# AUTOMATIC CLASSIFICATION OF MICROLENSING CANDIDATES

A Thesis

by

Virisha Timmaraju

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

| | |
|---|---|
| Chair of Committee, | Scott L. Miller |
| Co-Chair of Committee, | Louis E. Strigari |
| Committee Members, | Srinivas Shakkottai |
| | Jean-Francois Chamberland-Tremblay |
| Head of Department, | Miroslav M. Begovic |

August 2018

Major Subject: Electrical Engineering

## ABSTRACT

It is both exciting and important to look for life beyond our planet. To find signs of life on distant planets, there is a need to search across the vast space that surrounds us and find planets outside our solar system, called exoplanets. Among the many search techniques which have been developed to detect exoplanets, 'microlensing' holds the advantage of finding Earth-like planets. In order to detect a microlensing event, there is a need to scan millions of stars simultaneously for the case of perfect alignment of two stars. This chance alignment typically lasts for weeks or days, until the two stars move out of alignment. Hence, there is a need to follow up on all detected events in real-time, to capture information about the properties of the star system. Large scale astronomical surveys like the Global Astrometric Interferometer for Astrophysics (Gaia) mission and Large Synoptic Survey Telescope (LSST) will capture terabytes of data every night. Hence, building an automatic classifier, using tools from machine learning in order to sift through this data and detect microlensing events is crucial.

The scope of work includes identification and development of three appropriate methods to establish an automatic classifier. The first method makes classification decisions based on five characteristics of microlensing translated into statistical features. The second and third methods detect microlensing events without relying on any specific characteristics of microlensing, but differ in the way they handle data. These methods are applied to datasets from three different astronomical surveys and the results thus obtained are evaluated to make sure that all the occurrences of this rare event, microlensing, are detected. The third method uses an RNN to detect all the events in the training set. It is concluded that, this method can be easily extended to exoplanet detection.

## DEDICATION

To my mother, my father and the extraterrestrials. One step closer.

## ACKNOWLEDGEMENTS

I would like to extend my deepest gratitude to my advisors, Dr. Scott L. Miller and Dr. Louis E. Strigari, for providing the opportunity to pursue my passion and imparting the essence of academic research. Their technical expertise, encouragement, patience and constant belief in me were essential to the completion of this dissertation.

I wish to thank Dr. Srinivas Shakkottai and Dr. Jean-Francois Chamberland for serving as members of my thesis committee. I am forever indebted to Dr. James Long for introducing me to the Astro-Statistics committee at SAMSI. Those collaborations have greatly helped me understand the intricacies of my thesis domain and instilled in me, a fascination for academic research.

I am grateful to Avinash Vem, Surya Kiran Kanumilli, Aniket Bonde, Aditya Timmaraju, Marcin Jurek and Geethik Kamineni for their vital assistance at different stages of this work. Avinash and Surya provided me with invaluable resources for pursuing an interdisciplinary project. Aniket, Aditya and Marcin guided me in the machine learning ventures for my thesis. Geethik helped me deal with every roadblock in the thesis, discussing technical aspects about the ideation of some of the methods and always being my support system.

I wish to thank the ECE and Astronomy departments at TAMU for providing me with the opportunity to achieve my academic and professional goals. I would like to thank my family and friends for their constant support. Lastly, I am indebted to the StackOverflow community for steering my late-night trysts with programming and machine learning.

## CONTRIBUTORS & FUNDING SOURCES


### Contributors

The work was supported by a thesis committee consisting of Professor Scott Miller, Professor Louis Strigari, Professor Jean-Francois Chamberland and Professor Srinivas Shakkottai.

### Funding Sources

My graduate study was partly supported by a merit scholarship from the department of Electrical and Computer Engineering at Texas A&M University.

## NOMENCLATURE

| | |
|---|---|
| E.T | Extra-Terrestrial |
| GAIA | Global Astrometric Interferometer for Astrophysics |
| LSST | Large Synoptic Survey Telescope |
| OGLE | Optical Gravitational Lensing Experiment |
| SDSS | Sloan Digital Sky Survey |
| MACHO | Massive Compact Halo Objects |
| ML | Machine Learning |
| ACF | Autocorrelation Function |
| CCF | Cross-Correlation Function |
| TN | True Negative |
| TP | True Positive |
| FN | False Negative |
| FP | False Positive |
| LDA | Linear Discriminant Analysis |
| QDA | Quadratic Discriminant Analysis |
| RF | Random Forests |
| SVM | Support Vector Machines |
| NN | Neural Network |
| RNN | Recurrent Neural Network |

## TABLE OF CONTENTS

Page

## LIST OF FIGURES

FIGURE Page

## LIST OF TABLES

TABLE Page

# 1. INTRODUCTION

It is very exciting to look beyond the normal scope of life and get involved in the search for life beyond earth. Life that sustains on planets other than the Earth is called extraterrestrial life (E.T). There are many theories employed in extra-terrestrial search techniques; examining the atmospheres of various planets for gases that could cook life, by measuring the spectrum of the star it is orbiting. As the star emits these gases, the planets surrounding them absorb it in different combinations, thus forming unique signatures composed of gases. These planets, present outside our solar system, are called exoplanets.

While the search for E.T is one motivation to find them, another important reason to look for exoplanets that could host life is for us to inhabit; since the Earth may become uninhabitable as the Sun gets older, larger and warmer.

There are various exoplanet search techniques like radial velocity searches, transit photometry and microlensing. Radial velocity search refers to the variations in the speed with which a star moves towards or away from the Earth. These variations in a star's speed occur due to the gravitational pull of the planets orbiting it. Transit photometry refers to the brief decrease in brightness of a star, as a planet orbiting it crosses (transits). Microlensing refers to the magnification in the brightness of a star due to its alignment with another star whose gravitational pull acts like a lens.

While radial velocity searches can look for planets with a distance of upto 100 light years away and transit photometry can look for those which are several hundreds of light years away, microlensing can look for planets which are thousands of light years away from us.

Microlensing can help detect the farthest and smallest of planets. While radial velocity and transit photometry can well detect planets very closely orbiting their stars, microlensing is good at detecting planets orbiting their stars at moderate to large distances. This is another great advantage since planets orbiting stars at moderate distances have certain conditions like

temperature, which are very similar to those on Earth. Whereas closely orbiting planets would have very high temperatures, unsuitable for most forms of life. Hence, it is justified to say that microlensing plays a crucial role in exoplanet searches. The phenomena of microlensing is discussed in detail, in the following sub-section.

### 1.1 Microlensing

Microlensing is observed when, a massive object (lens star) bends the light of a bright background object (source star) with its gravitational pull, generating a magnified image of the source star as shown in figure 1 below. As the lens star moves in the foreground of the source star, the magnified image of the source star gets brighter, reaches a peak and then gets dimmer in a smooth way, over a period of few days or weeks.

Additionally, when a planet present close to the lens star passes through one of the source star's light streams, the planet's gravitational pull also affects the lensing phenomena. This indicates the presence of a planet. Based on the duration of the microlensing event and its intensity, the characteristics of a planet ,like its mass, orbit etc., can be derived.

Exoplanets are detected using microlensing as follows; During the initial experiments on microlensing events, a prominent brightening of star was observed in the pattern and it was concluded that it was due to the gravitational pull of a star resulting in gravitational microlensing. Later on, additional spikes of brightness were observed and it was concluded that there were two objects responsible for the spikes. Upon further observation of this brightness pattern, it was noted that the second object was smaller and of much less mass compared to that of the lensing star. This helped in concluding that the second object is a planet orbiting the lensing star. As planets affect the brightness of a background star, exoplanets can be easily detected by scanning millions of stars simultaneously for a case of perfect alignment.

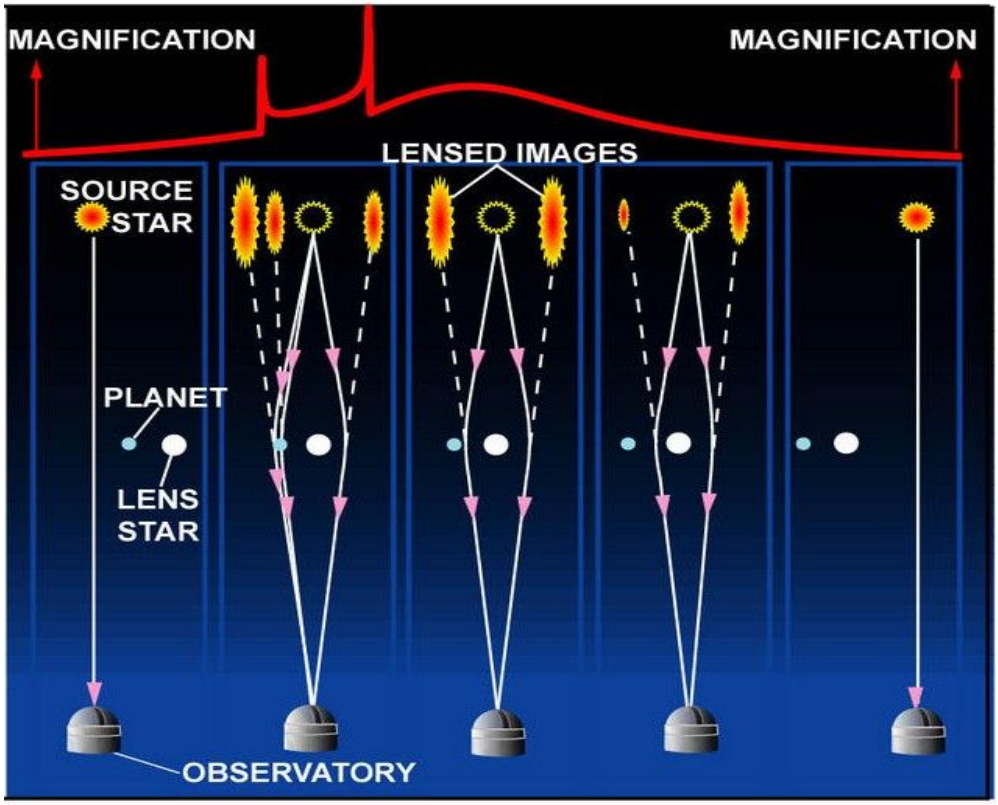


Figure 1: Gravitational microlensing phenomenon [1]

This entire process is depicted in figure 1 above, as we look from the rightmost column to the leftmost one. In the first step, an observatory on Earth is observing a bright source star. When another star comes to the foreground of the source star, it acts like a lens and bends the light, resulting in the magnification of source star's brightness ('magnitude'). It is important to note that, in astronomy, an increase in a star's brightness is observed as a decrease in magnitude. When a planet orbiting this lens star is struck by light from the source star, there is a brief increase in brightness again before the lens star moves out of alignment and the source star goes back to its normal brightness level. This measurement of brightness varying over time is called a light curve. The top part of this figure depicts the associated light curve. The objective is to detect the occurrence of a microlensing event by observing the light curve of the source star.

There are many challenges associated with the analysis of these light curves due to the problems encountered while recording them. Details of some of those challenges and ways to handle them are discussed in the following sub-section.

### 1.2 Challenges In The Analysis Of Astronomical Data

The light curve data of stars is observed, collected and documented by various astronomical surveys. Astronomical surveys like the Sloan Digital Sky Survey (SDSS), Global Astrometric Interferometer for Astrophysics (Gaia) mission and Large Synoptic Survey Telescope (LSST), send in terabytes of data; LSST every night and Gaia, SDSS, by the end of their operation. There are certain challenges while analyzing the data obtained through these surveys, some of which are described;

- Astronomical data is vast, as these surveys operate in specific regions of the universe and document data of most of the stars present there. Hence, the volume of astronomical data thus generated poses a challenge in the phase of analysis.
- Additionally, separating brightening effects due to physical phenomena like microlensing, from those occurring due to faulty mechanical objects in the telescopic equipment etc., is difficult.
- The light curve data collected is often unevenly sampled, having no observations for certain periods of time, due to bad weather conditions and other reasons. This is a challenge since there are many stars with missing observations.

Information can't be lost, by clouding it with ambiguity, drawing erroneous conclusions or waiting for astronomical phenomena to happen again. It is important to analyze data, efficiently and accurately, as it comes. For example, the LSST is going to scan the sky every night and is estimated to generate about 60 petabytes of data by the end of the project. With mere man-power it is impossible to manage the data generated through this survey. This is also the

case with data from the Gaia mission. This paves the way for efficient data analysis techniques like machine learning, which could reduce or eradicate manual intervention to a great extent.

Machine Learning, simply put, is a computer teaching itself to grow and change when exposed to new data. An introduction to all the steps involved in a typical machine learning pipeline is provided in the following sub-section.

### 1.3 Machine Learning Pipeline

The machine learning pipeline shown in figure 2 below, depicts the series of steps involved in solving a problem using machine learning.

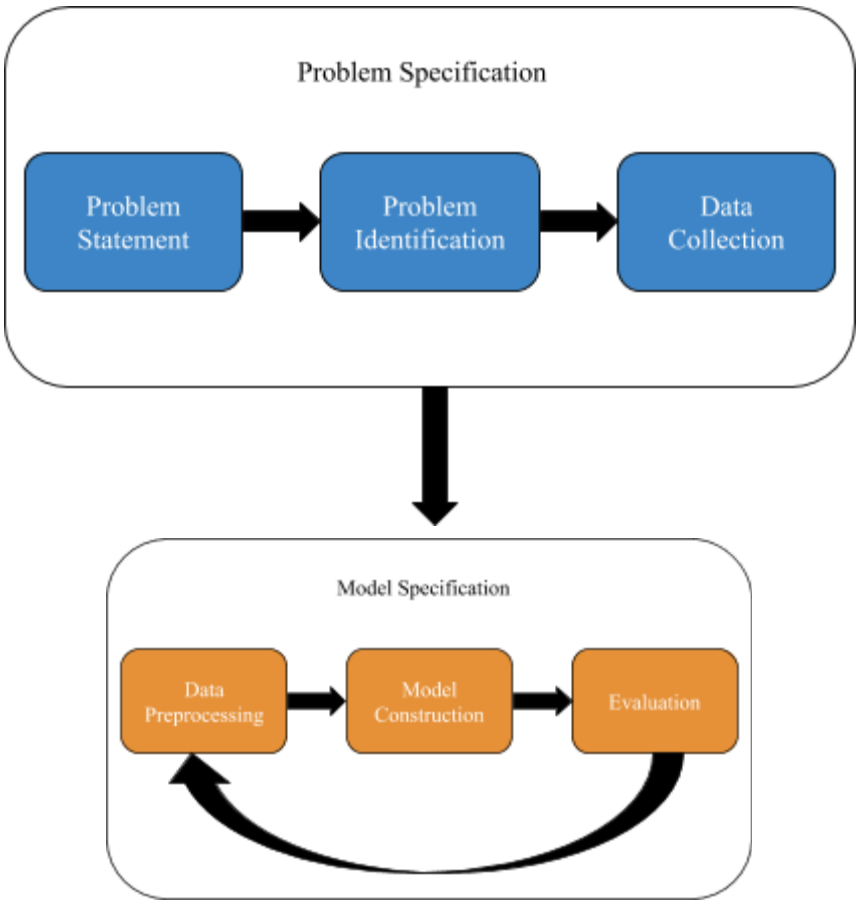


Figure 2: Machine learning pipeline

As depicted in figure 2 above, there are two major steps involved in a machine learning pipeline; problem specification and model specification. Each of the steps is described in detail as follows;

### 1.3.1 Problem Specification

This step involves the formulation of a problem statement, identification with one of the existing problem types in machine learning and collection of datasets helpful for the machine to learn. Each of these is described in detail as follows;

1. Problem Statement - A problem that one wants to solve needs to be defined informally. Then, the motivation for solving the problem and the impact of the solution needs to be known to the person who formulated the problem statement. In order to convert this formulated problem to a machine learning problem, the next sub-step, problem identification, is done.
2. Problem Identification - The formulated problem can be aligned with one of the commonly used types of machine learning problems; Supervised Learning, Unsupervised Learning etc. They are described as follows;
    - Supervised Learning - In this case, the machine is fed with examples of input-output pairs and is posed with the task of deciding outputs for new inputs given by the user. The machine decides outputs by inferring a function from the fed examples and mapping new inputs to it. There are two types of supervised learning, namely, Classification and Regression. Classification assigns the inputs to discrete categories (classes or labels), whereas Regression predicts the future values of a continuous quantity.
    - Unsupervised Learning - In contrast to supervised learning, where the machine finds patterns from existing correct outputs to a given set of inputs, unsupervised learning finds patterns without knowing any outputs to the inputs. Clustering is

the most popular type of unsupervised learning, which groups all the similar inputs together, such that each group is different.

It is clear from the above definitions that, to solve any problem using machine learning, it is required to feed example input-output pairs or just inputs (in the case of Unsupervised Learning) to the machine, to solve a problem. Hence, collection of datasets is an important step in machine learning and is described below.

3. Data Collection - A collection of input-output pairs is called a dataset. A typical dataset comprises samples, features and labels. They are described as follows;
    - Samples - Each example of input-output pair is termed as a sample. Many such samples are fed to the machine. Each sample is stored as a row in the dataset.
    - Features - The measurable properties or characteristics of inputs are termed as features. The number of features used to represent an input is under the discretion of the user. The set of features is chosen in such a way that it helps to differentiate between the categories of inputs. Each feature is stored as a column in the dataset.
    - Labels - The outputs or categories of inputs are called labels. Labels are fed to the machine only in supervised learning problems. Labels are stored as a single column in the dataset.

As the problem definition, problem identification and data collection necessary for the machine to solve the problem have been completed, a problem solving model is discussed in the next step.

### 1.3.2 Model Specification

Through this step, a relevant problem solving model is evolved and all the details about its construction are described. The datasets are processed to a form suitable for the problem solving model, post which, all the available models are evaluated iteratively based on their performance.

1. Data Preprocessing - The process comprises several steps; performing exploratory data analysis, cleaning the dataset, splitting into training, test and validation sets. Each of the steps is detailed below;
    - Exploratory data analysis - The dataset can be visualized using computer aided tools and programming to find the underlying patterns and distributions of data. This step helps to identify prominent trends in the data and exposes various issues which might exist in the data.
    - Cleaning the dataset - Data analysis is performed to tackle missing values in the data, remove abnormalities or outliers (data samples which are quite different from the rest of the samples).
    - Training set - The training set is a set of example input-output pairs fed to the machine to create a learning model. The training set is used to 'train' a machine learning model in order to help the machine learn patterns from the data and make decisions on new data.
    - Validation set - A sample of data held back from the training phase is called the validation set. The validation set is fed as an input to the trained model. The decisions of the trained model on these inputs are validated against their corresponding outputs, to evaluate the performance of the model. This is an iterative process, since the hyperparameters (configurable variables internal to the model) are continuously changed ('tuning') to ensure the best achievable performance for a model ('optimization'), with a given dataset. It is important to validate every model before selecting a final model.
    - Test set - The test set is independent from the training set and serves as the new data given to the trained and validated model. In brief, the decisions a machine learning model makes on the test set, comprises the solution to the defined problem.

2. Model Construction - Once the pre-processed training, validation and test sets are obtained, various existing machine learning models are considered, selected based on their applicability to the collected datasets ('Model Selection') and developed ('Model Building'). These two topics are detailed as follows;
    - Model Selection - The models to consider are narrowed down based on; the identification of problem, supervised or unsupervised, and the inferences drawn about the distribution and patterns in the data from exploratory data analysis. If it is a supervised learning problem, various classification and regression models like Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), Random Forests (RF), Support Vector Machines (SVMs), Neural Networks (NNs), Linear Regression, Logistic Regression etc., are used. If it is an unsupervised learning problem, various clustering models like K-Means, Hierarchical Clustering etc., are used. Further, in classification models, if the underlying distribution of training data forms a clear boundary between the classes or categories of input data, basic classifiers like LDA, QDA, RF etc., can be used.
    - Model Building - Essentially, a machine learning model, makes decisions about input data by forming a function between example inputs and outputs considering some factors ('parameters'). This function differs from model to model. The decisions made on input data are compared with the true outputs, by a learner system, in turn adjusting the parameters in the model until there is a fully optimized model.
3. Evaluation - The performance of the constructed model is evaluated using various metrics like classification accuracy, confusion matrix, area under the receiver operating characteristic curve (roc) etc., for classification models and mean absolute error, mean squared error etc., for regression models.

This pipeline is iterative. Based on the evaluation results, if the model doesn't perform satisfactorily, the next step is to revert to the model specifications part, define a new model and perform all other steps associated with it.

A detailed account of the machine learning pipeline has been given in the above section 1.3. The objective defined in section 1.1 is translated into a problem statement and solved using the machine learning pipeline. Each of the steps, are applied to this specific problem and described in detail in the following sections.

# 2. PROBLEM SPECIFICATION

This section discusses in detail, all the steps pertaining to the specification of a problem in a machine learning pipeline, applied to the case of detecting microlensing events.

## 2.1 Problem Statement

It can be inferred from section 1.1 and 1.2 that, in order to detect microlensing events by analyzing all the light curves in the great volume of data generated by astronomical surveys, creating an automated system, with no manual intervention, is mandatory. It was also mentioned that machine learning is the most effective way to establish an automated system when dealing with a large dataset. By binding all these ideas together, we arrive at the problem statement; building an automated system to detect microlensing events.

This problem statement is translated into a machine learning problem in the following sub-section.

## 2.2 Problem Identification

To find a machine learning problem suitable to achieve the goal of detecting microlensing events in a pool of many other types of stars (‘variable stars’), certain challenges need to be understood.

Complicating the search for microlensing, there are a lot of other stars which show a continuous change in their brightness. According to [2], it could happen either due to internal factors like the star shrinking and swelling periodically (‘Pulsating Variables’) or the light from it to the Earth getting blocked due to some external factors like having an orbiting partner eclipsing it (‘Eclipsing Binaries’). Some of the variable stars present in the pulsating variables group are;

- Cepheids - Young, massive and luminous stars.
- RR Lyrae stars - Less luminous than cepheids & have shorter periods.

- Delta Scuti stars - Much fainter & have much shorter periods.
- Mira (*"Mira"*, Latin, adj.) - Characterized by very red colors and have periods longer than 100 days.
- Slow irregular variable stars - Have little or no detectable periodicity, therefore being similar to microlensing events and causing a difficulty in distinction.

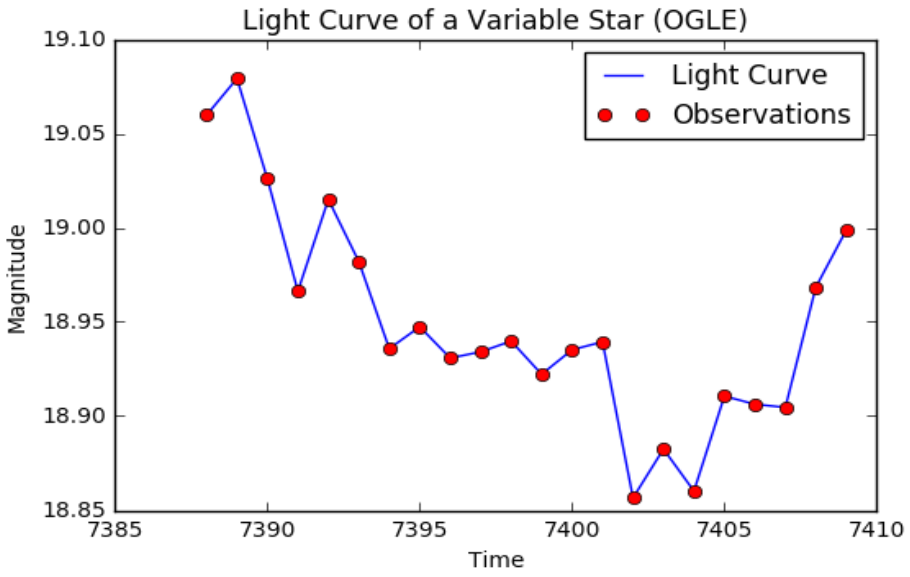


Figure 3: Light curve of a variable star

The example light curves of one such slow irregular variable star (in figure 3 above) and a microlensing event (in figure 4 below) demonstrate the difficulty in distinguishing them.

A microlensing event is characterized by a magnification of brightness for a brief period, only once in its lifetime and symmetric about its peak brightness value. The astronomical convention of a drop in magnitude corresponding to an increase in brightness can be observed in figure 4 below. Similarity is observed in the variable star displayed, as there is a magnification of

brightness for a brief period (between 7400-7405 on x-axis), after which it follows its usual pattern of brightness.

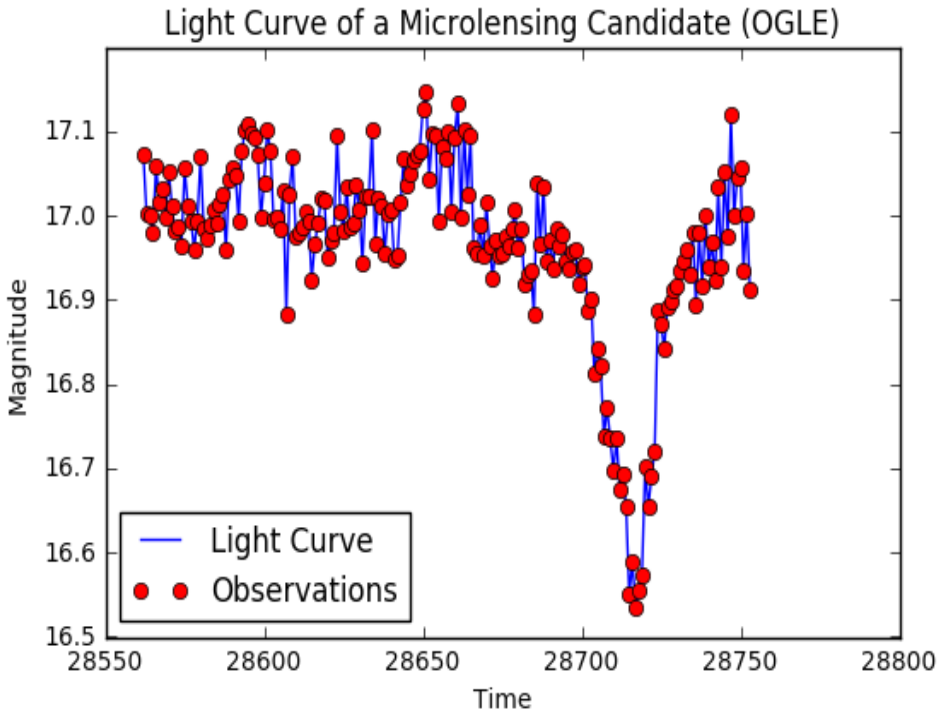


Figure 4: Light curve of a microlensing event

Therefore, given a representation of both variable stars and microlensing, the goal is to separate out microlensing events from all other types of variability. This clearly aligns with a classification problem. The formulated problem statement can now be translated into a machine learning problem as; automatic classification of microlensing candidates.

Given a classification problem, we need to collect datasets with representations of both variable stars and microlensing. Details about the collection of datasets is discussed in the following section.

### 2.3 Data Collection

In order to build an automatic classifier, datasets containing a representation of both variable stars and microlensing are chosen.

The training set [7,8] (for which we know the class, microlensing or variable star) consists of 1534 variable stars like RR Lyrae, Cepheids, MIRA, Delta Scuti, Eclipsing Binaries etc., (described in section 2.2) and 212 confirmed microlensing candidates from the first two operating seasons of an astronomical survey, Optical Gravitational Lensing Experiment (OGLE). The test set [9] (for which we don't know the classes), consists of 3194 stars from the first data release of Gaia. The number of observations for each sample is quite uneven, as mentioned in section 1.2; range of 12 to 49 for variable star light curve samples, 116 to 280 for microlensing light curve samples, 4 to 90 for Gaia light curve samples. It was ensured that aperiodic stars (like slow irregular variable stars described in section 2.2) are represented in the training dataset. About 76 such samples were included.

OGLE was chosen as the training dataset since this survey was dedicated to look for microlensing events and it has a rich database of variable stars as well as microlensing events. Gaia was chosen as the test dataset, since we intend to test it on a dataset with a huge number of samples to evaluate the scaling capability of our models to huge datasets. An additional dataset [17], MACHO (an astronomical survey looking for galactic dark matter in the form of Massive Compact Halo Objects, machos) was also used as a validation dataset for microlensing events since it has a significant number of observations (about 250-535 for each sample) compared to that in Gaia (4-90 for each sample). About 13 confirmed microlensing events were used from the survey data in the 'Large Magellanic Cloud'. This also serves the purpose of evaluating the scaling capability of our models.

Therefore, in line with data collection in the machine learning pipeline, the samples are light curves of stars, features are magnitude values in the light curve and labels are microlensing

and variable star. These labels are numerically represented as ‘0’ for variable stars and ‘1’ for microlensing.

This completes the problem specification part of the pipeline. The subsequent model specification step is discussed in the following section.

# 3. MODEL SPECIFICATION - A

A model (arbitrarily named 'A') is evolved, for solving the problem introduced in section 2.2, by using the resources mentioned in section 2.3. Based on the evaluation results, it is decided whether to use this model or iteratively start by evolving a new one.

The scope of this section includes application of each step in the model specification part of the general machine learning pipeline to the problem at hand.

## 3.1 Data Pre-Processing

The first step in data preprocessing, exploratory data analysis, is performed as follows;

### 3.1.1 Exploratory Data Analysis

Among the three data components, samples, features and labels, data patterns and distributions are mainly drawn from features. As specified in the previous section, features are the magnitude values or observations in the light curve. Section 2.3 mentions greatly varying number of features between datasets as well as within each dataset.

But, the input space on which supervised classifiers operate, typically contains fixed length input vectors. So, the number of features ('dimensions') need to be brought to a fixed length, in order to be fed to the classifiers. This can be done in two ways; use domain knowledge of the data to create features which can separate the two classes well ('feature engineering'), bring each of the samples to a fixed length ('interpolation'). Feature engineering is attempted and evaluated first.

Microlensing event is defined to be a single, symmetric, positive excursion from the baseline brightness level. From this definition, each of the five characteristics is mathematically formulated to create five features.[1] The features thus created, are evaluated using certain data

[1] Reproduced from Data from Paragraphs 3 & 4 in Section 4.2 (adapted). Vasily Belokurov et al. Light-curve classification in massive variability surveys — I. Microlensing. *MNRAS* (2003) 341 (4): 1373-1384. By permission of Oxford University Press on behalf of the Royal Astronomical Society. Available online at: https://academic.oup.com/mnras/article/341/4/1373/1039118?searchresult=1

visualization techniques to observe their ability to separate the two classes, variable stars and microlensing.

Feature Engineering

1. Distinction from baseline noise

The baseline noise is continuous and the microlensing peak significantly stands out from it. Traditionally, to detect these type of significant peaks present in noise, an autocorrelation function ('acf') is used. The correlation of magnitude values with delayed copies of themselves is called autocorrelation. The maximum value of this autocorrelation function, where the energy of the signal is concentrated, is the point of perfect match of the signal under consideration.

$$\text{acf} = r_k = \sum_{n=-\infty}^{\infty} Y[n] * Y[n-k]$$

$$\text{maxacf} = \text{maximum(acf)}$$

Here, the time delay is k and Y is a set of 'm' magnitudes. Y[n] is simply the nth sample. Thus, the 'maxacf' feature is computed.

2. Excursions

Excursion is the deviation of a signal from its usual path of values. The microlensing event is observed as an excursion from the baseline noise. The logarithm of ratio of the mean of measurements above the median and below the median is calculated to measure the excursion. This feature helps us determine whether the excursion is positive or negative. In the case of microlensing, we expect the excursion to be positive as the increase in brightness is measured as a decrease in magnitude (astronomical convention), resulting in a log argument greater than 1.

$$excursions = \log\left(\frac{mean(A_{med})}{mean(B_{med})}\right)$$

Here, $A_{med}$ is the set of observations (magnitude values) lying above the median of the observations and $B_{med}$ is the set of observations (magnitude values) lying below the median of the observations.

3. Symmetry

When the source star and lens star move out of alignment, the brightness of the source star gradually decreases and goes back to its normal path of values. This pattern of decrease in brightness is in symmetry with the increase in brightness as the stars move into alignment. Traditionally, to detect the similarity in two signals, a cross-correlation function ('ccf') is used. The correlation of the lightcurve with a time reversed copy of itself serves as the ccf in this case. The similarity is observed as a peak (maximum value) in the ccf value since when two positive peaks or two negative peaks align, they make a positive contribution to the integral.

$$\text{ccf} = \sum_{n=-\infty}^{\infty} Y[n] \ * \ Y[-n+k]$$

$$ccf \ = \ maximum(ccf)$$

Here, the time delay is k and Y is a set of 'm' magnitudes. Y[-n+k] is the time-reversed measurement of Y[n].

4. Periodicity

Microlensing has a single occurrence, as it is nearly impossible for the same stars to perfectly align, more than once. Whereas, all the variable stars exhibit a specific periodicity. The mean frequency calculated with the power spectrum as a weighting function is measured as periodicity ('omega_bar') [10].

$$Omega\_bar = \overline{\omega} = \left(1 - \frac{0.5}{1+\exp(-period)}\right)$$

In order to calculate the periodicity, most of the existing methods assume even sampling. So, we use the Lomb-Scargle Periodogram [11], which helps compute periods ('period' in the formula above) for unevenly sampled data by doing a least squares fit of a

single sinusoid to the data. The best fit occurs only at the correct periods. To find these correct periods, the periodogram uses a brute force grid search optimizer which searches over a grid of period values specified by the user. This grid should ideally cover all expected periodicities in the data. The period range is set to (0.2, 1.2) since the expected periodicities of almost all the variable stars are bounded by these values. A detailed explanation of the method is documented in [17].

5. Timescale of the event

Microlensing events usually do not last very long (last upto a maximum of few weeks), so we want to filter out the patterns that span a considerable time period (typical variable stars). Width of the autocorrelation function ('ACF_Width') measures the timescale of an event and is judged by its standard deviation. A narrow acf generally implies a broad signal spectrum and vice versa as shown in figure 5 below;

$$ACF\_Width = \sigma_{mag}\left(\sum_{n=-\infty}^{\infty} Y[n] * Y[n-k]\right)$$

Here, $\sigma_{mag}$ is the standard deviation of the acf at every sample 'mag' (magnitudes), and is called the width of the acf. The time delay is k and Y is a set of 'm' magnitudes. Y[n] is simply the nth sample.

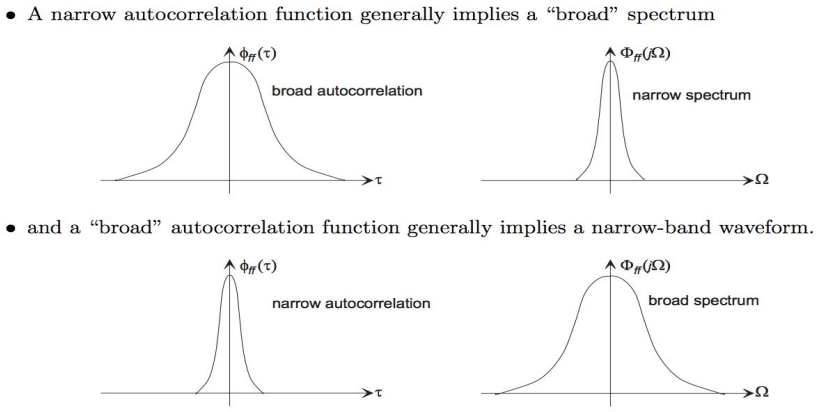


Figure 5: Width of autocorrelation function [18]

The goal of exploratory data analysis is to discover the underlying patterns or distributions in data using data visualization techniques. The scatterplots for some of the feature pairs (feature names, as designated above) calculated on the training set (depicted in figures 6-10 below), establish this.

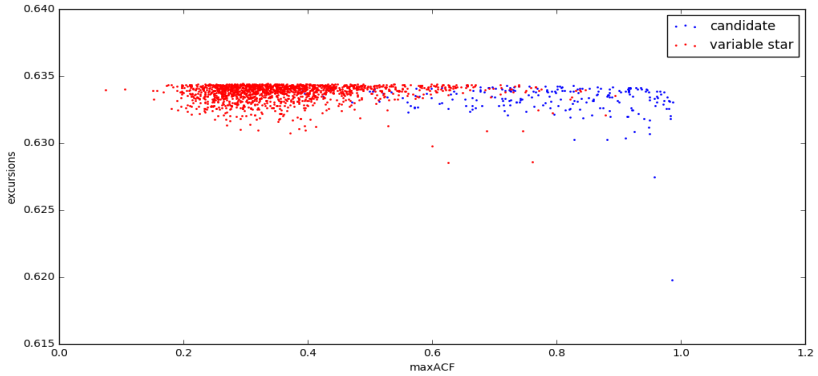


Figure 6: Scatter plot of maxACF vs Excursions

Figure 6 depicts the distribution of values for the features ‘distinction from baseline noise’ and the ‘excursion’.

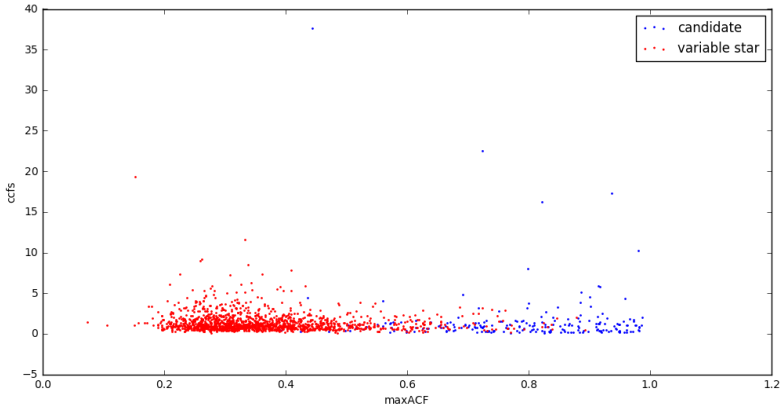


Figure 7: Scatter plot of maxACF vs ccf

Figure 7 depicts the distribution of values for the features ‘distinction from baseline noise’ and the ‘symmetry’.

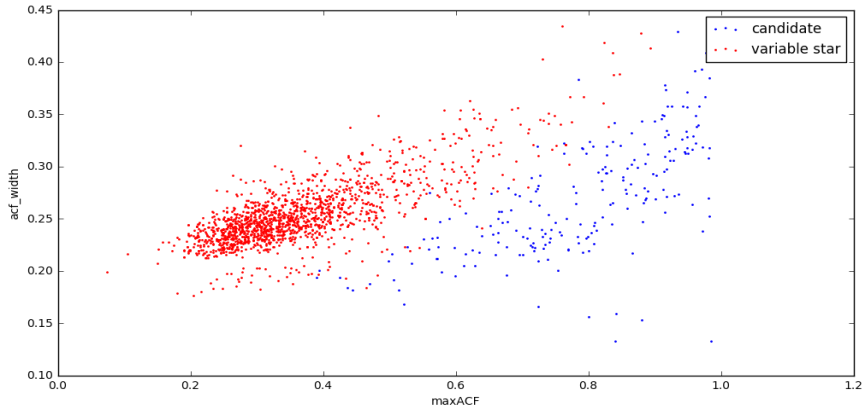


Figure 8: Scatter plot of maxACF vs acf_width

Figure 8 depicts the distribution of values for the features ‘distinction from baseline noise’ and the ‘timescale of the event’.

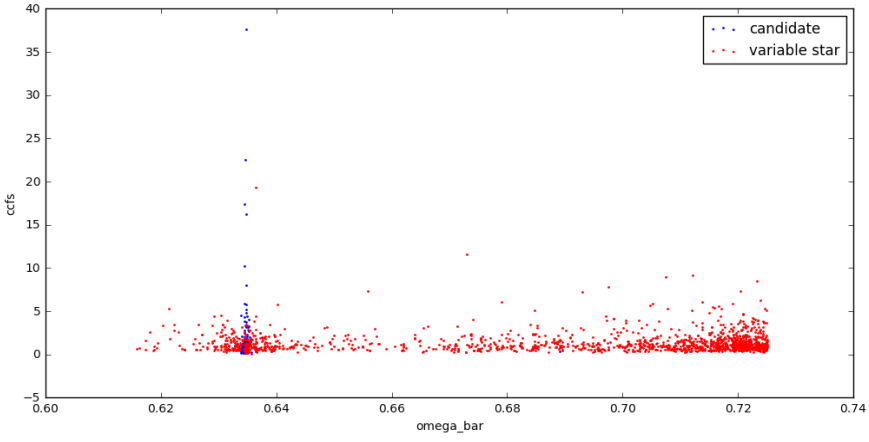


Figure 9: Scatter plot of omega_bar vs ccf

Figure 9 depicts the distribution of values for the features ‘periodicity’ and the ‘timescale of the event’.

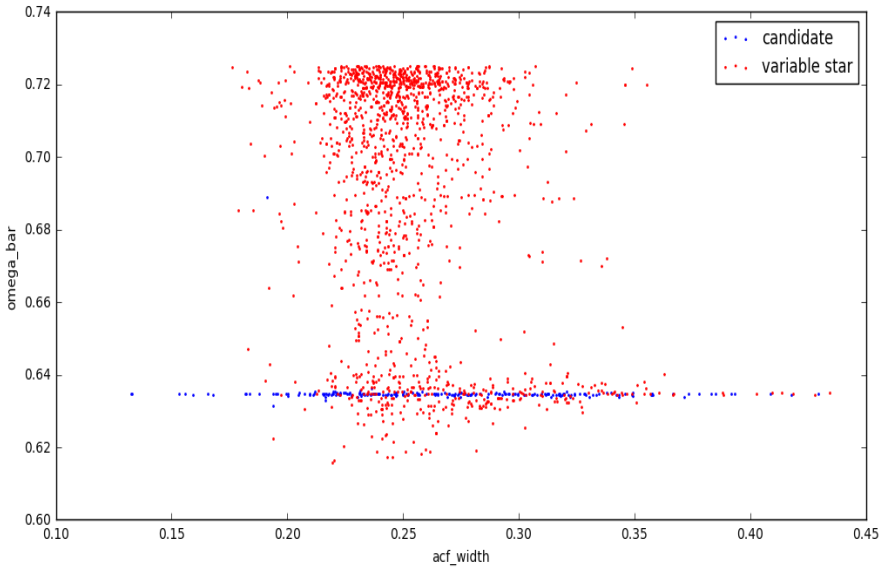


Figure 10: Scatter plot of acf_width vs omega_bar

Figure 10 depicts the distribution of values for the features ‘timescale of the event’ and the ‘periodicity’.

It can be inferred from these plots that some of the combinations of these features already partially separate the boundaries of microlensing and variable star classes (for example, maxACF vs acf_width) and some show regularities (for example, omega_bar vs acf_width), where the variable stars have different periodicities but bounded timescales, in contrast with microlensing candidates. The true separating efficiency of these features can only be determined in the evaluation stage (section 3.3). Hence, we proceed with the next step in data preprocessing, cleaning the dataset.

### 3.1.2 Data Cleaning

Any erroneous values like NaN (not-a-number type in the data), would come up in the

exploratory data analysis. But, nothing like that was found, so the next preprocessing step, splitting the dataset into training, validation and test set is performed.

### 3.1.3 Training, Validation and Test Data

Feature engineering is applied to the microlensing and variable star samples from the OGLE and Gaia datasets. All five features are computed for each sample in the OGLE dataset and are used as the training and validation datasets. And those computed for each sample in the Gaia dataset are used as the test dataset.

The validation dataset is created using a k-fold cross-validation technique as follows; the data set is split into k equal parts and the kth part is used as the validation dataset while the remaining k-1 parts are used as the training dataset. This is repeated k times (folds) such that the classifier is trained on all the observations, leaving one part out each time. This is done in order to generalize over the data. The choice for 'k' is not fixed and is at the discretion of the user. A large value of 'k' results in large computation time. The divisor of the dataset is commonly used as the value of k. Hence, in this case, k is chosen as 5.

This completes the step of data preprocessing and the next step in model specification, 'model construction' is discussed in the following section.

## 3.2 Model Construction

With the training, validation and test datasets obtained in the previous section, machine learning models applicable to the problem are identified ('model selection'), the working of each of the identified models is discussed in detail and their applicability to this specific problem is explained ('model building').

### 3.2.1 Model Selection

As specified in section 1.3.2, model selection is done based on the identification of problem

(supervised or unsupervised) and the inferences drawn from exploratory data analysis. The problem was identified to be supervised, specifically, a classification problem.

It was also mentioned that, if a clear separation of classes (variable stars and microlensing, red vs blue) is observed in the distribution of data, models like LDA, Logistic Regression, QDA, RF etc., can be used, due to their relationship with the distribution of data. In brief, while LDA and Logistic Regression generate a linear separating boundary between class distributions, QDA and RF generate complex separating boundaries between class distributions. The features represented in figures 6-8 seem to linearly separate the classes, whereas the ones in figures 9,10 do not show a linear separation. Each of these models and their applicability is studied in detail in the following section.

### 3.2.2 Model Building

In this section, each of the four machine learning models selected; LDA, Logistic Regression, QDA and RF are discussed.

#### 3.2.2.1 Linear Discriminant Analysis

The working and application of LDA are discussed separately as follows [13];

##### 3.2.2.1.1 Working

Discriminant analysis is a technique used to determine the set of independent features which help in distinguishing the classes. This analysis technique uses a discriminant function in finding the features which are independent. Discriminant function is a functional mapping to obtain a linear combination of features to discriminate between the classes well.

LDA comprises a linear discriminant function and has a linear separating boundary between the classes (in blue and red) as shown in the figure 11 below. LDA makes some assumptions about the data; the data comes from a Gaussian distribution and each class has the same covariance.

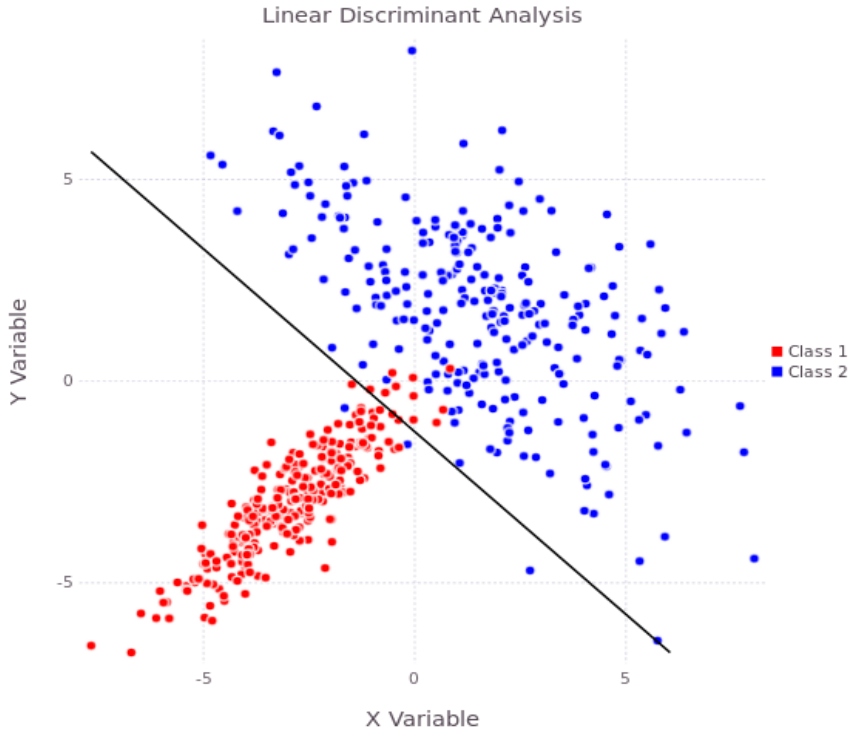


Figure 11: LDA boundary (black solid line)[13]

With the above mentioned assumptions, LDA performs the following steps;

- It computes the mean $(\mu_k)$ for each of the classes in the data.
- The covariance $(\Sigma_k)$ for each class in the data is also computed. LDA assumes $\Sigma_k = \Sigma$
- It makes predictions on the data by estimating the probability of each new input sample belonging to a certain class.
- The class probabilities are estimated using Bayes' Theorem;

$$\arg\max_k P\left(\frac{K=k}{X=x}\right) = \frac{\pi_k f_k(x)}{\sum_i (\pi_i f_i(x))}$$

- The class probability which maximizes the discriminant function below is predicted as the corresponding class for the input.

$$\arg\max_k \frac{\pi_k f_k(x)}{\sum_i \pi_i f_i(x)} = \arg\max_k \left(\log\left(f_k(x)\right) + \log\left(\pi_k\right)\right)$$

- Now, we assume the discriminant function to be;

$$\delta_k(x) = \log\left(f_k(x)\right) + \log\left(\pi_k\right)$$

- Substituting the assumptions of LDA in the discriminant function above;

$$\delta_k(x) = -\mu^T{}_k \Sigma^{-1} x + \tfrac{1}{2}\mu_k \Sigma^{-1} \mu_k + \ln\left(\pi_k\right)$$

where,

prior probability, $\pi_k = \frac{n_k}{n}$,

estimated probability of *'x'* belonging to class 'k' based on the first assumption of LDA,

$f_k(x) = \frac{\exp\left(\frac{-1}{2}(x-\mu_k)^T \Sigma^{-1}{}_k (x-\mu_k)\right)}{(2\pi)^{p/2}\left|\Sigma_k\right|^{1/2}}$,

number of samples of class 'k' in the training data, $n_k$,

total number of samples in the training data, $n$,

features in the training data, $x$

predicted class, $k$.

As the definition and working of LDA have been established, its applicability to the problem at hand is discussed.

**3.2.2.1.2 Application**

Each of the above mathematical steps is integrated into a 'Python' programming package, 'sklearn.discriminant_analysis.LinearDiscriminantAnalysis' .

Model Fitting

This step trains the LDA, in brief, helps it to learn the mapping between existing input-output pairs (training data). It accepts as inputs (parameter 'X' in the function), the 5 - feature training dataset and their corresponding labels (parameter 'y' in the function) and follows the working of LDA.

Model Predictions

This step tests the LDA, in brief, given the validation and test datasets (new inputs), it makes

predictions about their labels by examining their class probabilities (detailed in 'working' above).

An evaluation of these predictions will be discussed in section 3.3.

### 3.2.2.2 Quadratic Discriminant Analysis

The working and application of QDA are discussed separately as follows;

#### 3.2.2.2.1 Working

QDA comprises a quadratic discriminant function, unlike LDA, and has a quadratic separating boundary between the classes (as shown in figure 12 below, in blue and red). QDA makes almost similar assumptions about the data; the data comes from a Gaussian distribution but each class uses its own estimate of covariance. With these assumptions, QDA also performs the same steps as LDA. The only difference is that the class probabilities maximize a quadratic discriminant function below [4];

$$\delta_k(x) = -\frac{1}{2}\log\left|\Sigma_k\right| - \frac{1}{2}(x-\mu_k)^T\Sigma^{-1}{}_k(x-\mu_k) + \log(\pi_k)$$

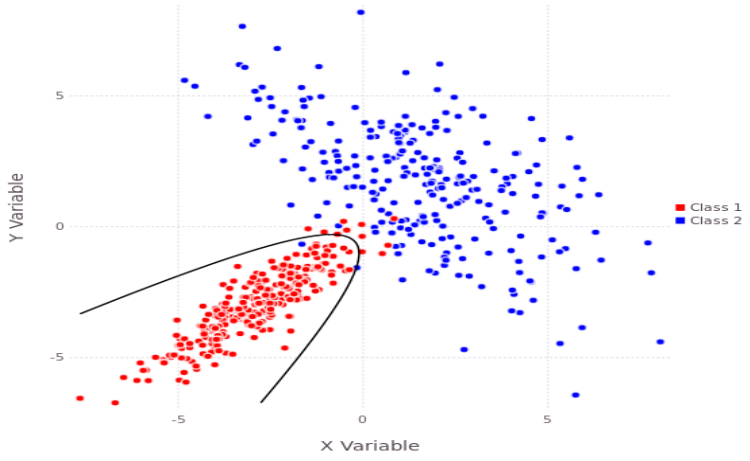


Figure 12: QDA boundary (black solid line)[13]

#### 3.2.2.2.2 Application

Each of the above mathematical steps is integrated into a 'Python' programming package, 'sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis'.

Model Fitting

This step trains the QDA, in brief, helps it to learn the mapping between existing input-output pairs (training data). It accepts as inputs (parameter 'X' in the function), the 5 - feature training dataset and their corresponding labels (parameter 'y' in the function) and follows the working of QDA.

Model Predictions

This step tests the QDA, in brief, given the validation and test datasets (new inputs), it makes predictions about their labels by examining their class probabilities (detailed in 'working' above). An evaluation of these predictions will be discussed in section 3.3.

### 3.2.2.3 Logistic Regression

The working and application of Logistic Regression are discussed separately as follows [12,18];

The function which maps the inputs to outputs while using logistic regression is called a logistic function and is derived in the 'working' section. A logistic or sigmoid function applied on an input 'x' is as follows [3];

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

The function is s-shaped and is depicted in figure 13 below. The function always maps the inputs to a range of outputs (0,1).

#### 3.2.2.3.1 Working

Given a set of two features $(x_1, x_2)$ separable by a linear boundary, the equation of the linear boundary would be $\beta_0 + \beta_1 x_1 + \beta_2 x_2$. Plugging in the values of a point (a,b) in the boundary equation would give $g = \beta_0 + \beta_1 a + \beta_2 b$. If (a,b) lies in the region where the class label is '1', the boundary equation would lie in the region $(0,\infty)$. If (a,b) lies in the region where the class label is '0', 'g' would lie in the region $(-\infty,0)$. And if (a,b) lies on the boundary then the class

label is undecided. Let P(X) denote the probability of occurrence of an event X (belonging to class '1'). The ratio of the probability of an event happening, to not happening is called the odds ratio, $\frac{P(X)}{1-P(X)}$ and needs to be mapped to the region (-∞,∞). Hence, we take a logarithm of the odds ratio (log-odds). Odds ratio = $e^g$, since 'g' is the logarithm of the odds ratio. Thus, the logistic function, P(X) = $\frac{e^g}{1+e^g}$ is obtained.

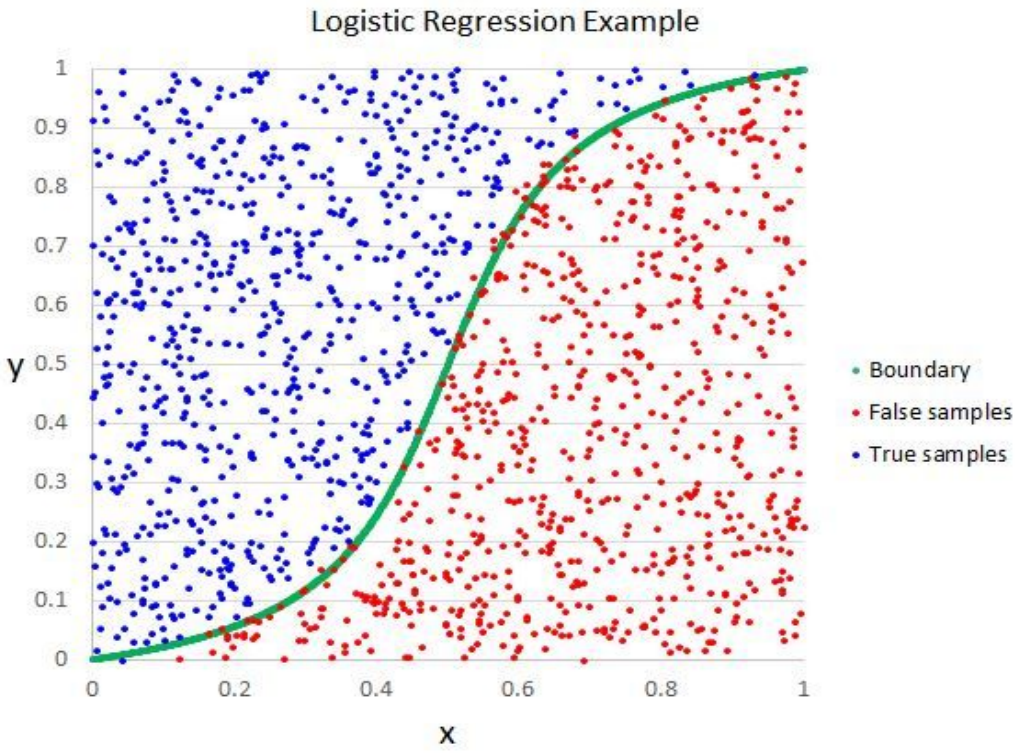


Figure 13: Logistic regression boundary (green solid line) [16]

The most important steps in logistic regression are;

- The logistic regression model makes a prediction about the probability of the input belonging to class '0'. If this probability exceeds 0.5, the output is a prediction for the default class (class '0'), otherwise it is for class '1'.

- The estimation of coefficients ( $\beta$ ) is often the most important step in logistic regression. These coefficients are used as weights to the input features ('x').
- Averaging P(X) over the entire training dataset will give the likelihood of a new data point belonging to the class '1'. The aim of logistic regression is to maximize this likelihood.
- Maximum likelihood estimation is used to obtain the best estimate of these coefficients. This is done by taking a derivative of the likelihood function described above and equating it 0.
- The estimation is an iterative process, minimizing the cost function while training and selecting the coefficients.
- The cost function is a sum of the squared errors between the predicted and true outputs. Gradient descent minimizes the cost function.
- Gradient descent computes the derivative of the cost function with every update of coefficients until a global minima of the cost function is reached.

**3.2.2.3.2 Application**

Each of the above mathematical steps is integrated into a 'Python' programming package, called scikit-learn. Specifically, 'sklearn.linear_model.LogisticRegression' is the function with all the above mathematical steps programmed.

Model Fitting

Since there are five input features, the logistic function would be;

$$f(x) = \frac{e^g}{1+e^g}$$

where $g = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5$ ,

$\beta_{0-5}$ represent the regression coefficients,

$x_{1-5}$ represent the five input features computed

This function is used to map the inputs to outputs (variable star and microlensing, class '0' and class '1').

Model Predictions

The predictions are made by evaluating the probability of an input belonging to class ‘0’.

#### 3.2.2.4 Random Forests

The working and application of Random Forests are discussed separately as follows;

##### 3.2.2.4.1 Working

Random forests, simply put, are an aggregation of decision trees. Decision tree is a supervised learning model, in which predictions are done based on decision rules on features (like setting a threshold value in case of numerical features). In brief, every decision tree has nodes, each of which is a decision rule. These decision rules are decided (deciding the threshold value, for example) based on their ability to split the classes well. Each node could have several daughter nodes under it. For example, if the main node has a decision rule for variable ‘x’ as $x<5$, possible daughter nodes would have decision rules like, $x>2$, $x<3$ etc.

In order to make predictions, each new sample is sent to the root node (first node in the decision tree, superset of all decision rules), and passed through the corresponding daughter nodes until it reaches a leaf node (last node, class label). Several such decision trees combined together make a random forest. Here, the new input sample is sent to each of the decision trees in the forest and the mode of the results from leaf node is selected as the final class.

It consists of the following steps;

- A subset of features ‘i’ from the total number of input features ‘j’ is randomly selected such that $i << j$.
- Calculate the node which can be the best split point.
- Split that node into daughter nodes using best split method.
- Repeat the above 3 steps until we are left with the leaf node (‘output’).
- One iteration of the above steps creates a ‘tree’. Create ‘n’ such random trees to create a random forest

  The best split is the value of a feature for which the cost is minimum. This completes the

creation of the random forest. The next step is to make predictions. It consists of the following steps;

- Pass the input features in the test dataset to each random tree, predict the outcome and save it.
- For each sample, calculate the number of votes for each class.
- The class with maximum votes is considered as the final prediction.

**3.2.2.4.2 Application**

Each of the above steps is integrated into a 'Python' programming package, called scikit-learn. Specifically, 'sklearn.ensemble.RandomForestClassifier' is the function with all the above steps programmed.

Model Fitting

Since this is a classification problem, 'gini index' is used as the cost function. It calculates the purity of the groups created by the best split point. A '0' gini index indicates perfect purity i.e., class samples are separated into exactly two groups (binary classification).

Model Predictions

The predictions on the test set are made by using the steps described in the predictions part of the random forest working.

Since all the classifiers have been defined and the application of training, test and validation datasets generated in section 3.1.3 is discussed, all of them are evaluated in the following section.

## 3.3 Evaluation

Cross-validation is a method which evaluates the efficiency of a built classification model on real data. The quality of each of these classifiers using 5-fold cross validation (discussed in section 3.1.3) is assessed. This is repeated 5 times (folds) such that the classifier is trained on all the

observations, leaving one part out each time. The results of all 5 folds are averaged to produce a single error estimate. In this way the error rate on the training data is determined.

The metrics evolved to evaluate the results of these four classifiers on the selected datasets are; total error rate, false alarm rate and missed detection rate. Each of these is defined, after which their values for model - A are specified.

Total error rate: The total proportion of wrong predictions in the dataset.

$$\text{Total Error Rate} = \frac{\sum|true\ labels - predictions|}{Total\ number\ of\ samples}$$

A true positive (tp) is an outcome where the model correctly predicts the positive class (label = '1'). Similarly, a true negative (tn) is an outcome where the model correctly predicts the negative class (label = '0'). A false positive (fp) is an outcome where the model incorrectly predicts the positive class. And a false negative (fn) is an outcome where the model incorrectly predicts the negative class.

False alarm rate: It represents the misclassification rate of the negative class. It is a ratio of number of false positives and sum of false positives and true negatives. It is mathematically defined as;

$$\text{False alarm rate} = \frac{fp}{fp+tn}$$

Missed detection rate: It represents the misclassification rate of the positive class. It is a ratio of number of false negatives and sum of false negatives and true positives. It is mathematically defined as;

$$\text{Missed detection rate} = \frac{fn}{fn+tp}$$

| | Training error | Total error rate | False alarm rate | Missed detection rate |
|---|---|---|---|---|
| LDA | 0.98669937 | 0.01487352 | 0.08559219 | 0.00529373 |
| QDA | 0.9742566 | 0.02688989 | 0.17571749 | 0.00266934 |
| Logistic Regression | 0.95451891 | 0.04632174 | 0.11251501 | 0.03944792 |
| Random Forests | 0.99914184 | 0.01601474 | 0.05743934 | 0.01113635 |

Table 1: Evaluation metrics for model - A

An important observation about the error rates, QDA is always expected to perform better than LDA since LDA is just a special case of QDA. This is observed in the training dataset but not particularly followed in the validation dataset, which indicates that it overfits on the training data. Additionally, we can't afford any missed detections of microlensing because the event is very rare (1 in 400,000 in the bulge, as reported by the MACHO project [1]). But, since the total error rates and missed detection rates were low, this model was applied on the test dataset.

In order to make predictions on the test set, each of the classification algorithms makes its predictions. First, for each light curve in the test set, we calculate its "score" or the number of classifiers that labelled it as a potential microlensing candidate. In an ideal case, we would see only 4's and 0's. The former corresponds to the case when all algorithms say that a given sample is a microlensing candidate, the latter to the case when all agree that it is not. As we might expect, there are also some 1's, 2's and 3's - samples for which different algorithms give us different predictions. We choose to focus only on the "strong candidates", i.e., samples which have been unanimously declared as microlensing candidates, to be very sure of the predictions. Using this method, 149/3194 candidates were classified as strong microlensing events. The histogram in figure 14 below, shows a detailed breakdown of these predictions.

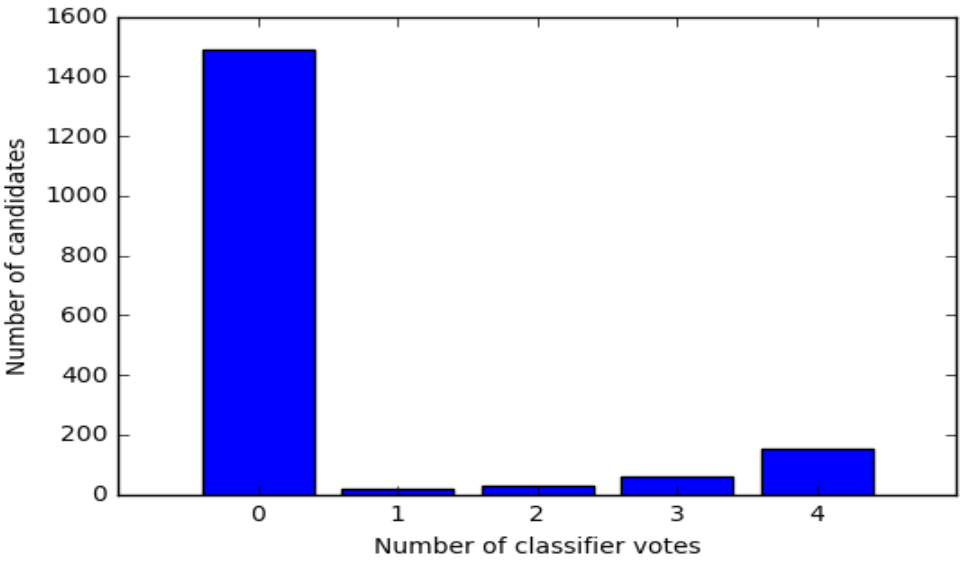


Figure 14: Classifier votes for candidates

The results are unsatisfactory, as we expect to see very few microlensing events in the whole dataset. In addition to that, to establish an automated classifier and perform better on unseen data (which is our intended use case- LSST, Gaia), it is important to avoid manually extracting features (as done in model A) and wave fitting; evaluating the fit of parameters relevant to a microlensing event, to light curves from the test set. Hence, this method was discarded in favor of a model - B described in the next section.

# 4. MODEL SPECIFICATION - B

Based on the evaluation results for model - A and the drawbacks mentioned, it is evident that a new model (arbitrarily named 'B') needs to be evolved. All the steps starting from data preprocessing to evaluation are performed again. Based on the evaluation results, it is decided whether to use this model or iteratively start by evolving another new one.

The scope of this section includes application of each step in the model specification part of the general machine learning pipeline to the problem at hand.

## 4.1. Data Pre-Processing

Each of the parts comprised; exploratory data analysis, data cleaning and data splitting specific to this model are discussed in detail.

### 4.1.1 Exploratory Data Analysis

Section 3.1.1 mentioned the need for fixed length input vectors and the two methods to achieve it. The first method, feature engineering, was performed and evaluated as a part of model A. In model B, the second method, interpolation, is performed and evaluated.

Interpolation is the method of inserting new data points into a range of discrete data points specified. Before directly performing interpolation on all the datasets, it is important to analyze the consequences of applying classifier models to this data. Hypothetically, interpolation is performed on the variable star and microlensing data from the training dataset specified in section 2.3. Of all the magnitude observations in a typical microlensing sample, only a few (part where the exact microlensing event occurs) actually relate to microlensing. The rest of the observations are a part of the baseline noise. As mentioned in section 3.2.2, classifiers learn the underlying class distribution of data. Therefore, given a set of microlensing samples, the classifiers learn an incorrect distribution of data. In order to avoid this, observations pertaining to

only the microlensing event are selected to represent the microlensing class, after which each of the samples is interpolated. The method to do this is explained below and the result is depicted in figure 15 below.

- A similarity measure (cross-correlation) is performed between, a reference, confirmed microlensing light curve (from OGLE) and microlensing samples in the training data set.
- We examine the peak locations (cross-correlation peak means maximum similarity as discussed in section 3.1.1). We then note the peaks before and after the microlensing peak (peak with minimum magnitude). The part of the light curve lying between these two peaks depicts the phenomena of microlensing, as shown in figure 14 below.
- In order to find the peaks, a peak detection function is used. It performs a continuous wavelet transform by convolving a ricker wavelet with each microlensing sample using various width sizes, to which the wavelet is stretched. Ricker wavelet is modeled as the following function; $A\left(1-\frac{x^2}{a^2}\right)\exp\left(-\frac{x^2}{a^2}\right)$, where $A=\frac{2}{(\sqrt{3a})\pi^{0.25}}$ and $a$ is the width size, $x$ is the input wave to the function.
- We need to choose the range of these widths such that they cover the expected widths of the peaks we want to detect. Therefore, tuning of peak widths in the peak detection function, is essential to get precise locations for the beginning and end of a microlensing event. In this specific case, a width range of 13-20 was found to be optimal in detecting the positive peaks accurately and a width range of 10-50 was found to be optimal in the case of negative peaks.
- These specific parts of the microlensing samples are all extracted to form the class of microlensing.
- The variable star samples, microlensing samples after cross-correlation (training dataset), and the samples in the test dataset are all linearly interpolated to contain the number of observations equal to the maximum number of observations present in their class samples (microlensing or variable stars).

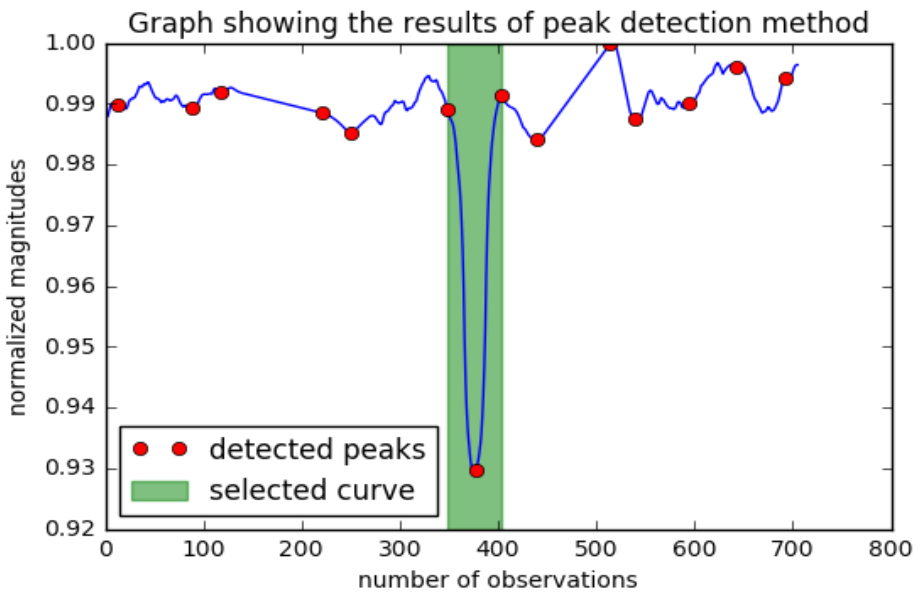


Figure 15: The part of curve selected by peak detection method

The goal of exploratory data analysis is to discover the underlying patterns or distributions in data using data visualization techniques. The mean vs variance plot for the data is depicted in figure 16 below.

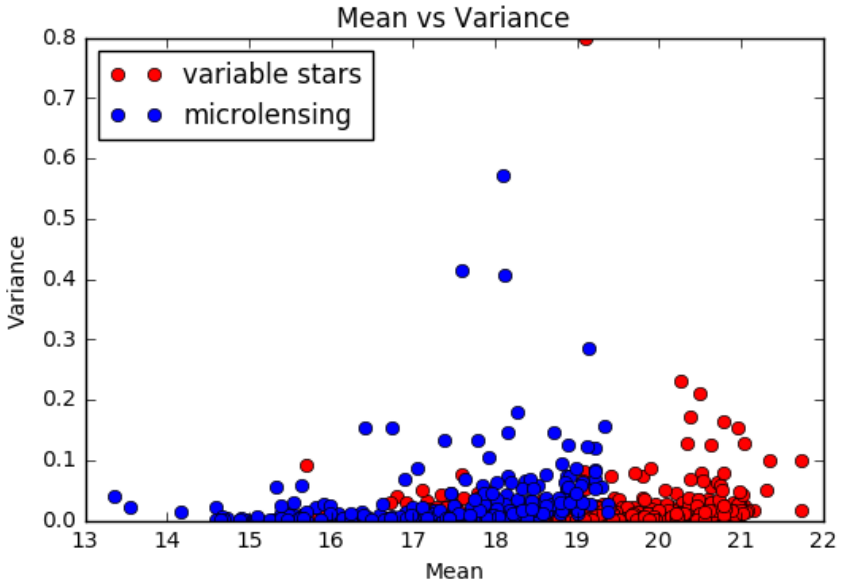


Figure 16: Mean vs Variance plot to examine class decision boundary

It can be inferred from this plot that the distributions of these datasets clearly separate the boundaries of microlensing and variable star classes. Hence, we proceed with the next step in data preprocessing, cleaning the dataset.

### 4.1.2 Data Cleaning

As there were no erroneous samples observed in the training dataset, there is no need for any data cleaning to be performed.

### 4.1.3 Training, Validation and Test Data

Interpolated magnitude observations for each sample in the OGLE dataset are used as the training and validation datasets. The microlensing class samples after cross-correlation are interpolated and used in the training dataset. And interpolated magnitude observations for each sample in the Gaia dataset are used as the test dataset.

The validation dataset was created as follows; A fraction of light curves each from variable stars and confirmed microlensing candidates (from the datasets mentioned above), were selected such that the number of resulting time segments for each class is balanced. In order to generate these time segments, a segmentation pipeline is used. It uses the sliding window technique to choose fixed length blocks of overlapping data, extracts time segments of these light curves and labels each one of them based on their class, variable star or microlensing candidate. The number of samples and time segments in training and validation for each of the classes are mentioned towards the end of section 4.1.3.

A block length of 48 with an overlap of 10 is chosen, based on classifier results for each length upto the number of observations in the variable stars class (since it is the lower of the two classes and increasing it beyond that would leave variable stars with no data for the remaining length). Thus, a training dataset for the baseline classifier was created using those labelled time segments. Similarly, using the remaining light curves, a validation dataset was created. This

method ensures that samples of both variable stars and microlensing candidates are contained in the training and validation datasets and there is no overlap of light curves between the two datasets.

The datasets are further preprocessed to organize them into a form required for a classifier operating on fixed length sequences. In particular, all the samples are shuffled to avoid any learning of order of data, each sample in the newly created training and validation dataset has 48 observations (equal to the chosen block length), labeled as 49 features (48 magnitude observations and 1 class label), for the classifier to learn. The class label feature is excluded from the validation dataset when given as input to the classifier, since it has to predict the labels on its own and the class label feature is used to evaluate the accuracy of those predictions. The details of the training and validation are;

| | Microlensing samples | Microlensing time segments | Variable stars samples | Variable stars time segments |
|---|---|---|---|---|
| Training | 30 | 720 | 767 | 767 |
| Validation | 31 | 744 | 767 | 767 |

Table 2: Training and validation dataset compositions

### 4.2 Model Construction

With the training, validation and test datasets obtained in the previous section, machine learning models applicable to the problem are identified (‘model selection’), the working of each of the identified models is discussed in detail and their applicability to this specific problem is explained (‘model building’).

### 4.2.1 Model Selection

As mentioned in section 3.2.1, in the presence of a clear separating boundary between the two classes, baseline classifiers like QDA can be used. The results from exploratory data analysis present a class distribution which seems to be separable by a quadratic separation boundary (intuitively). Hence, QDA is applied to the selected training, validation and test datasets.

### 4.2.2 Model Building

Since the relevant literature and application of QDA is clearly discussed in section 3.2.2, the model is directly applied to the training and validation data described in section 4.1.3. The built model is evaluated in the following section.

## 4.3 Evaluation

The metrics evolved to evaluate the results of QDA on the selected datasets are; false alarm rate, missed detection rate (defined in section 3.3) accuracy score and confusion matrix. Accuracy score and Confusion matrix are defined, after which the values of all the metrics for model - B are specified.

Accuracy score: It is a ratio of number of correct classifications (given label = predicted label) and total number of samples being classified. It is mathematically defined as;

$$accuracy\,(y,\hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} I_{(y_i=\hat{y})}$$

where, $I_{a=b}$ is an indicator function, takes the value of 1 only when $a$ is equal to $b$, otherwise 0.

Confusion matrix: It helps to evaluate the performance of a classifier in supervised learning problems. Each row of the matrix represents the samples in an actual class while each column represents the instances in a predicted class. The components of a confusion matrix are; true positive, true negative, false positive and false negative.

We achieve an accuracy score of 96.89% , false alarm rate of 1.04% (percentage of variable stars misclassified as microlensing) and missed detection rate of 5.24% (percentage of microlensing candidates misclassified as variable stars). The confusion matrix for this classification problem is as follows;

| | **Predicted Labels** | | |
|---|---|---|---|
| **Actual Labels** | | **Variable Stars** | **Microlensing** |
| | **Variable Stars** | 759 | 8 |
| | **Microlensing** | 39 | 705 |

Table 3: Confusion matrix of QDA

Upon inspecting the microlensing time segments incorrectly detected as variable star time segments, it was found that they belonged to 27 microlensing samples, out of a total of 31 samples in the validation dataset. This is a major disadvantage since the classifier is unable to detect most of the microlensing events. We can't afford any missed detections of microlensing because the event is very rare (1 in 400,000 in the bulge, as reported by the MACHO project [1]), as mentioned previously.

Another notable disadvantage was that any data set that needs to be passed through this classifier needs to have a fixed length, which indicates the need to interpolate. But, this could greatly disrupt the quality of data. In some cases, there could be a change of shape of the light curve itself by adding too many observations and stretching them out (interpolation) or by discarding important observations. We cannot afford to change the shape by attempting to interpolate it to a longer timescale since microlensing events don't usually last for longer than a month. And discarding observations could stray us away from very valuable discoveries as it

could contain exoplanet information, which would only be a small blip in the microlensing light curve.

Considering the above inferences, it is clear that another model to resolve the above mentioned issues, needs to be evolved. This is discussed in detail in the following section.

# 5. MODEL SPECIFICATION - C

A model (named ‘C’ arbitrarily) is evolved to address the issues in models A and B. All the steps pertaining to model specification in the machine learning pipeline are performed again.

## 5.1 Data Pre-Processing

All the steps starting from exploratory data analysis are performed again and described in the following sections.

### 5.1.1 Exploratory Data Analysis

Feature engineering and interpolation were proved unsuitable for this problem in sections 3.3 and 4.3 respectively. Therefore, in this model, data is used in its original form, without any preprocessing. The inputs are variable length (length of a sample equal to the number of observations present in it). Hence, the next step, data cleaning is performed.

### 5.1.2 Data Cleaning

There were no erroneous samples observed in the dataset. Hence, the next step, creation of training, validation and test data is performed.

### 5.1.3 Training, Validation and Test Data

The OGLE datasets mentioned in section 2.3, are used in their original form as the training and validation datasets. The Gaia dataset is used in its original form as the test dataset. Two cases are mentioned in the creation of training and validation datasets;

- Case-1: 30% of light curves from variable stars data and 50% of light curves from the confirmed microlensing events data (from the training dataset mentioned in section 2.3), were selected in order to achieve some balance between the training samples in two

classes, variable stars and microlensing. Here, each of these light curves is labelled based on their class, variable star or microlensing candidate. Thus, a training dataset was created using these labelled light curves. Similarly, using the remaining light curves from the samples of microlensing and 15% of remaining light curves from the samples of variable stars , a validation dataset was created.

- Case-2: In order to evaluate its accuracy for a highly imbalanced dataset, 80% of the samples each from training examples of variable stars and microlensing are used. Using the remaining 20% light curves in each class, a validation dataset was created.

The resulting data set compositions are as follows;

| | | Microlensing samples | Variable stars samples |
|---|---|---|---|
| Case-1 | Training | 112 | 534 |
| | Validation | 100 | 150 |
| Case-2 | Training | 169 | 1227 |
| | Validation | 43 | 307 |

Table 4: Dataset compositions

The generated training, validation and test datasets are used in the construction of a model, discussed in the following section.

### 5.2 Model Construction

With the training, validation and test datasets obtained in the previous section, machine learning models applicable to the problem are identified (‘model selection’), the working of each of the identified models is discussed in detail and their applicability to this specific problem is explained (‘model building’).

### 5.2.1 Model Selection

The model selected should be able to make predictions on variable length inputs. The most popular model which can achieve this is a Recurrent Neural Network (RNN). This model, applied to the datasets generated in section 5.1.3, is discussed in detail in the following section.

### 5.2.2 Model Building

This section will briefly describe the working of neural networks, followed by an improved version, recurrent neural networks and its application to the problem at hand. It is important to understand the elements and working of a neural network since they are the building blocks of RNNs as well.

#### 5.2.2.1 Neural Networks

It is a type of a machine learning model, which mimics the working of human brain in its operation and is used to recognize patterns in data. While all the models described in the previous sections map given inputs to outputs based on specific functions; neural network creates a function to map any input to any output. It is therefore called, Universal Approximator. It learns the correct function to relate all such known input-output pairs, given a training dataset.

The most basic element in a neural network is a node and is depicted in figure 17 below. A node is simply where computations happen. Several nodes combine to form a layer. Several such layers combine to form a network. The nodes in a neural network are like neurons in the human brain. Hence the name, neural network.

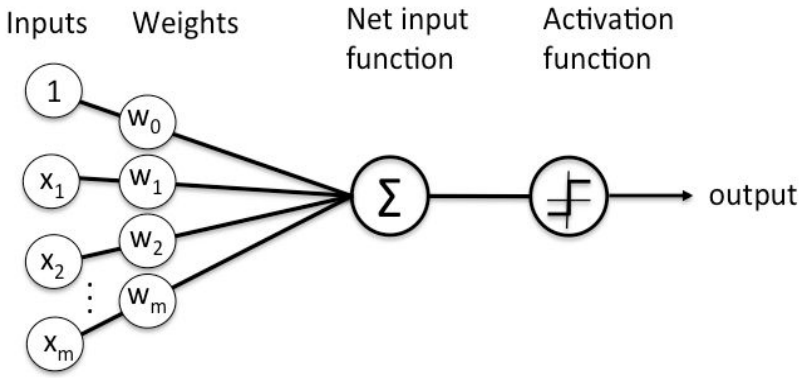


Figure 17: Elements in a node of the neural network [14]

#### 5.2.2.1.1 Working

At every node, the input data is combined with some coefficients ('weights'), which either amplify or dampen that particular input. This, simply put, is assigning significance to an input in solving a particular problem. A sum of the product of these combinations of inputs and weights is passed through the activation function. The activation function decides the extent of progress of this input (through the network) to affect the final output (like class prediction in a classification problem).

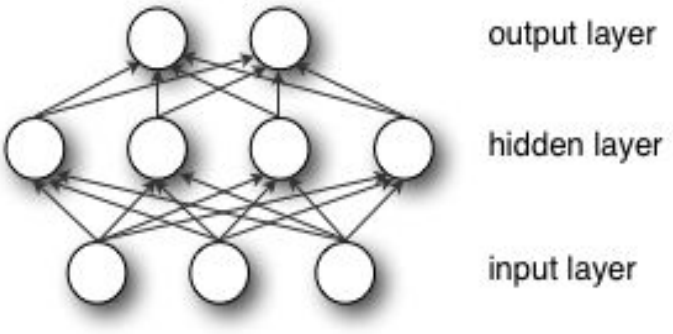


Figure 18: Basic structure of a neural network [14]

The structure of a neural network is depicted in figure 18 above. The inputs are fed to the input layer of a neural network. Based on the significance assigned to each of those inputs, they are either turned on or off. The results of this layer are passed to the next layer, hidden layer, where the same process is repeated and its results are given to the final output layer, where we obtain the class predictions as outputs (in case of a classification problem). Based on the error in classification, the weights are adjusted and the process is done again. It can be mathematically defined in three steps [14];

- Scoring the input: Input * Weight = Guess (Based on activation function)
- Calculating the associated loss: Truth - Guess = Error (Based on loss function)
- Updating the model: Error * Weight's contribution to error = Adjustment of weights (feedback)

This type of network is called a feed-forward network and act only on the current inputs and don't consider their past inputs to perceive the future inputs.

A neural network chooses initial weights at random and updates them by using gradient descent as discussed in section 3.2.2. The amount of change these coefficients should have in each update is decided by the step size (or learning rate). The learning rate is selected by using various optimization algorithms like Root Mean Squared Propagation (RMSprop) etc., which calculates the learning rate adaptively, dividing the learning rate by an average of squared gradients in each update.

#### 5.2.2.2 Recurrent Neural Network

Recurrent networks, another type of neural networks, remember the decision they made on the previous input fed to make a decision on the input at the next time step. This is very helpful in the case of sequential inputs like time series inputs where each of the inputs form a sequence and are related to each other.

##### 5.2.2.2.1 Working

They are neural networks with memory. A recurrent neural network is depicted in figure 19.

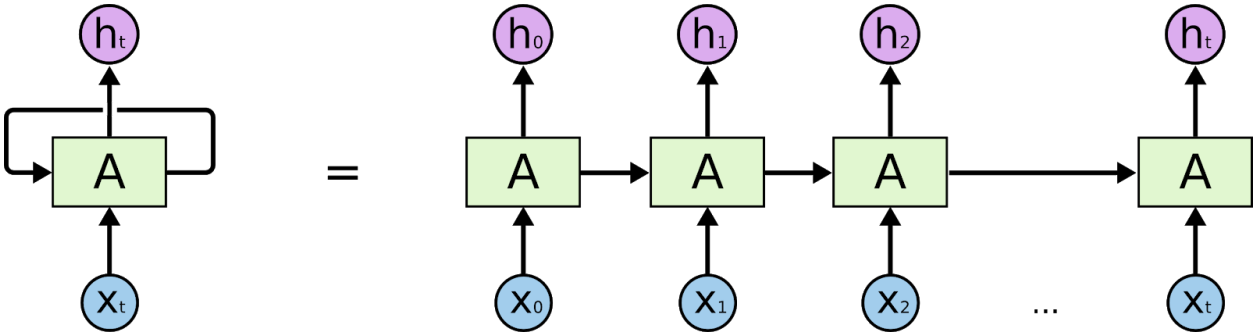


Figure 19: Recurrent Neural Network [15]

In the figure 19 depicted above, '$X_t$' is the input fed to a chunk of neural network 'A' (usually having a single layer), to give the output '$h_t$'. The loop around 'A' indicates the ability of the network to pass information from one time step to the next, as depicted in the right hand side.

This memory capability of recurrent neural networks reduces with increase in time i.e., they can't memorize the inputs for long periods, which could be an issue in the case of sparse inputs spread over a long time period. In order to address this issue, a class of RNNs called Long Short Term Memory Networks (LSTMs) were evolved.

#### 5.2.2.2.2 Applicability

The sequential learning of RNNs is very helpful for the problem at hand since, each sample in the datasets under consideration forms a time series. But, astronomical data could be sparse, depending on the sampling rate of the survey documenting the data. So, choosing an RNN which doesn't have the capability to perform well on sparse data, wouldn't be ideal. Because, the problem identified is to build an automatic classifier to detect microlensing candidates in any given dataset. The objective is to build a model which can scale well to a dataset having a high number of samples and a high number of observations in each sample. Hence, a model which can

scale well is required.

#### 5.2.2.3 Long Short Term Memory Networks

They are a special class of RNNs. Remembering information for long periods of time is the default behaviour of LSTMs.

##### 5.2.2.3.1 Working

As mentioned in the case of an RNN, the repeating neural network in each time step ('A'), usually consists of a single layer. Whereas, in the case of an LSTM, it consists of four layers interacting in a very different way. The core functionality of an LSTM is established by the 'cell state', to which information is added and removed using gates in the LSTM. This process is discussed in detail below through figure 20 [5].

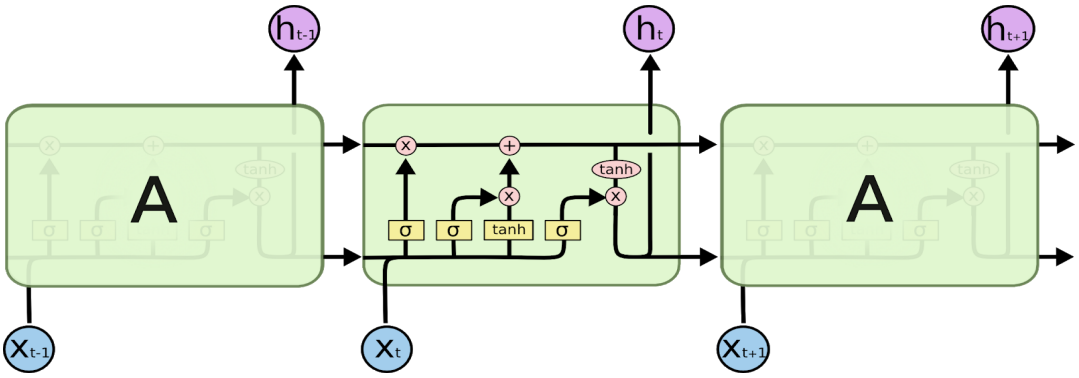


Figure 20: LSTM [15]

The four yellow boxes in a network indicate various layers in an LSTM. These layers decide which information needs to be evicted, stored in the cell state (the horizontal line at the

top carrying state information from one network to the other) and select the outputs accordingly. The first sigmoid layer (first yellow box) is the ‘forget gate’ and decides which information needs to be evicted from the cell state. The second sigmoid layer, the ‘input gate’ decides which weights we will update and the next tanh layer creates the new weights to be used. The ‘+’,‘x’ and ‘+’ operations are all used to update the cell state (forget and store information). The final sigmoid layer decides which parts of the cell state we will output. The ‘tanh’ in pink merely pushes the values to be in the range of -1 to +1 and is multiplied with the output of the sigmoid layer to create the final output and send to the next network in the loop.

Although LSTMs essentially use variable length inputs to generate class predictions (‘0’ or ‘1’ for binary classification), the inputs need to have a fixed length. Therefore, all the variable length inputs are padded with zeros to the left of the first observation in each sample, such that their length equals the maximum length of the sequences present. This is fed to the RNN along with a mask, containing ones only where the data is present and the padded values masked as zeros. The RNN simply copies the previous state as the current state whenever it encounters a ‘0’ in the mask i.e., it doesn’t affect the learning process. Whereas, whenever it encounters a ‘1’ in the mask, it memorizes the input and changes the state, i.e., it affects the learning process. In this way, LSTMs handle variable length inputs.

#### 5.2.2.3.2 Application

This is the most ideal model for the problem at hand because it can handle sparse time series data very well, very easily. It is applied to the problem at hand using the given resources. The training, validation and test datasets generated in 5.1.3 are padded with zeros as mentioned in the previous paragraph and then, given as an input to the LSTM model.

This LSTM model is used with the following hyperparameters (configurable variables as defined in section 1.3.2) crucial in training any neural network; activation function, loss function and dropout. The hyperparameters applicable to this problem are discussed;

- Sigmoid Activation Function: An activation function is responsible for generating the mapping function between inputs and outputs, and maintaining the universal approximation characteristic of the neural network. Hence, they need to be non-linear in nature. 'Sigmoid' is chosen as the activation function for the problem at hand. The activation function of a sigmoid is inversely proportional to $1 + e^{-x}$ [6]. The non-linear nature of the sigmoid function helps the output to stay in the range (0,1), which is our desired range of outputs.
- Binary Cross-Entropy Loss Function: A loss function computes the cost associated with every decision regarding the output. Some of the loss functions used are binary cross-entropy, squared error, euclidean distance. Cross-entropy loss is defined as;

  $$H(p,q) = \Sigma\, p(x)\, log\, q(x)$$

  Here, 'p' is the true label and 'q' is the predicted value from the model used. Binary cross-entropy is used when 'p' has only two possible labels, in short, for binary classification problems, such as the problem at hand.
- Optimization algorithm: RMSprop algorithm is one of the basic adaptive learning techniques, so this is used first as the optimization algorithm. If the results are not satisfactory, other optimization algorithms need to be implemented.
- Dropout: Adding dropout probability 'p' as a hyperparameter in a neural network means that nodes, are kept with a probability of 'p' during the training phase of a model. The nodes are chosen randomly during each epoch (iteration). A co-dependency is developed amongst nodes during training, resulting in 'over-fitting', which curbs a model's ability to perform with equal ability (as performed on training data) on unseen test data. Dropout helps to eliminate over-fitting. A dropout of 0.007 is finalized after tuning to several other values.

All the above parameters are carefully tuned to arrive at the best classification model. This completes the construction of model 'C'. An evaluation for both the cases (mentioned in section 5.1.3) applied to the built LSTM model, is discussed in the following section.

### 5.3 Evaluation

The metrics chosen to evaluate model 'C' are accuracy score and confusion matrix (described in section 4.3). The parameter values selected, training, validation and test split details, accuracy score and confusion matrix are tabulated below;

| | |
|---|---|
| **Training**<br>**[Variable Stars: Microlensing] 0:1** | **534:112** |
| **Validation**<br>**[Variable Stars: Microlensing] 0:1** | **150:100** |
| **No. of Nodes**<br>**Input-Hidden-Output** | **1-40-2** |
| **Batch Size** | **10** |
| **No. of Epochs** | **3** |
| **Dropout** | **0.1** |
| **Validation Accuracy** | **~99.2** |

Table 5: Parameters used in LSTM for case-1

| | **Predicted Labels** | | |
|---|---|---|---|
| **Actual Labels** | | **Variable Stars** | **Microlensing** |
| | **Variable Stars** | 148 | 2 |
| | **Microlensing** | 0 | 100 |

Table 6: Results for case-1

| | |
|---|---|
| **Training**<br>**[Variable Stars: Microlensing] 0:1** | **1227:169** |
| **Validation**<br>**[Variable Stars: Microlensing] 0:1** | **307:43** |
| **No. of Neurons**<br>**Input-Hidden-Output** | **1-40-2** |
| **Batch Size** | **10** |
| **No. of Epochs** | **3** |
| **Dropout** | **0.007** |
| **Validation Accuracy** | **~99.7** |

Table 7: Parameters used in LSTM for case-2

| | **Predicted Labels** | | |
|---|---|---|---|
| **Actual Labels** | | **Variable Stars** | **Microlensing** |
| | **Variable Stars** | 306 | 1 |
| | **Microlensing** | 0 | 43 |

Table 8: Results for case-2

It is clearly indicated that the classifier never misses the detection of microlensing events in the validation dataset (even when there is a high class imbalance), which is of primary interest to us.

Several attempts were made to apply models ‘B’ and ‘C’ to the Gaia dataset, but given the low number of recorded observations for each star, it is difficult to confirm an event. Interpolating the Gaia dataset was observed to stretch out the curve and entirely change the

structure of almost all the light curves since there are very few observations, so this is not a reliable pre-processing method.

Since the application of these models to the test dataset wasn't possible, another dataset with a high number of observations for each sample (MACHO), was considered. The details of this dataset are mentioned in section 2.3. This was used as an additional validation dataset in order to test if the built model can scale to datasets with a high number of observations per sample, with similar performance.

The training and validation datasets are created as follows; 80% of light curves from variable stars data and 100% of light curves from the confirmed microlensing events data (from the training dataset mentioned in section 2.3), were selected. Here, each of these light curves is labelled based on their class, variable star or microlensing candidate. Thus, a training dataset was created using these labelled light curves. Similarly, using the remaining 20% of light curves from the samples of variable stars and all the light curves from the samples of MACHO microlensing events, a validation dataset was created. The results are tabulated below;

| | |
|---|---|
| **Training**<br>**[Variable Stars: Microlensing] 0:1** | **1227:212** |
| **Validation**<br>**[Variable Stars: Microlensing] 0:1** | **307:13** |
| **No. of Neurons**<br>**Input-Hidden-Output** | **1-40-2** |
| **Batch Size** | **10** |
| **No. of Epochs** | **1** |
| **Dropout** | **0.007** |
| **Validation Accuracy** | **100** |

Table 9: Parameters used in LSTM for validation with MACHO

| | Predicted Labels | | |
|---|---|---|---|
| **Actual Labels** | | **Variable Stars** | **Microlensing** |
| | **Variable Stars** | 307 | 0 |
| | **Microlensing** | 0 | 13 |

Table 10: Results for MACHO

The built model ‘C’ was able to show similar performance on OGLE on MACHO datasets, even though the number of observations in each sample greatly increased. This proves that this model is the best of the three models built.

The built model ‘C’ serves as the perfectly automatic classifier to detect microlensing events with an accuracy of at least 99%, and 0% missed detection rate.

# 6. CONCLUSIONS

This built model ‘C’ can be used to detect exoplanets discovered through microlensing by providing it with confirmed light curve examples of the same and testing it on all the microlensing candidate light curves we have available. The fact that there are very limited number of exoplanets already detected through microlensing, makes it a challenging task to train the classifier.

By exploring more advanced pre-processing techniques, there is scope to apply it to datasets consisting of very less observations for each star, like Gaia, and obtaining reasonably accurate results.

## REFERENCES


1. The Planetary Society Blog. *Microlensing [Web log post]*. Retrieved on September 20, 2016 from http://www.planetary.org
2. Wikipedia, The Free Encyclopedia. *Variable Star*. Retrieved on May 15, 2018 from https://en.wikipedia.org
3. Raschka, Sebastian. *Machine Learning FAQ.* Retrieved from, www.sebastianraschka.com
4. The Pennsylvania State University. *Classification [Lecture Notes].* Retrieved on May 16, 2018 from https://newonlinecourses.science.psu.edu/stat857/node/80/
5. Skymind.ai. *A Beginner's Guide to LSTMs [Web log post]*. Retrieved from https://skymind.ai/wiki/lstm
6. Sharma, Avinash. (2017, Mar 30). *Understanding Activation Functions in Neural Networks [Web log post].* Retrieved from https://medium.com/the-theory-of-everything/
7. Pigulski A., Kolaczkowski Z., Kopacki G. (2003, Oct 20). *VI photometry of MM1 and MM7 OGLE fields.* Retrieved from http://cdsarc.u-strasbg.fr/viz-bin/Cat?J/AcA/53/27#sRM3.2
8. Udalski A. et al. (2006, Sep 21). *OGLE microlensing events in Galactic Bulge.* Retrieved from http://cdsarc.u-strasbg.fr/viz-bin/Cat?J/AcA/50/1#sRM3.1
9. Gaia Collaboration et al. (2016, Sep 14). *Gaia Data Release 1*. Retrieved from http://vizier.u-strasbg.fr/viz-bin/VizieR-3?-source=I/337/fov
10. Vasily Belokurov, N. Wyn Evans, Yann Le Du. (2003, June 1). *Light-curve classification in massive variability surveys -I*. Retrieved from https://doi.org/10.1046/j.1365-8711.2003.06512.x
11. VanderPlas, Jacob T. (2017, Mar 28). *Understanding the Lomb-Scargle Periodogram*. Retrieved from arXiv:1703.0984
12. Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. *Additive logistic regression: a statistical view of boosting*. The Annals of Statistics, 28(2):337407, 2000.

13. Thatcher, Tim. *Linear Discriminant Analysis*. Retrieved on (2018, May 17) from http://discriminantanalysis.readthedocs.io/en/latest/
14. Skymind.ai. *A Beginner's Guide to Neural Networks and Deep Learning [Web log post]*. Retrieved from https://skymind.ai/wiki/neural-network
15. Olah, Christopher. (2015, Aug 27). *Understanding LSTM Networks*. Retrieved from http://colah.github.io/posts/2015-08-Understanding-LSTMs/
16. Hussain, Mahboob. (2015, Oct 9). *About Logistic Regression [Web log post]*. Retrieved from http://mh-journal.blogspot.com/2015/10/about-logistic-regression.html
17. The MACHO Collaboration. (1997, Apr 10). *The Macho Project: 45 Candidate Microlensing Events From The First-year Galactic Bulge Data*. Retrieved from http://wwwmacho.anu.edu.au/Data/LMC/Lensing/
18. MIT OpenCourseWare. *The correlation functions (cont.) [Lecture Notes]*. Retrieved on May 17, 2018 from https://ocw.mit.edu/courses/mechanical-engineering/2-161-signal-processing-continuous-and-discrete-fall-2008/lecture-notes/lecture_22.pdf