Exoplanet Detection using Machine Learning

Abhishek Malik[1]*, Ben Moster[1] and Christian Obermeier[1]
[1] Universitäts-Sternwarte München, Ludwig-Maximilians-Universität, München, Germany

1 December 2020

ABSTRACT
We introduce a new machine learning based technique to detect exoplanets using the transit method. Machine learning and deep learning techniques have proven to be broadly applicable in various scientific research areas. We aim to exploit some of these methods to improve the conventional algorithm based approach used in astrophysics today to detect exoplanets. We used the popular time-series analysis library 'TSFresh' to extract features from lightcurves. For each lightcurve, we extracted 789 features. These features capture information about the characteristics of a lightcurve. We used these features later to train a tree-based classifier using a popular machine learning tool 'lightgbm'. This was tested on simulated data which proved it to be more effective than conventional box least squares fitting (BLS). It produced comparable results to the existing state-of-art models while being much more computationally efficient and without needing folded and secondary views of the lightcurves. On Kepler data, the method is able to predict a planet with an AUC of 0.948 which means that, 94.8% of the time a planet signal is ranked higher than a non-planet signal and Recall of 0.96 meaning, 96% of real planets are classified as planets. With the Nasa's Transiting Exoplanet Survey Satellite (TESS), a reliable classification system is much needed as we are receiving over a million lightcurves per month. However, classification is harder as lightcurves are shorter. Our method is able to classify lightcurves with an accuracy of 98% and is able to identify planets with a Recall of 0.82.

Key words: planets and satellites: detection, methods: data analysis, techniques: photometric

1 INTRODUCTION

Planets outside of our solar system are known as extra-solar planets or exoplanets. The discovery of the first planets in 1992 (Wolszczan & Frail 1992) opened our minds to the possibility of life beyond Earth. In the last three decades, planet detection has become a major research area in astrophysics and astronomers have developed various methods to detect exoplanets. As of July 2020, astronomers have discovered 4281 confirmed planets and a majority of those are detected by the transit method.

Depending on the observer's position, a planet may move in front of it's host star blocking a part of the star's light and causing a dip in it's brightness. In the transit method, we continually observe a star and look for such 'dips' in it's brightness. The proposed method in this paper is developed for the transit method but it can easily be adapted for other time series based method such as radial velocity method.

NASA's Kepler (Borucki et al. 2010) mission, and it's second survey program K2 (Howell et al. 2014), was a major step forward in the creation of a vast catalogue of exoplanet systems, which may give insights into the formation process of planets and the abundance of potentially habitable Earth-like analogues. The validation process of the candidates detected in those surveys is a still ongoing process, as ruling out false-positive detections is a time-consuming process and new candidates are still being discovered.

In April 2018, NASA launched Transiting Exoplanet Survey Satellite (TESS) as a successor of Kepler with the primary objective to survey some of the brightest stars. TESS (Ricker et al. 2015) will observe stars that are 30-100 times brighter than those selected by the Kepler mission. This enables us to identify targets that are far easier to follow up for detailed observation with other space based and ground based telescopes. It covers a sky area of 400 times larger than that covered by Kepler while producing around a million lightcurves per month.

One of the most commonly used methods for planet detection is Box-fitting Least Squares (BLS) (Kovács et al. 2002), where we attempt to fit a box model to the data as shown in Figure 1 (Obermeier 2016). Cases with a seemingly good fit can then be manually reviewed. However, BLS is limited in terms of signal to noise and data cadence and is vulnerable to false-positive detections created by cosmic, random noise patterns or stellar variability.

TESS and other similar surveys still rely on manual analysis. Yu et al. (2019) provided a good overview of this process. For a typical TESS sector, usually a group of experts manually eliminate obvious false positive cases, a process which alone can take a few days. From the remaining cases, each case has to be viewed by at

* E-mail: a.malik@usm.lmu.de

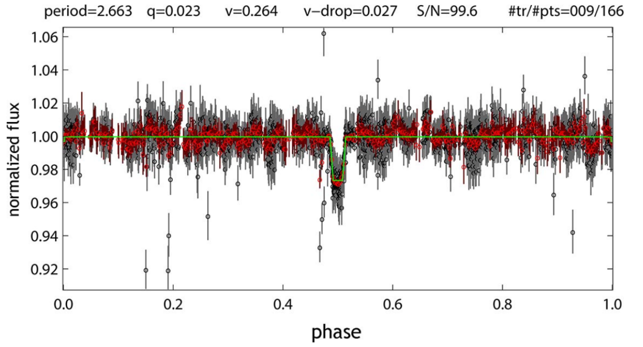

Figure 1. Box-fitting Least Squares (BLS) method in application. In this method, we try to find correlation of data to a box model.

least 3 experts. This kind of procedure can lead to disagreements on a particular case as people might not maintain the same definition for classification. A difference of opinion may arise in a team of experts on a particular case due to factors like the way a case is presented, how many TCEs viewed recently, or something as little as the time of day at which the TCE is being viewed (e.g. Coughlin et al. (2016a)).

For these reasons, we need a system that can reliably and repeatably select most important planet candidates for us, which can then be manually reviewed for confirmation at a later stage. Over the past years, there was a growing interest in building an automatic vetting system. Some of the notable earliest attempts include the Robovetter (Coughlin et al. 2016b), Autovetter project (McCauliff et al. 2015) and (Mislis et al. 2016) where a tree based model (decision tree or random forest) was trained for vetting. However, the research moved away from classical machine learning methods after that and deep learning methods became the new focus. The most notable work in this area was done by Shallue & Vanderburg (2018). They introduced a novel deep learning architecture *Astronet* which produced the best results to our knowledge on the Kepler data (Table 2). Their approach and their model architecture was adapted and applied to data from several different surveys such as Kepler's K2 mission (Dattilo et al. 2019), next generation transit survey (NGTS) (Chaushev et al. 2019) and TESS data (Yu et al. 2019).

Since the induction of Shallue & Vanderburg (2018), the community has moved on to deep learning, as deep learning methods tend to produce better results than classical machine learning approaches specially for more complex problems or data types. However, deep learning models are usually computationally expensive and require large amounts of data. In some cases, these models can be superfluous for the given problem and simpler or less computationally expensive approaches can perform equally well or even better. On the other hand, it's also important to sometimes move away from the currently best performing methods and search for new possibilities. With this motivation, in this paper we would like to propose a new direction to approach this problem using classical machine learning.

This paper is organised in five sections. Section 2 contains details about our methodology, specifically data preparation, feature extraction and model training. Section 3 explains the results achieved on simulated, Kepler and TESS data. It will also compare our results with some of the best performing models. In section 4, we will compare the main differences how our model differentiates from the other deep learning based methods being used in the area. It will also provide details of our vetting tool which can be used to make real time inference. Finally, we will conclude and discuss future steps in section 5.

2 METHODS

Machine Learning methods are widely used in scientific research areas to build classifiers i.e. an algorithm which separates data into two or multiple classes. In our case we are building a binary classifier which will separate each time-series photometry, a so-called lightcurve, into the classes 'planet candidate' and 'non candidate'. The current state-of-art machine learning methods (shown in Shallue & Vanderburg 2018) for planet detection utilises deep learning, a class of machine learning. However, our method is based on classical machine learning and one essential difference between our approach and the deep learning methods is that deep learning models are able to do feature extraction automatically, while we have to calculate features beforehand and provide them as input to the model. We used time series feature extraction based on scalable hypothesis tests (Christ et al. 2018, TSFresh), a python based library for feature extraction.

The idea behind the method was inspired by methodology used in time-series prediction projects (such as stock prediction) that uses feature engineering libraries like TSFresh. TSFresh is also used in projects like machine fault prediction, identifying epileptic seizures in EEG signals, Earthquake prediction, and Time series forecasting for business applications. Light curves are essentially a time series and so these tools can directly be used for our case.

We trained and tested our model on three kinds of data sets. The first stage used simulated data that used K2 photometry as a baseline with additional injected transit's. We then trained it on Kepler and finally TESS photometry. Each stage is split into three parts:

(i) processing and labelling the training data;
(ii) extract features from each lightcurve using TSFresh;
(iii) model training.

This workflow is also explained in the figure 2. Each of those steps will be explained in more detail in the following sections.

2.1 Preparing and Labelling Training Data

2.1.1 *Simulated Data:*

We obtained the K2 photometry from the Mikulski Archive for Space Telescopes (MAST) and used the calibration from Vanderburg & Johnson (2014). While the processing of Vanderburg & Johnson (2014) already removed the vast majority of systematic effects, we further cleaned up the data by identifying and removing remaining cosmics and creating a new noise model. If a point is an outlier with respect to it's neighbours, i.e. if it is more than 5σ above it's previous and following point, it was assumed to be a cosmic. Then, we removed the stellar variability that is common, depending on stellar type, with an iterative process. We smoothed the data, binned it, fit cubic splines to the bin, clipped negative 3σ outliers and iterated this process until it converges.

Then, we randomly injected transit signals in half of the processed lightcurves in an approach similar to Obermeier et al. (2016). We removed known planet systems from the data sample beforehand and then performed a blind search for eclipsing binary systems

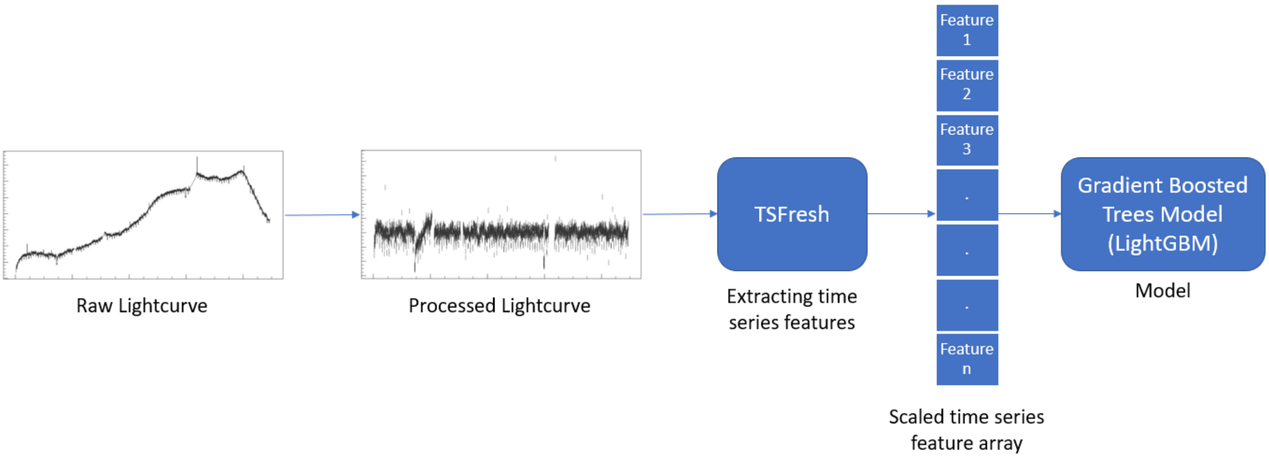
Raw Lightcurve
Processed Lightcurve
TSFresh
Extracting time series features
Feature 1
Feature 2
Feature 3
.
.
.
Feature n
Scaled time series feature array
Gradient Boosted Trees Model (LightGBM)
Model

Figure 2. Workflow of our method starting from raw lightcurves to performing inference on the sample.

based on our implementation of BLS. Any light curve with a detected signal to noise ratio of more than 12 was removed from the sample.

We then created a randomly filled space of orbital periods, stellar limb darkening and radii, planet radii and orbit inclinations. An example of the resulting simulated planet systems can be seen in Figure 3.

Each lightcurve with injected planets was labelled as class 1 and remaining curves were categorised as class 0. These labels were then used to train our classifier with the objective to identity the injected transit signals. In total, our training set consisted of 7873 lightcurves out of which 3937 belongs to class 1 (injected planet signal). This data set was then divided into training-validation set for 10-fold cross validation (90%) and test set (10%).

2.1.2 *Kepler Data:*

We used publicly shared dataset from the work of Shallue & Vanderburg (2018), the details about data processing can be found in section 3.2 of their paper. These lightcurves were produced by the Kepler pipeline (Jenkins et al. 2010), where each lightcurve contains around 70,000 data points equally spaced out at the interval of 29.4 minutes. The lightcurve were flattened and outliers were removed as mentioned in the previous section. The labels for the curves were taken from the Autovetter Planet Candidate Catalog (Catanzarite 2015). The catalog is divided into four lightcurve categories viz. planet candidate (PC), astrophysical false positive (AFP), non-transiting phenomenon (NTP) and unknown(UNK). All UNK were removed from the data set and all PC were given class label 1. All the remaining cases were assigned to the class 0. In total, there are 3600 PCs and 12137 non-PCs. The data is already divided into training set (80%), validation set (10%) and test set (10%). We used the same test set for our model performance as Shallue & Vanderburg (2018) in order to compare our model performance. Apart from this, we combined their training and validation set into a training-validation set for 10-fold cross validation.

2.1.3 *TESS Data:*

For the TESS data, we again used publicly available data provided by Yu et al. (2019) for their model *AstroNet-Vetting*. Details about their lightcurve processing can be found in the section 2.1, Where they used the MIT Quick Look Pipeline (QLP; Huang et al.) for processing their lightcurves. The QLP is designed to process the full-frame images (FFI) and produces lightcurves using it's internal calibrated images. The labelling was done by visually inspecting each lightcurve. Each lightcurve was categorised in 3 categories viz Planet Candidates (PC), Eclipsing Binary (EB) and Junk (J), where junk signals were the cases with stellar variability and instrumental noise. The dataset consists of 2154 EB, 13805 J and 492 PC signals, where PCs were labelled as class 1 and everything else was labelled as class 0. This data is also already divided into training set (80%), validation set (10%) and test set (10%). Like the previous case, we used the same test set and combined the training and validation set into training-validation set for 10-fold cross validation.

Some of the sample lightcurves containing transit signal(s) are shown in figure 3. All of these curves are taken from our test set of the respective dataset. It can be seen that the TESS (shown in blue) and Kepler (shown in green) data is a lot more noisy and likely to contain less number of transit's. On the other hand, simulated data is a cleaner as all the curves with S/N over 12 were removed from the dataset before the planet signals were injected. However, some of the injected transit's such as, the second lightcurve of simulated data were generated with low S/N values to imitate realistic scenarios.

2.2 Feature Calculation

Now we take the processed lightcurves and express them in form of features. These features capture information about the characteristics of the light curve and are used as input to our model for training and making inference. We used a popular python framework *TSFresh* Christ et al. (2018) to extract features such as features based on energy, fourier transform etc. Lightcurves might not consist of data points at a fixed time intervals. So at first, lightcurves are resampled to the frequency of 1 hour i.e. lightcurves are reformatted into windows of 1 hour to ensure that data points are uniformly distributed, this process is commonly referred as resampling. Although resampling is not mandatory, it is a standard practice and proven to produce a better representation of the data. In the resampling process, if multiple data points are inside one window, they are summed up and missing data points on the curves are interpolated. The frequency of one hour is chosen to ensure minimum data

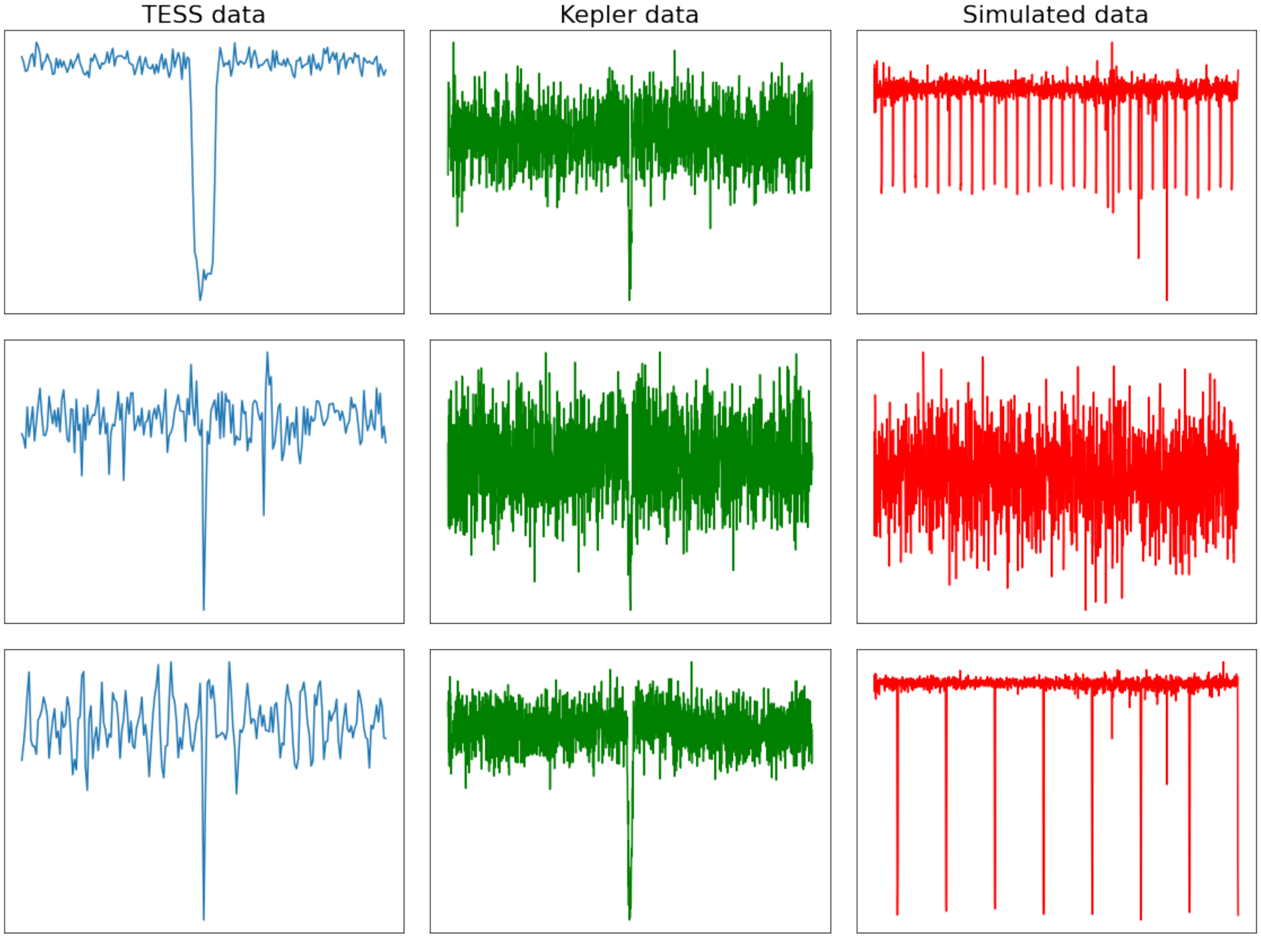

Figure 3. Sample lightcurves containing transit signal(s). All of these curves are taken from our test set of the respective dataset. It can be observed that the TESS (in blue) and Kepler (in green) data is a lot more noisy and likely to contain less number of transit's. Simulated data (in red) on the other hand is cleaner as all the curves with S/N over 12 were removed from the dataset before the planet signals were injected. However, some of the injected transit's were generated with low S/N values to imitate the realistic scenarios. The second lightcurve of simulated data is one such case.

loss while not increasing the data size significantly. This resampled time series can now be directly used with multiple time-series analysis tools. We used TSFresh's efficient feature extraction setting which extracted around 790 generalised time series features. Mathematical formulation and other details of these features can be found at: `https://tsfresh.readthedocs.io/en/latest/text/list_of_features.html` After the feature extraction, we implemented some standard data pre-processing steps for machine learning. We removed all features whose values were constant throughout the data set as it won't make a difference in the training and filled the gaps or missing values by interpolating. Lastly, we scaled the whole data set using a robust scalar. This data set was then used for training a tree based classifier.

2.3 Model Training

Our model is a binary classifier: it classifies every lightcurve into two classes i.e. 'planet candidate' or 'false positive'. We used a gradient boosted tree model using popular machine learning framework (lightGBM Ke et al. 2017) for our classifier. Gradient boosted tree model is an ensemble of decision trees which are trained in sequence. In each iteration, GBDT learns the decision trees by fitting the negative gradients (also known as residual errors Friedman (2001)).

Moreover, we used a 10 fold cross validation (10-fold CV) during training. This means the training set is split into 10 smaller sets and the model is trained using 9 sets at a time. The remaining last set is used to evaluate the model performance. This is done iteratively until the model is evaluated on all 10 sets. The final performance is then the average of all the values computed in the iteration.

We will use the following four metrics to evaluate our results:

- AUC: The area under the curve (AUC) tells us what % of times a planet candidate is ranked higher than a false positive.
- Recall: What % of total planet candidates in the validation set is recovered.
- Precision: Out of the total predicted planet candidates by model, what % are actually "planet candidates".
- Accuracy: The % of samples correctly classified, including both planet candidates and false positives.

We argue that out of the above four metrics AUC and Recall are most important as it is important for us to have a system which recovers a high percentage of lightcurves containing planet signals. In other words, we want to optimize our model for not missing

Table 1. Results on simulated data

AUC	Recall	Precision	Accuracy
0.92	0.92	0.94	0.91

lightcurves with planet signals i.e. a planet with high Recall on the planet class. This is important to consider as a model with high Recall might lead to a poorer Precision and vice versa. This is commonly known as Precision-Recall trade-off. For our use case, we rather have higher number of false positives than losing possible planet signals. Hence, our model is trained for a higher Recall as opposed to Precision. Lastly, accuracy is not a proper metric for our use case since most of the exoplanet detection datasets are unbalanced i.e. usually there are a lot more lightcurve without any planet signal than the cases with a planet. For instance, in our TESS dataset only 3% lightcurves are planet candidates, and if a classifier predicts "Non-planet" for all the cases it will still lead to an accuracy of 97%. Even though 97% appears to be a high accuracy but we know the classifier is not able to identify any planet candidates. Due to these reasons, our model is optimised to maximize the AUC.

3 RESULTS

3.1 Simulated data

In order to attain a proof of concept the method was first applied to simulated data. Since the data was derived from K2 lightcurves, most curves had gaps. We filled the missing data points by interpolating between those gaps. Overall, we used 7876 light curves from the K2 mission and removed all known planet signals from them. Then we removed 3σ outliers, i.e. discrepant points like cosmic ray hit's) and flattened the light curves. Lastly, we randomly injected transit signals in half of the lightcurves. The lightcurves were then used for feature extraction and for training a model to detect the cases with injected signals as described in chapter 2. Our training set consisted of 5907 samples and we used the remaining 1969 samples to evaluate it's performance. Our validation set consisted of 970 cases with an injected random transit and 999 cases without it. The model outputs the probability of a lightcurve containing the transit. Typically, for such a classification problem the default threshold is 0.5 i.e. if the model predicts a probability higher than 0.5, it is considered as a predicted planet. However, we decided to optimize that threshold according to Figure 4. In this case, choosing a threshold of 0.13 make gives us a Recall of 0.92 and a Precision of 0.94. This result is preferred over the ones with standard threshold of 0.5 as for our problem, it's preferred to extract the maximum possible lightcurves at the cost of a minor increase in false positives. The results on our validation set are shown in the Table 1:

We then used BLS on our simulated dataset. With Box-fitting Least Square (BLS), we were able to detect around 84% of total planets as opposed to 92% with our method. Overall our machine learning model failed to identify around 78 (out of 970) cases, while with BLS we failed to detect 155 cases. We also found an overlap in the cases that were not detected by our method and BLS. We investigated these cases manually and found that the transit's were injected randomly with random parameters which resulted in many non-detectable transit's such as:

- Cases with relatively low inclination angle for the given star, which resulted in very small planet signals.

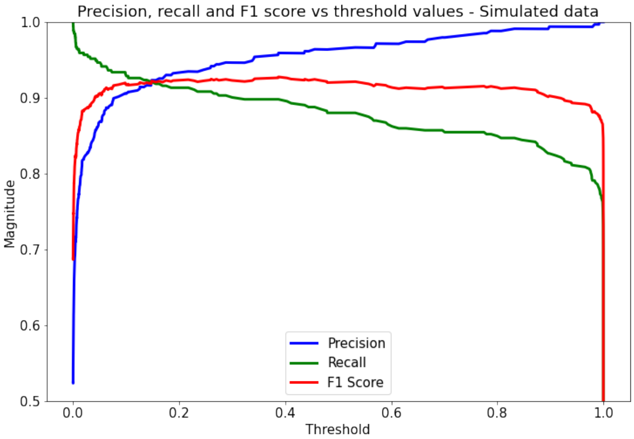

Figure 4. Precision, Recall and F1 Score (harmonic mean of Precision and Recall) against the prediction probability predicted by our model for test set of out simulated data. Typically, a threshold of prediction probability 0.5 for classification problems. This threshold can be adapted to increase or decrease the model sensitivity or depending on the problem at hand. In the above case, choosing a threshold of 0.13 make gives us a Recall of 0.92 and Precision of 0.94. As we want to retrieve maximum possible lightcurves containing transit signals, this result is preferred over the ones produced by a default threshold of 0.5.

- Cases with low S/N ratio where the injected transit signal was weaker than the noise, and hence it was not detected either by our method or by box least squares (BLS).

The majority of these cases had a S/N ratio < 12, which means that the signal strength was very little. With this, we are able to provide a proof of our hypothesis that our machine learning based method can detect planets more accurately and efficiently than BLS especially in cases with low S/N ratio. Similar results were also shown by Pearson et al. (2018) where they compared various machine learning methods with BLS. Their machine learning models were able to detect planet signals with higher signals and much lower false positive rate as compared to BLS. Therefore, our next step was to apply this method on realistic data: Kepler and TESS data whose results are shown in next sections.

3.2 Kepler data:

Shallue & Vanderburg (2018) produced very promising results on Kepler data using a deep learning model. These are also the best current results on this dataset as of this writing. They shared their training dataset publicly, which we used to evaluate our methods and used their results as a benchmark for our results. The dataset consists of 15737 lightcurves which were labelled into 3 classes for autovetter training data as PC (planet candidate), AFP (astrophysical false positive), NTP (non-transiting phenomenon) with 3600 PCs.

As discussed in the last section, we decided to optimize the threshold according to the Figure 5. In this case, choosing a threshold of 0.46 gives us a Recall of 0.96 and Precision of 0.82. This result is preferred over the ones with standard threshold of 0.5 as for our problem, it's preferred to extract maximum possible lightcurves at the cost of a minor increase in false positives. A comparison of the results from our method with Shallue & Vanderburg (2018) is summarised in Table 2. It can be seen that the results of both the methods are comparable even though the precision of our model is

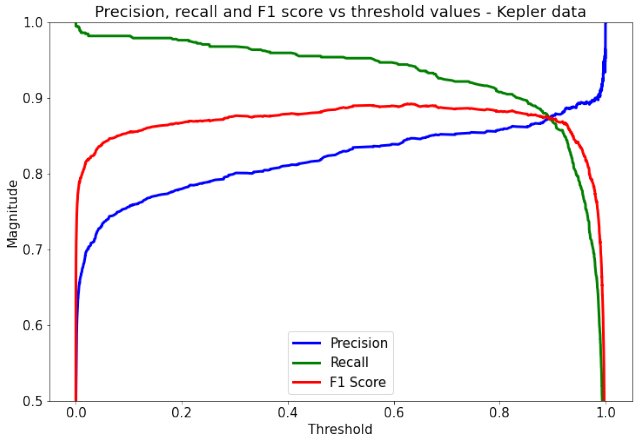

Figure 5. Precision, Recall and F1 Score against the prediction probability predicted by our model for test set of our Kepler dataset. Typically, a threshold of prediction probability 0.5 for classification problems. But this threshold can be adapted to increase or decrease model sensitivity or depending on the problem at hand. In the above case, choosing a threshold of 0.46 gives us a Recall of 0.96 and Precision of 0.82. As we want to retrieve the maximum possible Kepler targets, these results are preferred over the ones produced by a default threshold of 0.5.

Table 2. Results on Kepler data

Type	AUC	Recall	Precision
Shallue & Vanderburg (2018)	0.988	0.95	0.93
Our method	0.948	0.96	0.82

lower than the benchmark as it is least important for our use case as discussed earlier.

This is the first real dataset we encountered and we went further and plotted the lightcurves in a lower dimensional 2D space using the T-SNE (Maaten & Hinton 2008) algorithm in the Figure 6. To do this we extracted time series features from each lightcurves as described in section 2.2. These features were then scaled using a robust scaler, any missing values (NaNs) were filled using the mean of that column. This resulted in around 700 processed features. Now, top 70% of those features were selected based on the feature importance from the lightGBM model. This was done as after hypertuning the model, optimal value of the parameter *feature_fraction* was found to be 0.70. Which implies that the model performs best when only 70% of the features are taken into account at once. Hence, we selected top 70% features with highest information gain (or feature importance) and used t-sne on this data for dimensionality reduction, which is plotted in the Figure 6.

The points closer in the plot are also closer in the high dimensional space. However, the point far away in the 2D space might not be far away in high dimensional space. Based on this, we can argue that most of the lightcurves containing a transit are clustered in one region of the plot. This also validates our pre-processing and feature extraction process. Broadly there are 3 clusters in the plots and samples from these clusters can be analysed to understand how these clusters vary from each other. However, we left this part for future steps and continued validating the model on more dataset. We also applied our approach to the TESS data and it's results are presented in the next section.

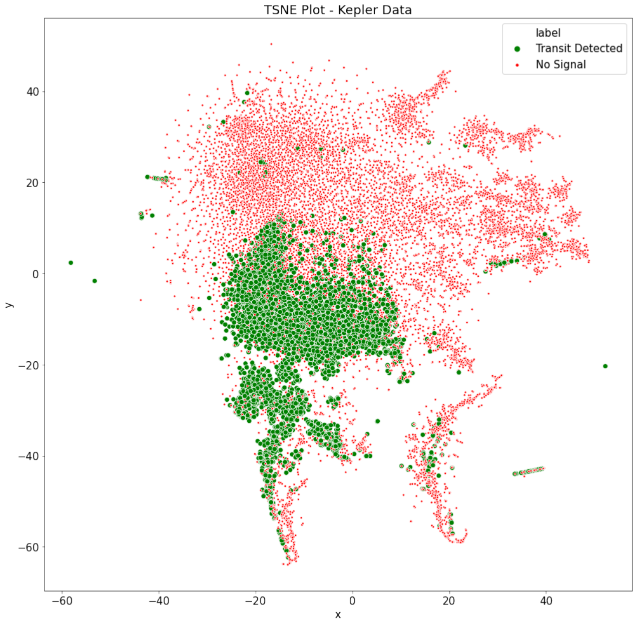

Figure 6. The above figure shows TSNE plot of Kepler data. Lightcurves with planet transit signal(s) are shown in green. It can be observed that these curves are clustered in one region of the space.

3.3 TESS data

3.3.1 Kepler vs TESS data

TESS lightcurves are in many way different from the Kepler data set. Kepler observed a fixed field of view in it's 4 year timeline and K2 observed each of it's 19 campaigns for 80 days. On the other hand, TESS observed each sector for 27 days. Longer baselines contain more details and lead to a much higher signal to noise in terms of detections. The short time span of TESS makes it harder for any classifier to detect planet signals as each lightcurve (and features extracted from it) tends to contain less data points from transit's, making it harder for a classifier to differentiate a case with planet signal from a case without it. Moreover, the short length of lightcurves makes the presence of multiple-periodic transit signals less likely to be observed. For longer exoplanet transit's, this problem is further compounded if only a single transit is recorded. In contrast, multiple transiting planets in the Kepler data may lead to an automatic confirmation due to the low probability of any fitting false-positive scenario.

Yu et al. (2019) introduced the first machine leaning classifier to be trained and tested on real TESS data. Their model is an adapted version of Shallue & Vanderburg (2018) as shown in Figure 2 of their paper. We again trained our model on the dataset publicly shared by Yu et al. (2019) and used their results as benchmark. The data consists of 16,500 lightcurves, with only 490 planet candidates. As discussed in last two section, we decided to optimize that threshold according to the Figure 7. In this case, choosing a threshold of 0.12 make gives us a Recall of 0.82 and Precision of 0.63. As explained earlier, this result is preferred over the ones with standard threshold of 0.5. A comparison of results from both the methods with and without optimal threshold is shown in the Table 3 and 4 respectively.

With the optimized thresholds, our model was able to find 40 out of 49 curves with planet transit in our test set as opposed to 44

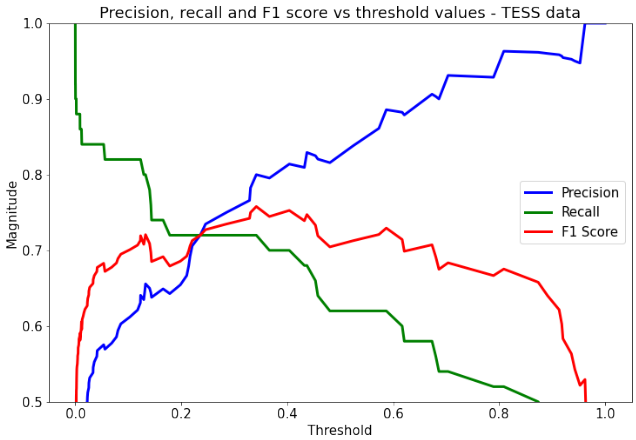

Figure 7. The above figure shows Precision, Recall and F1 Score (harmonic mean of Precision and Recall) against the prediction probability predicted by our model for test set of our TESS dataset. Typically, a threshold of prediction probability 0.5 for classification problems. But this threshold can be adapted to increase or decrease model sensitivity or depending on the problem at hand. In the above case, choosing a threshold of 0.12 make gives us a Recall of 0.82 and Precision of 0.63. As we want to retrieve maximum possible TESS targets, these results are preferred over the ones produced by a default threshold of 0.5.

Table 3. Results on TESS data with default threshold

Type	AUC	Recall	Average Precision
Yu et al. (2019)	0.984	0.5	0.69
Our method	0.80	0.61	0.83

Table 4. Results on TESS data with optimized threshold

Type	AUC	Recall	Average Precision
Yu et al. (2019)	-	0.89	0.44
Our method	0.80	0.82	0.81

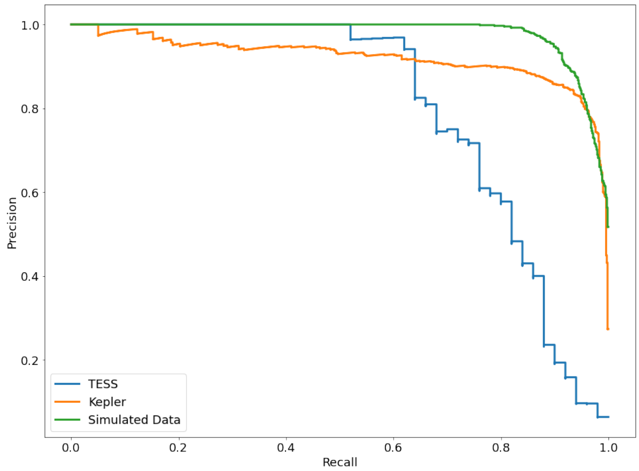

Figure 8. Precision vs Recall curve for test sets of TESS, Kepler and simulated data sets. It shows how the tradeoff between Precision and Recall varies in all three cases. We can see the performance on simulated and Kepler data is similar, there is slight increase in the number of false positives in the Kepler dataset. This is because Kepler dataset is more noisy than the simulated one. One the other hand, performance on TESS dataset is worst in all three cases, and high class imbalance is the primary factor for this. With a more balanced dataset, performance on TESS data is expected to approach the other two cases.

samples identified by Yu et al. (2019) on this test set. On the other hand, average Precision of our model is nearly double as high i.e. it will result in nearly a third of false positives as encountered by Yu et al. (2019). With default threshold, the deep learning method from Yu et al. (2019) produced a much higher AUC than our model but only produced a Recall of 0.5 that is, around 50% of all planets was found by the model. On the other hand, our machine learning method is able to Recall up to 61% of the cases. We can see that recalling 61% is still not ideal and there is a lot more work to be done before such system can be used in production. In this case, class imbalance is one of the biggest problems the model is facing. Now that we are finding more and more planets from TESS data, we should be able to gather more data which is expected to improve our results. This is also discussed in chapter 4. Even though we can't use it for production, we can certainly use this model for reliably eliminating false positives, so that we implement our current methods on a smaller set and humans vetters do not have to waste time on cases which are a clear false positive.

Performance of all three dataset can be compared from the Precision-Recall curve of all 3 datasets is shown in Figure 8. We can see the performance on simulated and Kepler data is similar, there is slight increase in the number of false positives in the Kepler dataset. This is because Kepler dataset is more noisy than the simulated one. One the other hand, the performance on TESS dataset is worst in all three cases, and high class imbalance is the primary factor for this.

4 DISCUSSION

4.1 Classical Machine Learning vs Deep Learning Approaches

One of the primary differences in the classical machine learning (CML) and deep learning (DL) methods is the fact that DL models are able to automatically extract useful features from raw data. On the other hand, CML methods usually can not deal with raw data and needs extracted features. Usually CML models don't work well when problems becomes complex such as language translation. In those cases we deep learning is pushing new frontiers. However, machine learning methods shouldn't be ruled out as they are much more efficient and can still produces good results for some problems.

Our CML approach has the following advantages over state-of-the-art deep learning methods:

- CML models can work with only a global view of the lightcurve. On the other hand, DL methods additionally require folded or secondary views.
- CML model training is less time consuming and takes less than 5 minutes to train on a 2 CPU (central processing unit) system. This also indicates that it can be quickly adapted to a new data source. On the other hand, DL models can take upto 5 hours for training and much longer to hypertune.
- Exactly same model setup and code can be used for data from different sources such as Kepler, K2 and TESS etc. There is only need to tune the parameters for the optimal parameters. However, DL models almost always need changes when the data source changes.

- For the CML model, no special hardware such a graphical processing unit's (GPUs) are required for training.

On the other hand, our approach has the following disadvantages over state-of-the-art deep learning methods:

- The biggest disadvantages is the lower performance. When DL models are properly trained, they usually leads to better results.
- Time series data such as the global view, and folded view can be directly used as an input in the DL model. While ML model require extracted features from the time series data.

4.2 The Vetting Tool

Machine learning methods are still relatively new in astrophysics and not yet fully accepted in the exoplanet community. There is a new paradigm of building classification techniques and it still has a long way to go before being used in production pipelines. One major factor is the non-deterministic or black-box nature of such methods. Unlike conventional algorithms, the outcome of such methods can't always be understood due to the non-linear and stochastic nature of such methods. It's harder for the user to explain the reason behind predictions made by the model. Even though these models can outperform conventional methods like BLS, they do not always make correct predictions which is another contributing factor for this. Hence, it is important to note that they cannot completely replace human vetting experts. However, if they are supported to be used alongside the domain knowledge, in that cases these techniques can help automate the processes like planet detection and enable us to deal with the growing data size in astronomy.

These tools are not a commonplace yet but for these systems to progress further, it is important to take feedback and expertise from the community on these methods. For this reason, we have developed a vetting tool where a few results from our model can be explored in an interactive way without any knowledge of the underlying machine learning method or the feature calculation technique. The tool is hosted at `https://github.com/abhmalik/Exoplanet-Vetting-Tool`.

The vetting tool allows the user to explore the cases where model assigns a high probability of being a planet candidate. It allows the user to view the lightcurve in global view and folded view (also called local view). It also gives a list of features that were important in the model prediction for that case. It can be easily used to make model inference on any new lightcurve.

4.3 Next steps

With this paper, we tried to provide a new direction in order to make a light and efficient automatic vetting system. However, there is a lot more that can be done in this direction. We list some of the suggested future steps below:

- Since our method is able to work with data from different sources such as Kepler, K2 and TESS etc. It is possible to construct a global classifier that can intake lightcurves of any length from any data sources. This enable us to use the same model in case the data format changes as happened in case of K2 when lightcurves had a time gap.
- As we saw, the performance of our model was worst for TESS out of all three datasets. The primary reason for this is the high class imbalance in TESS data i.e. less than 3% lightcurves contain transit signals and most of them are just single transit. Now that more and more planets are being confirmed from TESS data, we believe we can construct a dataset with better class balance which might help us to reach the performance levels we are reaching with the Kepler and the simulated data.
- As shown in the TSNE plot in Fig. 6, lightcurves are grouped in different regions. We can extract samples from these groups and analyse how the light curves in different group differ. This information can further be used to optimize the model performance for that particular dataset.

5 CONCLUSIONS

Machine learning methods have seen very active development in the last decade and now they are an essential part of our way of working. In fact, for everyone who interacts with a computer or smartphone, it's highly likely to interact with some sort of machine learning programs. They are also widely used in sciences for several use cases such as detecting diseases, creating new samples without long simulations, and finding string models in a string theory landscape etc. In astronomy, we have seen several application such as Galaxy Zoo (predicting galaxy morphology), identifying gravitational waves, and gravitational lens. With new and advanced telescopes, data in astronomy are growing at a fast pace. Conventional methods that involve human judgements are not efficient and prone to variability depending on the investigating expert. For example, the commonly used method for exoplanet detection, BLS, produces large number of false positives in case of noisy data, which has to be reviewed manually.

In this paper we proposed a novel planet detection methods based on classical machine learning. Our method consists of automatically extracting time series features from lightcurves which are then used as input to our machine leaning model. We were able to demonstrate that with our machine learning method, we could identify lightcurves with planet signals more accurately as compared to BLS while significantly reducing false positives. Some notable works in the area are based on Shallue & Vanderburg (2018) and use deep learning. Although deep learning methods are superior to classical machine learning approaches specifically in complex problems such as machine translation and object detection. However, such models are harder usually require a long training time on special hardware (graphics processing unit's or GPUs). They usually require large sets of training data and are more difficult to hypertune as they have to navigate a much bigger parameter space as compared to classical machine learning models. Hence, in this work, we attempted to move away from the standard deep learning approach and introduced a new direction that provides a light weight model.

The deep learning models require processed views of the data such as *local view* and *secondary view* along with the *global view*. On the other hand, our approach only requires the global view as input and automatically calculates the required features out of it, which are then used as input to the model. Our model is able to distinguish important features from unimportant ones and there is no need for manually selecting features. Moreover, our approach does not require special hardware like GPUs and it takes about 2 minutes to train our CML model on a dual core CPU system which means it can be easily trained even on low end computers and can be revised quickly in case it's required.

We compared the performance of our results with the state-of-the-art models and found that, on the Kepler data, our model is able to achieve comparable results and on the TESS data our model is able to out perform the deep learning model, after being trained on the same amount of data. However, the results are still not yet robust enough to be broadly applicable. Even if a machine learning model works well in the development environment, the models are prone to make mistakes on unseen data and such methods should be used alongside with human supervision. Nonetheless, at the current stage, these models can provide a very reliable system to rule out large number of false positives and can drastically reduce the number of cases requiring manual reviews.

ACKNOWLEDGEMENTS

We are grateful to Arno Riffeser and Jana Steuer for useful discussions. We would also like to thank Christopher Shallue and Liang Yu for publicly sharing their code and training data. This enabled us to quickly test our model on different datasets and provided a benchmark to compare our results.

We thank the developers of Lightkurve (Lightkurve Collaboration et al. 2018), NumPy (Walt et al. 2011), Matplotlib (Hunter 2007), Scikit-learn (Pedregosa et al. 2011) and TensorFlow (Abadi et al. 2015) for their very useful free software. The Astrophysics Data Service (ADS) and arXiv preprint repository were used intensively in this work.

REFERENCES

Abadi M., et al., 2015, TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, https://www.tensorflow.org/

Borucki W. J., et al., 2010, Science, 327, 977

Catanzarite J. H., 2015, Technical report, Autovetter Planet Candidate Catalog for Q1-Q17 Data Release 24. KSCI-19091-001 https://exoplanetarchive. ipac. caltech. edu/docs/KSCI-19091 ...

Chaushev A., et al., 2019, Monthly Notices of the Royal Astronomical Society, 488, 5232

Christ M., Braun N., Neuffer J., Kempa-Liehr A. W., 2018, Neurocomputing, 307, 72

Coughlin J. L., et al., 2016a, The Astrophysical Journal Supplement Series, 224, 12

Coughlin J. L., et al., 2016b, The Astrophysical Journal Supplement Series, 224, 12

Dattilo A., et al., 2019, The Astronomical Journal, 157, 169

Friedman J. H., 2001, Annals of statistics, pp 1189–1232

Howell S. B., et al., 2014, PASP, 126, 398

Hunter J. D., 2007, Computing in Science & Engineering, 9, 90

Jenkins J. M., et al., 2010, The Astrophysical Journal Letters, 713, L87

Ke G., Meng Q., Finley T., Wang T., Chen W., Ma W., Ye Q., Liu T.-Y., 2017, in Guyon I., Luxburg U. V., Bengio S., Wallach H., Fergus R., Vishwanathan S., Garnett R., eds, , Advances in Neural Information Processing Systems 30. Curran Associates, Inc., pp 3146–3154, http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf

Kovács G., Zucker S., Mazeh T., 2002, A&A, 391, 369

Lightkurve Collaboration et al., 2018, Lightkurve: Kepler and TESS time series analysis in Python, Astrophysics Source Code Library (ascl:1812.013)

Maaten L. v. d., Hinton G., 2008, Journal of machine learning research, 9, 2579

McCauliff S. D., et al., 2015, The Astrophysical Journal, 806, 6

Mislis D., Bachelet E., Alsubai K., Bramich D., Parley N., 2016, Monthly Notices of the Royal Astronomical Society, 455, 626

Obermeier C., 2016, Searching for hot Jupiter transits around cool stars, http://nbn-resolving.de/urn:nbn:de:bvb:19-199636

Obermeier C., et al., 2016, A&A, 587, A49

Pearson K. A., Palafox L., Griffith C. A., 2018, Monthly Notices of the Royal Astronomical Society, 474, 478

Pedregosa F., et al., 2011, Journal of Machine Learning Research, 12, 2825

Ricker G. R., et al., 2015, Journal of Astronomical Telescopes, Instruments, and Systems, 1, 014003

Shallue C. J., Vanderburg A., 2018, The Astronomical Journal, 155, 94

Vanderburg A., Johnson J. A., 2014, PASP, 126, 948

Walt S. v. d., Colbert S. C., Varoquaux G., 2011, Computing in science & engineering, 13, 22

Wolszczan A., Frail D. A., 1992, Nature, 355, 145

Yu L., et al., 2019, The Astronomical Journal, 158, 25

This paper has been typeset from a TeX/LaTeX file prepared by the author.