Exoplanet Vettor

Mini-Project

by

Tanmay Janbandhu
Roll. No. 200121021

under

Prof. Girish Setlur

**Department of Physics,
Indian Institute of Technology Guwahati
Guwahati – 781039, Assam, India
Jan-April 2023**

Author Note

This is a report for the Mini-Project in Semester 6 of Engineering Physics (2024 Batch) produced under supervision of Prof. Girish Setlur , IIT Guwahati. In this project I have tried to train a Machine Learning Model for classifying Exoplanets into Planet Candidate or False Positives. I have uploaded the workfiles regarding this mini Project in Drive folder.

Acknowledgement

I would like to thank Prof. Girish Setlur for considering me to do a mini-project under him and constantly asking to do something which could contribute to the community. This provided me with motivation to learn different things which would have been useful in this project.

I would also like to thank my friend Mihir Kumar singh , who always helped me out when I was stuck on something. I wouldn't be able to complete this without his help.

Lastly I would like to thank the developers of Lightkurve (Lightkurve Collaboration 2018) for their free software and data which were essential for data Visualization and Processing and Kaggle users for Datasets and insights.

Tanmay Janbandhu

April 29,2023

Contact (r.janbandhu@iitg.ac.in)

Abstract

This mini-project is focussed on analyzing the lightcurves obtained from kepler during its Operation.

Using various libraries such as astropy , matplotlib and Lightkurve I have obtained the timeseris of the flux data and extracted features out of it using the *TSfresh*, an open source feature extractor which provided us with ~784 features for our time series. Using all these features for training the XGBoost model I tested it for major discoveries using the *Astronet model*.

Keywords: exoplanets, Data Analysis

CONTENTS

1. Introduction

2. Data Collection.

3. Lightcurve Processing for the dataset

4. Model Training

5. Results

6. Discussion

Exoplanet Vettor

Introduction

In recent times, the search for exoplanets beyond our solar system has been a primary focus of astronomers. The exploration of new exoplanets with the possibility of sustaining life is a thrilling prospect given the vastness of the universe. Nevertheless, detecting and examining exoplanets poses a formidable challenge, often requiring complex equipment and methods. Machine learning, on the other hand, has emerged as a promising technique for exoplanet detection and classification. It leverages algorithms that can recognize patterns in extensive datasets and assists astronomers in analyzing large quantities of astronomical data to identify prospective exoplanets more efficiently than traditional methods. This report looks into the use of machine learning in exoplanet detection, including the associated challenges and opportunities in this fascinating field of study. Major focus of this project is to analyze the TCE produced by Kepler during its operation and thereby use the transit method (Charbonneau et al. 2000; Naef et al. 2001) for detection of exoplanets in this dataset.

1.1 Background

The various ways of detecting the exoplanet are through Radial Velocity (Campbell et al. 1988; Latham et al. 1989) , Transit Method (Charbonneau et al. 2000; Naef et al. 2001), Direct imaging(Chauvin et al. 2004; Thalmann et al. 2009) and Gravitational Microlensing (Mao & Paczynski 1991; Beaulieu et al. 2006)..Out of this radial Velocity and transit velocity can be said to be most successful . The Kepler Dataset I am using for this project is tracking the threshold-Crossing events in space. TCE is a transit-alike feature sequence with respect to time which is quite similar to sequences made by a transiting planet. Transit method observes the dip in the brightness of traversing star when a planet or astronomical body passes in between of the kepler and the target star.This flux data time- series is then fitted into Box-fitting Least Squares (BLS) and using this various parameters such as period and duration of transit, depth of transit and signal to noise ratio. Using this data Astronomers further calculate Radial and orbital distance with Planet density which could be helpful to figure if the planet is habitable and what kind of minerals it possesses.

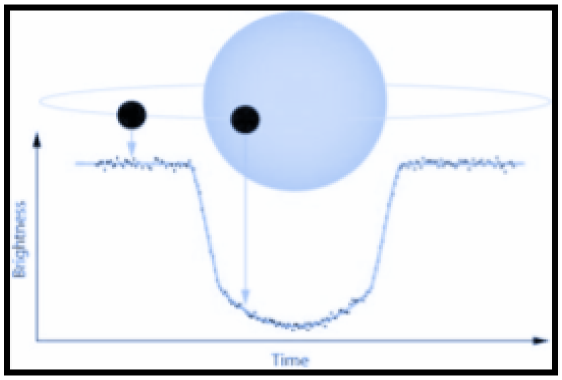


Figure 0. Brightness vs Time plot observed during the the transit of planet on any target star

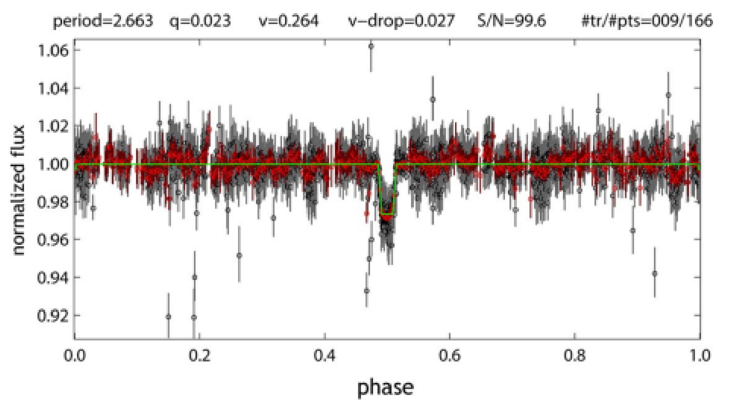


Figure 1. Phase-folded light curve of a K dwarf being orbited by a hot Jupiter candidate, observed by the Pan-Planets survey. The Box-fitting Least Squares (BLS) method has been used to fit the light curve. Shown at the top are the resulting parameters: period (days), transit duration q (in units of phase), transit v shape (0 corresponds to a box, 1 to a V), transit light drop, signal-to-noise (S/N) and number of transits/number of points in the transits. Binned data points are shown in red, while the green line shows the best-fit model. See Obermeier et al. (2016) for more details.

The Detection of exoplanets required Large number of Experts to confirm the status by analyzing the flux data time series of the TCE, and then it gets confirmed by TCERT. Due to the large number of experts involved to confirm a single Exoplanet, a need to automate the Classification process was required. So Using machine learning a way to classify the planets into Planet candidates and false Positives was used in the classifier named Autovettor. Following this many classification models using KNN, SVM were looked into and another classifier named robovettor was invented.

1.2 Machine Learning Models

Autovettor((McCauliff et al. 2015) is a ML based classifier which uses Random forest trees to classify the TCE into Planet candidate(PC) , False Positives (FP) or Non Transiting Planet(NTP). To train this Random forest model use attributes from the satellite data and dataset created by the TCERT(Terrestrial crossing events Review Team).
Robovettor(Coughlin et al. 2016b) is similar to autovettor , provides the disposition score for each of the TCE and helps classifying them into 4 categories.Recently the robovettor disposition has been also evaluated with synthetic data which consist of noise and thereby training the model to avoid some noise effect from the test Data. Another Model named SIDRA has been developed but was not that widely used.
The Astronet model by Shallue & Vanderburg (2018) is the most remarkable deep Learning model which uses CNN for analyzing the Light Curves and predicting the Exoplanet. This model has successfully confirmed planet Kepler 90i and so is the most successful model yet. Recent research has been going on to use this model for the data of various other Satellites such as TESS and K2.

Method

I have used Kepler data from NASA Data Archive with a predisposed Autovettor Score and Then used the kepler ids of Possible planet Candidates and False Positive to download the lightcurves of the respective stars. The Light Curve Now is needed to be Stitched and Normalised within each Quarter of Kepler operation. This will Provide us with

our Normalized Flux vs time plot for light curve Which is similar to the plot shown in Figure 1.
Then instead of sticking to the traditional Box Least Square Fitting to extract features of the time series We have used TSFresh , a Open Source Python Library which is useful to extract features of the Flux Time Series. This will extract nearly 1500 which we can then use to Train our XGBoost model and then Get our final Results.

Data Collection

From the NASA Data Archive download the Q1-Q24 DR24 dataset. I have preferred this dataset because it does not contain any Injected Synthetic Data points which have been processed by the robovetter Disposition and are available in Q1-Q24 DR25. Instead we download these Datasets with Autovettor Disposition. Another Reason to Choose Autovettor is that Robovettor Disposition Characterises some planets as False positives and this would just reduce the training data points of few known planets till date.

While downloading the data Points the following columns are must haves in the Data set:

- Kepler ID:

  Kepler ID is the identification marker for each of the TCE in the Data Set. It plays an important role when we download Light curve Corresponding to i and also when we get the final lightcurve flux array we assign if Planet Candidate or False positive Depending upon this Kepler Id.

- Tce_plt_num: This correspond the number of star for which we are observing the TCE
- Period(in days): Total Time period before the Quarterly change of orientation
- time(BJD); first detected event in BJD-offset

- Tce_duration:(in hours): Essential for Processing Light curve
- Av_training_set : Contains the Disposition of Planets as Planet Candidate (PC) or False Positives(FP)

  Using the Autovettor training set we use, Exclude all the Non transiting Planet (NTP) marked kepler Ids. And Now we are just left with False Positives and Planet Candidates. After this We could move ahead on downloading the Light Curves for the Data.

LightCurve Processing

To Download and Processes Light curves I have used An open Source software named LIghtKurve(Lightkurve Collaboration 2018). Each Kepler Id has a corresponding Lightcurve to it containing ~70000 data points each. This data About the Light Curves is uploaded on the MAST portal. We will have approx 9000 data with each of them having ~70 thousand data points in them, this makes the total data set size up to 90 Gbs. Making it difficult to download this Data . To tackle this problem I decided to use Kaggle dataset (flux-time series) to train my model and extract selected few kepler ids for testing. Starting with lightkurve, we first import libraries and then search and download the respective light curves. The data from Kepler can be found with two cadences. Cadence can be termed as the experiment time over which exposure is monitored. In Kepler, the short Cadence is data collected over exposure of 1 minute and the Long cadence the data is collected over exposure time of 30 minutes. For our model we consider the [1]Long Cadence as it provides longer exposure which could consider a long orbital period for some planets that could be missed out from short cadence.
After Downloading all the data we plot the curve to get a visual representation of the light curve.

[1] Figuring the cadence of the Dataset from Kaggle was difficult to predict as my calculation was not mating the exact size of consecutive fluxes.This was also the reason why the even after processing our own dataset we were unable to get accurate bins for final flux generation.

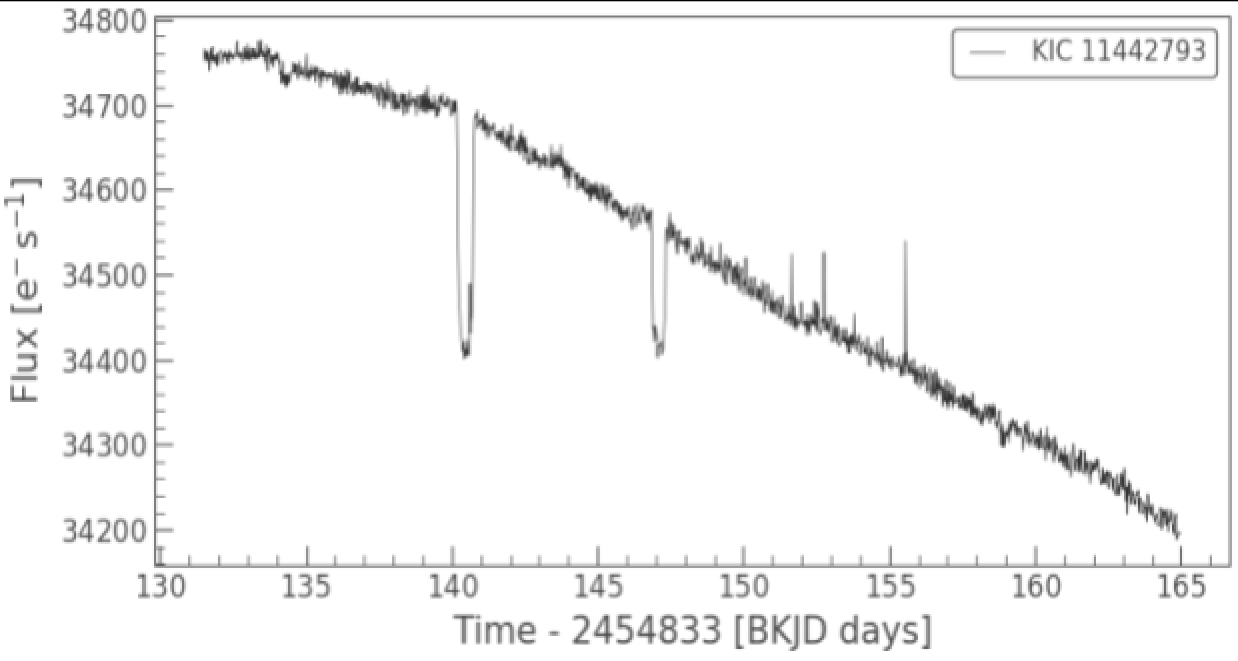


Figure 2. Lightcurve for KIC 11442793 (kepler 90)

This is the lightcurve plotted for a 1st Quarter of the 14 Quarter of TCE observed for this Kepler ID. Taking the Example of this Curve we see that the Brightness has been decreasing over the time and also there are some sudden spikes in the brightness which we can categorize mostly into the noise due to Cosmic radiation or stellar Debris. In order to remove this Noise we use the Remove outlier in which we neglect any value which is greater than specified standard Deviation from mean Value. Before doing all this we stitch all the flux Values from different quarters.After doing this the data is folded over its period.

To extract the final Lightcurve out of this we first need to normalize flux[2] from each Quarter in its own range and then stitch it together to get a normalized Flux vs time plot. In order to fit the Box least fitting Square(BLS) such format of the Light Curve is required so we can extract maximum data of these Curves.

[2] The flux mentioned here refers to the Simple Aperture Photometry flux (SAP flux).

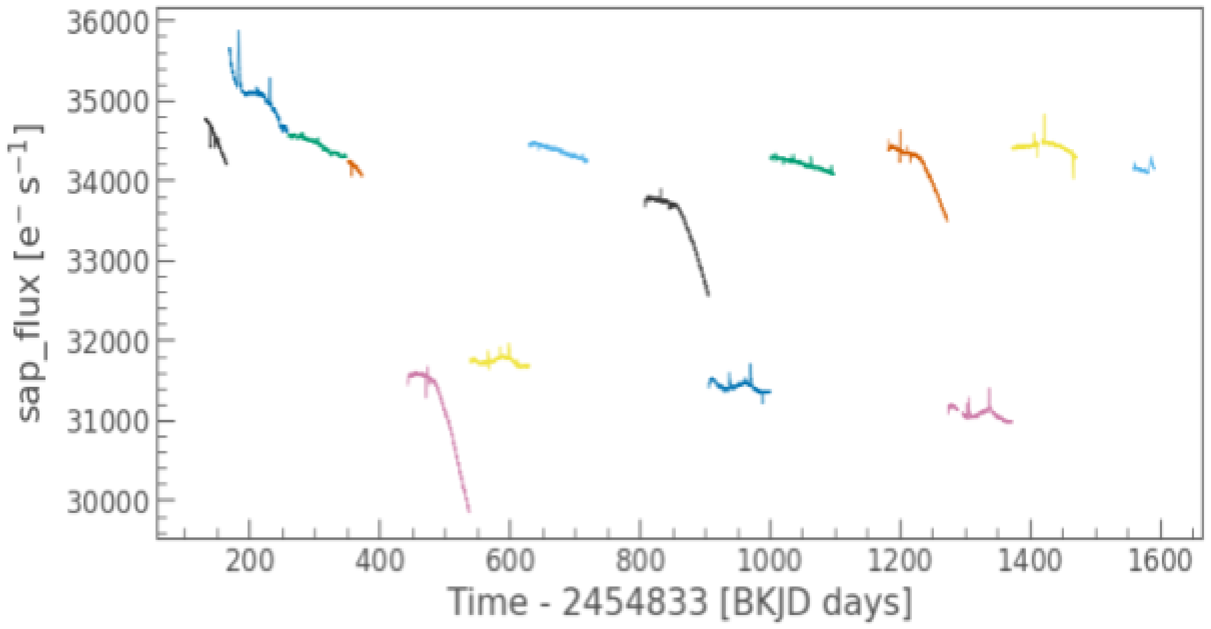


Figure 3. Flux timeseriesof different Quaters plotted in a single plot.

The final Plot is depicted in figure 4.The folding over the time scale is an important factor to get a continuous lightcurve, for which time period, and transit epoch are a contributing factor and reason we need them in the dataset.

The Light Curve processing I have done is not exactly the same as the previous Work done in this field which focuses on generating two Views Global and a Local View. The Local View found could be very useful for working on the problem with less resources and time and could give the Comparable results. But this Doesn't agree with our flux time series dataset downloaded from Kaggle

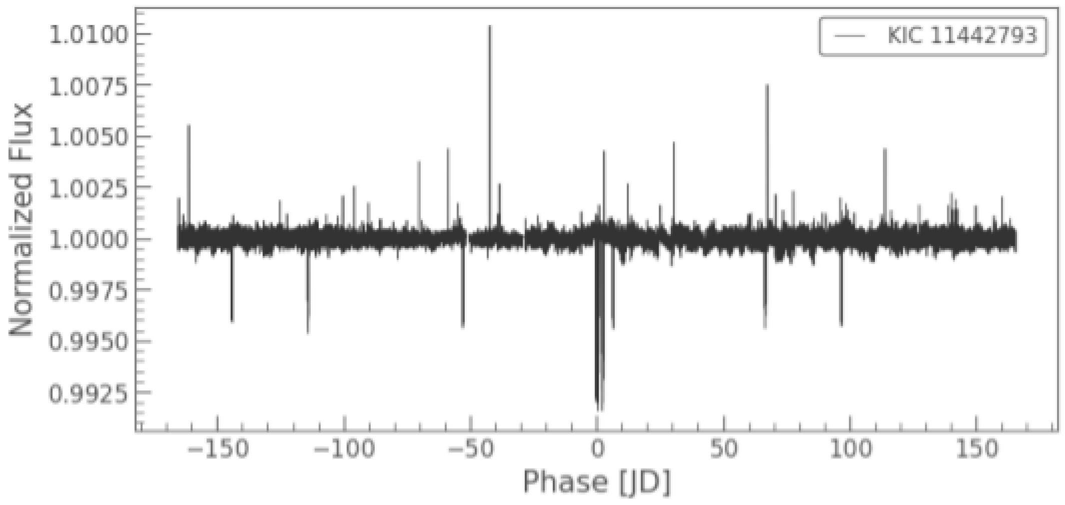


Figure 4. This is the Normalised stitched lightcurve we got after processing of the Light curve.

After all of this processing , I have exported the flux data in the form of a numpy array out of this. There are many data points in a light curve so we have also binned the data points by the factor of 20 so we get the final data points ~3500.

Binning is the process in which we Take mean of consecutive factor terms and then plot it accordingly.

The Code to do do most of this preprocessing part is provided below,

```
def flux_generator(k_id,period,t,transit_duration):
  # k_id = Kepler ID , Period = Orbital Period , t= Transit Epoch ( in BKJd)
  # searching and Downloading Lightcurve
  lcs = lk.search_lightcurve('KIC'+ tostring(k_id) , author='Kepler',
cadence='long').download_all()

  lc = lcs.stitch()# stitching data of different Quarters
  lc = lc.remove_outliers(sigma=20, sigma_upper=4) #removing all attributes
above mean level here we are just using upper sigma as if we use downward
sigma it may affect our transit.
```

```
  folded = lc_clean.fold(period, epoch_time=t) # folding over time to remove
discontinuity
  final= folded.bin(binsize=20) # reducing the number of data points

  flux= final.flux() #  flux values to an numpy array
  return  flux
```

We would get an output of Flux time series from the code provided above which we will be using to extract features of the time series. After this step we Could use Either Box Least fitting Squares , but we will extract features using TSFresh Library.

Model Training

Downloading all the lightcurves and extracting data out of it is Quite an Memory and GPU demanding process.So I have used flux time series available on Kaggle[3] for training this data set. This data set on plotting was not processed and so I have tried to process it to get our data favorable Light curve flux. For this I have used sklearn Preprocessing. I have basically tried to normalize the data. In Processing of this data binning it could have reduced feature extraction time.

After getting normalized flux time series. We move forward with using the TSFresh features extractor. So we took the transpose of the flux time series and reset the index of the dataframe to normal numbering. Changing Index is an important part as we will feed it as an index of time while extracting the Features.

```
dt = pd.melt(df.reset_index(), id_vars='index', value_vars=df.columns)
dt = dt.rename(columns={'index': 'time', 'variable': 'id', 'value': 'flux'})
dt['time'] = dt['time'].astype(int)
```

[3] This data set has a time series without a time index and consists of consecutive flux , so we need to change the index of data to get continuous time series data.

As we have now achieved the format suitable for the *Tsfresh* we then extract features using this.

```
extracted_features= extract_features(dt, column_id='id', column_sort='time')
impute(extracted_features)
```

Extracting features is a long process`and will take hours for just a few hundred lightcurves.

Why use Tsfresh?*[4]

- We could get information about which features are useful in the detection of light curves, if we study the SHAP. This could be helpful for researching theoretical physics models which might also help in the Deep learning based AstroNet..

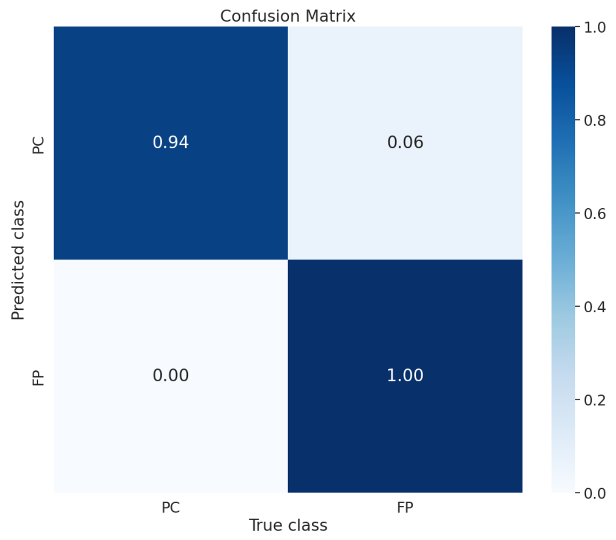
Using these features I scaled them using the Robust Scalar as a standard Machine Learning Practice. And then went on to Training the XGBoost Model.
In the XGboost model we have considered a logistic function to classify the flux series into Planet Candidate or false Positive.

```
xgbc = XGBClassifier(objective = 'binary:logistic', n_estimators= 1000,
scale_pos_weight=3.37, max_depth= 4, subsample= 0.8, colsample_bytree= 0.5,
gamma= 1)
xgbc.fit(x_train, y_train)
pred = xgbc.predict(x_test)
y_test_prob = xgbc.predict_proba(x_test)
score1 = [accuracy_score(y_test, pred), roc_auc_score(y_test,
y_test_prob[:,1])]
report1 = classification_report(y_test, pred)
```

[4] Not a Verified information, just my opinion on future works that could be carried out using this models

Results

The XGBoost model when tested on the test set of 400 lightcurves gave us the Accuracy of 93.78 %, and a ROC-AOC-Score of 99.69%.



```
array([[372,  25],
       [  0,   5]])
```

Based on these result we find that it has, Precision = 0.937 & F1 Score = 0.9675

The results I have obtained are comparable to the one officially published by Shallue and Vandelberg using their Astronet Model which gave Precision of 0.93 and AUC of 0.988. The accuracy of AstroNet was found to be 95%.

There might be some bias in the model due to limited training and testing Data of 1400 points while AstroNet being trained and tested on a dataset of ~8000.

Discussion

TSFresh Library can Extract many features of the Light curve which could count till 1500 So the feature Extraction has been found to detect different features in some Cases. So It is essential to minimize the number of features, except if our aim is to find new features that could be used for detecting exoplanets and build a theory on it. As per me the most important features are related to Depth of Transit, time length of the lightcurve, Frequency of similar transits occurrence.

Secondly, in the similar model by A. Malik , they used the extracted data from the light curves which was then converted to give two forms of Views: Global and local one. Based on this, every Lightcurve had Similar bins after the processing. But For the Light Curves I processed and extracted flux data from the Bins size was greater than expected values of the Data set which I acquired from Kaggle.

Further Using the SHAP analysis which is Game Theory Based Machine Learning Working Predictor, we could further Search for Features which could contribute for research of new astronomical models.

References

Shallue C. J., Vanderburg A., 2018, AJ, 155, 94

Mallick A.Exoplanet Detection using Machine Learning, 2021

Coughlin J. L., et al., 2016b, ApJS, 224, 12

Charbonneau et al. 2000; Naef et al. 2001

Lightkurve Collaboration 2018, Lightkurve: Kepler and TESS time series analysis in Python,

Astrophysics Source Code Library (ascl:1812.013)

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

TSfresh : Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests

XGBoost: A Scalable Tree Boosting System ,2016,