Final Project Report
Astrophysical Objects Classification

Mai Ngo

DePaul University - College of Computing and Digital Media (CDM)

DSC 540 Advanced Machine Learning - SEC 801

Prof. Casey Bennett

March 19, 2024

Tables of Contents

Abstract

The field of astronomy has such a captivating sense of mystery that for generations, scientists and astronomers have been and still finding breakthroughs and studying evidence of unknown extraterrestrial life and celestial bodies outside Earth. In this study, using a versatile dataset collected from a very highly efficient telescope name Large-aperture Synoptic Survey Telescope (LSST), we aim to classify fourteen astrophysical class objects of comets and asteroids. Implementing various machine learning (ML) classification models to examine characteristics like spatial positions, light curves, and motions of an object for the classification of its cosmic class.

The LSST is an integrated telescope designed not only for Cosmo observation but also with a built-in automated ML system that capable of fast time domain observed objects classification by utilizing extensive decade-long astrophysical data. By repeatedly scanning the optical sky, the LSST can simultaneously learn and detect light patterns of astrophysical objects over time. Findings from this study suggest that light curves and brightness motions have significant effect on astrophysical objects identification. Given the robustness of the dataset, certainly there are ample room for further studies, which will be discussed later in the paper.

Introduction

The LSST telescope, currently located at the Vera C. Rubin Observatory in Atacama desert of Chile, is a revolutionary ground-based telescope that was initially designed to focus on two cosmic surveys conducting for its first 10 years of operation. The first survey is Deep Drilling Fields which focuses on specific regions and aspects of the sky such as the Galactic plane or nearby galaxies like the Large and Small Magellanic Clouds. The second survey is Wide-Fast-Deep that aims for the bigger picture, capture vast areas with a regular cadence that allows to map and monitor movements of the sky over time (Bianco et al, 2021). The ultimate goal is to construct a comprehensive digital map of the universe and further learn more about the unknowns. In addition, the observatory is named after an American astronomer, Vera C. Rubin which is famous for her amazing lifework focuses on finding profound evidence on the existence of unseen "dark" matter in the Cosmo. With that, LSST operation objectives is to carry on Vera's legacy. The telescope can categorize a substantial number of observed celestial objects, between 60 - 90%, also capable of making assessment on potential asteroids threat to life on Earth.

In late 2018, the Vera C. Rubin Observatory organized a very highly successful astronomical time series data analytic competition on Kaggle known as PLAsTiCC (Kaggle, 2018). This was the competition that set LSST dataset publicly available, and quickly gain popularity among data scientists to sharpen their analytical skills. The dataset has a versatility since it encompasses a combination of static numerical and time series data, also the challenge was appealing due to best-model prize money. The competition opens doors to all individuals with different skill levels in data analysis. Many scientists, researchers, and data enthusiasts approached the PLAsTiCC competition in different ways and methodologies. Some only focused on time series aspects of the data, while others aims to find meaning on the static features. Furthermore, the competition objective also open for not just classifying astrophysical objects, but also encourage to find other meaningful patterns and connections the universe may entail given the released dataset. Many submissions did further research and applied different approaches of data pre-processing based on astronomy knowledge, beyond traditional classification tasks. Overall, the competition was a great open-source opportunity for data enthusiasts.

In these studies, the goal is to integrate static numerical data and time series data together, while still keep the same original astrophysical object classification task that PLAsTiCC comes with. We aim to assess whether leveraging the combined dataset would enhance classification performance of astrophysical objects.

Literature Reviews

To gain foundational domain knowledge of the dataset and adequately select ML models to achieve optimal classification performances. We examine three studies that mainly discuss and analyze the outcomes from the PLAsTiCC competition. Noticeably, majority of top ranked submissions employ various data engineering techniques, resulting in substantial enhancements in output performance. Furthermore, the use of weighted metrics for class targets significantly helps to neutralize the dominance of popular classes, feature selection also has an important role here (Hložek et al, 2023). Majority of models yield success output use Boosted Decision Trees, particularly Light Gradient Boosting, and not so much on Neural Networks. This preference could be attributed due to the scientific complexity of the data, basic domain knowledge is required for correct pre-processing as well as conduct feature importance.

Other research also find that features related to light, such as light curve characteristics and redshift, represents light movement of celestial objects over time hold a crucial influence for accurate classification. Given this, the second study suggests that focusing on light features, especially considering the temporal aspect rather than spatial locations will give significantly better classification output (Kessler et al, 2019). Indeed, the third study conducted by Qu et al conducted in in 2021 proposes an approach that firstly run a preliminary analysis on bright timeframe only, which means only using light features during time frame that the object was deemed bright – data can be determined using light statistical thresholds based on domain knowledge. Features selected from bright object study with spatial data given in bright condition could potentially give a solid starting point for developing classification models.

With the intention to implement Neural Networks (NNs), we examine two studies for insights. The first article discusses the gaining traction in forecasting using NNs. Emphasizes guidelines and best practices for leveraging existing Recurrent NN architectures, highlighting their competitiveness in specific situations compared to other established models with user-friendly attributes such as ETS and ARIMA. While the second study discusses parameter estimation, exploring both fundamental and advanced NNs model structures, showing that adding multiple convolutional layers and batch normalization notably boosts performance accuracy. NNs is a deep learning ML model that is notorious for its long training time, and a wide range of complex architecture. Given the paper, we want to learn more which is the best approach to implement NNs; especially on a massive hybrid dataset in this case.

NNs are known for good compatibility with large dataset, capable to find deep underlying data pattern thus, capture special relationship between the data features. However, its trade-off is longer running times thus, computationally expensive. The studies also emphasizes the hesitance of previous researchers in using NNs due to its extensive architecture, potentially give similar output just like other simple ML models. The conventional approach involves utilizing built-in functions or default parameters provided by the software in use to mitigate the risks of overfitting, underfitting, or potential crashes (Hewamalage et al, 2021). In considering of hyperparameters, the two key parameters: kernel size and number of units have significant influence over its neighbor parameters within the layer, and ultimately the overall performance of the model. Therefore, it is advisable to tune on these two parameters first then start stretching to others (Lee & Song, 2019). Overall, we decided to deploy NNs model using Keras due to their flexibility, simplicity and robustness.

Methodologies – Classification Models

To align with the objectives of this study which is creating an adequately optimal classification model on a massive hybrid dataset, as well as satisfied the requirement of the final project. Several ML models are considered and implemented for comparison. Accuracy and Log Loss (Cross Entropy Loss) will be used as evaluation metrics since this is multi-class classification problem.

Random Forest (RF)

RF is a popular ML algorithm that is adored by many data enthusiasts. Not just because of its technical efficiency, RF is also the best model that can be explain to non-technical individuals. The algorithm is based on multiple individual Decision Trees in which number of trees can be pre-determined. Overall, RF accounts for all trees prediction - the class that receives most votes from all trees is chosen as the final prediction. The goal is to improve classification accuracy and prevent overfitting. RF was trained using Scikit-learn 'RandomForestClassifier'. Two criterion parameters used are 'gini' and 'entropy'. This is the only model run additionally with cross-validation due to its adequate run times, also trained on both regular and down sampled data, with and without feature selection using 'SelectFromModel' method respectively for each dataset.

Gradient Boosting (GB)

Equally similar to RF, Gradient Boosting is also commonly preferred by data enthusiasts to approach classification problem. Same concept, GB consists of multiple week Decision Tree learners. However, the significant difference is while RF builds tree parallelly, GB builds tree sequentially, with the expectation that current tree should correct the errors or emphasize misclassifications from the previous one. The goal is to minimize errors by focusing on the residual errors of the prior tree, this technique is known as gradient descent. GB was also trained using Scikit-learn 'GradientBoostingClassifier'. Two criterion parameters used are 'friedman_mse', 'squared_error'. This model run without cross-validation, longer run time compared to RF. Additionally, trained on both regular and down sampled data, with and without feature selection respectively for each dataset.

Adaptive Boosting (AB)

The third ML model can be considered as a cousin to GB. AdaBoost also works with the setup of multiple week Decision Tree leaners. However, instead of an 'almost' fully grown tree, these week learners only operate at one level, or in another word, only one feature is considered at a tree. This concept is called a decision stumps – simple tree that makes a random split decision. Similar to GB, the tree are trained sequentially; however, AB different in a sense that it assigns weight to each instance or observation. Then adjust that weight based on the performance of prior tree. Overall, AB is a quick easy approach to get an initial benchmark classification performance. AB was also trained using Scikit-learn 'AdaBoostClassifier'. Due to quick run times, we were able to apply multiple combination of parameters tuning which are 'n_estimators' – number of trees with values [20, 50, 100, 200], and 'learning_rate' at [0.01, 0.3, 0.5, 1.0]. This model also run without cross-validation. Additionally, trained on both regular and down sampled data, with and without feature selection respectively for each dataset.

Neural Networks (NNs)

The last ML model is Neural Network, a deep learning algorithm that has been previously introduced in Literature Reviews section. To simplify, NNs map the idea of how human brain work in terms of information communication between neurons. The artificial neurons in this case are called nodes and the objective is to ensure they are conveying the information or data efficiently that the final neurons would make accurate predictions. There are three important layers given a layer consists of multiple nodes: input, hidden, and output layers. Since this is a multi-class classification problem, targets are encoded as one-hot vectors format using 'softmax' activation function in the output layer for probabilistic predictions.

Furthermore, since this is a hybrid dataset, time-series data was trained using a Long Short-Term Memory (LSTM) layer, a type of Recurrent NN layer. On the other hand, static data was trained using two Dense layers, a type of traditional NN layer. Finally, using a Dropout rate of 0.2 before merging the outputs of these layers into the final Dense layer for prediction. NNs was trained using Keras, specifically the Functional API to accommodate the hybrid data situation. This model also trained without cross-validation and only on down sampled data with all features included.

Exploratory Data Analysis

As briefly mentioned above, there are two datasets from LSST. The first dataset – meta data is a static numerical data set describe characteristics of the object, while the second dataset is a time-series data that the telescope collected during its observation time frame.

Meta Data

There are 7,848 observations and 12 features (See Appendix A - Dataset Introduction, Table 1). There are four static features 'ra' – right ascension, 'decl' - declination, 'gal_l' – galactic longitude and 'gal_b' – galactic latitude describe the location of the object. Another four static features 'hostgal_specz' - spectroscopic redshift, 'hostgal_photoz' - photometric redshift, 'hostgal_photoz_err' - uncertainty on hostgal_photoz, 'mwebv' - extinction of light due to Milky Way dust describe the light motion of the object. Two features 'ddf' and 'mwebv' are dropped since they are not aligned with the study's objective. We only want to keep features that represents the characteristics of the object.

For each set of features, we visualize correlations and distribution of each feature (See Appendix A - Dataset Visualization, Figure 1, 2, 3 and 4). Overall, there is no high correlation between these features. However, there is interesting pattern such as an upside-down parabola between galactic longitude and right ascension of an object We also notice all features has extreme outliers and consider it as a good thing since these significant statistical thresholds will help to classify and identify objects to a specific class.

Time Series Data

There are 1,421,705 observations and 6 features (See Appendix A - Dataset Introduction, Table 2). Two features 'passband' and 'detected' are dropped since they are not aligned with the study's objective - keep features that represents the characteristics of the object. In order to understanding time series visualization, there are six 'passband' in this data represents six color lenses of the telescope. Each of these lenses will capture the light curves of the object differently and put a number representation of these captures that represented as 'flux' features.

Appendix A - Dataset Visualization, Figure 5 shows the light curve distribution of six randomly chosen objects. We notice that they all have different light curves pattern, also different observation time frame. To ensure correct merging between time series and meta data, 'mjd' is kept and apply conversion to date time to ensure that each object ID has the same number of observed rows, time length. With respect to object ID: Get the appropriate MAXIMUM of minimum number of obs (95) since target class '53' has the least amount of total Object ID count.

Then we apply Time Rolling technique on ‘flux’ and ‘flux_err’ features using time steps of 5 for even division across all retained rows.

Class Imbalance – Down Sampled Data

We also notice data imbalance issue in the dataset. For each transformation step, class imbalance plots are generated to capture whether there is the distribution change. Overall, the imbalance distribution pattern stay the same. Surprisingly, the distribution patterns of total object count, and unique object count across classes are also similar.

As previously recommended in one of literature reviews (Hložek et al, 2023), we will apply weights on target class to tackle this problem. Using data imbalance handling technique from Google (Google Developers - Machine Learning, 2023), we implemented weights on target class with the objective is still preserve same proportional distribution yet bring down the total count of object ID per class. The same weight will be re-applied again on the ML side. The down sapling factors are: {90: 20, 42: 11, 65: 10, 16: 10, 15: 6, 62: 6, 88: 5, 92: 4, 67: 4, 52: 4, 95: 4, 6: 4, 64: 3, 53: 1} given keys are target classes. Appendix A – Class Imbalance, Figures 1 and 2 show the distribution of target classes in train and test data, respectively after and before applying down sampling.

Final Data

Ultimately, after merging and down sampling there are two set of final data. Regular train data consists of 667, 725 observations, 18 features. Down sampled data which is derived from regular train data consists of 81,155 observations and also 18 features.

The 18 features are 8 static features that describe the location and light motion characteristics of the object. Other 10 features are time-rolled out features using time step of 5 from the two time-series columns. With respect to ML models, a Train-Test Split with ratio 0.65 – 0.35 was applied, also ensure similar class distribution in both sets. Feature Selection is only performed on static features.

Classification Results

For this study we established a training framework, start with using parameter tuning to find the based best model. Then apply the chosen model on both data considering with and without feature selection. However, throughout the training process, we were able to apply all steps to three models: Random Forest, Gradient Boosting and Adaptive Boosting. Further details and parameters for each model are specified in Methodologies – Classification Models section.

Appendix B – Classification Results, Table 1 shows output from Random Forest Classifier. Overall, statistically, both data perform well with RF. A notice on down sampled with feature selection yield significantly lower accuracy of 64%. Log loss represents the probability confidence in correct prediction; overall small log loss further prove the data is performing well using RF. Adequate run time consider size of training data.

Appendix B – Classification Results, Table 2 shows output from Gradient Boosting Classifier. Overall, the model does not perform as well as RF, also much longer run times. Accuracy on all data also drop from the 90% range down to 60% range. We also notice down sampled with feature selection gives the same output as RF. Same two features that describe light motion of an object were selected as well. Appendix B – Classification Results, Table 3 shows output from Adaptive Boosting Classifier. This model does not yield optimistic output. All four data perform pretty poorly compared to RF, especially down sampled data have very low accuracy of 23% and significantly high log loss. AB also select an additional feature that also represent light motion of an object.

Lastly, Appendix B – Classification Results, Table 4 shows output from Neural Networks Classifier. Using 5 epochs which are five learning rounds, down sample data with full feature yield and accuracy of 60% considered weight application on target class, a moderate cross entropy loss of 9.6

Discussion – Best Model

Overall, considered run times and statistical output, Random Forest classifier will be the most suitable to this dataset. Specifically, the model performs well on both train and down sampled data with full features. Especially regular train data yield 97% of accuracy and very small log loss of 1.05. Down sampled data also performs well in this case further prove that applying weight on target class was a good decision. We also notice a pattern across all four models that regular train data with full feature yield the best accuracy and smallest log loss.

As each model algorithm train with different objective as specified above in in Methodologies – Classification Models section, we cannot definitely conclude whether there is a best model in this sense, only in terms of statistically optimal. We also attempted to perform Support Vector Machine Classifier; however, due to its nature of computationally expensive and large train data size, we could not get the output despite a long training time of 10 hours.

Conclusion

In conclusion, this study provides good insights into LSST dataset. This is just one of the approaches on how to utilize this dataset in classifying astrophysical objects. By merging the static and time series data, we were able to find a good classification model which is aligned with the objective of study. However, since LSST constantly learn new data, also reinforce it classification system using predictive data, there is still much more room for improvement and enhancing the model.

Future Work

Due to the versatility of the dataset which consists of both static numerical and time series data, there is substantial room for further refinement in data preprocessing, particularly different method of feature engineering. Despite conducting this study with limited domain knowledge, we are already able to achieve a high 97% accuracy using RF classifier. With appropriate data augmentation techniques and domain expert, creating new features that capture the essence of object characteristics perhaps using machine learning methods, we expect to achieve even higher classification performance.

Additionally, class imbalance can also be tackled in different way, for example using SMOTE. Explore other time rolling technique on time series data is necessary as well. Both traditional ML models and NNs offer valuable insights in this study, each with its advantages and considerations. NNs was trained using only four layers so perhaps deeper, finer tuned NNs would be able to achieve desired outputs. To sum up, we believe there are so much more room for future work, this study is only focus on raw characteristics of the data.

References

Bianco, F. B., Ivezić, Ž., Jones, R. L., Graham, M. L., Marshall, P., Saha, A., ... & Willman, B. (2021). Optimization of the Observing Cadence for the Rubin Observatory Legacy Survey of Space and Time: a pioneering process of community-focused experimental design. *The Astrophysical Journal Supplement Series*, *258*(1), 1.

Google Developers - Machine Learning. (2023, June 9). Imbalanced Data. https://developers.google.com/machine-learning/data-prep/construct/sampling-splitting/imbalanced-data

Hewamalage, H., Bergmeir, C., & Bandara, K. (2021). Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting, 37*(1), 388-427.

Hložek, R., Malz, A. I., Ponder, K. A., Dai, M., Narayan, G., Ishida, E. E. O., ... & Zuo, W. (2023). Results of the photometric lsst astronomical time-series classification challenge (plasticc). *The Astrophysical Journal Supplement Series*, *267*(2), 25.

Kaggle. (2018). PLAsTiCC Astronomical Classification. Kaggle. https://www.kaggle.com/c/PLAsTiCC-2018

Keras. (n.d.). KerasTuner. https://keras.io/keras_tuner/

Kessler, R., Narayan, G., Avelino, A., Bachelet, E., Biswas, R., Brown, P. J., ... & Transient and Variable Stars Science Collaboration. (2019). Models and simulations for the photometric LSST astronomical time series classification challenge (PLAsTiCC). *Publications of the Astronomical Society of the Pacific*, *131*(1003), 094501.

Lee, H., & Song, J. (2019). Introduction to convolutional neural network using Keras; an understanding from a statistician. *Communications for Statistical Applications and Methods*, *26*(6), 591-610.

Qu, H., Sako, M., Möller, A., & Doux, C. (2021). SCONE: supernova classification with a convolutional neural network. *The Astronomical Journal*, *162*(2), 67.

Appendix A. Data

Appendix A – Dataset Introduction

Features	*Description*	*Data Type*
object_id	Unique object identifier.	Int32
ra	Right ascension, sky coordinate: co-longitude in degrees.	Float32
decl	Declination, sky coordinate: co-latitude in degrees.	Float32
gal_l	Galactic longitude in degrees.	Float32
gal_b	Galactic latitude in degrees.	Float32
ddf	A flag to identify the object as coming from the DDF survey area (with value DDF = 1 for the DDF, DDF = 0 for the WFD survey).	Boolean
hostgal_specz	The spectroscopic redshift of the source. This is an extremely accurate measure of redshift.	Float32
hostgal_photoz	The photometric redshift of the host galaxy of the astronomical source. While this is meant to be a proxy for hostgal_specz, there can be large differences between the two and should be regarded as a far less accurate version of hostgal_specz.	Float32
hostgal_photoz_err	The uncertainty on the hostgal_photoz based on LSST survey projections.	Float32
distmod	The distance to the source calculated from hostgal_photoz and using general relativity.	Float32
mwebv	MW E(B-V). this 'extinction' of light is a property of the Milky Way (MW) dust along the line of sight to the astronomical source, and is thus a function of the sky coordinates of the source ra, decl. This is used to determine a passband dependent dimming and reddening of light from astronomical sources.	Float32
target	The class of the astronomical source.	Int8

Table 1: Meta Data - Features names, descriptions and data types.

Features	*Description*	*Data Type*
object_id	Unique object identifier.	Int32
mjd	The time in Modified Julian Date (MJD) of the observation. Can be read as days since November 17, 1858. Can be converted to Unix epoch time formula unix_time = (MJD−40587)×86400.	Float64
passband	The specific LSST passband integer, such that u, g, r, i, z, Y = 0, 1, 2, 3, 4, 5 in which it was viewed.	Int8
flux	The measured flux (brightness) in the passband of observation as listed in the passband column. These values have been corrected for dust extinction (mwebv), though heavily extinct objects will have larger uncertainties (flux_err) in spite of the correction.	Float32
flux_err	The uncertainty on the measurement of the flux listed above.	Float32
detected	If 1, the object's brightness is significantly different at the 3-sigma level relative to the reference template. Only objects with at least 2 detections are included in the dataset.	Boolean

Table 2: Time Series Data - Features names, descriptions and data types.

Appendix A – Dataset Visualization

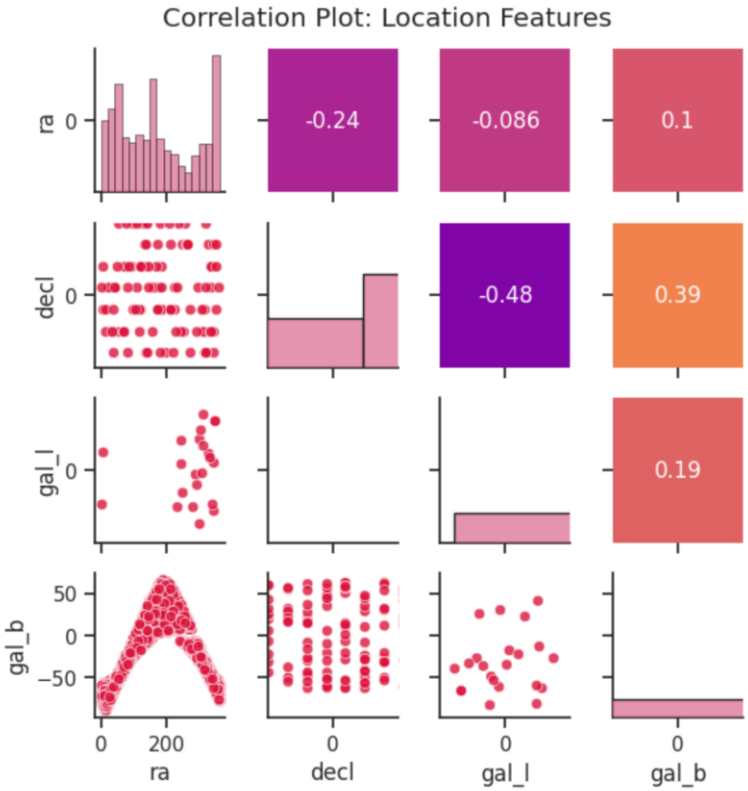

Figure 1: Meta Data – Location Features correlation plot.

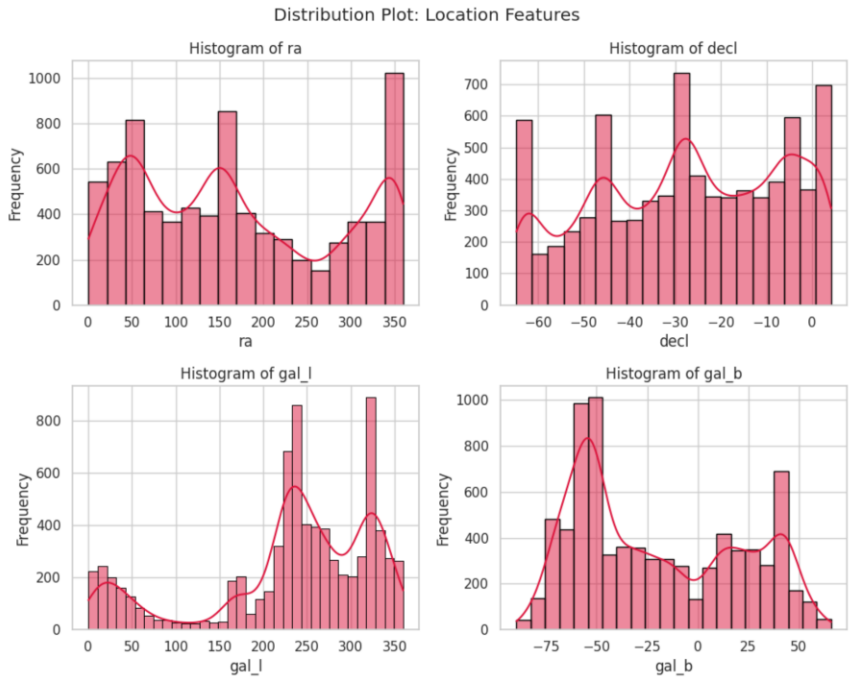

Figure 2: Meta Data – Location Features distribution.

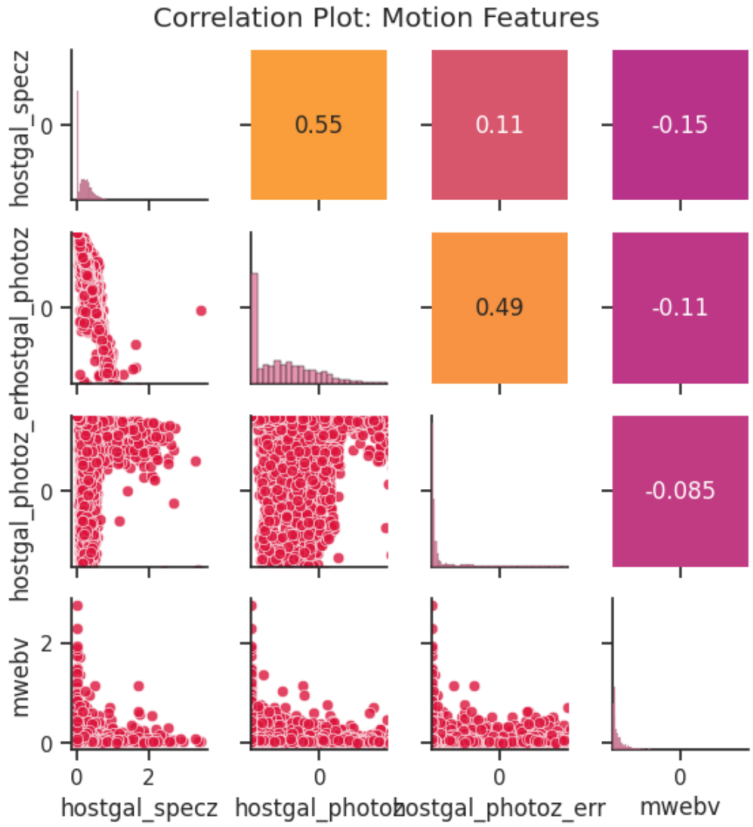

Figure 3: Meta Data – Light Motion Features correlation plot.

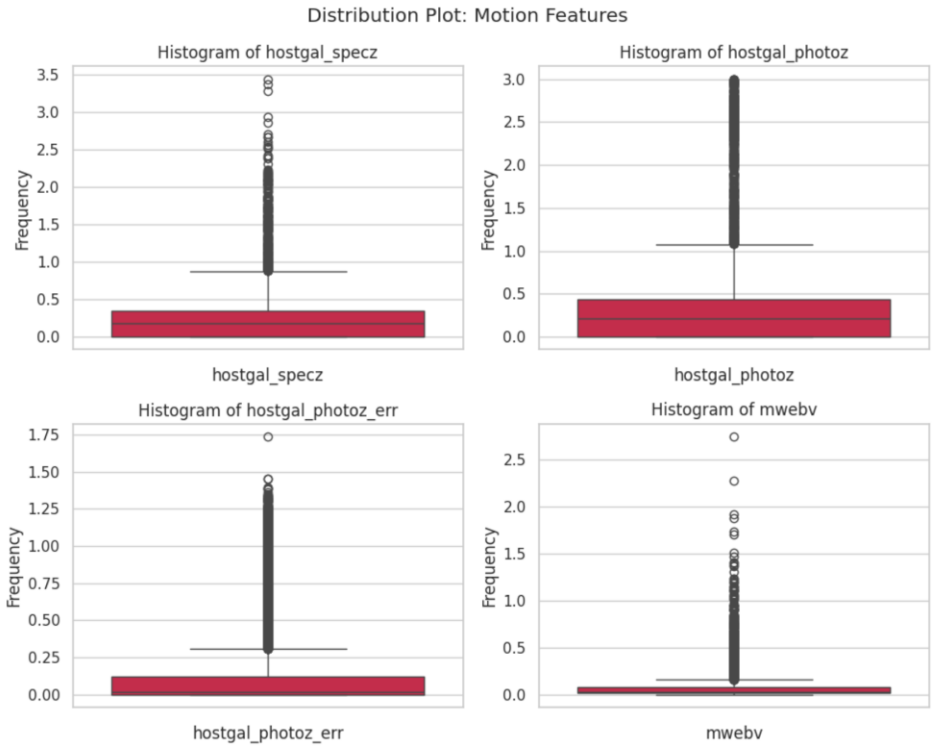

Figure 4: Meta Data – Light Motion Features distribution.

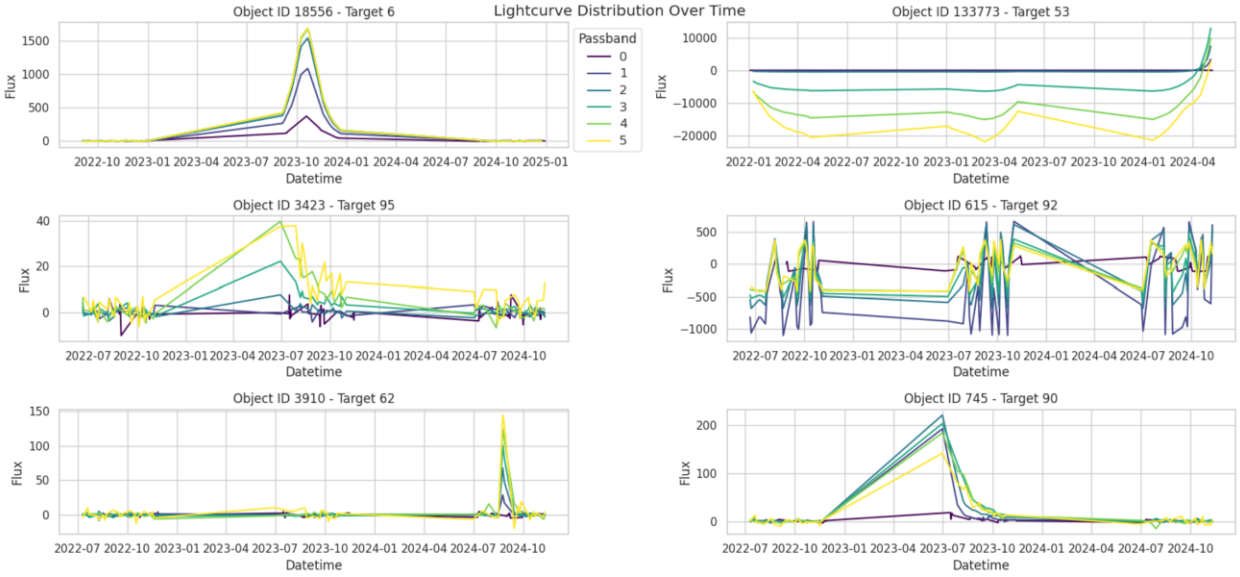

Figure 5: Time Series Data – Light Curves Distribution of six objects.

Appendix A – Class Imbalance

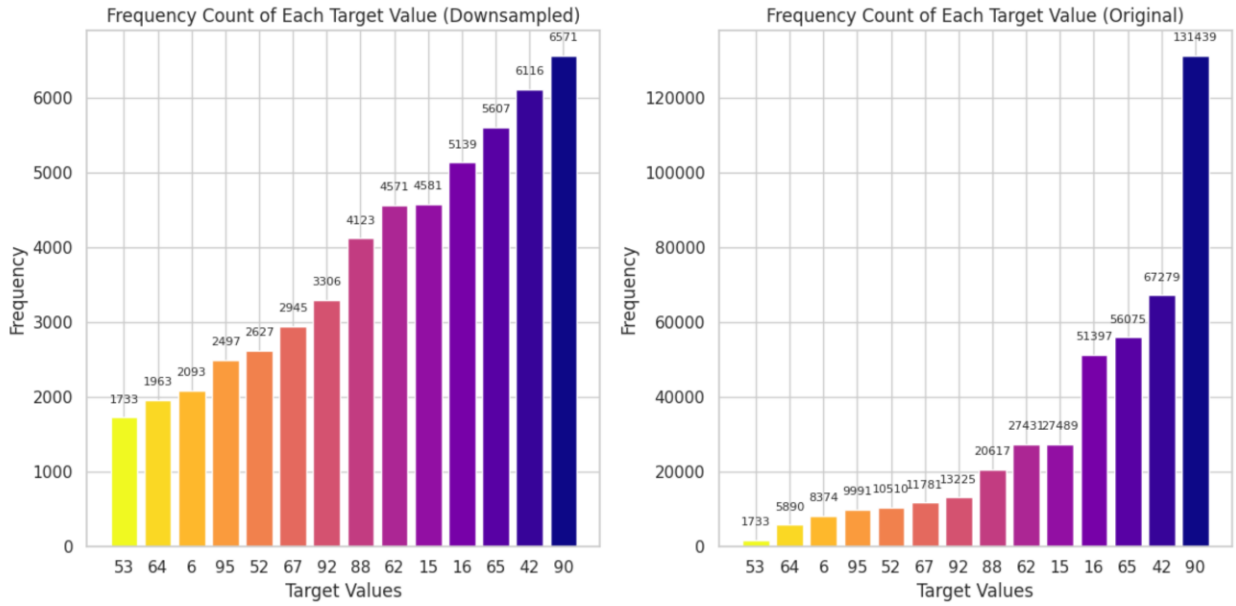

Figure 1: Train data – After and Before Down Sampling.

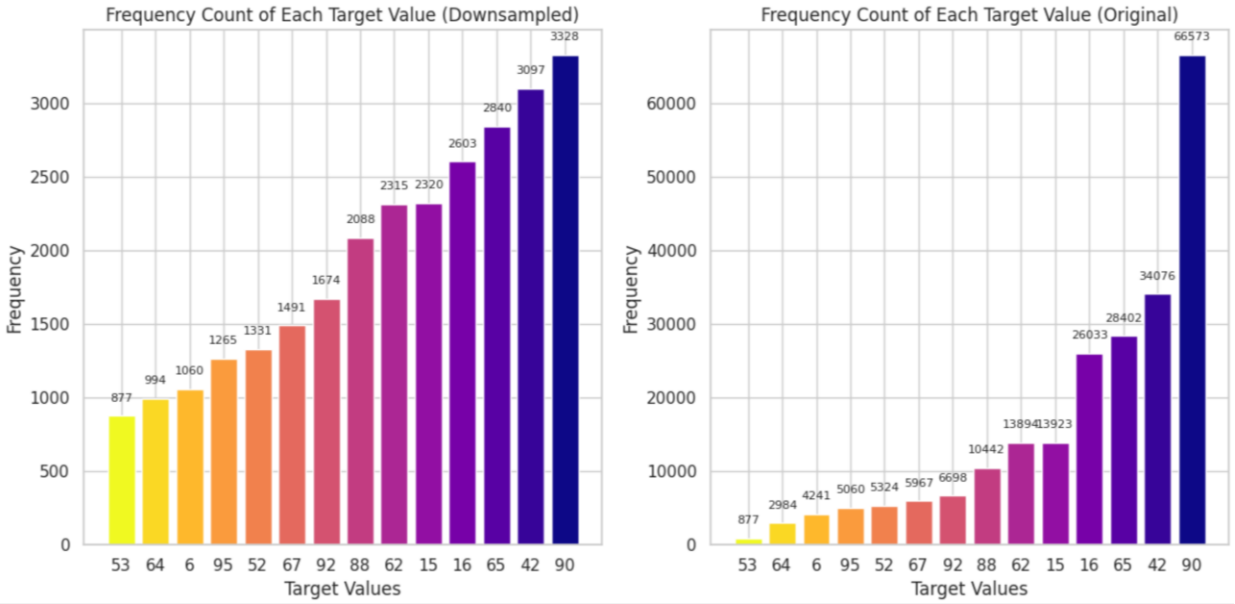

Figure 2: Test data – After and Before Down Sampling.

Appendix B. Classification Results

<table>
<tr><th colspan="2">Random Forrest Classifier</th></tr>
<tr><td>With CV</td><td>

	Criterion	Accuracy	Log_Loss	Runtime
0	gini	0.968527	1.054847	2295.272728
1	entropy	0.974020	0.873420	3256.859522

</td></tr>
<tr><td>Without CV</td><td>

	Criterion	Accuracy	Log_Loss	Runtime
0	gini	0.970734	1.054847	459.186801
1	entropy	0.975768	0.873420	696.432265

</td></tr>
<tr><td>Feature Selection</td><td>

```
Selected Features:
['hostgal_specz', 'hostgal_photoz']

Not Selected Features:
['ra', 'decl', 'gal_l', 'gal_b', 'hostgal_photoz_err', 'mwebv']
```

</td></tr>
<tr><td>Train Data (Full Features)</td><td>Acc: 0.97 | Log Loss: 1.05</td></tr>
<tr><td>Train Data (FS)</td><td>Acc: 0.81 | Log Loss: 6.78</td></tr>
<tr><td>Down Sampled Data (Full Features)</td><td>Acc: 0.91 | Log Loss: 3.17</td></tr>
<tr><td>Down Sampled Data (FS)</td><td>Acc: 0.64 | Log Loss: 2.89</td></tr>
</table>

Table 1: Random Forrest Classifier Output.

<table>
<tr><th colspan="2">Gradient Boosting Classifier</th></tr>
<tr><td>Without CV</td><td>

	criterion	Accuracy	Log_Loss	Runtime
0	friedman_mse	0.677087	11.638959	4274.507610
1	squared_error	0.677087	11.638959	4272.144021

</td></tr>
<tr><td>Feature Selection</td><td>

```
Selected Features:
['hostgal_specz', 'hostgal_photoz']

Not Selected Features:
['ra', 'decl', 'gal_l', 'gal_b', 'hostgal_photoz_err', 'mwebv']
```

</td></tr>
<tr><td>Train Data (Full Features)</td><td>Acc: 0.68 | Log Loss: 11.64</td></tr>
<tr><td>Train Data (FS)</td><td>Acc: 0.65 | Log Loss: 12.55</td></tr>
<tr><td>Down Sampled Data (Full Features)</td><td>Acc: 0.65 | Log Loss: 12.53</td></tr>
<tr><td>Down Sampled Data (FS)</td><td>Acc: 0.64 | Log Loss: 13.12</td></tr>
</table>

Table 2: Gradient Boosting Classifier Output.

Adaptive Boosting Classifier	
Without CV	

	n_estimators	learning_rate	Accuracy	Log_Loss	Runtime
0	20.0	0.01	0.423063	20.794934	79.554674
1	20.0	0.30	0.385182	22.160295	65.526772
2	20.0	0.50	0.219462	28.133425	53.503718
3	20.0	1.00	0.254858	26.857655	44.903271
4	50.0	0.01	0.423063	20.794934	108.427271
5	50.0	0.30	0.247797	27.112135	128.946388
6	50.0	0.50	0.195729	28.988863	102.836081
7	50.0	1.00	0.282783	25.851135	94.903687
8	100.0	0.01	0.457968	19.536824	207.686962
9	100.0	0.30	0.216785	28.229919	227.684102
10	100.0	0.50	0.209596	28.489055	215.905324
11	100.0	1.00	0.283674	25.819024	263.817219
12	200.0	0.01	0.278306	26.012493	520.437495
13	200.0	0.30	0.220723	28.087988	418.908736
14	200.0	0.50	0.206402	28.604173	538.301139
15	200.0	1.00	0.283674	25.819024	507.682440

Feature Selection	`Selected Features:`
	`['hostgal_specz', 'hostgal_photoz', 'hostgal_photoz_err']`
	
	`Not Selected Features:`
	`['ra', 'decl', 'gal_l', 'gal_b', 'mwebv']`

Train Data (Full Features)	**Acc: 0.46 | Log Loss: 19.54**
Train Data (FS)	**Acc: 0.31 | Log Loss: 24.74**
Down Sampled Data (Full Features)	**Acc: 0.23 | Log Loss: 7.78**
Down Sampled Data (FS)	**Acc: 0.23 | Log Loss: 27.78**

Table 3: Adaptive Boosting Classifier Output.

Neural Networks Classifier	

Without CV	`Trial 5 Complete [00h 03m 54s]`
	`val_accuracy: 0.45695120096206665`
	
	`Best val_accuracy So Far: 0.4680570363998413`
	`Total elapsed time: 00h 23m 37s`

Down Sampled Data (Full Features)	**Acc: 0.46 | Acc(weighted): 0.59 | Log Loss: 9.6**

Table 4: Neural Networks Classifier Output.