Inter-IIT ISRO

Team Number 17
Submission Code : MP_ISRO_T17
March 18, 2022

1 Introduction

The given problem statement is about identifying and characterizing X-ray bursts from light curves that are obtained from the observations made by Chandrayaan-II's Solar X-ray Monitor. The problem can be categorized broadly into the following parts:

- Developing a pipeline to process the noisy light curve and identifying the flares with low false negatives and false positives.
- Estimating the characteristics of the flare candidates identified above, and using the information obtained to remove false identification cases, if possible.
- Developing a statistical/machine learning model to classify flares based on their characteristics, as estimated above.
- Deploying the complete solution approach on a stand-alone application as well as on a web-based tool.

The following diagram depicts the workflow adopted by our team for the aforementioned tasks:

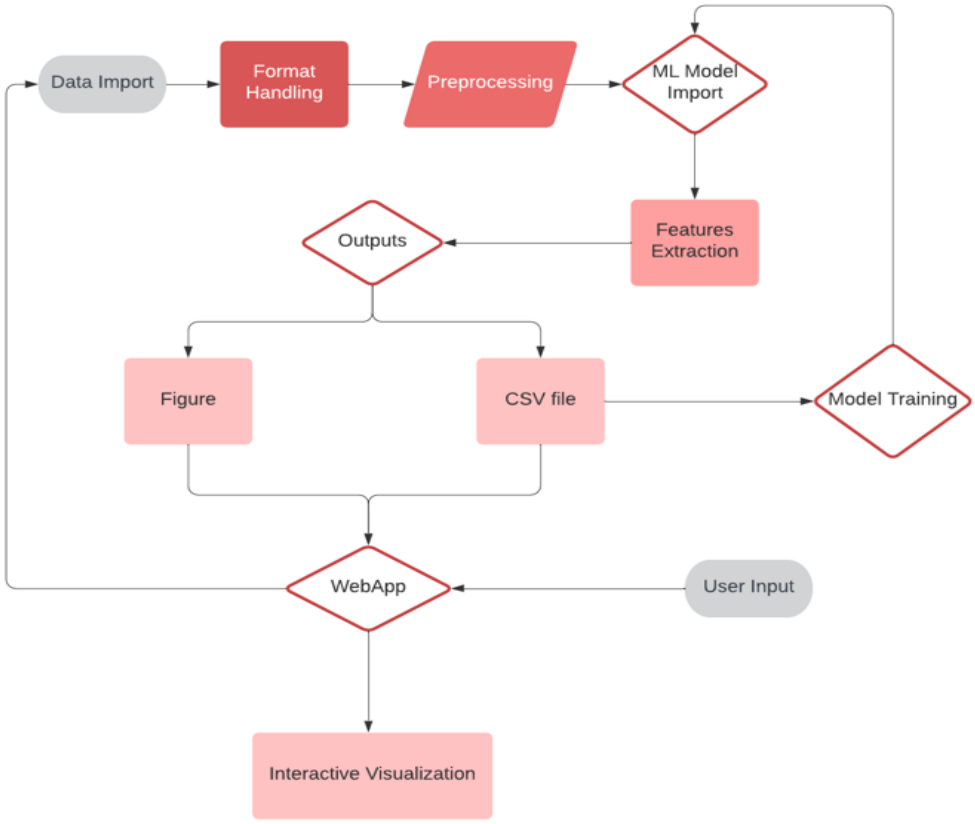

Figure 1: Flow Chart of the entire Approach

2 Dependencies

In order to use the standalone application, python should be installed (preferably the latest version). Furthermore, the following python libraries are required to be installed:

- astropy
- numpy
- pandas
- matplotlib
- scipy
- time
- os
- glob
- warnings
- pyfiglet
- pickle
- itertools

For installing any python module, one needs to run the following command using Terminal:

```
$ pip3 install <module name>
```

3 Internal Tool Workflow

3.1 Approach

Our algorithm uses the following basic stages in the mentioned order while locating flare candidates, characterising and classifying them.

- Pre-processing of noisy data to reduce chances of false flare detection by spurious maxima introduced by noise.
- Characterising the flares by curve fitting to estimate flare parameters
- Unsupervised Classification using a machine learning model
- Saving the results

3.2 Detailed workflow

The detailed workflow of the approach used in our solution is described as following:

1. The algorithm begins by identifying the continuous stretches of data based on successive differences between the recorded timestamps. This obviates the need for supplying a *.gti* file to the tool for identifying the good time intervals. Since the continuous stretches of data have been recorded, they can be processed independently.
2. A Savitzky Golay filter is employed for noise reduction, and for smoothening the noisy light-curve. It ensures that the overall data trend isn't affected, by fitting low order piecewise polynomaials to the data provided. The data must be resampled to ensure uniform sampling durations before using this filter.
3. The obtained light-curve after Savitzky Golay filtering has several local maximas that can create problems while using local maximas for identifying actual flares. Thus, we downsample the data to 80 times the original sampling duration, and employ Random Forest Regression to identify trends in individual continuous data stretches.

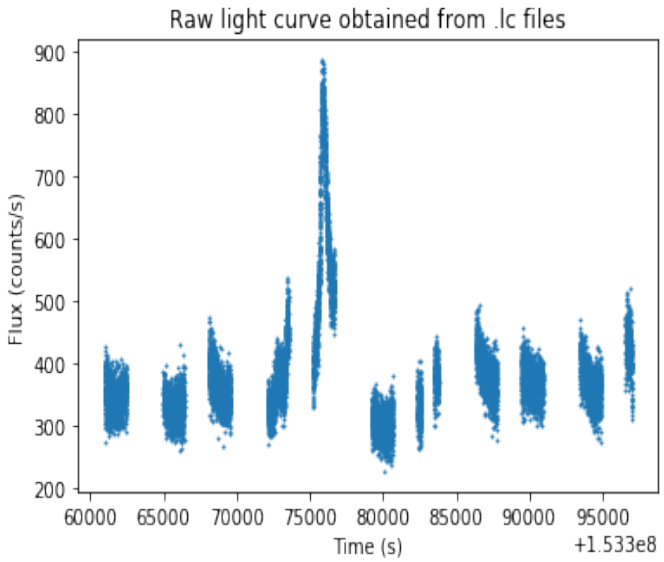

Figure 2: Raw light curve

4. The Random Forest Regression curve needs further smoothening. For this we use Cubic spline interpolation to smoothen out the rough edges obtained from Random Forest Regression.

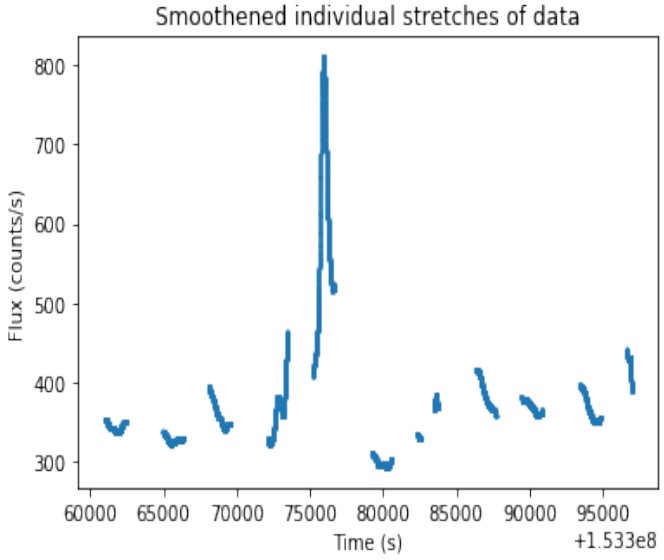

Figure 3: Smoothened light curve

5. As is evident from the above figure, the smoothened light curve can be used for identifying the flares, by looking at the local extrema within a certain neighbourhood of the maxima. The size of the neighbourhood would on one hand help us reject some artefacts of the curve pre-processing, but would create issues in resolving two very closely occurring flares.
6. The next stage is identification of flare candidates. We find the local extrema and adjust them depending on whether any maximas are present in the interval spanned in the neighbourhood of the aforementioned extrema by the adjacent minima. Thus, we find our flare candidates and proceed to fitting.
7. We fit an elementary flare profile (Gryciuk, M., Siarkowski, M., Sylwester, J. et al. Flare Characteristics from X-

ray Light Curves. Sol Phys 292, 77 (2017). https://doi.org/10.1007/s11207-017-1101-8) to the identified extrema and find the characteristics such as decay rate, rise time, decay time, peak flux, and peak flux over pre-flare background. These parameters characterise the shape of the flare and can be used later on in classification. The flare candidates giving non-physical estimates of flare characteristics and those having a Signal to Noise Ratio (SNR) of below 0 dB are discarded by the algorithm.

8. Using a clustering algorithm, we identify the prime clusters present in our collected data. The data is trained over 141 days out of 713 days of provided data. This small amount of data is used in order to meet the requirement of using only 20% of the available dataset. The clusters represent the categorisation of the flares based on flare shape. Further investigation into the clusters can help identify the differences in physical processes causing the flares, between different clusters.

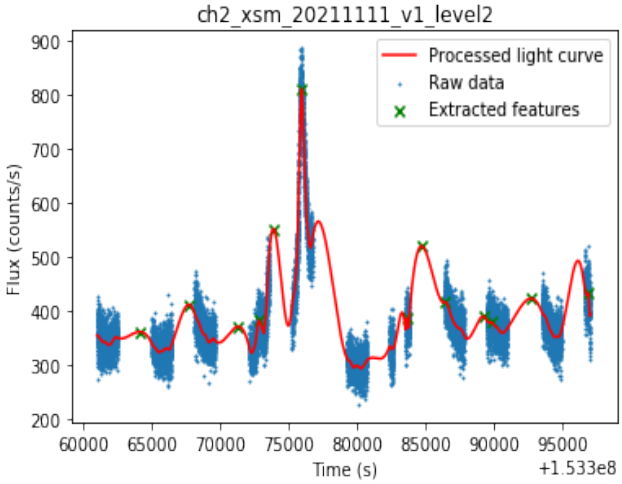

Figure 4: Identified flare candidates

4 Manual

4.1 Stand-alone application

4.1.1 Step 1: Download and Setup

Download the **xsmwas** from the github repository. Unzip it to desired directory. The newly extracted directory has the following contents:

- An **lc_files** directory, containing a readme file. **Please delete this file before proceeding further, or else the script would throw an error**. This is the directory within which the user will store all the light curve files that are to be processed. Currently, the application processes files having only the following extensions: **.lc** and **.fits** (which are assumed to correspond to FITS files), **.txt** and **.dat** (which are assumed to correspond to ASCII tables).
- A csv file **final_data.csv** containing the features of all the files on which the ML model is to be trained. This shall be used just once during initialization.
- A python script **xsmmodgen.py** which generates the pickle file **xsm_model.pkl** that will be used by the script **xsmwas.py** for the purpose of predicting the class of the flares that will be identified. It should be noted that the pickle file is generated locally for use by the model to prevent version inconsistencies.
- A python script **xsmwas.py** which processes all the light curve files stored in the **lc_files** directory. First, the script creates the directories **csv_files** and **fig_files** if they are not already present. Then, the extracted features of all the X-ray bursts of a particular light curve file, say **fileName.extension** are stored in a csv file in the **csv_files** directory with the name **fileName.csv**. The script also has the optional functionality (based

on user input) of:

- Interactively showing light curves visualizing the flares and its peaks in a matplotlib window (Figure 5). The user can zoom in to particular portions to see specific bursts of interest.
- Saving the light curves visualizing the flares and its peaks in the **fig_files** directory as images with name **fileName.jpg**.

4.1.2 Step 2: Generate Pickle File

Execute the **xsmmodgen.py** script to generate the pickle file **xsm_model.pkl** by writing the following command in terminal (ensure that the current working directory is **xsmwas**):

```
$ python3 xsmmodgen.py
```

4.1.3 Step 3: Process all the light curves stored in the lc_files directory

First ensure that all the light curve files are added in the **lc_files** directory having the appropriate extentions. Then, execute the **xsmwas.py** script (Figure 6) to process each and every light curve file stored in the **lc_files** directory and save the features of the flares in the **csv_files** directory by writing the following command in terminal (ensure that the current working directory is **xsmwas**):

```
$ python3 xsmwas.py
```

In order to use the additional functionalities of:

- Saving the images, enter 'y' when the following line is displayed: Save images? (y/n)
- Interactively viewing images, enter 'y' when the following line is displayed: Interactively view images? (y/n)

Close the opened plot once you want to move on to analyse the next file. After all the files have been processed, the final constituents of the **xsmwas** directory after all steps are followed should look like:

```
xsmwas
├── lc_files
│   └── fileName.extension (multiple files)
├── csv_files
│   └── fileName.csv (multiple files)
├── fig_files
│   └── fileName.jpg (multiple files)
├── final_data.csv
├── xsm_model.pkl
├── xsmmodgen.py
└── xsmwas.py
```

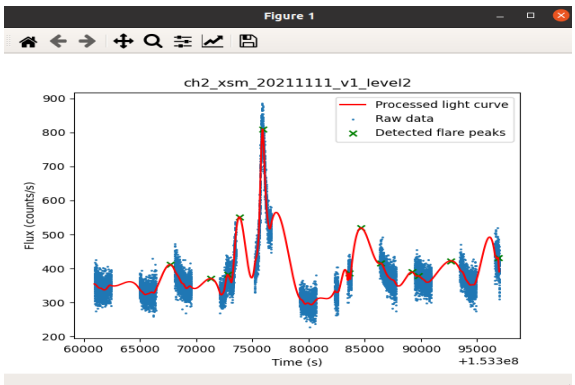

Figure 5: Interactive matplotlib window visualizing flares

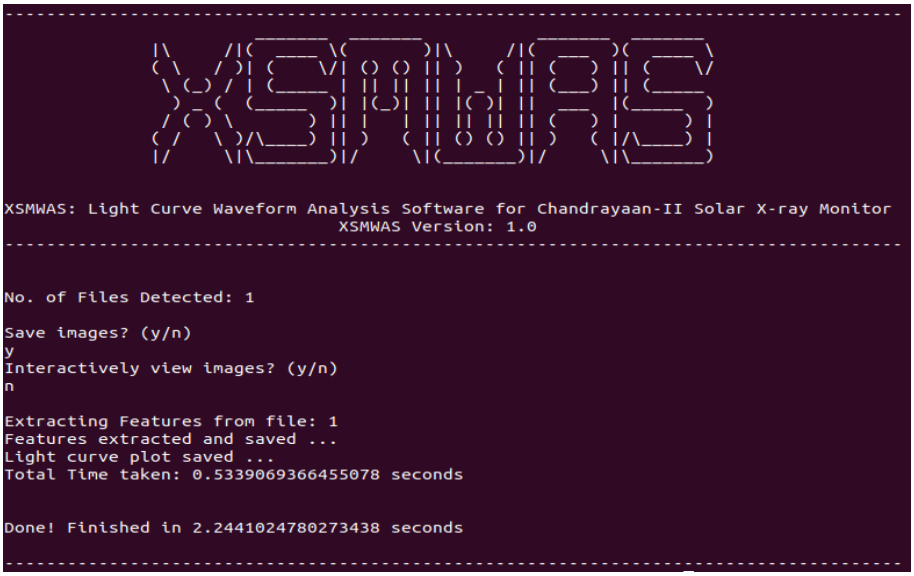

Figure 6: Example output in terminal after executing **xsmwas.py**

4.2 Web Application

4.2.1 Step 1: Download the Repository/Git Clone

User can either download the Repository or clone it using GitClone

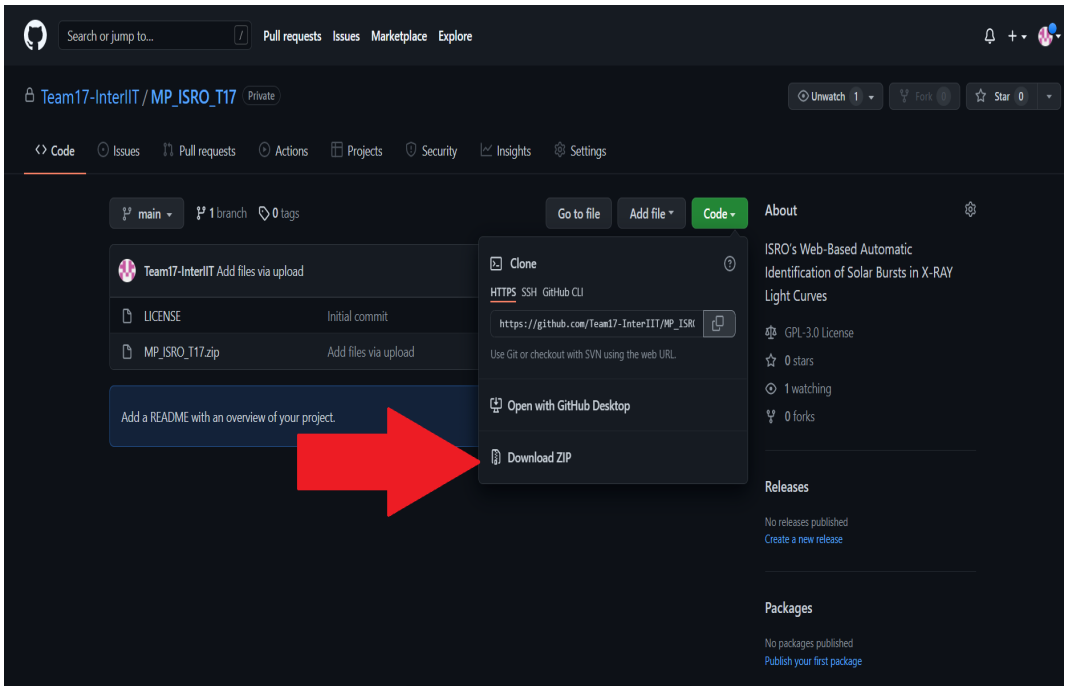

Figure 7: Downloading the Repository

4.2.2 Step 2: Installing all the requirements

Navigate to the root folder and open Terminal in it. Then run the following command there:

```
$ pip install -r requirements.txt
```

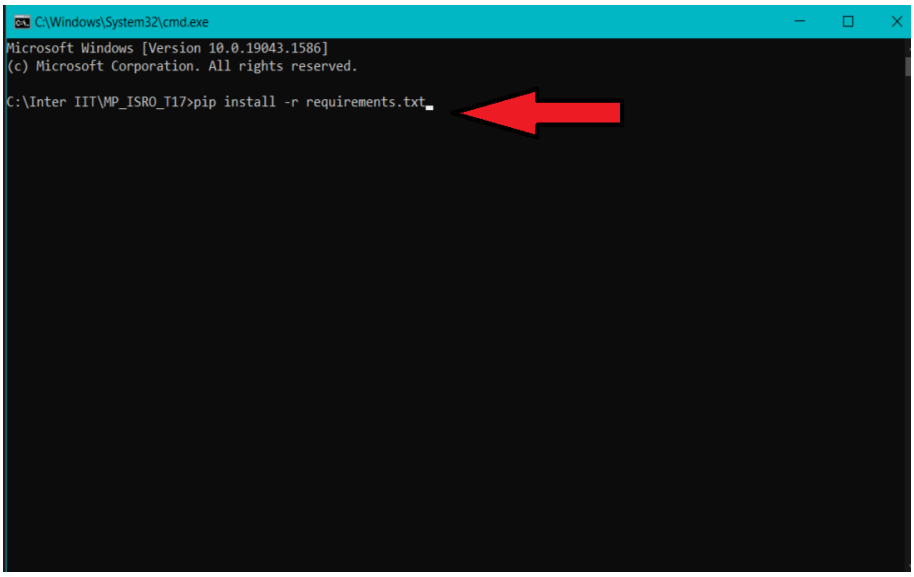

Figure 8: Installing requirements

4.2.3 Step 3: Running the server

Open the root folder in Terminal and run the following command:

```
$ python manage.py runserver
```

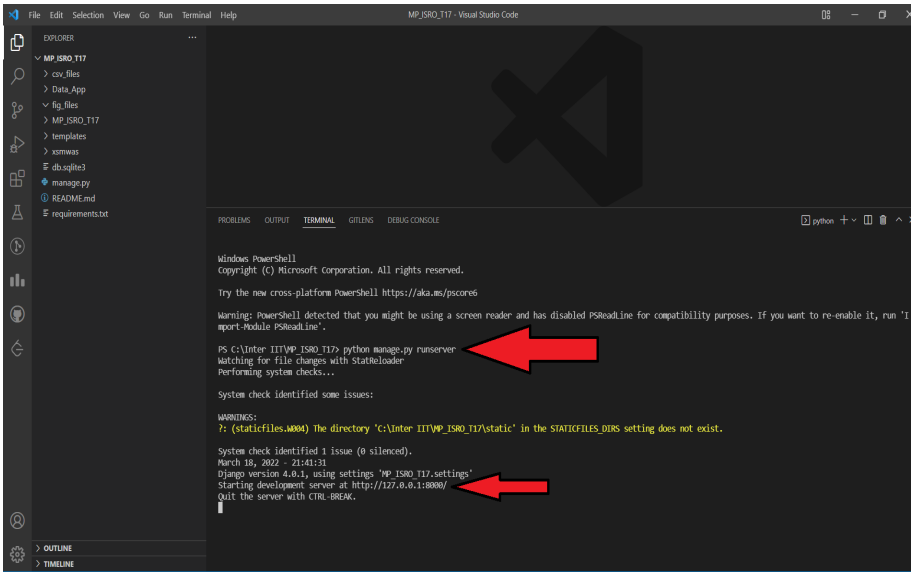

Figure 9: Running the Server

Click on the link in terminal to open the Website

4.2.4 Step 4: Adding the files

Click on ”Add Files” button to upload ls files

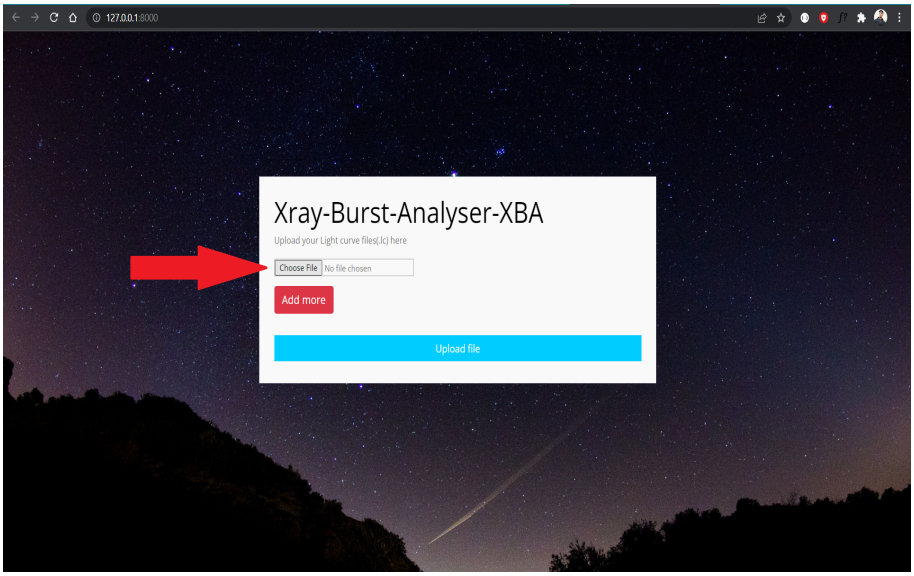

Figure 10: Adding the files

Note: The user can upload multiple ls files

4.2.5 Step 5: Uploading Files

Click on "Upload Files" button to upload process files. The user gets directed to next page automatically once files are processed.

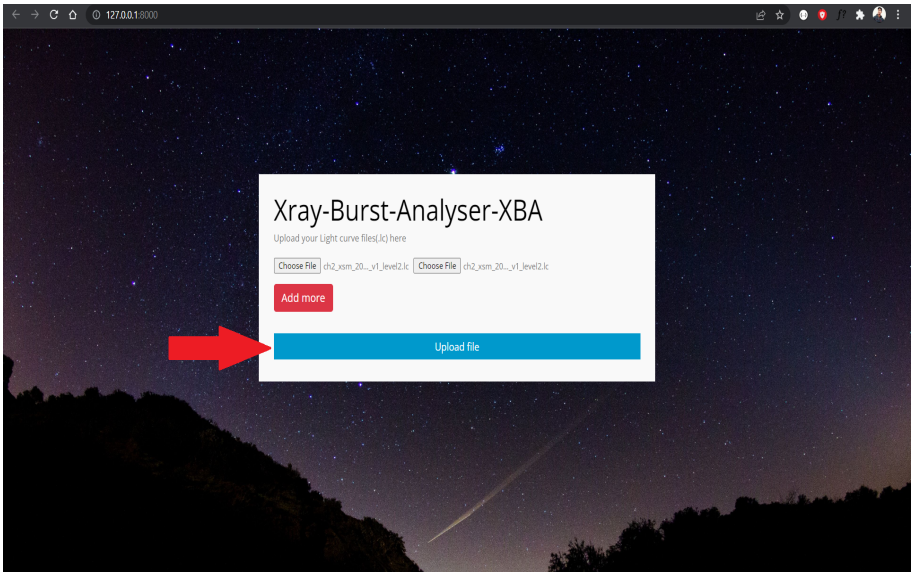

Figure 11: Uploading the files

4.2.6 Step 6: The Graphs for each ls files can be visualized

We get the graphs corresponding to each Data file.

Note: For a closer look, hover the cursor on section of graph you want to zoom.

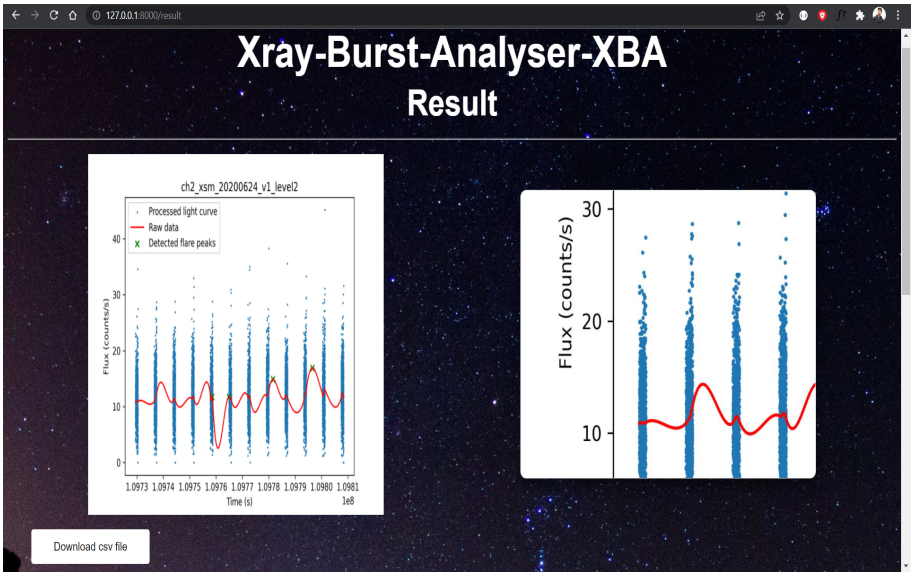

Figure 12: Visualizing processed data

4.2.7 Step 7: Downloading csv file

Processed CSV file for each graph can be downloaded by clicking on ”Download csv file” button

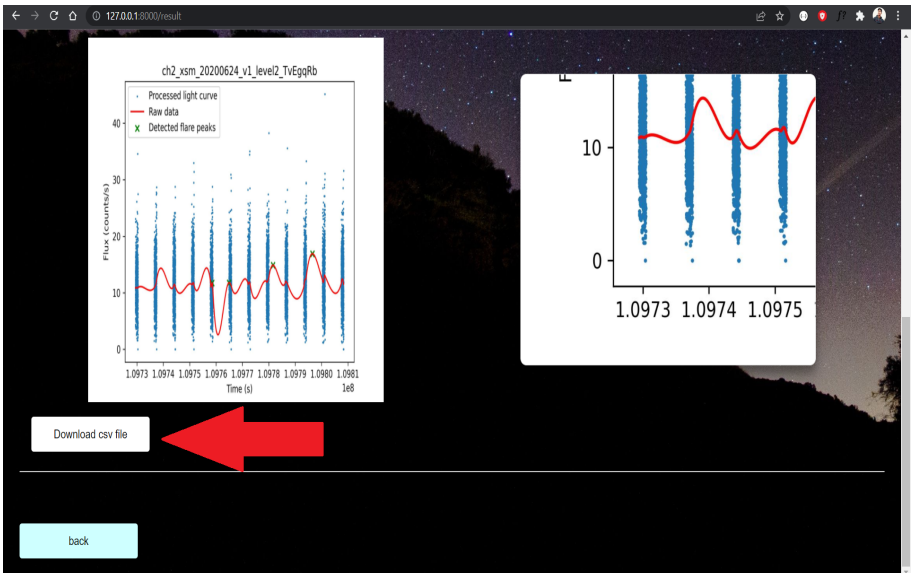

Figure 13: Downloading csv file

4.2.8 Step 9: Back to Home Page

Click on ”Back” button to go back to Home Page

Note: Old uploaded files get removed once user go backs to Home page

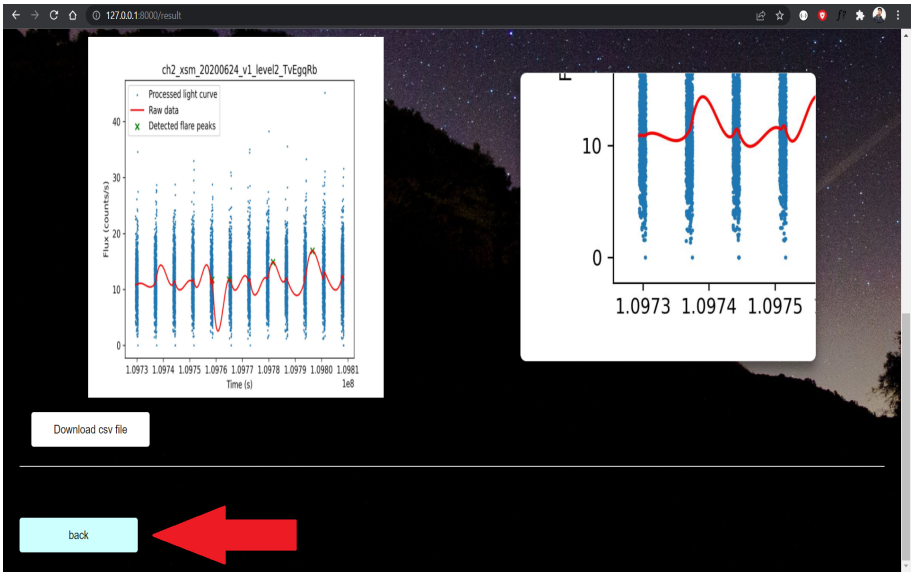

Figure 14: Back to Home Page

5 Understanding flare categories

We have 5 major classes of data, which can be differentiated on the basis of the height of the burst (peak flux above background level), the rise/ decay times for the burst and the decay rate of the burst. These parameters characterize the shape of the burst and can be taken to be a hallmark of the physical processes causing the flare.

An overview of these flare types are as follows:

5.1 Fast bursts (F-bursts) (Plotted in Purple)

This class is primarily characterized by its low decay time. The time doesn't exceed 5000 s and is typically less than 4000 s. The rise times also fall in the same range. This class is also distinctive in having the widest range of decay rates. In general, any burst having a decay rate above 0.005/s will fall in this class. It has a reasonably wide range of peak event counts similar to all the other classes.

5.2 Slow Symmetrical (SS) bursts (Plotted in Yellow)

These are bursts with both rise and decay times above 15000s. They tend to be rare with very low decay rates.

5.3 Slow Asymmetrical (SA) bursts (Plotted in Blue)

These are similar to the SS bursts in having very low decay rates and high decay times (in this case, above 20000s), In contrast to SS bursts, these flares have rise times below 10000s.

The last two classes are intermediate between the slow and fast bursts.

5.4 Intermediate Type A (Plotted in Green)

These are bursts with decay times between 10000 and 20000s. (around 3 to 6 hours). They can be symmetrical, but are mostly asymmetrical with the sow dacay rates being in contrast with short rise times (<5000s).

5.5 Intermediate Type B (Plotted in Teal)

This the second interdiate class with dacay times between 5000 and 10000s. The rise times remain similar to the Type A intermediates, but they have a wider range of decay rates (intermediate to the fast and Type A bursts).

The plots below are useful for demonstrating the salient features of each class:

The 3 plots involving decay time on the Y-axis (Figures 14, 15, 16) demonstrate how decay rate serves as the primary differentiator between curves. High decay rates providing further characterization of the F-bursts. High and low rise times allow differentiation of SS and SA bursts.

The plot between decay rate and rise time (Figure 17) shows how rise time serves as a powerful characterizer for SS bursts while Decay rate serves to characterize F bursts.

Finally, the the last few curves (Figures 18, 19, 20) make it clear that there isn't much variation in peak size between flares except for some exceptional bursts with very high peaks. Thus, peak size is not a very useful classification criteria.

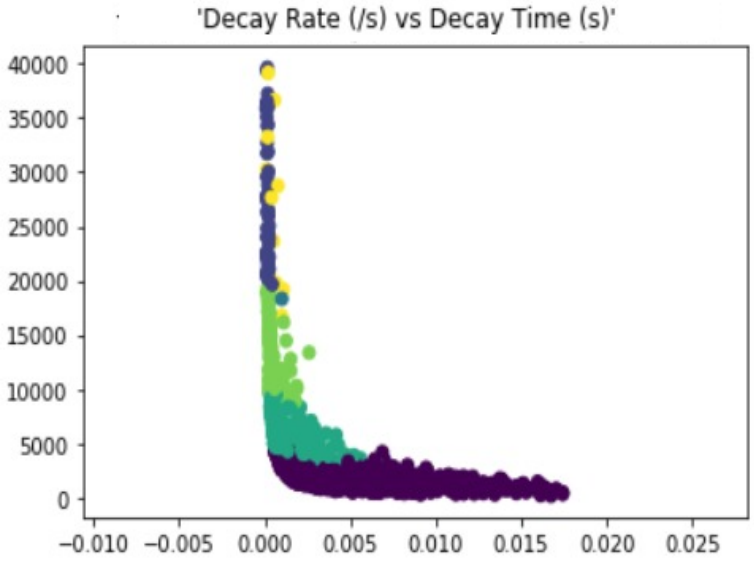

Figure 15: Decay rate v/s Decay time

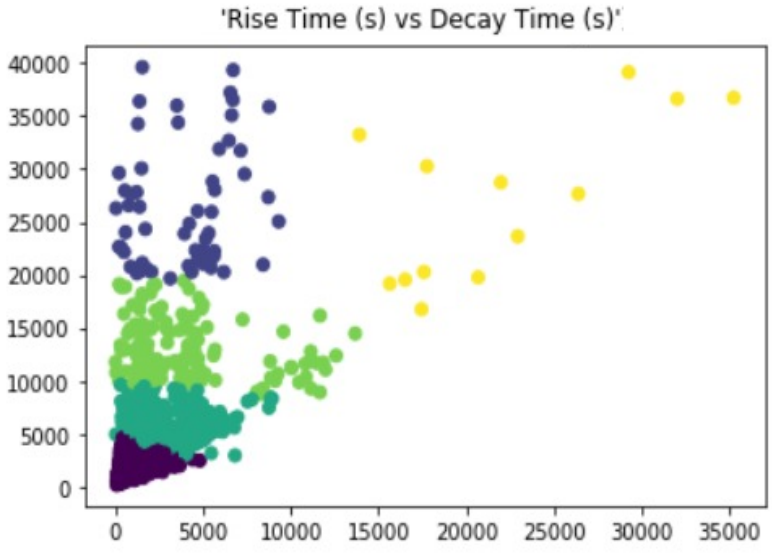

Figure 16: Rise time v/s Decay time

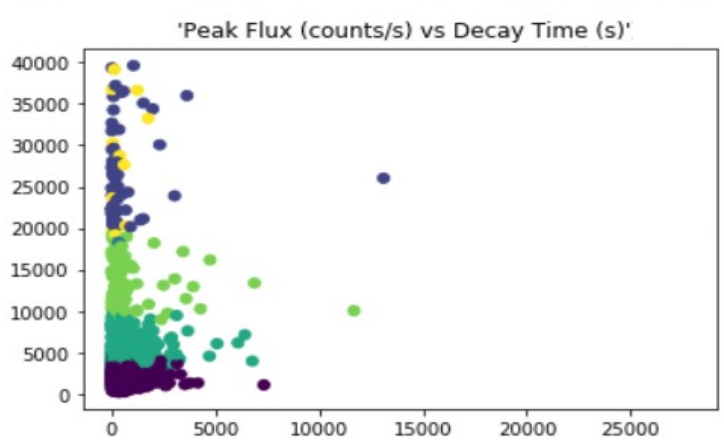

Figure 17: Peak flux v/s Decay time

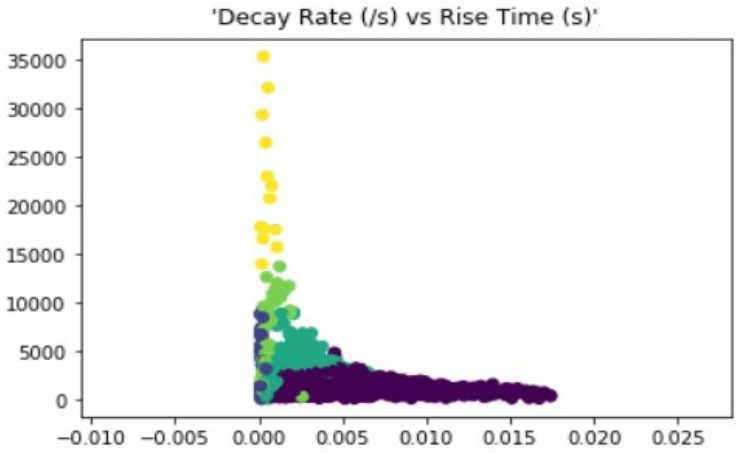

Figure 18: Decay rate v/s Rise time

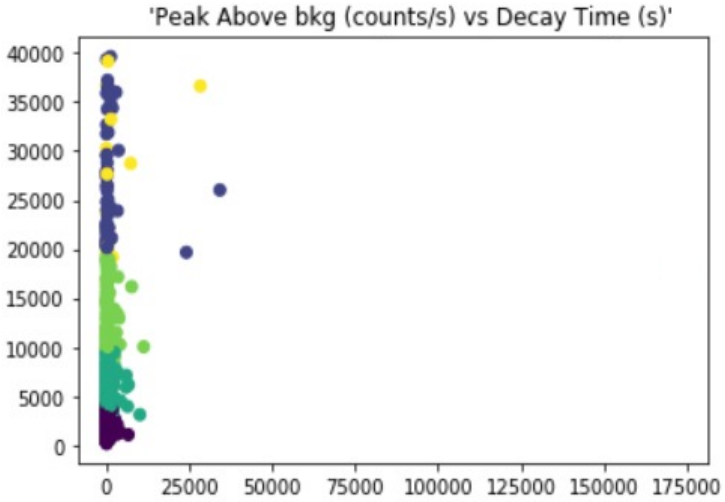

Figure 19: Peak above bkg v/s Decay time

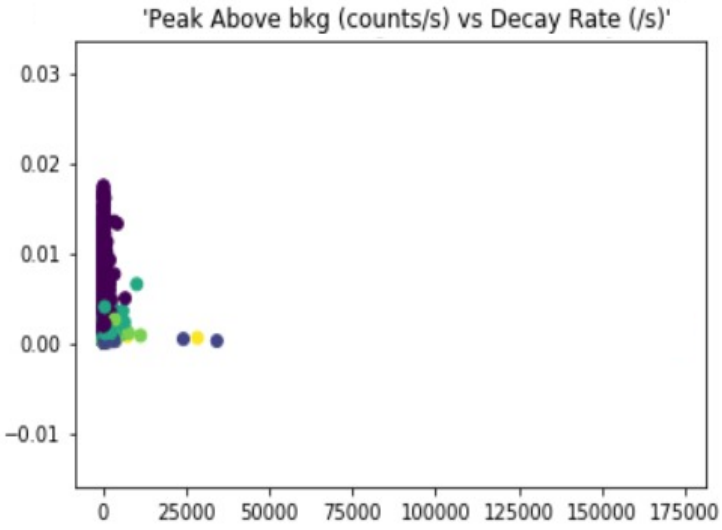

Figure 20: Peak above bkg v/s Decay rate

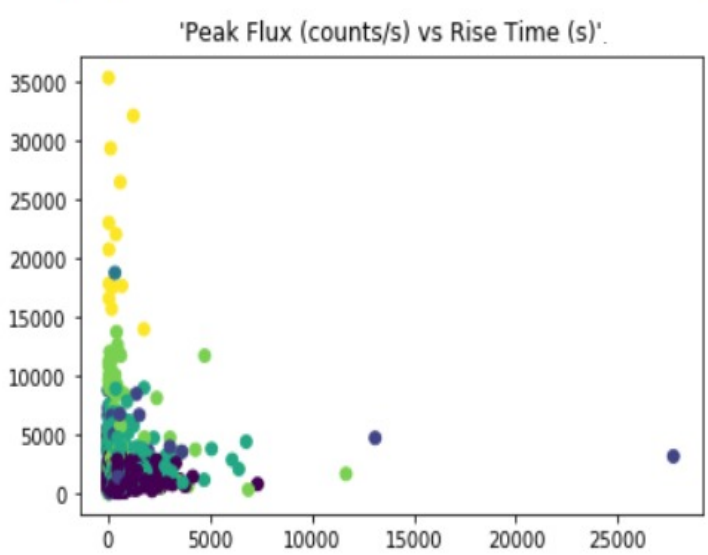

Figure 21: Peak flux v/s Rise Time

6 Caveats

The flare detection and characterization tool developed by us relies on the averaged trend of the continuous stretches of data (a stretch of data is considered continuous if the maximum difference between any two timestamps is less than 100s, which is a value selected by testing several light curve files). So, if any blank stretch of time is flanked on both sides by an increasing and decreasing trend respectively, the algorithm identifies it as a flare candidate. While this can help identify flares whose peaks haven't been recorded, there is a possibility of some false positives occurring. The tool uses some fitting based, and SNR based criteria to pare down the list of flare candidates to suitable flares. This method isn't perfect, and is sometimes unable to distinguish between actual small-peak flares, and the artefacts of the light curve processing stages. Further work is required in this direction. We believe that the usage of machine learning or statistical methods in detecting outliers in vector distributions can be helpful for reducing the false positives during flare identification.

Further work is also required in refining the light curve processing stages, which employ several standard numerical methods, smoothing filters, and regression techniques. We believe that exploring different curve smoothing and regression techniques during the pre-processing stage can help us identify the correct strategy to reduce false flare detection.

The figure (Figure 23) shown below indicates how false positives are interspersed between actual flares of low peak magnitudes.

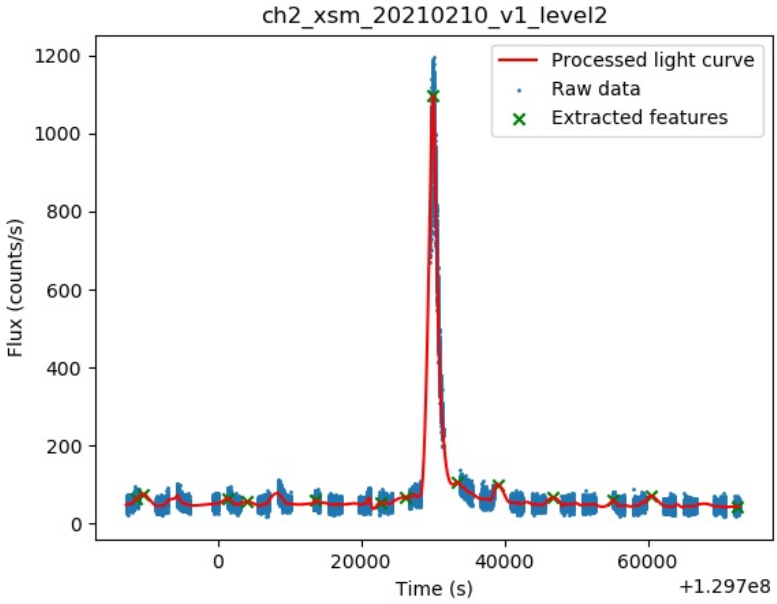

Figure 22: *A case of false positive detection*