CS342 Machine Learning Assignment 2 Report
Kaggle ID: cs342u1500212

Abstract

The LSST telescope is an important tool utilised to decipher the domain of our universe. It captures a glimpse of astronomical phenomenons, such as light sources and dark energy. Whilst it is impossible cover every aspect of the LSST, this report aims to handpick certain things to enrich our understanding of it.

The methods used for this Kaggle competition can be summarised in three main steps. In the first step, we focus on an in-depth understanding of our classes, exploring concepts such as time series flux/passbands, Redshift and Galactic/Extragalactic objects. Next, we generate new features based on different shapes of its time series, depending on whether an object is periodic or a burst. Finally, five different models were developed (sklearn_RF, sklearn_MLP, keras_MLP, keras_CNN, light_gbm). These were tuned methodologically using graphs and a forward/backward stepwise selection approach, followed by an analysis on the impact of feature engineering. Scores and submissions were documented accordingly throughout the process. The best model developed scored a weighted multi-log loss of 1.052 on the leaderboard, equivalent to approx. 131st place on Kaggle or Top 14% in the world.

1. Data Exploration

1.1 Time Series

With a total of 15 different classes, the basic building block for identifying light sources is a time series of flux and flux error based on 6 different passbands. These passbands are given by:

$$\{u \rightarrow 0,\ g \rightarrow 1,\ \ r \rightarrow 2,\ i \rightarrow 3,\ z \rightarrow 4,\ y \rightarrow 5\}$$

Of which are measured over a Modified Julian Date (MJD) unit for time. The LSST takes a picture of one region of the sky for 15-30 seconds followed by a gap of 30 minutes, hence resulting in three different patches illustrated in the Figure 1. Flux is a measure of brightness and its error has a 68% confidence interval. Further, objects of the 15 classes are either constituted as a "burst" or "cyclic". Burst objects have a peak values/decay rate whilst cyclic objects are concerned with periodicity. This is a key distinction that gives us a clue on how to convert the time series into features.

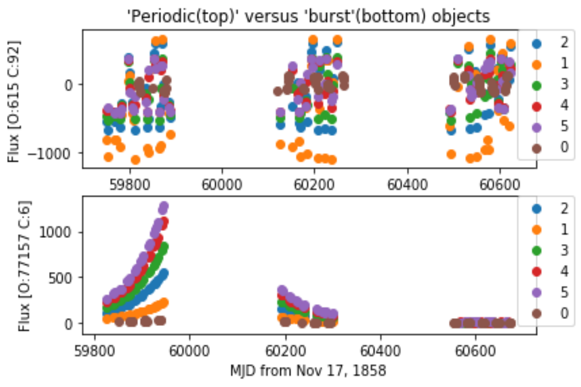


Figure 1: Comparing a 'cyclic' and 'burst' object

1.2 Redshift

In addition to the time-series data, its metadata contains a plethora of useful information. Particularly, redshift is a concept worth paying strong attention to. Photometric redshift of the host galaxy, known as 'hostgal_photoz', is a measure of how fast a distance source is moving relatively to the earth. This has a one to one correlation with another important feature known as 'distmod', which measures the brightness of the flux. The plot in Figure 2 illustrates the relationship between redshift/error and dismod. We can observe how the error associated with photometric redshift has an increasing but rather spurious trend with distmod.

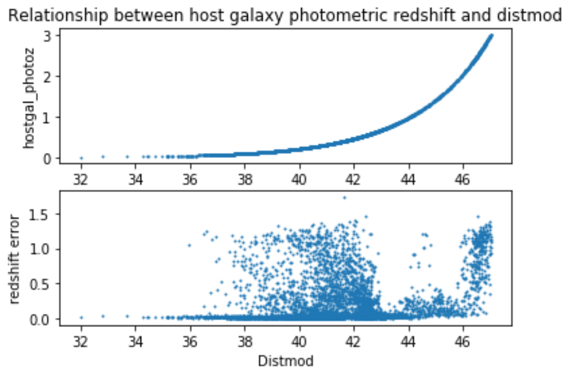


Figure 2: Relationship between Distmod vs Redshift

The main takeaway from this entails such that different classes are brighter or fainter within the flux/redshift space; hence a combination of flux values and hostgal_photoz is distinctive. Distmod is not particularly useful when used alone because it is already represented by hostgal_photoz, but can still be used as a function to calculate other features. Another feature known as spectrometric redshift or 'hostgal_specz' can also be used to classify extragalactic objects better; however this is mostly unavailable in the test data given.

1.3 Galactic vs Extragalactic

Generally, objects have a binary classification between galactic and extragalactic based on its redshift values. A zero valued hostgal_photoz implies it is galactic and vice versa. Galactic objects are known to be “ cyclic” while extragalactic objects are “bursts”. This leads us to another useful hint: after we categorize objects as galactic/extragalactic, we may extract more information about the shape and relative strengths of its flux and passbands.

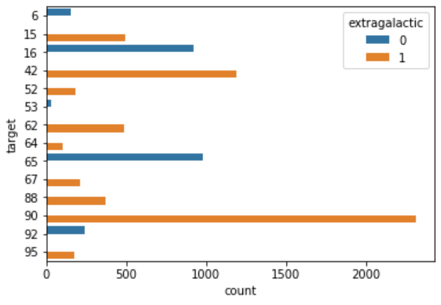


Figure 3: Extragalactic (1) vs Galactic (0)

To visualize how our training set looks like, Figure 3 illustrates the distribution of the training classes based on galactic/extragalactic. A nonuniform distribution of classes is shown, for example, class 53 has less than 100 rows whereas class 99 consists of 2000+ rows

1.4 Coordinates

As a side note, Figure 4 shows the coordinate of objects provided by the metadata, which is densely concentrated in areas known as the “DDF”. These coordinates do not provide useful information but is worth being aware of.

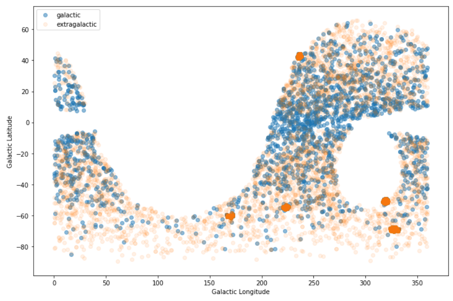


Figure 4: Locations based on galactic/extragalactic

2. Feature Engineering

2.1 Method I

To capture periodicities of “cyclic” objects outlined in our data exploration section, we compute the Fourier Transform coefficients for each object flux per passband. This stems from the fact that any time series can be represented as a sum of different cos and sin curves. More formally, we can define any time series as in (Eq1). In our case, we apply a discrete fourier transform version shown in (Eq2) to our dataset using an external library ‘tsfresh’ to find the real and imaginary values of these coefficients. Kurtosis and skew is also calculated to indicate symmetry of some underlying probability density function of the amplitude of flux.

$$(Eq1)\ x(t) = A_0 + \sum_{n=1}^{\infty} A_n\, cos(\omega nt) + \sum_{n=1}^{\infty} B_n\, sin(\omega nt)$$

$$(Eq2)\ A_k = \sum_{m=0}^{n-1} a_m\, exp\,\{-2\pi i(mk/n)\}\ ,\ s.t.\ k = 0, ..., n-1$$

Overall, this method is somewhat a dimensionality reduction technique. It can be used to reconstruct an approximation of a periodic cycle since it converts the time series into its frequency domain.

2.2 Method II

As mentioned in the lecture by Dr. David Armstrong, the brightness of astronomical objects varies over scale, hence we need a way to work under its magnitude. This can be calculated using the equation in (Eq3). Further, absolute magnitudes can also be calculated by taking the magnitude minus the distmod of the object. Instead of calculating the abs magnitude for all magnitudes, we limit the scope of absolute magnitudes to extragalactic objects only. This is primarily to help find its peak values to identify “burst” objects better. To do this, we apply the equation (Eq4) on the minimum magnitudes to find its peak, due to the flipped y-axis log transformations.

A new version of distmod called ‘distmod_lambdacdm’ is calculated using the astropy library. This is because the distmod given by the dataset is approximated and tends to be noisy, as mentioned by Kyle Boone (2nd place).

$$(Eq3)\ Magnitude = -2.5\ log\,(Flux)$$

$$(Eq4)\ Peak\ Abs.\ Magnitude = Min.\ Magnitude - Distmod$$

2.3 Method III

To further separate burst objects such from cyclic objects, a method to identify sudden increments of intensity is highly desirable. This can be achieved by extracting time series values where ‘detected’ equals one, which basically implies its brightness is significantly different than usual. We find the time difference of this, named ‘mjd_diff_det’, for the detected time series values to find out how long a “burst” sustains or its rate of decay. This helps separate single event objects such as ‘supernovae’ from cyclic events such as ‘cephoids’, as we expect different extragalactic objects to have different “bursts” sizes or decay rates.

Algorithm :

```
1. $For\ all\ objects$ :
        1.1 $Extract\ rows\ of\ time\ where\ detected\ = 1.$
        1.2 $Find\ its\ mjd\ max\ and\ mjd\ min$
        1.3 $Set\ mjd\_diff\_det\ := mjd\ max\ -\ mjd\ min$
2 $Merge\ new\ data\ frame\ with\ metadata\ on\ object\ id$
```

3. Data Augmentation

Window warping and window slicing were implemented to simulate new samples for the training set. For each object in the training set, a random 20% chunk of time was chosen. This chunk is then either dropped or 'squeezed' using a custom minMaxScaler() function and appended into two new training sets, stored locally. The end result of this method produced an additional approx. 15696 rows of data. This new data is used by importing, preprocessing and merging it with the original training data.

4. Random Forests and MLP Classifier

As a starting point, simple functions of raw time series values were used to train three different models: one Random Forest model and two Multilayer Perceptron models from sklearn and keras. The simple functions of raw time series include aggregates such as the min, max, median, std, skew, size on values flux, flux magnitude, flux error, flux ratio and flux by flux ratio.

A stratified cross-fold validation approach was adopted to tune the models, which basically rearranges the data to ensure each fold has a good representation of all classes. Only 14 classes are available in the test set, hence motivating the need for novelty detection. In particular, a simple method taken by olivier (34th place) was adopted to predict the missing class 99, which is a product of probabilities based on the other 14 classes outlined below.

$$class99\ = \prod_{k=1}^{14} \mathbb{P}(1 - another\ class\ k)$$

$$\mathbb{P}\,(class99)\ = 0.18\ \times\ class99 \div normalizing\ constant$$

4.1 sklearn RF

The Random Forest model was tuned primarily using graphs on two parameters, n_estimators and max_depth. By isolating other parameters and varying these two, this allows us to find a range of parameters to find the point of overfitting and underfitting. Next, a grid search parameter search technique was used based on the graphs in Figure 5, around values n_estimators > 80 and max_depth between 5 and 15. This is a necessary step as different combinations of parameters produce different results. After choosing its parameters of (n_estimator=100, max_depth=7), the leaderboard submission score for this model was 1.932.

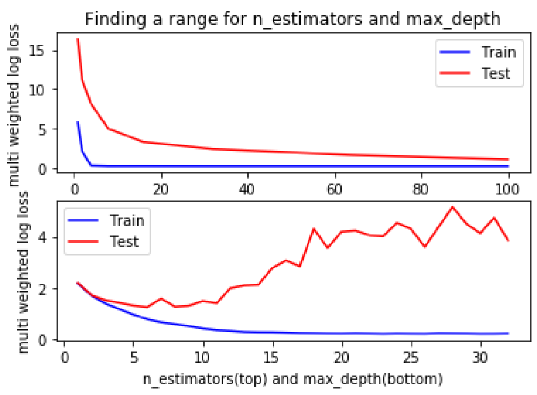


Figure 5: Finding the optimal range for grid search.

4.2 sklearn MLP

Next, the Multilayer Perceptron by sklearn was tuned using a forward/backward stepwise selection approach. The Rectified Linear activation function and Adam optimizer worked best for this model although improvements were only marginal. The Hidden Layers seemed to have a bigger impact; complex models with more than 5 layers or more than 30 input nodes per layer tend to underperform compared to simpler models. Using a simple (20, 20, 20) hidden layer, the leaderboard score for this model was 1.999.

4.3 keras MLP

For comparison purposes, the keras MLP model was implemented based on a public kernel from Siddhartha (9th place). This model performed the best amongst the three models, with a leaderboard score of 1.815. Compared to the sklearn MLP model, this model had a greater complexity of a (512, 256, 128, 64) layer, a dropout rate of 0.25 and batch normalization, hence better performance and stability for neural networks. Additionally, training time for this MLP took substantially longer to converge.

5. Incorporating Feature Engineering

After incorporating feature engineering or FE, the leaderboard scores for all three models have improved significantly. The sklearn RF model improved from 1.932 to 1.679, sklearn MLP from 1.999 to 1.679 and keras MLP from 1.815 to 1.205 as shown in Figure 6. To investigate why our models improved by an average of 0.3-0.4 loss score, we need a form of methodology to identify which classifications are performing well.

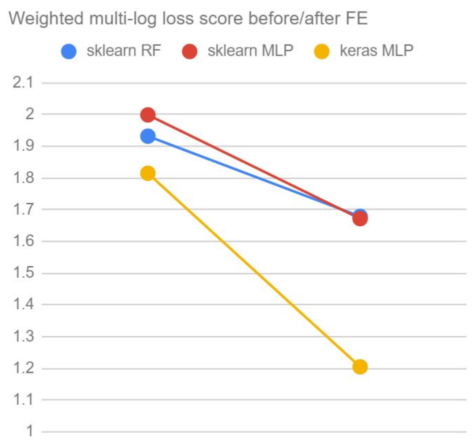
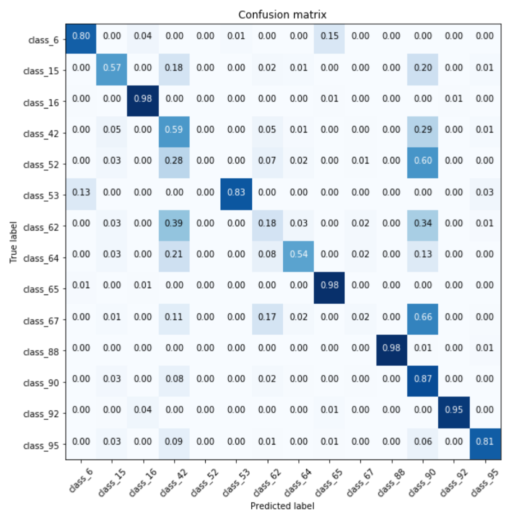


Figure 6: Before and after feature engineering



	class_6	class_15	class_16	class_42	class_52	class_53	class_62	class_64	class_65	class_67	class_88	class_90	class_92	class_95
class_6	0.80	0.00	0.04	0.00	0.00	0.01	0.00	0.00	0.15	0.00	0.00	0.00	0.00	0.00
class_15	0.00	0.57	0.00	0.18	0.00	0.00	0.02	0.01	0.00	0.00	0.00	0.20	0.00	0.01
class_16	0.00	0.00	0.98	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.01	0.00
class_42	0.00	0.05	0.00	0.59	0.00	0.00	0.05	0.01	0.00	0.00	0.00	0.29	0.00	0.01
class_52	0.00	0.03	0.00	0.28	0.00	0.00	0.07	0.02	0.00	0.01	0.00	0.60	0.00	0.00
class_53	0.13	0.00	0.00	0.00	0.00	0.83	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03
class_62	0.00	0.03	0.00	0.39	0.00	0.00	0.18	0.03	0.00	0.02	0.00	0.34	0.00	0.01
class_64	0.00	0.03	0.00	0.21	0.00	0.00	0.08	0.54	0.00	0.02	0.00	0.13	0.00	0.00
class_65	0.01	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.98	0.00	0.00	0.00	0.00	0.00
class_67	0.00	0.01	0.00	0.11	0.00	0.00	0.17	0.02	0.00	0.02	0.00	0.66	0.00	0.00
class_88	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.98	0.01	0.00	0.01
class_90	0.00	0.03	0.00	0.08	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.87	0.00	0.00
class_92	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.95	0.00
class_95	0.00	0.03	0.00	0.09	0.00	0.00	0.01	0.00	0.01	0.00	0.00	0.06	0.00	0.81

Figure 7: Confusion matrix for sklearn RF after FE.

Figure 7 shows the confusion matrix for the sklearn RF model after incorporating feature engineering. To highlight its key points, galactic objects such as classes 16, 65, 88 and 92 have a high true positive score of 0.95+. Contrastingly, some extragalactic objects such as class 52, 62 performed poorly at a dead 0 true positive score, often overlapping with class 90. Two implications can be inferred from this. First, it seems that the Fourier Transform coefficients mentioned in Method I seems to help classify galactic objects. Second, the feature engineering Methods II and III seemed to inconsistently work for some "burst" objects such as class 15 only.

Including FE however, has a significant drawback - it is computationally slow. Any features generated by the training model must also be done in the test set. Overall the predicting time for the test set after FE ranges from 180 - 210 minutes, compared to 50 minutes without FE.

6. Convolutional Neural Network

A CNN was implemented based on a public kernel by (Kaggle ID: higepon). Essentially, the flux, flux error and detected values from each row of the time series are converted into 71 bins for each passband. In total this generated a training set of dimensions (X, 6, 216), where X is the number of objects available from the training set. After data augmentation, there are 23128 objects compared to the initial 7848 which is a 209% increase, hence the final unflattened dimensions for the training data is (23128, 6, 216).

The CNN can be deconstructed into three convolution modules. Each module contains a one dimensional convolution filter of size 256 that extracts features based on each window, a one dimensional max pool filter of size 4 that extracts a max value on each window, batch normalisation for faster convergence and a dropout rate of 0.25 to avoid overfitting. At the end of the modules, there is a fully connected dense layer using the softmax activation function. Figure 8 shows the cross-validation training score of the keras CNN model, with a best CV loss score of 1.69 - which is a significant improvement from the original kernel starting at approx 2.0.

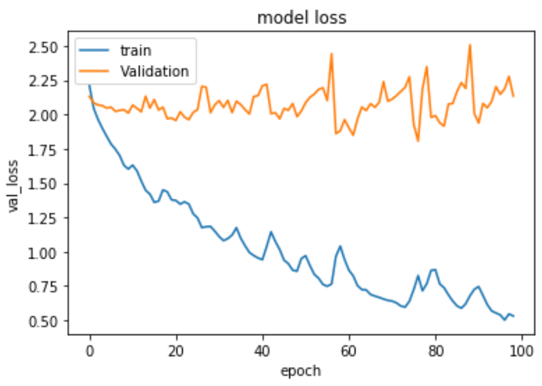


Figure 8: Validation score after training CNN model.

Unfortunately, predictions for the CNN model were not submitted due to difficulties translating the test set into correctly shaped input layers. Regardless, data augmentation was tested on the keras MLP model to test the validity of the new sample data. After adding data augmentation, the keras_MLP (Task 4) improved its loss from 1.815 to 1.369, which shows fairly promising results. However, similar improvements did show up for other models such as sklearn_RF - this implies that Data Augmentation is not as effective as previously thought.

Comparing the CV scores between the keras_CNN and keras_MLP model, the latter performs much better using generated features instead of binarized time series values. This is expected as CNNs do not normally work well with time-series data, because it does not preserve essential dependencies for time series information. Alternatively, a Recurrent Neural Network would have been a more ideal model for this Kaggle challenge.

7. Progression Graph

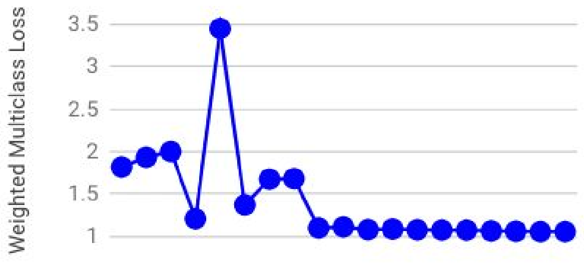


Figure 9: Overall Progression from 25 Nov to 5 Dec.

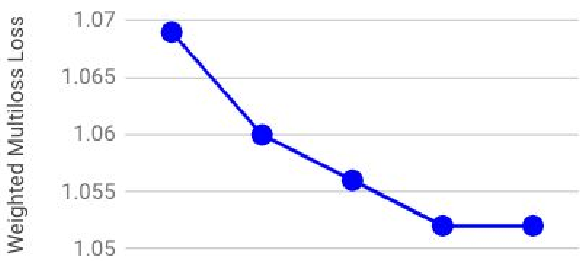


Figure 10: Top 5 most recent submissions.

A full description of every submitted model is available in the appendix. 19 submissions in total were attempted between 25 Nov to 5 Dec 2018. To highlight the main trends of the progression graph in Figure 9: the model initially started slow at 1.815. A peak at 3.456 was due to an syntax error with class 99 predictions. Before the 1st of December, the best score was 1.205 for the keras_MLP model after feature engineering, which was ranked third in the module at that time. From the 9th submission onwards, there is a steady decrease up until the best model developed on the 5th Dec illustrated in Figure 10.

Best Model

The best model in this project exploits a Light GBM gradient boosting framework that employs a tree based learning algorithm. It essentially works by splitting the tree leaf wise instead instead of depth/level wise; hence being faster without compromising accuracy. The best submission for this model was 1.052 (131st place on Kaggle on 6th Dec), using a chain of convoluted methods described in a list format below:

Best Light GBM based model 131 ▲359 cs342u1500212 1.052 19 13h	
1.	3 Feature Engineering approaches
2.	Finding the passband label of the 'max' and 'min' flux values. Some bursts objects have orders between passbands i.e. passband 5 is usually the highest flux value for some objects.
3.	Finding the longest strike above mean for flux to identify decay of burst objects.
4.	Adding additional features such as host_gal_certain and milky_way.
5.	Lowering the depth of the tree to reduce overfitting.
6.	**Advanced:** Outlier detection using Isolation Forest to remove noisy data from extragalactic classes 15, 42 and 90 from training set.
7.	**Advanced:** Predicting hostgal_specz on the test set by extracting 120,000 of the available hostgal_specz in the test set to use as training data.

Figure 11 shows a confusion matrix for this model. Mainly, we observe that the true positive scores for extragalactic objects such as 15, 42, 52, 62, 67 and 90 have improved tremendously compared to the previous confusion matrix in Figure 7, i.e. the true positive score for class 67 improved from 0 to 0.55.

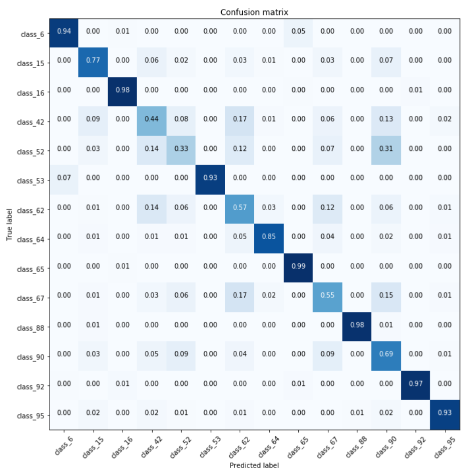


Figure 11: Confusion matrix for Best Model

Final Words

In retrospect, feature engineering seems to have a much greater effect than data augmentation in this challenge. Galactic objects are classified well, although a top 100 submission would have classified extragalactic objects even better. Future directions would include more ways to encode "burst" objects, such as finding the Dynamic Time Warping distance between each passband. However, such an implementation would have taken days to predict, as computation time is heavy on the test set. Interestingly, computation time was a major consideration for every strategy used in this project.

References

https://www.kaggle.com/c/PLAsTiCC-2018/kernels

https://www.kaggle.com/c/PLAsTiCC-2018/discussion

https://www.kaggle.com/michaelapers/the-plasticc-astronomy-starter-kit, Kaggle Starter Kit.

https://warwick.ac.uk/fac/sci/dcs/teaching/material/cs342/plasticc_compress.pptx, Dr. David Armstrong powerpoint slides.

https://www.kaggle.com/iprapas/ideas-from-kernels-and-discussion-lb-1-135, feature engineering method 1 on Fourier Coefficients.

https://www.kaggle.com/c/PLAsTiCC-2018/discussion/71871#423698, feature engineering method II on absolute magnitudes.

https://www.kaggle.com/c/PLAsTiCC-2018/discussion/69696#410538, feature engineering method III on mjd_diff_det.

https://aaltd16.irisa.fr/files/2016/08/AALTD16_paper_9.pdf, reference for window warping/slicing.

https://www.kaggle.com/meaninglesslives/simple-neural-net-for-time-series-classification, MLP reference.

https://www.kaggle.com/higepon/updated-keras-cnn-use-time-series-data-as-is, CNN template.

https://tsfresh.readthedocs.io/en/latest/ , tsfresh library.

http://docs.astropy.org/en/stable/api/astropy.cosmology.FlatLambdaCDM.html, astropy library.

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html, outlier detection using Isolation Forest.

https://www.kaggle.com/cttsai/forked-lgbm-w-ideas-from-kernels-and-discuss, reference for LGBM and other useful features.

Appendix

Date	Score	Description
25 Nov	1.815	keras MLP + Functions of time series
28 Nov	1.932	sklearn RF + Functions of time series
29 Nov	1.999	sklearn MLP + Functions of time series
30 Nov	1.205	keras MLP + Feature Engineering
1 Dec	3.456	keras MLP + Failed anomaly detection
2 Dec	1.369	keras MLP + Functions of times series + Data Augmentation
2 Dec	1.672	sklearn MLP + Feature Engineering
2 Dec	1.679	sklearn RF + Feature Engineering
2 Dec	1.094	Light GBM + Feature Engineering + More features
3 Dec	1.107	Light GBM + Feature Engineering + More features + Data augmentation
3 Dec	1.075	Light GBM + Feature Engineering + More features + Passband feature
3 Dec	1.082	Keras MLP + Feature Engineering + More features
3 Dec	1.073	Light GBM + Feature Engineering + More features + Passband feature
4 Dec	1.07	Best submission + Better Tuning
4 Dec	1.069	Best submission + Better Tuning
5 Dec	1.06	Light GBM + Feature Engineering + More features + Passband feature
5 Dec	1.056	Best submission + predicted hostgal_specz
5 Dec	1.052	Best submission + predicted hostgal_specz + higher class99 values
5 Dec	1.052	Best submission + predicted hostgal_specz + higher class99 values