DOI: http://doi.org/10.34614/2023IYRC_F48

COMPARATIVE ANALYSIS OF MACHINE LEARNING MODELS FOR PREDICTING EXOPLANETS WITH KEPLER

Prathika Rajkumar

International School of the Sacred Heart, 4-3-1 Hiroo, Shibuya-Ku, Tokyo, Japan
(prathika.rajkumar@gmail.com)

ABSTRACT: The search for exoplanets has, in recent years, become a rising subject of interest in the field of space exploration. With NASA's Exoplanet Space Program launching missions like Kepler and TESS, both dedicated to the detection of exoplanets, it is vital for scientists to accurately interpret the data recorded (NASA, 2023). This paper addresses the construction of machine learning and deep learning algorithms that contribute to a more accurate means of exoplanet classification. Upon numerous examinations and refinements, we were able to produce two algorithms (Decision Tree Classifier and XGB classifier) with a classification accuracy (macro f1 score) of 100%.

1. INTRODUCTION

An exoplanet is a planet outside of our solar system which orbits a star. The first exoplanet was detected in 1992 and there have been more than 5,000 exoplanets confirmed by NASA since then (NASA, 2022). Detecting exoplanets is important because they are the basis for possible life and its detection gets us closer to the discovery of life outside Earth. Additionally, it provides us with a greater understanding of the origin of our solar system and creation of the universe (NASA, 2019).

Kepler, the largest space mission that was used to uncover such exoplanets, confirmed 2,662 (about half) of our verified exoplanets (NASA, 2020). The Kepler Space Telescope collected data for over nine years beginning in March 2009 and was designed to inspect our local region of the Milky Way galaxy and locate exoplanets using the transit method.

With a maximum orbital height of 0.917AU from Earth, the Kepler Space Telescope orbits our sun, collecting the data of the brightness of stars deep within our galaxy (NASA, 2009). If a planet passes in front of a star that is being observed (a phenomenon known as transit), the Kepler telescope, with its 95-megapixel array of CCDs (charge-coupled devices), detects a decrease in the brightness of the star. This is because when a planet travels between the star and the Kepler Space Telescope Camera, some of the light that was initially passing to the telescope gets blocked. When reformatting this data that Kepler picks up, we can create light curves (a graph with the independent variable being time and the dependent variable being brightness) with vertical dips at points of decreased brightness. When observing light curves, there will be noticeable static along the lined graph (known as noise) which do not correlate to the existence of exoplanets themselves. This noise is produced because of inconsistent factors like surrounding and internal heat, cosmic radiation, and background stars.

Consequently, for exoplanets, there is optimally a significant dip in the brightness of the light curve. For non-exoplanets, there should no, very little, or unusual dip in the brightness of the light curve. The detection of these two cases are complicated by the variance in size, distance, and shape of each of the exoplanets and non-exoplanets and by the noise that correlates with its data.

In this paper, we will use and compare specific generic and novel machine learning and deep learning models to classify exoplanets and non-exoplanets from the Kepler dataset.

2. RELATED WORKS

Due to the massive amount of data that scientists need to decipher for each space exploration mission, there have been attempts to use machine learning to classify obvious exoplanet cases without supervision.
Malik et al. (2021) created a gradient boost classifier using Time Series Feature Extraction, a common tool to examine time series data, and the *lightgbm* machine learning tool to efficiently utilize the characteristics of light curves (which is a time-series graph that records the intensity of light). However, they failed to account for the magnitude of abundant noise in raw data by splitting the data into discrete segments (which leads to a high number

of false-positive cases). In our research, we will aim to take static into more consideration through the application of data processors such as the Gaussian Filter.

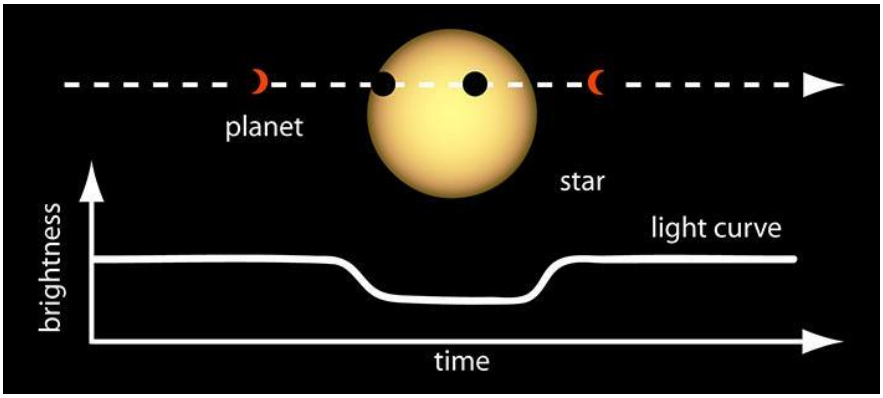

Figure 1. *A simplified diagram representing how the motion of exoplanets orbiting a star affects the observed light curve and its fit line* [NASA, 2022].

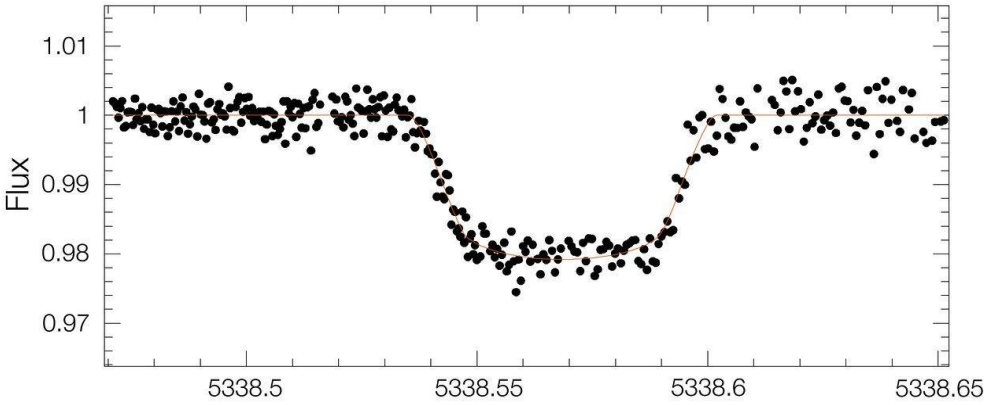

Figure 2. *The light curve of exoplanet WASP-19b. The black dots are a direct translation of the recordings and the red line represents its fit line. The deviation of the recordings to the fit line resemble the effects of noise* [TRAPPIST, 2010].

Cui et al. (2021) explored the field of computer vision and employed object detection framework for their classification. Although this achieves high accuracy metrics, the model uses an explicit (case-by-case) network architecture. Since the model must process to render an image and is already a complex structure with complex data processors, this method requires an immense amount of computational power. In our work, we will look to achieve higher accuracy metrics without the necessity for high computational power.

Tiensuu et al. (2019) utilizes both CNN (a deep learning model) and SVM (a machine learning model) with relatively low computational processors and loss functions but produces low and inconsistent accuracy metrics. As addressed by Malik et al. (2021), the common process of determining exoplanets is not optimal because even experts, who manually review cases altogether, do not maintain the same criteria for classification, and "something as little as the time of day or their mood" (Malik et al. 2021, pg.2) can affect their decisions. While this strengthens our understanding of the need for objective systems in this field, it also highlights the importance of an accurate model (with no room for discrepancies in false positive or false negative cases).

There are numerous ways we can work to enhance these systems' recall, precision, and other metrics (more on accuracy metrics in Section 4.1 of Results/Conclusion). In our research, we will work to create models with values of 1.00 in accuracy metrics (to overcome the issue brought up in Tiensuu et al.'s 2019 discussion) that do not require customized qualities. Instead, we will use standard processing techniques from *sklearn* (to overcome the issue brought up in Cui et al.'s 2019 discussion), work on a spectrum of machine and deep learning models, refine our data processing steps, and adjust the pipeline features for each algorithm. Unlike our related works, we will also utilize the creation of synthetic data to compensate for the major imbalance between the number of exoplanets and non-exoplanets (an issue mentioned in all three works).

Table 1. *Overall precision and recall scores comparing related works and our research*

	Overall Precision	Overall Recall
Malik et al. (2021)	0.63	0.82
Cui et al. (2021)	0.9	0.9
Tiensuu et al. (2019) (CNN)	0.574	1.00
Tiensuu et al. (2019) (SVM)	0.637	0.973
Our research (Decision Tree Classifier)	1.00	1.00
Out research (XGB Classifier)	1.00	1.00

3. METHODS/MODELS

Creating an algorithm to successfully classify exoplanets and non-exoplanets from a bundle of transit data involves numerous steps. The main stages are data collection, data correction, data processing, model construction, model training, and model testing. Within each of these stages, we can use numerous counts and combinations of methods and qualities. Upon going through this cycle of stages, we refine our results through trial and error by substituting different values and techniques or using completely different models altogether.

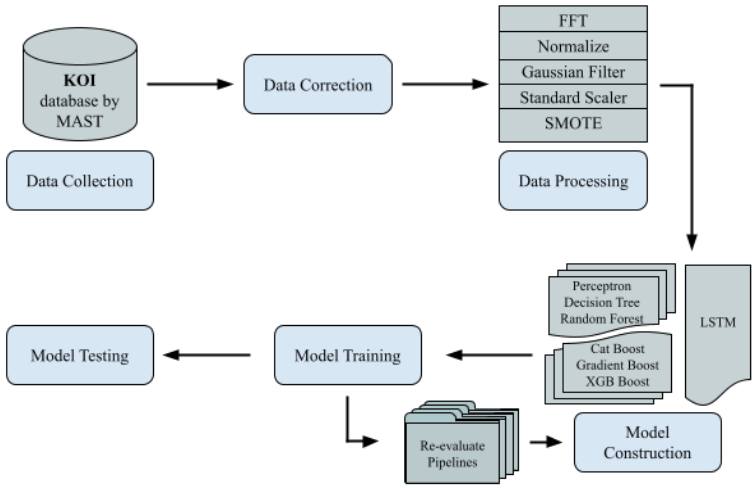

Figure 3. *A diagram summarizing the process of our methods and models. We will continue to loop the "re-evaluate pipelines" function until our training scores pass a certain accuracy threshold.*

3.1 Data Collection

Data collection establishes the dataset that the program will be learning and practicing off of. Using the Kepler Objects of Interest database (KOI), the largest labeled database for NASA's Exoplanet Program provided by MAST (Mikulski Archive for Space Telescope), we downloaded a data package that was recorded by the Kepler Space Telescope. This is also the database that each of our related works utilized (Malik et al. 2021, Cui et al. 2021, Tiensuu et al. 2019). For our program, we used a mixed-label dataset of 5,087 exoplanet and non-exoplanet cases, each with 3,197 flux recordings (the measurement of light at specific timestamps). We then split this data into training and testing datasets with a ratio of 7:3 (respectively).

3.2 Data Correction

Data correction is the process in which we re-format our datasets for them to be functional in the rest of our program. First, we searched for null data (a case with missing values) to erase them, as they may cause confusion within the program and lead to an undesirable output. Next, we re-labeled the original class labels to suit our

preferences: exoplanets, originally registered as class 2 in KOI, were re-labeled to class 1; non-exoplanets, originally registered as class 1 in KOI, were re-labeled to class 0. This is because when using machine learning techniques, the convention is to have class 0 set to a negative class and class 1 set to a positive class.

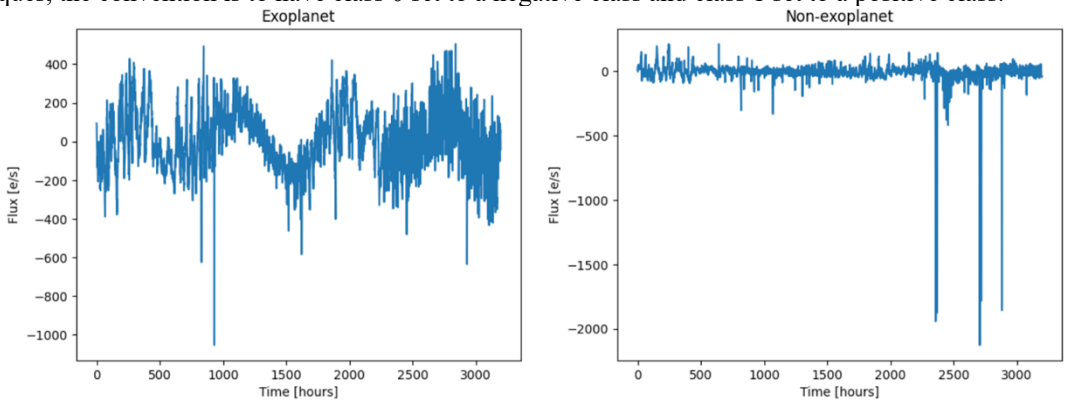

Figure 4. *The light curves for an exoplanet (class 1) and non-exoplanet (class 0) over an extended time frame. In our research, we will use key features present or absent in the light curves in relation to class 1 and class 0 cases (eg. periodicity in the exoplanet light curve is clear but non-existent in the non-exoplanet light curve).*

3.3 Data Processing

Data processing is an important procedure for evaluating and refining our datasets to make it functional in the context of our program. Depending on the total number of data, difference between classes, and intended purpose and output, the data processors we use within a program can vary tremendously. After numerous trials, we found the following transitions to suit best with our purposes: FFT, Normalizing, Gaussian Filter, Standard Scaler, and SMOTE.

FFT (Fast Fourier Transform) (*scipy.fftpack.fft*, 2023) is a conversion method that processes the periodic modulation of a given data and transfigures them into graphs with clear frequencies. We apply this to our program to create a distinction between the two classes because the light curves of exoplanets have periodicity and those of non-exoplanets do not.

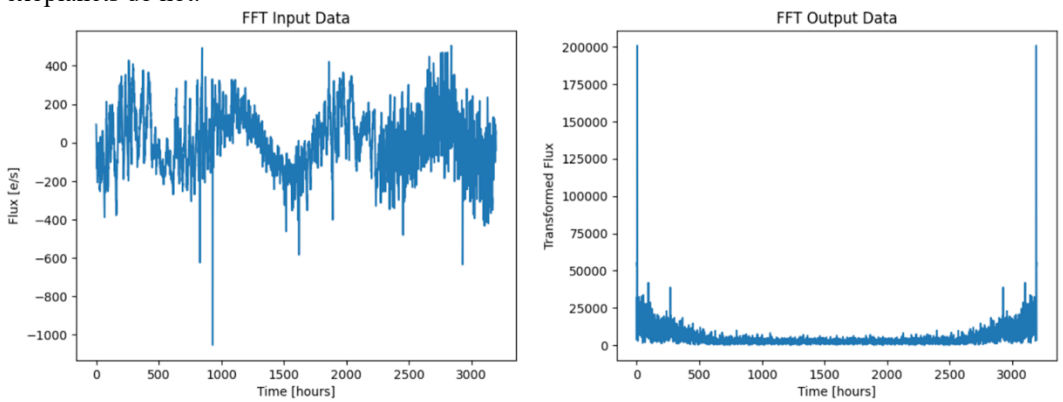

Figure 5. *The light curve for an exoplanet case before (left) and after (right) FFT. The transformed points on the x-axis (a specific time) with a higher y value (flux) are indicators of the light curve's periodicity.*

Next, we normalize our data (*sklearn.preprocessing.normalize*, 2023) to reformat the FFT outputs into a more suitable graph (used in both Cui et al. 2021 and Tiensuu et al. 2019 for the same purpose). We must perform this step because we only need to feed the program extracts of the graph produced by the FFT step (as seen in Fig. 6).

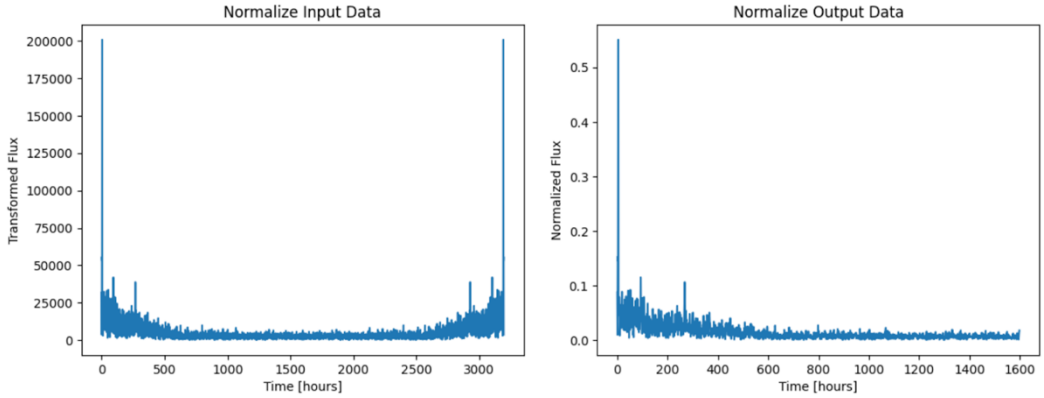

Figure 6. *The light curve for an exoplanet case before (left) and after (right) normalization. The transformed axis is scaled down with the curve's symmetry (across the x-axis) removed.*

The Gaussian Filter (*scipy.ndimage.filter.gaussian_filter*, 2023) is then applied on our normalized FFT outputs to remove noise. By applying a Gaussian Filter, we dimmed this noise by blurring all data points below a certain magnitude, so that any point that is not beyond a fixed height becomes a smooth line. In our program, we set this height to a sigma of ten (a standard deviation of 10%). This reduces the number of false positive cases by canceling the effects of false periodicity created by obstructions (such as heat and radiation).

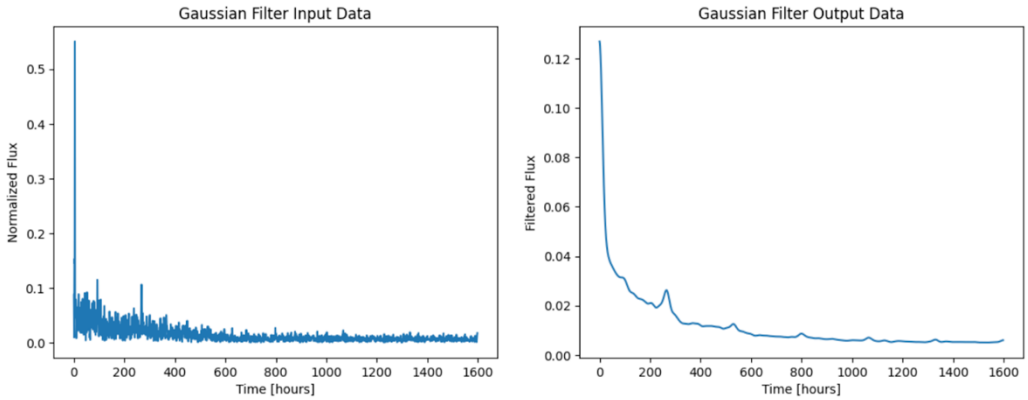

Figure 7. *The light curve for an exoplanet case before (left) and after (right) the application of the Gaussian Filter. The transformed curve results in one with decreased noise.*

Next, we apply the Standard Scaler to our datasets. The Standard Scaler (*sklearn.preprocessing. StandardScaler*, 2023) feature re-evaluates our data points by rescaling the case's mean to zero and its standard deviation to one. We performed this step since both machine learning and deep learning algorithms generally perform better when the data is standardized and scaled relative to a point and its surroundings.

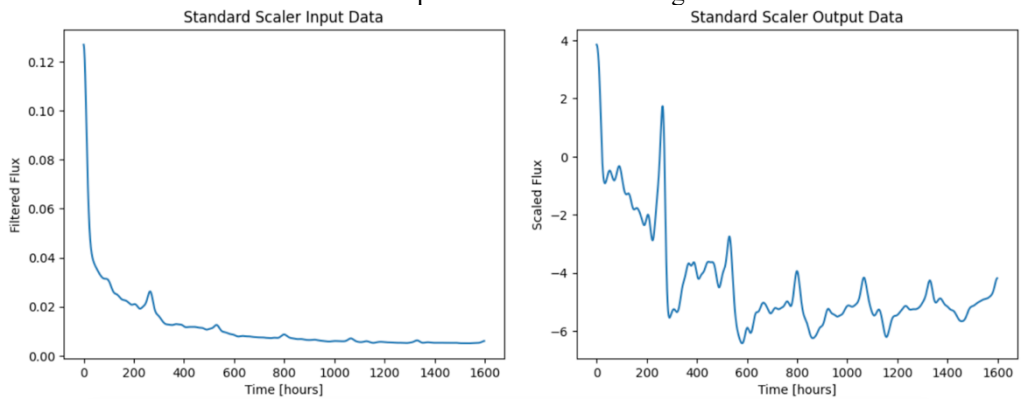

Figure 8. *The light curve for an exoplanet case before (left) and after (right) the application of Standard Scaler. The transformed axis is re-scaled and the points of standard deviation are removed.*

SMOTE (Synthetic Minority Oversampling Technique) (*imblearn.over_sampling.SMOTE*, 2023) is the final and most complex data processing unit in our program. In our data, the majority class is class 0 (non-exoplanets) and

the minority class is class 1 (exoplanets); the ratio of majority to minority class is 5050:37 in our overall dataset. As there is a major imbalance between our classes, our program will not perform as intended because even if the program were to classify every exoplanet incorrectly, it can still attain a 99% accuracy score. Using SMOTE, we make up for this by undersampling the majority class and oversampling the minority class.

To oversample the minority class data, SMOTE creates synthetic cases with data points that are close to the standard deviation and reflective of the original class 1 dataset. For our program, we oversampled the minority class up to 20% of the entire dataset and undersampled our majority class by decreasing its size by 30%. By the end of our SMOTE function, we created mixed datasets (of synthetic and original cases) with a new ratio of the majority to minority class being 3366:1010. This procedure of generating synthetic data is a key aspect which differentiates our research from our related works (Malik et al. 2021, Cui et al. 2021, Tiensuu et al. 2019).

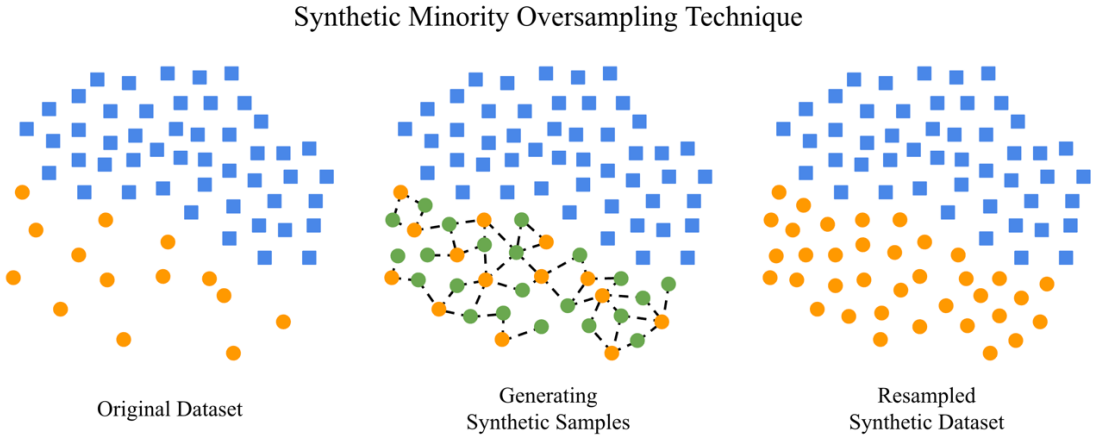

Figure 9. *A diagram representing the overall function of SMOTE on an abstract level.*

3.4 Model Construction

There are many deep learning and machine learning methods we can use to create our algorithm. So, we experimented with many to find which ones were the most suited for our program through trial and error. Upon constructing many different models and training and testing them to see their output, we refined our data processing methods and altered the qualities of our models to see what worked best. Upon noting the sequences which lead to a deep learning or machine learning model's best performance, we moved onto the next model and performed the same process. Our generic machine learning models include Perceptron (*sklearn.linear_model.Perceptron*, 2023), Logistic Regression (*sklearn.linear_model. LogisticRegression*, 2023), Random Forest (*sklearn.ensemble.RandomForestClassifier*, 2023), and Decision Tree (*sklearn.tree.DecisionTree*, 2023) classifiers; our novel machine learning models include Cat Boost (*catboost.CatBoostClassifier*, 2023), Gradient Boost (*sklearn.ensemble.GradientBoostingClassifier*, 2023) (as used in Malik et al. 2021), XGB Boost (*xgboost.XGBClassifier*, 2023) classifiers; our deep learning model is LSTM (*tensorflow.keras.wrappers. scikit_learn.KerasClassifier*, 2023).

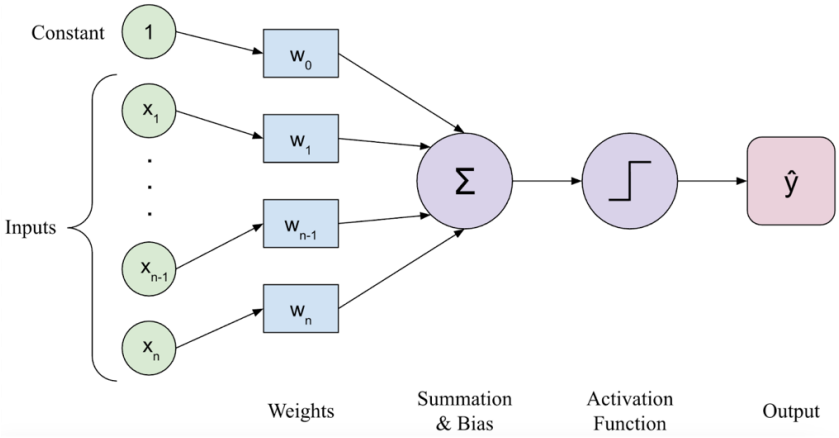

Figure 10. *A diagram representing the overall classification process of a Perceptron model. The Perceptron receives a number of input signals, and it either generates a signal or produces no output at all if the total of the input signals is greater than a predetermined threshold.*

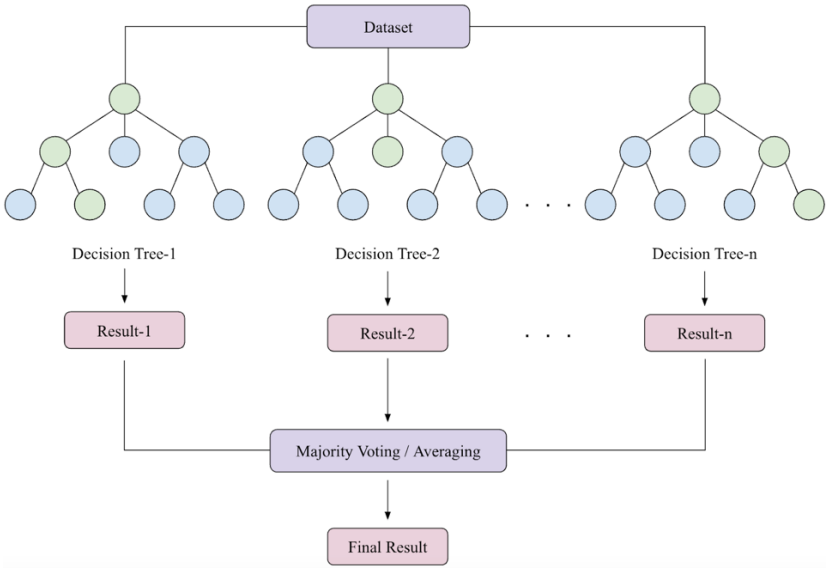

Figure 11. *A diagram representing the overall classification process of a Random Forest Classifier (consisting of multiple branches of Decision Tree Classifiers). The Decision Tree Classifier learns by creating nodes and categorizing features of different classes until it finds a path that is distinct to each of its classes.*

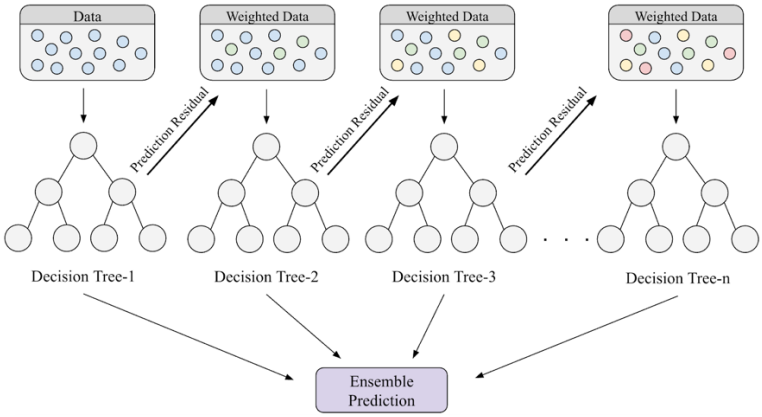

Figure 12. *A diagram representing the overall classification process of an XGB. In this classifier, a subset of the data is used to train each decision tree and the aggregated forecasts of all the trees yield the final prediction.*

3.5 Model Training

To train each of our models, we used the training dataset that we finished assessing at the data processing stage. Each model that we constructed in the previous stage trains from a given data differently (as seen in Figure 10., Figure 11., and Figure 12.), so we created pipelines with different qualities to maximize each model's performance. For example, we used a max depth (the maximum number of subdivisions created) of eight for our Decision Tree classifier but six for our Gradient Boost, because this led to their respective optimal performances.

3.6 Model Testing

To test each of our models, we used the testing dataset that we finished assessing at the data processing stage. To measure the quality of each model and its training process, we prompted several modes of diagnosis explained in the following section.

4. RESULTS/CONCLUSION

Upon refining each of our machine learning and deep learning algorithms, we produced some very high performing models. To evaluate the quality of our models, we used several metrics including accuracy score (sklearn.metrics.accuracy_score), balanced accuracy score (sklearn.metrics. balanced_accuracy_score), and classification report (*sklearn.metrics.classification_report*, 2023) and graphic diagnostic tools such as ROC

(Receiver Operating Characteristic) curves (*sklearn.metrics.roc_curve*, 2023) and Precision-Recall curves (*sklearn.metrics.precision_recall_curve*, 2023).

4.1 Numerical Metrics

Each numerical metric is represented as a decimal between 0.00 to 1.00 with 1.00 representing 100% accuracy in each category. Accuracy, balanced accuracy, precision, and recall are all commonly used metrics in machine learning. The f1 score combines the precision and recall metrics by applying their harmonic mean for each class. Unlike an arithmetic mean, the harmonic mean ensures a well-adjusted scale of measurement, as the weight of a smaller value will more significantly decrease the overall value. The macro average f1 score is the arithmetic mean of the combinations of f1 scores. For our program, we deemed that the macro average f1 score would take precedence in determining the performance of each model. This is because our data and other raw exoplanet program datasets have a major imbalance in the cases per class, and macro average f1 scores take this imbalance into account by treating binary classes equally (0.5:0.5) regardless of their support values.

In Table 2., Table 3., and Table 4., each model is ranked in order of the overall lowest to highest performing metrics from top to bottom within each of the three categories: generic machine learning models, novel machine learning models, and deep learning models. Our best performing models across these categories were the Decision Tree Classifier and the XGB Classifier. Both algorithms had a 1.00 score in every metric, classifying every exoplanet and non-exoplanet of our testing dataset correctly. The Random Forest Classifier, Gradient Boost Classifier, and LSTM model were also highly performing, having macro average f1 scores of 0.95 or higher. The cat boost classifier and logistic regression classifier were also well performing. The only algorithm that did not show high execution was the perceptron algorithm.

Table 2. *Accuracy, balanced accuracy, and classification report of generic machine learning algorithms*

	Accuracy Score	Balanced Accuracy Score	Class 0 Precision	Class 1 Precision	Class 0 Recall	Class 1 Recall	Class 0 f1 score	Class 1 f1 score	Macro Average f1 Score
Perceptron Algorithm	0.909	0.954	1.00	0.09	0.91	1.00	0.95	0.16	0.56
Logistic Regression	0.997	0.995	1.00	0.62	0.99	1.00	1.00	0.77	0.88
Random Forest Classifier	1.00	0.999	1.00	0.83	1.00	1.00	1.00	0.91	0.95
Decision Tree Classifier	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table 3. *Accuracy, balanced accuracy, and classification report of novel machine learning algorithms*

	Accuracy Score	Balanced Accuracy Score	Class 0 Precision	Class 1 Precision	Class 0 Recall	Class 1 Recall	Class 0 f1 score	Class 1 f1 score	Macro Average f1 Score
Cat Boost Classifier	0.996	0.998	1.00	0.71	1.00	1.00	1.00	0.83	0.92
Gradient Boost Classifier	1.00	0.999	1.00	0.83	1.00	1.00	1.00	0.91	0.95
XGB Classifier	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table 4. *Accuracy, balanced accuracy, and classification report of deep learning algorithms*

	Accuracy Score	Balanced Accuracy Score	Class 0 Precision	Class 1 Precision	Class 0 Recall	Class 1 Recall	Class 0 f1 score	Class 1 f1 score	Macro Average f1 Score
LSTM	0.98	0.96	0.98	0.97	0.99	0.93	0.99	0.95	0.97

4.2 Graphical Analysis

Figure 13. represents ROC curves of three generic machine learning algorithms with performance differences: Decision Tree Classifier, Logistic Regression model, and Perceptron algorithm. The Decision Tree Classifier has the best performing ROC curve output as it has the highest recall to false-positive rate. It also has the highest AUC value of 1.00. The Perceptron algorithm has the lowest performing ROC curve output as it has the lowest recall to false-positive rate. It also has the lowest AUC value of 0.95. The ROC curve and AUC values helped us to align our understanding of the performance of models with the numerical metrics. However, when considering models such as the logistic regression, these metrics were not enough to make a key distinction between itself and the Decision Tree Classifier, as the ROC curves vary by a fraction and the AUC value varies by 0.01. To further analyze the performance of well performing models and create distinctions, we used the Precision-Recall curves.

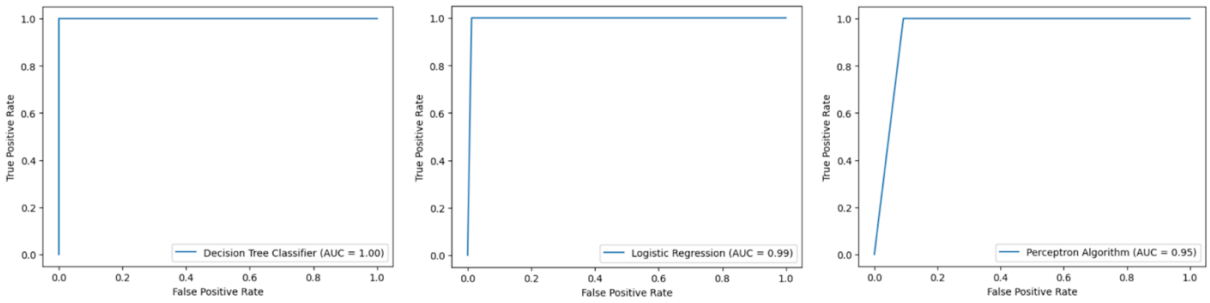

Figure 13. *The ROC curves for three generic machine learning algorithms. Each ROC curve produces its AUC (Area Under the ROC Curve)—a two dimensional area which acts as the aggregate measure of performance.*

Figure 14. represents the Precision-Recall curves for the Decision Tree Classifier and the Logistic Regression model. Although there was no major visible distinction between the two models' ROC curves, there are key differences between their Precision-Recall curves. Using the same principles to estimate the performance of a model using the ROC curve and its AUC, it is clear that the performance of the Decision Tree Classifier is higher than the performance of the Logistic Regression model.

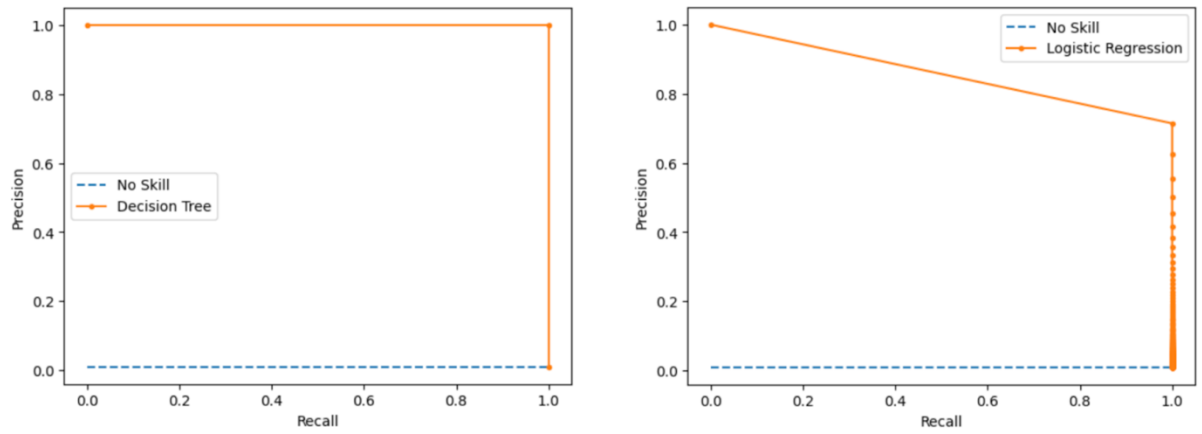

Figure 14. *The Precision-Recall curves for two of the well performing generic machine learning algorithms. Precision-Recall curves are predominantly used for datasets with a high imbalance between classes because precision does not consider true negatives. The no skill graph represents a Precision-Recall curve of a model that has not been trained and would predict a random or constant class for every case.*

4.3 Conclusion

Upon using diagnostic tools such as numerical metrics and graphical analysis, we concluded that our best performing models were the Decision Tree Classifier (a generic machine learning model) and the XGB Classifier (a novel machine learning model); both models had a value of 1.00 for every metric (accuracy score, balanced accuracy score, and classification report) and each AUC value (for the ROC and Precision-Recall curves). So, both the Decision Tree Classifier and XGB Classifier were able to identify every exoplanet and non-exoplanet correctly.

5. FUTURE WORKS

Our research was able to create an extremely accurate model with more fundamental models (compared to our related works). Therefore, we were successful in achieving our goal of creating a precise algorithm that is computationally less expensive. Due to our model's low computational power and use of general processors, we may be able to directly use these pipelines within the telescope as it gathers information, opening doors to new exploration techniques with this novel form of data collection.

Furthermore, although we worked on a relatively large portion of the Kepler dataset, we would like to train and test our algorithms on an even larger dataset (the complete Kepler dataset) in future investigations. We would also like to create a processing system that is compatible with multiple telescope and data formats, including SuperWasp and TESS. We would also like to add to this enhanced program by making it compatible for future Exoplanet Program explorations that are yet to launch

.

6. REFERENCES

decision trees. (n.d.). Scikit-Learn. Retrieved September 24, 2023, from https://scikit-learn.org/stable/modules/tree.html

Administrator, N. (2013, June 7). Five things about kepler. NASA. https://www.nasa.gov/mission_pages/kepler/news/keplerf-20090305.html

CatBoostClassifier. (n.d.). CatBoost. Retrieved September 24, 2023, from https://catboost.ai/en/docs/concepts/python-reference_catboostclassifier

Cui, Kaiming, Liu, Junjie, Feng, Fabo, Liu, & Jifeng. (n.d.). Identify light-curve signals with deep learning based object detection algorithm. I. Transit detection. The Astronomical Journal, 163(1). https://doi.org/10.3847/1538-3881/ac3482

Historic timeline. (n.d.). Exoplanet Exploration: Planets Beyond Our Solar System. Retrieved September 24, 2023, from https://exoplanets.nasa.gov/alien-worlds/historic-timeline/#first-exoplanets-discovered

How many exoplanets are there? (n.d.). Exoplanet Exploration: Planets Beyond Our Solar System. Retrieved September 24, 2023, from https://exoplanets.nasa.gov/faq/6/how-many-exoplanets-are-there/

KEPLER. (n.d.). MAST. Retrieved September 24, 2023, from https://archive.stsci.edu/missions-and-data/kepler

Kepler Space Telescope. (n.d.). Exoplanet Exploration: Planets Beyond Our Solar System. Retrieved September 24, 2023, from https://exoplanets.nasa.gov/keplerscience/

Malik, A., Moster, B. P., & Obermeier, C. (2020, November 28). Exoplanet Detection using Machine Learning. arXiv.Org. https://arxiv.org/abs/2011.14135

NASA. (2015, April 14). Mission overview. NASA. https://www.nasa.gov/mission_pages/kepler/overview/index.html

Python API Reference — xgboost 2.0.0 documentation. (n.d.). Retrieved September 24, 2023, from https://xgboost.readthedocs.io/en/stable/python/python_api.html

scipy.fftpack.fft — SciPy v1.11.2 Manual. (n.d.). Retrieved September 24, 2023, from https://docs.scipy.org/doc/scipy/reference/generated/scipy.fftpack.fft.html

scipy.ndimage.gaussian_filter — SciPy v1.11.2 Manual. (n.d.). Retrieved September 24, 2023, from https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.gaussian_filter.html

sklearn.ensemble.GradientBoostingClassifier. (n.d.). Scikit-Learn. Retrieved September 24, 2023, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

sklearn.ensemble.RandomForestClassifier. (n.d.). Scikit-Learn. Retrieved September 24, 2023, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

sklearn.linear_model.LogisticRegression. (n.d.). Scikit-Learn. Retrieved September 24, 2023, from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

sklearn.linear_model.Perceptron. (n.d.). Scikit-Learn. Retrieved September 24, 2023, from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Perceptron.html

sklearn.metrics.classification_report. (n.d.). Scikit-Learn. Retrieved September 24, 2023, from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html

sklearn.metrics.precision_recall_curve. (n.d.). Scikit-Learn. Retrieved September 24, 2023, from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_curve.html

sklearn.metrics.roc_curve. (n.d.). Scikit-Learn. Retrieved September 24, 2023, from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html

sklearn.preprocessing.normalize. (n.d.). Scikit-Learn. Retrieved September 24, 2023, from https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.normalize.html

sklearn.preprocessing.StandardScaler. (n.d.). Scikit-Learn. Retrieved September 24, 2023, from https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

SMOTE — version 0.11.0. (n.d.). Retrieved September 24, 2023, from https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html

tf.keras.layers.LSTM. (n.d.). TensorFlow. Retrieved September 24, 2023, from https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM

Tiensuu, J., Linderholm, M., Dreborg, S., & Örn, F. (2019, January 1). Detecting exoplanets with machine learning: A comparative study between convolutional neural networks and support vector machines. DIVA. https://uu.diva-portal.org/smash/record.jsf?pid=diva2%3A1325376&dswid=-1776

Why do scientists search for exoplanets? Here are 7 reasons. (2019, November 19). Exoplanet Exploration: Planets Beyond Our Solar System. https://exoplanets.nasa.gov/news/1610/why-do-scientists-search-for-exoplanets-here-are-7-reasons/