Diploma Thesis

Höhere Technische Bundeslehranstalt Leonding
Abteilung für Informatik

Exohunter

Submitted by: **Bernhard Reinsprecht, 5BHIF**
Julian Hautzmayer, 5BHIF
Date: **April 4, 2018**
Supervisor: **Herbert Lackinger**

Declaration of Academic Honesty

Hereby, I declare that I have composed the presented paper independently on my own and without any other resources than the ones indicated. All thoughts taken directly or indirectly from external sources are properly denoted as such.

This paper has neither been previously submitted to another authority nor has it been published yet.

Leonding, April 4, 2018

Bernhard Reinsprecht, Julian Hautzmayer

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorgelegte Diplomarbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Gedanken, die aus fremden Quellen direkt oder indirekt übernommen wurden, sind als solche gekennzeichnet.

Die Arbeit wurde bisher in gleicher oder ähnlicher Weise keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Leonding, am 4. April 2018

Bernhard Reinsprecht, Julian Hautzmayer

Abstract

Due to the fast technological improvements in the last few years we are nowadays able to look deep into space with the help of new high precision tools and telescopes here on earth or in the orbit of our planet. This gives us the opportunity to collect a huge amount of data on a daily basis to improve our understanding of the universe. Even huge computer systems cannot cope with the enormous supply of data and the most advanced analyses sometimes overlook anomalies or patterns in the data.

Since decades the search for planets outside our solar system is a major part of astronomy. The enormous amount of data is an important resource for finding evidence pointing to the existence of these planets. Especially lightcurves, the time series data of the emitted light of a star, play an important role in the search for exoplanets. By looking for periodic changes in the light such patterns can indicate unknown planets. Not only does this field of science strongly depend on state organisations but also on private persons and hobbyists who want to participate in the search for exoplanets.

This project aims to create a network for these planethunters and provide a tool which uses advanced concepts like neural networks and time series analysis to specifically analyse the light curves with the user's computer processing power. For the implementation of this software new as well as state of the art technologies were used to create a smooth user experience and provide a solid scientific background.

Zusammenfassung

Durch die technischen Entwicklungen der letzten Jahre ist es uns heute möglich mit hoch präzisen Instrumenten und Teleskopen hier auf der Erde oder im Orbit unseres Planeten in die Tiefen des Alls zu sehen. Durch diesen Fortschritt sind wir in der Lage tagtäglich Unmengen an Daten zu sammeln um unser Verständnis über unser Universum zu verbessern. Doch die schiere Menge an Daten überfordert auch die größten Rechenanlagen und auch die besten Analysen übersehen immer wieder Anomalien und Mysterien in den Datensätzen. Gerade deshalb ist die Vielfalt der Analysemethoden und Missionen die sich mit der Analyse dieser Daten beschäftigen so wichtig.

Seit jeher ist die Suche nach Planeten außerhalb unseres Sonnensystems eine wichtige Bestandteil der Astronomie. Gerade hier sind diese Unmengen an Daten eine wichtige Resource um nach Indizien für die Existenz anderer Planeten zu suchen. Vor allem Lichtkurven(Lightcurves) spielen hier eine wichtige Rolle. Diese über einen bestimmten Zeitraum aufgezeichneten Daten repräsentieren das emittierte Licht von Sternen und geben uns die Möglichkeit durch periodische Änderungen im Licht unbekannte Planeten zu finden. Nicht nur sind in diesem Feld der Wissenschaft staatliche Organisationen von immenser Wichtigkeit sondern auch private Personen und Hobbyisten die sich bei der Suche nach Exoplaneten beteiligen.

Dieses Projekt hat es sich zum Ziel gemacht ein globales Netzwerk für diese Planetenjäger zu schaffen und ihnen ein Tool zur verfügung zu stellen welches neuartige Methoden wie Neuronale Netze und Time series Analysen verwendet um gezielt Lichtkurven mithilfe der Rechenleistung ihrer Computer zu analysieren. Bei der Realisierung dieser Software wurden sowohl neuartige wie auch 'stand of the Art' Technologien verwendet um eine angenehme Nutzererfahrung und eine soliden wissenschaftlichen Grundlage zu bilden.

Acknowledgments

We would like to express our special thanks of gratitude to our teacher and mentor Dipl.-Ing. Prof. Herbert Lackinger who helped and supported us during the whole process of making this project become reality.
Secondly we would also like to thank our families and friends for their help and support.

Contents

1 Theoretical Background **1**

1.1 Exoplanets . . . 2
1.1.1 What is an exoplanet . . . 2
1.1.2 Habitable zone . . . 2
1.1.3 Why humanity needs to find earth-like exoplanets . . . 2
1.1.4 History of finding exoplanets . . . 2
1.1.5 Exoplanet finding methods . . . 3
1.2 Kepler, K2 Mission and NASA . . . 17
1.2.1 What are the Kepler and K2 Mission . . . 17
1.2.2 Goal of the Kepler and K2 Mission . . . 18
1.2.3 Usage of K2 and Kepler lightcurves . . . 18
1.2.4 APIs and Databases . . . 19
1.3 Neural Networks . . . 21
1.3.1 The basic concept of Neural Networks . . . 21
1.3.2 Convolutional Neural Networks . . . 24
1.3.3 Human versus artificial intelligence . . . 25
1.3.4 Technologies for implementing Neural Networks . . . 26
1.3.5 Importance of data . . . 28
1.4 Time series analysis . . . 30
1.4.1 Time series . . . 30
1.4.2 Preparing time series . . . 31
1.4.3 Moving average . . . 34
1.5 Clustering . . . 35
1.5.1 Supervised vs. unsupervised learning . . . 35
1.5.2 What is clustering? . . . 35
1.5.3 Clustering algorithms . . . 36
1.5.4 Cost functions . . . 38
1.5.5 Dynamic time warping algorithm . . . 38
1.6 Electron desktop client . . . 40
1.6.1 Electron . . . 40
1.6.2 Canvas.js . . . 41
1.6.3 Calling processes from Electron . . . 41
1.7 JavaEE Server . . . 44

1.7.1 Wildfly . . . 45
1.7.2 MariaDB . . . 45
1.7.3 Docker . . . 46
1.7.4 JPA and Hibernate . . . 47
1.7.5 Java REST . . . 47
1.7.6 Jsoup . . . 48
1.7.7 Light curve fetching . . . 49
1.8 Website . . . 50
1.8.1 Chartist . . . 50

2 Practical implementation **51**
2.1 System architecture . . . 52
2.1.1 Desktop Client . . . 53
2.1.2 Server . . . 54
2.1.3 Website . . . 54
2.2 Diagrams . . . 55
2.2.1 Entity relationship diagram(ERD) . . . 55
2.2.2 LADataPoint . . . 56
2.2.3 Userstories . . . 59
2.3 Implementation of the light-curve classifier . . . 61
2.3.1 General . . . 61
2.3.2 Data . . . 62
2.3.3 Preparing data . . . 63
2.3.4 The Convolutional Neural Network . . . 66
2.3.5 Training the Neural Network . . . 68
2.3.6 Testing the Neural Network . . . 69
2.3.7 Using the Neural Network . . . 70
2.4 Implementation of the Electron client . . . 72
2.4.1 Manual mode . . . 72
2.4.2 Batch mode . . . 74
2.4.3 Fetching light curves and displaying . . . 77
2.4.4 Scanning for exoplanets . . . 78
2.4.5 Calling python processes . . . 83
2.4.6 User input feedback . . . 84
2.4.7 Logging . . . 87
2.5 Implementation of the Time Series Analysis . . . 88
2.5.1 General . . . 88
2.5.2 Preparing the light-curve . . . 89
2.5.3 Create light-curve inclination map . . . 91
2.5.4 Searching for dips . . . 92
2.5.5 Calculate dip parameters . . . 92
2.5.6 Hierarchical dip clustering . . . 92
2.5.7 Cluster dip period check . . . 94

2.5.8 Calculate candidate parameters 94
2.6 Implementation of the Server 99
2.6.1 General 99
2.6.2 Rest interfaces for the most important classes 99
2.6.3 K2 epic number retrieval 101
2.6.4 Add new evaluation and user feedback 103
2.6.5 Exoplanetary statistics 105
2.6.6 Improving the light-curve classifier 106
2.6.7 Evaluation probability calculation 106
2.7 Implementation of the Website 109
2.7.1 General 109
2.7.2 Exoplanet statistics 111
2.7.3 List of evaluations 114
2.7.4 Evaluation detail page 114

3 Attachments **119**
3.1 Protocols 119

Chapter 1

Theoretical Background

1.1 Exoplanets

1.1.1 What is an exoplanet

An exoplanet is a planetary celestial body orbiting a foreign star outside the gravitational influence of the sun. Scaling up from out solar system, which contains 8 planets, to the milky way, which has again about 100 billion stars we get a total of 100 to 200 billion planets. When taking the whole universe into account, which contains again about 100 billion galaxies, we get an estimated total of 10^{21} to 10^{22} exoplanets.

Of this enormous amounts of planets, humanity has confirmed 3709 discovered exoplanets. Exoplanets are not to be confused with rogue planets which do not orbit a host star unlike exoplanets.

1.1.2 Habitable zone

Chances for an earth-like planet increase when the celestial body orbits its host star in range of the habitable zone. This range varies with size and temperature of the host star. Within the habitable zone planets support water in a liquid state - too cold and it is frozen, too hot and it becomes gas. It is known that water is an essential component for life, so the chances of intelligent life also increase if the planet is located in range of the habitable zone.

1.1.3 Why humanity needs to find earth-like exoplanets

It has always been human nature to explore and expand. The things we can learn by looking at the stars and orbiting exoplanets are astounding. When analysing foreign planets we could find other species, broaden our knowledge about space and get to understand our universe better.

Another aspect is that earth will turn uninhabitable in the future. Whether it is due to climate change or mass extinction events, either way someday our sun will explode and wipe off every living creature on earth along with it. Until then humanity needs to find habitable planets that could prolong humankind's existence in the universe by offering a new place to settle. Colonizing a planet like Mars would act as backup, in case something were to happen.

1.1.4 History of finding exoplanets

In January 1992 Alex Wolszczan and Dale Frail made history when they confirmed the first ever exoplanets using the Arecibo Observatory located in Puerto Rico. They discovered the first two exoplanets located in the Virgo constellation. The distance between earth and the Virgo constellation is about 2,300 light years and the two newly found planets orbited around a pulsar. (A pulsar is a neutron star which forms out of the collapse of a star's core. Therefor the star must be more massive then our own sun.

Those pulsars send out two light beams and spin with an enormous velocity of up to once a second. Alien life on planets orbiting such monsters seems nearly impossible.)

Long before this breakthrough scientists theorized about the existence of planets beyond our solar system, moving alone through space or orbiting another star. But never before was an exoplanet confirmed until January 1992. This was the first step in a long journey to better understand our universe and ultimately find extraterrestrial life.

> "From the very start, the existence of such a system carried with it a prediction that planets around other stars must be common, and that they may exist in a wide variety of architectures, which would be impossible to anticipate on the basis of our knowledge of the solar system alone."
>
> (Alex Wolszczan)

In October 1995 Micheal Mayor and Didier Qualozm made another major discovery when they found the first ever exoplanet orbiting a sun-like star. Since then the search for exoplanets went on, supported by the fast technological improvements in the upcoming years and the growing public interest in astronomy. In March 2009 NASA launched the Kepler space telescope which carried out the Kepler and the currently ongoing K2 mission, both of which played an important part in the search for exoplanets.

Currently(15.February.2018) there are 3,605 confirmed exoplanets some of which could support life and are very similary to our own earth. One of the most interesting findings in the last few years were the in early 2017 discovered 7 earth-like exoplanets orbiting a star that is just 40 light-years away in the Aquarius constellation called Trappist-1. In terms of astronomical units this is only a stone's throw from our earth.

1.1.5 Exoplanet finding methods

There are several different techniques for finding exoplanets. Every one of them has advantages and disadvantages and can only be applied on a fixed domain of exoplanets which fulfill certain critera. For example the transit method one of the most common ways for finding exoplanets only looks at the light emitted by a star and the small irregularities created when an exoplanet passes in front of the star.

Thereby, exoplanets that orbit their host in such a way that they are never aligned with their star from the perspective of the observer can't be detected by this method. Fortunately, there are many different methods and humanity has the ability to apply them with the help of the most modern telescopes and observation systems.

1.1.5.1 Different Techniques

Currently (15.February.2018) there are 3,605 confirmed exoplanets. Compared with the roughly estimated number of terrestrial planets (terrestrial planets are planets that are

primarily composed of rocks and metals) in our galaxy the Milky Way of about 10 billion, this number seems very, very small. That means that humanity has currently only found about $3 \times 10^{-4}\%$ of that estimated 10 billion.

The most effective and popular exoplanet finding methods are:

1. **Radial Velocity**
   Every body with mass affects nearby objects with its gravitational forces. This force can be determined with Newton's gravitational law $F = G \times \frac{m_1 \times m_2}{r^2}$ with G the Gravitational constant($G = 6.674 \times 10^{-11} m^3 kg^{-1} s^{-2}$), m_1 and m_2 the two masses of the objects and r the distance between them. Based on this principal, even a planet which is significantly smaller and has a lot less mass than its host star applies a force on the star and changes the center of mass of the system so that the star itself rotates around a new center of mass. This circular movement leads to variations in the speed with which the star moves away from and towards earth. These changes in velocity can be determined and analysed to confirm the existence of a nearby planet.

2. **Transit Method**
   The Transit Method aims to detect exoplanets by looking for irregularities in a star's light. Such irregularities appear when a planet orbits, from the perspective of the observer, in front of the star and reduces the light that travels to earth. Therefore the emitted light of a certain star is monitored over a period of time by a telescope on earth or in earth's orbit. Afterwards, this data is analysed to find unusual or repeating patterns to identify planets orbiting a particular star. With the help of the Transit Method properties like the radius of the planet can be determined, in contrast to the Radial Velocity method where the mass of the planet can be determined(with a certain amount of error).

   Important for the Transit Method is the position of the observer(the telescope or system monitoring the star). If a planet is orbiting its host star in such a way that for the observer it will never pass in front of the star the planet is undetectable for the transit method. That is a major drawback for this method although this method is currently the most effective one and missions like the Kepler Mission or the ongoing K2 Mission have found hundreds of new exoplanets. A more in depth explanation of the Transit Method can be found in section 1.1.5.2 "The Transit Method".

3. **Direct Imaging**
   In contrast to the other exoplanet finding methods, Direct Imaging really means to "see" the exoplanet, if it exists. The method can be described with the well known saying "Seeing is believing". However, the success chance of this method is minor and the difficulty to find exoplanets with Direct Imaging is high, which can be directly seen by the number of confirmed exoplanets found with Direct

Imaging. The reason for that lies in the fact that planets are very small and they easily get lost in the glare of their host star. However, with the existing telescope technology there can be special situations in which the Direct Imaging method leads to success.

4. **Gravitational Microlensing**
   Gravitational Microlensing has a major difference from all of the other methods. It is capable of finding exoplanets at a truly great distance from our home planet. In comparison, the Transit Method and the Radial Velocity method can find exoplanets at a distance of hundreds of light years away from earth. Gravitational Microlensing on the other hand is capable of finding exoplanets near the center of the Milky Way. The theory behind Gravitational Microlensing focuses on Einsteins's General Theory of Relativity. According to Einstein, if a light beam from an observer from earth passes near a star it will be slightly bent by the gravitational forces of the star it is passing. If now an exoplanet would exist near the star the light beam would also be affected by the gravitational forces of the planet.

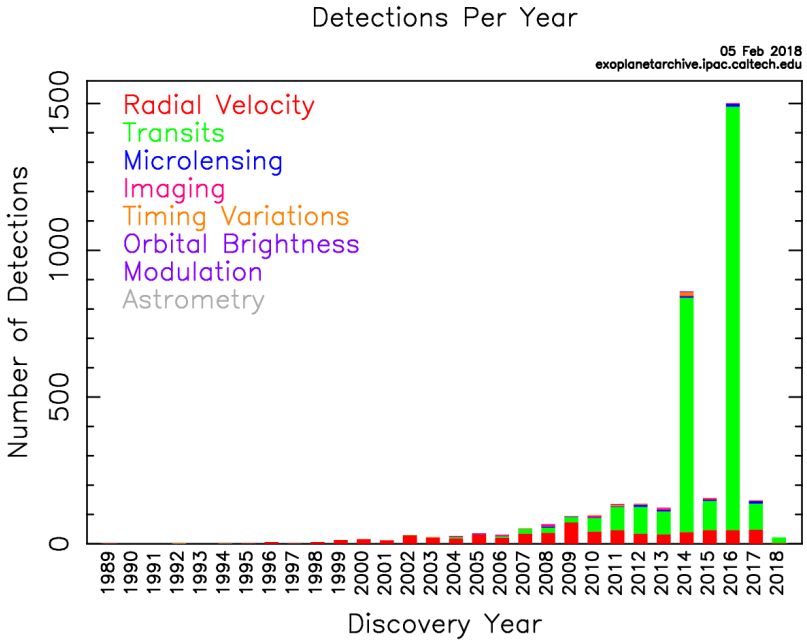

Figure 1.1: Distribution of confirmed exoplanets by year and finding method.

When looking at the number of confirmed exoplanets and their referenced finding methods these four methods can be easily ranked. The most effective one(if effectiveness is defined by the number of confirmed exoplanets) is the Transit Method with a total of 2804 exoplanets. The next one is the Radial Velocity method with about 665 confirmed exoplanets and a significant delta between the first and the second rank in the number of planets. On the third rank is the Gravitational Microlensing with 53 and on the fourth

place Direct Imaging with 44 exoplanets. The rest of the 38 exoplanets where found with other methods that will not be mentioned here. Therefore this project chose to employ the Transit Method to search for exoplanets.

1.1.5.2 The Transit Method

1.1.5.2.1 What is a star's light-curve The most important part of information used in the Transit Method is the so called light-curve. A light-curve is a discrete function of data points collected over a certain amount of time. Every point in the light-curve represents the magnitude of the star's light at a certain time. These graphs are not only used for exoplanet detection but also for detecting and researching novae, supernovae and variable stars. Light-curves are very simple but valuable tools for scientists in the fields of astronomy.

Figure 1.2 shows the light-curve of the star EPIC 211311380 (HIP 41378). This star is particularly interesting because it has five exoplanets which can be easily seen in the displayed light-curve. The y-axis of the plot describes the magnitude of the star's light with the unit flux and the x-axis describes a point of time; this time can be measured in days, hours or even minutes. In this case the x-axis or time-axis is measured in days.

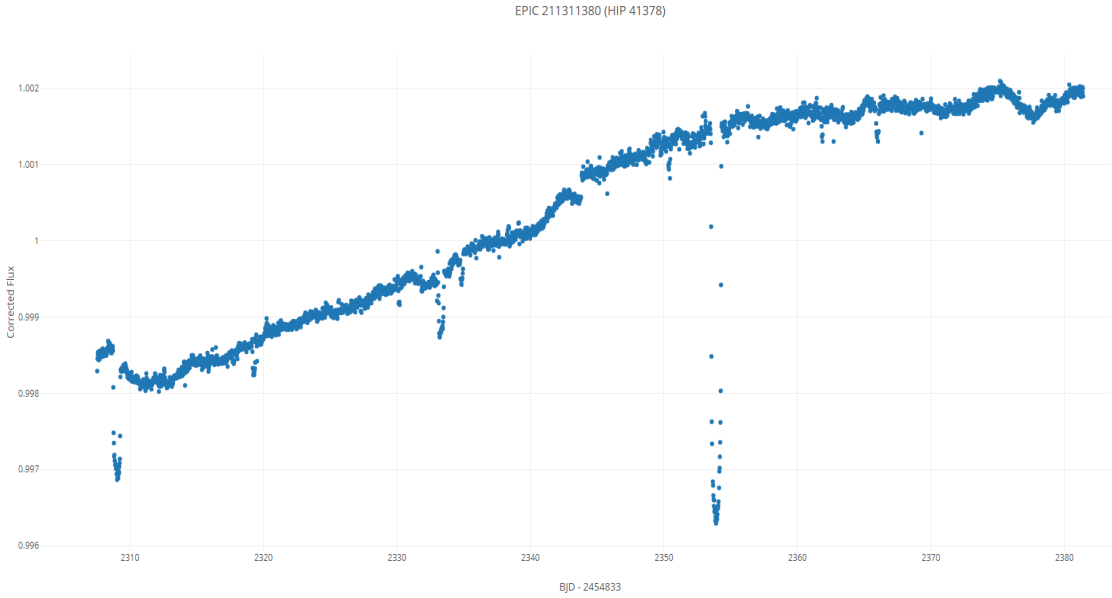

Figure 1.2: Light-curve of the star EPIC 211311380 (HIP 41378).

The big question is now, what can we learn from ordinary light-curves? First we can find periodic events like the orbit of a planet, we can determine the duration of an event by looking at the x-axis and the change of magnitude by calculating the delta value of the

light. Additionally we can find irregularities, unusual events or patterns that indicate other events or, in the example of the Transit Method, the crossing of a celestial body in front of the star. Important to know is that light-curves or their preamble, the time series, are extremely powerful and strong pieces of data which can be found everywhere in modern day computer science.

1.1.5.2.2 How does the Transit Method work The basic idea of the Transit Method consists of three major steps:

1. Monitoring the target
2. Preparing the collected data
3. Analysing the data

The **first step** can be either done with a telescope stationed on earth or with a telescope in outer space like the Kepler telescope. Monitoring the target from earth sounds a lot easier than building a rocket that shoots the telescope into space. However, the earth's atmosphere, clouds, smog and other natural or men made factors can obstruct the monitoring process. On the other hand, telescopes in space have no major atmosphere decreasing their view or other negative factors.

In case of the Kepler or K2 Mission the telescope is not pointed at a target in particular, like for example a certain star. The telescope is pointed at a particular cluster in space which it is observing for a certain amount of time. In the case of the Kepler Mission this monitoring process lasted for years, but in the newer K2 Mission the monitoring process only lasts for about 8 months per cluster. After one cluster is finished the next cluster will be monitored for another 8 months.

The duration of observation plays an important role in the amount and type of planets that can be found with the data collected by the Transit Method. Data that observes a target over a very short time interval means that the probability of finding exoplanets with a very fast orbital period(the time a planet needs to make a full orbit around its host star) is very high. However, planets that have a very long orbital period are much less likely to be found with the help of the data.

A very simple example of this situation would be if we try to observe our sun from many light-years away for 4 months to find our earth in the resulting light-curve. Earth has an orbital period of $365\frac{1}{4}$ days, compared to that the observation time of 4 months represents only $\frac{1}{3}$ of the orbital period of earth. The planet must be in the correct starting position at the beginning of the observation to intersect the view of the observer in front of the sun within the observation time, which is not always the case for planets with a higher orbital period like for example earth or, to name another more extreme example from our solar system, Neptune, which orbits our sun in 60,190.03 days or 164.8 years.

The **second step** is the preparation of the collected data. The raw data which is collected by a telescope consists only of numbers and can be full of errors and anomalies that have to be cleaned up before the final analysis of the data. Additionally the raw data must be converted into a light-curve data format which is saved in the so called .FITS format. In the case of the Kepler Space Telescope the collected data is processed and calibrated by the Kepler Science Pipeline. This Pipeline consists of many different modules and steps to clean the data, for example:

1. Correct systematic errors
2. Estimate and subtract background light
3. Detect outlying data points
4. Handle cosmic rays
5. Finally create light-curves with the corrected flux values

The **last step** and at the same time the most important step is to analyse the light-curve of the targeted star. To understand the analysis of the light curve it is important to understand how the light of a light-curve reacts if a planet crosses in front of its host star and which effects occur.

When a planet crosses in front of its host star it dims the light emitted by the star. This can be seen in the light-curve as a dip, a reduction followed by an increase in the flux value of the star's light. Every dip describes an object that is passing in front of the star. That also means that it is possible that non-planet like objects like dust, clouds or other celestial bodies can cross in front of the star and create a dip. Therefore it is important to ensure that the depth of the dip, the delta value before the entrance of the body and the bottom of the dip, is not too small and at the same time not too large. A too small dip can point to non-planet activity and a too large dip can for example unveil a binary star system(two stars rotating around each other and also blocking the light to create a periodic pattern in the light-curve). It is very important to note that the size of the exoplanet strongly affects the shape of the dip. This behavior can be seen in Fig 1.3. An increase in the radius of a planet also increases the depth of the dip.

These dips are the most important indicators for a planet in the Transit Method. A planet that makes multiple orbits will be represented in the light-curve with multiple very similar dips. A planet that only finishes one orbit or only intersects the view of the observer because of a high orbital period will be only seen as one dip. By comparing the dips with each other it is possible to find similar ones and to group them together. This can give information about the amount of planets in the current system and the orbital period of a planet.

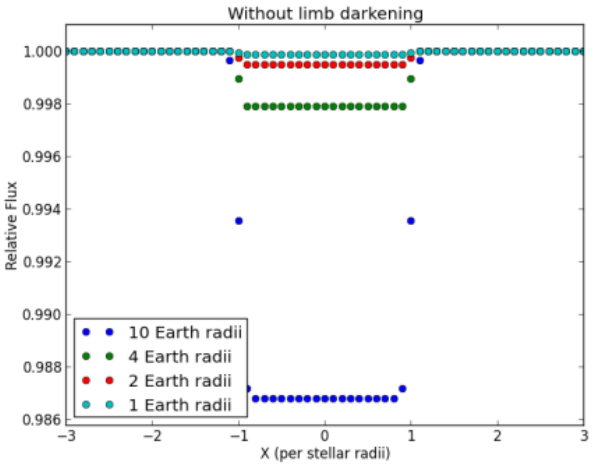

Figure 1.3: Comparision of different planet radii.

Additionally the dips help us to determine some of the properties of an exoplanet like the radius, the orbital period or the orbital velocity(the calculation of these values will be explained in the next section). Fig 1.4 shows a dip with all of its important parameters that can be used to determine the planet's properties. Most important is the difference of flux between the star's light before and during the transit(a transit is the event when a planet passes in front of its star), which can be used to determine the radius of the exoplanet in respect to the star's radius.

Furthermore there are four timestamps in the Fig 2.3 t_I, t_{II}, t_{III} and t_{IV}. t_I is the time when the edges of the planet and the star start to intersect, this event is called ingress. t_{II} is the time when the planet has completely passed the left outer edge of the star. t_{III} and t_{IV} describe the same event in reversed order, first the planet right edge and the stars left edge align before the planet leaves the position in front of the star where it dims the star's light; this event is called egress. The time from t_I to t_{IV} is called the total time which describes the time the planet needs to pass fully in front of its host star. The time from t_{II} to t_{III} is called full time which describes the time the planet needs after it finishes the ingress until it starts the egress.

One last very important parameter displayed in Fig 1.4 is b which is called the impact parameter. It is basically the horizontal offset from the planet in respect to the center of the star. The impact parameter has a direct impact on the duration of the transit. When the horizontal offset of the planet changes, the transit duration will also change. The domain of b starts from b=0 which means the center of the stellar disk up to b=1 which is the cusp of the disc. Fig 1.5 describes the change of the impact parameter and the resulting changes of the transit duration. If b gets bigger the transit time will increase.

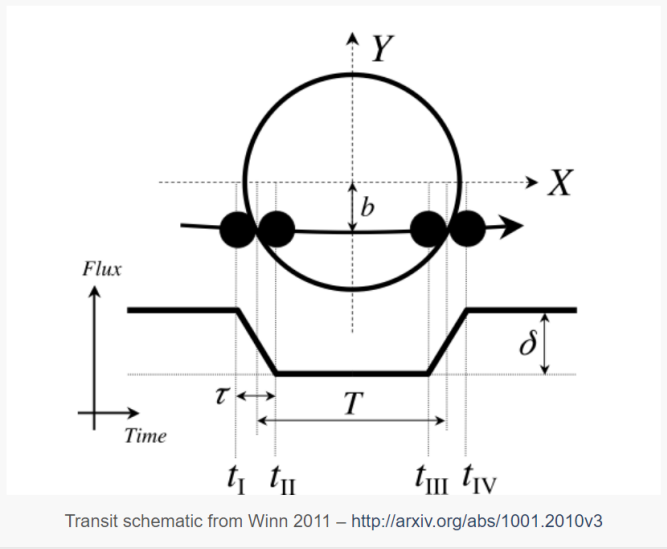

Figure 1.4: Detailed graphic of a transit.

Another important effect in the Transit Method is called Limb Darkening. It basically describes that the stellar disk of a star is brighter in the center than the limb of the disk. This means that through the effect of Limb Darkening the bottom of a dip will not be as rectangular as in Fig 1.4, instead it will become more round and it will be harder to determine when to start and end the full time. This effect is displayed in Fig 1.6.

The Transit Method is very effective and has already led to great success, however there are some problems with the Method. First, the Transit method can only detect planets that are passing in front of the star from the perspective of the observer. Secondly, an orbital inclination of the planet when executing a transit or a non-circular orbit of the planet make the analysis harder and can create a big error in the calculated parameters of the exoplanet. Important to note is also that a real light-curve does not look like in Fig 1.4. The light of a star does not follow such a steady and constant behavior. Instead there are several trends that can be found in the light-curve and many very big periodic patterns that point to star cycles or sunspots. The complexity of a star's light-curve strongly depends on the number of planets in a system. With a growing number of planets in a solar system the complexity of a light-curve also increases. Not only because more planets can mean more dips but also because every planet is unique, every planet has its own orbital velocity and radius which can lead to overlapping dips. However, based on the currently more then 3000 confirmed exoplanets it can be said that most of the planetary systems currently known to us consist of 1 or 2 planets.

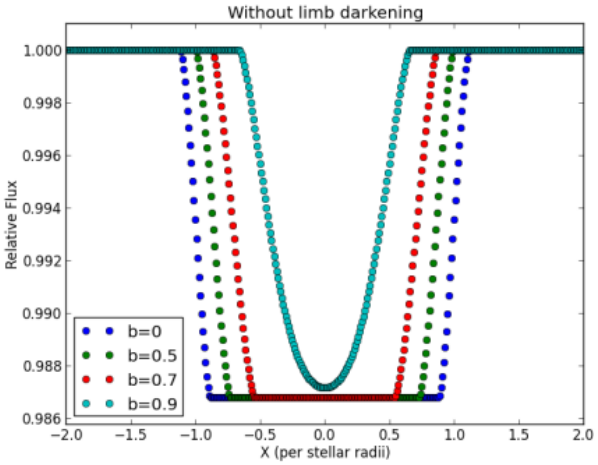

Figure 1.5: Different values of the impact parameter.

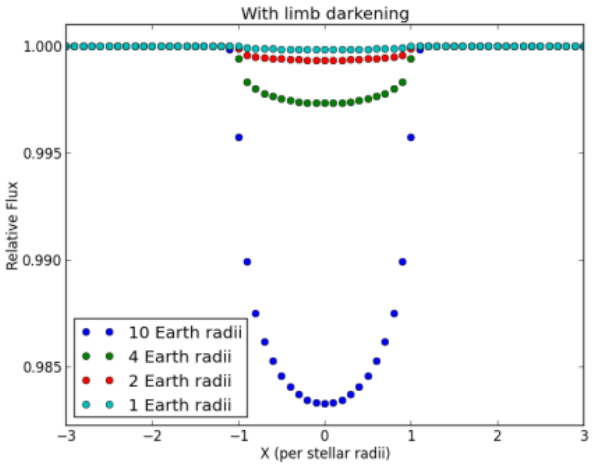

Figure 1.6: Different planet radii with the effect of Limb Darkening.

1.1.5.2.3 Mathematical background and candidate parameter calculation

A very important step in the search for exoplanets is analysing all the data that could potentially indicate the existence of a planet. Therefore basic properties should be determined or estimated to better understand the system.

To determine all the important and with the Transit Method calculable properties of an celestial object the first step is to find all abnormal, periodic dips or pattern that can be found in the light-curve of a star. Afterwards these dips or patterns get grouped together by comparing their depth, shape, total time and full time to create clusters with the associated evidence which points to the existence of a planet.

In this special process many not relevant or very uncertain pieces of information can be eliminated to underline the important and meaningful parts of the information found in the light-curve, like too small or nearly unrecognizable dips or patterns that could be small, insignificant objects crossing in front of the observed star, solar flares, sunspots or basic noise.

The parameter calculation is done in two steps. In the first step all dips will be used to determine the full time, total time and delta flux value. Afterwards all the clusters of dips will be used to determine the average period between the dips. The resulting information would be an array of dip information containing:

1. t_F - The full time, the time a planet needs after ingress until egress in days.
2. t_T - The total time, the duration of a planet's full transit in days.
3. ΔF - The delta flux value which describes the depth of a dip compared to the local, normal flux value of the host star in flux.

For more information on these properties see Fig 1.4.

All of this information is then used to determine the major properties of a potential planet like:

1. R_P - The radius in R_{Jup} which is calculated with the help of the star's radius.
2. t_{dur} - The duration of a transit which is basically the total time t_T.
3. v_P - The orbital velocity of the planet in m/s.
4. P - The period which describes the time of one full orbit around the host star in days.
5. a - The semi-major axis of a planet which describes the "average" distance between the planet and its host star in AU.

6. T_P - The thermal equilibrium temperature which is a theoretical temperature of a planet if it would be considered as a simple black body being heated by its host star.

For further calculations some major constants und units must be known:

1. $R_{Sun} = 695,700km$ - The radius of our Sun in kilometers.
2. $R_{Jup} = 69,911km$ - The radius of Jupiter in kilometers.
3. $R_{Earth} = 6,371km$ - The radius of our Earth in kilometers.
4. $M_{Sun} = 1.98855 \times 10^{30}kg$ - The mass of our Sun in kilograms.
5. $AU = 1.49597870700 \times 10^{11}m$ - The astronomical unit which is a unit of length describing roughly the distance between Earth and Sun.
6. $G = 6.674 \times 10^{-11}m^3kg^{-1}s^{-2}$ - The gravitational constant from Keplers's gravitational law.
7. $pc = \frac{648,000}{\pi} \times 1AU = 3.08567758149137 \times 10^{16}m$ - A parsec is a unit of length for measuring extremely large distances of objects outside our own solar system.

To determine the properties of a planet the most important information of the host star also has to be defined, namely:

1. M_S - The star's mass in sun masses(M_{Sun}).
2. R_S - The star's radius in sun radii (R_{Sun}).
3. R_{S_ERR1} - The star's positive error in sun radii (R_{Sun}).
4. R_{S_ERR2} - The star's negative error in sun radii (R_{Sun}).
5. D_S - The distance between earth and the star in parsec (pc).
6. T_S - The effective temperature of a star in Kelvin ($0K = -273.15°C$).

   Though stars are definitely no perfect black-bodies they can be seen as such to calculate their surface temperature with the help of the Stefan-Boltzmann Law: $L = 4\pi R^2 \sigma T_e^4$ with L the luminosity of the star, R the star's radius, σ the Stefan-Boltzmann constant($\sigma = 5.67 \times 10^{-8} Wm^{-2}K^{-4}$) and T_e the effective temperature of the star.

If some of the information of the host star is missing some of the properties of the planet cannot be calculated.

After defining all needed properties of the star, significant constants and units the properties of the planet can be calculated.

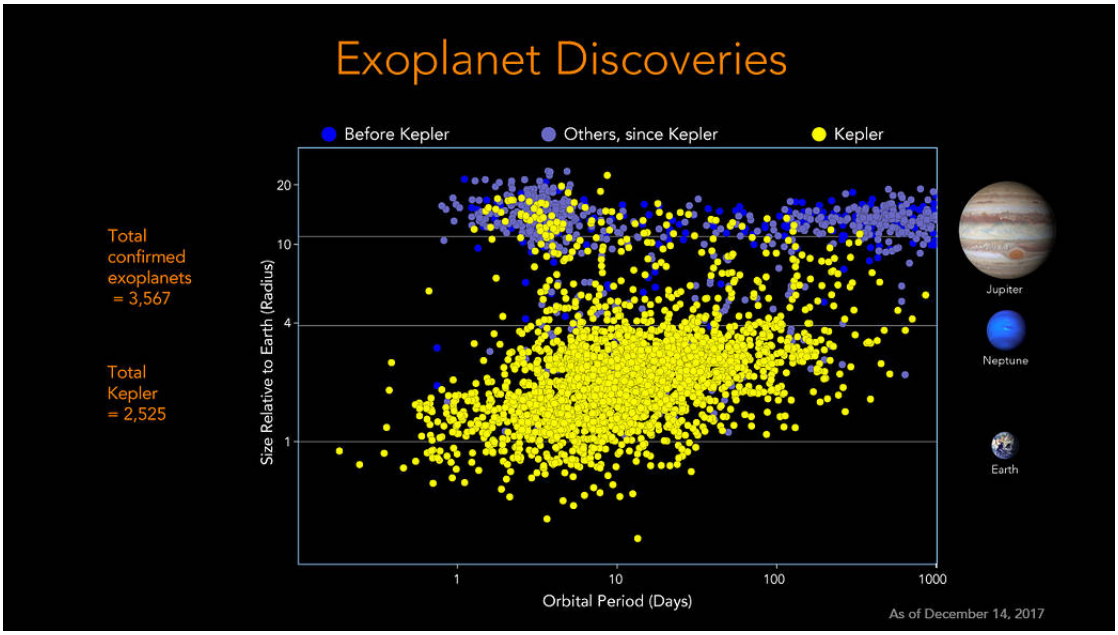

Figure 1.7: Confirmed exoplanets with radii and period.

Period of a planet - The period of a planet P is the duration of a full orbit. The period of a planet can only be determined if two or more dips were found that point to a planet. The number of dips and the accuracy of the planet's period are strongly connected; with an increasing number of dips the period's accuracy also increases.

When looking at Fig 1.7 it can be clearly seen that the majority of the currently confirmed exoplanets have a very short period, most of them between 1-100 days, which is compared to earth is a very short time. The reason for this it that most of the missions including the K2 Mission only watch a star for a short period of time, not allowing every planet of the system to cross the star during that observation time.

Radius of a planet - Avery significant parameter of exoplanets is the radius which can be calculated when using the Transit Method. Therefore the radius of the star needs to be given else the planet's radius cannot be determined. Often a single value for the star's radius is not accurate enough, so a positive and negative error has to be used to create an interval of error for both the star's radius as well as the planet's radius. Furthermore, the average of delta flux values of all dips of a planet have to be considered in the equation, for simplification the average of delta flux values will be named as ΔF. The radius of a planet is defined as:

$$R_P^2 = \frac{\Delta F}{F_S} \times R_S^2 \;/\; R_P^2 = \frac{\Delta F}{F_S} \times (R_S + R_{S\_ERR1})^2 \;/\; R_P^2 = \frac{\Delta F}{F_S} \times (R_S + R_{S\_ERR2})^2$$

Where R_P is the radius of the planet, ΔF is the delta value of fluxes, F_S is the local, normal value of flux, R_S is the star's radius and $R_{S\_ERR1/2}$ is the positive/negative error of the planet's radius. In Fig 1.7 the distribution of radii of all currently confirmed exoplanets can be seen.

Duration of transit - The duration of a single transit is basically the total time t_T of the dip. Therefor the average value of all dips must be determined.

Semi-major axis - The semi-major axis of a planet is its "average" distance from the star. It is often said that the semi-major axis is the "average" distance, although this is not quite right. The semi-major axis of a planet can be calculated by its orbital period and the star's mass the planet is orbiting. A planet's semi-major axis is defined as:

$$a = \sqrt[3]{\frac{P^2 G M_S}{4\pi}}$$

Where a is the semi-major axis, P is the planet's orbital period, G is the Gravitational constant and M_S is the star's mass. Together with the orbital period, the semi-major axis is one of the most important parameters describing a planet's orbit. The formula of the semi-major axis is related to Kepler's third law:

$T^2 \alpha a^3$ which is a simplification of Newton's two-body problem,

$T^2 = \frac{4\pi^2}{G \times (M+m)} \times a^3$ where M is the mass of the central object and m is the mass of the orbiting object. However m compared to M is insignificant and can be ignored.

Orbital velocity of a planet - The orbital velocity is the planet's tangential velocity. Therefore the planet's orbit must be assumed as a circular path around the star, although many planets circle their host star in a rather elliptical orbit. The average orbital speed of the planet is defined as:

$$\overline{v} = \frac{2\pi a}{P}$$

Where $\overline{v}$ is the average orbital velocity, a is the semi-major axis and P is the orbital period of the planet.

Equilibrium temperature of a planet - The equilibrium temperature of a planet basically describes the planet's surface heat after energy from the sun in form of sunlight hits the surface, some of the light is reflected back into space and another part is absorbed by the planet. This behavior is shown in Fig 1.8 The equilibrium temperature is defined as:

$$T_P = T_S \times \sqrt[4]{(1-a)} \times \sqrt{\frac{R_S}{2 \times D}}$$

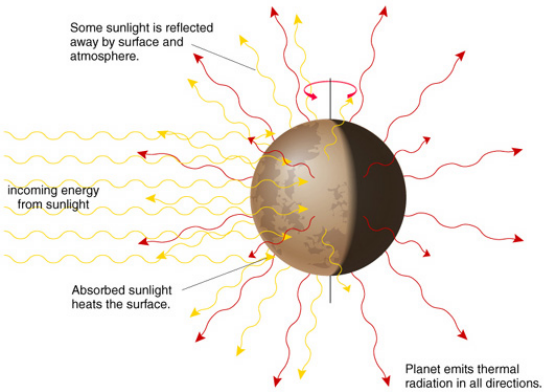

Figure 1.8: Equilibrium temperature simple explained.

Where T_P is the equilibrium temperature of the planet, T_S is the effective temperature of the star, R_S is the star's radius, D is the distance between the planet and the star(in this case the semi-major axis) and a the ALBEDO, a number describing the reflectivity of a planet(ALBEDO is in the interval of [0, 1] where 0 is a black object that absorbs 100% of light and reflects 0% and 1 is a white body that absorbs 0% but reflects 100% of the light).

1.2 Kepler, K2 Mission and NASA

The supply of data is an important part in analysing stars. The data is essential to train the neural network and also to run an analysis on. Luckily NASA offers APIs which provide all the data needed. NASA collects data from the Kepler Space Telescope which is constantly collecting the brightness information of each star with a photometer while having over 145000 stars in its field of view.

1.2.1 What are the Kepler and K2 Mission

The Kepler mission successfully sent a telescope to sun-orbit, free from all disturbances at earth in order to observe stars and their exoplanets. The telescope named after astronomer Johannes Kepler launched in march 2009 and was threatened to stop in 2013 due to technical difficulties as two out of four rotation wheels broke down.
In may 2014 K2 (Kepler 2), an upgrade to the broken Telescope, was launched. Currently K2 is operating but will shut down in late 2018 due to fuel shortage. The telescope weighs roughly 1040 kilograms and features 21 photometric cameras that combined have a field of view of about 12 degrees, which is approximately 0.25 percent of the whole sky.

The telescope is observing star clusters (as seen in figure 1.9) and collects the brightness of each star inside that group. Every cluster is followed for approximately 83 days, until a rotation is required to prevent light from the sun entering the telescope's lens. Each period of observing a cluster is called a campaign. Currently K2 is collecting data for the 15th campaign.

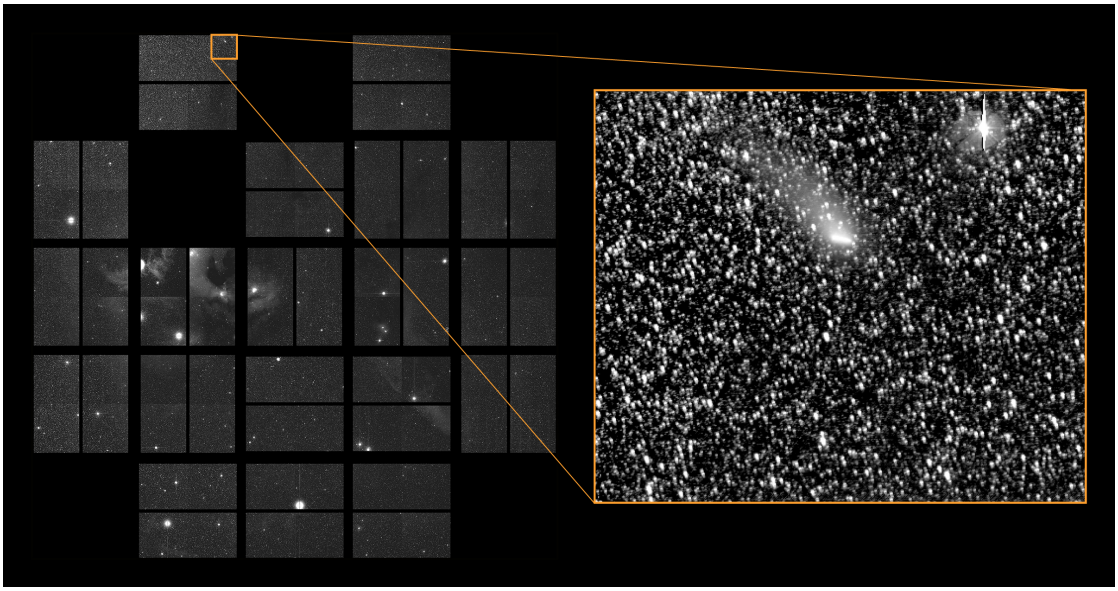

Figure 1.9: Cluster observed by the Kepler Telescope for the duration of one campaign.

1.2.2 Goal of the Kepler and K2 Mission

The goals of Kepler and K2 are to research planetary systems and identify their structures and diversities. These goals are achieved by

1. finding exoplanets of different host stars
2. identifying earth-like planets in the habitable zone
3. analysing the distribution of exoplanet's dimensions
4. finding the average number of planets in a star system
5. analysing the different properties of the observed stars
6. identifying additional attributes of planets orbiting their host star

1.2.3 Usage of K2 and Kepler lightcurves

The lightcurves display a star's brightness over a period of about 83 days. These graphs can be analysed by the human eye or by machines to find all kinds of anomalies. Lightcurves can be used to find planets in front of stars and also to approximately calculate the following parameters:

- size of the planet (radius)
- distance to the star
- orbital velocity
- equilibrium temperatures
- semi major axis

More details on parameter calculation can be found in section **??**

1.2.4 APIs and Databases

When querying the NASA exoplanet archive a certain structure has to be followed. By calling the base URL `https://exoplanetarchive.ipac.caltech.edu/cgi-bin/nstedAPI/nph-nstedAPI?` and adding all the parameters needed separated with *?* the database can be queried.
There are following parameters:

1. table (mandatory)

   table is the first and most important parameter. Querying the archive requires restriction to a certain table, like any database would. The NASA exoplanet archive provides following tables:

   - ***exoplanets*** Confirmed Planets
   - ***multiexopars*** Extended Planet Information
   - ***keplernames*** Kepler Numbers
   - ***koi*** Kepler Objects of Interest
   - ***stellar*** Kepler Stellar Tables
   - ***tce*** Threshold-Crossing Events
   - ***superwasptimeseries*** SuperWASP Time Series
   - ***k2targets*** K2 Targets
   - ***k2candidates*** K2 Candidates
   - ***k2names*** K2 Names
   - ***mission_exocat*** Mission Stellar
   - ***kelt*** KELT Time Series

2. select (optional)

   By choosing the columns with **select** the query only returns the columns stated. Columns are separated by commas. *select=** can be used to return all columns of a table.

3. order

   To order the result by a column **order=col** is used

4. format

   Definition of the output format can be done with **format**. Default option: csv

When using the NASA exoplanet archive only meta information about stars and planets can be retrieved. For actual raw time series data, the archive for *Reduced Light Curves from K2 Campaigns on MAST* (`https://www.cfa.harvard.edu/~avanderb/k2.html`) can provide.

The data is separated by campaigns and contains all stars with their light curves. The light curves are adjusted to remove incorrect data during thruster fire when turning the Kepler telescope. There is a .txt file for each lightcurve, which contains the time values along the x-axis and the flux values along the y-axis.

1.3 Neural Networks

1.3.1 The basic concept of Neural Networks

Neural Networks are very powerful tools to solve classification, recognition and many other problems of modern day computer science which are hard to solve with the help of ordinary algorithms. Neural Networks are inspired by the human nervous system and try to mimic the network of neurons in our brain as exactly as possible.

The long term target of Neural Networks is to ultimately create a super artificial intelligence that could outperform humans in every thinkable task and would from one day to next change humanity forever.

Although today's Neural Networks have not yet reached the point of a super artificial intelligence, they aim at and cannot compete with human intelligence in general or our collective human intelligence. However they are extremely good at solving single task problems like image recognition, classification, pattern recognition, regression, stock prediction, etc...

Dr. Robert Hecht-Nielsen who invented the first ever neurocomputer defines Neural Networks as:

> "...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs." (Dr. Robert Hecht-Nielsen)

1.3.1.1 Structure of a Neural Network

As stated above Neural Networks work based on the basic concepts of the human brain. A human brain typically contains about 86 billion nerve cells which are called neurons. These neurons are connected to thousand of other neutrons with the help of so called Axons to create a giant network which makes up our logical intelligence, memory and probably our consciousness.

The network gets its information from external sensors like our taste, our sense of hearing or our sense of touch. The collected information is sent from neuron to neuron, gets analysed by the network and gets outputted as a result. Basically this network has a bunch of inputs, analyses them and returns an output which can consist of one or multiple output values.

A Neural Network consists of multiple nodes which imitate the biological neurons. These nodes are connected with each other to send information through the network. This information is fed into the Neural Network through input nodes. Every node performs an operation on the input it gets and outputs a result which is typically called activation value.

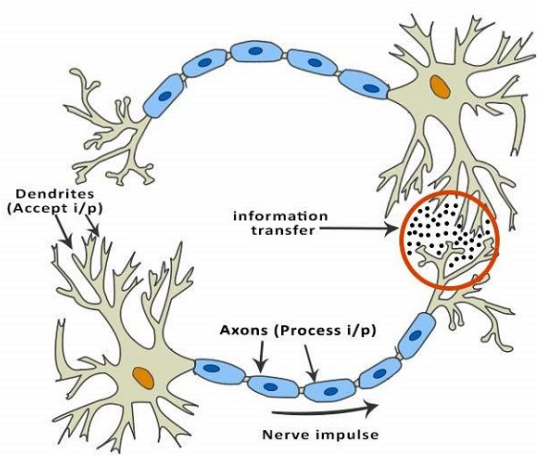

Figure 1.10: Structure of human neurons.

To give Neural Networks the ability to learn and remember information and associations each one of the connections between two nodes is associated with a so called weight. A weight is basically a numeric value which will be changed by the network to learn and adjust its behavior.

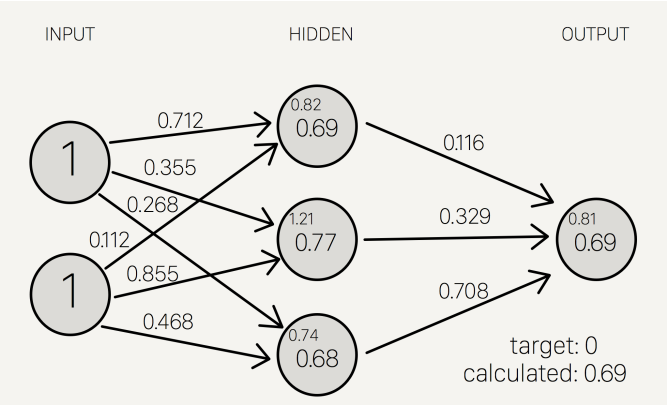

Figure 1.11: Structure of a simple Neural Network.

Fig 1.11 describes a simple Neural Network. It can clearly be seen that every connection between two nodes is associated with a numeric value, the so called weight.

The basic structure of a neural network does not consist of random connected neurons which create an obscure network. The contrary is true, a Neural Network is made up of layers of nodes. Every node in a layer is connected to every node from the layer before and after. The first layer is called the input layer which is responsible for inputting

the information into the network. The last layer is called output layer. It is responsible to for outputting the result of the whole network in an appropriate matrix. Within the network there are zero or several hidden layers which transport the information to the other nodes after applying an operation on the data.

1.3.1.2 Learning Neural Networks

When starting to train a Neural Network the first important thing is the initialization of the weights. There are several approaches and every one of them has advantages and disadvantages. One way is to randomly initialize the weight's matrix or to assign each weight a pre-defined value. The way the weights of a Neural Network get first initialize has a major impact on the accuracy of the finally trained network.

There are two major categories of learning methods used to modify the weights of a Neural Networks. (Important for this diploma thesis are only the supervised learning methods like Backpropagation.)

1. Supervised Learning
2. Reinforcement Learning

1.3.1.2.1 Supervised Learning needs, like the name already suggests, a teacher who helps the Neural Network to learn things. This particular teacher has to have a better knowledge of the problem and its solutions then the Network. The learning progress of the Neural Network begins by feeding the Network with a large amount of training data. The training data is a set of data in the domain of the targeted task which contains the problems X and the right solutions for each problem Y.

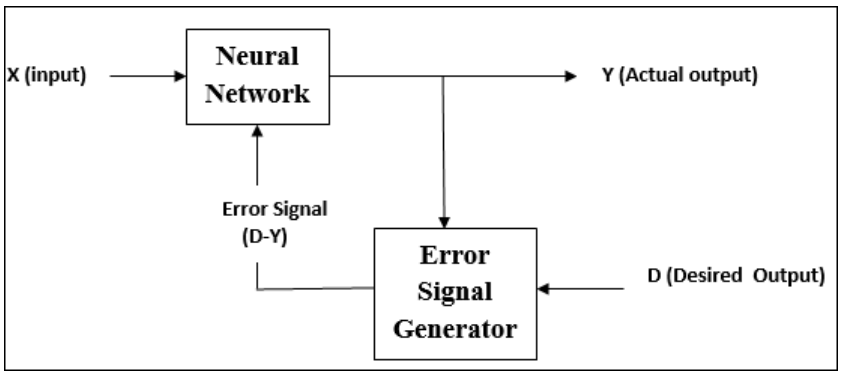

Figure 1.12: Diagramm of supervised learning.

Every time the Network outputs a solution, the result of the network gets compared with the result of the training data. If the output of the Network does not match the

solution of the training data the weights of the Neural Network will be adjusted until the output of the Network and the result of the training data matche to a certain degree. A very famous algorithm used to modify the weights of a Neural Network is called Backpropagation. This method is the perfect way to easily train a Neural Network. Backpropagation is often used in image recognition, classification problems and regression tasks.

The speed with which the Neural Network will learn depends on the learning rate. It is important to know that a too high learning rate can teach the Neural Network very fast but can also miss the aimed targeted aim of accuracy. On the other hand a too slow learning rate will increase the time it takes to teach the Neural Network and could lead to overlearning the network.

It is always important not to overlearn the Neural Network so it is able to recognize similar data(data it has already seen) and at the same time can work well with new data. Therefore it is important to test the Neural Network with appropriate test data(data which the Neural Network has never seen before) to determine the true accuracy of the Neural Network(testing the accuracy of a neural network with the training set has no real informative value).

1.3.2 Convolutional Neural Networks

Convolutional Neural Networks are a category of Neural Networks especially used in image recognition, speech recognition and classification. They have many advantages; for example they are easier to train, need fewer parameters and have many different types of modules to construct complex CNNs which can be applied to many problems.

Furthermore, the architecture of a Convolutional Neural Network is made for taking advantage of the two-dimensional structure of the input which will be fed to the Network, like 2D images or speech sources.

The basic idea behind CNNs is to subdivide the data into smaller parts called filters (the size of these filters is called kernel size or filter size). Then the Convolutional Neural Network searches in each of the filters for features. These features then create the new data and get also divided into smaller chunks of data to repeat the process over and over again. A single transit of all of these actions is called a convolutional layer. These layers are stacked upon one another to form the Convolutional Neural Network.

One of the most important tools of CNNs is called pooling. Pooling means that the data is minimized by not losing any of the important data. This is done to bundle the important data and to eliminate local and not necessary information. Pooling is often applied between two convolutional layers.

The tail of the Convolutional Neural Network is made up of fully connected layers or also

called dense layers. These layers take the information of the convolutions and put it all together for classification. Fully connected layers are similar to layers in a normal deep neural network with weights, biases and activation functions. The last fully connected layer represents the output of the Convolutional Neural Network and consists of different neurons each presenting a category in the classification. Fig 1.13 shows again the above described process of a Convolutional Neural Network.

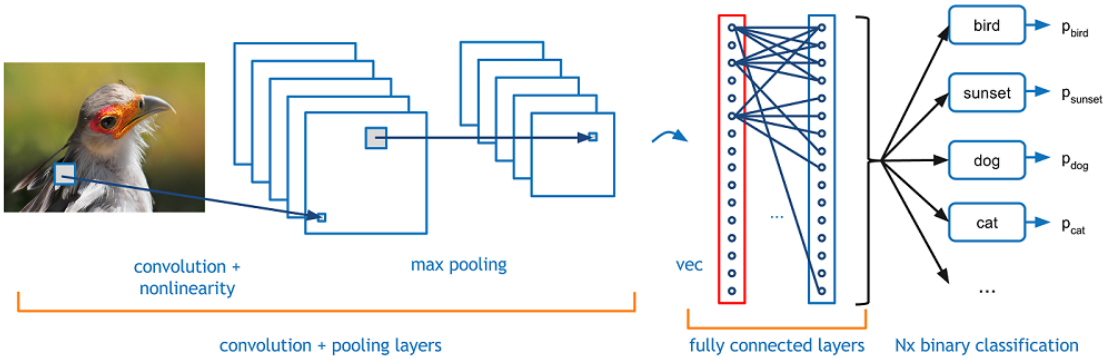

Figure 1.13: Process of a CNN.

There are some other modules to use in CNNs like:

1. **Batch Normalization**
   Provides faster learning and a higher overall learning rate.

2. **Flattening**
   Converts the data of convolutions to the input of fully connected layers

3. **Dropout**
   Mimics the loss of information in a human brains and prevents the network from overtraining.

1.3.3 Human versus artificial intelligence

What are the main differences between artificial intelligence and Human intelligence nowadays? This question is pretty hard to answer because the research in the field of artificial intelligence is a fast growing multi million dollar market which will expand even further in the next few years.

> "Artificial intelligence will reach human levels by around 2029. Follow that out further to, say, 2045, we will have multiplied the intelligence, the human biological machine intelligence of our civilization a billion-fold."
>
> (Ray Kurzweil)

When thinking about the most advanced strategic games humanity has ever created like chess or Go there is no game that a current, state of the art artificial intelligence would not be able to beat even the world's top players. Some years ago it seemed impossible for a machine to beat the world's top Go players. However, in 2017 this barrier was breached.

The big difference between humans and current artificial intelligence is that an artificial intelligence has no consciousness and lacks the strong logical intelligence humans have, but is still capable of solving problems a lot faster than a human being. Furthermore the learning process of an artificial intelligence compared to a human is also a lot faster and the information and logic stored can be easily transferred and replicated.

This big advantage of modern day artificial intelligence and the fast improvements that are made in this field will make sure that a vast majority of our jobs will be done by artificial intelligence in the near or far future. This is due to the fact that an artificial intelligence is faster, more cost efficient, will make a lot less errors and will never have to sleep or take a break.

Research into a super artificial intelligence has not yet created the ultimate machine that is smarter and more intelligent in every single problem that we could imagine. However, there are hundreds of examples of AIs that easily beat humans in isolated problems like pattern recognition, image recognition, speech recognition, games, automotive driving and many other modern day problems which are hard to be solved with the help of ordinary straight forward algorithms.

1.3.4 Technologies for implementing Neural Networks

Creating an Neural Network every time from scratch is very time consuming and very inefficient when there are many structures that every Neural Network has in common. Furthermore there are many mathematical functions, for example working with matrixes, which would be overwhelmingly difficult if every time someone wants to create an Neural Network he would have to implement all functions again. Therefore every major programming language has its own packages, libraries or modules to work with and create Neural Networks. For this diploma thesis the decision was made to use Python based technologies to work with Neural Network. There are three major Python based technologies which are great when trying to create and work with Neural Networks.

1.3.4.1 TensorFlow

TensorFlow is an open source library which was create from the Google Brain team in November 2015. Since then it has created a big community which led to hundreds and thousand of tutorials, blog entries and examples which, in combination with a very good and detailed documentation, made TensorFlow a perfect tool to solve machine learning problems. There are several well known companies that use TensorFlow like airbnb, nvidia, user, sap, dropbox, ebay, google, twitter, snapchat and many more.

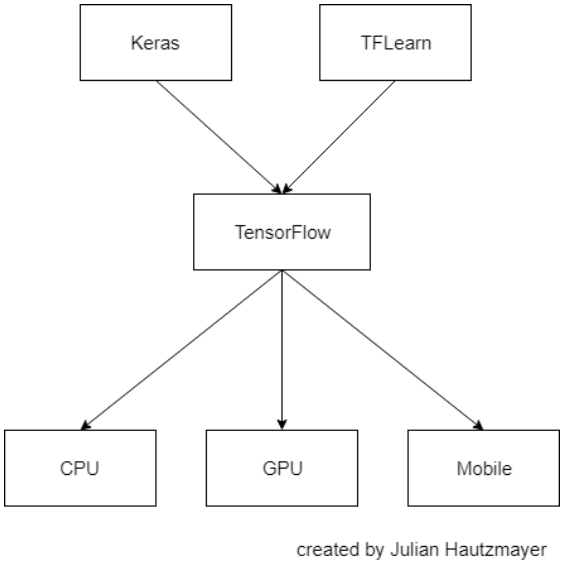

Figure 1.14: Diagramm of TensorFlow, TFLearn and Keras.

The big question is now what is TensorFlow's functionality all about? TensorFlow is a library for doing complex numerical computations using data-flow graphs to solve problems in the domain of modern computer science and to build machine learning models like Neural Networks.

One of TensorFlow's big advantages it that it is cross-platform and runs on nearly every GPU, CPU and mobile platform. The data-flow graph models make is easy to distribute across GPUs, CPUs and multiple systems which is another big advantage of TensorFlow. Furthermore the graph models make it a perfect tool for working with Neural Networks.

Additionally, TensorFlow has a visualization software called TensorBoard which makes analysing, understanding and debugging of machine learning models a lot more easier.

1.3.4.2 TFLearn

TFLearn is a High-Level API built on top of TensorFlow to make creating, working with, testing and training Neural Network models a lot easier while at the same time keeping the transparency and usability that TensorFlow provides. Furthermore TFLearn supports the developer with a whole bunch of tutorials, examples and a good and easy-to-read documentation for implementing deep neural networks.

1.3.4.3 Keras

Keras is also a High-Level API developed in Python for creating deep neural networks. It can run either on TensorFlow, CNTK or Theano and was developed to enable fast experimentation. The guiding principles of Keras are:

1. **User friendliness**
   Trying to minimize user action for common use cases.
2. **Modularity**
   Easy to build neural networks with the help of different building blocks like neural layers, cost functions, activation functions, optimizers and a lot more.
3. **Extensibility**
   Easy to add new modules with the help of new classes and functions.
4. **Working with Python**
   Neural Network Models are defined in Python which makes it easier to work with.

1.3.5 Importance of data

Data is the most important thing when talking about Neural Networks. It is the information that was previously collected which can be learnt and mastered by such a network. That means that the best Neural Network can only be as good as the data it was trained with. Therefore the quantity and the quality of the data play an important role in creating and training a Neural Network.

1.3.5.1 Data preparation

is a major task to accomplish before the data can be used to train a Neural Network. Often the raw data is not in a useable format or has errors that have to be eliminated to increase the quality of the data. Errors of sensors, background sounds in a sound file, basic random noise or other phenomena should be removed before the data is used.

Also other tasks like padding all incoming data to a certain length, shrinking the values by scaling each to zero mean and unit variance or normalizing all values between an interval for example [-1, 1], can improve the quality of the data.

1.3.5.2 Training and testing data

are an important part in training a Neural Network. The wrong set of training data can lead to a badly trained network and can tamper the accuracy reached by the network. If a set of data with the expected output exists it would make no sense to completely use it for training the Neural Network because there would not be any "new" data left to test the Network with. That is why it is normal that the data is divided in a training and a testing set.

The training data will be used to train the Neural Network and the test data will be used after training to test the accuracy of the Network. It is important that the test data is not used during the training phase to get the pure accuracy of the trained Neural Network. There are many different ways how to separate the data. Sometimes it is enough to split 70% training data and 30% testing data, for other networks or problems 50% / 50% is necessary to get the best results.

Sometimes the testing data itself is divided into two parts. The first part is used to select the best working approach for your model(validation) and the second part is only used to determine the accuracy(test). In such a case the data could be distributed 50% training data, 25% validation data and 25% testing data.

1.4 Time series analysis

1.4.1 Time series

1.4.1.1 What are time series

A time series is an ordered, discrete function of values that were monitored over a fixed period of time with equal space between those values. Time series are an important tool in computer science for collecting time dependent data of a variable. They can be applied on every variable that changes its value over time like wind speed, power usage, the stock market or the weather.

Time series are mainly used for:

1. understanding the underlying structures, forces, the composition and the behavior of what creates the time series itself
2. monitoring, tracking metrics, analysing and making decisions based on the behavior and trend of the underlying data
3. trying to forecast future events and behavior of the data based on historical data

Especially in the age of the Internet of Things movement, time series are an important asset because they allow the systems to better understand for example the power usage, heat or humidity in a smart home, the heart rate of a patient or the sleep pattern of a person.

The trends, periodic patterns and events that can be extracted out of this data are used to better understand diseases, the stock market(to understand the effect on a company's stock after the release of a new product), the economy or to recognize signs of heart attacks early. The data can be used to feed it to Neural Networks which can learn from the data or forecast major events in the future.

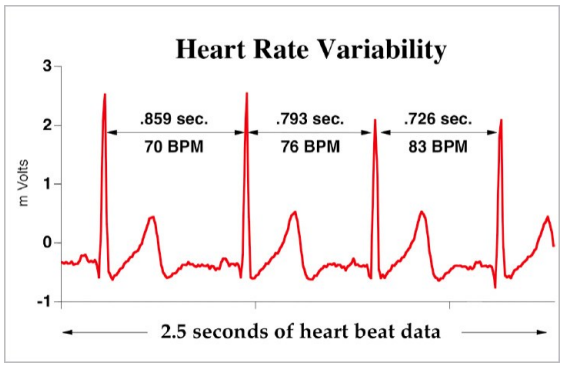

Figure 1.15: Time series describing the heart beat of a patient.

1.4.1.2 What is time series analysis

Time series analysis is the task of preparing the monitored data and analysing it. As discussed above, time series can contain sensor errors, anomalies, noise or other forms of error inside them that can obstruct the analysing process. Additionally by changing the scale of the data or removing insignificant information the data can be simplified and the computing power of the analysis can be decreased.

There are several methods for preparing and afterwards analysing time series which will be discussed in the coming section.

In the case of the Transit Method the light-curve created for example by the K2 Mission is the time series data which will be analysed with the methods and techniques used in ordinary time series analysis.

1.4.2 Preparing time series

A good preparation and cleaning of the data used for the analysis is an essential task because the results of the analysis can only be as good as the data. Therefore it is extremely important to bring the data in a clean form without major errors, noise or too much insignificant information which could obstruct the analysis process.

1.4.2.1 Global and local information

There are two important types of information in time series, the first is global information and the second is local information. Global information means information that applies to all the data of a time series like the global mean or the global standard deviation which can be calculated by using all data points of the series.

Local information however only applies to a subset of data points. Typically a single point which is the "center of gravity" and to the left and right a number of n neighbors. The bigger n gets the more global the information becomes again. When searching for patterns in the time series to find a planet, local information is often more important than global.

1.4.2.2 Detrend data

The global trend of a time series describes the trend of the value the time series has monitored. Whether or not this information is needed heavily depends on the purpose of the analysis.

In the case of the Transit Method the information of the global trend can increase the difficulty of the analysing process, therefore the time series gets detrended to remove the global trend of the flux values without losing essential information concerning the exoplanets that could hide inside the data. Through the activity of a star the flux values

can change over a period of time which affects the flux values a lot more than a tiny exoplanet crossing in front of it. Furthermore the very common binary systems(two stars or more orbiting each other) can create large periodic patterns in the light monitored by the light curve which can distract the analysis.

In Fig 1.16 a time series with a simple linear trend is shown. After identifying and removing the trend without losing important information the resulting data is now a simplification of the old data that can be seen in Fig 1.16.

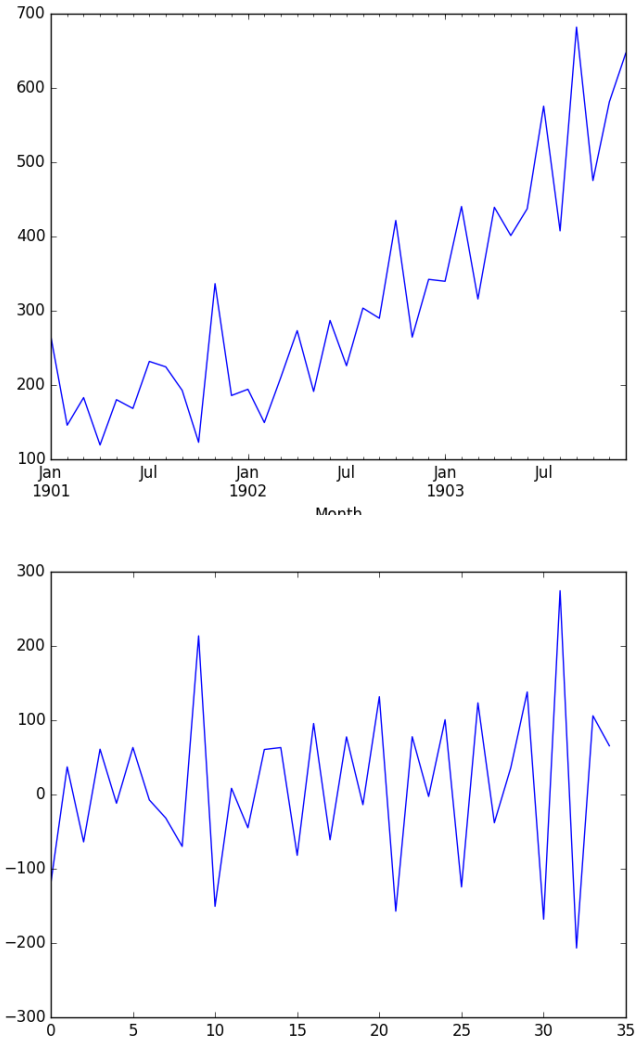

Figure 1.16: Time series with and without a linear trend.

1.4.2.3 Noise reduction

Simplifly put, noise is random data; in this case noise is random and very insignificant data. When working with the Transit Method noise represent changes in the flux values that are too weak to indicate the existence of a planet. By removing this noise the time series can be simplified and is easier to analyse. There is a very small risk to lose information that could be important; however, this loss can be accepted.

Noise reduction can be done by iterating through the time series and removing every data which is beneath a certain threshold, like a portion of the local mean. Therefore the local mean is calculated by summing up the Y values of a subset of the data points, which are direct neighbors of the data point which is currently checked and dividing the sum by the count of used points.

1.4.2.4 Outlier detection

In contrast to noise reduction, outlier detection means to remove data points which completely ignore the trend of the local information. A good example would be a nearly linear function and a suddenly appearing data point which is 3-4 times that of the mean value.

Especially with the Transit Method these outliers(when using the Transit Method outliers are only points which go in the positive direction of the Y-axis because every negative movement on the Y-axis could be an indicator of an exoplanet) can tangle the calculation of the local mean(moving average).

Outlier detection can be done by iterating through the time series and removing all data points(or setting them to an appropriate value like the local mean) by comparing them with a certain threshold like the multiple of the local mean.

1.4.2.5 Z-normalization

Z-normalization or "Normalization of zero mean and unit of energy" is a method which helps the analysis to focus on the structural information of the data rather than on the Y values.

When working with the Transit Method, Z-normalization is an important preparation process because the structural information in the time series is of great value.

To achieve this the time series data is subtracted by the global mean and after that divided by the global standard deviation. The result is a time series whose mean is roughly 0 and has a standard deviation of about 1. In Fig 1.17 the results of the Z-normalization on two different time series can be seen.

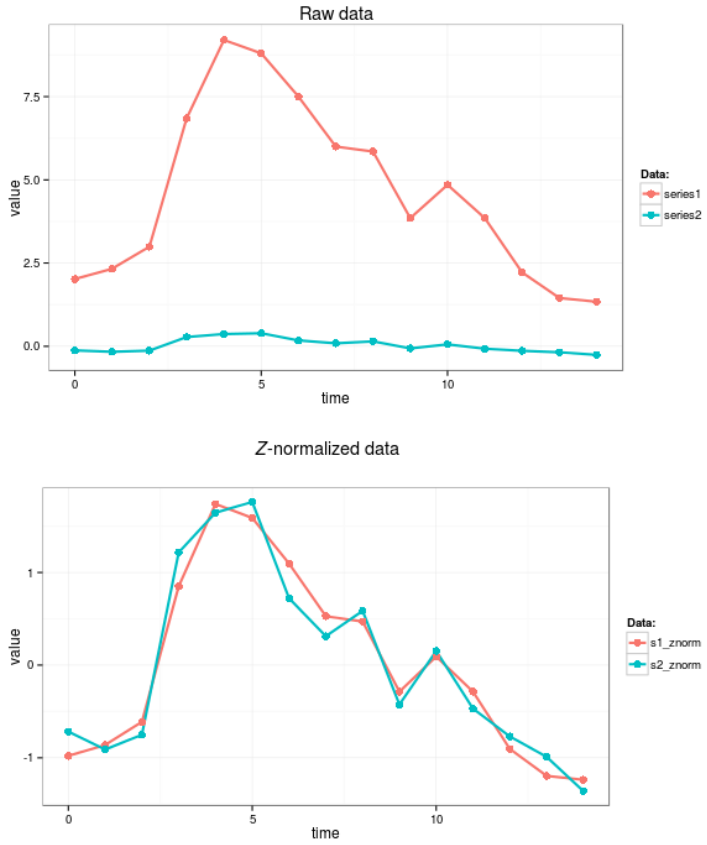

Figure 1.17: Time series with and without Z-normalization.

1.4.3 Moving average

The moving average is a powerful method to obtain local information of the data by determining the average of a subset of data points. It is often used for forecasting or, in the case of the Transit Method, getting information of the local trend of the data to find dips that refer to exoplanets.

The calculation of the moving average is very simple, it is basically the local mean of a subset of data points from the time series.

1.5 Clustering

1.5.1 Supervised vs. unsupervised learning

The field of machine learning can be separated in two big clusters. On the one hand there is supervised machine learning and on the other side there is unsupervised machine learning. Both of them contain very powerful tools to solve modern day problems of computer science.

1.5.1.1 Supervised machine learning

Supervised machine learning is the more widely used form of machine learning. In the field of supervised machine learning every problem needs to have a basic set of training data to train a model with. Like a Neural Network which should learn how to solve a certain problem.

Normally an input vector of X is given, the model tries to solve the problem and finally outputs an output vector Y. The main difference to unsupervised learning is that the solution of the problem is already known so the model can learn how to solve such problems.

Famous examples of supervised learning are classification and regression problems.

1.5.1.2 Unsupervised machine learning

Unsupervised machine learning is the other field of machine learning. When using unsupervised learning an input vector X is given, but the solution to the problem is unknown. There is no training data a model could learn from; instead the algorithm has to figure out a solution on its own.

The purpose of unsupervised machine learning is to understand the underlying structure and schema of a problem and to better understand the data. There is no teacher like in supervised learning who can help the model learn how to solve the problem, the algorithm is on its own.

A very important example of unsupervised learning is clustering which will be further discussed in the next section.

1.5.2 What is clustering?

Clustering is an important unsupervised machine learning problem the purpose of which is to find the underlying structures and similarities in an array of data points. Essential to this problem is that the input data is not labeled or grouped beforehand.

Basically clustering converts a collection of data points into a collection of clusters,

where a cluster is an array of data points which are classified by the clustering algorithm to be similar in structure or based on their properties. A simple clustering example can be seen in Fig 1.18.

Clustering can have many different goals like finding data classes in the data, better understanding the structure and distribution of the data, finding groups in the data, finding a certain pattern or behavior in the data, reducing the data to clusters, searching for unknown properties in the data or detecting outliers(data points which are hard to associate with other clusters).

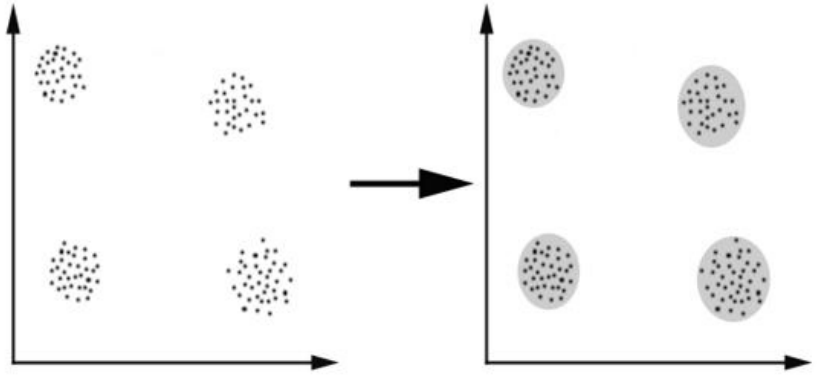

Figure 1.18: Simple clustering example.

A major part in clustering is to decide how to measure the distance between two data points or clusters. The distance can be referred to as a value which describes the similarity or dissimilarity of two data points. The function calculating this number is called cost function of the clustering algorithm.

1.5.3 Clustering algorithms

There are several clustering algorithm types which all have their advantages and disadvantages. Two of them are the hierarchical algorithms and the partitive algorithms.

1.5.3.1 Hierarchical algorithms

Hierarchical algorithms can be top down(divisive) or bottom up(agglomerative). The hierarchical bottom up algorithm for example follows a simple pattern. At first every data point gets its own cluster. After that the centroid of every cluster is calculated. The cluster's centroid is the average of all data points in a cluster(the centroid can also be the data point that has the smallest distance to all other points in the cluster; there are many different approaches).

Then the distance between the cluster's centroids is calculated and the two clusters with the smallest distance get merged and a centroid for the new cluster is determined(There are again many different approaches to determine which two clusters have the smallest distance like single linkage, complete linkage or average linkage).

This cycle is repeated over and over again until there is only a single cluster left. The whole clustering process can also be stopped before only one cluster is left by breaking out of the loop if a certain threshold or criteria is reached. This whole process can be seen in Fig 1.19.

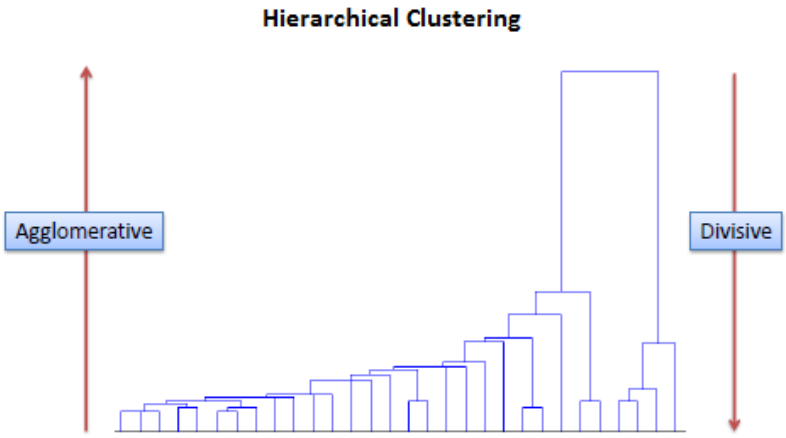

Figure 1.19: Hierachical clustering algorithms.

1.5.3.2 Partitive algorithms

Partitive algorithms like K-Mean clustering tries to cluster all the data points into k clusters. Every point of the input belongs to the cluster which has the nearest mean compared to the data point's own value. The big difference between hierarchical and partitive is that the partitive method can only create k different clusters where k must be defined first. The best number of clusters must be determined by trying different values for k.

The K-mean algorithm can be easily explained. At first k centroids each representing a cluster will be randomly placed in the data. After that there are two major steps that will be repeated over an over again until the clusters stop changing or another stopping condition occurs.

In the first step the clusters will be assigned data points. That means, that every

point will be added to one of the k clusters with the shortest distance to the data point. The second step involves moving the centroid of every cluster by calculating the average of all data points in the cluster. In Fig. 1.20 the single steps of an K-mean algorithm can be seen.

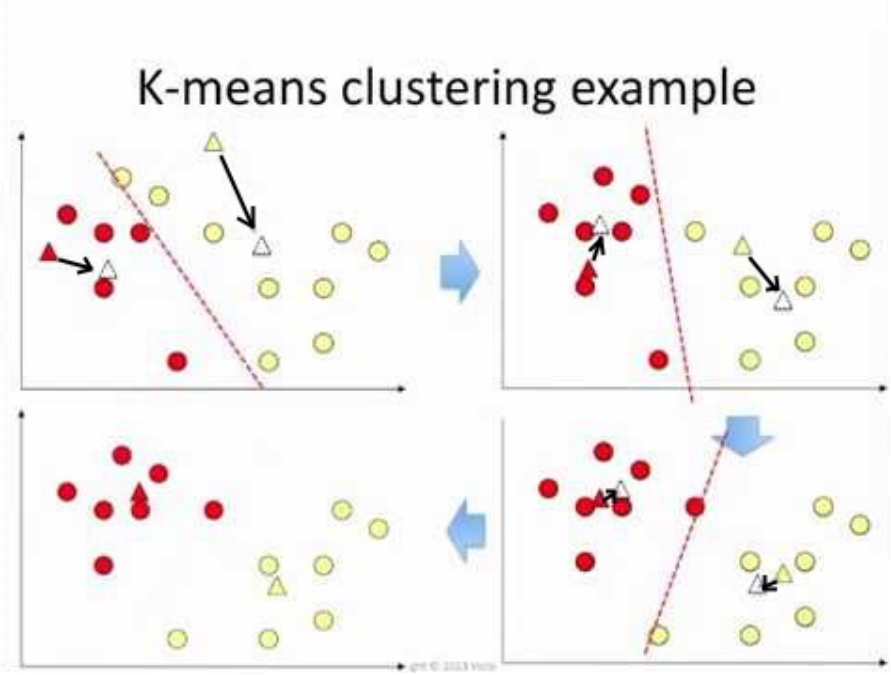

Figure 1.20: The single steps of a K-mean clustering algorithm.

1.5.4 Cost functions

Cost functions help to calculate a value which describes the distance between two data points or clusters. There are several mathematical ways to determine the distance like:

1. Euclidean distance $\sqrt{\sum_{j=1}^{n}(x_j - y_j)^2}$
2. Manhatten distance $\sum j = 1^n |x_j - y_j|$
3. Dynamic time warping algorithm (see Dynamic time warping algorithm)

1.5.5 Dynamic time warping algorithm

The dynamic time warping algorithm aims to find the optimal non-linear alignment of two different time series. Compared to the Euclidean distance the algorithm is much more interested in the structural information by coping with the problem of pessimistic similarity measurements created by distortion on the x-axis(time-axis).

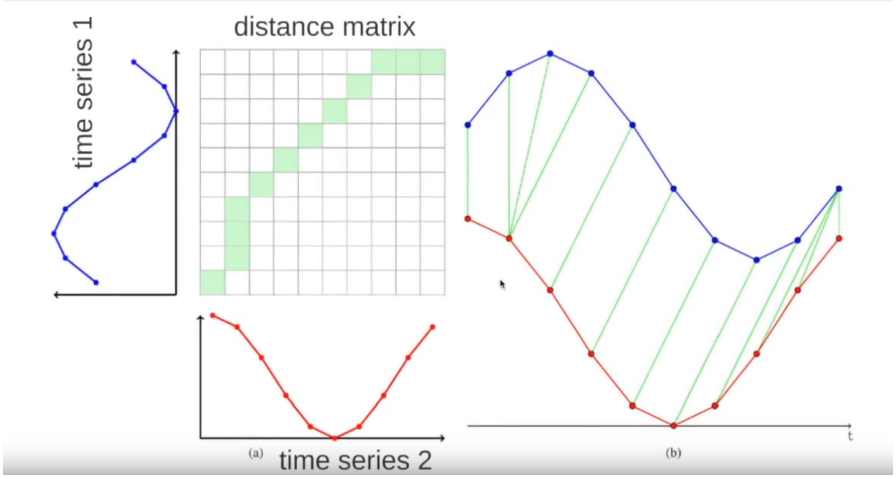

Figure 1.21: Dynamic time warping algorithm example.

1.5.5.1 How does DTW work?

Imagine two input vectors A and B with the same size s. At first the algorithm creates an matrix M of size $s \times s$. For every element $M_{i,j}$, the Euclidean distance between A_i and B_j will be calculated. Afterwards the path through the matrix M, describing the minimal cumulative distance will be determined. This path can then be used to calculate the distance between the two time series. The whole process can be seen in Fig 1.21.

1.6 Electron desktop client

A client with a sophisticated user interface is a necessity so users do not have any difficulties with using the application. The Electron Framework comes in handy, as it allows for a cross platform implementation of a download-able desktop client with HTML,CSS and JavaScript. The client mainly focuses on the user interface since the logic is completely encapsulated in python scripts. The separation of logic and UI makes individual implementation a lot easier.

1.6.1 Electron

Electron is an open-source framework developed by GitHub and allows web-programming with HTML, CSS and JavaScript on a desktop environment. Electron uses Node.js for the back-end and the chromium engine for the front-end to render the HTML DOM. Electron is available under the MIT-license and can be found on the public git repository `https://github.com/electron/electron`

The basic components of an Electron project are **package.json**, **main.js** and **index.html**. The metadata of the project such as name, version and used dependencies are written in the package.json. The main.js is the Node.js back-end of the Electron application. The structure of the user interface is described in the index.html just like a web-page would be designed. It is also possible and recommended to include CSS style-sheets and JS-scripts in the index.html.
The view is rendered via the chromium engine into a so called BrowserWindow, which frames our web-app into a desktop window. Chromium is the rendering engine that the Google Chrome browser uses which means that Electron and Google Chrome share the same web compatibilities.
A problem with Electron is that the main.js back-end runs in a so called main process and all view-related scripts run in separate render-processes as seen in figure 1.22. This makes data-flow between renderer processes and the main process a little difficult but is possible via the Electron remote module.

The last step of making an Electron app is to build it for a certain operating system. Electron can build executables for Windows, Linux and iOS.

Considered alternatives for a desktop application are:

- C# WPF
- C# WinForms
- C# Xamarin
- JavaFX
- sciter

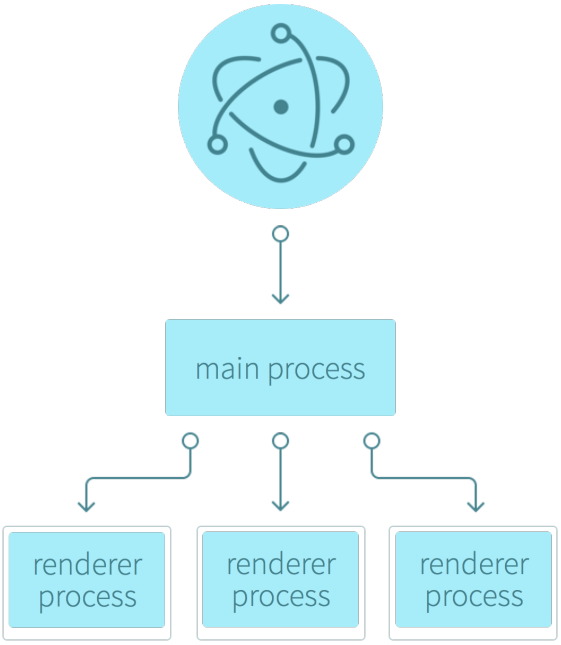

Figure 1.22: Electron's main process and multiple renderer processes

1.6.2 Canvas.js

In order to display massive amounts of data on the client user interface, a good-looking, well performing chart library which could handle large amounts of data on top of it seems like a big requirement. Luckily Canvas.js covers these needs and provides a wide range of chart types with a free payment plan. Canvas.js also offers an excellent documentation and support. Big companies like NASA, Microsoft, Apple, Intel and many more also use Canvas.js due to its overall superior qualities.

Another great feature of canvas.js is the ability to modify datasets dynamically by adding and removing data entries without having to reload the complete chart. Also panning and zooming can be enabled in the configuration, which is very important for this projects, since a lot of dips are very small and hard to spot without being able to zoom into a more detailed view.

1.6.3 Calling processes from Electron

In order to call python scripts from the Electron client a package that allows calling processes is necessary. In this case **child_process** is the perfect module to use. child_process.spawn allows for calling any process that would run in the command prompt. Spawn also allows the data exchange between client and instantiated processes via Inter-Process Communication (ICP) over input, output and error streams. As seen in figure 1.24, there is the input stream via stdin, which is the data stream from client

to process, the output stream as stdout, which is the data received from the process and an error stream as stderr, which transfers errors that occur in the process to the client.

For each stream there are two events. The "data" event collects chunks of bytes that have to be put together in memory. The "end" event defines the end of the last transmission, meaning that the collected chunks in memory are now complete and can be converted to strings that make sense.

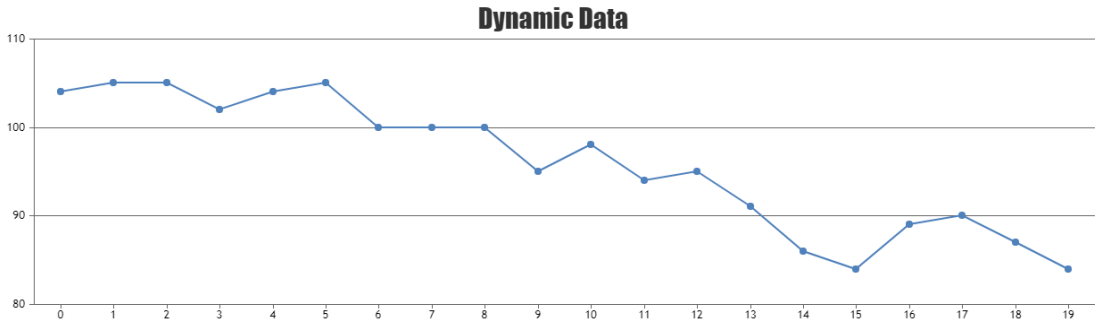

Figure 1.23: Example of a canvas.js dynamic chart.

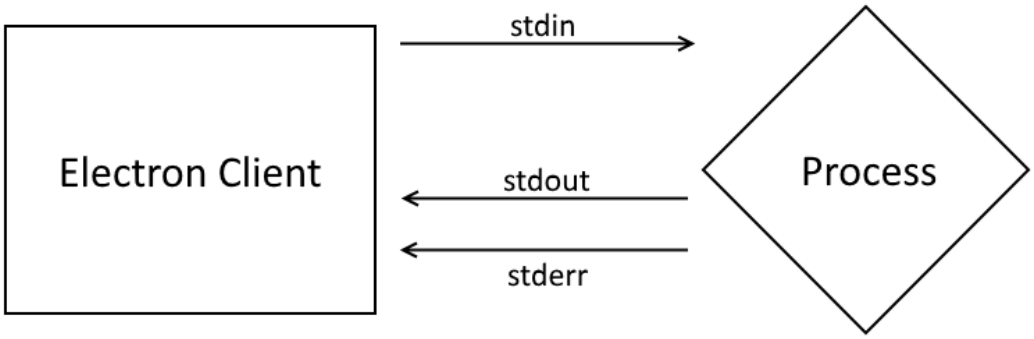

Figure 1.24: Available streams for communication between processes and client

1.7 JavaEE Server

The server for this diploma thesis is built on a Java Enterprise server and allows the data transfer between website, client, database and online archives. As seen in figure 1.25, the K2DataProvider centralizes all data flow. Technologies like REST, JPA and Jsoup are used to communicate.

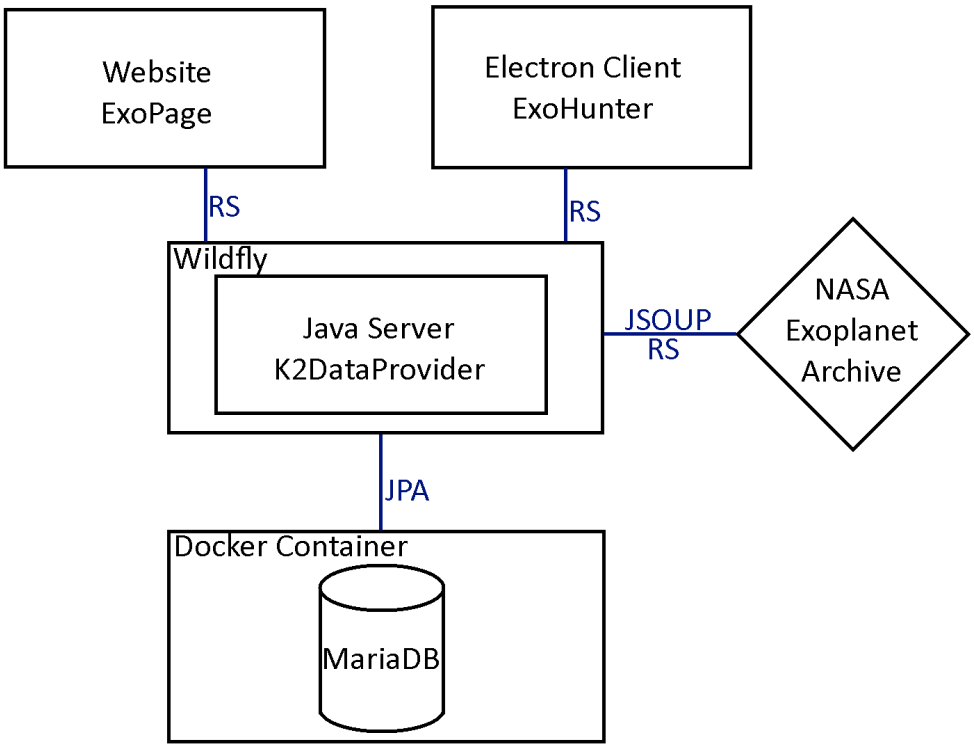

Figure 1.25: Technologies used of the Java server

1.7.1 Wildfly

Wildfly is an application server for Java applications similar to Glassfish which originated from EJB-OSS (Enterprise Java Bean Open Source Software) in 1999. Authored by JBoss and developed by Red Hat, Wildfly is available for free and is an open-source software. It can be used on all platforms and allows the deploying of Java EE web archives.
Features of the Wildfly application server are:

- Clustering
- Load balancing
- Administration API
- Deployment API
- Hibernate for JPA, as seen in section 1.7.4
- etc.

Familiar alternatives for the Wildfly application server are Glassfish, Tomcat, Payara and IBM WebSphere.

1.7.2 MariaDB

MariaDB is a relational database management system (RDBMS) developed by the MariaDB Foundation, mostly community-driven. As a fork of MySQL it shares the same features as MySQL before version 5.5. MariaDB forked from MySQL because developers feared the acquisition by Oracle and wanted their individual branch. As a result MariaDB changed their current version to 10.x to signalize the independence of MySQL which is at 5.x at the moment. Currently MariaDB's latest release is 10.2.

MariaDB's benefits over MySQL are:

- transparent development
- improved performance
- better clustering
- increased amount of storage engines
- more features

Alternatives for a RDBMS are Oracle DB, MySQL, Apache Derby DB, SQLite, Microsoft SQL Server and many more. But there are not only relational database systems, but also NoSQL databases like MongoDB or Apache's CouchDB that come to mind. It is more beneficial to use relational databases when the schema and structure is known. If this is not the case, a NoSQL DB might be more suitable.

1.7.3 Docker

Docker allows for virtualization on an operating-system level, on a very lightweight scale. Virtualization means that a virtual "container" is used to simulate an environment for another operating system within the host operating system. Programs that run inside the virtual container are unable to tell that they run in a simulated environment. Virtualization is a great thing if a program only runs in one environment and has to be used in another. Docker does, unlike other virtual machines, provide a very lightweight layer, which does not consume a lot of computer processing power but is in exchange only meant for a single process, even though multiple processes are possible as well.
Docker's first release was in early 2013 and is still actively developed by Docker Inc. and is available under open-source.

This is how Docker works:

Docker is mainly based on containers and images. Containers, which mostly contain a single program, can be thought of an entire encapsulated operating system. Running multiple programs in one container is possible but not recommended. Containers are instances of so called images which are basically snapshots of a file system, containing the information about operating system, installed programs, and configurations. Images can be based on other images, while storing only the changes to the image that they were inherited from.

Another great feature of docker is dockerfiles. A Dockerfile stores the instruction to build an image in a text format. The instructions can tell docker what operating system to use, which programs to be installed and generally what commands should be executed.

A thing to consider is that docker containers run in their own environments, which means they have their own network. This allows for multiple containers to internally use the same ports, but externally use different ports on the host, as they can be rerouted.

1.7.4 JPA and Hibernate

Hibernate is a framework which is responsible for relational data management in Java. Hibernate is a so called ORM, an Object-Relational-Mapper, meaning that it is responsible for the mapping of Java objects to data objects in relational databases and the other way around. It is available under the GNU Lesser General Public License and is usable for free.

Figure 1.26: The Hibernate framework is an Object-Relational-Mapper

JPA, meaning Java Persistence Api, uses Hibernate to simplify the steps of mapping objects to the database. By using annotations, classes/entities can be assigned to tables and fields can be assigned to columns. JPA also eases id-management and relations between entities. By those annotations, JPA can implicitly convert between Java objects and database objects. Transactional control is also made easy, since JPA has built in functions that define transactions starts and commits.

JPA provides a higher comfort level than JDBC, which only gives access to databases via SQL. It builds on top of JDBC and extends its features. For advanced queries JPA offers JPQL, meaning Java Persistence Query Language, which allows for a object-oriented SQL-mix.

Performance wise, JPA has a higher performance cost than JDBC, sacrificing speed for comfort. This is the main reason that it is sometimes wiser to use JDBC when a high performance database access is necessary.

1.7.5 Java REST

REST, meaning **REpresentational State Transfer**, is a web service protocol based on HTTP, which allows stateless access to web-resources via different operations. The data sent over REST are mainly text-based formats, such as XML, JSON and HTML. REST's core feature is a request-response-message exchange, meaning that for every request with any operation a response returns. As already mentioned REST has different operations for different applications. The most typical operations for restful services are GET, PUT, POST, DELETE. The most important one is a GET-request, the operation that lies underneath every web-page call. When entering a URL into a browser, a GET-request gets sent to the web-server that is reachable at the entered URL and then sends a response to the caller, containing a HTML string, which is then rendered on the

clients browser.

In Java, REST on the server-side is commonly implemented with JAX-RS, a widely used library for restful services. REST endpoints can be accessed with paths in a tree-like structure and by defining the operation like GET, PUT, POST, DELETE, etc.. When using JAX-RS, endpoints are defined by annotations, like @GET for a GET-endpoint and @Path to describe the navigation to that endpoint. Additionally there are @Consumes and @Produces which implicitly convert in- or output to the wanted data format, like convert JSON-strings to Java objects and vice versa.

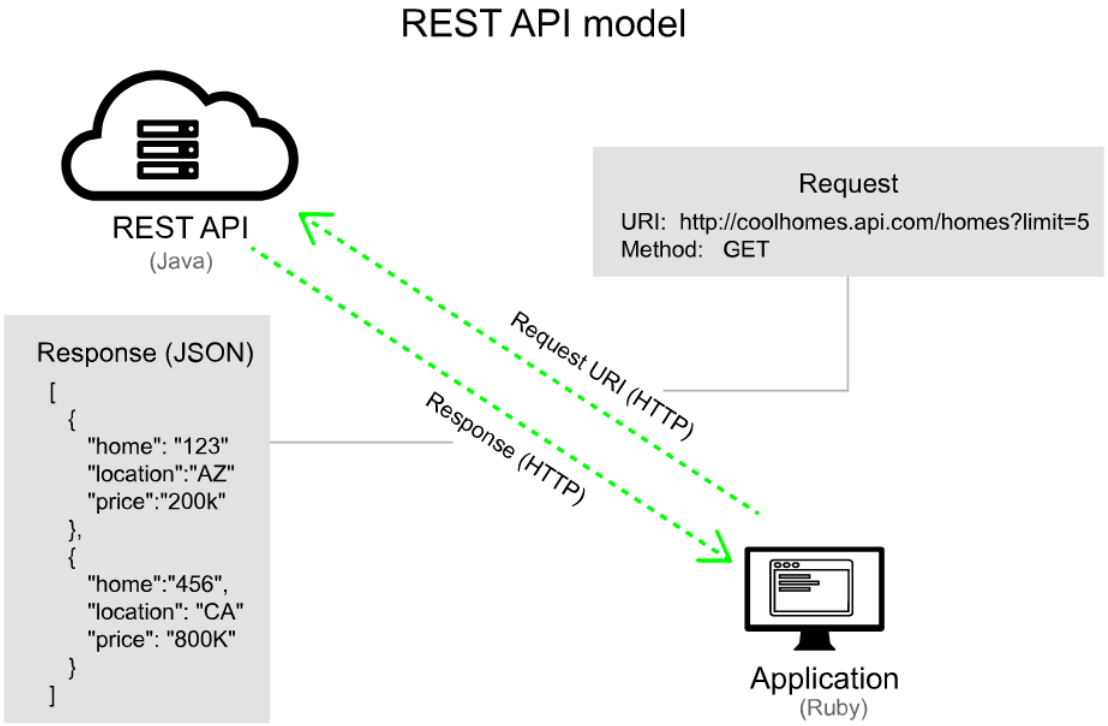

Figure 1.27: Structure of a REST API

1.7.6 Jsoup

Jsoup is an open-source library for Java that allows the downloading of a web-page to the memory and the navigation through its document object model (DOM). The DOM gets converted to a Java object which is then navigable and queryable. Written in 2009 by Jonathan Hedley, a developer at Amazon, Jsoup is available under the MIT-license.

The downloading of web-pages by software is also known as web scraping, web data extraction or web harvesting and is mainly used by bots and web-crawlers. By having the tree-like structure in memory, the data can be analysed, parsed, searched, modified

and so on.
cURL is a well known example of a web crawler that downloads a page by URL.

1.7.7 Light curve fetching

As seen in figure 1.28, multiple steps are necessary to download the brightness values of a star.
The first step is to find the resource location of the txt file that contains the brightness data. To do this, the epic number and the campaign in which the star was observed must be given to query the Harvard archive for K2 photometry. Since the client does not know the campaign in which the required star was observed, the server uses the NASA exoplanet archive to fetch the campaign number and then goes on to query the K2 photometry archive. The result is a page which contains all kinds of data for that star, but only the raw brightness data is needed. To extract the download link for the txt file, Java Jsoup is used and the result is then sent back to the client.

The second step is to use the URL received from the server and download the actual raw values. This can be done by simply calling a GET request to that link. When the client has downloaded the data to the memory it can be further used by charts, analysis and so on.

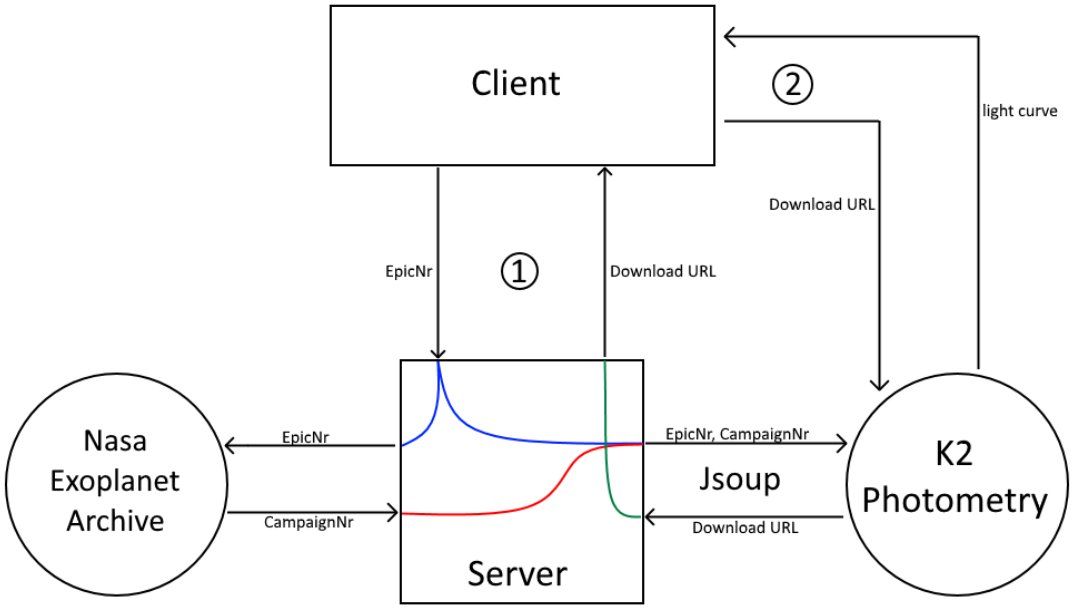

Figure 1.28: Steps of downloading a light curve

1.8 Website

1.8.1 Chartist

Important for the development of the website was to be able to draw the light-curves of the different stars and additionally to plot other statistical data like the distribution of exoplanets by their radius and orbital period, or by their finding method. The need to quickly and easily create good looking charts which are simple to configure, style and animate, is a perfect fit for the use of Chartist.

Chartist is a simple responsive charting library that allows the developer or the designer to easily create powerful charts of different types. These charts are very efficient, straightforward to configure in the JavaScript code and perfect to style with JavaScript or CSS.

A big advantage of Chartist is that it is built with SVG which, compared to other chartist libraries allows it to work natively with the browser. Another big plus point for Chartist is how easy it is to animate the charts within the JavaScript code, which gives the developer countless new options of how to present the data.

Chartist has a great community, a lot of examples and documentation online which makes it fast and easy to use. There are also some great plugins like tooltips, point labeling, axis titles, zoom and further amazing animations. Chartist also provides many different types of charts like line, scattered line, bar and pie charts which can be modified in many ways to create different and new chart types. In Fig 1.29 a very simple line chart can be seen.

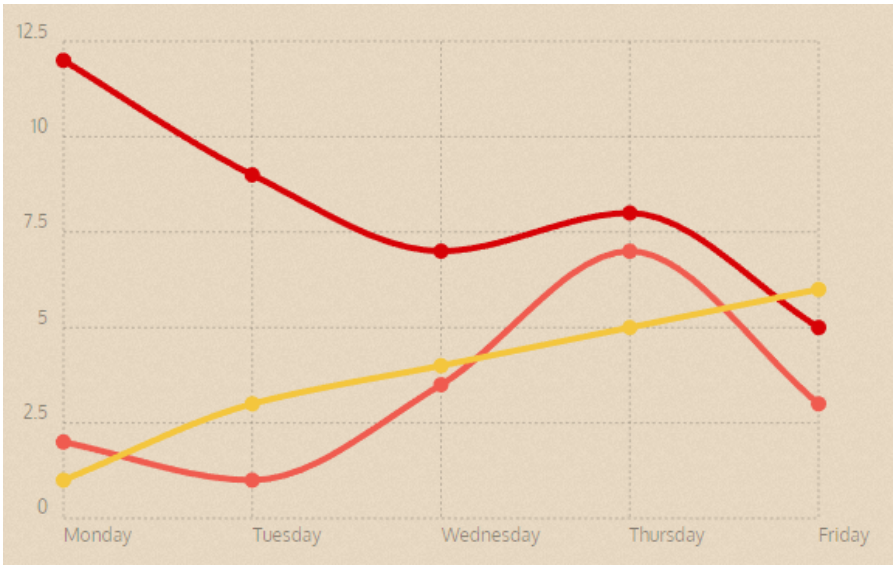

Figure 1.29: A simple line chart with Chartis.

Chapter 2

Practical implementation

2.1 System architecture

This section deals with the general system architecture of this project. There are six major parts that make up the whole project:

1. Server
2. Database
3. Desktop client
4. Time Series Analysis
5. Light-curve classifier(CNN)
6. Website

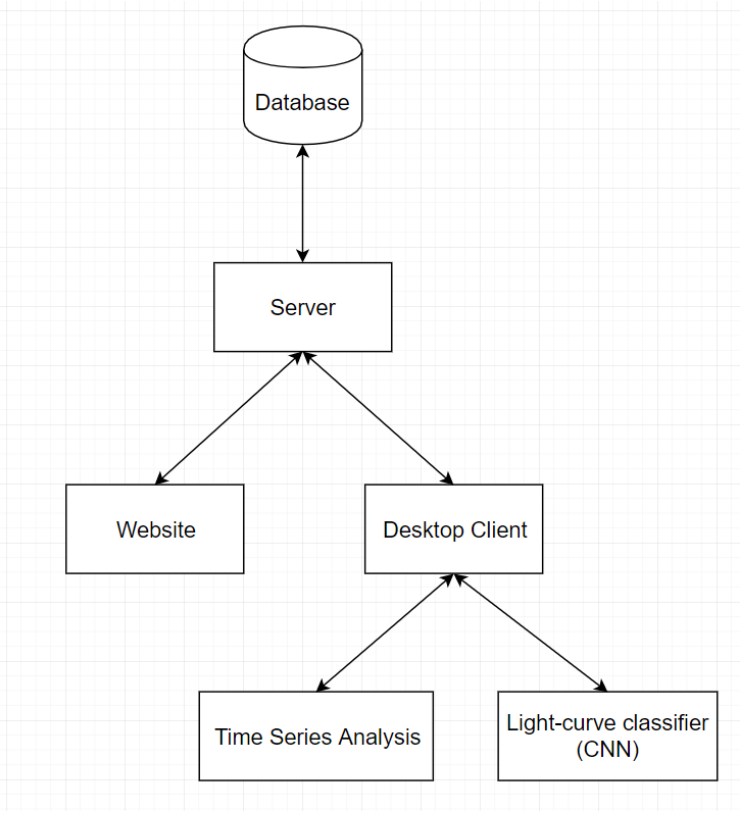

Figure 2.1: Diagram of the system architecture.

2.1.1 Desktop Client

The desktop client is the core part of the diploma thesis. It allows the user to investigate light-curves with modern technologies and to search for indicators for the existence of planets outside our solar system. The client consists of three subparts: the client itself which is the graphical interface for the user and two background processes which handle the main logic of the project, the Time Series Analysis and the light-curve classifier. The user interface was written in Electron which makes it a multiplatform client able to run on multiple operation systems. Furthermore the desktop client gives the user the ability to search for exoplanets in manual mode or batch/autonomous mode.

In manual mode the user can select a star by him or herself and start the analysis of the light-curve. In the batch mode the client communicates with the server which tells the client which light-curves are best to analyse. The batch mode can be seen as a kind of data mining mode in which the user has many options of how the process will run, for example how long, if results will be shown or if the light-curves of the stars should be displayed in a chart. After the batch mode has finished, the user gets a summary of all the analysed stars and the analysed data is sent to the server.

A very significant functionality of the system is the ability to take user feedback and learn from it. After every analysis the user has the ability to give feedback if he or she thinks that the results are wrong or contain an error. This feedback will be sent to the server to train the light-curve classifier and to create a newer and better version over time which will be automatically distributed to every client.

2.1.1.1 Light-curve classifier

The light-curve classifier is a trained convolutional neural network which is responsible for classifying the light-curve by determining a value which describes the probability for the existence of one or more exoplanets orbiting a certain star.

The deep neural network was implemented using Python, Keras and TensorFlow. With the help of the user and the server collecting feedback, the accuracy of the convolutional neural network will increase over time by learning from the new data. The current "intelligence" of the deep neural network is serialized on a simple text file which makes it very easy to replace if an updated version of the classifier is available.

2.1.1.2 Time Series Analysis

If the light-curve classifier decides that there is a certain probability that a star has orbiting planets the light-curve will be further analysed by the Time Series Analysis. Like explained in Section 1.4 "Time Series Analysis", a Time Series Analysis examines a time series in this case the light-curve to better understand the underlying structures or to extract crucial information. Here the analysis tries to find evidence which points

to the existence of an exoplanet by searching for patterns in the light-curve.

Simply put, the light curve analysis searches for dips in the light curve, groups matching dips and afterwards calculates the parameters of the found candidates. Important to know is that the Time Series Analysis was also written in Python and gets called by the desktop client.

2.1.2 Server

The server is written in JavaEE and is deployed on a Wildfly application server. The server is responsible for managing a rest interface which gives access to all collected evaluations of the clients.

It also keeps track of all the incoming new evaluations by persisting them to a Maria database. Every newly analysed star will be grouped with all analysis of the same star and matched with all confirmed exoplanets and candidates in the NASA database. This task is called the evaluation candidate check.

The server is also responsible to fetch data from the NASA Open API, other databases and APIs which have important astronomical data that is needed for the search for exoplanets. Furthermore the server handles all the user feedback which helps advancing the light-curve classifier and trains the deep neural network over time. Additionally the server helps to distribute the newest model to every client. Last but not least the server keeps track of all confirmed exoplanets to provide up-to-date statistics for the website and the rest endpoints.

2.1.3 Website

The Website is the place where all evaluations made by any user can by seen. With the help of a list view with many filter options the user has access to all evaluations he or she wants to view. By clicking on a single evaluation the user opens a detail page which shows the light-curve of the current star and all the information that can be found in the system like the number of candidates or the number of already confirmed exoplanets that are associated with the evaluation.

The website is also an important tool to introduce new users to the Exohunter project and to give them the opportunity to download the desktop client which helps them to search for exoplanets. Furthermore the website has a lot of useful information concerning the topic exoplanets.

Additionally the website has many up-to-date statistics like a bar chart showing the number of found exoplanets per year, pie charts showing the distribution of exoplanet finding methods and many others. The used data is provided by the Server which keeps track of the data with the help of NASA's Open API.

2.2 Diagrams

2.2.1 Entity relationship diagram(ERD)

An essential part of this project is the entity relationship model which can be seen in Fig 2.2. It shows the entities that are used in the database to persist the results of the analysis, classifications, the user feedback, the users, exoplanet statistics and a lot more. Therefore it is crucial to go in depth and explain what each of the entities represents and what they are used for.

2.2.1.1 LAEvaluation

The entity LAEvaluation consists of many LACandidates and represents the report of a single analysis. It holds the timestamp of when the analysis was made, which user performed the analysis, all results that were uncovered by the analysis and the classification result of the light-curve classification.

2.2.1.2 LACandidate and LACandidate_units

The entity LACandidate is the most important in the whole system. It represents all the information and evidence for the existence of a single exoplanet. Within this entity, all the calculated parameters like the orbital velocity, period, semi-major axis, equilibrium temperature and planet radius(with error 1 and 2) are saved.

All the discovered dips that belong to a candidate are saved in the entity LADip. Furthermore the average delta flux, full time and total time of all dips are stored in LACandidate. To track the unit in which the candidate's properties are stored they are saved in a HashMap, for which JPA creates an extra table called LACandidate_units. The description and calculation of the candidate's parameters can be found in Section 1.1.5.2.3.

2.2.1.3 K2LACandidateMatchingResult

A major part in searching for exoplanets is to check if someone else has already found evidence for the existence of an exoplanet or the exoplanet has already been confirmed. Therefore the entity K2LACandidateMatchingResult was created to check if a candidate that was analysed by the Exohunter system matches with any record on NASA's database. By re-discovering a candidate(evidence for an exoplanet that has not yet been confirmed yet) the newly found data could push it further to become a new confirmed exoplanet.

2.2.1.4 LADip

The entity LADip represents a single dip that is extracted by the Time series analysis from the light-curve. The important parts of a dip are the start point and end point on

the x-axis of the light curve, the delta flux value, full time and total time which were discussed in Section 1.1.5.2.3. The flux and x values that belong to the dips are stored in the entity LADataPoint.

2.2.2 LADataPoint

The entity LADataPoint is important for saving the flux, x and flag values of a single data point in the light-curve. It is important to store this data to recreate the dip and draw it into a chart or use it to calculate properties, it is important to store this data.

2.2.2.1 Star and Star_units

The star is a central entity in the system. The star's data is important for the calculation of every single planetary parameter. Therefore the data of every star that was ever analysed also remains on the database. To calculate all parameters of an exoplanet, the distance, effective temperature, mass, radius, radius error 1 and radius error 2 of a star are needed. Furthermore the star's epic number and the star's K2 campaign get stored in the database. To track the unit in which the star's properties are saved they are saved in a HashMap, for which JPA creates an extra table called Star_units.

2.2.2.2 User

The user entity is a very simple entity which only saves the id and the username to know which user analysed what and to keep track of who gave which UserReport.

2.2.2.3 UserReport

The UserReport entity handles the user's feedback which trains the light-curve classifier over time. The entity holds the epic number of the star that was wrongly classified according to the user, the classification of the neural network and the classification the user would give the light-curve. Also a timestamp is saved, a variable 'used' which tracks if the feedback was already used to advance the neural network and a reference to the user to know which user made the report.

2.2.2.4 ExoplanetStat and ExoplanetStatRow

When the website or another client tries to get exoplanet statistics via the rest endpoint it would make no sense to fetch this data from the NASA Open API. Therefore the ExoplanetStat and ExoplanetStatRow entities were created, to hold the necessary information(in which year(year), how many exoplanets(count) where confirmed with which method(method)). This data helps to create the statistic shown on the front page of the website.

To keep the data up-to-date a scheduled task runs on the server and updates the data in the database if something changes.

2.2.2.5 CnnVersion

The entity CnnVersion is a very simple datatype which basically stores the version of the light-curve classifier, the timestamp of creation and the file of the neural network model. With the help of this entity the system is able to determine what is the latest model and can tell the clients if they need to update their neural network model.

2.2.2.6 TrainData and TrainData_seq

To be able to store the training and testing data it was important to create an entity that could hold the epic number of the star, the classification result and a list of float values to hold the light-curve.

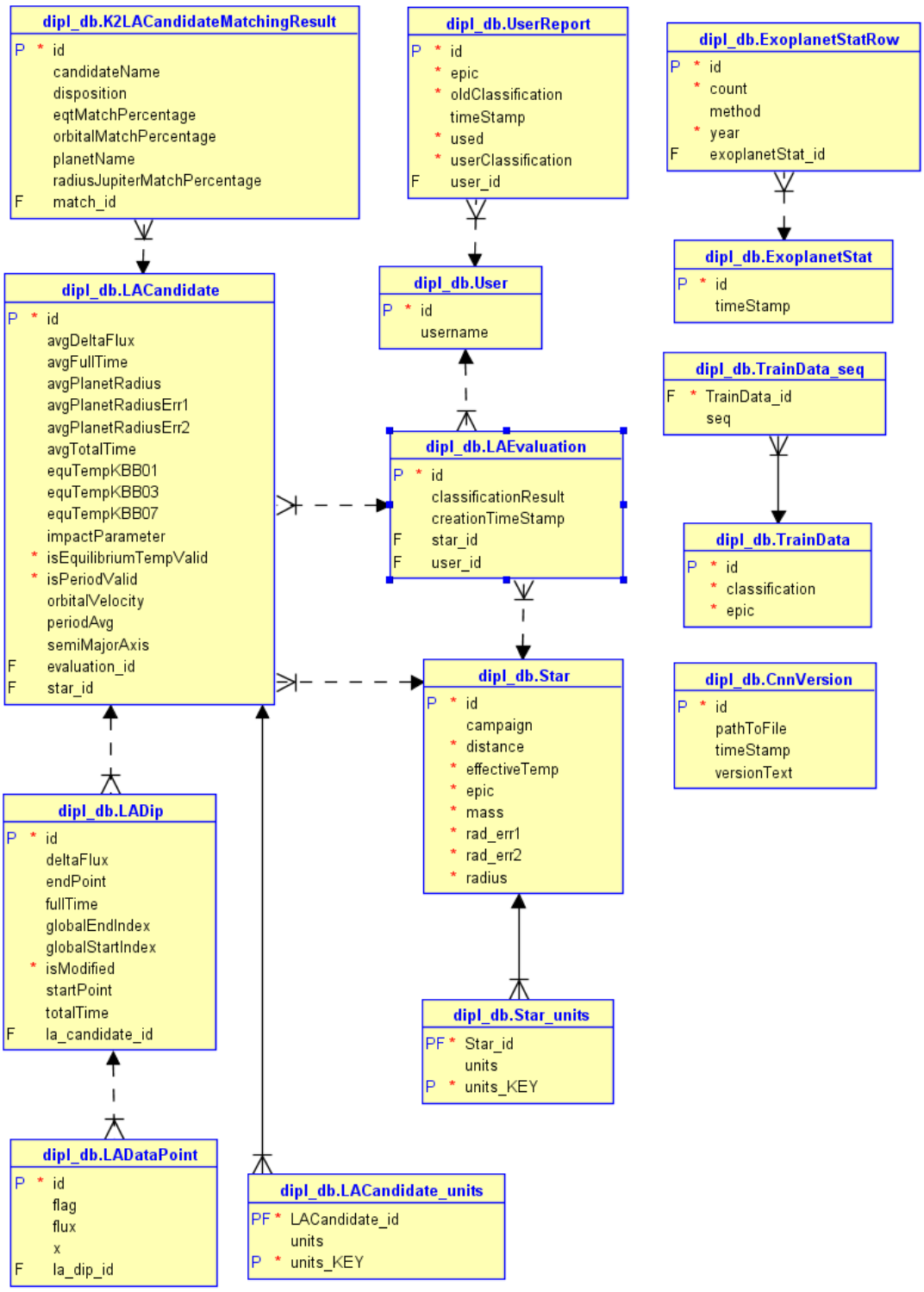

Figure 2.2: Diagram of the system's entity relationship diagram.

2.2.3 Userstories

2.2.3.1 Electron client

The main components of the electron client can be seen in figure 2.3. A more detailed explanation can be found in section 2.4.

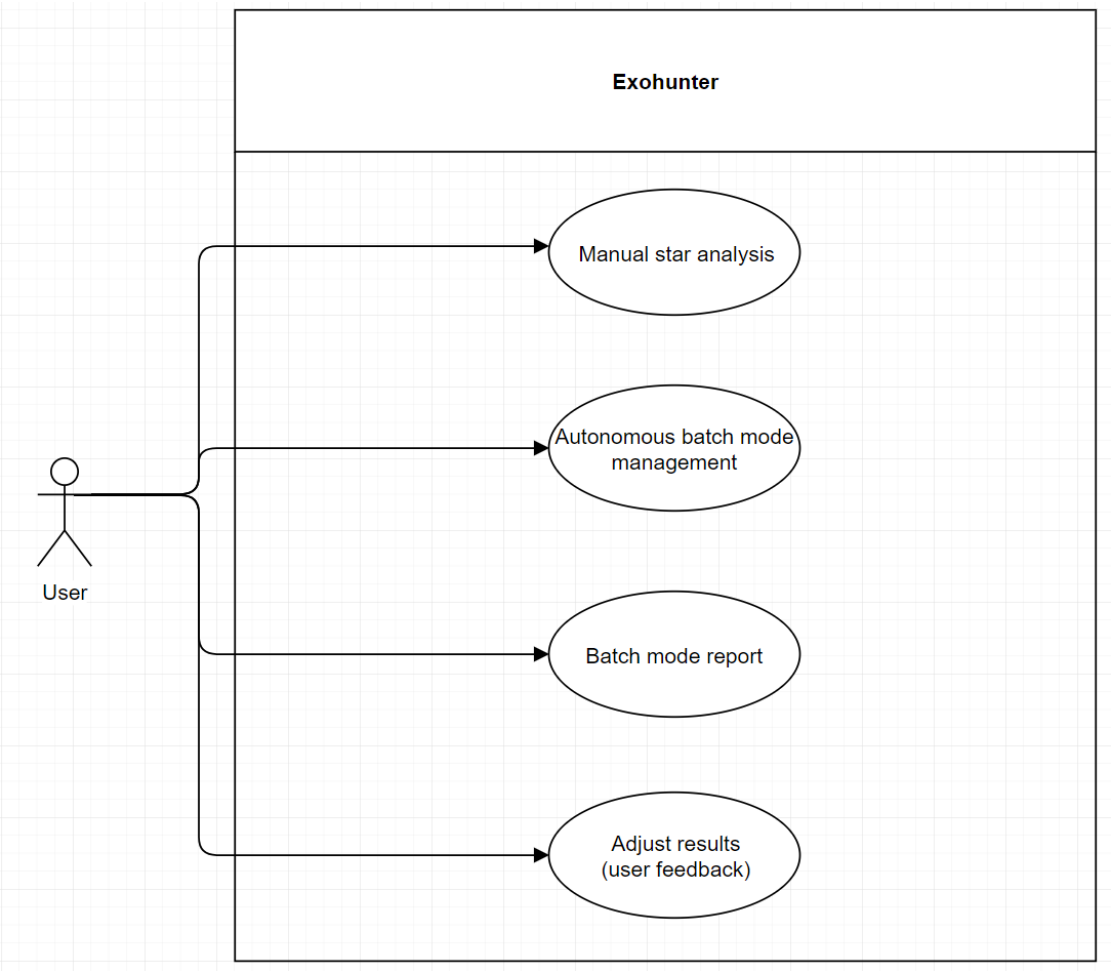

Figure 2.3: UCD of the electron client.

2.2.3.2 Website

The most important userstories of the Website can be seen in Fig. 2.4. Every single userstory in this figure is described in section 2.7 "Implementation of the website".

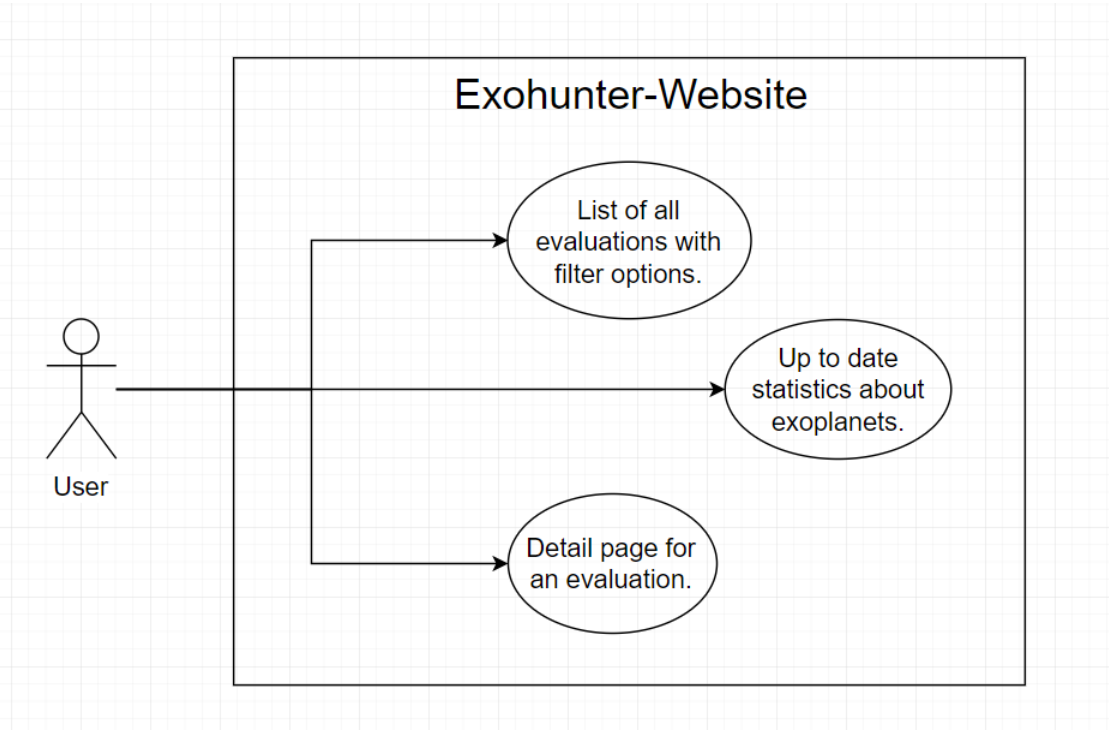

Figure 2.4: UCD of the website.

2.3 Implementation of the light-curve classifier

2.3.1 General

The light-curve classifier plays an essential role in this project. Its purpose is to decide if a certain light-curve shows evidence for the existence of an exoplanet orbiting a star. Therefore the core of the light-curve classifier is a Convolutional Neural Network which was trained with the help of pre-analysed light-curves to classify every light-curve a user wants to examine.

The whole system also works together to further advance the Neural Network by collecting user feedback, which the user can give if he or she thinks that the classifier outputs a wrong result. This feedback is then used by the server to train the Neural Network to achieve a higher accuracy than before.

A very important question when discussing the light-curve classifier is, why does the system need a Neural Network to do the work? The answer is very simple. Although humanity has up to this point never created a super intelligence that could outperform humans in every possible domain, Neural Networks of today are capable of achieving the same and even higher accuracy in single domain problems than humans. Such problems include for example image recognition, speech recognition and pattern recognition.

Another big advantage of Neural Networks compared to humans is the speed in which they classify a single data set. Therefore, the use of an Neural Networks seems to be a perfect fit, to accelerate the search for exoplanets by automating the task and to make it easier for the user to search for these mysterious objects.

The light-curve classifier consists of 4 different parts(see Fig 2.5 for the first three parts). Each of them plays an important role for the light-curve classifier. The first one is called the data preparation part. Here the light-curves, that were previously classified by a human, get separated in a training data and testing data set. After that each of the data sets will be used for training and testing the Convolutional Neural Network. The preparation of the data contains many methods that were introduced in the Time Series Analysis section. The second part is called the training section of the Neural Network. Here the training data is fed to the Convolutional Neural Network to let it learn how to solve the problem if there is an exoplanet orbiting a star. The third part is called the testing section, of the Neural Network. In this section, which is normally performed after the training section, the newly trained model will be tested. By doing so the prepared testing data is used to determine the accuracy and loss of the Convolutional Neural Network. The last part is the direct usage of the classifier within the desktop client to classify a light-curve which the user wants to examine. Each of these sections, the structure of the Convolutional Neural Network and the structural information of the data later fed to the Neural Network, will be discussed in the next few sections of this chapter.

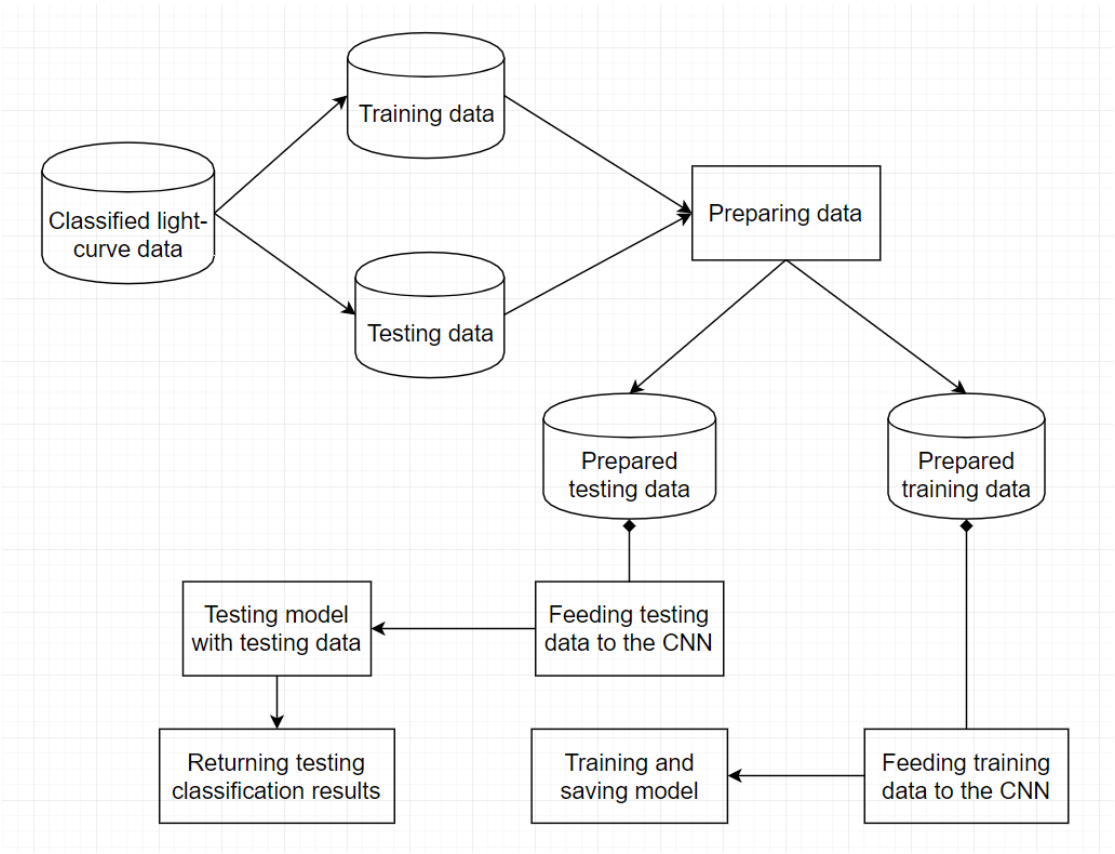

Figure 2.5: Data preparation, training and testing the model.

2.3.2 Data

The main core of the data consists of pre-classified light-curves from a website called Kaggle and all currently confirmed exoplanets. Furthermore some very old light-curves from the first few K2 campaigns are included which do not show any indicator for an exoplanet.

Important for a dataset to be used to train and test a Neural Network with is the size, the distribution and the quantity. Quantity in this case means that a light-curve with a confirmed exoplanet really does have an exoplanet orbiting the light-curve's star and distribution means that the percentage between all data groups is equal. To achieve an equal distribution the dataset itself can be equally distributed or the function preparing the single batches that are fed to the Neural Network creates a kind of equal distribution of the data classes.

The structure of every row in the dataset follows the same pattern:
201287802;1;0.991901569;0.996373881;0.99578215;0.983168331;0.983547719;0.984678149;0.984890406;...

The first number is the epic number of the star, followed by the classification result. 0 stands for no exoplanet and 1 stands for confirmed exoplanet. The remaining elements of the row are the light-curve as a sequence of flux values. The time-axis is not important for the classification and can be omitted.

2.3.3 Preparing data

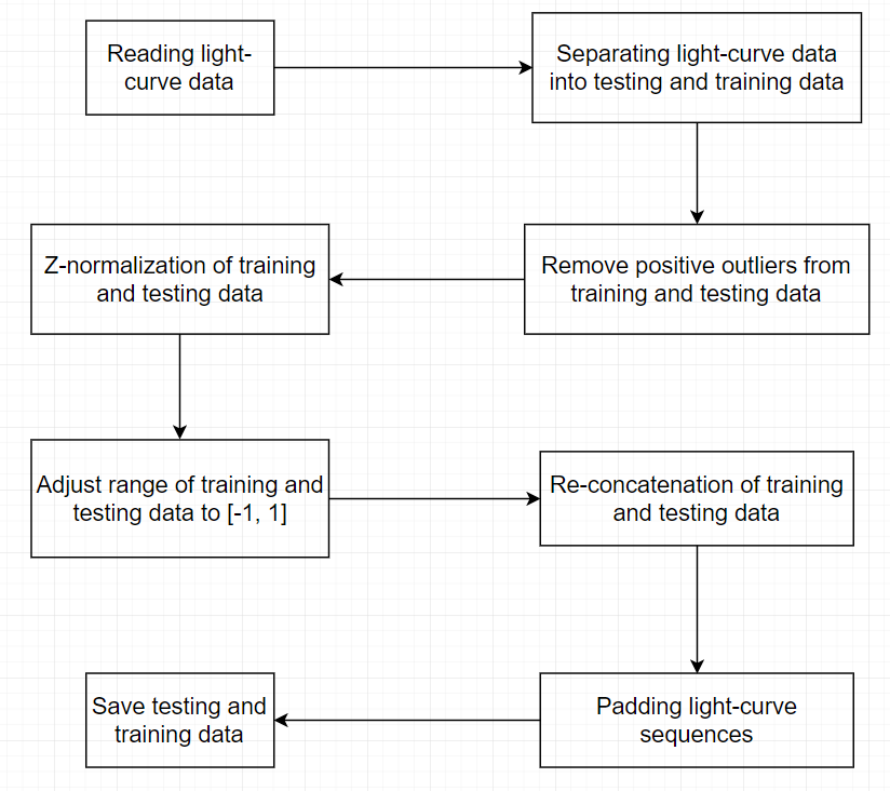

Figure 2.6: Each step of the data preparation section for the CNN.

The preparation of the collected data in order for it to be fed to the Neural Network afterwards is an essential task to guarantee the success of the light-curve classifier. As discussed in section 1.3.5 "Importance of data", a Neural Network can only work as well as the data it is learning from. Therefore the preparation of the data does not only mean separating the previously collected data into testing and training data. A lot has to be done before feeding the raw data to the network like normalizing and padding the single data rows. Each of these steps can be seen in Fig. 2.6.

2.3.3.1 Reading light-curve data

The first thing to do is to simply read all files that contain the raw data and merge them into a single list.

2.3.3.2 Seperating light-curve data

After that all the data is separated into two parts: the training data set, which will later be used to train the Convolutional network with, and the testing data, which helps the trained model to test how it performs on data it has never seen before.

2.3.3.3 Remove positive outliers

As discussed in section 1.4 "Time Series Analysis", the light-curve can contain unnecessary information that could obstruct the classification process. If that is the case, data points that have a higher flux value then the local average flux values and standard deviation should be removed and normalized to clean the data.

This is done with the help of the remove_positive_outliers(y) method which can be seen in Fig 2.7. The method was implemented in Python. The first parameter y is the array or sequence of float values that should be normalized with the function. At first the count of elements belonging to the local mean is defined called padding_size=10. Then the sequence is iterated. For every element the local mean and local standard deviation is calculated. If the current element minus the local mean is greater than the local standard deviation it will be normalized with the local mean value. At the end the normalized array is returned.

```
def remove_positive_outliers(y):
    padding_size = 10
    for i in range(0, len(y)):

        b_start = i - padding_size if i - padding_size >= 0 else 0
        b_end = i
        a_start = i + 1
        a_end = i + padding_size if i - padding_size <= len(y) else len(y)

        mean = np.mean(list(y[b_start:b_end]) + list(y[a_start:a_end]))
        std = np.std(list(y[b_start:b_end]) + list(y[a_start:a_end]))

        if (y[i] - mean) >= std:
            y[i] = mean
    return y
```

Figure 2.7: Python implementation of positive outlier detection.

2.3.3.4 Z-normalization

The next step is to apply the Z-normalization to each light-curve. The Z-normalization is, as previously discussed in section 1.4 "Time Series Analysis", a method to subtract the global mean of the light-curve from every single flux value and divide it by the global standard deviation. This is done to help the Neural Network focus on the structural information rather than on the flux values.

Fig. 2.8 shows a very simple implementation of Z-normalization in python. This method is used to apply the Z-normalization on a single light-curve. The parameter lc stands for light-curve, this list of floats is then iterated implicitly by python and every value of lc is subtracted with the mean flux value and divided by the standard deviation of the flux values in lc.

```
def z_normalization(lc):
    return (lc - np.nanmean(lc)) / np.nanstd(lc)
```

Figure 2.8: Python implementation of Z-normalization.

2.3.3.5 Adust range

After focusing on the structural information with the Z-normalization it is also important to reduce the range of values by rescaling the light-curve to the range [-1, 1]. In Fig. 2.9 the method normalize_nom(nom, min_nom, max_nom) rescales the current number between [-1, 1] with the help of the min. and max. value of the light-curve. After the method definition the method is called by applying it to every single element of arr.

```
def normalize_nom(nom, min_nom, max_nom):
    return 2*(nom - min_nom) / (max_nom - min_nom) - 1

min_nom = np.min(arr)
max_nom = np.max(arr)
arr = [normalize_nom(nom, min_nom, max_nom) for nom in arr]
```

Figure 2.9: Adust range to [-1, 1].

2.3.3.6 Re-concatenation and padding

To increase the quantity of the light-curve data and to get a fixed size to feed the data into the Neural Network it is important to pad the data until it has the right length. There are different approaches on how to accomplish this, such as filling up the array with null values until it has the right size. Another way would be to attach the sequence

to itself only reversed until the array has the correct length.

Fig. 2.10 shows the implementation of the pad_sequence(seq, cur_len) method which tries to pad the sequence with itself until the MAX_PADDING size(a fixed size for all input data) is reached. The variable 'switch' keeps track of how the sequence should be appended onto itself(reversed or not). Depending on the value of switch the maximal possible number of elements of the sequence will be appended to the sequence to reach the correct length.

```
def pad_sequence(seq, cur_len):
    switch = True
    while(cur_len < MAX_PADDING):
        diff_len = MAX_PADDING - cur_len

        if(switch):
            if cur_len - diff_len < 0:
                diff_len = cur_len
            test = seq[cur_len - diff_len:cur_len]
            test = test[::-1]

            seq[cur_len:cur_len+diff_len] = test
        else:
            if diff_len > cur_len:
                diff_len = cur_len
            test = seq[0:diff_len]

            seq[cur_len:cur_len+diff_len] = test

        cur_len += diff_len
        switch = not switch
    return seq
```

Figure 2.10: Padding light-curve.

2.3.3.7 Saving testing and training data

When the raw data is separated into a training and testing set and both have been prepared, the two sets can be saved in two different .csv files.

2.3.4 The Convolutional Neural Network

Like discussed in Section 1.3.2 "Convolutional Neural Networks" there are many different tools and layers that can be used when creating a Neural Network, especially a

Convolutional Neural Network. Furthermore there are many different ways to structure the Neural Network and the designing process of creating a good solution often needs many tests to determine the accuracy and efficiency of a model's structure. Fig. 2.11 shows the structure of the Neural Network used in this project.

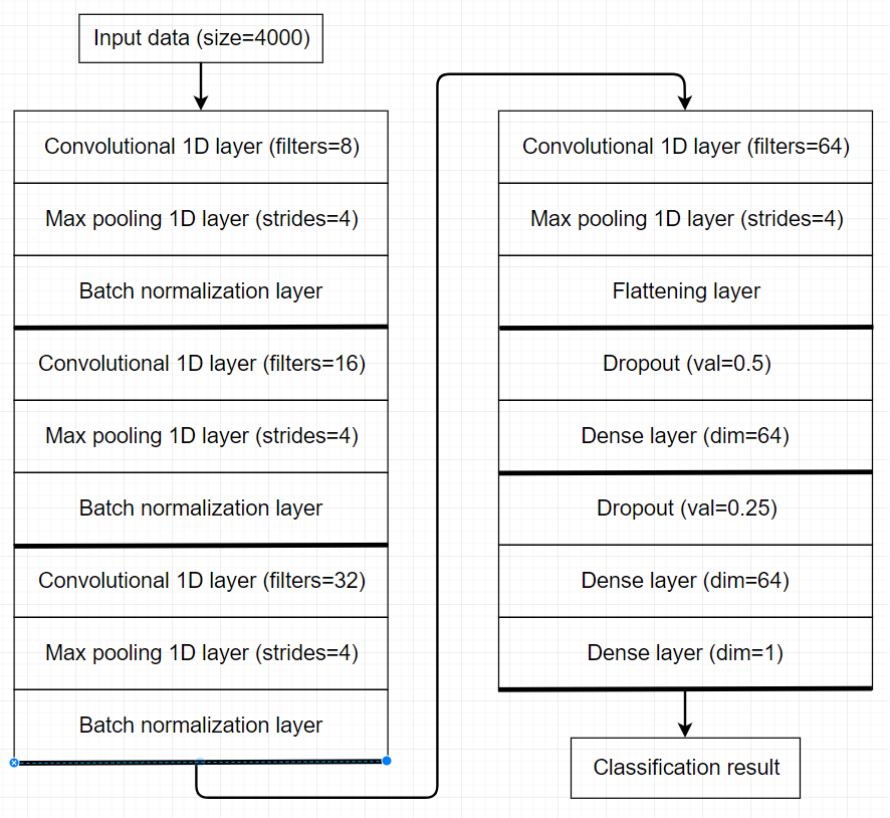

Figure 2.11: Structure of the CNN.

First the Convolutional neural is fed with a vector with dimensions 1x4000. 4000 is the maximal size of the input vector and like previously mentioned it is the number every light-curve is padded to. After that follow a convolutional layer, a max pooling and a batch normalization.(The different layers and methods used here were already described in section 1.3.2 "Convolutional Neural Networks"). Basically the convolutional layer subdivides the input into smaller parts and searches for important features to create new data out of these features. The max pooling reduces the data without loosing important information and the batch normalization is used for faster learning and a higher overall learning rate.

These three parts are repeated three times, every time with slightly different parameters to filter the important data from layer to layer. The fourth major part starts with a convolutional layer followed by a max pooling, but instead of a batch normalization a flattening layer is used to convert the data created by the convolutions into a format that is used by the fully connected layer. (see Fig. 1.12) After that the fully connected layer consists of a dropout which mimics the memory loss of some of the information in our brain and is primarily used to prevent the Neural Network from over-training and a dense layer which is the standard layer of a Fully Connected Neural Network with weights, biases and activation functions. These two parts are repeated twice before the current result of the Convolutional Neural Network will be transformed with the help of another dense layer and the sigmoid activation function to a single number. This number is the final result of the Neural Network, the probability of a star being orbited by an exoplanet.

The Fig. 2.12 shows the Python and Keras implementation of the previously described model. The train_x parameter is the single light-curve vector to feed the CNN with.

2.3.5 Training the Neural Network

Training the network is the most important part of every Neural Network. Because Neural Networks belong to the supervised machine learning category they are not able to solve the problem they are created for without proper training. Therefore, training the network is essential for the success of the model.

The basic steps of a training process can be seen in Fig. 2.5. The training takes the previously prepared training data set and feeds every single row of data to the network. If the output of the Neural Network and the result of the training data do not match the weights of the Neural Network will be updated. The speed and method of how the correcting step will be executed depends on the learning rate and the optimizer.(1.3.1.2 "Learning Neural Networks")

Often there are multiple learning steps with different learning rates to pre-train the network first, creating a basic knowledge of problem solving and then further advancing the process in more detail. Fig. 2.13 shows the implementation of the train() method.

First the method load_and_proprocess_train_test() loads the training and testing data set. The result of the function will be saved in four lists, the train_x which holds the light-curve sequences, train_y which holds the classification results for training, test_x and test_y. Then the model of the Convolutional Neural Network will be defined.

The training process consists of two steps, first the Convolutional Neural Network will be trained with the Adam-optimizer and a learning rate of 10^{-5} for 10 epochs. After that the network will be trained again with the Adam-optimizer and a learning rate of 4×10^{-5} for 75 epochs. Every epoch, a batch of 32 data rows will be randomly selected

```python
def create_model(train_x):
    model = Sequential()
    # first conv. step
    model.add(Conv1D(filters=8,
    kernel_size=11, activation='relu',
    input_shape=train_x.shape[1:],
    kernel_initializer='random_uniform', padding='valid'))
    model.add(MaxPool1D(strides=4))
    model.add(BatchNormalization())
    # second conv. step
    model.add(Conv1D(filters=16,
    kernel_size=11, activation='relu',
    kernel_initializer='random_uniform', padding='valid'))
    model.add(MaxPool1D(strides=4))
    model.add(BatchNormalization())
    # thrid conv. step
    model.add(Conv1D(filters=32, kernel_size=11,
    activation='relu', kernel_initializer='random_uniform',
    padding='valid'))
    model.add(MaxPool1D(strides=4))
    model.add(BatchNormalization())
    # last conv. step -> convert to fully connected layer
    model.add(Conv1D(filters=64, kernel_size=11, activation='relu',
    kernel_initializer='random_uniform',
    padding='valid'))
    model.add(MaxPool1D(strides=4))
    model.add(Flatten())
    # fully connected layer
    model.add(Dropout(0.5))
    model.add(Dense(64, activation='relu'))
    model.add(Dropout(0.25))
    model.add(Dense(64, activation='relu'))
    # convert to final CNN result
    model.add(Dense(1, activation='sigmoid'))
    return model
```

Figure 2.12: Python and Keras implementation of the model.

and slightly modified to be fed to the Neural Network. When the training is completed the current model will be saved to a file and the metrics of the training will be plotted on the screen.

2.3.6 Testing the Neural Network

The testing of a Convolutional Neural Network is a major part of ensuring that the network is accurate and efficient at solving the problem it is made for(see Fig. 2.5). With the help of the prepared testing data, a data set the Neural Network has never seen before during training the accuracy and loss of the network can be calculated.

Fig. 2.14 shows the implementation of the testing method. First the testing data gets loaded and the model gets defined. Afterwards the model tests the testing data set

```
def train():
    # load training data
    train_x, train_y, test_x, test_y = pp.load_and_preprocess_train_test()
    model = create_model(train_x)

    # first step of training
    model.compile(optimizer=Adam(1e-5), loss='binary_crossentropy', metrics=['accuracy'])
    res = model.fit_generator(pp.create_batches(train_x, train_y, 32),
                              validation_data=(test_x, test_y),
                              verbose=2, epochs=10,
                              steps_per_epoch=train_x.shape[1] // 32)

    # second step of training
    model.compile(optimizer=Adam(4e-5), loss='binary_crossentropy', metrics=['accuracy'])
    res = model.fit_generator(pp.create_batches(train_x, train_y, 32),
                              validation_data=(test_x, test_y),
                              verbose=2, epochs=75,
                              steps_per_epoch=train_x.shape[1]//32)

    model.save(current_model_to_use)

    # plot loss and accuracy of training
    plt.plot(res.history['loss'], color='r')
    plt.plot(res.history['val_loss'], color='g')
    plt.show()

    plt.plot(res.history['acc'], color='r')
    plt.plot(res.history['val_acc'], color='g')
    plt.show()
```

Figure 2.13: Training the CNN.

before the accuracy and the loss of the network will be printed on the console.

```
def test():
    train_x, train_y, test_x, test_y = pp.load_and_preprocess_train_test()
    model = load_model(current_model_to_use)

    hist = model.evaluate(train_x, train_y, batch_size=32, verbose=0)
    print('Training data\n Loss:', hist[0], ' Accuracy:', hist[1])
```

Figure 2.14: Testing the CNN.

2.3.7 Using the Neural Network

After the model is properly trained and tested the Desktop Client can call the light-curve classifier to classify a single light-curve. Before the light-curve which is provided by the desktop client can be fed to the Convolutional Neural Network it has to run through the same preparation process the training and testing data were normalized with. The preparing process of the single light-curve can be seen in Fig 2.15. Afterwards the pre-

pared light-curve data can be fed to the Neural Network to classify it. The result of the Neural Network will then be returned to the desktop client which will call the Time Series Analysis if the prediction exceeds a certain threshold.

Fig 2.16 shows the implementation of the test_single_light_curve(arr_y, verbose=0) method. First the model gets loaded, then the prepared light-curve is transformed into a two dimensional array before being fed to the Neural Network.

```
def pre_process_single(arr_y):
    # remove outliers
    arr_y = utils.remove_positive_outliers(arr_y)
    arr_y_len = len(arr_y)
    # Z-normalization
    help = (arr_y - np.nanmean(arr_y)) / np.nanstd(arr_y)
    # fix range
    min_nom = np.min(help)
    max_nom = np.max(help)
    arr_y = [utils.normalize_nom(nom, min_nom, max_nom) for nom in help]
    # padding the light-curve
    arr_y = fix_zero_padding(arr_y)
    arr_y = pad_sequence(arr_y, arr_y_len)
    # uniform filter
    arr_y = np.stack([arr_y, uniform_filter1d(arr_y, axis=0, size=200)], axis=1)

    return arr_y
```

Figure 2.15: Prepare single light-curve.

```
def test_single_k2_light_curve(arr_y, verbose=0):
    model = load_model(current_model_to_use)
    arr = np.empty((1, arr_y.shape[0], arr_y.shape[1]), dtype='float32')
    arr[0] = arr_y
    hist = model.predict(arr, verbose=verbose)
    return hist
```

Figure 2.16: Test single light-curve with CNN.

2.4 Implementation of the Electron client

The core features of the Electron client are the manual and batch/autonomous mode for analysing stars. The electron client provides access to these features while offering a user interface that makes the functionalities easy to use.

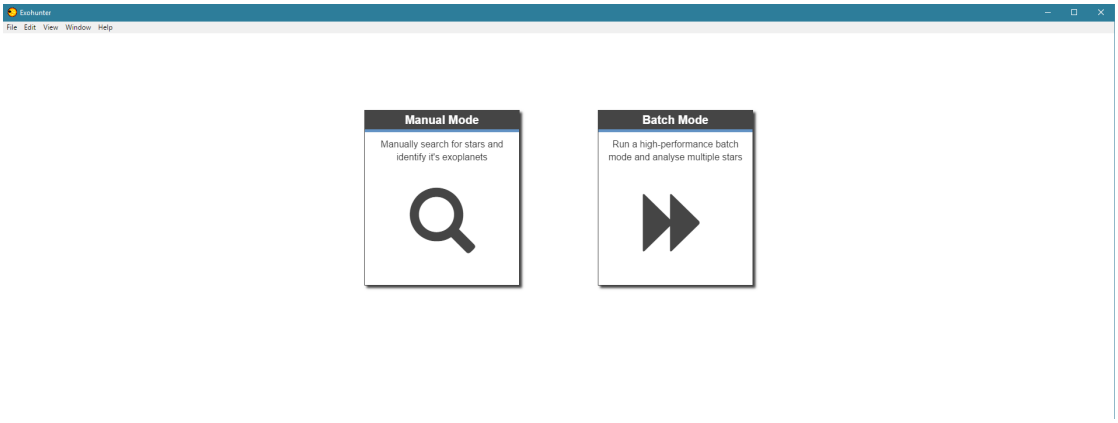

Figure 2.17: mode selection on the start screen

2.4.1 Manual mode

Manual mode allows for users to analyse stars with a known epic number or epic name. After entering the identification of a certain star, the light curve will render and the user will have the ability to resize and pan the chart.

Additionally, after searching a star, a button will appear to commence the scanning process, which then allows the classifier to take over. After the Classifier returns a probability for unusual patterns in the chart, the client decides the next steps to take. If the probability is higher than a given threshold, which means that a single or multiple exoplanets are likely, the analysis gets initiated. If the probability is lower than the

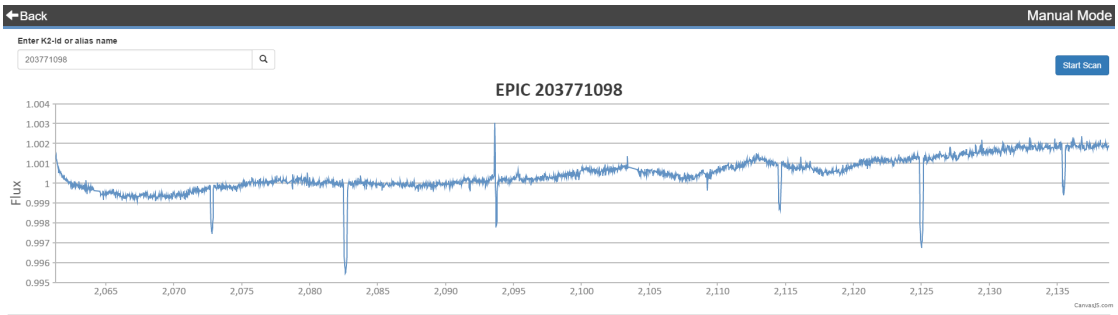

Figure 2.18: chart after entering a star id in manual mode

threshold, the chart shows most likely no anomalies and the user gets informed that nothing of interest was found.

As soon as the analysis gets initiated, all parameters and dips get calculated in the background. Once this is done, the results are rendered to the client, placed underneath the chart. Furthermore, all dips get plotted into the chart, displaying the intervals where exoplanets possibly transited the star. The first entry is the host star, displaying its mass, radius and temperature. Each candidate is then displayed under the host star, containing informations about:

- average period duration
- average delta flux
- average full/total time
- average planet radius
- semimajor axis
- orbital velocity
- estimated temperatures

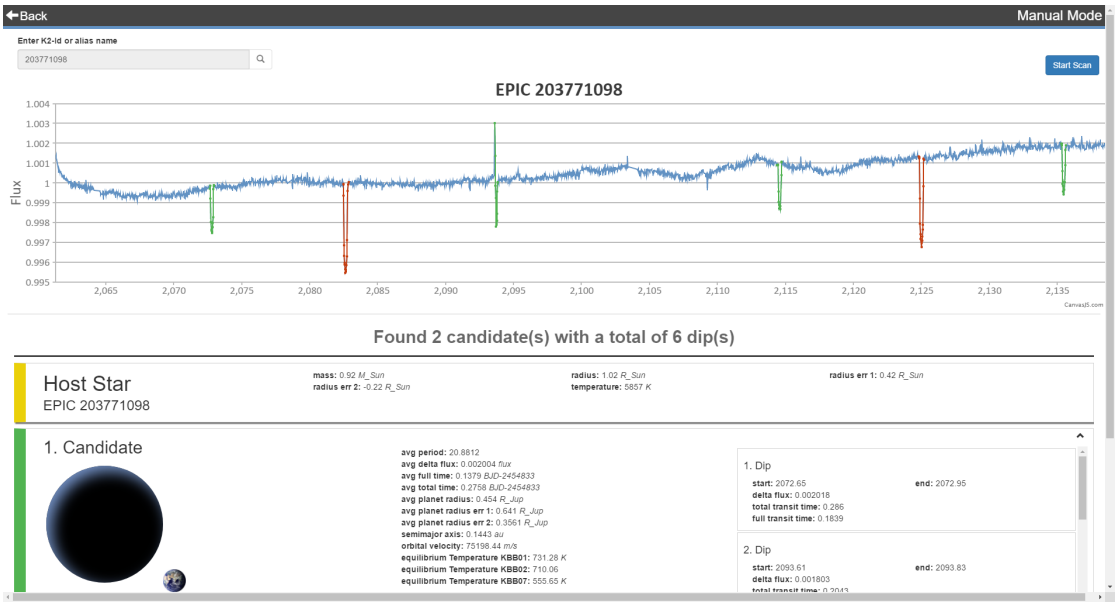

Figure 2.19: results are displayed after analysis is complete

The last step a user can take is to modify the results and notify the server about a wrong result. The new result is then taken into account when optimizing the neural network and improving its accuracy. The user interface for this process can be looked up in section 2.4.6.

2.4.2 Batch mode

The batch mode is very similar to the manual mode in terms of structure. The main difference is that the whole process gets looped while the epic numbers are distributed by the server. Between iterations there is no user interaction but as soon as the batch mode is exited, a report of all analyses is rendered to the screen. The batch mode is an autonomous process that allows users to run Exohunter in the background, for example when they are asleep. When a user checks the batch mode results, he or she may have found an exoplanet.

Before starting batch mode, the user has the ability to configure the settings for the process. There are the following options available:

- **Show charts:** Enables/Disables the display of dynamic charts during batch mode. Disabling may lower the processing costs.
- **Show progress:** Enables/Disables the display of progression in batch mode.
- **Show report:** Enables/Disables the report at the end of a batch. Disabling removes the ability to adjust entries after finishing batch mode.
- **Number of iterations:** Sets the number of iterations after which the batch mode is stopped. Leave blank to ignore the number.
- **Time limit in minutes:** Sets the time limit for the batch. Leave blank for indefinite duration. Batch mode can be manually stopped or automatically stops when either iteration limit or time limit are reached.

During batch mode, every iteration result is sent to the server, to keep track of all already analysed stars. The results contain star id(epic number), star meta data, and the result of the Time Series Analysis, which contains candidates with dips. Additionally all entries are saved in memory to serve as data for the report.

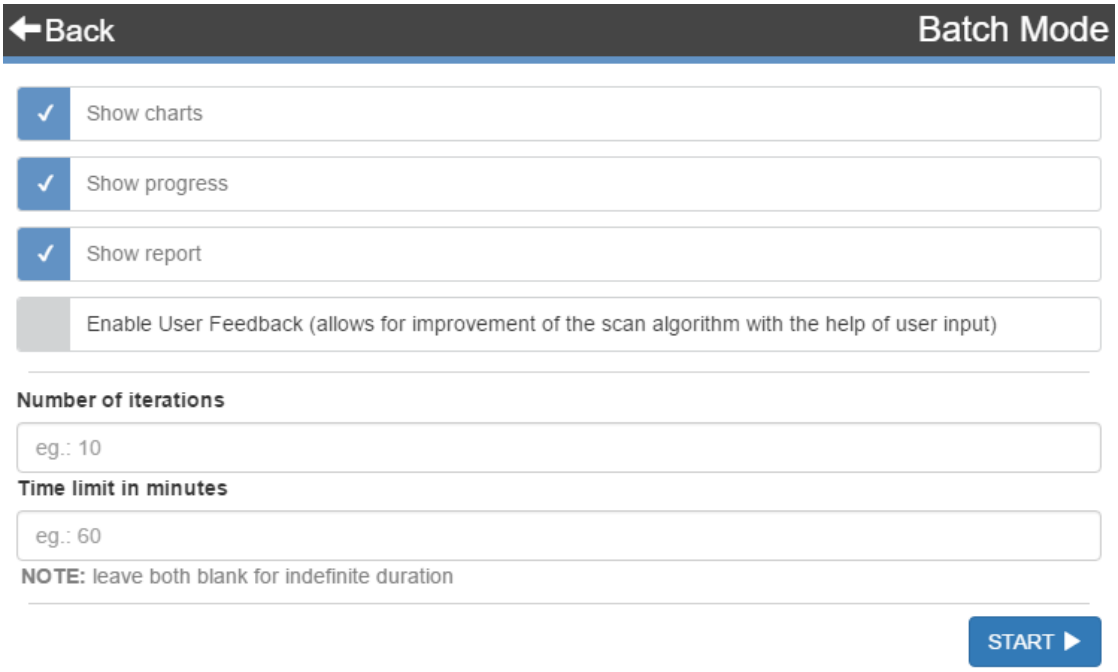

Figure 2.20: settings for the batch mode

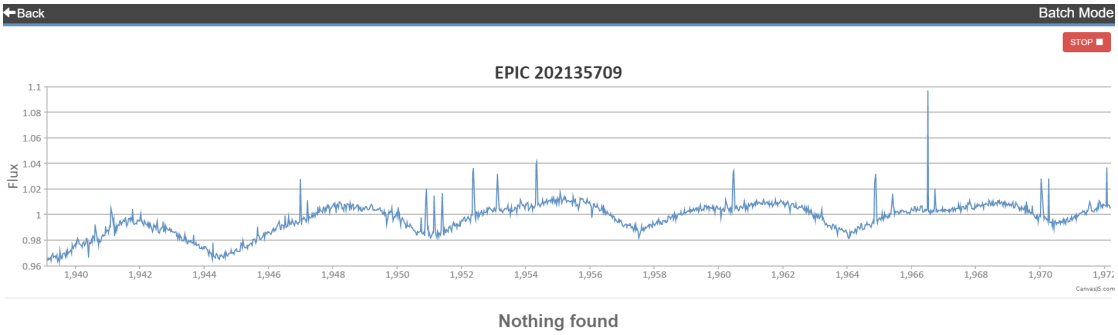

Figure 2.21: example for a batch iteration where nothing was found

The batch report is a key part of the batch mode, enabling the user to look back at all analysed stars. In figure 2.22, an example of a batch report is given. In the top right the user can export the data to a JSON-file, which contains information about each star, the cnn probability, candidates and dips. In the rendered view, an overview about the batch mode is displayed and each star is listed. The user then has the ability to adjust each result to match his desire. The adjustment interface is further explained in section 2.4.6.

s

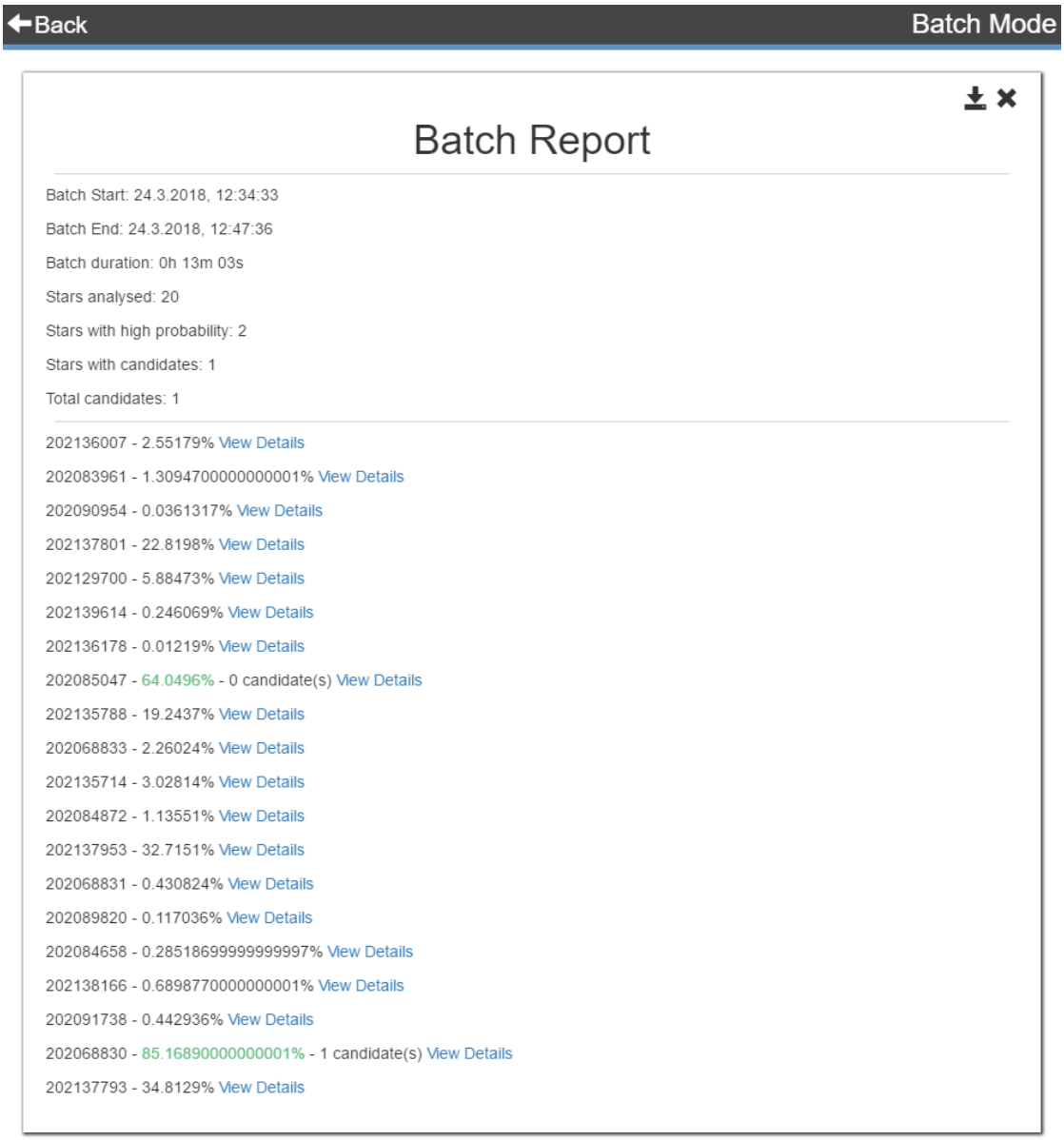

Figure 2.22: example report after finishing batch mode

2.4.3 Fetching light curves and displaying

As explained in section 1.7.7, the URL for a light curve is retrieved from the server after calling the REST endpoint with the epic number of the star which should be fetchedü. After receiving said URL, the client can simply download the raw data which contains x-values for time and y-values for star brightness(flux) as seen in figure 2.23. After receiving the raw data, the lines have to be split up and stored in an array of x and y values which is done in the *parseData* function.

```
//receives the URL to the desired star's lightcurve
exports.getLightCurveURL = function (id, callback) {
    server.sendGetRequest(SERVER_URL + "k2/fetchLightcurveURL/" + id, callback);
}

//downloads the raw data with a given URL
exports.getLightCurve = function (url, callback) {
    server.sendGetRequest(url, function (data) {
        parseData(data, callback);
    });
}
```

Figure 2.23: fetching light curves with a star id

To render the raw data, which is now stored in memory, a chart library called canvas.js, as explained in section 1.6.2, is used. Each canvas.js chart has a configuration which describes chart type, colours, shapes and most importantly the actual data which is copied from memory. In this case, a line chart with x and y values is used and the data array containing x and y points is copied into the configuration.

2.4.4 Scanning for exoplanets

In order to scan for exoplanets, a star's light curve has to be first evaluated for exoplanet probability by the CNN. If positive, the Time Series analysis get initiated, which then calculates those exoplanets in detail.

In the first step, as seen in figure 2.24, the light curve URL for a given epic number gets fetched, which is then used to download the raw data. After that, the view gets updated and the values gets split up into individual x and y arrays. The y values need to be separated from the x values, since the CNN, which is looked into in the next step, only uses y values.

```
//continue by downloading the light curve
datafetcher.getLightCurveURL(id, function (url) {
    datafetcher.getLightCurve(url, function (data) {
        //update the view
        var accuracy = display.accuracy;
        var status = display.status;

        status.innerHTML = `Scanning EPIC ${id}`;

        //prepare data
        var y = [];
        var x = [];
        for (var i = 0; i < data.length; i++) {
            x.push(data[i].x);
            y.push(data[i].y);
        }

        //continue by spawning a python process for the cnn
        createPythonProcess(cnnPath, y, function (result, process) {···
        });
    });
});
```

Figure 2.24: values are prepared for the CNN after downloading the light curve

After the CNN is finished and returned its probability value, the return value can be used to determine the next steps to take. If the accuracy is greater than a given constant threshold, the analysis continues - if the accuracy is lower, the whole process finishes, since nothing of interest was found. These steps can be examined in figure 2.25.

```
//continue by spawning a python process for the cnn
createPythonProcess(cnnPath, y, function (result, process) {
    if (result.length > 0) {
        //check if accuracy of the cnn is over a given threshold
        //if the accuracy is greater than the threshold, the time series analysis gets initiated
        if (acc > cnn_threshold) {
            skip = false;
            str = "RUNNING ANALYSIS";
        }
        //update view
        accuracy.innerHTML = `Probability: <strong>${Math.round(acc * 1000000) / 10000}%` +
            `</strong> <span class='glyphicon glyphicon-menu-right'></span> ${str}`;
        //wait 3 seconds and then continue with the next steps
        setTimeout(function () {
            if (!skip) {
                //potential exoplanets found, continue with analysis
                status.innerHTML = `Analysing EPIC ${id} for exoplanets`;
                //before starting analysis, the star meta data needs to be fetched
                datafetcher.getStarMetaData(id, function (meta) {···
                });
            } else {
                //finished, exit this function
                status.innerHTML = "Nothing found";
                finish(callback);
            }
        }, 3000);
    } else {
        finish(callback);
    }
});
```

Figure 2.25: further action is decided by the exoplanet probability

In the case of a high probability, the time series analysis is required to calculate parameters for potential exoplanets. Before starting the analysis process, the view gets updated and star meta data is fetched from the NASA API, since the analysis needs informations about the star for parameter calculation, as seen in image 2.26.

```
//before starting analysis, the star meta data needs to be fetched
datafetcher.getStarMetaData(id, function (meta) {
    //build star configuration
    var star = {
        "units": {
            "mass": "M_Sun",
            "radius": "R_Sun",
            "rad_err1": "R_Sun",
            "rad_err2": "R_Sun",
            "effectiveTemp": "K",
            "distance": "pc"
        },
        "epic": id,
        "mass": meta.k2_mass,
        "radius": meta.k2_rad,
        "rad_err1": meta.k2_raderr1,
        "rad_err2": meta.k2_raderr2,
        "effectiveTemp": meta.k2_teff
    };
    var analysisParams = { "star": star, "x": x, "y": y };

    //run analysis with paramters
    runAnalysisProcess(analysisPath, analysisParams, status, chart, data, callback);
});
```

Figure 2.26: fetching of star meta information

In figure 2.27, the analysis python process is instantiated and results are rendered to the view. Depending on the outcome of the analysis, the chart either shows occurrences of dips or not. If the scan was started during manual mode, a more detailed information is displayed than in batch mode. This is done by rendering a host star entry and underneath all entries for found candidates. Each candidate shows information about exoplanet parameters and dips.

```
//run the analysis process for a certain star and callback the result
function runAnalysisProcess(analysisPath, analysisParams, status, chart, data, callback) {
    //spawn the python process for the time series analysis
    createPythonProcess(analysisPath, analysisParams, function (analysisResult, process) {
        if (analysisResult.length > 0) {
            //parse result
            var json = utils.parseToJson(analysisResult);
            //render result
            if (displayManualMode) {
                manualResult.style.display = 'inline-block';
                utils.addStarEntry(manualResult, json.star);
            }
            var candidatecount = json.candidates.length;

            //render result and charts
            for (var i = 0; i < candidatecount; i++) {
                var candidate = json.candidates[i];
                var color = utils.getColorForCandidate(i);
                utils.addDipsToChart(chart, data, candidate.dips, color);
                if (displayManualMode) {
                    utils.addCandidateEntry(manualResult, candidate, i, color);
                }
            }
            //update status
            status.innerHTML = `Found ${json.candidates.length} candidate(s)` +
                ` with a total of ${dipcount} dip(s)`;
        }
        //exit function
        setTimeout(() => {
            finish(callback);
        }, 5000);
    });
}
```

Figure 2.27: running the analysis process

The finish function marks the end of the scan and is responsible for transmitting the results to the server. It does not matter if candidates where found in the light curve or not, as both outcomes are important for evaluation. As seen in figure 2.28, the computed results are bundled and sent via POST for further use. After this step, the scanning progress is completely finished.

```
//send computed evaluation result to server, automatically after an analysis is finished
function sendComputedEvaluationToServer(evalEntity) {
    server.sendPostRequest(server_url + "LAEvaluation/addSingleComputed", {
        userId: 1,
        star: evalEntity.star,
        candidates: evalEntity.candidates,
        id: evalEntity.id,
        classificationResult: evalEntity.accuracy * 100,
    }, function () {
        logger.log_info("Succesfully sent analysis to server");
    });
}
```

Figure 2.28: the computed evaluation gets forwarded to the server

2.4.5 Calling python processes

To call python processes, a node library called child_process is utilized as discussed in section 1.6.3. When using the library, a process is instantiated in the system environment, which can then be accessed via inter process communication (ICP). In figure 2.29 the usage of streams via stdin, stdout and stderr is illustrated. After a data stream is finished all chunks, which are collected in data events, get combined to form a string of text. When the data stream (stdin) is finished transmitting data, the text is returned via callback.

```
//create a python process with a .py path and parameters
function createPythonProcess(path, params, callback) {
    var py = spawn('python', [path]);
    var output = '';
    var error = '';

    py.stdout.on('data', function (data) {
        output += data.toString();
    })

    py.stdout.on('end', function () {
        callback(output, py);
    })

    py.stderr.on('data', (data) => {
        error += data.toString();
    });

    py.stderr.on('end', (data) => {
        logger.log("PYTHON", error);
    });

    py.stdin.write(JSON.stringify(params));
    py.stdin.end();
}
```

Figure 2.29: handling of process streams in electron

2.4.6 User input feedback

As mentioned before, user feedback is very important to improve the accuracy of the classifier, making it better and better over time. The user has the ability to edit results after analysing in manual mode and also for each entry after a batch is finished. The interface for both possible options is the same.

In said interface, as seen in figure 2.30, the user can edit, remove and add candidates, which are groups of dips that could potentially be a planet. In each candidate there is at least one occurrence of a dip but theoretically there is no limit. The user is given the ability to select a candidate and then edit its dips. When editing a candidate, the user is able to add, remove and also to shift and resize dips.

Figure 2.30: Interface for user adjustments

Custom dips are also plotted to the chart, making it easy for the user to move dips to the right place. When modifying a dip, the corresponding part of the chart also updates in realtime. To modify a dip, the user can either change the start position or the dip duration, while the end position is calculated by those two.

After the user is satisfied with the new configuration, saving it will send the custom data to the server (figure 2.31 and figure 2.32), which will then be taken into account for improvements.

```
//save adjustments of an analysis result and notify server
function saveAdjustments() {
    recalculateParams({
        star: detailEntity.star,
        candidates: detailEntity.candidates
    }, function (data) {
        //set recalculated parameters
        if ("candidates" in data) {
            detailEntity.candidates = data.candidates;
        }

        //override old entity
        entity = clone(detailEntity);

        //update manual page
        if (displayManualMode) {
            manualResult.innerHTML = "";
            utils.addStarEntry(manualResult, entity.star);
            utils.removeAllCandidatesFromChart(manualChart);
            for (var i = 0; i < entity.candidates.length; i++) {
                var c = entity.candidates[i];
                var color = utils.getColorForCandidate(i);
                utils.addDipsToChart(manualChart, manualData, c.dips, color);
                utils.addCandidateEntry(manualResult, c, i, color);
            }
        }

        //send to server
        sendCustomEvaluationToServer(entity);
        //hide modal
        hideModal();
    });
}
```

Figure 2.31: saving process of adjustments after a user is satisfied with the new result

```
//send a custom evaluation result to server after user adjustments
function sendCustomEvaluationToServer(evalEntity) {
    server.sendPostRequest(server_url + "LAEvaluation/addSingleCustom", {
        userId: 1,
        star: evalEntity.star,
        candidates: evalEntity.candidates,
        id: evalEntity.id,
        classificationResult: evalEntity.accuracy * 100,
    }, function () {
        logger.log_info("Succesfully sent adjustments to server");
    });
}
```

Figure 2.32: the new custom results get sent to the server

2.4.7 Logging

Logging was a necessity for the electron client, to log informations, ranging from quick status reports to warnings and errors. This is why a simple logging functionality was implemented. The client always saves a log file when running and also keeps the log file of the previous run. This made development a lot easier and should give users the ability to check for any problems.

```
//log a message with a severity-level and message to a log file
function log(level, message) {

    if (message === undefined || message.length == 0) {
        return;
    }

    var line = "[" + new Date().toLocaleString() + "] " + level + ": " + message.replace(/(?:\r\n|\r|\n)/g, ' ') + "\n";

    if (level == "ERROR") {
        console.error(line);
    } else {
        console.log(line);
    }

    //manage the log files
    if (init) {
        init = false;
        if (fs.existsSync(filename)) {
            if (fs.existsSync(filenameold)) {
                //delete old log file
                fs.unlinkSync(filenameold);
            }
            //rename latest log file to old log file
            fs.renameSync(filename, filenameold);
        }
    } else {
        //write in the current log file
        fs.writeFile(filename, line, { 'flag': 'a' }, (ret) => { });
    }
}
```

Figure 2.33: a simple logger for the electron client

2.5 Implementation of the Time Series Analysis

2.5.1 General

The Time Series Analysis is a major part of the system's logic. Its purpose is to extract important evidence from the light-curve that could point to the existence of an exoplanet orbiting a star. The theory, methods and use of Time Series Analysis was previously described in section 1.4 "Time Series Analysis" and will be further discussed here with a heavy emphasis on the practical implementation for the usage as a tool for the hunt for exoplanets.

Like mentioned in the system architecture part, if the light-curve classifier returns a classification result that exceeds a certain threshold(the probability for the existence of an exoplanet is high) the desktop client will apply the Time Series Analysis onto the light-curve to extract further information from it. The target of the analysis is to find periodic patterns called dips with which a candidate can be described and its parameters can be calculated.

Fig. 2.34 shows the different parts of the Time Series Analysis which will be described in more detail in the next few sections.

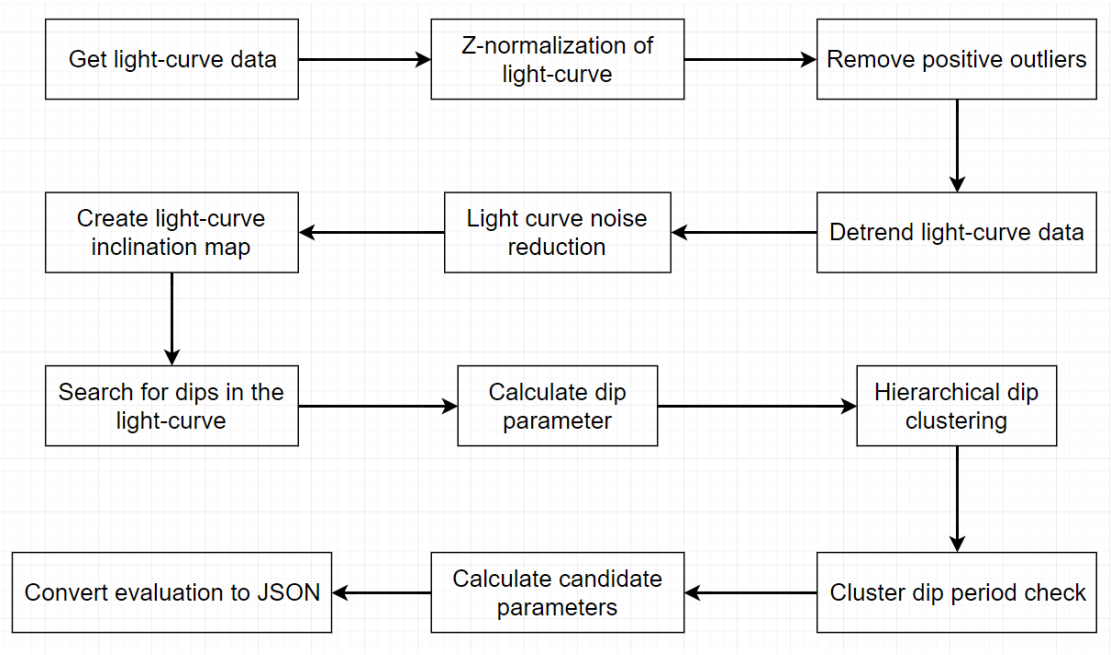

Figure 2.34: The different parts of the Time Series Analysis.

At first the light-curve gets sent from the desktop client to the Python process where the light-curve gets prepared before being fed to the analysis. To do so, many methods

typically used in Time Series Analysis which were explained in section 1.4 "Time Series Analysis" are used(some of them were already described in detail, including a Python implementation, in section 2.3 "Implementation of the light-curve classifier"). These methods include Z-normalization, positive outlier detection, detrending methods for the light-curve and noise reduction.

After data preparation an inclination map of the normalized light-curve data gets created to help find dips in the light-curve data. With the help of the normalized data and the inclination map the light-curve will be examined to search for dips and unusual patterns. All these discovered indicators will be clustered with an hierarchical clustering algorithm and post-checked with the so called cluster period check.

Finally all the candidates that were created through the analysis will be taken to determine all the important parameters of the exoplanets that could be hiding in the light-curve.

2.5.2 Preparing the light-curve

Preparing the light-curve data is an essential task before analysing it. Noise and unnecessary data like outliers could obstruct the analysis and cause unwanted errors in the results. To do so several Time Series Analysis methods get applied to the light-curve data.

2.5.2.1 Z-normalization

The first method for preparing the data that is applied to the light-curve is the Z-normalization. It helps to focus on the structural information of the data instead of the Y value. The Z-normalization was already covered in section 2.3.3 "Preparing data - CNN" and 1.4 "Time Series Analysis".

2.5.2.2 Reduce positive outliers

Positive outlier detection is also an important part of the data preparation task. It helps the analysis not to be handicapped by data points that could be created by external influences. The positive outlier detection was also covered in section 2.3.3 "Preparing data - CNN" and 1.4 "Time Series Analysis".

2.5.2.3 Detrend the light-curve

Detrending is a major part in data preparation because it helps to significantly simplify the light curve data before the analysis. The process of detrending removes the global trend of the flux values from the data without loosing essential structural information. Detrending was already discussed in section 1.4 "Time Series Analysis".

For this Time Series Analysis two different detrending methods were used. The first

one, implemented in Python can be seen in Fig. 2.35. This detrending method works by subtracting the (i-1) element from the (i) element. This detrending method has a small disadvantage because the output length of the sequence is length-1 instead of the original length of the sequence y.

```
# de-trend by differencing, cur - point before
def de_trend_other(x, y):
    diff_y = np.zeros(len(y) - 1)
    for i in range(1, len(y)):
        diff_y[i - 1] = y[i] - y[i - 1]
    return x[1:], diff_y
```

Figure 2.35: Light-curve detrending algorithm.

The second detrending algorithm can be seen in Fig. 2.36. Important to know is that both of these algorithms have different results which have important structural information the Time Series Analysis works with. That is the reason why both of them are used to find dips in the light-curve.

```
def de_trend(data, degree=10):
    detrended = [None] * degree
    for i in range(degree, len(data) - degree):
        chunk = data[i - degree:i + degree]
        chunk = sum(chunk) / len(chunk)
        detrended.append(data[i] - chunk)
    return detrended + [None] * degree
```

Figure 2.36: Light-curve detrending algorithm.

2.5.2.4 Light-curve noise reduction

Noise is a disturbing factor that can create errors within the analysis results. Fig. 2.37 shows the implementation of the noise reduction function which handles the noise in the positive and negative direction of the Y-axis separate to prevent the noise reduction from eradicating necessary information.

First the function separates all flux values into negative and positive values to determine the global positive and negative mean of the already detrended data. Afterwards the method iterates every flux value from the light-curve to check if the value is below a certain threshold. If the value is below this threshold the value will be replaced by 0 to reduce the noise in the light-curve. Furthermore the function also applies another layer of outlier detection on the light-curve to remove any outlier that could be created

by detrending. Finally the function returns the negative and positive mean and the light-curve data.

```
def remove_noise(x, y):
    pos_mean = np.mean([n for n in y if n is not None and n > 0])
    neg_mean = np.mean([n for n in y if n is not None and n < 0])

    real_pos_y = list()
    real_neg_y = list()

    for i in range(0, len(x)):
        if y[i] is not None and 0 < y[i] <= 2 * pos_mean:
            real_pos_y.append(y[i])
        elif y[i] is not None and 0 > y[i] >= 2 * neg_mean:
            real_neg_y.append(y[i])

    pos_mean = np.mean(real_pos_y)
    neg_mean = np.mean(real_neg_y)

    y = [0 if n is not None and (0 < n <= 3 * pos_mean or 0 > n >= 3 * neg_mean) else n for n in y]

    return neg_mean, pos_mean, x, y
```

Figure 2.37: Light-curve noise reduction algorithm.

2.5.3 Create light-curve inclination map

The inclination map is an important tool for finding dips in the light-curve. It is basically a 1D vector which describes the inclination of every flux value in the light-curve. By analysing the inclination map the algorithm can find local and unusual patterns like dips.

The implementation for creating an inclination map is very straight forward and can be seen in Fig 2.38. The algorithm simply iterates through the light-curve and calculates the inclination at the current point. The resulting inclination map gets returned at the end of the function.

```
def create_inclination_map(x, y):
    inclinations = list()
    for i in range(1, len(y)):
        inclinations.append((y[i] - y[i - 1]) / (x[i] / x[i - 1]))
    return inclinations
```

Figure 2.38: Algorithm for creating an inclination map.

2.5.4 Searching for dips

The search for dips is one of the key functionalities of the Time Series Analysis. The algorithm uses many different dimensions, like the inclination map, the detrended and normalized light-curve and the raw input data to search for unusual patterns that could point to the existence of an exoplanet.

The basic concept of the search is quite simple, the algorithm loops through the light-curve data and takes all dimensions known about a current item(like the current inclination and the current flux value) to assigns it a certain flag, resulting in a sequence of flags. Afterwards this sequence is used to extract certain flag patterns that could be potential candidates.

These potential candidates are then separated into different dip classes depending on different conditions like a minimal or maximal dip length, a minimal dip depth and many more. Depending on the matching conditions and the amount of satisfied conditions every potential candidate gets assigned a certain class. Each of these classes is weighed differently and describes how likely it is for a certain candidate to be a real dip.

2.5.5 Calculate dip parameters

Like mentioned in section 1.1.5.2.3 "Mathematical background and candidate parameter calculation" every dip has three important parameters that must be determined. These three are the total time, the full time and the delta flux value. The full time is the time a planet needs from ingress to egress, the total time is the duration of a full transit and the delta flux value is the depth of the dip in respect to the local average before the dip. All three values are essential for the clustering and the candidate parameter calculation.

Fig. 2.39 shows the implementation of the dip parameter calculation. First the algorithm calculates the average of the ingress and egress values to get the top line for calculating the delta flux value. After that the dip parameter calculation adds all data points which form the bottom of the dip to the list flat_fluxes. With the help of this list the bottom line of the dip is determined as the average of flat_fluxes. With the help of the bottom and top value of the dip the delta flux can be defined as the difference between them. Finally the first and last data point's x values are used to calculate the full time and the first and last data point's x values of the original dip are used to calculate the total time. At the end of this function the newly calculated parameters are returned.

2.5.6 Hierarchical dip clustering

After all dips are extracted out of the light-curve and their parameters are calculated the dips can be clustered into groups to form candidates. The algorithm type used for the dip clustering is an hierarchical clustering algorithm which was covered in section

```
def calc_dip_params(d):
    try:
        r_y = d.r_y
        r_x = d.r_x
        # return 0, 0, 0 if dip is too small
        if len(r_y) is 1 or len(r_y) is 2:
            return 0, 0, 0

        max_avg = 0
        if len(r_y) is 3:       # calculate dip top average
            max_avg = (r_y[0] + r_y[-1]) / 2
        else:
            max_avg = (r_y[0] + r_y[-2]) / 2

        offset = 0
        if len(r_y) is 3:   # determine dip offset
            offset = 0
        else:
            offset = 2
        all_avg = np.average(r_y[0:len(r_y) - offset])
        flat_fluxes = list()
        flat_xs = list()

        for i in range(0, len(r_y)): # get flat bottom of dip
            if r_y[i] < all_avg:
                flat_fluxes.append(r_y[i])
                flat_xs.append(r_x[i])

        ingress_x = flat_xs[0]
        egress_x = flat_xs[-1]
        # calculate float average of dip
        min_avg = np.average(flat_fluxes)
        # calculate delta flux
        delta_flux = max_avg - min_avg
        # calculate full time
        full_time = egress_x - ingress_x
        # calculate total time
        total_time = 0
        if len(r_x) is 3:
            total_time = r_x[-1] - r_x[0]
        else:
            total_time = r_x[-2] - r_x[0]
        # return results
        return delta_flux, full_time, total_time
    except:
        return None
```

Figure 2.39: Calculate dip parameters.

1.5.3.1 "Hierarchical Algorithm".

The basic principle of a hierarchical clustering algorithm is to start with n clusters where n is the number of elements to cluster. Then the algorithm merges the two clusters with the minimum distance until there is only one cluster left. However, the algorithm can also be executed if the distance exceeds a certain threshold. In this particular case the clustering algorithm starts with as many clusters as there are dips and starts to merge the two dips that have the minimal distance between them until a certain distance threshold is reached. Fig. 2.40 shows the clustering algorithm used to cluster the dips.

First the two clusters with the minimal distance get selected in nested for-loops. Afterwards the algorithm checks if the minimum distance exceeds the maximum distance that is allowed. If the max. distance is exceeded the clustering will end. Else the two selected clusters get merged together to form a new cluster.

An important part of the clustering algorithm is the cost function. In this example the structural information of the dips is the perfect information to measure the distance between two clusters. Therefore the dynamic time warping algorithm is used to determine the distance. The DTW algorithm was previously discussed in section 1.5.5 "Dynamic time warping algorithm". Fig. 2.41 shows a simple Python implementation of the DTW algorithm.

2.5.7 Cluster dip period check

The cluster period check is the last testing of the different clusters before the candidate parameter calculation. Its purpose is to look for falsely clustered dips and to correct them. The algorithm calculates the periods between each dip of a cluster, determines the average period of a cluster and checks if the periods between the dips match the average period. If the algorithm detects an unusual period between the dips it looks at the other clusters and their dips and tries to switch them around or removes the wrong dip until the error is resolved. Thereby dips cannot be added to clusters if the difference between the average parameters of the cluster and the single dip is too large. If the error cannot be resolved the cluster period check tries to simulate a new dip to correct the error.

2.5.8 Calculate candidate parameters

One of the most important parts of the Time Series Analysis is the parameter calculation. The porpuse of the parameter calculation is to calculate all essential parameters of a candidate. There are several important properties like the period, orbital velocity, radius or equilibrium temperature which must be calculated to create a good evaluation of a candidate. All of the parameters and the mathematical background that will be dealt with in this section was previously discussed in section 1.1.5.2.3 "Mathematical background and candidate parameter calculation".

```
is_running = True
cnt = 0
while is_running:
    min_cluster_from = None
    min_cluster_to = None
    min_sum_cluster_dist = float('inf')
    min_dist_str = ''
    min_comp = None
    # search for the two clusters with the min. distance
    for ci in clusters:
        for cj in clusters:
            if ci.id != cj.id:
                # calculate the distance between two clusters
                cur_sum_cluster_dist, cur_comp, dist_str =
                calc_dip_dist(ci.min_dip, cj.min_dip, True)
                comp_res = check_comps(cur_comp,
                cur_sum_cluster_dist,
                min_comp,
                min_sum_cluster_dist)
                # compare the current distance with the current min. distance
                if comp_res == 'min':
                    # replace min. two clusters with new
                    min_comp = cur_comp
                    min_sum_cluster_dist = cur_sum_cluster_dist
                    min_cluster_from = ci
                    min_cluster_to = cj
                    min_dist_str = dist_str
    # check if the min. distance is exceeds the max. possible distance
    if min_sum_cluster_dist == float('inf') or min_sum_cluster_dist > MAX_DISTANCE_ERROR:
        print_all_clusters(clusters)
        is_running = False
        continue
    else:
        # merge two clusters
        new_cluster = Cluster(max_cluster_id)
        max_cluster_id += 1
        new_cluster.dips.extend(min_cluster_from.dips)
        new_cluster.dips.extend(min_cluster_to.dips)
        # get all clusters that are in the pool now
        clusters = [c for c in clusters if c.id != min_cluster_from.id and c.id != min_cluster_to.id]
        clusters.append(new_cluster)
    cnt += 1
```

Figure 2.40: Dip clustering algorithm.

2.5.8.1 Orbital period

The first parameter is the orbital period which will be determined with the average of all periods between the dips of a candidate. If a candidate has at least two dips, the period can be determined.

Fig. 2.42 shows the implementation of the function calc_periods(cluster) which calculates the orbital period of a cluster/candidate. This is done by first determining all periods between the dips before calculating the average value to get the orbital period.

```
def dynamic_time_warping(dip1, dip2):
    dtw = {}

    for i in range(len(dip1)):
        dtw[(i, -1)] = float('inf')
    for i in range(len(dip2)):
        dtw[(-1, i)] = float('inf')
    dtw[(-1, -1)] = 0

    for i in range(len(dip1)):
        for j in range(len(dip2)):
            dist = (dip1[i] - dip2[j]) ** 2
            dtw[(i, j)] = dist + min(dtw[(i - 1, j)],
            dtw[(i, j - 1)],
            dtw[(i - 1, j - 1)]
            )

    return np.sqrt(dtw[len(dip1) - 1, len(dip2) - 1])
```

Figure 2.41: Dynamic time warping algorithm.

```
def calc_periods(cluster):
    if len(cluster.dips) <= 1:
        return [], None
    periods = list()
    for i in range(0, len(cluster.dips) - 1):
        periods.append(cluster.dips[i + 1].startPoint - cluster.dips[i].startPoint)
    avg_period = np.mean(periods)
    return periods, avg_period
```

Figure 2.42: Calculating the orbital period.

2.5.8.2 Radius, error 1 and error 2

A very meaningful parameter of the candidate is the radius. The radius depends on the depth of the dip and the star's radius. Fig. 2.43 shows the implementation of the calc_radius(cluster, star, error=0.0) function which determines the planet's radius. First the local average flux value is calculated before determining the planet's radius and returning it. The error variable helps when calculating with the positive and negative error of the star.

2.5.8.3 Semi-major axis

The semi-major axis is the radius between the star and the candidate. Fig. 2.44 shows the implementation of the calc_semi_major_axis(candidate, star) function which deter-

```
def calc_radius(dip, star, error=0.0):
    if star.radius is None:
        return None
    if error is None:
        error = 0.0
    main_flux = (dip.r_y[0] + dip.r_y[-1]) / 2
    planet_radius = np.sqrt((dip.deltaFlux / main_flux) *
        np.power((star.radius + error) *
            utils.RADIUS_SUN, 2)) / utils.RADIUS_JUPITER
    return planet_radius
```

Figure 2.43: Calculating the radius.

mines the planet's semi-major axis. The semi-major axis cannot be calculated if the orbital period was not calculated; also it needs the mass of the star.

```
def calc_semi_major_axis(candidate, star):
    if candidate.periodAvg is None:
        return None
    else:
        if star.mass is None:
            return None
        a = np.power(candidate.periodAvg * utils.DAY_TO_SECOND, 2) *
            utils.GRAVITATIONAL_CONSTANT *
                star.mass * utils.MASS_SUN
        a = np.power(a / (4 * np.pi * np.pi), 1.0 / 3.0)
        a = a / utils.ASTRONOMICAL_UNIT
        return a
```

Figure 2.44: Calculating the semi-major axis.

2.5.8.4 Orbital velocity

The orbital velocity is the velocity with which an exoplanet orbits its star. For the calculation of the orbital velocity it is taken for granted that the planet has a perfect circular orbit; a more accurate determination of the average velocity is not possible with the value that can be derived from the light-curve. Fig. 2.45 shows the implementation of the function calc_orbital_velocity(candidate, star) which calculates the orbital velocity of the candidate. The orbital velocity cannot be determined if the average period or the semi-major axis are not available.

2.5.8.5 Equilibrium temperature

The thermal equilibrium temperature is a theoretical temperature of a planet if it were considered a simple black body being heated only by its host star. Fig. 2.46 shows the

```
def calc_orbital_velocity(candidate, star):
    if candidate.periodAvg is None or candidate.semiMajorAxis is None:
        return None
    else:
        v = (2 * np.pi * candidate.semiMajorAxis * utils.ASTRONOMICAL_UNIT) /
            (candidate.periodAvg * utils.DAY_TO_SECOND)
        return v
```

Figure 2.45: Calculating the orbital velocity.

implementation of the function calc_equilibrium_temp(candidate, star) which calculates the equilibrium temperature for a spherical black body with reflection of 0.1, 0.3 and 0.7. The equilibrium temperature cannot be calculated if the semi-major axis, the star's effective temperature or the star's radius are missing.

```
def calc_equilibrium_temp(candidate, star):
    if candidate.semiMajorAxis is None or star.effectiveTemp is None or star.radius is None:
        return None, None, None
    else:
        candidate.isEquilibriumTempValid = True
        temp1 = star.effectiveTemp * np.power(1-0.1, 1/4) *
            np.sqrt((star.radius * utils.RADIUS_SUN) /
                (2 * candidate.semiMajorAxis * utils.ASTRONOMICAL_UNIT / 1000))
        temp2 = star.effectiveTemp * np.power(1-0.2, 1/4) *
            np.sqrt((star.radius * utils.RADIUS_SUN) /
                (2 * candidate.semiMajorAxis * utils.ASTRONOMICAL_UNIT / 1000))
        temp7 = star.effectiveTemp * np.power(1-0.7, 1/4) *
            np.sqrt((star.radius * utils.RADIUS_SUN) /
                (2 * candidate.semiMajorAxis * utils.ASTRONOMICAL_UNIT / 1000))
        return temp1, temp2, temp7
```

Figure 2.46: Calculating the equilibrium temperature.

2.6 Implementation of the Server

2.6.1 General

The server is the core part for distributing data to the different clients, both the desktop client and the website. It is also responsible for persisting evaluations, resulting from the light-curve classifier and the Time Series Analysis. Furthermore the server applies additional methods on the analysed data to map it with already confirmed exoplanets or corresponding candidates from the NASA databases.

2.6.2 Rest interfaces for the most important classes

Like previously mentioned, the server is the main engine for the distribution of data between the clients. Therefore every important analysis result that is saved on the database or any statistical data can be accessed via simple REST-Endpoints. That allows every client independent of its implementation language to access the data and users or scientists that want to use the Exohunter data for further research.

The most important REST-endpoints are as follows:

1. /LAEvaluation, handles the basic CRUD operations for LAEvaluations as well as other important functionalities
   (a) /addSingleComputed, adds the computed(results of the desktop client) analysis results to the LAEvaluation data
   (b) /addSingleCustom, adds the changes of the LAEvaluations by the user to the LAEvaluation data and creates a user feedback
   (c) /K2LACandidateMatchingResultsOfEvaluation/id, returns all confirmed exoplanets or candidates from the NASA data that could correspond with the LAEvaluation of id
   (d) /calcProbability/{epic}, calculates the probability for the existence of exoplanets in a system with the corresponding epic number
2. /LACandidate, handles the basic CRUD operations for LACandidates
3. /LADip, handles the basic CRUD operations for LADips
4. /Star, handles the basic CRUD operations for Stars
5. /K2LACandidateMatchingResult, handles the basic CRUD operations for K2LACandidateMatchingResults
6. /ExoplanetStats, handles the basic CRUD operations for ExoplanetStats and further statistical rest interfaces
   (a) /findCurrentExoplanetStat, get the latest exoplanet statistics

   (b) /findCntOfConfirmed, get count of currently confirmed exoplanets
   (c) /findCntPerYear, get the count of confirmed exoplanets per year
   (d) /findCntPerYearAndMethod, get the count of confirmed exoplanets per year and method
   (e) /findCntPerYearAndMethodPerc, get the percentage of confirmed exoplanets per year and method
7. /Search, handles the filtering of LAEvaluations with /findAll
8. /CnnVersion, handles the CNN version control and the basic CRUD operations for CnnVersions
   (a) /findLatestVersion, get the latest version number of the Cnn
   (b) /downloadLatestVersion, downloads the latest model file of the Cnn

2.6.3 K2 epic number retrieval

The client needs a continuous supply of K2 epic numbers, to feed the batch mode with stars to analyse. In order to provide for multiple users, a list of epic numbers is loaded into memory which is then used as a cache. When the client calls the REST endpoint for an epic number, one entry of the cache gets removed and returned to the caller. Initially, 100 epic numbers are loaded into a list in memory, which is refilled once the cache size reaches a small size of about 20. To maintain functionality, all load-processes are done in background using threads, so REST endpoints are not getting blocked.

```
//Loads a given amount of ids into the cache in a background thread
@Override
public void run() {
    List<String> total = new LinkedList<>();

    List<String> resultSet1 = getEpicIds(getCampaignLink(campaign), startIndex, length);
    total.addAll(resultSet1);

    if (resultSet1.size() < length) { //Not enough data in current campaign => load next campaign
        //get the next campaign and fill up with remaining ids
        int newCampaign = incrementCampaign();
        //Only load the remaining data to reach cache size
        List<String> resultSet2 = getEpicIds(getCampaignLink(newCampaign), 0, length - total.size());
        total.addAll(resultSet2);
        root.setCurrentCampaign(newCampaign);
        root.setCurrentCampaignIndex(resultSet2.size());
    } else {
        root.setCurrentCampaignIndex(root.getCurrentCampaignIndex() + length);
    }

    System.out.println("Loaded " + total.size() + " Entries into cache");
    //add 100 entries to cache
    root.getCache().addAll(total);
}
```

Figure 2.47: Background thread that refills the cache

As seen in figure 2.47, the run-method is implemented, which means it runs in the background. A campaign has a limited amount of epic numbers and when there are less than 100 entries left to fetch, the next campaign has to be queried for the remaining numbers. The combined result is then added to the cache. Additionally, the cursor of the current campaign and entry position has to be persisted, so that the server can continue where it stopped when it restarts.

To fetch epic numbers from a campaign, the URL to the web page of the campaign is required. Additionally start and end point are necessary to limit the result set, because fetching all epic numbers from a campaign is not purposeful. A Java library called Jsoup is used, as explained in section 1.7.6, to fetch only the epic numbers from the website, which contains a lot of other information that is not needed. How the URLs are fetched

```java
//Loads Ids from a campaign from a given start point and length
public List<String> getEpicIds(String path, int start, int len) {
    try {
        List<String> out = new LinkedList<>();

        //Load all ids from a campaign-page into a list via Jsoup
        Elements elements = Jsoup.connect(path).get().getElementsByTag("a");

        int end = start + len;

        //Filter out the needed elements
        for (int i = start; i < elements.size() && i < end; i++) {
            Element a = elements.get(i);
            out.add(a.html().replaceAll("\\D+", ""));
        }
        return out;
    } catch (Exception e) {
        System.out.println(e);
        return new LinkedList<>();
    }
}
```

Figure 2.48: Implementation of epic number fetching

is explained in section 1.7.7.

2.6.4 Add new evaluation and user feedback

After the desktop client has analysed a light-curve with the light-curve classifier and the Time Series Analysis it sends the results in form of an evaluation to the server. However it is important to note that there are two similar rest endpoints which perform a very different logic. These two endpoints are the /addSingleComputed and the /addSingleCustom endpoint.

The principal behind these two endpoints is very simple. After the desktop client has analysed the light-curve which does not involve the user's input the results will be sent to the /addSingleComputed endpoint where the evaluation will be checked further. In doing so the algorithm checks if there is already an evaluation for the current star, if not the star data is also persisted to the database. Afterwards the newly added evaluation will be checked against the already confirmed exoplanets and candidates to find corresponding matches.

If the user decides that the current evaluation has an error he or she can create a user feedback with which he or she can actively edit the results of the analysis. This user feedback is then sent to the /addSingleCustom endpoint where the changes made in the user feedback are applied to the already existing evaluation with the same id. Furthermore a user feedback entry is created which will be used to advance the light-curve classifier.

Fig. 2.49 shows the method addSingleComputed which takes in a json object, creates an evaluation and persists it on the database. In Fig. 2.50 the Java implementation of the addSingleCustom method can be seen. At first the new evaluation with the changes is created, then the old evaluation of the user of the current star is fetched from the database to check against the new one. If the classification of the user and the analysis do not match a user report is created to further advance the light-curve analysis.

Every time a new evaluation is sent to the server to persist it is essential to link it

```
public LAEvaluation addSingleComputed(JSONObject json) {
    User user = null;
    if(json.has("userId")) {
        user = userFacade.findById(json.getLong("userId"));
    }
    LAEvaluation e = LAEvaluation.convertFromJson(json, user);
    save(e);
    candidateCheckController.checkLAEvaluation(e, true);
    return e;
}
```

Figure 2.49: Java implementation of method addSingleComputed.

with all already confirmed exoplanets and candidates from the NASA database to check

```
public LAEvaluation addSingleCustom(JSONObject json) {
    User user = null;
    if(json.has("userId")) {
        user = userFacade.findById(json.getLong("userId"));
    }
    LAEvaluation e = LAEvaluation.convertFromJson(json, user);

    // find other evaluation with user and star
    List<LAEvaluation> others = em.createNamedQuery("LAEvaluation.findByUserIdAndStar")
        .setParameter("userId", e.getUser().getId())
        .setParameter("epic", e.getStar().getEpic())
        .getResultList();
    LAEvaluation other = others.get(0);

    DateTimeFormatter dtf = DateTimeFormatter.ofPattern("dd.MM.yyyy HH:mm:ss");

    save(e);
    candidateCheckController.checkLAEvaluation(e, true);    // apply candidate check

    // if user classification of user and CNN don't match -> create user report
    if(Math.round(e.getClassificationResult()) != Math.round(other.getClassificationResult())) {
        UserReport ur = new UserReport();
        ur.setUsed(false);
        ur.setUser(user);
        ur.setEpic(e.getStar().getEpic());
        ur.setTimeStamp(LocalDateTime.parse(LocalDateTime.now().format(dtf), dtf));
        ur.setOldClassification(other.getClassificationResult());
        ur.setUserClassification(e.getClassificationResult());

        userReportFacade.save(ur);
        userReportFacade.checkForCnnUpdate();   // check if CNN can be updated
    }

    em.remove(other);

    return e;
}
```

Figure 2.50: Java implementation of method addSingleCustome.

if there is any additional evidence for the existence of an exoplanet. Therefor every confirmed exoplanet and candidate which belongs to the analysed star gets fetched from the NASA database and is compared with the different candidates of the evaluation. Every match between a candidate and the analysis results gets also saved into the database and is from then on part of the analysis result.

The code of the LAEvaluation candidate check can be found in the CandidateCheckController.java method checkLAEvaluation(LAEvaluation e, boolean persistMatching) where e is the LAEvaluation that should be checked and persistMatching says if the results of this check should be saved to the database.

2.6.5 Exoplanetary statistics

On the Exohunter website there are two charts, the first shows the distribution of confirmed exoplanets over the last few years and the second shows the percentage of each exoplanet finding method. To create these two charts and potentially many more, the decision was made to create an entity to save planetary statistics, the ExoplanetStats. This entity saves the year, method and count of found exoplanets with which many different statistics can be created like the count of all confirmed exoplanets, the count of exoplanets per year or the count of exoplanets per year and method.

To keep this data up-to-date the server runs a scheduled tasks which refreshes the planetary stats once every day with the help of the NASA database. The code of this updating task can be seen in Fig 2.51. The complexity and length of the method do not make it possible to show the full code. However the code in the for loop is only a simple parsing algorithm to parse the data fetched from NASA into the format year - all methods - count of confirmed exoplanets.

```
public ExoplanetStat updateExoplanetStat() {
    // fetch data from nasa
    Client client = Client.create();
    System.out.println(ExoplanetStat.URL_GET_WITH_YEAR);
    WebResource resource = client.resource(ExoplanetStat.URL_GET_WITH_YEAR);
    ClientResponse response = resource.accept("application/json").get(ClientResponse.class);
    if (response.getStatus() != 200) {
        throw new RuntimeException("Failed : HTTP error code : " + response.getStatus());
    }
    String str = response.getEntity(String.class);
    String[] lines = str.split("\n");

    Map<Integer, Map<String, Integer>> meta = new HashMap<>();

    // parse the result into the year/method/count data
    for(int i = 0; i < lines.length; i++) {···
    }

    // convert the map into the exoplanet stat data
    List<ExoplanetStatRow> esr = new LinkedList<>();
    for(int year: meta.keySet()) {
        for(String method: meta.get(year).keySet()) {
            ExoplanetStatRow row = new ExoplanetStatRow(year, method, meta.get(year).get(method));
            esr.add(row);
        }
    }
    ExoplanetStat es = new ExoplanetStat();
    es.setTimeStamp(LocalDateTime.parse(LocalDateTime.now().format(ExoplanetStat.dtf), ExoplanetStat.dtf));
    es.setExoplanetStatRows(esr);
    return es;
}
```

Figure 2.51: Java implementation of method updateExoplanetStat.

2.6.6 Improving the light-curve classifier

An important part of the server is to advance the light-curve classifier with the help of the user feedback. Therefore every time a new user feedback is added to the database, the server checks if already enough user feedback has been collected to re-train the CNN. If there is enough user feedback, an algorithm converts all the user reports into training data, by calculating the user classification value of every star that was mentioned in the feedback and loading the light-curve of the star. These training data gets persisted in the database and is used to train the CNN. Furthermore after training a new CnnVersion is created so that all clients can download the new model file.

Fig. 2.52 shows the implementation of the previously described user feedback check. First the algorithm checks if enough user feedback is available for re-training the network. If so, all the user reports are converted into training data which is used to train the CNN.

2.6.7 Evaluation probability calculation

An important feature of the server is to calculate the probability information data if exoplanets are orbiting a certain star. To do so, the server uses the analysis results of the Exohunter projects and the references with the NASA database(already confirmed exoplanets and candidates which are already linked with the evaluations).

First the algorithm fetches all K2LACandidateMatchingResult(the already confirmed exoplanets and candidates from the NASA database) and all evaluations of a certain star. Afterwards the function calculates the weighted average value of the evaluation classification results and determines the count of confirmed, candidate and false positive K2LACandidateMatchingResults. All of this data, the count of evaluations, count of confirmed, candidate and false positive matching results, average classification value, the star data and the distinct candidate names gets pushed into a json object and returned by the function. Fig. 2.53 shows the data fetching and calculation part of the algorithm.

```
public void checkForCnnUpdate() {
    // fetch not used data
    List<Object[]> columns = em.createNamedQuery("UserReport.findUpgradeableData")
        .setParameter("cnt", NUM_OF_USER_REPORTS_FOR_CLASSIFICATION)
        .getResultList();
    // check if enough data is available
    if(columns.size() >= NUM_OF_MIN_CNN_UPGRADE_USER_REPORTS) {
        for(Object[] colRow: columns) {
            int epic = (int)colRow[0];
            Long cnt = (Long)colRow[1];
            List<UserReport> unusedUserReports = findByStarUnused(epic);

            // calculate new classification result from user reports
            double resClassification = 0;
            for(UserReport ur : unusedUserReports) {
                ur.setUsed(true);
                merge(ur);
                resClassification += ur.getUserClassification();
            }
            resClassification /= unusedUserReports.size();
            resClassification /= 100;
            int result = (int)Math.round(resClassification);

            // create new training data set
            TrainData td = trainDataFacade.findByStar(epic);
            if(td == null) {
                List<Float> list = getSeqFromEpic(epic);
                TrainData newTd = new TrainData(epic, result, list);
                trainDataFacade.save(newTd);
            } else {
                td.setClassification(result);
                trainDataFacade.merge(td);
            }
        }
        // re-traing CNN
        trainCnn();
    }
}
```

Figure 2.52: Java implementation of the re-traing check.

```
List<LAEvaluation> es = laEvaluationFacade.findAllByStar(epic);
List<K2LACandidateMatchingResult> candidates =
    k2LACandidateMatchingResultFacade.findAllByStar(epic);

List<String> alreadyTaken = new LinkedList<>();

int cntOfConfirmed = 0;
int cntOfCandidates = 0;
int cntOfFalsePositive = 0;

for(K2LACandidateMatchingResult c: candidates) {
    if(!alreadyTaken.contains(c.getCandidateName())) {
        if (c.getDisposition() == CandidateDisposition.CONFIRMED) {
            cntOfConfirmed++;
        } else if (c.getDisposition() == CandidateDisposition.CANDIDATE) {
            cntOfCandidates++;
        } else if (c.getDisposition() == CandidateDisposition.FALSE_POSITIVE) {
            cntOfFalsePositive++;
        }
        alreadyTaken.add(c.getCandidateName());
    }
}

// calculate weighted average if evaluation classification results
double averageClassification = 0;
int weight = es.size();
for(LAEvaluation e: es) {
    averageClassification += e.getClassificationResult() * weight;
    weight--;
}
int div = 0;
for(int i = es.size(); i >= 0; i--) {
    div += i;
}
averageClassification /= div;

int numberOfEvaluations = es.size();
```

Figure 2.53: Java implementation of method calcPropability.

2.7 Implementation of the Website

2.7.1 General

The Exohunter website is the main client for users to inform about the project, to look at the analysis results, to download the desktop client and much more. The purpose of the front-page is to inform new users about the Exohunter project, describe what the Transit method is and how exoplanets can be found. The website was created with JavaScript, CSS3, HTML5 and JQuery.

The first part of the front-page can be seen in Fig. 2.54. It is basically a simple banner with a network animation in the background. On the top is a navbar which helps the user navigate to the download page of the desktop client, the API to see the list of all evaluations and the About-page which provides additional information about the creators of the Exohunter project.

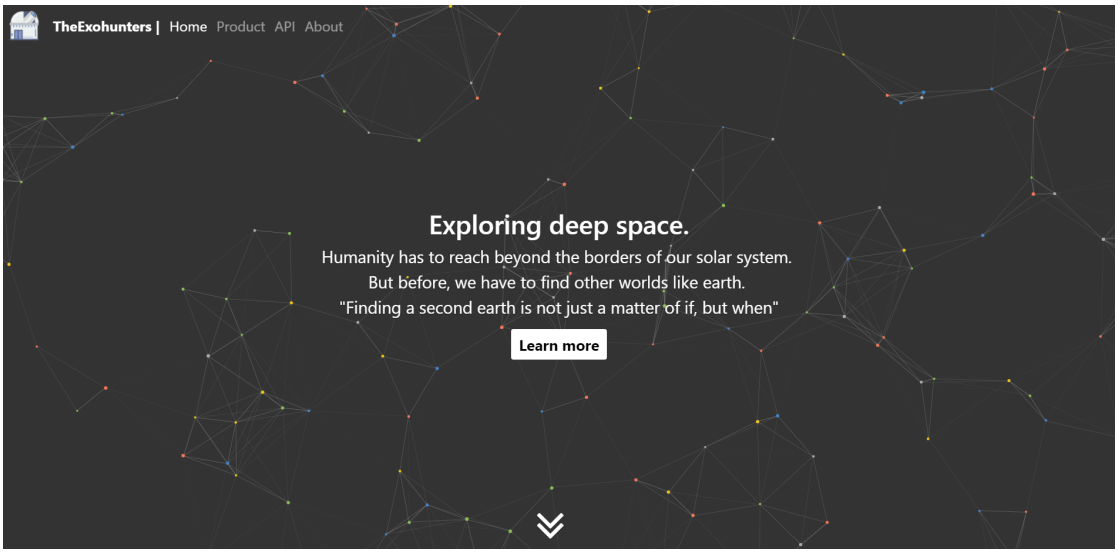

Figure 2.54: The banner of the front-page.

The second part of the front-page is a brief description of the Exohunter project that informs the user about the most important information. This section of the website can be seen in Fig. 2.55.

Fig. 2.56 shows the third part of the front-page, which is another simple banner that fo-

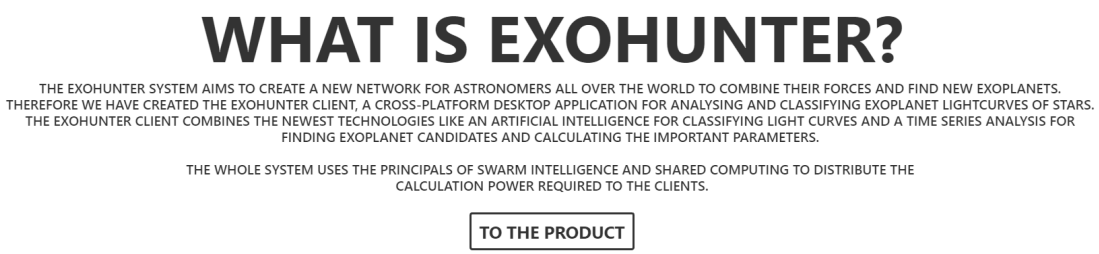

Figure 2.55: The brief project description.

cuses on the main technological parts this project is built with or wants to be associated with. That involves artificial intelligence and learning AI with the light-curve classifier, the light curve analysis and the parameter calculation, the crossplatform desktop client and the idea to create a network for people who want to search for exoplanets, who are called Exohunters.

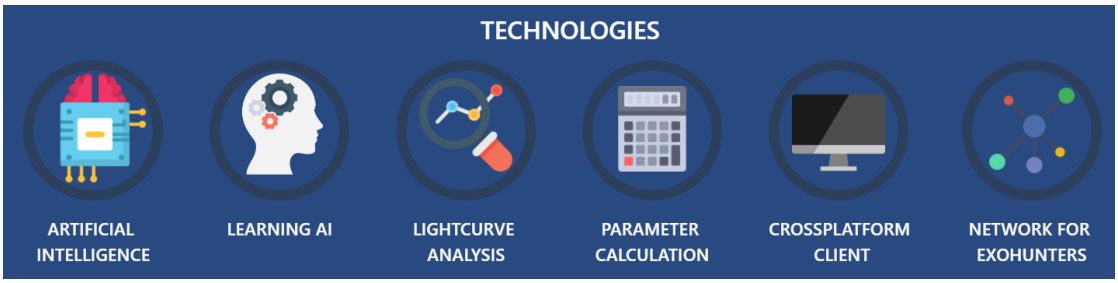

Figure 2.56: The most important technologies and goals of Exohunter.

The fourth part of the front-page is a quick explanation of what an exoplanet is. Followed by a short explanation of the Transit method with an additional GIF which shows the transit of an exoplanet around its host star. The exoplanet description can be seen in Fig. 2.57 and the Transit method explanation can be seen in Fig. 2.58.

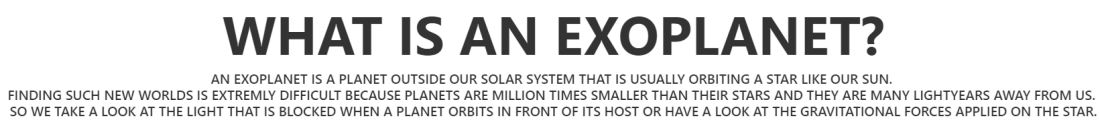

Figure 2.57: Short description of an exoplanet.

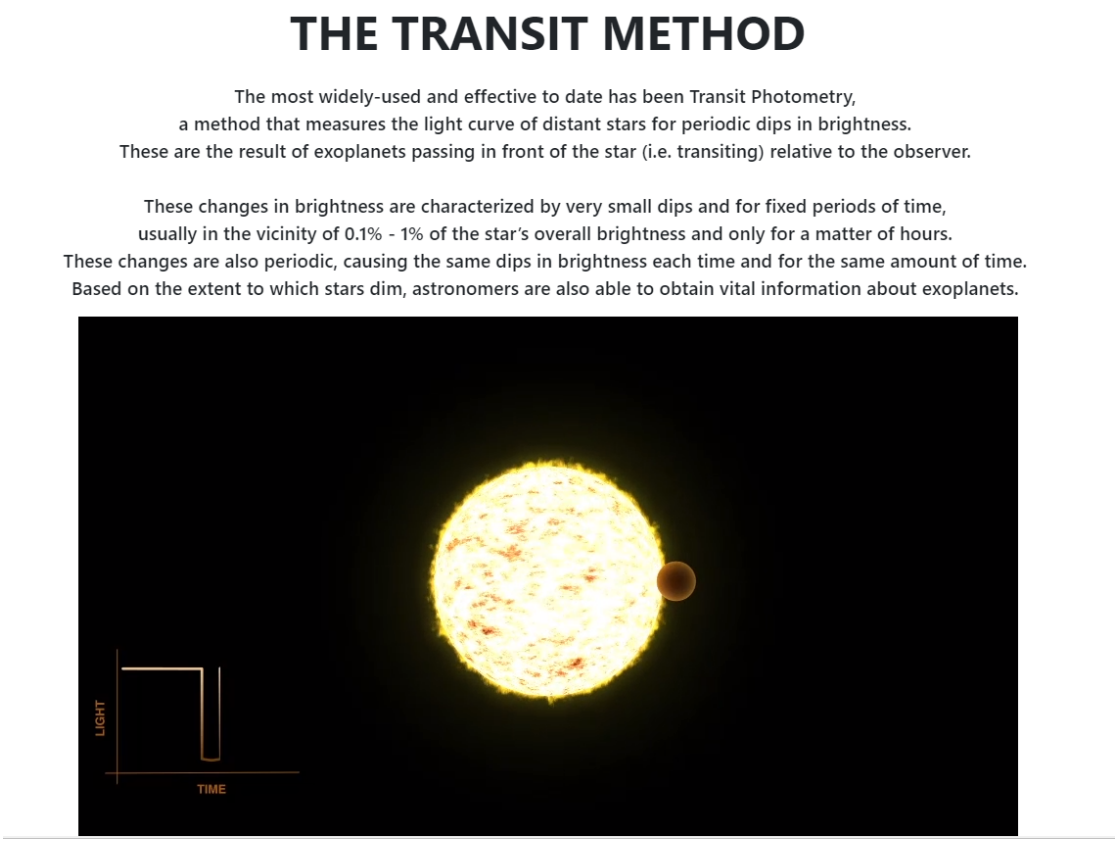

Figure 2.58: Explanation of the transit method.

2.7.2 Exoplanet statistics

The front-page also shows important exoplanet statistics, like the count of already confirmed exoplanets or the distribution of confirmed exoplanets over the last few years. To supply this information the website accesses the exoplanetary statistics which are saved on the server and updated every day. Every time the website is loaded the JavaScript in the background loads the statistical data from the server. First the count of confirmed exoplanet, then the count of exoplanets of the last few years and at the end the exoplanet finding method distribution data. The code which handles the loading of the data can be seen in Fig. 2.59.

2.7.2.1 Count of exoplanets

The first statistical data is a simple integer which shows the current number of confirmed exoplanets which can be seen in Fig. 2.60

```
function getExoplanetMetaData() {
  // get count of confirmed exoplanet
  sendGetRequest(url_findCntOfConfirmed, function(data) {
    cntOfConfirmed = data.count
    $("#exo-stats-number-confirmed-exo").html(numberFormat(cntOfConfirmed) + " ")
  })
  // get statistics for exoplanets per year chart
  sendGetRequest(url_findCntPerYear, function(data) {
    var years = []
    var count = []
    for(var i in data) {
      years.push(data[i].year)
      count.push(data[i].count)
    }
    chart_labels = years
    chart_series = count
    chart_data = {
      labels: years,
      series: [count]
    };
    handleCountOfExoplanetChart()
  })
  // get statics for exoplanet finding method distribution
  sendGetRequest(url_findCntPerYearAndMethodPerc, function(data) {
    exoMethodPerc = data
    $('#circle-chart-transit-val').attr('stroke-dasharray', data.percentageTransit + ', 100')
    $('#circle-chart-transit-text').html(data.percentageTransit + '%')

    $('#circle-chart-radial-velocity-val').attr('stroke-dasharray', data.percentageRadialVelocity + ', 100')
    $('#circle-chart-radial-velocity-text').html(data.percentageRadialVelocity + '%')

    $('#circle-chart-imaging-val').attr('stroke-dasharray', data.percentageImaging + ', 100')
    $('#circle-chart-imaging-text').html(data.percentageImaging + '%')

    $('#circle-chart-other-val').attr('stroke-dasharray', data.percentageOther + ', 100')
    $('#circle-chart-other-text').html(data.percentageOther + '%')
  })
}
```

Figure 2.59: Load all planetary statistics for the website.

Figure 2.60: Count of confirmed exoplanets.

2.7.2.2 Exoplanets per year

The second statistical data is a chart which shows the count of confirmed exoplanets found per year in the last few years. The chart that is created by the website can be seen in Fig. 2.61. The bar chart was created with Chartist.js which is described in section 1.8.1 "Chartist". Chartist makes the creation, handling and animation of charts

very easy. The code for the creation, filling and animation of the Chartist bar chart can be seen in Fig. 2.62.

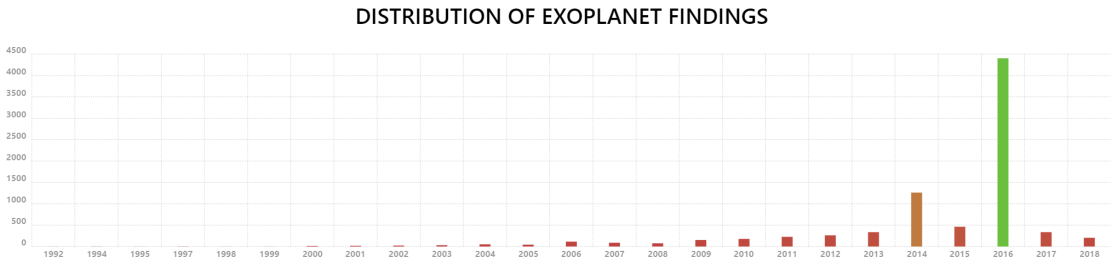

Figure 2.61: Count of confirmed exoplanets per year.

```
function handleCountOfExoplanetChart() {
  // define chart options
  var chart_options = {
    width: window_width * 0.8,
    height: 400,
    plugins: [
        Chartist.plugins.tooltip()
    ]
  };
  // create bar chart
  chart = new Chartist.Bar(
    ".ct-chart",
    chart_data,
    chart_options,
    chart_responsive_options
  );
  // handle bar chart animations
  chart.on("draw", function(context) {
    if (context.type === "bar") {···
    }
  });
}
```

Figure 2.62: Creation of the bar chart with Chartist.

2.7.2.3 Exoplanet finding method distribution

The last statistical data are four percentage charts which show the distribution of the exoplanet finding methods. The charts can be seen in Fig. 2.63.

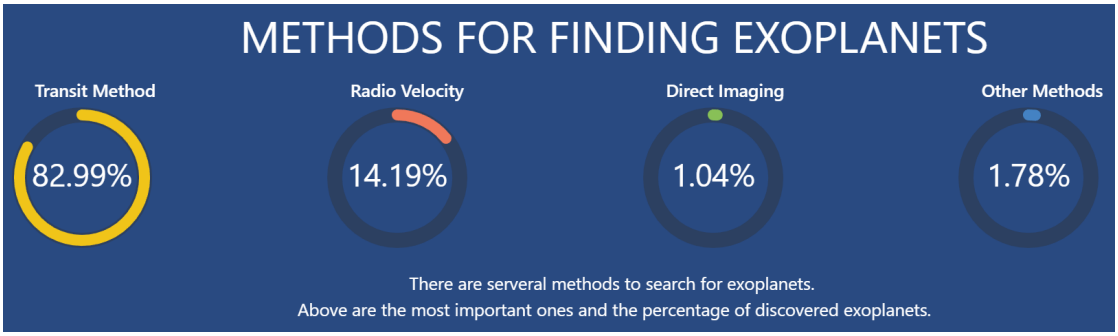

Figure 2.63: Distribution of exoplanet finding methods.

2.7.3 List of evaluations

The evaluation list view is a very essential part of the website. It allows the user to access every analysis that was ever performed with the Exohunter system. The list view can be seen in Fig. 2.64. The single evaluations are displayed as a simple rectangle and contain the most important information of the evaluation like the author name, the number of found candidates, found dips and references to other already confirmed exoplanets or candidates. Furthermore the rectangle shows the classification result for the light-curve from the user him- or herself or the light-curve classifier, the timestamp of the analysis and the epic number of the star.

The navbar of the list view consists of different elements that help to filter the evaluations that will be displayed below. The user can filter by the epic number, the username, the maximum classification result and the number of candidates. Furthermore the user selects the way the evaluations are ordered; either by timestamp, epic number, username or number of candidates.

When hovering over one of the rectangles it will float a little bit to the right to show that it is active. If the user clicks on the rectangle he or she will be redirected to the detail view of the clicked evaluation. The detail view is described in the next section.

2.7.4 Evaluation detail page

The evaluation detail page is also an important part of the website. It shows the user all information of a single evaluation like the light-curve, all the candidates, dips, references, exoplanet parameters and others. The detail page can be separated into two parts. The first one is the light-curve chart, a simple chart that displays the light-curve and the second one is the information part. This part shows all the essential analysis information in different rectangular boxes.

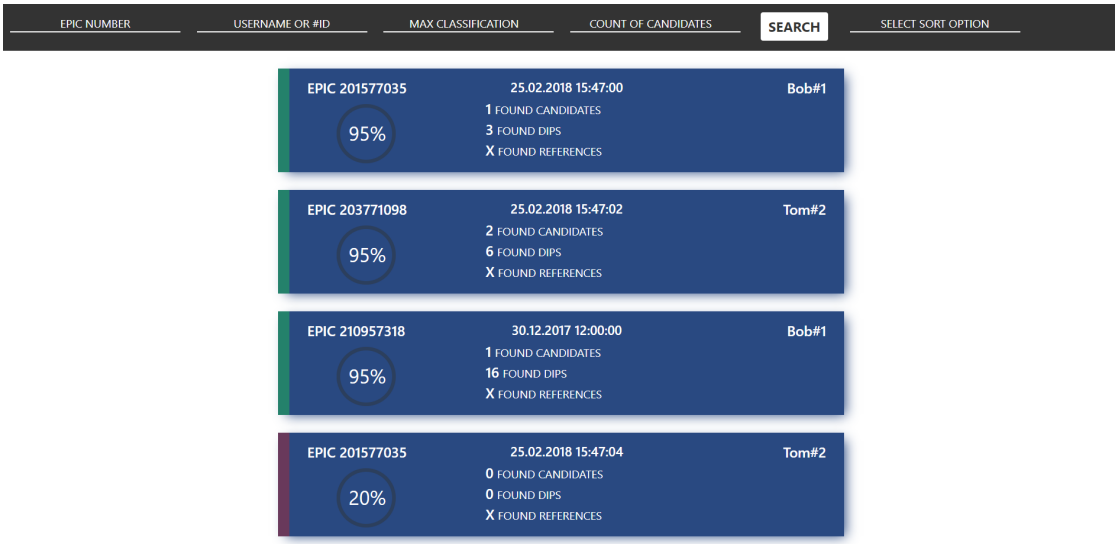

Figure 2.64: Evaluation list view.

2.7.4.1 Light-curve chart

The light-curve chart shows the light-curve of the current evaluation. The user can zoom in and out in the chart, can reset the zoom or hide the whole chart. The light-curve chart can be seen in Fig. 2.65.

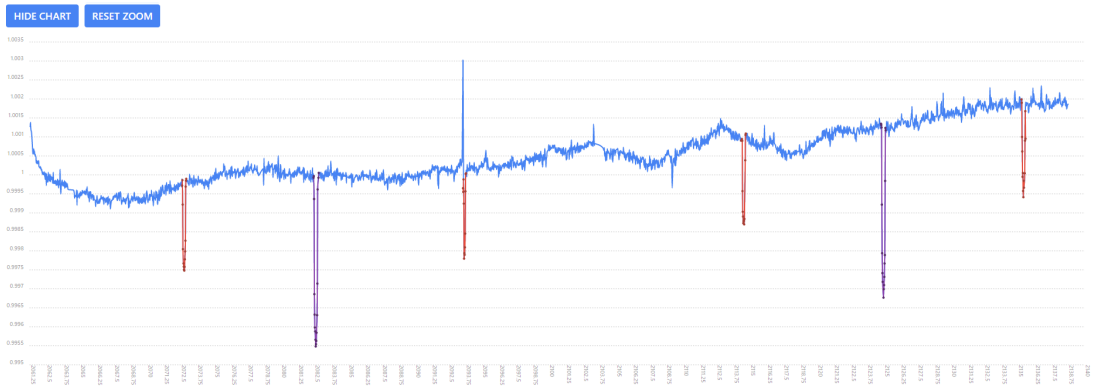

Figure 2.65: Light-curve of the evaluation.

2.7.4.2 Evaluation information

There are several rectangles on the detail page and each of them shows specific information from the evaluation. In this section every single one of them will be explained.

The first box is the star data, which displays all the important star properties with the appropriate units. (see Fig. 2.66)

The second is the evaluation data box, which displays the classification value, the

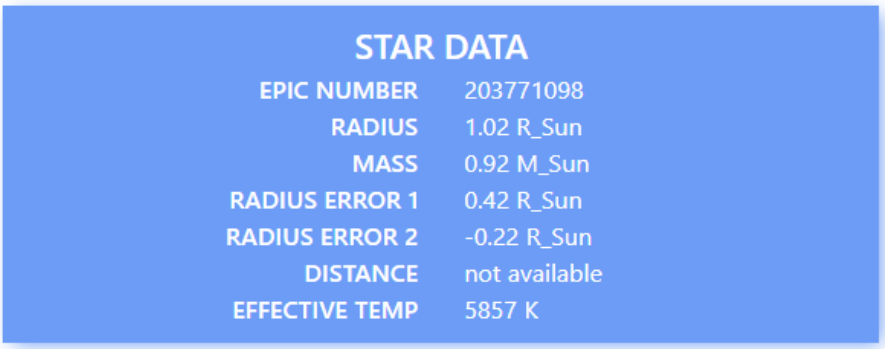

Figure 2.66: Detail page star data.

creation timestamp and the author of the evaluation. (see Fig. 2.67)

The third box is the candidate list box, which shows a little rectangle for every candi-

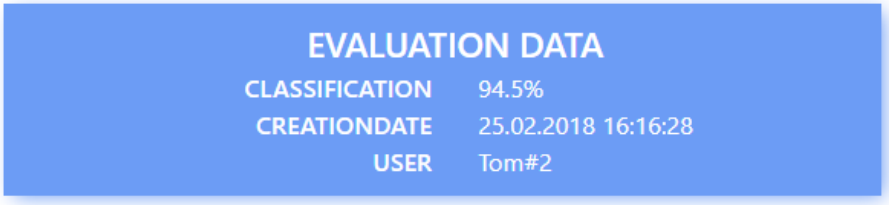

Figure 2.67: Detail page evaluation data.

date found in the light-curve. When clicking on these little rectangles the dips of the appropriate candidates will be shown in the light-curve and all the candidate associated data will be displayed in the candidate detail list on the right side of the screen. The candidate detail list shows all important candidate data, such as the calculated parameters. (see Fig. 2.68 and Fig. 2.69)

The fourth box shows the probability data of the evaluation. The probability data represents the number of evaluations of a star, the weighted average classification, the number of references, number of confirmed, false positive, candidate references and many more. (see Fig. 2.70)

The last box is the candidate reference data. This box shows all the referenced candidates and already confirmed exoplanets from the NASA database as well as the difference between them and the calculated evaluation candidates. (see Fig. 2.71)

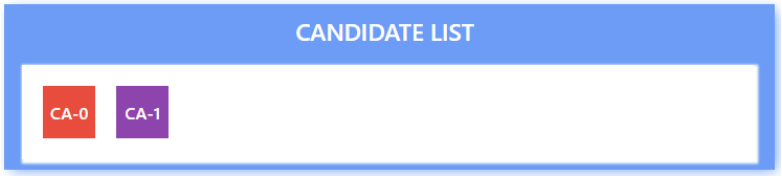

Figure 2.68: Detail page candidate list.

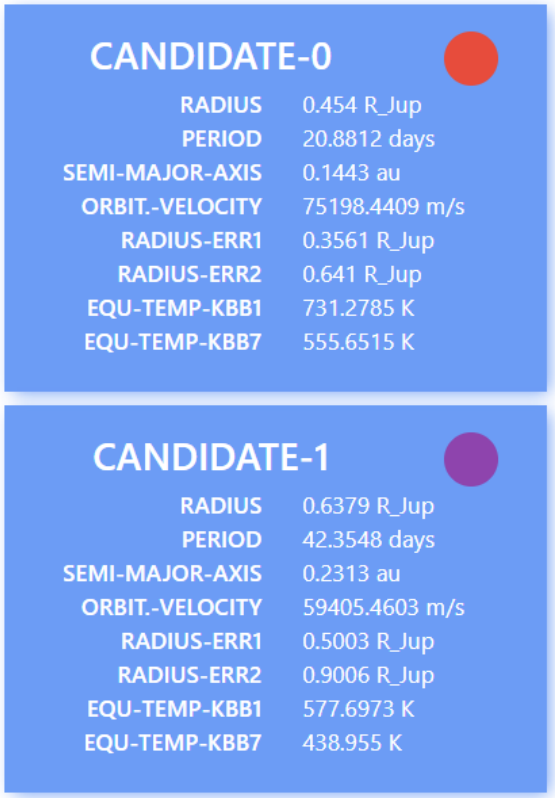

Figure 2.69: Detail page candidate detail list.

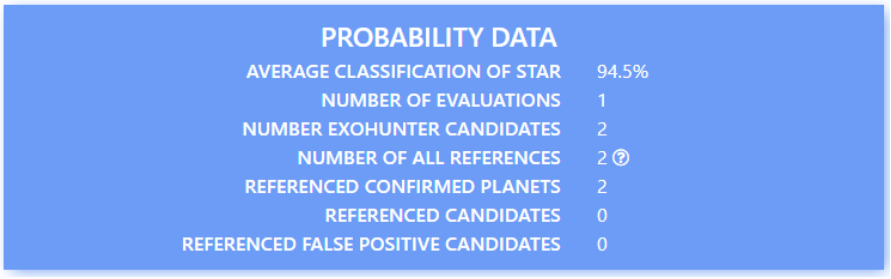

Figure 2.70: Detail page probability data.

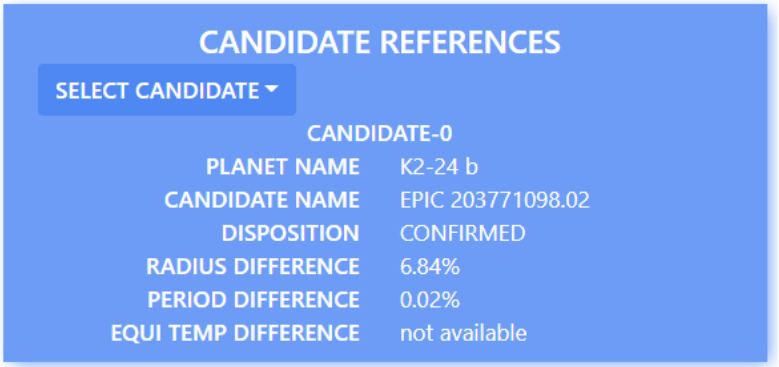

Figure 2.71: Detail page referenced candidate data.

Chapter 3

Attachments

3.1 Protocols

Meeting, 4th September, 2017
10:00 - 10:30

Participants:

- Prof. DI Herbert Lackinger
- Julian Hautzmayer
- Bernhard Reinsprecht

The first thing being discussed was the necessity of the server in the system architecture. The idea was to download all data directly from the NASA APIs on the client side, in order to avoid excessive bandwidth. Another aspect that was treated was the analysis, whose task is to find dips in a light curve. The analysis should be able to calculate period, radius, velocity, distance to the star, temperature and transit duration of a planet.

Prof. Lackinger wanted to know about the performance of the neural network and the analysis and if it was realistic, that a planet scan only takes about 30 seconds. Another suggestion was the implementation of a batch-mode, which would be able to autonomously scan for exoplanets without any user interaction.

Meeting, 27th September, 2017
17:00 - 17:10

Participants:

- Prof. DI Herbert Lackinger
- Julian Hautzmayer
- Bernhard Reinsprecht

During this short meeting, a summary of all finished tasks was given and prof. Lackinger suggested the use of the TensorBoard for the presentation.

Meeting, 21st March, 2018
16:00 - 16:30

Participants:

- Prof. DI Herbert Lackinger
- Julian Hautzmayer
- Bernhard Reinsprecht

The core topic of this meeting was the discussion about the presentation of ExoHunter for the ProjectAward. Prof. Lackinger suggested to focus more on the core features of our system to keep the presentation short. The website should not be shown, neither as live demo nor as video. Another idea was the usage of dark backgrounds on our slides, maybe even astronomical pictures like stars. In contrast, a bright, white text should be used. The usage of vector images is better avoided, otherwise it looks too childish. Prof. Lackinger went into detail about each slide and suggested to mention alien life on our last slide, in order to make it more interesting.

Bibliography

[ACC+13] R. L. Akeson, X. Chen, D. Ciardi, M. Crane, J. Good, M. Harbut, E. Jackson, S. R. Kane, A. C. Laity, S. Leifer, M. Lynn, D. L. McElroy, M. Papin, P. Plavchan, S. V. Ramirez, R. Rey, K. von Braun, M. Wittman, M. Abajian, B. Ali, C. Beichman, A. Beekley, G. B. Berriman, S. Berukoff, G. Bryden, B. Chan, S. Groom, C. Lau, A. N. Payne, M. Regelson, M. Saucedo, M. Schmitz, J. Stauffer, P. Wyatt, and A. Zhang. The NASA Exoplanet Archive: Data and Tools for Exoplanet Research. *Publications of the Astronomical Society of the Pacific*, 125(930):989–999, August 2013. URL: http://arxiv.org/abs/1307.2944, doi:10.1086/672273.

[Adm15] NASA Content Administrator. *The Milky Way's 100 Billion Planets*. April 2015. URL: http://www.nasa.gov/multimedia/imagegallery/image_feature_2233.html.

[APPE09] Cesare Alippi, Marios Polycarpou, Christos Panayiotou, and Georgios Ellinas. *Artificial Neural Networks – ICANN 2009: 19th International Conference, Limassol, Cyprus, September 14-17, 2009, Proceedings*. Springer Science & Business Media, September 2009.

[Bar] Geert Barentsen. *NASA - Kepler/K2 Guest Observer Program*. URL: ./pages/kepler-and-k2-data-processing-pipeline.html.

[Bre] Pat Brennan. *Exoplanets 101*. URL: https://exoplanets.nasa.gov/the-search-for-life/exoplanets-101.

[Bri15a] Denny Britz. *Recurrent Neural Network Tutorial, Part 4 – Implementing a GRU/LSTM RNN with Python and Theano*. October 2015. URL: http://www.wildml.com/2015/10/recurrent-neural-network-tutorial-part-4-implementing-a-grulstm-rnn-with-pytho

[Bri15b] Denny Britz. *Recurrent Neural Networks Tutorial, Part 1 – Introduction to RNNs*. September 2015. URL: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/.

[Bri15c] Denny Britz. *Recurrent Neural Networks Tutorial, Part 2 – Implementing a RNN with Python, Numpy and Theano*.

September 2015. URL: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-2-implementing-a-language-model-rnn-wi

[Bri15d] Denny Britz. *Recurrent Neural Networks Tutorial, Part 3 – Backpropagation Through Time and Vanishing Gradients*. October 2015. URL: http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-van

[Bri17] Denny Britz. *tf-rnn: Practical Examples for RNNs in Tensorflow*. July 2017. URL: https://github.com/dennybritz/tf-rnn.

[Bro16a] Jason Brownlee. *How to Normalize and Standardize Time Series Data in Python*. December 2016. URL: https://machinelearningmastery.com/normalize-standardize-time-series-data-python/.

[Bro16b] Jason Brownlee. *How to Use and Remove Trend Information from Time Series Data in Python*. December 2016. URL: https://machinelearningmastery.com/time-series-trends-in-python/.

[Bro16c] Jason Brownlee. *Supervised and Unsupervised Machine Learning Algorithms*. March 2016. URL: https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/.

[CAE] Calla Cofield, Space com Senior Writer \textbar April 21, and 2016 10:57pm ET. *What Are Pulsars?* URL: https://www.space.com/32661-pulsars.html.

[Car] Daniel Smilkov and Shan Carter. *Tensorflow — Neural Network Playground*. URL: http://playground.tensorflow.org.

[com] 09 Nov 2017 Amy Unruh Feed 71up 2 comments. *What is the TensorFlow machine intelligence platform?* URL: https://opensource.com/article/17/11/intro-tensorflow.

[Del16] Carlos Delgado. How to execute a Python script and retrieve output (data and errors) in Node.js, October 2016. URL: https://ourcodeworld.com/articles/read/286/how-to-execute-a-python-script-and-retrieve-output-data-and-errors-in-node-js.

[Des] Adit Deshpande. *A Beginner's Guide To Understanding Convolutional Neural Networks*. URL: https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/.

[GA16] Bui Cong Giao and Duong Tuan Anh. Similarity search for numerous patterns over multiple time series streams under dynamic time warping which supports data normalization. *Vietnam Journal of Computer Science*, 3(3):181–196, August 2016. URL: https://link.springer.com/article/10.1007/s40595-016-0062-4, doi:10.1007/s40595-016-0062-4.

[Joh15a] Michele Johnson. Comet Siding Spring Passes Through K2's Field-of-View, April 2015. URL: http://www.nasa.gov/ames/kepler/comet-siding-spring-passes-through-k2s-field-of-view.

[Joh15b] Michele Johnson. Mission overview, April 2015. URL: http://www.nasa.gov/mission_pages/kepler/overview/index.html.

[Kun18] Gion Kunz. *chartist-js: Simple responsive charts*. February 2018. URL: https://github.com/gionkunz/chartist-js.

[noaa] URL: http://www.astro.princeton.edu/strauss/FRS113/writeup3/.

[noab] *11.2. Dynamic Time Warping*. URL: http://web.science.mq.edu.au/cassidy/comp449/html/ch11s02.html.

[noac] *12.7 - Pseudo Code \textbar STAT 897D*. URL: https://onlinecourses.science.psu.edu/stat857/node/108.

[noad] *(384) How DTW (Dynamic Time Warping) algorithm works - YouTube*. URL: https://www.youtube.com/watch?v=_K1OsqCicBY&t=90s.

[noae] *3D Visualization of a Fully-Connected Neural Network*. URL: http://scs.ryerson.ca/aharley/vis/fc/.

[noaf] *5 Ways to Find a Planet*. URL: https://exoplanets.nasa.gov/interactable/11/.

[noag] *6 - Equilibrium Temperature*. URL: http://lasp.colorado.edu/bagenal/3720/CLASS6/6EquilibriumTemp.html.

[noah] *6.4.4.2. Stationarity*. URL: http://www.itl.nist.gov/div898/handbook/pmc/section4/pmc442.htm.

[noai] *artificial intelligence - Role of Bias in Neural Networks - Stack Overflow*. URL: https://stackoverflow.com/questions/2480650/role-of-bias-in-neural-networks.

[noaj] *AXA*. URL: http://brucegary.net/AXA/x.htm.

[noak] *Backpropagation*. URL: http://home.agh.edu.pl/vlsi/AI/backp_t_en/backprop.html.

[noal] Building a desktop application with Electron – Developers Writing – Medium. URL: https://medium.com/developers-writing/building-a-desktop-application-with-electron-204203eeb658.

[noam] *caesar0301/awesome-public-datasets: An awesome list of high-quality open datasets in public domains (on-going). By everyone, for everyone!* URL: https://github.com/caesar0301/awesome-public-datasets.

[noan] *chart chartist line*. URL: https://gionkunz.github.io/chartist-js/examples.html.

[noao] *chart clustering*. URL: https://sites.google.com/site/dataclusteringalgorithms/home.

[noap] *chart clustering bottom up*. URL: http://www.saedsayad.com/clustering_hierarchical.htm.

[noaq] *chart clustering k mean*. URL: https://www.youtube.com/watch?v=_aWzGGNrcic.

[noar] *chart clustering yes*. URL: https://www.datascience.com/blog/k-means-clustering.

[noas] *chart dtaw*. URL: https://www.youtube.com/watch?v=_K1OsqCicBY.

[noat] *chart exoplanet by orbit and radius*. URL: https://www.nasa.gov/content/kepler-multimedia.

[noau] *chart exoplanet finding methods*. URL: https://exoplanetarchive.ipac.caltech.edu/exoplanetplots/exo_dischist.png.

[noav] *chart heart beat time series*. URL: https://alcoholicsguidetoalcoholism.com/2015/05/04/journey-from-the-head-to-the-heart-and-back/.

[noaw] *chart impact parameter*. URL: https://blog.planethunters.org/2013/01/21/what-factors-impact-transit-shape/.

[noax] *chart planet full transit description*. URL: https://arxiv.org/abs/1001.2010v3.

[noay] *chart planet radii*. URL: https://blog.planethunters.org/2013/01/21/what-factors-impact-transit-shape/.

[noaz] *chart radii with limb*. URL: https://blog.planethunters.org/2013/01/21/what-factors-impact-transit-shape/.

[noa—] *chart time series trend*. URL: https://machinelearningmastery.com/time-series-trends-in-python/.

]noauthor$_c hart_n odate - 9 charttimeseriesznormno.URL$: .

Chartist - Examples. URL: https://gionkunz.github.io/chartist-js/examples.html.

chart_time_series_z_norm_yes. URL: https://jmotif.github.io/sax-vsm_site/morea/algorithm/znorm.html.

Clustering. URL: http://www.saedsayad.com/clustering.htm.

Clustering - Introduction. URL: https://home.deib.polimi.it/matteucc/Clustering/tutorial_html/.

Confirmed Planets. URL: https://exoplanetarchive.ipac.caltech.edu/cgi-bin/TblView/nph-tblView?app=ExoTbls&config=planets.

CS231n Convolutional Neural Networks for Visual Recognition. URL: http://cs231n.github.io/.

Data Clustering Algorithms. URL: https://sites.google.com/site/dataclusteringalgorithms/home.

Deep Learning. URL: http://www.deeplearningbook.org/.

Der Gradient. URL: http://statistik.wu-wien.ac.at/leydold/MOK/HTML/node123.html.

Die FFT mit Python einfach erklärt. URL: http://www.cbcity.de/die-fft-mit-python-einfach-erkl

Direct Imaging. URL: http://www.planetary.org/explore/space-topics/exoplanets/direct-imaging.html.

Distance between signals using dynamic time warping - MATLAB dtw - MathWorks Deutschland. URL: https://de.mathworks.com/help/signal/ref/dtw.html.

Dynamic time warping example. URL: http://www.phon.ox.ac.uk/jcoleman/old_SLP/Lecture_5/DTW_explanation.html.

Effective Temperature \textbar COSMOS. URL: http://astronomy.swin.edu.au/cosmos/E/Effective+Temperature.

Electron. URL: https://electron.atom.io/.

EPIC 211311380 (HIP 41378) \textbar scatter chart made by Andrew418 \textbar plotly. URL: https://plot.ly/andrew418/3/.

EPIC Search. URL: http://archive.stsci.edu/k2/epic/search.php.

ExoFOP. URL: https://exofop.ipac.caltech.edu/k2/.

Exoplanet Archive Planet Counts. URL: https://exoplanetarchive.ipac.caltech.edu/docs/counts_detail.html.

Exoplanet Hunting in Deep Space. URL: https://www.kaggle.com/keplersmachines/kepler-labelled-time-series-data.

Extract Seasonal & Trend: using decomposition in R - Anomaly. URL: https://anomaly.io/seasonal-trend-decomposition-in-r/.

The Extrasolar Planet Encyclopaedia — Catalog Listing. URL: http://exoplanet.eu/catalog/.

Getting Started - TFLearn. URL: http://tflearn.org/getting_started/.

HEC: Periodic Table of Exoplanets - Planetary Habitability Laboratory @ UPR Arecibo. URL: http://phl.upr.edu/projects/habitable-exoplanets-catalog/media/pte.

Hibernate. Everything data. - Hibernate. URL: http://hibernate.org/.

Hierarchical Clustering. URL: http://www.saedsayad.com/clustering_hierarchical.htm.

How is deep learning different from multilayer perceptron? - Quora. URL: https://www.quora.com/How-is-deep-learning-different-from-multilayer-perceptron.

img cnn. URL: https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/.

img equ temp. URL: http://lasp.colorado.edu/bagenal/3720/CLASS6/6EquilibriumTemp.html.

img neurons human brain. URL: https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_neural_networks.htm.

img supervised learning. URL: https://www.tutorialspoint.com/artificial_neural_network/images/supervised_learning.jpg.

Index of /pub/kepler/lightcurves. URL: http://archive.stsci.edu/pub/kepler/lightcurves/.

interpretation - What is the difference between "mean value" and "average"? - Cross Validated. URL: https://stats.stackexchange.com/questions/14089/what-is-the-difference-betwee

ISS Group at the University of Texas. URL: http://iss.ices.utexas.edu/?p=projects/galois/benchmarks/agglomerative_clustering.

JavaScript Live / Dynamic Charts & Graphs. URL: https://canvasjs.com/html5-javascript-dynamic-

K-Means. URL: http://www.saedsayad.com/clustering_kmeans.htm.

K2 Field Descriptions. URL: https://archive.stsci.edu/search_fields.php?mission=k2.

K2 Photometry. URL: https://www.cfa.harvard.edu/~avanderb/k2.html.

K2_planets Field Descriptions. URL: https://archive.stsci.edu/search_fields.php?mission=k2_planets.

K2's First Five-Planet System. URL: http://aasnova.org/2016/08/08/k2s-first-five-planet-syste

K2sff Search. URL: https://archive.stsci.edu/k2/hlsp/k2sff/search.php.

machine learning - What is the difference between test set and validation set? - Cross Validated. URL: https://stats.stackexchange.com/questions/19048/what-is-the-difference-betwee

MAST K2. URL: https://archive.stsci.edu/k2/.

MAST Web Services. URL: https://archive.stsci.edu/vo/mast_services.html.

Microlensing. URL: http://www.planetary.org/explore/space-topics/exoplanets/microlensing.html.

NASA Exoplanet Archive. URL: https://exoplanetarchive.ipac.caltech.edu/index.html.

NASA Exoplanet Archive. URL: https://exoplanetarchive.ipac.caltech.edu/.

NASA Exoplanet Archive: Exoplanet Transit Survey Service. URL: https://exoplanetarchive.ipac.caltech.edu/applications/ETSS/Kepler_index.html.

node.js - Run python script in Electron app - Stack Overflow. URL: https://stackoverflow.com/questions/41199981/run-python-script-in-electron-app.

normalization - How to normalize data to 0-1 range? - Cross Validated. URL: https://stats.stackexchange.com/questions/70801/how-to-normalize-data-to-0-1-range.

Open Data. URL: https://open.nasa.gov/open-data/.

Outlier Detection in Time-Series Signals using FFT and Median Filtering \textbar Bugra Akyildiz. URL: https://bugra.github.io/work/notes/2014-03-31/outlier-detection-in-time-ser

Piecewise Aggregate Approximation (PAA) \textbar SAX-VSM. URL: https://jmotif.github.io/sax-vsm_site/morea/algorithm/PAA.html.

Python Programming Tutorials. URL: https://pythonprogramming.net/.

Semi-Major Axis Calculator. URL: http://orbitsimulator.com/gravity/articles/smaCalculator.html.

Spectral Leakage \textbar Bugra Akyildiz. URL: http://bugra.github.io/work/notes/2012-09-15/Spectral-Leakage/.

Symbolic Aggregate Approximation (SAX) \textbar SAX-VSM. URL: https://jmotif.github.io/sax-vsm_site/morea/algorithm/SAX.html.

Tangential- und Winkelgeschwindigkeit. URL: https://elearning.physik.uni-frankfurt.de/data/FB13-PhysikOnline/lm_data/lm_282/auto/kap07/cd179.htm.

TFLearn \textbar TensorFlow Deep Learning Library. URL: http://tflearn.org/.

Time Series Classification and Clustering with Python. URL: http://alexminnaar.com/time-series-classification-and-clustering-with-python.html.

Time Series Clustering and Classification - RDataMining.com: R and Data Mining. URL: http://www.rdatamining.com/examples/time-series-clustering-classification.

Transit Light Curve Tutorial. URL: https://www.cfa.harvard.edu/avanderb/tutorial/tutorial.html.

Understanding LSTM Networks – colah's blog. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Unsupervised Feature Learning and Deep Learning Tutorial. URL: http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/.

Using the Application Programming Interface (API). URL: https://exoplanetarchive.ipac.caltech.edu/docs/program_interfaces.html.

What is a REST API? | Document REST APIs. URL: https://idratherbewriting.com/learnapidoc/docapis_what_is_a_rest_api.html.

What is bias in artificial neural network? - Quora. URL: https://www.quora.com/What-is-bias-in-artificial-neural-network.

What is meant by feature maps in convolutional neural networks? - Quora. URL: https://www.quora.com/What-is-meant-by-feature-maps-in-convolutional-neural-networks.

What is TensorFlow? - Quora. URL: https://www.quora.com/What-is-TensorFlow-1.

Z-normalization \textbar SAX-VSM. URL: https://jmotif.github.io/sax-vsm_site/morea/algorithm/znorm.html.

Orbital Periods of the Planets. June 2014. URL: https://space-facts.com/orbital-periods-planets/

Jupiter radius. December 2016. URL: https://en.wikipedia.org/w/index.php?title=Jupiter_radius&oldid=755105092.

An overview of gradient descent optimization algorithms. January 2016. URL: u=http://ruder.io/optimizing-gradient-descent/.

Earth radius. September 2017. URL: https://en.wikipedia.org/w/index.php?title=Earth_radius&oldid=798370959.

Euclidean distance. August 2017. URL: https://en.wikipedia.org/w/index.php?title=Euclidean_distance&oldid=798215348.

The First Exoplanet Was Discovered 25 Years Ago Today. January 2017. URL: https://futurism.com/the-first-exoplanet-was-discovered-25-years-ago-today/.

Gravitationskonstante. January 2017. URL: https://de.wikipedia.org/w/index.php?title=Gravitationskonstante&oldid=161557545.

How to Create Convolutional Neural Networks Using Java and DL4J. March 2017. URL: http://progur.com/2017/03/how-to-create-convolutional-neural-networks-java-dl4j.html.

Parsec. September 2017. URL: https://en.wikipedia.org/w/index.php?title=Parsec&oldid=799462650.

Astronomical unit. February 2018. URL: https://en.wikipedia.org/w/index.php?title=Astronomical_unit&oldid=825718354.

Cluster analysis. January 2018. URL: https://en.wikipedia.org/w/index.php?title=Cluster_analysis&oldid=821127746.

Dynamic time warping. January 2018. URL: https://en.wikipedia.org/w/index.php?title=Dynamic_time_warping&oldid=822466906.

Gravitational constant. February 2018. URL: https://en.wikipedia.org/w/index.php?title=Gravitational_constant&oldid=825688516.

Methods of detecting exoplanets. January 2018. URL: https://en.wikipedia.org/w/index.php?title=Methods_of_detecting_exoplanets&oldid=822113186.

Planetary equilibrium temperature. January 2018. URL: https://en.wikipedia.org/w/index.php?title=Planetary_equilibrium_temperature&oldid=822465263.

Sebastian Raschka. *python-machine-learning-book: The "Python Machine Learning" book code repository and info resource*. June 2017. URL: https://github.com/rasbt/python-machine-learning-book.

Brian Resnick. *NASA has discovered 7 Earth-like planets orbiting a star just 40 light-years away.* February 2017. URL: https://www.vox.com/2017/2/22/14698030/nasa-seven-exoplanet-discovery-trappist-1.

rpantaleo. *Kepler Discovers More Exoplanets / Scientists Classify Types of Planets – Science World.* URL: https://blogs.voanews.com/science-world/2017/06/22/kepler-discovers-more-exoplanets-scientists-classify-types-of-planets/.

Scott. *Detrending Data in Python with Numpy.* June 2010. URL: https://www.swharden.com/wp/2010-06-24-detrending-data-in-python-with-numpy/.

Siraj Raval. *The Best Way to Prepare a Dataset Easily.* URL: https://www.youtube.com/watch?v=0xVqLJe9_CY.

Investopedia Staff. *Time Series.* March 2006. URL: https://www.investopedia.com/terms/t/timeseries.asp.

Stephanie. *Detrend Data.* URL: http://www.statisticshowto.com/detrend-data/.

Thales Sehn Körting. *How DTW (Dynamic Time Warping) algorithm works.* URL: https://www.youtube.com/watch?v=_K1OsqCicBY.

tutorialspoint.com. *Artificial Intelligence Neural Networks.* URL: https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_neural_networks.htm.

ujjwalkarn. *A Quick Introduction to Neural Networks.* August 2016. URL: https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/.

Victor Lavrenko. *K-means clustering: how it works.* URL: https://www.youtube.com/watch?v=_aWzGGNrcic.

List of Figures

1.1 Distribution of confirmed exoplanets by year and finding method. 5
1.2 Light-curve of the star EPIC 211311380 (HIP 41378). 6
1.3 Comparision of different planet radii. 9
1.4 Detailed graphic of a transit. 10
1.5 Different values of the impact parameter. 11
1.6 Different planet radii with the effect of Limb Darkening. 11
1.7 Confirmed exoplanets with radii and period. 14
1.8 Equilibrium temperature simple explained. 16
1.9 Cluster observed by the Kepler Telescope for the duration of one campaign. 17
1.10 Structure of human neurons. 22
1.11 Structure of a simple Neural Network. 22
1.12 Diagramm of supervised learning. 23
1.13 Process of a CNN. 25
1.14 Diagramm of TensorFlow, TFLearn and Keras. 27
1.15 Time series describing the heart beat of a patient. 30
1.16 Time series with and without a linear trend. 32
1.17 Time series with and without Z-normalization. 34
1.18 Simple clustering example. 36
1.19 Hierachical clustering algorithms. 37
1.20 The single steps of a K-mean clustering algorithm. 38
1.21 Dynamic time warping algorithm example. 39
1.22 Electron's main process and multiple renderer processes 41
1.23 Example of a canvas.js dynamic chart. 42
1.24 Available streams for communication between processes and client 43
1.25 Technologies used of the Java server 44
1.26 The Hibernate framework is an Object-Relational-Mapper 47
1.27 Structure of a REST API 48
1.28 Steps of downloading a light curve 49
1.29 A simple line chart with Chartis. 50

2.1 Diagram of the system architecture. 52
2.2 Diagram of the system's entity relationship diagram. 58
2.3 UCD of the electron client. 59

2.4 UCD of the website. 60
2.5 Data preparation, training and testing the model. 62
2.6 Each step of the data preparation section for the CNN. 63
2.7 Python implementation of positive outlier detection. 64
2.8 Python implementation of Z-normalization. 65
2.9 Adust range to [-1, 1]. 65
2.10 Padding light-curve. 66
2.11 Structure of the CNN. 67
2.12 Python and Keras implementation of the model. 69
2.13 Training the CNN. 70
2.14 Testing the CNN. 70
2.15 Prepare single light-curve. 71
2.16 Test single light-curve with CNN. 71
2.17 mode selection on the start screen 72
2.18 chart after entering a star id in manual mode 72
2.19 results are displayed after analysis is complete 73
2.20 settings for the batch mode 75
2.21 example for a batch iteration where nothing was found 75
2.22 example report after finishing batch mode 76
2.23 fetching light curves with a star id 77
2.24 values are prepared for the CNN after downloading the light curve 78
2.25 further action is decided by the exoplanet probability 79
2.26 fetching of star meta information 80
2.27 running the analysis process 81
2.28 the computed evaluation gets forwarded to the server 82
2.29 handling of process streams in electron 83
2.30 Interface for user adjustments 84
2.31 saving process of adjustments after a user is satisfied with the new result 85
2.32 the new custom results get sent to the server 86
2.33 a simple logger for the electron client 87
2.34 The different parts of the Time Series Analysis. 88
2.35 Light-curve detrending algorithm. 90
2.36 Light-curve detrending algorithm. 90
2.37 Light-curve noise reduction algorithm. 91
2.38 Algorithm for creating an inclination map. 91
2.39 Calculate dip parameters. 93
2.40 Dip clustering algorithm. 95
2.41 Dynamic time warping algorithm. 96
2.42 Calculating the orbital period. 96
2.43 Calculating the radius. 97
2.44 Calculating the semi-major axis. 97
2.45 Calculating the orbital velocity. 98
2.46 Calculating the equilibrium temperature. 98

2.47 Background thread that refills the cache . . . 101
2.48 Implementation of epic number fetching . . . 102
2.49 Java implementation of method addSingleComputed. . . . 103
2.50 Java implementation of method addSingleCustome. . . . 104
2.51 Java implementation of method updateExoplanetStat. . . . 105
2.52 Java implementation of the re-traing check. . . . 107
2.53 Java implementation of method calcPropability. . . . 108
2.54 The banner of the front-page. . . . 109
2.55 The brief project description. . . . 110
2.56 The most important technologies and goals of Exohunter. . . . 110
2.57 Short description of an exoplanet. . . . 110
2.58 Explanation of the transit method. . . . 111
2.59 Load all planetary statistics for the website. . . . 112
2.60 Count of confirmed exoplanets. . . . 112
2.61 Count of confirmed exoplanets per year. . . . 113
2.62 Creation of the bar chart with Chartist. . . . 113
2.63 Distribution of exoplanet finding methods. . . . 114
2.64 Evaluation list view. . . . 115
2.65 Light-curve of the evaluation. . . . 115
2.66 Detail page star data. . . . 116
2.67 Detail page evaluation data. . . . 116
2.68 Detail page candidate list. . . . 117
2.69 Detail page candidate detail list. . . . 117
2.70 Detail page probability data. . . . 117
2.71 Detail page referenced candidate data. . . . 118