Machine Learning in the Hunt for Exoplanets

A Novel Approach

I. Shrimpton

A 40 credit research project for the degree
of Bachelor's Computer Science

Supervised by Peter Tino
Word Count: 4,390

School of Computer Science
University of Birmingham
United Kingdom
March 2024

Abstract – The Kepler missions aimed to search for transiting exoplanets across hundreds of thousands of stars, however the data processing pipeline introduced for these missions required weeks worth of human input for manual verification. In this paper, the ExoplaNet VErification and Orbital Period Estimation (ENVELOPE) framework is proposed to provide an alternative to NASA's Transiting Planet Search module, using a combination of data transformation and machine learning methods in order to verify the existence of a transiting planet, and learn the orbital period. Promising results are seen, and the orbital periods of transiting planets are found to a high degree of precision. With further studies, this proposed framework could provide a strong alternative to the Transiting Planet Search module.

Keywords: exoplanets, transit method, machine learning, support vector machine, DBSCAN, Fast Fourier transform

---

I would like to sincerely thank my supervisor, Professor Peter Tino, for his support and guidance throughout the past six months, without whom I would not have been able to complete this project.

Contents

1 Introduction

Since the first exoplanet was discovered in 1992 [1], nearly 5,600 further discoveries have been confirmed across thousands of planetary systems [2]. So far, this decades-long process has involved surveying millions of stars, producing large amounts of data that cannot humanly be processed. Historically, exoplanet surveying missions have relied on substantial human input to confirm candidates and manually analyse features of these planets. For example, the Kepler missions relied on the Transiting Planet Search (TPS) module developed by NASA to identify planetary candidates [3]. After these ∼4,800 planetary candidates were identified [4], they had to be manually verified and further studied to find precise parameters and filter out false positive results. Most Kepler discoveries were made this way, with heavy reliance on the TPS Module for initial parameter estimation.

In this research project, the concept of applying machine learning (ML) to further automate this process is explored. Specifically, we propose the novel framework, ExoplaNet VErification and Orbital Period Estimation - ENVELOPE. The contributions made by this framework include:

- An alternative to the TPS module, searching for transits requiring no prior knowledge of the star or planet candidate parameters
- An iterative ML algorithm designed to verify the existence of a transiting planet and learn its orbital period
- An approach with explainable AI in mind, allowing for human-interpretable processes and visualisations throughout the process

The proposed framework uses a range of data transformation and ML methods such as support vector machines. Limitations are set in order to achieve a realistic standard, wherein the range of orbital period searched is between 0.5-21 Days, and as a soft goal, the target of detecting any transit with a depth greater than 1,500 parts per million (ppm) is set. Results prove an effective search method, optimising the orbital period to an accuracy of up to 6 significant figures.

2 Background & Related Work

2.1 Transiting Planets

Defined as any planet outside of our solar system, exoplanets have a 30-year history of study. During this time, a range of detection methods have been introduced, each of which can reveal various properties of exoplanets. The focus of this paper is the transit method, which is the most prominent detection method, making use of observing changes in flux (apparent brightness) of a target star.

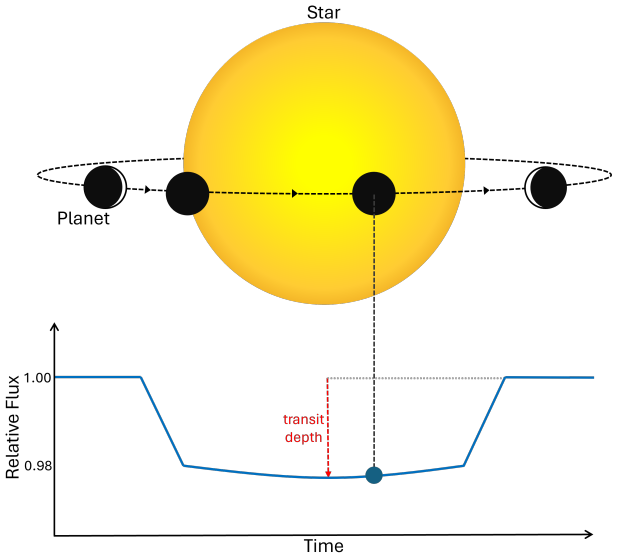

Figure 1. Light curve formed whilst observing transiting planet from Earth

When an exoplanet crosses in front of, i.e. transits its parent star, a small amount of light coming from the star is blocked out, as shown in Figure 1. It is possible to measure this change in flux and infer planetary parameters from it, for example orbital period based on how often the transit occurs, or planetary radius based on the transit depth.

The transit method was first devised by Charbonneau et al., 2000 [5], who observed two full transits of planet HD 209458b. Since then, there have been plenty of studies regarding this method, including the Kepler mission, a space telescope built by NASA to search for transiting exoplanets, active between 2009-2018.

During this time, it surveyed over 500,000 stars, discovering more than 2,600 exoplanets [6]. The data obtained by this mission is still used widely today as both benchmarks and to discover new planets which haven't yet been found within the data. In order to decrease the manual workload, NASA developed the Kepler data processing pipeline [3], which is the initial automatic processing pipeline for the Kepler space telescope's data. Jenkins et al., 2010, [3] details the Kepler data processing pipeline; after initially cleaning the data and removing any anomalies, it executes a "Transiting Planet Search" (TPS), which consists of a wavelet-based, adaptive filter. From this search, a series of Threshold Crossing Events (TCEs) are outputted. These events represent a candidate exoplanet transit, each of which is then processed and validated through the Data Validation stage.

However, this pipeline does have limitations; the threshold is non-adaptive and has a high false positive rate [7, 8], due to it misclassifying a range of other variable star types. For example, eclipsing binary stars are a system made up of two stars orbiting each other, which when viewed along its orbital plane will eclipse each other, temporarily occluding some of the system's light. The light curve produced from an eclipsing binary system is similar, but not the same, as a transit curve.

Once each of the ∼20,000 individual candidate transit events have been validated within this pipeline, manual human input is required to confirm the existence of an exo-

planet or whether it is a false positive, and further manual work is needed to identify features of the newly discovered exoplanet. This tedious process requiring weeks worth of human input is prone to human error and potentially biased classification.

In light of this, the application of ML has been introduced to the field of exoplanet studies in more recent years. Shallue & Vanderburg, 2018 [9] designed a deep learning model to classify potential planet signals into true positives and false positives with an accuracy of 98.8%. Hussain et al., 2017 [10] compare supervised ML approaches to classifying transit curves, also with high precision rates. Although these two studies produce good results on candidate analysis, there are downsides; training a supervised learning model takes time and a large amount of labelled data. Both studies also only focus on filtered datasets and rely on information provided by autovetting pipelines, such as information on TCEs and Kepler Objects of Interest (KOIs), for information on a transiting planet candidate.

On the other hand, Malik et al., 2021 [11] approach the light curve data with no prior knowledge, applying a classical machine learning method to classify light curves into 'candidates' or 'non-candidates'. Their proposed method produces results with a recall of 82%, however its precision is only 63%; this means that, although it can predict a true transit with decent accuracy, there is still a high false positive rate. Coughlin et al., 2016 [12], in association with NASA, works on improving the automation of the initial Kepler data processing pipeline by developing the "robovetter", a series of decision trees aimed to mimic the human decision process.

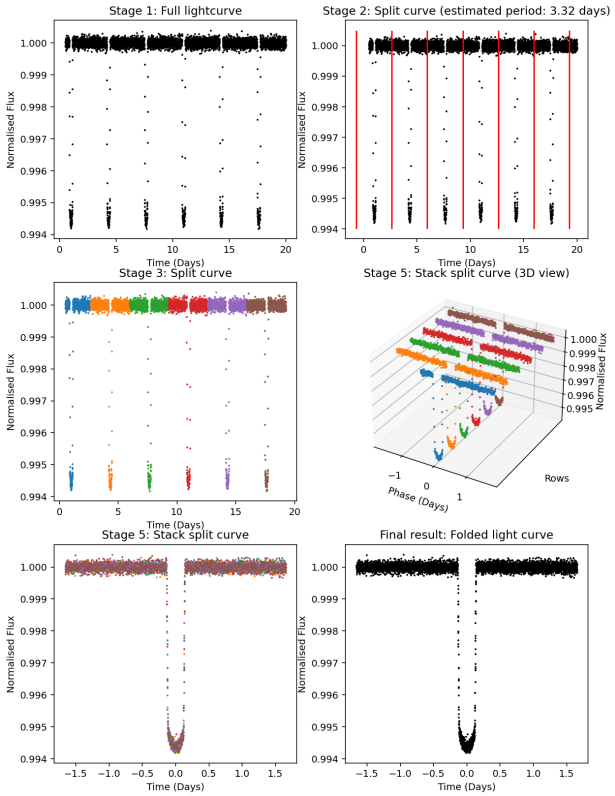

Figure 2. Step-by-step process of phase folding a light curve

This robovetter provides high accuracy results and significantly reduces the amount of human verification required, however it is not perfect and each planet is still manually checked before confirming its existence.

2.2 Phase Folding

When observing a transit, it may be that only a few data points are recorded throughout its duration. Since observations are generally made over several days (or even months), multiple transits tend to be observed. Therefore, it is possible to phase fold a light curve, splitting one curve into several based on an estimated orbital period, and "stacking" parts on top of one-another to create a single transit curve, as shown in Figure 2. From this folded light curve, planetary parameters can be found with more certainty.

3 Proposed Framework

3.1 Data Pre-Processing

The data used in this framework comes from the Kepler space telescope, obtained through NASA's Exoplanet Archive [4]. A Python package called Lightkurve [13] is used to query this API and perform some initial data cleansing. Firstly, several light curves may be available, for example Kepler space telescope's observations are separated into quarters, therefore any observations made over the space of more than 3 months will inherently come as two or more "pieces" of time series data. In order to obtain a single light curve, each individual time series is flattened and normalised before "stitching" together. The flattening stage removes any low frequencies observed from the star, for example stellar activity, certain types of variable stars, or variations in pixel sensitivity on the telescope are all reasons why a star may appear to have low frequency changes in flux.

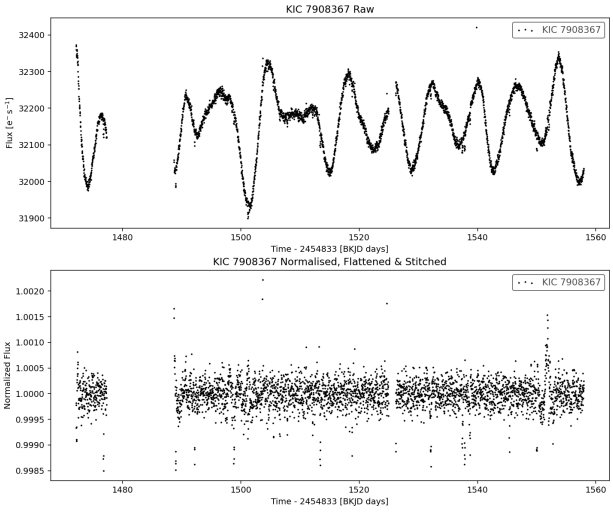

Figure 3. Before and after a raw light curve is flattened, normalised and stitched.

Once the stitched light curve has been obtained, outliers are then removed via sigma clipping. During this project, it was found that the default sigma value of 5.0 from the Lightkurve package was too harsh and would often remove full transits from the data. On the other hand, outliers lying above the median are not as important in this process,

and can be subject to a harsher sigma value. Therefore, the lower sigma value was adjusted to 15.0 while the upper value was set to 10.0, which generally worked well.

The final light curve is normalised to between 0 and 1, having been stitched together and has had underlying low frequency variations removed. There may be gaps in the data; this may be due to observational gaps (i.e. the star is out of scope of the telescope's sensors for some time), or due to cleaned anomalies, for example some values directly from the Kepler pipeline are NaN, which are removed entirely in the pre-processing process.

Although observational gaps do not affect further processes in this framework, the light curve is linearly interpolated using SciPy's `interpolate.interp1d` method [14], as shown in Figure 4. This is done to create an evenly-spaced time series, which allows for the usage of signal processing methods within this framework. Linear interpolation was chosen since higher order interpolation was found to create high amplitude peaks when large gaps were present, interfering with the transit data.

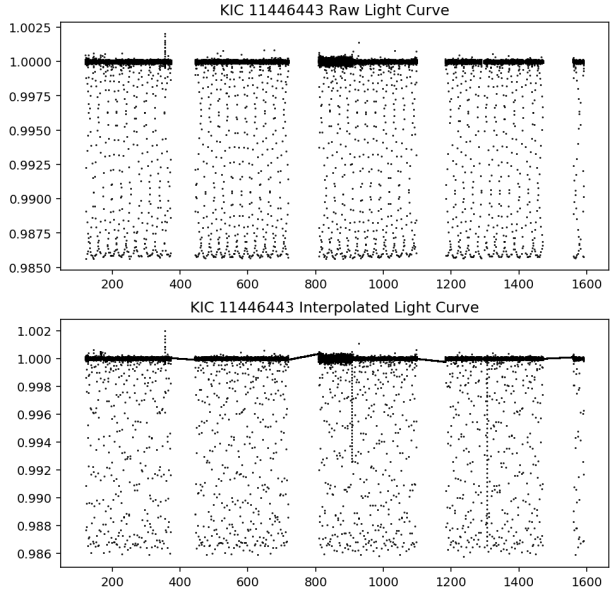

Figure 4. Linear interpolation on the light curve

3.2 ExoplaNet VErification and Orbital Period Estimation (ENVELOPE)

3.2.1 Initial Estimations

The first step of the proposed ENVELOPE framework is to search for patterns in the whole light curve using a Fast Fourier Transform (FFT), an algorithm designed to transform a signal into the frequency domain, breaking it down into separate sinusoidal waveforms of different powers [15]. This is done using SciPy's `fft` module [14], and the frequency domain is limited to between 0.048 days^{-1} and 2 days^{-1}, in other words the initial time period estimate only searches for time periods of between 0.5 days and 21 days.

After performing the FFT, all prominent peaks are found by using SciPy's `signal.peak_prominences` method, then the top N most prominent peaks are found, where $N = min(len(peaks)//10, 30)$. The threshold was chosen to be more lenient, meaning that some irrelevant frequencies may still be found, however this is done since some light curves have high noise and the transit frequency isn't always prominent.

The top N most prominent peaks are now defined as the initial orbital period estimates. As is shown in Figure 5, the linearly interpolated observational gaps in the time series do not cause issue in this step, and peaks are present at the frequency (and multiples) of the transiting planet.

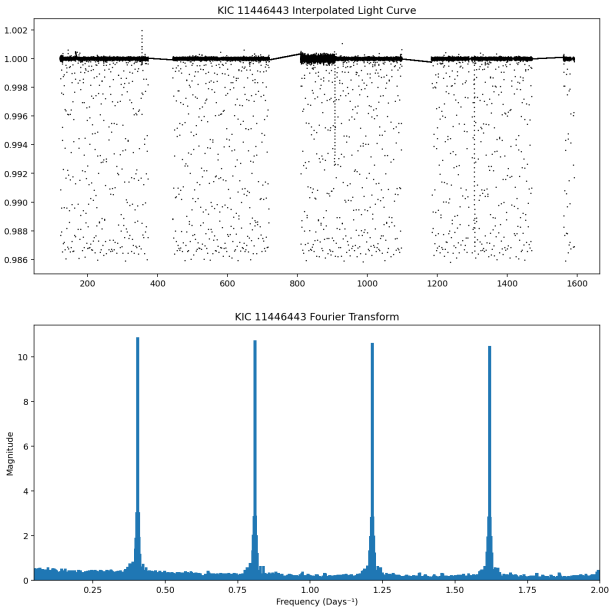

Figure 5. Fast Fourier Transform on KIC 11446443

3.2.2 Transit Curve Transformation

For each estimate obtained from the FFT, the orbital period estimation algorithm is applied. The first step in this process is to phase fold the light curve based on the estimated period. Assuming an exoplanet is present in the data, depending on the value of the orbital period estimate, the transit curve may not be lined up well. Due to large noise and gaps in the data, the FFT often detects multiples or factors of the actual orbital period; if this happens, the folded transit curve may display multiple transits or half-transits, as shown in Figures 6 and 7. A method devised to counter this problem is discussed further on in Section 3.2.6. If the orbital period estimation is not a multiple or factor of the actual orbital period, the transit curve may not show any correlation when viewed, as is seen in Figure 8.

In this ENVELOPE framework, an alternative viewpoint of the data is used; rather than viewing the folded light curve as a flattened time series, the data is not immediately flattened, such as in Stage 5 of Figure 2. Instead, the folded transit curve is viewed from top-down as a two dimensional image, each pixel representing the normalised relative flux of a specific row in the fold at that point.

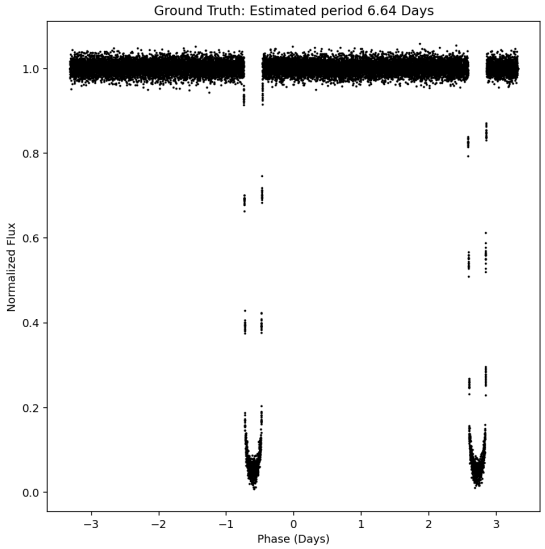

Figure 6. A transit curve phase folded to twice the actual orbital period

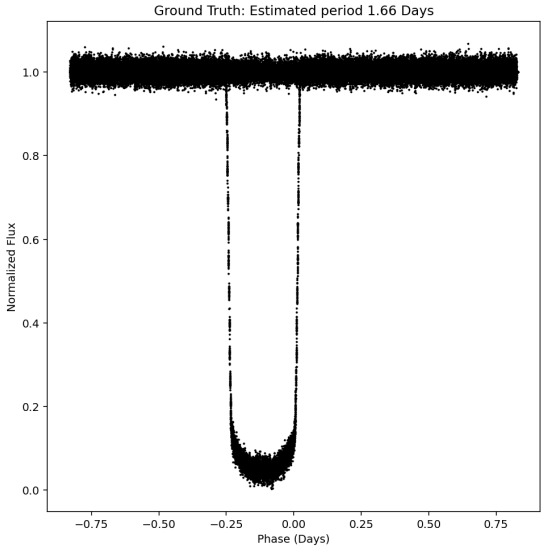

Figure 7. A transit curve phase folded to half of the actual orbital period

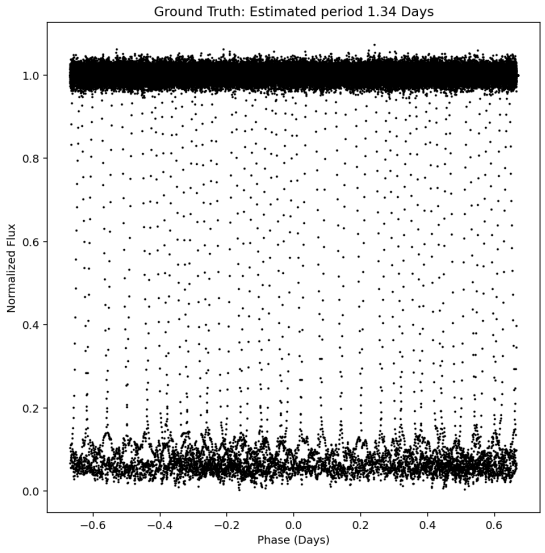

Figure 8. A badly folded light curve

With this transformation, it is possible to understand correlations which weren't necessarily apparent beforehand. As seen in Figure 9, this allows for a more comprehensive understanding of how accurate the current orbital period estimation is. If the fold is good, a straight vertical line is apparent, otherwise there is some displacement between each "row" where the transit exists. The aim of this algorithm is to learn the orbital period (if any planet is present) by aligning the transits to a straight, vertical line from this transformed viewpoint.

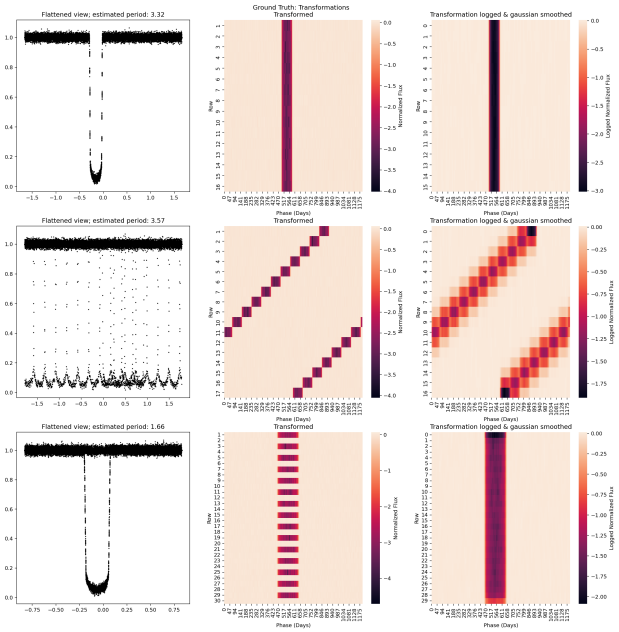

Figure 9. Transforming phase fold

In order to achieve this, the image is thresholded to obtain a set of coordinates where the values are lowest. The threshold is found via the following:

```
allowed_points_per_fold = 30
threshold_index = num_rows *
                  allowed_points_per_fold
threshold =
    sorted(image_values)[threshold_index]
```

After finding the thresholded coordinates, the following sequence is applied to the data:

1. DBSCAN to find clusters
2. If `num_clusters` $>$ 40 and `average_cluster _size` $<$ 100: Stack clusters
3. Reduce cluster(s)
4. Find gradient through a support vector machine (SVM)

The following sections cover these points in more detail.

3.2.3 DBSCAN

It is not always true that a single line will be present in the transit image. If the light curve is not folded on the actual orbital period (or multiples/factors of), the line will be skewed to some degree, sometimes not even recognisable as a line, such as in Figure 10.

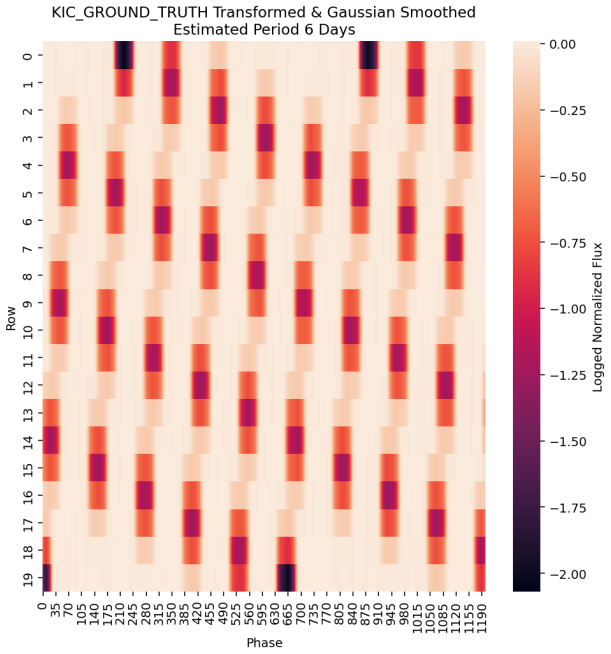

Figure 10. Badly folded light curve with no strong correlation

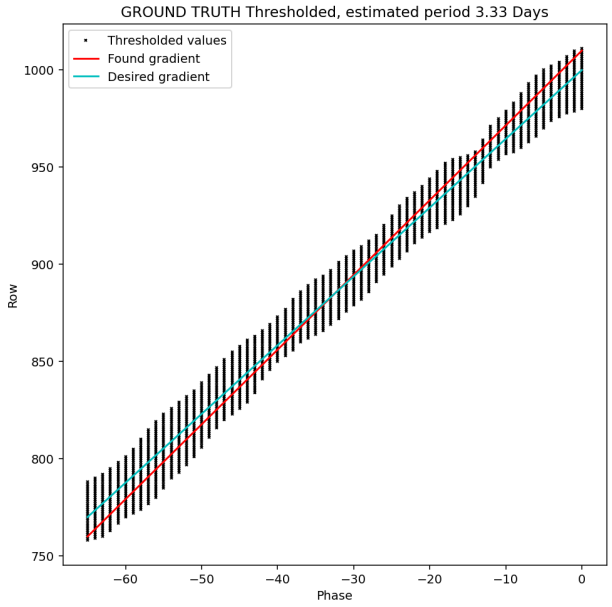

Figure 11. Inaccurate gradient found without reduced cluster

In these cases, it is difficult to find the gradient by blindly applying a linear regression, hence a clustering algorithm is first applied to the thresholded coordinates in order to determine the next step. DBSCAN was used in this framework due to its ability to easily find density-based clusters without prior input of cluster sizes or number of clusters, since this information is not yet known.

The next step of the framework depends on the outcome of this clustering algorithm; if there are few clusters which are relatively large, it is likely that these somewhat lined-up transit curves, and hence can continue straight to the reduction stage (Section 3.2.4).

However, if there is a large number of small clusters, no strong correlation is found. It is likely that the estimated orbital period is not close to the actual period, so this orbital period estimate is skipped.

3.2.4 Cluster Reduction

Since the clusters produced from the thresholded values can produce a large number of coordinates, each cluster is reduced before a SVM is applied. This entails including only the middle $x\%$ values per row per cluster, where $x = 1$ if one cluster is present, otherwise $x = 50$. This reduction stage was introduced after experimentation saw that applying the SVM without reduction would cause the optimal gradient to be found diagonally within a wide cluster line, rather than straight down the centre, as shown in Figure 11.

3.2.5 Support Vector Machine

Before applying a SVM to find the gradient of the line(s), the axis are inverted. Rows become the x variable as it is called in Scikit-Learn's `svm` module [16], whereas the phase value is now called y. They are swapped due to the fact that it is much easier to optimise a SVM to a horizontal line of gradient 0, rather than a vertical line of gradient infinity. Scikit-Learn's `LinearSVR` class is used in this step.

A support vector machine was chosen for this due to its simplicity and efficiency. The value of epsilon was chosen to be the standard deviation of the cluster, calculated via the following:

```
flattened_rows = []
for row in rows:
    row_values = phase_vals_in_row
    centre = median(row_vals)
    row_vals = row_vals - centre
    flattened_rows.append(row_vals)
std_dev = std(flattened_rows)
```

If there are multiple clusters, a SVM is applied to each cluster individually. The weighted average is taken to obtain the found gradient, where the weights are based on the cluster size to avoid smaller, more inaccurate clusters skewing the average.

The current orbital period estimation is adjusted by a factor of this gradient; a decaying learning rate is applied to this, under the assumption that as the learnt value tends towards the correct orbital period, it can be adjusted with more fine tuning in mind.

3.2.6 Transit Verification

As previously mentioned, it is difficult to differentiate between a correct orbital period and a factor of the actual period. Another difficulty is multiples of the orbital period; as in Figure 12, there are two lines visible which, with this

current framework, will simply be clustered into two lines and be treated as a single transit dip.

In order to combat this, a verification step has been added to the iteration process. This step checks first for multiples of the orbital period, then factors. In order to achieve this, a normalised histogram of the transformed smoothed image is taken, via:

```
histogram = sum(smoothed_image,
          axis=0) / len(smoothed_image)
```

This produces a single time series which resembles a phase-folded transit curve. Two filters are then applied to this histogram to produce two separate results: Sobel and Laplacian of Gaussian, in order to obtain the first and second order derivatives respectively. Figure 12 demonstrates these derivatives. A candidate transit curve can be detected using this histogram and its calculated derivatives.

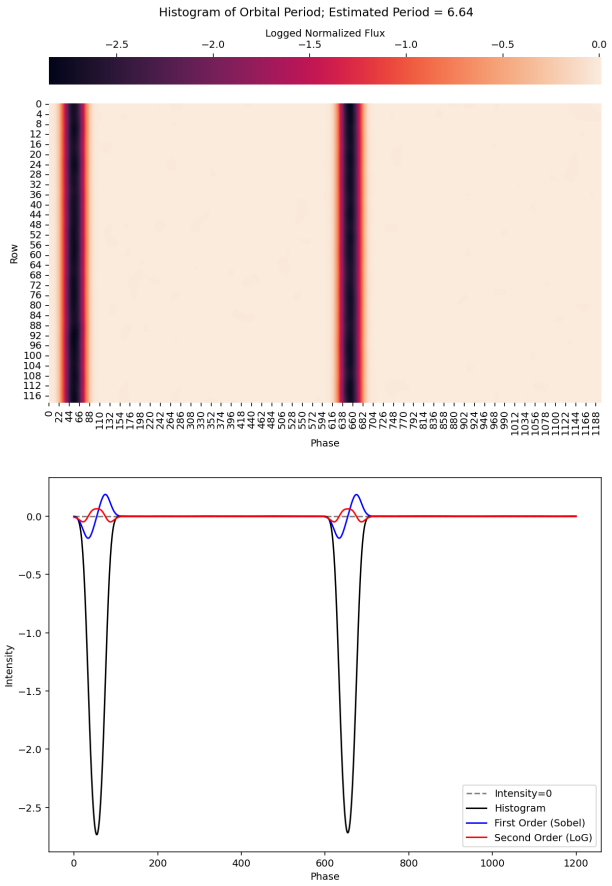

Figure 12. Estimated orbital period is double the actual period

Firstly, as the transit reaches its midpoint, the curve reaches its turning point and the gradient of the line (first derivative) crosses from negative to positive; therefore zero-crossings are searched for in the first derivative. The next step is to review the second derivatives, measuring the rate of change of the histogram. If the second derivative is greater than 0, the curve at that point is concave. When comparing the two derivatives, checking for anywhere with first order derivative 0 and second order derivative greater than 0, it can be determined that these coordinates contain a dip in the histogram. An extra check is added to this detection method: simply that the histogram is in the bottom 10% of all histogram values. This check confirms that the dip is significant enough to qualify as a potential transit curve. The final check in this step is for noise; if there are several turning points present within one transit-like dip, such as in Figure 13, only one transit is counted. The orbital period is then divided by the number of transits found within this method (provided it's greater than 1).

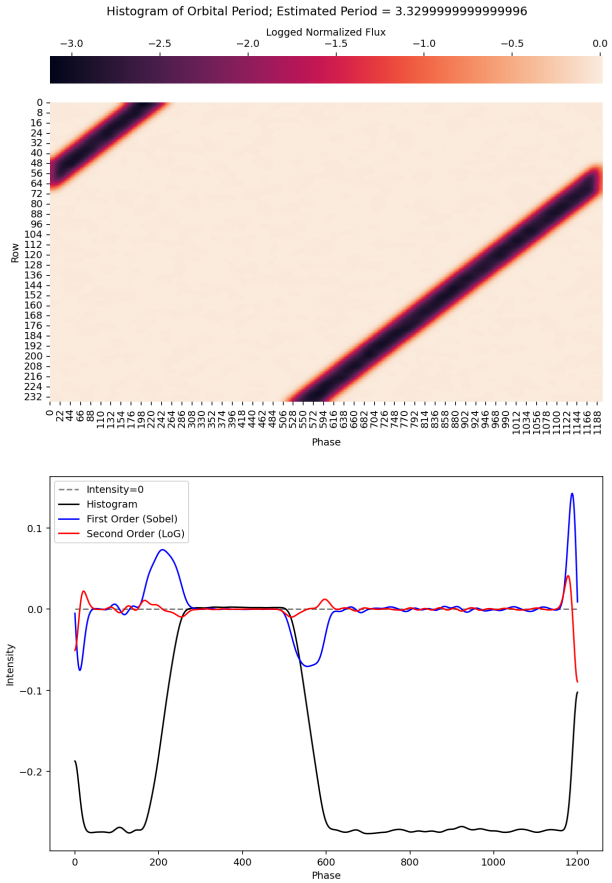

Figure 13. A badly folded light curve has some noise as a histogram

The second part to this step is to check for factors; this is done by calculating the "transit density" over the folded, non-smoothed image. The same derivative algorithm as above is applied, except to each row in the image. The average number of transits per row, or transit density, is found; if the estimated period is correct, the transit density is 1.0. If the estimated period is a factor of the actual period, the reciprocal of the factor would be present, for example if the estimated period is half of the actual period, the transit density would be 0.5; for a third, it would be 0.33.

If the transit density is found to be less than 0.6, the reciprocal is taken and rounded to the nearest whole number, then the estimated orbital period is divided by that number.

3.2.7 Optimising Orbital Period Estimation

The steps detailed from Section 3.2.2 to Section 3.2.6 are repeated n iterations, where n is arbitrarily set as 10. At the end of the iteration, the existence of a singular good transit is verified using the derivatives method as described in Section 3.2.6. If a single transit is found, the star is marked to

have a transiting exoplanet present, with its orbital period being the learnt value.

4 Results

To evaluate the effectiveness of the proposed framework, a set of test data was processed. The first observation was that each test returned a positive result for the existence of a transit. Upon further inspection, this is likely due to a bad implementation of the transit verification stage.

However, when applying KOI test data to the algorithm to test the orbital period estimation module stage, promising results were seen. Of the 113 test KOI data, ENVELOPE correctly found the period of 12 transit-like light curves to a high level of accuracy. Meanwhile, a further 72 orbital periods were found to be exact factors of the known period. This further suggests an issue in the verification step of the algorithm, as factors of the orbital period are not being corrected properly. It should be noted that no multiples of the orbital period were returned as a single transit here; this information combined with the high factors rate implies that the verification step is over-sensitive to multiple transit detections, which could easily be fixed in further studies.

The gradient optimisation step of the algorithm is determined to be 74% accurate with the provided results, since this part of the algorithm does not take into account multiples or factors of the actual period. Some results can be visualised in order to understand in what circumstances the algorithm works well. As a starter, the error between ENVELOPE's calculated value and the Kepler estimated value can be calculated, using Equation 1.

$$error = \frac{|actual - estimated|}{actual} \tag{1}$$

The error can be visualised as transit duration and transit depth (real values taken from the test database) [17], with the error on each value represented as a colour scale, as in Figure 14. The general trend in this graph shows that a higher transit depth tends to yield a more accurate value, along with a lower orbital period.

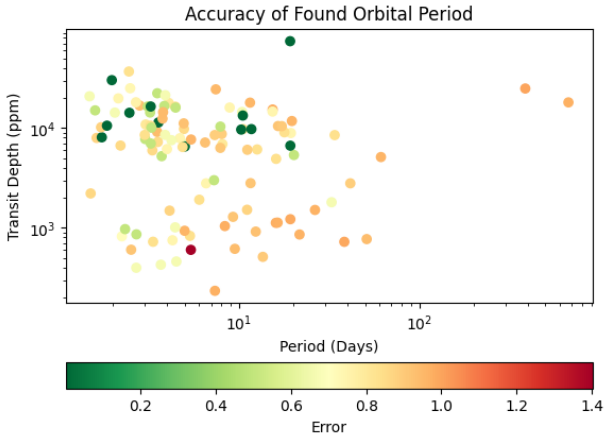

Figure 14. Error of each orbital period when comparing to labelled test data

Overlooking the factors of the actual orbital period, the range of use of the algorithm dramatically increases; Figure 16 demonstrates the effectiveness of the orbital period estimation across a wide range of periods and transit depths. With further work on the algorithm, this error can be fixed and will therefore be able to automatically determine the orbital period with high accuracy.

Filtering out the incorrect estimates, it is possible to calculate the error on the estimated orbital period, taking into account the factor errors via Equation 3. As seen in Figure 16, the resulting errors are extremely low, proving the effectiveness of the ENVELOPE algorithm.

$$factors = \frac{actual}{estimated} \tag{2}$$

$$error = \frac{|actual - (estimated \cdot round(factors))|}{actual} \tag{3}$$

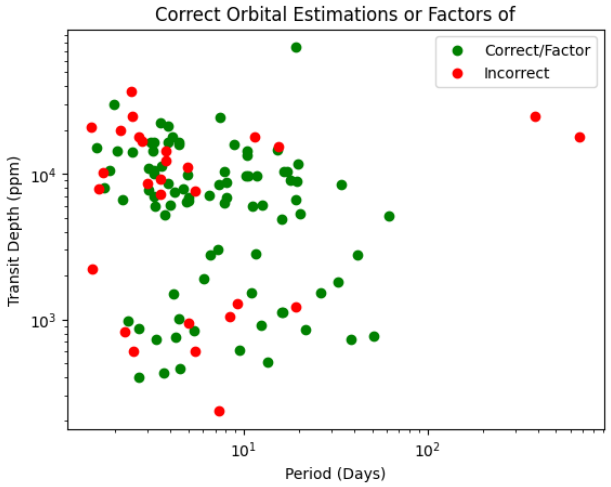

Figure 15. Correct orbital period estimations and correct factors of such are portrayed in green

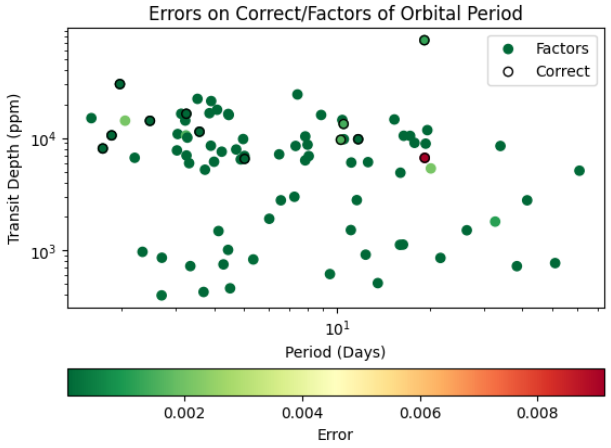

Figure 16. Errors on would-be orbital period estimations, overlooking

On the other hand, the estimations that don't return good approximations have various reasons for their incorrect results. Two common issues were high levels of noise and low value outliers, both of which were able to skew the results. Although ENVELOPE is less sensitive to noise due to the application of Gaussian smoothing, a small transit depth combined with high noise may cause the algorithm

to miss the transit altogether, such as in Figure 17. Realistically, this result should not have been accepted in the validation stage of the algorithm, however this is again due to the issues present in the validation stage. There are a number of results similar to this one, with no correlation visible within the transformed view.

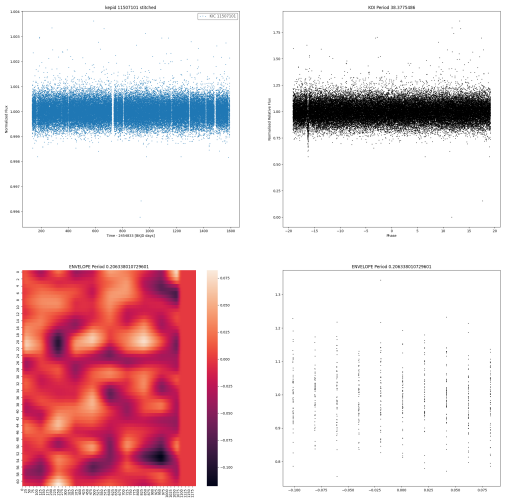

Figure 17. Result of KIC 11507101. Top left shows full light curve of the star, top right shows fold based on Kepler vetted parameters, while the bottom row shows ENVELOPE's results

Some results, however, show good promise of being able to find a transiting planet with high noise levels. Figure 18 demonstrates a result that has optimised towards a factor of the actual orbital period.

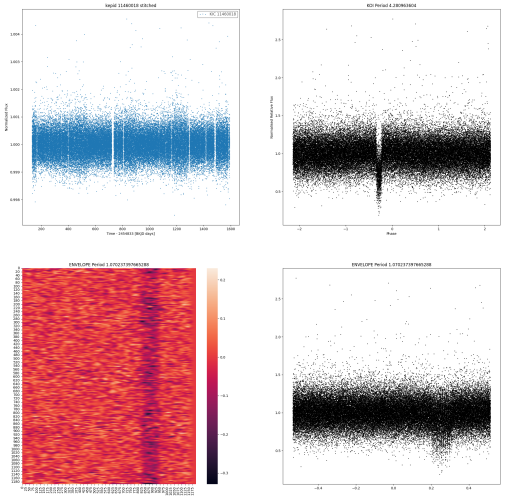

Figure 18. Result of KIC 11460018. Top left shows full light curve of the star, top right shows fold based on Kepler vetted parameters, while the bottom row shows ENVELOPE's results

A handful of estimated orbital period were returned as slightly off the actual period, as shown in Figure 19. This is clearly a non-vertical line, and hasn't been optimised correctly, likely due to too few iterations used within the algorithm. A fix to this could be to increase the iteration count, however the trade-off between time efficiency and improvement on a handful of results must be taken into consideration.

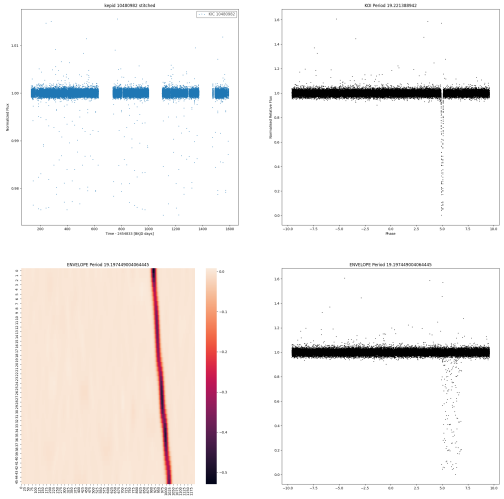

Figure 19. Result of KIC 10480982. Top left shows full light curve of the star, top right shows fold based on Kepler vetted parameters, while the bottom row shows ENVELOPE's results

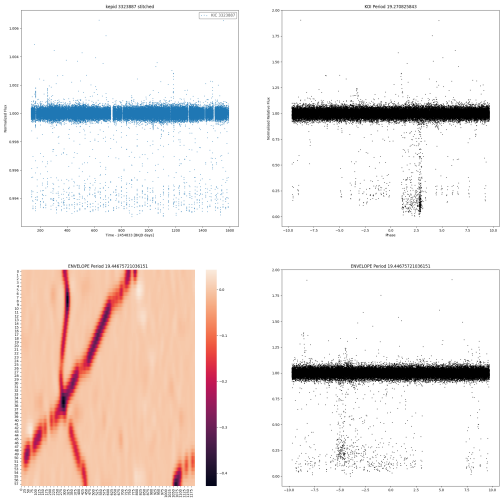

Figure 20. Result of KIC 3323887. Top left shows full light curve of the star, top right shows fold based on Kepler vetted parameters, while the bottom row shows ENVELOPE's results

Although it is possible to find multiple planets around one star using this method, due to Gaussian smoothing reducing the effect of the non-correlated transit of the secondary planet, sometimes the orbital periods are closely interlinked, for example when orbital resonance is occurring within the planetary system [9], the ENVELOPE algorithm may not be able to discern the two planetary transits, such as in Figure 20.

An example of a well-estimated orbital period is Figure 21. The estimated orbital period, 1.97376038 Days, matches the given Kepler orbital period, 1.97376085 Days, to 7 significant figures, proving the potential for accuracy of this framework when implemented well. Figure 22 also visualises 110 phase folds of the test data used.

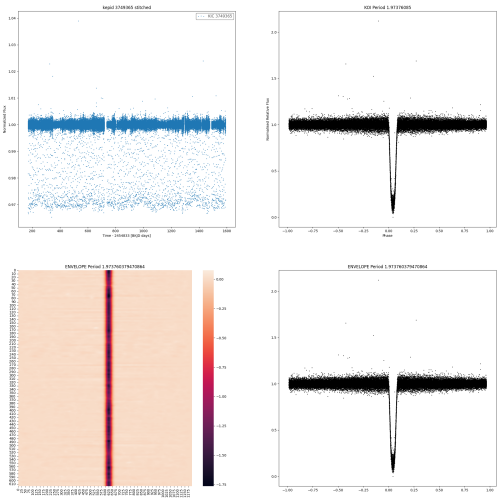

Figure 21. Result of KIC 3749365. Top left shows full light curve of the star, top right shows fold based on Kepler vetted parameters, while the bottom row shows ENVELOPE's results

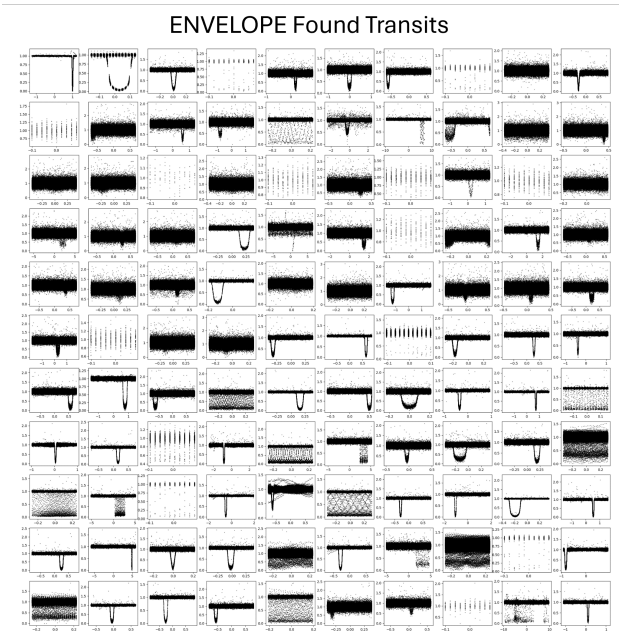

Figure 22. Phase folds of 110 found transit curves, according to the ENVELOPE algorithm

5 Conclusion

The proposed framework in this paper introduces a novel method to detect exoplanet candidates, providing a highly accurate orbital period estimation with the right circumstances, occurring about 74% of the time. This statistic is overlooking the existence of a bad implementation of the verification module, which proved to be overly sensitive when attempting to detect the existence of multiples of a planet, causing the estimated orbital period to be a divisible factor of the actual period roughly 64% of the time. However, this issue would be fixable if more time was available to fix the implementation of this stage. The framework was also found to be somewhat sensitive to noise, unable to identify a transiting planet when the transit depths are shallow.

5.1 Discussion & Future Work

Overall, this framework provides a possible alternative to the TPS module from the Kepler data processing pipeline, which with further research may provide a reliable auto-vetting method for finding transiting exoplanet candidates.

The issues discussed in this paper regarding the transit verification stage could easily be fixed with some more time; the derivative method works in theory, however the issue is finding the thresholds that work well as a general rule to classify a dip as a transit or a bad fold. Alternatively, other computer vision or ML approaches could be used to verify the transit curves in this stage, which may prove to be more effective.

The concept of false positives is only briefly touched on in this paper. A truly good auto-vetting framework would be able to disregard false positive values, such as eclipsing binary stars. Although not covered in this paper, future work could implement a stage to the ENVELOPE framework in which features of the folded transit curve are analysed to distinguish between a transiting planet and a false positive.

Although noise was found to be an issue in some cases, the goal of identifying planets with a transit depth of greater than 1,500 ppm was broadly achieved. However, with further studies it is possible that these smaller, shallower transits may be more easily detected using the ENVELOPE framework. For example, the thresholds set throughout the framework could become more adaptive, rather than generally selecting the top x%, etc.

Even with the imperfections, the results proved to be impressive, learning the orbital period to a high level of precision with only a few iterations in some cases. The exploration of machine learning in the application of exoplanets within this paper produced some novel viewpoints of transit data, allowing for explainable processes along every step of the framework, and has the possibility to be an alternative to Kepler's Transiting Planet Search module with further studies.

References

[1] A. Wolszczan and D. A. Frail. "A planetary system around the millisecond pulsar PSR1257 + 12". In: *Nature* 355.6356 (Jan. 1992), pp. 145–147. ISSN: 1476-4687. DOI: 10.1038/355145a0. URL: https://doi.org/10.1038/355145a0.

[2] NASA. *NASA Exoplanets*. 2023. URL: https://exoplanets.nasa.gov/discovery/exoplanet-catalog/ (visited on 10/20/2023).

[3] Jon M. Jenkins et al. "OVERVIEW OF THE KEPLER SCIENCE PROCESSING PIPELINE". In: *The Astrophysical Journal Letters* 713.2 (Mar. 2010), p. L87. DOI: 10.1088/2041-8205/713/2/L87. URL: https://dx.doi.org/10.1088/2041-8205/713/2/L87.

[4] NASA. *NASA Exoplanet Archive*. URL: https://exoplanetarchive.ipac.caltech.edu/index.html.

[5] David Charbonneau et al. "Detection of Planetary Transits Across a Sun-like Star". In: *The Astrophysical Journal* 529.1 (Jan. 2000), pp. L45–L48. ISSN: 0004-637X. DOI: 10.1086/312457. URL: http://dx.doi.org/10.1086/312457.

[6] NASA. *Kepler Space Telescope*. Sept. 2020. URL: https://exoplanets.nasa.gov/keplerscience/.

[7] Andrew Vanderburg et al. "A Habitable-zone Earth-sized Planet Rescued from False Positive Status". In: *The Astrophysical Journal Letters* 893.1 (Apr. 2020), p. L27. DOI: 10.3847/2041-8213/ab84e5. URL: https://dx.doi.org/10.3847/2041-8213/ab84e5.

[8] Joseph R. Schmitt, Jon M. Jenkins, and Debra A. Fischer. In: *The Astronomical Journal* 153.4 (Mar. 2017), p. 180. DOI: 10.3847/1538-3881/aa62ad. URL: https://doi.org/10.3847%2F1538-3881%2Faa62ad.

[9] Christopher J. Shallue and Andrew Vanderburg. "Identifying Exoplanets with Deep Learning: A Five-planet Resonant Chain around Kepler-80 and an Eighth Planet around Kepler-90". In: *The Astronomical Journal* 155.2 (Jan. 2018), p. 94. DOI: 10.3847/1538-3881/aa9e09. URL: https://doi.org/10.3847%2F1538-3881%2Faa9e09.

[10] Naireen Hussain, Chelsea Huang, and Kristen Menou. "Identifying Exoplanets from Transits with Supervised Machine Learning". In: *Galbraith Society Undergraduate Engineering Journal* 1.3 (2017), pp. 23–31.

[11] Abhishek Malik, Benjamin P Moster, and Christian Obermeier. "Exoplanet detection using machine learning". In: *Monthly Notices of the Royal Astronomical Society* 513.4 (Dec. 2021), pp. 5505–5516. ISSN: 0035-8711. DOI: 10.1093/mnras/stab3692. eprint: https://academic.oup.com/mnras/article-pdf/513/4/5505/43865055/stab3692.pdf. URL: https://doi.org/10.1093/mnras/stab3692.

[12] Jeffrey L. Coughlin et al. "Planetary Candidates Observed by Kepler. VII. The First Fully Uniform Catalog Based on the Entire 48-month Data Set (Q1-Q17 DR24)". In: *The Astrophysical Journal Supplement Series* 224.1 (May 2016), p. 12. ISSN: 1538-4365. DOI: 10.3847/0067-0049/224/1/12. URL: http://dx.doi.org/10.3847/0067-0049/224/1/12.

[13] Geert Barentsen et al. *Lightkurve*. 2018. URL: https://github.com/lightkurve/lightkurve.

[14] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[15] Ronald Newbold Bracewell and Ronald N Bracewell. *The Fourier transform and its applications*. Vol. 31999. McGraw-Hill New York, 1986.

[16] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[17] NASA. *Kepler and K2 Data Processing Pipeline*. URL: https://keplergo.github.io/KeplerScienceWebsite/pipeline.html.