Hunting Extra-Solar Planets using Machine Learning

Shivansh Mathur[1] ,Anurag Malik[2], Ishita Rastogi[3], Ishvinder Singh[4] , Kartik Tiwari[5] , Sanskriti Agarwal[6]
Associate Professor[2]
B.Tech Scholar[1,3,4,5,6]
Department of Computer Science & Engineering, Moradabad Institute of Technology[1,2,3,4,5,6]
Moradabad, India
anurag_malik@rediffmail.com[2] , ishitarastogi2324@gmail.com[3] , ishvinder133@gmail.com[4],
tkartik88@gmail.com[5] , sanskriti02911@gmail.com[6] , shivanshmathurr@gmail.com

Abstract*—In this paper, we will use machine learning techniques to detect exoplanets based on the dataset provided by Kepler. Astronomers are already working on detecting such planets using different techniques, but here, we will be using the transit approach and training different models for given data. The different models will have different accuracy and will be compared after the complete training of the model, and the model with the maximum accuracy will be considered. According to the data till November 18, 2021, Kepler has confirmed the discovery of 2402 planets and has shown 2361 planets as true positives. We aim at discovering more such exoplanets or searching about the discovered true positives. Our approach will be comparatively less time-consuming as compared to the already used approaches.

Keywords—exoplanets, machine-learning, Kepler, TESS, astronomy, transit approach

I INTRODUCTION

The study of planets, stars, and other astronomical objects has always fascinated the human species. From the discovery of the Moon in 1610 till today, we have come a long way in our search of celestial bodies. Just like Earth and its other siblings that revolve around our native star Sun, we have discovered other stars and planets that revolve around them. The planets that revolve around stars other than the Sun are called Exoplanets or Extra-Solar Planets.

Since the launch of the Kepler satellite, the number of discovered exoplanets has increased significantly, and in 2018 NASA launched TESS (Transiting Exoplanet Survey Satellite) aimed explicitly to find more such planets, and the list is expected to grow more. However, the task is not very simple due to complex data analysis and manual processing of this vast data. However, with advancements in technology, the process is being machine-controlled with the assistance of Machine Learning and Deep Learning techniques.

There are various ways to detecting exoplanet signals. Some of them are :

Direct Imaging[4]: Exoplanet imaging is done directly with the help of massive telescopes that are fitted with accommodative optics. This technique is sensitive to the hotter, brighter, and larger exoplanets on wide and eccentric orbits. The separation from its native star allows for spectra to be obtained directly, which enables the direct activity of light.

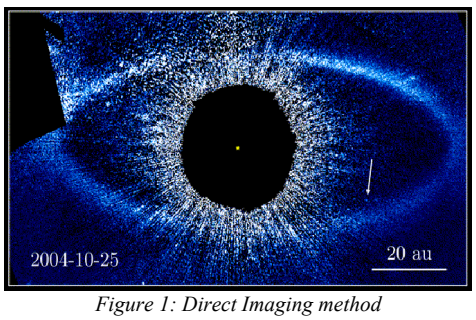

Figure 1: Direct Imaging method

Radial Velocity[5]: The exoplanet detection is done by measuring the Doppler effect within the host starlight due to the attractive force between two bodies. The radial velocity technique permits for a minimum mass (dependant on orbital inclination) to be calculated. This technique is responsive to exoplanets with an outsized mass orbiting near their host star perpendicular to the sky plane.

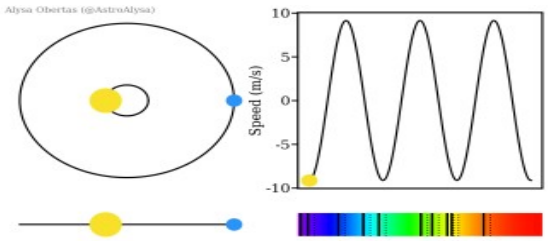

Figure 2: Radial Velocity and Doppler Effect

Microlensing[6]: Exoplanet detection is done by measuring characteristic light curve changes caused by changes within the lensing effect ascertained once a star with a planet passes in front of a distant star. The technique is restricted to distant one-time events and by the shortage of correct determination of the world and orbit parameters. It is, however, a unique technique because of the lack of strong radii or mass biasses, creating it ideal for applied math population studies.

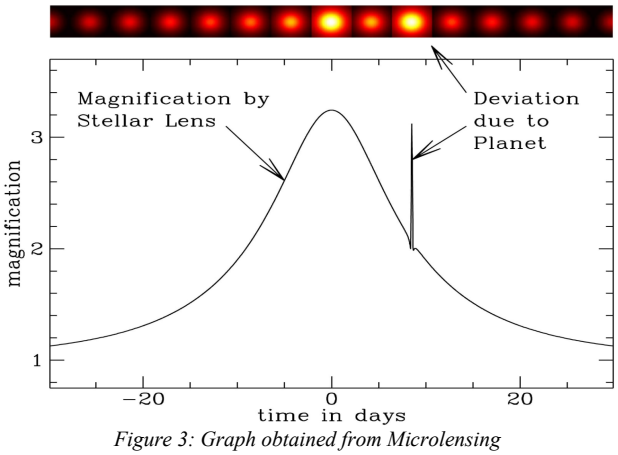

Figure 3: Graph obtained from Microlensing

Transit[7]: The exoplanet detection is done by measuring a periodic decrease within the flux received from the host star due to the object transiting ahead of the host star. The transiting technique is susceptible to massive exoplanets orbiting near their host star and correctly determines parameters like planetary radius, planetary mass, atmosphere, composition relative to the host star. It is also

the most popular and successful method, and to date, more than 2000 exoplanets have been found using Transit Method.

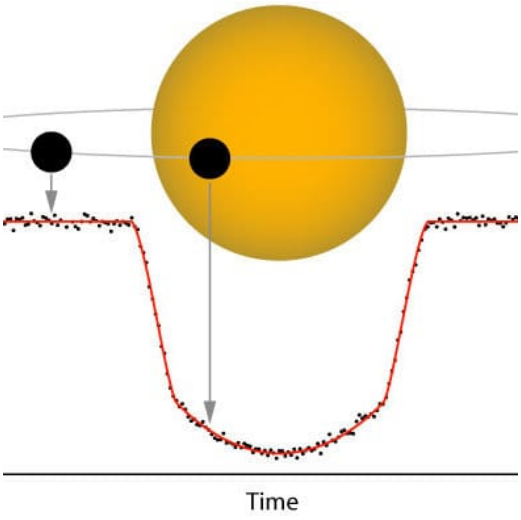

Figure 4: The Transit method

In this paper, we are going train some Machine Learning models to sight whether or not a given object is an exoplanet candidate or not using the data available on Mikulski Archive for Space Telescopes also called as MAST archive.

The general flowchart of our approach is as follows

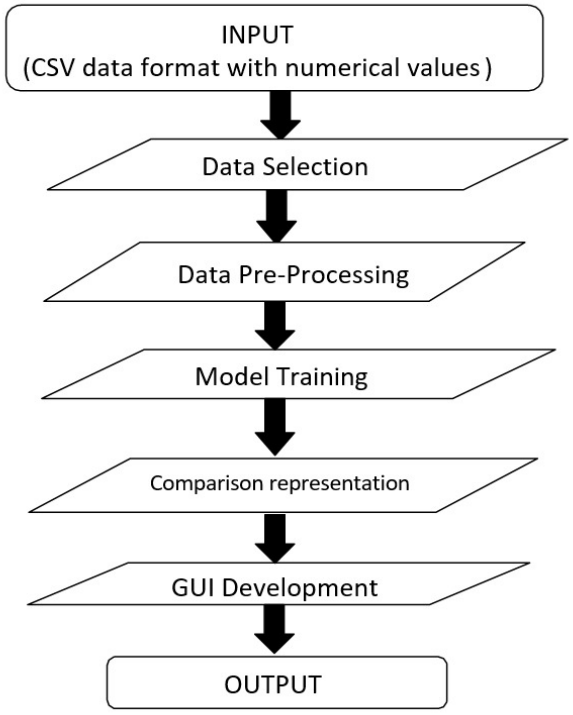

Figure 5: General solution flowchart

II Problem Statement

We are trying to determine whether a signal is an exoplanet candidate or not using the transit approach with the help of different machine learning models. The aim is to find planets that may be a bit similar to Earth and help us identify if there is any other system existing that is similar to our solar system. The principal technique used till now is the Transit approach, which includes monitoring dips in the brightness of the stars and identifying the appearance of some planets causing these dips. However, currently, the graphs of these dips are plotted and monitored manually, so we focus on discovering these planets without using a manual approach.

III Existing Solutions

According to our literature survey, there have been multiple solutions for automating exoplanets' vetting process, and even today, new exoplanets are being discovered. There are different methods for detection of detecting exoplanets. However, in our survey, we found that Transiting Method is the most used and most successful method among all other methods.

Deep Learning approaches made by Christopher J Shallue and Andrew Vanderberg[1] showed high results with around 98% accuracy. Their model successfully discovered two new exoplanet belts around stars, Kepler-80 and Kepler-90.

Other approaches used classification models like Decision Tree, Random Forest, SVM, KNN, and many more were trained and tested. In a paper by Abhishek Malik, Benjamin P Moster, and Christian Obermeier[2], a gradient boosted tree model was trained over 789 extracted features, and it yielded an accuracy of 91%.

In his article 'Exoplanet detection using AI', Shafi[3] talked about hunting extrasolar planets using CNN and SVM and trained them on combined data of Kepler, K2, and TESS.

Pat Brennan[8] found a circumbinary planet called 'TIC 172900988-b', which revolves around two stars and is hence categorized as Circumbinary, which means a planet revolving around two stars. The planet has been found using the transit method and is a gas giant like Jupyter but even massive in size.

IV The Proposed Solution

After studying the early methods used for the exoplanet discovery and their needs, we have found the transit method to be the most effective way to be used for the exoplanet discovery at the best success rate. It will be based on different dips caused by the planet crossing a star. These dips will be monitored continuously, and the values received will be used for further detection. For the implementation, we are using the data provided by MAST (Mikulski Archive for Space Telescopes). The data contains the values of various transit parameters. We will use machine learning techniques to train different models like logistic regression, decision tree classifier, and random forest classifier. These models will classify all the signals as either exoplanet candidates or false positives. Now, these models will be working on raw data, standardized data, and normalized data. To check which model will be the best, we will compare them based on their precision score, f1 score, accuracy, recall, and confusion matrix. The one with the best values will be selected for detection. Also, we will be checking on how our models are more efficient than the models mentioned.

A. Data Gathering

Various publicly hosted resources are available that contain loads and loads of data. NASA's Exoplanet Exploration Program, or ExEP, is an online archive that holds observational data from satellites like Kepler, K2, and TESS. Also, the California Institute of Technology (CalTech)

hosts an online archive that contains tools and data captured by Kepler and TESS satellites. However, we have used another publically available resource called Mikulski Archive for Space Telescopes that contains a plethora of data from various satellites and missions, including Kepler, K2, TESS, and many more. The data is available in raw format, CSV format, .tpf(Target Pixel File) format, and we can also obtain data obtained through different pipelines. It is swift and has a great search feature that helps find a particular range of data and a friendly user interface. Hence MAST is the platform of choice for this project.

Table 1: Some important features in dataset[10]

S No	Feature	Description
1	koi_disposition	Exoplanet Archive Disposition
2	koi_period	Orbital Period
3	koi_time0bk	Transit epoch
4	koi_duration	Transit Duration
5	koi_depth	Transit Depth
6	koi_prad	Planetary Radius
7	koi_teq	Equilibrium Temprature of planet
8	koi_model_snr	Transit signal to noise ratio

B. Data Cleaning and Processing

We downloaded the data in CSV format from MAST, and the data contains 9564 rows and 50 columns[10]. However, upon close observation, we found that some of the columns were null and some irrelevant for our purposes; hence we dropped some of the columns.

To fill columns having some null values, we used two methods, if data was numerical, we filled it using the mean, which is the average of the column, and if the data were categorical, we used mode, which is the most frequent data in the column.

Also, we found that the categorical data was nominal; hence to convert it from categorical to numerical data, we used a one-hot encoding technique. Also, we split the data into training and testing set in the 70-30 fashion, i.e., 70% percent data for training the model and 30% remaining for testing. During splitting, we set randomness to 1 and set the shuffle flag to True.

Lastly, since our data contains very different values, we decided to perform normalization and standardization using MinMaxScaler and StandardScaler, respectively. The data comes down to a more reasonable range. The MinMaxScaler changes all the values to range between 0 and 1, whereas StandardScaler ranges down the values between -1 and 1.

C. Model Selection

For our project we decided to train three different models on three datasets (main data, normalized data and standardized data). The three models are :-

a. Logistic Regression - Logistic regression is a method of modeling the probability of a distinct outcome given an associate input variable. The foremost standard logistic regression models a binary outcome, which may take two values like true/false and yes/no. Multinomial logistic regression can model eventualities with more than two potential distinct outcomes. Logistic regression could be a helpful analysis technique for classification issues wherever we attempt to determine if a brand new sample fits best into a class. As aspects of cyber security are classification issues, like attack detection, logistic regression could be a helpful analytic technique.

Input: Training data

1. For i←1 to k
2. For each training data instance d_i:
3. Set the target value for the regression to
   $z_i \leftarrow \frac{y_j - P(1 \mid d_j)}{[P(1 \mid d_j) . (1 - P(1 \mid d_j))]}$
4. nitialize the weight of instance d_j to $P(1 \mid d_j). (1 - P(1 \mid d_j)$
5. finalize a f(j) to the data with class value (z_j) & weights (wj)
   Classification Label Decision
6. Assign (class label:1) if $P(1|d_j) > 0.5$, otherwise (class label: 2)

Figure 6: Logistic Regression Algorithm[9]

b. Decision Tree Classifier - It is a type of supervised machine learning algorithm that may be used for both Classification and Regression issues. It is a tree-structured classifier, where internal nodes depict the data features, branches depict the decision rules, and every leaf node depicts the result. In a decision tree, there are two nodes: the Decision Node and Leaf Node. Decision nodes are accustomed to creating any decision and have multiple branches, whereas Leaf nodes are the output of these choices and do not contain any further branches. The decisions are performed based on features of the given data. It is a representation for obtaining all the potential solutions to a problem/decision supported by conditions. It is known as a decision tree because, just like a tree, it starts with the root node, which expands on more branches and constructs a tree-like structure.

GenDecTree(Sample S, Features F)
Steps:
1. ***If*** *stopping_condition(S, F) = true* ***then***
   a. *Leaf = createNode()*
   b. *leafLabel = classify(s)*
   c. ***return*** *leaf*
2. *root = createNode()*
3. *root.test_condition =findBestSpilt(S,F)*
4. *V = {v | v a possible outcomecfroot.test_condition}*
5. ***For each*** *value v ∈ V:*
   a. S_v = {s | root.test_condition(s) = v and s ∈ S };
   b. Child = TreeGrowth (S_v, F);
   c. *Add child as descent of root and label the edge {root → child} as v*
6. ***return*** *root*

Figure 7: Decision Tree Algorithm[9]

c. Random Forest Classifier - The fundamental idea behind random forest is a straightforward, however powerful, one — the knowledge of crowds. It is a supervised machine learning algorithm in which multiple Decision Tree Models work together as a committee. Every individual tree within the random forest spits out a {class or a category} prediction, and also, the class with the most votes becomes our model's prediction. In data science-speak, the rationale that the random forest model works so well is "A large variety of comparatively unrelated models (trees) operating as a committee can exceed any of the individual constituent models."

Algorithm 1: Pseudo code for the random forest algorithm

To generate c classifiers:
for $i = 1$ to c **do**
  Randomly sample the training data D with replacement to produce D_i
  Create a root node, N_i containing D_i
  Call BuildTree(N_i)
end for

BuildTree(N):
if N contains instances of only one class **then**
  return
else
  Randomly select x% of the possible splitting features in N
  Select the feature F with the highest information gain to split on
  Create f child nodes of N, N_1,..., N_f, where F has f possible values (F_1, ... , F_f)
  for $i = 1$ to f **do**
    Set the contents of N_i to D_i, where D_i is all instances in N that match
    F_i
    Call BuildTree(N_i)
  end for
end if

Figure 8: Random Forest Algorithm[9]

E. Evaluation

After training our models with data we tested them using our test data set and evaluated Confusion Metrix, Accuracy, Precision, Recall and F1 Score.

The test results of all are three models is given below

Table 2: Results for Logistic Regression Model

Dataset	Accuracy	F1 Score
Main	79.634%	.7975
Normalized	79.540%	.7968
Standardized	79.751%	.81673

Table 3: Results for Decision Tree Model

Dataset	Accuracy	F1 Score
Main	75.657%	.7665
Normalized	75.978%	.7708
Standardized	76.596%	.7751

Table 4: Results for Random Forest Model

Dataset	Accuracy	F1 Score
Main	79.887%	.8054
Normalized	80.985%	.8114
Standardized	82.318%	.8326

V. Conclusion And Future Scope

From our results, we can say that in our case random forest classifier works the best with an accuracy of 82% and F1 score of .82. Also, we have chosen the classical machine learning approach instead of the deep learning method like Convolutional Neural Network (CNN) because classical machine learning algorithms are easy to implement, far less complex, and require less time as well space to train. They also require less data as compared to deep learning algorithms. Also, our scores were comparable to the results of other algorithms we have discussed in this paper.

Our understanding of exoplanets has come a long long way since the discovery of the first exoplanet in 1995. In the early stages, the only extrasolar planets found were big gas giants and hot Jupiters but now with the advancement in our technology and sophisticated satellites, we can detect more Earth-like exoplanets as well. The data has been pouring in since then and has helped to improve our understanding of our solar system and its creation. Also, with the launch of the James Webb Telescope, there will be an enormous increase in the amount of data that we are currently receiving hence we plan on improving our system to predict the data from additional satellites like TESS and also improve the overall accuracy of the model.

ACKNOWLEDGEMENT

We are grateful to our Institution for guiding us to use this opportunity. We would also like to thank our guide Mr. Anurag Malik and our Project Committee for continuous guidance provided by them.

We, as a team, thank each other for all the co-operation and support required to complete this paper. All the authors whose papers were used for reference for our paper were a great source of information.

References

1 Christopher J. Shallue, Andrew Vanderburg, "Identifying Exoplanets with Deep Learning "

2 Abhishek Malik, Benjamin P. Moster, Christian Obermeier, "Exoplanet Detection using Machine learning "

3 Shafi, "Exoplanet Detection using Artificial intelligence ".

4 https://manyworlds.space/tag/direct-imaging/

5 https://blog.planethunters.org/2019/05/17/the-exoplanet-hunt-the-radial-velocity-method/

6 https://nexsci.caltech.edu/workshop/2011/

7 https://www.researchgate.net/figure/A-light-curve-showing-the-transit-method-of-detecting-exoplanets_fig1_312524607

8 https://exoplanets.nasa.gov/news/1688/discovery-alert-a-giant-planetand-three-eclipses/

9 https://researchgate.net/different-types-of-supervised-algorithms/

10 https://exoplanetarchive.ipac.caltech.edu/cgi-bin/TblView/nph-tblView?app=ExoTbls&config=koi