FINAL REPORT PERSONAL CHALLENGE ARTIFICIAL INTELLIGENCE

EXOPLANET DETECTION

BY MARTIN KIRYAKOV

STUDENT #: 3780090

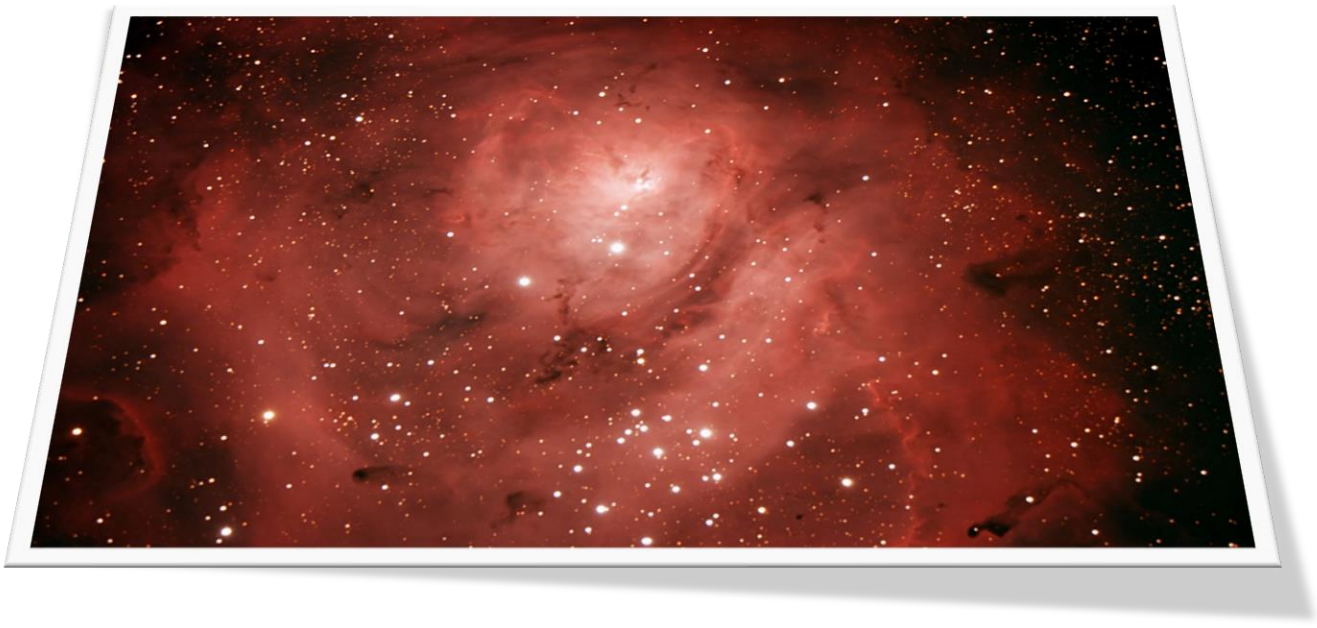
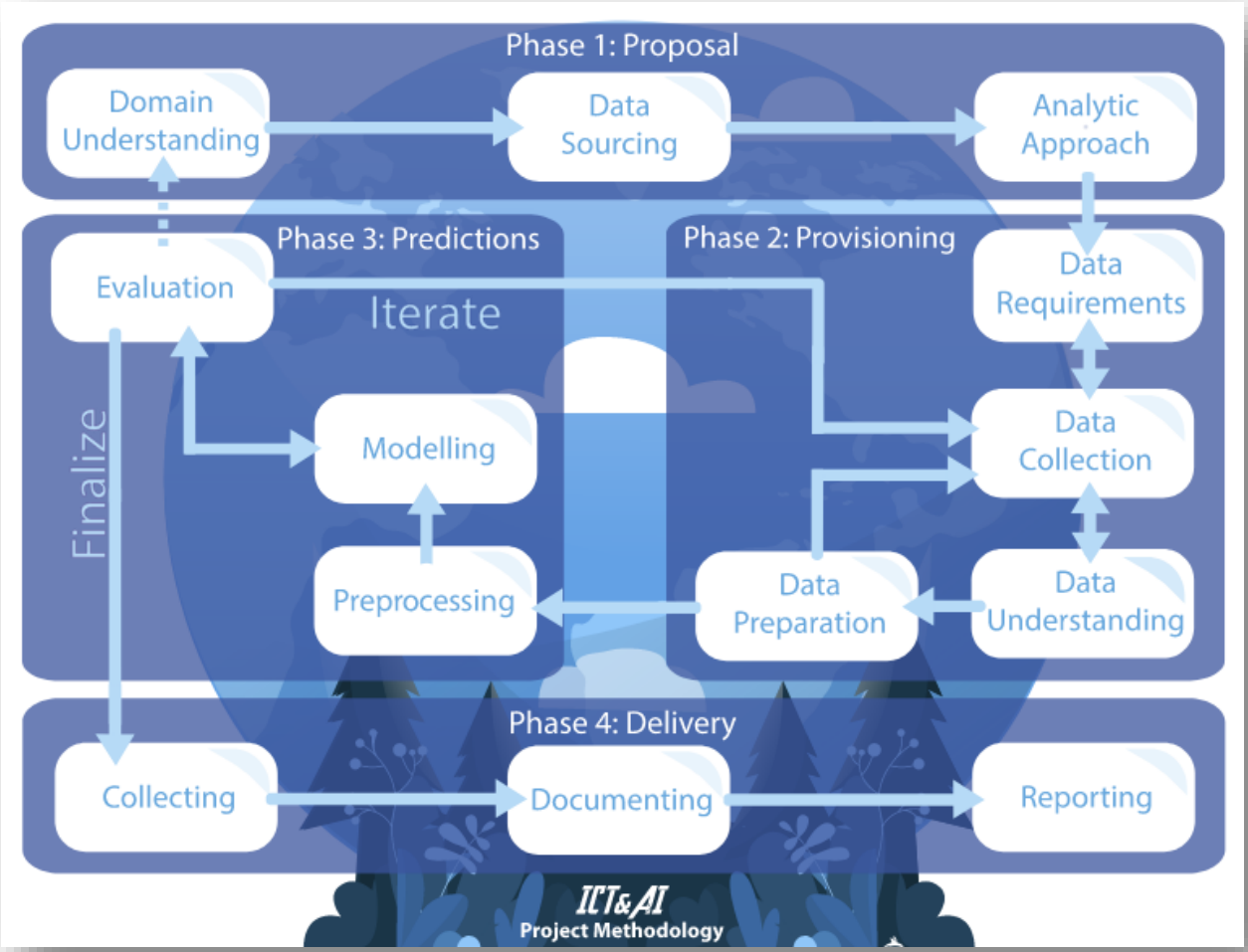


Table of Contents

1. Project description and understanding

Over the course of this semester my understanding of this project changed dramatically. It evolved from a mere classification problem to a full scale analysis with an iterative cycle of work, much akin to the "AI methodology" presented by the Fontys graphic.



I'll be honest, in the beginning I viewed this as just another graph, one that doesn't really correlate with the actual, real-world application of an AI project. I couldn't have been more wrong. Working on my personal challenge has made me realize just how accurate this flowchart actually is. I've went through every single stage listed here, and can confidently point to the how, why and when of every step, which is exactly what I aim to do with this document. I will explore the stages of the project and try to explain my thought process along the way.

Note: This document is best read alongside the Jupyter notebook of this project.

2. Initial data collection and domain understanding

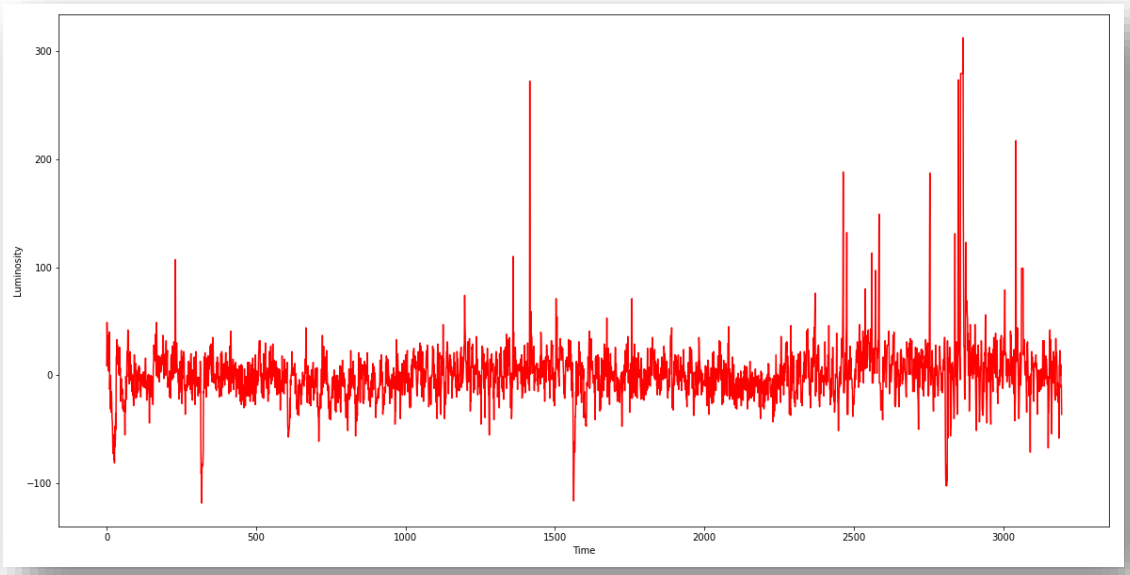
In the beginning, I struggled a lot with choosing a topic and dataset for this project. I knew that I wanted something to do with astronomy, as it's a topic that I'm very enthusiastic about, so I searched far and wide for different datasets, including asteroids, meteorite impacts, solar system moons, and more. None of these really hit the mark though, until I found the Kepler space telescope exoplanet dataset.

I immediately knew that this would be the dataset for me. That said, I also knew the challenges that would come with it, especially with it's bizarre shape – 5087 rows and 3198 columns for the train set. This proved to be a problem on several occasions.

The first step with any AI project is to understand the data you're dealing with, both in a literal sense, and in a societal impact perspective. The latter already has it's own document, so lets tackle the former.

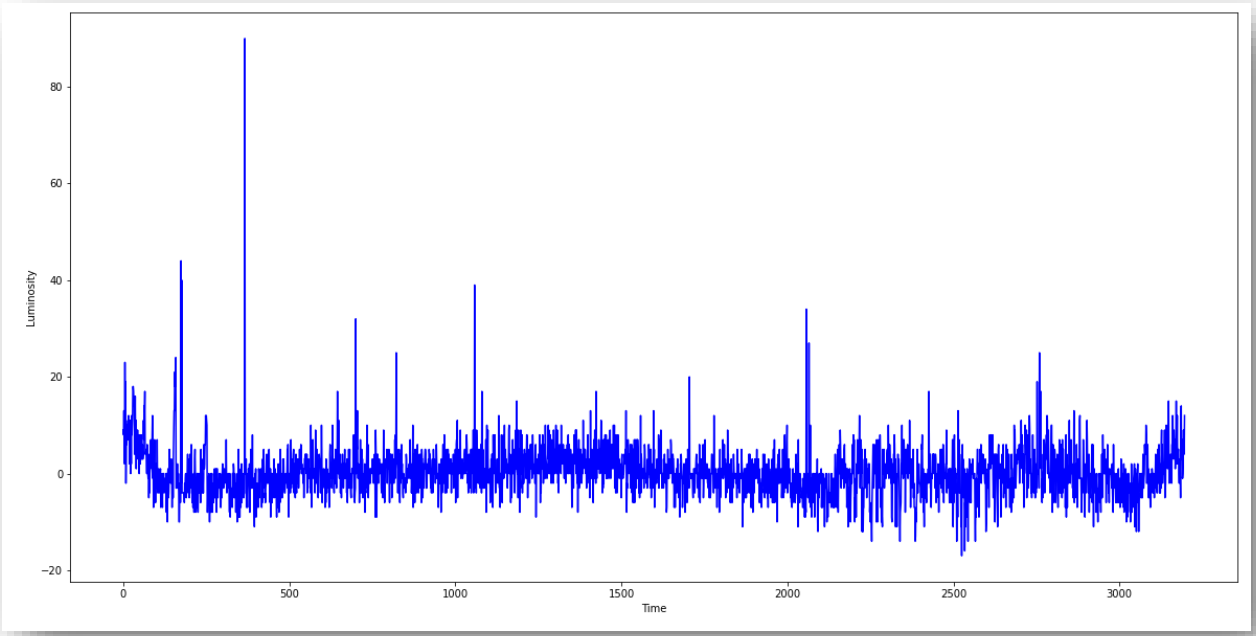
The way this dataset needs to be treated as, is a collection of time series. These time series represent a period of 80 days, in which the Kepler space telescope has observed a distant star and collected readings from it. Each observation features a label, followed by 3197 FLUX readings. The label is simple – a value of 2 means that the observed star has an exoplanet orbiting around it, while a value of 1 means the opposite. The FLUX readings are a bit more complex. They each represent the quantity of light emitted from an observed star, i.e., its luminosity. These readings can fluctuate drastically, depending on what is happening in the distant solar system. Space debris such as gas clouds, asteroid belts, or even binary systems can greatly affect the luminosity of an observed star, and this is what we aim to classify.

If a planet passes in front of its parent star in such a way that its in-line with the Kepler telescope, it will cause a short, noticeable decrease in the perceived FLUX value. Let's take a look at one of the graphs I generated from the raw data of star 6:



We can see three clear downward spikes in the reading. Additionally, they are all spaced at a very equal interval, around 1225 readings. Using some math (described in greater detail in the notebook), we can deduce that the orbital period of this body is right around 30 Earth days. This is insanely quick. Mercury, the fastest orbiting planet in our solar system has an orbital period of 88 days.

Now let's take a look at another graph – one from a star without an exoplanet.

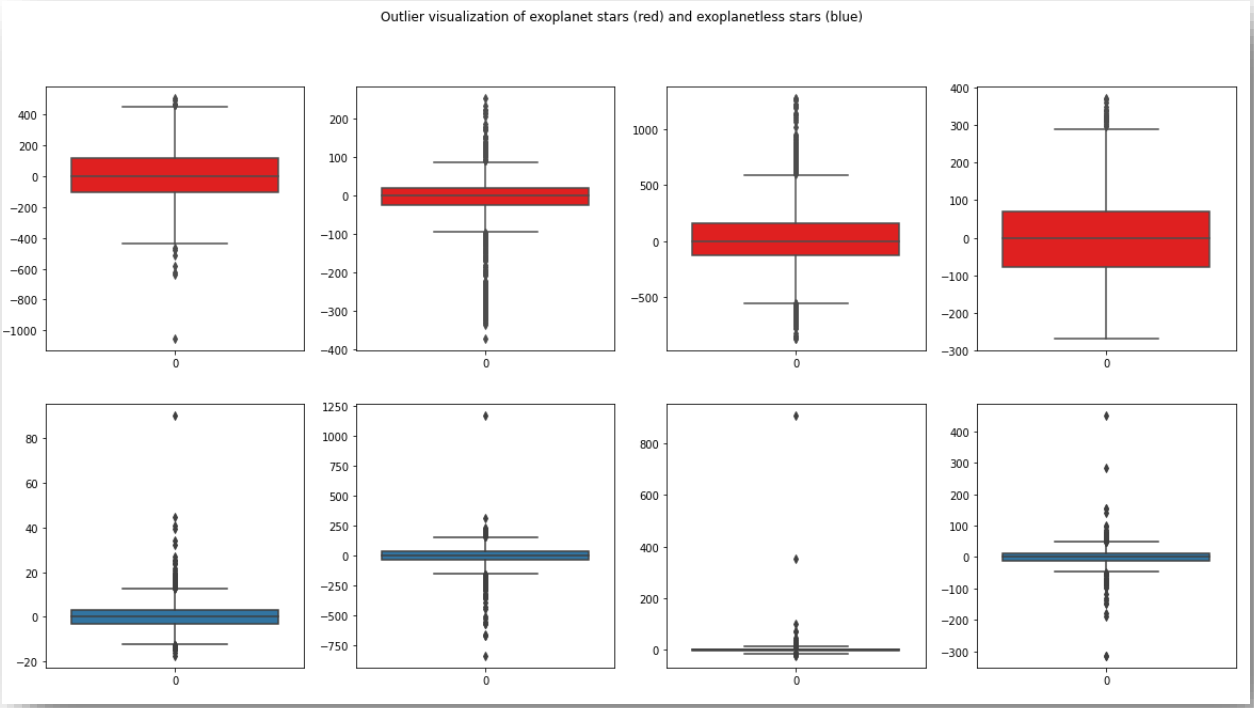


We see a much different shape here. There are no downward spikes, and the graph seems to have a very aperiodic, trendless nature. This key difference is what I attempted to quantify and classify in my project.

3. Data preparation

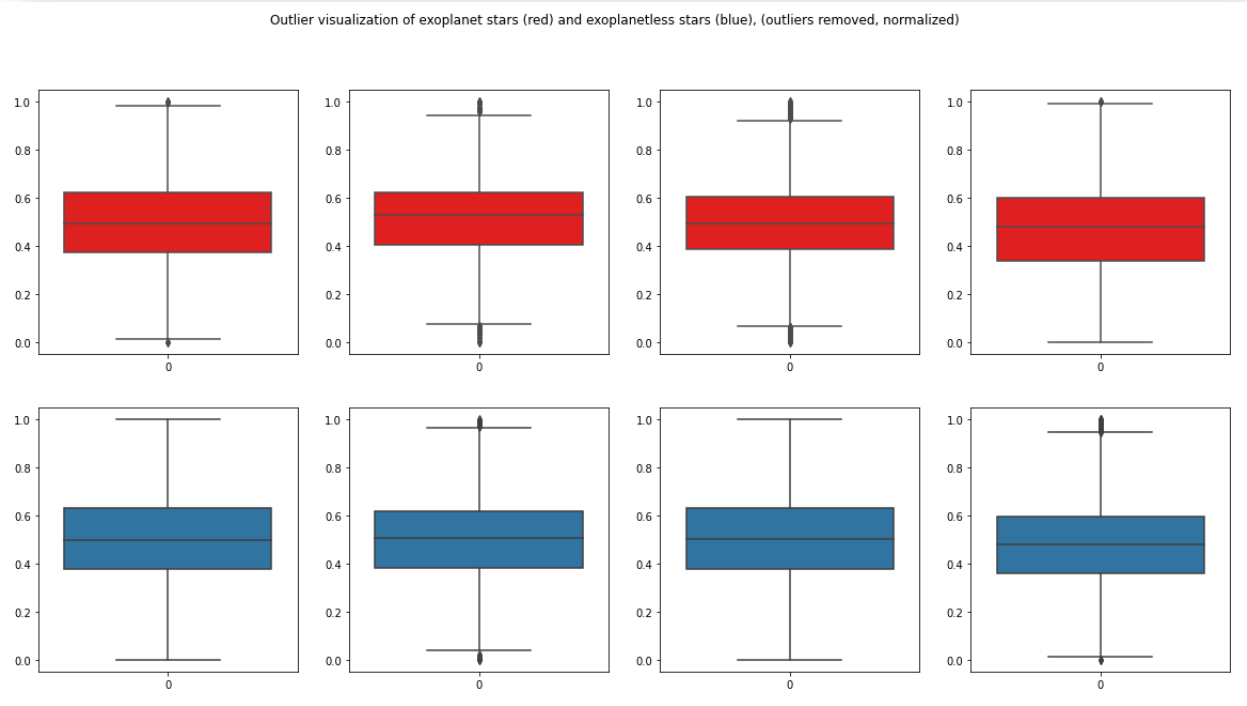
Now that I had a general idea of what my dataset represents, it was time to prepare it for machine learning. The dataset was already relatively clean, it had no missing values. The only real thing that separated it from a truly ML-ready set was its lack of normalization, as was also stated on the download page.

I started by using seaborn to get some boxplots of 4 stars from each label value:



As we can see, there are a lot of issues here. As mentioned before, the lack of normalization means that there is no common y axis for all of these. Additionally, there are quite a few outliers here.

During my initial round of outlier removal, I ran a fairly standard IQR algorithm. However, in doing so, I actually removed some key points of data. This is because the very thing we're looking for is outliers – large downward spikes of FLUX readings that represent a very small portion of all data. I decided with going for a much less strict outlier removal pattern, and after running a normalization algorithm I ended up with this:



As you can see, there are still a few outliers here and there, but I believe that its for the best, as these are crucial for ML performance.

Another problem that I ran into here was the aforementioned ‘weirdness’ of this dataset. Because the rows should be treated as time series, the columns don’t really mean much. The problem arises when one realizes that most python libraries for data manipulation are created around columns, since most datasets use them for machine learning. Here, when I first ran my scaler, I got some very bizarre values, because it was interpreting the columns as all part of the same variable, when in fact they had nothing to do with one another as they were readings from different stars. I solved this by transposing the entire dataframe, performing the operation, and then transposing it again to return it to its original shape.

This was of course only the first and most basic rounds of data preparation. In the later stages of this project, I realized that this was still not quite enough for a good ML performance, so through iteration I further improved the dataset.

4. First round of machine learning

I settled on two algorithms – nearest neighbors and decision trees. I tried a few others, such as linear regression and SVMs, but I received very limited performance from them.

During this initial run, I got an accuracy of 99.1%. This of course, proved to be a common mistake. Because we were dealing with very imbalanced datasets, both for training and testing, any algorithm we used was going to be severely biased towards the majority class, in this case exoplanet negative stars. This is exactly what I observed – my algorithms had simply classified everything as exoplanet negative, and the accuracy score showed this. I developed my own function, which would only focus on the 5 exoplanet positive stars in the test set. I later also found out about a built-in function, called balanced_accuracy_score, which aims specifically at solving this problem. Using these, I understood that what I had done was practically worthless.

I got around to solving this imbalance issue.

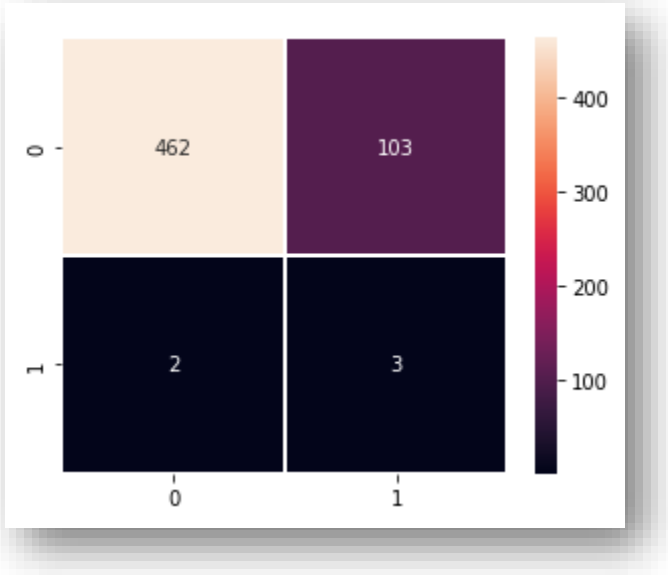
4.1 Imbalance solution

After much research, I settled on using SMOTE, which is a popular oversampling tool. But what is oversampling? It is a method of generating new data for a minority class, so that it may become equal in size to a majority class, thus leading to balanced ML. It does this, by looking at the existing data and shifting it around in such a way as to generate new observations, without actually inducing any new or interpolated data which may affect the outcome of machine learning.

I applied this process to the training set, while leaving the test set untouched, which was also vital in ensuring the quality of prediction data.

5. Second round of machine learning

This time around I got much more promising results. Below is a confusion matrix of the better of the two predictions, which was the one made by the decision trees classifier.



We can see that of the 5 exoplanet stars, 3 were predicted correctly, while 2 weren't. Additionally, there are 103 exoplanet negative star which the algorithm predicted as exoplanet positive – these are our false positives.

Overall, the accuracy here is not great. The algorithm's behavior now seems erratic, almost random. I looked at some of the false positives, as well as the false negatives and could still see similarities, meaning that my algorithm wasn't really trained well and was basically 'guessing'. This led me to the final trick up my sleeve – frequency analysis.

6. Frequency analysis

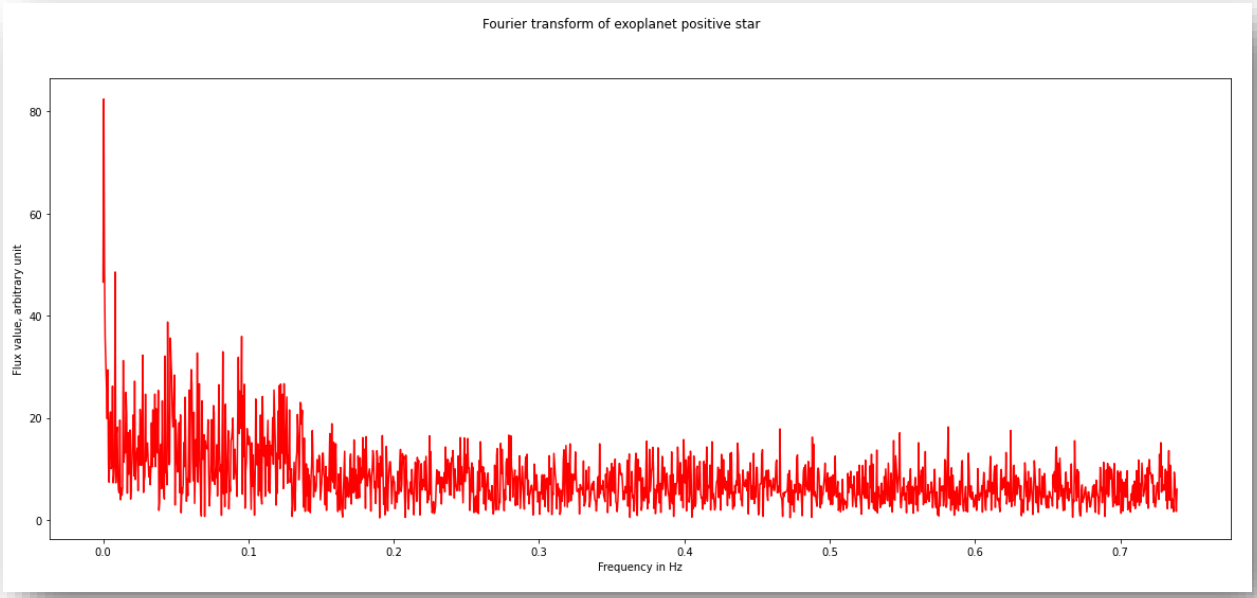
Frequency analysis is what I had aimed for all along with this project, without even knowing it. The subtitle of my notebook that I came up with in the beginning reads:

"Quantification of the coefficient of periodicity in a time series"

Let's break this down. The coefficient of periodicity is simply the answer to "How periodic are the events in this time series?". For example, a clock has an extremely high coefficient of periodicity. Its minute hand has a period of 60 minutes, its hour hand – a period of 12 hours. On the other end is something like simple white noise, or static. It has a coefficient value of near zero, as it is almost completely random, and it doesn't loop or repeat in a predictable manner. To quantify such a coefficient simply means to put it into a language that a machine learning algorithm can understand.

I again went through a lot of extensive research on this topic. There was a surprisingly low amount of information regarding this subject, but I eventually found Fourier transforms. A Fourier transform is a method of deconstructing an aperiodic function down to its basic, sinusoidal components. Essentially, this means that it I can generate information about which frequencies are overrepresented in my data. This was useful, because lower frequencies can generally be attributed to periodic, orbital behavior, whereas higher frequencies are tied to noise and interference in the measurement, allowing us to filter them out.

After many hours of trial and error, I was able to generate a full Fourier transform of all our stars. Below you will see an example of one of these transforms, done on star 6, which is the same star as the example above.

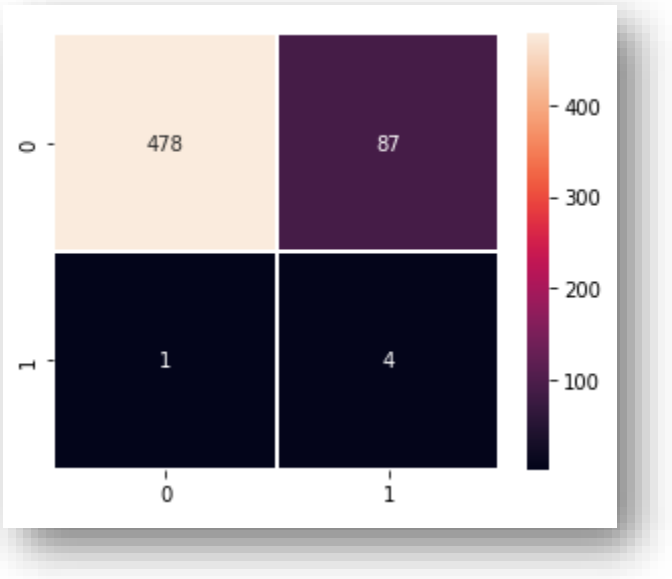


We can see that the readings in the lower frequency ranges are indeed a bit higher, possibly relating to the exoplanet in orbit around this star. Now it was time to put this new data to good use.

7. Third round of machine learning

I now had a powerful tool to quantify the periodicity of my stars. I first ran my algorithms only on the Fourier transforms. In both cases, the number of false positives dropped by 37-47%, which was a huge improvement. In the decision trees algorithm, this also came with a significant hit on true positive accuracy however – I now only detected one of the five exoplanet positive stars.

I decided to combine both datasets, so that each row would consist of the 3197 raw FLUX observations, appended by their Fourier transform. This gave my algorithms the best possible chance to train well, as it provided them with the largest quantity of data. I again saw some marginal improvements in both algorithms' predictions. I also tweaked their parameters in order to get as much performance as possible. Finally, I decided to combine both algorithms' predictions, in order to get the largest sample size possible, leading to this final confusion matrix:



I now had successfully predicted 4 of the 5 exoplanet positive stars, while limiting the false positives to 87.

8. Results and evaluation

Let's look at the generated classification report.

	precision	recall	f1-score	support
0.0	1.00	0.85	0.92	565
1.0	0.04	0.80	0.08	5
accuracy			0.85	570
macro avg	0.52	0.82	0.50	570
weighted avg	0.99	0.85	0.91	570

We can now see that our overall weighted recall accuracy is 85%. This may not seem like much, but for strictly no-neural-network algorithms, its on-par and even better than most other notebooks of this dataset. I'm quite happy with the results, and I do believe that they can be applied to significantly improve the time it takes to detect an exoplanet. While the results themselves are not nearly accurate enough to be completely reliable, they could still be of great use to a human operator observing the data.

I believe this project was a success.