University of Cape Town

Department of statistical sciences

AST5005W

Data Sci: Minor Dissertation

Machine Learning techniques to discover and understand the population of flare stars in MeerLICHT data

---

Name: *Aphiwe Bangiso*

Student No: *BNGAPH002*

Supervised by

- Prof. Paul Groot
  Department of Astronomy

Co-supervised by

- Dr. David Buckley
  Department of Astronomy
- Dr. Cole Johnston
  Radboud University

18 October 2022

Acknowledgments

I would like to thank my supervisors for their continuous support during the course of this project, it would not have been possible without their support. I would like to also thank the Inter-university Institute for Data-Intensive Astronomy (IDIA) for providing computational resources for this project. This project was funded by the National Astrophysics and Space Science Programme (NASSP) and the department of astronomy at UCT.

Contents

List of Figures

List of Tables

Abstract

In this era of information overload, machine learning and artificial intelligence have been increasingly popular in various fields, including the field of astronomy. These approaches attempt to extract meaningful information from the data through automated means. In this work, we develop generic machine learning models that classify a given transient object from the observed light curve. We train random forest (sect 4.1.1) and multilayer perceptron neural network (sect 4.1.3) models on simulated LSST PLAsTiCC data and real data from the MeerLICHT survey. We found that the random forest model outperforms the neural network model in both data sets, achieving test accuracy of 66.0% and 98.0% in the PLAsTiCC and MeerLICHT data respectively. On the other hand, the neural network model achieved test accuracy of 65.7% and 86.6 % in the PLAsTiCC and MeerLICHT data respectively. For PLAsTiCC simulated data, we also show that grouping all types of supernovae into one aggregate class and discarding distance information improves the performance of both models to 96.5% and 96.0% for random forest and neural networks respectively. As additional work, we attempt to find sub-classes within the M-type class in MeerLiCHT data using k-means and hierarchical clustering algorithms. We find two distinct sub-classes in this data. Namely variable and non-variable M-type stars.

1 Introduction

Many astronomical surveys are collecting large amounts (multiple terabytes) of data for various research purposes. Some examples of such surveys include (but not limited to) Zwicky Transient Facility [Bellm et al. 2018], MeerLICHT [Bloemen et al. 2016], ASAS-SN [Jayasinghe et al. 2018], future BlackGem [Groot et al. 2019], the Sloan Digital Sky Survey [York et al. 2000], Dark Energy Spectroscopic Instrument [Collaboration et al. 2016, Abareshi et al. 2022], The 2df galaxy redshift survey [Colless et al. 2001], SAGES Legacy Unifying Globulars and GalaxieS Survey [Brodie et al. 2014], VLA sky survey [Condon et al. 1998], etc. This data needs massive amounts of processing to achieve a scientific goal, i.e. object classification, time-series analysis, etc. In traditional settings, astronomers perform the classification of sources in the data manually. This process may be effective as long as the amount of data remains relatively small. Sometimes through this process, however, classification efforts may be limited to known sources and properties. Astronomers may be biased by identifying sources with well characterised behaviour, thus reducing the chances of discovering new and unknown sources. Due to these considerations, many researchers have proposed machine learning approaches to automate classification and remove potential bias. These methods have proven to be very effective in classifying and processing large amounts of data. Automated machine learning classifiers can easily be integrated with the astronomical data processing pipelines as well. Such an integrated scheme enables real-time classification and the generation of automated alerts for the detection of interesting and time-critical astronomical sources.

A lot of work has been carried out to attempt to develop automated means for classifying variable objects. Orwat-Kapola et al. (2021) developed a pipeline for efficient classification of X-ray binaries based on light curve data. Bassi et al. (2021) have applied a hybrid neural network of one-dimensional convolutional neural network and a Long Short Term Memory network which used the raw time-series data from variable stars and reported a mean classification accuracy of 85%. Sreehari and Nandi (2021) have applied unsupervised machine learning methods to find subgroups in black hole binaries. They found that k-means clustering

provides more reliable results compared to other standard clustering techniques. Tachibana and Miller (2018) compared random forest algorithm with the SDSS and PS1 photometric classification models. They found that the random forest model outperforms these models. Hinners et al. (2018) used machine learning methods to predict and classify steller properties of objects in Kepler data using representation learning and feature engineering. They were able to successfully predict stellar density, stellar radius, and effective temperature with errors below 2%-4%. Pasquet et al. (2019) developed a model called PELICAN, for the characterization and the classification of supernovae time series data. They have proven that PELICAN is capable of dealing with sparsity and irregular light curve samples. They apply their model to other datasets such as the Supernova Photometric Classification Challenge and LSST Deep Fields achieving the accuracy of 81.1% and 96.5% respectively. Burhanudin et al. (2021) trained a recurrent neural network model to perform real-time classification of objects in Gravitational-wave Optical Transient Observer data using photometric time series and additional contextual information. The model achieved maximal Area Under the Curve of 0.972. There are many other attempts to characterize and classify astronomical sources through automated means, some are reported by A. Miller et al. (2015), Barbara et al. (2022), Linares et al. (2020), Mahabal, Rebbapragada et al. (2019), Guglielmetti et al. (2022), etc.

In this research, we will be developing generic machine learning models to classify different types of astronomical objects given an observed light curve. Our general aim is to identify flaring variable stars amongst other variable and transient sources observed with the MeerLICHT telescope, and sources in simulated LSST (Rubin Observatory Legacy Survey of Space and Time) PLAsTiCC datasets. Flares are characterized by a rapid increase in brightness, followed by a slower decrease in brightness over an extended period of time (tens of minutes to hours)[Dzombeta and Percy 2019]. Often characterised by a fast rise and exponential decay (FRED) morphology [Goad et al. 2007]. In this way, flares are morphologically similar to other explosive astronomical events, such as novae or supernovae. While these have different physical mechanisms and occur on different time scales, they are also characterized by a rapid increase in brightness followed by a slower dimming. An

M dwarf flare is an explosive phenomenon that results from magnetic reconnection events on the surface of the star. They are thought to be similar to solar flare with energies ranging from 10^{29} erg to 10^{32} erg [Y. Notsu et al. 2016]. The charged particles spiral down the field lines from the reconnect point. These particles heat up surrounding material as they spiral down a newly reconnected magnetic field line. They get decelerated as they enter a dense plasma causing flare emission [Jackman et al. 2021].

Flares are thought to emanate mostly from M-type stars. The low mass M-type stars make up more than 70% of the stars in the Universe and they host exoplanets. For this reason, we observe a high rate of occurrences of flares and due to their abundance M-type stars are a primary target for searching for life forms in the Universe. Astronomers have been interested in studying how flares influence the evolution of exoplanets in the vicinity of the host star and their habitability zones. The high energy emission from flare events may alter the chemistry and structure of the atmosphere of exoplanets that are situated in the vicinity of the host star. The high radiation from the flare event may cause atmospheric escape processes to exoplanets with thin atmospheres. This has detrimental effects on astrobiology life forms [Venot et al. 2016]. On the other hand, flare events release chemical elements that can have a positive contribution on the chemistry of the exoplanets. These can initiate prebiotic chemistry in the planetary atmosphere [Yamashiki et al. 2019]. We are particularly interested in studying the influence of flare events on the habitability zones in the exo-planets. A clear understanding of the impact of flares on the habitability zone will give us clues or hints on how life started on our own planet.

Our goal is to develop machine learning models on the simulated LSST PLAsTiCC dataset [Malz et al. 2019, Allam Jr et al. 2018] and then apply it to MeerLICHT data [Bloemen et al. 2016]. We also apply clustering techniques to identify subgroups in M types stars in this data. Additionally, we perform feature selection techniques to extract features that are more relevant. Here relevant means features that have some correlation with the response variable. This also ensures the effective use of computational resources.

The layout of this thesis is as follows, we introduce astronomical concepts including variable types in section 2. We present the properties of the data in section 3. Section 4 gives a quick overview of the machine learning algorithms (including metrics) that will be developed in this project. In section 5 we present and discuss the results. Finally, we conclude in section 6.

2 Astronomy

Stars are the fundamental building blocks of the visible Universe. They are responsible for the formation of all chemical elements heavier than lithium, such as carbon, nitrogen, oxygen, etc. These elements are produced via nuclear fusion in the stellar core at different stages of a star's evolution. Stars are formed from the gravitational collapse of a cool and dense molecular cloud of gas. As the gas collapses, in-falling material clumps together and rotates. As more and more material continues to contract, the temperature begins to increase, which then increases the local opacity. Eventually, the heat from gravitational collapse begins to produce mechanical equilibrium, forming a proto-star. As the protostar spins, a disc of material is formed around it. This disc will continue to feed material to the proto-star until its core becomes massive and hot enough for nuclear fusion to occur. At this point, hydrogen fusion will begin in the star's core. The star is now born and moves to the main sequence of the Hertzsprung-Russell diagram (see sect 2.4). It will spend roughly 90% of its life on the so-called main sequence where it burns hydrogen to helium in its core. Generally speaking, the mass and chemical composition of a star determines its evolutionary path, while its temperature and brightness determine its position on the HR diagram at any point in its evolution. For convenience, stellar masses are measured in units of solar masses. Stars with less than one solar mass are cooler and fainter than the Sun when they are born. Conversely, more massive stars are hotter and brighter than the Sun when they are born.

Eventually, the star will run out of hydrogen in its core. At this point, the star moves off the main sequence as it develops a hydrogen burning shell around the core. The remaining evolution of the star depends strongly on its mass. Stars

less than 9 times the mass of our Sun, will end their life as a small, hot white dwarf. Stars more massive than this, however, will end their lives explosively as supernovae, creating either a neutron star or black hole remnant [Cherchneff 2009].

2.1 Stellar radiation

A black-body is a hypothetical object that absorbs and re-emits all the electromagnetic radiation incident to it without scattering or reflecting the radiation. Stars are considered to be a good approximation of black bodies. According to Planck's Law, the electromagnetic radiation emitted by an object depends on its temperature. Hot objects emit more radiation at a short wavelength while cooler objects produce radiation at the longer wavelength of the electromagnetic spectrum. Using Planck's radiation law, the black-body radiation models the relationship between the temperature and wavelength (or frequency) as:

$$B_\lambda = \frac{2hc^2}{\lambda^5} \times \frac{1}{e^{\frac{hc}{\lambda K_B T}} - 1}. \tag{2.1.1}$$

Here k_B is the Boltzmann constant, h the Planck constant, and c the speed of light in vacuum. Generally, this formula allows us to understand the observed light of stars.

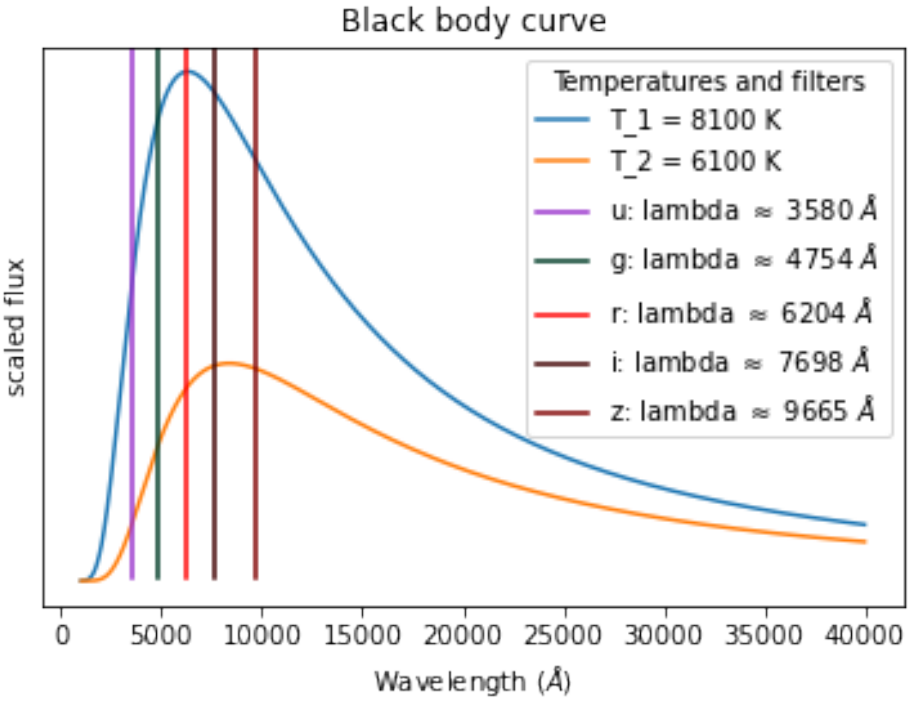
Photometric astronomical observations are taken in specific, discrete wavelength ranges. This is achieved through the use of filters, which block light at all wavelengths except for the desired range. Some of the more commonly used set of filters are the Sloan *u, g, r, i,* and *z* filters. These filters are centered at 358.0, 475.4, 620.4, 769.8, and 966.5 nanometers respectively, covering a large portion of the optical region.

As aforementioned, depending on the temperature, an object can be brighter in one filter compared to others. We measure the magnitudes of stars (see sect 2.3 for a detailed description of stellar magnitudes) at a specific wavelength using filters. Using this information one can derive colour information of the stars by taking the difference between any two magnitudes of the same star measured at different filters. Since the magnitude decreases with brightness, when taking the difference

between the red and blue magnitude for the blue star then $g - r$ will be small. Conversely, $g - r$ will be larger for a red star. In this way, the colour $(g - r)$ of a star can act as a proxy for that star's surface temperature.

Suppose we measure the black body curves of two stellar bodies with surface temperatures 8100 and 6100 Kelvin, let's call them star A and B respectively. The black-body curves for these objects are shown in figure 2.1.1. Suppose we use the filters *u, g, r, i*, and *z*. From figure 2.1.1, it can be shown that taking the color difference $u - g$ for both objects one would get a more positive value implying the objects are fainter in u than in g. Similarly, the difference $g - r$ is positive since g is fainter than r. On the other hand, for the star with temperature $T_1 = 8100$ K, the difference between $i - z$ would be larger implying the object is much fainter in z compared to i. However, this is not the case for the star with temperature $T_2 = 6100$ K. $i - z$ is equal to zero, implying the object (with $T_2 = 6100$ K) is equally bright in the near infra-red (i) as it is in the infra-red (z). Using just this information, one would easily determine that star A is bluer than star B. We have used colour information as features in our models. The colour information offers the advantage that it does not depend on a distance since all magnitudes for a single object are measured at the same distance.

Figure 2.1.1: Simulated black body curve



2.2 Luminosity and flux

In order to study stellar bodies we measure their radiation. From equation 2.1.1 it can be shown that the total energy emitted per unit area by a blackbody is given by 2.2.1. This is called Stefan–Boltzmann law [e.g Landsberg and De Vos 1989].

$$F \propto \frac{T^4 R^2}{d^2} \tag{2.2.1}$$

Where F is the flux received on earth, R is the radius of the star and d is the distance between the observer and the star. For a star with known temperature T and radius R, the bolometric luminosity is given by:

$$\begin{aligned} L &= A \times F \\ &= 4\pi R^2 \sigma T^4 \end{aligned} \tag{2.2.2}$$

2.3 Absolute and apparent magnitude

The observed brightness of a star is typically expressed in magnitudes. The magnitude can be related with the distance by:

$$m = -2.5 \times log_{10}(\frac{L}{d^2}) + C \tag{2.3.1}$$

Where L is the luminosity, d is the distance from the star, and C is a constant. In order to measure the absolute magnitude, d is set to equal 10 parsecs while for apparent magnitude d is the distance between target star and observer. The apparent magnitude is usually denoted with m and absolute magnitude with M. The difference between apparent and absolute magnitude is called distance modulus [e.g Hogg 1999]. It is given by:

$$m - M = 5 \times log_{10}(d) - 5 \tag{2.3.2}$$

The distance modulus is a measure of distance to a star. It can be used to infer whether an object is at, closer, or further away from 10 parsecs. The distance modulus with 0 value implies that the star is exactly at 10 parsecs, negative value implies it is closer than 10 parsecs while positive value implies it is further than 10 parsecs.

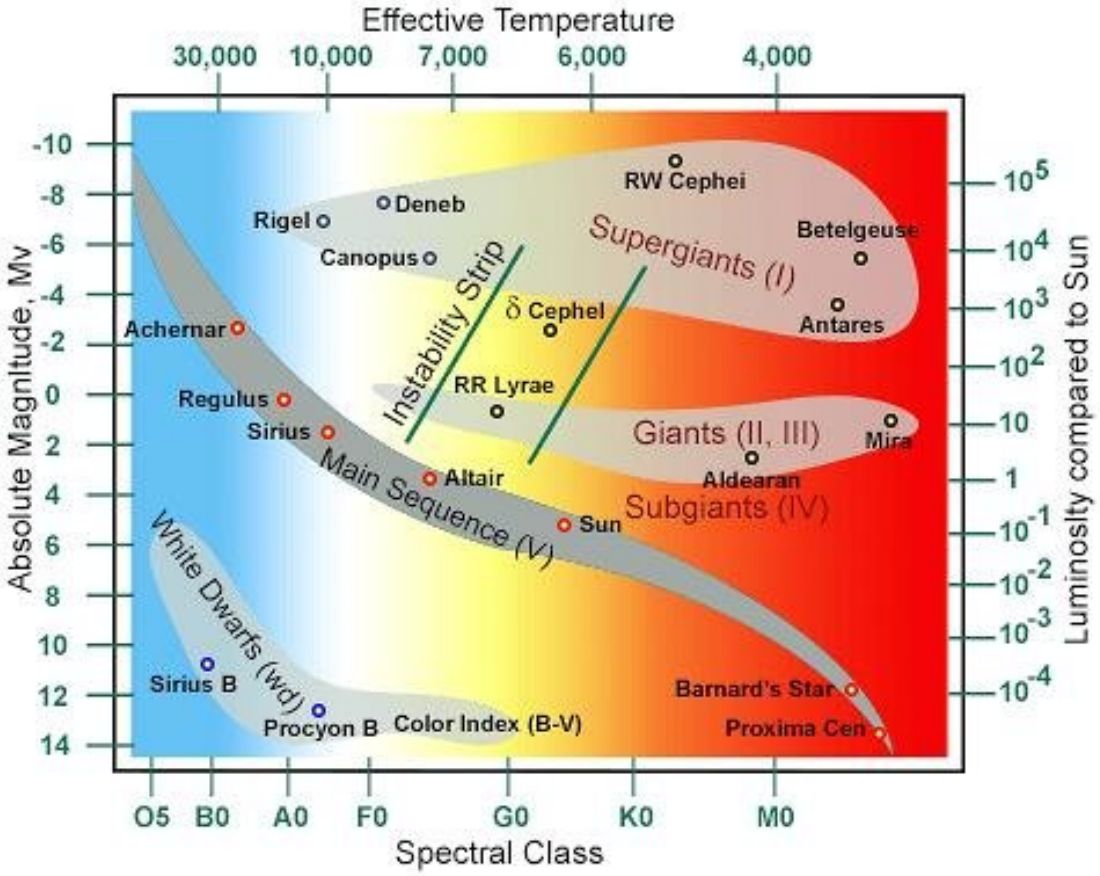
2.4 Hertzsprung-Russell Diagram

The Hertzsprung-Russell diagram is a graphical representation that plots the intrinsic brightness against the spectral type or temperature of celestial objects [Arp 1958]. Figure 2.4.1 illustrates the HR diagram taken from Agrawal (2018) . It is a great tool for visualizing the evolutionary track of a group of stars. Some of the most prominent features of the HR diagram include the main sequence, supergiant, red giant, and white dwarfs. The main sequence runs from the top left corner to the bottom right corner. This is where stars spend most of their lives burning hydrogen into heavier elements. Once the star runs out of hydrogen it moves off the main sequences to other stages of evolution. Stars enter the supergiant and red giant stage when they have exhausted the hydrogen in their core and start to burn

heavier elements in their cores. These stages are found above the main sequence. Stars in this stage have high luminosities and large radii due to expansion. The white dwarf region is found below the main sequence. This is the final stage of the Sun-sized star when it has lost its shell exposing a non-burning core.

The instability strip is the region above the main sequence occupied by pulsating variable stars. The stars vary in luminosity as well as in radii. Massive stars join this region after they left the main sequence. Horizontal-branch is an evolution star that is subsequent to the red giant stage. These stars experience structural changes that lead to a reduction in size, and luminosity which in turn increase their temperatures.

Figure 2.4.1: Hertzsprung-Russell Diagram [Agrawal 2018]



2.5 Variable stars

A variable star is a star that is varying in brightness with time. This variation in brightness can be caused by something extrinsic to the star, like on an eclipsing binary, or by something intrinsic to the star, such as flares, spots, or pulsations. The goal of this thesis is to investigate how to classify different types of variable stars with a time series of stellar brightness observations (magnitudes), namely light curves. Here, we briefly describe some types of variable stars that we will be classifying in this project. Of course there are other types of stars which we will not cover here.

2.5.1 Cepheids

Cepheids are pulsating stars that vary in both diameter and temperature. These are found in the instability strip of the HR diagram. They are burning helium to carbon and oxygen in their cores. They exhibit a period luminosity relationship [e.g Yuan et al. 2022], which states that objects with long period have greater intrinsic luminosity. They are very similar to RR Lyrae however Cepheids are more luminous. They are used as standard candles for measuring intergalactic and intragalactic distances because their pulsation period correlates with their luminosity.

2.5.2 RR Lyrae

RR Lyrae are variable type stars that are found in the horizontal branch of the HR diagram. They are suspected to have been main sequence stars with the mass and size similar to that of the Sun. They have a period ranging from few hours to 1.2 days [e.g H. A. Smith 2004]. They also exhibit the period luminosity relationship found in Cepheids but are much fainter than Cepheids [e.g Catelan et al. 2004]. The RR Lyrae variables are found in any stellar population of sufficient age, which include the Galactic halo and globular clusters.

2.5.3 Eclipsing binaries

Eclipsing binaries describe a star system in which two stars orbit around a common center of mass. Depending on the inclination of their orbits, this can cause the star system to vary in brightness. This variation in luminosity is caused by blockage along the line of sight by either one or both stars in the system if the orbital plane is on our line of sight. When observing these objects over a period of time one might see a dip in light observed. Any star can be in an eclipsing binary system, but it is observed that 30% of stars like our Sun are in a binary [e.g Raghavan et al. 2010], whereas most if not all O and B type stars are thought to be in binaries [e.g Sana, De Mink et al. 2012, Sana, Le Bouquin et al. 2014].

2.5.4 Supernovae

A supernova is a luminous explosion that results from a dying massive star. Stars generate energy while balancing pressure and gravity. The gas pressure is pushing the star outwards due to the heat generated when fusing elements and gravity tries to squeeze inwards causing the star to shrink. Once the star runs out of fuel the pressure drops significantly. At this point, gravity wins and shrinks the star. The star suddenly collapses causing a massive explosion, this is a supernova. Supergiant stars are believed to be Supernova progenitors. Supernovae may also be formed in a binary system accreting mass. This occurs when a white dwarf accretes mass from a nearby star causing a thermal explosion. These processes may lead to different types of supernova. We will not go through these here.

2.5.5 Active galactic nuclei

The Active Galactic Nuclei, also known as AGN, are active supermassive black holes in the galaxy that are powered by mass accretion [e.g Padovani et al. 2017]. They are thought to be the most energetic objects in the Universe [e.g Beckmann and Shrader 2013]. The AGNs are detectable as light in all wavelengths along the electromagnetic spectrum [e.g Padovani et al. 2017].

2.5.6 Tidal disruption events

Tidal disruption events occur when a star is ripped apart by tidal force from a supermassive black hole [e.g Mattila et al. 2018]. The TDEs are usually accompanied by transient flares when material from the companion star is accreted onto the black hole.

2.5.7 M type stars

M-type stars, also known as M dwarfs are the most common types of stars and are making up more than 70% of all the stars in the universe. They are small in size, cool, and dim. They have a long life span, with low temperature and luminosity peaking in the red [e.g Tarter et al. 2007]. They are also active flaring stars, which opens an area for studying their habitability. M-type stars often host exoplanets, making them a target for searching for life forms in the universe.

2.5.8 Kilonovas

A Kilonova event is an astronomical event that results from a merging compact binary star system that has at least one neutron star. The binary system may include a neutron star and a black hole or two neutron stars. When the two objects collide they release high-energy radiation and gravitational waves [e.g Kessler et al. 2019].

2.5.9 Mira variables

Mira variable stars are cool stars that vary in brightness for a period ranging from 150 to 500 days. These stars are found in the asymptotic giant branch of stellar evolution [e.g Castelaz et al. 2000].

2.5.10 Microlensing

Microlensing point sources are sources that are due to the gravitational lensing effect. Gravitational lensing occurs when a foreground object crosses the line of sight of a distant object, curving the space [e.g Kessler et al. 2019]. This causes a double image of the background object.

3 Data sources and properties

In this section, we describe the sources and properties of the data on which we will be developing machine learning models. Our goal is to try machine learning models on the simulated LSST PLAsTiCC dataset [Malz et al. 2019, Allam Jr et al. 2018] and then apply it to real MeerLICHT [Bloemen et al. 2016] data. We first develop models on the LSST data to evaluate our results on an established series of training and test data and compare against results by other groups. When developing machine learning models, we need some way to evaluate model performance. To do this the data is split into two sets, where one set is used to train the model called the training set, and the other set is called the testing set, which is used to evaluate model performance on the unseen data. Note that the testing data is not used in the training step. The model performance is assessed on the testing data. See section 4.2 for more on model evaluation.

Below, we briefly describe the sources of data, PLAsTiCC data is presented in section 3.1 and MeerLICHT data in section 3.2. The individual features of each data set are explained in table 3.3.1. It must be noted that for both data sets, we only considered time domain components of the light curves since we are interested in the actual representation of the light curves. We did not consider Fourier components of the light curve since the majority of our variable types do not have periodic variation. This comes from the fact that the Fourier transform assumes that the signal is periodic. The Fourier transformation of the periodic signal is incomparable Fourier transformation of the non-periodic signal.

In both data sets, we have dealt with missing data by removing all the rows with one or more missing values. For instance, when there are no light curve measurements for a specific filter say g, we can not compute statistical features for this filter. So we drop all the observations that belong to this object from the data. There are many alternatives to this approach such as mean imputation. Mean imputation refers to the scenario in which the missing values are replaced by the mean. The light from some objects might have to go through a lot of interstellar dust before we can detect it. This may arise due to several factors including distance, position in the sky, etc. We can not assume the mean value for

a given light curve since some objects may be reddenned. Because of this, filling in any missing values may be yield inaccurate measurements.

For both data sets, we did not balance the data due to some drawbacks associated with data balancing. In the case where there are outliers (or any bogus objects) in the data, this could reproduce these outliers when oversampling the minority classes. The model might pick random patterns in these objects during the training step. When undersampling the majority classes we might lose useful data, especially since we have limited training samples.

3.1 PLAsTiCC data

The Photometric LSST Astronomical Time-Series Classification Challenge is a software challenge that encourages researchers from all professions to develop automated machine learning models that are capable of processing and classifying large amounts of astronomical data. The data in this challenge was simulated to mimic thousands of transients and variable sources that will be captured by the LSST survey [Allam Jr et al. 2018]. The Legacy Survey of Space and Time is expected to discover tens of thousands of variable sources every night [Malz et al. 2019]. Thus there is a need for automated means to classify these sources since the traditional manual vetting approaches will be inefficient for large data. The PLAsTiCC data was made available to several data science platforms, inviting science enthusiasts to develop efficient machine learning models that classify this data.

The PLAsTiCC data was provided in 11 test sets and a single training set in CSV files. For both training and test sets, time series light curves in 6 astronomical filters and metadata with information like distance, variable types, flux, etc, was provided. In this work, we used 5 filters with large sky coverage in the training data, these are g, r, i, z, and y. The u filter did not have light curve samples for some of the sources, hence we did not include it. The variable classes consist of 14 different transients or variable sources coded as numbers. There are 5731 and 2213990 samples in the training and the test set respectively. In this work, light curve features (described in section 3.3) will be derived for each filter. That is, for each bandpass we will compute statistical features such as mean, skewness,

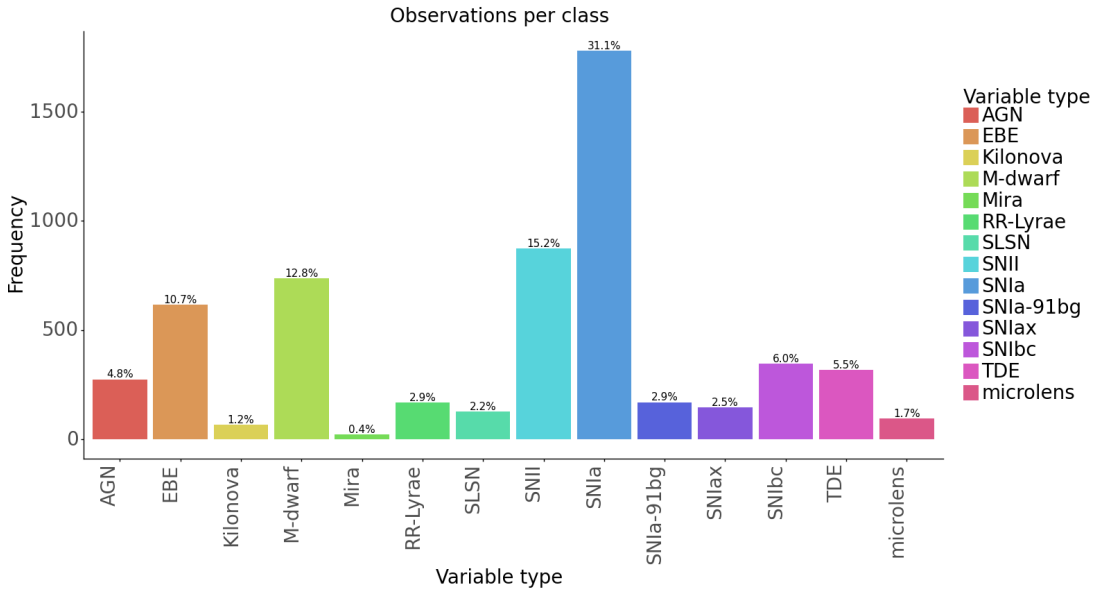
kurtosis, flux ratio, etc. We considered light curves with at least 10 data points. Once these features are obtained, the resulting data is merged with metadata for additional information. The goal of this work is to develop generic machine learning models to classify transients using cross-matched data.

To have a rough idea of how many samples are in each class, we computed the percentage of instances in each class. Table 3.1.1 and figure 3.1.1 shows the number of training samples in each class. There are classes with observations of less than 2% of the training data. The class with the largest number of samples is class Supernova Type Ia (SNIa) making 31.1% of the training data. The Mira variable has the least number of samples in the data. There are 21 samples in this class which constitute 0.4% of the training data. We fitted the machine learning models on the raw data, we did not attempt to balance classes.

Table 3.1.1: Class labels and descriptions on the PLAsTiCC data set

Name	**Class label**	**Number of samples**	**Percentage (%)**
Mira Variable	53	21	0.4
Kilonova (KN)	64	67	1.2
Point source µ-lensing	6	95	1.7
Super Luminous Supernova (SLSN)	95	126	2.2
Supernova Type Ia-x (SNIax)	52	145	2.5
RR Lyrae	92	167	2.9
Supernova Type Ia-91bg (SNIa-91bg)	67	169	2.9
Active galactic nucleus (AGN)	88	274	4.8
Tidal disruption event (TDE)	15	316	5.5
Core-collapse Supernova Type Ibc (SNIbc)	62	346	6.0
Eclipsing binary event (EBE)	16	615	10.7
M-dwarf	65	736	12.8
Core-collapse supernova Type II (SNII)	42	873	15.2
Supernova Type Ia (SNIa)	90	1781	31.1

Figure 3.1.1: Samples in each class



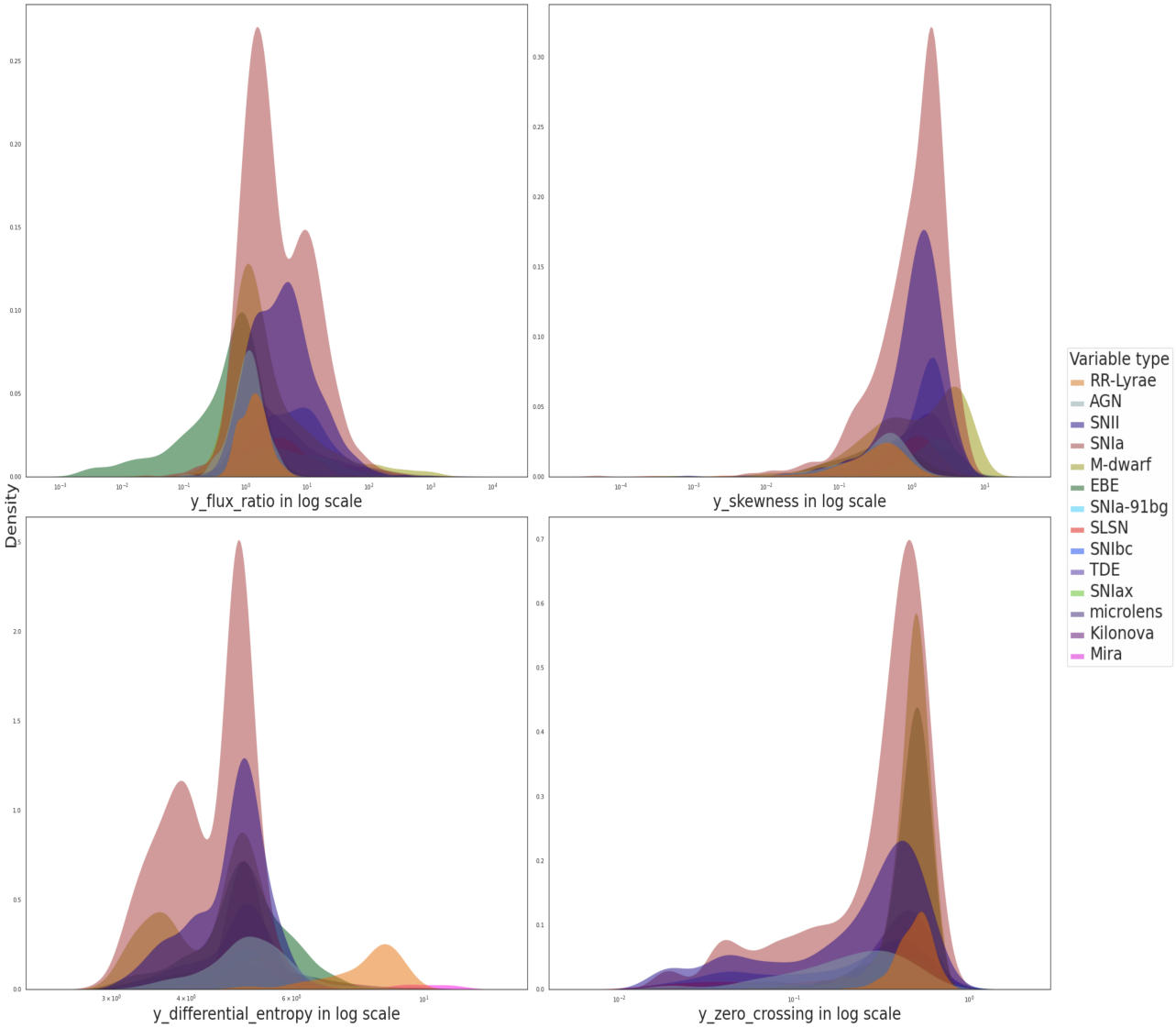
As for the test data, there is no need to balance the data as long as the target class exists in the training data. The test target classes must all exist in training data in order to perform classification.

3.1.1 Class distribution

We plot a kernel density estimation in figure 3.1.2 to have an idea of how the classes are distributed on individual features. The KDE plots for the rest of the features are attached in the appendices (fig: 8.1.1). The features are described in table 3.3.1. Ideally, we want the classes to vary among each other on the individual features; the observations in one class must be similar to each other and dissimilar to observations in a different class. Suppose we have two classes A and B, we want the sample distribution for observations in class A to be different from observations in class B. This allows the machine learning models to easily discern the individual classes. From figure 3.1.2 below, it can be observed that the classes have a varying probability distribution on the majority of the features. The Supernova Type Ia takes a large probability distribution on the majority of the light curve features. The flux at specific filters e.g u, g, r, i, and z are dominated by the

probability distribution for M-dwarf, Eclipsing binaries, and Supernova Type Ia-x. While these classes may dominate in some features, the probability distribution varies enough for a machine learning model to pick variation among the classes.

Figure 3.1.2: Kernel density estimation per feature (see table 3.3.1)



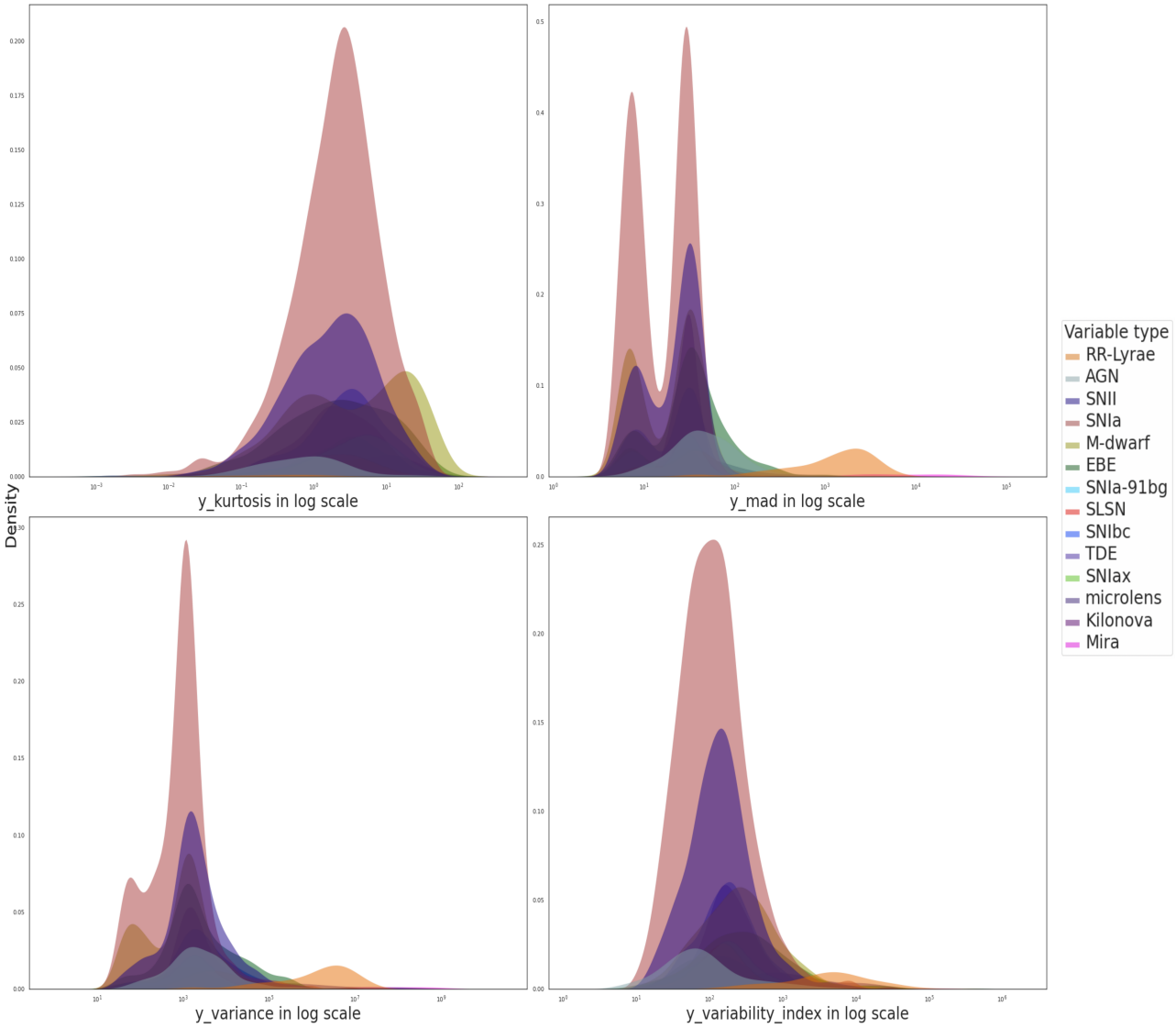


Figure 3.1.2 continued

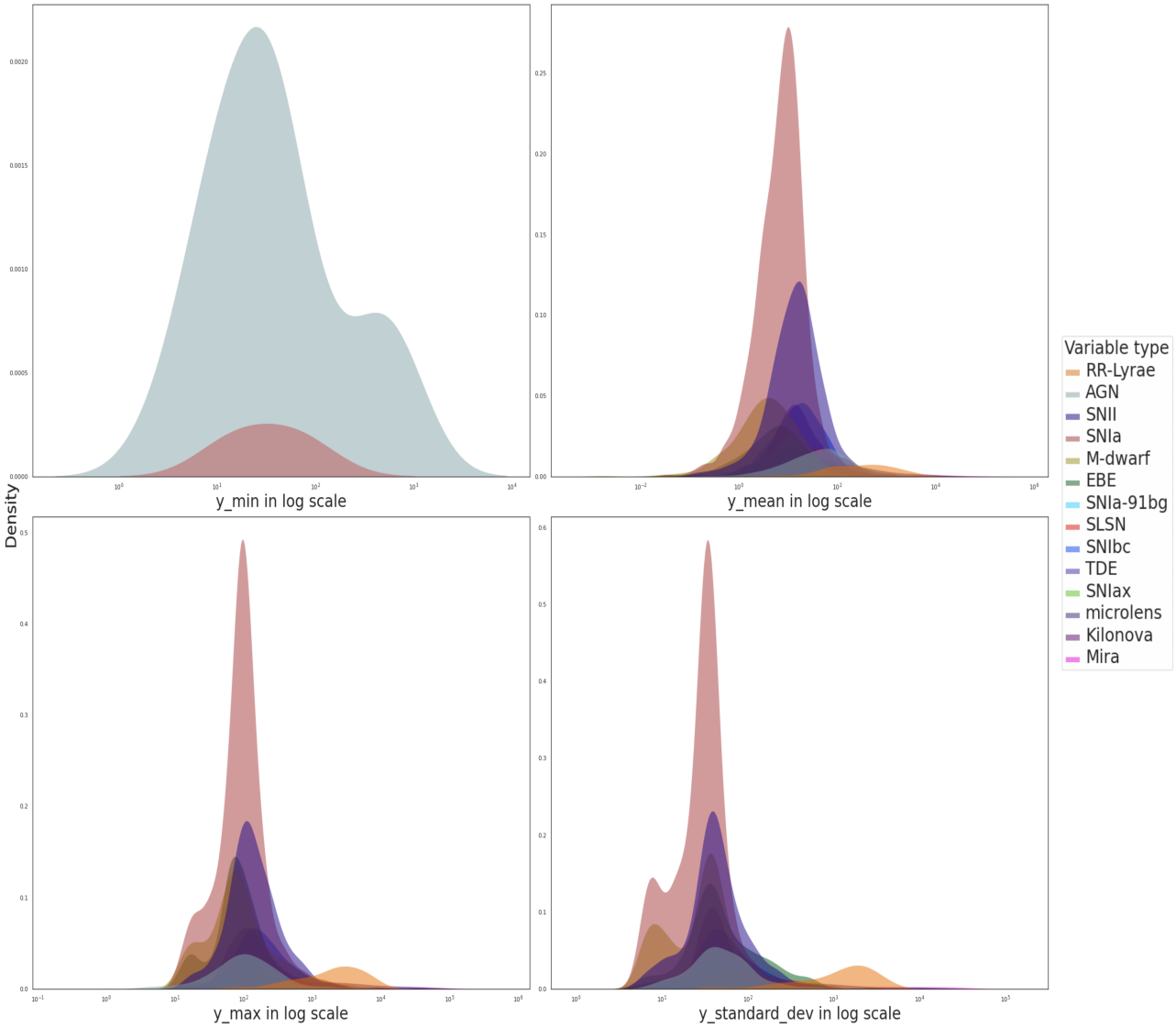


Figure 3.1.2 continued

3.2 MeerLICHT sources

For this data, we will be using the light curve data observed by the MeerLICHT telescope. The MeerLICHT optical telescope is situated in Sutherland, South Africa, have a 65 cm primary mirror and 2.7 square degrees of field of view [Bloemen et al. 2016]. The MeerLICHT data was cross-matched with General Catalogue of Variable Stars [Samus et al. 2001] catalog for class labels. The GCVS

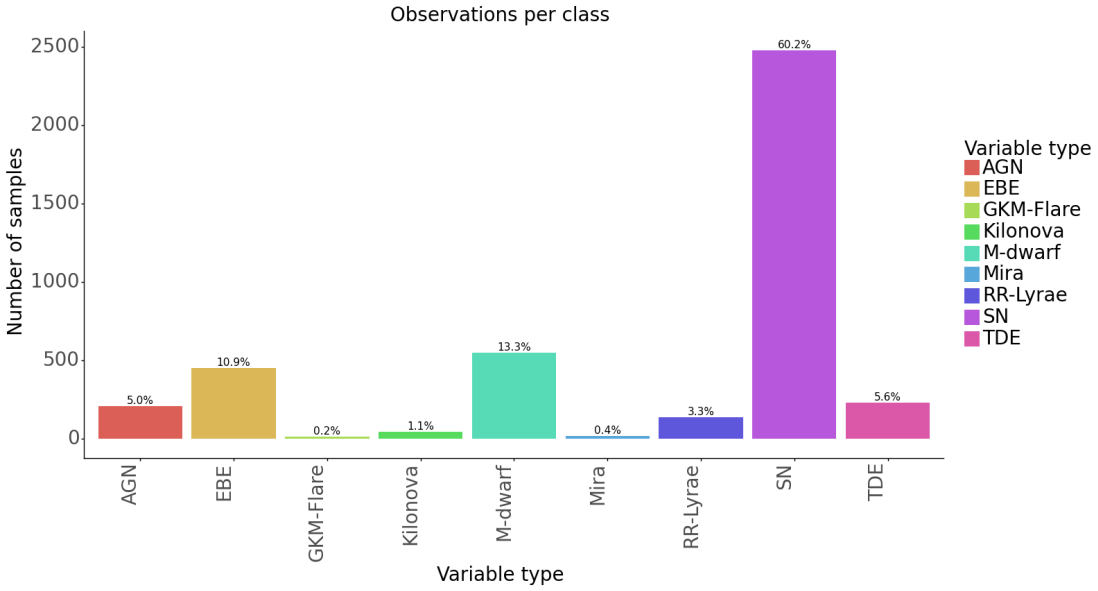
catalog contains labeled variable types which were collected by Moscow variable-star researchers on behalf of the International Astronomical Union since the year 1946 [Samus' et al. 2017]. Because the MeerLICHT data has low sky coverage, we attempted to increase the sample size by combining it with LSST PLAsTiCC simulated data. By combining the data, we mean we stacked the two datasets. We did not cross-match the datasets. We have performed this to test how the combined real and simulated data would affect the performance of the machine learning models. Roughly about 70% of the data is from PLAsTiCC and 30% from MeerLICHT survey. The overall training and testing data had 4119 and 1429 samples respectively. The data consist of the following transient variable sources Supernovae, RR-Lyrae, Eclipsing binaries, Tidal Disruption Events, GKM flaring stars, Active Galactic Nuclei, and M-dwarf stars. Note that the GKM flare stars are specifically late-type flaring stars, whereas the M-dwarfs are generally M-dwarfs that don't have to be flaring.

As with PLAsTiCC data, we check how many samples are in each class. Table 3.2.1 shows the number of observations that are in each variable type. We also plot this information in figure 3.2.1. It can be observed that the Supernovae class comprises 60.2% of the data. While this is a large fraction of the data, it will not have a significant impact on the predictive power of the model since we have aggregated the subclasses for supernovae. The GKM flares, Mira, RR-Lyrae, and Kilonova make less than 5% of the data. Despite the imbalances in the data, we will go ahead and fit the models.

Table 3.2.1: Class labels and descriptions on the MeerLICHT-PLAsTiCC sources

Variable type	**Number of samples**	**Percentage (%)**
Mira	16	0.4
GKM-Flare	10	0.2
Kilonova	44	1.1
RR-Lyrae	135	3.3
AGN	207	5.0
TDE	230	5.6
EBE	448	10.9
M-dwarf	548	13.3
SN	2481	60.2

Figure 3.2.1: Samples in each class



3.3 Time series data and feature definitions

From the light curve data, we can derive statistical features like mean, skewness, kurtosis, dispersion, etc. Using the statistical features we can study how the light curves differ among the classes. With just the derived statistical measures it is easy to cross-match with other catalogs for additional information like distance, velocity, redshift, etc. These metrics can help in training machine learning models. We can feed derived features to machine learning methods without having to construct a sparse metric with columns as time and values as flux or magnitudes. This section presents statistical features derived from light curve data. A subset of the chosen features is similar to those used by Hinners et al. (2018) and Hosenie et al. (2019). These features have shown promising results in the literature. In section 5.3 we perform feature selection to identify which of these features are relevant.

Skewness

Skewness is a statistical measure of the symmetry of a normal distribution. It can be described mathematically as:

$$Skewness = \frac{\sum_i^N \left(X_i - \bar{X}\right)^3}{(N-1) \times \sigma_3} \tag{3.3.1}$$

Here $\bar{X}$ is the mean of the distribution and X_i is the i^{th} sample in distribution X. N is the number of samples in X and σ is the standard deviation of X. The skewness is zero for a symmetric distribution [Cain et al. 2017].

Kurtosis

Kurtosis is a statistical measure of how heavily tailed a distribution is compared to a normal distribution. It is also associated with the shoulder and peakedness of a distribution [Cain et al. 2017]. It is formulated as:

$$Kurt = \frac{\mu_4}{\sigma_4}$$

The kurtosis has large values for a distribution with high peak and small values for a flat distribution [Cain et al. 2017].

Flux ratio

Flux ratio is the ratio of squared mean subtracted flux values above and below the mean flux. For a sinusoidal wave, the flux value is 1 while for a non-sinusoidal wave it is less than 1.

$$FR = \frac{\sum_i^N \left(X_i - \bar{X}\right)^2}{\sum_j^N \left(X_j - \bar{X}\right)^2}$$

Where X_i and X_j are flux values above and below the mean respectively.

Zero-crossing

Zero-crossing measures the rate at which the signal changes the sign from negative to positive and vice versa [Bachu et al. 2008]. In simple terms, it is the number of times the signal passes the zero in a given interval divided by the number of samples in that interval [Gouyon et al. 2000]. It can be used to measure the noisiness of the signal, noisy signals tend to have high zero crossing while low noise signals tend to have low zero crossing.

Zero-crossing can be described mathematically as:

$$Z_n = \sum_{i=-\infty}^{\infty} |sgn[x_i] - sgn[x_{m-1}]|\, w(m-n) \tag{3.3.2}$$

Where

$$sgn[x] = \begin{cases} 1, & \text{if } x \geq 0 \\ -1, & \text{Otherwise} \end{cases}$$

and

$$w(n) = \begin{cases} \frac{1}{2N}, & \text{for } 0 \leq n \leq N-1 \\ 0 & \text{Otherwise} \end{cases}$$

Differential entropy

The differential entropy for a continuous random variable x with probability density function p is given by:

$$H(x) = \int p(x) lnp(x) dx \tag{3.3.3}$$

The Kozachenko-Leonenko entropy is a more convenient estimation of differential entropy. It offers the advantage that it is flexible with respect to the dimensions of the input data [Gautama et al. 2003]. It is given by:

$$H(x) = \sum_{i=1}^{N} ln(Np_i) + ln(2) + C_E \tag{3.3.4}$$

According to Gautama et al. (2003) "N is the number of samples in the data set, p_i is the Euclidean distance of the i-th delay vector to its nearest neighbour, and C_E is the Euler constant". Differential Entropy is used to measure amount of disorder in light curve data [Gautama et al. 2003]. In this work, we use the differential entropy implementation provided by the scipy package.

Shapiro Wilk test

The Shapiro Wilk test is a null hypothesis test that tests whether a given distribution is drawn from a normal distribution [Razali, Wah et al. 2011]. A signal sampled from a random process forms a normal distribution. Using the Shapiro Wilk test one can discern between noise and actual signal. One can train a model on the output of the Shapiro Wilk test, which are test statistic and p-values. An extremely small p-value implies the distribution is sampled from a normal distribution, thus the null hypothesis is rejected.

Scaled flux

We also compute a color-independent feature which is a fraction of the difference between two filters over the difference between two corresponding wavelengths. For filters i and j with fluxes f_i, f_j and wavelengths λ_i, λ_j we calculate:

$$df_\lambda = \frac{f_i - f_j}{\lambda_i - \lambda_j} \tag{3.3.5}$$

We used the maximum flux of the light curve.

Data normalization

In order to find patterns in the data, machine learning techniques often compute statistical measures such as distance, variance, etc. For accurate results, the input features need to be measured on the same scale, since features with large values tend to dominate the solution. This normalization removes the fact that features are measured on different scales. To achieve this, we normalize or standardize the data to have a common scale. In this work, we used min-max normalization. This method normalizes the data to have values between 0 and 1 while retaining the relationship between the features. For vector x with entries $x_{1,2,3,.....,N}$, a normalized value of x is given by:

$$norm(x) = \frac{x_{i \in N} - min(x)}{max(x) - min(x)} \tag{3.3.6}$$

For each column in the data set, we apply equation 3.3.6 to normalize the data.

3.3.1 Data features

Table 3.3.1 below describes features that our models were trained on. We give a checkmark if the feature was included in the models trained either on PLAsTiCC or MeerLICHT data. We leave it blank if the feature was not included. The mathematical descriptions of these features is presented in the previous section (sect 3.3).

Table 3.3.1: Data features (see section 3.3)

Name	Description	PLAsTiCC	MeerLICHT
filterName_flux_ratio	Flux ratio for the named filter	✓	✓
filterName_skewness	Skewness for the named filter	✓	✓
filterName_differential_entropy	Differential entropy for the named filter	✓	✓
filterName_zero_crossing	Zero crossing for the named filter	✓	✓
filterName_kurtosis	Kurtosis for the named filter	✓	✓
filterName_mad	Median flux for the named filter	✓	✓
filterName_variance	Variance for the named filter	✓	✓
filterName_variability_index	Variability index for the named filter	✓	✓
filterName_min	Minimum flux for the named filter	✓	✓
filterName_mean	Average flux for the named filter	✓	✓
filterName_max	Maximum flux for the named filter	✓	✓
filterName_standard_dev	Standard deviation for the named filter	✓	✓
filterName_shapiro_stats	Shapiro statistics for the named filter	✓	✓
filterName_shapiro_pvalue	Shapiro pvalue for the named filter	✓	✓
flux_filterName	Flux at specific filter i.e u, g, r, i, z or q	✓	✓
df_ij	Scaled flux between maximum flux at filter i and j, see equation 3.3.5. Examples are df_gr, df_ri, df_iz, df_zq	✓	✓
hostgal_specz	accurate spec-redshift for small subset	✓	
hostgal_photoz	photometric host-redshift	✓	
hostgal_photoz_err	uncertainty on photometric host-redshift	✓	
distmod	distance modulus computed with hostgal_photoz	✓	
mwebv	Galactic E(B-V) extinction	✓	

4 Machine learning

Before delving into fitting machine learning models, this section provides a brief overview of the machine learning algorithms that will be used to perform classification in this project. These include random forest, artificial neural networks, K-means clustering and hierarchical clustering. Machine Learning describes a set of tools and techniques that are used to teach computers or machines to mimic human intelligence. Machine learning is divided into supervised (sec. 4.1) and unsupervised (sec. 4.3) learning techniques.

4.1 Supervised learning techniques

Supervised learning is a set of machine learning techniques that seek to find a mapping between input and output variables. The mapping can then be applied to previously unseen data to make the prediction [Cunningham et al. 2008]. Supervised learning can be applied in either a classification or regression problem. In the regression problem the output is a continuous variable while in classification the output is qualitative [Mohamed 2017]. An example of a classifier would be a function that classifies a given input as a star, quasar or galaxy while a regression problem would be a function that predicts redshift for a given object. This section describes a high level overview of supervised learning algorithms that will be applied in this work.

4.1.1 Random forest

A random forest classifier is an ensemble model that trains multiple decision trees on randomly selected subsets of the input data and combines the results. A decision tree is a non-parametric supervised learning technique that stratifies the data using features during the training stage. This forms a top-down tree-like structure. Decision trees can be used for both classification and regression. The decision trees recursively partition feature space into j regions such that observations in the jth region are as similar as possible [Gelfand et al. 1989]. In the case of classification, we want to maximize homogeneity in each node. Ideally, classes in the jth node must belong to a single class. In other words, we want the classes to

be as homogeneous or similar as possible in the jth node. Often we use gini and entropy (see section 4.1.2) to measure the homogeneity in a particular node. The process of measuring homogeneity is called node impurity. A decision tree consists of consecutive nodes with a given splitting feature and a splitting condition on that feature. The terminal nodes do not contain a splitting condition, they assign labels to objects for a particular path in the tree. The feature that results in the best separation of the training examples is at the top of the tree i.e best separation will be at the root, the second-best separation will be on the node below the root node, etc. In this way, decision trees can be used as a feature selection technique. An important feature will be located at the root node, feature importance reduces with depth. For example; in figure 4.1.1, x_3 will be more important than the rest of the features since is at the root node.

To make a prediction for a new observation, it passes through a series of splits in which the path of the observation in a tree is decided based on the values of the independent variables [Moisen 2008]. Eventually, the observation is assigned a label at the terminal node. In practice, decision trees are easy to understand and interpret. They do not assume an underlying structure to data. On the other hand, the trees are prone to overfitting as there is no restriction to the size of the tree. Furthermore, they are also sensitive to outliers.

Figure 4.1.1 presents an example of the binary decision tree for data with feature X=$\{x_1, x_2, x_3, ..., x_p\}$ and response variable with classes A, B, and C. Here p is the number of columns in matrix X. This diagram was taken from Jemwa and Aldrich (2005). The values a, b, c are entries in the vectors x_3, x_1, and x_j respectively. In order to make a prediction for a vector $< x_1 = b, x_3 = a, x_j = c >$, the classifier would evaluate to yes in both the root and child (i.e $x_1 \leq b$) node which eventually classifies the instance to be in class A.

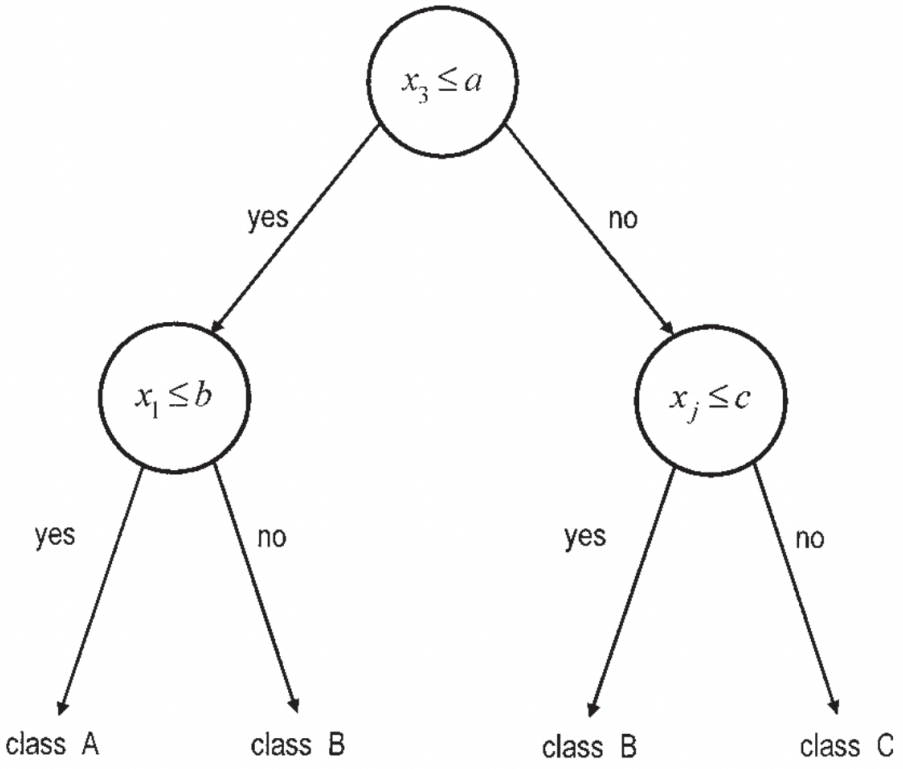


Figure 4.1.1: Typical binary decision tree structure [Jemwa and Aldrich 2005]

Random forest is an ensemble machine learning method that grows and combines multiple decision trees on bootstrapped samples of the training data for each tree [Gislason et al. 2004]. The bootstrapped (i.e resampling by replacement) samples reduce the correlation between individual trees in the forest [Archer and Kimes 2008]. For this reason, a random forest tends to generalize well on previously unseen data, thus improving the performance. Whenever the tree is grown, the random forest technique only considers a subset of the features in the data when looking for the best split. This is the model hyper-parameter that is set by the user, a suitable maximum number of the features to consider at each split can be determined by cross-validation. The commonly used maximum number of features is $\sqrt{p}$ and $log_2(p)$ where p is the total number of features in the training data. There are other hyper-parameter for the random forest such as the number of trees in the forest, maximum depth, node impurity measure, etc. We will be varying these

hyper-parameters in the search for the best model (sect 5).

4.1.2 Node impurity measures

When developing decision trees, we choose features based on how well they split the data on individual nodes. The best feature is chosen such that the observations landing in each of the resulting nodes are as similar as possible. In other words, we want a feature that minimizes heterogeneity in the resulting nodes. This is often referred to as node impurity. The choice of node impurity measure depends on the problem at hand. In the regression task the variance is used while in the classification task the entropy or Gini is used. In this section, we present entropy and Gini since we will be solving a classification problem in this work.

- Gini Index
  Gini coefficient is given by:

$$G(p) = 1 - \sum_j p_j^2$$

  For a homogeneous node the Gini index is zero.

- Entropy
  Entropy is given by:

$$E(p) = -\sum_j p_j ln(p_j)$$

Where p_j is the probability of class j in that node [Sundhari 2011, Muchai and Odongo 2014, Breiman 1996].

4.1.3 Artificial neural networks

An Artificial neural network, often referred to as multilayer perceptron, is a machine learning technique with an architecture that is inspired by the functionality of the human brain. Due to their non-linearity, neural networks can model complex relationships between the input and output data very well compared to traditional machine learning algorithms. Neural networks consist of an input layer, hidden

layers, and an output layer. The data is presented into the input layer, which is then propagated through the hidden layers until it reaches the output layer. The input layer is connected to each neuron in the first hidden layer by weights that transmit information to the succeeding layers. Similarly for the second, third layer, etc until the data reaches the output layer. The output of each hidden layer is an input to the succeeding layer. The hidden layers have an activation function that dictates how the weights will be altered per layer.

The number of hidden layers, number of neurons in each layer, and activation function are called hyper-parameters of the neural network. Some of the most common activation functions are the sigmoid, rectified linear unit function, hyperbolic tan function, and softmax function [Baron 2019]. Neural networks suffer from a lack of interpretability due to their complexity. They are also prone to overfitting because they have a large number of parameters to estimate. Due to a lack of general interpretability, it is not trivial to find the right combination of hyper-parameters for a given application.

A graphical presentation of the neural network is displayed in figure 4.1.2 and was taken from Rafiq et al. (2001) . The raw data is presented to the input layer to hidden layers where the activation is applied and then to the output. The input from the previous layer is linked to each neuron by weights. The weights are linearly combined with the input then the activations are applied then passed to the succeeding layer. The weights are randomly initialized and then are optimized during the learning process. The model makes predictions on the output layer, in the case of the classification there are n nodes in the output and one node in the regression task. Here n is the number of classes or categories in the data.

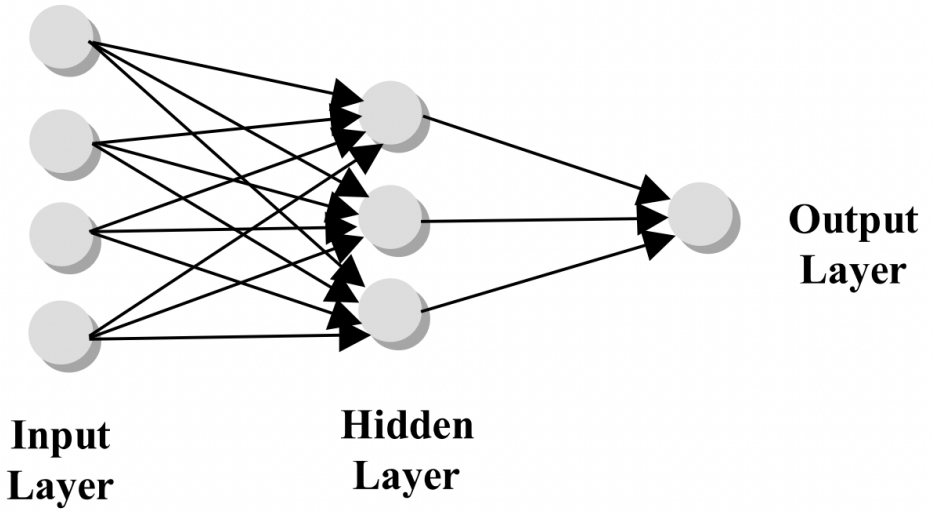


Figure 4.1.2: General neural network architecture [Rafiq et al. 2001]

4.1.4 Types of activation functions

Activation functions in neural networks add the nonlinear transformation to the weighted sum of the input vector. If no activation is used it is called a linear function since the input is mapped to the output without applying any transformation. They strongly influence the capability and performance of the neural network. Hence it is important for a suitable function to be chosen wisely. Some of the popular activation functions are the sigmoid, rectified linear unit, and hyperbolic tangent function. The choice of an activation solely depends on the nature of the problem, especially of the last layer of the network. In a regression task, a linear function must be applied on the last layer while in a classification problem softmax function shall be used. In this section, we briefly describe activation functions and their drawbacks. We mainly focus on the sigmoid, hyperbolic tangent, and rectified linear unit function.

The sigmoid function

The sigmoid is also known as logistic function, is a nonlinear function that outputs values between 0 and 1 for a given input [Tong and Mintram 2010]. In this function, large input values are compressed to 1 while small negative values are compressed to 0. It has an S-shaped profile and is suitable for a layer that outputs probabilities.

The logistic function is given by:

$$g(x) = \frac{1}{1 + e^{-x}}$$

Where x in the input or linear combination of summed weights from the previous layer.

The hyperbolic tangent function

The hyperbolic tangent function is similar to the sigmoid function except that it outputs values between -1 and 1 for a given input [Tong and Mintram 2010]. The tan function differs from the sigmoid function in that it is symmetric around the origin while the sigmoid function only outputs positive values [Sharma et al. 2017]. The tan function is given by:

$$f(x) = \frac{2}{1 + e^{-2x}} - 1$$

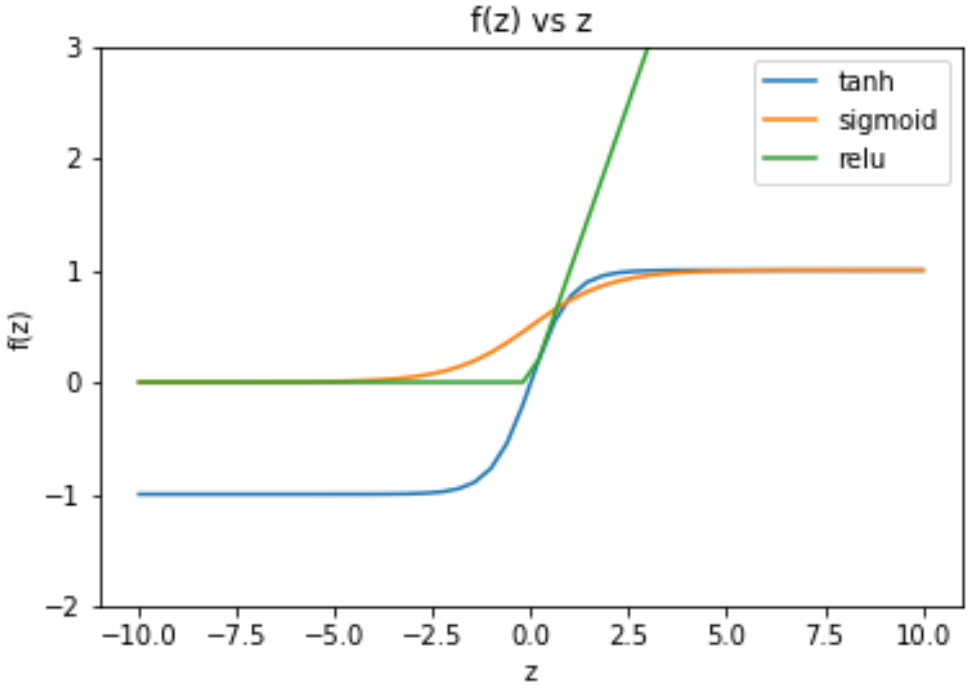
The rectified linear unit function

The rectified linear unit function is a modified linear function that outputs values between 0 and positive infinity. This function has shown to improve the learning of the model and hence performance. It is a default function for most neural networks models. It is given by:

$$h(x) = max(0, x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{Otherwise} \end{cases}$$

Figure 4.1.3 demonstrates the profiles of the sigmoid, tanh, and rectified linear unit function. As stipulated above, it can be observed that, for input z, the sigmoid, tanh, and rectified linear unit function outputs values in the range [0,1], [-1,1], and [0,∞] respectively.

Figure 4.1.3: Activation functions profiles



4.1.5 Optimization algorithms

When developing machine learning methods we need an efficient strategy to minimize a loss function. There are various techniques that available for this. These include Stochastic Gradient Descent, Limited memory Broyden Fletcher Goldfarb Shanno, Adaptive Moment estimation, etc. This section gives a quick overview of these optimization algorithms. We will not give an in-depth description since this is beyond the scope of this project (but see Bottou 2012, Kingma and Ba 2014, Xiao et al. 2008, Morales 2002). Our main focus is on the optimization algorithms for neural networks available on scikit-learn. Scikit-learn is an open-source python package for developing machine learning models [Pedregosa et al. 2011].

- **The Stochastic Gradient Descent**
  The Stochastic Gradient Descent (sgd) is a simplification of the gradient descent algorithm. It seeks to minimize the objective function by computing a partial derivative of the objective function with respect to the weights on

the randomly selected subset of the full data [Bottou 2012]. This process is repeated until the entire data is iterated through. The results are combined later for a final estimate. This tends to be very efficient when the data becomes extremely large.

- **The Adaptive Moment Estimation**
  The Adaptive Moment Estimation (adam) is a stochastic optimization algorithm which is ideal for large-scale parameter search with noisy and non-stationary gradients. It is designed with two independent adaptive learning rates for the first and second-order moments of the gradient [Kingma and Ba 2014]. It offers the advantage that is easy to implement, requires less memory, computational efficiency, etc.

- **The Limited Memory Broyden Fletcher Goldfarb Shanno**
  The Limited Memory Broyden Fletcher Goldfarb Shanno (lbfgs) is a quasi-Newtonian optimization algorithm that approximates the Broyden Fletcher Goldfarb Shanno where the computer memory is restricted. It aims to minimize a differentiable objective function f on unconstrained real-valued vector x. It is suitable for large-scale problems [Xiao et al. 2008, Morales 2002].

4.2 Model evaluation

In the past sections, it was mentioned that some machine learning methods may perform better than others, but how do we compare the algorithms and how do we determine how effective and truthful our comparison is? In this section, we describe the statistical techniques that are used to address this, in order to perform model selection. It is generally thought that more flexible and complex machine learning algorithms can perform relatively well compared to more simple algorithms. However, this is not always the case since some algorithms are data-specific. Hence we can not directly choose a machine learning method that is complex over a simple algorithm. We need some ways to compare the methods. Often, this is carried out by training the methods to be compared on the same data and assessing how accurate it predicts previously unseen data. There are several metrics that are used to quantify model performance. The most common metrics are the confu-

sion matrix, the classification error rate, receiver operating characteristic curve (ROC) for classification task and root mean square error or mean absolute error for regression task.

4.2.1 Confusion matrix

The confusion matrix compares the true response with the prediction by creating a square matrix that consists of the number of correctly classified classes along the diagonal and incorrectly classified along off-diagonals. For a perfect classifier one would expect maximal entries along the diagonal and zero entries off-diagonal. For a binary classifier with true classes say 0 and 1 and predicted classes $1'$ and $0'$ as shown in table 4.2.1, one would have a 2×2 confusion matrix. In this setting the entries a_{11}=TP and a_{22}=TN is the number of correctly classified entries and a_{12}=FP and a_{21}=FN represent the number of incorrectly classified classes. From table 4.2.1, the notation TP stands for "true positive" which refers to instances that the model correctly predicted to be in the positive class. TN stands for "true negative" which are instances that the model correctly predicted to be in the negative class. On the other hand, FP (False Positive) are instances the model incorrectly classified as positive and FN (False Negative) are instances that the model incorrectly predicted to be in the negative class. From the confusion matrix we can derive the following metrics: Accuracy, misclassification error rate, precision and recall [Ahmad et al. 2018]. These metrics are briefly described below.

Table 4.2.1: Confusion matrix for binary classifier

		Predicted value	
		$1'$	$0'$
Actual value	1	TP	FN
	0	FP	TN

- The accuracy measures the number of times the model has made correct predictions. It is computed by calculating the number of correct classifica-

tions divided by the number of observations in the positive class, plus the number of observations in the negative class as shown in equation 4.2.1. The accuracy has the disadvantage that it can not capture the limitations and shortcomings of the model since it is a single scalar quantity. We can account for this by using precision as well.

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{4.2.1}$$

- Precision is computed as the fraction of positive class over the total number of all positive predictions. The precision is given by equation 4.2.2.

$$Precision = \frac{TP}{TP + FP} \tag{4.2.2}$$

- Misclassification error rate measures the rate at which the model incorrectly classified the observations. This is computed as the fraction of incorrect predictions (false positive plus the false negative) over the total number of observations in the data. Misclassification error rate is given by 4.2.3.

$$Misclassification\ error\ rate = \frac{FP + FN}{TP + TN + FN + FP} \tag{4.2.3}$$

- Recall is also known as sensitivity. It is computed as the fraction of positive classes over the total number of actual positive classes. The Recall is given by equation 4.2.4.

$$Recall = \frac{TP}{TP + FN} \tag{4.2.4}$$

- F1-score is also known as F-score. It is defined as a harmonic mean of precision and recall [Huang et al. 2015]. It is given by equation 4.2.5.

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4.2.5}$$

4.2.2 The receiver operating characteristic curve

The metrics presented in section 4.2.1 are handy when evaluating the performance of the classification model on a single probability threshold. An alternative to these metrics is the receiver operating characteristic curve(or ROC curve, section 5.2.1) which plots the trade-off between obtaining the true positive rate at the cost of incurring an additional false positive rate at all possible probability thresholds [Fogarty et al. 2005]. It is used to infer which probability threshold is suitable for a specific problem. For example in a binary task with positive and negative classes, one might want to determine a threshold that strictly classifies objects as negative. This can be performed by assigning all the instances with a probability of less than say 90% to the negative class. 90% is the probability threshold in this example. It can also be used to compare the machine learning models. A perfect classifier is the one with a line passing through the top left corner of the ROC curve. While the model with a straight line curve is equivalent to a random assignment of objects to a class. It is referred to as a non-discriminatory test. We want to keep the false positives relatively small for all the values. If the false positives occur at the same rate as the true positive we would have a straight line and hence a bad fit. In other words, we want a classifier whose true positive rate is as close as possible to 1.

4.3 Unsupervised learning technique

Unsupervised learning techniques are a set of techniques that seek to find patterns, cluster, groups and subgroups in unlabeled data. To mention a few, these techniques include principal component analysis, k-means clustering, hierarchical clustering and self-organizing maps [Miljković 2017, Glielmo et al. 2021]. Unsupervised learning techniques are often used as a pre-processing step before supervised learning is applied. These algorithms can also be useful for clustering, dimension reduction and data visualization. In this project we attempted to identify subclasses in the M type stars. To do this, the k-means and hierarchical clustering algorithms were fitted to the data. This section describes these techniques, starting with k-means in section 4.3.1 and hierarchical clustering in section 4.3.2.

4.3.1 K-means clustering

This is a clustering algorithm that seeks to find k number of clusters in a given data [Likas et al. 2003]. This is performed by randomly initializing k cluster centers and assigning observations to the nearest cluster. Once all the observations are assigned to a cluster, cluster centers are calculated as the mean of the data points in the kth cluster and the observations are assigned to the closest centroid. This process is repeated (assigning observations to the closet centroid and then recalculating the centroid as the mean of the data points in the kth cluster) until the calculated cluster centers do not change or a desired variance is attained [Kodinariya and Makwana 2013]. The k-means algorithm is named after the fact that cluster centers are the average of the data points in the kth cluster. To determine closest data points, the algorithm uses distance metrics such as Euclidean [Kumar et al. 2014]. It is important that the data is measured in similar scale to avoid biasing the clusters on features with large values. As such, the data need to be standardized before it is fed to the algorithm.

While the k-means clustering method is very simple and intuitive, it has some drawbacks. One major drawback is that the user has to know the underlying number of clusters in advance. Even if the data does not have the desired number of clusters, the k-means algorithm solution always contain k clusters. Furthermore, the solution of the k-means method solely rely on the initial randomized cluster centers [Pena et al. 1999].

4.3.2 Hierarchical clustering

Hierarchical clustering is a distance-based clustering algorithm that seeks to find patterns and subgroups in the data [Reddy et al. 2017]. This section describes agglomerative hierarchical clustering, although there are other clustering approaches such as divisive hierarchical clustering [Marinova–Boncheva 2008]. The agglomerative hierarchical clustering stratifies the data into clusters using the bottom-up approach. At first, each data point is its own cluster such that there are N clusters (here N is the number of data points). In the next step, two similar points are merged to form a cluster such that there are N-1 clusters. The process of merging

the closest clusters is repeated until all the observations belong to a single cluster [Zhao and Karypis 2002, Dawyndt et al. 2005]. The distance in which the clusters are merged is recorded and can be used to determine the number of clusters in the data. The hierarchical clustering solutions result in a hierarchy of clusters that can then be visualized with a dendrogram.

Hierarchical clustering introduces a computational challenge in repeatedly computing the distance between clusters at each iteration. Linkage clustering methods were introduced to overcome this issue. To mention a few, these are average linkage, single linkage, complete linkage, and centroid linkage. The average linkage method merges any two clusters that maximize the intercluster average similarity [Charikar et al. 2019]. The single linkage method merges any to cluster with the minimum similarity between all pairs points in the clusters while the complete linkage takes maximum similarity between all pair points in each cluster [Ros and Guillaume 2019]. The centroid linkage measures the similarity between cluster centers. In the centroid linkage method, we do not need to compute similarity between all pair points in each cluster [Jarman 2020]. Unlike the k-means clustering algorithm, in hierarchical clustering, the user does not need to have prior knowledge about the number of clusters in the data. Due to distance computation between each pair of data points, the hierarchical clustering does not scale well when the data becomes extremely large.

5 Results and Discussions

This section presents the results obtained by fitting the random forest and neural network model on PLAsTiCC 5.1 and MeerLICHT 5.2 data. For both data sets, it must be noted that we have fitted both random forest and neural network models on the light curve statistical features described in section 3.3. For each object in the data, we computed statistical features for individual filters ($g, r, i, z,$ and y) such that we have skewness, kurtosis, mean, deferential entropy, etc for all the light curves. In this way, for each bandpass, we have vectors of mean, skewness, kurtosis, etc for all the objects in the data. We then cross-matched these light curve statistical features with metadata for additional information such as distance,

Galactic extinction, scaled flux, etc. Finally, we have 85 features (independent variables X) for PLAsTiCC and 80 features for MeerLICHT data modeling the response variable Y (i.e transient objects classes). We did not feed the light curves directly to the models. The features of each data set are described in table 3.3.1.

5.1 PLAsTiCC

As presented in section 3.1, the PLAsTiCC data was already split into training and test sets. The data had 1 training set and 11 test sets provided by PLAsTiCC challenge developers. All the 11 test sets were combined to form a set of data. For both training and test sets, only light curves with at least 10 data points were considered and light curve features (see section 3.3) were computed for these. By considering only sources with at least 10 data points we lost 2117 and 1278900 objects from the training and test data respectively. We chose 10 data points in order to have enough sample points to compute metrics such as mean, zero-crossing, flux ratio, etc. For example, when computing flux ratio we take a fraction of data points above and below the mean. In the case where there are few data points and are all greater than the mean, we would have division by zero. Choosing 10 data points reduced the chances of this encounter.

Finally, we end up with a training and test set consisting of 5731 and 2213990 samples respectively. Once the data was ready to be fed to machine learning models, random forests and neural network models were developed.

The configurations for each models is presented in table 5.1.1. The neural network consisted of 3 hidden layers each with 120, 98, 56 neurons in the first, second, and last layer respectively. The activation function was set to be a rectified linear unit, while the chosen optimizer is stochastic gradient descent. For the random forest, the model was configured to have 200 trees, the number of features to consider at each split was $log_2(p)$ where p is the number of features. Our model uses entropy to calculate node impurity, a measure of the homogeneity of each node in the grown trees.

Table 5.1.1: Model configurations for PLAsTiCC training data

Method	**Hyper-parameters**	**Train accuracy**
neural network	**Hidden layers** : 3 **Neurons per layer**: (120, 98, 56) **Solver**: stochastic gradient descent **Activation**: relu **Number of iteration**: 10 000	78.206%
Random forest	**Features at each split**: log_2 **Node impurity measure**: entropy **Number of trees**: 200	100%

Table 5.1.2 tabulates metrics computed for each model, namely recall, precision, f1 score for test data, and accuracies. The test accuracy for both models is comparable, it is 65.7% for neural networks and 66.0% for the random forest. While the test accuracy for the random forest is high, there is a large variation between the training and test accuracy. Generally, if a model performs well on the training data, one would expect it to perform well on the test data. However, since we find a large discrepancy between the training and testing accuracy for the random forest model, we can conclude that the random forest model is overfitting the training data. Therefore, it does poorly when generalized to unseen data. A model with a high variation between training (high) and testing (low) accuracy is considered to be overfitting. The difference between the two accuracy scores must be small with both scores being as high as possible. There are many reasons for the model to overfit, these include 1) If the test samples that are chosen during the splitting of data into train and test data are due to noise. The test samples are not full representations of the training data and vice versa. 2) The hyperparameters combination is not good enough to pick the patterns in the data. 3) The size of the data is small, etc. In section 5.1.1 we attempt to improve model performance by conducting a hyperparameter search using cross-validation.

For the neural network model, it can be seen that there is less variation between

training and test accuracy which is 78.2% and 65.7% respectively. The model is generalizing well enough to pick up patterns in the unseen data. For the above-mentioned reasons, in terms of the bias–variance trade-off, we say the neural network has a high bias while the random forest has a low bias. High bias implies that the model is underfitting, the model is simple enough to pick patterns in the data such that it is insensitive to varying observations. On the other hand, low bias implies the model is too complex for estimating objective function, it may be picking up patterns that are due to random chance. Thus this model is overfitting, it is sensitive to varying samples since some of the patterns are due to random chance that may not exist in the new unseen data.

Table 5.1.2: Metrics for PLAsTiCC test data

Method	**recall**	**precision**	**f1-score**	**test accuracy**
Neural network	65.695%	63.645 %	62.663 %	65.695%
Random forest	65.953 %	62.979 %	61.731 %	65.953 %

One potential source of error in classification models is confusion amongst classes when the selected features do not have sufficient discriminating power. To this end, we investigated which classes introduce errors in the predictions by predicting individual classes then evaluated model performance for each class. Table 5.1.3 below, presents the test accuracy per class per method. This presents how likely each model would correctly classify a new sample per class from previously unseen data. It can be observed that both methods are performing poorly when classifying SNIax, SLSN, SNIbc, Kilonova, SNIa-91bg, and TDE with accuracies below 10%. Neither of the two methods was able to correctly classify classes SNIax and SNIa-91bg, with a test accuracy of 0. While the random forest model correctly classified 5.814 % of observations in the Kilonova class, the neural network incorrectly classified all the observations in this class. A potential cause for this could be that since we are doing top-level classification, the subclasses in the training sample might be different from the subclasses in the test data. A good example of this is the RR Lyrae type which consists of RRa, RRb, and RRc subgroups.

These subclasses have different light curves, RRc has a light curve close to sinusoidal while RRa has an asymmetric light curve. If the training data is composed of mostly RRc and test data of RRa, the model may perform poorly on the test data. A solution to this would be to make sure that the subclasses are balanced between the data sets or in the training set. We could not do this since we did not have direct access to the subgroups.

There is some variability in terms of identifying classes among the algorithms. It can be seen that while one method may achieve high accuracy for one class, the other algorithm might penalize this class heavily or equally achieve the same prediction accuracy. The random forest and the neural network model correctly classified 97.9 $\pm$ 2% of samples in M-dwarf class. Both methods also performed equally in classifying the Core-collapse Supernova Type Ibc variable type. The converse can be observed for classes RR Lyrae, Supernova Type Ia, and Point source µ-lensing with respective accuracy of 98%, 93.5% and 75.6% for neural network and 59%, 88.5% and 43% for random forest model. This could be due to the fact that models are sensitive to patterns found in specific variable types. We have not investigated if this is really the case as we are limited to this data. This can be tested on other catalogs.

Table 5.1.3: Test accuracy per class in PLAsTiCC data

Variable type	**Number of samples**	**Neural network**	**Random forest**
Kilonova	86	0.0 %	5.814%
M-dwarf	58968	97.856 %	99.942%
Supernova Type Ia-91bg	25580	0.0 %	0.0%
Point source µ-lensing	775	75.613 %	43.097%
Core-collapse supernova Type II	652227	22.804 %	43.047%
Tidal disruption event	8714	0.195 %	5.199%
Eclipsing binary event	57540	63.994 %	98.318%
Supernova Type Ia-x	40692	0.0 %	0.0%
Mira Variable	855	72.515 %	86.199%
Active galactic nucleus	63650	65.018 %	95.125%
Supernova Type Ia	1054562	93.483 %	88.492%
RR Lyrae	116194	98.996 %	59.024%
Core-collapse Supernova Type Ibc	111813	0.026 %	0.026%
Super Luminous Supernova	22334	7.173 %	0.134%

5.1.1 Cross validated results

It is often the case in machine learning applications that the specific set of hyper-parameters used for a model can influence the outcome. To address this, we performed hyper-parameter tuning to attempt to find the parameters that best classify the data. We apply 5-fold cross-validation [Berrar 2019]. 5-fold cross-validation was chosen for efficient utilization of computational resources otherwise other k-fold values may be used. For each of the previously developed machine learning methods, the models were cross-validated on hyper-parameters that are suspected to contain the optimal solution for the data. The hyper-parameters, as well as their associated mean cross-validation and test accuracy are listed in tables 5.1.4 and 5.1.5 for the neural network and random forest models, respectively.

The neural network model with the highest cross-validation accuracy has 3 hidden layers each with 120, 99, and 40 neurons in layers 1, 2, and 3 respectively. This model has the rectified linear unit activation function and adaptive moment estimation optimizer. While this model seems to be doing well on the cross-validated data with a mean cross-validation accuracy of 75.57%, it is not correctly classifying as many sources as the other models in the test data. It can be observed that model nnm_120_98_56_relu_sgd has a test accuracy of 65.695% which is greater than that of the model with the highest cross-validation accuracy. The reason for this could be that the training samples in model nnm_120_98_56_relu_sgd as similar to those in the test data. It might not do well on the new unseen data set. On the other hand, the best model performed relatively well on newly unseen data set. The best model has a better generalization compared to other models.

Table 5.1.4: Neural network cross-validated results in PLAsTiCC data

Hidden			**activation**	**optimizer**	**cv mean accuracy (%)**	**test accuracy (%)**
layer 1	**layer 2**	**layer 3**				
120	99	49	relu	sgd	73.023	59.933
120	99	44	relu	sgd	72.901	63.051
120	98	56	relu	sgd	73.825	63.812
120	99	60	tanh	sgd	75.012	59.118
120	99	45	tanh	sgd	74.401	55.421
120	97	59	tanh	sgd	74.872	60.77
120	99	48	tanh	sgd	73.983	60.818
100	70	48	logistic	sgd	31.075	47.632
95	89	56	logistic	sgd	31.075	47.632
120	99	40	relu	adam	75.57	53.794
120	100	68	relu	adam	74.785	56.78
120	98	41	relu	adam	75.134	48.194
120	98	21	relu	adam	74.418	54.184
120	98	35	tanh	adam	72.9	52.833
120	98	25	tanh	adam	73.232	47.973
120	100	25	logistic	adam	74.017	55.507
120	97	66	logistic	adam	73.739	47.911
120	99	20	logistic	adam	73.616	56.04
120	99	46	relu	lbfgs	70.859	50.215
120	99	44	relu	lbfgs	70.457	53.199
120	99	38	relu	lbfgs	69.707	56.49
120	97	63	relu	lbfgs	70.161	48.767
120	98	25	tanh	lbfgs	71.783	49.638
120	99	39	tanh	lbfgs	72.045	53.195
120	99	43	logistic	lbfgs	69.724	43.812
120	97	42	logistic	lbfgs	68.538	39.569
120	98	43	logistic	lbfgs	70.579	48.163

Table 5.1.5 presents hyper-parameter tuning results using 5 fold cross-validation for the random forest method. We are varying the hyper-parameters, number of trees, maximum depth, number of features to consider at each split, and node impurity measure. For each iteration, we are measuring mean cross-validation and test accuracy. It can be observed that the majority of the model configurations have mean cross-validation accuracy greater than 70%. The test accuracies for all model configs are comparable with that of the random forest model fitted previously, reaching a maximum of 65.875%. While this test accuracy does not correspond with that of the model with the largest mean cross-validation accuracy, the best model is doing pretty well with the mean cross-validation and test accuracy of 77.699% and 65.116% respectively. This model is configured to have 1000 trees, a maximum depth of 25, and the maximum number of features to consider at each split is 7 features. The node impurity measure is set to be gini. The top 6 best models have the maximum depth, criterion, and maximum feature at split equalling 25, gini, and 7 respectively.

Table 5.1.5: Random forest cross-validated results in PLAstiCC data

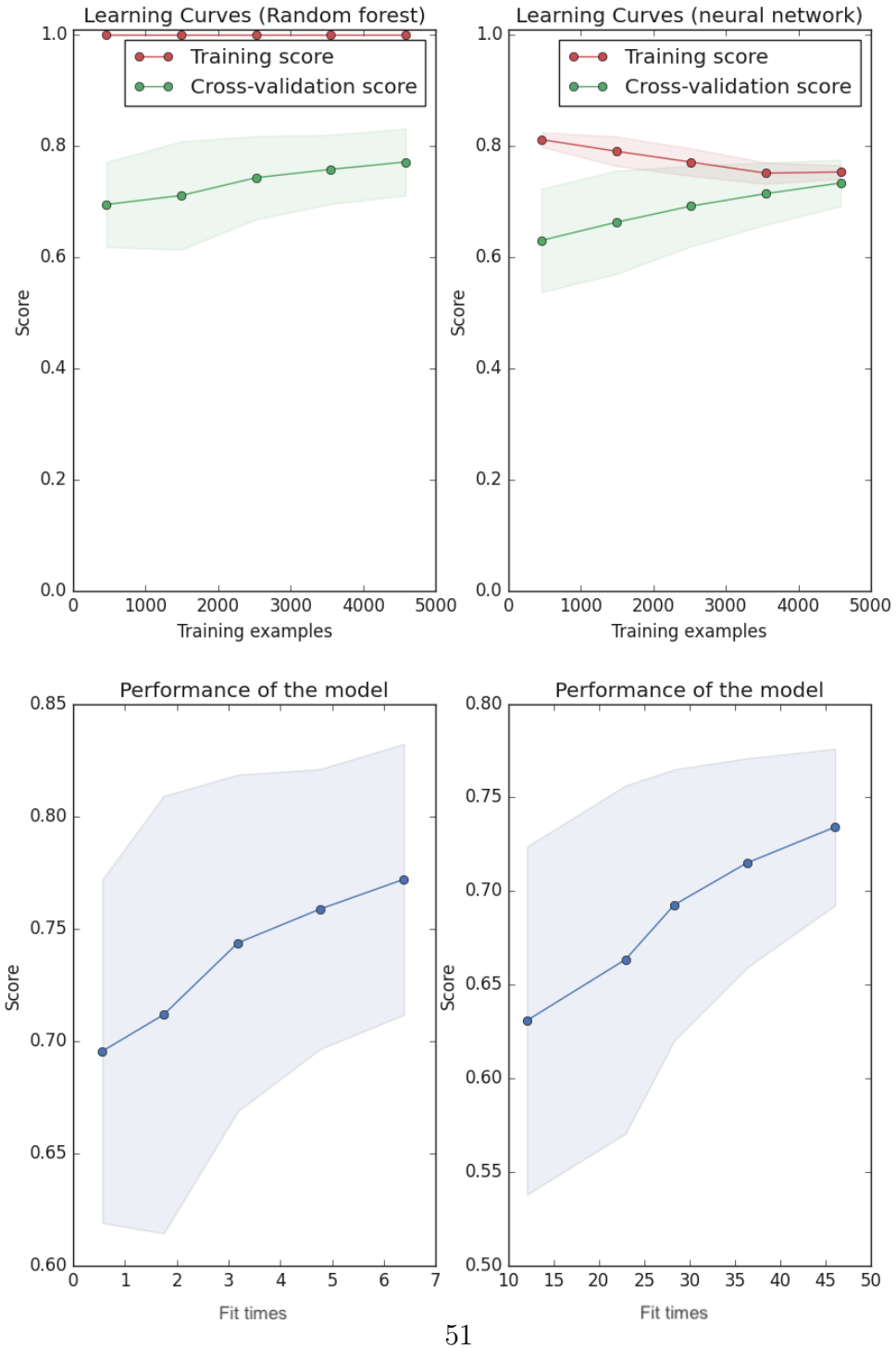
number of trees	**max depth**	**number of features**	**criterion**	**cv mean accuracy (%)**	**test accuracy (%)**
1000	25	7	gini	77.699	65.116
950	25	7	gini	77.699	64.98
900	25	7	gini	77.682	65.624
850	25	7	gini	77.664	65.191
750	25	7	gini	77.542	65.131
700	25	7	gini	77.507	65.126
400	55	7	entropy	77.455	64.577
400	85	7	entropy	77.455	64.577
400	70	7	entropy	77.455	64.577
300	10	13	gini	76.757	65.844
250	10	13	gini	76.687	67.444
750	10	13	gini	76.548	65.082
200	10	13	gini	76.53	66.111
700	10	13	gini	76.513	64.87
650	10	13	gini	76.478	64.608
550	10	13	gini	76.478	64.553
450	10	13	gini	76.425	64.976
600	40	4	gini	76.425	64.714
600	55	4	gini	76.408	64.714
600	70	4	gini	76.408	64.714
600	85	4	gini	76.408	64.714
600	25	4	gini	76.373	64.738
200	25	4	gini	76.268	65.875
500	25	4	gini	76.146	64.793
450	25	4	gini	76.041	65.015

In both methods, there is no variability in the cross-validated results. The cross-validation mean test accuracy for all the fitted hyper-parameter is almost the same.

5.1.2 Learning curves

In an attempt to investigate how the number of samples influences the training and cross-validation accuracy, we plot a learning curve in figure 5.1.1. The shaded areas is the confidence interval with one standard deviation of mean. The 1σ confidence interval is computed by adding or subtracting the one standard deviation from the mean [Donner and Zou 2012]. For the neural network model, it can be observed that there is negative correlation between cross-validation and training accuracy for all the samples. The training accuracy slightly decreases for samples from 0 to 3900 and eventually starts to increase for samples greater than 3900. On the other hand, the training accuracy for the random forest model remains relatively high as the number of samples increases. The cross-validation accuracy increases with the number of samples. For both models, the performance score increases with the fit times. The performance score reaches up to approximately 78% and 74% for random forest and neural network model respectively.

Figure 5.1.1: Learning curve for neural network and random forest algorithm



5.1.3 Modified PLAsTiCC data

Following our results from classifying the original PLAsTiCC data, we concluded that it would be worthwhile to investigate whether the presence of "sub-classes" confuses our algorithms, given the set of features we use to perform the classification. We investigate this by testing whether grouping all types of supernovae (see table 3.1.1) into one aggregate class would improve the performance of the models. We also drop distance information as well as the μ-lensing point sources. Fitting the random forest model to this modified data, we improve the training and test accuracy to 100% and 96.451% respectively and for neural network method the training and test accuracy increase to 92.087% and 96.006% respectively. These results are comparable with those obtained by Hosenie et al. (2019) . Hosenie et al. (2019) found that using a hierarchical classification scheme, where aggregate classes based on general morphologies were classified before searching for sub-classes, provided better results than trying to classify numerous sub-classes from the start.

We also investigate how well the two models classify individual classes on the modified test data. Table 5.1.6 presents the test accuracy per class. The neural network model correctly classified 88.387 % of aggregated supernovae while random forest classified 99.887%. The neural network model seems to be doing well in classifying the eclipsing binary stars. It is with maximum test accuracy of 98.417 % which was 63.994 % in the previous setting. The test accuracy for the number of correctly classified M dwarfs reduced from 97.856 % to 82.485 % for the neural network model while it remained relatively high for the random forest model. Both models performed poorly when classifying Kilonova class, there was no improvement. The reason for this is that there are very few Kilonova in the training data, so there is not enough samples for the model to discriminate between various types of Kilonova. Even though we have reduced the features, the performance of the two models remained relatively high.

Table 5.1.6: Test accuracy per class in the modified PLAsTiCC data

Variable type	**Number of samples**	**Neural network**	**Random forest**
Kilonova	86	0.0 %	0.0%
M-dwarf	58968	82.485 %	99.679%
Supernova	1907208	88.387 %	99.887%
Tidal disruption event	8714	0.218 %	0.184%
Eclipsing binary event	57540	98.417 %	99.597%
Mira Variable	855	68.772 %	87.485%
Active galactic nucleus	63650	85.921 %	89.436%
RR Lyrae	116194	83.089 %	47.388%

The table above does not give us any information about which classes the incorrectly classified instances are mistaken for. To help visualize incorrect predictions we compute a normalized confusion matrix for each model. The normalized confusion matrices of the random forest and neural network model are shown in figure 5.1.2 and 5.1.3 respectively. As mentioned in section 4.2.1, for a perfect classifier we want maximal values along the diagonal of the matrix and zeros off-diagonal. It can be observed that both model makes correct predictions for most of the classes while for some classes the models are making incorrect predictions. The large majority of the Kilonova and Tidal disruption event sources were incorrectly classified, all these classes were incorrectly predicted to be in the aggregated Supernova class. Explosive phenomena such as TDE, Kilonova, and Supernovas tend to have light curves with exponential decline. The models might have trouble discerning these features since statistical features such as Skewness, kurtosis, differential entropy etc for these variable types are similar. In our case, the probability distribution in each feature for Kilonovas and TDEs falls within the distribution of the Supernova class. Since the Supernova covers a wide range of distribution all these classes are assigned to Supernova class. This result just makes sense, and future work might benefit from either putting Kilonova and Tidal disruption events into the aggregate supernova class initially or searching for best feature combination.

Figure 5.1.2: Confusion matrix for random forest algorithm

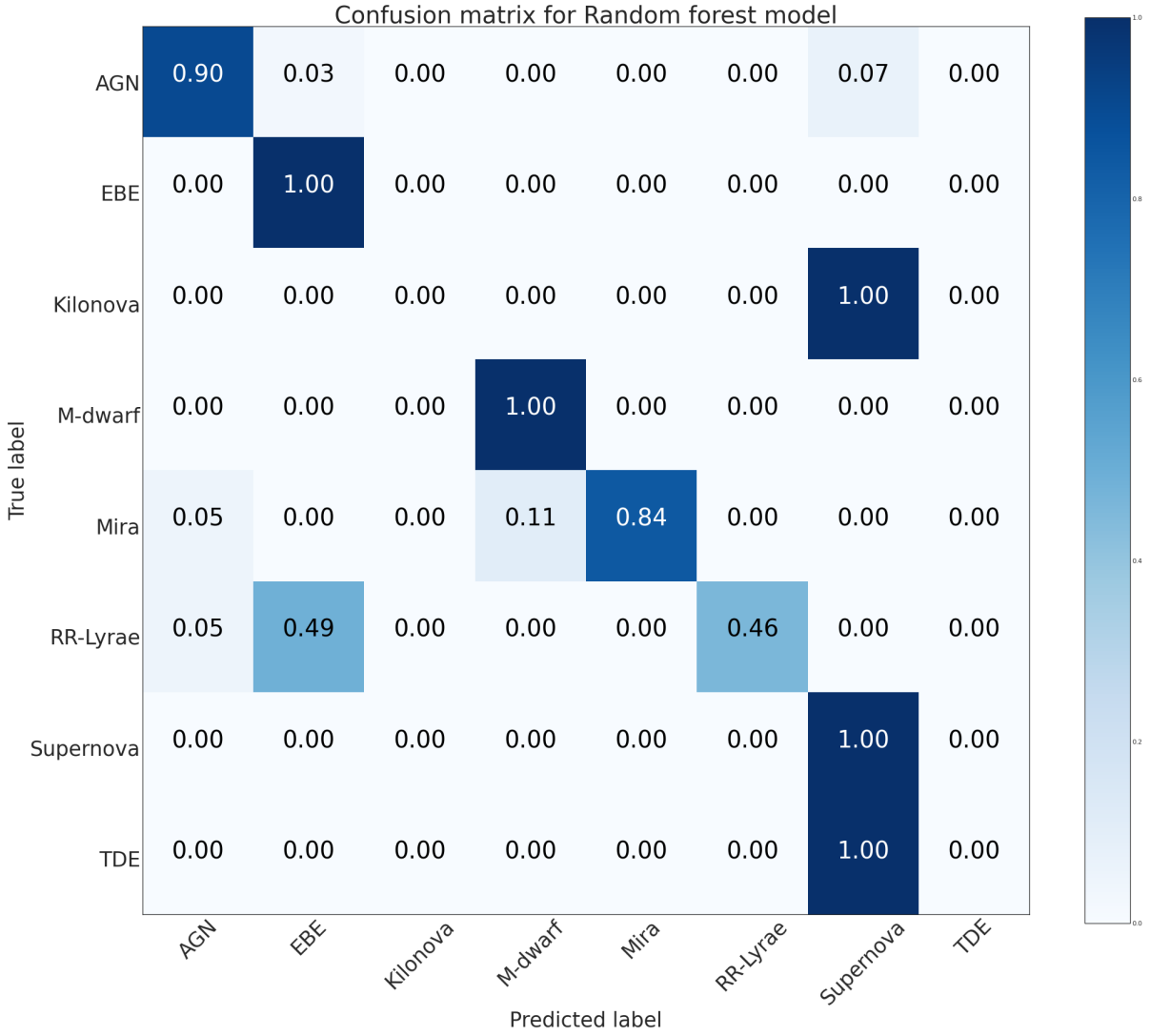
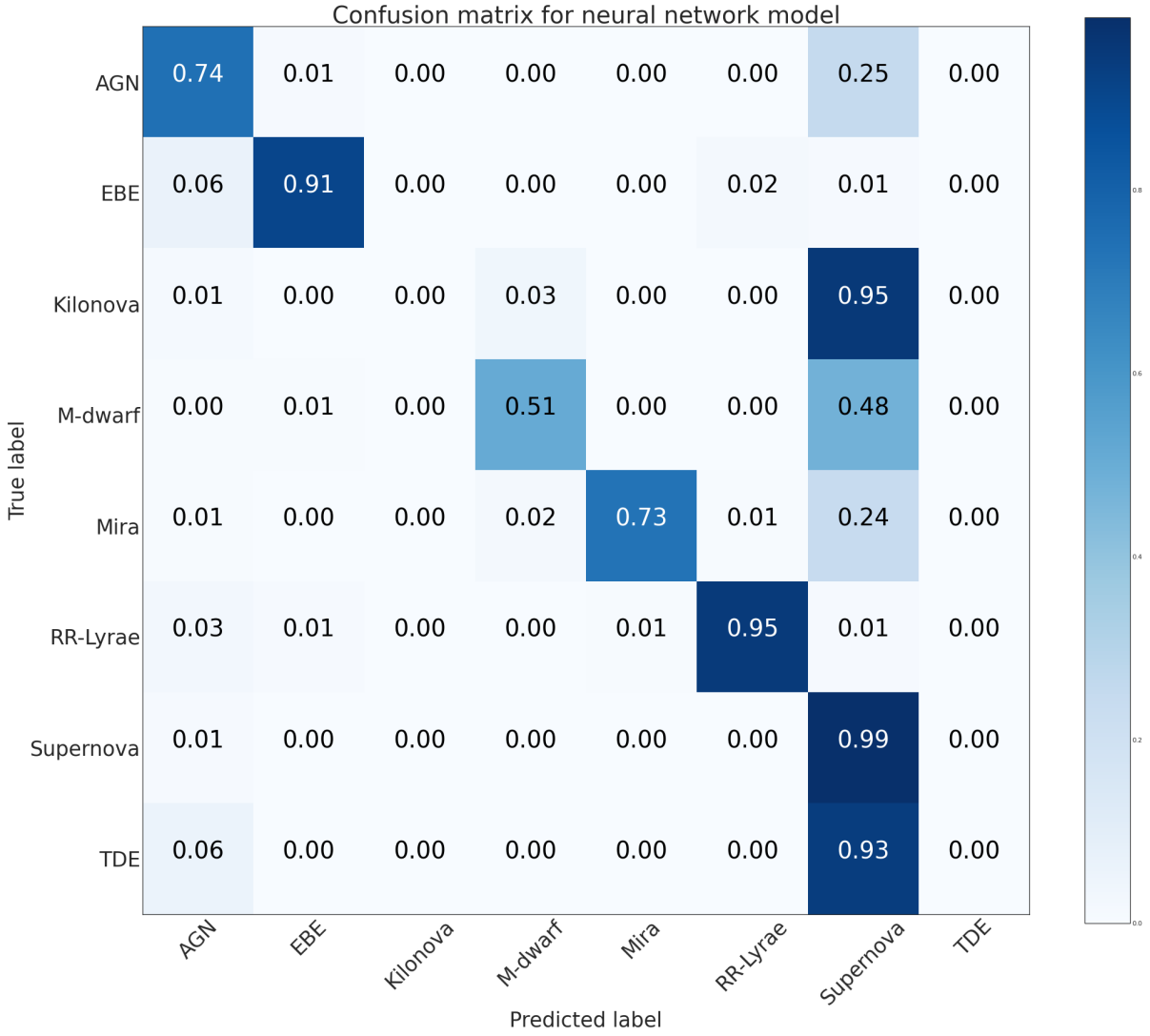


Figure 5.1.3: Confusion matrix for neural network algorithm



For each model, we also compute the logarithmic loss metric provided by scikit-learn [Pedregosa et al. 2011]. The log-loss for the neural network is 0.1534 while the log-loss for the random forest is 0.7667. These results are comparable with those presented by Hložek et al. (2020) in table 2. Hložek et al. (2020) reports some of the initial attempts to classify the PLAsTiCC data.

5.2 MeerLICHT

The random forest and neural network algorithms were applied to the MeerLICHT data. The data is described in section 3.2. This data is a composite of real and simulated data. The real data was observed by MeerLICHT optical telescope, this was cross-matched with General Catalogue of Variable Stars [Samus' et al. 2017, Samus et al. 2001] catalog for class labels. This data was small due to the lack of samples and sky coverage. Some sources do not have observations in all the filters. We combined the MeerLICHT data with LSST PLAsTiCC simulated data to 1) increase the sample size, and 2) test if such a combined data set works well for training models. By combining the data, we mean we stacked the two datasets. We did not cross-match the two datasets. The data was provided as training and test sets with number of samples of 4119 and 1429 respectively. Roughly about 70% of the data is from PLAsTiCC and 30% from MeerLICHT survey. As with the PLAsTiCC data, we fitted multiple random forest and neural network models to the data and varied the hyper-parameters to determine the best model. Varied hyper-parameters for neural network were number of neurons in each layer, activation and solver and for random forest hyper-parameters were number of trees, maximum depth, node impurity metrics and number of features to consider at split. At each step we evaluated the models by computing test and training accuracy. Here we present the model configuration with the highest test accuracy.

The model configuration for each technique is presented in table 5.2.1. The best neural network was fitted with three hidden layers, rectified linear unit activation function, and Limited-memory Broyden optimizer. On the other hand, the best random forest was fitted with 13 maximum features to consider at each split, 100 tree, maximum depth of 10 and entropy for node impurity. The random forest and neural network model test accuracy is 98.041% and 86.564% respectively. Both models correctly classified all the training observation, with training accuracy of 100%.

Table 5.2.1: Model configurations

Method	**Hyper-parameters**	**train accuracy (%)**	**test accuracy (%)**
neural network	**Hidden layers** : 3 **Neurons per layer**: (100, 86, 50) **Solver**: Limited-memory Broyden **Activation**: relu **Number of iteration**: 10 000	100	86.564
Random forest	**Features at each split**: 13 **Node impurity measure**: entropy **Number of trees**: 100 **Maximum depth**: 10	100	98.041

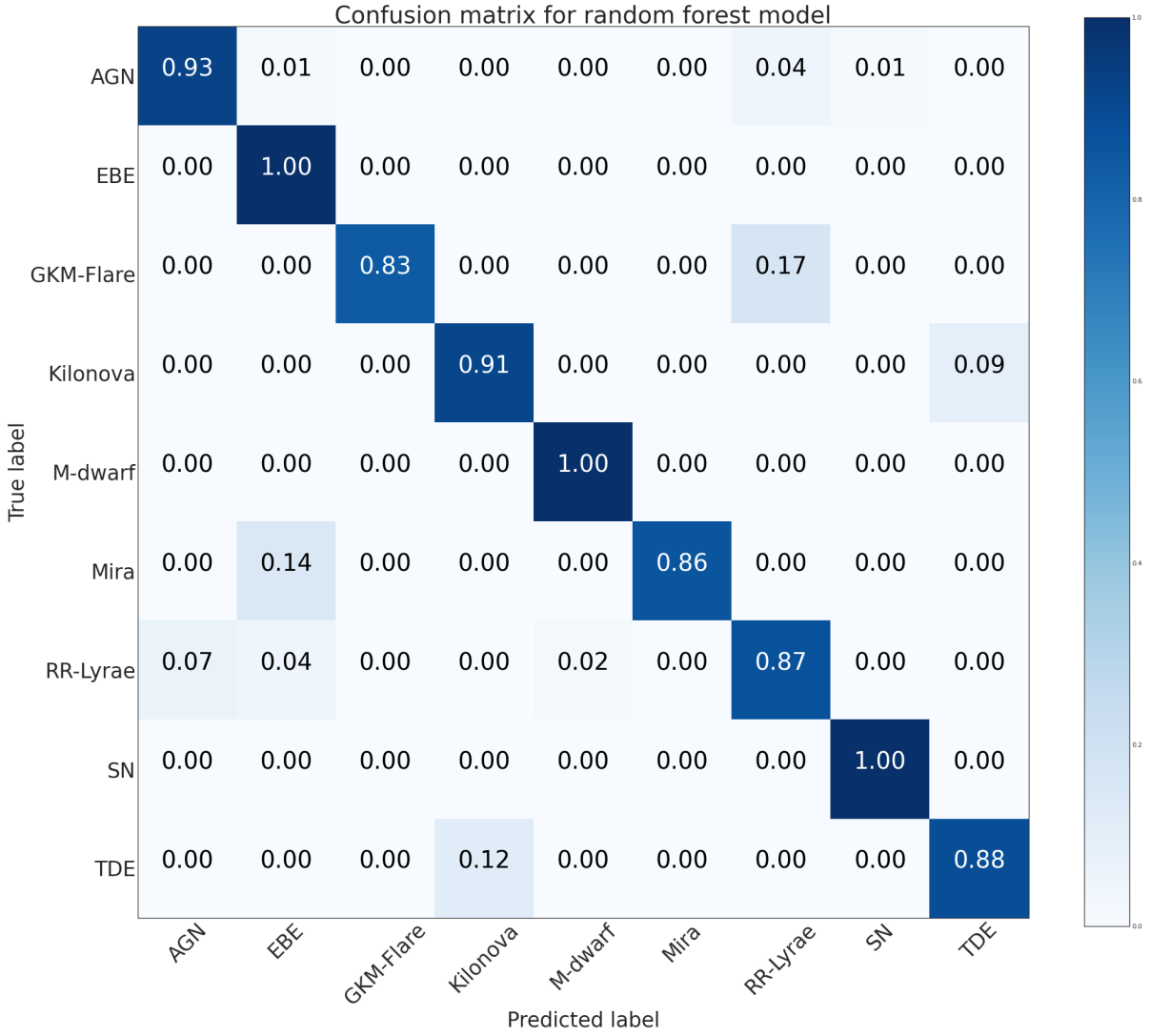
We also investigated which classes the models were most likely to classify correctly. Table 5.2.2 shows the test accuracy per variable type for each model. It can be observed that both models are performing relatively well in classifying M dwarf, Supernovae, AGN, EBE, and RR Lyrae variable stars with test accuracies greater than 80%. The neural network model incorrectly classify all the Kilonovas and TDEs in the test data while the random forest was able to correctly classify 91.304% of Kilonovas and 88.372% of TDEs in the test data. The random forest model is correctly classifying majority of the variable type. The lowest test accuracy for this model is 83.333%.

Table 5.2.2: Accuracy per class

Variable type	**Number of samples**	**Neural network (%)**	**Random forest (%)**
Kilonova	23	0.0	91.304
M-dwarf	188	84.043	100.0
SN	832	97.476	99.639
GKM-Flare	6	66.667	83.333
TDE	86	0.0	88.372
EBE	174	91.954	100.0
Mira	7	71.429	85.714
AGN	67	89.552	92.537
RR-Lyrae	46	84.783	86.957

We plot a normalized confusion matrix to visualize to which class the incorrectly classified instances are assigned to. Figures 5.2.1 and 5.2.2 demonstrate normalized confusion matrices for random forest and neural network models. The majority of the off diagonal elements of the confusion matrix for random forest model are zero while the diagonal elements are close to 1. This indicates that we are making little to no mistakes when classifying sources using this model.

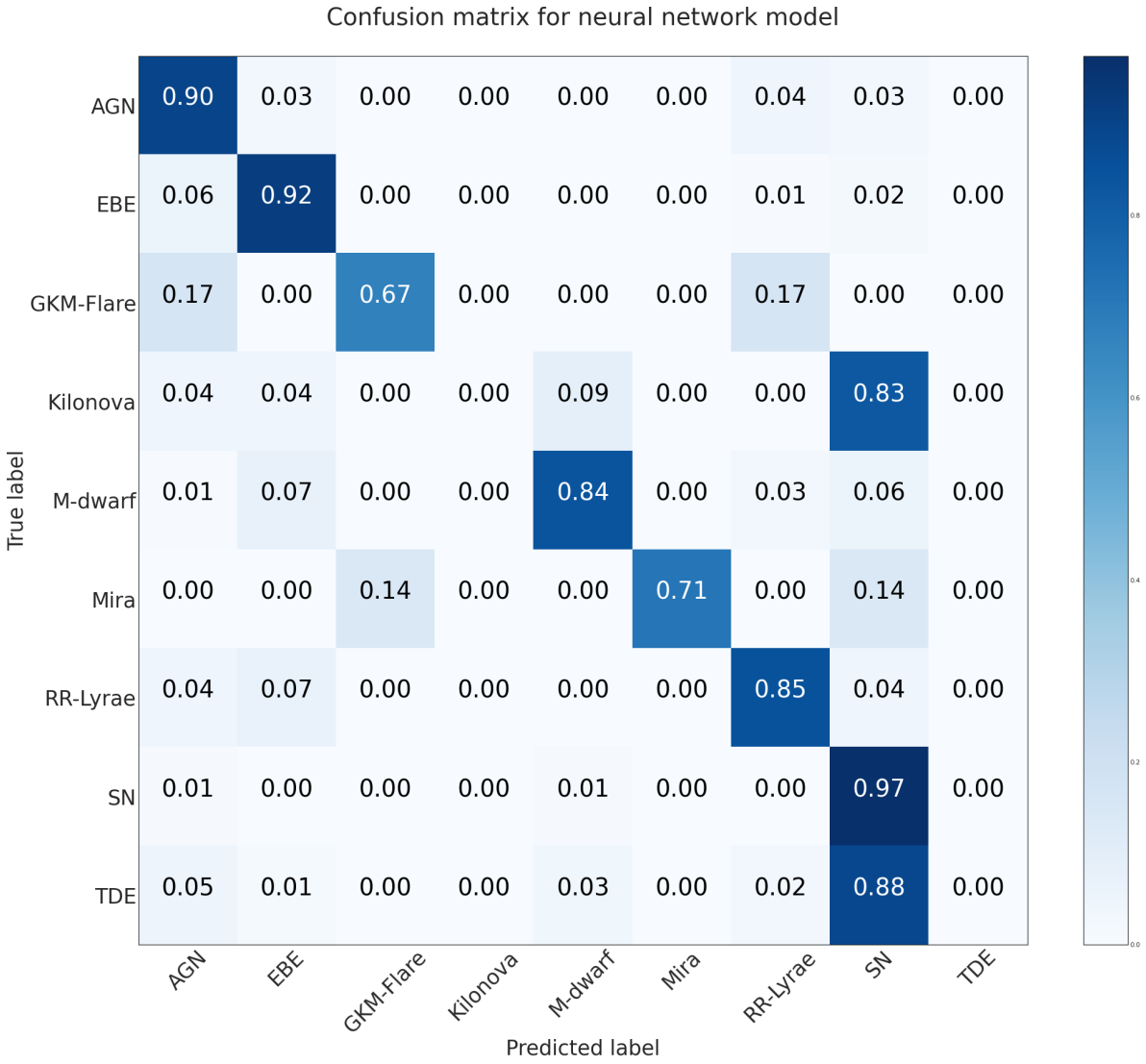
Figure 5.2.1: Confusion matrix for Random forest model



On the other hand, the neural network model was unable to correctly classify Kilonovas and TDEs. The majority of Kilonovas and TDEs instances where incorrectly classified to be in Supernovae class. More than 80% of these classes were incorrectly predicted as Supernova. The neural network model is having difficulty

separating Kilonovas and TDEs from the Supernovas. We have observed this behaviour on the PLAsTiCC data. The potential reason for this is that the probability distribution in each feature for Kilonova and TDE events is embedded within the distribution of the Supernova class. Hence all the majority of these classes are incorrectly classified as Supernovae. The random forest model might have been complex and flexible enough to distinguish between these classes.

Figure 5.2.2: Confusion matrix for Neural network

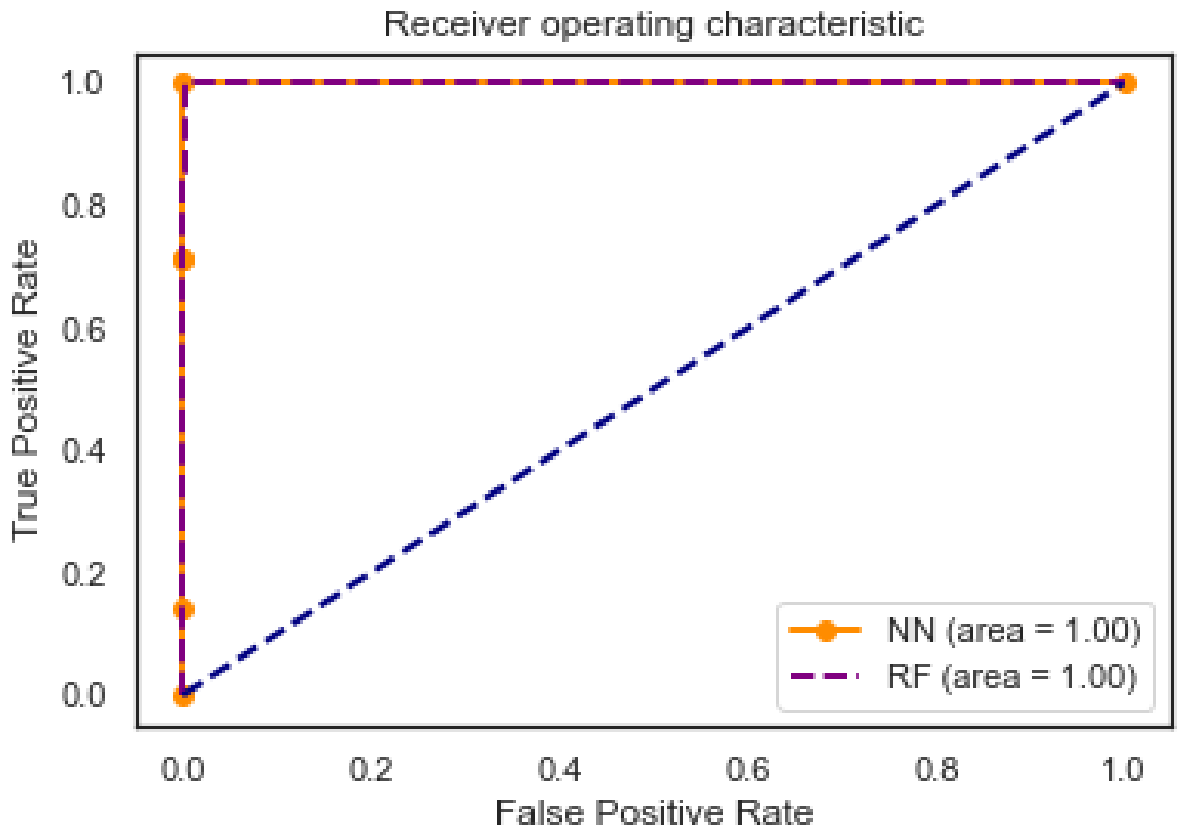


5.2.1 The ROC curve

To investigate how the model would classify the test observations at different probability thresholds, a receiver operating characteristic(ROC) curve was constructed. As outlined in section 4.2.2, the ROC curve provides a visual representation of how the model would classify the test observations at different thresholds. Figure 5.2.3 demonstrates a ROC curve for neural network (orange curve) and random forest (purple curve). It can be observed that both models are performing equally well, both models have the overlapping ROC curve close to the top left corner. Both neural network and random forest have equal area under the curve of 1. Even though the random forest outperformed the neural network on a single threshold, the neural network model would perform equally on an adjusted threshold on this data.

The dashed line corresponds to the ROC curve of a classifier that randomly assigns labels to the instances. It is referred to as a non-discriminatory test. The region below this line has the area under the curve less than or equal to 0.5 [Mathur et al. 2021]. It belongs to a poor classifier.

Figure 5.2.3: ROC curve for neural network and random forest algorithm

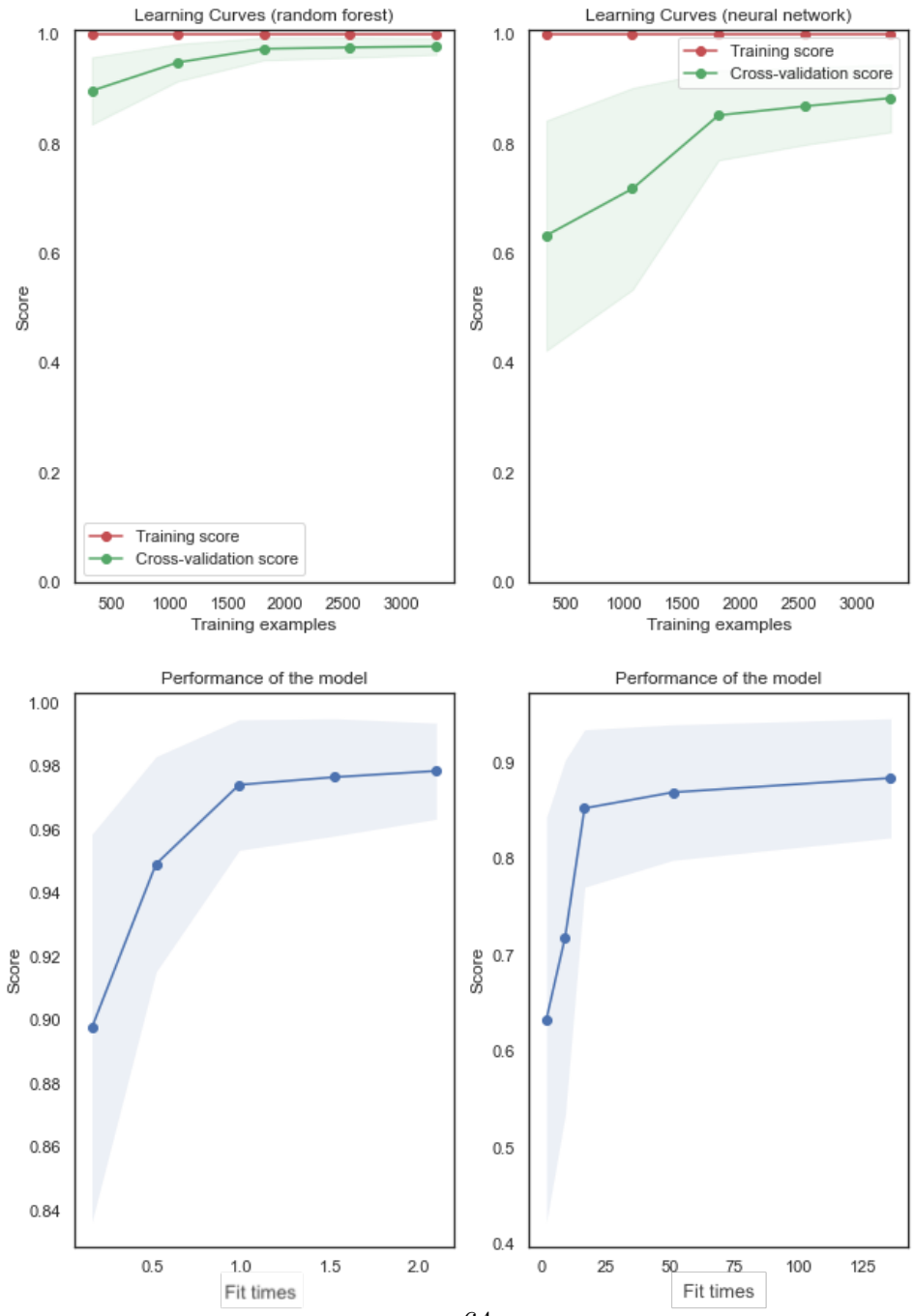


5.2.2 Learning curves

In an attempt to investigate how the number of samples influences the training and cross-validation accuracy, we plotted a learning curve shown in figure 5.2.4. The shaded areas is the confidence interval for the respective scores. For both the neural network and random forest model model, it can be observed that the training accuracy remains high for all the samples. The cross-validation accuracy is increasing with the number of samples. On the other hand, the random forest model attains a cross-validation score of more than 90% for samples size below 500 while neural network starts with low cross-validation score of less than 80%. This means neural network model would need larger sample size to attain cross-validation score equal to that of random forest model. For both models, the

performance score increases logarithmically with the fit times.

Figure 5.2.4: Learning curve for neural network and random forest algorithm in Meerlicht data



5.2.3 Clustering M dwarf stars

After supervised learning models were applied to the data, unsupervised learning methods were fitted to identify clusters of M-type stars. Due to the fact that we do not have prior knowledge about the number of clusters to expect in the data, the number of clusters varied from 2 to 8, and the Silhouette score was used to evaluate how well separated the classes are. The Silhouette score is a metric that measures how dissimilar are the clusters identified by the clustering algorithm. It outputs values between [-1, 1] with 1 meaning the clusters are perfectly separated (good fit) while -1 implies clusters are overlapping(bad fit)[Shahapure and Nicholas 2020].

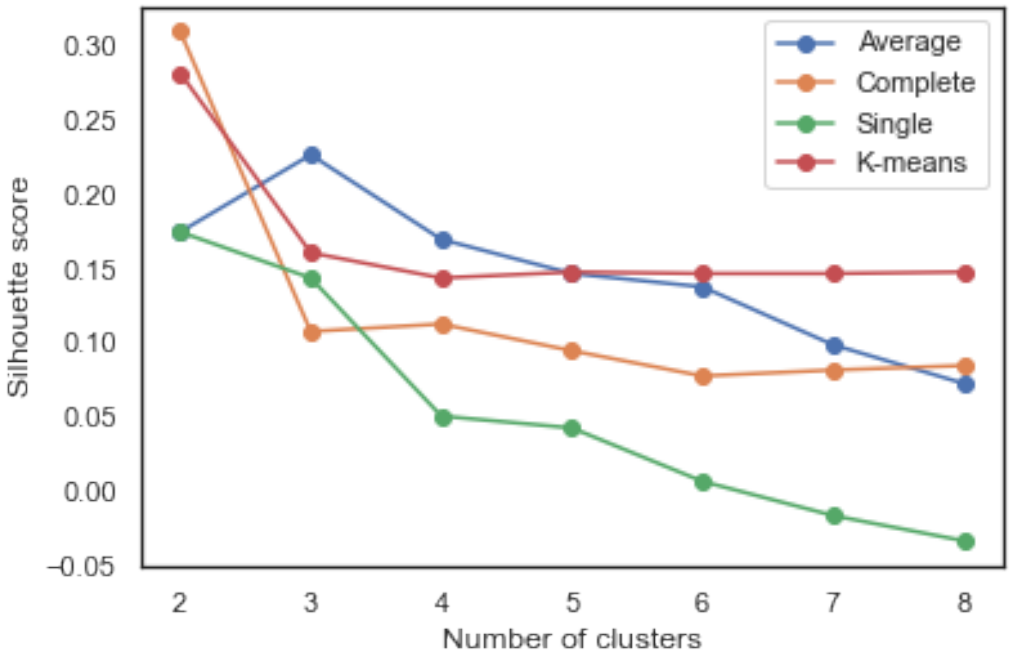
Table 5.2.3 presents the resulting Silhouette score for each method. We fitted k-means and hierarchical clustering linkage methods available on scikit-learn [Pedregosa et al. 2011]. It can be seen that the complete linkage method have the maximum score of 0.31 for 2 clusters. K-means clustering attained second highest score of 0.282 for 2 clusters. The single and average linkage method has attained a equal score of 0.175 for 2 clusters. The single linkage method finds the clusters greater than 7 to be inseparable. These clusters have negative scores. Based on the Silhouette score from K-means and complete linkage Hierarchical clustering method, it is evident that there are 2 clusters in the M dwarf variable sources.

Table 5.2.3: Silhouette score

Number of clusters	**K-means**	**Hierarchical: complete**	**Hierarchical: average**	**Hierarchical: single**
2	0.282	0.31	0.175	0.175
3	0.161	0.108	0.227	0.144
4	0.144	0.113	0.17	0.051
5	0.148	0.095	0.147	0.043
6	0.147	0.078	0.138	0.007
7	0.147	0.082	0.099	-0.016
8	0.148	0.085	0.073	-0.033

In figure 5.2.5 we plot Silhouette scores against number of clusters for the data presented in table 5.2.3. Ideally, we want the Silhouette score to be as large as possible for distinct clusters. As mentioned previously, all clustering methods attain a large score for the number of clusters equal to 2. We conclude, that there are 2 clusters in M type sources. While the clustering techniques do not tell us any information about the types of these subgroups, we suspect these subgroups are variable and non-variable M type stars. We suggest follow-up for these objects to investigate their properties. We were not able to do that on this project due to time constraints.

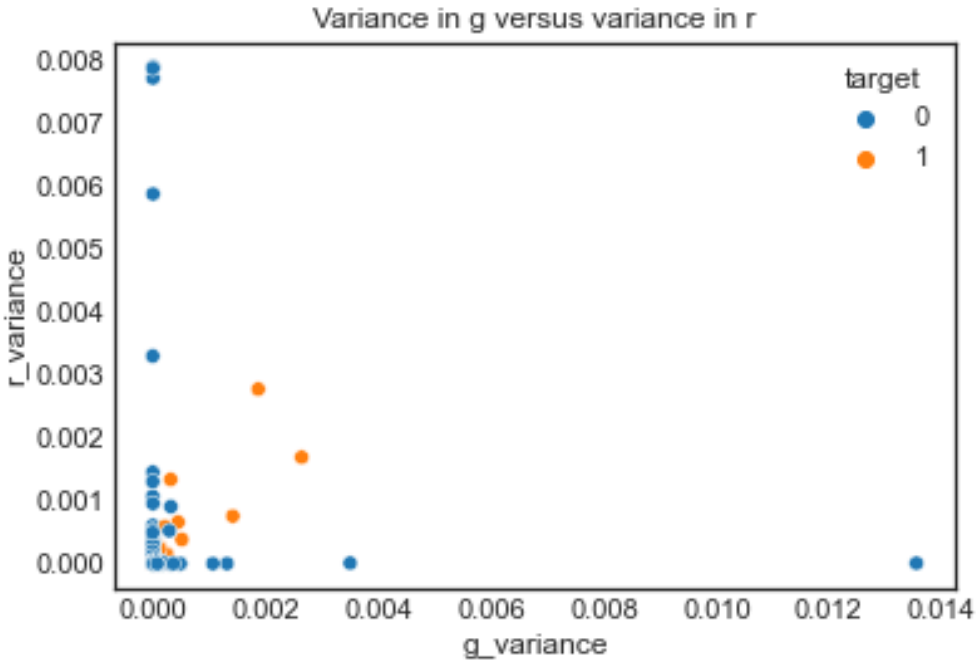
Figure 5.2.5: Silhouette score by number of clusters



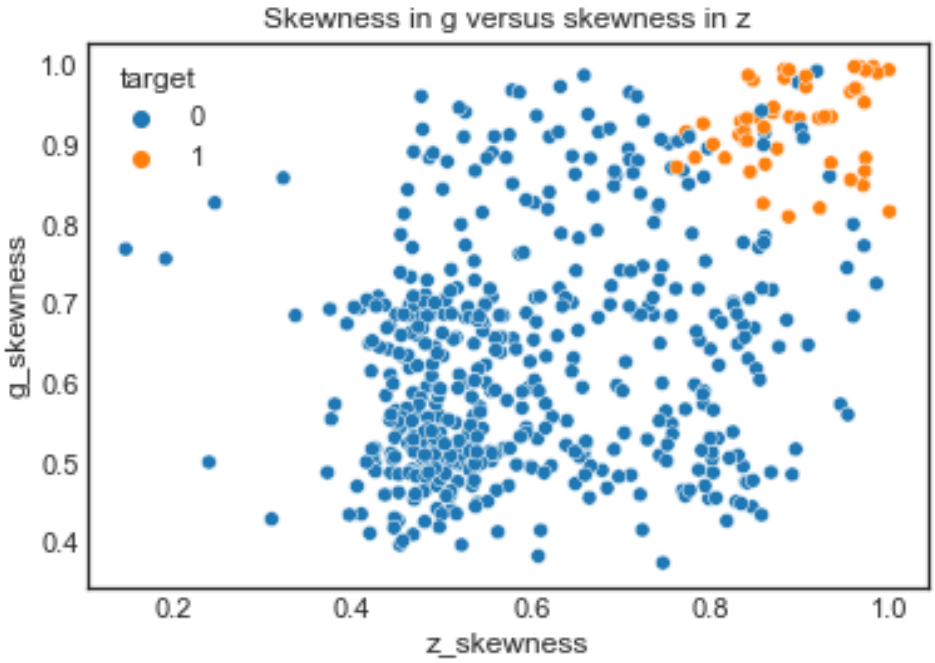
In figure 5.2.6, we are plotting 2D mapping between r and g variance and z and g skewness. It can be observed that in both plots the classes are separable. Figure 5.2.6a, it can be observed that the two classes can be separated by an exponential decision boundary on these features. On the other hand figure 5.2.6b, the instances

of class 1 are grouped towards the top right-hand corner. This illustrates that any instance in this class has an expected skewness value that is approximately between 0.8 and 1 in both filters. The identified classes are separable on various 2D feature combinations.

Figure 5.2.6: 2D plots of clusters



(a) Variance in g versus variance in r



(b) Skewness in g versus skewness in z

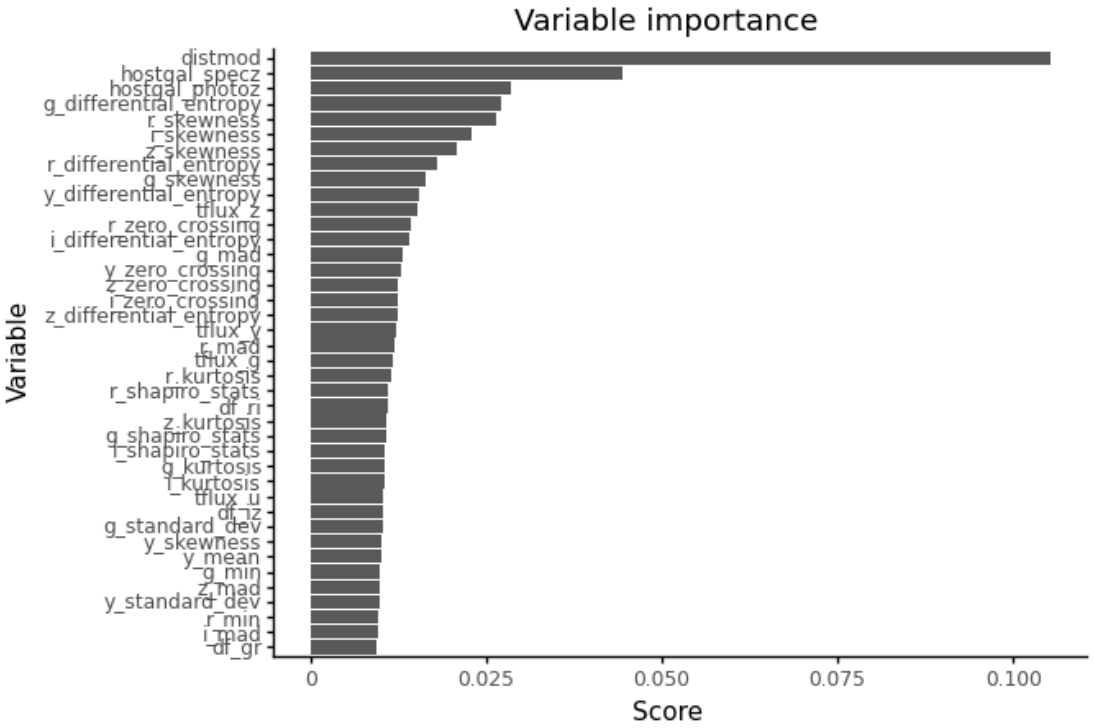
5.3 Feature selection

The models developed in the previous sections were fitted on all the features in the training data. However, there might be features that are not correlated with the response or some might be proxied by another independent variable. That is an independent variable is correlated with one or more other variables (this is detailed at end of this section), so now they appear to be related to the response. However, this relationship is influenced by the strong predictor. Therefore, it is important to include only the features that describe the true behavior of the response.

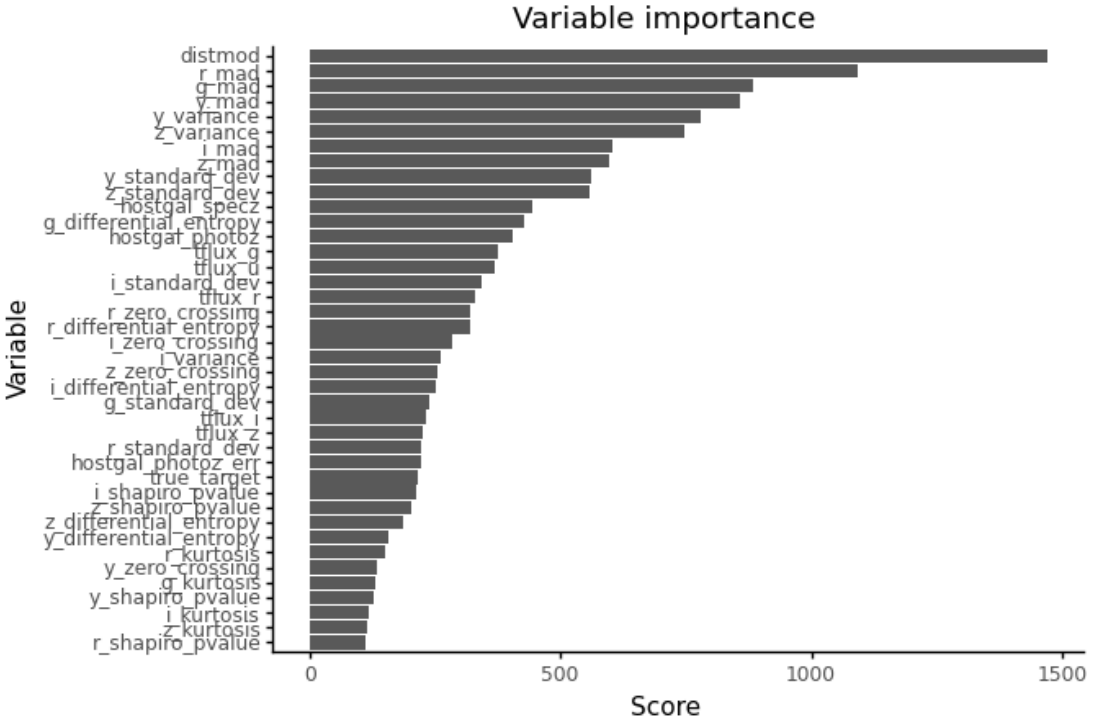
This section applies some of the feature selection techniques to identify features that exhibit a correlation with the response. We apply a univariate selectkbest and extra-trees feature selection methods. The important features identified by the two methods are presented in figure 5.3.1 for PLAsTiCC and figure 5.3.2 for MeerLICHT data set. Selectkbest is a feature selection method that chooses k best features based on univariate statistical tests [Desyani et al. 2020]. It retains k features with the highest score. The extra-trees classifier is a tree-based ensemble method that can be used for feature selection [Naseri et al. 2021]. The extra-trees classifier is similar to the random forest (sect 4.1.1) except that the number of features to consider at each split is randomized [Naseri et al. 2021, Abubaker et al. 2020]. In the extra-trees classifier, the feature importance is computed by taking a normalized contribution of each feature in the forest. The best feature will be the one with the highest contribution.

Figure 5.3.1 displays best 40 features for each aforementioned methods in PLAsTiCC data. Figures 5.3.1a and 5.3.1b are the feature set from extra-trees classifier and selectkbest features respectively for PLAsTiCC data. Both feature selection methods determined distance modulus as the first most important feature. The distance is very important in astronomy, as mentioned in section 2.5, variable stars including RR Lyrae, Cepheids, etc are used as standard candle for measuring distances. So one would expect a strong relationship between the distance and variable type. Derived light curve features like median, kurtosis, skewness, standard deviation, differential entropy appear in the top 5% of the most important feature for both models.

Figure 5.3.1: Feature importance PLAsTiCC data



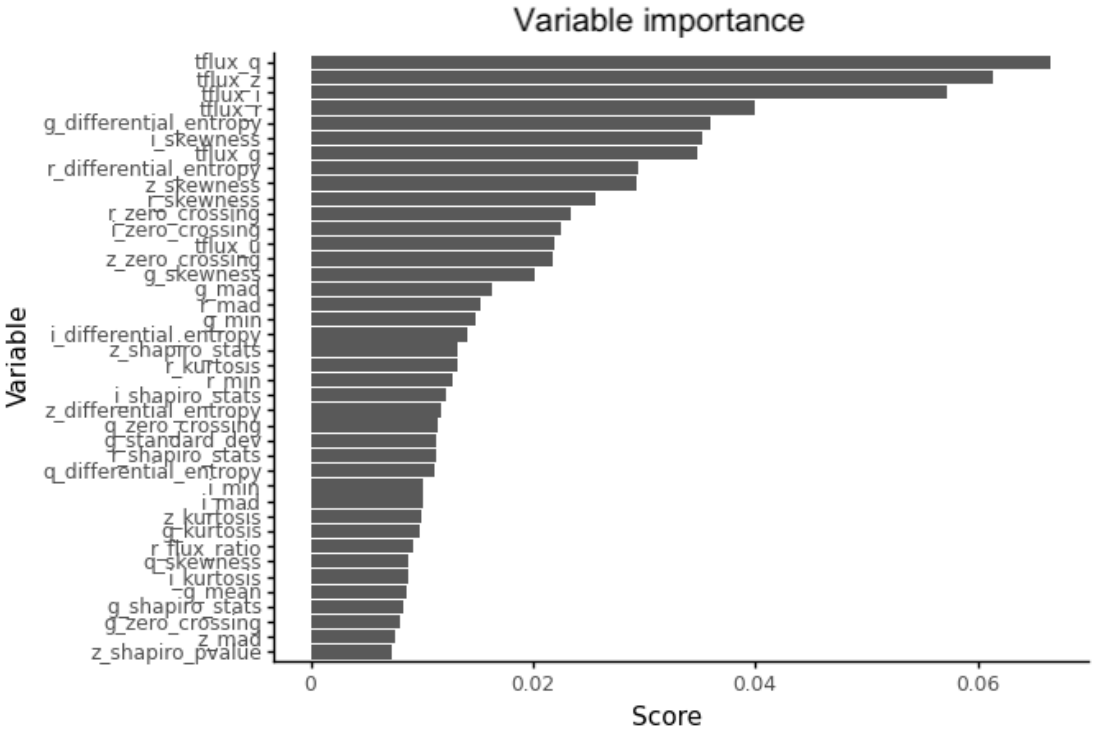
(a) Extra-trees classifier for PLAsTiCC



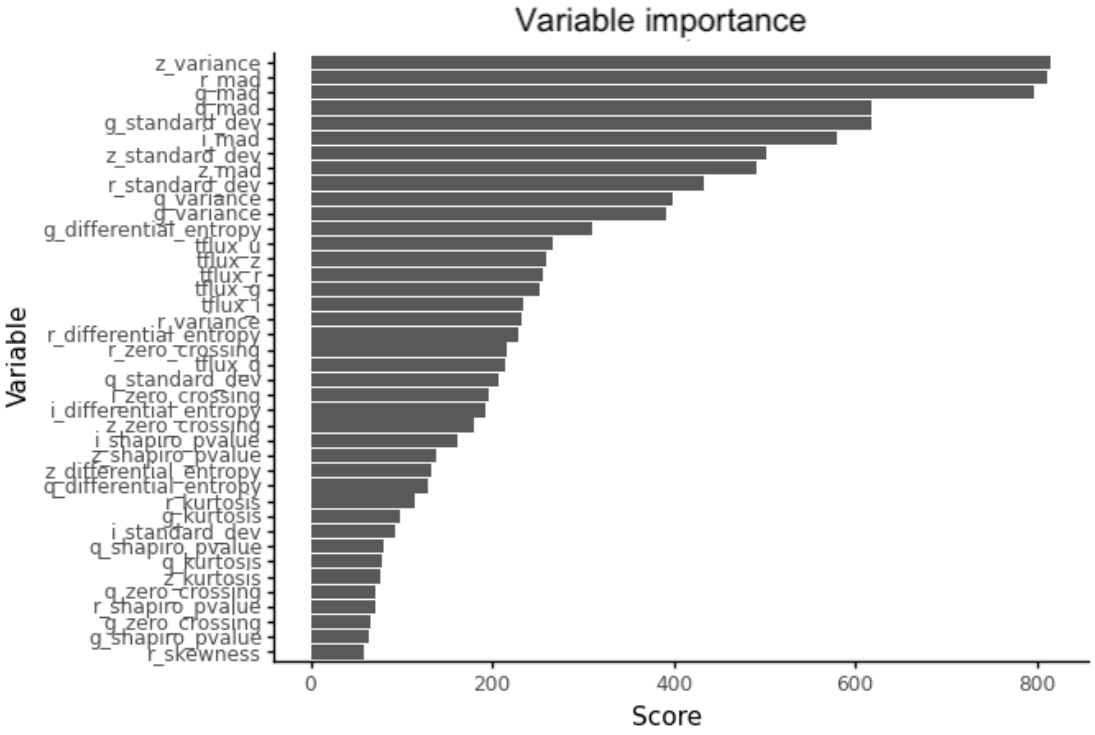
(b) Select k best for PLAsTiCC

Figure 5.3.2 illustrates the top 40 best features in the MeerLICHT data as determined by selectkbest and extra-trees classifier methods in figures 5.3.2a and 5.3.2b respectively. It can be seen that the extra-trees classifier method have retained flux in q, z, i, and r as the top 4 most important features with flux in q being the top 1. On the other hand, the selectkbest method retained variance in the z filter as the top 1 most important predictor in the data set. In both methods, the important features are predominantly statistical features from z filter, e.g z_variance, z_kurtosis, z_standard_dev etc. This might be an indication that we could be able to classify variable sources only using light curve data observed by z filter. Some derived light curve features such as flux ratio, deferential entropy, Shapiro stats, variance, etc for various filters also appear in the to 40 most important features.

Figure 5.3.2: Feature importance MeerLICHT data



(a) Extra tree classifier for MeerLICHT



(b) Select k best for MeerLICHT data

The resulting feature set from the two methods is different for each data set. This might be due to the existence of multicollinearity in the data. Suppose we are given an independent variable X=$\{x_1, x_2, ...x_p\}$ which is modeling a response variable y. Here p is the number on columns in matrix X. Suppose we have a multicollinearity case such that x_1 and x_2 are correlated and x_1 is a strong predictor of y. In the univariate test, x_2 will also appear as a strong predictor of y even though this is not be a true relationship. We say x_1 is a proxy for x_2. Using univariate-based feature selection methods such as selectkbest in this setting would yield misleading results. Both x_1 and x_2 would be identified as important features. However tree-based methods are not affected by multicollinearity since the features are selected based on information gain. When there is no multicollinearity the results of the two methods might be similar.

6 Conclusions and Future work

In this work, we developed generic machine learning models that classify sources in two data sets, real MeerLICHT and simulated LSST PLAsTICC data set. We fitted the random forest and multilayer perceptron neural network. Our findings are as follows:

- For PLAsTICC data, we fitted random forest and neural network models with randomly chosen hyper-parameters on the PLAsTICC data. We obtain a test accuracy of 65.695% and 65.953% for neural network and random forest model respectively. In an attempt to improve model performance, we also performed 5-fold cross-validation. For the neural network model, we obtain maximum mean cross-validation and test accuracy of 73.023% and 59.933% respectively. On the other hand, the best random forest model has the mean cross-validation and test accuracy of 77.699% and 65.116% respectively.
- By aggregating the supernova sub-classes into one class while not using distance information, we improved the test accuracy to 96.006% and 96.451% for neural network and random forest model respectively.
- Similar to PLAsTICC data, we fitted random forest and neural network

models on the MeerLICHT data. We achieve a test accuracy of 86.564% and 98.041% for the neural network and random forest model respectively. We use the ROC curve to investigate how these models would perform on an adjusted probability threshold. We find that with this model configuration, the two models would perform equally.

- By plotting a confusion matrix, we find that the two models are unable to discern a difference between explosive events in both data sets. More than 80% of the Kilonovas and Tidal disruption events were incorrectly classified as Supernovae. This is due to the fact that the probability distribution for Kilonova and TDE events is embedded inside that of Supernova class on the majority of features. This is not surprising since these events are morphologically similar in their light curves.

- We also attempted to find clusters within the M type variable in MeerLICHT data. To do this, we performed clustering using hierarchical and k-means clustering algorithms. Both clustering methods identified 2 distinct clusters within the M type sources. We suspect that these are variable and non-variable M type stars (see sect 5.2.3).

- We have also shown that the model performance could be improved by increasing the number of samples in the data. In both data sets, the random forest model is outperforming the neural network model. There are number of factors that may cause this poor performance. The model hyper-parameters that were chosen may have not contained the solution or we did not have enough samples for the neural network to discern between different classes. One may investigate if adding or reducing complexity on/from the neural network model can yield better results. Due to time constraints, we could not do this.

- We performed feature selection technique using selectkbest and extra tree classifier for both data sets. We obtained different feature sets from each data set. We suspect that this is due to the existence of multicollinearity in the data since univariate feature selection methods are heavily affected by multicollinearity. The tree-based feature selection methods may be preferred

when there is multicollinearity in the data since they are not affected by this.

We have successfully classified the M-dwarf flaring star with the random forest and neural network model. We have achieved best model performance when aggregating all supernova sub-classes into a single class and ignoring distance information on the data. With this setting, the model performance improved significantly. The random forest performs slightly better than the neural network based on the accuracy score.

We have developed our models on statistical features derived from the light curve, while this seems to be effective it does not allow for classification using deep learning methods such as the deep convolutional neural networks. Instead, we can transform the light curve into a two-dimensional image mapping which we can use for deep learning as outlined by Mahabal et al. (2017) . For this, the light curves are required to have relatively the same time measurements to avoid sparsity. Another issue we encountered was dealing with unbalanced classes during the training process. We can use metrics outlined by Boone (2019) to assess the performance of deep learning methods. The metrics account for under-represented classes by penalizing the logarithmic loss components of over-represented classes. Using 2-d transformed data, we can then perform hierarchical classification similar to Hosenie et al. (2019) while optimizing the weighted logarithmic loss function.

7 References

Abareshi, B., Aguilar, J., Ahlen, S., Alam, S., Alexander, D. M. [David M], Alfarsy, R., Allen, L., Prieto, C. A. [C Allende], Alves, O., Ameel, J. et al. (2022). Overview of the instrumentation for the dark energy spectroscopic instrument. *arXiv preprint arXiv:2205.10939*.

Abubaker, H., Ali, A., Shamsuddin, S. M. & Hassan, S. (2020). Exploring permissions in android applications using ensemble-based extra tree feature selection. *Indonesian Journal of Electrical Engineering and Computer Science*, *19*(1), 543–552.

Agrawal, D. C. (2018). Apparent and absolute magnitudes of stars: A simple formula. *World Scientific News*, *96*, 120–133.

Ahmad, I., Basheri, M., Iqbal, M. J. & Rahim, A. (2018). Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection. *IEEE access*, *6*, 33789–33795.

Allam Jr, T., Bahmanyar, A., Biswas, R., Dai, M. [Mi], Galbany, L., Hložek, R. [Renée], Ishida, E. E., Jha, S. W. [Saurabh W], Jones, D. O. [David O], Kessler, R. [Richard] et al. (2018). The photometric lsst astronomical time-series classification challenge (plasticc): Data set. *arXiv preprint arXiv:1810.00001*.

Archer, K. J. & Kimes, R. V. (2008). Empirical characterization of random forest variable importance measures. *Computational statistics & data analysis*, *52*(4), 2249–2260.

Arp, H. C. (1958). The hertzsprung-russell diagram. *Astrophysics ii: Stellar structure/astrophysik ii: Sternaufbau* (pp. 75–133). Springer.

Bachu, R., Kopparthi, S., Adapa, B. & Barkana, B. (2008). Separation of voiced and unvoiced using zero crossing rate and energy of the speech signal. *American Society for Engineering Education (ASEE) zone conference proceedings*, 1–7.

Barbara, N. H., Bedding, T. R., Fulcher, B. D., Murphy, S. J. & Van Reeth, T. (2022). Classifying kepler light curves for 12,000 a and f stars using supervised feature-based machine learning. *Monthly Notices of the Royal Astronomical Society*.

Baron, D. (2019). Machine learning in astronomy: A practical overview. *arXiv preprint arXiv:1904.07248*.

Bassi, S., Sharma, K. & Gomekar, A. (2021). Learning to classify variable stars light curves using long short term memory network. *Frontiers in Astronomy and Space Sciences*, *8*, 168.

Beckmann, V. & Shrader, C. (2013). *Active galactic nuclei*. John Wiley & Sons.

Bellm, E. C., Kulkarni, S. R., Graham, M. J., Dekany, R., Smith, R. M., Riddle, R., Masci, F. J., Helou, G., Prince, T. A., Adams, S. M. et al. (2018). The zwicky transient facility: System overview, performance, and first results. *Publications of the Astronomical Society of the Pacific*, *131*(995), 018002.

Berrar, D. (2019). Cross-validation.

Bloemen, S., Groot, P., Woudt, P., Wolt, M. K., McBride, V., Nelemans, G., Körding, E., Pretorius, M. L., Roelfsema, R., Bettonvil, F., Balster, H., Bakker, R., Dolron, P., van Elteren, A., Elswijk, E., Engels, A., Fender, R., Fokker, M., de Haan, M., ... Sybilski, P. W. (2016). MeerLICHT and BlackGEM: custom-built telescopes to detect faint optical transients. In H. J. Hall, R. Gilmozzi & H. K. Marshall (Eds.), *Ground-based and airborne telescopes vi* (pp. 2118–2126). SPIE. https://doi.org/10.1117/12.2232522

Boone, K. (2019). Avocado: Photometric classification of astronomical transients with gaussian process augmentation. *The Astronomical Journal*.

Bottou, L. (2012). Stochastic gradient descent tricks. *Neural networks: Tricks of the trade* (pp. 421–436). Springer.

Breiman, L. (1996). Some properties of splitting criteria. *Machine Learning*, *24*(1), 41–47.

Brodie, J. P., Romanowsky, A. J., Strader, J., Forbes, D. A., Foster, C., Jennings, Z. G., Pastorello, N., Pota, V., Usher, C., Blom, C. et al. (2014). The sages legacy unifying globulars and galaxies survey (sluggs): Sample definition, methods, and initial results. *The Astrophysical Journal*, *796*(1), 52.

Burhanudin, U., Maund, J., Killestein, T., Ackley, K., Dyer, M., Lyman, J., Ulaczyk, K., Cutter, R., Mong, Y., Steeghs, D. et al. (2021). Light-curve classification with recurrent neural networks for goto: Dealing with imbalanced data. *Monthly Notices of the Royal Astronomical Society*, *505*(3), 4345–4361.

Cain, M. K., Zhang, Z. & Yuan, K.-H. (2017). Univariate and multivariate skewness and kurtosis for measuring nonnormality: Prevalence, influence and estimation. *Behavior research methods*, *49*(5), 1716–1735.

Castelaz, M. W., Luttermoser, D. G., Caton, D. B. & Piontek, R. A. (2000). Phase-dependent spectroscopy of mira variable stars. *The Astronomical Journal*.

Catelan, M., Pritzl, B. J. & Smith, H. A. (2004). The rr lyrae period-luminosity relation. i. theoretical calibration. *The Astrophysical Journal Supplement Series*, *154*(2), 633.

Charikar, M., Chatziafratis, V. & Niazadeh, R. (2019). Hierarchical clustering better than average-linkage. *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2291–2304.

Cherchneff, I. (2009). Dust formation in massive stars and their explosive ends. *arXiv preprint arXiv:0909.0164*.

Collaboration, D., Aghamousa, A., Aguilar, J. [Jessica], Ahlen, S. [Steve], Alam, S., Allen, L. E., Prieto, C. A. [Carlos Allende], Annis, J., Bailey, S., Balland, C. et al. (2016). The desi experiment part i: Science, targeting, and survey design.

Colless, M., Dalton, G., Maddox, S., Sutherland, W., Norberg, P., Cole, S., Bland-Hawthorn, J., Bridges, T., Cannon, R., Collins, C. et al. (2001). The 2df galaxy redshift survey: Spectra and redshifts. *Monthly Notices of the Royal Astronomical Society*, *328*(4), 1039–1063.

Condon, J. J., Cotton, W., Greisen, E., Yin, Q., Perley, R. A., Taylor, G. & Broderick, J. (1998). The nrao vla sky survey. *The Astronomical Journal*, *115*(5), 1693.

Cunningham, P., Cord, M. & Delany, S. J. (2008). Supervised learning. *Machine learning techniques for multimedia* (pp. 21–49). Springer.

Dawyndt, P., De Meyer, H. & De Baets, B. (2005). The complete linkage clustering algorithm revisited. *Soft Computing*, *9*(5), 385–392.

Desyani, T., Saifudin, A. & Yulianti, Y. (2020). Feature selection based on naive bayes for caesarean section prediction. *IOP Conference Series: Materials Science and Engineering*, *879*(1), 012091.

Donner, A. & Zou, G. (2012). Closed-form confidence intervals for functions of the normal mean and standard deviation. *Statistical Methods in Medical Research*, *21*(4), 347–359.

Dzombeta, K. & Percy, J. (2019). Flare stars: A short review.

Fogarty, J., Baker, R. S. & Hudson, S. E. (2005). Case studies in the use of roc curve analysis for sensor-based estimates in human computer interaction. *Proceedings of Graphics Interface 2005*, 129–136.

Gautama, T., Mandic, D. P. & Van Hulle, M. M. (2003). A differential entropy based method for determining the optimal embedding parameters of a signal. *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, *6*, VI–29.

Gelfand, S. B., Ravishankar, C. & Delp, E. J. (1989). An iterative growing and pruning algorithm for classification tree design. *Conference Proceedings., IEEE International Conference on Systems, Man and Cybernetics*, 818–823.

Gislason, P. O., Benediktsson, J. A. & Sveinsson, J. R. (2004). Random forest classification of multisource remote sensing and geographic data. *IGARSS 2004. 2004 IEEE International Geoscience and Remote Sensing Symposium*, *2*, 1049–1052.

Glielmo, A., Husic, B. E., Rodriguez, A., Clementi, C., Noé, F. & Laio, A. (2021). Unsupervised learning methods for molecular simulation data. *Chemical Reviews*, *121*(16), 9722–9758.

Goad, M., Page, K., Godet, O., Beardmore, A., Osborne, J., O'Brien, P., Starling, R., Holland, S., Band, D., Falcone, A. et al. (2007). Swift multi-wavelength observations of the bright flaring burst grb 051117a. *Astronomy & Astrophysics*, *468*(1), 103–112.

Gouyon, F., Pachet, F., Delerue, O. et al. (2000). On the use of zero-crossing rate for an application of classification of percussive sounds. *Proceedings of the COST G-6 conference on Digital Audio Effects (DAFX-00), Verona, Italy*, *5*.

Groot, P., Bloemen, S. & Jonker, P. (2019). Blackgem telescope array. *The La Silla Observatory-From the Inauguration to the Future*, 33.

Guglielmetti, F., Arras, P., Veneri, M. D., Enßlin, T., Longo, G., Tychoniec, Ł. & Villard, E. (2022). Bayesian and machine learning methods in the big data era for astronomical imaging. *arXiv preprint arXiv:2210.01444*.

Hinners, T. A., Tat, K. & Thorp, R. (2018). Machine learning techniques for stellar light curve classification. *The Astronomical Journal*, *156*(1), 7.

Hložek, R., Ponder, K., Malz, A., Dai, M., Narayan, G., Ishida, E., Allam Jr, T., Bahmanyar, A., Biswas, R. & Galbany, L. (2020). Results of the photometric lsst astronomical time-series classification challenge (plasticc). *arXiv preprint arXiv:2012.12392*.

Hogg, D. W. (1999). Distance measures in cosmology. *arXiv preprint astro-ph/9905116*.

Hosenie, Z., Lyon, R. J., Stappers, B. W. & Mootoovaloo, A. (2019). Comparing multiclass, binary, and hierarchical machine learning classification schemes for variable stars. *Monthly Notices of the Royal Astronomical Society*, *488*(4), 4858–4872.

Huang, H., Xu, H., Wang, X. & Silamu, W. (2015). Maximum f1-score discriminative training criterion for automatic mispronunciation detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *23*(4), 787–797.

Jackman, J. A., Shkolnik, E. & Loyd, R. P. (2021). Stellar flares from blended and neighbouring stars in kepler short cadence observations. *Monthly Notices of the Royal Astronomical Society*, *502*(2), 2033–2042.

Jarman, A. M. (2020). Hierarchical cluster analysis: Comparison of single linkage, complete linkage, average linkage and centroid linkage method.

Jayasinghe, T., Kochanek, C., Stanek, K., Shappee, B., Holoien, T. W., Thompson, T. A., Prieto, J., Dong, S., Pawlak, M., Shields, J. et al. (2018). The asas-sn catalogue of variable stars i: The serendipitous survey. *Monthly Notices of the Royal Astronomical Society*, *477*(3), 3145–3163.

Jemwa, G. T. & Aldrich, C. (2005). Improving process operations using support vector machines and decision trees. *AIChE journal*, *51*(2), 526–543.

Kessler, R., Narayan, G., Avelino, A., Bachelet, E., Biswas, R., Brown, P., Chernoff, D., Connolly, A., Dai, M., Daniel, S. et al. (2019). Models and simulations for the photometric lsst astronomical time series classification challenge (plasticc). *Publications of the Astronomical Society of the Pacific*.

Kingma, D. P. & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kodinariya, T. M. & Makwana, P. R. (2013). Review on determining number of cluster in k-means clustering. *International Journal*, *1*(6), 90–95.

Kumar, A., Ingle, Y. S., Pande, A. & Dhule, P. (2014). Canopy clustering: A review on pre-clustering approach to k-means clustering. *Int. J. Innov. Adv. Comput. Sci.(IJIACS)*, *3*(5), 22–29.

Landsberg, P. T. & De Vos, A. (1989). The stefan-boltzmann constant in n-dimensional space. *Journal of Physics A: Mathematical and General*, *22*(8), 1073.

Likas, A., Vlassis, N. & Verbeek, J. J. (2003). The global k-means clustering algorithm. *Pattern recognition*, *36*(2), 451–461.

Linares, R., Furfaro, R. & Reddy, V. (2020). Space objects classification via light-curve measurements using deep convolutional neural networks. *The Journal of the Astronautical Sciences*, *67*(3), 1063–1091.

Mahabal, A., Rebbapragada, U., Walters, R., Masci, F. J., Blagorodnova, N., van Roestel, J., Ye, Q.-Z., Biswas, R., Burdge, K., Chang, C.-K. et al. (2019). Machine learning for the zwicky transient facility. *Publications of the Astronomical Society of the Pacific*, *131*(997), 038002.

Mahabal, A., Sheth, K., Gieseke, F., Pai, A., Djorgovski, S. G., Drake, A. J. & Graham, M. J. (2017). Deep-learnt classification of light curves. *2017 IEEE symposium series on computational intelligence (SSCI)*, 1–8.

Malz, A., Hložek, R. [Renée], Allam Jr, T., Bahmanyar, A., Biswas, R., Dai, M. [Mi], Galbany, L., Ishida, E., Jha, S., Jones, D. et al. (2019). The photometric lsst astronomical time-series classification challenge plasticc: Selection of a performance metric for classification probabilities balancing diverse science goals. *The Astronomical Journal*, *158*(5), 171.

Marinova–Boncheva, V. (2008). Using the agglomerative method of hierarchical clustering as a data mining tool in capital market.

Mathur, N., Shankar, K., Kapoor, S. & Varughese, B. (2021). Evaluation of diagnostic accuracy of biomarkers of inborn errors of metabolism in sick neonates: A prospective observational study. *Clinical Epidemiology and Global Health*, *10*, 100707.

Mattila, S., Pérez-Torres, M., Efstathiou, A., Mimica, P., Fraser, M., Kankare, E., Alberdi, A., Aloy, M. Á., Heikkilä, T., Jonker, P. G. et al. (2018). A dust-enshrouded tidal disruption event with a resolved radio jet in a galaxy merger. *Science*, *361*(6401), 482–485.

Miljković, D. (2017). Brief review of self-organizing maps. *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 1061–1066.

Miller, A., Bloom, J., Richards, J., Lee, Y., Starr, D., Butler, N., Tokarz, S., Smith, N. & Eisner, J. (2015). A machine-learning method to infer funda-

mental stellar parameters from photometric light curves. *The Astrophysical Journal*, *798*(2), 122.

Mohamed, A. E. (2017). Comparative study of four supervised machine learning techniques for classification. *Information Journal of applied science and technology*, *7*(2).

Moisen, G. G. (2008). Classification and regression trees. *In: Jørgensen, Sven Erik; Fath, Brian D.(Editor-in-Chief). Encyclopedia of Ecology, volume 1. Oxford, UK: Elsevier. p. 582-588.*, 582–588.

Morales, J. L. (2002). A numerical study of limited memory bfgs methods. *Applied Mathematics Letters*, *15*(4), 481–487.

Muchai, E. & Odongo, L. (2014). Comparison of crisp and fuzzy classification trees using gini index impurity measure on simulated data. *European Scientific Journal*, *10*(18).

Naseri, H., Waygood, E. O. D., Wang, B., Patterson, Z. & Daziano, R. A. (2021). A novel feature selection technique to better predict climate change stage of change. *Sustainability*, *14*(1), 40.

Notsu, Y., Maehara, H., Shibayama, T., Honda, S., Notsu, S., Namekata, K., Nogami, D. & Shibata, K. (2016). Statistical properties of superflares on solar-type stars with Kepler data. https://doi.org/10.5281/zenodo.59138

Orwat-Kapola, J. K., Bird, A. J., Hill, A. B., Altamirano, D. & Huppenkothen, D. (2021). Light curve fingerprints: An automated approach to the extraction of x-ray variability patterns with feature aggregation–an example application to grs 1915+ 105. *Monthly Notices of the Royal Astronomical Society*.

Padovani, P., Alexander, D., Assef, R., De Marco, B., Giommi, P., Hickox, R., Richards, G., Smolčić, V., Hatziminaoglou, E., Mainieri, V. et al. (2017). Active galactic nuclei: What's in a name? *The Astronomy and Astrophysics Review*, *25*(1), 1–91.

Pasquet, J., Pasquet, J., Chaumont, M. & Fouchez, D. (2019). Pelican: Deep architecture for the light curve analysis. *Astronomy & Astrophysics*, *627*, A21.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. et al. (2011). Scikit-

learn: Machine learning in python. *the Journal of machine Learning research*, *12*, 2825–2830.

Pena, J. M., Lozano, J. A. & Larranaga, P. (1999). An empirical comparison of four initialization methods for the k-means algorithm. *Pattern recognition letters*, *20*(10), 1027–1040.

Rafiq, M., Bugmann, G. & Easterbrook, D. (2001). Neural network design for engineering applications. *Computers & Structures*, *79*(17), 1541–1552.

Raghavan, D., McAlister, H. A., Henry, T. J., Latham, D. W., Marcy, G. W., Mason, B. D., Gies, D. R., White, R. J. & Theo, A. (2010). A survey of stellar families: Multiplicity of solar-type stars. *The Astrophysical Journal Supplement Series*, *190*(1), 1.

Razali, N. M., Wah, Y. B. et al. (2011). Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *Journal of statistical modeling and analytics*, *2*(1), 21–33.

Reddy, M., Makara, V. & Satish, R. (2017). Divisive hierarchical clustering with k-means and agglomerative hierarchical clustering. *Int J of Comp Science Trands and Tech (IJCST)*, *5*(5), 5–11.

Ros, F. & Guillaume, S. (2019). A hierarchical clustering algorithm and an improvement of the single linkage criterion to deal with noise. *Expert Systems with Applications*, *128*, 96–108.

Samus', N. N., Kazarovets, E. V., Durlevich, O. V., Kireeva, N. N. & Pastukhova, E. N. (2017). General catalogue of variable stars: Version GCVS 5.1. *Astronomy Reports*, *61*(1), 80–88. https://doi.org/10.1134/S1063772917010085

Samus, N., Kazarovets, E. & Durlevich, O. (2001). General catalogue of variable stars. *Odessa astronomical publications*, *14*, 266–269.

Sana, H., De Mink, S., de Koter, A., Langer, N., Evans, C., Gieles, M., Gosset, E., Izzard, R., Le Bouquin, J.-B. & Schneider, F. (2012). Binary interaction dominates the evolution of massive stars. *Science*, *337*(6093), 444–446.

Sana, H., Le Bouquin, J.-B., Lacour, S., Berger, J.-P., Duvert, G., Gauchet, L., Norris, B., Olofsson, J., Pickel, D., Zins, G. et al. (2014). Southern massive stars at high angular resolution: Observational campaign and companion detection. *The Astrophysical Journal Supplement Series*, *215*(1), 15.

Shahapure, K. R. & Nicholas, C. (2020). Cluster quality analysis using silhouette score. *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*, 747–748.

Sharma, S., Sharma, S. & Athaiya, A. (2017). Activation functions in neural networks. *towards data science*, *6*(12), 310–316.

Smith, H. A. (2004). *Rr lyrae stars* (Vol. 27). Cambridge University Press.

Sreehari, H. & Nandi, A. (2021). A machine learning approach for classification of accretion states of black hole binaries. *Monthly Notices of the Royal Astronomical Society*, *502*(1), 1334–1343.

Sundhari, S. S. (2011). A knowledge discovery using decision tree by gini coefficient. *2011 International Conference on Business, Engineering and Industrial Applications*, 232–235.

Tachibana, Y. & Miller, A. A. [Adam A]. (2018). A morphological classification model to identify unresolved panstarrs1 sources: Application in the ztf real-time pipeline. *Publications of the Astronomical Society of the Pacific*, *130*(994), 128001.

Tarter, J. C., Backus, P. R., Mancinelli, R. L., Aurnou, J. M., Backman, D. E., Basri, G. S., Boss, A. P., Clarke, A., Deming, D., Doyle, L. R. et al. (2007). A reappraisal of the habitability of planets around m dwarf stars. *Astrobiology*, *7*(1), 30–65.

Tong, D. L. & Mintram, R. (2010). Genetic algorithm-neural network (gann): A study of neural network activation functions and depth of genetic algorithm search applied to feature selection. *International Journal of Machine Learning and Cybernetics*, *1*.

Venot, O., Rocchetto, M., Carl, S., Hashim, A. R. & Decin, L. (2016). Influence of stellar flares on the chemical composition of exoplanets and spectra. *The Astrophysical Journal*, *830*(2), 77.

Xiao, Y., Wei, Z. & Wang, Z. (2008). A limited memory bfgs-type method for large-scale unconstrained optimization. *Computers & Mathematics with Applications*, *56*(4), 1001–1009.

Yamashiki, Y. A., Maehara, H., Airapetian, V., Notsu, Y., Sato, T., Notsu, S., Kuroki, R., Murashima, K., Sato, H., Namekata, K. et al. (2019). Impact

of stellar superflares on planetary habitability. *The Astrophysical Journal*, *881*(2), 114.

York, D. G., Adelman, J., Anderson Jr, J. E., Anderson, S. F., Annis, J., Bahcall, N. A., Bakken, J., Barkhouser, R., Bastian, S., Berman, E. et al. (2000). The sloan digital sky survey: Technical summary. *The Astronomical Journal*, *120*(3), 1579.

Yuan, W., Macri, L. M., Riess, A. G., Brink, T. G., Casertano, S., Filippenko, A. V., Hoffmann, S. L., Huang, C. D. & Scolnic, D. (2022). Absolute calibration of cepheid period-luminosity relations in ngc 4258. *arXiv preprint arXiv:2203.06681*.

Zhao, Y. & Karypis, G. (2002). Evaluation of hierarchical clustering algorithms for document datasets. *Proceedings of the eleventh international conference on Information and knowledge management*, 515–524.

8 Appendices

8.1 Class distribution

Figure 8.1.1: Kernel density estimation per feature (see table 3.3.1)

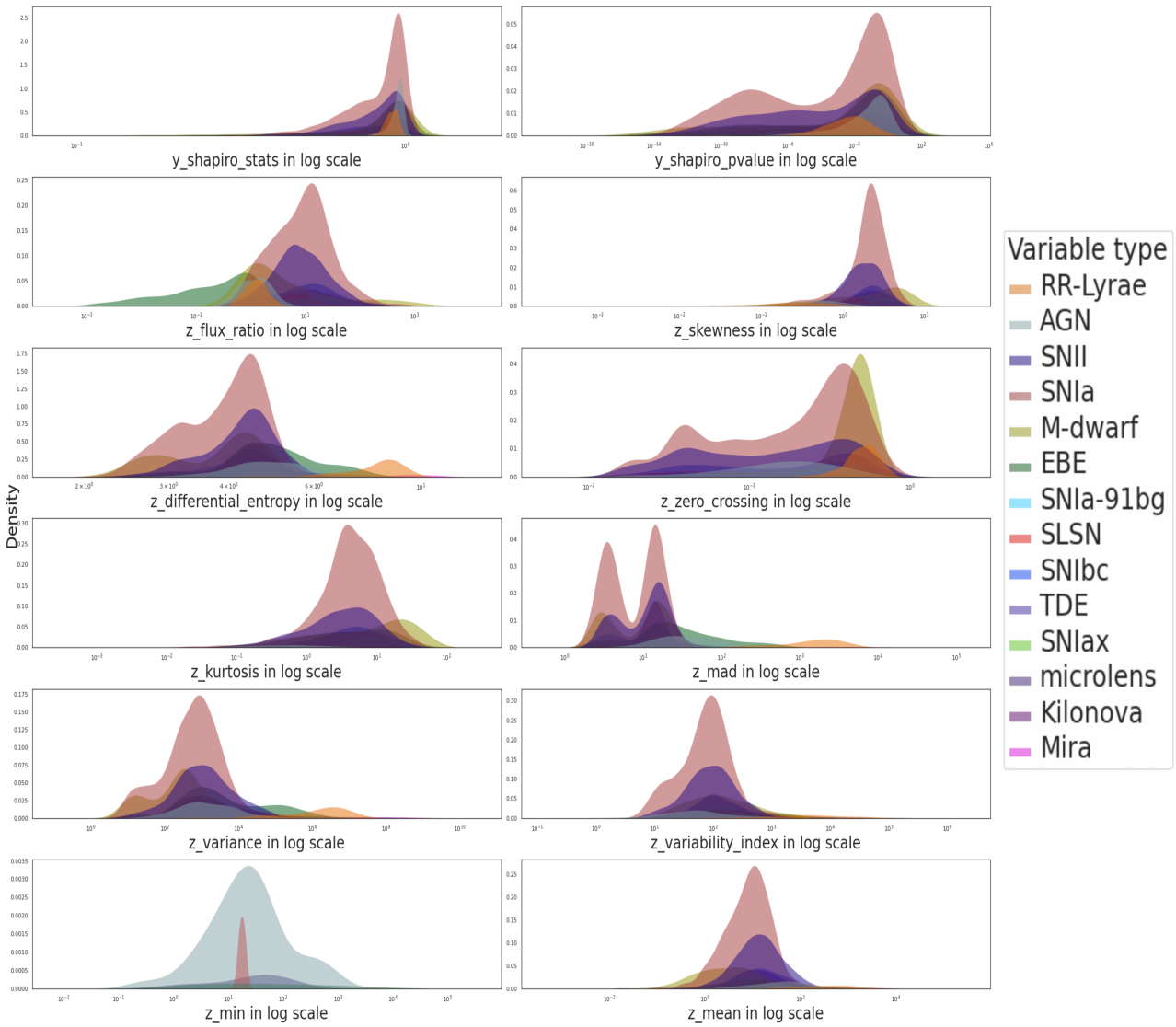


Figure 8.1.1 continued

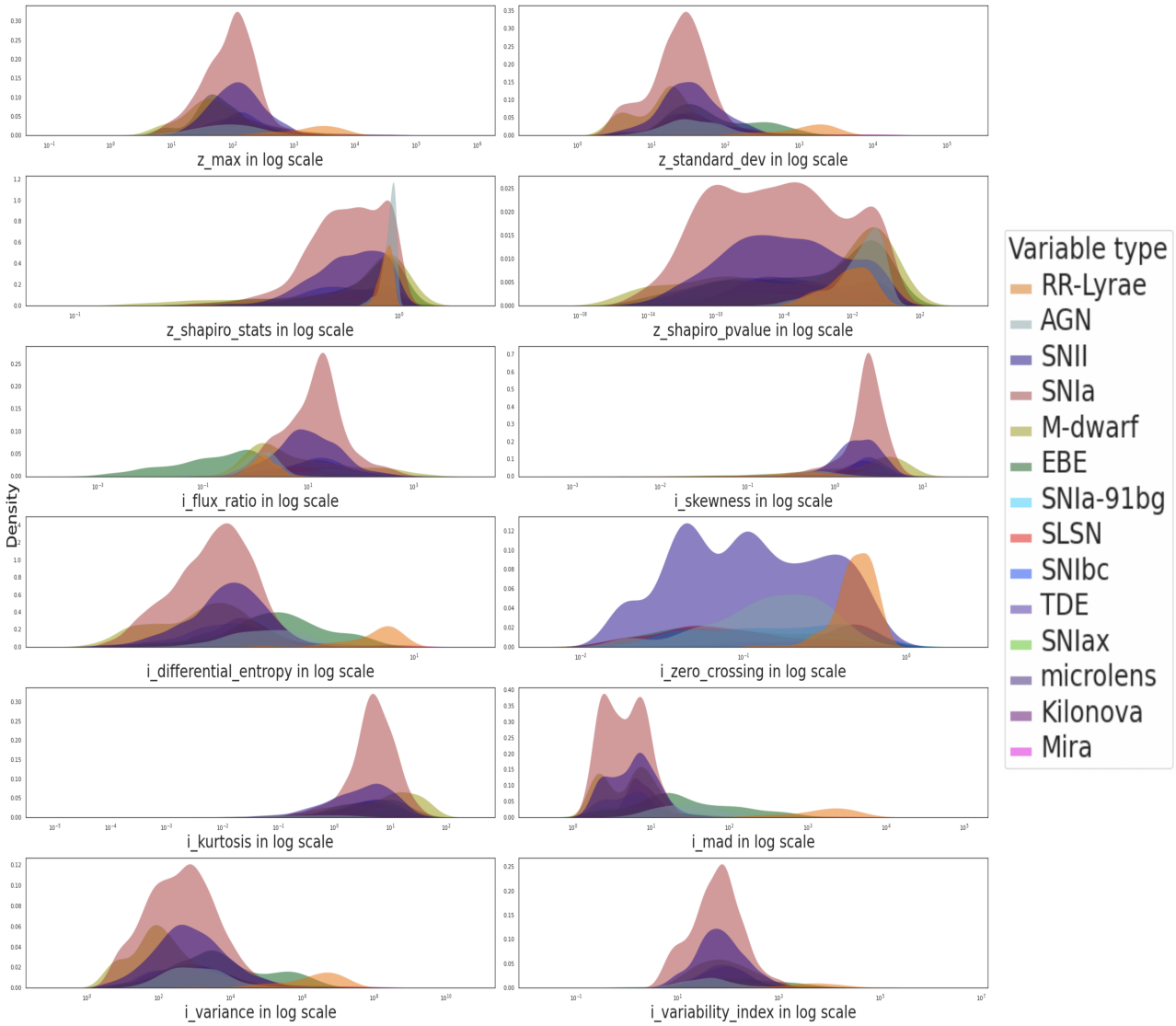


Figure 8.1.1 continued

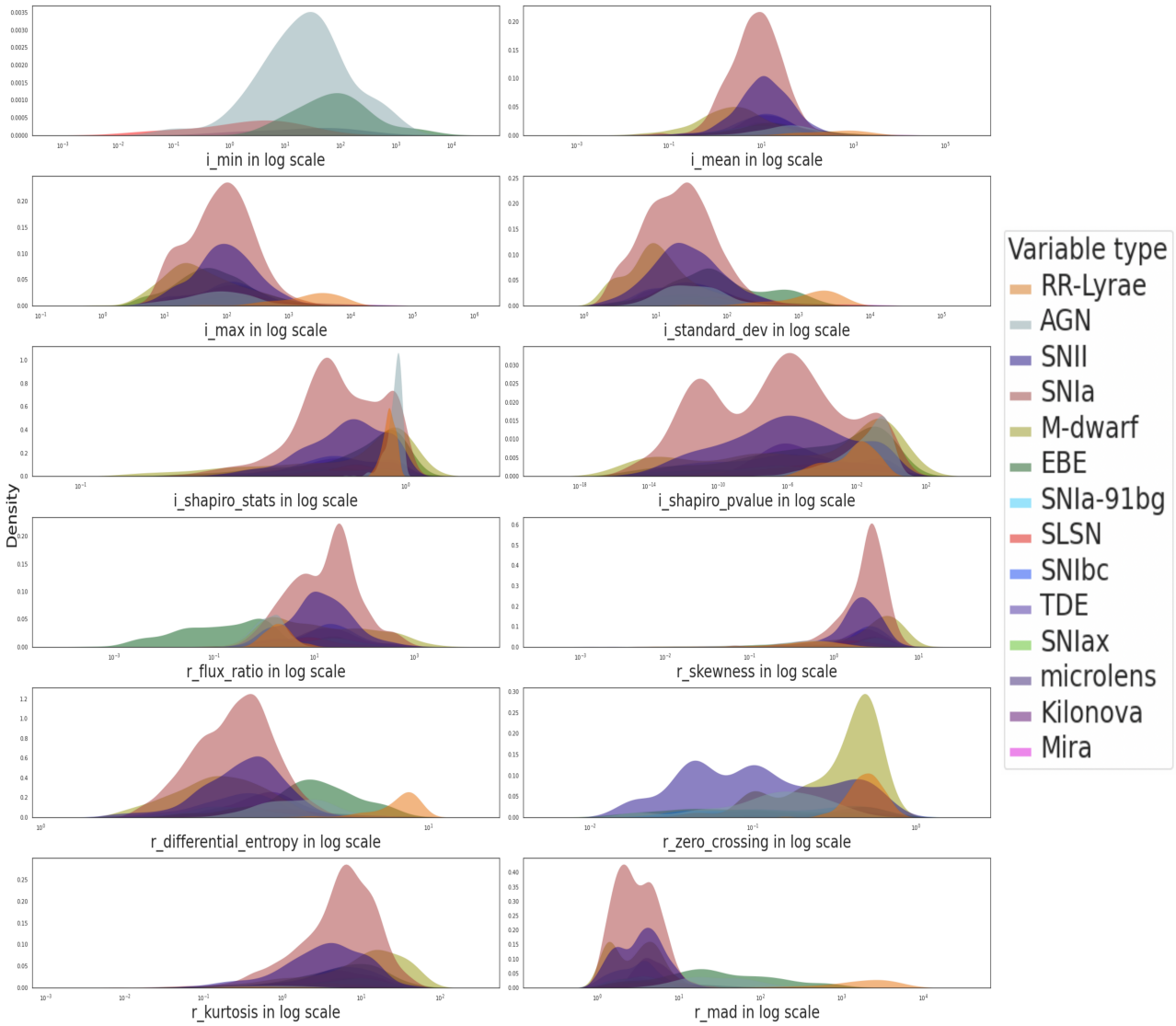


Figure 8.1.1 continued

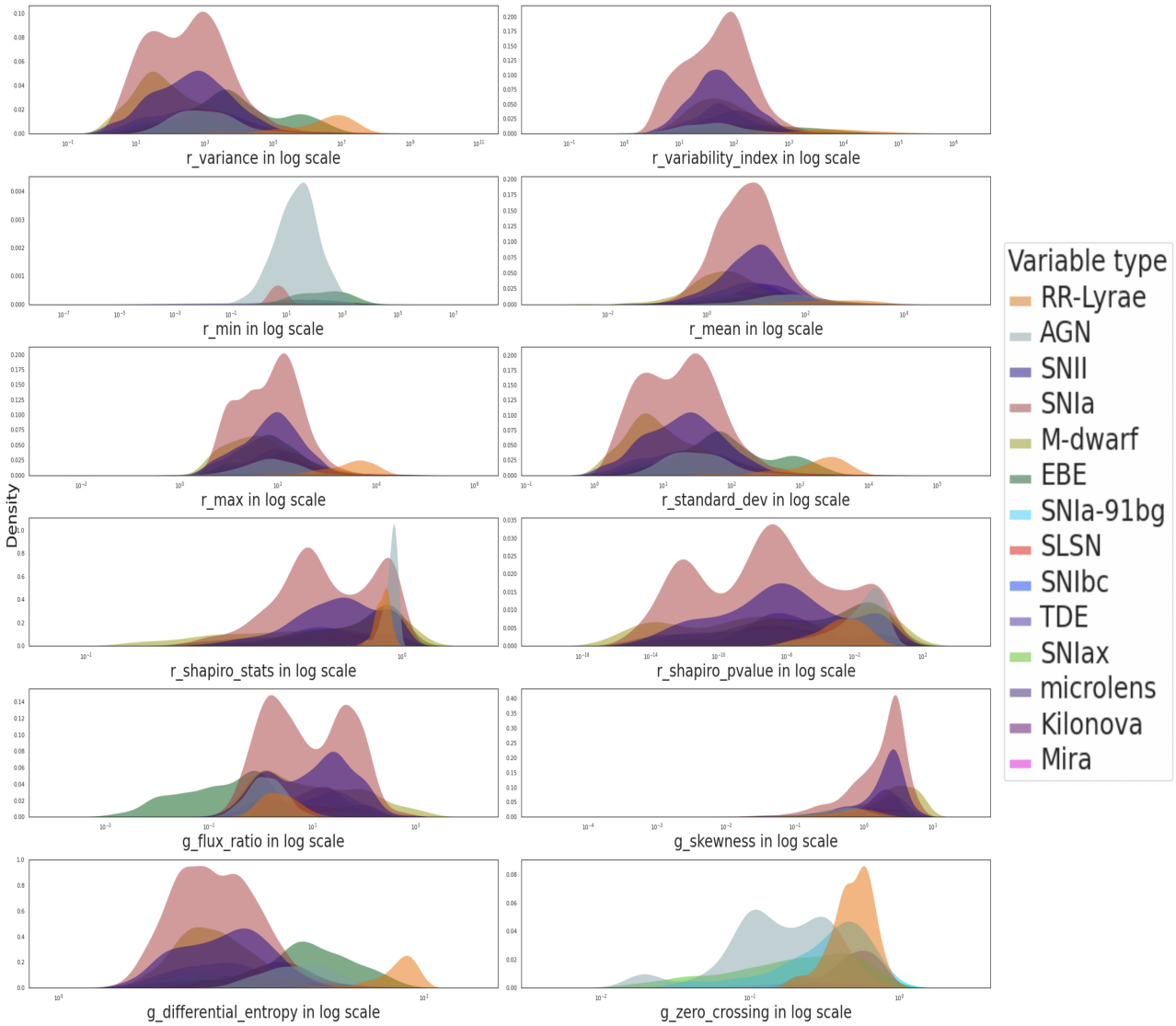


Figure 8.1.1 continued

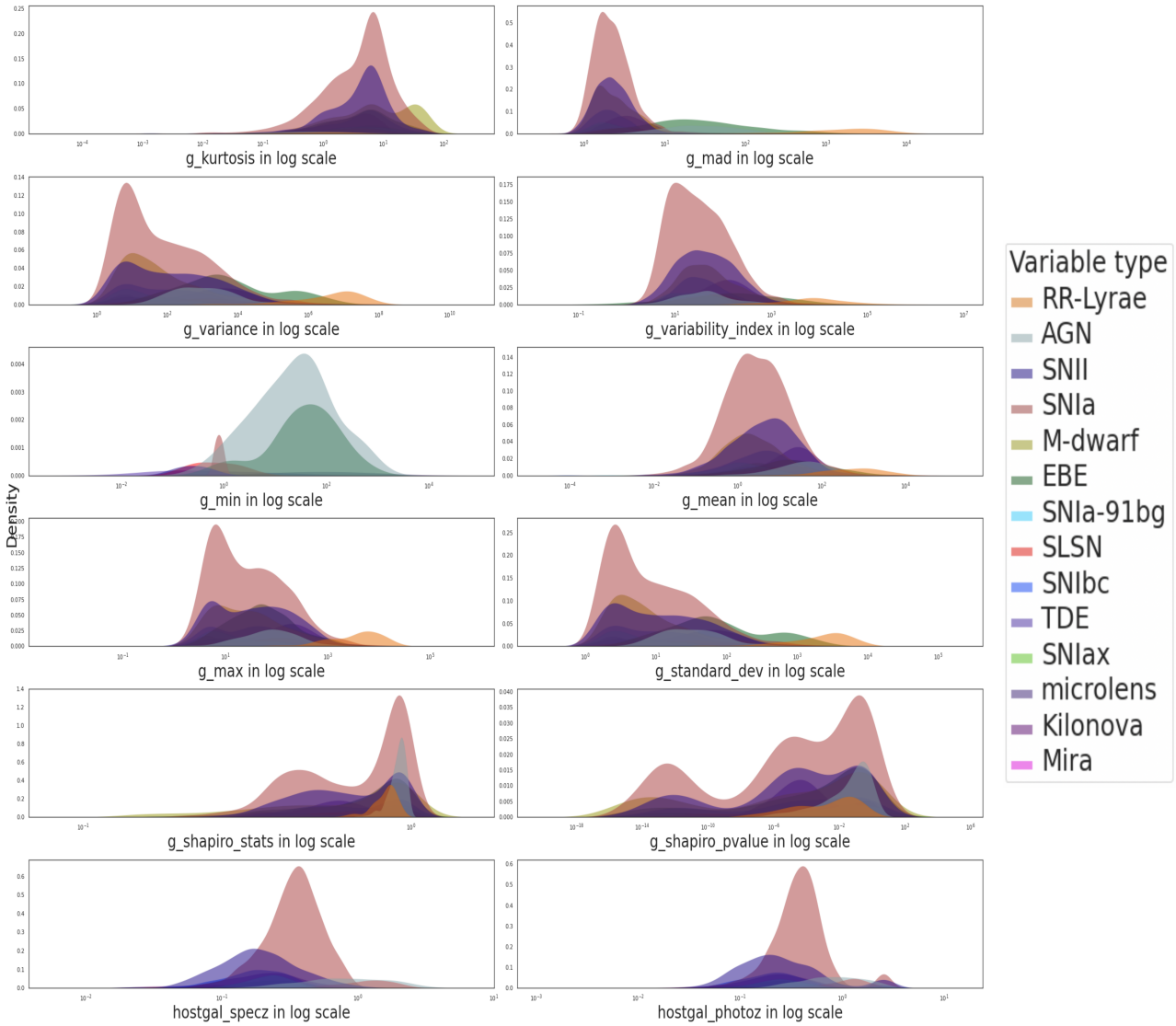


Figure 8.1.1 continued

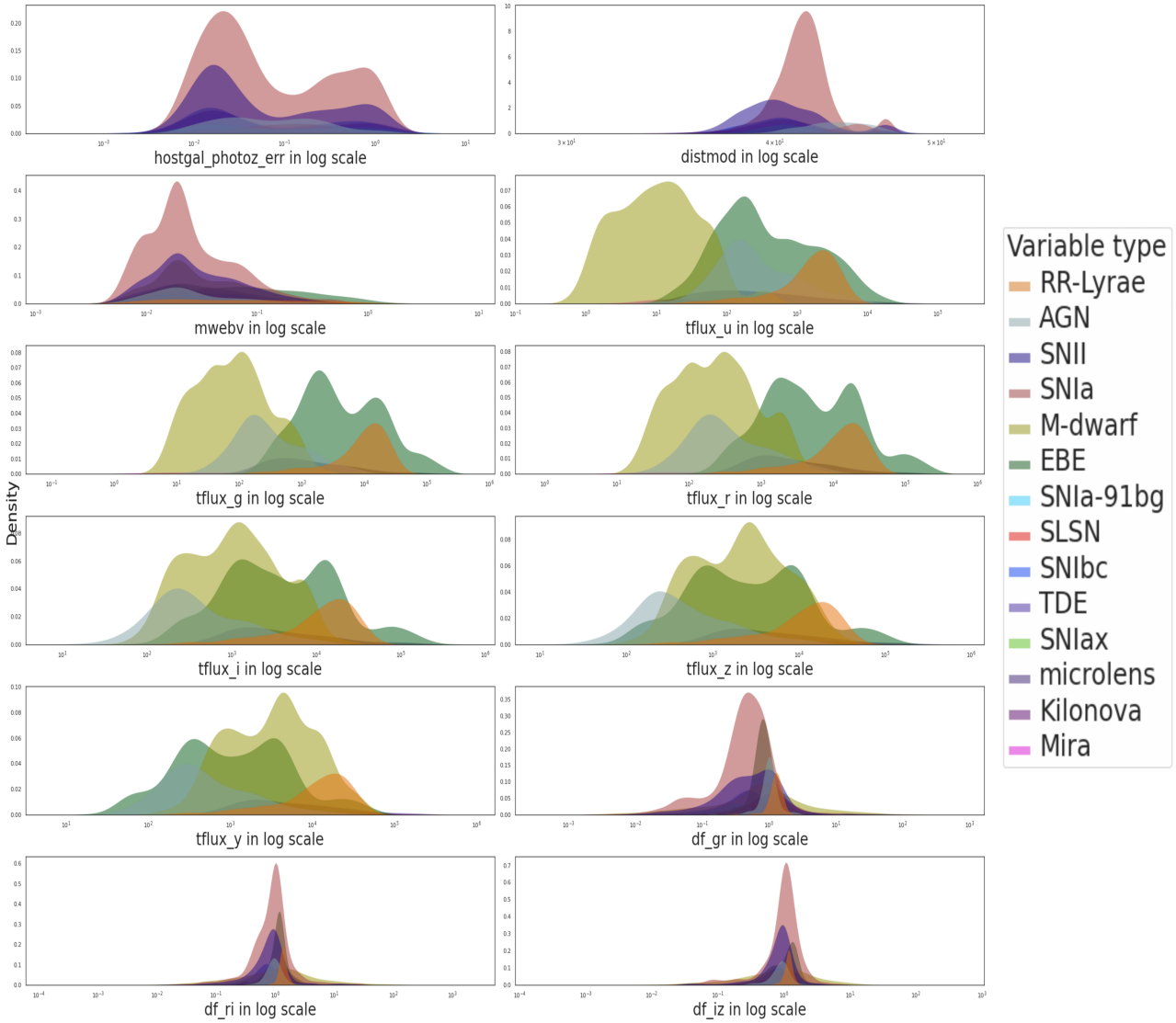


Figure 8.1.1 continued

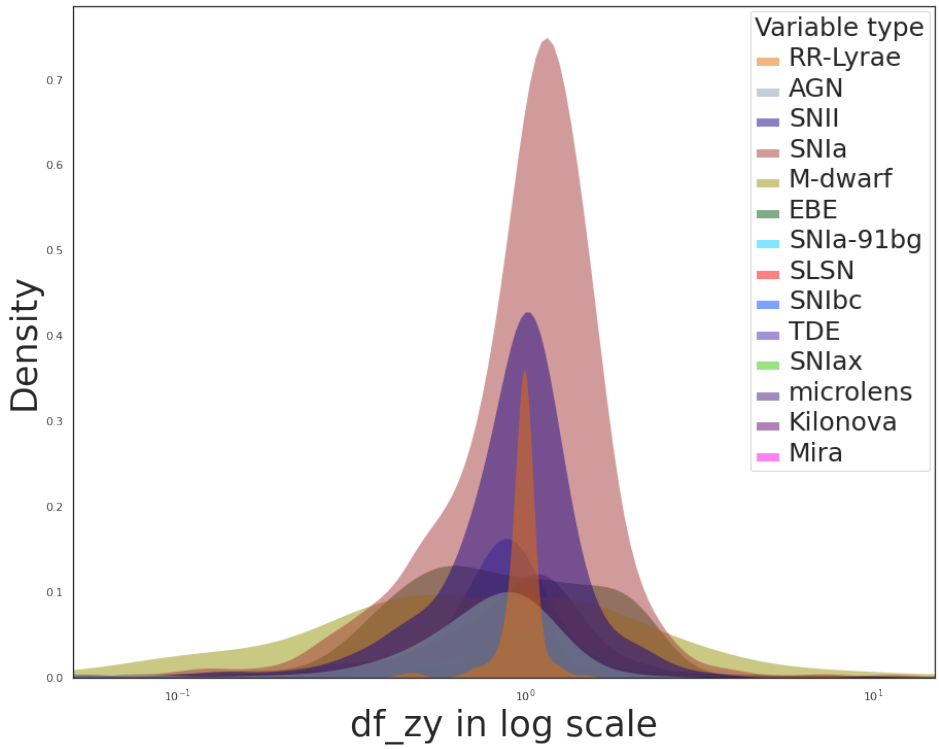


Figure 8.1.1 continued