# Automatic Classification of Kepler Planetary Transit Candidates Using Multilayred Neural Networks

Prabath Peiris

Capstone Project

Machine Learning Nanodegree

Udacity

# Contents

# Chapter 1

# Introduction

✓ Introduction

Kepler is a space observatory launched by NASA in 2009 to discover Eath-sized exoplanets orbiting other stars [12]. Kepler mission developed several decades to answer the centuries-old question: How frequent are other Earth-like planets in Milkyway galaxy? In particular, what is the frequency of Earth-size planets in the Habitable-Zone of solar-like stars? There are three different types of exoplanets are common in our universe: gas giants, hot-super-Earths in short period orbits, and ice giants. The challenge is to find the terrestrial planets that are in the habitable zone of their host stars where liquid water might exist on the surface of the planet.

The scientific objective of the Kepler mission is to explore the structure and diversity of planetary systems. This mission surveys a large sample of stars to determine the percentage of terrestrial and large planets that are in or near the habitable zone of a wide variety of stars and determine the distribution of size and shapes of the orbits of these planets. Kepler mission also estimates how many planets are in multiple-star systems. After collecting a large number of data points, using many techniques scientists determine the properties of those stars that harbor

planarity systems including the planets itself.

After preprocessing raw data, the goal is to classify each detection into one of the three different categories: Planetary Candidate (PC), Astrophysical False Positive (AFP) and None transiting phenomena (NTP ). Historically this process is done by researchers looking at each observation. This is a very slow and time consuming process. During this project, I am attempting to automate this classification process using machine learning algorithms.

## 1.1 Search for Earth-Like Planets

✓ What Problem we are solving - Project Origin, Background problem domain

✓ - define the problem : explain in detail about the exoplanets

One of the most fundamental and intriguing philosophical questions has remained for at least 2300 years is whether life exists outside of our Solar System. With the science and technology has grown exponentially within the last couple of hundred years, it is now, we have the theoretical knowledge, practical and feasibility to seek answers to this question from a scientific perspective. NASA's Navigator Program is the long term project that over-seeing the missions related to detecting and characterization of Earth-like planets. These missions consist of multidisciplinary suite of research efforts, centering on finding exoplanets that could harbor biological activity similar to terrestrial life.

In the process of searching for Earth-like planets, we will encounter a spectrum of planets. The planetary population conceivably outnumbers stellar population since a star could host many planets such as our solar system. Unlike stars[1], plan-

[1]Vogt-Russell theorem states that the properties of a star are fully determined by the mass and chemical composision.

ets can have many different characteristics, careful examination of the planets in our solar system show us that full understanding of planets, in general, requires a working knowledge of diverse fields including star formarion, orbital mechanics, geology, grophysics, climatology, agronomy, chemistry, biology and various engineering disciplines.

we need to address a couple of questions in the search for habitable worlds:

- How do planetary systems form and evolve?
- Are there other planetary systems like our own?
- Is there life elsewhere in the Universe?

Search for habitable planets begins with an understanding of the formation of planetary systems in protoplanetary disks to learn how planetary systems form, around which types of stars planets form, how often planets form, and how the disk properties effect the distribution of final planets sizes and orbits[27].The primary mission for observing the formation of planetary systems will be the James Webb Space Telescope (JWST) [8]. JWST will allow us to study the earliest moments of star and planet formation.

In the quest of finding an extraterrestrial life form, the current observations suggest that Earth-sized rocky planets may be common; however, their abundance is quite uncertain. Kepler is a one the space-based mission[12] that is under NASA's navigator program which observes planetary systems in our solar neighborhood. Kepler also finds correlations between the presence of Eath-like planets and both stellar characteristics and the presence and orbits of giant planets. It is important to understand that no single instrument or technique is capable of finding all planetary system components around stars of all ages. Many space and ground bases

missions will use complementary instrumentations and techniques to explore majority of planetary discoveries. There are four fundamental techniques will be used to determine the architecture of planetary systems: (i) *Radial Velocity Measurements* (ii) *Transit Observations*, (iii) *Astrometry* (iv) *Direct Detection*.

To answer the question "Is there life elsewhere in the universe?", we need to look for habitable planets. For us to determine whether a planet is habitable, we must build observations capable of directly detecting the light from the planet, with the planet illuminated by the light from its parent star. Direct detection of these planets is an enormous technical challenge. Another way to infer habitability of a planet is to check if the planet is located in the *Habitable Zoned* of the host star. Habitable Zone is the range in the distance from a star where liquid water could exist on the surface of a planet orbiting a star that possibly supports life. Liquid water is essential to all life on Earth, and so the definition of a habitable zone is based on the hypothesis that extraterrestrial life would share this requirement [14]. This is a very traditional definition, as a planet surface temperature may depend on other factors such as greenhouse gas abundance, its reflectivity, atmospheric and oceanic circulation, radioactive decay, and tidal heating within the planet. These energy sources can be easily allowed the planet to have subsurface liquid water reservoirs. Jupiter's moon Europa has liquid water ocean tens of kilometers below its surface that may well be habitable for some organisms. More than 20 planets, including the nearest extrasolar planet, Proxima Centauri b [1], have been found that are both roughly Earth-sized and orbiting within a Habitable Zoned of their stars.

NASA's navigator program is a scientific program whose primary goal are to detect and characterize Earth-like exoplanets and understand the formation, and

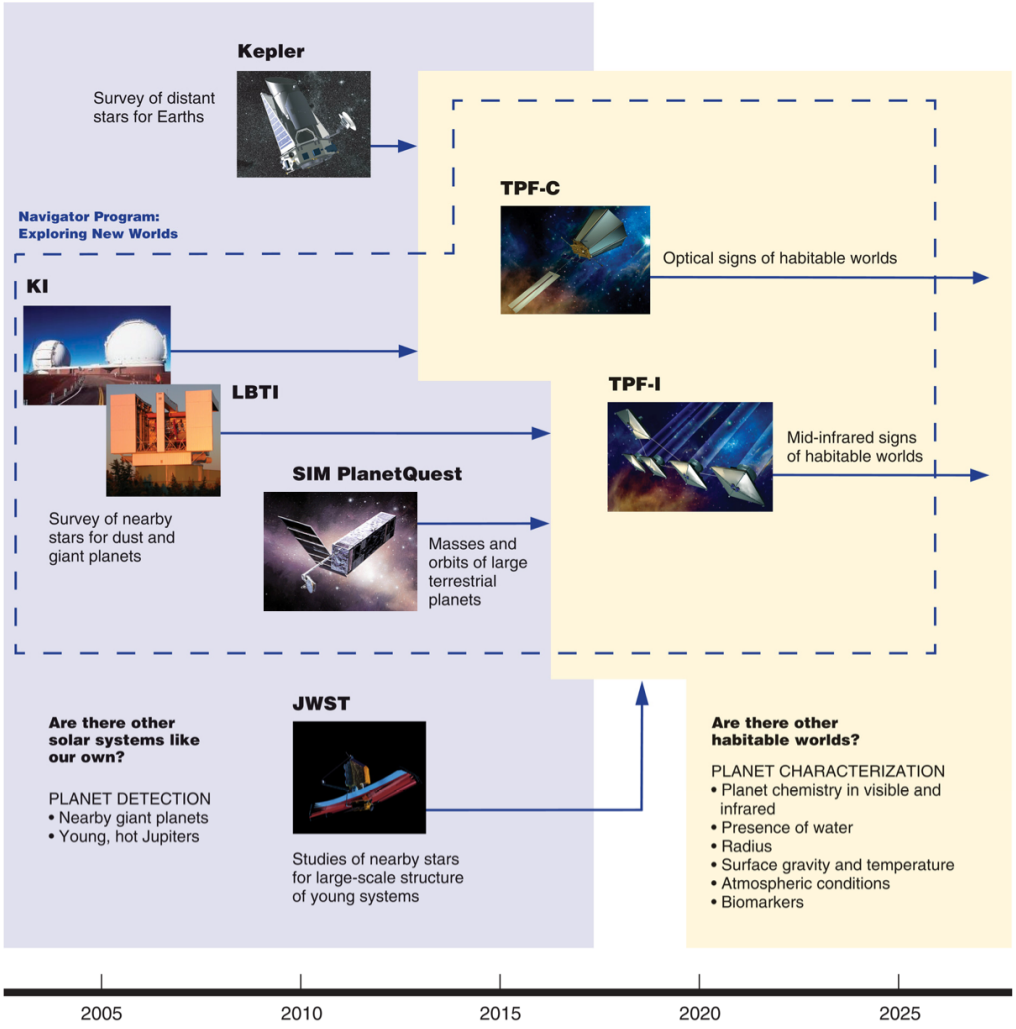



Figure 1.1: Navigator Program. National timeline including the flow of science between missions, including the *Kepler* Discovery mission and JWST. *Image Credit*: NASA

distribution of planetary systems in our Galaxy. Key features of the navigator program includes, integration of space and ground activities into a cohesive effort to find and characterize the planetary systems in our solar neighborhood. Multi-project

approach to managing risk across the Navigator program by identifying the scientific and technological dependecies across the program and developing alternatives and descope to provide robustness and flexibility. The relationship between the different missions is illustrated in figure 1.1

## 1.2 Kepler Mission

Kepler mission and the spacecraft was named after one of history's revolutionary German astronomer and mathematician, Johannes Kepler[26]. Kepler space observatory was launched in March of 2009 into an Earth-trailing heliocentric orbit from Cape Canaveral Air Forced Station, Florida using Delta II rocket. Spacecraft design to observe fixed field of view for an extended period. Original mission duration was planned for 3.5 years; however, the mission elapsed 7 years and 11 months. On June 19, 2009, the spacecraft sent its first science data to Earth. Spacecraft continually collect science data downlink back to Earth per month. Each dataset roughly 12 gigabytes in size. On July 14, 2012, one reaction wheel (out of four) use for pointing of the spacecraft failed; however, the spacecraft only require three wheels to operate accurately. Spacecraft continually collected data from the original field of view until a second reaction wheel failed on May 11, 2013. This ends the Kepler's primary mission. At this point spacecraft no longer point to its original field of view, and this led to the "K2" follow-on mission [11] observing different fields near the elliptic orbit.

## 1.3 Photometry

The most basic information we can measure about celestial objects is the amount of energy that coming from it in the form of electromagnetic radiation. This quantity is known as flux. The science of measuring the flux from a celestial object is called Photometry. The photometer is an instrument that use to measure light intensity coming from an object. A Charge Coupled Devices (CCD) camera is essentially a grid of photometers that record and measure photons are coming from the sources that are in its field of view. The primary instrument of the Kepler spacecraft is a CCD photometer[19]. This CCD has 0.95-meter aperture and a 105 square degree field of view (FOV). This instrument has the sensitivity to detect Earth-like planet transit that host by a solar-like star in 6.5 hours of integration.

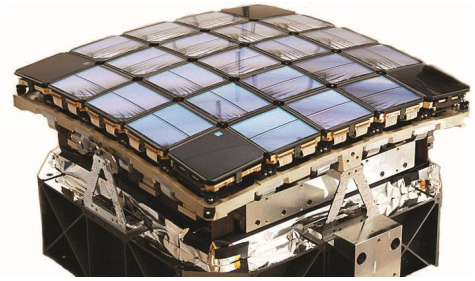


Figure 1.2: The focal plane consists of an array of 42 CCDs. Each CCD is 2.8 by 3.0 cm with 1024 by 1100 pixels. The entire focal plane contains 95 mega pixels. mage Credits: NASA Ames and Ball Aerospace

## 1.4 The Transit Method of Detecting Extrasolar Planets

The Kepler spacecraft detects exoplanets using transit photometry [4]. If a planet orbiting a star on the plain of view and when the it moves between the

detector and the star, the light that is coming from the star partially get blocked. This event is called a "*transit*". For example, we can observe an occasional Venus or Mercury transit from Earth as a small black dot creeping across the Sun. During a transit, the flux we receive from the star reduce due to the transiting planet. When this happen, we say the planet transit the star, and can be detected using transit photometry. During these events, Kepler CCD collects raw data form of a sequence of stellar images, which are processed into "light-curves" tracking the brightness of a star over time. Light-curves are graphs that show the intensity of the light that observes from the star on the y-axis and the observation time on the x-axis (Figure 1.3). These transit data are rich with information about the planet-stellar system. The depth of the dip in the light curve and the size of the star together can be use to measure the size or the radius of the planet. The orbital period of the planet can be determined by measuring the time different between transits. Once the orbital period of the planets is known, Kepler's Third Low of planetary motion can be use to determine the average distance of the planet from its hosting star.

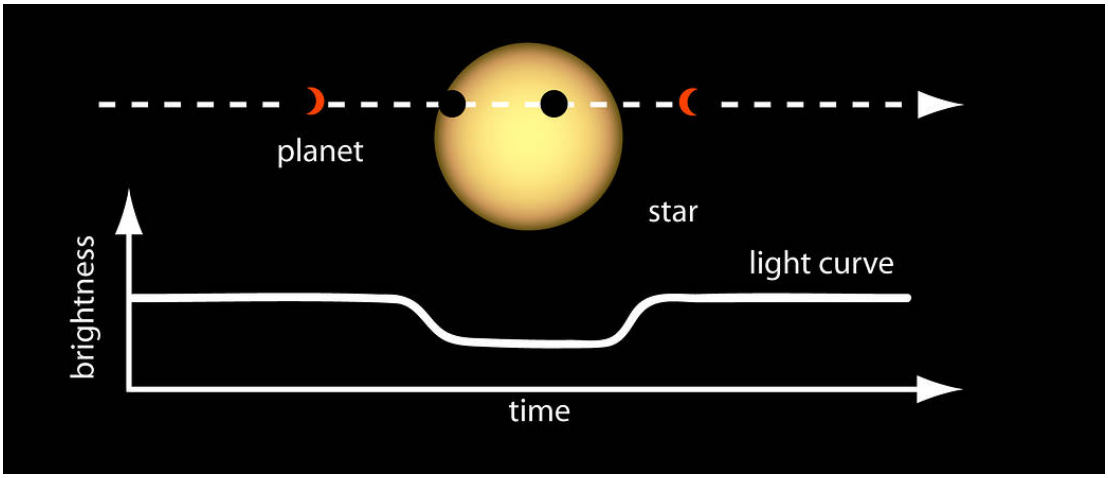



Figure 1.3: Light Curve of a Planet Transiting Its Star. Image Credit: NASA

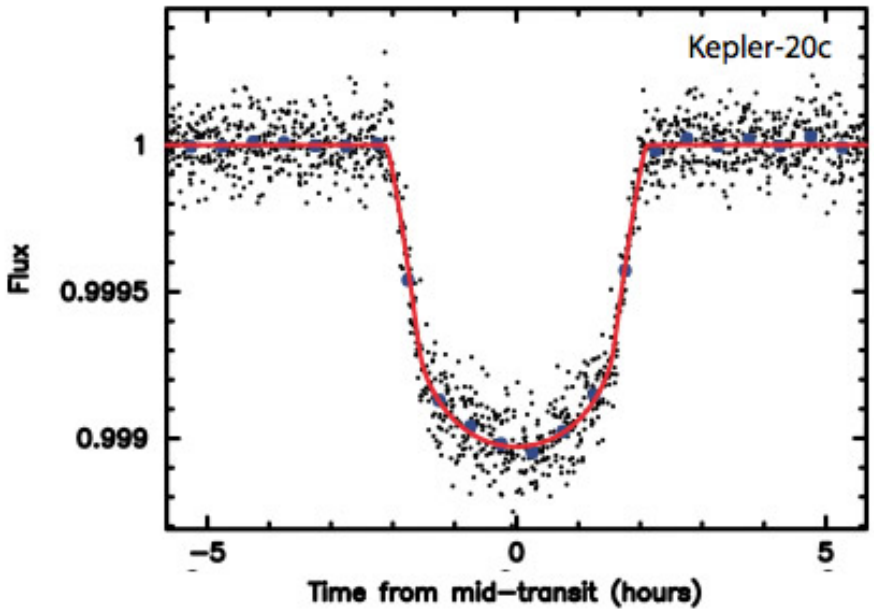



Figure 1.4: The light curve shown here was made from brightness data gathered by the Kepler Mission for discovery of planet Kepler-20c. Image Credit: NASA

Figure 1.4 shows the light curve made from the brightness data from Kepler Mission for the discovery of a planet named *Kepler-20c*. Data clearly show the flux of the host star drop notability when the planet is in transit. This is the primary signature we are looking in the transit photometry to discover planets that are in orbits around stars. These planets create this photometric signature periodically while they are orbiting around the star. To make things more complicated, the light curves of some other objects also create similar photometric signatures: eclipsing binaries stars, certain other variable stars. During the classification stage, we need to classify these object as false positives.

## 1.5 Kepler Field of View (FOV)

When it comes to where to look for exoplanets, there is a vast area in the sky we could point the spacecraft; however, there are a couple of technical requirements need to satisfy for the spacecraft to collect accurate data. The primary requirement is the field of view is always clear of the Sun and the Moon. This is preventing the Sunlight enter int to the CCD array. Kepler points away from the ecliptic, the line in the sky where the Sun, Moon, and the solar system planets traverse. Additionally, Kepler chose to look at an arm of the Milky Way galaxy that has stars similar in age and composition to our Sun and have the largest possible number of stars. The field of view region the Kepler mission is in the constellations Cygnus and Lyra, north of the visible band of the Milky Way. Figure 1.5 show the Field of View superimposed over Milky Way Galaxy, Figure 1.6 show the FOV relative to our Sun, Figure 1.7 show the FOV relative to Cygnus.

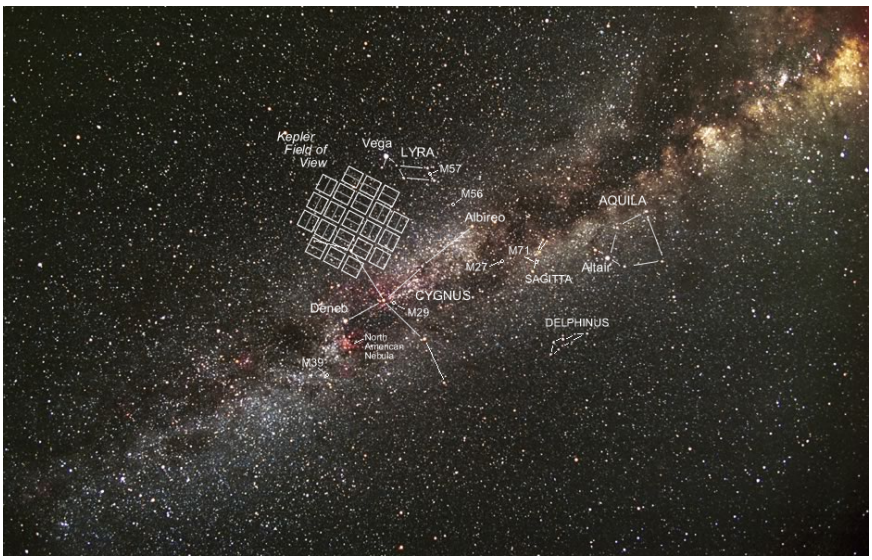



Figure 1.5: The Kepler field of view superimposed over a photograph of the Milky Way Galaxy

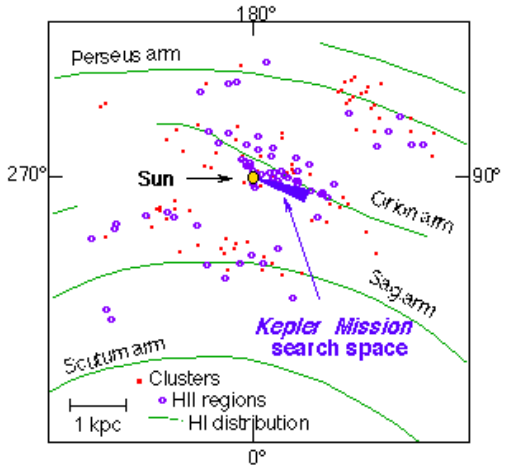



Figure 1.6: Kepler FOV in extended solar neighborhood

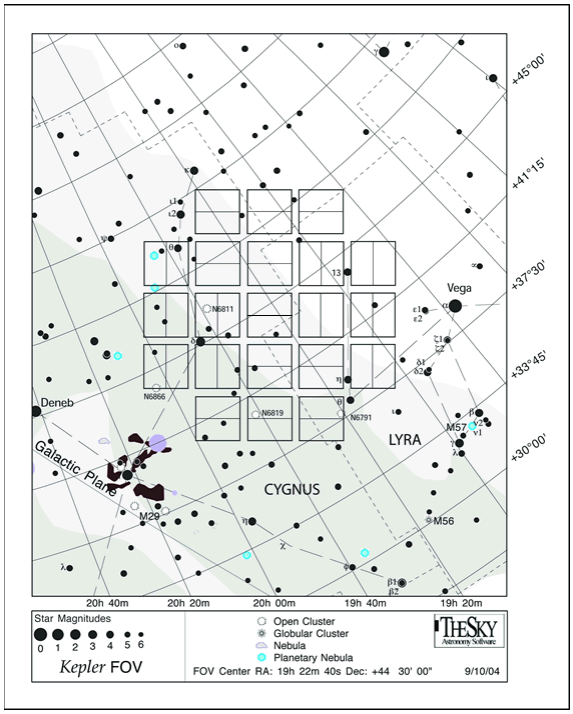



Figure 1.7: Squares are 21 CCD modules.

## 1.6 Kepler Pipeline

The Kepler Pipeline [12] is a data reduction pipeline used for translating the Kepler raw pixel data into possible transiting planet detections. Kepler mission perform photometric observations of carefully selected stars (around 156,000) using its 115 deg$^2$ field of view (FOV) as reviewed in Borucki et el. (2010) [3] and Koch et al. (2010) [13]. The Kepler Mission Science Operations Center (SOC) at NASA Ames Research Center performs major functions on these datasets including calibrate CCD array, download data (light curves) from the spacecraft periodically, remove systematic noise [23] and perform statistical tests to reject false positives and establish accurate statistical confidence in each detection.

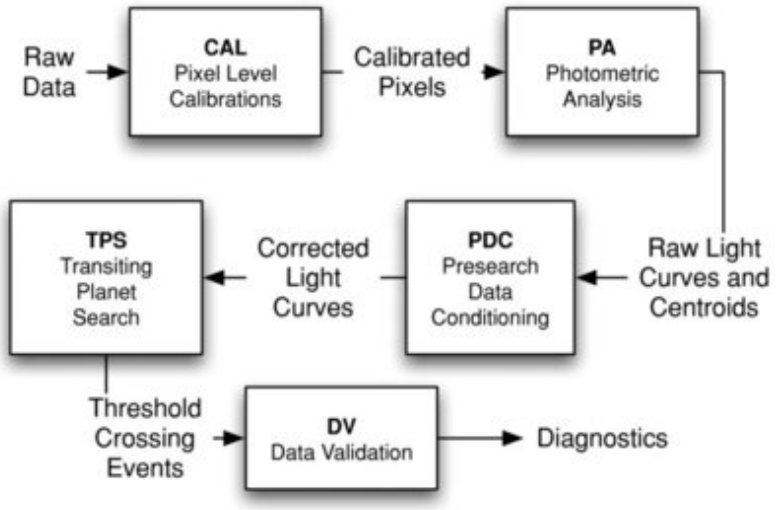



Figure 1.8: Data flow diagram for the SOC Science Pipeline. Image Credit: The American Astronomical Society

Figure 1.8 show the major steps and modules of the pipeline. In particular the last two modules of the pipeline; those that identify as Threshold Crossing Events (TCEs) and their subsequent transit model fiting. TCE is a sequence of significant, periodic, planet transit-like features in the light curve of a target star. Transiting Planet Search module takes systematic error-correlated light curve for a star and seach parameter space for possible transit signatures. This module outputs a TCE or say that does not exists TCE event on the target star. This produce smaller subset of target stars what is given to the Data Validation (DV) module. DV module takes initial TCE and gaps the transit signatures from the light curve and uses the Transiting Planet Search to find additional TCEs on the same target star. This process repeats until it finds all the TCEs on given star. More details on this process explained by Mandel and Agol (2002) [16] and Claret and Bloemen (2011) [5].

TPS algorithm detects transit-like features in light curves by applying noise compensating, wavelet-based matched filtering. TPS characterized the power spectral density (PSD) of the observation noise as a function of time to implement a whitening filter in the wavelet domain. The trial transit pulse is whitened and correlated against the whitened flux time series. Features with correlations above the threshold of $7.1\sigma$ are flagged as potential threshold crossing events and subjected to additional tests in TPS to guard against false alarms.

Algorithm searches a parameter space with varying transit durations $D$ and produce a Single Event Statistics (SES) time series that is the significance of the detection of the reference transmit pulse centered at that particular time for each $D$

$$SES(t) = N(t)/\sqrt{D(t)} \tag{1.1}$$

$\sqrt{D(t)}$, is the expected signal to noise ratio of a signal that exactly matches the template pulse and $N(t)$ is the correlated time series.

Multiple Event statistics (MES) is constructed that characterizes a significant detection in a search over varying orbital period $p$ and epochs (phase) $t_0$ by folding $N(n)$ and $D(n)$. MES $> 7.1\sigma$ may produce a TCE if it also passes additional statistical tests. SES and MES are the basis of some of the attributes used in the training set.

## 1.7 KOI and TCE Attributes

The Threshold Crossing Events (TCE) catalog contain a sequence of transit-like features in the flux time series of a given target star. These TCE data can download

from NASA Exoplanet Archive databases [2], and also the detail description of the table fields [3] are listed as public data. Kepler Object of Interest (KOI) catalog contains object data including many attributes. The detail attributes are listed on NASA Exoplanet Archive website [4], and dataset can be download from the arcive tables [5]. All these data cab be download as bulk using data tools available in the archive website.

## 1.8 TCE Classification Labels

Each Threshold Crossing Event (TCE) is subject to a vetting process performed by the Kepler TCE Review Team (TCERT). During the triage (Initial) vetting stage, all TCEs are partition into two different sets: Problematic Ligh Curves that has instrumental noise and Kepler Object of Interest. KOI is a TCE that contains convincing transit-like features that do not present obvious evidence that the TCE was generated from non-transiting phenomena such as instrumental noise. These KOIs moves to next level of the vetting process performed by individuals manually inspecting light curves using detection statistics. Any indication they see the signal came from an eclipsing binary star or more complex forms of instrumental noise removed from the list. TCEs that survive this removal process are classified as Planet Candidates (PC).

We can identify three different types of classification labels in the processed dataset: Planetary Candidates (PC), Astrophysical False Positives (AFP) and non-

[2] http://exoplanetarchive.ipac.caltech.edu/cgi-bin/TblView/nph-tblView?app=ExoTbls&config=q1_q17_dr24_tce

[3] http://exoplanetarchive.ipac.caltech.edu/docs/API_tce_columns.html

[4] http://exoplanetarchive.ipac.caltech.edu/docs/API_kepcandidate_columns.html

[5] http://exoplanetarchive.ipac.caltech.edu/cgi-bin/TblView/nph-tblView?app=ExoTbls&config=cumulative

transiting phenomena (NTP). PCs are confirmed as planets, statistically validated as planets or determined to be a planet candidate by the TCERT. AFPs are those TCEs that have been shown to be eclipsing binary stars or have shown evidence that the transiting object being detected is not located around the target tart. NTP are those TCEs that failed the initial vetting process.

## 1.9 The Problem statement

Kepler is a single instrument spacecraft that collect most contiguous and long-running photometric time series possible. Kepler observe approximately 170,000 stars simultaneously while it is operating. The fundamental objective of the Kepler mission is to detect a large number of transiting exoplanets. The ultimate goal of the primary mission was a characterize the frequency of exoplanets on diameter, orbital period and host star. The Manual classification of the findings of Kepler object has proven very time-consuming. The new space-based, transit photometry missions such as K2 [11], TESS [21], and PLATO 2.0 [20] also produce a large number of the dataset that demands some level of automation to do the classification.

Using machine learning classification techniques, we can speed up the process and provide a more continuous rating of planarity candidates. There are various of machine learning classification techniques has been applied to Kepler dataset including random forests, SVM, K-mean clustering [25, 17]. In this project, we are attempting to train a Multilayered Neural Network to identify the potential planetary candidates in the Kepler dataset.

## 1.10 Solution Statement

Machine learning techniques contribute a way to automate some step of exoplanet discovery. The TCE vetting process is a tedious and time-consuming (mostly a manual) process. Modern astronomical observational instruments generate a large amount of data within a short period of observational time. These observations may contain such a crucial events that need future follow-up observations using other telescopes (using other wavelengths); hence, processing these time series data and extracting meaningful information is a time sensitive process. Solution to this is to process Kepler data using machine learning algorithms to express the classification while reducing human errors. I am attempting to trained Neural Network to process the TCE catalog to automate the classification process.

# Chapter 2

# Analysis

A Threshold-Crossing Events (TCE) are built using flux time series that has a sequence of transit-like features. The flux time series of a target star that resembles the signature of transiting planet with a high degree of confidence passed on for further analysis. Each TCE identified by a Kepler ID (KID). In the case of a star that holds multiple planets may have a many TCEs. Figure 2.1 show examples of two classes of TCEs. In the top plot is the flux time series for a true transiting planet candidate, with transit-like features of comparable depth. In the bottom plot is a flux time series containing a single very large feature and a bump that is slight above the noise floor; the Kepler pipeline identifies this flux time series as containing a transit signature despite the fact that it is just the artifact.

## 2.1 Threshold Crossing Event Catalog

Figure 2.2 show a sample of TCE datalog from Exoplanet Archive from NASA Exoplanet Science Institute. This is an interactive table that has exoplanet archive including information provided by the original sources. Each row that has unique KeplerID is a TCE. In Figure 2.2 show only a few columns out of many. Each col-

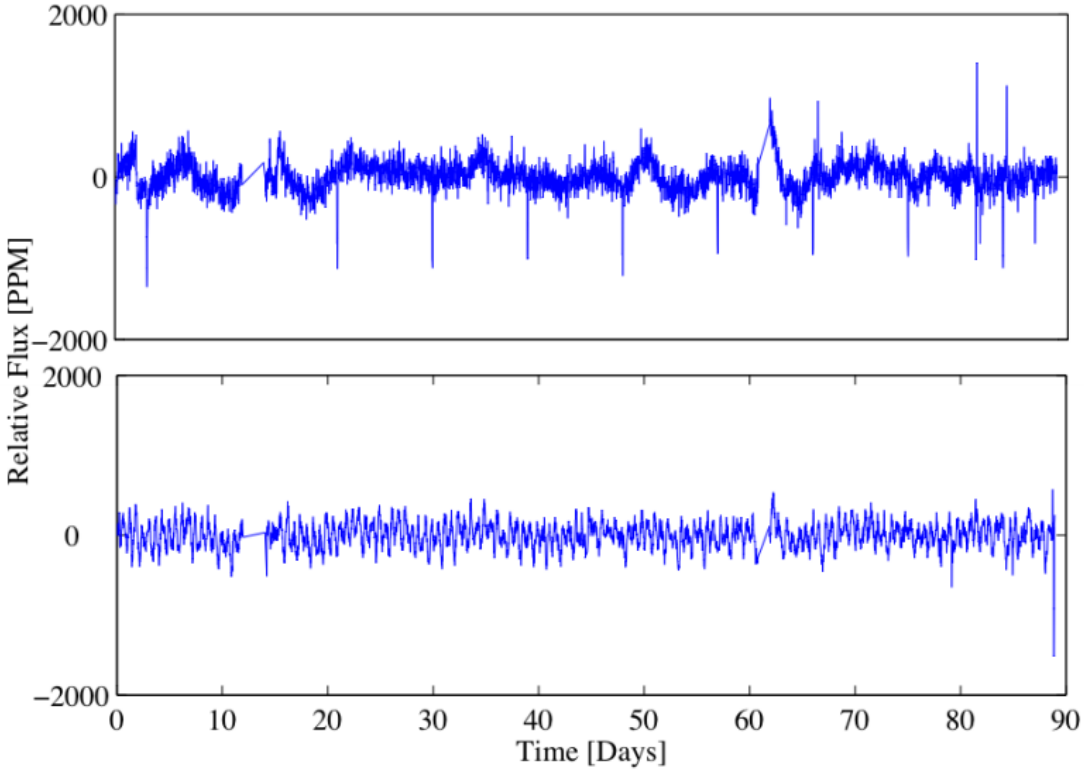
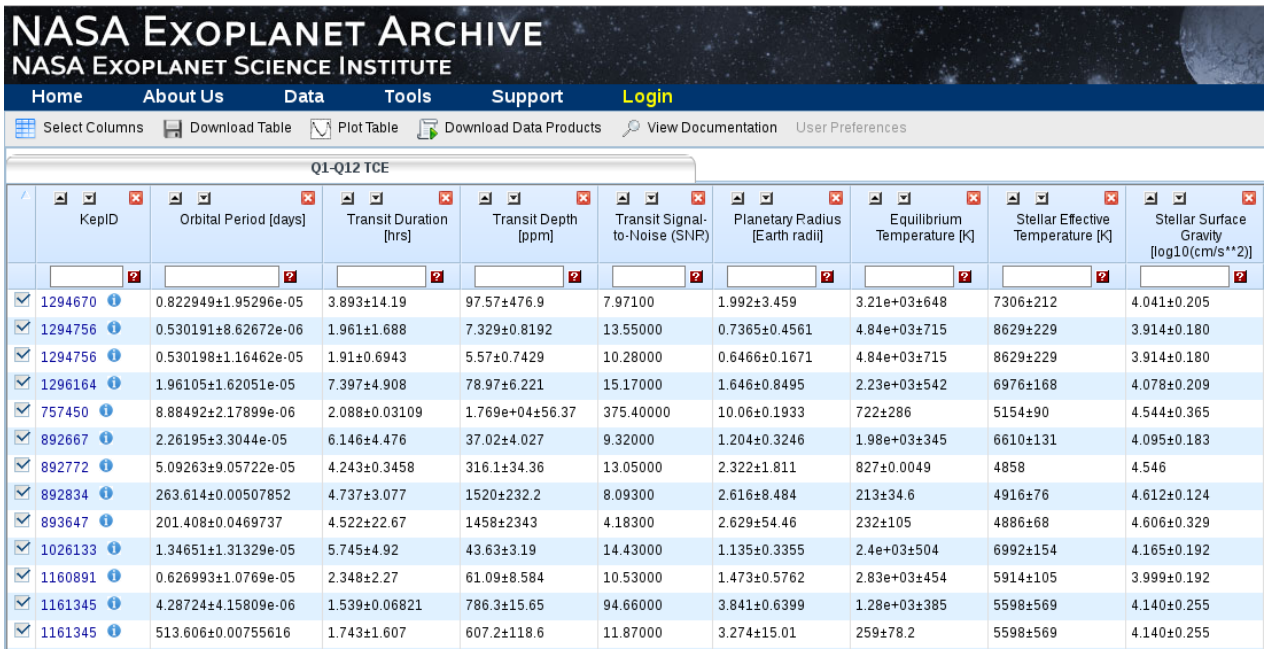



Figure 2.1: Two examples of light curves which produce TCEs. *Top*: a planet candidate, with multiple transit-like features of comparable depth. *Bottom*: a spurious candidate, with a single large feature at the end of the light curve and a small bump at 79 days that are combined and interpreted as a TCE.

umn is an observed or a computed data point that related to the given TCE. Using the upper left hand "Select Column" menu, we can select any fields to appear on the table. By hovering over a KeplerID, you can explore more data that is belong to this particular TCE (show in Figure 2.3). Kepler target overview page show more complete stellar parameters including stellar 2MASS images. Kepler TCE Overview page shows in-depth details of the TCE, this includes Kepler data validation reports and Kepler Planet Detection Metrics and time series data including various downloading options. We can download all the time series data from these




| KepID | Orbital Period [days] | Transit Duration [hrs] | Transit Depth [ppm] | Transit Signal-to-Noise (SNR) | Planetary Radius [Earth radii] | Equilibrium Temperature [K] | Stellar Effective Temperature [K] | Stellar Surface Gravity [log10(cm/s**2)] |
|---|---|---|---|---|---|---|---|---|
| 1294670 | 0.822949±1.95296e-05 | 3.893±14.19 | 97.57±476.9 | 7.97100 | 1.992±3.459 | 3.21e+03±648 | 7306±212 | 4.041±0.205 |
| 1294756 | 0.530191±8.62672e-06 | 1.961±1.688 | 7.329±0.8192 | 13.55000 | 0.7365±0.4561 | 4.84e+03±715 | 8629±229 | 3.914±0.180 |
| 1294756 | 0.530198±1.16462e-05 | 1.91±0.6943 | 5.57±0.7429 | 10.28000 | 0.6466±0.1671 | 4.84e+03±715 | 8629±229 | 3.914±0.180 |
| 1296164 | 1.96105±1.62051e-05 | 7.397±4.908 | 78.97±6.221 | 15.17000 | 1.646±0.8495 | 2.23e+03±542 | 6976±168 | 4.078±0.209 |
| 757450 | 8.88492±2.17899e-06 | 2.088±0.03109 | 1.769e+04±56.37 | 375.40000 | 10.06±0.1933 | 722±286 | 5154±90 | 4.544±0.365 |
| 892667 | 2.26195±3.3044e-05 | 6.146±4.476 | 37.02±4.027 | 9.32000 | 1.204±0.3246 | 1.98e+03±345 | 6610±131 | 4.095±0.183 |
| 892772 | 5.09263±9.05722e-05 | 4.243±0.3458 | 316.1±34.36 | 13.05000 | 2.322±1.811 | 827±0.0049 | 4858 | 4.546 |
| 892834 | 263.614±0.00507852 | 4.737±3.077 | 1520±232.2 | 8.09300 | 2.616±8.484 | 213±34.6 | 4916±76 | 4.612±0.124 |
| 893647 | 201.408±0.0469737 | 4.522±22.67 | 1458±2343 | 4.18300 | 2.629±54.46 | 232±105 | 4886±68 | 4.606±0.329 |
| 1026133 | 1.34651±1.31329e-05 | 5.745±4.92 | 43.63±3.19 | 14.43000 | 1.135±0.3355 | 2.4e+03±504 | 6992±154 | 4.165±0.192 |
| 1160891 | 0.626993±1.0769e-05 | 2.348±2.27 | 61.09±8.584 | 10.53000 | 1.473±0.5762 | 2.83e+03±454 | 5914±105 | 3.999±0.192 |
| 1161345 | 4.28724±4.15809e-06 | 1.539±0.06821 | 786.3±15.65 | 94.66000 | 3.841±0.6399 | 1.28e+03±385 | 5598±569 | 4.140±0.255 |
| 1161345 | 513.606±0.00755616 | 1.743±1.607 | 607.2±118.6 | 11.87000 | 3.274±15.01 | 259±78.2 | 5598±569 | 4.140±0.255 |

Figure 2.2: Threshold Crossing Event Catalog: `http://exoplanetarchive.ipac.caltech.edu/cgi-bin/TblView/nph-tblView?app=ExoTbls&config=q1_q12_tce`

catalogs.

## 2.2 TCE Attributes

Initial TCE attribute set contains 237 attributes that are based on the wavelet matched filter use by TPS, transit model fitting, difference image centroids, and some additional tests. Each of these attributes has different strength of prediction values. The importance of these attributes is selected using historical literature on attributes and performing a Principal Component Analysis. We discuss some of the most important attributes in this section.

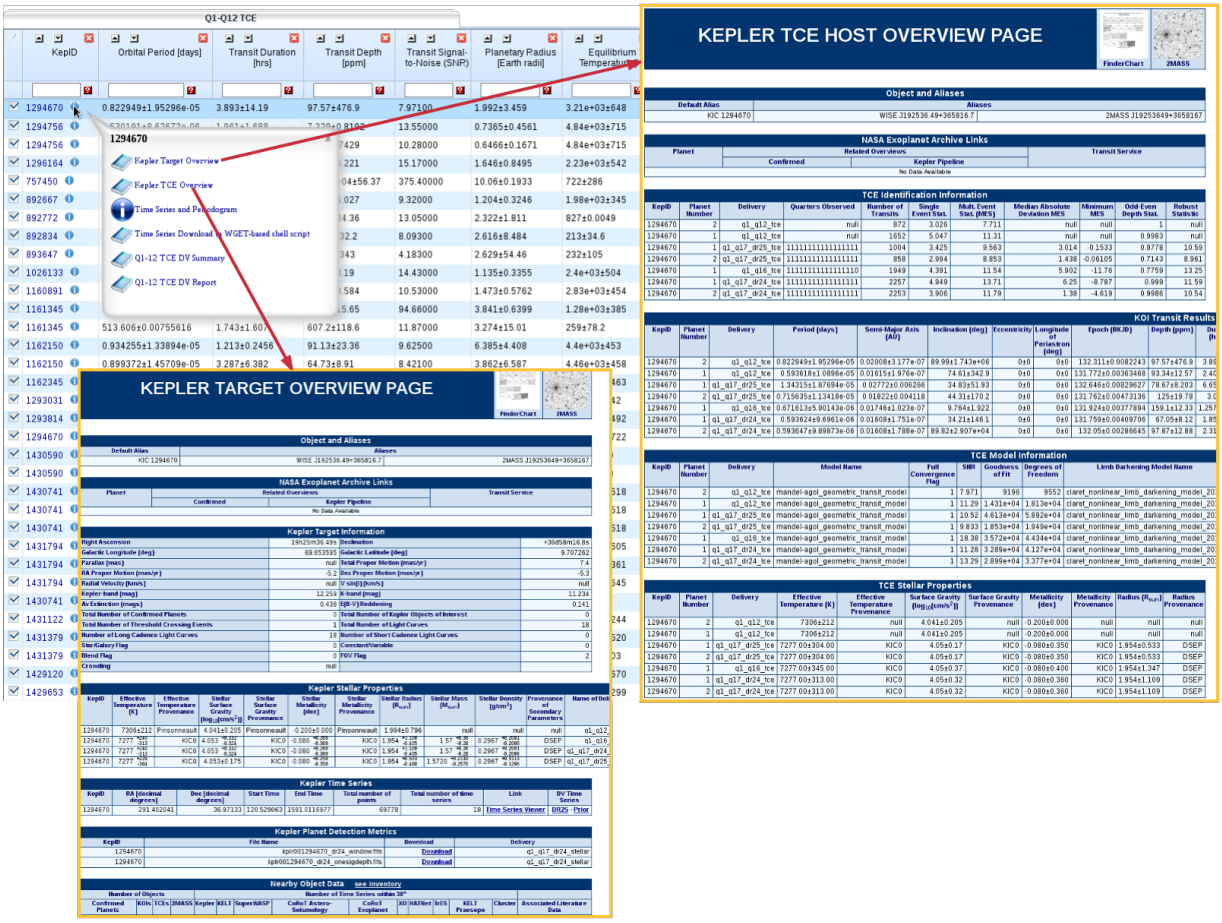



Figure 2.3: Threshold Crossing Event Catalog Details

### 2.2.1 Transit Fit Parameters

This paper presents exact analytic formulate for the eclipse of a star described by quadratic or nonlinear limb darkening. The Kepler Project derives transit parameters from best-fit parameters produced by this analytic formula. Some of the transit parameters are calculated directly; others are derived from the best-fit parameters. Limb-darkening coefficients are fixed and pre-calculated from host stellar

properties.

### Ratio between planet radius and stellar radius (*tce_ror*)

This attribute is calculated using planet radius divided by its hosting the stellar radius. Both planet and stellar radius are in Eath-radii units. Plot 2.4 show the histogram of the *tce_ror* attribute in the TCE catalog (x-axis is in log scale and y-axis is in linear)

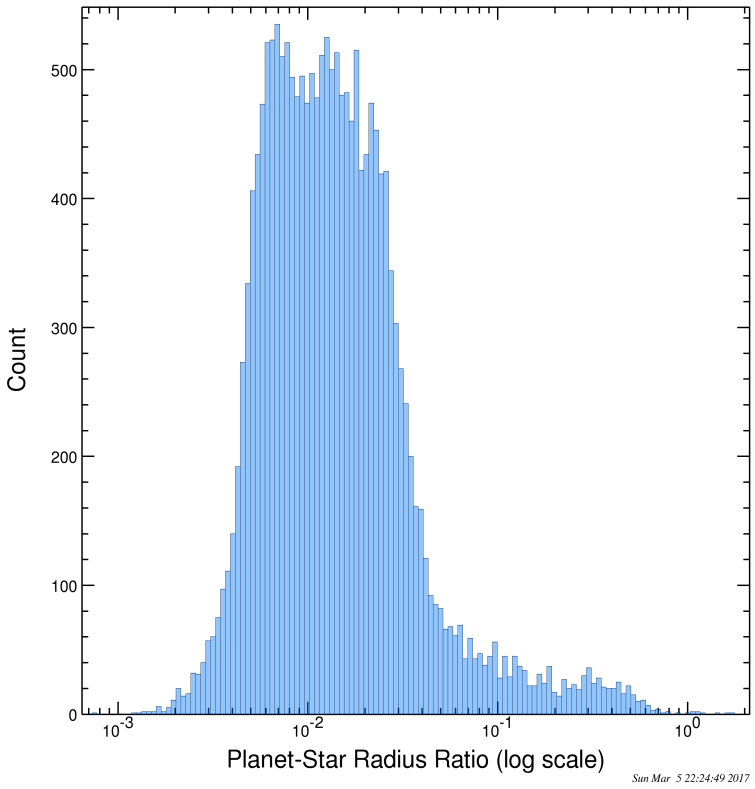



Figure 2.4: Ratio between planet radius and stellar radios

### Transit Duration (*tce_duration)*

Transit Duration is the duration of the observed transits. Duration is measured from the first contact between the planet and star until the last contact. The duration is measured by hours. The plot 2.5 show the distribution of the transit duration of the TCE catalog. We can observe the most of the transit are fall under 10-hour duration.

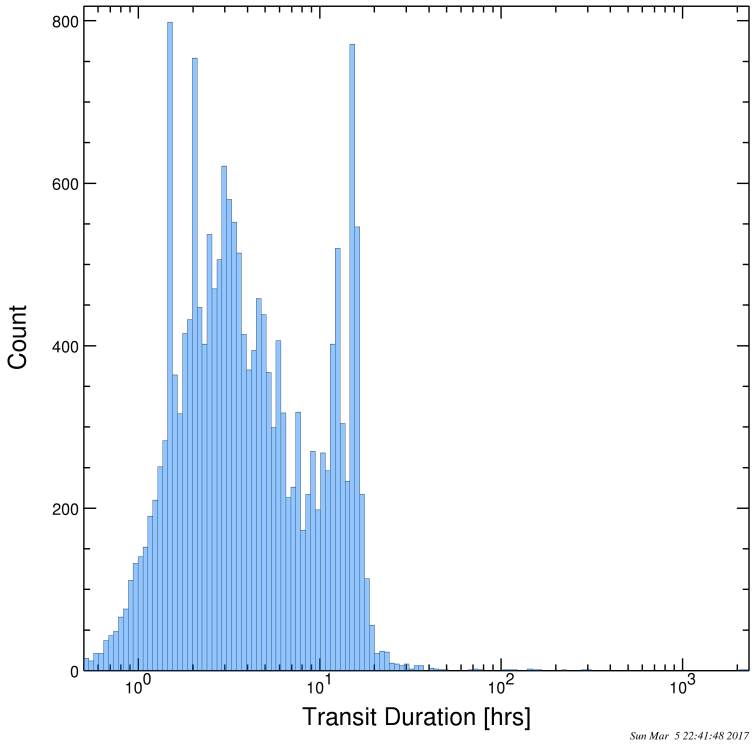



Figure 2.5: Transit Duration

### Impact Parameter(*tce_impact*)

Impact Parameter is the sky-projected distance between the center of the stellar disc and the center of the planet disc at conjunction, normalized by the stellar radius.

### Transit SNR (*tce_model_snr*)

Transit depth normalized by the mean uncertainty in the flux during the transits. Plot 2.6 show the distribution of the SNR.

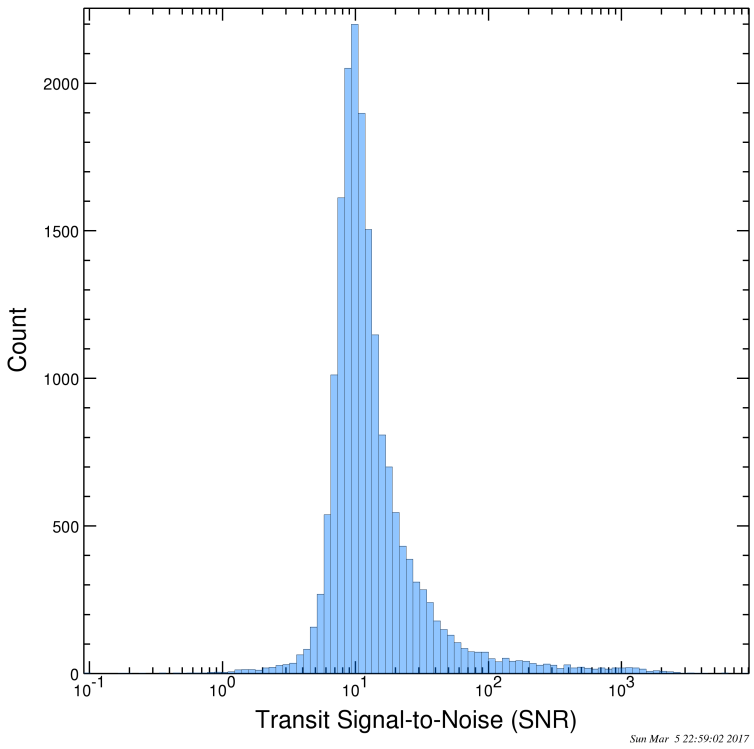



Figure 2.6: Transit SNR

### 2.2.2 Stellar Parameters

Best-fit transit parameters are normalized to the size of the host star. The size of the hosting star is determined by its radius. Physical planet parameters may be derived by scaling to the star's size and temperature. Stellar effective temperature, surface gravity, metallicity, radius, mass, and age should comprise a consistent set. This section describe and visualize some of the stellar parameters from the TCE catalog.

**Stellar Effective Temperature (K) (*tce_steff*)**

Plot 2.7 show the photospheric temperature of the stars in the TCE catalog. Kepler mission target sun-like stars and the photospheric temperature of the sun is 5,775 K. If we plot the sun on the 2.7, it will fall close to the apparently large number count that shows on the plot. This plot 2.7 show the TCE catalog stellar temperatures are falling between 3000K and 17000K

**Stellar Surface Gravity (*tce_slogg*)**

The base-10 logarithm of the acceleration due to gravity at the surface of the star. The surface gravity of our sun is 274.0 cm/$s^2$ (4.44 log10(cm/$s^2$)). Plot 2.8 show the distribution of the TCE catalog stellar surface gravity.

### 2.2.3 Light Curve Based Vetting Statistics

Kepler data pipeline performs detection tests using gap-filled flux time series. These time series usually build using multi-quarter data that observed by the space-craft. In the process of combining these datasets, it removes edge effects around the data gaps and then combines the segments together. Gaps are filled by interpolating

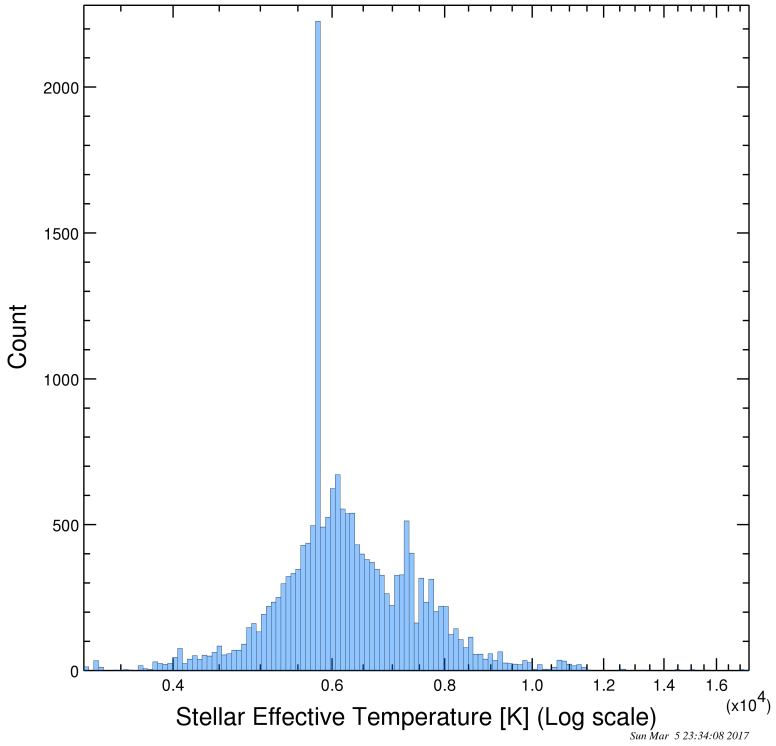



Figure 2.7: Stellar Effective Temperature (K)

data segments. TPS module of the pipeline estimates the Power Spectral Density of the flux tine series as a function of time. This generates coefficients for Single Event Statistics (SES) time series components. These can be interpreted as measurements fo the statistical significance of the presence of a transit of trial duration at each point in the time series.

**Single Event Statistics (SES)**

The maximum calculated value of the SES. Maximum SES statistics for different TCEs from the same target differ because the most significant TCE is removed from the time series before repeating the test for further and weaker transit signals.

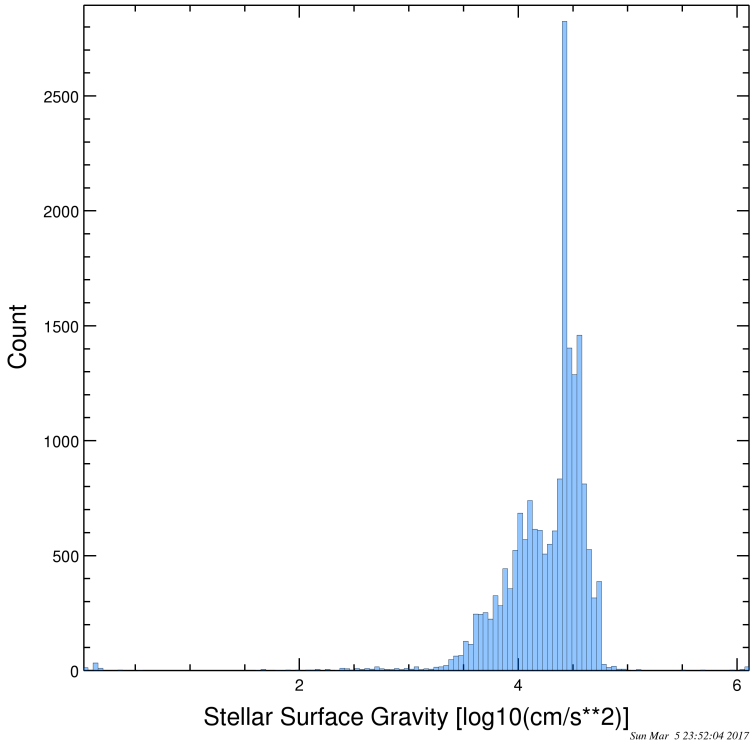



Figure 2.8: Stellar Surface Gravity log10(cm/$s^2$)

### Multiple Event Statistic (MES)

Multiple Event Statistic (MES) is the ratio of the detected signature's strength to the noise limit for a transit signature of the selected period and duration, or equivalently the SNR for the detection of the series of transits. This field contains the maximum calculated value of the MES.

## 2.3 Missing Attributes

In some circumstances, there are missing attributes in the dataset due to missing information in the stellar catalogs; the data validation fit fails to converge, or a

processing timeout is reached. These missing attributes can be fill using reasonable methods such as mean, median, and in case of missing stellar attributes can be filled using sun's parameters. These various substitutions will be tested when training the network.

## 2.4 Principal Component Analysis

TCE attributes set contains well over 200 attributes. I use the Principal Component Analysis to reduce the dimention of this dataset. Principal Component Analysis (PCA) is a standard method use in modern data analysis. PCA provides a roadmap for how to reduce complex data set to a lower dimension to reveal the something hidden, and simplified structures that often underline it. PCA reduces the dimensionality of the data set while preserving most of the variation in data. This talk is accomplished by identifying directions, called Principal Components, along which the variation in the data is maximal. Using this method, a dataset that has a large number of variables can be express using fewer variables (components).

### 2.4.1 Naive Basis

The goal of PCA is to identify the most meaningful basis to re-express the data set while filtering out the noise and expose hidden structures. Lets assume we have a dataset with $m$ number of variables. We can represent these data set using a $m$ orthogonal basises. without loss of generality, we can visualize a data set with a 3 variables using $x, y, z$ axises. $x, y, z$ axises are being the vatiables (Figure 2.9). In other words, the basis of this dataset is (1, 0, 0), (0, 1, 0), (0, 0, 1). We can raise the question why we need to pick this basis over any other basis? The reason is that the naive basis reflects the method we gathered data. We can express this naive

basis using matrix as follows

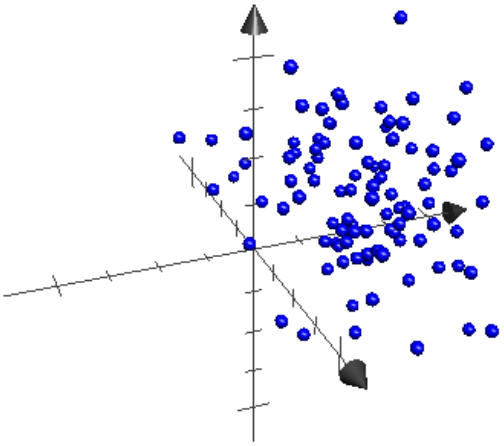


Figure 2.9: Ax

$$B = \begin{bmatrix} b_1 \\ b_2 \\ . \\ b_m \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix} = I \tag{2.1}$$

Each row of this matrix is an orthonormal basis vector $b_i$ with $m$ components. All of our data has been recorded on this basis, and it can be expressed as a linear combination of $b_i$. Generalize form of the matrix can be written as a $m \times m$ identity matrix (Equation 2.1).

### 2.4.2 Change of Basis

All the data has been recorded using naive basis; however, using linear algebra, we can find many other bases where these data can be re-express. For example, any

rotation of naive basis will be perfectly valid basis to re-express the data. This is the question PCA precisely asking: Is there another basis, which is a linear combination of the original basis, which best re-express the data set?

Let us assume $X$ is our original data set and $Y$ is a new representation of the same data set. We can perform a linear transformation ($P$) to produce $Y$ from $X$,

$$PX = Y \tag{2.2}$$

In matrix form, $P$ is a matrix that transforms $X$ into $Y$. Geometrically speaking $P$ rotate and stretch $X$ to produce $Y$. The rows of $P$ are a set of new basis vectors for expressing the columns of $X$.

$$PX = \begin{bmatrix} p_1 \\ \vdots \\ p_m \end{bmatrix} \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix} \tag{2.3}$$

$$Y = \begin{bmatrix} p_1.x_1 & \dots & p_1.x_n \\ \vdots & \ddots & \vdots \\ p_m.x_1 & \dots & p_m.x_n \end{bmatrix} \tag{2.4}$$

Each column of $Y$ is a dot product of $x_i$ with the corresponding in $P$. Therefore, the rows of $P$ are a new set of basis vectors for representing of columns of $X$. It is important to mention that we are assuming the linearity and this reduce the problem to finding the appropriate change of basis. During this process, the row vector in the matrix P will become principal components. Now the question is what the criteria to pick the best possible basis P? The answer to this question is base on what features we would like Y to exhibit.

Regardless the method of data analysis, the noise in the data must me low. If the noise in the data is significant, then the signal can not accurately extract. The signal to noise ratio ($SNR$, Equation 2.5) or the proportion of variance sigma is high ($SNR >> 1$) for high precision data and low for noisy data. best possible rotation for the naive basis is to find the basis that maximizes the $SNR$. This by assuming the dynamics of interest exists along directions with the largest variance. Therefore maximizing the $SNR$ allow us to find the best possible principal components. Finally, we can assume the Principal Components are orthogonal; this assumption provides an intuitive simplification that makes PCA solvable by using linear algebra decomposition techniques.

$$SNR = \frac{\sigma^2_{signal}}{\sigma^2_{noise}} \tag{2.5}$$

### 2.4.3 Solving PCA using Eigenvalue Decomposistion

One of the ways to find the PCA is using eigenvalue decomposition techniques. The first step is to find some orthonormal matrix $P$ (Eq 2.2) by satisfying

$$C_Y = \frac{1}{n} Y Y^T \tag{2.6}$$

$C_Y$is a diagonal matrix. The rows of $P$ are the principal components of $X$. Using simple substitutions using equation 2.2 and 2.6, we can derive $C_Y$ as,

$$C_Y = P(\frac{1}{n} X X^T) P^T \tag{2.7}$$

$\frac{1}{n} X X^T = C_X$ is the covariance matrix of X. Covariance matrix is a square matrix that has the variance of particular measurement types in diagonal terms, and off-diagonal terms are wth covariance between measurement types. $C_X$capture the

covariance between all possible pairs of measurements. Covariance values reflect the noise and redundancy in the measurements. In the diagonal terms, by assumptions, large values correspond to interesting structure, and in the off-diagonal terms, large values correspond to high redundancy.

During the AVD, choice of $P$ diagonalizes $C_Y$. This was the goal of PCA. The results of the PCA can summarize as,

- The principal components of $X$ are the eigenvectors of $C_X$
- The ith components of $C_Y$ is the variance of $X$ along $p_i$

In practice computing PCA of a data set X is:

- Subtract off the measurement types
- Computing the eigenvectors $C_X$

## 2.5 Application of Artificial Neural Networks in Astrophysics

In recent years, artificial neural networks (ANNs) have emerged as one of the potentially most successful modeling approaches in engineering and science. In this project, I am using backpropagation multilayered perceptron artificial neural network classifier to classify TCE catalog to identify planetary candidates. By using ANN, my objective is to apply machine learning technique to astrophysical datasets.

ANNs are model after human brain and nervous system to simulate complex networks using interconnected modules. ANNs learn by examples; we call this a training data in which an actual measured set of input variables and the corresponding outputs are represented to determine the rules that govern the relationship

between the variables. ANNs are well suited to modeling complex problems where the relationship among variables are not apparent and non-linear [15]. The concept of artificial neurons was first introduced in 1943 [18], and since the back-propagation training algorithm for feed-forward was published in 1986 [22], ANNs has taken a significant advancement within last decade thus be considered as relatively new tool in the field of machine learning.

### 2.5.1 Artificial Neural Networks

ANN consists of a number of artificial neurons (also known as processing elements, nodes, or units). These neurons are usually arranged in layers. There are three different layers we can identify in ANN: an input layer, output layer and more layers between input and output collectively call hidden layers (Figure 2.10).

Each neuron is a layer partially or entirely connected to many other neurons via weighted connections. The scalar weights determine the strength of the connections between interconnected neurons. A zero weight means no connection between neurons and a negative weight refers to a prohibitive relationship. As shown in the Figure 2.11, a neuron receives its weights inputs from other neurons which are summed, and a bias unit or threshold is added to subtracted. Bias us used to scale the input to a useful range to improve the convergence properties of the network (Equation 2.8).

$$I_j = \theta_j + \sum_{i=1}^{n} w_{ji} x_i \tag{2.8}$$

Summed inputs $I_j$ pass through a transfer function to produce the output of the neuron. Figure 2.11 show this process for a neuron $j$ (Equation 2.9).

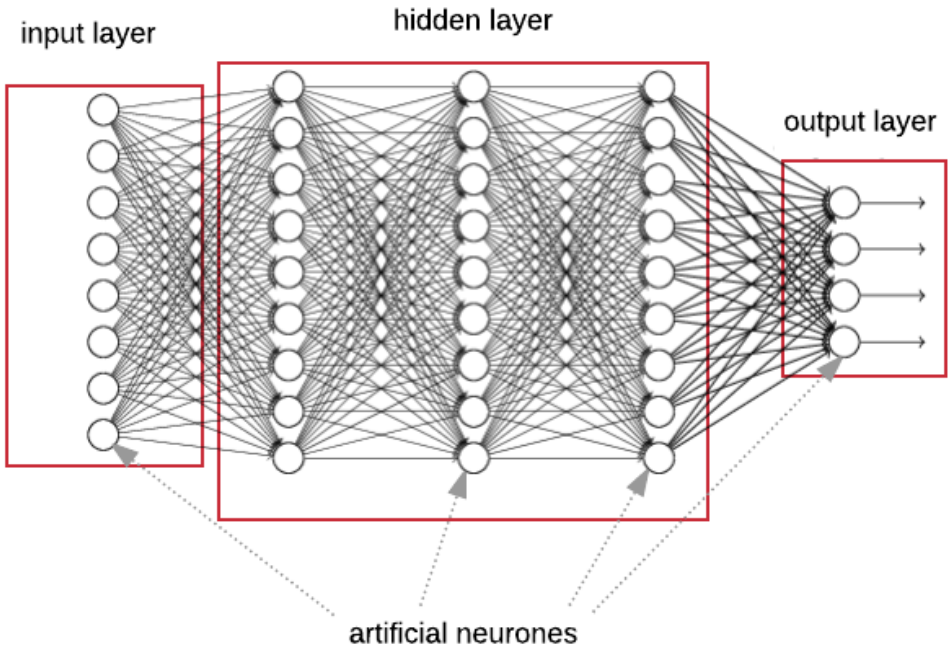



Figure 2.10: Typical Structure of ANNs

$$y_j = f(I_J) \tag{2.9}$$

Using a predefined data set, that has known outputs for given inputs, and the network can be trained. The training steps are follows.

- Information propagation starts from the input layer where the data being fed to the network.
- Inputs are weighted and received by each node in the next layer.
- Weighted inputs are summed and pass through a transfer function to produce

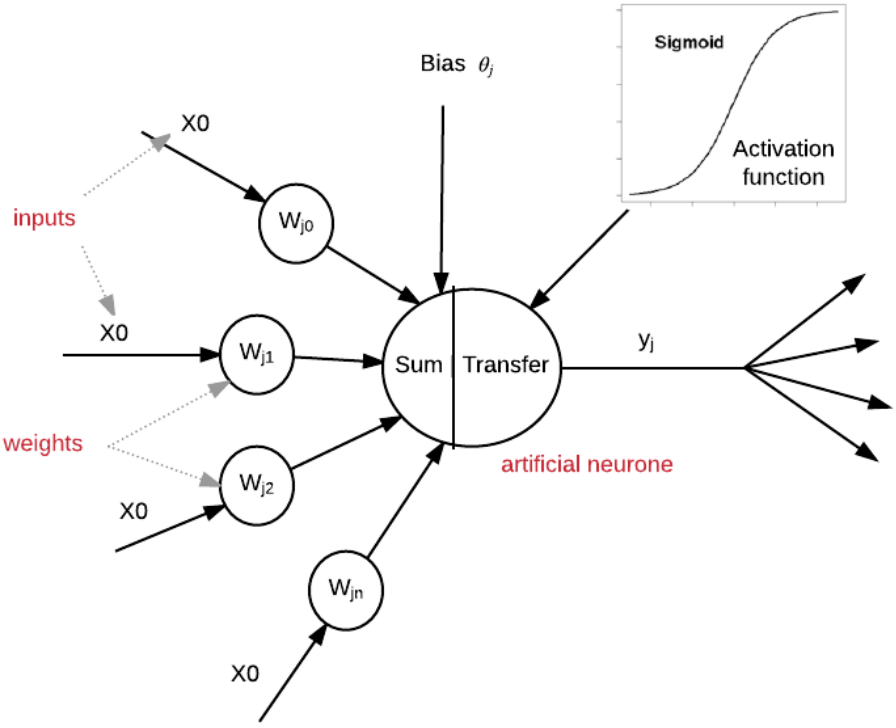



Figure 2.11: Artificial Neuron

neuron outputs, and pass to the neuron in the next layer.

- This process will be repeated while adjusting the weights to produce the predefined output of the training dataset. These weights are calculated by reducing the error.

Learning in ANNs is usually divide into two separate groups: supervised and unsupervised [18]. Supervised learning neural network is given with a dataset with knows classifications (outputs), this dataset is known as the training data. During this project, I am using the supervised learning method to train the network. In this case, the network has been presented with TCE dataset with known planetary clas-

sifications. Network compare the known classification with the network output and compute the error. This error is used to adjust the weights among the neurons until the network produces the same results as the knows classifications in the training dataset. Two examples of Supervised learning ANNs are Multi-Layer Perceptrons (MLP) and neurofussy networks. In unsupervised learning, the network is only presented with the inputs without any desired outputs. The system itself adjusts the connection weights and cluster the input data records into classes of similar features. ANNs also can be divided into two separate groups based on connection types among neurons: feedback networks and connections are in both forward and backward directions. During this project, I am using MLPs trained with a back-propagation algorithm. This has a high capacity of data mapping capabilities.

### 2.5.2 Multi-layer Perceptrons

In multi-layer perceptrons (MLPs) are fall under supervised feed-forward category in which the processing elements are arranged in a multilayered structure [10] Figure 2.10. As described in the equation 2.8 and 2.9, the input to a neuron is weighted and biased and run through an activation function before it passes on to the next layer. The output of a one neuron layer provides the input to the other next layer. The global error between the input and the output of the network is calculated using an error function. For example, the error function, for node j, is calculated using the equation 2.10.

$$E = \frac{1}{2} \sum (y_j - d_j)^2 \tag{2.10}$$

where, $E$ is the global error function, $y_j$ is the predicted output by the network and $d_j$ is the desired output (from the training data set).

The objective is to minimize this error $E$ between the predicted and actual outputs. By reducing this error, the network can produce more accurate results. The minimization of this error function achieves with respect to all variables in the neural network such as connection weights, architecture, learning rate and threshold. Among these variables, connections weights are the most influential variable; thus back-propagation training algorithms are minimized the error function with respect to the connection weights only. Back-Propagation uses the gradient descent technique to adjust the weights in which the global error function, $E$ is minimized by modifying the weights using the equation 2.11.

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} \tag{2.11}$$

where $\Delta w_{ji}$ is the weight increment from node $i$ to node $j$ and $\eta$ is the learning rate: the size of the step taken along the error surface is determined.

The weights are then updated by adding the delta weight, xx to the corresponding previous weight as equation 2.12

$$\Delta w_{ji}(n+1) = \Delta w_{ji}(n) + \Delta w_{ji}(n+1) \tag{2.12}$$

where, $w_{ji}(n)$ is the value of the weight from node $i$ to node $j$ at step $n$ before the adjustment and $w_{ji}(n+1)$ is the value of the weight at step $(n+1)$ after adjustment.

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} + \mu \Delta w_{ji} \tag{2.13}$$

First, the weights between the hidden and the output layers are adjusted, and then weights between the hidden layer and the input layers are adjusted. Learning rate is an another parameter we need to adjust in this process. Usually, the learning

rate is adjusted by trial and error method. Smaller learning rate slows down the convergence even though the convergence can be achieved; however, it is subject to the local minima in the error surface that is closest to the random starting position. On the other hand, if the learning rate is large, the weight changes will be also large, causing the error to go up rather than down the thus convergence might never occur. However, larger learning rated might enable the model to jump out of local minima. Trial and error method to figure out the correct learning rate might lead us to oscillations, which can be overcome by adding another variable to the equation known as the momentum term ($\mu$). The process is to add a momentum term to the weight adjustment that is proportional to the amount of the previous weight change, Equation 2.13. Once the adjustments are carried out, it is saved and used to modify all the subsequent weight adjustments. This means that the weight change of the current step should carry some momentum of the weight change from the previous step.

The process of adjusting weights is repeated until the network produce the desired results by minimizing the global error. Desired results are the output that given by the training data set. Once this process successfully calculates the proper weights, network continually can use these weights to make predictions, and at this point, we call this is a trained network that is ready to use in production.

Multi-layered perceptrons with back propagation algorithms are very efficient in many engineering problems; however, there are some limitations of this kind of neural networks. One of the limitation is that MLPs can get trapped in a local minimum in the process of finding the global minimum of the error surface. Exaggerated illustration of this issue is shown in figure 2.12. Gradient descent is working on reducing the error and trying to find the lowest error possible. As it shown in

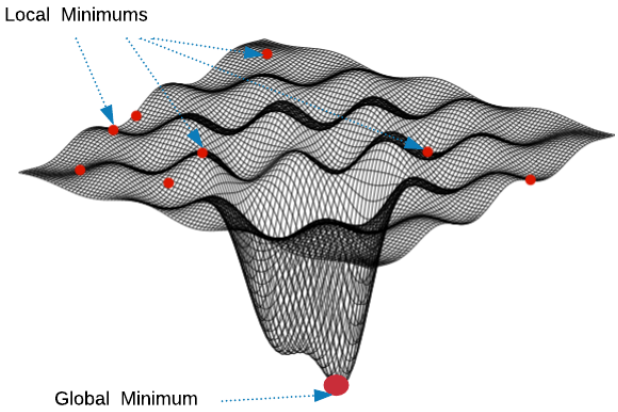



Figure 2.12: Simulated error surface with local and global minimums. (Image credit: `https://projects.g-node.org/emoo/`)

the figure 2.12, there can be many local minimums presents in the error surface where the gradient descent may get trapped in local minimum even though there is a much apparent global minimum present. In the literature, there are several ways proposed to escape from this local minima, including the learning rate, adding the momentum term, adding a small amount of random noise to the input patterns to shake the network from the line of steepest descent, adding more hidden nodes to relocating the network along the error surface by randomising the initial weights and retraining are some of them.Another limitation of MLPs is that feed-forward neural networks that are trained with the back-propagation are often criticized for being black boxes. The knowledge acquired by these networks during training is stored in their connection weights and bias values in a complex manner that is often difficult to interpret.

To improve the performance of the ANN models, we need to determine the model inputs carefully. Careful selection of the model inputs will significantly improve the performance of the network. Mostly we can choose the model input based on the

prior knowledge of the problem we are aiming to solve. Another technique is to train many neural networks with different combinations of input variables and to select the network that has the best performance [24]. There are many other techniques can be found in the literature to the best method to pick model inputs. Selecting a large number of input variables usually increase the network size resulting in a decrease in processing speed and reducing the efficiency of the network.

Once the input parameters are selected, the dataset is divided into two separate sets: training and testing sets. Training set will be used to train and find out the connections weights and other network parameters by minimizing the error between model outputs and the corresponding measured values. Since ANN models have a large number of models parameters, this may cause to overfit the training data, especially if the training date are noisy. In other words, if the number of degrees of freedom of the model is huge compared to training date points, the model might no longer fit the general trend of the data. The testing data set used to make sure the model can generalize within the range of the data used for calibration. Usually, 2/3 of the data are recommended for a training dataset, and 1/3 will use as testing data. If the available data are small, it is difficult to have enough data points in the training dataset hence the k-folding method can be used to separate dataset and train the network.

Once the available data have been divided into training and testing, it is important to pre-process the data in a suitable form before they are applied to the ANN. Data pre-processing usually speed up the learning process. Pre-processing can be done in the form of data scaling (this allow the transfer function to perform well), normalization (in some cases the data need to be normally distributed to obtain optimal results) and transformation (transform the input data into some known

form may be helpful to improve ANN performance).

For free-forward neural networks, the most common method to find the optimum weight is the first order gradient descent. The advantage of this approach is that they have the ability to escape local minima in the error surface and thus produce optimal or near-optimal results. However, this also has a slow convergence rate. The stopping criteria are used to decide when to stop the training process. There are many criteria can be used to determine when to stop training: When the training error reaches a sufficiently small value. Or when not or slight changes in the training error occurred. However, the above example of stopping criteria may lead to the model ending permanently or over-training, and cross-validation approach can be used to overcome these problems. The cross-validation required to divide the data into three sets: training, testing and validating.

### 2.5.3 Model Validation

The performance of the model validation ensures that the model has learned the complex and non-linear relations among variables using training data and is capable of generalized within limits. The standard approach is to achieve this is to test the performance of trained ANNs on the independent validation set, which has been using as part of the model building process. Prediction performance of ANN models are often evaluated using a couple of different methods: The coefficient of correlation (Equation 2.14) , root mean squared error (RMSE, Equation 2.15) and mean absolute error (MAE, Equation 2.16).

$$r = \frac{C_{y_j d_j}}{\sigma y_j \sigma_{d_j}} \tag{2.14}$$

where, $y_j$ - model (predicted) output, desired (observed) output, $C_{y_j d_j}$ covariance

between the model output and desired output. Suggested values for $r$ are,

- $|r| \geq 0.8$ - Strong correlation exists between two sets of variables.
- $0.2 < |r| < 0.8$ - Correlation exists between eh two sets of variables
- $|r| \leq 0.2$ - weak correlation exists between the two sets fo variables

RMSE is the most accepted measure of error and has the advantage that large errors receive much greater attention than small error.

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - d_j)^2} \tag{2.15}$$

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_j - d_j| \tag{2.16}$$

### 2.5.4 Benchmark Model

Benchmarking the neural network by performing many training trials by varying number of training data sets to reach the optimal network. Initial weight will be randomly chosen [9]. Comparing the performance of the NN with other classification algorithms also provide a degree of freedom to benchmark our model. In the literature there are many classification domains can be found that perform classification task on KOI dataset: K-nearest neighbors (K-NN) [6], Naive Bays [7], Random Forests [17].

Figure 2.13 show PC or AFP Detection Rate vs False Alarm Rate of three different classification results performances. K-NN does not produce a ranking of predictions and so is represented as a single point in the plot. The resulting error rates for K-NN and Naive Bays are 3.15% and 2.73% respectively. I will be using

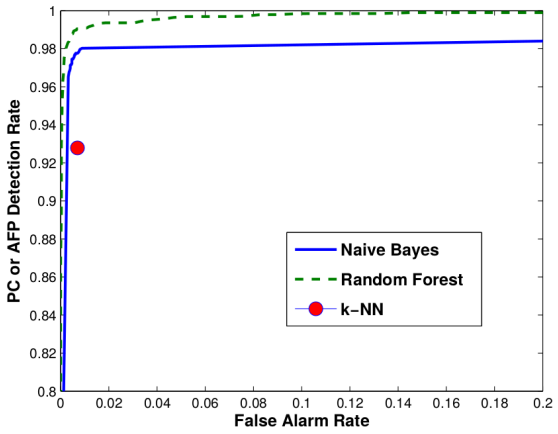



Figure 2.13: Comparison of Random forest, Naive Bayes and K-NN. Image Credit: McCauliff et al. 2015 [17]

these statistics that found in the literature as the benchmark for Neural Network using in this project.

# Chapter 3

# Methodology

Initial Threshold Crossing Events (TCE) catalog contains 237 attributes. Each of these attributes has different strengths when it comes to classifying the ephemerides. For us to develop and train a proper network model to classify Kepler dataset, we need to understand the variables in the catalog carefully. As describe in the previous section, there are many variables in the dataset that needed to be in the final input dataset of the network; however, dependencies between variables and non-significant variables need to remove from the dataset of entry in order to increase the performance of the network. To achieve high performance, we reducing the dataset using Principal Component Analysis and selecting top components to use as features.

## 3.1 TCE Catalog data cleanup

Once download the TCE catalog from the NASAs data archives, I do process the dataset and clean them up. The initial raw data is in comma separated format where the comments are included at the beginning of the file with the descriptions of the columns and their units. I separated these comment section and created a separate data file with each row consists of a column name, its description and given

units. This information later uses as a reference to get the description of the TCE catalog variables. The data values are loaded to a *DataFrame*(using python *pandas* module). There are some handful of columns are not completed in this dataset and left empty (and with zero value), these columns are dropped since these *nul*l and zero values do not play any role.

Once we reduce these null values, we standardize the all the values in the *DataFrame*. Standardization of the data is done using the `StandardScaler` function in the `sklearn.preprocessing` module. Standardization of datasets is a general requirement for many machine learning estimators. Unstandardized data might behave poorly if the individual features do not more or less look like standard normally distributed data: Gaussian with zero mean and unit variance. These standardized data has been saved as a comma separated dataset into a file.

## 3.2 Apply PCA to TCE Catalog

Next step is to perform Principal Component Analysis (PCA) on the standardized TCE dataset. Standardized data set loaded from the file and apply the Principal Components Analysis using sklearn.decomposition PCA method. All the results of the PCA have been saved to a file comma separated files.

### 3.2.1 Data Analysis

I examine eigenvalues of PCA to determine how many principal components should be considered. Table 3.1 show the Eigenvalues, its proportions and the cumulative values of principal components. The proportion of variance calculated by Equation 3.1. The first eigenvalue explains about 19% of the variance. The cumulative percentage calculated by adding the successive proportions of variations

explained to obtain the running total. For instance the 19.00% + 13.60% equate to 32.60% for the second eigenvalue ($\lambda_2$). This means the first two eigenvalues represent 32.60% variances of the dataset. To determine the number of principal components we need to use, we can plot the Cumulative variance vs the principal components as shown in the figure 3.1. The Cumulative variance will add up to 100%, this mean, the all the PCs contain the 100%of the variance of the dataset as it should be. It is apparent from the plot when we reach the $20^{th}$ component we almost capture total variance of the dataset. From the table 3.1 we can see $20^{th}$ PC has the 94.57% of the variance captured. Using this method, I have experimented with the number of PCs need to include as the input the neural network to produce proper results.

$$P_i = \frac{\lambda_i}{\sum_{i=1}^{n} \lambda_i} \tag{3.1}$$

$$C_i = P_i + C_{i-1} \tag{3.2}$$

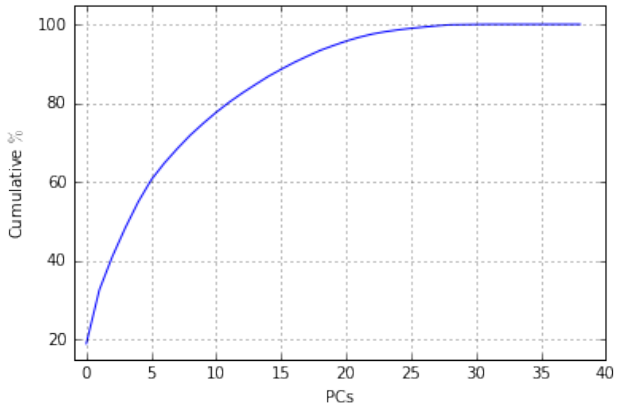



Figure 3.1: Cumulative %

| **Component** | **Eigenvalue** ($\lambda_i$) | **Proportion** % ($P_i$) | **Cumulative** % ($C_i$) |
|---|---|---|---|
| PC1 | 7.4114 | 19.00 | 19.00 |
| PC2 | 5.30315 | 13.60 | 32.60 |
| PC3 | 3.30398 | 8.47 | 41.07 |
| PC4 | 2.84707 | 7.30 | 48.37 |
| PC5 | 2.57537 | 6.60 | 54.97 |
| PC6 | 2.17668 | 5.58 | 60.55 |
| PC7 | 1.63915 | 4.20 | 64.75 |
| PC8 | 1.43721 | 3.69 | 68.44 |
| PC9 | 1.33558 | 3.42 | 71.86 |
| PC10 | 1.16966 | 3.00 | 74.86 |
| PC11 | 1.09168 | 2.80 | 77.66 |
| PC12 | 1.0003 | 2.56 | 80.22 |
| PC13 | 0.901048 | 2.31 | 82.53 |
| PC14 | 0.836118 | 2.14 | 84.67 |
| PC15 | 0.81133 | 2.08 | 86.75 |
| PC16 | 0.7114 | 1.82 | 88.57 |
| PC17 | 0.678007 | 1.74 | 90.31 |
| PC18 | 0.602895 | 1.55 | 91.86 |
| PC19 | 0.584217 | 1.50 | 93.36 |
| PC20 | 0.472781 | 1.21 | 94.57 |
| PC21 | 0.447496 | 1.15 | 95.72 |
| PC22 | 0.380196 | 0.97 | 96.69 |
| PC23 | 0.315781 | 0.81 | 97.50 |
| PC24 | 0.234446 | 0.60 | 98.10 |
| PC25 | 0.183911 | 0.47 | 98.57 |
| PC26 | 0.153009 | 0.39 | 98.96 |
| PC27 | 0.12802 | 0.33 | 99.29 |
| PC28 | 0.123365 | 0.32 | 99.61 |
| PC29 | 0.0969348 | 0.25 | 99.86 |
| PC30 | 0.0174445 | 0.04 | 99.90 |
| PC31 | 0.0152885 | 0.04 | 99.94 |

Table 3.1: Eigenvalues, and the proportion of variation produced by the principal components (Only first 31 components are listed).

| | PC0 | PC1 | PC2 | PC3 | PC4 | PC5 | PC6 | PC7 | PC9 |
|---|---|---|---|---|---|---|---|---|---|
| **boot_messtd** | 0.985574 | -0.0272193 | -0.0382496 | -0.020681 | -0.0615102 | -0.03871 | -0.00229867 | -0.122926 | 0.000757585 |
| **tce_slogg** | -0.00837695 | -0.196582 | -0.0305392 | 0.582 | -0.426112 | 0.453474 | 0.108778 | -0.0888948 | -0.00840663 |
| **tce_mesmad** | 0.108973 | -0.139566 | 0.285851 | 0.204781 | 0.689681 | 0.177966 | 0.00629496 | -0.239434 | 0.013128 |
| **tce_model_snr** | 0.00763855 | 0.0293483 | 0.275337 | 0.429099 | 0.03533 | -0.414147 | 0.0292706 | -0.0503864 | -0.157082 |
| **tce_sma** | 0.0363605 | 0.717049 | 0.469579 | -0.350958 | -0.103719 | 0.119687 | 0.0494162 | -0.0416602 | -0.0126404 |
| **tce_duration** | 0.0239144 | 0.254927 | 0.352592 | -0.0572724 | -0.133736 | 0.0310973 | 0.321123 | 0.0486368 | 0.483072 |
| **tce_max_sngle_ev** | 0.996833 | -0.0295747 | -0.0252242 | -0.0141228 | -0.026105 | -0.00990162 | 0.00111957 | -0.0354227 | -0.00719581 |
| **tce_depth** | 0.0281772 | 0.119758 | 0.373656 | 0.489657 | 0.0218678 | -0.386995 | -0.190517 | -0.0463958 | 0.0776003 |
| **tce_minmesd** | 0.0262406 | 0.451038 | 0.346278 | -0.30045 | -0.0515882 | 0.101336 | 0.0603943 | -0.0998856 | -0.029149 |
| **tce_rsnrmes** | -0.0352112 | -0.133232 | -0.297088 | -0.00298871 | -0.0810706 | 0.178459 | 0.3902 | 0.0947168 | 0.12978 |
| **tce_ldm_coeff4** | -0.0247076 | -0.758891 | 0.487764 | -0.283137 | -0.0818988 | -0.0532799 | -0.0518798 | 0.0150193 | 0.0276525 |
| **tce_period** | 0.0326134 | 0.683568 | 0.466122 | -0.371192 | -0.101386 | 0.142346 | 0.0364275 | -0.0582389 | -0.0371005 |
| **tce_ror** | 0.0401246 | 0.190918 | 0.43162 | 0.619899 | -0.0131904 | -0.472477 | 0.00302559 | -0.0200604 | 0.0855508 |
| **tce_sradius** | 0.0102952 | 0.402955 | 0.0484603 | -0.436943 | 0.337178 | -0.439099 | -0.113251 | 0.16843 | -0.114113 |
| **tce_robstat** | 0.0853849 | -0.00985225 | 0.323367 | 0.274418 | 0.720758 | 0.363828 | 0.0457526 | -0.153424 | -0.0173252 |
| **tce_incl** | 0.0211386 | 0.405332 | 0.106486 | 0.256101 | -0.191366 | 0.196405 | -0.587207 | 0.0630756 | 0.222331 |
| **tce_rminmes** | -0.00971034 | -0.187554 | 0.0401913 | -0.493746 | 0.192367 | -0.0315114 | -0.00935625 | -0.0677288 | 0.27249 |
| **tce_eqt** | -0.0276139 | -0.500732 | -0.261984 | -0.375837 | 0.418173 | -0.346343 | -0.15574 | -0.0438917 | 0.0503787 |
| **tce_model_chisq** | 0.0107883 | 0.0323824 | 0.115602 | 0.112077 | -0.0173611 | -0.161593 | 0.0767894 | -0.0145846 | 0.518074 |
| **tce_bin_oedp_stat** | 0.000602256 | 0.000917913 | 0.0346378 | 0.0552496 | 0.0104175 | -0.0829708 | 0.024532 | -0.0233276 | -0.147528 |
| **tce_max_mult_ev** | 0.989803 | -0.0294388 | -0.0284572 | -0.0140514 | -0.0282611 | -0.015705 | 0.000712676 | -0.110846 | -0.00947719 |
| **tce_dor** | 0.0265683 | 0.561023 | 0.37098 | -0.274905 | -0.086844 | 0.120624 | -0.104687 | -0.0711477 | -0.151388 |
| **tce_chisq1** | 0.185803 | -0.0124196 | 0.177382 | 0.106373 | 0.379572 | 0.271653 | 0.0354826 | 0.798786 | 0.0190882 |
| **tce_steff** | -0.0230577 | -0.576066 | -0.00637538 | -0.312562 | 0.197058 | -0.159451 | -0.023146 | -0.126958 | 0.272363 |
| **tce_impact** | -0.017385 | -0.0823197 | -0.00365774 | -0.0532586 | 0.0383543 | -0.187421 | 0.818341 | -0.0216904 | -0.345382 |
| **tce_minmes** | -0.996345 | 0.0296795 | 0.0203883 | 0.0125951 | 0.0174457 | -0.000444007 | -0.00270876 | -0.0529541 | 0.00417214 |
| **tce_prad** | 0.0205213 | 0.287963 | 0.196505 | -0.0353396 | 0.228277 | -0.536481 | -0.0150112 | 0.1776 | -0.0510106 |
| **tce_smet** | 0.00421656 | 0.260043 | -0.471499 | -0.0443286 | 0.201787 | -0.0474384 | 0.0539475 | -0.131872 | 0.291863 |
| **tce_chisq2** | 0.0761657 | -0.0185815 | 0.272596 | 0.211934 | 0.728614 | 0.420771 | 0.0682671 | -0.191158 | -0.0141495 |
| **tce_rmesmad** | 0.691863 | -0.0210194 | 0.0464222 | 0.0158418 | 0.0773508 | 0.0876311 | 0.0096273 | 0.665474 | 0.0145089 |
| **boot_mesmean** | -0.983517 | 0.0263375 | 0.0397527 | 0.0223489 | 0.0692185 | 0.0439375 | 0.00275629 | 0.126684 | -0.00331832 |
| **tce_maxmes** | 0.991296 | -0.0291171 | -0.0369399 | -0.0203358 | -0.0472821 | -0.0256594 | -0.000341935 | -0.0924002 | -0.00836846 |
| **boot_fap** | 0.000269175 | 0.0310663 | -0.127392 | -0.230928 | -0.0553696 | 0.29396 | 0.0894548 | 0.0490162 | 0.191893 |
| **boot_mesthresh** | 0.985848 | -0.0277626 | -0.0392151 | -0.0210757 | -0.0602013 | -0.037186 | -0.00240554 | -0.122536 | -0.00228809 |

Table 3.2: Correlation between Principal Components and Variables

Furthermore, we can interpret each of the principal components using the correlations between the original data for each variable and principal components. The correlations can be obtain using correlations procedure. The Table 3.2 show the correlation coefficients between PCs (first 10) and original variables (rows). Due to the standardization, all PCs will have mean of zero. Standard Deviation is also given for each of the components and these will be the square root of the eigenvalue. The interpretation of principal components is based on the variables are most strongly correlated with each component, i.e, which of these number are large in magnitude, the farthest from zero in either positive or negative direction. Table 3.2 show the correlation values greater than 0.7 in one color (dark blue) and between 0.4 and 0.7 (lighter blue) in one color and the negative correlations using other colors (dark yellow and lighter yellow). These interpretation allow us to have a clear understanding which variables are correlated with which principal component. However, to continue with the calculated principal components as input to the neural network, this knowledge is not nessasary, this is purely to understand the dataset in depth to interpret physics behind it.

## 3.3 Training and Testing Data

Before we train the neural network with the transformed TCE dataset, we ned to split the data into two sections: training and testing data. Training the network with one dataset and testing with another dataset allow us to avoid overfitting. To split the data into these sets, we are using the `scikit-learn` a random `split` `train_test_split` helper function. We can tell this method what ratio it should need to split the data. Once we do that, the method split the data according to the ratio randomly selecting a number of data records as training and testing. The

helper method takes the TCE catalog data and the labels as parameters.

## 3.4 Training the Neural Network

Once the data has been cleanup, standardized, transform using Principal component analysis and split into testing and training data sets, next step is to train a neural network. We create a Multi-layered perceptron (MLP) that trains using backpropagation network using `MLPClassifier` in `sklearn.neural_network` package. When creating the MLP, there are several parameters we need to define. These parameters are picked after analyzing the behavior of them with the dataset.

### 3.4.1 Activation Function

There are several built-in activation functions are in the scikit learn. Primarily these activations functions are divided into two different categories for two different output layers: non-linear and linear or softmax layers. We compute the loss function for activation functions where we can see the how the loss function behave. List of activation functions:

- *Identity*: this is a linear function without changing the input, this is useful to implement linear bottleneck, Equation 3.3

$$f(x) = x \tag{3.3}$$

- *Logistic*: This is a logistic sigmoid function, Equation 3.4

$$f(x) = \frac{1}{(1 + e^{-x})} \tag{3.4}$$

- *Relu*: This is a rectified linear unit function, Equation 3.5

$$f(x) = max(0, x) \tag{3.5}$$

- *tanh*: The hyperbolic tan function, Equation 3.6

$$f(x) = tanh(x) \tag{3.6}$$

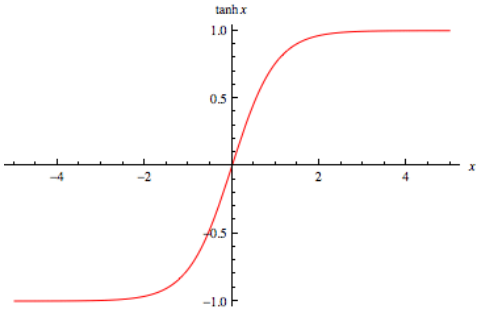



Figure 3.2: *tanh* function

After some experiments and the nature of the *tanh* function (Figure 3.2), we have picked the *tanh* function as the activation function for the network.

**Loss Function**

Log loss is also known as logistic regression loss or cross-entropy loss is defined on probability estimate. Usual application for logistic regression is a binary classification, however, it is easy to extend this to a multiclass case [2], Equation 3.7.

$$L_{log}(Y, P) = -logPr(Y|P) = -\frac{1}{N}\sum_{i=0}^{N-1}\sum_{k=0}^{K-1} y_{i,k} log p_{i,k} \tag{3.7}$$

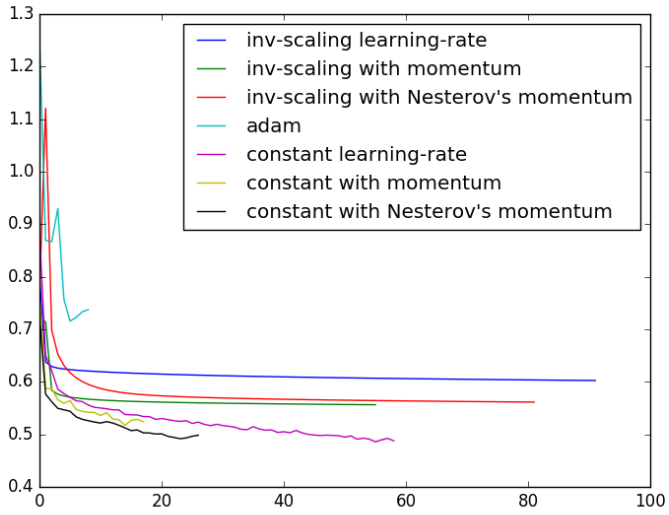



Figure 3.3: Loss function with some learning rate methods. x-axis: number of iterations. y-axis: loss function value

### 3.4.2 Learning Rate - $\eta$

It was appetent the behavior of the network with the activation function being the tanh highly depend on the learning rate. Learning rate ($\eta$) is explained using the equation 2.11 in the section 2.5.2. scikit lean allow to configure the neural network with several different learning rate methods:

- *constant*: A constant learning rate, the value is taken as an integer value
- *invscaling*: Gradually decreases the learning rate at each time step $t$ using an inverse scaling exponent of *power_t*.
- *adaptive*: Keep the learning rate constant as long as training loss keeps decreasing. Each time two consecutive epochs fail to decrease training loss by at least *tol* ($= 0.0001$), or fail to increase validation score by at least an amount

that defined as tol, or if *early_stopping* is *on*, the current learning rate is divided by 5.

Figure 3.3 show the behavior of the loss function for several learning rates we have experimented with the dataset. The goal is to bring the loss function to close to zero that indicates the convergence of the network. The x-axis shows the number of iterations.

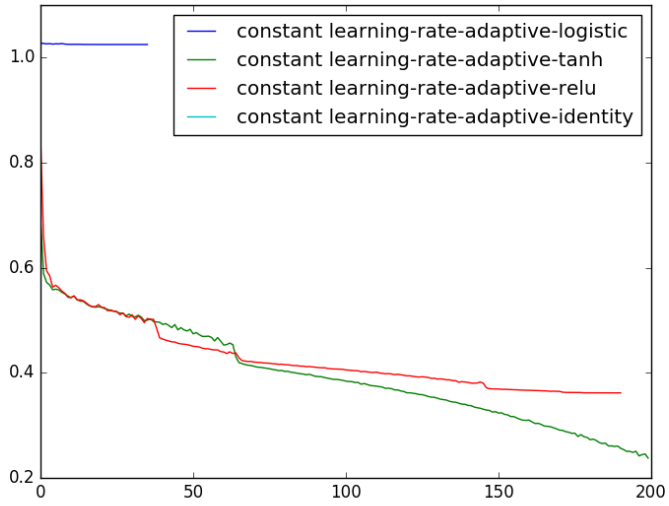



Figure 3.4: Loss function with constant learning rate with different activation methods. x-axis: number of iterations. y-axis: loss function value

After seeing the constant learning rate is keep reducing the loss function, I have experimented with the constant learning rate (set for 0.2) with different activation functions. Figure 3.4 show some results of this work. It is clear that constant learning rate with adaptive method for the learning rate and *tanh* for the activation function is quite impressively kept reducing the loss function. I have tested with this particular combination further and the result is shown in the figure 3.5. Constant

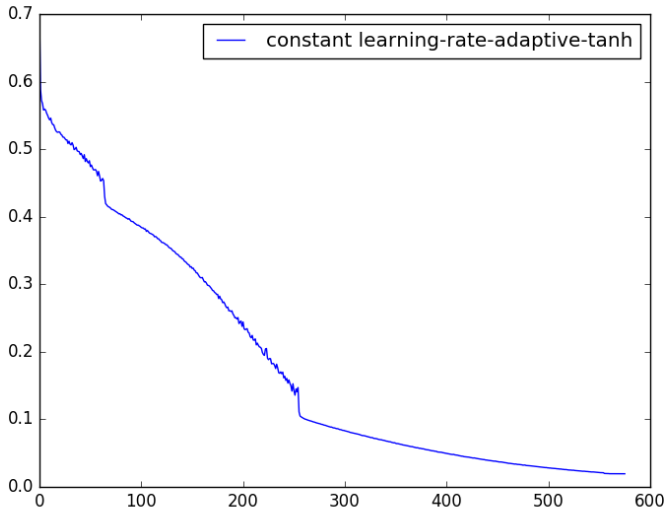



Figure 3.5: Loss function with constant adaptive learning rate with *tanh* activation method converge the network. x-axis: number of iterations. y-axis: loss function value

learning rate with adaptive method combine with the *tanh* activation method converge the network close to 600 iterations, thus we are settling with this configuration for to train the neural network with TCE catalog data.

MLPClassifier trains iteratively since each time step the partial derivative of the loss function with respect to the model parameters are computed to update the parameters. Once we fix the parameters of the network, we train the network with training dataset we pick out of the entire catalog. This training process took about 15 minutes to complete, once this complete I save the trained classifier as a python `pickle` object where the program can load the train network to test the test dataset without keep training every single time. The program config file let you pick a name for the train network where the program looks for a `pickle` file under that name at the beginning of the program to load.

# Chapter 4

# Results

★ Model evaluation and validation

★ - Explain the final model paramters, explain

★ - Explain the and show the loss funstion, show the confision matrix

★ justification

★ - Compare the final results to the bench mark - compare the confustion matrix to the once we have on the paper, also compare to the graph the score value, explain and show the problem is adiculately solved

## 4.1 Confusion Matrix

At this point, we have a trained model with optimized parameters. In order for the model to be evaluated, we compute the confusion matrix $C$ is also known as error matrix. This matrix allows us to visualize of the performance of the algorithm. Each column of the matrix represents the instances in a predicted class while each row represents the instances in an actual class. This matrix shows the True Positive (TP), False Negative (FN), False Positive (FP) and True Negative (TN).

The diagonal elements of the matrix represent the number of points for which the predicted label is equal to the true label, while off-diagonal elements are those that are mislabeled by the classifier. Figure 4.1 show the normalized confusion matrix for a MLP network which is not optimized yet. This kind of normalization can be interesting in case of class imbalance to have a more visual interpretation of which class is being misclassified. Labels are: PC (Planetary Candidates), NTP (None Transiting Phenomena), AFP (Astrophysical False Positive)

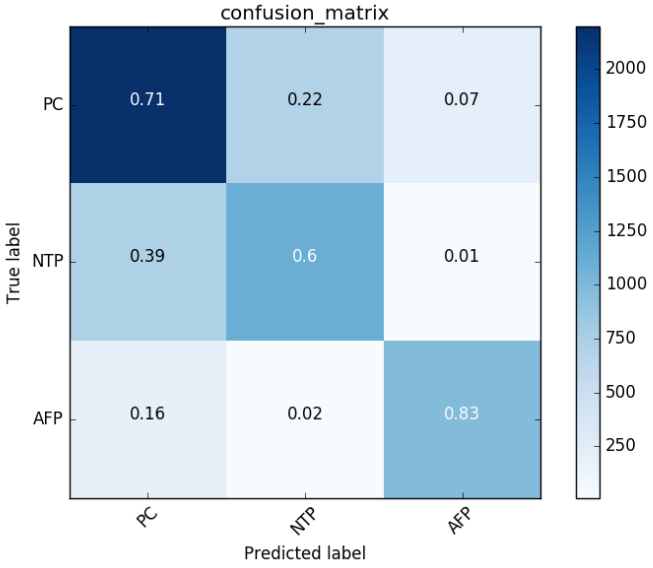



Figure 4.1: Normalized Confusion Matrix. Unoptimized network predictions . Model Parameters are: `activation = identity, learning_rate_init_num = 0.2, momentum_num = 0`

I also compute the accuracy (score) of the predictions. In multilabel classification, the function returns the subset accuracy. If the entire set of predicted labels for a sample strictly match with the true set of labels, then he subset accuracy is 1.0; otherwise, it is 0.0. The accuracy is computed using equation 4.1. For the example shown in the figure 4.1 has the accuracy score of 0.687.

$$accuracy(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}} I(\hat{y}_i = y_i) \tag{4.1}$$

where, $\hat{y}$t is the predicted value of the $i^{th}$ sample $y_i$ is the corresponding true value and $I(x)$ is the indicator function.

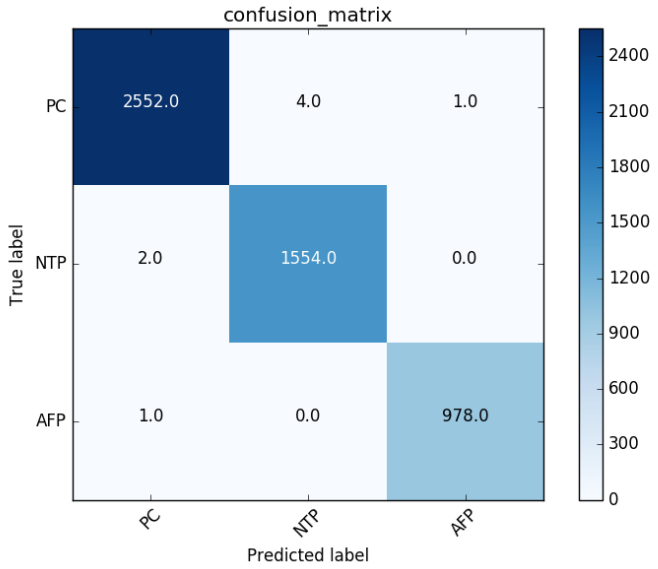



Figure 4.2: Optimized MLP network predictions.

Finally, the confusion matrix that produced by the classifier that is properly tuned is shown in the figure 4.2. I am showing the confusion matrix without normalizing it. The accuracy score that produces by this classifier is 0.998. Due to high accuracy, the normalized matrix will have 1.0 for the diagonal values and the 0.0 for the off-diagonal values. It is clear the network is capable of classifying the Kepler Threshold Crossing Event Catalog with a high degree of accuracy. Glancing at the confusion matrix, only 5 of the event are falsely labeled for the known planetary candidates (4 as NTP and 1 as an AFP). For the NTP its only 2 of the events are incorrectly classify as PCs and for the AFP there is only 1 one incorrect

classification.

# Chapter 5

# Conclusion

★ Conclusion discussion - based on the conclusion section of the paper

★ Describe the entier process - highlight whats interesting and whats difficult, interesting - Astrophysics can replace the physical data processing piplelines with machine learning and allow real time dataprocessing with the large data comming from the telescopes

★ How can I Improve - this is where explain we can now do this using light curves instead using TCE catalog

# Bibliography

[1] G. Anglada-Escudé, P. J. Amado, J. Barnes, Z. M. Berdiñas, R. P. Butler, G. A. L. Coleman, I. de La Cueva, S. Dreizler, M. Endl, B. Giesers, S. V. Jeffers, J. S. Jenkins, H. R. A. Jones, M. Kiraga, M. Kürster, M. J. López-González, C. J. Marvin, N. Morales, J. Morin, R. P. Nelson, J. L. Ortiz, A. Ofir, S.-J. Paardekooper, A. Reiners, E. Rodríguez, C. Rodríguez-López, L. F. Sarmiento, J. P. Strachan, Y. Tsapras, M. Tuomi, and M. Zechmeister. A terrestrial planet candidate in a temperate orbit around Proxima Centauri. *nat*, 536:437–440, August 2016.

[2] Christopher M Bishop. Pattern recognition. *Machine Learning*, 128:1–58, 2006.

[3] William J. Borucki, David Koch, Gibor Basri, Natalie Batalha, Timothy Brown, Douglas Caldwell, John Caldwell, Jørgen Christensen-Dalsgaard, William D. Cochran, Edna DeVore, Edward W. Dunham, Andrea K. Dupree, Thomas N. Gautier, John C. Geary, Ronald Gilliland, Alan Gould, Steve B. Howell, Jon M. Jenkins, Yoji Kondo, David W. Latham, Geoffrey W. Marcy, Søren Meibom, Hans Kjeldsen, Jack J. Lissauer, David G. Monet, David Morrison, Dimitar Sasselov, Jill Tarter, Alan Boss, Don Brownlee, Toby Owen, Derek Buzasi, David Charbonneau, Laurance Doyle, Jonathan Fortney, Eric B. Ford, Matthew J. Holman, Sara Seager, Jason H. Steffen, William F. Welsh, Jason

Rowe, Howard Anderson, Lars Buchhave, David Ciardi, Lucianne Walkowicz, William Sherry, Elliott Horch, Howard Isaacson, Mark E. Everett, Debra Fischer, Guillermo Torres, John Asher Johnson, Michael Endl, Phillip MacQueen, Stephen T. Bryson, Jessie Dotson, Michael Haas, Jeffrey Kolodziejczak, Jeffrey Van Cleve, Hema Chandrasekaran, Joseph D. Twicken, Elisa V. Quintana, Bruce D. Clarke, Christopher Allen, Jie Li, Haley Wu, Peter Tenenbaum, Ekaterina Verner, Frederick Bruhweiler, Jason Barnes, and Andrej Prsa. Kepler planet-detection mission: Introduction and first results. *Science*, 327(5968):977–980, 2010.

[4] D. Charbonneau, T. M. Brown, D. W. Latham, and M. Mayor. Detection of Planetary Transits Across a Sun-like Star. *apjl*, 529:L45–L48, January 2000.

[5] A. Claret and S. Bloemen. VizieR Online Data Catalog: Limb-darkening coefficients (Claret+, 2011). *VizieR Online Data Catalog*, 352, March 2011.

[6] P. Dubath, L. Rimoldini, M. Süveges, J. Blomme, M. López, L. M. Sarro, J. De Ridder, J. Cuypers, L. Guy, I. Lecoeur, K. Nienartowicz, A. Jan, M. Beck, N. Mowlavi, P. De Cat, T. Lebzelter, and L. Eyer. Random forest automated supervised classification of Hipparcos periodic variable stars. *mnras*, 414:2602–2617, July 2011.

[7] Eric D. Feigelson and G. Jogesh Babu. *Modern Statistical Methods for Astronomy: With R Applications*. Cambridge University Press, Cambridge, 007 2012.

[8] Jonathan P. Gardner, John C. Mather, Mark Clampin, Rene Doyon, Matthew A. Greenhouse, Heidi B. Hammel, John B. Hutchings, Peter Jakob-

sen, Simon J. Lilly, Knox S. Long, Jonathan I. Lunine, Mark J. Mccaughrean, Matt Mountain, John Nella, George H. Rieke, Marcia J. Rieke, Hans-Walter Rix, Eric P. Smith, George Sonneborn, Massimo Stiavelli, H. S. Stockman, Rogier A. Windhorst, and Gillian S. Wright. The james webb space telescope. *Space Science Reviews*, 123(4):485–606, 2006.

[9] Leonard GC Hamey. Benchmarking feed-forward neural networks: models and measures. In *NIPS*, pages 1167–1174. Citeseer, 1991.

[10] John Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the theory of neural computation*, volume 1. Addison-Wesley Pub. Co, Redwood City, Calif, 1991.

[11] S. B. Howell, C. Sobeck, M. Haas, M. Still, T. Barclay, F. Mullally, J. Troeltzsch, S. Aigrain, S. T. Bryson, D. Caldwell, W. J. Chaplin, W. D. Cochran, D. Huber, G. W. Marcy, A. Miglio, J. R. Najita, M. Smith, J. D. Twicken, and J. J. Fortney. The K2 Mission: Characterization and Early Results. *pasp*, 126:398–408, April 2014.

[12] J. M. Jenkins, D. A. Caldwell, H. Chandrasekaran, J. D. Twicken, S. T. Bryson, E. V. Quintana, B. D. Clarke, J. Li, C. Allen, P. Tenenbaum, H. Wu, T. C. Klaus, C. K. Middour, M. T. Cote, S. McCauliff, F. R. Girouard, J. P. Gunter, B. Wohler, J. Sommers, J. R. Hall, A. K. Uddin, M. S. Wu, P. A. Bhavsar, J. Van Cleve, D. L. Pletcher, J. A. Dotson, M. R. Haas, R. L. Gilliland, D. G. Koch, and W. J. Borucki. Overview of the Kepler Science Processing Pipeline. *apjl*, 713:L87–L91, April 2010.

[13] D. G. Koch and et al. Kepler Mission Design, Realized Photometric Performance, and Early Science. *apjl*, 713:L79–L86, April 2010.

[14] R. K. Kopparapu, R. M. Ramirez, J. SchottelKotte, J. F. Kasting, S. Domagal-Goldman, and V. Eymet. Habitable Zones around Main-sequence Stars: Dependence on Planetary Mass. *apjl*, 787:L29, June 2014.

[15] Holger R Maier. *Use of artificial neural networks for modelling multivariate water quality times series/by Holger Robert Maier.* PhD thesis, 1995.

[16] K. Mandel and E. Agol. Analytic Light Curves for Planetary Transit Searches. *apjl*, 580:L171–L175, December 2002.

[17] S. D. McCauliff, J. M. Jenkins, J. Catanzarite, C. J. Burke, J. L. Coughlin, J. D. Twicken, P. Tenenbaum, S. Seader, J. Li, and M. Cote. Automatic Classification of Kepler Planetary Transit Candidates. *apj*, 806:6, June 2015.

[18] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[19] E. A. Petigura and G. W. Marcy. Identification and Removal of Noise Modes in Kepler Photometry. *pasp*, 124:1073, October 2012.

[20] H. Rauer, C. Catala, C. Aerts, T. Appourchaux, W. Benz, A. Brandeker, J. Christensen-Dalsgaard, M. Deleuil, L. Gizon, M.-J. Goupil, M. Güdel, E. Janot-Pacheco, M. Mas-Hesse, I. Pagano, G. Piotto, D. Pollacco, Ċ. Santos, A. Smith, J.-C. Suárez, R. Szabó, S. Udry, V. Adibekyan, Y. Alibert, J.-M. Almenara, P. Amaro-Seoane, M. A.-v. Eiff, M. Asplund, E. Antonello, S. Barnes, F. Baudin, K. Belkacem, M. Bergemann, G. Bihain, A. C.

Birch, X. Bonfils, I. Boisse, A. S. Bonomo, F. Borsa, I. M. Brandão, E. Brocato, S. Brun, M. Burleigh, R. Burston, J. Cabrera, S. Cassisi, W. Chaplin, S. Charpinet, C. Chiappini, R. P. Church, S. Csizmadia, M. Cunha, M. Damasso, M. B. Davies, H. J. Deeg, R. F. Díaz, S. Dreizler, C. Dreyer, P. Eggenberger, D. Ehrenreich, P. Eigmüller, A. Erikson, R. Farmer, S. Feltzing, F. de Oliveira Fialho, P. Figueira, T. Forveille, M. Fridlund, R. A. García, P. Giommi, G. Giuffrida, M. Godolt, J. Gomes da Silva, T. Granzer, J. L. Grenfell, A. Grotsch-Noels, E. Günther, C. A. Haswell, A. P. Hatzes, G. Hébrard, S. Hekker, R. Helled, K. Heng, J. M. Jenkins, A. Johansen, M. L. Khodachenko, K. G. Kislyakova, W. Kley, U. Kolb, N. Krivova, F. Kupka, H. Lammer, A. F. Lanza, Y. Lebreton, D. Magrin, P. Marcos-Arenal, P. M. Marrese, J. P. Marques, J. Martins, S. Mathis, S. Mathur, S. Messina, A. Miglio, J. Montalban, M. Montalto, M. J. P. F. G. Monteiro, H. Moradi, E. Moravveji, C. Mordasini, T. Morel, A. Mortier, V. Nascimbeni, R. P. Nelson, M. B. Nielsen, L. Noack, A. J. Norton, A. Ofir, M. Oshagh, R.-M. Ouazzani, P. Pápics, V. C. Parro, P. Petit, B. Plez, E. Poretti, A. Quirrenbach, R. Ragazzoni, G. Raimondo, M. Rainer, D. R. Reese, R. Redmer, S. Reffert, B. Rojas-Ayala, I. W. Roxburgh, S. Salmon, A. Santerne, J. Schneider, J. Schou, S. Schuh, H. Schunker, A. Silva-Valio, R. Silvotti, I. Skillen, I. Snellen, F. Sohl, S. G. Sousa, A. Sozzetti, D. Stello, K. G. Strassmeier, M. v Svanda, G. M. Szab, A. Tkachenko, D. Valencia, V. Van Grootel, S. D. Vauclair, P. Ventura, F. W. Wagner, N. A. Walton, J. Weingrill, S. C. Werner, P. J. Wheatley, and K. Zwintz. The PLATO 2.0 mission. *Experimental Astronomy*, 38:249–330, November 2014.

[21] G. R. Ricker, J. N. Winn, R. Vanderspek, D. W. Latham, G. Á. Bakos,

J. L. Bean, Z. K. Berta-Thompson, T. M. Brown, L. Buchhave, N. R. Butler, R. P. Butler, W. J. Chaplin, D. Charbonneau, J. Christensen-Dalsgaard, M. Clampin, D. Deming, J. Doty, N. De Lee, C. Dressing, E. W. Dunham, M. Endl, F. Fressin, J. Ge, T. Henning, M. J. Holman, A. W. Howard, S. Ida, J. Jenkins, G. Jernigan, J. A. Johnson, L. Kaltenegger, N. Kawai, H. Kjeldsen, G. Laughlin, A. M. Levine, D. Lin, J. J. Lissauer, P. MacQueen, G. Marcy, P. R. McCullough, T. D. Morton, N. Narita, M. Paegert, E. Palle, F. Pepe, J. Pepper, A. Quirrenbach, S. A. Rinehart, D. Sasselov, B. Sato, S. Seager, A. Sozzetti, K. G. Stassun, P. Sullivan, A. Szentgyorgyi, G. Torres, S. Udry, and J. Villasenor. Transiting Exoplanet Survey Satellite (TESS). In *Space Telescopes and Instrumentation 2014: Optical, Infrared, and Millimeter Wave*, volume 9143 of *procspie*, page 914320, August 2014.

[22] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.

[23] J. C. Smith, M. C. Stumpe, J. E. Van Cleve, J. M. Jenkins, T. S. Barclay, M. N. Fanelli, F. R. Girouard, J. J. Kolodziejczak, S. D. McCauliff, R. L. Morris, and J. D. Twicken. Kepler Presearch Data Conditioning II - A Bayesian Approach to Systematic Error Correction. *pasp*, 124:1000–1014, September 2012.

[24] CI Teh, KS Wong, ATC Goh, and S Jaritngam. Prediction of pile capacity using neural networks. *Journal of computing in civil engineering*, 11(2):129–138, 1997.

[25] G. Torres, D. M. Kipping, F. Fressin, D. A. Caldwell, J. D. Twicken, S. Ballard, N. M. Batalha, S. T. Bryson, D. R. Ciardi, C. E. Henze, S. B. Howell, H. T. Isaacson, J. M. Jenkins, P. S. Muirhead, E. R. Newton, E. A. Petigura, T. Barclay, W. J. Borucki, J. R. Crepp, M. E. Everett, E. P. Horch, A. W. Howard, R. Kolbl, G. W. Marcy, S. McCauliff, and E. V. Quintana. Validation of 12 Small Kepler Transiting Planets in the Habitable Zone. *apj*, 800:99, February 2015.

[26] J.R. Voelkel. *Johannes Kepler and the New Astronomy*. Oxford Portraits in Science. Oxford University Press, USA, 2001.

[27] A. N. Youdin and F. H. Shu. Planetesimal Formation by Gravitational Instability. *apj*, 580:494–505, November 2002.