CS342 Machine Learning – Assignment 2

u1600779

Abstract

This report summarises progress through the PLAsTiCC Astronomical Classification Challenge [1] following the tasks outlined in Assignment 2 alongside any other undertaken work. Exploratory data analysis was initially carried out alongside research into the relevant astrophysics behind the challenge; this research was supported by insight from astrophysicist Dave Armstrong.

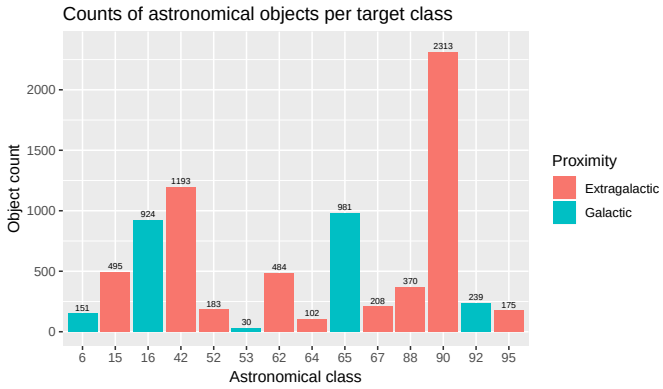
The challenge itself was concerned with classifying different types of astronomical objects based on flux values represented as light curves across astronomically filtered frequency intervals (referred to as passbands) of light. Numerous classifiers including a Random Forest, Multi-Layer Perceptron, Light Gradient Boosting Machine and Convolutional Neural Network were built and submissions made for each. All of these were somewhat successful in terms of surpassing the simpler benchmarks, with notable performance from LightGBM and Random Forest once feature engineering was carried out. The challenge was a struggle given the context, but rewarding in terms of the progression seen throughout the project.

Exploratory Data Analysis

The EDA carried out mirrors the EDA done by Danilo de Silva in places as his exploratory resource [2] is a fairly comprehensive one. The first step taken was to read the documentation about the data to determine the names and relevance of the columns in the provided datasets. Further reading was done around the topic and the provided Starter Kit [3] was followed in order to gain familiarity with the data and task. The data was split into two main parts: a metadata file which contained all the object-specific features (including a target class for the training data), and a larger file containing multiple passband-flux observations for each of these objects which were to be used to classify them. This simulated data was built to represent the time delays between each object's passband flux readings; some of the readings were very spaced apart and others taken relatively regularly (depending on how often the telescope looks at that location with a specific filter), which was something to consider during feature engineering and classification.

With such a huge dataset, it seemed sensible to first investigate different splits and identifiers amongst the data, so that it might be observed with greater clarity and eventually classified in more manageable ways than attacking it all at once. The first and perhaps most obvious way to split the data was into galactic and extragalactic objects, this was decided after observing that a large number of rows in the training set contained *NA* values for `distmod` which is a statistic derived from red-shift values.

Galactic objects (those present in the Milky Way) were given a red-shift of zero due to their relative proximity which resulted in the observed *NA* values used to split the data. This split was backed up by the discussion provided by Dave Armstrong who also highlighted potential difficulties in classifying some of the extragalactic classes that likely correspond to different types of supernovae. Indeed, upon summarising subsetted data which contains only the proposed galactic classes, the red-shift values can be observed to all be 0. The split between galactic and extragalactic classes and the counts of each can be seen in the plot below.

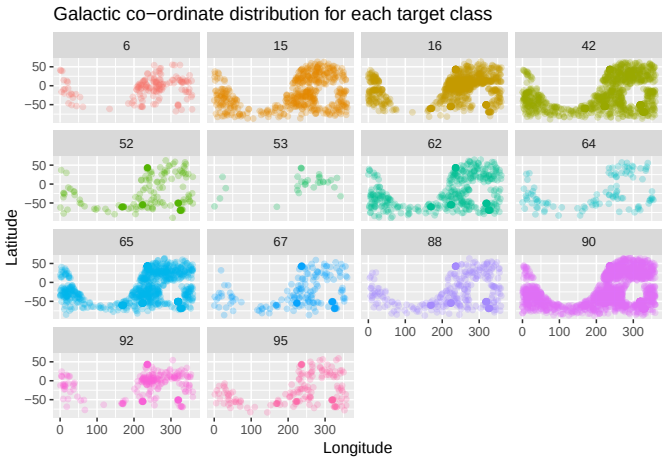


From this plot it can be observed that there are a wide range of counts for each class, and there are substantially more extragalactic data points than galactic ones. This could be to aid in the difficulty in differentiating between some of the extragalactic classes which mainly seem to correspond to various types of supernovae. Other classes could have their flux values plotted over time to reveal their 'shape' and in some cases their periodicity. It was suggested that some of the classes correspond to things such as cepheids, asteroids, lyrae and more, so it was worth investigating the differences in flux patterns between them in the hopes that this could aid in classifying them down the line. *See the Appendix for the plots of flux over time, sourced from Danilo's EDA* [2]. In practice it proved challenging to directly utilise the knowledge of these shapes, though it can be seen clearly that some of them do have unique signatures in the flux-by-time domain.

Periodicity as a concept did prove useful and a feature was introduced to help separate the periodic classes from the burst event / singular occurrence ones. Insight from Dave Armstrong was invaluable here in opening up this avenue for exploration, knowledge of periodicity of different classes (cepheid's being greater than 2, whilst lyrae being around 0.5) made this seem like an area of great potential for

classification. It will be assumed throughout that the reader has a reasonable knowledge in astrophysics and may have seen Dave Armstrong's proposed identifications of each class.

Part of the data corresponded to the galactic co-ordinates of different objects; it seemed wise to plot the distributions of objects by class, again in the hopes that obvious patterns would emerge. However, as can be seen in the plot below, this was unfortunately not the case.



Some of the remaining features included a flag to show whether the object corresponded to the DDF (Deep Drilling Field) or WFD (Wide-Fast-Deep) Survey, this could be useful in classification as it is an explanatory variable for the uncertainty in light-curve readings. DDF was a more focussed look at a small part of space whilst WFD observed a huge number of points but less frequently, increasing the uncertainty in measurements taken. This could allow for the inference that if the uncertainty is high for other reasons something else may be causing the uncertainty. It is also worth noting the presence of class 99 which was not present in the training set and meant to represent any objects which fall outside of the definitions of the other classes; allowances had to be made for membership of this class in the classification stage.

Feature Engineering

An important initial stage to take with the data was to aggregate it into statistics such as means, medians, skewness etc. These aggregations allowed for classification to occur on an object-by-object basis given that the data provided contained multiple completely separate flux readings for every object across different passbands and time scales, meaning that it was necessary for this information to be aggregated in order for it to be used in any meaningful way towards classification. In this particular time series data set, taking moments beyond proved to be a valuable strategy; as was determining ranges of values over time and how these interact with the calculated moments (means, skews, etc.). These higher order moments helped capture some of the dependence between readings and relationships over time present in the data.

As the EDA might suggest, some obvious feature engineering strategies to try included a means of determining whether an object was galactic or extragalactic via the use of `distmod`. This feature was implemented but eventually superseded by some of the strategies discussed later on - at least explicitly - though it may be assumed that the other features made this separation occur implicitly.

The *Time Warp Edit Distance* algorithm mentioned in the suggested reading [4] showed promise and was initially manually implemented in the hopes that it would aid in classification. However, perhaps due to a lack of subject knowledge, or inappropriate applications of these techniques, submitted scores seemed to worsen with this algorithm included.

As an alternative to this brute force, manual approach, the `tsfresh` Python package which is designed to handle some of the intricacies of extracting features from time series data was used for the majority of the following feature engineering. A resource [5] on time series analysis proved useful in deciding which features in particular were worth extracting and feeding to the models.

Through familiarity and knowledge within the domain of mathematical statistics alongside evidence from the attempts made by other high-performing entrants in the challenge it seemed that moments (namely mean, skewness and kurtosis) of `flux` data made for good features to extract from the aggregated data. Ranges in `mjd` were also extracted in order to determine lengths of things like strikes above or below a mean as well as to be used in determining the presence of periodicity in an object's flux over time. Moving averages of `flux` values also proved effective; all of these terms are described in the resource referenced above but are quite standard in the realm of aggregate feature engineering for time series data.

Flux-per-passband ratios were also calculated as relative measures to give information that is directly comparable between objects in terms of the amount of flux present in each passband. This allows a model to pick out features of classes which behave uniquely in specific passbands, e.g. being particularly active in the ultraviolet range.

Possible shape-matching avenues were explored based on the flux over time graphs mentioned previously but these seemed too complex to implement directly; flux ratios and derivatives of these showed more promise alongside the aggregate moments mentioned above.

Another notable created feature separated single event occurrences such as supernovae from the cyclic event categories such as cepheids. This feature was discovered through the Kaggle discussion boards and proved useful to some of the top performing challenge entrants [6]. It is constructed by defining a feature as the difference in maximum and minimum `mjd` for objects which have the detected flag set affirmatively. This acts to allow the

model to differentiate between those objects which are repeatedly detected with a period and those which have a single main detection (bursts, comets, supernova etc.).

There were a lot of different options in terms of features to extract and as the competition progressed it became apparent that certain combinations worked well in terms of activating each others' effectiveness and aiding in classification.

Data Augmentation

In order to be able to use Convolutional Neural Networks, some data augmentation was required. To achieve this, each object was iterated through and each observation's `mjd` recorded. These were used to calculate the maximal length of the time-series corresponding to this object which was then used to place each object's aggregates into bins of fixed size, leading to a new representation of the data through transformation. The bins contained data for an object's `flux`, `flux_err` and `detected` flag which were deemed to be most informative in this context for feeding to a CNN. Means were calculated and the flux was normalised for consistency between bins.

Classifiers

*The Kaggle username for this competition was **CS342u1600779**, display name was **CS342 u1600779** and the best ranking was somewhere around 400th. This was due to an earlier RFC submission mainly, at time of writing the ranking is 504th. For this reason, I would suggest that looking at rank number is somewhat arbitary, but my maximum score was 1.467. In the archive for this assignment, `<name>bare.py` corresponds to code for the attempts made before proper feature engineering whilst `<name>features.py` is hopefully self-explanatory.*

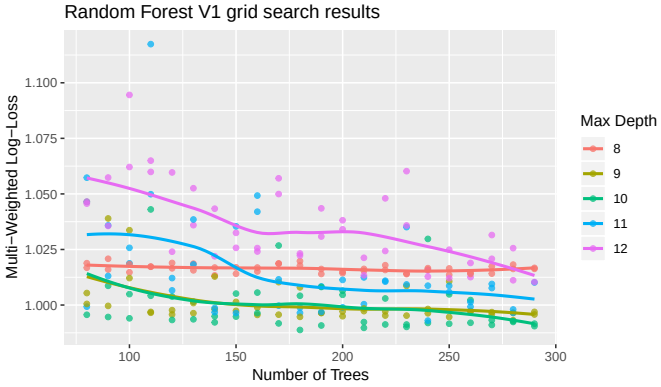
Complex time series data presents many challenges; the potential lack of independence between observations can leads difficulties due to high levels of autocorrelation in the data. Different classifiers were implemented and tested in order fulfil the tasks laid out in the assignment as well as to explore the potential of each in this particular scenario as often different classifiers can be better suited to different data.

Random Forest

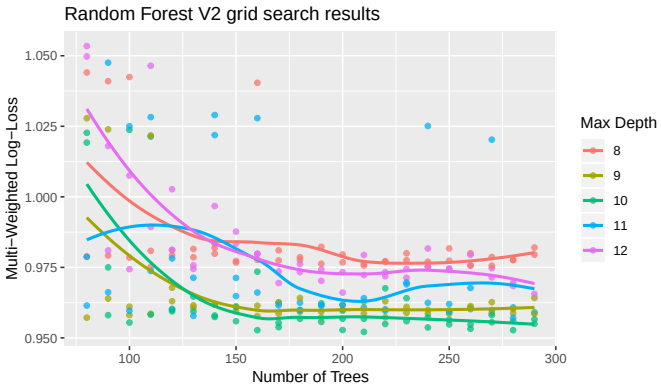
Random Forests are a familiar and relatively robust means of classification when feature selection is used. The first forays into classification were somewhat unsuccessful as aggregates were not properly applied; after these issues were resolved, a custom grid search function built to utilise a multi-weighted log-loss function designed to match that specified in the challenge was performed with a wide range of hyper-parameters to be tuned.

It became evident that the `entropy` criterion tended to perform better on the training data. Depths ranging from 8 to 16 were attempted alongside `sqrt` and `log2` max feature specifications, and a number of trees ranging from 80 to 300. A balance between the number of trees and performance was found at around the 200 mark alongside a maximum depth of 10; a lower number of trees was preferred to avoid overfitting.

The first iteration of this classifier - denoted as **RF V1** - did not properly utilise aggregate statistics and also suffered from some overfitting issues, meaning that it scored relatively poorly, similar to the naive benchmark. The aggregates used in classification were over `flux`, `flux_err` and `detected`, they included means, medians, extrema, skew and so on (see implementation for specifics).



After rectifying these issues and introducing some feature engineering, **RF V2** attained a score of around 1.7 after grid searching again through the parameters mentioned above. As the challenge progressed and the feature engineering efforts were refined, a score of 1.467 was eventually reached which remains the best submission score throughout this assignment.



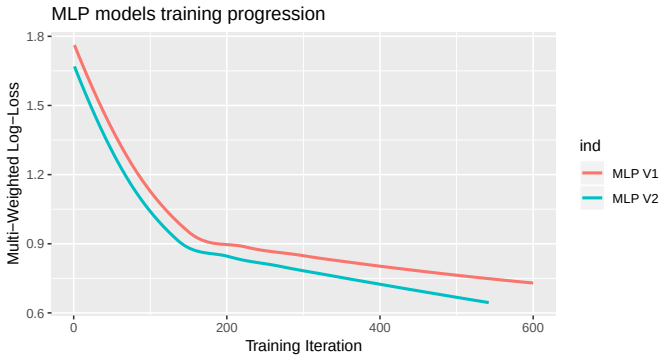
Multi-Layer Perceptron

As a first exploration into implementing an Artificial Neural Network, this one proved itself to be relatively simple. Using `sklearn` allowed for a grid search approach to be

taken again in terms of the hidden layer sizing and count, as these configurations could easily be iterated through. Four hidden layers of sizes $(256, 128, 64, 32)$ were eventually settled upon. The advantage of using an MLP is that it can identify data that is not linearly separable, and it can be trained easily using back-propagation, the number of iterations was throttled as the competition progressed to avoid overfitting as this seemed to be a potential issue with the classifier. Its performance in training was somewhat erratic but did not lead to massive fluctuations in the submission score which suggests more tuning is required to ensure training scores are representative.

It is worth noting that some additional aggregates were used for the MLP model to feed it information on `mjd` and `passband` as well as those mentioned for the RF classifier. Scaling was also used to ensure the inputs and outputs of the model were correct. This model did not seem as effective as any of the other approaches, perhaps due to its simplicity and the comparatively small training set relative to the whole dataset.

The graph below shows the MLP's training results before and after the inclusion of the features discussed earlier. Here, **MLP V1** is the first iteration of the model, whilst **MLP V2** represents an almost unchanged model with the sole addition of feature engineering.



Interestingly, the benefit of training beyond around 200 iterations decreases noticeably. This could suggest that training could be stopped earlier but further testing would be required to confirm this. It can clearly be seen that **MLP V2** is superior in training; this was reflected in the submission scores as well. The MLP with and without feature engineering gave disappointing results. This could be due to potential overfitting on the data and perhaps some feature augmentation to increase the effective training set size would have helped improve the approach.

Convolutional Neural Network

A Convolutional Neural Network has fixed input size and is usually comprised of convolutional layers which are then flattened or averaged to a 2D dense / fully connected section of the network. These types of Neural Networks are traditionally associated with images but can sometimes be effective at classifying other types of data if it can first be put in the correct format. They excel where traditional NNs would become overwhelmed with an abundance of weightings in hidden layers dealing with multi-dimensional data. A CNN can deal with inferred spatial dimensions (width, height etc.) in the data to handle it more effectively. In this challenge, they are powerful in terms of extracting their own features in the pooling layers as they are trained, potentially leading to previously unnoticed features and better scores.

The model itself was built as a sequential set, starting with three *2D Convolutional* layers which have neurons arranged to better deal with dimensions as mentioned above; these are separated by *Batch Normalisation* and *Leaky Rectified Linear Unit (ReLU)* layers. After this, a transformation was required to constrain the dimension of the data moving through the network in order to eventually lead to a valid output. A *Flatten* layer was trialled (**CNN Flatten**) but generated a huge number of parameters and seemed to overfit on the training data, so instead *2D Global Average Pooling* (**CNN 2GAP**) was used to again bring the data into the necessary dimensions but do so by taking means rather than simply generating all possible combinations within the input kernel.

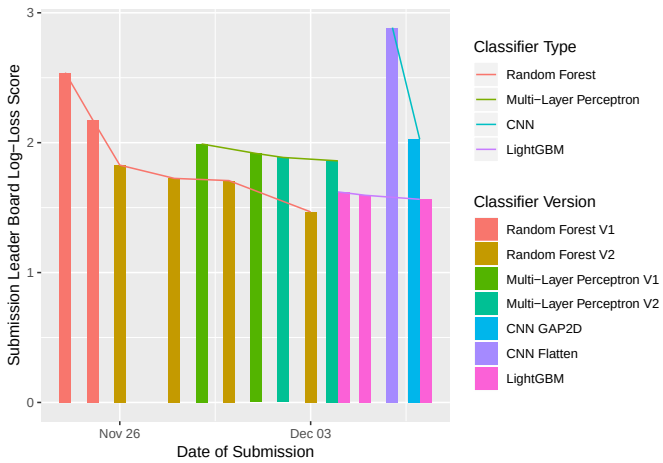
This approach brought the training and validation scores more in-line. Following this flattening / averaging, a fully connected section could be built with three *Dense* layers separated by *Dropouts* and *Leaky ReLU* activation layers; the final Dense layer utilising a softmax activation function to lead to the desired output of multi-class probabilities (softmax outputs a probability distribution which ensures the output values sum to 1).

The design was partially inspired from reading an influential paper on Deep Residual Learning [7] which refers to a similar initial setup. The data augmentation technique described earlier made the topics and structures of CNNs discussed in this paper applicable as the time series data was transformed in such a way that it could be treated as an image or equivalent format with respect to training a CNN to recognise features.

The models were both trained using categorical cross-entropy as their loss function, due to its parallels with the challenge's defined loss function, and were given 100 by 2 epochs of training time. The progression through training with both choices of flattening layer can be seen in the graph overleaf, along with the final multi-weighted log-loss score for both trained models on some validation data, justifying the final choice of flattening layer.

Table 1: Multi-weighted log-loss scores after training

Flatten Layer Type	Training Score	Validation Score
Global Avg. Pooling 2D	1.373189	1.617589
Flatten	2.089344	1.017481



Extension Task : Light Gradient Boosting Machine

A relatively new and very interesting approach was discovered through Kaggle discussion and other research: the Light Gradient Boosting Machine model (**LGBM**). There are some explanations to its structure online [8] but it is essentially a gradient-boosting, tree-based learning algorithm which builds its trees vertically (leaf by leaf) rather than the majority of alternatives which only deal with depth (level by level). It proved effective upon initial testing, and is relatively performant in terms of running and training time compared to the models discussed above. One of the downsides of this model were the sheer number of parameters to be tuned; other kernels were used as starting points for this but it seemed like this model did not reach its full potential due to time constraints on tuning. It is still shown in the progression graph for reference and code for this classifier alongside all of the others mentioned above is present in the archive of this assignment.

Progression Graph

The best submissions for each model from each day are recorded in the graph below. *Despite this being one of the most challenging and engaging projects undertaken during my time at university, I unfortunately had to deal with a lot of competing deadlines and other circumstances which made it difficult to begin making leader board submissions for some of the models as early as I would have liked.*

Despite this, there is still some clear progression from the first submission to the last and the differences and development between models can be seen. The key corresponds to the iteratively developed versions of each model, which have been mentioned in their relevant sections.

The Random Forest classifier in particular shows great progress. This is testament to the time spent on it as it was the first model attempted and thus the one that gave the most opportunity to reflect on hyper-parameter tuning and how features might interact with it. The CNN showed good progression through changing only a single layer; this could suggest that further changes might bring the score down even more, it is unfortunate that a working CNN was only reached near the deadline. The extension attempted surrounding the LGBM model also showed promise but despite numerous configurations of parameters being attempted it could not compete with the Random Forest classifier.

Given the progression present over time in all of the approaches it may be assumed that better scores *are* possible; perhaps just a few parameter or feature alterations away. All of the submissions were improved upon after initial submissions, indicating that if these trajectories were to be followed into the future better results could be obtained.

Conclusions

There were some inherent limitations in the data due to the way in which it is collected. Namely, in the WFD Survey some of the errors were extremely high and may have hindered classification; developing better ways to deal with this uncertainty could have greatly improved classification. Another limitation is that it seemed (based on research and expert opinion) that many of the extragalactic classes corresponded to different types of supernovae. It is difficult to differentiate between these classes without a strong foundation in astrophysics and it is likely these that stopped the submissions seen above from reaching as low of a loss as some of the top entrants. Often a lot of trial and error was involved in deciding on what to aggregate and which features to build, more time to gain domain knowledge would have streamlined this process and perhaps made it easier to differentiate between some of the classes if the models could be guided more effectively from the start.

The training results for CNNs especially were promising

in terms of initial training scores and improvements, but unfortunately this was not reflected in the first submissions made to the leader board. Given more time, the CNN would hopefully grow into a robust and novel classifier for a challenge of this type, further data augmentation would have helped as these models require a lot of training data to be effective. This disparity between training results and the actual submission score plagued a lot of the models especially in the early stages of their development.

It is somewhat unclear as to which model may be the best to pursue in the future. However, the more complex models were likely not used to their full potential in a project this short in length and it is likely that with time they could attain leader board scores less than 1. Long Short Term Memory nets and Recurrent Neural Networks are also of interest for future development, these approaches should work well on this particular task without forcing the data into a specific format like the CNN did. They could potentially deal with some of the autocorrelation present in time series data where recurrent readings are not necessarily independent, which a lot of the classifiers built above could not deal with particularly well. However, due to prioritising the models that the tasks requested, these were not implemented in time. Exploration into these ideas would definitely be something to pursue in the future.

Building on this point of a lack of clarity or opportunity to rigorously form comparisons between models, the feature engineering remained as a black box of sorts throughout the assignment with it being applied in full to all of the models attempted (except CNN), moving forward it would be wise to investigate which features work best for specific types of classifier as it is likely that some features may benefit more, less or even hinder classification depending on the workings of the underlying classifier.

Overall, the scale and nature of the data in this challenge proved to be the biggest difficulty. The tasks offered some structured progression but also left a lot of work in terms of tuning. Additionally, due to the size of the dataset, some models took time in the order of hours to do a complete training and prediction run which made it difficult to carry out in-depth testing. Despite this, a respectable final score was gained alongside a lot of knowledge in previously unseen classification areas, leading to an overall positive result for the assignment.

References

[1] Allam JrT. *et al.*, "The photometric lsst astronomical time-series classification challenge (plasticc): Data set," *arXiv preprint arXiv:1810.00001*, 2018.

[2] D. de Silva, "PLAsTiCC Dataset - Complete EDA." https://www.kaggle.com/danilodiogo/the-astronomical-complete-eda-plasticc-dataset.

[3] The PLAsTiCC Team, "PLAsTiCC Starter Kit." https://github.com/LSSTDESC/plasticc-kit.

[4] W. Vickers, "Collection of Time Series Classification Algorithms." http://www.timeseriesclassification.com/algorithm.php.

[5] Z. Rong, "Introduction to Time Series Analysis." https://netman.aiops.org/~peidan/ANM2018Fall/5.KPIAnomalyDetection/LectureCoverage/2017Tutorial_Introduction%20to%20Time%20Series.pdf.

[6] G. Sionkowski, "Kaggle Discussion - Event Type Separation Feature." https://www.kaggle.com/c/PLAsTiCC-2018/discussion/69696#410538.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016 [Online]. Available: http://dx.doi.org/10.1109/CVPR.2016.90

[8] P. Mandot, "Medium - LightGBM Explained." https://medium.com/@pushkarmandot/.