Saif Ali Rahman

Ub number: 18010207

Random forest approach for solar flare type predictions

(C-Class Flares)

**Abstract**

This research report will include an analysis of the Solar Flare dataset, accompanied by a demonstration of a working Ai system, in this case a Random Forest approach. This report will provide a system that is able to predict the event of a solar flare (specifically C-Class Flares) accurately according to various predictors. The proposed system will be using a Random Forest Classifier approach and trained using the solar flare dataset provided by the UCI Machine Learning Data Set Repository.

12th January 2022

1. Introduction

A solar flare is natural event that occurs when stored energy near sunspots in the form of ‘twisted’ magnetic fields is released suddenly, leading to an explosion which we call solar flares (What are solar flares?, 2022). Flares are categorised into 3 categories based on the brightness in the x-ray wavelengths. ‘C-Class Flares’ which is my focus, is the most common type of flare that is small and with few noticeable consequences here on earth. The second type is ‘M-Class Flares’ - medium sized and can cause brief radio blackouts in earths polar regions with the possibility of some minor radiation storms in the upper atmosphere. The final and largest type is the ‘X-Class Flares’ - considered to be major events that can trigger radio blackouts throughout the entire planet followed by enduring radiation storms in the upper atmosphere (What are solar flares?, 2022).

This report will demonstrate a Python solution, developed with the goal to be able to predict the event of ‘C-Class Flares’. The program will use the 10 attributes/predictors provided by the dataset to be able to accurately predict the event of ‘C-Class Flares’.

The solution is using a Random Forest classifier approach and trained against the Solar Flare dataset. The goal is to be able to determine and evaluate the accuracy of the model with the goal of being able to accurately detect the occurrence of ‘C-Class’ solar flares.

1. Background

The Solar Flare dataset for predicting the event of a solar flare based upon provided predictors, includes 1066 instances and 10 attributes (13 including the 3 types of flares). The attributes and what they represent are as follows:

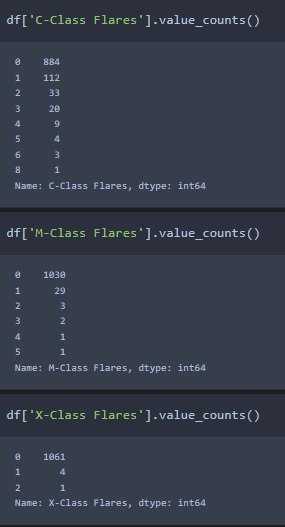
* Modified Zurich class – sunspot classification, code (A,B,C,D,E,F,H)
* Largest spot size – size of sunspot, code(X,R,S,A,H,K)
* Spot distribution – how open or compact the space between sunspots is, code(X,O,I,C)
* Activity – 1 = reduced, 2 = unchanged
* Evolution – 1 = decayed, 2 = no growth, 3 = growth (of a sunspot)
* Previous 24 hour flare activity code – checks if a flare is smaller(1), as large as(2) or larger than an ‘M1’ class flare(3)
* Historically complex – 1 = yes, 2 = no
* Did region become historically complex on this pass across the sun’s disk –

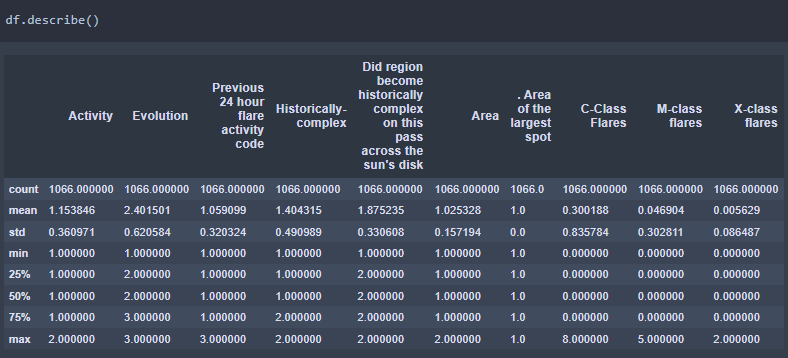
1 = yes, 2 = no

* Area – 1 = small, 2 = large (of flare)
* Area of the largest spot – 1 = ‘<=5’, 2 = ‘>5’

Each of these instances have different outcomes depending on the class of flare. For example the ‘C-Class Flares’ outcome can range from 0-8. The following figure 1 displays how many times (out of 1066 instances) the value for each class of flare was either ‘0’ or above. The program also checks ,for example, how many times the value was ‘1’ for ‘C-Class Flares’, it was ‘112’. The dataset is described in figure 2.

Figure 1 showing the value count of each flare type

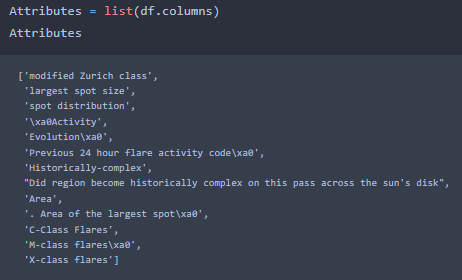


Figure 2 describing the dataset

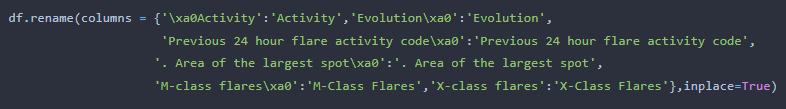
1. Methodology
   1. Preprocessing of the data

One of the most important aspects of implanting a solution into a data set is to first preprocess the data. The data set was first observed and visualized by using methods such as ‘df.describe()’, ‘df.head(5)’ and by listing all the attributes. By creating a Attributes method and calling it, the data showed that some of the attributes contained a \xa0 unicode which is essentially a space in the name, as shown in figure 3.

Figure 3 showing the attributes as a list.

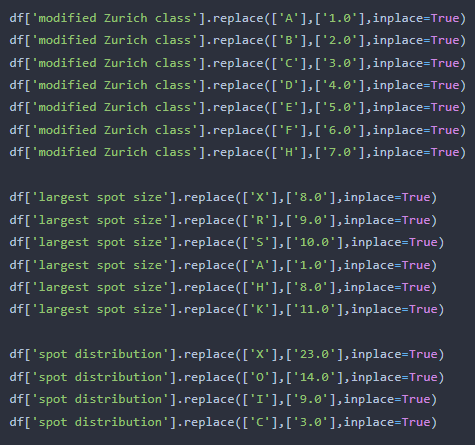


Some attributes weren’t consistently named either, for example C-Class Flares contained capital letters for each word however M-Class Flares and X-Class Flares did not. This was the first issue resolved by simply renaming each attribute as shown in figure 4.

Figure 4 showing how the columns were renamed.

By running the ‘df.describe()’ method my data set displayed all the numerical information available to us from the data set, however as shown in figure 2 the first 3 predictors(modified Zurich class, Largest spot size and spot distribution) are missing and show no data. This is because the first 3 columns are none numerical values and can therefore not be displayed in the same way as the other columns. This proved to be a problem as the Ai model cannot fit attributes that are strings, in this case they were letters (A,B,C…), it requires integers or floats to be able to implement the attribute/predictor into the model. To solve this issue, the letters were replaced with numerical values as shown in figure 5. This does not affect the integrity of the data as the Ai will simply see the new numerical data as a simple value to create a prediction with. Fortunately, this dataset did not contain any other issues such as zero values or nonsensical data and was now ready to be implemented.

Figure 5 showing the letter values being replaced with numerical values.(this was kept consistent through each attribute, for example, A = 1 in both the first and second attribute)

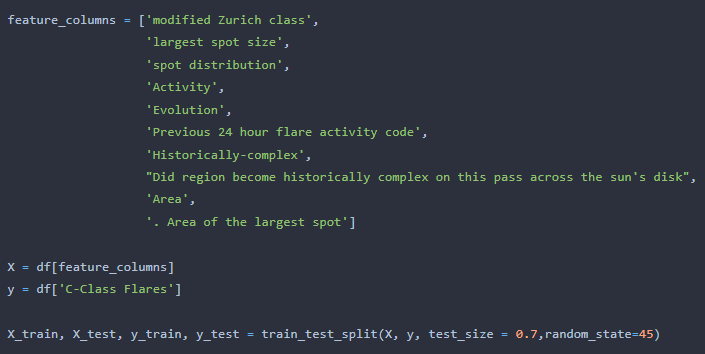


* 1. Developing the solution

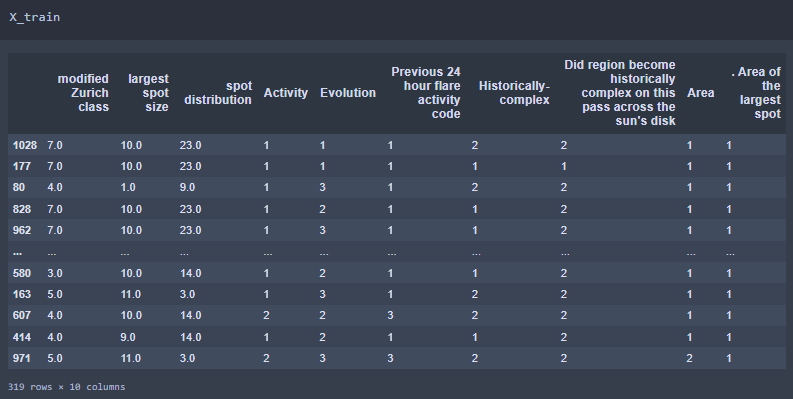
Before implementing a solution, we must first understand what the inputs and outputs of the system are. In the case of this dataset we have 10 attributes that act as predictors for 3 possible outcomes. These outcomes can be outputted several times in each instance, this is what needs to be predicted. The outputs in question being C,M,X class flares. Noticing that the outputs can range from 0-8 depending upon the class of flare, it was evident that a binary solution would not suffice such as logistic regression. Instead, a Random Forest classifier was chosen for its easy to fit structure and the ability to measure the relative importance of features easily which is important when trying to predict multiple outcomes. This solution will allow us to accurately predict against C-Class Flares.

* 1. Creating and training the solution

In terms of training, the data was first split by using the train-test-split method. This is done so that the 10 attributes/predictors can be trained separately from the target data to create an accurate prediction model, as shown in figure 6. They are split by naming the attributes as ‘X’ and naming the output, in this case C-Class Flares, as ‘y’. This method will split arrays into random train and test subsets which gives us the ability to create such accurate models due to the random splitting of data.

Figure 6 showing the train-test-split method.

In this example, a testing ratio of 70:30 was used. 70% of the data was split into a test group and the other 30% into a training group with a random state of 45. We can clearly see that there has been a split in data as shown in figure 7 where ‘X\_train’ is called.

Figure 7 showing the training data with every attribute visible.

Now that we have our ‘X’ and ‘y’ split, we can input our data into our Random Forest Classifier Ai model, as shown in figure 8. However, before the model could be fitted the data had to first be ‘normalized’. To do this a ‘MinMaxScaler()’ method was used. This method normalizes data by transforming all the values in range between 0-1. The method ‘StandardScaler()’ was also used to standardize the data. This method removes the mean and scales each variable to unit variance(Loukas, 2020). These methods were fit by using the ‘make\_pipeline()’ method, which is a great tool that allows the code to be written neater and it runs each method in steps, as shown in figure 9. This creates an Ai model that can predict how many times a C-Class Flare event occurs.

Figure 8 showing how the Random Forest Classifier model was created.

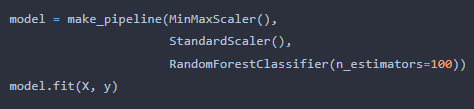
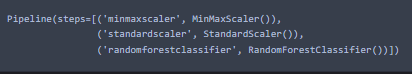


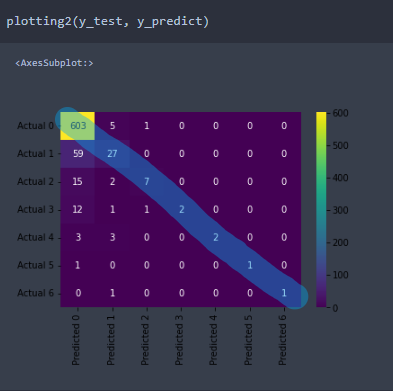
Figure 9 showing make\_pipeline steps.



1. Analysis
   1. Testing

To test the system, the split data was utilized. As mentioned before there is now a test and train portion of data. The training portion, in this case 30%, would be trained however the other 70% could be used for testing only. By doing so I created a y\_predict variable which will be used as a comparison against the y\_test (C-Class Flares output) values. This data was used to create a confusion matrix to illustrate the accuracy of my model as shown in figure 10.

Figure 10 showing the confusion matrix.



* 1. Results

The confusion matrix here shows the ‘predicted’ result on the ‘X’ axis and the ‘Actual’ result on the ‘y’ axis. For example, we can see that my model predicted that the number of C-Class Flares is ‘1’, five times when it was actually ‘0’. Any value that is within the blue line is a correct prediction. This displays that the model is an accurate one as shown in figure 11 which is the accuracy of the test and training sets. And Figure 12 shows a classification report which gives a more detailed insight into the accuracy of the model. Pairing the confusion matrix with the classification report gives a better visualization of the data as the classification report shows how many values there were in total, for example there were 609 cases of ‘0’, and the confusion matrix displays how many of those my model was able to predict.

Figure 11 showing the accuracy score of the test and train set.

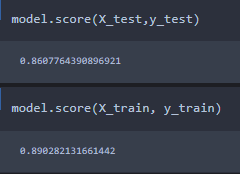
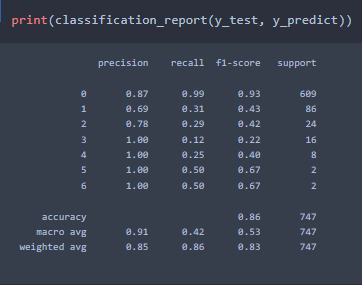


Figure 12 showing the classification report.



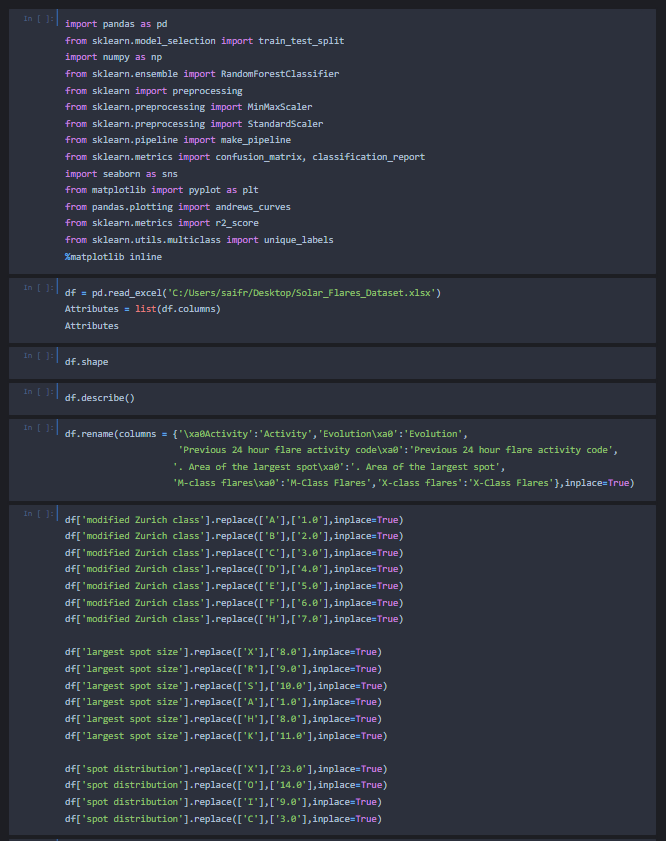
1. Conclusions

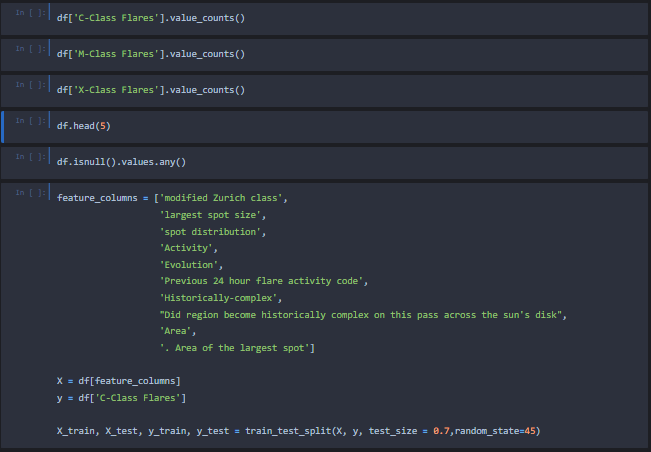
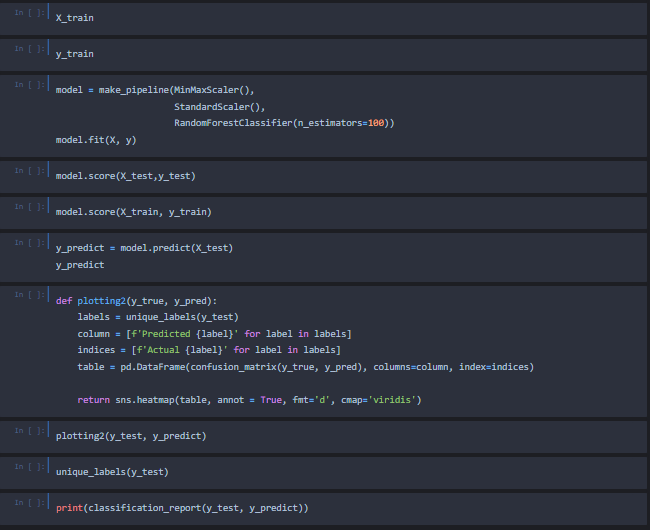
In conclusion, the development of the Random Forest Classifier approach has provided an accurate prediction model for solar flares. The model can successfully predict the event of C-Class Flares and how many occur by using 10 different types of attributes as predictors as was the original goal for this model.

* 1. Future work

The model is not without its flaws. The model lacks in being able to predict a more accurate number of values above ‘0’. Why this is could be because this dataset contains many ‘0’ values which may have hindered the Ai’s performance by creating a bias. This needs to be investigated by using another dataset in future. The model could have instead been made with a Random Forest Regressor as this method is more suited for datasets that contain continuous variables whereas the classifier model is more suited to discrete variables. This may have been an oversight in my work and will need to be improved.

1. Appendices





1. Bibliography

Bradshaw, G., 1989. *UCI Machine Learning Repository: Solar Flare Data Set*. [online] Archive.ics.uci.edu. Available at: <http://archive.ics.uci.edu/ml/datasets/solar+flare> [Accessed 12 January 2022].

The European Space Agency. 2022. *What are solar flares?*. [online] Available at: <https://www.esa.int/Science\_Exploration/Space\_Science/What\_are\_solar\_flares#.Yd8vvvfuys0.link> [Accessed 12 January 2022].

Wwwbis.sidc.be. n.d. *User guide from SIDC - Royal Observatory of Belgium*. [online] Available at: <https://wwwbis.sidc.be/educational/classification.php> [Accessed 12 January 2022].

Loukas, S., 2020. *How Scikit-Learn's StandardScaler works*. [online] Medium. Available at: <https://towardsdatascience.com/how-and-why-to-standardize-your-data-996926c2c832#:~:text=StandardScaler%20removes%20the%20mean%20and,standard%20deviation%20of%20each%20feature.> [Accessed 13 January 2022].

scikit-learn. n.d. *sklearn.preprocessing.StandardScaler*. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html> [Accessed 13 January 2022].

scikit-learn. n.d. *sklearn.preprocessing.MinMaxScaler*. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html> [Accessed 13 January 2022].

Brownlee, J., 2020. *Train-Test Split for Evaluating Machine Learning Algorithms*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/> [Accessed 29 November 2021].

scikit-learn. n.d. *sklearn.ensemble.RandomForestClassifier*. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html> [Accessed 13 January 2022].

scikit-learn. n.d. *sklearn.pipeline.make\_pipeline*. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.make\_pipeline.html> [Accessed 13 January 2022].