients exhibit a rapid decay, and there will be millions of transient alarms

every night in the future sky survey [2]. Therefore, it is necessary to use machine learning technology to study the real-time classification algorithm and determine the classes of transients at an early epoch. This real-time classification algorithm is also helpful to make a follow-up observation of objects and promote the study of their physical properties and precursor systems [2].

Currently, optical transient classification relies on spectroscopic and photometric data. The spectroscopic classification is the most accurate. But spectroscopic observation is expensive, which has high requirements for telescopes and observation time. Therefore, spectroscopic classification can only be applied to limited objects [16]. Compared to spectroscopic observation, the photometric method has a lower accuracy but higher efficiency. Photometric classification mainly includes the template fitting method [17] and machine learning method [18,19]. The above classification algorithms in Refs [17–19] require that transients have a complete light curve with full phase coverage, so these algorithms cannot classify transients in real time. They are also not suitable for sparse data [2]. In order to solve the above problems, it is necessary to study the real-time light curve classification algorithms of transients, which can classify transients before obtaining the complete light curve. In the recent five years, Muthukrishna et al. proposed the RAPID algorithm based on two unidirectional gated recurrent unit (GRU) layers [2]. Möller and de Boissière proposed the supernova framework, which is an SN classifier based on a Bayesian recurrent neural network [20]. Godines et al. used Random Forest (RF) to classify gravitational microlensing [21]. Ishida et al. used active learning to achieve early classification of SN [22]. Villar et al. used a variety of feature extraction methods and data augmentation algorithms combined with the Support Vector Machine (SVM), RF, and Multilayer Perceptron (MLP) to classify SN. The experiment results proved that RF is the optimal classification algorithm. This algorithm can realize early classification in theory, but its accuracy needs to be verified [23]. Hosseinzadeh et al. named the above algorithm Superphot [24]. Stachie et al. improved the Rapid algorithm [25]. The weighted formula is added to the improved algorithm, aiming at identifying KN. Takahashi et al. used the deep neural network with highway layer to classify SN [26]. Villar et al. proposed a semi-supervised classification algorithm SuperRAENN, which is based on recurrent autoencoder neural networks (RAENNs) [27]. Baldeschi et al. used RF to classify SN and host galaxies [28]. Andreoni et al. used a convolutional neural network (CNN) to detect transients and identified KN through the cross-matching method of the star catalog [29]. Burhanudin et al. used long short-term memory (LSTM) and GRU to classify transient sources, which are suitable for unbalanced data [30]. Allam et al. used the Transformer framework to classify transients, which were provided via the Photometric LSST Astronomical Timeseries Classification Challenge (PLAsTiCC) [31]. Later, deep compression technology was used to improve the practicability of the algorithm [32]. Qu et al. proposed a classifier SCONE [33] based on CNN. Burhanudin et al. used Gaussian regression to generate heat maps of SN photometric data and used CNN to classify the heat maps [34]. Gomez et al. proposed a FLEET algorithm based on RF to identify TDE, which needs to use light curves and host galaxies of objects [35]. Gagliano et al. improved the Rapid algorithm, which used the single-layer LSTM algorithm and added the SN host photometric information to solve the problem of sparse data [36]. Pimentel et al. proposed a deep attention model (TimeModAttn) to classify supernovae [37]. Kisley et al. proposed a nuclear density estimation method which only relies on the host photometric information [38].

To sum up, the popular real-time classifiers mainly include adding host feature information [28,35,36,38] or using machine learning techniques such as recurrent neural network (RNN) [2,20,25,27,30,36], CNN [29,33,34], Transformer [31,32], RF [21,23,24,28,35], active learning [22], etc. The above methods have different advantages: RNN is suitable for a time series data modeling task; CNN has better feature extraction capability than a cyclic neural network; Transformer is usually based on multi-head attention, which can better capture the global information; and RF has high accuracy and strong robustness. We propose a new high-accuracy and efficient classification algorithm: the Temporal Convolutional Network

and Light Gradient Boosting Machine combined with Weight module (TLW) algorithm, which is not only suitable for real-time light curves but also has good feature extraction capability and can reduce generalization error. The TLW algorithm will be applied to the ground-based robotic follow-up telescope (C-GFT) in the SVOM mission in the future.

The organization of the paper is as follows: in Section 2, we introduce a new real-time light curve classification algorithm TLW, which can obtain the type of the transient object rapidly. In Section 3, we describe the data and the experimental process. In Section 4, we present an analysis of the experimental results. In Section 5, we use the TLW algorithm to classify ZTF data, aiming to verify the effectiveness of the algorithm. In Section 6, we give the conclusions.

2. TLW Algorithm

The TLW algorithm has good feature extraction capability. It can quickly extract relevant features when the sample data are small and sparse. It also has the advantages of high classification accuracy, strong robustness, and strong anti-noise capability. Moreover, the TLW algorithm has high speed, high efficiency, and low memory occupation. The advantages of the algorithm are introduced in Section 2.4, and the experimental results are shown in Sections 4 and 5. Its model architecture is shown in Figure 1. The TLW algorithm is based on a Temporary Convective Network (TCN) [39] and Light Gradient Boosting Machine (LightGBM) [40,41]. In addition to input and output modules, the TLW model includes the TCN module, LightGBM module, and Weight module.

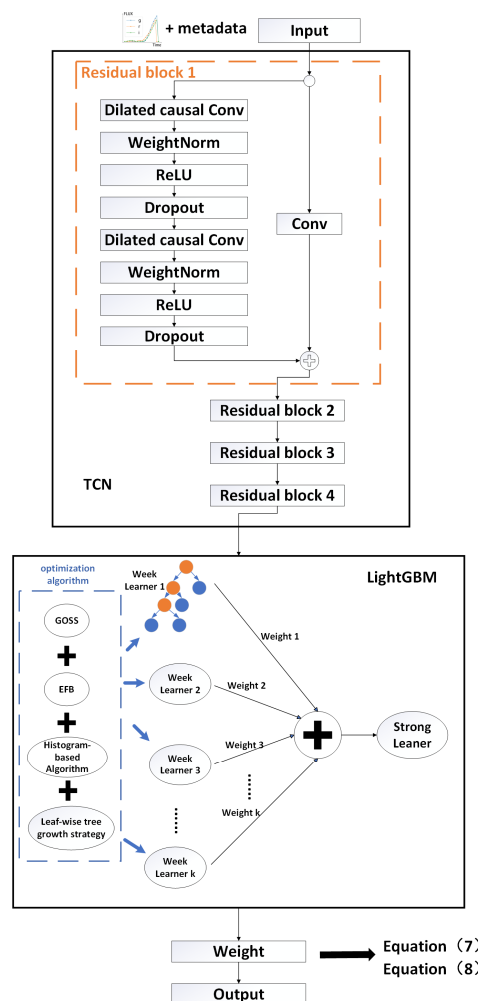


Figure 1. Model architecture used in the TLW algorithm. We input the preprocessed light curve and other metadata into the model and then the TLW model outputs the classifications of transients.

2.1. TCN Module

The TCN module is used for feature extraction. It can reduce the workload of manual feature extraction and automatically extract more effective and complex temporal features from samples.

The module mainly relies on the TCN model with four residual blocks. Based on CNN, TCN combines causal convolutions, dilated convolutions, and residual connections [39]. Causal convolutions can strictly constrain time, which only learn the current moment and previous elements and do not need future data. Therefore, causal convolution is suitable for a time series data modeling task, which can learn the effective history. However, each causal convolution neuron can only capture the elements at the time points near the current moment in the sequence, and the receptive field size is small. Dilated convolutions can expand the receptive field size exponentially to achieve a longer effective history. It can reduce information loss without adding parameters and complexity of models. The receptive field size is related to the depth of the network, so it is necessary to solve the problem of gradient vanishing caused by the extremely deep network. Residual connections can map elements across layers, which is helpful to train deep networks and solve the above problem. As shown in Figure 1, the TCN module consists of four Residual blocks. Each Residual block contains two layers of dilated causal convolutions (Dilated causal Conv) to extract the features of the input sequence. The number of neurons in each layer is 100; the convolution kernel size is 2; and the dilation factors of the four residual blocks are 1, 2, 4, and 8 respectively. The output of each Dilated causal Conv layer is normalized using the Weightnorm layer, aiming at accelerating network training and stabilizing gradient. Then the ReLU activation function is used to enhance the nonlinear ability of the model, and Dropout is used to avoid over-fitting.

2.2. LightGBM Module

The LightGBM module is used to preliminarily classify transients. The features obtained using the TCN module are input into this module, namely the LightGBM model. The module outputs the probability scores of each candidate class of the objects.

LightGBM is a boosting algorithm based on a decision tree [40,41]. Different from other decision tree algorithms, the LightGBM algorithm uses Gradient-based One-Side Sampling (GOSS), Exclusive Feature Bundling (EFB), a leaf-wise tree growth strategy with depth limitation, and a histogram algorithm to optimize the decision trees. The GOSS algorithm can select samples according to the gradient of data instances [41]. In order to reduce the sample ensemble size and improve the efficiency of the algorithm, the GOSS algorithm keeps samples with large gradients and randomly drops samples with small gradients. The EFB algorithm bundles mutually exclusive features within a certain conflict rate, which can reduce feature dimension without information loss [41]. It can generate a new transformed set of features. Leaf-wise tree growth strategy with depth limitation is splitting the leaf nodes with the largest splitting gain, while other leaf nodes do not split [41]. The typical decision trees, such as Canonical ID3 and C 4.5, are based on the level-wise tree growth strategy, which can split multiple leaf nodes in the same level. In the typical decision trees, some leaves have low splitting gains. Splitting these leaves will increase unnecessary calculation. As shown in Figure 1, the orange dots are the leaf nodes with the largest splitting gain, and the blue dots are other leaf nodes. With the same number of splits, the leaf-wise tree growth strategy has a smaller error than the level-wise tree growth strategy, owing to the fact that this growth strategy splits all leaves with the largest splitting gain. This growth strategy can also accelerate the learning speed of the algorithm. But other leaves are not splitting, which will make the decision tree not comprehensively learn all the features. And splitting only one leaf in each level will increase the depth of the tree and make the model over-fitted. Therefore, it is necessary to limit the depth of the tree to avoid over-fitting. The histogram algorithm is used to find the split node. The histogram algorithm needs to discretize continuous floating-point eigenvalues into integers. We construct a histogram with a width equal to the number of integers mentioned above.

When traversing all the data for the first time, the discrete values are used as indexes to accumulate the statistics of the histogram. After that, we can find the optimal segmentation point according to the discrete value of histogram without traversing all the data. The algorithm can divide continuous eigenvalues into finite discrete values, which can reduce memory consumption and accelerate the training speed.

In addition, the LightGBM algorithm uses the second-order Taylor expansion to calculate the first and second derivatives of the objective function, which improves the efficiency of the algorithm, in order to save time in calculating the minimum value of the objective function, as shown in Equations (3)–(5). The objective function includes loss function and the L2 regularization term, as shown in Equation (6). The LightGBM algorithm also uses regularization parameters to avoid over-fitting. Firstly, the LightGBM module builds a lot of leaf-wise trees in sequence as shown in Figure 1, namely Weak Learner. Secondly, the LightGBM module trains each of the weak classifiers until the minimum of the objective function is calculated. Finally, all Weak Learners are weighted for boosting via the tree boosting technique to become a strong learner, which can predict the final results. The training steps are as follows:

Step 1: Grow an initial decision tree, namely Weak Learner 1. Use the sample data obtained using the TCN module to train Weak Learner 1. The predicted values are obtained. The sample residuals are calculated with the real values.

Step 2: Grow a new decision tree, train this decision tree with current sample residuals, and divide the sample into leaf nodes according to the tree structure. According to the samples and residuals on leaf nodes, the weight of the leaf nodes is calculated. Stop training until the objective function is minimum.

Step 3: Add the prediction result of the current tree and the weight of leaf nodes to the prediction result of the current model and update the model.

Step 4: When the total number of decision trees reaches the maximum value k , the model is established and the final prediction result \hat{y}_i ($i = 1, 2, \dots, n$, where n is the number of transients) is output, as shown in Equation (1). Otherwise, create a new decision tree, update sample residuals and repeat steps 2–4.

$$\hat{y}_i = \sum_{k=1}^K f_k(x_i), \tag{1}$$

where \hat{y}_i represents the prediction probability result of the i transient, x_i is the feature data set of the i transient, and f_k is the k decision tree. It can be seen from Equation (1) that the total predicted value $\hat{y}_i^{(k)}$ of tree K is equal to the sum of the total predicted value $\hat{y}_i^{(k-1)}$ of tree K and the predicted value $f_k(x_i)$ of tree K , as shown in Equation (2).

$$\hat{y}_i^{(k)} = \hat{y}_i^{(k-1)} + f_k(x_i) \tag{2}$$

Leaf-wise strategy is a greedy algorithm, and each selection can minimize the features as in functional analysis and segmentation points of the loss function. This selection strategy enables the tree to learn the relationship between the details and features of data more quickly. The objective function of the k th tree is shown in Equation (3).

$$O^{(k)} = \sum_{i=1}^n l\left(y_i, \hat{y}_i^{(k)}\right) + \Omega(f_k), \tag{3}$$

where $l\left(y_i, \hat{y}_i^{(k)}\right)$ is the loss function between the real value and the predicted value, which is the quadratic softmax loss function, and $\Omega(f_k)$ is a regularization term, that is, the model

complexity of the k tree, as shown in Equation (4). The smaller the complexity, the stronger the generalization ability of the model.

$$\Omega(f_k) = \gamma T + \frac{1}{2} \lambda \|w\|^2 = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2, \tag{4}$$

where γ and λ are penalty coefficients, T is the number of leaf nodes of the k tree, w_j is the weight of the j leaf node, and w represents the output score on each tree leaf node. L2 regularization is related to $\lambda \|w\|^2$.

In order to obtain the minimum value of the objective function, the LightGBM algorithm substitutes Equations (2) and (4) into Equation (3) and simplifies it via Taylor expansion. The result is shown in Equation (5).

$$O^{(k)} = \sum_{i=1}^n l\left(y_i, \hat{y}_i^{(k-1)} + f_k(x_i)\right) + \gamma T + \frac{1}{2} \lambda \|w\|^2 = \sum_{i=1}^n \left[l\left(y_i, \hat{y}_i^{(k-1)}\right) + g_i f_k(x_i) + \frac{1}{2} h_i f_k^2(x_i) \right] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2, \tag{5}$$

where $g_i = \partial_{\hat{y}_i} l\left(y_i, \hat{y}_i^{(k-1)}\right)$ represents the first derivative, and $h_i = \partial_{\hat{y}_i}^2 l\left(y_i, \hat{y}_i^{(k-1)}\right)$ represents the second derivative. Ignoring the constant term, the objective function can be rewritten as Equation (6).

$$O^{(k)} = \sum_{j=1}^T \left[\sum g_j w_j + \frac{1}{2} (\sum h_j + \lambda) w_j^2 \right] + \gamma T \tag{6}$$

It can be seen from Equation (6) that when $w_j = -\frac{\sum g_j}{\sum h_j + \lambda}$, the minimum value

$$O_{\min} = -\frac{1}{2} \sum_{j=1}^T \frac{(\sum g_j)^2}{\sum h_j + \lambda} + \gamma T.$$

The LightGBM model parameters are as follows: the number of decision trees is 200, and the maximum leaf node is 60, which is the largest number of leaves in a tree. The max depth is 30, which is the depth of the dendrogram. The minimum child sample is 20, which is the minimum number of examples per leaf.

2.3. Weight Module

The Weight module in the TLW algorithm plays a crucial role in enhancing classification accuracy via mitigating noise effects. It does this via applying a weighting operation to the class probability scores obtained from the LightGBM module, based on the features extracted using the TCN module.

Due to the influence of image artifacts, the deviation introduced via the telescope CCD camera, and cosmic rays, the observed photometric data will be mixed with noise. Moreover, due to the short timescale characteristics of transients, especially KN, the decay rate of brightness is very fast, and the actual observation time is often as long as several days or dozens of days. Therefore, only the first few data points in some photometric data time series contain object information, and the rest only contain noise in the infinite time series of this event. When the TCN module extracts feature information, it will misjudge noise as the signal feature of the transient for learning and training, thus affecting the accuracy of the whole model. Therefore, a new weighting algorithm is proposed in this paper, which gives greater weight to the class probability scores obtained at the time points with signals, thus reducing the noise impact, as shown in Equation (7).

$$\left\{ \begin{array}{l} W_{i,t,c} = \max(\text{CP}_{i,t,c}) \times A_{\text{argmax},c} + W_{i,t-1,c} \\ \text{when } \sum_{f=1}^F x_{i,t,f} > \sum_{f=1}^F \alpha_f \times \max(X_{i,t,f}) \text{ and submaximum}(\text{CP}_{i,t,c}) < \max(\text{CP}_{i,t,c}) - \beta \text{ and } t > 1 \\ W_{i,t,c} = 1 \quad t = 1 \\ W_{i,t,c} = W_{i,t-1,c} \quad \text{others} \end{array} \right. , \tag{7}$$

where $CP_{i,t,c}$ is the category probability score obtained using the LightGBM module. $W_{i,t,c}$ is the weight value of the class C at the time t of the i object. $A_{\text{argmax},c}$ is the activation coefficient. When $\text{argmax} = c$, $A_{\text{argmax},c} = 1$, and the rest are 0. C is the number of categories. $x_{i,t,f}$ is the light curve flux value of f bandpass at time t of the i object, $X_{i,t,f}$ is the light curve of f bandpass at time t before the i object, f is the number of bandpass, α_f is the characteristic coefficient, and β is a constant.

We use Equation (8) to obtain the final prediction result of TLW algorithm, where $W_{i,t,c}$ is defined using Equation (7).

$$y_{i,t,c} = \frac{W_{i,t,c} \times CP_{i,t,c}}{\sum_{c=1}^C W_{i,t,c} \times CP_{i,t,c}} \quad (8)$$

2.4. Advantages of the TLW Algorithm

Compared with GRU used by cyclic neural networks, such as Rapid, TCN is parallel, and has no gating unit. Therefore, the implementation timings of TCN are short. It can learn longer historical information and have the advantage of the strong feature extraction ability of a convolutional neural network [39]. The size of receptive field can be adjusted with the number of network layers, expansion factor, and convolution kernel size, and the memory size can be flexibly controlled. And the gradient in TCN is more stable because of residual connection. Moreover, TCN can accept input of any length via sliding a one-dimensional convolution kernel. TCN structure is simpler and clearer than RNN structure. To sum up, TCN is superior to RNN networks such as LSTM and GRU. Therefore, the TLW algorithm uses the TCN model to realize feature extraction. Multi-neurons and multi-layer residual blocks in TCN can extract more accurate and deeper related features that are difficult to find manually under the condition of small sample data and sparse time series data.

Compared with GRU, the LightGBM algorithm is more suitable for dealing with high-dimensional data, and a decision tree combined with gradient boosting algorithm can effectively deal with nonlinear relationships and complex feature interactions, providing more accurate prediction results. And it is robust to missing values and abnormal values. Some abnormal values are the points values where the slope of the light curve jumps abnormally, the points values with a high outlier rate, and the points values beyond three standard deviations.

To sum up, the TLW algorithm takes advantage of TCN, which not only effectively extracts the time series characteristics of the light curve of the transient but also overcomes the shortcoming that LightGBM ignores the correlation between the attributes in the data set, which improves that accuracy of the algorithm. And Residual blocks in the algorithm solve the problem that convolutional networks are easy to over-fit. The TLW algorithm also has the advantages of LightGBM, including high accuracy and strong robustness. GOSS, FEB, and other optimization algorithms make the TLW algorithm have the advantages of high speed, high efficiency, and low memory occupation [40]. Considering the characteristics of a short timescale of the transient, the Weight module is introduced to solve the problem that the signal source disappears prematurely, and the noise is misjudged as a feature. In other words, the Weight module makes the algorithm have the advantage of strong anti-noise capability.

However, several optimization algorithms in the TLW algorithm will abandon some feature information. When the sample ensemble size is small, the feature set used for model training will be incomplete, which will affect the accuracy. Therefore, the TLW algorithm is more suitable for early classification of transients with a large number of light curves.

3. TLW Algorithm Experiment

3.1. Sample Data

The sample data used in the experiment are the open-source multi-band transient simulation data [42] provided via the LSST astronomical photometry time series classification challenge PLAsTiCC. The PLAsTiCC data set provides the simulated photometric

time series data of multi-class astronomical transients and variables what we expect from the 8 m telescope with a 3 billion pixel camera on LSST [42]. The simulated time series data are a function of the object’s brightness with time via measuring the photon flux in six different astronomical filters on LSST. In addition, the data set also provides the simulated astronomical information header file about each simulated object, including right ascension (RA), declination (DEC), galactic longitude (l), galactic latitude (b), redshift, extinction value (MWEBV), and so on [42]. We use the simulated photometric transient data of g, r, i bands in the PLAsTiCC data set because these three bands are consistent with the ground-based telescope observation bands of the SVOM mission. Among them, the main samples of transient celestial bodies are Tidal Disruption Events (TDEs), Kilonovae (KN), Type Ia supernova (SNIa), and Type I Super-luminous supernovae (SLSN-I), and an example of their g, r, and i bands photometric time series data is shown in Figure 2. In this paper, some objects without obvious transient signal peaks are removed from the data of the PLAsTiCC open-source training set. Because these time series data of objects have no obvious features of light curves of transients (such as signal peak), the data quality is poor. If the model learns poor quality data, its accuracy will be reduced. The remaining objects’ classes labels; modified Julian day time (MJD) series in g, r, and i bands; flux series and flux error series; redshift value (z); extinction value (MWEBV); and galactic latitude (b) are counted. The sample set includes 40 TDE, 42 KN, 30 SNIa, and 175 SLSN-I.

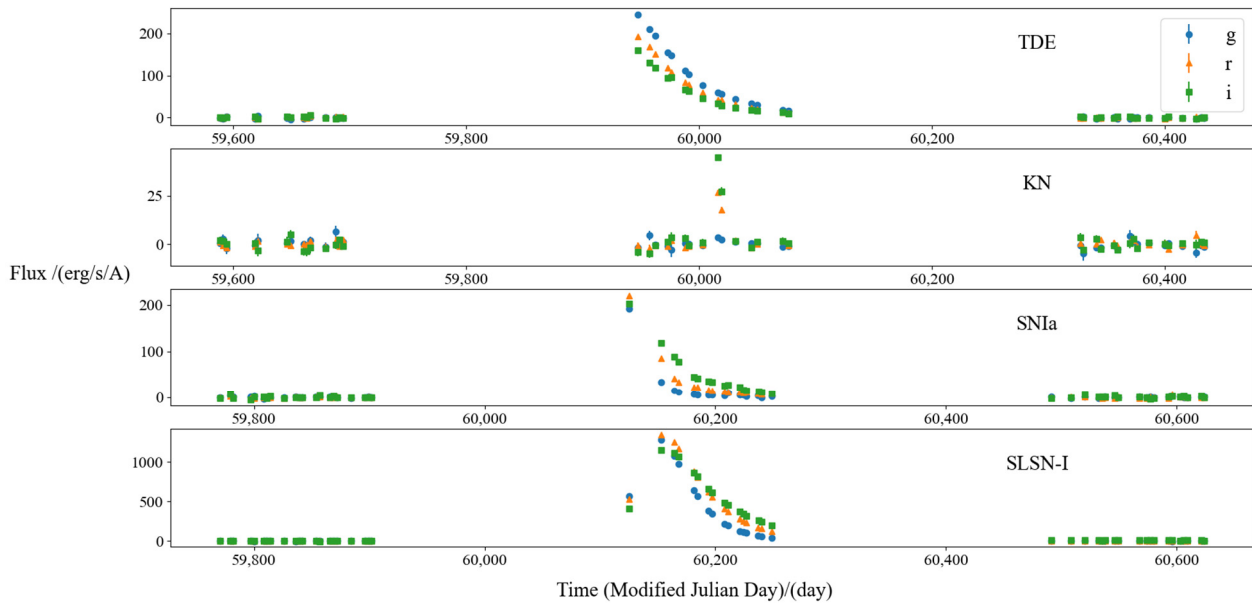


Figure 2. Examples of transients. They are the simulation data of PLAsTiCC. The horizontal axis is the modified Julian day time, the unit of which is days. The vertical axis is the original flux, the unit of which is erg/s/A.

3.2. Experiment

We make use of the fitting method of light curves from reference [3], and the complete experimental flow is shown in Figure 3. The main steps are as follows:

Step 1: Correct the time, that is changing the MJD time into a time interval sequence when the trigger time MJD_0 is 0. Then we use the redshift z to correct the rest-frame time caused by the expansion of cosmic time, as shown in Equation (9).

$$t = \frac{MJD - MJD_0}{1 + z} \tag{9}$$

Step 2: Use the fluxerr for 3 sigma clipping and repeat the clipping 5 times.

Step 3: Use the extinction function of Fitzpatrick [43] to de-redden the galaxy and correct the flux.

Step 4: Establish the light curve model, as shown in Equation (10).

$$L_m^\lambda(t; t_0, a^\lambda, c^\lambda) = [a^\lambda(t - t_0)^2] \cdot H(t - t_0) + c^\lambda, \quad (10)$$

where λ is the g, r, and i bands; L_m^λ is the analog luminosity of λ band; t_0 is the explosion time of the transient object; a^λ and c^λ are constants; and $H(t - t_0)$ is the Heaviside function.

The parameter results of t_0 , a^λ , and c^λ are estimated using the chi-square function χ^2 , as shown in Equation (11).

$$\chi^2(t_0, a^\lambda, c^\lambda) = \sum_\lambda \sum_{t=-\infty}^{t_{\text{peak}}} \frac{[L^\lambda(t) - L_m^\lambda(t; t_0, a^\lambda, c^\lambda)]^2}{[\sigma^\lambda(t)]^2}, \quad (11)$$

where $L^\lambda = (f - f_{\text{med}}) \frac{d^2}{10^{18}}$ is the real luminosity value converted from flux f and median flux f_{med} , d is the luminosity distance calculated using the cosmological constant, σ^λ is the luminosity error of L^λ , and t_{peak} is the corresponding time of peak flux.

Step 5: Fitting the light curve, which includes corrected time, corrected flux, and t_0 .

Step 6: The data augmentation method proposed in Boone [18] is adopted to expand the data set and solve the problem of unbalanced sample categories. Gaussian process model fitting is carried out on the optical curve of the existing sample, which is down-sampled, some points on the light curve are randomly selected for many times, and noise is added to generate a group of synthetic optical curves with similar characteristics to the original curve. The number of synthetic data of different categories is inversely proportional to the original data to solve the problem of sample imbalance. As shown in Figure 3, the curves in the figure are Gaussian fitting diagrams of light curve time and flux, the shaded parts are Gaussian fitting parts of time and fluxerr, and the dots are the points of the down-sampled synthetic light curve.

Step 7: Delete the full light curves of samples which have high redshift ($z > 0.5$) and have few points on the light curve; delete samples of less than 3 single-band points, galactic latitude $|b| < \pm 15^\circ$; and delete samples of less than 4 epochs [2].

Step 8: Selecting temporal segments of the time series, we cut the series data with $t < -70$ days and $t > 80$ days. Perform uniform sampling on the rest of the time series ($-70 \leq t \leq 80$) with an interval of 3 days to obtain the time points of the new series. Linear interpolation is carried out via using the function of flux and time, and the flux values of three bands at each time point are obtained. Sampling and interpolation can solve the problem of sparse data, and cutting the series can solve the problem of excessive memory.

Step 9: Establish a matrix with the size of 50×4 as the model input matrix, in which the first three columns are the flux of the interpolated light curve in different bands at each time point. If the length of a curve is less than 50, the rest is filled with 0. The 50 elements in the fourth column of the matrix are all repeated values of redshifts. Establish a matrix with the size of 50×5 as the model label matrix. The binary code is used as the label. For example, the label of pre-explosion is [1, 0, 0, 0, 0], the label of TDE is [0, 1, 0, 0, 0], etc. Same as the input matrix operation, if the curve length is less than 50, the rest is filled with 0.

Step 10: Divide the data set into 80% training set and 20% testing set and train the TLW model. It can real-time classify transients which have light curves with different lengths. Obtain the prediction result of the object at each time point.

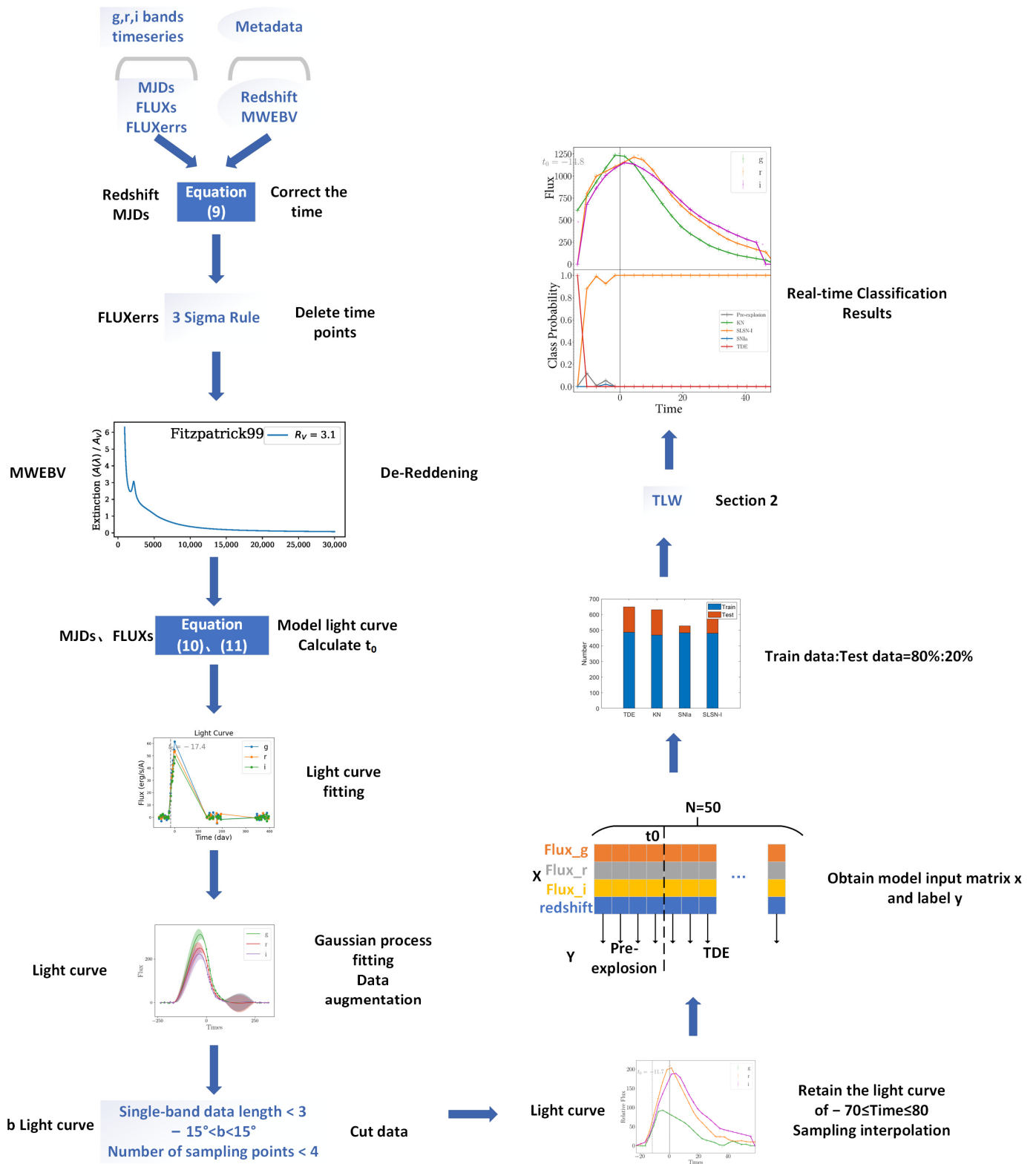


Figure 3. Experimental steps. The parameters needed are at the left of the step, and the functions of the step are at the right.

4. Result Analysis

All algorithms are trained on TensorFlow architecture in Windows and Python. We use data augmentation technology to obtain four groups of data sets from simulated PLAsTiCC data sets, which are all stratified on class balance. There are 362 samples in Group I,

1195 samples in Group II, 2390 samples in Group III, and 3218 samples in Group IIII. Each group is randomly divided into 80% training set and 20% testing set. After testing the parameters of four training sets, we find that the optimal parameters of the four models are similar. The four groups of models are trained via their own training sets with the same hyperparameters. We mainly adjust the parameters of the number of neurons in each layer of TCN module (Neurons number), the number of LightGBM trees (Tree number), and the parameters of α_f and β . The result of the parameter adjustment is shown in Figure 4. We show the results of Group III with the highest accuracy under different parameters. When the Neurons number is too small, we extract fewer features which cannot fully learn the original features of the light curves, so the accuracy is small. When the Neurons number is too large, too many features are extracted, which is easy to over-fit. When the Tree number is too small, fewer decision trees are boosting, and the model will be under-fit. When the Tree number > 50 , the difference in model accuracy is small. So, we choose the parameter Tree number = 200 when the accuracy is the highest. α_f , β , and other parameters (such as max depth) have little influence on the model accuracy. To sum up, the Neurons number is 100, the Tree number is 200, α_f is 0.1, and β is 0.1.

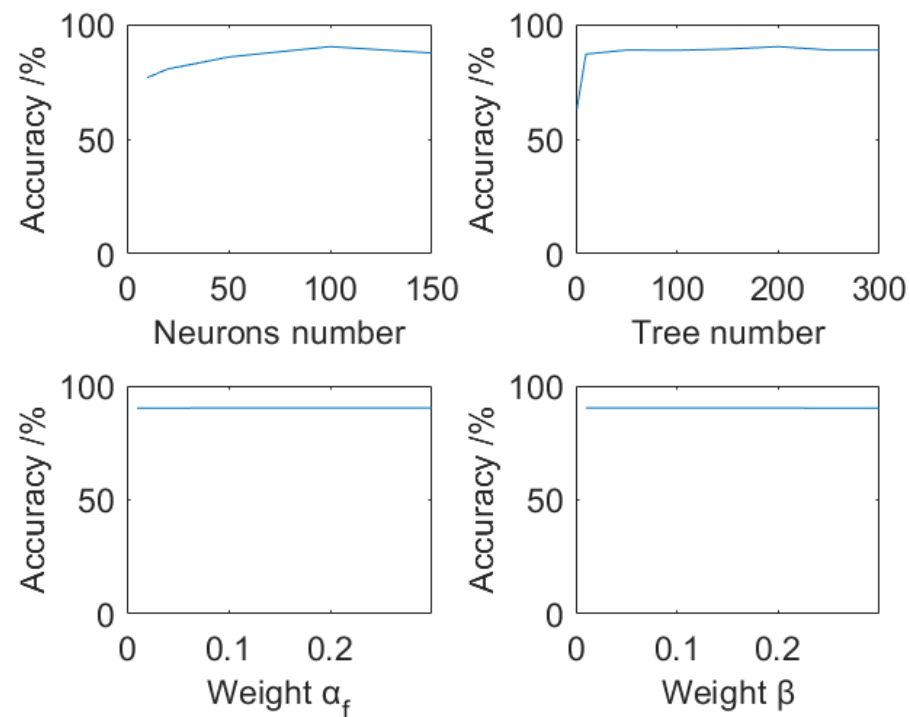


Figure 4. The result of parameter adjustment. On the upper left is the diagram of Accuracy vs. Neurons number. On the upper right is the diagram of Accuracy vs. Tree number. On the bottom left is the diagram of Accuracy vs. α_f . On the bottom right is the diagram of Accuracy vs. β .

After completing the experiments in Section 3.2, the classification results of object transients at each moment can be obtained, and the function of automatic real-time classification can be realized. An example of the results is shown in Figure 5. The upper figure shows the light curve of the object after correction, and the lower figure shows the probability of the object classes predicted at each moment. The class with the highest probability is considered as the object class. In Figure 5, the same example objects using two kinds of algorithms all achieve correct classification in a short time since the trigger, which can prove that the experiment is reasonable and effective. In order to eliminate the interference of other factors, we introduce the same data into TLW and Rapid respectively after using the same fitting light curve method and same preprocess. As shown in Figure 5, the TLW algorithm can correctly classify the example object as SLSN-I before reaching the peak of flux, and its SNSL-I class probability score is much higher than other classes. The

TLW algorithm can correctly classify the transient object earlier than the Rapid algorithm. And in the TLW algorithm, the class probability of SLSN-I is greater than 0.8, while that of the Rapid algorithm is less than 0.6. And the object is wrongly classified as SNIa when Time = 35 days. In addition, the classification probability of the correct SLSN-I in the Rapid algorithm is lower than that in the TLW algorithm, and the classification probability of the wrong SNIa is higher. As far as this example, the TLW algorithm shows better classification than the Rapid algorithm.

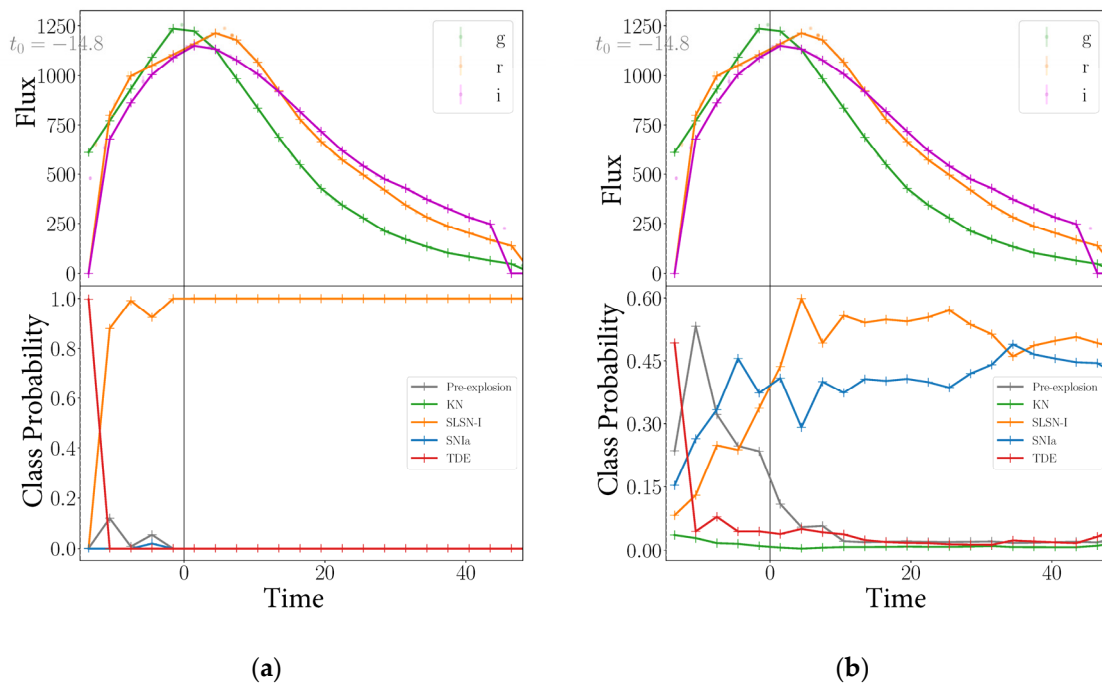


Figure 5. Classification result example diagram. (a) TLW; (b) Rapid.

In order to assess the performance of the TLW algorithm, we use TLW, LSTM, Transformer, CNN, RF, Rapid, and another classification algorithm proposed by our group, the Temporary Convective Network and Extreme Gradient Boosting Combined with Weight Module Algorithm (TXW), to compare four groups of data sets with different sample ensemble sizes of transients. The same data are used to train the above seven classification models respectively. In order to reduce the influence of other factors, we adopt the same experimental process and only use different models. We count the classification results of each transient at each moment and take the ratio of the number of correctly classified results to the total number as the accuracy of the algorithm. The accuracy and implementation timings of the algorithm are shown in Tables 1 and 2.

The results of four groups show that the accuracy of TLW is 11.55%, 21.05%, 25.15%, and 21.39% higher than that of LSTM, with an average increase of 19.79%, and the implementation timing of TLW is 5.17%, 86.95%, 86.55%, and 87.42% less than that of LSTM, with an average decrease of 86.52%. Compared with Transformer, the accuracy of the TLW algorithm is improved by 10.77%, 16.05%, 20.18%, and 20.85% respectively, with an average increase of 16.97%, and the implementation timing is reduced by 14.71%, 30.79%, 19.81%, and 11.99%, with an average decrease of 19.33%. Compared with CNN, the accuracy of the TLW algorithm is improved by 7.48%, 17.24%, 20.35%, and 20.36%, with an average increase of 16.36%, and the implementation timing is reduced by 91.63%, 94.21%, 93.8%, and 94.18%, with an average decrease of 93.46%. Although the efficiency of TLW is lower than that of RF and Rapid, its accuracy is higher. The accuracy of TLW is 4.25%, 11.07%, 13.02%, and 14.15% higher than that of RF, with an average increase of 10.63%. The accuracy of TLW is 4.38%, 5.74%, 7.46%, and 1.4% higher than that of Rapid, with an average increase of 4.75%. The TLW algorithm will discard some feature information. When the sample ensemble size

is small, TLW is difficult to obtain a complete feature set, which will affect the accuracy. Therefore, when TLW is applied to Group I and Group II with small sample ensemble sizes, the accuracy values of results are slightly lower than those of TXW. With the increase in sample ensemble size, the classification results of Group III show that the accuracy of TLW is 1.88% higher than that of TXW, and the classification results of Group III show that the accuracy of TLW is 1.17% higher than that of TXW. It is due to the fact that the leaf-wise growth strategy can generate more complex trees, thus improving the accuracy of the algorithm. The efficiency of TLW is higher than that of TXW. Compared with TXW, the implementation timing of TLW algorithm is reduced by 2.6%, 19.89%, 15.22%, and 19.6%, with an average decrease of 14.33%. Although the sample ensemble size of Group III is larger, the sample data are mixed with more noise, and the data quality is worse than that of Group II and Group III, so the classification accuracies of different algorithms applied to Group III are mostly lower than those of Group II and Group III. However, the sample ensemble size of Group I is too small, so the influence of sample ensemble size factor on the result accuracy exceeds that of the noise factor. Therefore, in order to improve the accuracy of the model, we not only need to expand the sample ensemble size of the data set but also need to select the training data with high quality and remove the samples polluted with excessive noise.

To sum up, the average accuracy of TLW is 84.54%, which is the highest among the seven algorithms, and the average implementation timing is 123.09 s. The results can prove that the TLW algorithm has the best comprehensive performance and has the advantages of high precision and high efficiency. The TLW algorithm includes many effective techniques to mitigate over-fitting, such as residual block, dropout layer, leaf-wise tree growth strategy with depth limitation, L2 regularization in Equation (4), etc. We also use a variety of techniques to mitigate under-fitting, such as using the TCN module, which can extract a large number of features; using a boosting algorithm (LightGBM); and using a data augmentation method. The training set accuracy and testing set accuracy of the TLW model are both high, which shows that the model is low deviation and low variance. The loss vs. training epoch diagrams of TLW and Rapid is shown in Figure 6. The loss function of Rapid and TCN module in TLW is categorical cross entropy, and the loss function of the LightGBM module and Weight module is softmax. When the epoch is 100, the Rapid and the TCN module in TLW has approximate minimum loss, and the Rapid model and TLW feature extraction model are trained to be optimal. The features extracted from the TCN module are used to train the LightGBM module and Weight module in TLW. When the number of decision trees is 200, loss is the lowest, and the TLW model is trained to the optimal. In the results of repeated training, the minimum loss values of the models are similar. We choose the minimum loss values as the minima of the training and testing loss function.

Table 1. Comparison of classification accuracy of different algorithms.

	I	II	III	IIII	Average
LSTM Accuracy	68.03%	63.63%	65.20%	62.14%	64.75%
Transformer Accuracy	68.81%	68.63%	70.17%	62.68%	67.57%
CNN Accuracy	72.10%	67.44%	70.00%	63.17%	68.18%
RF Accuracy	75.33%	73.61%	77.33%	69.38%	73.91%
Rapid Accuracy	75.20%	78.94%	82.89%	82.13%	79.79%
TXW Accuracy	79.99%	84.73%	88.47%	82.36%	83.89%
TLW Accuracy	79.58%	84.68%	90.35%	83.53%	84.54%

Table 2. Comparison of implementation timings of different algorithms.

	I	II	III	IIII	Average
LSTM Time/s	166.92	506.36	1294.99	1806.92	3775.19
Transformer Time/s	29.02	95.5	217.13	258.35	150
CNN Time/s	295.80	1141.5	2810.44	3905.41	2038.29
RF Time/s	0.91	4.92	10.79	16.8	8.36
Rapid Time/s	18.07	63.22	163.91	204.44	112.41
TXW Time/s	25.41	82.51	205.38	282.80	149.03
TLW Time/s	24.75	66.1	174.12	227.37	123.09

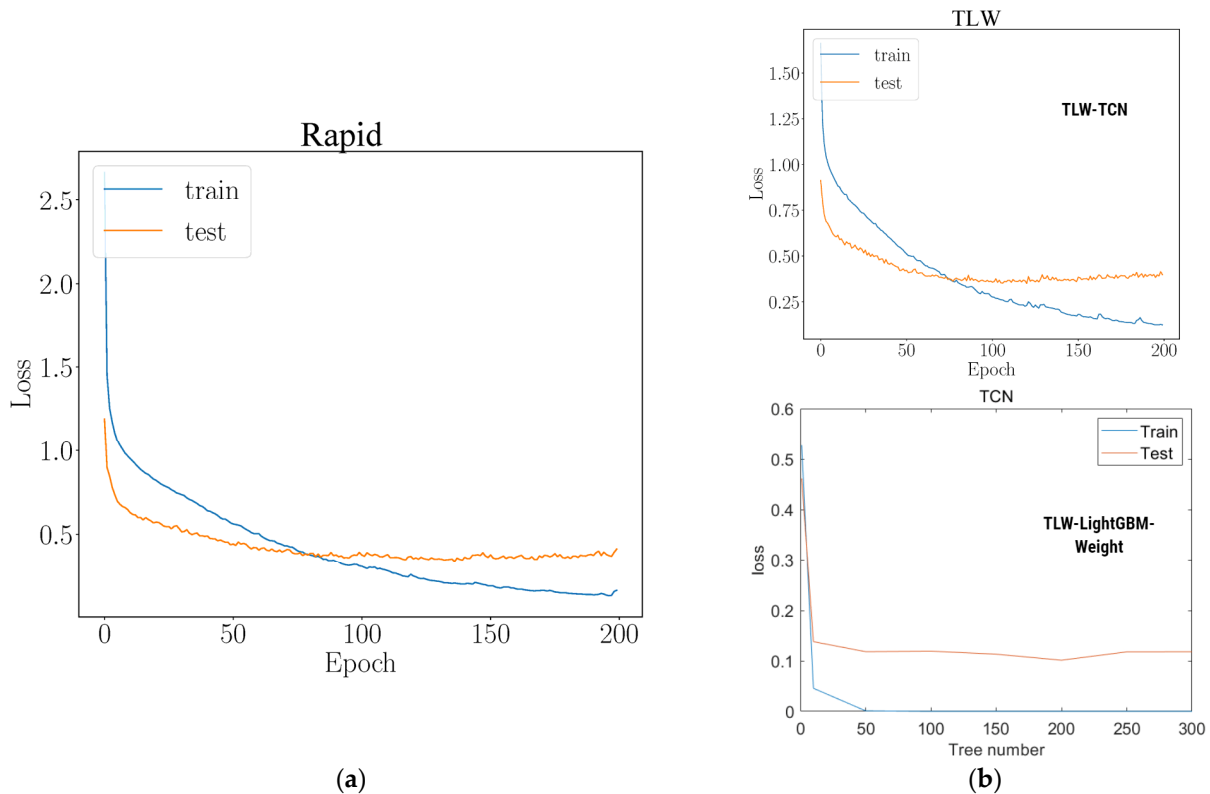


Figure 6. The loss vs. training epoch diagrams. (a) Rapid; (b) TLW. The upper diagram is the loss vs. training epoch diagram of the TCN module of the TLW algorithm. The upper diagram is the loss vs. training epoch diagram of the TCN module in the TLW algorithm. The lower diagram is the loss vs. number of decision trees of the LightGBM and Weight module in the TLW algorithm.

The TLW algorithm can not only classify the real-time light curves of transients but also obtain high-precision classification results in 1–2 days since trigger time MJD_0 ($t = 0$), which can be used for early classification. In order to evaluate the performance of the TLW real-time classification algorithm, this paper selects the Group III data set with the highest accuracy and uses Confusion Matrix, Precision-Recall (PR) curve and Receiver Operating Characteristic (ROC) curve as the performance indicators of the algorithm. The classification results of the TLW algorithm and Rapid algorithm were compared and analyzed at 1, 4, 8, 15, 20, 25, and 30 days since the trigger. The results show that the comprehensive performance of the TLW algorithm is better than that of Rapid algorithm.

Among them, Confusion Matrix is shown in Figures 7–13, and the numerical values of cells represent the proportion of each real label classified as a prediction label. The bluer the cell color, the higher the correct rate and the redder the error rate. The results show that at 1, 4, 8, 15, 20, 25, and 30 days since the trigger, the classification results of the TLW algorithm are higher than those of the Rapid algorithm. In the result of applying the TLW algorithm, the classification accuracy of SNIa, TDE, and SNSL-I is equal to or more than 0.5 at 1 day

since the trigger. Because the decay rate of brightness of KN is too fast, the light curve we extracted contains fewer points with signal feature of transient than other classes, so some KNs are misjudged as pre-explosion. With the increase in time, the points with signal contained in the optical curve increase, and the accuracy of various transients is improved. At 30 days since the trigger, the accuracy of KN also reached 0.89, and the accuracy of the other three categories exceeded 0.9. However, the accuracy of various classes of the Rapid algorithm is relatively low, and KN classes are not correctly classified. It may be because the Group III data set only contains 2390 samples, while the Rapid paper uses tens of thousands of samples, which leads to the low accuracy of the Rapid model. Due to the noise mentioned in Section 2.3, the accuracy of the results of the Rapid algorithm is the highest on the 20 days since the trigger, and the accuracy of the results decreases on the 25 and 30 days since the trigger. The TLW algorithm solves this problem. The experimental results show that the TLW algorithm has a high accuracy and is anti-noise, and high-accuracy results can be obtained in the early stage of transient.

The horizontal axis of PR curve is recall rate R (Recall), and the vertical axis is precision rate P (Precision). P represents the accuracy of predicting the correct positive samples, which is defined as $P = \frac{TP}{TP+FP}$, where TP is True Positive, the classifier predicts positive samples and is actually positive samples. FP is False Positive, and the classifier predicts a positive sample, but it is actually a negative sample. R represents the coverage of positive samples with correct prediction and is defined as $R = \frac{TP}{TP+FN}$, where FN is False Negative, and the classifier predicts negative samples, which are actually positive samples. A high-performance algorithm should have both high accuracy and high recall. In order to analyze PR curve, we give the average accuracy (AP) below the curve and summarize PR curve into a single value representing the average of all the accuracy.

$$AP = \sum_{k=0}^{k=n-1} [R(k) - R(k+1)] \times (k),$$
 where n is the threshold number of PR curve. The larger the AP, the higher the accuracy and the better the performance of the algorithm. As shown in Figures 14–20, the micro-average AP of TLW is 0.21, 0.21, 0.19, 0.17, 0.2, 0.2, and 0.19 higher than that of Rapid at 1, 4, 8, 15, 20, 25, and 30 days since the trigger. The AP of KN is 0.21, 0.3, 0.32, 0.17, 0.19, 0.19, and 0.09 higher respectively. The AP of SLSN-I is improved by 0.25, 0.22, 0.16, 0.12, 0.2, 0.2, and 0.19 respectively. The AP of SNIa is improved by 0.16, 0.22, 0.29, 0.17, 0.18, 0.21, and 0.28 respectively. The AP of TDE is improved by 0.05, 0.05, 0.01, 0.06, 0.04, 0.07, and 0.05 respectively. With the increase in the number of days since the trigger, the micro-average AP values of both algorithms increase, which also proves that the more complete the light curve, the better the classification results. Moreover, the AP of the four classes of TLW is higher than Rapid at each moment.

The horizontal axis of ROC curve is False Positive Rate (FPR), and the vertical axis is True Positive Rate (TPR). TPR represents the proportion of samples that are actually positive, and it is defined as $TPR = \frac{TP}{TP+FN}$, and TPR and R are the same. FPR represents the proportion of samples that are actually negative and are wrongly predicted to be positive, and it is defined as $FPR = \frac{FP}{FP+TN}$, TN is True Negative, and the classifier predicts negative samples, which are actually negative samples. The higher the TPR and the lower the FPR, the better the performance of the algorithm. In order to analyze ROC curve, we give the area under the ROC curve (AUC) below the curve, and the larger the AUC value the better. Macro-average value is the average of all ROC-like curves. Micro-average value is the weighted average of ROC curve considering the number of each class. As shown in Figures 21–27, compared with Rapid, ROC curve shows that the micro-average AUC of TLW is improved by 0.12, 0.11, 0.09, 0.06, 0.07, 0.08, and 0.07 respectively, and the macro-average AUC is improved by 0.12, 0.1, 0.08, 0.06, 0.07, and 0.06 respectively at 1, 4, 8, 15, 20, 25, and 30 days since the trigger. The AUC of KN improved by 0.21, 0.21, 0.13, 0.07, 0.08, 0.09, and 0.01; the AUC of SLSN-I improved by 0.16, 0.12, 0.09, 0.06, 0.09, 0.08, and 0.08; the AUC of SNIa improved by 0.1, 0.11, 0.12, 0.1, 0.08, 0.09, and 0.11; and the AUC of TDE improved by 0.08, 0.07, 0.04, 0.06, 0.06, 0.07, and 0.05. To sum up, the AUC of four classes in TLW is higher than Rapid at every moment.

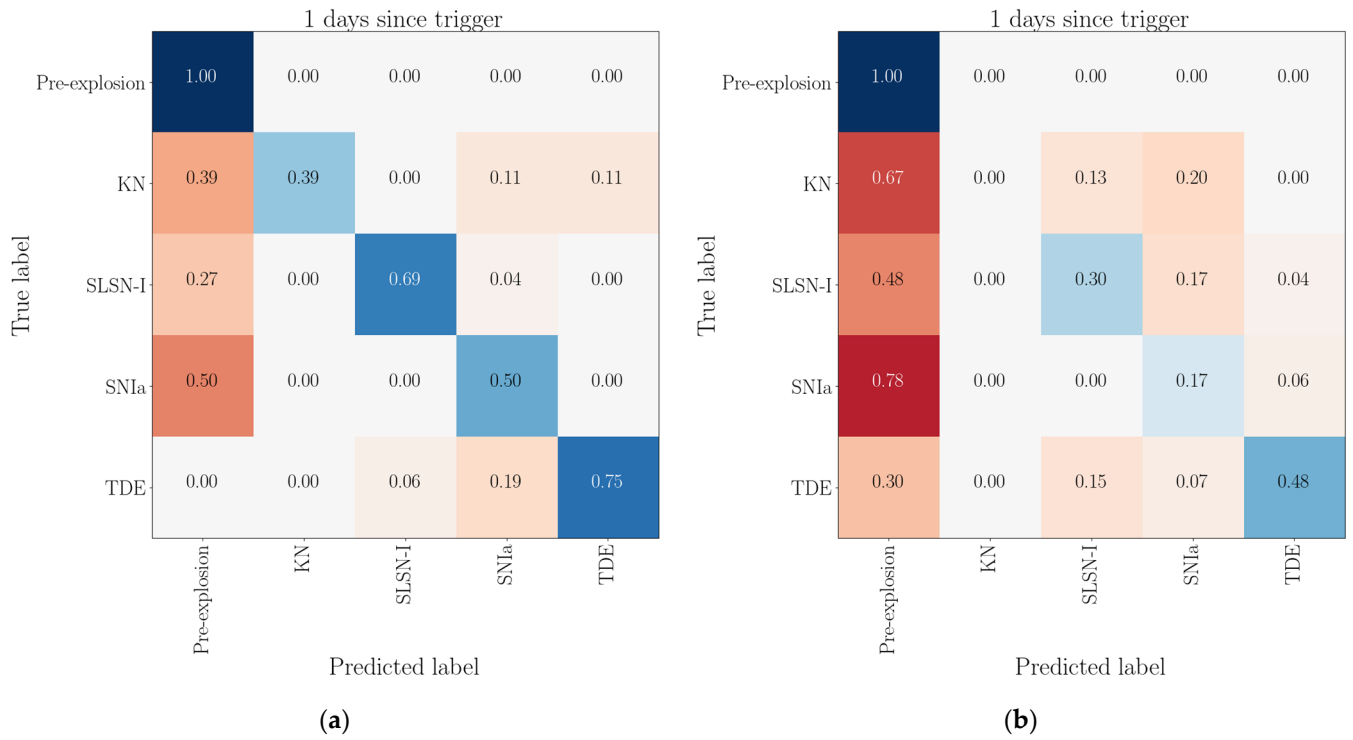


Figure 7. Confusion Matrix at 1 day since trigger. (a) TLW; (b) Rapid.

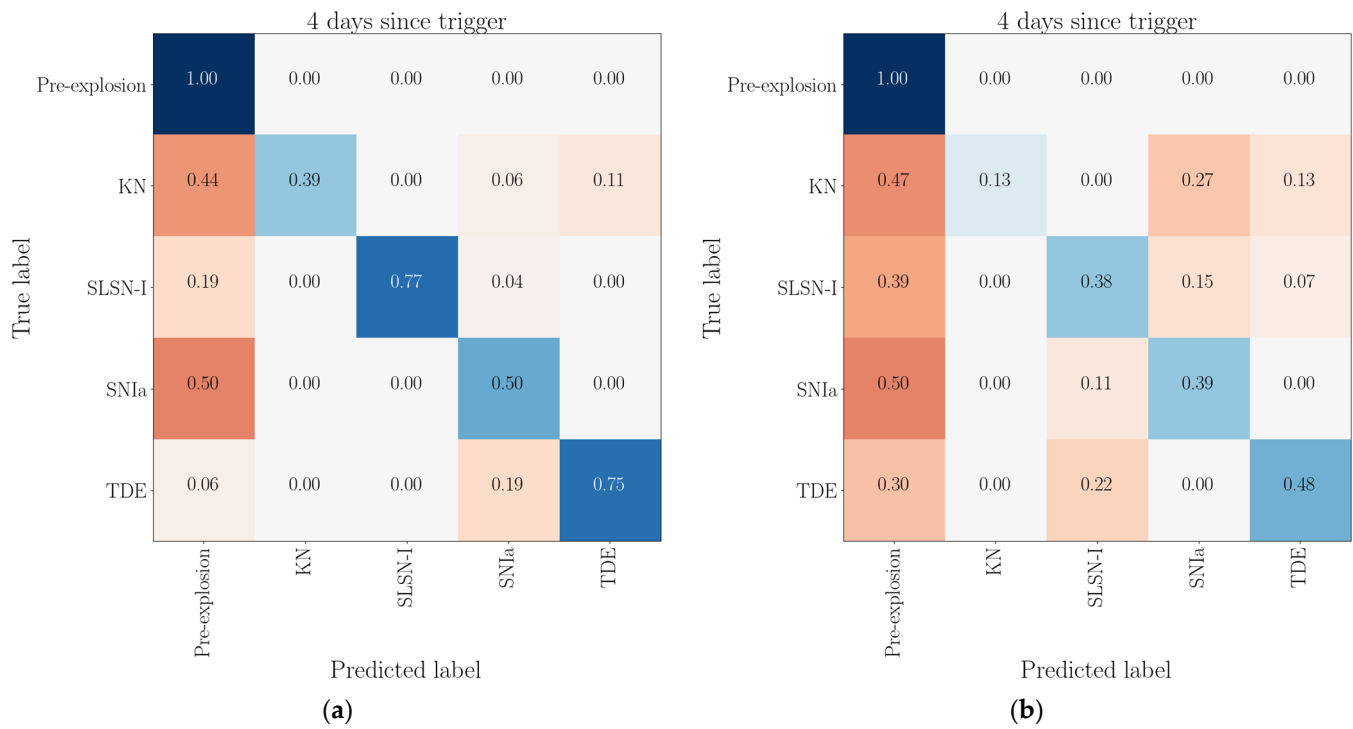


Figure 8. Confusion Matrix at 4 days since trigger. (a) TLW; (b) Rapid.

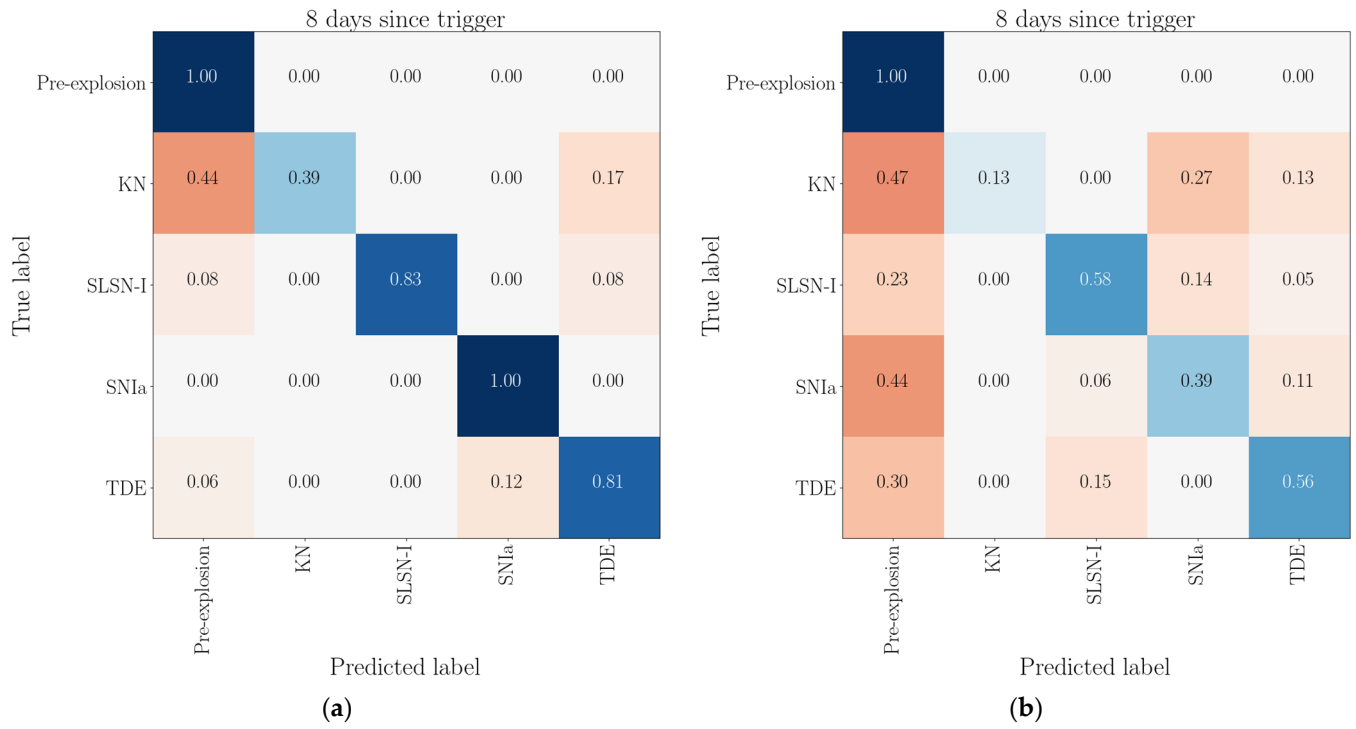


Figure 9. Confusion Matrix at 8 days since trigger. (a) TLW; (b) Rapid.

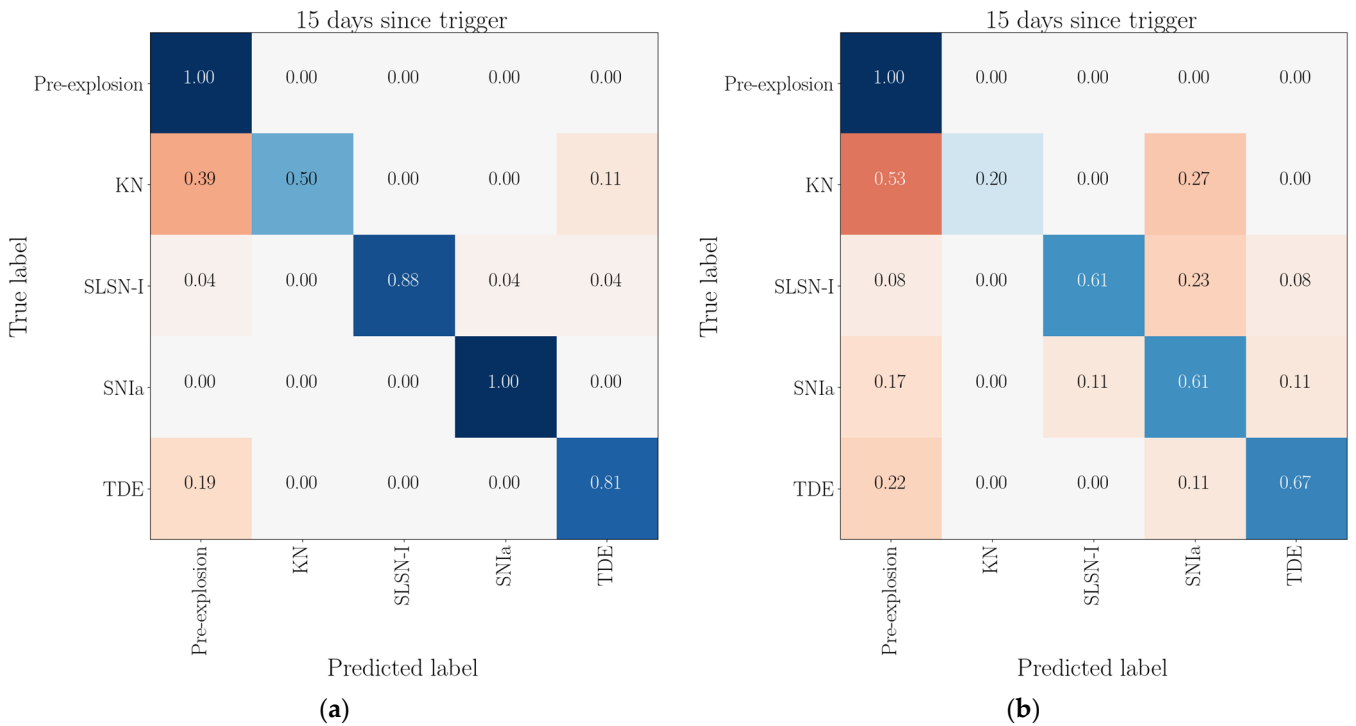


Figure 10. Confusion Matrix at 15 days since trigger. (a) TLW; (b) Rapid.

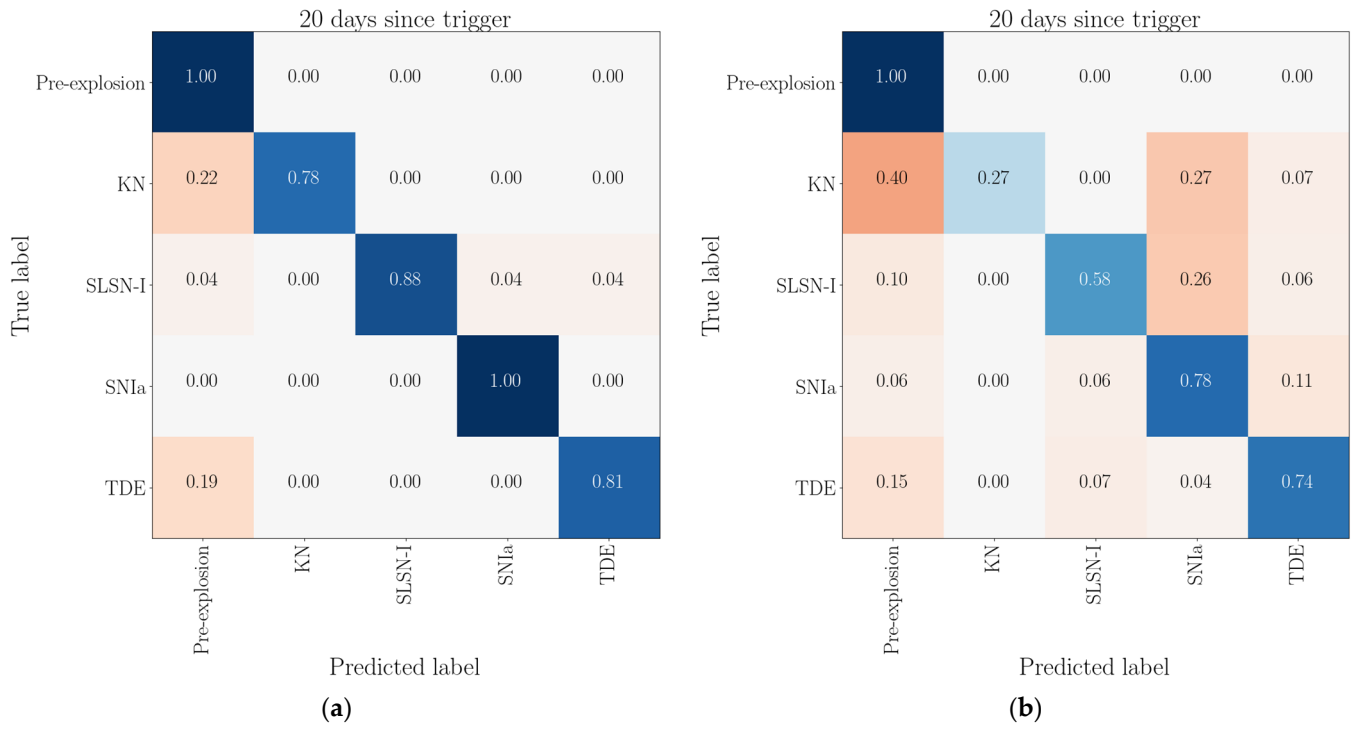


Figure 11. Confusion Matrix at 20 days since trigger. (a) TLW; (b) Rapid.

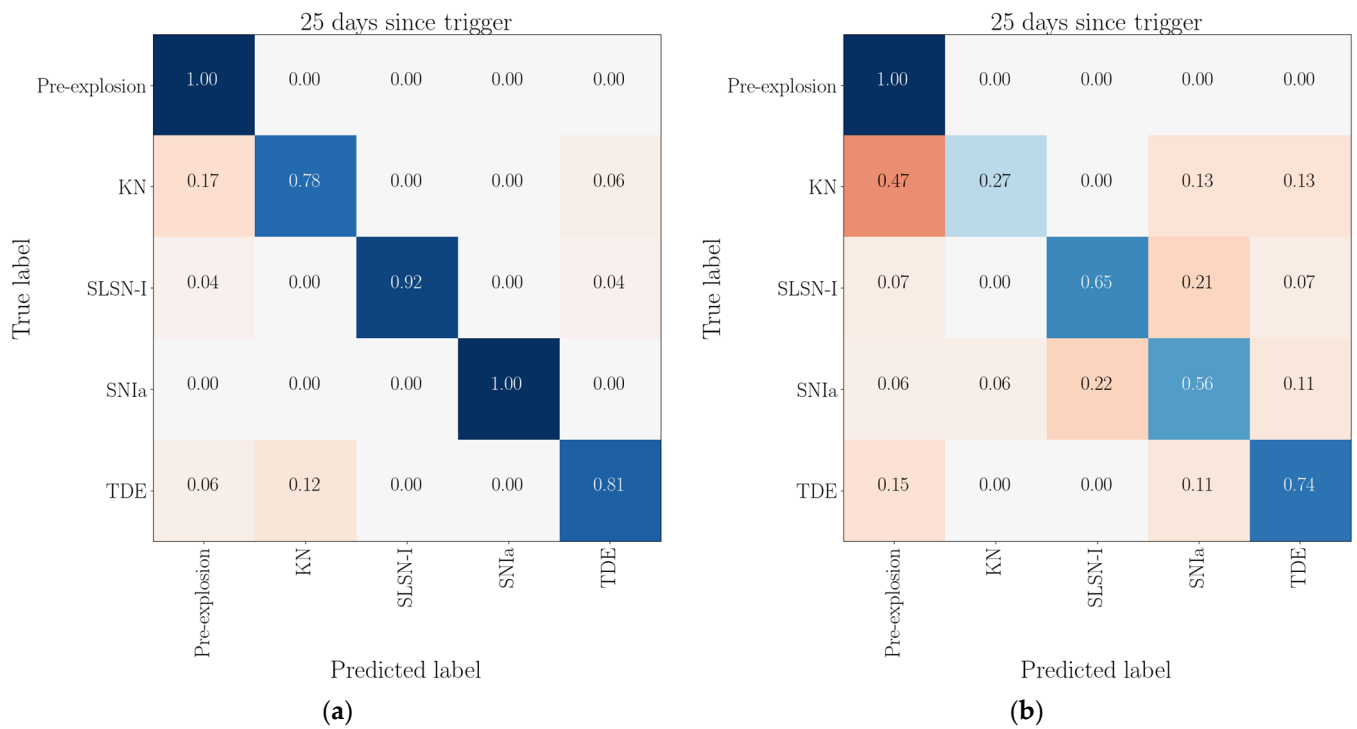


Figure 12. Confusion Matrix at 25 days since trigger. (a) TLW; (b) Rapid.

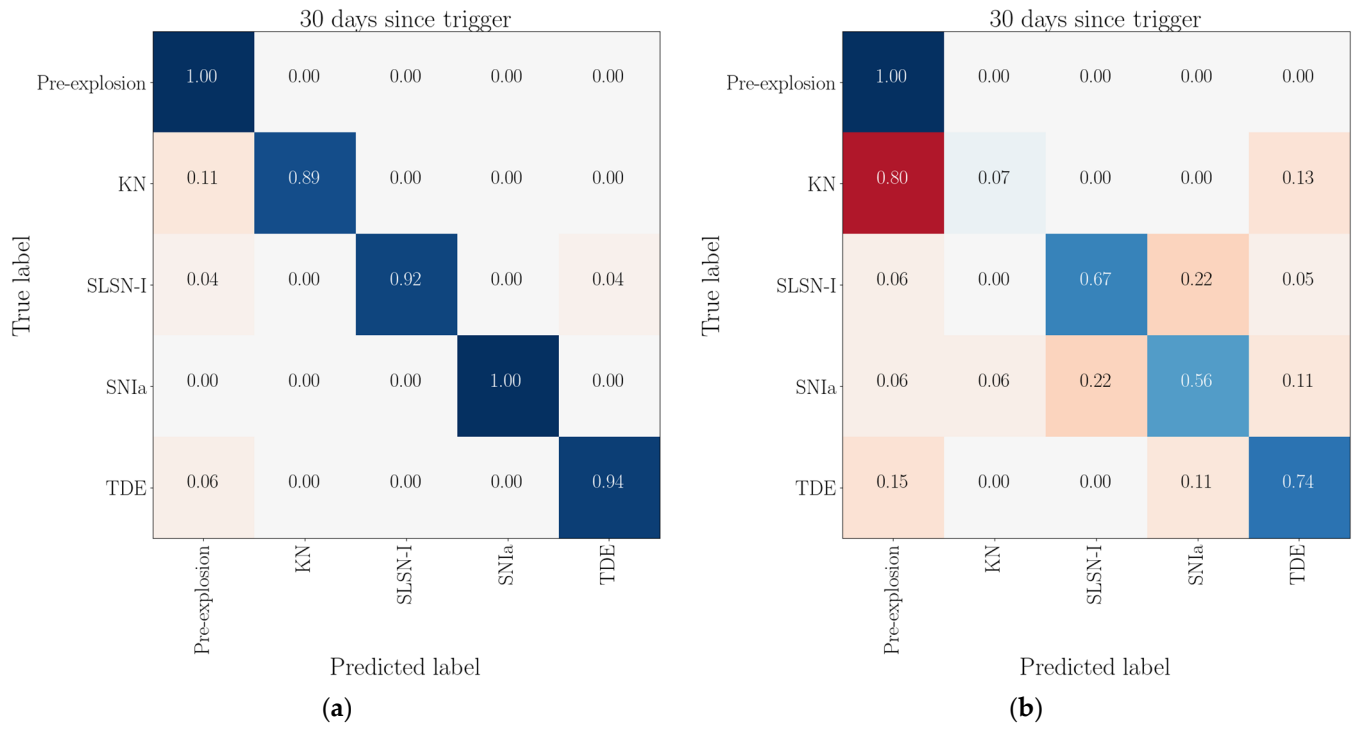


Figure 13. Confusion Matrix at 30 days since trigger. (a) TLW; (b) Rapid.

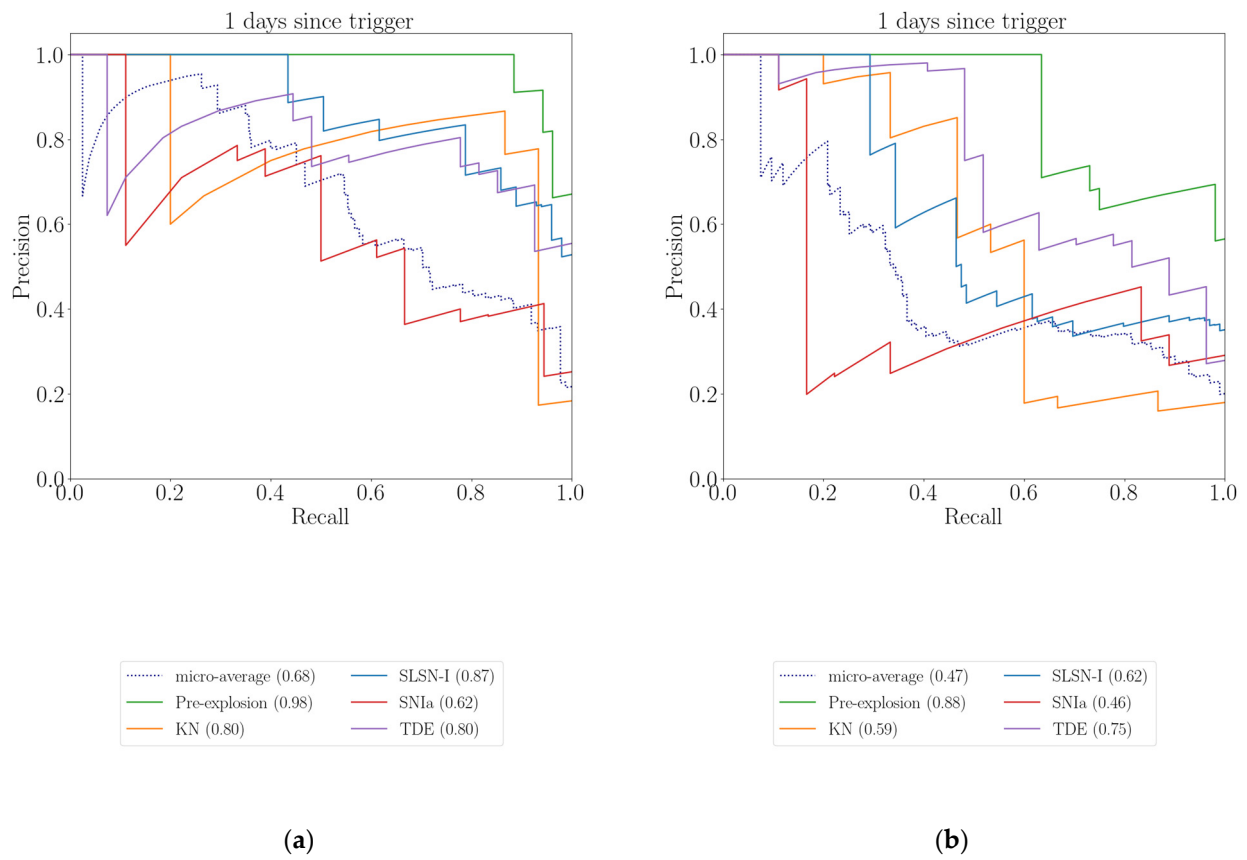


Figure 14. PR curve at 1 day since trigger. (a) TLW; (b) Rapid.

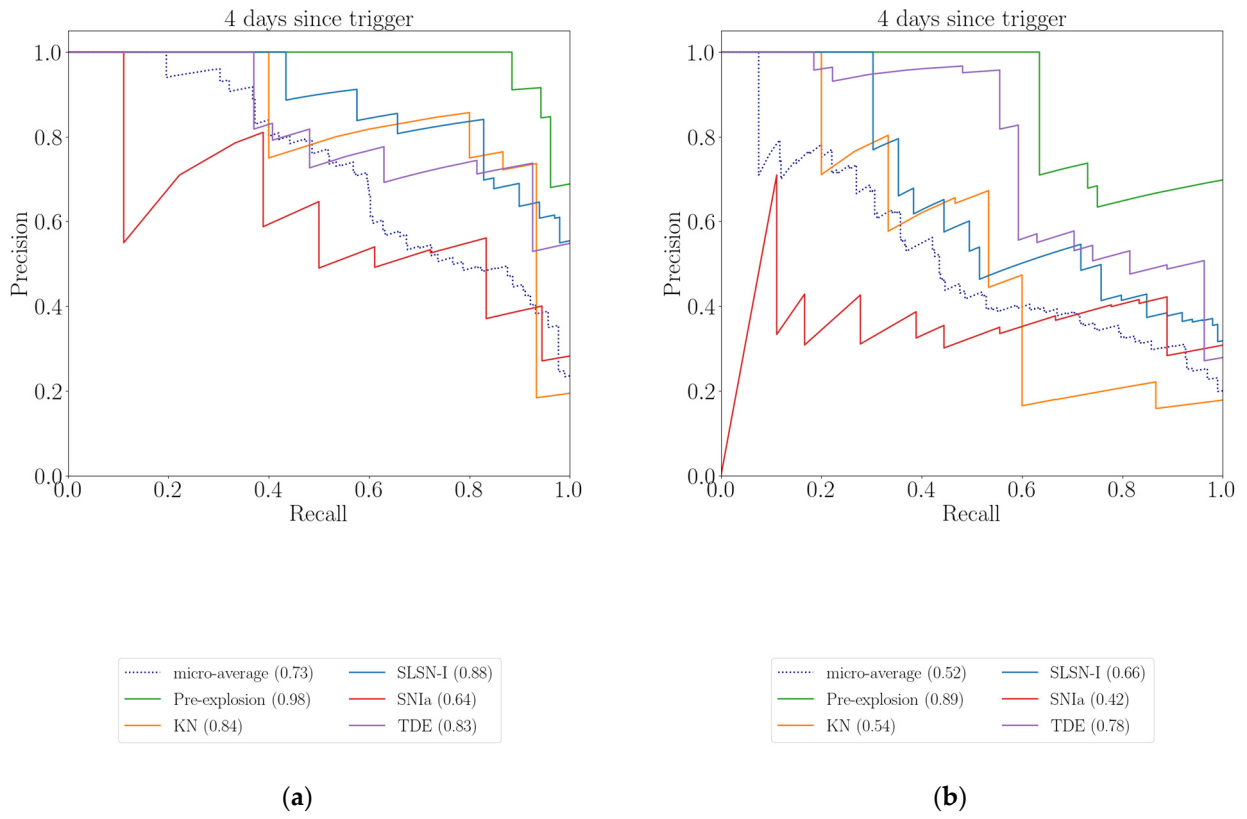


Figure 15. PR curve at 4 days since trigger. (a) TLW; (b) Rapid.

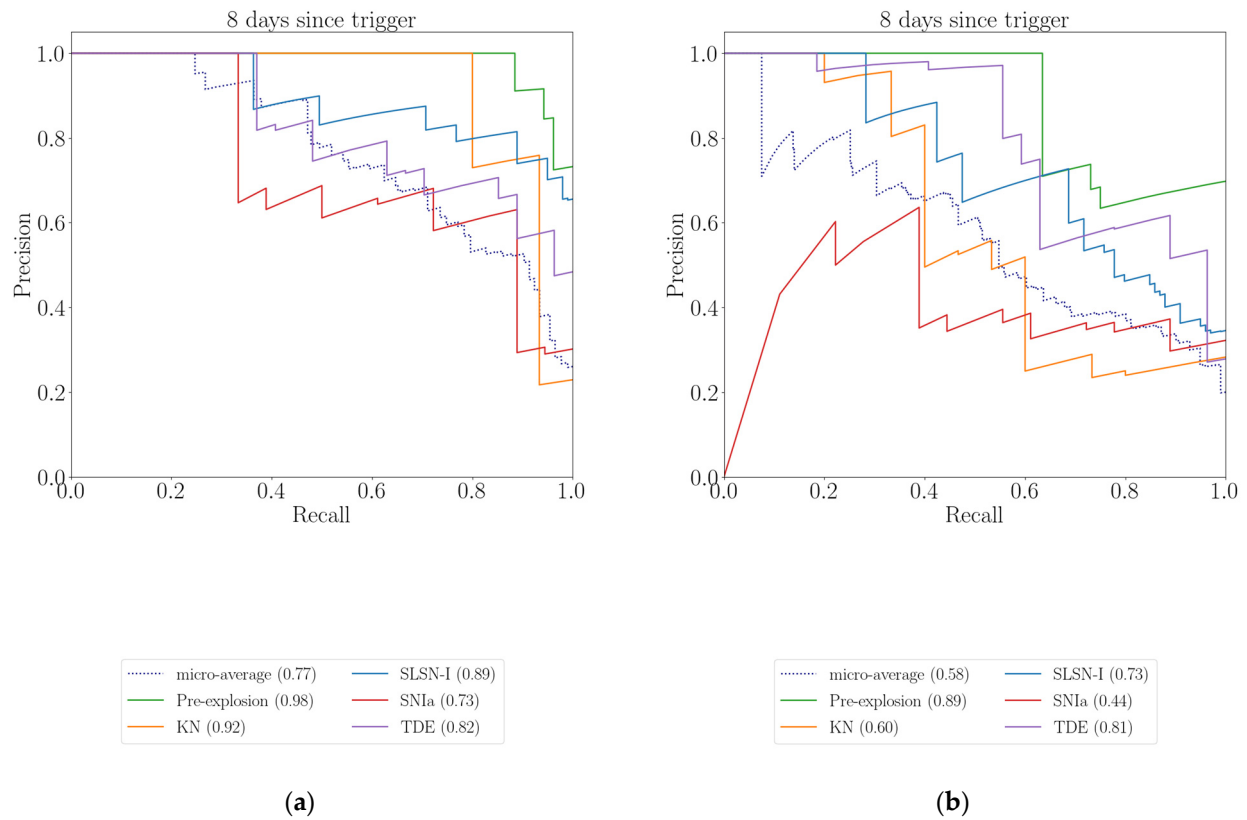
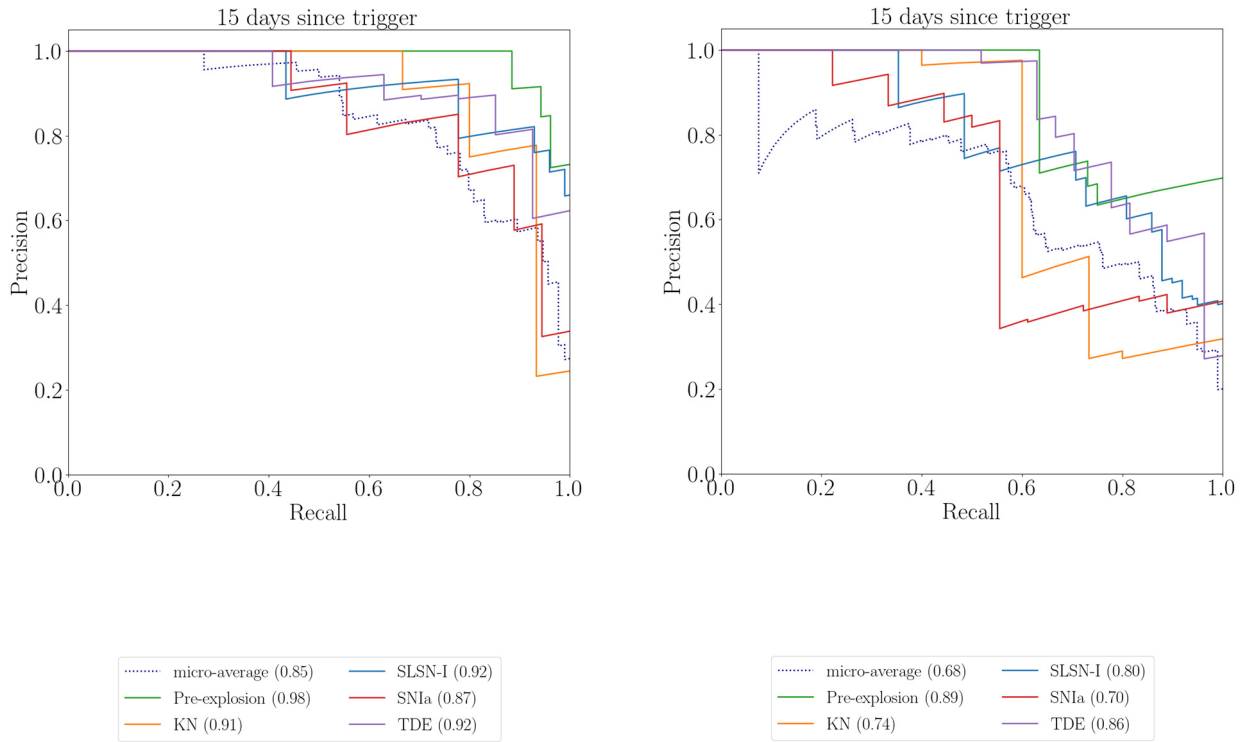
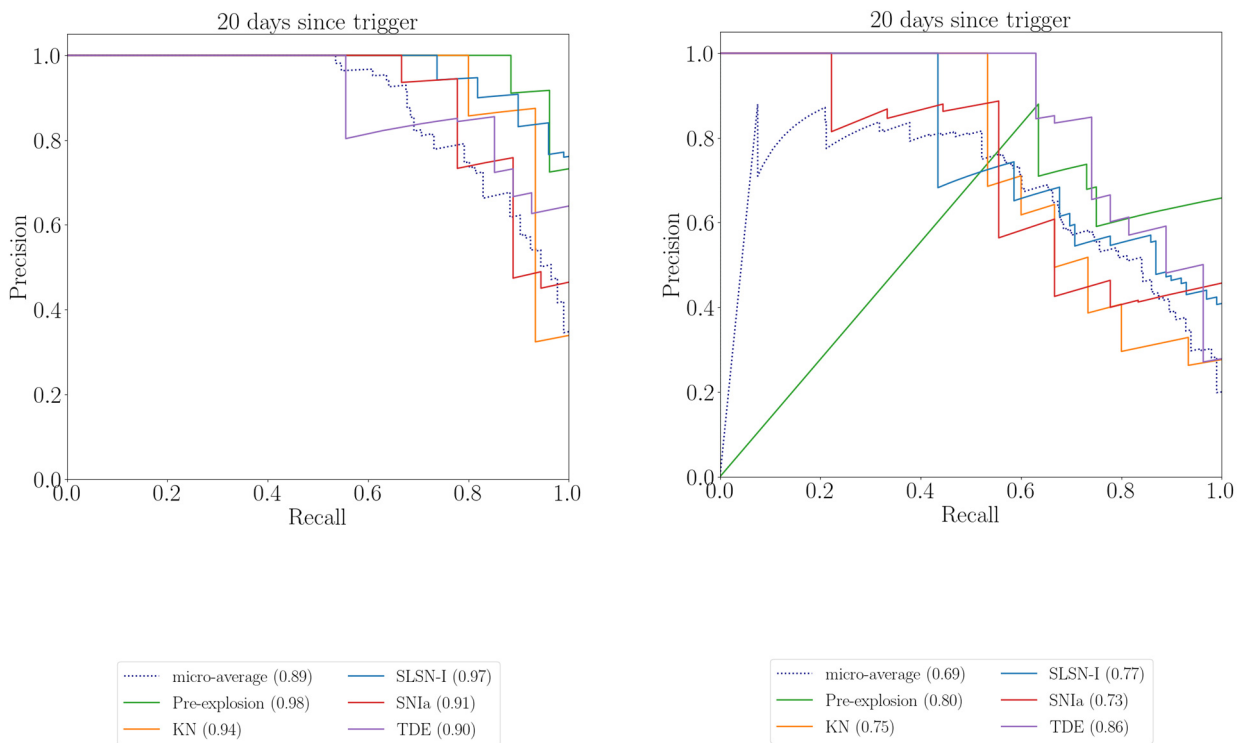


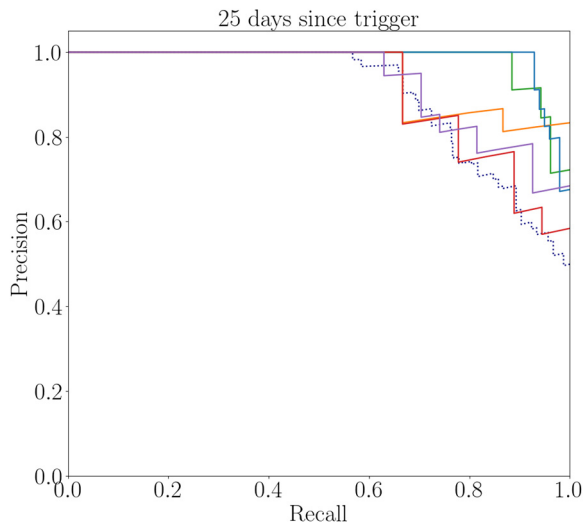
Figure 16. PR curve at 8 days since trigger. (a) TLW; (b) Rapid.



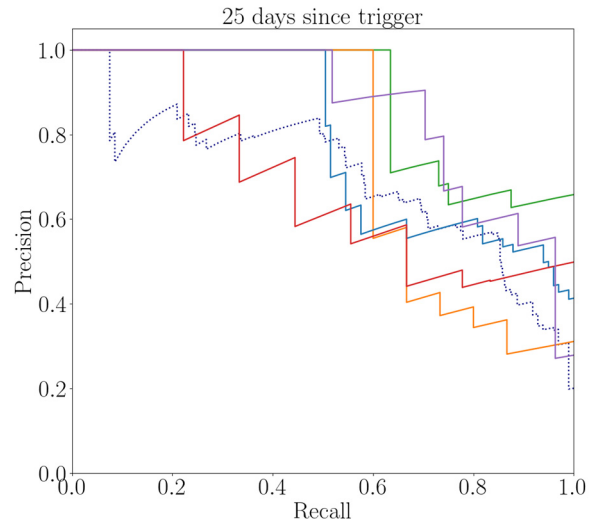
(a) (b)
Figure 17. PR curve at 15 days since trigger. (a) TLW; (b) Rapid.



(a) (b)
Figure 18. PR curve at 20 days since trigger. (a) TLW; (b) Rapid.

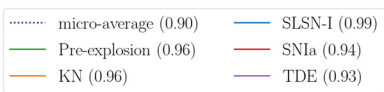
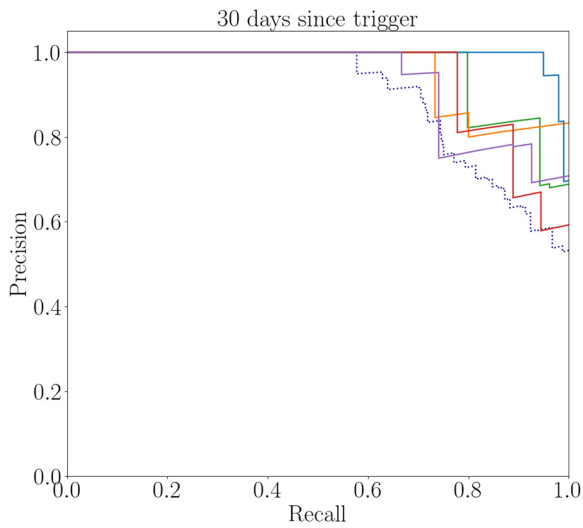


(a)

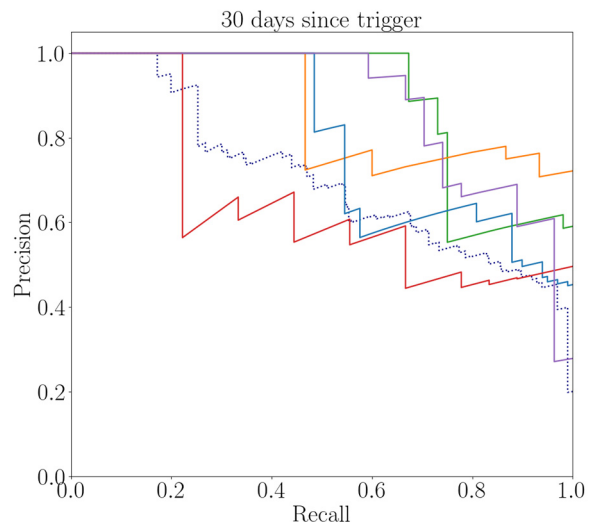


(b)

Figure 19. PR curve at 25 days since trigger. (a) TLW; (b) Rapid.



(a)



(b)

Figure 20. PR curve at 30 days since trigger. (a) TLW; (b) Rapid.

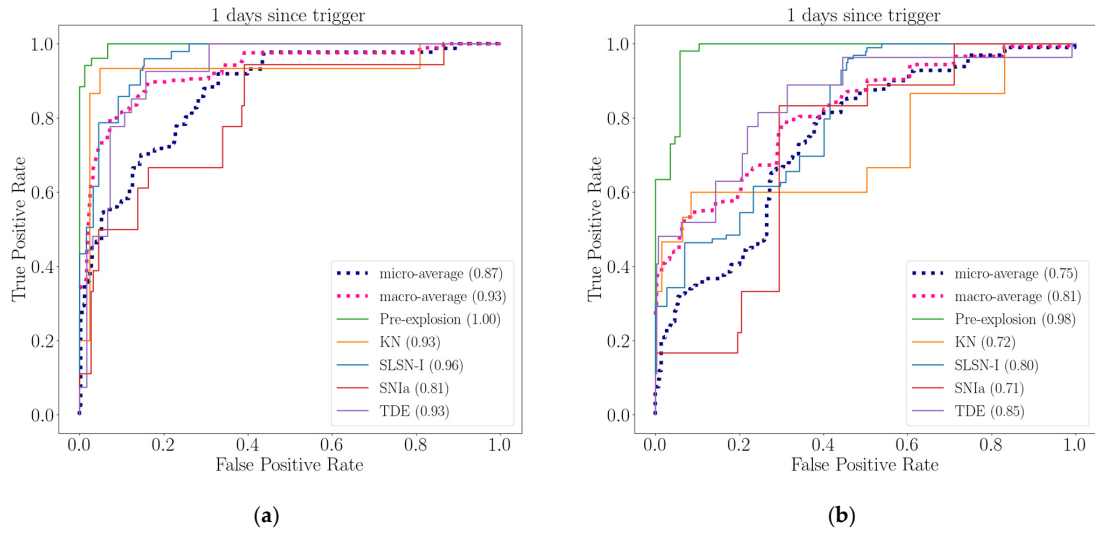


Figure 21. ROC curve at 1 day since trigger. (a) TLW; (b) Rapid.

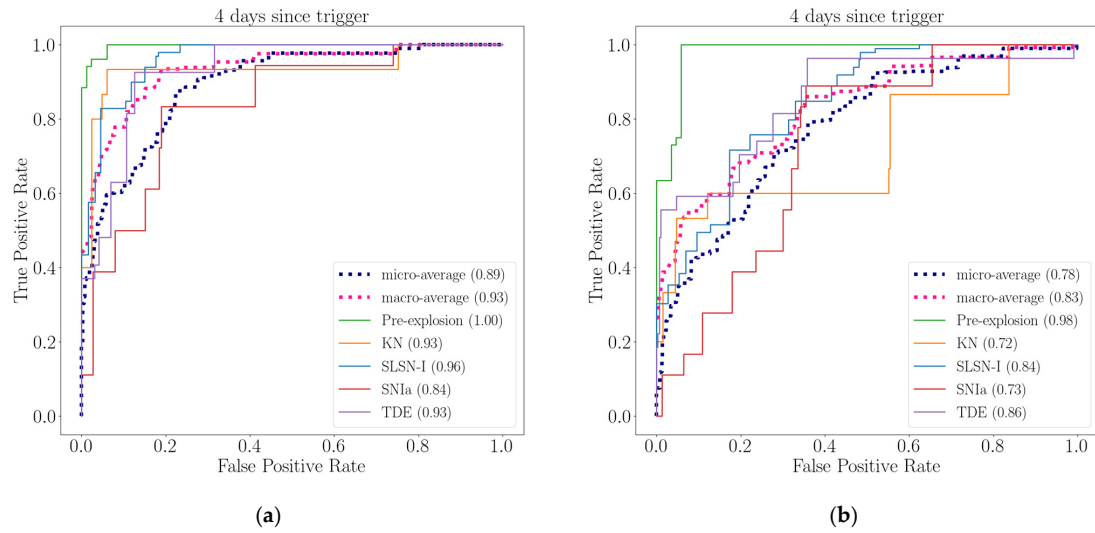


Figure 22. ROC curve at 4 days since trigger. (a) TLW; (b) Rapid.

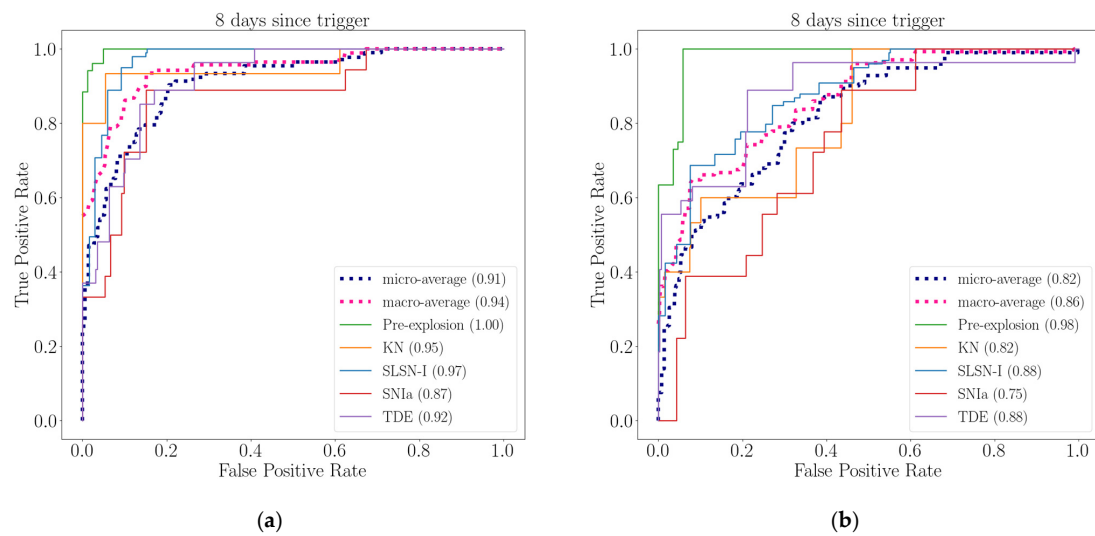


Figure 23. ROC curve at 8 days since trigger. (a) TLW; (b) Rapid.

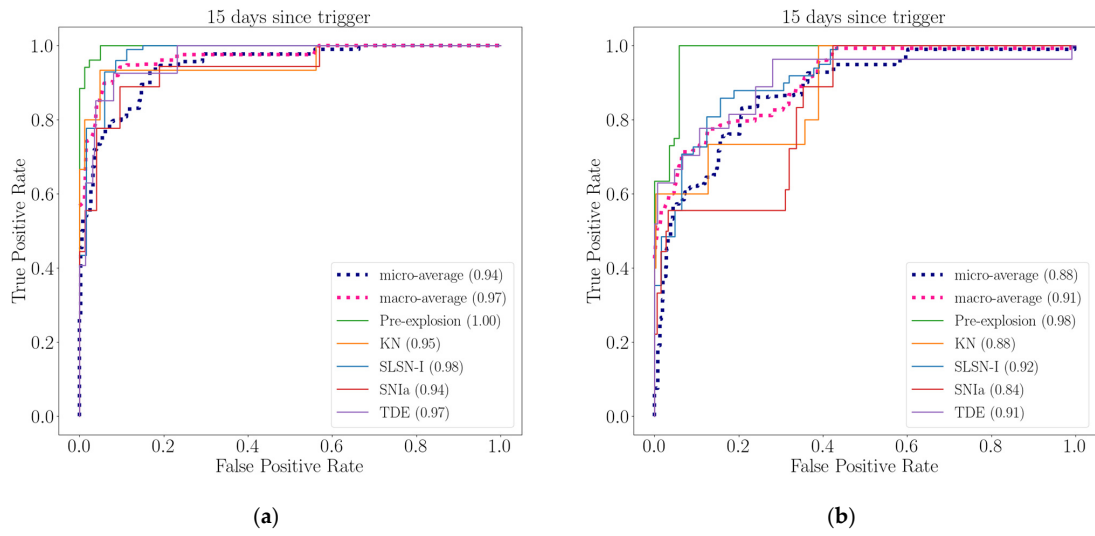


Figure 24. ROC curve at 15 days since trigger. (a) TLW; (b) Rapid.

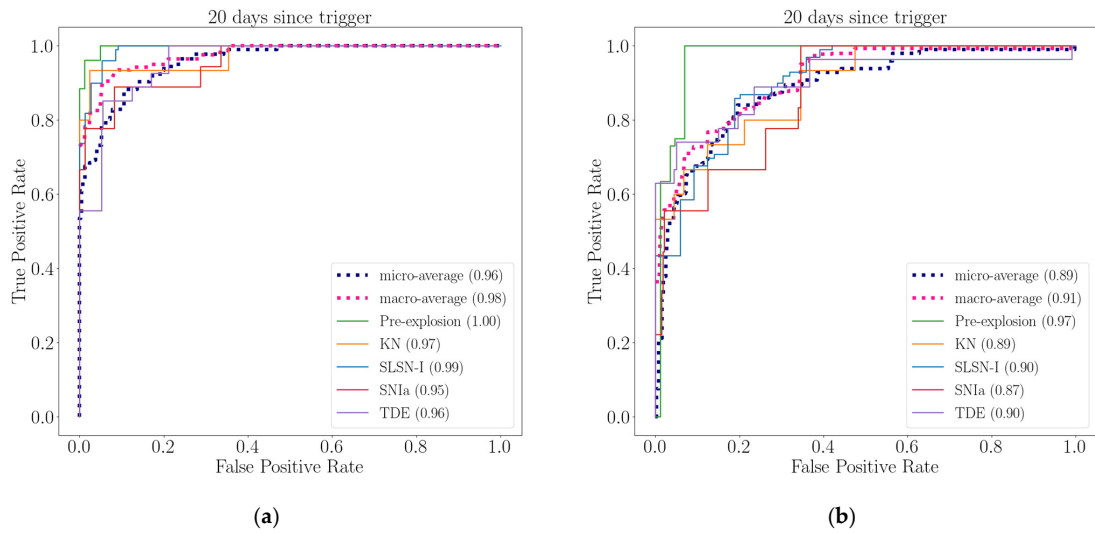


Figure 25. ROC curve at 20 days since trigger. (a) TLW; (b) Rapid.

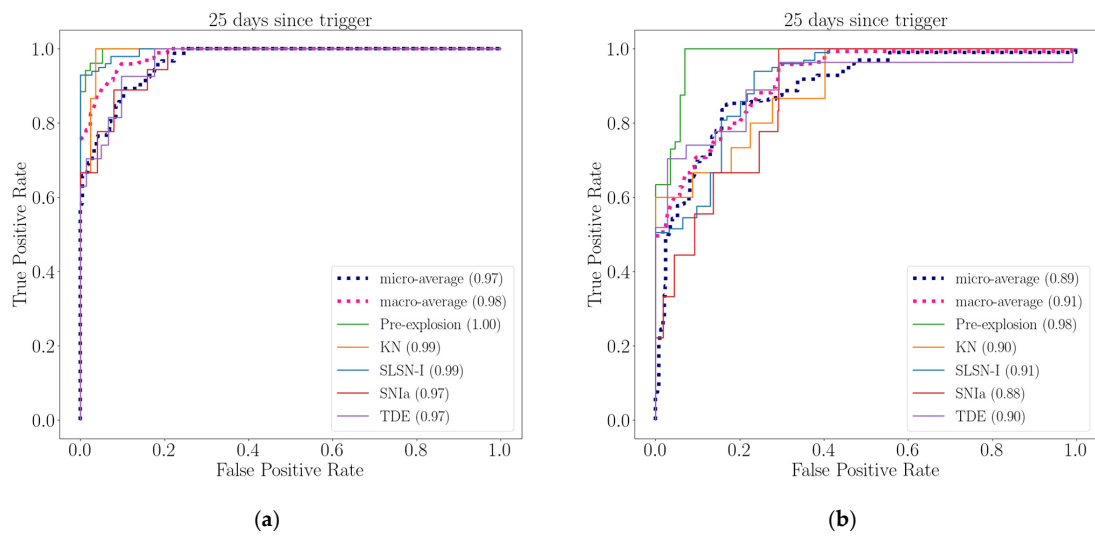


Figure 26. ROC curve at 25 days since trigger. (a) TLW; (b) Rapid.

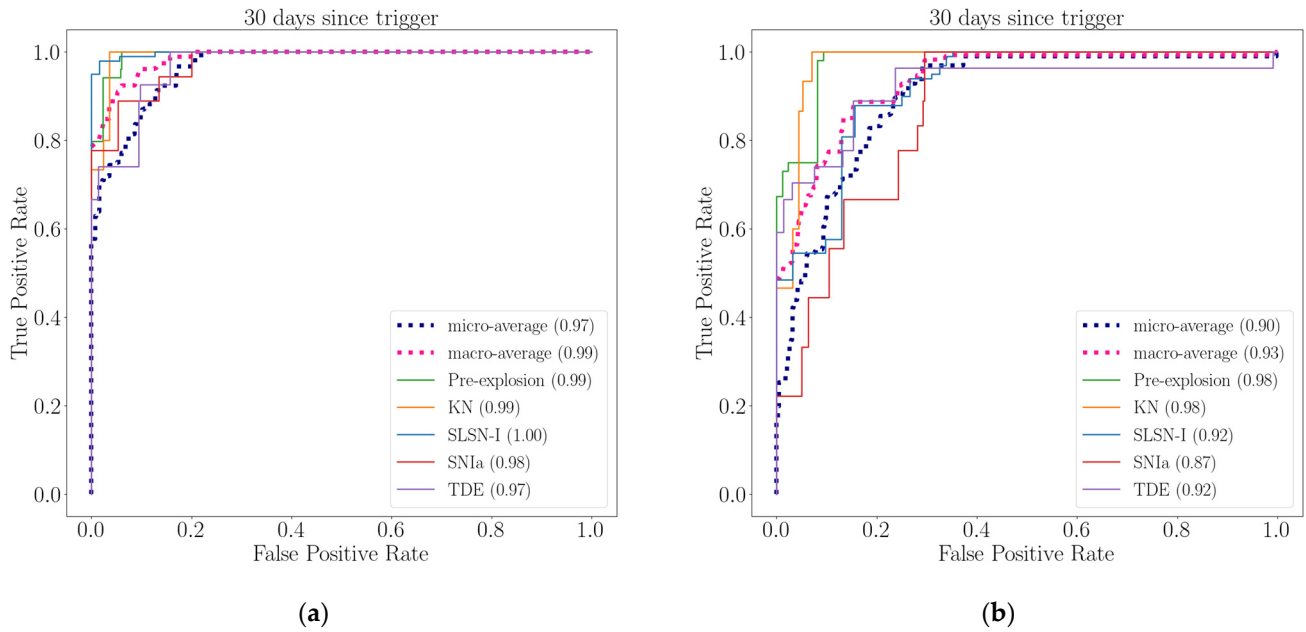


Figure 27. ROC curve at 30 days since trigger. (a) TLW; (b) Rapid.

We also compare the time–AUC curves of TLW and Rapid. As shown in Figure 28, with the increase in time, each AUC of TLW and Rapid has an upward trend. After $t > 40$ days, the AUC value of the Rapid algorithm decreases due to the influence of noise, and the TXW algorithm weakens the influence of noise. The maximum AUC of the Rapid algorithm is greater than 0.85, while that of the TLW algorithm is approximately 1.

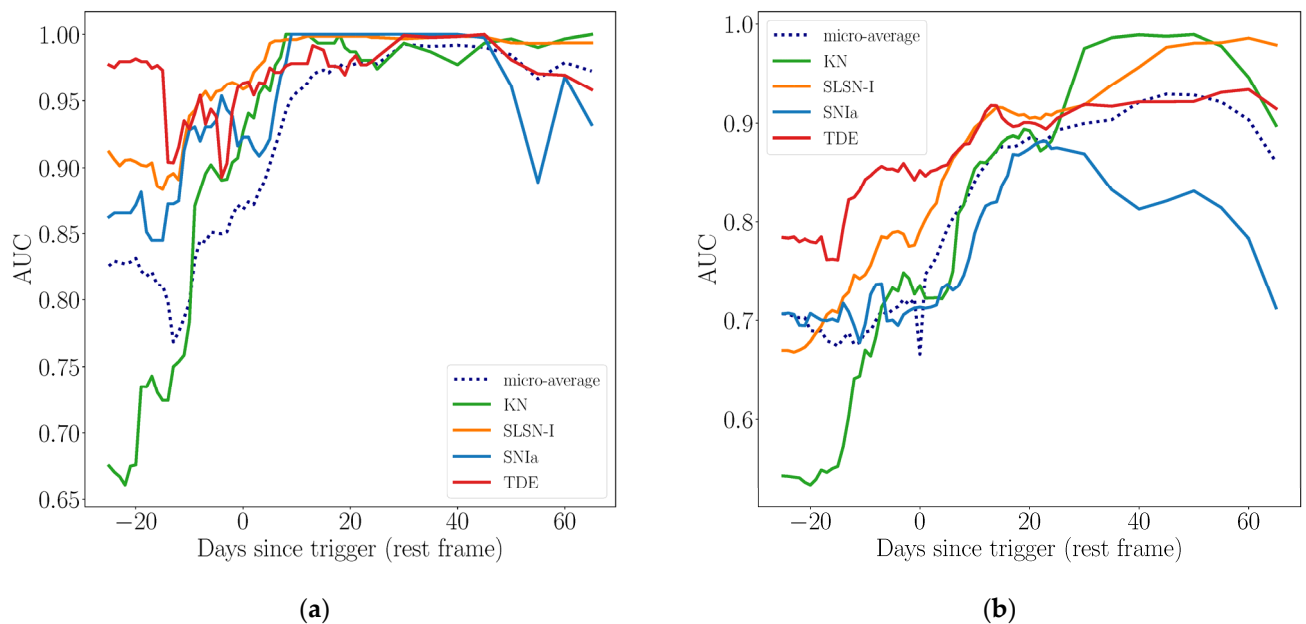


Figure 28. Time–AUC curve. (a) TLW; (b) Rapid.

To sum up, the performance of TLW is better than that of Rapid on all days since the trigger, with higher accuracy and better anti-noise ability, which can realize real-time classification of transients.

5. Application to ZTF Observational Data

In order to verify the availability of the model in actual observation, we apply the TLW algorithm to the measured data of ZTF. ZTF is a time-domain survey telescope at Paloma Observatory in Southern California, USA. ZTF is equipped with a 47 deg² sky survey camera, which can survey the sky faster to find and measure transients with high cadence. In order to prove that the classifier can classify rare transients, we choose the SNIi and SNIbc supernovae provided by Zhang [44]. After data selecting, the data set includes 88 SNIi and 126 SNIbc, and the examples of observed light curves are shown in Figure 29. The classification results are shown in Figure 30.

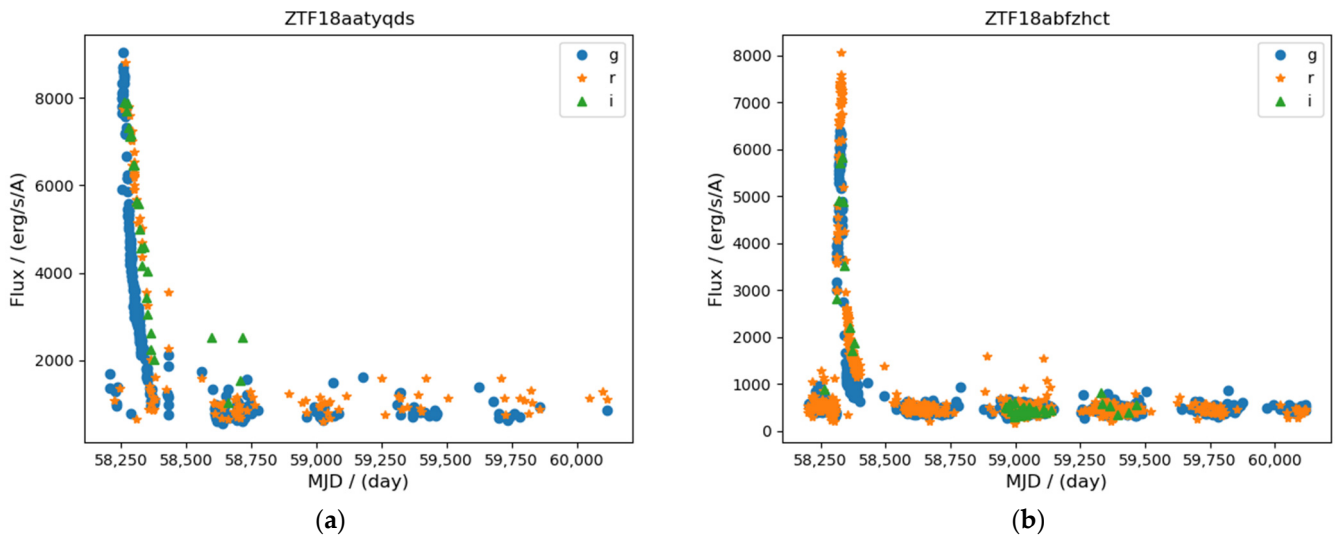


Figure 29. Examples of ZTF observed light curves. (a) SNIi; (b) SNIbc.

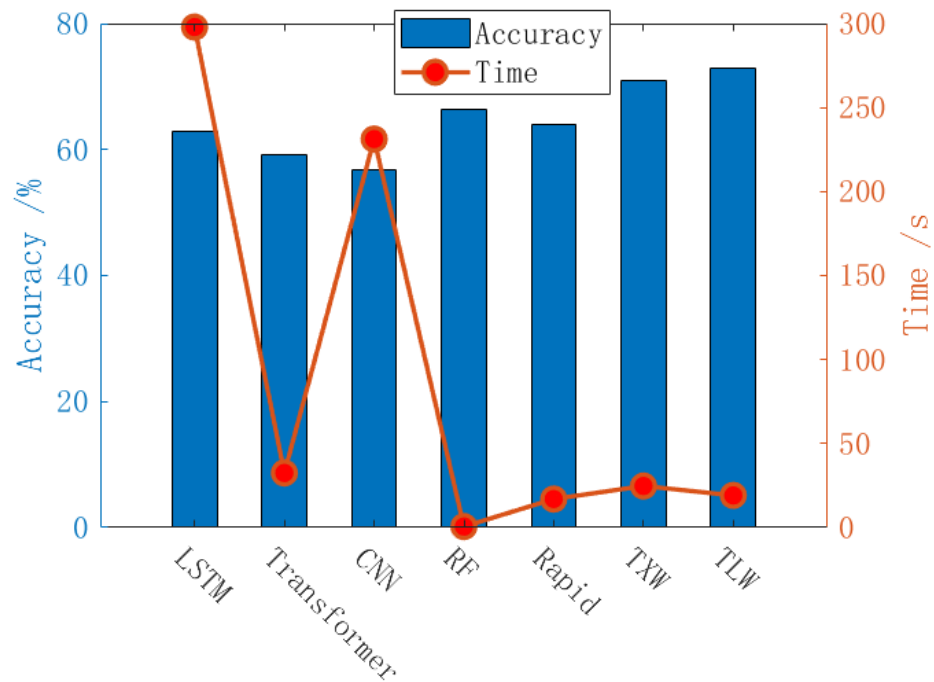


Figure 30. ZTF data classification results. On the horizontal axis are seven classification algorithms. The blue column is the classification accuracy of various algorithms, and the left vertical axis is its scale. The red dots are the implementation timings of various algorithms, and the right vertical axis is its scale.

The results show that the accuracy of TLW is the highest in the ZTF data classification task, which is 9.91% higher than LSTM, 13.6% higher than Transformer, 16.16% higher than CNN, 6.43% higher than RF, 8.85% higher than Rapid, and 1.87% higher than TXW. The implementation timing is reduced by 93.58% compared with LSTM, 41.17% compared with Transformer, 91.73% compared with CNN, and 22.72% compared with TXW. Although the implementation timing is longer than that of RF and Rapid, its accuracy is higher. To sum up, the comprehensive performance of TLW is better than other algorithms. But it can also prove that the TLW algorithm is available in the measured data.

6. Conclusions

We propose a real-time light curve classification algorithm TLW for transients. The TLW algorithm is based on machine learning. Firstly, the TCN module is used to extract the features of transient photometric data. Secondly, the LightGBM module is used to calculate the probability scores of each candidate class of transients. Finally, the Weight module is used to reduce the noise influence and obtain the real-time classification results of transients. Compared with the classical algorithms, the TLW algorithm has stronger feature extraction ability than GRU, which can effectively extract the time series features of light curves belonging to small sample transients. The TLW algorithm also overcomes the shortcomings that RF and XGBOOST ignore some correlations between attributes in data sets. It solves the problem that CNN is easy to over-fit. The TLW algorithm has good feature extraction ability. It can quickly extract related features when the sample ensemble size is small and the time series data are sparse. Moreover, it has the advantages of high classification accuracy, strong robustness, and effectively reducing the noise influence. Moreover, the TLW algorithm is fast, efficient, and requires little memory.

We use TLW to classify the open-source photometric simulation transient data in *g*, *r*, and *i* bands provided via PLAsTiCC, typing TDE, KN, SNIa, and SNSL-I. In order to verify whether TLW is suitable for data sets with any sample ensemble sizes, comparative experiments are designed to use four groups of data with different sample ensemble sizes to build a classifier. There are 362 samples in Group I, 1195 samples in Group II, 2390 samples in Group III, and 3218 samples in Group IIII. After preprocessing progresses with four sets, such as correcting time, de-reddening, fitting light curve, and data augmentation, we established four different data sets suitable for the model, and each data set is divided into 80% training set and 20% testing set respectively. We use training sets to train seven classifiers, including TLW, LSTM, Transformer, CNN, RF, Rapid, and another classification algorithm proposed via our research team, namely TXW. Then we use these classifiers to classify four groups of testing sets respectively. We compare the accuracy and implementation timings to prove the superiority of TLW algorithm. The results of four groups show that the accuracy of TLW is 11.55%, 21.05%, 25.15%, and 21.39% higher than that of LSTM, with an average increase of 19.79%, and the implementation timing of TLW is 5.17%, 86.95%, 86.55%, and 87.42% less than that of LSTM, with an average decrease of 86.52%. Compared with Transformer, the accuracy of the TLW algorithm is improved by 10.77%, 16.05%, 20.18%, and 20.85% respectively, with an average increase of 16.97%, and the implementation timing is reduced by 14.71%, 30.79%, 19.81%, and 11.99%, with an average decrease of 19.33%. Compared with CNN, the accuracy of the TLW algorithm is improved by 7.48%, 17.24%, 20.35%, and 20.36%, with an average increase of 16.36%, and the implementation timing is reduced by 91.63%, 94.21%, 93.8%, and 94.18%, with an average decrease of 93.46%. Although the efficiency of TLW is lower than that of RF and Rapid, its accuracy is higher. The accuracy of TLW is 4.25%, 11.07%, 13.02%, and 14.15% higher than that of RF, with an average increase of 10.63%. The accuracy of TLW is 4.38%, 5.74%, 7.46%, and 1.4% higher than that of Rapid, with an average increase of 4.75%. Although the accuracy of TLW is less than that of TXW, the implementation timing is 2.6%, 19.89%, 15.22%, and 19.6% less than that of TXW. The average accuracy of TLW is 84.54%, which is the highest among the seven algorithms. In order to verify the real-time classification performance of the TLW algorithm, we select the experimental results of

Group III for in-depth analysis and assess the performance of the algorithm by using Confusion Matrix, PR curve and ROC curve of transient at 1, 4, 8, 15, 20, 25, and 30 days since the trigger. Confusion Matrix shows the accuracy of the classification results. We found that the accuracy of TLW is higher than that of Rapid. Compared with Rapid, PR curve shows that the micro-average AP of TLW is improved by 0.21, 0.21, 0.19, 0.17, 0.2, 0.2, and 0.19 at 1, 4, 8, 15, 20, 25, and 30 days since the trigger. Compared with Rapid, ROC curve shows that the micro-average AUC of TLW is improved by 0.12, 0.11, 0.09, 0.06, 0.07, 0.08, and 0.07 respectively, and the macro-average AUC is improved by 0.12, 0.1, 0.08, 0.06, 0.07, and 0.06 respectively at 1, 4, 8, 15, 20, 25, and 30 days since the trigger. To sum up, the performance of the TLW algorithm is better than that of the Rapid algorithm at all days since the trigger, with higher accuracy and better anti-noise ability. The TLW algorithm can realize real-time classification of transients.

We also apply the TLW algorithm to classify the ZTF real transient. The results show that the TLW algorithm has the highest accuracy, which is 9.91% higher than LSTM, 13.6% higher than Transformer, 16.16% higher than CNN, 6.43% higher than RF, 8.85% higher than Rapid, and 1.87% higher than TXW. Furthermore, the implementation timing of TLW is 93.58% less than LSTM, 41.17% less than Transformer, 91.73% less than CNN, and 22.72% less than TXW. Although the implementation timing of TLW is longer than that of RF and Rapid, the accuracy of TLW is high. To sum up, the TLW algorithm has the best comprehensive performance indexes and has the advantages of high precision and high efficiency.

TLW is a real-time classification algorithm, which can provide transient classes at an early stage. In the future, the TLW algorithm will be applied to the SVOM mission to identify ToOs. Due to its wide field camera mounted on the primary focus, C-GFT excels in searching candidate of ToOs, particularly when dealing with significant localization errors. The real-time light curve classification method is well-suited for identifying true transients within the light curves of objects in the localized region. This study is also helpful to make follow-up observation of transients and promote the study of their physical properties and precursor systems.

The TLW algorithm can classify transients preliminarily and rapidly using the shapes of light curves, aiming to obtain the classes of transients such as TDE and SN. However, many types of transients have special subclasses. For example, the TDE could be expected to be a repeating periodic event. The TLW algorithm can confirm that the type of the object is TDE within an outbreak period of the repeating periodic TDE. In future work, we will consider combining the astrophysical properties of the target to propose an improved TLW algorithm, aiming at real-time identification of the subclass of the transient object, such as repeating periodic TDE, SN Ib, SNIc, etc.

Author Contributions: Conceptualization, M.L., C.W. and Z.K.; methodology, M.L.; software, M.L.; validation, M.L., C.W., Z.K., C.L. and S.D.; formal analysis, M.L. and C.W.; investigation, M.L., C.W., and Z.K.; resources, M.L., C.W., Z.K., C.L., S.D. and Z.L.; data curation, M.L. and S.D.; writing—original draft preparation, M.L.; writing—review and editing, M.L., C.W., Z.K., C.L., S.D. and Z.L.; visualization, M.L.; supervision, M.L. and Z.K.; project administration, M.L., C.W., Z.K. and C.L.; funding acquisition, C.W., Z.K., S.D. and Z.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, grant numbers U2031129, U1931133, 12273080, and 12203078. Chinese Academy of Sciences and local government cooperation project, grant numbers 2023SYHZ0027 and 23SH04. Strategic Priority Research Program of the Chinese Academy of Sciences, grant number XDB0550401. Natural Science Foundation of Jilin Province, grant number 20210101468JC. This work is supported by the SVOM project, a mission in the Strategic Priority Program on Space Science of CAS.

Data Availability Statement: We are grateful for PLAsTiCC, which provided us with the simulated data of transients. We are also grateful for ZTF that provide us with real transients. PLAsTiCC website can be found at <https://www.kaggle.com/c/PLAsTiCC-2018/data> (accessed on 2 June 2022) and the ZTF website at https://irsa.ipac.caltech.edu/docs/program_interface/ztf_lightcurve_api.html (accessed on 18 October 2023).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Wu, C.; Ma, D.; Tian, H.J.; Li, X.R.; Wei, J.Y. Study and Development of a Fast and Automatic Astronomical-transient-identification System. *Acta Autom. Sin.* **2017**, *43*, 2170–2177.
2. Muthukrishna, D.; Narayan, G.; Mandel, K.S.; Biswas, R.; Hložek, R. RAPID: Early Classification of Explosive Transients Using Deep Learning. *Publ. Astron. Soc. Pac.* **2019**, *131*, 118002. [[CrossRef](#)]
3. Abbott, B.P.; Abbott, R.; Abbott, T.D.; Abernathy, M.R.; Acernese, F.; Ackley, K.; Adams, C.; Adams, T.; Addesso, P.; Adhikari, R.X.; et al. Observation of Gravitational Waves from a Binary Black Hole Merger. *Phys. Rev. Lett.* **2016**, *116*, 061102. [[CrossRef](#)] [[PubMed](#)]
4. Riess, A.G.; Filippenko, A.V.; Challis, P.; Clocchiatti, A.; Diercks, A.; Garnavich, P.M.; Gilliland, R.L.; Hogan, C.J.; Jha, S.; Kirshner, R.P.; et al. Observational Evidence from Supernovae for an Accelerating Universe and a Cosmological Constant. *Astron. J.* **2018**, *859*, 1009–1038. [[CrossRef](#)]
5. Dai, L.X.; McKinney, J.C.; Roth, N.; Ramirez-Ruiz, E.; Miller, M.C. A Unified Model for Tidal Disruption Events. *Astrophys. J. Lett.* **1998**, *116*, L20. [[CrossRef](#)]
6. Cowperthwaite, P.S.; Berger, E.; Villar, V.A.; Metzger, B.D.; Nicholl, M.; Chornock, R.; Blanchard, P.K.; Fong, W.; Margutti, R.; Soares-Santos, M.; et al. The Electromagnetic Counterpart of the Binary Neutron Star Merger LIGO/Virgo GW170817. II. UV, Optical, and Near-infrared Light Curves and Comparison to Kilonova Models. *Astrophys. J. Lett.* **2017**, *848*, L17. [[CrossRef](#)]
7. Kasen, D.; Metzger, B.; Barnes, J.; Quataert, E.; Ramirez-Ruiz, E. Origin of the heavy elements in binary neutron-star mergers from a gravitational-wave event. *Nature* **2017**, *551*, 80–84. [[CrossRef](#)]
8. Abbott, B.P.; Abbott, R.; Abbott, T.D.; Acernese, F.; Ackley, K.; Adams, C.; Adams, T.; Addesso, P.; Adhikari, R.X.; Adya, V.B.; et al. GW170817: Observation of Gravitational Waves from a Binary Neutron Star Inspiral. *Phys. Rev. Lett.* **2017**, *119*, 161101. [[CrossRef](#)] [[PubMed](#)]
9. Bellm, E. The Zwicky Transient Facility. In Proceedings of the Third Hot-wiring the Transient Universe Workshop (HTU-III), Santa Fe, NM, USA, 13–15 November 2013.
10. Ivezić, Ž.; Kahn, S.M.; Tyson, J.A.; Abel, B.; Acosta, E.; Allsman, R.; Alonso, D.; AlSayyad, Y.; Anderson, S.F.; Andrew, J.; et al. LSST: From Science Drivers to Reference Design and Anticipated Data Products. *Astrophys. J.* **2019**, *873*, 111. [[CrossRef](#)]
11. Dark Energy Survey Collaboration; Abbott, T.; Abdalla, F.B.; Aleksić, J.; Allam, S.; Amara, A.; Bacon, D.; Balbinot, E.; Banerji, M.; BBechtol, K.; et al. The Dark Energy Survey: More than dark energy—An overview. *Mon. Not. R. Astron. Soc.* **2016**, *460*, 1270–1299. [[CrossRef](#)]
12. Chambers, K.C.; Magnier, E.A.; Metcalfe, N.; Flewelling, H.A.; Huber, M.E.; Waters, C.Z.; Denneau, L.; Draper, P.W.; Farrow, D.; Finkbeiner, D.P.; et al. The Pan-STARRS1 Surveys. *arXiv* **2016**, arXiv:1612.05560.
13. Chen, W.; Li, B.; Huang, Y.; Duan, J.; Zhang, P.; Guo, D.Y.; Sun, G.X.; Wang, P.; Song, L.; Li, X.B.; et al. Introduction of the scientific application system of GECAM. *Sci. Sin. Physic. Mech. Astron.* **2020**, *50*, 129512. [[CrossRef](#)]
14. Gong, Y.; Liu, X.K.; Cao, Y.; Chen, X.L.; Fan, Z.H.; Li, R.; Li, X.D.; Li, Z.G.; Zhang, X.; Zhan, H. Cosmology from the Chinese Space Station Optical Survey (CSS-OS). *Astrophys. J.* **2019**, *883*, 203. [[CrossRef](#)]
15. Wei, J.; Cordier, B.; Antier, S.; Antilogus, P.; Atteia, J.L.; Bajat, A.; Basa, S.; Beckmann, V.; Bernardini, M.G.; Boissier, S.; et al. The Deep and Transient Universe in the SVOM Era: New Challenges and Opportunities—Scientific prospects of the SVOM mission. *arXiv* **2016**, arXiv:1610.06892.
16. Moriwaki, K.; Nishimichi, T.; Yoshida, N. Machine learning for observational cosmology. *Rep. Pro. Physic.* **2023**, *86*, 076901. [[CrossRef](#)] [[PubMed](#)]
17. Perrefort, D.; Zhang, Y.; Galbany, L.; Wood-Vasey, W.M.; González-Gaitán, S. A Template-based Approach to the Photometric Classification of SN 1991bg-like Supernovae in the SDSS-II Supernova Survey. *Astrophys. J.* **2020**, *904*, 156. [[CrossRef](#)]
18. Boone, K. Avocado: Photometric Classification of Astronomical Transients with Gaussian Process Augmentation. *Astron. J.* **2019**, *158*, 257. [[CrossRef](#)]
19. Sánchez-Sáez, P.; Reyes, I.; Valenzuela, C.; Förster, F.; Eyheramendy, S.; Elorrieta, F.; Bauer, F.E.; Cabrera-Vives, G.; Estévez, P.A.; Catelan, M.; et al. Alert Classification for the ALerCE Broker System: The Light Curve Classifier. *Astron. J.* **2021**, *161*, 141. [[CrossRef](#)]
20. Möller, A.; de Boissière, T. SuperNNova: An open-source framework for Bayesian, neural network-based supernova classification. *Mon. Not. R. Astron. Soc.* **2020**, *491*, 4277–4293. [[CrossRef](#)]
21. Godines, D.; Bachelet, E.; Narayan, G.; Street, R.A. A machine learning classifier for microlensing in wide-field surveys. *Astron. Comput.* **2019**, *28*, 100298. [[CrossRef](#)]

22. Ishida, E.E.O.; Beck, R.; González-Gaitán, S.; de Souza, R.S.; Krone-Martins, A.; Barrett, J.W.; Kenamer, N.; Vilalta, R.; Burgess, J.M.; Quint, B.; et al. Optimizing spectroscopic follow-up strategies for supernova photometric classification with active learning. *Mon. Not. R. Astron. Soc.* **2019**, *483*, 2–18. [[CrossRef](#)]
23. Villar, V.A.; Berger, E.; Miller, G.; Chornock, R.; Rest, A.; Jones, D.O.; Drout, M.R.; Foley, R.J.; Kirshner, R.; Lunnan, R.; et al. Supernova Photometric Classification Pipelines Trained on Spectroscopically Classified Supernovae from the Pan-STARRS1 Medium-deep Survey. *Astrophys. J.* **2019**, *884*, 83. [[CrossRef](#)]
24. Hosseinzadeh, G.; Dauphin, F.; Villar, V.A.; Berger, E.; Jones, D.O.; Challis, P.; Chornock, R.; Drout, M.R.; Foley, R.J.; Kirshner, R.P.; et al. Photometric Classification of 2315 Pan-STARRS1 Supernovae with Superphot. *Astrophys. J.* **2020**, *905*, 93. [[CrossRef](#)]
25. Stachie, C.; Coughlin, M.W.; Christensen, N.; Muthukrishna, D. Using machine learning for transient classification in searches for gravitational-wave counterparts. *Mon. Not. R. Astron. Soc.* **2020**, *497*, 1320–1331. [[CrossRef](#)]
26. Takahashi, I.; Suzuki, N.; Yasuda, N.; Kimura, A.; Ueda, N.; Tanaka, M.; Tominaga, N.; Yoshida, N. Photometric classification of Hyper Suprime-Cam transients using machine learning. *Publ. Astron. Soc. Jpn.* **2020**, *72*, 89. [[CrossRef](#)]
27. Villar, V.A.; Hosseinzadeh, G.; Berger, E.; Ntampaka, M.; Jones, D.O.; Challis, P.; Chornock, R.; Drout, M.R.; Foley, R.J.; Kirshner, R.P.; et al. SuperRAENN: A Semisupervised Supernova Photometric Classification Pipeline Trained on Pan-STARRS1 Medium-Deep Survey Supernovae. *Astrophys. J.* **2020**, *905*, 94. [[CrossRef](#)]
28. Baldeschi, A.; Miller, A.; Stroh, M.; Margutti, R.; Coppejans, D.L. Star Formation and Morphological Properties of Galaxies in the Pan-STARRS 3π Survey. I. A Machine-learning Approach to Galaxy and Supernova Classification. *Astrophys. J.* **2020**, *902*, 60. [[CrossRef](#)]
29. Andreoni, I.; Coughlin, M.W.; Kool, E.C.; Kasliwal, M.M.; Kumar, H.; Bhalerao, V.; Carracedo, A.S.; Ho, A.Y.Q.; Pang, P.T.H.; Saraogi, D.; et al. Fast-transient Searches in Real Time with ZTFreST: Identification of Three Optically Discovered Gamma-Ray Burst Afterglows and New Constraints on the Kilonova Rate. *Astrophys. J.* **2021**, *918*, 63. [[CrossRef](#)]
30. Burhanudin, U.F.; Maund, J.R.; Killestein, T.; Ackley, K.; Dyer, M.J.; Lyman, J.; Ulaczyk, K.; Cutter, R.; Mong, Y.L.; Steeghs, D.; et al. Light-curve classification with recurrent neural networks for GOTO: Dealing with imbalanced data. *Mon. Not. R. Astron. Soc.* **2021**, *505*, 4345–4361. [[CrossRef](#)]
31. Allam, T.J.; McEwen, J.D. Paying Attention to Astronomical Transients: Introducing the Time-series Transformer for Photometric Classification. *arXiv* **2021**, arXiv:2105.06178. [[CrossRef](#)]
32. Allam, T.J.; Peloton, J.; McEwen, J.D. The Tiny Time-series Transformer: Low-latency High-throughput Classification of Astronomical Transients using Deep Model Compression. *arXiv* **2023**, arXiv:2303.08951.
33. Qu, H.; Sako, M. Photometric Classification of Early-time Supernova Light Curves with SCONE. *Astron. J.* **2022**, *163*, 57. [[CrossRef](#)]
34. Burhanudin, U.F.; Maund, J.R. Pan-chromatic photometric classification of supernovae from multiple surveys and transfer learning for future surveys. *Mon. Not. R. Astron. Soc.* **2023**, *521*, 1601–1619. [[CrossRef](#)]
35. Gomez, S.; Villar, V.A.; Berger, E.; Gezari, S.; van Velzen, S.; Nicholl, M.; Blanchard, P.K.; Alexander, K.D. Identifying Tidal Disruption Events with an Expansion of the FLEET Machine-learning Algorithm. *Astrophys. J.* **2023**, *949*, 113. [[CrossRef](#)]
36. Gagliano, A.; Contardo, G.; Foreman-Mackey, D.; Malz, A.I.; Aleo, P.D. First Impressions: Early-time Classification of Supernovae Using Host-galaxy Information and Shallow Learning. *Astrophys. J.* **2023**, *954*, 6. [[CrossRef](#)]
37. Pimentel, Ó.; Estévez, P.A.; Förster, F. Deep Attention-based Supernovae Classification of Multiband Light Curves. *Astron. J.* **2023**, *165*, 18. [[CrossRef](#)]
38. Kisley, M.; Qin, Y.J.; Zabludoff, A.; Barnard, K.; Ko, C.L. Classifying Astronomical Transients Using Only Host Galaxy Photometry. *Astrophys. J.* **2023**, *942*, 29. [[CrossRef](#)]
39. Bai, S.J.; Zico Kolter, J.; Koltun, V. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv* **2018**, arXiv:1803.01271.
40. Fan, J.L.; Ma, X.; Wu, L.F.; Zhang, F.C.; Yu, X.; Zeng, W.Z. Light Gradient Boosting Machine: An efficient soft computing model for estimating daily reference evapotranspiration with local and external meteorological data. *Agric. Water Manag.* **2019**, *225*, 105758. [[CrossRef](#)]
41. Ke, G.L.; Meng, Q.; Finley, T.; Wang, T.F.; Chen, W.; Ma, W.D.; Ye, Q.W.; Liu, T.Y. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS), Long Beach, CA, USA, 4–9 December 2017.
42. The PLAsTiCC team; Allam, T.J.; Bahmanyar, A.; Biswas, R.; Dai, M.; Galbany, L.; Hložek, R.; Ishida, E.E.O.; Jha, S.W.; Jones, D.O.; et al. The Photometric LSST Astronomical Time-series Classification Challenge (PLAsTiCC): Data set. *arXiv* **2018**, arXiv:1810.00001.
43. Fitzpatrick, E.L. Correcting for the Effects of Interstellar Extinction. *Publ. Astron. Soc. Pac.* **1999**, *111*, 63–75. [[CrossRef](#)]
44. Zhang, J.Y.; Zhang, Y.X.; Kang, Z.H.; Li, C.H.; Zhao, Y.H. A Catalog of Young Stellar Objects from the LAMOST and ZTF Surveys. *Astrophys. J. Suppl. Ser.* **2023**, *267*, 7. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.