Abstract

The main steps behind the given classification task were to understand the data and the meanings behind each feature, filter them, create new ones and identify simple "super-class" classifications based on the gained knowledge, augment the training set, explore with various models, tune parameters and finalize predictions.

The models that seem to perform best on this challenge are the Random Forest based ones. However my best score so far (aprox. 1.5) has been obtained by performing RF Classifier with augmented data, but without some parts of the feature engineering. The Convolutional Neural Network performs better than the Multi-Layer Perceptron and could probably be improved even further.

The following report will give some conclusions about the features, explain how those conclusion were reached, analyse the models in task, provide comparisons and outline certain properties of these.

Task 1: Preliminary data analysis

The given data is split across two files, the one containing the metadata and the one containing measurements. The metadata of an object describes properties of an object that do not change through time, such as location (ra, decl, gal_l, gal_b), distance from source (distmod), extinction of light (mweb) etc.. However, the most relevance for the predictions is in the measurements. For each object (identified based on object_id) there are several entries (from 1 to 352) with information about the flux of the object (the change in brightness), with an error margin (flux_err). These measurements are taken with 6 different passbands ("filters") and have the time at which they were taken recorded (mjd). There are some other features, such as detected, which gives information about the brightness at the time of the first measurement (detected = 0, the brightness was near zero, thus the object un-detectable).

One of the first observations to be made is that the data can be split in two different "super-classes". Based on the redshift (hostgal_specz or photoz) of an object, we can determine wether it belongs to our galaxy or not. Figure 1, from a public kernel [2] shows nicely that there is no overlap in terms of classes for object with zero and non-zero hostgal_photoz. This feature is a less accurate version of hostgal_specz, which is unfortunately not available for the entire test set. However, in the described models, hostgal_specz has been used and when not available, hostgal_photoz.

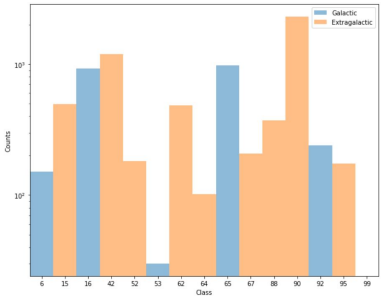

Figure 1: "super-classes"

After separating the data in the two distinct "super-classes", the analysis was conducted separately. The galactic objects are mostly periodic, however some seem to be bursts. My models seem to be quite accurate when deciding whether an object has a burst or not and the idea behind how it decides is looking at the uncertainty of modelling a period.

Extra-galactic objects however are harder to classify. This is because there are more classes with even more subtle differences, as well as the set size being much larger than for galactic objects. Some differences could be the different shapes of the flux, all of which are different distributions, for example plateau or gamma distributions.

Task 2: Feature engineering

In order to make more meaningful predictions on the data, the features have been changed, aggregated etc. Based on background knowledge on the data, from the guest lecture as well as reading the available documents [4], new features were added to better show certain properties of the data, that are not immediately obvious from the aggregates.

Before providing details about the feature engineering, I will detail some other changes made to the data, that helped improve predictions. The first changes on the data were computing aggregates of the flux measurements.

Initially it may seem like leaving all data in, rather than aggregating is a good idea, but there is a different number of entries for each object, meaning that the only way we could bring all the objects to the same format would be augmenting the set by adding more measurements for the objects with under 352 entries. To do this, one would require modelling the flux change (periodic or not) and pick some points on the resulting model, add some noise and introduce them in the set. However, besides this taking very long (even with multithreading.. around 300 hours), there is a high likelihood that it would add more noise rather than actual information and turn out to be irrelevant. Having this in mind, aggregating the time series was the way to move forward. The basic aggregate functions performed for each object were: flux_ratio_sq, flux_by_flux_ratio_sq, flux_diff, flux_w_mean etc.

Another meaningful observation was that each object had measurements taken with 6 different passbands, and combining the passbands loses a lot of information, whereas keeping all the aggregates for each passband separately would perhaps give more insight into the classification of objects. For example, some bursts may only be seen through one passband etc. For this reason, the data was pivotted by passband, thus having 6 separated sets of aggregates for each passband.

The first two feature engineering approaches were regarding the galactic objects. Given that the set is fairly small in comparison with the extra-galactic data, this allowed for long computation to be performed. The first feature added was the period of an object. According to the lecture on astrophysics, the galactic objects are periodic or bursts of light. Using the Multiband Lomb-Scargle Periodogram, available from [5], each object had it's flux values modelled and then the period measured. However, calculating the period helped predict some of the galactic data, but it couldn't identify the bursts, which is why the second feature is relevant. This consisted of calculating a score of the period for each object. If the score was high, it meant the period was most likely correct and the object strongly periodic, whereas a low score will mean that the object doesn't have a period, thus the chances of it having bursts of light are quite high.

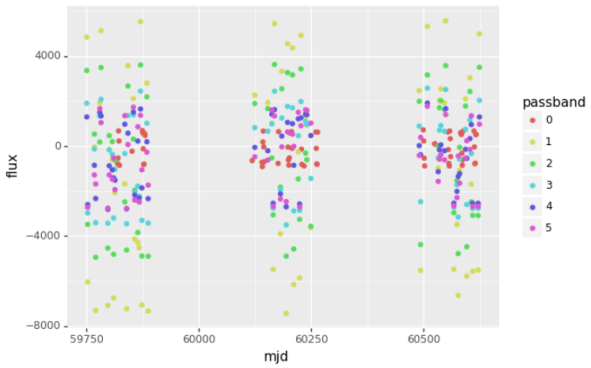

Figure 2: Obj 8442 - phase against time

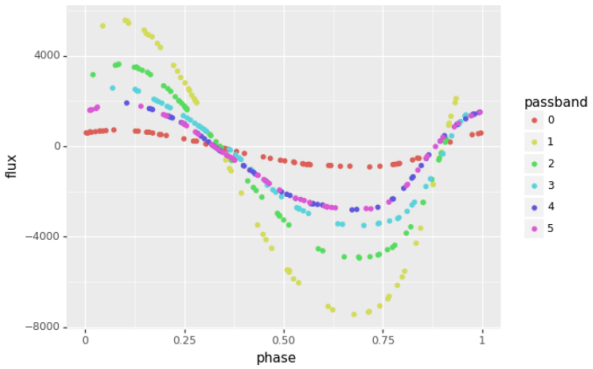

Figure 3: Obj 8442 - phase against period 0.311

The third feature that was added was the absolute magnitude of the objects. However, this seemed to mostly add noise to the data, perhaps because of the way it adjusts to objects which have negative flux. To calculate the absolute magnitude of an object, the formula given in the astrophysics lecture, by by Dr. David Armstrong, was used, after modifying the data. The absolute magnitude was calculated based on the maximum flux of the object, after adding a large constant (modulus of min flux + 1) to ensure that the values are positive before applying log. Perhaps a better idea would have been computing the magnitude for each measurement and then aggregating the values, to give a better idea of the magnitude than the max flux, because this will have no significance for objects with some bursts of energy or an outlier measurement with really high flux values.

One idea of feature engineering that didn't work out in the end was shifting the flux for objects with a 1 as the detected flag, by a large constant. The reason behind this is that when the detected flag is 1, the object was already bright and then the value changed by the flux of the first measurement. By adjusting the values to include this detected flag, there would be a better idea of the actual brightness of an object. However, the submissions didn't include this change, because of insufficient understanding of what the value to shift by could be.

Task 3: Data augmentation

In order to augment the training set data, more objects were created, by doubling the set - creating a new object for each existing one. To do so, each initial object is copied, the id changed and some noise is added to the data. Adding noise should change the object measurements, such that the model can still determine similarities in the basic traits of the two objects, but avoid overfitting. With these considerations, the noise is added to the flux as samples of a normal distribution with the mean 0 and the standard deviation the flux_err for that object. The intuition behind this object is that the generated object would be what the initial one could have been without the error. Noise sampled from a normal distribution (0, 0.5) was added to distmod and to the flux_err. However, perhaps a better idea would have been to modify the flux_err based on the difference between the initial flux and the generated flux.

After augmenting the data once, i.e. doubling the size of the training set, some small improvement in the CV scores as well as LeaderBoard scores was noticed. This is why I decided to try and augment the training set even more and see if this will help the predictions or cause the models to overfit on the training data. After analysing the scores obtained by various models on larger training sets, the observation was that the models seem to perform slightly better (0.07 improvement). Even though the improvement is not very significant, it seems to grow exponentially (the larger the training set, the higher difference in improvements). This is why the final submission has the training set size four times larger than the initial one. A reason for this could be that having more information for each object, helps the models disregard the noise and focus on the more important recurring patterns in the data.

Task 4: RF & MLP on raw data

On raw data, with only some simple aggregates, the RF got a LeaderBoard score of 5.32, whereas the MLP performed worse scoring 6.17.

The raw data consisted of the same number of features on both sets (extra-galactic and galactic), which were simple aggregates of all the initial values. As discovered when predicting using a Random Forest and Cross Validating on the training set, most of these features are insignificant, create noise and therefore have been eliminated.

One possible improvement could be using tsfresh [6] to extract features. The intuition behind this became clear when dropping colomns (see below) and even better understood when working on the engineered features, which were glanced over by the model.

Random Forest Classifier

Initially,the Random Forest Classifier was not performing very well (getting scores of around 12) for the engineered and augumented dataset. However, the impact of changing one parameter was very significant, reducing the scores to 1.73. This was setting the max_depth to be 20 instead of the default None, which proved to be so significant because it avoids overfitting. If the maximum depth of an estimator is too large, then models look into all the different feature combinations possible, however trimming it down forces a higher impact on the more relevant features.

One method that helped improve my cross validation and LeaderBoard scores was by dropping multiple columns for the galactic objects. Dropping most of the augmented features, such as the differences between flux values had a positive impact on the model predictions, thus concluding that for galactic data most of them were just adding noise.

Multi-Layer Perceptron Classifier

Upon seeing the effect of parameter tweaking on the Random Forest, attempts were made to set the right parameters for a Multi-Layer Perceptron Classifier. The impact was smaller than the perviously described max_depth change, but still significant. The final parameters were the maximum number of iterations set to 5 and the hidden layer sizes of (75,75,75) for extra-galactic objects and (85,85,85) for galactic objects.

Based on the observations made when attempting the Random Forest Classifier, more columns have been dropped or adjusted (filled null values, increased by constants), but this did not seem to make an impact on the

performance of the model.

The most important part about this task was standardizing the input. Without standardizing it, the cross validation log loss is around 21.3, whereas afterwards it becomes 1.03.

Task 5: RF & MLP on modified data

Modifying the data has been done incrementaly, i.e. attempting to add a new feature, performin cross validation and checking the feature importance to determine the impact of that particular feature.

After adding the described feature engineering methods and data augmentation, the RF improved significantly to a score of 1.73, whereas the MLP performed even worse, with a score of 6.31. Given that the log loss scores for MLP were not much worse than the RF ones, the MLP model is probably overfitting on training data.

The RF showed a much higher improvement on the augmented and engineered data set.

One method that could help my prediction accuracy increase could be using gradient boosting to sum up more weaker models into a stronger one. According to [7] might perform better on very complex datasets (if they include ranking etc.) Gradient boosting would create trees that would improve on previous ones, rather than pooling multiple trees. It is uncertain if it would perform better, but perhaps it would have been worth looking into (if more time given)

Random Forest Classifier

When adding the periodicity to the galactic objects, I would have thought that it would make a significant impact on feature importance for the RF Classifier, however this was false. Out of 20 features (all aggregates, period, period_score), the period and period_score features were amongst the less relevant ones, with importance around 0.03, where the most important feature being the mean flux value (aprox. 0.8)

Multi-Layer Perceptron Classifier

One perhaps meaningful observation is that the MLP Classifier seemed to perform worse when given augmented data, rather than on raw data. This could be because the model is fitting to the noise of the training set and therefore making worse predictions.

Overall the MLP Classifier performs significantly worse than the RF Classifier implemented. Even though on the raw data, the scores the two models obtain are not that different, the RF makes more of the feature engineering and the data augmentation. One reason could be in fact that the RF avoids overfitting better than the MLP, by limiting the maximum depth of an estimator.

Task 6: CNN on augmented data

- improv: class weights

The Convolutional Neural Network performed worse than the Random Forest Classifier, but better than the Multi-Layer Perceptron Classifier. Despite the accuracy when cross validating being quite low (0.45 using the scikit-learn wrapper for a Keras Classifier [8], the LeaderBoard score was 3.53.

The CNN Was based on a simple Neural Net that was submitted slightly earlier and obtained a better score (3.1). This means that the convolutional layers added losing accuracy. One reason behind this could be that the kernel size is fairly small.

The layers used are Conv1D, Dropout (to avoid overfitting), BatchNormalization (to improve stability), Activation (to introduce non-linearity), Conv1D, Flatten and Dense (to output).

The CNN model could have been improved by applying class weights, to overcome the imbalance between the training set and the test set. This will be detailed in the "Other mentions" section.

Task 7: Progress graph

The graph in Figure 4 shows the progress made throughout the assignment, starting from the date of the first submission (23/11) to the date of the last submission (05/12).

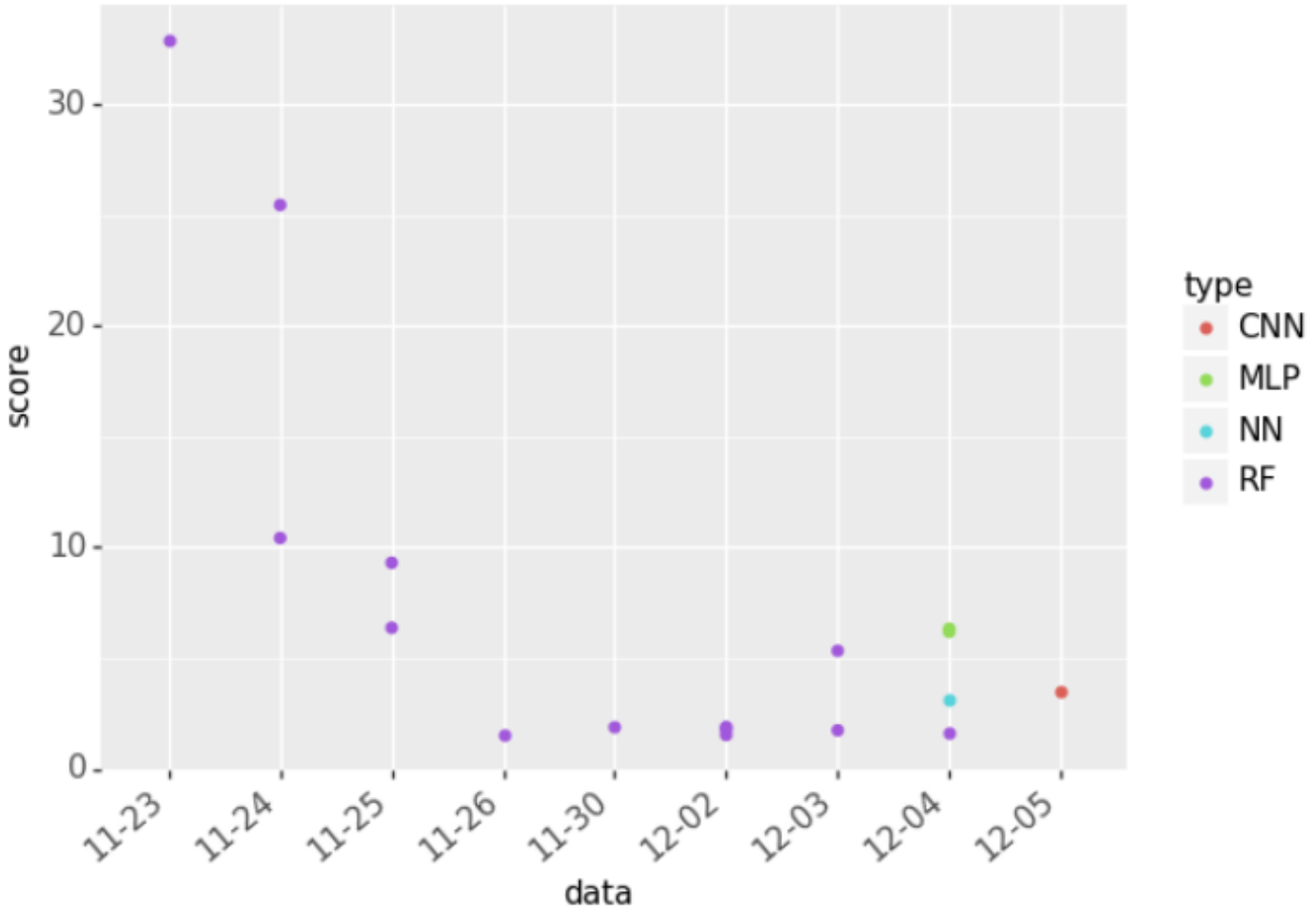

Figure 4: Progress - score against time

Most of the experimenting with feature engineering and data augmentation were done on a Random Forest Classifier, which is why most submissions are labelled as RF. In the second half of the assignment period, models corresponding to the assignment tasks were submitted.

The first 4 models (23/11-24/11) were faulty in the formatting of predictions. The models were predicting fine, but the order of columns in the final DataFrame were shifted in various ways, due to multiple bugs.

The improvement from 9.293 to 6.358 LeaderBoard score was achieved by adding predictions for class_99 (as mentioned below). The next improvement from 6.358 to 1.565 was due to the max_depth parameter, whose change was detailed previously in this report.

Model	**RAW**	**ENG**	**AUG**	**FE&AUG**
RFC	5.32	1.82	-	1.73/1.55
MLP	6.17	-	-	6.31
CNN	-	-	3.53	-

The above table presents nicely the scores obtained for each model, according to the given tasks. The columns represent the various modifications to data - raw data, data with feature engineering, augmented data, and augmented data with feature engineering.

Other mentions & ideas

Below is some information about what improved my models and some improvement ideas that did not quite fit within any of the tasks.

One important addition to the models was predicting class 99 (the surprise class that is not included in the training set). To do so, after making the predictions for each class, I apply the following formula and then scale the results such that all the probabilities add up to 1.

$$\prod_{i=0}^{n}(1 - prob_i); n = classes$$

This equation essentially multiplies how uncertain the model is of certain classes and allocates the remaining "overall uncertainty" to class 99.

Another notable observation regarding the data was that the classes are not equally distributed. For example there are very few samples of class 53. There are a number of public kernels [1] that detail how to compute class weights. These numbers can be calculated by sending repeated submissions to see the score obtained if only one class is predicted. This gives a fair idea of how many objects of that class are in the test set, which can then be compared against the number of objects of that class in the training set, thus resulting class weights.

One possible improvement that was not mentioned above could be anomaly detection, to rule out the outliers in the data set. These could be passing objects, one-time bursts of light etc.

Some of the difficulties encountered were around processing the very large amount of data. This is now done in batches using an available kernel [3] and processing 10000 objects at a time. Another problem encountered when completing the assignment was that calculating the period for galactic object for a test set would take around 300 hours. This was overcome by multithreading the calculation with a thread-pool approach, the different periods were returned by the function and then put together, meaning no mutex locks were needed.

References

[1] Kaggle Discussion Topic, Let's probe Public LB class weights. [online]
Available at: (https://www.kaggle.com/c/PLAsTiCC-2018/discussion/67194)

[2] Kaggle Kernel, The PLAsTiCC Astronomy "Starter Kit". [online]
Available at: (https://www.kaggle.com/michaelapers/the-plasticc-astronomy-starter-kit)

[3] Kaggle Kernel, PLAsTiCC data reader. [online]
Available at: (https://www.kaggle.com/gregbehm/plasticc-data-reader-get-objects-by-id)

[4] The Photometric LSST Astronomical Time-series Classification Challenge (PLAsTiCC): Data set. [online]
Available at: (https://arxiv.org/abs/1810.00001)

[5] Multiband Lomb-Scargle Periodogram. [online]
Available at: (https://www.astroml.org/gatspy/periodic/lomb_scargle_multiband.html)

[6] Tsfresh. [online]
Available at: (https://tsfresh.readthedocs.io/en/latest/)

[7] Gradient Boosting vs Random Forest. [online]
Available at: (https://medium.com/@aravanshad/gradient-boosting-versus-random-forest-cfa3fa8f0d80)

[8] Keras Wrapper in scikit-learn. [online]
Available at: (https://keras.io/scikit-learn-api/)