Department of Computer Science
University of Warwick

Student ID: 1610375
Module Organiser: Theo Damoulas

CS342 Machine Learning: Assignment 2

PLAsTiCC Astronomical Classification

Final Report - December 2018

Abstract
PLAsTiCC is a large data challenge that attempts to classify astronomical objects by analysing the time series measurements of the 'light curves' data (intensity of photon flux) emitted by cosmological objects using six different astronomical passbands[1]. The flux may be decreasing or increasing over time, the pattern of the changes in its brightness acts as a good indicator of the underlying object. Each object in the set of training data belongs to one of the 14 classes, in the test data there is one additional 15th class that is meant to capture "novelties" (objects that are hypothesised to exist).

Throughout the analysis it was identified that domain specific feature engineering is of high relevance to identifying cosmological objects. The absolute magnitude of a flux has proved to be a very significant feature that lets us compare cosmological events that might be at different distances from the telescope. The difference between the maximum and minimum MJD values for detected observations of flux is an extremely useful domain specific feature that can separate a "one event" from a "cyclic event". Not to mention, a fourier transform of the time series data is yet another feature engineering approach that proved to be very useful in the analysis. Light Gradient Boosting framework that incorporates the feature engineering has proved to be the best approach to tackle the challenge.

Data Exploration, Feature Engineering, and Time Series Analysis
Large Synoptic Survey Telescope (LSST) will have two different survey regions[2]: Deep Drilling Fields (have well determined light-curve points, small errors in flux) and Wide-Fast-Deep survey (covers larger portion of the sky, produces larger flux measurement uncertainties). The time-series flux data produced by LSST will have gaps between observations (makes the analysis much less trivial) since the telescope does not constantly observe one portion of the sky.

There are different types of events that can happen in the cosmos. Some of them are called *transients* (violent deep-sky events, such as supernovae), others are called *variables* (vary in brightness in a periodic way, pulsating stars known as Cepheids). In order to characterise light from cosmological objects we either use 'spectroscopy' (high precision and very accurate, but extremely time-consuming) or 'photometry' (taking an image of an object through different passbands where each passband is responsible for a specific wavelength range). Due to the large volume of data, the measurements produced by LSST will be photometric data.

An extremely important feature of an object is its redshift. Redshift is the rate at which photons arrive to the telescope (compared to the rate of their emission). Objects that are farther away

[1] See figure 1 in appendix (6 passbands, each responsible for its wavelength range)
[2] See figure 2 in appendix

(with higher redshift) from the earth can appear to look fainter[3], however, if we look at how bright an object is as a function of its redshift, we get a more accurate representation of the true brightness. Based on the above analysis of redshift, a feature called absolute magnitude[4] was applied to the raw time-series flux data in order to improve the accuracy of the classification. For each object in the training and test sets, a difference between the maximum and minimum flux values was generated (to make sure it is a non-negative value, before applying log transformation according to the absolute magnitude formula). Then the absolute magnitude variable was generated. It allows us to hypothetically place all cosmological objects at a standard scaled distance from the telescope and directly compare the simple functions of their flux values.

Another relevant piece of information regarding an object is its 'MWEBV' (larger values of MWEBV mean more Milky Way dust along the line from the telescope to the object). The dust tends to make the objects appear redder than they actually are (similar to the effect of a large redshift). Not to mention, each object has its coordinates that identify its location in the universe. There are two ways to measure the coordinates (sky coordinates and galactic coordinates), both of them do not really help in classifying objects since objects of different classes tend to be randomly scattered in the universe (there are no specific patterns regarding their coordinates[5]).

The light curve tables[6] include information regarding the flux of an object (measure of brightness corrected for MWEBV), flux error (uncertainty of the flux measurement), passband (photometric filters, 6 different types), detected (if the brightness is significantly different at the 3σ level relative to the reference template). An extremely useful feature has been produced by calculating the difference between the maximum and minimum values of mjd (measure of time[7]) given the observations were detected, this acted as a characteristic that can separate a "one event" object such as supernovae from "cyclic events" such as Cepheids.

During the feature engineering stage of the analysis, the Lomb Scargle algorithm was used to produce periodograms[8] of the light-curve time series. Then the best (high-powered) periods and their respective powers were extracted to be used as new features in classification. However, these features did not prove to be very useful. This might be because, in general, extragalactic objects tend to be non-periodic, whereas galactic ones are, and in order to

[3] See figure 1 in appendix (the same supernovae event looks different depending on its redshift (z)
[4] Absolute Magnitude = -2.5log(Flux) - Distance Modulus, where Distance Modulus is a function of redshift computed using knowledge of General Relativity
[5] See figure 3 in appendix (objects of different types have a scattered pattern in their sky coordinates)
[6] A few data caveats: data gaps, galactic vs. extragalactic, negative flux
[7] MJD is the time in Modified Julian Date, defined to have starting point of midnight of November 17, 1858
[8] Periodogram is used to identify the dominant periods (or frequencies) of time-series data

separate those we can just use the value of a redshift (0 implies a galactic object). Not to mention, it might be the case that period data can be interacted with other features to become useful, however, after several attempts such features were not established.

The tsfresh library's extract_features method of the feature_extraction module was used to calculate the fourier coefficients of the one-dimensional discrete Fourier Transform for the time series light curve data by passband. Discrete Fourier Transform (DFT) of the series provides a 'compact' representation of the data (given its gaps and sparsity). DFT transforms a time series into a constant and a series of sines and cosines, after which the coefficients of the sine and cosine terms of the resulting approximation can be used in the classification. Furthermore, a technique called low pass filtering could also be applied (was not implemented in code) to only capture a certain number of coefficients in order to drop noise from the data and allow for less overfitting.

There are two possible data augmentation approaches that can be used to expand a time-series dataset. These techniques are called window slicing (data is sliced into substructures that are given the same target as a parent time-series) and window warping (speeding up or down a given time-series at a randomly selected location). In the analysis, window warping with different scaling factors (to stretch the series) was applied to generate new observations. However, the approach did not prove to be useful when using a Convolutional Neural Network, the reason might be that the time-series data had too many gaps and was of different lengths.

Machine Learning Models: Artificial Neural Networks and Deep Learning

- Kaggle username: cs3421610375; displayname: cs3421610375 (or Artur Begyan)
- Best ranking result: loss of 1.080, overall ranking at the time of writing: 320th place

For Task 4, the raw light curve time-series data was converted into a simple features dataframe that includes the mean, maximum, minimum and standard deviation for the flux for each object and passband. That data was then merged with the metadata of each object[9]. The first model to fit the data was a Random Forest (RF) Classifier. After performing cross validation, the decision was made to use One-Vs-The-Rest classifier to split the task of multiclass classification into a set of binary classifiers. This approach yielded better CV scores. Since the dataset of simple features merged with the metadata is rather large, both GridSearchCV and RandomizedSearchCV were not feasible. Therefore, a method of Bayesian hyperparameter

[9] After merging, the data was normalised even though RF is invariant to monotonic transformations of individual features (did not affect analysis/results in any way)

tuning[10], where the results of past evaluations are kept and a surrogate probability of the objective function is maintained and updated, was used. This approach yielded 10,000 estimators with a max depth of 7 for the RF classifier. Not to mention, a decision was made to use a calibrator[11] for the classifier in order for the predicted probabilities of the RF to match the expected distribution of probabilities for each class.

The Multi-Layer Perceptron (MLP Classifier) was then used to make classifications using the same simple features passed earlier to the RF classifier. The MLP Classifier was inspired by the "Simple Neural Net for Time Series Classification" kernel[12]. Before passing the data to the MLP, the features were normalized because an ANN does a bad job when the features have very different scales since it might find correlations between features that have a large scale, during the learning process. The vector of target classes for each object was converted into a one-hot encoding form (matrix with a boolean for each class value) using the to_categorical method from keras.utils in order to match the dimensions accepted by the Keras MLP.

The MLP has a dropout rate of 0.25, it forces the ANN to learn more robust and useful features by ignoring 25% of the neurons during each backward and forward pass. Dropout tends to reduce overfitting since in a fully connected network neurons tend to develop codependency amongst each other during training. Rectified linear unit (ReLU) was used as the activation function in order to reduce the likelihood of a vanishing gradient, also the constant gradient of ReLU results in faster learning (critical for the challenge since we deal with vast amounts of data, especially in the test set). The model has one input layer, three hidden Dense (fully connected) layers and one output layer that uses a softmax activation function to ensure that the output values are within a range from 0 to 1 to be used as predicted probabilities. A Batch Normalisation layer was applied after each Dense layer (except for the output layer) in order to normalise the activations applied by previous layers of each batch. It reduces the amount by which the hidden unit values shift (covariance shift) and speeds up the training time.

After all the layers and hyperparameters of the MLP were defined[13], it was the time for its complication, fitting and test set predictions. Adam[14] (adaptive moment estimation) was chosen to be the optimisation algorithm (extension to stochastic gradient descent). The method calculates individual learning rates for different parameters using the estimates of first and

[10] A surrogate probability model of the objective function is built and iteratively updated once new info is incorporated during the hyperparameter tuning and evaluations of the true objective function
[11] Calibration is a scaling operation that was applied after the predictions were made by the RF model, a simple Platt Scaling was used to calibrate the probabilities
[12] https://www.kaggle.com/meaninglesslives/simple-neural-net-for-time-series-classification
[13] See figure 4 in the appendix
[14]Computationally more efficient, require little memory, well suited for large data/parameter problems, requires little tuning

second moments of the gradients. The multi-weight log-loss function[15] was used; the fitting of the model consisted of 600 epochs[16] and was done by batches[17] of size 100. Bayesian hyperparameter optimisation was applied to determine the dropout and the activation function for the MLP. Techniques such as learning rate decay schedule (to tune we need to examine the cost to check if it is dropping faster) and early stopping (stop the training if the validation errors are increasing persistently) could be used to potentially improve the model.

Both when simple features were used and when features produced during the feature engineering stage were used, the MLP turned out to be a better model for classification. Data augmentation applied earlier (using the window warping approach) did not improve the predictions in any setting. Random forest tends to work worse if there are rare outcomes (outliers or novelties that exist in the test set). On the other hand, neural networks tend to work really well with large amounts of data, however the computational cost is higher for an ANN.

At the final stage of analysis a Convolutional Neural Network[18] (ConvNet) was implemented to classify the cosmological objects from the challenge following the example from the kernel[19]. The data augmentation approach did not turn out to be useful, therefore model fitting was done using the original series. During the preprocessing stage the flux and flux error data was scaled using the StandardScaler of sklearn.preprocessing. Then, the train time series data was transformed into a 2D image (for each object_id) with the following dimensions: [number of passbands, length of flux + length of flux error + length of detected variable). After the transformed time series data was merged with the meta train data, and the one-hot encoding were applied to the targets, it was time to build, compile and fit the ConvNet.

There are a few nuances that make the ConvNet different from the MLP analysed above (focusing on the differences in this paragraph). When building the model, 1D layers were used because we are not dealing with images and 1D layers work better when the location of a feature within a portion of the dataframe does not convey much significance (applies well to time series analysis). The first conv layer has a kernel size of 80 (40 for second conv layer), meaning that it will apply 80 different filters and extract 80 features from the input layer, the size of the strides was chosen to be 4 (1 for second conv layer), meaning that the filter will take a step of size 4 (produces smaller output volumes spatially) when it moves along the input data. “Same” padding was used for the first layer to try to evenly pad the left and right of the input

[15] https://www.kaggle.com/c/PLAsTiCC-2018/discussion/69795
[16] An epoch is when an entire dataset is passed both forward and backward through a neural network once (the larger the number, the more likely the model is to overfit)
[17] A batch size is the total number of training observations that are present in a single batch
[18] See figure 5 in the appendix
[19] https://www.kaggle.com/higepon/updated-keras-cnn-use-time-series-data-as-is

data with zeros to make the width of the input a multiple of the filter size. A Max pooling layer (with a pool size of 4) was applied after each convolutional layer to decrease the feature map size while keeping the significant information and preventing overfitting.

Progression Graph[20] & Discussion

The lowest score on the progression graph is based on a naive approach of classifying objects based on a single variable: the boolean for being a galactic object (redshift == 0), this approach yielded a multiclass log-loss of 2.158. CNN approach was an improvement, however the log-loss value was still relatively high, this could be because the dataset is too small for a CNN to be able to extract robust features, moreover, given the sparse nature of time series data and its differences in length, data augmentation did not fix the issue. A simple neural network that used simple functions of the light curve data alongside the metadata resulted in a noticeable improvement of the log-loss score (1.368). However, after conducting the necessary feature engineering outlined above, there has been further progression in the accuracy of results (log-loss score of 1.135). Finally, the best score was achieved using an LGBM[21] classifier that used previously engineered features (hyperparameters were truncated from the best-performer, calculated and selected by Bayesian optimisation).

In the future a Recurrent Neural Network (RNN)[22] could be used to potentially achieve better results because RNNs tend to deal with sequence problems (e.g. time-series data), i.e. state can be retained from one iteration to the other. Not to mention, a novelty detection technique such as Local Outlier Factor[23] could be used to more accurately predict the 99th class in the test set. More domain related features need to be designed in order to distinguish between “hard” classes (42, 90, 67, 95, 62, 15, and 52). Furthermore, separate classifiers could potentially be designed for galactic and extragalactic objects due to their fundamental differences (e.g. galactic objects tend to be more periodic). Another interesting approach could be to use a ConvNet to extract features from the training data and then feed these features into a classifier such as the LGBM that proved to be the best for the challenge data.

[20] See figure 6 in the appendix (the progression graph)
[21] Light Gradient Boosting framework that uses a tree based learning algorithm (grows the tree in a vertical manner, while others focus on horizontal growth)
[22] Long Short Term Memory networks
[23] Calculates the score of abnormality of novel data → the ratio of the average local density of the K nearest neighbours to the local density of a given observation

Appendix

Figure 1[24]

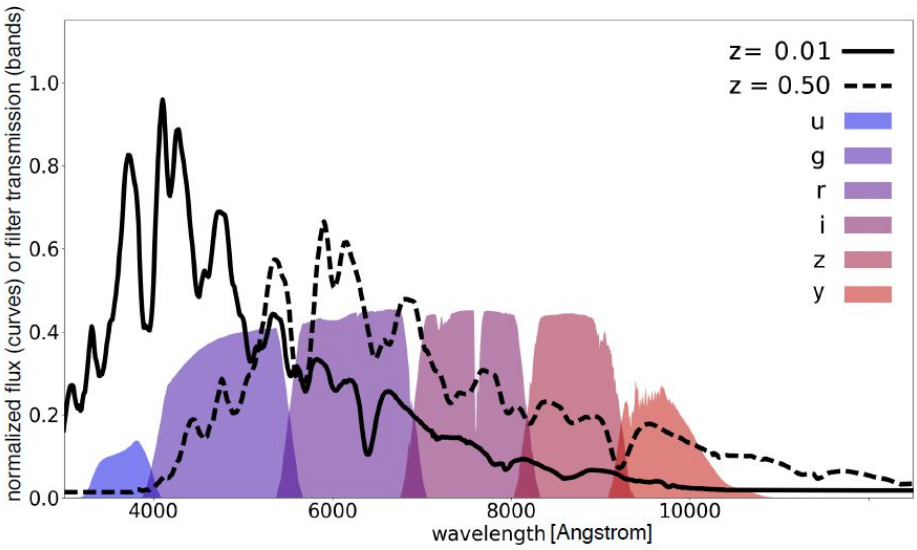
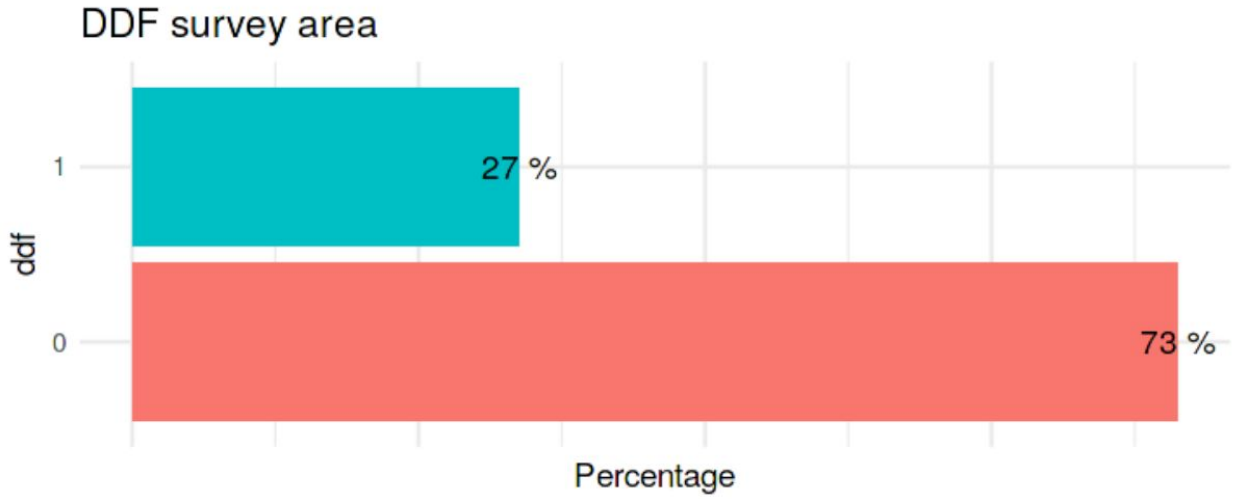
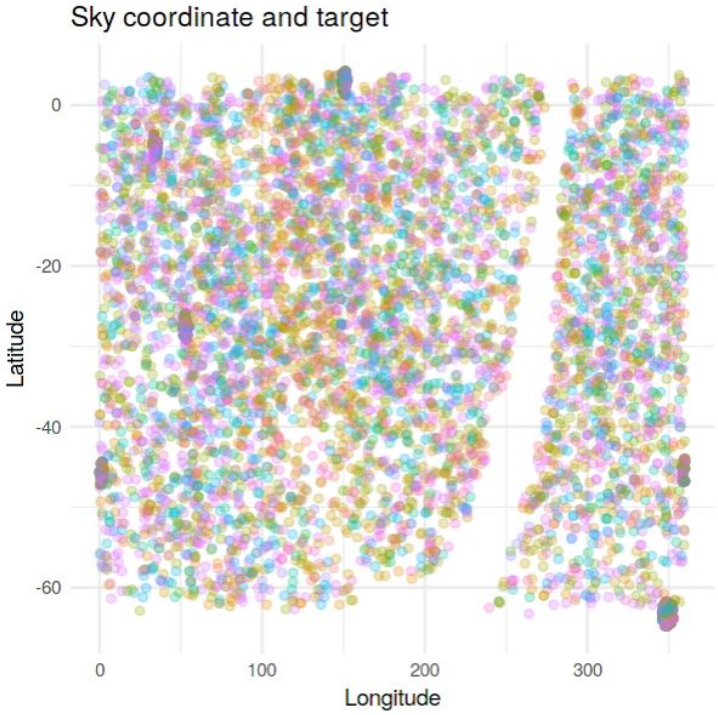


Figure 2[25]



[24] Taken from: https://arxiv.org/abs/1810.00001
[25] Taken from: https://www.kaggle.com/danilodiogo/the-astronomical-complete-eda-plasticc-dataset

Figure 3[26]



[26] Taken from: https://www.kaggle.com/danilodiogo/the-astronomical-complete-eda-plasticc-dataset

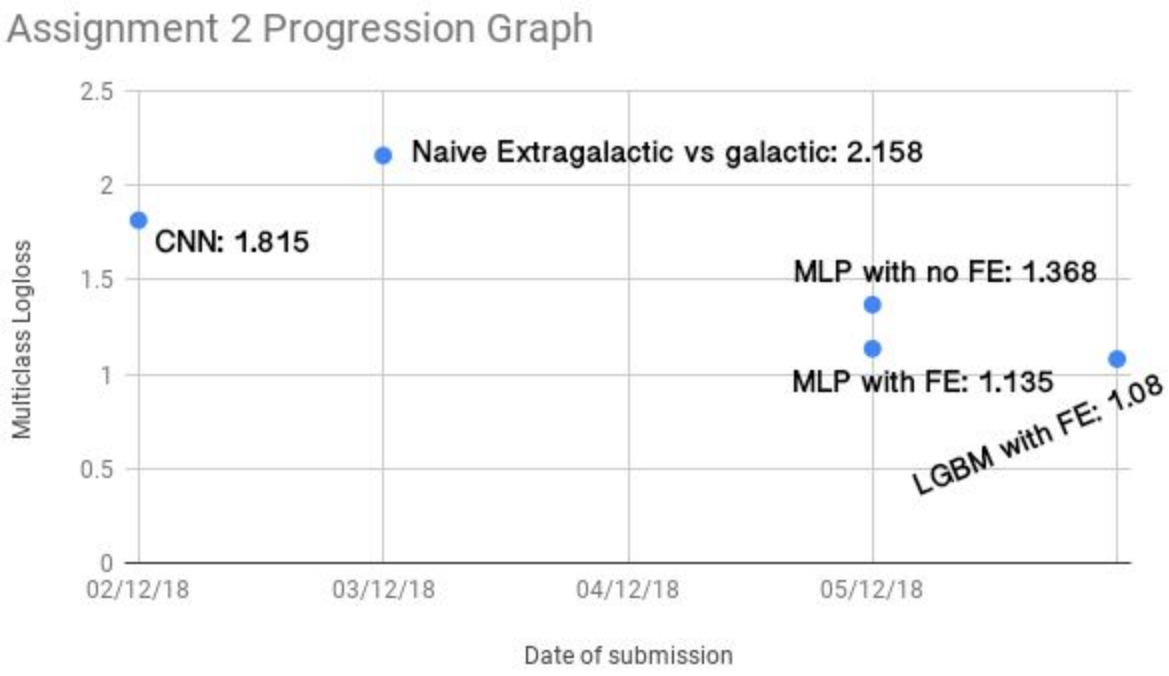
Figure 4

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 512)               16384
_________________________________________________________________
batch_normalization_1 (Batch (None, 512)               2048
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328
_________________________________________________________________
batch_normalization_2 (Batch (None, 256)               1024
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 128)               32896
_________________________________________________________________
batch_normalization_3 (Batch (None, 128)               512
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_4 (Dense)              (None, 64)                8256
_________________________________________________________________
batch_normalization_4 (Batch (None, 64)                256
_________________________________________________________________
dropout_4 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_5 (Dense)              (None, 14)                910
=================================================================
Total params: 193,614
Trainable params: 191,694
Non-trainable params: 1,920
_________________________________________________________________
```

FIgure 5

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input0 (InputLayer)          (None, 216, 6)            0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 54, 256)           123136
_________________________________________________________________
batch_normalization_1 (Batch (None, 54, 256)           1024
_________________________________________________________________
activation_1 (Activation)    (None, 54, 256)           0
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 13, 256)           0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 13, 256)           196864
_________________________________________________________________
batch_normalization_2 (Batch (None, 13, 256)           1024
_________________________________________________________________
activation_2 (Activation)    (None, 13, 256)           0
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 3, 256)            0
_________________________________________________________________
lambda_1 (Lambda)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 14)                3598
=================================================================
Total params: 325,646
Trainable params: 324,622
Non-trainable params: 1,024
```

Figure 6 (progression graph and data that the progression graph is based on)



Date	Multiclass log-loss	Model + FE Used
02/12/18	1.815	CNN with no data augmentation
03/12/18	2.158	Naive “Galactic vs Extragalactic” classifier
05/12/18	1.368	Simple Neural Net with no feature engineering (using simple functions of the light curve flux data)
05/12/18	1.135	Neural Net with feature engineering (outlined in the FE section of the report)
06/12/18	1.08	LGBM + FE (Light Gradient Boosting framework that uses the same features outlined in the feature engineering section of the report)