Overview of the 1st Place Solution to the Kaggle PLAsTiCC Astronomical Classification challenge

Kyle Boone

January 12, 2019

A MODEL SUMMARY

A.1 Background on you/your team

Competition Name: PLAsTiCC Astronomical Classification challenge
Team Name: Kyle Boone
Private Leaderboard Score: 0.68503
Public Leaderboard Score: 0.67056

Name: Kyle Boone
Location: Albany, California, USA
Email: kboone@berkeley.edu

A.2 Background on you/your team

What is your academic/professional background?
I am currently a Ph.D. candidate in the physics department at the University of California, Berkeley. I previously earned a B.A.Sc. in Engineering Physics from the University of British Columbia in Vancouver, BC and a M.A. in Physics from the University of California, Berkeley. My research is on improving cosmological distance measurements with Type Ia supernovae under the supervision of Saul Perlmutter.

Did you have any prior experience that helped you succeed in this competition?
My research involves building models of Type Ia supernovae, so I am very familiar with the challenges in this competition. I did not work on classification of transients before this challenge, but I did have lots of experience with modeling time series with Gaussian Processes which proved to be very beneficial to this challenge.

What made you decide to enter this competition?
I originally entered this competition to teach myself more about machine learning. For my research, I had previously used various basic machine learning algorithms, but I had a lot to learn about state-of-the-art classification methods and how to tune models. I was interested in trying to apply insights that professional data scientists were posting on Kaggle in the kernels and discussions to my research.

How much time did you spend on the competition?
I spent a lot of time thinking about the challenge when it first came out, trying different ideas and learning

more about machine learning. At that point, I was mainly focused on teaching myself and seeing how I could apply the techniques that data scientists were sharing on the Kaggle platform to my other research problems, so I wasn't really directly working on the competition. I started seriously competing around the start of November, and ramped up my contributions from there. I spent most of my weekends in November and December coding for this challenge, and was by late November I was basically devoting all of my spare time to the challenge.

If part of a team, how did you decide to team up?
I did not join a team. I am on the job market, and thought that a solo finish in the top 10 would look better for me than one where I joined up with professional data scientists.

If you competed as part of a team, who did what?
I did not join a team.

A.3 Summary

My solution involves several steps. First, I augment the training dataset to make it more representative of the test dataset. I then fit Gaussian processes to each lightcurve in the sample. From the Gaussian process predictions, I calculate a set of 188 features that I designed to capture the shape of the lightcurve, the color of the object and information about where there is and isn't enough data to constrain the lightcurve. These features are then fed into a LightGBM classifier that I train to perform the classification.

The time to train the model and do classifications is entirely limited by the Gaussian process fits and predictions. On my machine (an Intel Xeon E3-1270), I can fit approximately 10 lightcurves per second which implies that I need $\sim$100 hours of time on one of these machines to do all of the fits for the training and test sets. This task can easily be parallelized because every fit is independent. All of the other steps in the solution are relatively quick and complete in under an hour.

A.4 Features Selection / Engineering

As a researcher of Type Ia supernovae, my main focus in this competition was to classify the various kinds of supernovae and transients with similar lightcurves. That turned out to be the main challenge for this competition. I added some additional features to help distinguish between the non-supernova like classes, but they were all relatively easy to classify by comparison. Unless specified otherwise, the features were all calculated on the Gaussian process prediction rather than on the raw data. The features that I used in my final model are:

- `hostgal_photoz`: Host galaxy photo-z
- `hostgal_photoz_err`: Host galaxy photo-z error
- `distmod`: Distance modulus (degenerate with photo-z, but I found a very slight improvement if I included both).
- `count`: Number of observations.
- `gp_fit_[id]`: Gaussian process fit parameters (kernel length scale and amplitude).
- `max_flux_[band]`: Maximum flux in each band.
- `max_dt_[band]`: Time of maximum flux relative to the median time for each band.
- `min_flux_[band]`: Minimum flux in each band.

- `positive_width_[band]`: Integral of the positive fluxes of the lightcurve divided by the maximum flux for each band. This gives a measure of the "width" in time of the lightcurve that can handle lots of different kinds of objects.
- `negative_width_[band]`: Same as above, but for the negative fluxes in the lightcurve.
- `frac_time_[direction]_[fraction]_[band]`: Before or after maximum flux, how many days does the lightcurve stay above a specific fraction of the maximum flux. The fractions used were 0.2, 0.5 and 0.8.
- `count_s2n_[threshold]`: How many data points had a signal-to-noise above a threshold. Thresholds of 3, 5, 10 and 20 were used for positive and negative fluxes separately.
- `frac_background`: Fraction of data points that have a signal-to-noise below 3.
- `total_s2n_[band]`: Total signal-to-noise in each band.
- `time_width_s2n_[threshold]`: Time delay between the first and last flux above a given signal-to-noise threshold.
- `count_max_[label]`: Count of how many data points fall in some bin in time. eg: `count_max_center` counts the number of points within 5 days of maximum light, `count_rise_20` counts the number of points between 5 to 20 days before maximum light, and so on. These features are essential for the classifier to understand when the GP fit is not able to constrain the rise or fall times of a lightcurve. The bins extend out to 800 days away from maximum to be able to capture the rough shape of non-supernova like objects.
- `mean_[label]`: Same as above, but the mean of all fluxes in the bin divided by the maximum flux of the object.
- `std_[label]`: Same as above, but the standard deviation of all fluxes in the bin divided by the maximum flux of the object.
- `peaks_[sign]_[band]_count`: Apply a peak detection algorithm to each band, and count the number of peaks found in both the positive and negative directions.
- `peaks_[sign]_[band]_frac_[index]`: Fractional height of the peaks after the first. This is calculated for the 2nd and 3rd peaks.

There are many large redundancies between features. The Gaussian process is relatively smooth across the different bands, and hence the various features that are calculated for the different bands are very highly correlated with each other. While many of these features are somewhat redundant, the LightGBM algorithm is designed to be able to handle correlated features so we did not do any preprocessing to remove redundant features. The model can likely be simplified by removing many of these redundancies in future work.

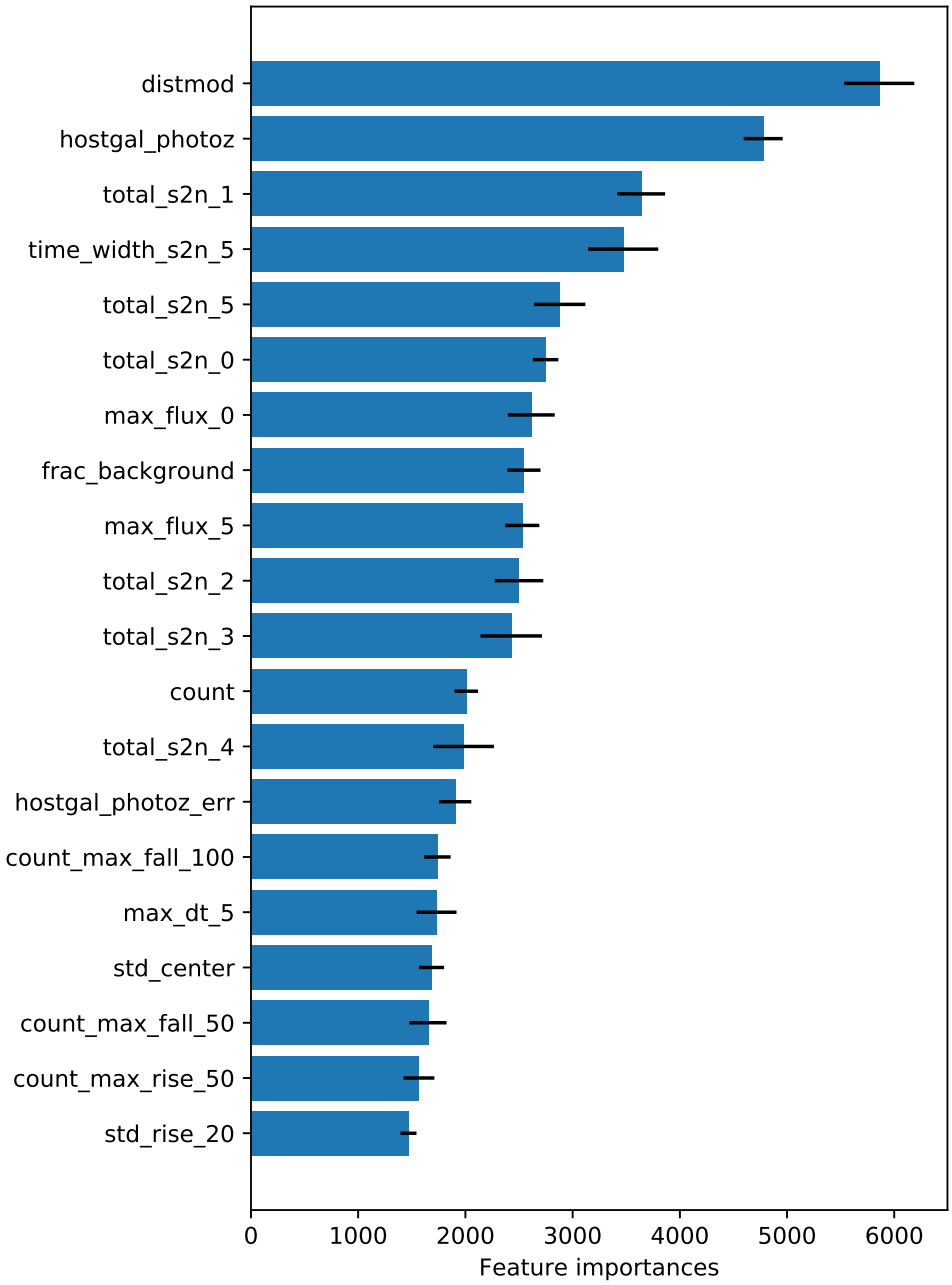
The 20 most important features are shown in Figure 1. We find that the most important features are the distance modulus / photo-z. When these are coupled with the maximum fluxes (eg: max_flux_0), the model is effectively discovering the absolute luminosity of the various objects which is a well-known way to distinguish between different transients. The total signal-to-noises in each band are all in the top features. Near the bottom of the top 20, various measures of the lightcurve width begin to appear.

I did not use any external data for this analysis.

A.5 Training Methods

I trained a LightGBM model on the feature set described above. The classifier was trained using the `multi_logloss` metric with weights for each sample designed to match the competition metric. I tuned the LightGBM parameters to optimize the cross-validation performance on the training set, and ended up with the following values:

Figure 1: 20 most important features for the model, ranked by their importances.



- `learning_rate`: 0.05
- `colsample_bytree`: 0.5
- `min_split_gain`: 10
- `min_child_weight`: 2000
- `max_depth`: 7
- `num_leaves`: 50
- `early_stopping_rounds`: 50

All other parameters were set to their default values. The model was trained with early stopping after 50 rounds of no improvement, and required ∼600 rounds to converge. The training takes ∼20 minutes on my machine with a 40-fold augmented sample. Increasing the number of augmentations past 40 could give additional benefits. My runs were limited to 40 augmentations because that uses all of the RAM on my system (16 GB).

A.6 Interesting findings

There are several different aspects to my approach that are very different that what was done by my competition. The first major challenge is dealing with sparse observations of the lightcurves. There are lots of gaps in the observations, and sequential measurements are often in different bands. The different classes of objects vary on wildly different timescales, so some are adequately sampled while others are heavily undersampled. Furthermore, some bands are much noisier than others so it is essential to take the uncertainties into account. For these reasons, I chose to use a Gaussian process model to fit the lightcurves. Gaussian processes are able to naturally handle all of the issues previously described and produce robust estimates of the true lightcurve. Examples of the Gaussian process predictions are shown in Figures 2 and 3.

Another major challenge is handling the differences between the training and test sets. I noticed that the training set typically has much higher signal-to-noise than the test set along with spectroscopic redshifts. The main issues with the test set are that its observations are typically at further distances than the training set and are at much lower signal-to-noises. My approach to resolve these differences was to augment the training set by degrading each observation in the training set to be more representative of the test set. For each object in the training set, I made 40 versions of it where I threw out large blocks of observations, threw out random observations, changed the distance (and redshift if applicable) and added noise. I calibrated all of these steps to be representative of the test set, and found that this procedure was very effective at improving performance on the test set.

My algorithm was focused on classifying supernova-like transients, and I did not put any significant effort into classifying other transients. I expect that additional features like measurements of Lomb-Scargle periodograms or fitting periodic-kernel Gaussian processes could be very useful in distinguishing different kinds of variable stars or other periodic transients. I found that my model was able to classify these transients well enough that they were not limiting me in this competition, so I did not pursue any of these additional features.

The confusion matrix for my final classifiers on the training set can be seen in Figure 4. The objects that look most supernova-like (42, 52, 62, 67 and 90) are the ones that are hardest for the model to distinguish between.

One unfortunate aspect of this competition is that the class 99 "unknown" objects can be identified by probing the leaderboard. I generated two different final submissions. My "real" prediction that is actually

Figure 2: Example of a Gaussian process fit to a class 52 object. The Gaussian process produces a robust estimate of the lightcurve, correctly handling one very high flux measurement with large uncertainty in band 5.

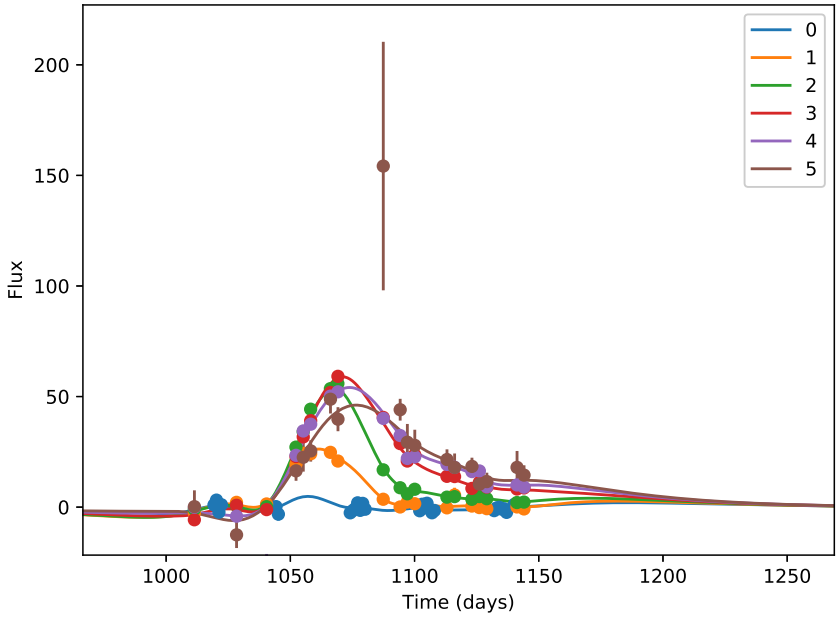


Figure 3: Example of a Gaussian process fit to a class 92 object. These objects vary on a timescale much smaller than the observations, and hence the lightcurve appears to be a sequence of random spikes. The Gaussian process fit identifies this, and drives the length scale down to very small values.

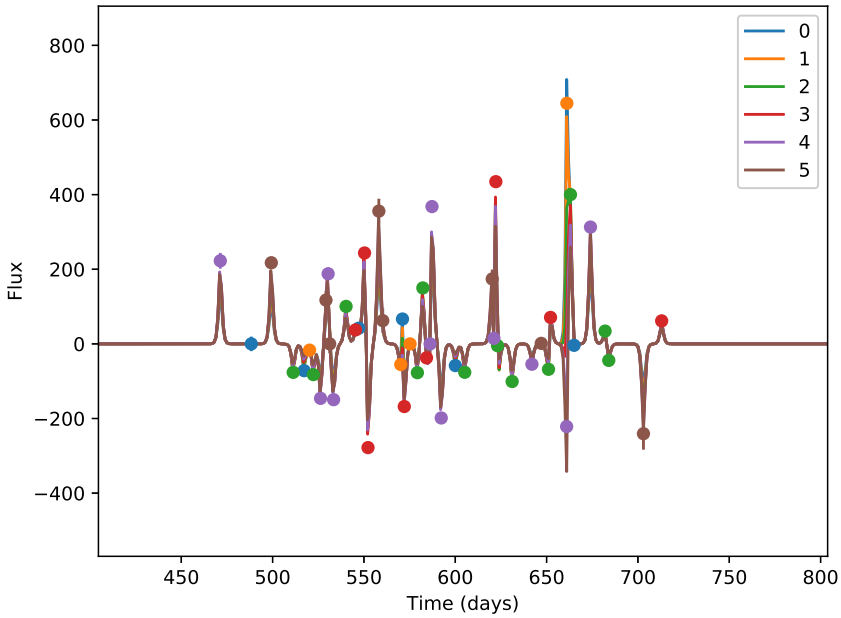
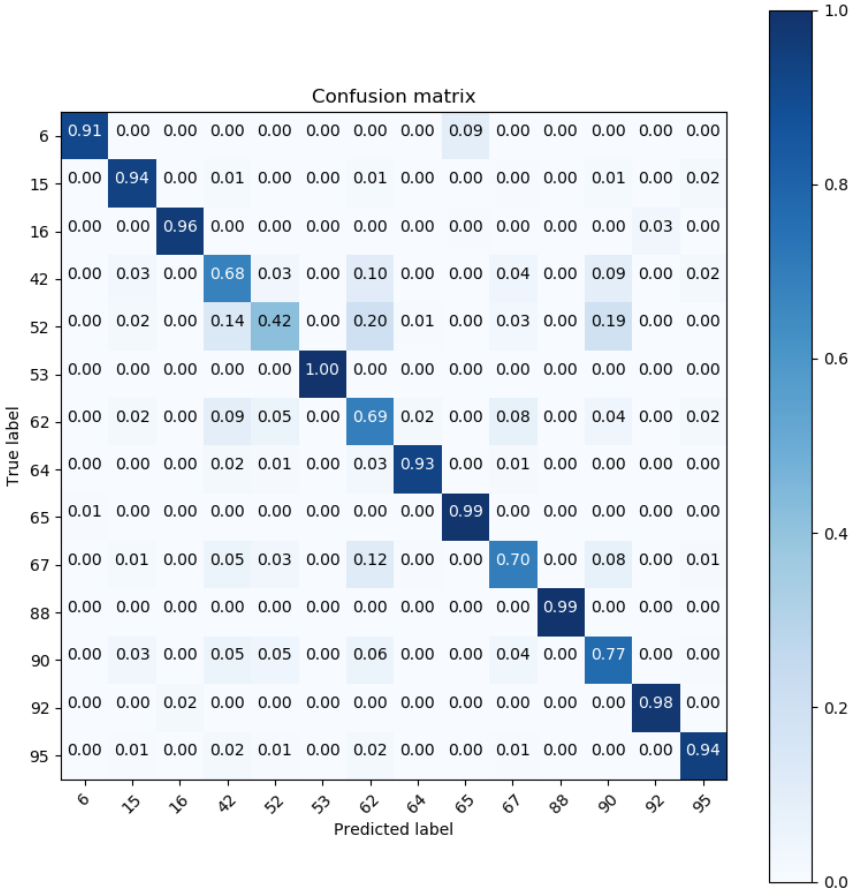


Figure 4: Confusion matrix for the final classifiers.



useful for science identifies outliers by modeling the expected scores of the class 99 objects and looking for predictions with anomalously low confidences. However, I discovered that predicting the class 99 objects using the following linear combination of the predictions for other objects gives an improvement to the final score of approximately 0.05:

$$p99 = 1.0 \cdot p42 + 0.6 \cdot p62 + 0.2 \cdot p52 + 0.2 \cdot p95 \quad (1)$$

My code outputs both final predictions.

A.7 Simple Features and Methods

The limiting computational component of this model is the Gaussian process fit which is required for most of the features. Computing features and performing predictions takes a negligible amount of time compared to the Gaussian process fits, so there is no computational advantage to not computing all of the features. The computation time for this model should not be an issue because the organizers will have access to a wide range of supercomputers for LSST and the total computational time for running this model on the full dataset is only roughly 100 hours.

A.8 Model Execution Time

As previously described, the slowest part of the model is the Gaussian process fits. My machine can perform approximately 10 fits per second, so it takes approximately 100 hours to perform the fits for the full train and test dataset.

Training the LightGBM classifier takes approximately 20 minutes once the features have been computed, and generating predictions takes approximately 1 hour, so these times are negligible compared to the step of fitting the Gaussian processes.

A.9 References

My code was originally based off of Olivier Grellier's kernel found at: https://www.kaggle.com/ogrellier/plasticc-in-a-kernel-meta-and-data.

There were several papers that I relied on heavily in getting inspiration for my solution:

- Charnock, T., Moss, A., 2017, ApJ, 837, 28
- Lochner, M., McEwen, J. D., Peiris, H. V., Lahav, O., Winter, M. K., 2016, The Astrophysical Journal Supplement Series, 225, 2, 31
- Malz, A., Hložek, R., Allam, T. Jr., et al., 2018, arXiv:1809.11145
- Naul, B., Bloom, J. S., Pérez, F., van der Walt, S., 2018, Nature Astronomy, 2, 151-155
- Revsbech, E., Trotta, R., van Dyk, D. A. 2017, MNRAS, 473, 3969
- The PLAsTiCC team, Allam, T. Jr., Bahmanyar, A., et al., 2018, arXiv:1810.00001

The code has been shared in a public repository on github at: https://github.com/kboone/plasticc

Additionally, I wrote up an overview of the solution which prompted a lot of interesting discussion at: https://www.kaggle.com/c/PLAsTiCC-2018/discussion/75033

A.10 Thoughts on the competition

Overall, I think that this competition went very well and that the algorithms that came out are a huge improvement over what was previously available in the field. Interestingly, the top 5 solutions were very different, and we were able to improve the final score by over 10% by using a weighted sum of the different solutions. Combining the best parts of the different solutions will lead to even better models for LSST.

My only issue with this challenge was that probing the "unknown" class 99 objects was a much more effective strategy than actually coming up with a real prediction for them. In most of the top solutions, competitors described different ways that they did this. Note that many of the methods that competitors came up with, like looking for gaps between the best and next best prediction, only worked because of the class 99 objects happened to look like supernovae and thus often have classification probabilities spread across several objects. This appears to be a legitimate algorithm, but it has no theoretical grounding and would not work if the unknown objects looked like eg: class 88 objects which are easy to classify. Algorithms like this, which were widely used and shared on the discussion boards, are only justified by leaderboard probing.

In the end, a "naive" class 99 algorithm like the ones shared on the discussion board only performs $\sim$0.05 worse on the competition metric compared to more sophisticated leaderboard-probing algorithms. The final results of the competition are therefore not significantly affected by this. If future competitions attempt to include a similar challenge, I would highly suggest putting different types of outliers in the public and private leaderboards so that users are required to come up with real algorithms rather than just probing the leaderboard. The consequences of this should be thought out carefully.

Thank you to Kaggle and to the organizers of this competition. I learned a lot by doing this competition, and really enjoyed the competitive yet friendly and collaborative atmosphere on the discussion boards and kernels. I hope that the output of this competition will be useful for LSST, and would be very interested to compete in another round of the competition!