MASARYKOVA UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA
ÚSTAV TEORETICKÉ FYZIKY A ASTROFYZIKY

Diplomová práce

BRNO 2018 **MARTIN VO**

MASARYKOVA UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA
ÚSTAV TEORETICKÉ FYZIKY A ASTROFYZIKY

Klasifikace světelných křivek kvasarů s použitím metod data-miningu

Diplomová práce

Martin Vo

Vedoucí práce: Mgr. Viktor Votruba Ph.D. **Brno 2018**

Bibliografický záznam

Autor:	Bc. Martin Vo Přírodovědecká fakulta, Masarykova univerzita Ústav teoretické fyziky a astrofyziky
Název práce:	Klasifikace světelných křivek kvasarů s použitím metod data-miningu
Studijní program:	Teoretická fyzika a astrofyzika
Studijní obor:	Astrofyzika
Vedoucí práce:	Mgr. Viktor Votruba Ph.D.
Akademický rok:	2017/2018
Počet stran:	xiv + 63
Klíčová slova:	Světelné křivky; Data Mining; Machine Learning; Convolutional Neural Network; Generative Adversarial Neural Network; SAX; Quasars

Bibliographic Entry

Author:	Bc. Martin Vo Faculty of Science, Masaryk University Department of theoretical physics and astrophysics
Title of Thesis:	Classification and detection of the quasar light curves with help of data-mining methods
Degree Programme:	Theoretical Physics and Astrophysics
Field of Study:	Astrophysics
Supervisor:	Mgr. Viktor Votruba Ph.D.
Academic Year:	2017/2018
Number of Pages:	xiv + 63
Keywords:	Light Curves; Data Mining; Machine Learning; Convolutional Neural Network; Generative Adversarial Neural Network; SAX; Quasars; Be stars

Abstrakt

Cílem této práce je představit různé přístupy strojového učení ke klasifikaci světelných křivek kvazarů. Tyto objekty jsou ideální pro testování klasifikačních metod, protože jejich proměnnost je nepravidelná a bez nápadných charakteristických rysů. Navíc jsou velmi podobné světelným křivkám Be hvězd, které stále tvoří početnou skupinu mezi kandidáty na kvazary v mnoha astronomických automatizovaných přehlídkách. Podobnost je většinou tak velká, že i pro oko odborníka jsou tyto objekty nerozlišitelné. Z tohoto hlediska není cílem této práce pouhá automatizace klasifikačních procesů, ale zároveň vytvoření modelu, který překoná i odborný lidský úsudek. Model musí být schopen se naučit i nepatrné závislosti mezi jednotlivými vzory, které lidskému oku mohou uniknout, a proto jsou kladeny vysoké nároky na zpracování dat, způsob reprezentace objektů pro klasifikaci a komplexnost modelu.

Za tímto účelem byl vytvořen *Light Curves Classifier* – program pro klasifikaci světelných křivek s využitím metod data miningu a strojového učení. Projekt byl vyvíjen přes 3 roky a je nyní k dispozici pro širokou veřejnost jako webová aplikace, Pythonovská knihovna nebo jako aplikace příkazového řádku. Program je schopen se naučit z trénovacích dat na základě specifikovaných rysů popisujících zkoumané objekty. Natrénované modely mohou být použity k filtrování výstupů z astronomických databází nebo z lokálně uložených dat. Implementované databázové konektory lze použít ke stahování světelných křivek z různých databází jako OGLE II, OGLE III, MACHO, Kepler, ASAS či CoRoT.

Za pomocí tohoto balíku byl natrénován klasifikační model pro rozpoznávání kvazarů od Be hvězd, proměnných hvězd a neproměnných hvězd. Jednotlivé hvězdy byly pro klasifikaci reprezentovány různými parametry jako je například odlišnost od vzorku známých kvazarů. Natrénovaný model byl pak použit k porovnání výsledků s podobnou prací za účelem ověření přesnosti modelu.

Druhá část práce je věnována aplikaci konvolučních neuronových sítí na světelné křivky. Přestože jsou konvoluční neuronové sítě široce využívány pro obrazová data a časové řady, v oblasti světelných křivek jde stále o poměrně netradiční přístup. Důvodem je kvalita světelných křivek, která je pro neuronové sítě zpravidla nedostačující. Světelné křivky jsou většinou značně zašuměné a obsahují mnoho chybějících hodnot. V této práci je ukázáno, že i takováto data mohou za pomocí důkladného zpracování vést k vytvoření kvalitního modelu.

Klasifikační modely predikují pravděpodobnost příslušnosti do jednotlivých tříd, ale zpravidla nám neumožňují lépe porozumět zkoumaným objektů. V práci je ukázáno, že vizualizace konvolučních filtrů je jeden ze způsobů, jak odhalit na jaké vzory se neuronové sítě dívají. Generativní adversariální neuronové síťě mohou být vhodnou metodou k lepšímu porozumění proměnnosti kvazarů. Skládají se ze dvou modelů, které se navzájem trénují. V našem případě, první model klasifikuje světelné křivky a druhý vytváří nové,

které se po zdárném natrénování podobají světelným křivkám kvazarů. Analýza aplikace této metody pro kvazary může být použita pro tvorbu modelů i pro jiné astronomické objekty, jedná se tak o zcela nový přístup ke světelným křivkám.

Abstract

The goal of this work is to present machine learning approaches for classification of light curves of quasars. These objects are ideal for testing classification models because their variability is irregular and without any obvious characteristic patterns. Moreover, light curves of Be stars are very similar to the light curves of the quasars and they are usually indistinguishable even for the experts in the field. It means that the aim is not only to build an automatic classification pipeline to process a huge amount of data but to build a model which even outperform a human. Therefore the model has to be much more complex in order to detect tiny dependencies between particular patterns which are almost invisible to the human eye.

In this work, Light Curves Classifier is presented – a package which utilizes various modern instruments and methods of data mining and machine learning in order to build efficient classification pipelines. It has been developed over 3 years and it can be used as Python package, web application or command line application. The program is capable of learning from the given training sample based on specified features describing inspected objects. The trained classifiers can be used to filter the outputs from an astronomical database or from the data stored locally. The database connectors can be also employed to download queried light curves and all available information from sources such as OGLE II, OGLE III, MACHO, Kepler, ASAS and CoRoT.

Finally, a classification pipeline was developed to distinguish quasars from Be stars, variable stars and stars without a variability. Particular stars were described by various features such as dissimilarity from template quasars. The trained model was then used to compare results from similar work in order to determine the precision of the model.

The second part of this work is dedicated to Convolutional Neural Networks and their application for light curves. Despite the fact that they are widely used for sequential data, it is still a quite new approach for light curves, because they are often noisy and contain lots of missing values which can be very problematic. In this work, it is shown that by proper feature engineering even low-quality data from ground-based observations can be employed to make a decent model.

Classification models usually do not provide any knowledge about the inspected objects. Since neural networks are complicated and hard to interpret, one can hardly explain decisions of the model. A new proposal of this work is to use Generative Adversarial Neural Network (GAN) which is designed exactly for such kind of tasks. It combines two models in order to make a powerful pair of classifier and generator which are trained together. Analysis of the application of the method can be used to develop custom models for arbitrary astronomical objects because for light curves it is a completely new approach.

MASARYKOVA UNIVERZITA
Přírodovědecká fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Akademický rok: 2017/2018

Ústav: Ústav teoretické fyziky a astrofyziky

Student: Bc. Martin Vo

Program: Fyzika

Obor: Teoretická fyzika a astrofyzika

Směr: Astrofyzika

Ředitel *Ústavu teoretické fyziky a astrofyziky* PřF MU Vám ve smyslu Studijního a zkušebního řádu MU určuje diplomovou práci s názvem:

Název práce: Klasifikace světelných křivek kvasarů s použitím metod data-miningu

Název práce anglicky: Classification and detection of the quasar light curves with help of data-mining methods

Oficiální zadání:
Úkolem studenta bude aplikovat data-miningové metody na dostupné fotometrické přehlídky (OGLE,MACHO,ASAS..) a na základě stanovených kriterií a metod data-miningu vybrat kandidáty reprezentující kvazary. Dále student provede cross-matching jim vybraných kandidátů s verifikovaným katalogem kvazarů - The half million Quasars (HMQ) Catalogue a provede srovnání obdobnou prací Eyer(2002),Geha(2003)

Jazyk závěrečné práce: angličtina

Vedoucí práce: Mgr. Viktor Votruba, Ph.D.

Konzultant: RNDr. Petr Škoda, CSc.

Datum zadání práce: 28. 11. 2016

V Brně dne: 11. 5. 2018

Souhlasím se zadáním (podpis, datum):

Bc. Martin Vo **student**	Mgr. Viktor Votruba, Ph.D. **vedoucí práce**	prof. Rikard von Unge, Ph.D. **ředitel Ústavu teoretické fyziky a astrofyziky**

Poděkování

I would like to extend my thanks to Mgr. Viktor Votruba Ph.D. for sharing his wisdom with me and his great support. Moreover, I would like to thanks to RNDr. Petr Škoda, CSc. for his help.

Prohlášení

Prohlašuji, že jsem svoji diplomovou práci vypracoval samostatně s využitím informačních zdrojů, které jsou v práci citovány.

Brno 13. května 2018 .....

Martin Vo

Contents

Introduction

Motivation

This work is built upon my bachelor thesis which was dedicated to an analysis of light curves of quasars using machine learning. However, this thesis is much more mature in terms of methods, feature engineering and developed tools. It utilizes results obtained from the previous work, for instance, a sample of quasars, which were crossmatched from various sources.

This work is not focus only on presenting new methods and approaches for using machine learning for analysis of light curves, but it also presents issues related to this topic such as data quality, feature engineering, representation of a physical phenomena by a machine learning model and quasars. Since machine learning methods tend to be ”black boxes”, great focus is put on evaluating each step in the classification pipeline from data engineering to model selection. For example, it could easily happen that the data is biased or it becomes to be biased due to incorrect data preparation.

The first part of this thesis is dedicated to *Light Curves Classifier* – a package for classification of light curves. The early version of this package was already introduced in the previous work, but since then the package was significantly upgraded. For instance, the command line interface (CLI) and the web interface were implemented in order to make the package accessible by multiple ways and also to non-programmers. Moreover, the quality of the package was improved by the documentation, extensive test coverage, pypi repository and GitHub repository. It employs modern technologies such as Redis Queue for processing queries or Docker for containerization of the web interface. More details are provided in the section 1. An article about the package, which is part of this thesis, was submitted to Astronomy & Computing and after many changes suggested by the reviewers, it is now in the state of the final revision.

A new approach of using Generative Adversarial Network (GAN) for generating light curves is presented in the second part of this thesis. GANs are becoming widely used for image data, but sequential data and moreover light curves are the new domain for this approach. Light curves as sequential data are not generally suitable input for neural networks. In most of the cases, the reason is quality of data which is very important for neural networks. However, all of these challenges can be overcome, which is discussed in more detail in section 3.1.

Data mining

Huge astronomical surveys represent endless possibilities for data analysis. Despite the huge spectroscopic databases, light curves are still the dominant source of information about the universe and their processing is beyond human capabilities. Therefore, it is essential to have automated pipelines to process, analyze, and classify data from astronomical surveys.

A typical example of the necessity for automated classification is photometric data collected during the space missions. The ESA Gaia mission [1] will produce photometric data on about 1.5 billion stars with the sampling around 80 measurements for each star during 5 years. Such a huge database (petabytes) needs to be checked and processed automatically on the High-Performance Computing Cluster (HPCC). Debosscher [2][3] showed that Bayesian Neural Networks and Gaussian Mixture have the best performance for automated supervised classification, according to their analysis and classification of photometric data from OGLE database and associated OGLE variable star catalogue (see [4]). Later, they successfully applied and used both of the classifiers for an automated supervised classification of variable stars in the COROT fields and produced the catalogue of classified variable stars together with their light curves [5]. It helps improve the training set for a similar classification pipeline developed for GAIA. It is based on same principles and methods applied in case of COROT photometry but with adaptation to different light curve sampling and with additional information, namely colours and spectroscopy. It is important to mention that some of the variability class types are missing in the GAIA analysis, as high-frequency multi-periodic pulsators (like roAp -stars, sdBV-stars) cannot be captured with GAIA photometry sampling.

Another example is Catalina survey [6] which contains 500 million sources with a median of 320 observations per time series over an 11-year baseline. CRTS (Catalina Real-time Transient Survey) detects [7] and openly publishes all transients within the minutes of the observation so that all astronomers can follow the ongoing events. Besides their main focus on transients, they have developed methods for classification of periodic variable stars [8], close supermassive black hole binaries [9] and others.

The success of object classification strongly depends on object representation. Light curves are usually represented by feature vectors of statistical parameters. Graham [10] reviewed methods for quasar classification by using many common statistical descriptors such as Autoregressive models or Slepian wavelet variance.

However, the most straightforward approach is to represent them as vectors of magnitudes and use them for machine learning as feature vectors. However, there are many challenges such as different time lags between measurements - which can vary from seconds to days, missing values (e.g. due to big gaps during daytime for ground observations or maintenance downtimes) or erroneous values (bad pixels in the image). Missing data can be with certain error obtained by some machine learning models. For example, Pichara [11] used Bayesian networks to predict missing values and dependency relationships between variables. This topic is in more detail discussed in the second part of this thesis.

Quasars

Quasars are mysterious objects from the early universe when conditions were so extreme that objects such as quasars could be created. According to current theories quasars are massive black holes in galactic centres surrounded by accretion discs of diameters of several light weeks. Nearby stars are caught by strong gravitational field and forced to revolt around the black hole in such speed that they are suddenly torn apart and supply accretion disc by their mass. The surrounding mass is gradually falling into the black hole which produces huge amounts of energy. Typical luminosity is around 10^{40} W which is luminosity of approximately 100 billions of Suns. In order to supply the luminosity at such scale, it would be necessary for a quasar to consume the mass of approximately 10 stars per year (for the brightest quasars even thousand stars per year). [12]

Variability of quasars is most likely caused by transmission of change of luminosity from a local source to other parts of the accretion disc. If we consider that the change of luminosity is transmitted by near speed of light, we can estimate the size of particular quasars from observed light curves. Variation of brightness is irregular and light curves are composed of various patterns. Example of the light curve of a quasar can be found in figure 1. The light curve is composed of both long patterns (several hundred days) and short-term patterns in a scale of weeks. Combination of these patterns is unique for each quasar. Therefore it is very challenging to find common features for them.

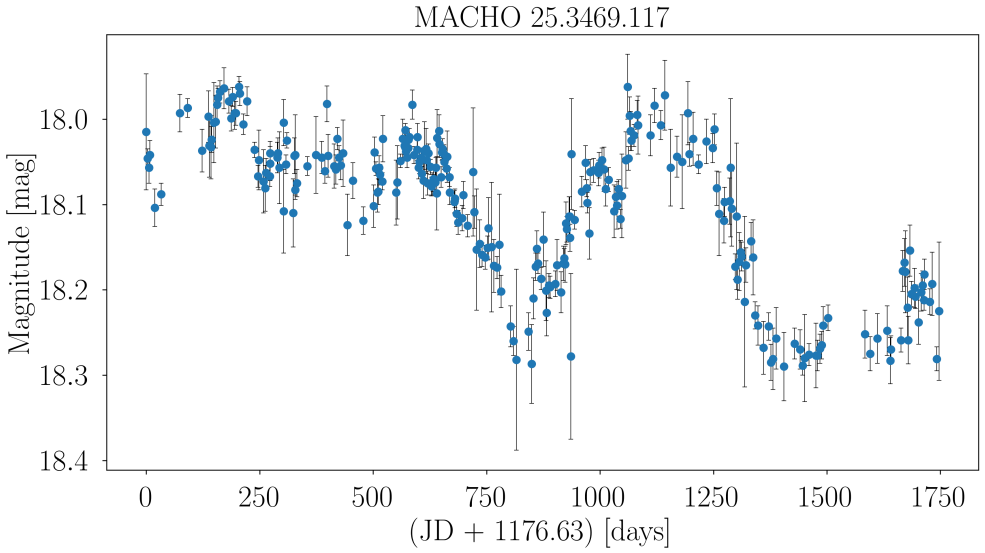


Figure 1: Example of light curve of a quasar

Despite all efforts with building classifiers of light curves of quasars, the only conclusive evidence for confirmation of quasars is the spectral measurement. It allows us to calculate distances by measuring shifts of wavelengths of known absorption or emission lines. If we consider that light observed wavelength $\lambda_{observed}$ were originally emitted at wavelength $\lambda_{emitted}$, then redshift z is given by:

$$z = \frac{\lambda_{observed}}{\lambda_{emitted}} - 1. \tag{1}$$

Employing relativistic Doppler formula for motion completely in the radial direction we can estimate radial velocity v_r:

$$z = \frac{1 + \frac{v_r}{c}}{1 - \frac{v_r}{c}}, \tag{2}$$

where c is speed of light. Redshift is caused by the expansion of the universe which was derived from the general relativity equations in 1922 by Alexander Friedmann [13]. Georges Lemaître proposed rate of the expansion which is now known as Hubble constant H_0. Speed of expansion v is then given by:

$$v = H_0 d, \tag{3}$$

where d is the distance of the object. Note that it is true just longer distances such as distances between galaxies. Luminosity L of the object can be obtained from observed flux F at distance d:

$$L = 4\pi d^2 F. \tag{4}$$

Combining 1, 2, 3 and 4, we can determine luminosity of the object from the difference of observed and actual wavelength of given spectral line. Since quasars are the only objects with such extreme luminosity we can use this procedure to confirm quasars candidates.

Chapter 1

Light Curves Classifier

In this era of Big Data, enormous amounts of data are collected every day. The light curves are one of the most common products of the universe observations gathered by the astronomical instruments. The most fundamental task is to classify them in order to identify the type of an observed object. Despite the effort to categorize particular light curves, there is still no tool which unifies classification procedures into one powerful instrument.

Despite the fact that machine learning methods are self-learning, there are generally vast of parameters which have to tune. They vary task to task and they depend on the training sample and given objective. The primary goal was to develop a simple yet powerful tool capable of solving all usual problems related to classifying objects in the astronomical surveys - i.e. extracting features from data, tuning the parameters of methods, machine learning, data-mining and classification. Developed *Light Curve Classifier* tool covers a wide range of tasks. In particular, it includes searching and downloading light curves from various sources like OGLE II, OGLE III, MACHO, Kepler, ASAS, CoRoT and other surveys, the statistical and physical description of their properties, visualization and (the most) notably, the classification of the light curves based on their properties.

Furthermore, *Light Curve Classifier* can be easily extended by implementing new database connectors, descriptors, etc. So far there are 12 types of features which can be calculated from both light curves and star's metadata. However, new methods of extracting and calculating features can be easily added. The program is even capable of handling the tasks it was not originally intended for. For example, the current version could be used for pattern mining of any time series which emerge in other fields of science and engineering.

It is important to note that the program can also be accessed through a web interface or with a command line interface (CLI). A filter pipeline composed of custom machine learning methods (such as Neural Network, Gradient Boosting etc.) and descriptors extracting selected features from data files can be made just with a few clicks. Then, the filter file can be used to query astronomical databases by selecting predefined data sources and writing a common query. Star objects meeting the filter parameters are stored, and, subsequently, they can be used for another training, validation, etc.

Another convenient feature of the program is the seamless integration of parameters associated with methods (such as a number of neurons in the hidden layer of NN, the smoothing parameters of light curves etc.) The package can evaluate the combinations

of the given ranges of parameters by splitting labelled stars into training and testing (validation) samples. Once the training has been done, the most successful combination of parameters is saved and can be used to search in astronomical databases. Also with the web interface and CLI, all the combinations of parameters can be visualized. Last but not least, models can be explored by probability surface plots which show model predictions for particular positions in parameters space. High dimensional data (more than 2 features) can be visualized in 2D/3D projections by a dimensionality reduction technique such as PCA.

1.1 Web Interface

A working service can be found as a part of VO-CLOUD system of the Stellar Department of Astronomical Institute of the Czech Academy of Sciences in Ondřejov [14]. However, the system runs only on a modest computer, so the users are therefore advised to install own environment from the GitHub repository [15]. The docker image is freely available on the Dockerhub [16].

```
docker run -d -p 80:80 mavrix93/lcc_web
```

The command above will run the web application on the background with exposed port 80. Therefore the website can be found at `localhost/lcc`. Using the web application is the easiest way how to explore the package. Therefore it is employed to introduce usage of the package in this section.

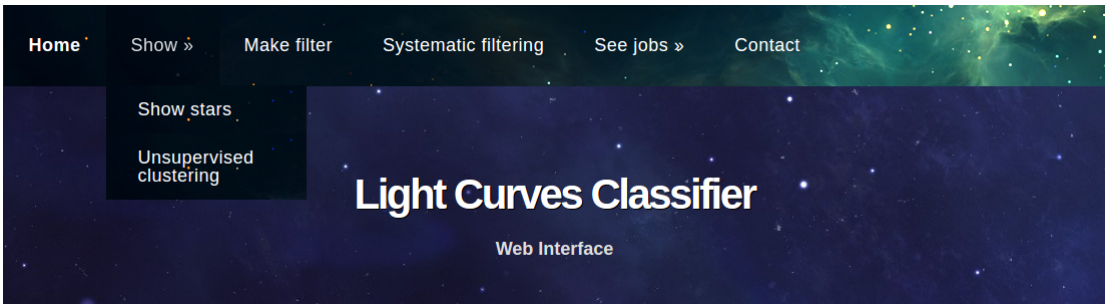


Figure 1.1: Menu of the Web Interface

The web interface provides an access to all main features offered by the package. Particular users have their own environments, thus the outputs and jobs are private. Calculations of the hosted service take place in a computational cluster and jobs are executed in threads so multiple calculations can be run simultaneously.

One of the basic tasks is to create a filtering pipeline to extract desired features and employ a classifier to make a prediction. In the terms of the package, the basic object is *filter* which consists of classification models and the feature extraction pipeline. These two parts of the filter are designated as deciders and descriptors. It will be discussed in more details in the following sections.

Figure 1.2 shows a part of the page where descriptors (feature extractors) and deciders (classifiers) are selected. Parameters values or their ranges have to be specified for both descriptors and deciders. In case of providing ranges, all combinations of parameters are evaluated and the best one is used to create the filter.

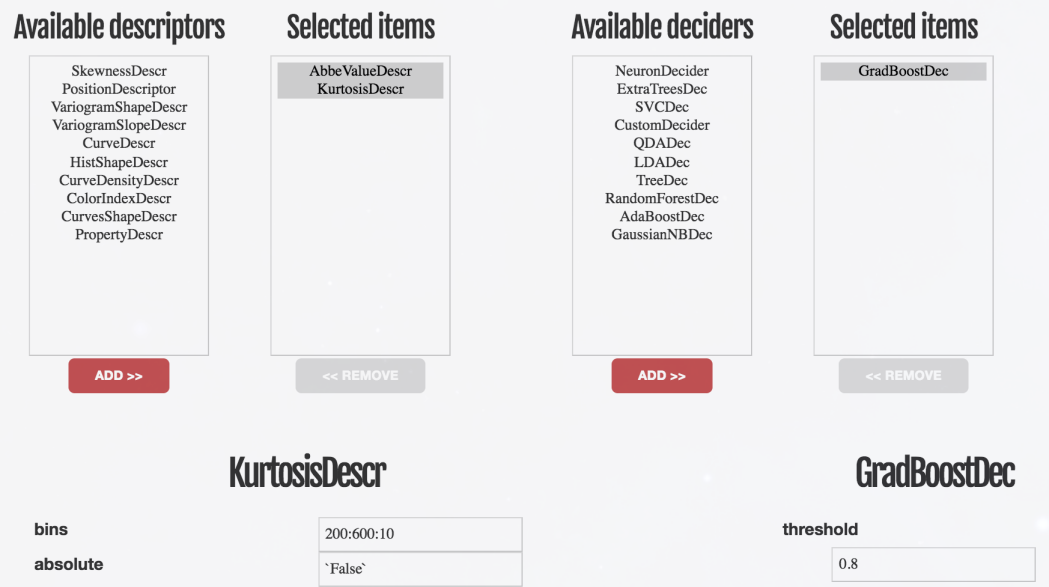


Figure 1.2: Upper part of the page for creating filters. The Arbitrary number of descriptors and deciders can be selected.

Once the form has been submitted an asynchronous job is created. Note that tuning of parameters can take a long time since all possible combinations have to be evaluated. Information about the progress of all jobs can be monitored from the jobs overview. Figure 1.3 shows an example of part of the result page. It consists of probability plot of the data in the feature space and light curve plot which is changed interactively based on the selected object in the feature space plot. The upper part of the page contains the table of all evaluated combinations of parameters and estimated accuracy.

A Created filter can be stored as a file and used later for filtering both the output from astronomical surveys and locally stored stars. It can be done by the database query form 1.4 which can be also used to simply download star objects with light curves without filtering. Stars are represented by FITS files (see 1.6) with light curves stored in binary extensions. It allows storing all metadata about the objects. Note that there is no standard format for storing light curves. Therefore it was decided to employ FITS as the format for light curves data.

Note that result stars files, filters etc. generated by the web interface can be loaded by the Python package, the command line application and vice versa.

1.2 Package Description

Following sections describe the key parts of the package. For a fully comprehensive guide how to use it, see the package repository [15]. The *Light Curve Classifier* is a Python

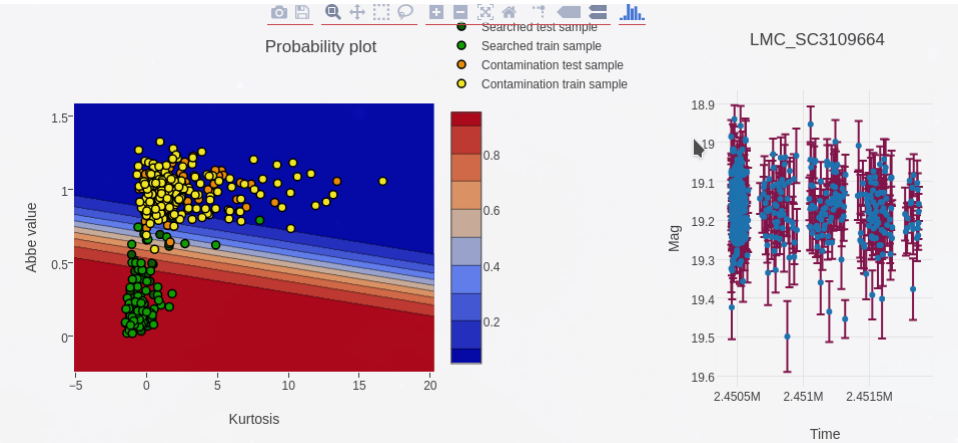


Figure 1.3: Result of training of the model. Kurtosis and Abbe Index are calculated for particular objects and plotted for both quasars (green) and non-variable stars (yellow/orange). Colours on the background represent the probability that an object with the given coordinates in the feature space is a quasar. It was generated by Linear discriminant analysis classifier. The graph is interactive and can be used to visualize light curves of selected objects.

package with a command line and web interface. The package is available freely at [15].

It consists of three main functions:

1. Feature extraction from light curves and their metadata
2. Searching astronomical surveys
3. Machine learning — classification

1.2.1 Database connectors

There are several predefined implementations of connectors to OGLE II [17], OGLE III [18], Kepler Archive [19], Catalina [20], MACHO [21], ASAS [22] and CoRoT [23] databases. Light curves can also be loaded from locally stored files such as FITS or csv and dat files.

The connectors can be used to download stars based on the query, or they can be implemented into a pipeline ”download-describe-classify”. Star objects are stored as FITS files, hence all-star attributes can be preserved and loaded again for next processing.

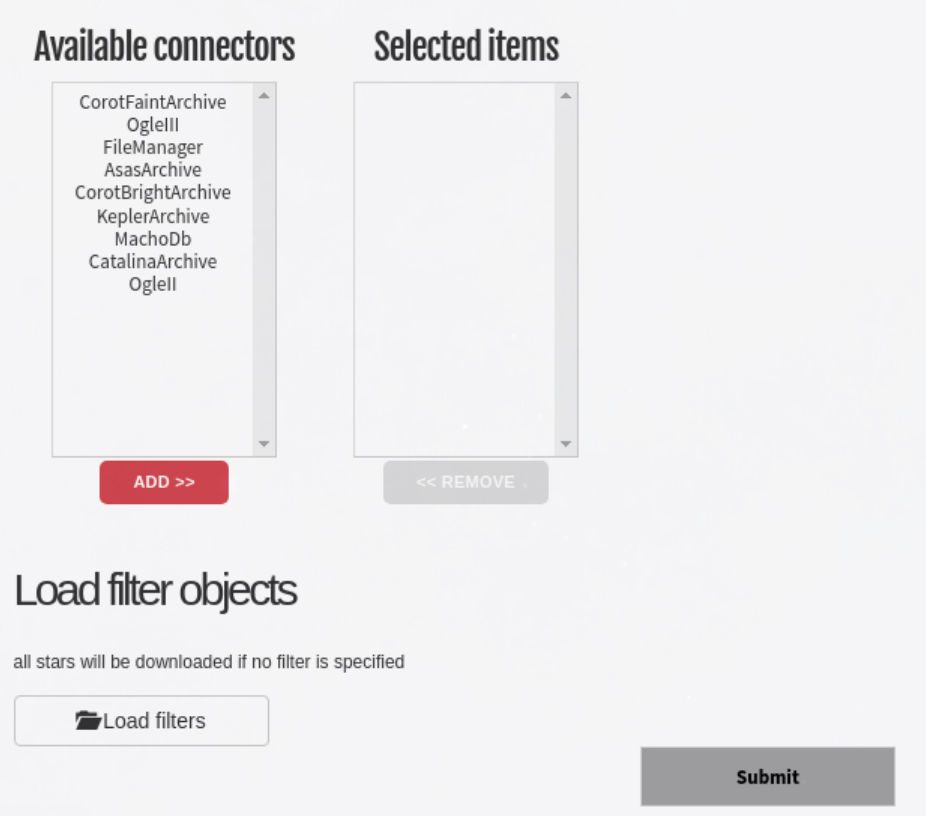


Figure 1.4: A Common query interface for astronomical databases. The results of queries can be filtered with trained filters. The results are stored as FITS files with all available data about the objects.

```
SIMPLE  =                    T / conforms to FITS standard
BITPIX  =                    8 / array data type
NAXIS   =                    0 / number of array dimensions
EXTEND  =                    T
IDENT   = 'LMC_SC2_12'
RA      =    82.57134000000001
RA_UN   = 'deg     '
DEC     =             -70.30103
DEC_UN  = 'deg     '
CLASS   = ''
HIERARCH OgleII_name = 'LMC_SC2_12'
HIERARCH OgleII_id_field = 'LMC_SC2 '
HIERARCH OgleII_id_target = 'lmc     '
HIERARCH OgleII_id_starid = '12      '
B_MAG   =                18.011
I_MAG   =                14.566
V_MAG   =                16.383
```

Figure 1.6: The header of a PrimaryHDU of a star downloaded from OGLE II.

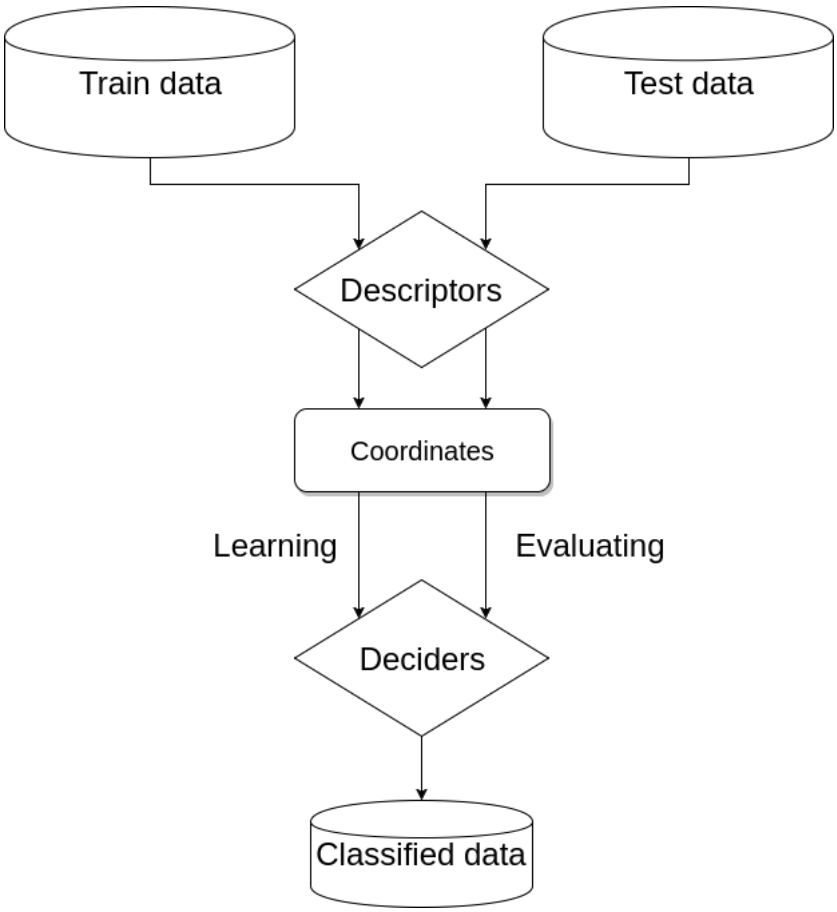


Figure 1.5: Filtering pipeline workflow: Input data are split into train and test samples. The descriptors extract feature vectors (coordinates in the feature space) for both training and testing samples. The deciders are first trained on the training features, and then used to estimate the precision on the test sample.

Stars can be downloaded by command line interface with the following command:

```
lcc filter_stars
    -q query.txt
    -r output_folder
    -d db_name
```

query.txt is a csv-like file with particular queries, *output_filder* is the name of the folder where result FITS files are saved and *db_name* is the name of the database from which stars are downloaded.

The same result can be achieved using the package directly:

```
client = StarsProvider.getProvider(db_name,
                                   queries)
stars = client.getStars()
```

Listing 1.1: Obtaining of stars

The syntax for querying particular databases is the same. Just database names and queries can differ. However syntax for coordinates cone search can be used for all databases (see 1.2.1).

For more extensive searches it is recommended to use *StarsSearcherRedis* which implements Redis job queue. Redis database is employed as a storage of jobs for workers. Since pickled jobs are stored in the persistent database, even after turning the computer off they are still queued.

```
searcher = StarsSearcherRedis(star_filters,
                              db_connector="OgleII")

queries = [{"field": "LMC_SC3",
            "starid": i} for i in range(1, 10000)]
searcher.queryStars(queries)
stars = searcher.getStars()
```

Listing 1.2: Obtaining of stars

The example below shows a query for the nearest object to the given coordinates.

```
queries = {"ra": 75.456,
           "dec": -70.214,
           "delta": 10,
           "nearest":True}
```

Listing 1.3: A sample query with a cone search

OGLE II BVI database

The Optical Gravitational Lensing Experiment [17] provides light curves in I-band of more than 40 million objects in the Magellanic Clouds and the Galactic bulge.

```
queries = {"starid" : 1456,
           "field" : "LMC_SC2"}
```

Listing 1.4: OGLE II sample query

OGLE III Catalog of Variable Stars

The main goal of this catalogue is to record all sources from the OGLE III [18] fields - the Magellanic Clouds and the Galactic bulge. It contains photometry in V band, I bands and the periods of the sources. The data include several types of objects:

- classical Cepheids in the Galactic Bulge, LMC and SMC
- type II Cepheids in the Galactic Bulge, LMC and SMC
- anomalous Cepheids in LMC and SMC
- RR Lyrae stars in the Galactic Bulge, LMC and SMC
- Long Period Variables in the Galactic Bulge, LMC and SMC
- Double Period Variables in LMC
- R CrB stars in LMC
- δ Sct stars in LMC

All these types can be specified as the input parameters of the query.

```
queries = {"ra": 69.329, "dec": -69.81986,
           "delta": 5, "nearest": True,
           "types": ["ACep", "LPV"]}
```

Listing 1.5: OGLE III sample query

CoRoT Archive

A space mission [23] dedicated to the exoplanet search and stellar physics. Light curve files are available on the CoRoT scientific archives website [24]. The catalog is also available in Vizier [25]. There are two separated tables: *Bright stars* (171 rows) and *Faint stars* (177382 rows). Light curves include no calculated magnitudes, just fluxes.

```
queries = {"CoRot": 116}
```

Listing 1.6: CoRoT query example

Kepler Archive

Kepler Objects of Interest catalogue [19] consists of stars observed by Kepler spacecraft. These objects were selected based on suspicion of hosting one or more transiting planets. The connector is based on the implementation of *kplr* [26] package.

```
queries = {"kic_num": 9787239}
```

Listing 1.7: Kepler Archive sample query

Catalina Archive

Catalina Archive consists of light curves from the second public data release (CSDR2) from the Catalina Surveys [20]. It contains photometry of a 7 -year observation taken by the CSS telescopes. The database includes about 500 million objects with the median of 80 measurements per object. It covers the area of 33 square degrees of objects between $-75^\circ < \text{Dec} < 70^\circ$ and $|b| > 15^\circ$ and with V magnitudes between 11.5 mag and 21.5 mag.

```
queries = {"id": 1001006023682}
```

Listing 1.8: Catalina Archive sample query

MACHO

The primary aim of this project is to test the hypothesis that a significant fraction of the dark matter in the halo of the Milky Way is made up of objects like brown dwarfs or planets, which can be revealed for example by microlensing events. [21]

The package uses the database of variable stars in the Large Magellanic Clouds contains 21474 objects.

```
queries = {"Field": 1 ,
           "Tile": 3441,
           "Seqn": 25}
```

Listing 1.9: MACHO query example

ASAS Catalog of Variable Stars

The All Sky Automated Survey (ASAS) is a low-cost project dedicated to constant photometric monitoring of the whole available sky, which is approximately 10^7 stars brighter than14 magnitude. The project's ultimate goal is detection and investigation of any kind of the photometric variability. One of the main objectives of ASAS is to find and catalogue variable stars. [22]

The package uses ASAS Catalog of Variable Stars which contains 50122 objects.

```
queries = {"ASAS": "000030-3937.5"}
```

Listing 1.10: ASAS sample query

File Manager

Files stored locally can be queried in the same way as other connectors. The *File Manager* can handle three types of files:

1. FITS files
2. Text files
3. Serialized Python star objects

```
queries = {"path": "/tmp/my_stars"}
```

Listing 1.11: Files sample query

1.2.2 Features extractors

The main purpose of the program is to search a big storage of light curves and look for objects of interest which is a quite challenging task. One of the most important steps is the characterization and description of a particular light curve. It can be done by their physical or statistical properties. For example colour indexes, skewness, variance, dissimilarity of a

light curve from the template, etc. Having specified the features, all stars can be described with the associated feature vectors.

All these tasks are reflected in the implementation of particular descriptors, which can be divided into two main groups:

1. Physical descriptors
2. Statistical descriptors

Physical descriptors

They extract features related to the physical properties of the stars.

A Short Introduction to SAX

Since SAX method is used in many descriptors, a few paragraphs are dedicated to this topic. The SAX method [27] transforms a sequential numeric data to a string representation.

The procedure follows simple steps. Firstly, the number of points of a light curve n is reduced. Secondly, the data are binned into w bins, where each value of a representative point corresponds to the average in a given bin window. This procedure is also known as Piecewise Aggregate Approximation (PAA) [28]. Note that modified version of PAA can be used to preserve time domain by averaging magnitudes for particular time buckets (see 1.2.2).

This step is necessary for simplification of the light curve L and the reduction of the points used for the string representation S. Also, it helps us to suppress the influence of the noise present in the dataset. The number of representative points w must reflect the time domain range of the searched variability. Such variability must be preserved in the representative data points.

$$L = l_j \Rightarrow S = s_i, \tag{1.1}$$

where indexes $j = 0\ldots,n$ and $i = 0,\ldots,w$.

Then we assign a letter to each bin value according to the specific rules. Namely, for the given size of the alphabet A we calculate breakpoints β_k with k running from 0 to A. Those breakpoints produced A equal-sized areas under a Gaussian curve. By definition, the breakpoints are a sorted list of numbers $B = \beta_1,\ldots,\beta_{A-1}$ such that the area under a $N(0,1)$ Gaussian curve from β_i to β_{i+1} is equal to $1/A$. (β_0 and β_A are defined as $-\infty$ and ∞). The word representation of the light curve W is obtained as follows

$$w_i = \alpha_k \quad if \quad \beta_{k-1} \leq b_i < \beta_k, \tag{1.2}$$

where α_k is k-letter from alphabet. We now have a string representation of a light curve. Not only does this method works with the light curves, but also with series such as histograms or others. Thus we have switched from a curve similarity detection problem to the problem of searching similarities between words. As a measure of similarity for this purpose, a specific distance measure which can be constructed using a lookup table was used. The more detailed explanation is in [27]. A smaller value of the distance measurement means

more similar light curves, where zero distance means almost identical light curves. Light curves can have different lengths and timescales, which can be an issue since SAX needs to have two aligned light curves of the same length. Therefore the package uses *days_per_bin* ratio which keeps the time scale. In case of comparing light curves with different lengths, the shorter light curve is successively compared to the parts of the same length of the longer light curve. Then the best match is taken as the dissimilarity of the two light curves. It is important to note that SAX does not preserve the scale of light curve's magnitudes. This could be beneficial in most cases, as the observed magnitudes depend on distance.

Processing of light curves

Light curves data are from their nature noisy and often they contain missing values. Therefore they need to be preprocessed first. More thorough analysis about processing light curves data is done in the section 3.1.1, where data are prepared for neural networks.

In this section, we will not use neural networks to feed them by the whole light curves, but rather calculate features for the machine learning models. It means that the demands for a preprocessing the data are much lower then it will be for neural networks in the chapter 3.

The only preprocessing method we need to use is PAA. It both smooth the light curves and fill some smaller gaps. Since features are calculated from the light curves, we do not need to replace missing values. For example variance of a light curve does not change much if few measurements are missing.

Curve Shape Descriptor

The descriptor measures dissimilarity of the inspected light curve and the template light curves. The comparison is executed in a symbolic representation using the SAX method described above.

Firstly, the dimension of the light curves of the length n is reduced into a new light curve of the length n' according to the *days_per_bin* ratio. So the length of the result words depends on the time scale of the measurement. While comparing two words of different sizes, the shorter word shifts through the longer word, letter by letter in order to find the best match.

The template can consist of multiple light curves. The dissimilarity of the inspected light curve is calculated as the mean value (by default, it can also be the lowest dissimilarity or for example mean of the best 5 matches).

Histogram Shape Descriptor

Another member of the family of "comparative descriptors" is the Histogram Shape Descriptor. It also uses symbolic representation, but in this case, the histograms of the light curves are compared.

The histograms are normalized on zero mean magnitudes and unit standard deviation.

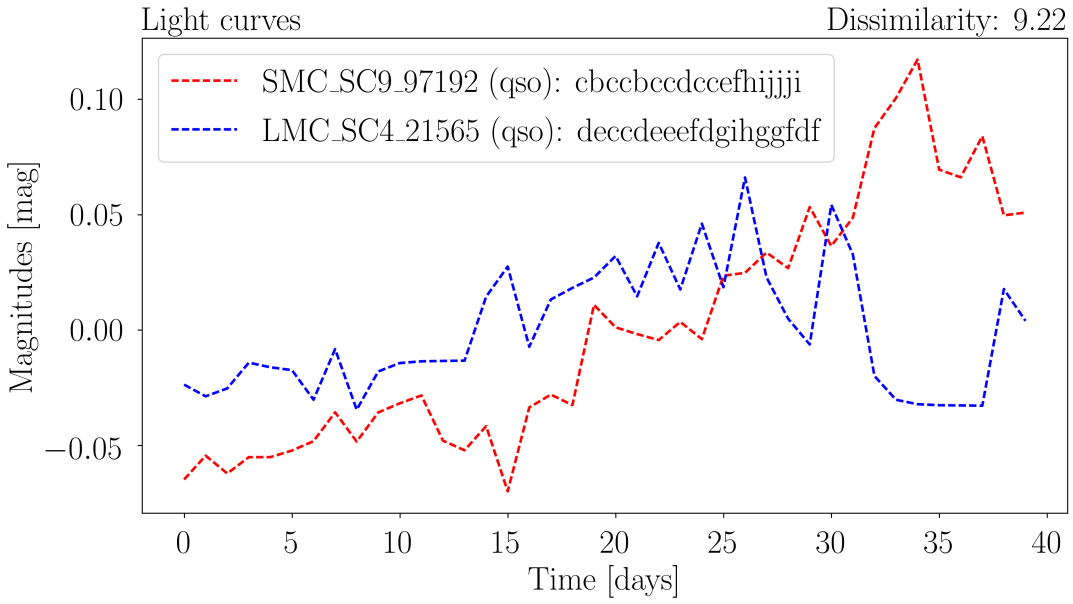


Figure 1.7: Comparison of light curves of two quasars: $days\_per\_bin = 60$, $alphabet\_size = 10$

Variogram Shape Descriptor

The Variogram Shape Descriptor uses symbolic space to compare the variograms of two light curves. Variogram is a function showing the differences of magnitudes in different time lags. Consider a light curve $I(t)$ with time lags defined as $\Delta t_{ij} = t_i - t_j$ and magnitude difference defined as $\Delta I_{ij} = \left(I_i - I_j\right)^2$. Hence function $\Delta I_{ij}(\Delta t_{ij})$ is designated as the variogram.

Color Index Descriptor

The *Color Index Descriptor* extracts the mean magnitudes of the light curves in certain bands.

Curve Descriptor

A Light curve can be also represented as the vector of magnitudes. To achieve that, the light curve has to be transformed in the way that the distances between the times of measurements are the same. This can be done, using, for example, PAA which creates a new light curve by averaging certain time windows. However, the gaps between the measurements can be that long that it is impossible to use the method.

Nevertheless, in most cases it can be applied successfully. After making the light curve equidistant we can omit the time vector, because the time information is already encoded in the gaps between the values in the magnitude vector. For example, a light curve described with time and magnitude vectors, $\vec{t}$ and $\vec{m}$ is transformed into a magnitude vector $\vec{m'}$ as follows:

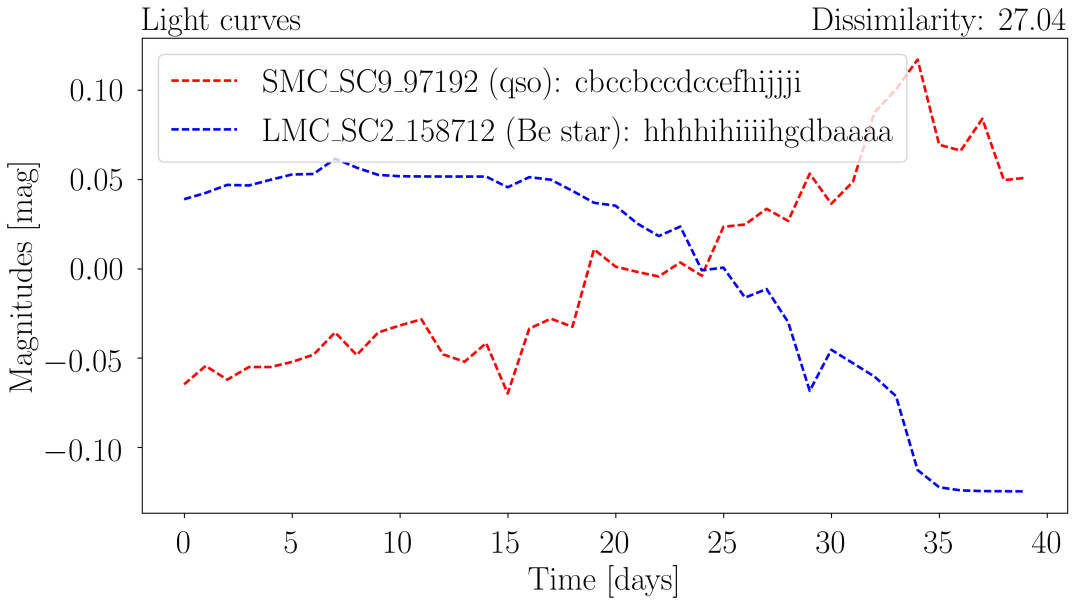


Figure 1.8: Comparison of light curves of a quasar and a Be star: $days\_per\_bin = 60$, $alphabet\_size = 1$

$$\vec{t} = [10, 20, 29, 30, 31]$$
$$\vec{m} = [1, 5, 8, 9, 10]$$
$$\rightarrow \vec{m'} = [1, 5, 9]$$

Consequently, the dimension of an equidistant magnitude vector can be reduced by binning — averaging of the adjacent values. The reduced light curve can be either directly used as features for a neuron network to analyze, or it can be reduced again with the reduction dimensionality methods such as Principal component analysis.

Property Descriptor

Stars obtained from the astronomical databases contain various attributes such as colour indexes, temperatures, distances, mass etc. The Property Descriptor formally extracts these metadata features.

Position Descriptor

The position in the sky can be also an important feature for some cases. For example, it can be used (with other features) to predict the probability of membership of a star into a cluster of stars.

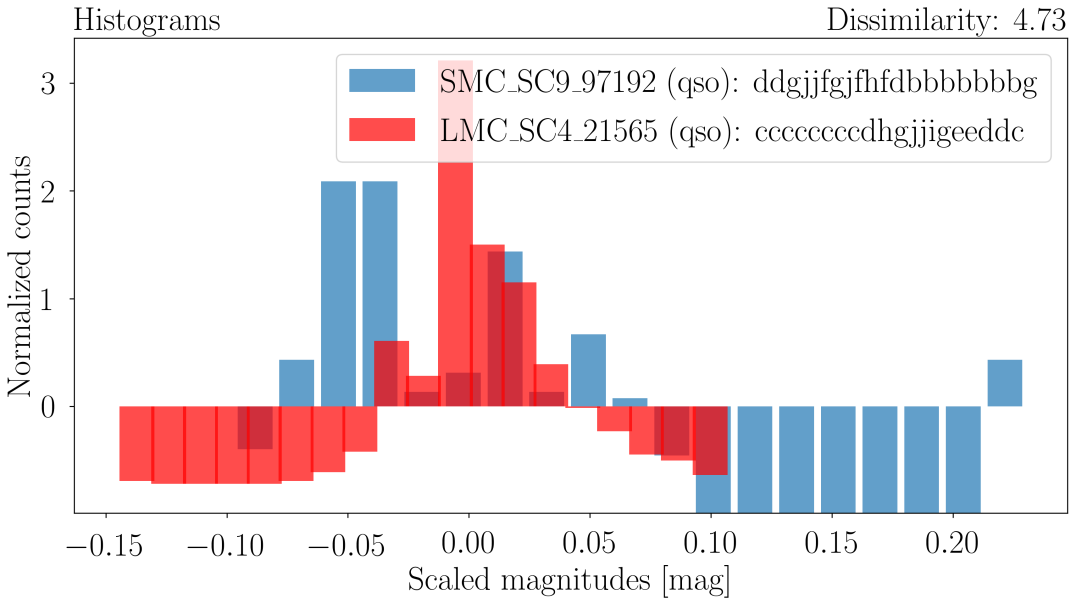


Figure 1.9: Comparison of histograms of two quasars: $bins = 20$, $alphabet\_size = 10$

Statistical descriptors

These descriptors calculate statistical values from the light curves.

Abbe Value Descriptor

Parameter $\mathscr{A}$ - the Abbe value, is defined as a fraction of the mean square successive difference and variance:

$$\mathscr{A} = \frac{n}{n-1} \frac{\sum_{\mu=1}^{n-1} \left(x_{\mu+1} - x_\mu\right)^2}{\sum_{\mu=1}^{n} \left(x_\mu - \bar{x}\right)^2}, \tag{1.3}$$

As it was shown by [29], this statistic can serve as a criterion whether observations are independent or correlated. Correlated light curves have $\mathscr{A}$ close to 0, while, non-variable light curves have $\mathscr{A}$ close to one. In case of searching variable light curves, we can provide a train sample of light curves to allow the filter to find the limit value.

Kurtosis Descriptor

This descriptor quantifies whether the shape of the data distribution matches the Gaussian distribution. The data with Gaussian distribution have a kurtosis of zero (since excess kurtosis is used), the data with a flatter distribution have a negative kurtosis and more peaked distributions have a positive kurtosis. For N samples data $\vec{X} = x_i, ..., x_N$ with the mean value $\bar{x}$ and standard deviation s, the kurtosis can be computed as follows:

$$k = \frac{1}{N} \frac{\sum_{i=1}^{N} \left(x_i - \bar{x}\right)^4}{s^4} - 3 \tag{1.4}$$

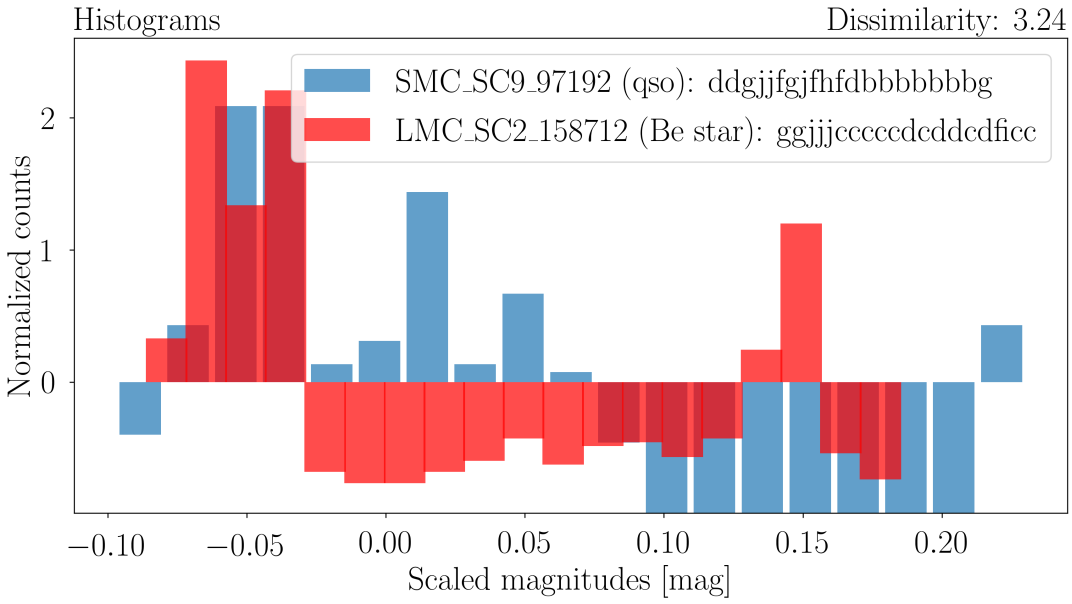


Figure 1.10: Comparison of histograms of a quasar and a Be star: $bins = 20$, $alphabet\_size = 10$

Skewness Descriptor

This descriptor quantifies how symmetrical the data distribution is. A symmetrical distribution has a skewness of zero, an asymmetrical distribution with a long tail to the right (higher values) has a positive skewness, an asymmetrical distribution with a long tail to the left (lower values) has a negative skewness. For n samples data $\vec{X} = x_i, ..., x_N$ with mean value $\bar{x}$ and standard deviation s, the skewness can be computed as follows:

$$k = \frac{1}{\sqrt{N(N-1)}} \frac{\sum_{i=1}^{N} (x_i - \bar{x})^3}{s^3} \tag{1.5}$$

Variogram Slope Descriptor

Light curves with trends have graduating variograms which can be fitted in a linear function $f(x) = ax + b$. The slope represents the steepness of the magnitude changes in different time lags (see image 1.11). Such criterion can be used for example for the classification of quasars [30].

Reduced Curve Descriptor

In the section 1.2.2 it was shown that the light curves can be represented as vectors. They can be handled in many ways such as introduced symbolic representation for comparing with a template or as the input for a convolutional neural network. The whole vector is not generally suitable for machine learning methods. However we can use for example PCA to reduce the dimension of the vector, but with preserving as much information as possible.

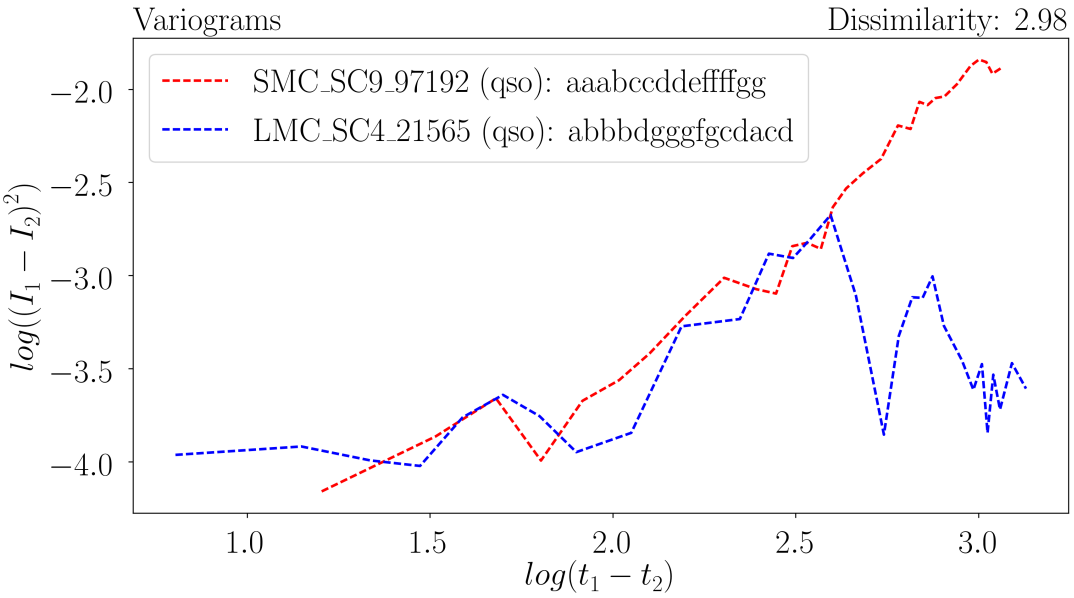


Figure 1.11: Comparison of variograms of two quasars: $bins = 15$, $alphabet\_size = 7$

1.2.3 Deciders

Deciders (classifiers) are classification machine learning methods. After the training, the deciders are able to calculate probabilities with which the inspected stars belong to the searched group of objects. See the figures 1.13 or 1.15.

For many cases, it is suitable to use more deciders simultaneously. The resulting probability can be then combined.

Neuron Decider

The Neuron Decider is simple *Keras* [31] implementation of the feed-forward neural network with two hidden layers. Moreover, the architecture of the classifier can be upgraded into much more complex.

Also, a new decider with a convolutional neural network (CNN) is about to become a part of the new version of the package. In this approach, the whole light curve is processed and taken without any feature extraction. It means that no information is lost. All the patterns can be learned with the arbitrary number of convolutional filters which are automatically build up.

The main issues with applying neural networks to the whole light curves are missing data and light curves of different lengths. So the CNN cannot be used as an ultimate solution, but in some cases such as the case of light curves with no big gaps, it could be a very powerful instrument.

Supervised scikit-learn classifiers

There are many classifiers included in the package from scikit-learn [32].

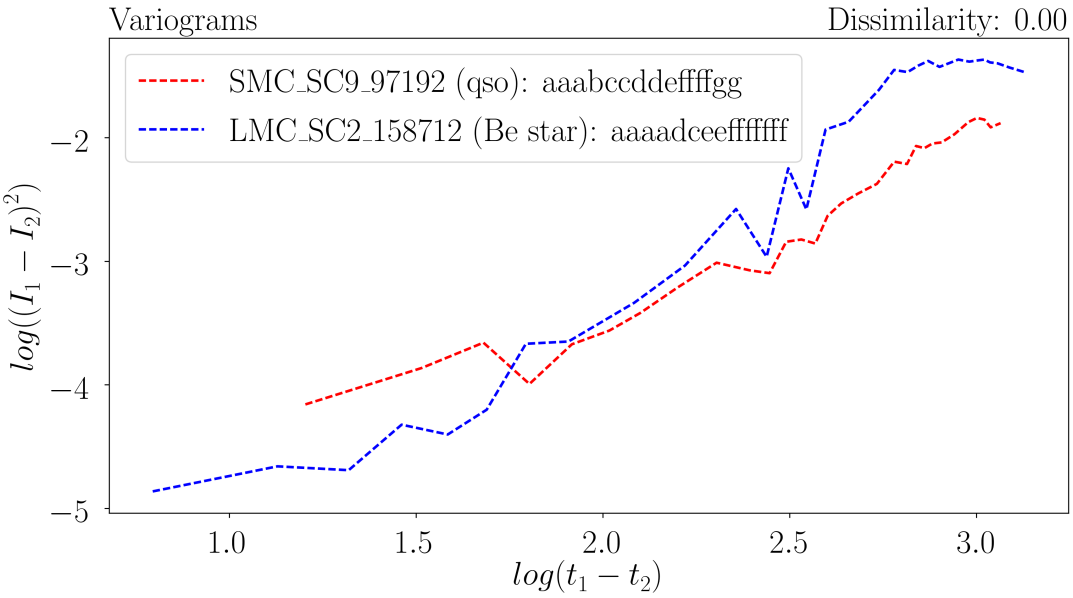


Figure 1.12: Comparison of variograms of a quasar and a Be star: $bins = 15$, $alphabet\_size = 7$

- Linear Discriminant Analysis
- Quadratic Discriminant Analysis
- Naive Bayes
- Gaussian Naive Bayes
- Support Vector Machine
- Decision Tree
- Random Forest
- Gradient Boosting
- Extra Trees
- AdaBoost

Custom Decider

This pseudo decider can be used to classify stars based on the user's inputs ranges. Its purpose is to allow the users to set the boundaries manually.

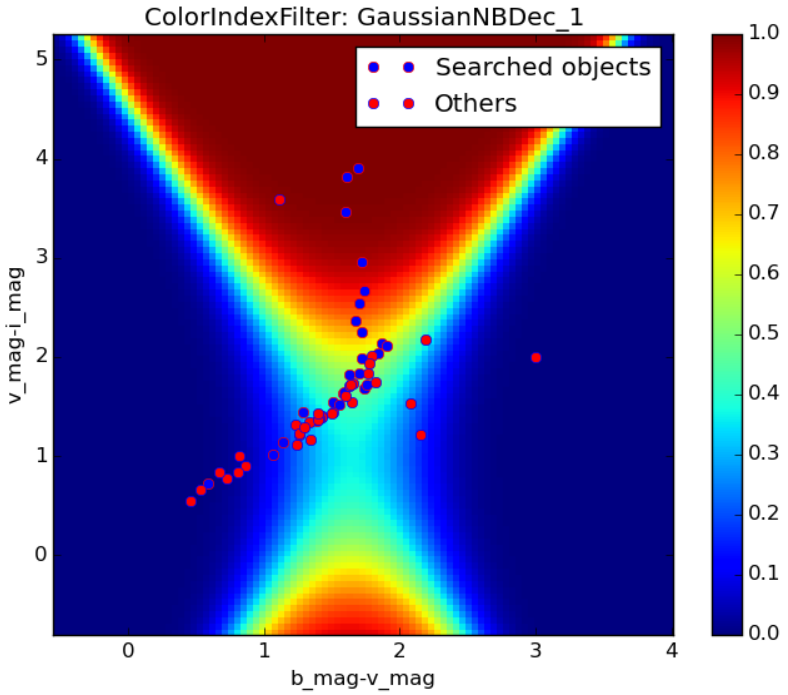


Figure 1.13: Random stars from Ogle II. The color background represents the probability of membership in the searched group (the blue dots).

1.2.4 Filter

A filter is the core object of the star classification process. It is composed of descriptors and deciders. Once it is defined it can be used to tune parameters, train the models and finally make predictions to inspected stars. The sample code below demonstrates the philosophy of filtering.

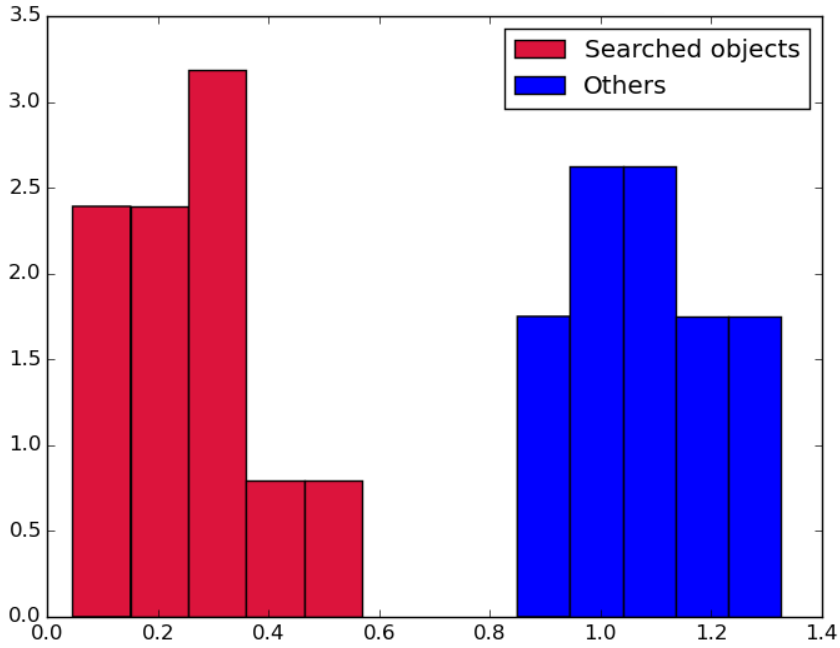


Figure 1.14: Distribution of $\mathscr{A}$ values for sample of quasars (red) and non-variable stars (blue)

```
bins = 50

# Loading deciders and descriptors
deciders = [LDADec(), SVCDec()]
descriptors = [AbbeValueDescr(bins), HistShapeDescr(bins)]

# Construction of the filter
stars_filter = StarsFilter(deciders, descriptors)

# Training the filter on the train sample
# searched_stars - stars we are looking for
# contamination stars - some other stars
stars_filter.learn(searched_stars, contamination_stars)

# Filtering of the inspected stars
passed_stars = stars_filter.filterStars(inspected_stars)
```

Listing 1.12: Star filter is constructed by deciders and descriptors. Consequently, it is trained with a sample of searched and contamination objects. After training completion, the filter is capable of filtering stars

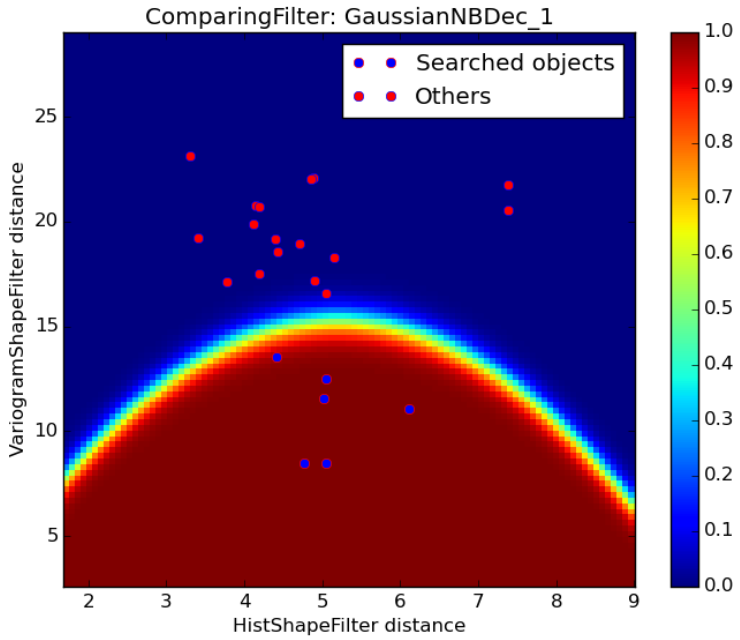


Figure 1.15: Histogram-variogram space. The feature coordinates represent dissimilarity of the inspected stars from the template sample. The background probability distribution was generated by Gaussian Naive Bayes decider.

1.2.5 Estimating the optimal parameters

Descriptors and deciders can contain parameters which have to be tuned. Tuning can be done automatically with the *ParamsEstimator*. All combinations of the parameters are evaluated, and the best ones are adopted for constructing the filter. All the combinations are then described with statistical values (true/false positive/negative rates, accuracy, etc.) used for the evaluation of particular combinations. By default, precision is used as the criterion:

$$PPV = \frac{TP}{TP+FP}. \tag{1.6}$$

where TP is true positive and FP is false positive. For an example code of tuning parameters see 3.3.4.

1.3 Usage

A general workflow using the command line application:

1. Create a project
2. Prepare combinations of parameters for tuning descriptors and deciders
3. Prepare queries for a database

4. Make a filter

5. Execute the systematic search in a database

Each step corresponds to the single execution of a script.

Example

In this example, our goal is to search the OGLE II and look for the light curves of the shape similar to our sample of light curves. For this purpose we can use *CurvesShapeDescriptor* which tells us how much the inspected light curves are dissimilar to the template light curves. Then we need a classifier which could be for example the *Quadratic Discriminant Analysis*. Note that the classifier has to be selected based on the complexity of data and separation of the classes.

Project preparation

Let's start by creating a new project *ExampleProject* in the current directory.

```
lcc create_project ExampleProject
```

Preparation of queries

Before running the systematic filtering in a database, we have to find optimal parameters of the descriptors and deciders for our task. *Curve Shape Filter* has two parameters - the length of alphabet and the dimension reduction ratio — a number of bins per day). Moreover, the decider has one more parameter which is *treshold*. A file with the parameters combinations can be generated by this command:

```
lcc prepare_query
    -o params_comb.txt
    -p CurvesShapeDescr:days_per_bin
        -r 50:150:15
    -p CurvesShapeDescr:alphabet_size
        -r 8:19:2
    -p QDADec:threshold -r 0.1:0.99:0.08
    -f t
```

Also a file for the database query can be generated similarly:

```
lcc prepare_query
    -o ogle_query.txt
    -p target -r lmc
    -p starid -r 1:100
    -p field_num -r 1:3
    -f q
```

Creating the filter

Now, we can train a filter with tuned parameters by a single command:

```
lcc make_filter
    -i params_comb.txt
    -s my_lcs_folder
    -c contamination_lcs_folder
    -t template_lcs_folder
    -f CurvesShapeDescr
    -d QDADec
    -n MyCurveShape
```

Of all the parameter combinations in *params_comb.txt*, the best one is stored as a filter object file *MyCurveShape_filter*. *QDADec* is used as a decider and light curves stored locally in *my_lcs_folder* and *contamination_lcs_folder* are loaded as train and contamination sample.

Systematic search

Finally, the created filter can be employed to systematically filter light curves in the OGLE II:

```
lcc filter_stars
    -q query_ogle.txt
    -d OgleII
    -f MyCurveShape/MyCurveShape.filter
    -r FirstRun
```

The status files and light curves are saved into the *FirstRun* folder in the project directory.

Result files

The performance of the filter is described with various files such as the ROC curve, the status file (a file with the statistics of every combination), the histograms of the distribution of described values, and the probability surface plot.

During the filtering, a status file is updated with information about the progress of searching/filtering.

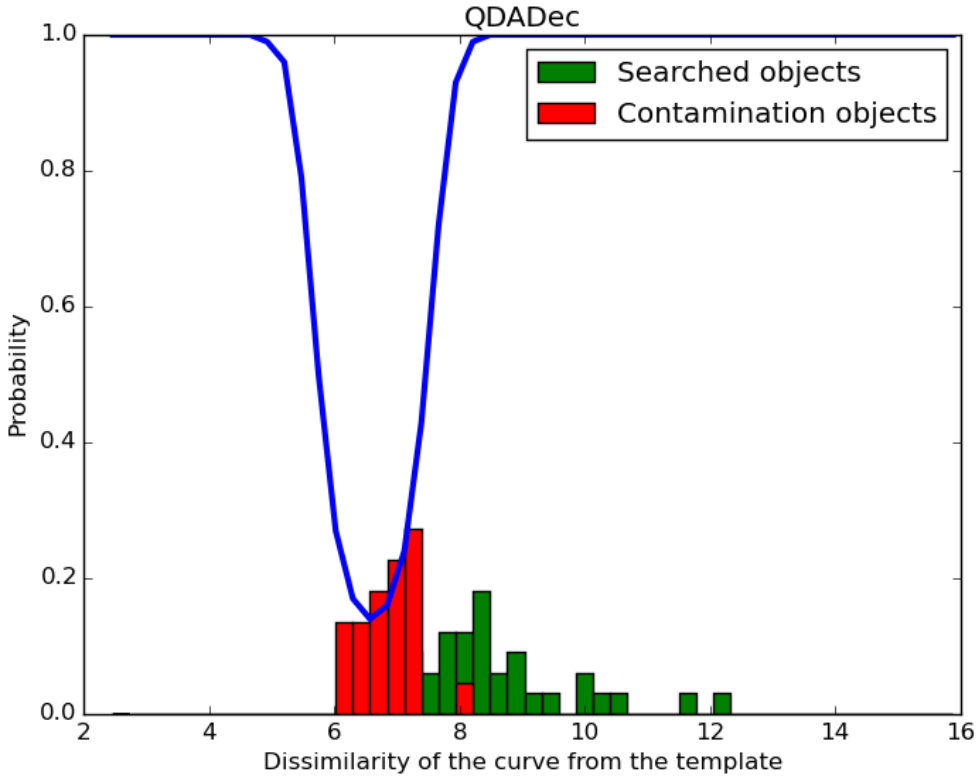


Figure 1.16: The probability plot created with a trained curve shape filter with training sample

#starid	target	name	found	filtered	passed
2	lmc	LMC_SC2_2	True	True	True
3	lmc	LMC_SC2_3	True	True	False
5	lmc	LMC_SC2_5	True	True	True
6	lmc	LMC_SC2_6	False	False	False
7	lmc	LMC_SC2_7	True	True	False

Table 1.1: Status file generated during filtering OGLE II. *found* column says whether the queried object was found, *filtered* says whether the object has a light curve and *passed* says whether the object passed through the filtering.

Chapter 2

Quasars classification

Quasars are objects suitable for developing and testing new techniques and methodologies, because of the irregular variations of the magnitudes and the fact that physical phenomena behind luminosity of quasars are still poorly understood.

In this chapter, the package is employed to classify quasars in the sample of Be stars, periodically variable stars and non-variable stars. Firstly the trained model is evaluated on the test sample, secondly, it is compared with results of Eyer [30], who classified quasars and Be stars by variograms slope.

2.1 Data

Spectroscopically confirmed quasars were gathered from multiple sources. The biggest source of quasars was HMQ which is a catalogue of almost million confirmed quasars. However it does not contain light curves data, hence it was crossmatched by coordinates with OGLE II. Quasars sources:

- MACHO Quasars Behind the Magellanic Clouds [33]
  - 59 quasars from MACHO (by identifier)
- The Magellanic Quasars Survey (MQS) [34]
  - 25 quasars from OGLE II (by identifier)
  - 1 quasar from MACHO (by identifier)
- The Half Million Quasars (HMQ) catalogue [35]
  - 94 quasars from OGLE II (by crossmatch)

Sources of the other stars:

- Be Stars [36]
  - 221 stars from the OGLE II
- Variable stars with double period [18]

  - 68 stars from the OGLE III

- Non variable stars
  - 232 random stars from the OGLE II

The sample was randomly split into the train and test samples (with the ratio of 0.7). Despite the fact, that it was gathered several hundred light curves from many sources, it is still not a lot. However, the main criterion is not the sample size, but its ratio to the number of used features and the complexity of the model. It means, that we cannot build for example a very deep neural network with thousands of features with such a small sample.

2.2 Model

The model consists of 7 descriptors which extract 8 features for each object. The selection of descriptors and their importance in the model is discussed in 2.2.1.

Parameters tuning

Despite the fact, that machine learning models can learn on training data by themselves, there are many parameters which have to to be set and can vary from task to task. Since most of the parameters are dependent on each other optimization of particular parameters need to be done together. However, there are some exceptions, namely SAX descriptors, because they provide an output which can be evaluated independently without a model – dissimilarity from the template. Hence our loss function can be defined as follows:

$$loss^{-1} = \frac{\sum_{i=1}^{n_{others}} y_i}{n_{others}} - \frac{\sum_{i=1}^{n_{qso}} x_i}{n_{qso}}, \tag{2.1}$$

where x is a set of particular dissimilarities of training quasars from the template quasars, y is a set of particular dissimilarities of training non-quasars from the template quasars, n_{qso} is a number of training quasars and n_{others} is a number of training non-quasars.

Graph 2.1 shows difference ($loss^{-1}$) of histogram shape for various *bins* (the length of the SAX world). A higher difference means better separability of these groups, hence better score for the given parameter. The other parameters can be tuned for example by the exhaustive Grid Search, Random Search or Genetic Optimization.

Descriptors

Since the classification of quasars, Be stars, periodically variable stars and non-variable stars is not an easy task, it is necessary to extract such features which differ for the particular classes. For example skewness, kurtosis and Abbe Value are important for classification of non-variable. Moreover, Abbe Value can distinguish periodically variable stars. It means that with these features we are able to easily find quasars and Be stars.

The most challenging part is to distinguish between quasars and Be stars. Since we know how light curves of some quasars look like, we can randomly select a template of

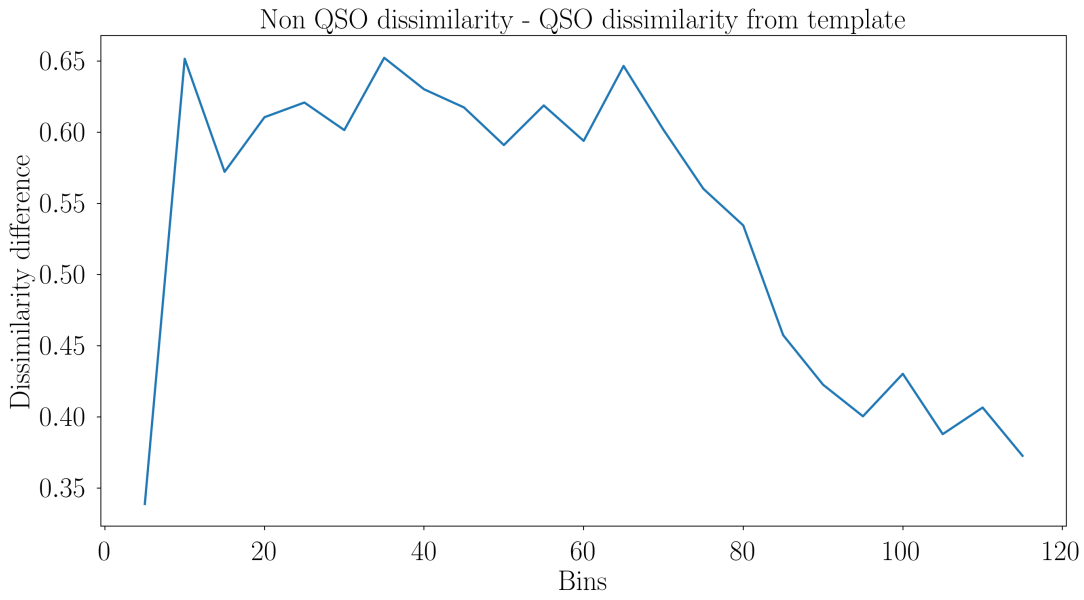


Figure 2.1: Dependence of histogram bins size for SAX precision.

quasars and compare inspected stars with these representatives. It is done in the symbolic representation as was discussed in the section 1.2.2.

Since some light curves of quasars look similarly, we can estimate dissimilarities of the light curves. However lots of them are unique, but the distribution of magnitudes can be similar to the others, hence we can compare even histograms. Variograms represent nature of the variability which can differ for quasars and Be stars. Eyer [30] has noticed it and used variogram slopes for quasars/Be stars classification. However, we can get even more information from the variograms then just their slopes. The third feature of the comparing can be variogram dissimilarity.

The last two features come from transformation light curves to two-dimensional space as was described in 1.2.2. The pipeline of descriptors with tuned parameters:

1. AbbeValueDescr
   - $bins = 80$
   - KurtosisDescr
   - $bins = 40$
2. SkewnessDescr
   - $bins = 40$
3. CurvesShapeDescr
   - $days\_per\_bin = 10$
   - $alphabet\_size = 18$

4. HistShapeDescr
   - $bins = 12$
   - $alphabet\_size = 12$
5. VariogramShapeDescr
   - $bins = 50$
   - $alphabet\_size = 12$
6. CurveDescr
   - $bins = 200$
   - $red\_dim = 2$

The extensive parameters tuning was performed at the web interface hosted at the computational cluster and took about 30 hours.

Color index

To complete the enumeration of methods for the classification of quasars and Be stars, it needs to be noted that colour index is one of the most efficient methods. It is no surprise because it represents an information about the distribution of radiation which is specific for particular object types. However, the descriptor is not used in the classification pipeline, because it is a parameter which is not retrieved from the light curves and also it is not available for most of the objects.

Classifier selection

Despite the fact that Random Forest (RF) is widely used in astronomy, Gradient Boosting (GB) currently belongs to the prominent ML methods for feature-based models. Table 2.1 shows, that for this task it even outperforms RF (see discussion below). Therefore it was used for quasars classification pipeline.

	Precision	Accuracy	F1-Score
GB	90.9	91.3	58.8
RF	100.0	89.5	41.4

Table 2.1: Comparison of Gradient Boosting and Random Forest

One of the most important metrics for this task are accuracy, precision and F1 Score. Accuracy is defined as fraction of the correct predictions:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{2.2}$$

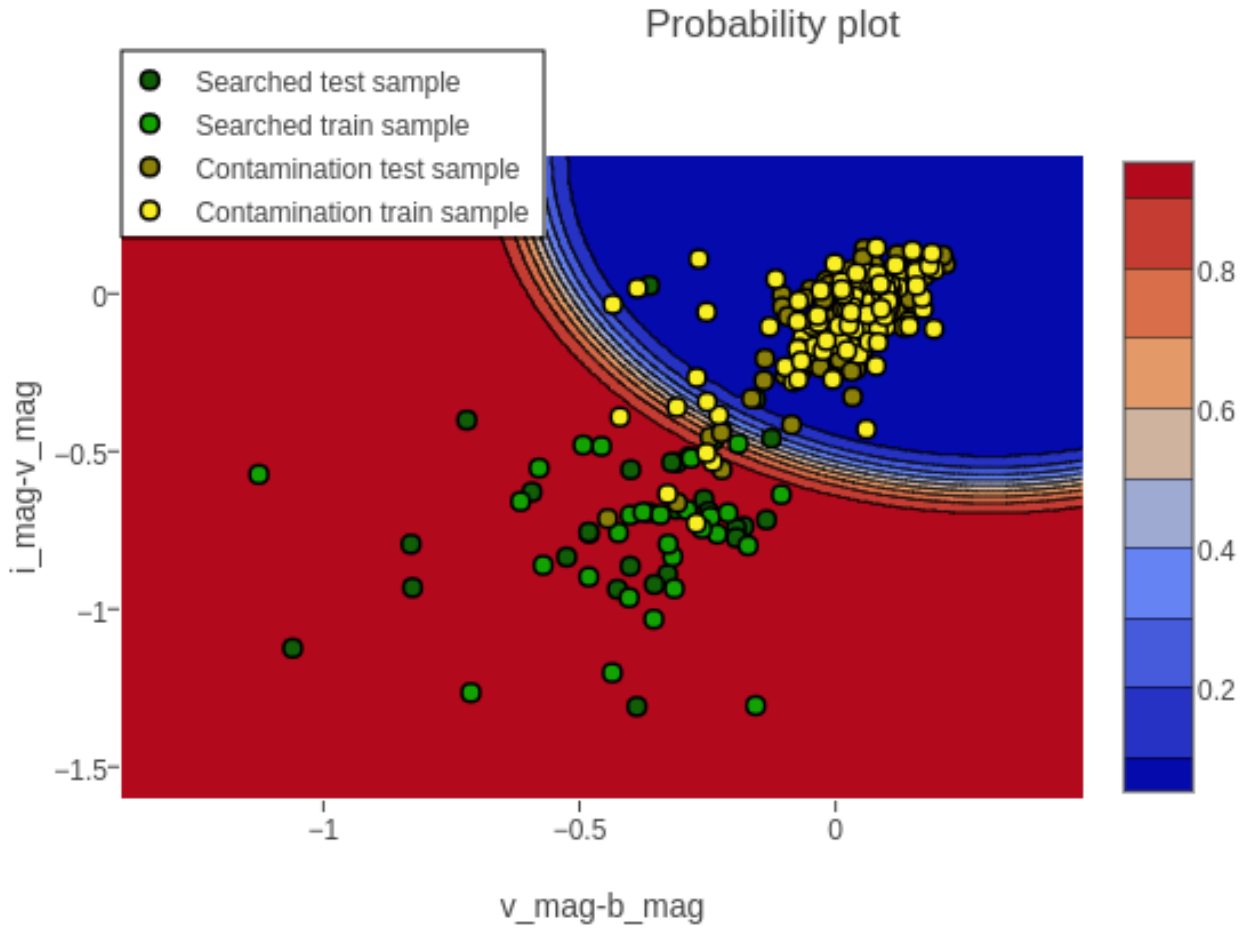


Figure 2.2: The color diagram of quasars (the searched sample) and Be Stars (the contamination sample). The color scale in the background represents the probability of being a quasar based on the color index.

where TP are true positives, TN are true negatives, FP are false positives and FN are false negatives. Precision is defined as fraction of correctly classified positive samples to the number off predicted positive samples:

$$precision = \frac{TP}{TP+FP}. \tag{2.3}$$

F1 Score is defined as the weighted average of precision and recall:

$$F1\_Score = 2\frac{recall \cdot precision}{recall + precision}, \tag{2.4}$$

where $recall = \frac{TP}{TP+FN}$. It takes into account both false positives and false negatives. Moreover, it can handle uneven distribution of classes in contrary to accuracy. Since we have much more negative samples, we can use this metric for selecting the model.

2.2.1 Evaluation

The test sample consists of 23 quasars and 139 non-quasars.

	True quasar	True non quasar
Predicted quasar	10	1
Predicted non quasar	13	138

Table 2.2: Evaluation of the model 1

Precision	90.9%
Accuracy	91.3%
F1-Score	58.8%

Table 2.3: Evaluation of the model 2

Roughly 43% of all evaluated quasars are expected to be correctly classified as quasars with contamination of 0.7% by other objects. It means that for a sample of 1000 quasars and 1000 other objects, the quasar's candidates sample will consist of 430 quasars and 7 non-quasars.

Feature importance

Importance of features in the decision tree based classifiers are proportional to the number of samples which were split by the nodes of particular features. Graph 2.3 shows importance of the features extracted by the descriptors:

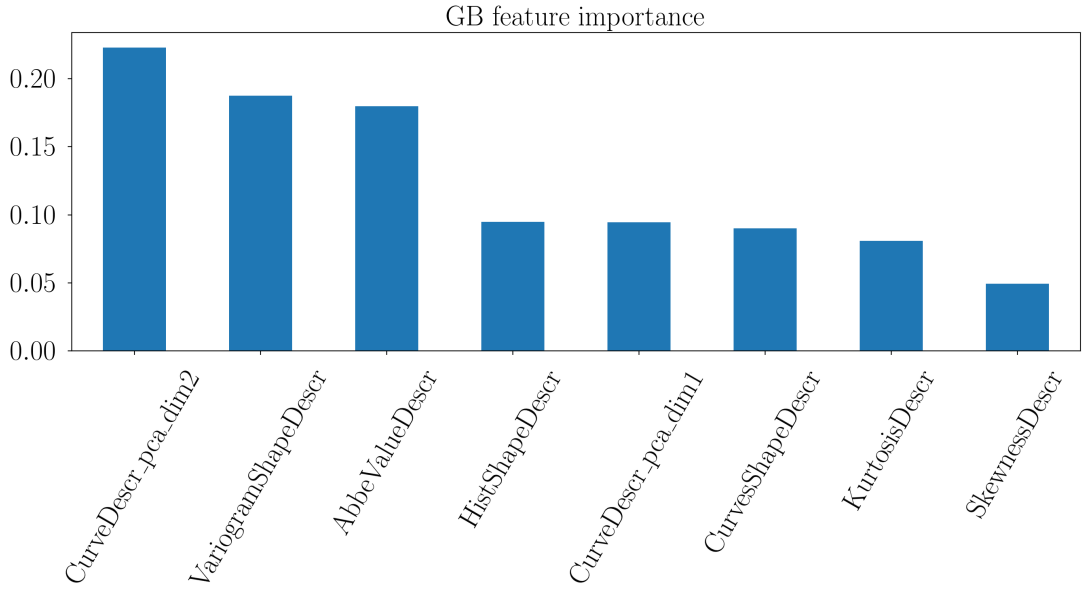


Figure 2.3: Feature importance of Gradient Boosting

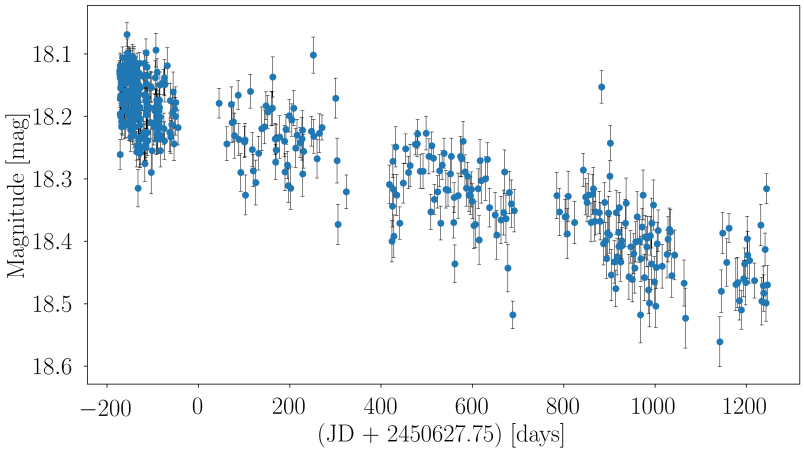
2.2.2 Testing on Eyer's data

The Eyer's sample consists of 133 quasars and 36 Be stars classified with variogram slopes and colour indexes. The trained model from the previous section was used to classify this sample. The threshold of the classifier is set strictly in order to get rid of as many false positive matches as possible. Lots of misclassified light curves of quasars look exactly the

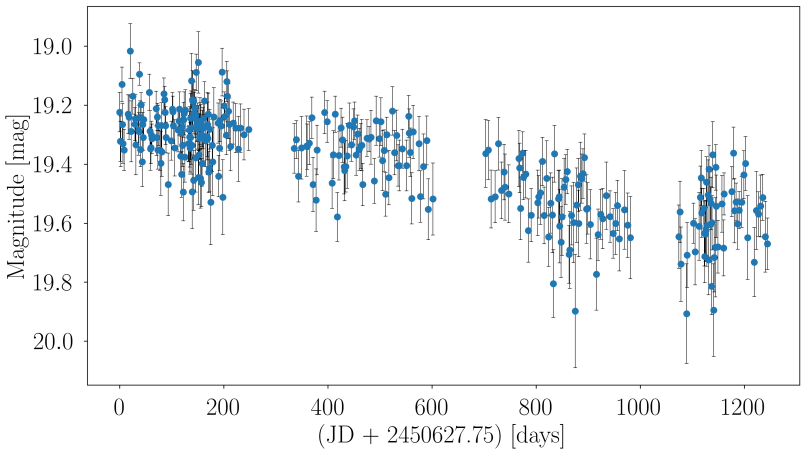
	True quasar	True Be star
Predicted quasar	75	3
Predicted Be star	58	33

Table 2.4: Evaluation of the Eyer's sample

same as the light curves of the Be stars and vice versa. Since these Eyer's quasars are not spectroscopically confirmed, the actual precision of the model could be much higher than it appears according to the table 2.4. However, the sample of objects predicted as quasars consists of 75 quasars and just 3 Be stars. Quite good result, when we consider that the threshold was set to minimize false positives.



(a) LMC_SC4_364710



(b) SMC_SC4_33863

Figure 2.4: The light curve a) was classified as quasar, but Eyer classified it as a Be star. b) Spectroscopically confirmed quasar (from Milliquas).

Chapter 3

Light curves and Neural Networks

Quasars are still mysterious objects and physical nature of their brightness variation is still not understood well. Since the shape of the light curves of the quasars differs a lot, it is very difficult to distinguish which features are characteristic for them.

3.1 Data

Nowadays neural networks are one of the most effective methods for sequential data. However, for light curves classification, they are still not widely used mostly because of their sensitivity to data quality which puts great demands on data preprocessing. Moreover, in most of the cases, another more simple methods can be used, because of lower complexity of the task. For example, there are many methods for periodic variable stars based on the periodicity, hence it makes no sense to use neural networks.

Sources of astronomical data can vary a lot - from space surveys with perfect observation conditions to noisy data caused by the atmosphere. Moreover, ground observatories are limited by the daylight which prevents continuous observations. Despite the fact that space telescopes could track objects for a long time, it is not happening in the most of the cases due to busy observation schedules and occasional maintenance. Therefore measurements can contain huge gaps and can be sampled irregularly.

3.1.1 Features engineering

Data are a fuel for the machine learning and without proper data cleaning and their representation, it is almost impossible to build a good model. Therefore a great effort was dedicated to thoroughly process data. The processing pipeline is composed of the following steps:

1. Split data into training, testing and validation sample
2. Transform light curves to have the same sampling
3. Split particular light curves into many shorter light curves in order to get data of the same length
4. Omit light curves with lots of missing values

5. Replace missing measurements of light curves
6. Scale the data

Data split

Data were split into three groups:

- Training sample - 50%
  - used for the training of the discriminator
- Validation sample - 25%
  - used as metrics during training and as evaluation criterion for models architecture
- Testing sample - 25%
  - used for final evaluation of the models

Sampling transformation

Light curves were transformed by PAA in the same way as it was shown in 1.2.2. The transformation parameter is the ratio of days per one bin which was set as a compromise between the resolution needed for quasars classification and smoothing which fix some missing values.

The parameter was set to $days\_per\_bin = 3$ which provides resolution sufficient for quasars classification and on the other hand it smooths noisy light curves and "clean" missing values. Note that this approach was also used for estimating remaining parameters of data transformation.

Light curves split

Dimension for the input light curves was set to $length = 100 \rightarrow$ all light curves have the same length of $days\_per\_bin \cdot length = 300$ days (shorter ones were omitted). Splitting was done by moving the window with $sliding\_step = 4$ which means that from 653 stars were obtained 20271 light curves.

Removing low-quality data

Many missing values were "fixed" during the sampling transformation (see 3.1.1), but still lots of parts of the light curves are contaminated by the missing measurements. Moreover, light curves with more than 15 consequences missing values were marked as "beyond repair" and filtered out. Still, the longest gaps can be 45 days long which is a quite large gap to fill (see example light curve in the figure 3.3).

Data from OGLE II contains lots of missing values, hence it is no surprise that after the cleaning almost 70% light curves were filtered.

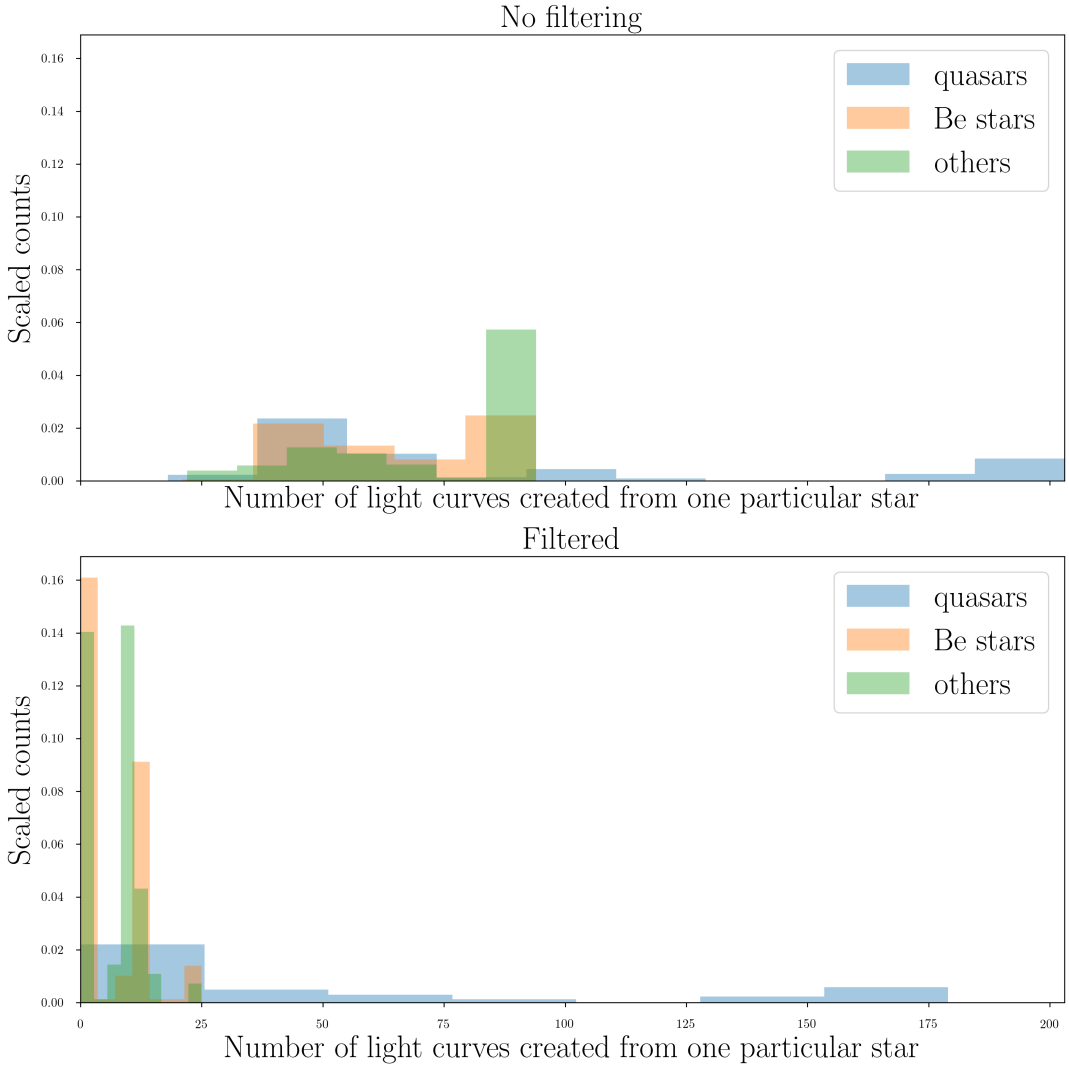


Figure 3.1: Amount of light curves created from one particular star – on the left side without filtering. On the right side with filtering of light curves with more than 15 subsequent missing values.

Missing values

The standard approach for interpolation of light curves data with a pattern is to employ sinusoidal harmonic fit. Since light curves of the quasars do not follow any permanent patterns, making such a fit would be inaccurate in the larger timescales.

The more complex approach could be a regression model trained on known light curves of known quasars to predict missing values. For example, the input of the model could be last n measurements and the output magnitude of the next measurements. This could work well for measurements with lower error, but for ground-based observations with so much noise, it is not the right choice. Training a machine learning model on data generated by an inaccurate model would just create a new source of the error.

Quasars brightness seems to change quite randomly, so it could imply to use a stochastic

method such as random walk to simulate the missing part of the light curve. Differences between consequent measurements of training light curves can be used as a set of walks. Consider a gap between two measurements which we want to fill by some amount of values. We can randomly select a walk, add it to the last value (first would be magnitude before the gap), fill the first missing by this value and so on. The procedure can be repeated multiple times and just these combinations which match measured values on both sides can be considered. However most of these light curves would look very chaotic and despite the fact that one of them can be very similar to the real one, the chance is low.

Methods mentioned above can theoretically give precise results, but there is still significant chance that the results would differ from the reality and cause that the light curves would appear as completely different objects for the classifier.

Since a convolutional neural network was used as the classifier, a conservative method would be the most accurate in terms of misclassification rate. The reason is properties of convolutional filters which are discussed in more details in the 3.2. Basically, their activation is proportional to the match of the filter with the light curve. Therefore we are looking for a fit which just does not differ a lot.

The method which was used in this work was *filling by weighted mean* – consider last point before the gap $[time_i, mag_i]$ and the first point after the gab $[time_{i+n}, mag_{i+n}]$, where i is index of the last measurement before the gap and n is number of missing values to fill. Then the formula for magnitude mag_{i+j} in time $time_{i+j}$ is following:

$$mag_{i+j} = \frac{\left(time_{i+j} - time_{i+j-1}\right) mag_{i+j-1} + \left(time_{i+n} - time_{i+j}\right) mag_{i+n}}{time_{i+n} - time_i}, \tag{3.1}$$

where $j > i$ and $j < i+n$.

Longer gaps of filled parts of the light curves can look artificially (see Figure 3.3) because the aim of this method is not to make a realistic fit, but rather minimize the error of the classifier. A quick fix could be adding noise to particular values from a random distribution.

This noticeable anomaly in the light curves can be spotted by the CNN and theoretically used as a proxy for classification. For example, if positive samples and negative samples would come from different sources, one with perfect dataset without gaps and the second one with many gaps, the model could learn to count artificially fixed gaps and used it as a shortcut for classification of these two classes.

Therefore it is necessary to investigate whether there are characteristic lengths of gaps in the training data across the classes which could lead to biasing of the model. Fortunately figure 3.2 shows that this is not the case – distribution of the gap lengths is very similar for all classes and there is no way how the model could be biased by this phenomenon.

Scaling and information preservation

Light curves have various scales of observed magnitudes caused by both different luminosity and different distance. The magnitudes of the data are relative, hence the absolute value of magnitudes can be omitted by adjusting brightness in the way that means magnitude is zero.

Next step is to scale ranges of magnitudes. In this case, dividing particular light curves by standard deviation (or any another scaling), one by one would cause losing information

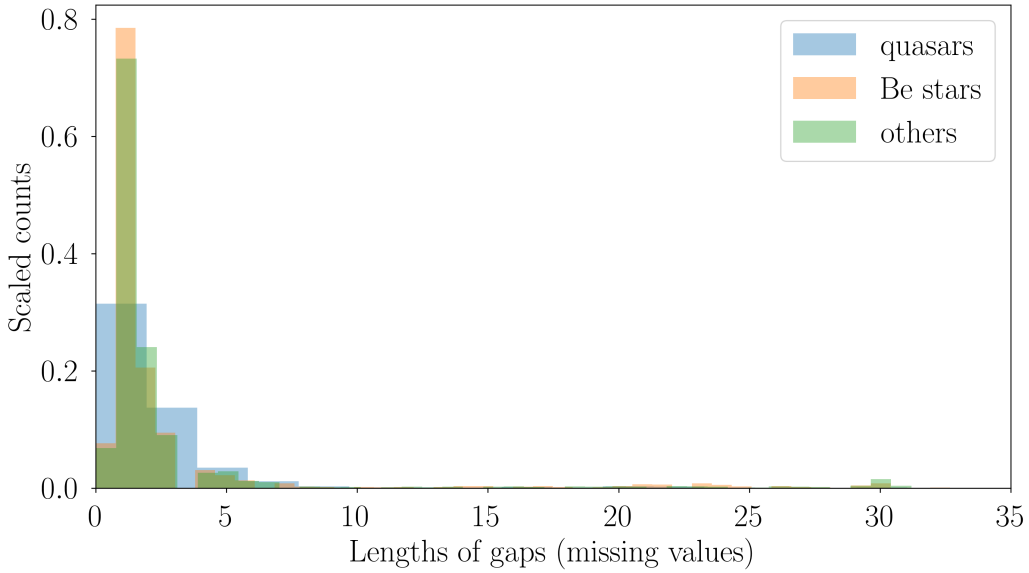


Figure 3.2: Lengths of sequences of missing values (gaps)

about the scale of the brightness variation. However, they can be scaled globally by computing global standard deviation and then dividing all magnitudes in the light curves. Note that similar process can be done if we also would like to preserve information about the absolute magnitudes. One important presumption we made for described global scaling is that all inputs (training, testing and evaluating) are expected to have variations of magnitudes in the similar ranges. Otherwise, we would have to scale particular light curves individually which would cause loss of the information about absolute scales.

3.1.2 Data quality

Quality of the machine learning models is proportional to the quality of the data. One cannot build a perfect model trained on biased and noisy data with missing values. Despite to all the all efforts which were done to clean the data, they are still far from perfect (see image 3.3). In this point, we are hitting the limit given by the precision of measuring instruments and observation conditions. Hence there is nothing more what can be done to improve quality of our data.

3.2 Convolutional Neural Network

Convolutional Neural Network (CNN) is the standard architecture for image related machine learning tasks. However, it is still quite new approach in the field of the one-dimensional data and even more uncommon in the astrophysics of the light curves.

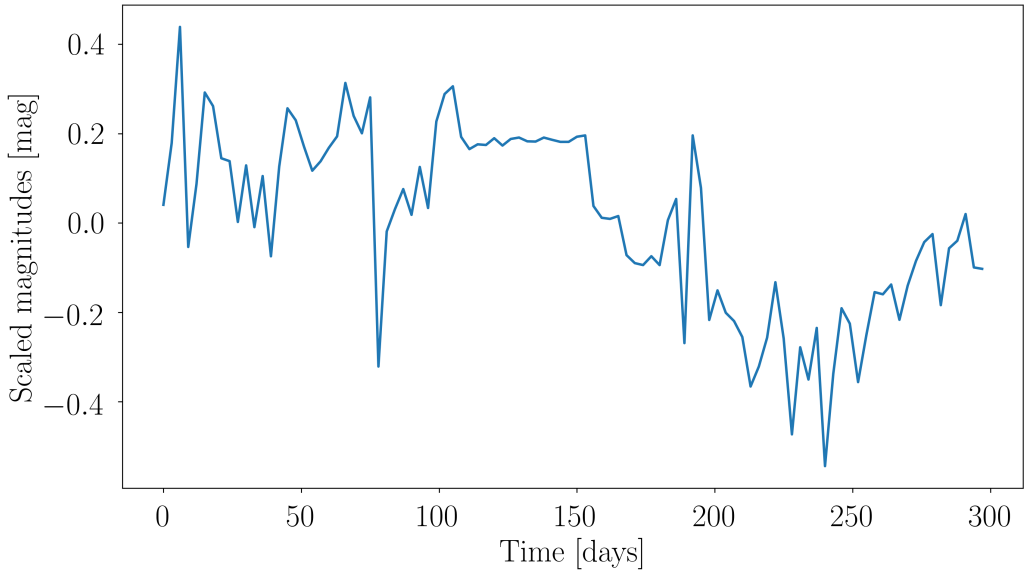


Figure 3.3: Example light curve after the transformations and data cleaning. Notice that it is still very noisy and "fixed" gap between days 120 and 150 is quite noticeable.

3.2.1 Convolutional filters

Fully connected neural networks do not perform well in learning patterns of the spatial data because connecting all neurons between each other cause losing information about the order and time distance of the measurements. Therefore neural networks can be easily overfitted by learning relations which make no sense from the physical point of view.

CNN preserves spatial distribution of the data by using the convolutional filters. In the following descriptions, we assume one-dimensional input if it is not said otherwise since the domain of this work is light curves.

Output values $\vec{y}$ are result of application of a convolutional filter $\vec{w}$ to the input $\vec{x}$. Consider input $\vec{x} = (x_1, x_2, x_3, x_4)$, convolutional filter $\vec{w} = (w_0, w_1, w_2)$ and output $\vec{y} = (y_0, y_1, y_2, y_3, y_4, y_5)$, then the convolution would looks like as follows:

$$\begin{bmatrix} w_0 & 0 & 0 & 0 \\ w_1 & w_0 & 0 & 0 \\ w_2 & w_1 & w_0 & 0 \\ 0 & w_2 & w_1 & w_0 \\ 0 & 0 & w_2 & w_1 \\ 0 & 0 & 0 & w_2 \end{bmatrix} \times \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix} \tag{3.2}$$

Matrix interpretation above is illustrated in the figure 3.4, where we can notice grey squares of zeroes at the beginning and at the end of the input vector which is not part of the input vector, but they are imaginary borders of zeros called *padding*. It allows starting sliding by the filter from a starting point beyond the input vector boundaries in order to control dimension of the output. For example when many subsequent filters are applied it is

convenient to keep input and output dimension the same. Another parameter is *stride* which controls steps length of sliding of the filter through the input.

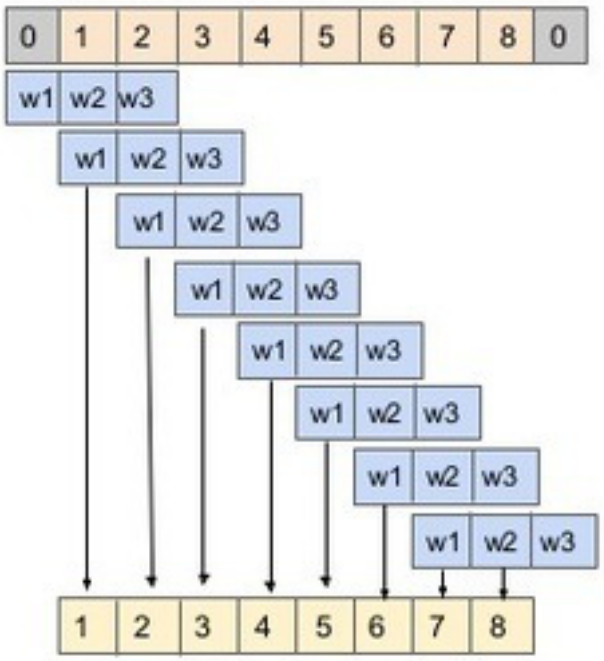


Figure 3.4: Example of applying convolutional filter $\vec{w} = (w_1, w_2, w_3) = (0,1,0)$ with zero padding 1 and stride 1. As can be seen the first value of the output (vector at the bottom) is calculated as the vector multiplication $(0,1,2) \cdot (w_1, w_2, w_3) = (0,1,2) \cdot (0,1,0) = 1$, the second values as $(1,2,3) \cdot (w_1, w_2, w_3) = (1,2,3) \cdot (0,1,0) = 2$ etc.

An example more related to time series can be found in the figure 3.5. Dimension x (could be for example time) is omitted and the remaining vector of outputs of the function $\frac{x}{10} sin(x)$ is used as the input for the convolution filters. Output vectors of the convolution represent how much the filters were activated for particular parts of the input.

Weights of the filters are tuned automatically by back propagation during training of the CNN. Therefore the model determines which patterns are important by its own. Usually many filters are applied and then the dimension of the input is increased to $n_{input} \times n_w$, where n_{input} is size of the input vector and n_w is number of filters.

3.3 Generative Adversarial Network

Classification algorithms are designed to distinguish between objects within the classes. Unfortunately, they cannot be directly used to generate new inputs which would look like objects of the desired class. However, this may be very interesting information because it can bring us a deeper understanding about the nature and typical features of objects we are interested in.

A naive approach would be to feed classifier by random light curves and look for those with the highest probability of being a quasar. Consider a sequential input of length 100

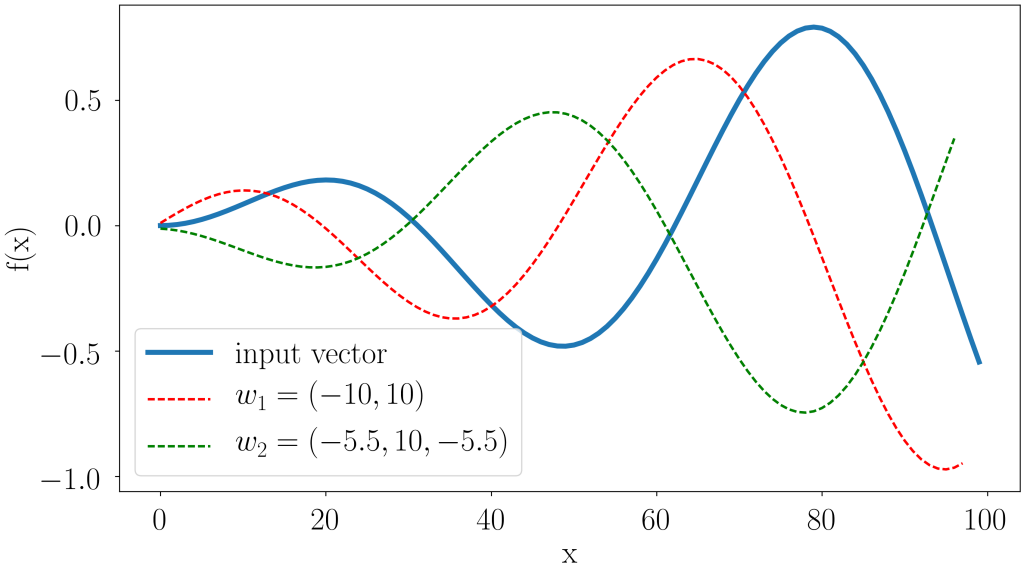


Figure 3.5: Input vector (generated from $\frac{x}{10}sin(x)$; $x \in \langle 0, 10)$) and output vectors after applying the filters. First filter $w_1 = (-10, 10)$ is detecting transitions from negative values to positives - see a maximum at $x = 60$. Second filter $w_2 = (-5.5, 10, -5.5)$ is little bit more complicated (but still very simple) is detecting inverse peaks (local minimum - see $x = 50$)

bins and scaled values discretely distributed into 30 different values (similar approach as in the section 1.2.2). The number of possible combinations for this approximate task is 30^{100} which is approx 5×10^{147}. Evaluating all of them would take a huge amount of time and computational resources. However, there are methods which do not need explore whole input space but can leverage from gradients to reach the local minimum of the input space.

Quite recent discovery are Generative Adversarial Networks (GANs) [37]. They are composed of two models - discriminator and generator. The generator is trying to create fake data which looks exactly same as the real positive data. Role of the discriminator is to classify given objects and detect whether they are fake or not. Both models are trained together by gradient descent. GANs were originally designed for two-dimensional image data and to this date, nobody used it for a task such as light curves generation.

The generator makes fake light curves from random noise which are evaluated by the discriminator and the prediction is used to calculate the error (gradient) of the generator. The gradient is then backpropagated through the network in order to update weights of the generator. This way the generator keeps making more realistic light curves of quasars.

The discriminator is trained by labelled light curves and fake light curves created by the generator which are labelled as the negative cases. It allows the discriminator to learn from previous mistakes caused by the still improving generator.

3.3.1 Discriminator

The discriminator **D** is a classifier which predict probabilities $\vec{y}$ whether the inspected light curve $\vec{x}$ is a quasar.

$$\vec{y} = \mathbf{D}(\vec{x}) \tag{3.3}$$

Before training the GAN, the discriminator is pretrained on the real training data to get some accuracy before facing the fake light curves.

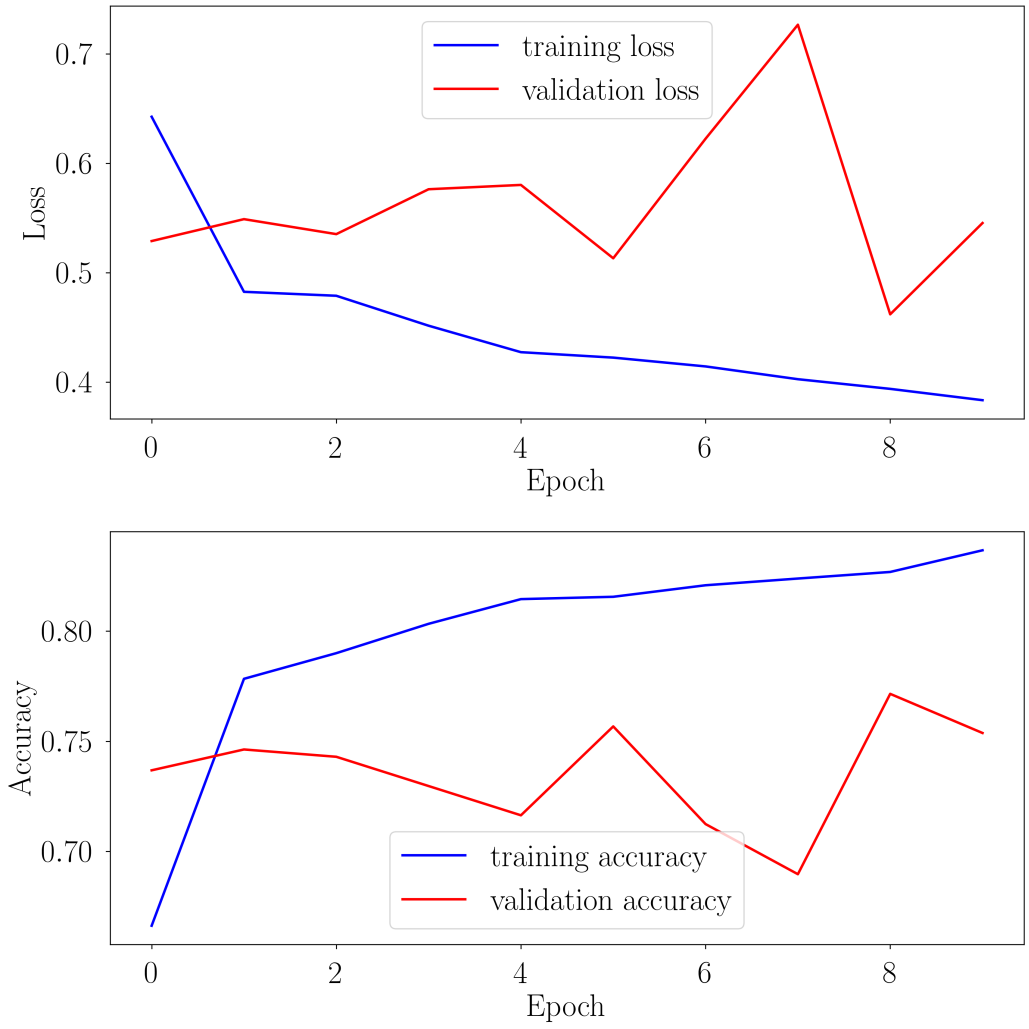


Figure 3.6: Discriminant pretraining

3.3.2 Generator

Generator **G** takes random noise $\vec{z}$ as the input and the output is a sequence $\vec{x}_z$ with the same dimension as the light curves. The dimension of the $\vec{z}$ was set to 32 which means

that the model increases dimension from the input 32 to the output 100.

$$\vec{x_z} = \mathbf{G}(\vec{z}) \tag{3.4}$$

3.3.3 GAN

Since the generator output has the same dimension as the discriminator input, these two layers can be connected together. It means that input of the GAN **A** is a noise of the length 32 and the output is probability whether the generator created a light curve which looks like a quasar for the discriminator.

$$\vec{y} = \mathbf{A}(\vec{x_z}) = \mathbf{D}(\mathbf{G}(\vec{z})) \tag{3.5}$$

There are two steps in the particular learning epochs. In the first one discriminator is trained on both real and fake light curves (labelled as negatives). Then GAN is trained with fixed weights of the discriminator part of the model. Input is a sample of random noises designated as positives. Calculated errors are used for gradient descent which updates the weights of the model.

The GANs have a problem with convergence, because of two loss functions competing between each other. Convergence of one prevents second to converge. Proposed loss function in the original article [37] was binary cross entropy. However, as it can be seen in the figure 3.7 the learning of both GAN and Discriminator is not stable. Also, it is hard to tune learning rates for both models, because it can easily happen that one model is converging, but the second diverges.

Solution for these problems is to use more suitable loss function for such model. It was shown that Wasserstein Loss [38] can be very effective for this kind of task. Figure 3.8 shows that both models are converging with this loss function.

3.3.4 Result

Generator

The accuracy of the discriminator is related to the generator accuracy. With increasing quality of the generated light curves also the quality of the training set for the discriminator is improving. Note that the discriminator is in every epoch trained by different light curves created by the improving generator. At the beginning generated light curves look like random noise – similar to non-variable stars. However, with improving generator, they start to look more like variable stars and after many more epochs, they look almost like the quasars.

Example light curves created by the generator can be found in the figures 3.9 and 3.10. They are both quite noisy which is no surprise when one considers that the training light curves are noisy as well.

Discriminator

Despite the quality of the data, the performance of the discriminator is relatively good. Distribution of predictions for both positive and negative samples can be seen in the figure

3.12 and evaluation table in 3.1. As was expected several percents of the light curves parts are indistinguishable because no all parts of the light curves of quasars show any detectable variability. Moreover, light curves of Be stars and quasars are very similar which is the second biggest source of the misclassification.

Visualizing convolutional filters of convolutional neural networks is one way how to get an idea what the neural network is doing. Despite all the efforts input light curves are still noisy (see Fig. 3.3) and it cannot be expected that convolutional filters will not be affected by this.

Figure 3.11 shows a sample of convolutional filters created by the model for quasars classification. It can be seen that they are detecting patterns such as increasing, decreasing, a transition from increasing to decreasing, etc. in the light curves.

	precision	recall	f1-score	support
quasars	0.92	0.77	0.84	1002
others	0.85	0.95	0.90	1357
avg / total	0.88	0.87	0.87	2359

Table 3.1: Evaluation of the discriminant model on the test sample for particular classes. *Others* is designation for Be stars, double period variables and non variables.

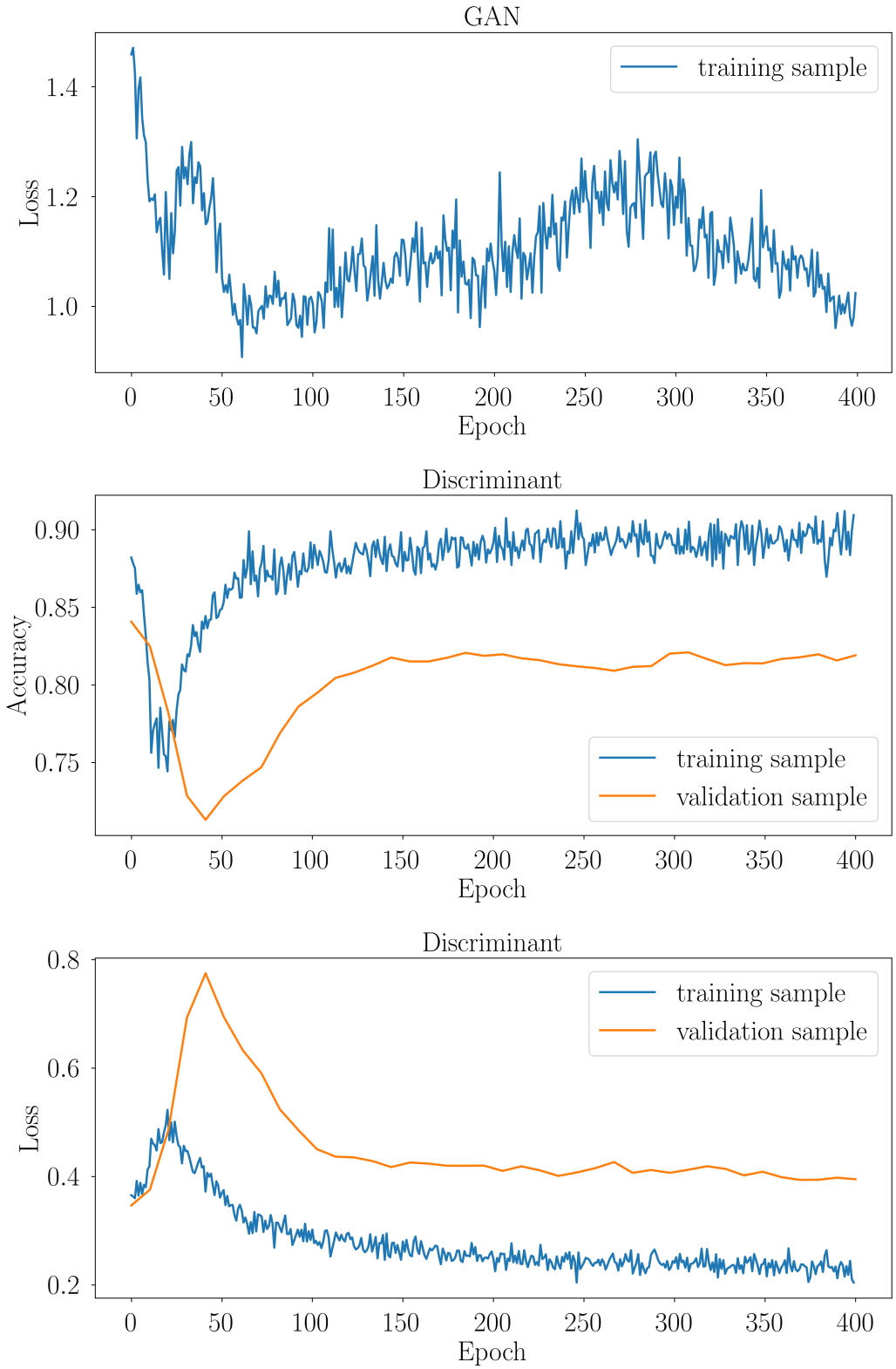


Figure 3.7: Learning curves of the discriminator and the GAN with used binary crossentropy as the loss function

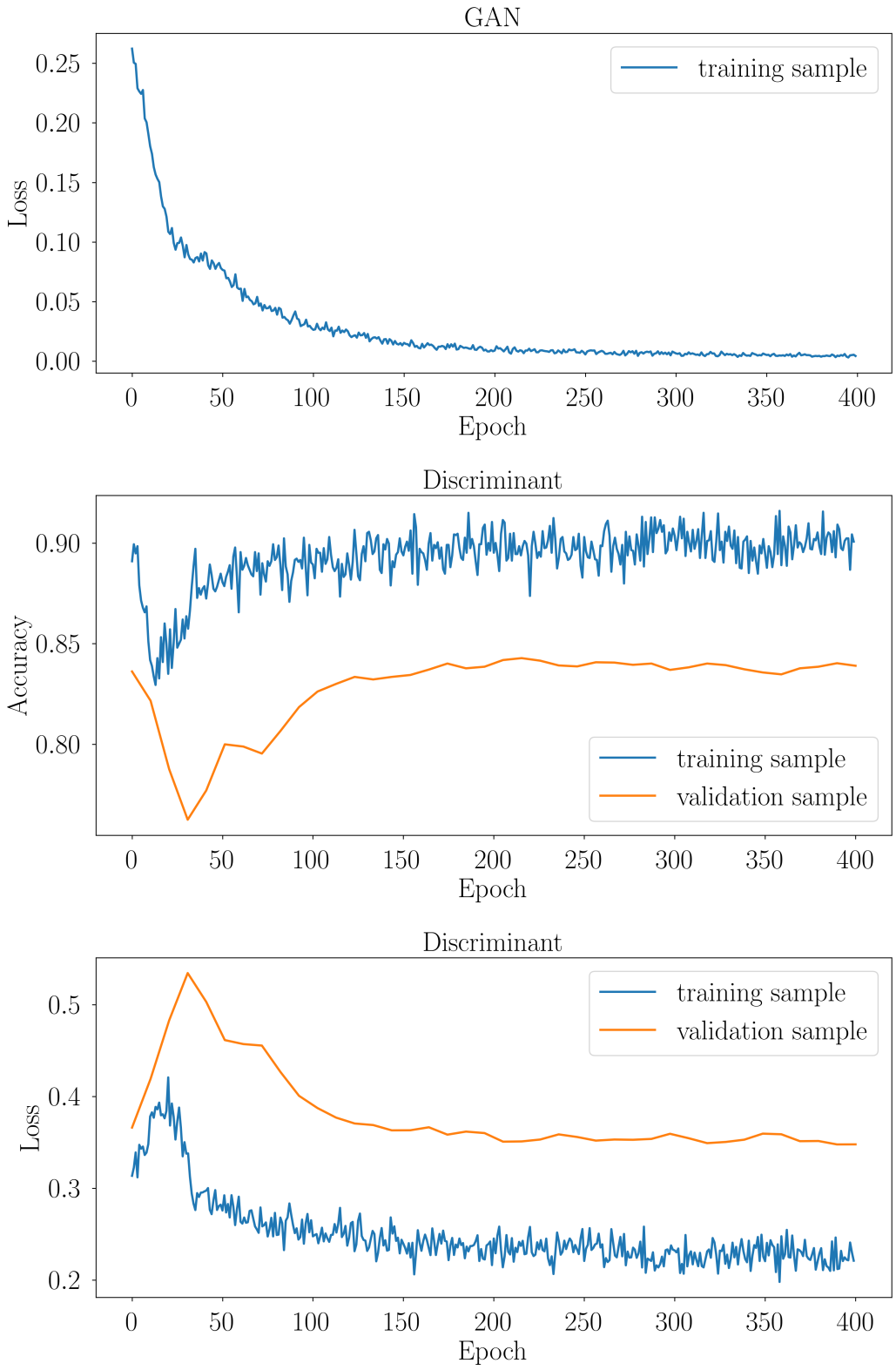


Figure 3.8: Learning curves of the discriminator and the GAN with used Wasserstein Loss

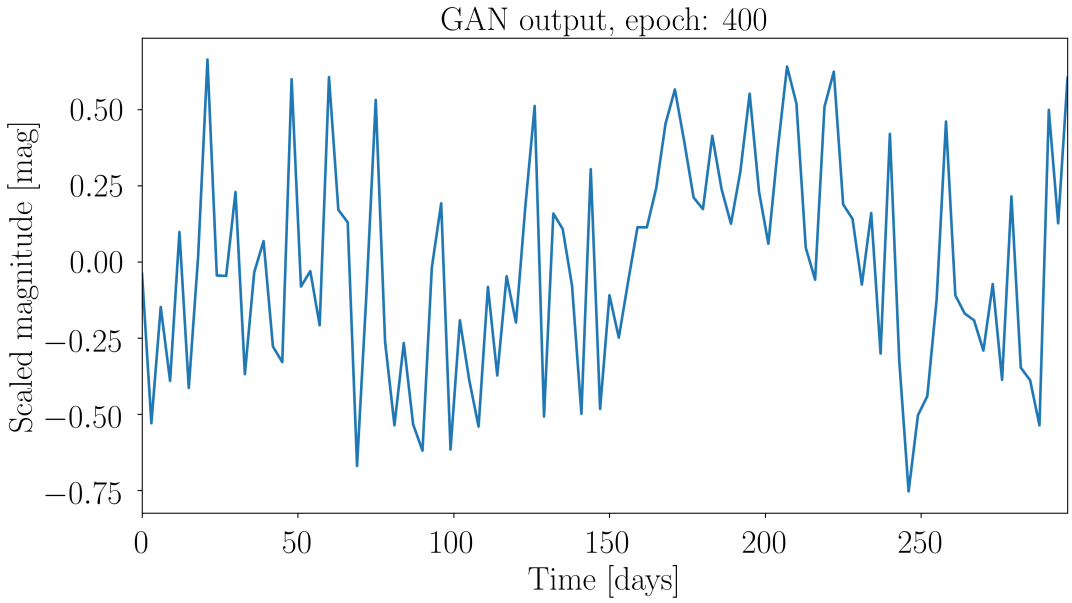


Figure 3.9: Fake light curve of a quasar generated by the generator after 400 epochs

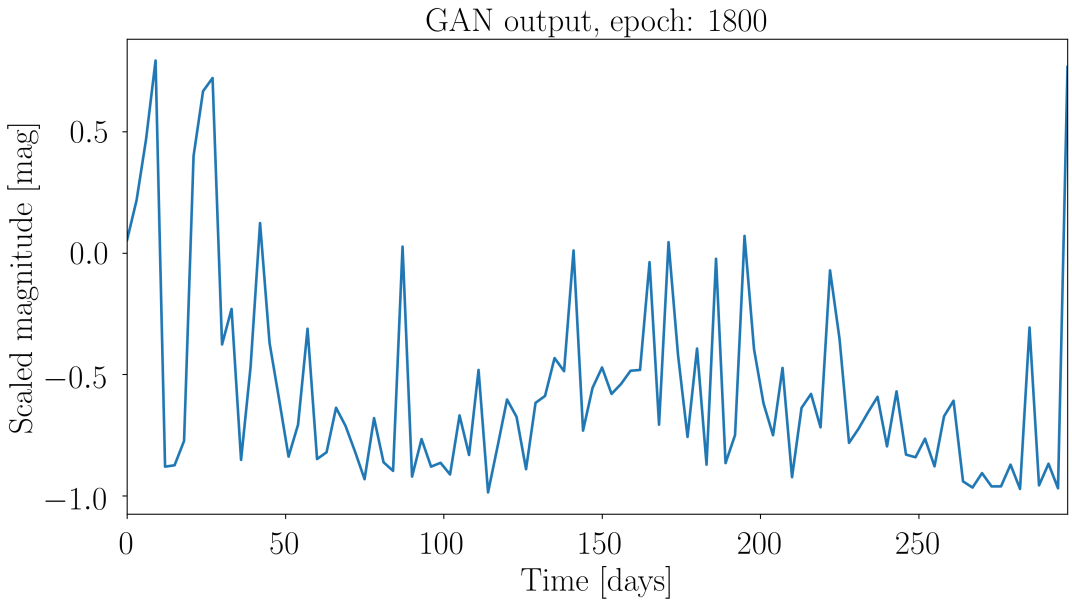


Figure 3.10: Fake light curve of a quasar generated by the generator after 1800 epochs

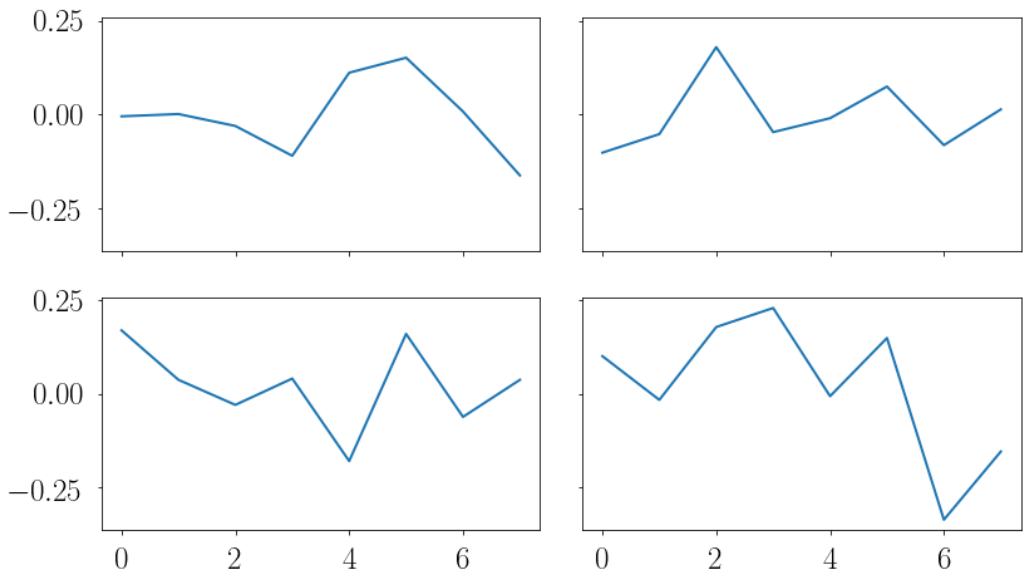


Figure 3.11: Selected convolutional filters created by the discriminator

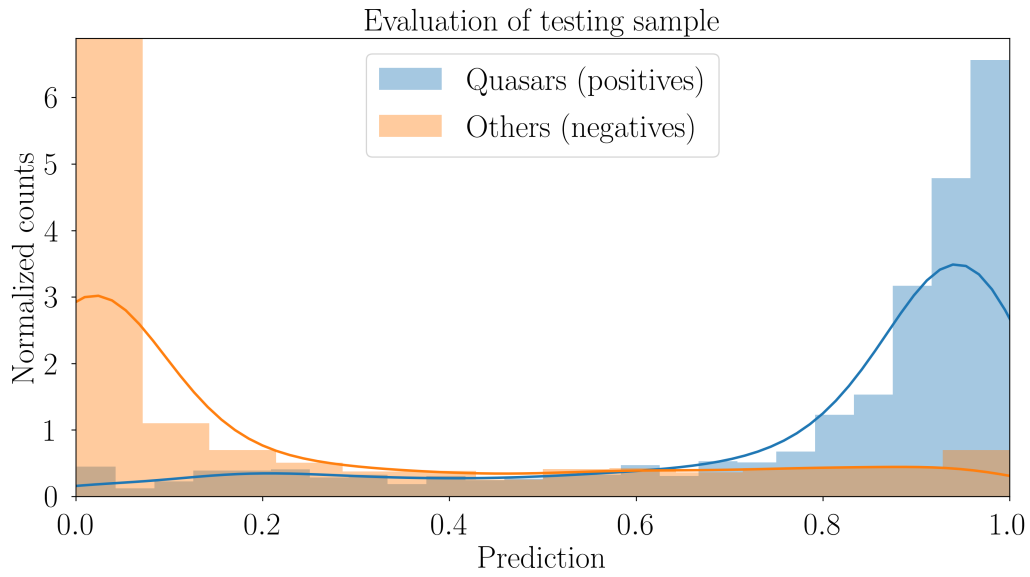


Figure 3.12: Distribution of predictions for the test sample. Quasars were labeled as 1 and others (Be stars, double period variables and non variables) were labeled as 0. Classification threshold could be for example set to 0.7 which would filter out most of negative samples and keep most of the quasars.

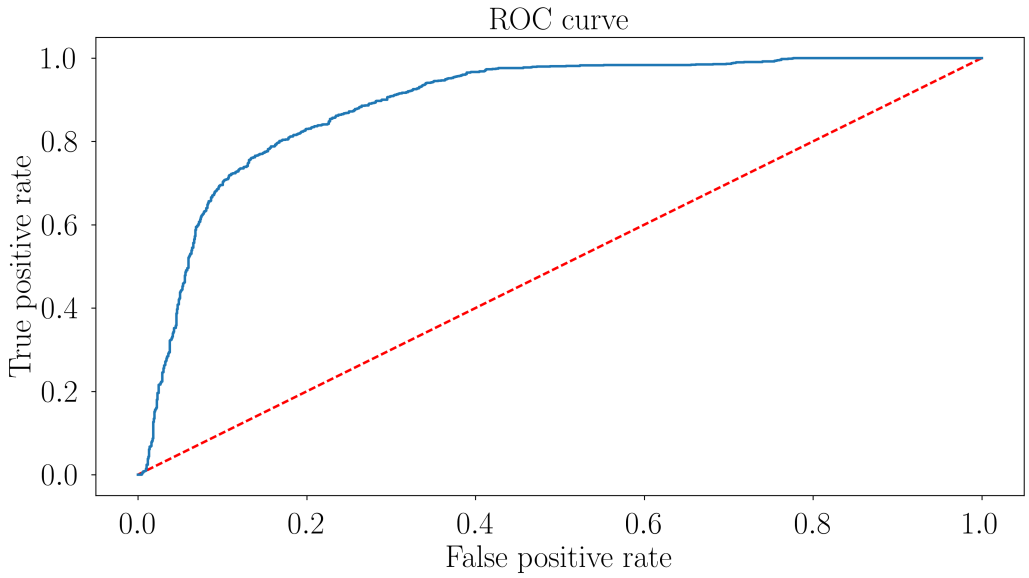


Figure 3.13: Receiver operating characteristic of the discriminant. Blue curve shows dependency between true positive rate and false positive rate for various thresholds. Red dashed line represents baseline for the model – in this case it would be random guessing.

Conclusion

In the era of Big Data, we need efficient and extendable tools to manage data in a reasonable time without wasting any time on the part of "tedious work". The *Light Curves Classifier* can make tasks related to data mining and machine learning easier and much more effective. An article about the package was submitted to Astronomy & Computing and after many suggested changes by the reviewers, it is now in the state of the final revision.

The package provides definable classification pipelines using modern libraries very quickly. The tool can be used programmatically as Python package, command line application or web application which is now hosted at [14] or can be run locally from Docker image as was shown in 1.1.

Moreover, it contains many feature extractors as well as numerous classifiers. All the methods can be combined and extended by users into very complex classification pipelines. The further contribution is the connectors to the largest databases of light curves which can be queried with the common interface.

In the future, the package will be extended with other methods, connectors, visualization tools and other features. However, it is not possible to develop such a huge package by one person anymore. Therefore it was released on Github as open source package with an aim to encourage the community to work on this project together.

Pipeline for classification of quasars was developed and evaluated on both test sample and the result of Eyer [30]. Filtering threshold of the model was set strictly in order to maximize ratio of correctly classified negative samples. Based on the evaluation of test data it is expected that from a sample of 1000 quasars and 1000 non-quasars (Be stars, variable stars with double period and non-variable stars) the passed sample consists of 430 quasars and 7 non-quasars.

However there are some misclassified objects according to Eyer, but since they are just candidates and very few of them were confirmed spectroscopically, it can be used to prove that the developed methods are better.

The second part of this work was dedicated to Convolutional Neural Networks and Generative Adversarial Networks. Generally, neural networks can be very powerful for more complex tasks such as light curves classification. However, there are many challenges which have to be handled during the process. Since the quality of astronomical data is usually quite low, the most focus needs to be put on the data preprocessing. Moreover, representation of the objects is very important. In terms of the light curves, one can extract some features (see 2) or use the whole light curve as the input vector. Both approaches were applied and discussed in this work.

Data processing is the crucial step in the machine learning pipelines and without considering all effects of each step it can have serious consequences for the final model

such as biasing of the data, which is often very hard to detect. It was shown that with thorough data engineering one can build accurate predictive model even with low-quality data. The essential component of convolutional neural networks is convolutional filters which are capable of detecting patterns, trends and other similarities. Therefore they can be effective in the classification of light curves.

Finally, Generative Adversarial Network was proposed as a new method in the field of light curves for training both model for generating and model for classification of light curves. It was shown that by the competition of these two models between each other, both models can be improved far beyond the capabilities of the standard approach of training them alone. Table 3.1 shows that very good results can be achieved by using this method, despite the quality of the data and the fact that no all considered parts of the quasars light curves show any variability. Moreover, GAN can be used for exploring the nature of the quasars. This can be done by both studying generated light curves and visualization of convolutional filters of the discriminator.

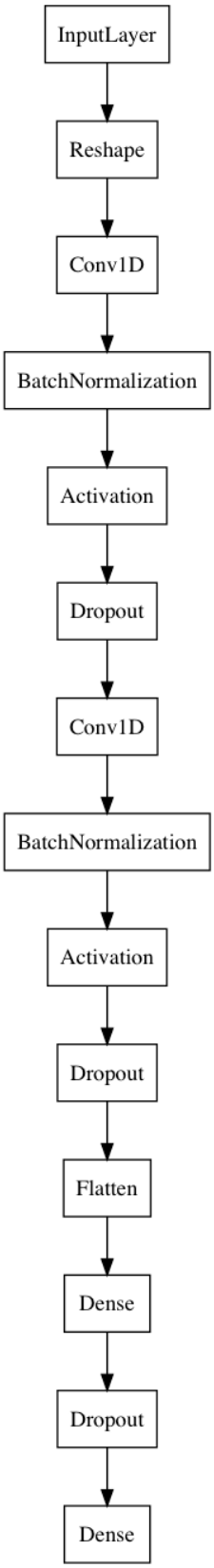
Appendix

```
# Tunning parameters
param_names = ["CurvesShapeDescr:alphabet_size",
               "CurvesShapeDescr:days_per_bin"]
alph_from, alph_to, alph_step = 10, 150, 5
dpb_from, dpb_to, dpb_step = 10, 100, 10

# Descriptor and decider
descr_name = "CurvesShapeDescr"
decid_name = "QDADec"

# Loading training stars
LCS_PATH = <light curve folder path>
obt_method = "FileManager"
be_stars_path = os.path.join(LCS_PATH, "be_stars")
stars_path = os.path.join(LCS_PATH, "some_stars")

# Get all permutations of the given parameter ranges
comb = get_combinations(param_names,
                        range(alph_from, alph_to, alph_step),
                        range(dpb_from, dpb_to, dpb_step))
tun_params = parse_tun_query(comb)

# Obtain the descriptor and the decider
pr = PackageReader()
descriptor = pr.getClassesDict("descriptors").get(descr_name)
decider = pr.getClassesDict("deciders").get(decid_name)
```

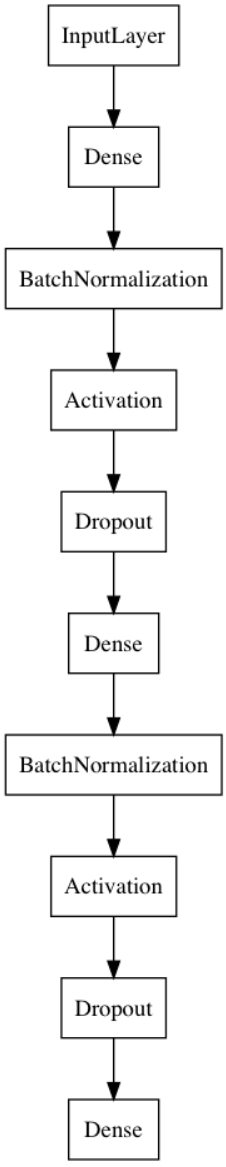
Listing 3.1

```
# Get search stars, some other stars (for training/testing)
# and template stars for comparing
be_provider = StarsProvider.getProvider(obt_method,
                                        {"path": be_stars_path})
stars_provider = StarsProvider.getProvider(obt_method,
                                           {"path" : stars_path})
templ_provider = StarsProvider.getProvider(obt_method,
                                           {"path" : stars_path})

be_stars = be_provider.getStars()
stars = stars_provider.getStars()
templ_stars = templ_provider.getStars()

# Load template stars as static parameter - this will not be tuned
static_params={"CurvesShapeDescr" : {"comp_stars" : templ_stars}}

# Estimate all combinations and get the best one
pes = ParamsEstimator(searched=be_stars,
                      others=stars,
                      static_params=static_params,
                      descriptors=[descriptor],
                      deciders=[decider],
                      tuned_params=tun_params)
pes.fit()
```

Listing 3.2



(a) Discriminator



(b) Generator

Figure 3.14: Architecture of the GAN model. GAN is composed of generator and the discriminator. Note that last layer of the generator is the first layer of the discriminator. In order to accelerate and stabilize learning batch normalization [39] was used.

References

[1] Gaia Collaboration, T. Prusti, J. H. J. de Bruijne, A. G. A. Brown, A. Vallenari, C. Babusiaux, C. A. L. Bailer-Jones, U. Bastian, M. Biermann, D. W. Evans, and et al., "The Gaia mission," *Astronomy and Astrophysics*, vol. 595, p. A1, Nov. 2016.

[2] J. Debosscher, L. M. Sarro, C. Aerts, J. Cuypers, B. Vandenbussche, R. Garrido, and E. Solano, "Automated supervised classification of variable stars. I. Methodology," *Astronomy and Astrophysics*, vol. 475, pp. 1159–1183, Dec. 2007.

[3] L. M. Sarro, J. Debosscher, M. López, and C. Aerts, "Automated supervised classification of variable stars. II. Application to the OGLE database," *Astronomy and Astrophysics*, vol. 494, pp. 739–768, Feb. 2009.

[4] K. Zebrun, I. Soszynski, P. R. Wozniak, A. Udalski, M. Kubiak, M. Szymanski, G. Pietrzynski, O. Szewczyk, and L. Wyrzykowski, "The Optical Gravitational Lensing Experiment. Difference Image Analysis of LMC and SMC Data. The Catalog," *Acta Astronomica*, vol. 51, pp. 317–329, Dec. 2001.

[5] J. Debosscher, L. M. Sarro, M. López, M. Deleuil, C. Aerts, M. Auvergne, A. Baglin, F. Baudin, M. Chadid, S. Charpinet, J. Cuypers, J. De Ridder, R. Garrido, A. M. Hubert, E. Janot-Pacheco, L. Jorda, A. Kaiser, T. Kallinger, Z. Kollath, C. Maceroni, P. Mathias, E. Michel, C. Moutou, C. Neiner, M. Ollivier, R. Samadi, E. Solano, C. Surace, B. Vandenbussche, and W. W. Weiss, "Automated supervised classification of variable stars in the CoRoT programme. Method and application to the first four exoplanet fields," *Astronomy and Astrophysics*, vol. 506, pp. 519–534, Oct. 2009.

[6] A. J. Drake, S. G. Djorgovski, A. Mahabal, J. L. Prieto, E. Beshore, M. J. Graham, M. Catalan, S. Larson, E. Christensen, C. Donalek, and R. Williams, "The Catalina Real-time Transient Survey," in *New Horizons in Time Domain Astronomy* (E. Griffin, R. Hanisch, and R. Seaman, eds.), vol. 285 of *IAU Symposium*, pp. 306–308, Apr. 2012.

[7] A. Mahabal, S. G. Djorgovski, M. Turmon, J. Jewell, R. R. Williams, A. J. Drake, M. G. Graham, C. Donalek, E. Glikman, and Palomar-QUEST Team, "Automated probabilistic classification of transients and variables," *Astronomische Nachrichten*, vol. 329, pp. 288–291, Mar. 2008.

[8] A. J. Drake, M. J. Graham, S. G. Djorgovski, M. Catelan, A. A. Mahabal, G. Torrealba, D. García-Álvarez, C. Donalek, J. L. Prieto, R. Williams, S. Larson, E. Christen sen,

V. Belokurov, S. E. Koposov, E. Beshore, A. Boattini, A. Gibbs, R. Hill, R. Kowalski, J. Johnson, and F. Shelly, "The Catalina Surveys Periodic Variable Star Catalog," *The Astrophysical Journal Supplement Series*, vol. 213, p. 9, July 2014.

[9] M. J. Graham, S. G. Djorgovski, D. Stern, A. J. Drake, A. A. Mahabal, C. Donalek, E. Glikman, S. Larson, and E. Christensen, "A systematic search for close supermassive black hole binaries in the Catalina Real-time Transient Survey," *Monthly Notices of the Royal Astronomical Society,*, vol. 453, pp. 1562–1576, Oct. 2015.

[10] M. J. Graham, S. G. Djorgovski, D. J. Stern, A. Drake, and A. Mahabal, "Detection of quasars in the time domain," in *IAU Symposium*, vol. 325 of *IAU Symposium*, pp. 231–241, June 2017.

[11] K. Pichara and P. Protopapas, "Automatic Classification of Variable Stars in Catalogs with Missing Data," *The Astrophysical Journal*, vol. 777, p. 83, Nov. 2013.

[12] P. B. Ryden, "Quasars." http://www.astronomy.ohio-state.edu/~ryden/ast162_8/notes36.html, 2003. [Online; accessed 8-May-2018].

[13] Wikipedia, "Alexander Friedmann." https://en.wikipedia.org/wiki/Alexander_Friedmann. [Online; accessed 8-May-2018].

[14] M. Vo, "The light curves classifier - web interface." http://vocloud-dev.asu.cas.cz/lcc, 2018.

[15] M. Vo, "The light curves classifier." http://ascl.net/1708.017, 2018.

[16] M. Vo, "Dockerhub: The light curves classifier - web interface." https://hub.docker.com/r/mavrix93/lcc_web/, 2018.

[17] I. Soszyński, "Results of the OGLE-II and OGLE-III surveys .," *Memorie della Società Astronomica Italiana, v.77, p.265 (2006)*, vol. 77, p. 265, 2006.

[18] A. Udalski, M. K. Szymanski, I. Soszynski, and R. Poleski, "The Optical Gravitational Lensing Experiment. Final Reductions of the OGLE-III Data," *Acta Astronomica*, vol. 58, pp. 69–87, June 2008.

[19] T. M. Brown, D. W. Latham, M. E. Everett, and G. A. Esquerdo, "Kepler Input Catalog: Photometric Calibration and Stellar Classification," *The Astronomical Journal*, vol. 142, p. 112, Oct. 2011.

[20] A. J. Drake, S. G. Djorgovski, A. Mahabal, E. Beshore, S. Larson, M. J. Graham, R. Williams, E. Christensen, M. Catelan, A. Boattini, A. Gibbs, R. Hill, and R. Kowalski, "First Results from the Catalina Real-Time Transient Survey," *The Astrophysical Journal*, vol. 696, pp. 870–884, May 2009.

[21] K. Cook, C. Alcock, R. Allsman, T. Axelrod, D. Bennett, S. Perlmutter, S. Marshall, M. Pratt, C. Stubbs, K. Griest, W. Sutherland, K. Freeman, P. Quinn, B. Peterson, and A. Rodgers, "The MACHO Collaboration Microlensing Survey: Results Toward the LMC and the Bulge," vol. 26, p. 912, May 1994.

[22] G. Pojmanski, "The All Sky Automated Survey. Catalog of about 3800 Variable Stars," *Acta Astronomica, v.50, pp.177-190, (2000).*, vol. 50, pp. 177–190, June 2000.

[23] A. Baglin and G. Vauclair, "The space stellar photometry mission corot: Asteroseismology and search for extrasolar planets," *Journal of Astrophysics and Astronomy*, vol. 21, no. 3, pp. 319–322, 2000.

[24] "Corot light curves," 2009.

[25] Ochsenbein, F., Bauer, P., and Marcout, J., "The vizier database of astronomical catalogues," *Astron. Astrophys. Suppl. Ser.*, vol. 143, no. 1, pp. 23–32, 2000.

[26] D. Foreman-Mackey, "A python interface to the kepler data." https://github.com/dfm/kplr, 2015.

[27] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, DMKD '03, (New York, NY, USA), pp. 2–11, ACM, 2003.

[28] M. Kantarcioglu and O. Kardes, "Preserving data mining in the malicious model," *Int. J. Inf. Comput. Secur.*, vol. 2, pp. 353–375, Jan. 2008.

[29] J. von Neumann, R. H. Kent, H. R. Bellinson, and B. I. Hart, "The mean square successive difference," *Ann. Math. Statist.*, vol. 12, pp. 153–162, 06 1941.

[30] L. Eyer, "Search for QSO Candidates in OGLE-II Data," *Acta Astronomica*, vol. 52, pp. 241–262, Sept. 2002.

[31] F. Chollet *et al.*, "Keras." https://github.com/keras-team/keras, 2015.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[33] M. Geha, C. Alcock, R. A. Allsman, D. R. Alves, T. S. Axelrod, A. C. Becker, D. P. Bennett, K. H. Cook, A. J. Drake, K. C. Freeman, K. Griest, S. C. Keller, M. J. Lehner, S. L. Marshall, D. Minniti, C. A. Nelson, B. A. Peterson, P. Popowski, M. R. Pratt, P. J. Quinn, C. W. Stubbs, W. Sutherland, A. B. Tomaney, T. Vandehei, and D. L. Welch, "Variability-selected Quasars in MACHO Project Magellanic Cloud Fields," *The Astronomical Journal*, vol. 125, pp. 1–12, Jan. 2003.

[34] S. Kozlowski, C. A. Onken, C. S. Kochanek, A. Udalski, M. K. Szymański, M. Kubiak, G. Pietrzyński, I. Soszyński, L. Wyrzykowski, K. Ulaczyk, R. Poleski, P. Pietrukowicz, J. Skowron, OGLE Collaboration, M. Meixner, and A. Z. Bonanos, "The Magellanic Quasars Survey. III. Spectroscopic Confirmation of 758 Active Galactic Nuclei behind the Magellanic Clouds," *The Astrophysical Journal*, vol. 775, p. 92, Oct. 2013.

[35] E. W. Flesch, "The Half Million Quasars (HMQ) Catalogue," *Publications of the Astronomical Society of Australia*, vol. 32, Mar. 2015.

[36] K. T. Paul, A. Subramaniam, B. Mathew, R. E. Mennickent, and B. Sabogal, "Study of candidate Be stars in the Magellanic Clouds using near-infrared photometry and optical spectroscopy," *Monthly Notices of the Royal Astronomical Society*, vol. 421, pp. 3622–3640, Apr. 2012.

[37] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," *ArXiv e-prints*, June 2014.

[38] C. Frogner, C. Zhang, H. Mobahi, M. Araya-Polo, and T. Poggio, "Learning with a Wasserstein Loss," *ArXiv e-prints*, June 2015.

[39] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *ArXiv e-prints*, Feb. 2015.