Analysis of Kepler Objects of Interest using Machine Learning for Exoplanet Identification

Aditeya Baral
Department of Computer Science
PES University, Bangalore, India
aditeya.baral@gmail.com

Ameya Rajendra Bhamare
Department of Computer Science
PES University, Bangalore, India
ameyarb1804@gmail.com

Saarthak Agarwal
Department of Computer Science
PES University, Bangalore, India
saarthak1607@gmail.com

***Abstract*—For several decades, planet identification has only been performed by astronomical experts and researchers with the help of specialized equipment. With the advent of computational methods and access to satellite data from space missions, this trend has changed. For instance, NASA's Exoplanet Exploration [1] program has provided us with vast amounts of data on celestial objects to assist in space exploration. One such mission of interest is the Kepler mission. Over 4000 such transiting exoplanets have been discovered, identified and catalogued since the mission commenced in 2007. It's focus lay on exploring planets and planetary systems. It has provided us with an extensive database of discoveries that help in computing planet occurrence rates as a function of an object's parameters such as the size, insolation flux, star type and orbital period. This information is catalogued in the Cumulative Kepler Object of Information dataset available for public domain use on NASA's exoplanet archive. Four basic models have been compared. Namely, Support Vector Machines, Random Forest Classifiers, AdaBoost and Deep Neural Networks. The AdaBoost classifier was selected as the optimum machine learning model and returned an F-1 score of 98% on the dataset.**

***Index Terms*—Kepler, NASA, planetary systems, satellites, exoplanet discovery**

I. INTRODUCTION

For ages, astronomy has fuelled our inquisition and has made us question and ponder over the mystery surrounding our existence and role in this universe. It is natural human tendency to question if the celestial objects that we know of are planets or not considering the fact that we inhabit one - and may soon need to find another. This question has stimulated the minds of researchers for centuries and the task of identifying exoplanets, which are essentially planets apart from the ones we know, way beyond our solar system, and has led to novel discoveries. Until now, this has been a time-intensive and tiring task, reserved only for astronomical experts with substantial domain knowledge and included the use of specialized equipment. These experts study images collected by satellite based telescopes like the Hubble. With the launch of a whole new generation of modern astronomy and technologically advanced telescopes and satellites such as the Kepler, a new door has opened up for exoplanet exploration with the ultimate goal being to automate the analysis of scientific observations and data generation pertaining to exoplanet identification.

These satellites and telescopes apart from taking images, also process them using astronomical and mathematical techniques to produce data with a variety of features for identifying these exoplanets. This data has been made publicly available and democratized the once arduous task of exoplanet exploration among data scientists, engineers and statisticians alike. Modern intelligent algorithms and techniques such as Machine Learning and Deep Learning can be applied to exoplanet data to detect overlooked and misclassified exoplanets and objects of interests in data archives or automate the pipeline.

The models explored include Support Vector Machines, Random Forest Classification, AdaBoost and Deep Neural Networks. They were used to classify objects found in the Kepler Cumulative Object of Interest [2] (KCOI) table. The KCOI table contains 50 features recorded from Kepler data. This data has been preprocessed to retain the most important features, 19 to be specific. Finally, models have been fit on this data and eventually used to assign the probability for an object in the KCOI table being an exoplanet.

Traditional methods of discovery such as obtaining and analysing images of distant stars are now outdated. A digital transformation in astronomy is underway. As opposed to satellite based telescopes from the past, the primary function of these satellites is to record and process a variety of data, not just images. We now find various powerful and state-of-the-art Machine Learning techniques at our disposal to look for exoplanets. As this data has been made publicly available, this makes the task all the more easy.

The Kepler Space Telescope was launched by NASA in 2009. To date, it has been the most successful telescope to aid the discovery of exoplanets. It has identified several thousand objects of interest, with over 4300 of them confirmed exoplanets. The catalog has a high reliability rate, averaging 85%-90% over the radius plane. It continues to improve as follow-up observations continue. While Kepler has been officially decommissioned as of October 2018 as it ran out of fuel, the statistical data it has accumulated is capable of providing insights for new exoplanet discoveries for years to come.

II. BACKGROUND

A. NASA Exoplanet Program

The NASA Exoplanet Exploration Program (ExEP) undertakes missions to answer humankind's most timeless questions. These questions include the kind of planetary systems that

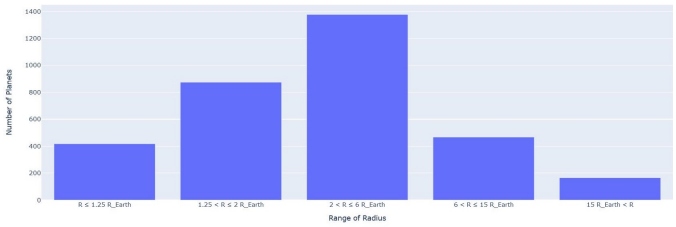

Fig. 1. Comparison of Exoplanets Discovered

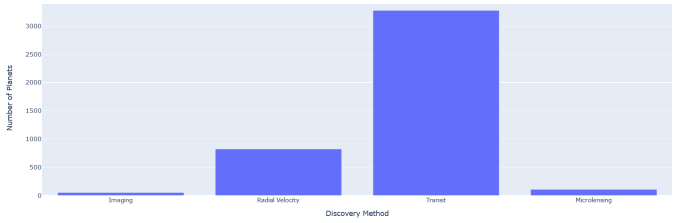

Fig. 2. Number of Planets by Discovery Method

orbit other stars in our galaxy, what exoplanets are like and whether humans are the only species to exist. The ultimate aim of these explorations is to be able to discover and characterize planets similar to Earth, search for habitable conditions on those planets, and reveal signatures of life. For instance, Kepler-22b was the first exoplanet to be ever considered to contain the necessary ingredients needed to support life.

B. The Kepler Mission

The Kepler Space Telescope launched in 2009, has been the most successful telescope to aid the discovery of exoplanets. The mission has been designed to survey a portion of the Milky Way galaxy and discovers hundreds of Earth-size and smaller planets in or near the habitable zone. It additionally determines the fraction of the billions of stars in our galaxy that might have their own solar system.

A dip in a star's brightness could indicate one of its planets is passing between a and the observing telescope. Transit time is the time it takes for the planet to pass between the star and an observing telescope. The magnitude of reduction in brightness coupled with transit time can provide clues to the relative size and position of the planet relative to its central star.

The collected data is periodically released and is hosted by CalTech under contract with NASA. The satellite was officially retired in October 2018 because it ran out of fuel. Years later, the statistical data that Kepler produced continues to produce new exoplanet discoveries.

C. Exoplanet Identification Techniques

Microlensing measures the bending of light as energy from a star passes a planet making it an indirect method of planet detection. It has the ability to detect the smallest and most distant of planets. The shift of a star resulting from the gravitational pull of the planets orbiting it is measured using the Radial Velocity. It is positively correlated to both, the mass and orbital period of a planet.

Direct imaging is an age old method used to identify exoplanets. It involves using extra-terrestrial telescopes to capture high resolution pictures of star fields. The pictures are analysed by scientists to determine if planets exist. While good for detecting stars, this method has proven to be inadequate for exoplanet identification. A solar eclipse occurs when the moon passes directly in front of the sun, blocking its light. This is quite similar to how the transit method finds exoplanets. When a planet passes between an observer and a star, it blocks some of that star's light. The star gets dimmer briefly. Though it is a small change in brightness, it hints to astronomers the presence of an exoplanet around a distant star.

Planetary transits furnish information which leads to an estimate of the object of interest's size, mass, speed and period of orbit and density of its star. Traditional methods like direct imaging and radial velocity techniques have been replaced. The reason being that they are biased towards the detection of large exoplanets. In contrast, Kepler's transit time data allows for the detection of smaller Earth-sized exoplanets. This has opened up a new window of planets to be discovered.

III. PREVIOUS WORK

Natalie M. Batalha, in her paper titled 'Exploring Exoplanet Populations with NASA's Kepler Mission' [3] reports on the progress made by Kepler in measuring the prevalence of exoplanets orbiting within 1 AU of their host stars. This strengthens NASA's long-term goal of finding habitable planets beyond our solar system. She talks about exoplanet confirmation and characterization and the requirements needed for a reliable exoplanet occurrence rate. Another paper titled 'Advances in Exoplanet Science from Kepler' [4] by Jack J. Lissauer et. al discusses the highlights of the Kepler mission. While the two papers discuss the highlights of the mission, they fail to talk about any exoplanet detection methods.

'Identifying Exoplanets with Deep Learning: A Five Planet Resonant Chain around Kepler-80 and an Eighth Planet around Kepler-90' [5] by Christopher J. Shallue and Andrew Vanderburg presents a method for classifying signals from potential planets using Deep Learning. Convolutional Neural Networks (CNNs) are used to predict if signals caused by either astrophysical or instrumental phenomena correspond to an exoplanet. The model was able to correctly classify plausible planet signals on the test set 98.8% of the time. Additionally, they validate two new exoplanets that are identified, with high confidence by their model. This paper explores Deep Learning approaches but does not compare its performance with other Machine Learning models.

Miguel Jara-Maldonado et. al, in their paper 'Transiting Exoplanet Discovery Using Machine Learning Techniques' [6] propose a model to create synthetic datasets of light curves. Several Machine Learning algorithms are used to identify transiting exoplanets. They conclude that multi-resolution analysis in the time-frequency domain improves exoplanet signal identification, due to the various characteristics of light curves

and transiting planet signals. A Gaussian process classifier reinforced by other models is used to perform probabilistic exoplanet validation by accounting for prior probabilities for possible false positives. This paper identifies the caveats against the use of single-method planet validation techniques and cautions against it.

'False Positive Probabilities for all Kepler Objects of Interest: 1284 newly validated and 428 likely false positives' [7] by Timothy Morton et. al, presents astrophysical false positive probability calculations for every recorded Kepler Object of Interest. It was the first large-scale demonstration of a fully automated transiting exoplanet validation procedure.

IV. PROPOSED SYSTEM

There are three primary categories of machine learning algorithms - regression, clustering, and classification. Classification is a supervised machine learning technique where observations are assigned a known label based upon their characteristics, often known as dependent variables. The classes can either be binary, or even multi-class. Classification is one of the major applications of machine learning algorithms and is used in everyday life to solve large scale real world problems across various domains. This study focuses on a binary classification of Objects of Interest as "FALSE POSITIVE" or "CONFIRMED" exoplanets. NASA uses the label of "FALSE POSITIVE" to indicate the satellite incorrectly tracked an object. We do not consider the observations labelled as "CANDIDATE" since these are yet to be labelled by NASA and hence, are unknown to us. For our analysis, we have used four models, each with its own unique characteristics to tackle the problem at hand from different angles. The four models used are Support Vector Machines (SVM), Random Forest, AdaBoost and a Feed-Forward Neural Network.

A. Support Vector Machine

Support Vector Machine is a machine learning algorithm which seeks to identify a hyperplane between classes and attempts to maximise the distance between the hyperplane and the opposing classes. It also possesses a property, known as the "kernel trick". It is able to tackle the issue of classes being non-linearly separable by mapping the input feature space to a higher dimensional feature space by performing various transformations. It then uses a maximal distance hyperplane in the transformed space to separate the classes.

B. Random Forest

The Random Forest classifier consists of a large number of individual Decision Trees that operate together. Each of these trees makes a class prediction and then averages them to obtain the final prediction. They utilize Bagging and randomized feature selection to build the ensemble model of Decision Trees. Bagging ensures only a portion of the dataset is utilized in any single Decision Tree in the ensemble. Randomized feature selection allows selection of only a random subset of the features from the dataset to construct individual Decision Trees as part of the ensemble. This improves the model as randomization decreases the effects of correlation and reduces overfitting.

C. AdaBoost

AdaBoost, short for "Adaptive Boosting" is an ensemble learning method. It uses an iterative approach to learn from the mistakes of weak classifiers. A weak classifier is one that performs better than random guessing, but performs poorly at classifying objects. A single classifier might not be able to accurately predict the class of an object. However, by combining multiple weak classifiers such that each one progressively learns from the others' wrongly classified objects, we can build a strong model. These classifiers could be any of the classifiers that we know of, Decision Trees being the default.

D. Neural Network

A Neural Network is an attempt to create and simulate the network of neurons inside a human brain. This enables computers to learn things and make decisions in a human like manner. These are genetically inspired algorithms developed to train machines to behave as though they are interconnected brain cells. A neural network is capable of generalising well and is known to be a Universal Function Approximater and can easily learn non-linear complex relationships between the dependent and independent variables.

E. Majority Voting Based Learning

To complement the already present algorithms, we also propose an ensemble of these algorithms by combining them together in a sequentially. The final predictions obtained will be a majority vote across its individual classifier predictions.

V. WORKFLOW

The Cumulative Kepler Object of Information dataset is not without faults. The dataset is a raw one, obtained directly from the readings measured by Kepler after assigning labels, and hence has not only missing values but a lot of unnecessary columns as well. Some of these can be dropped since they are ID attributes assigned but a majority of them are crucial to our analysis need to be filled in appropriately. This step helps us tune our model's performance and remove redundant features which do not add any information to our analysis.

A. Data Preprocessing

We begin by first observing the columns that have missing values. We observe that though quite a lot of these columns do contain missing values, most of these missing values correspond to the errors associated with attribute values.

We also observe that two error attributes are filled with missing values - the positive and negative errors associated with Equilibrium Temperature. Hence we decide to drop them completely, since their values cannot be imputed. For the remaining error attributes, we observe a highly skewed distribution of a few attributes. It would be ill-advised to replace these values with their average, and hence we decide to fill up their values with the median error value. This handles

Attribute	Number of Missing Values
kepler_name	7270
koi_depth	363
koi_depth_err1	454
koi_depth_err2	454
koi_duration_err1	454
koi_duration_err2	454
koi_impact	363
koi_impact_err1	454
koi_impact_err2	454
koi_insol	321
koi_insol_err1	321
koi_insol_err2	321
koi_kepmag	1
koi_model_snr	363
koi_period_err1	454
koi_period_err2	454
koi_prad	363
koi_prad_err1	363
koi_prad_err2	363
koi_score	1510
koi_slogg	363
koi_slogg_err1	468
koi_slogg_err2	468
koi_srad	363
koi_srad_err1	468
koi_srad_err2	468
koi_steff	363
koi_steff_err1	468
koi_steff_err2	483
koi_tce_delivname	346
koi_tce_plnt_num	346
koi_teq	363
koi_teq_err1	9564
koi_teq_err2	9564
koi_time0bk_err1	454
koi_time0bk_err2	454

TABLE I
COUNT OF MISSING VALUES IN ATTRIBUTES

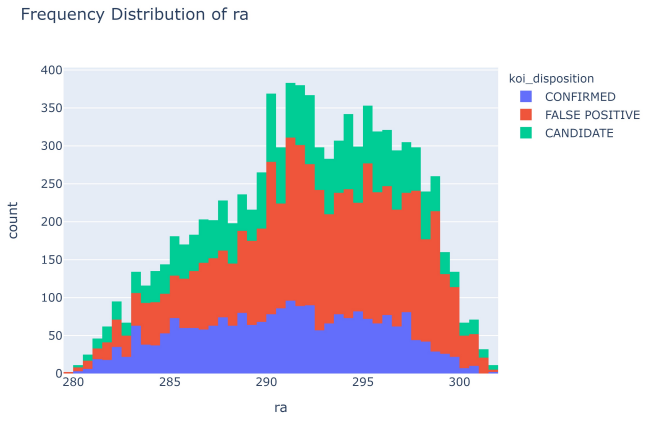

Fig. 3. Right Ascension of Exoplanets showing near Normal Distribution

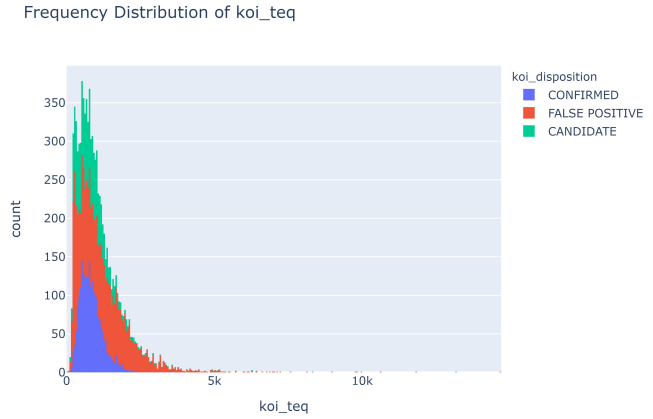

Fig. 4. Equilibrium Temperature of Exoplanets showing Skewed Distribution

the case for near normally distributed data, as well as skewed data.

The attribute kepler_name, koi_tce_delivname and koi_tce_plnt_num are dropped from our analysis. These are ID attributes which have been assigned to the objects of interest after analysis has been performed and they have been labelled (into exoplanets and non-exoplanets), and hence do not play a part in the building of the classification model. These attributes are thus removed from the analysis.

We finalise the data preprocessing step by standardising our data, to ensure all attributes are on the same scale. This increases performance and also decreases computational effort required by the models, thus speeding up the process of classification.

Additionally, it is to be noted that since the majority of the error values have a low order of magnitude, we propose to build two separate variations of models - one considering these error metrics and one without. This helps us analyse the effect of these attributes.

B. Feature Correlation and Variance

To reduce redundancy in the number of attributes chosen, we study the correlation between the attributes and observe the results. We see that while there is a correlation present between the attributes picked in our study, most of the correlations are not of a high order.

Additionally, only a few attributes exhibit high correlation with the dependent variable. Most attributes are observed to have only a medium or low correlation. We also observe that most attributes show a negative correlation with the dependent variable, suggesting that lower values of these attributes correspond to higher probabilities of an object being an exoplanet.

To observe the variance added to the dataset by the attributes, we perform Principal Component Analysis (PCA) on our dataset and choose the number of Principal Components as the number of chosen attributes. We observe that a full

Attribute	Description
dec	KIC Declination
koi_depth	Transit Depth (parts per million)
koi_duration	Transit Duration (hours)
koi_fpflag_co	Centroid Offset Flag
koi_fpflag_ec	Ephemeris Match Indicates Contamination Flag
koi_fpflag_nt	Not Transit-Like Flag
koi_fpflag_ss	Stellar Eclipse Flag
koi_impact	Impact Parameter
koi_insol	Insolation Flux [Earth flux]
koi_kepmag	Kepler-band
koi_model_snr	Transit Signal-to-Noise
koi_period	Orbital Period (days)
koi_prad	Planetary Radius
koi_slogg	Stellar Surface Gravity
koi_srad	Stellar Radius
koi_steff	Stellar Effective Temperature
koi_teq	Equilibrium Temperature (Kelvin)
koi_time0bk	Transit Epoch
ra	KIC Right Ascension

TABLE II
ATTRIBUTES CHOSEN FOR ANALYSIS

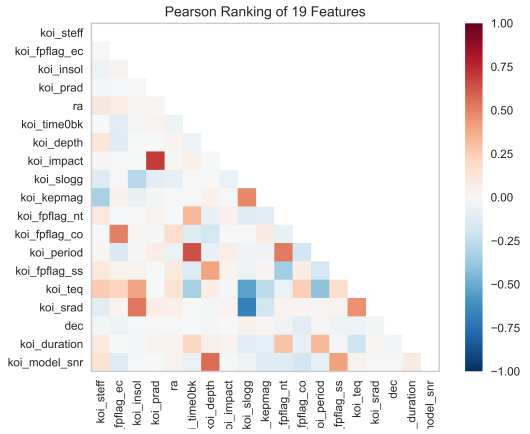

Fig. 5. Pearson Correlation without Error Attributes

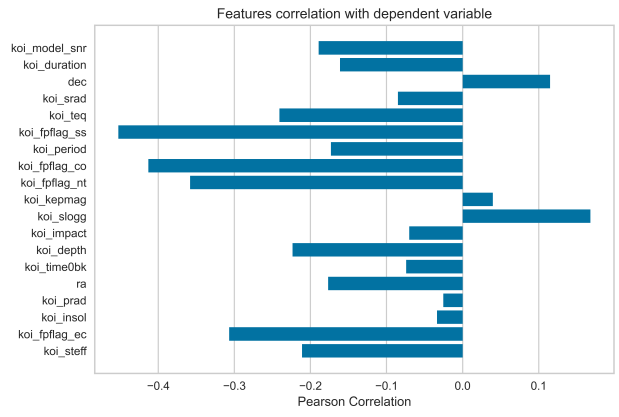

Fig. 6. Correlation of Non-Error Attributes with Dependent Variable

100% variance is reached only after taking into account every single attribute or component and the first principal component explains just 15% of the overall variance. The curve is a smooth decline, thus showing that each component individually adds significant variance to the dataset. We hence decide to move on with our set of attributes without removing any of them.

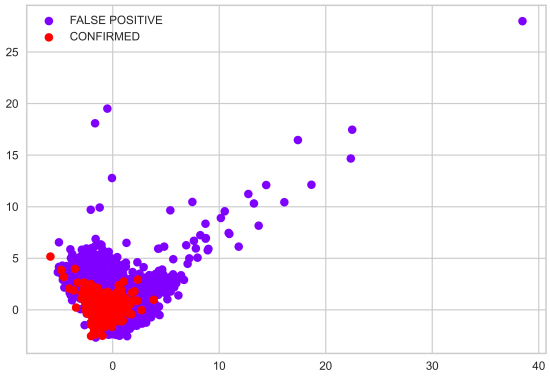

Fig. 7. PCA Decomposition

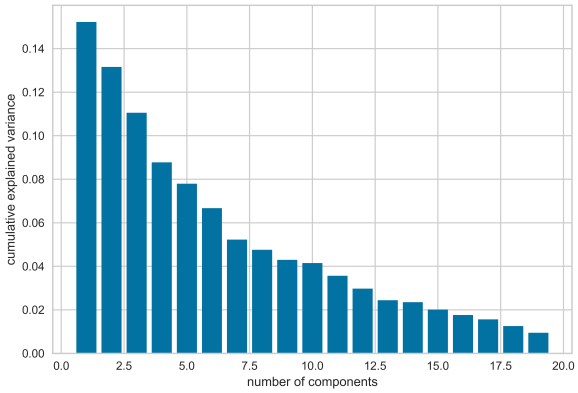

Fig. 8. Variance Distribution without Error Attributes

However, this changes when we consider the error attributes as well, as we observe that the first 30 components contribute to a full 100% of the variance, out of the complete set of 39 attributes. For this case, we consider only the first 30 principal components for our analysis and thus remove the remaining components. This step allows us to retain information from all the attributes while reducing the number of features as well.

Final observations also indicate that the spread of values for non-exoplanets is far greater than the spread of values for confirmed exoplanets. This shows that the magnitude of attributes for exoplanets is lesser compared to other objects of interest.

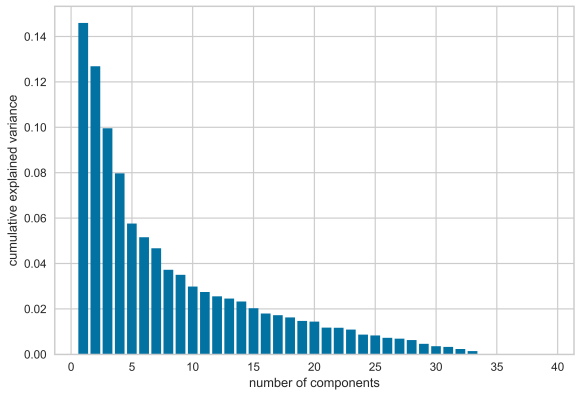

Fig. 9. Variance Distribution with Error Attributes

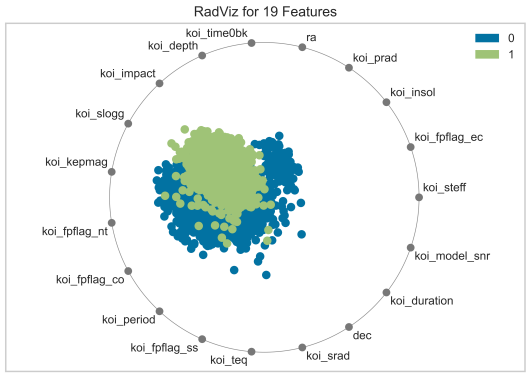

Fig. 10. Radial Visualisation without Error Attributes

C. Evaluation Metrics and Cross Validation

To counter the imbalance of the dataset, we propose different evaluation metrics, which take in account the imbalance. These include the F1 Score, Cohen Kappa score, Balanced Accuracy Score and finally the Confusion Matrix.

Additionally, to test out our classifier on different sets, we use F-Fold cross-validation across our entire dataset to ensure that we are not underfitting our classifier by introducing high bias. Since the dataset is imbalanced, we again use both - a non-stratified as well as a stratified splitting to ensure that within each fold the number of positive and negative examples are equal. We measure our classifier's performance across each split and finally take the mean of the performance achieved.

D. Classification Models

Preliminary visualisations showed us that the dataset is imbalanced .i.e the distribution of examples for each class is skewed towards the negative class. This owes to the abundance of non-exoplanet bodies in the universe compared to actual exoplanets. There is also an abundance of noise in the dataset,

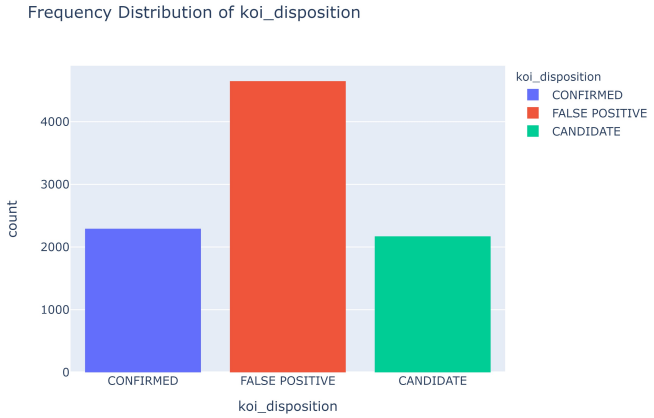

Fig. 11. Distribution of Classes

TABLE III
GRIDSEARCH PARAMETERS FOR SVM

Attribute	Possible Values
C	1, 1.05, 1.10, ..., 3
gamma	scale, auto
shrinking	True, False
tol	0.001, 0.01, 0.1, 1, 10, 100, 1000
class_weight	None, balanced

with inter-mixed classes present. To observe how different classifiers tackle this issue, we propose to use classifiers working on different approaches - traditional classifiers like the SVM, Bagging Ensemble models like the RandomForest, Boosting Classifiers like the AdaBoost and finally a Feed-Forward Neural Network. Additionally, we also create an Ensemble majority vote classifier that combines all of these together to take the majority vote and return the predicted label.

The attributes being considered in our study include all attributes except the ID attributes which assign names and ID's to each object of interest, along with any other attributes which are assigned by NASA after studies have been performed on these data points. We add to the aforementioned classifiers by training our model using two sets of considered attributes - one with and without the error attributes present. Finally, we perform GridSearch on each classifier with different values for its hyperparameters to choose the optimum set of values that provide maximum performance.

1) Support Vector Machine: We deploy a Support Vector Machine (SVM) as our first classifier. We additionally use a Gaussian or an RBF kernel to tackle the issue of our classes being non-linearly separable. This kernel performs a Gaussian transformation to map the input feature space to a higher dimensional space. We then tune our model by performing cross-validated GridSearch on it, allowing it to run through multiple combinations of parameters and finally choosing the best model based on its performance.

TABLE IV
CHOSEN GRIDSEARCH PARAMETERS FOR SVM

Attribute	Chosen value
C	1.85
class_weight	None
gamma	scale
shrinking	true
tol	1
class_weight	None, balanced

TABLE V
GRIDSEARCH PARAMETERS FOR RANDOM FOREST

Attribute	Possible Values
num_estimators	100, 200, 300, 400, 500, 600
max_depth	1, 2, 3 , 4, 5, 6, 7, 8, 9
min_samples_leaf	0, 0.2, 0.4, 0.6, 0.8, 1.0
max_features	None, sqrt, log2
class_weight	None, balanced, balanced_subsample

2) Random Forest: Our second classifier is an Ensemble based Bagging Classifier based on the Decision Tree. The RandomForest works by training multiple individual Decision Trees on each subset of the dataset and finally averaging the result obtained on classifying the input test example. Each Decision Tree in the Random Forest is alike, and can be tuned based on its hyperparameters. The Random Forest is extremely powerful and hence requires less fine tuning to build an accurate model. We again deploy GridSearch to tune each Decision Tree's hyperparameters to choose the most optimum set which returns the best results.

3) AdaBoost: AdaBoost is the third classifier we use. AdaBoost is a powerful Ensemble Boosting algorithm which also uses multiple Decision Trees, similar to Random Forest. However, instead of training each Decision Tree on a subset of the training data, each Tree is individually and sequentially trained on the entire dataset, such that the "harder examples" (which consist of the examples predicted wrongly) are given larger importance. This is done by assigning weights to each training example, and modifying these weights based on the current classifier's predictions. A corresponding weight for an example is increased for the next classifier if the current classifier predicts it incorrectly and vice versa. However this also makes it highly prone to overfitting, and hence we choose the average out the results using cross-validation.

We tune AdaBoost by using the following parameters. Since AdaBoost has its own hyperparameters (apart from the Decision Tree hyperparameters), we focus on tuning the AdaBoost classifier itself, and only vary the depth of the Decision Trees used.

TABLE VI
CHOSEN GRIDSEARCH PARAMETERS FOR RANDOM FOREST

Attribute	Chosen value
num_estimators	100
criterion	gini
max_depth	None
min_samples	2

TABLE VII
GRIDSEARCH PARAMETERS FOR ADABOOST

Attribute	Possible Values
No. of estimators	70, 80, 90, 100, 110, 120
learning_rate	0.8, 0.85, 0.9, 0.95, 1, 1.05, 1.1, 1.15
algorithm	SAMME, SAMME.R

TABLE VIII
CHOSEN GRIDSEARCH PARAMETERS FOR ADABOOST

Attribute	Chosen value
Algorithm	SAMME
base_estimator	DecisionTreeClassifier(max_depth=3)
learning_rate	0.95
n_estimators	80

4) Neural Network: As our final model, we train a regular Neural Network architecture to perform classifications. To tune our neural network, we vary the number of layers as well as the neurons per layer. However, we fix our activation functions for all layers except the final output layer to be ReLu and the final layer to be Sigmoid since we are performing binary classification. Finally, we choose Adam as the optimizer and Binary Cross Entropy as the Loss Function.

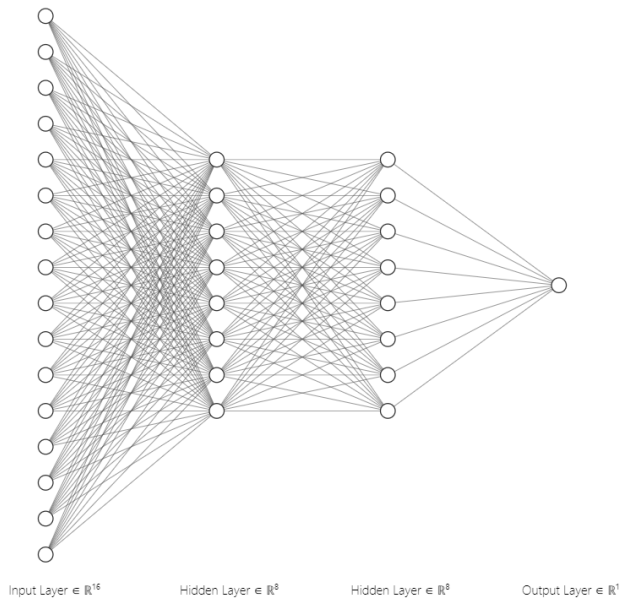

Fig. 12. Feed-Forward Neural Network Architecture (scaled by 8)

VI. RESULTS

A. Model Performance

We propose to test using 10-Fold cross-validation on the entire set of observations. Cross-validation helps analyse the model's performance against each subset of the dataset by training the model on n-1 folds and testing the model on the nth fold. Cross-validation also creates stratified splits, thus

TABLE IX
HYPERPARAMATERS FOR NEURAL NETWORK

Hyperparameter	Chosen
Layer Dimensions	256, 128, 128, 1
Optimizer	Adam
Loss	binary crossentropy
Metrics	accuracy
Epochs	20

Model	Stratified F-1 Score	Non-Stratified F-1 Score
SVM	97.72%	97.66%
RandomForest	98%	98.13%
AdaBoost	98.03%	98.11%
Neural Network	97.78%	97.46%
Majority - Vote	98.28%	98.29%

TABLE X
MODEL PERFORMANCE WITHOUT ERROR ATTRIBUTES

creating a pipeline to estimate the model's accuracy even on imbalanced datasets.

Each of the aforementioned models are subjected to cross-validation using both stratified as well as non-stratified splits. Additionally, error attributes are also considered to build additional models. We use the F-1 Score to mark a model's performance, since the dataset is imbalanced and a plain accuracy score would paint the wrong picture.

We observe that on a dataset without the error attributes, all the models work really well. However, in terms of numbers the AdaBoost classifier obtains the best performance with an average F-1 score of 98%. The majority vote ensemble model outperforms the rest by a good margin, attaining an F-1 score of 98.3%. This shows that models which rank features differently can be combined together sequentially to improve overall performance.

On considering the error attributes as well, it is observed that the performance increases. This is because PCA helps us maximise variance and retain an uncorrelated and independent feature set. In this case as well, we see that AdaBoost outperforms the others with a greater F-1 score. It even beats its own F-1 score achieved on removing the error attributes. The majority vote ensemble model again performs well, but fails to achieve the same scores as done previously.

We can conclude that the models built with the Error attributes tend to do better than the models built after removing the error attributes. Although all models perform almost equally well, the AdaBoost classifier outperforms the rest and also performs well across all cases, while the majority vote ensemble model is consistent as well.

Model	Stratified F-1 Score	Non-Stratified F-1 Score
SVM	98.28%	98.31%
RandomForest	97.68%	97.61%
AdaBoost	98.01%	98.17%
Neural Network	98.16%	98.27%
Majority - Vote	98.29%	98.32%

TABLE XI
MODEL PERFORMANCE WITH ERROR ATTRIBUTES

B. Analysis of Feature Importance

Different machine learning algorithms work differently. They all differ in the method used to classify, the sampling as well as initialisation of variables used for modelling. This also leads to differences in the features being analysed, and the priority or importance given to each feature while performing a prediction or classification.

We analyse the differences in the feature importance by performing a recursive elimination of features and observing the features that provide maximum changes to the model's performance. These features are then plotted in order of their relevancy to show how each model ranks the features. This however, is not possible with the Support Vector Machine, since the features are transformed into a higher dimensional feature space.

The AdaBoost and Random Forest classifiers assign the highest importance to flag variables namely the Centroid Offset flag and Non-Transit like Flag. We also observe a steep decline in the importance of attributes, beyond the set of flag attributes thus suggesting that the AdaBoost and Random Forest classifiers are extremely powerful while working with categorical variables and uses them to make the core decisions while splitting the dataset into decision nodes. This also proves that categorical variables help perform classifications faster since unlike continuous variables, they do not have to be binned.

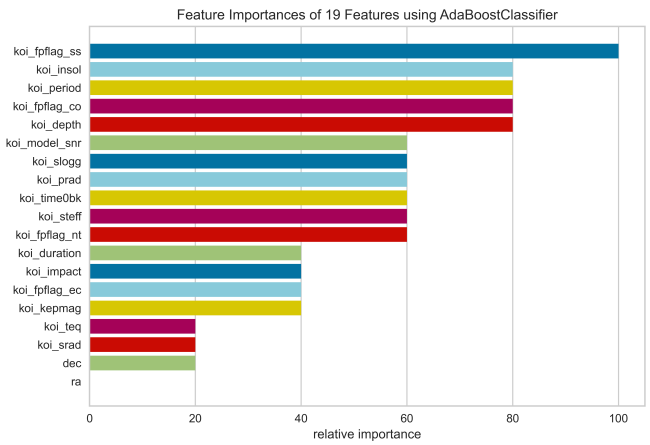

Fig. 13. Feature Importance for AdaBoost Classifier

The AdaBoost classifier displays a different order of feature importance, even though it uses a Decision Tree, just like the Random Forest classifer. Flag variables although exhibiting a high level of importance, do not take the top spot. This is accounted by the difference between Boosting and Bagging. Unlike Random Forest, we also observe that the feature importance tends to show a pattern, with the magnitude of importance taking a discrete set of proportions. Additionally, we also observe that the AdaBoost classifier can assign an importance of zero to any feature.

The Neural Network again returns a completely different order of ranked features. However, we observe that the flag

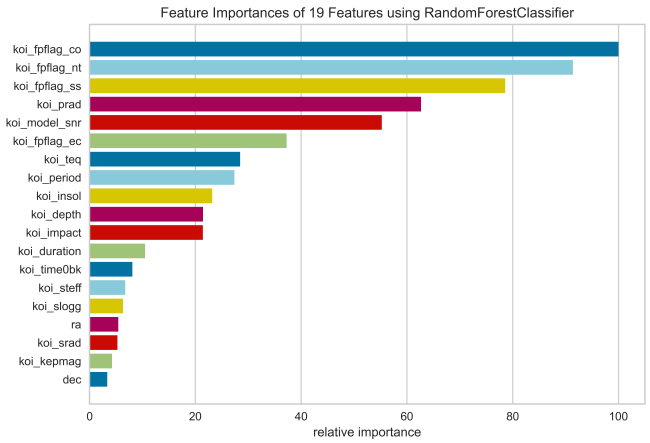

Fig. 14. Feature Importance for Random Forest

attributes are ranked the highest, similar to the Random Forest classifier. This confidently shows that in classification of observations, categorical variables drive the process of making well-informed decisions leading to accurate predictions. Categorical variables help form decisions faster, and are quick to decrease the overall entropy in a set of observations regardless of the algorithm deployed.

Additionally, because each algorithm ranks features differently, there is a possibility of combining together all these algorithms sequentially since an example classified wrongly by one algorithm may be classified correctly by another. This will also average out the feature importance for low scoring attributes while retaining the high scores for important ones. This leads to our majority-vote ensemble classifier, which performs a majority vote among its models to classify observations.

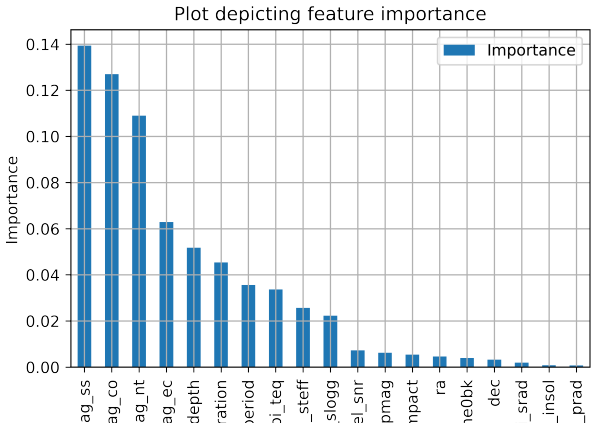

Fig. 15. Feature Importance for Neural Network

VII. OBSERVATIONS

There are many criteria that exist which are used to group planets. Planets are grouped based on their mass, orbital range and composition to name a few. Planets which are part of the same group or cluster often exhibit similar properties as its neighbours and are used to classify or group newer planets. Over the years, scientists have tried to analyse readings to group together such planets, in an effort to make analysis of these heavenly bodies simpler and organised. We can today categorise the different exoplanets into 4 broad categories.

1) Gas Giant - similar to Jupyter or Saturn
2) Neptune Like - similar to Uranus or Neptune
3) Super Earth - more massive than Earth but lighter than Neptune
4) Terrestrial - Rocky and Earth-like

In recent times to make study of these exoplanets easier, these have been further separated into a total of 8 categories. Since these additional categories are branched off from the main ones, there is quite a bit of overlap between them.

The 8 categories are Gas Giant, Mesoplanet, Mini Neptune, Planemo, Planetar, Super Jupyter, Super Earth and Sub Earth.

We can analyse the distribution of the exoplanets in the dataset using a cluster analysis. This will allow us to visualise the differences in characteristics between these exoplanets as well as observe their distribution into categories.

A. K-Means Clustering

K-Means Clustering is an iterative clustering algorithm used to group together similar objects and identify patterns. Clusters are formed by assigning and recalculating centroids iteratively until convergence. These centroids are highly representative of a cluster, and are often used to represent characteristics for the entire group as well.

K-Means however requires the number of clusters (or categories) in this case to be fed into the model as a hyperparameter. There are different techniques to find out the optimum number of clusters, the most common being the Sum of Squares Error (SSE). This method calculates the SSE with every batch of clustering across various number of clusters and plots them. A sharp bend in this curve marks the optimal number of clusters. At this point, both the value of the SSE as well as the number of clusters is balanced to neither underfit nor overfit the set of observations.

We observe that the optimal number of clusters returned is 7 instead of 8. This is because of the noise present in the set of observations in the form of overlaps between the classes. A visualisation of the distribution shows us that there are 4 primary groups of planets present in the dataset, while more groups have been constructed by breaking down one of the classes.

There is also an uneven distribution of exoplanets present. We see that the sum of squares error for each cluster is not uniform. Although most clusters have a low error suggesting that those groups consist of planets that share common properties and are very similar to each other, the largest cluster also has the highest sum of squares error, suggesting that a majority of exoplanets have properties vastly different from the other exoplanets, but are themselves not very similar to each other.

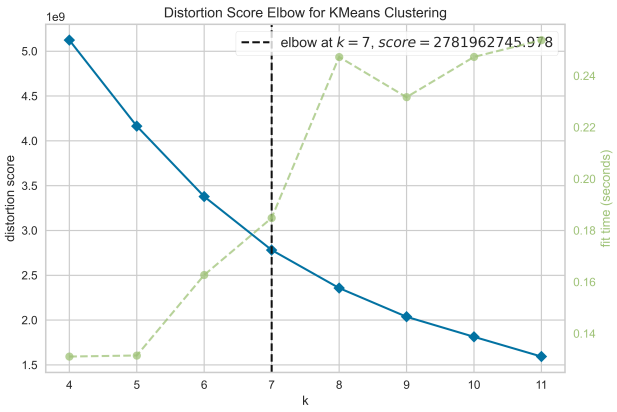

Fig. 16. SSE Method to Find Optimal Number of Clusters

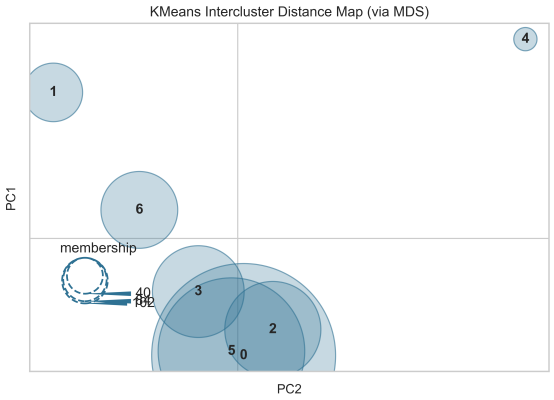

Fig. 17. Inter Cluster Distances

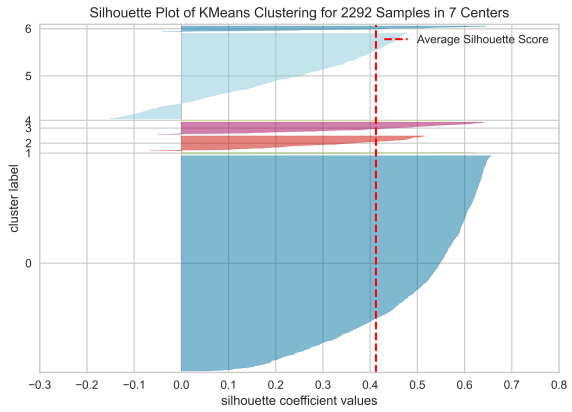

Fig. 18. Within Cluster Sum of Squares Error

B. Agglomerative Hierarchical Clustering

Agglomerative Hierarchical Clustering is another clustering algorithm that builds clusters by iteratively combining observations similar to each other into a single cluster, until there is only one cluster left. However, unlike K-Means, Hierarchical Clustering does not need the number of clusters to group together observations. We can obtain the desired number of clusters needed from dendogram, which is a hierarchical tree storing the order in which the clusters were formed.

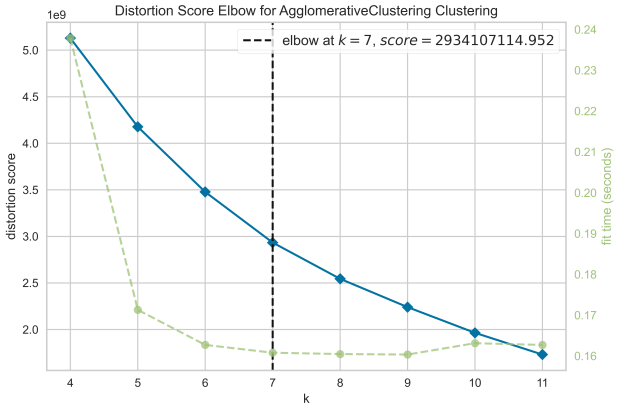

Fig. 19. Within Cluster Sum of Squares Error

However, we can still find the optimal number of clusters recommended using a similar approach. We again obtain the optimal number as 7, thus proving without doubt about the overlap between the classes.

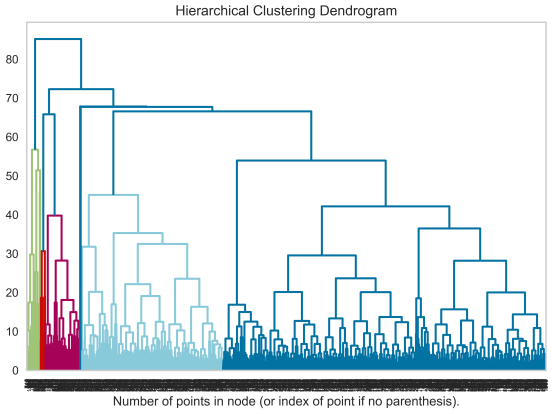

Fig. 20. Dendogram of Merges Obtained

C. Analysis of Characteristics of Grouped Exoplanets

We can observe that some attributes are vastly different across classes compared to the remaining attributes such as Insolation Flux and Equilibrium Temperature, while some attributes are very similar across classes such as the Right Ascension and the Kepler Band Magnitude.

This shows that there is no significant trend between attributes themselves, since exoplanets of different sizes and compositions, can have a similar set of attributes. This makes it increasingly difficult for researchers and scientists to pinpoint planets and declare them habitable because of the variation in characteristics.

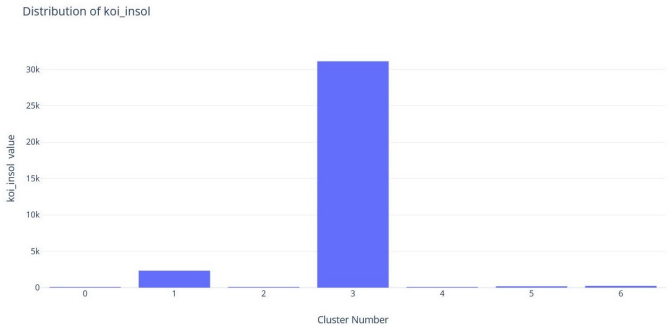

Fig. 21. Median values for Insolation Flux across Clusters

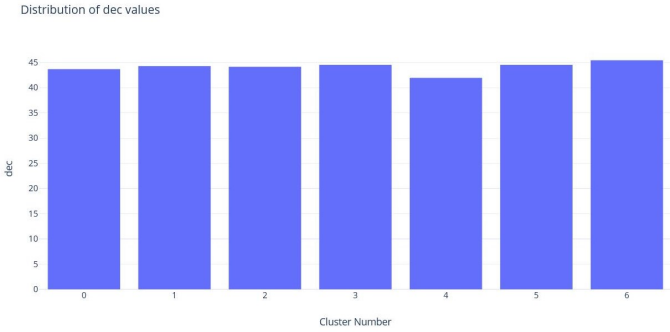

Fig. 22. Median values for KIC Declination across Clusters

VIII. CONCLUSION

The analysis performed was in depth and provided extremely valuable insights about exoplanets and the methodologies used to classify them. This could not have been possible without a well-thought out plan of action and efficient teamwork among the members.

Aditeya created the data preprocessing pipeline which allowed for removal of noise and redundant features in the dataset. Additionally, he worked on classification models based on traditional Machine Learning approaches.

Ameya worked on the literature review to provide us with background information about exoplanets which proved to be extremely useful and helped us plan and build our models. He also came up with the various hypotheses to be tested and also worked on Deep Learning based approaches.

Saarthak provided us with the necessary visualisations to help us understand the data better, without which it would not be possible to form our hypotheses. Additionally, he worked on the analysis of the models themselves, including the ranking of feature importance to give us greater insights on our classification.

REFERENCES

[1] NASA, "Kepler and k2 mission."

[2] CalTech, "Cumulative kepler object of interest data," https://exoplanetarchive.ipac.caltech.edu/cgi-bin/TblView/nph-tblView?app=ExoTblsconfig=cumulative, 2020.

[3] N. M. Batalha, "Exploring exoplanet populations with nasa's kepler mission," https://www.pnas.org/content/111/35/12647, 2014.

[4] S. T. Jack J Lissauer, Rebekah I Dawson, "Advances in exoplanet science from kepler," 2014. [Online]. Available: https://www.nature.com/nature/about

[5] A. V. Christopher J. Shallue, "Identifying exoplanets with deep learning: A five planet resonant chain around kepler-80 and an eighth planet around kepler-90," 2018. [Online]. Available: https://arxiv.org/abs/1712.05044

[6] M. J.-M. Vicente Alarcon-Aquino, "Transiting exoplanet discovery using machine learning techniques," 2020. [Online]. Available: shorturl.at/mG069

[7] S. T. B. Timothy D. Morton, "False positive probabilities for all kepler objects of interest: 1284 newly validated planets and 428 likely false positives," 2016. [Online]. Available: https://iopscience.iop.org/article/10.3847/0004-637X/822/2/86/meta