




## Article

# Assessment of Ensemble-Based Machine Learning Algorithms for Exoplanet Identification

Thiago S. F. Luz <sup>1</sup>, Rodrigo A. S. Braga <sup>1</sup> and Enio R. Ribeiro <sup>2,\*</sup><sup>1</sup> Institute of Technological Sciences, Federal University of Itajuba, Itajuba 7500-903, Brazil; thiagosfluz@unifei.edu.br (T.S.F.L.); rodrigobraga@unifei.edu.br (R.A.S.B.)<sup>2</sup> Institute of System Engineering and Information Technology, Federal University of Itajuba, Itajuba 37500-903, Brazil

\* Correspondence: enio.k@unifei.edu.br; Tel.: +55-35-3629-1195

**Abstract:** This paper presents a comprehensive assessment procedure for evaluating Ensemble-based Machine Learning algorithms in the context of exoplanet classification. Each of the algorithm hyperparameter values were tuned. Deployments were carried out using the cross-validation method. Performance metrics, including accuracy, sensitivity, specificity, precision, and F1 score, were evaluated using confusion matrices generated from each implementation. Machine Learning (ML) algorithms were trained and used to identify exoplanet data. Most of the current research deals with traditional ML algorithms for this purpose. The Ensemble algorithm is another type of ML technique that combines the prediction performance of two or more algorithms to obtain an improved final prediction. Few studies have applied Ensemble algorithms to predict exoplanets. To the best of our knowledge, no paper that has exclusively assessed Ensemble algorithms exists, highlighting a significant gap in the literature about the potential of Ensemble methods. Five Ensemble algorithms were evaluated in this paper: Adaboost, Random Forest, Stacking, Random Subspace Method, and Extremely Randomized Trees. They achieved an average performance of more than 80% in all metrics. The results underscore the substantial benefits of fine tuning hyperparameters to enhance predictive performance. The Stacking algorithm achieved a higher performance than the other algorithms. This aspect is discussed in this paper. The results of this work show that it is worth increasing the use of Ensemble algorithms to improve exoplanet identification.



**Citation:** Luz, T.S.F.; Braga, R.A.S.; Ribeiro, E.R. Assessment of Ensemble-Based Machine Learning Algorithms for Exoplanet Identification. *Electronics* **2024**, *13*, 3950. <https://doi.org/10.3390/electronics13193950>

Academic Editor: Kefeng Ji

Received: 21 July 2024

Revised: 5 October 2024

Accepted: 5 October 2024

Published: 7 October 2024



**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

**Keywords:** Ensemble algorithms; machine learning; cross-validation; exoplanet prediction; Kepler object of interest; confusion matrix

## 1. Introduction

Exoplanets are planets outside of our solar system. The discovery of exoplanets has been an active field of research in recent decades and has leveraged the development of various techniques to detect and characterize these distant planets. In 1992, the first exoplanet was discovered in a closed orbit around a pulsar [1]. Among the exoplanet detection methods were the following: detection by radial velocity, host star influence (astrometry), and the transit method. The radial velocity technique detects a planet through the star's radial velocity relative to the planetary system's rotation at its center of mass. In astrometry, it was discovered that it would be possible to detect an exoplanet via the influence of its host star on its orbit and luminosity. The transit method detects exoplanets by identifying the periodic variations in the star's light as the planet passes in front of the star during its orbit [2].

In 2009, NASA launched the Kepler telescope in a region of space with more than 200,000 stars, aiming to detect exoplanets. From Kepler's data, over 1000 candidates for exoplanets were detected using the transit method. These exoplanets underwent analysis to verify whether they would indeed be confirmed exoplanets or false-positive

data originating from different sources, such as binary eclipses or background binary eclipses. The dataset created by the Kepler program is known as the Kepler Object of Interest (KOI) [3].

In the past, exoplanet detection relied on manual methods. Currently, researchers are using Machine Learning algorithms due to the advancements in artificial intelligence [4]. These algorithms are capable of analyzing datasets faster than manual analysis [5]. Supervised classification is a commonly used Machine Learning technique for dataset analysis. It can identify and classify data according to their respective classes [6]. Ensemble algorithms, among others, are a type of classification algorithm. They combine predictions from multiple algorithms to improve their predictive performance [7].

Many fields, such as medicine, finance, and pattern recognition, have used Ensemble algorithms. For example, Farooq et al. [8] compared Machine Learning algorithms used for predictive maintenance of ball bearing systems. They mentioned these algorithms enhance the efficiency of modern industrial systems. In that paper's results, the Ensemble algorithm Extreme Gradient Boosting achieved the highest prediction accuracy of 96.61%. In spite of that, the application of Ensemble algorithms for exoplanet detection is still scarce. Several scientific sources support this observation. In a study published by Nigri, E. and Arandjelovic, O. [9], the researchers only used one Ensemble algorithm. The others were non-Ensemble. In this paper, the Ensemble algorithm achieved the best performance. Nonetheless, other researchers have not mentioned the Ensemble algorithms as a viable approach for identifying exoplanets. Fluke, C. and Jacobs, C. [10] published a literature review on the artificial intelligence used in astronomy. They identified the five most typical algorithms for exoplanet detection. However, none of them were Ensemble algorithms. Schanche et al. [11] compared different Machine Learning algorithms for exoplanet detection. They provided a comprehensive overview of the techniques used. However, they did not mention Ensemble algorithms as an approach that has been explored in the scientific literature related to exoplanet identification.

The scarcity of research using Ensemble algorithms in exoplanet detection is a relevant issue. These algorithms could improve the accuracy and reliability of the results. Considering the complexity of astronomical data, combining multiple algorithms could reduce the challenges faced using traditional methods. Therefore, it is evident that there is a need for further research and studies that explore the use of Ensemble algorithms for exoplanet identification.

Even though the use of the Ensemble algorithm is still scarce, some papers have shown that the Ensemble algorithm's predictive performance is superior to traditional algorithms in exoplanet databases. Priyadarshini et al. [4] described the use of an Ensemble technique for a neural network. Their paper's results showed that the Ensemble algorithm achieved a better prediction performance than the traditional algorithms. Similarly, Bhamare et al. [12] compared the performance of four algorithms to predict exoplanets. The two types of algorithms used were Ensemble and two non-Ensemble. The article results revealed the Ensemble algorithm Adaboost achieved the best predictive performance.

This paper aims to conduct a comprehensive assessment and comparative analysis of five Ensemble algorithms—Adaboost, Random Forest, Stacking, Random Subspace Method, and Extremely Randomized Trees—focusing on their effectiveness in classifying exoplanets from the KOI dataset. The primary goal was to identify which algorithm performs best in terms of exoplanet identification. To achieve this, we implemented each algorithm using cross-validation, followed by the generation of confusion matrices to capture both correct classifications and errors. From these results, we computed key performance metrics, including accuracy, sensitivity, specificity, precision, and F1 score, to evaluate and compare the strengths and weaknesses of each algorithm.

## 2. KOI Dataset

The exoplanet dataset used in this paper was the KOI. The NASA Exoplanet Science Institute (NExSci) developed this dataset. It accumulates information on confirmed exo-

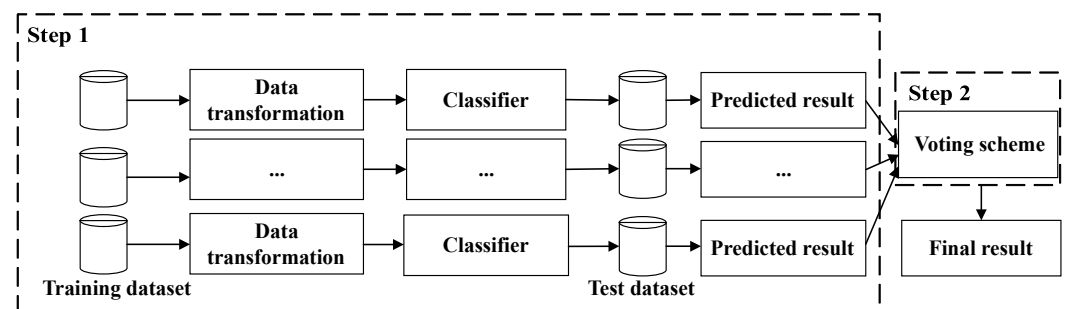
planets, candidate exoplanets for further analysis, and false-positive data. The false-positive data have patterns similar to exoplanets. However, they originate from other sources, such as eclipses. The dataset is publicly available for research and provides some tools for data analysis. The dataset contains fifty columns and 9.654 rows. Some features of this dataset include the following: stellar parameters (position, magnitude, and temperature), exoplanet parameters (mass and orbital information), and categorization of the data (light curve or radial velocity curve) [13].

### 3. Ensemble Classification Algorithms

The fundamental concept of the Ensemble algorithm is to combine multiple algorithms. This combination compensates a wrong prediction from an algorithm with a correct prediction from another. Therefore, the final prediction performance of the Ensemble algorithm will also be better than that of a single algorithm named inductor [7].

Figure 1 illustrates the concept of a typical Ensemble model for classification, comprising two steps [14]:

1. Classifying data using classification algorithms. These algorithms are called classifiers.
2. Unifying multiple prediction results from the classifiers into a single function and achieving the overall result with a voting scheme.



**Figure 1.** Ensemble algorithm concept for predicting data.

Five Ensemble algorithms from the Python library scikit-learn were selected for study based on their diversity in three characteristics: fusion method, dependency, and training approach [7]. The chosen algorithms were Adaboost, Random Forest, Stacking, Random Subspace Method, and Extremely Randomized Trees. They are described in Table 1.

**Table 1.** Characteristics of the Ensemble algorithms.

Algorithm	Fusion Method	Dependency	Training Approach
Adaboost	Weighting	Dependent	Input manipulation
Random Forest	Weighting	Independent	hybridization
Stacking	Weighting	Independent	hybridization
Random Subspace Method	Weighting	Independent	Partitioning
Extremely Randomized Trees	Meta-learning	Independent	Manipulated learning

During the development of an Ensemble Algorithm, it is necessary to consider diversity and predictive performance. Diversity means to unite many inductor models to predict the data with high performance. To include inductors in the Ensemble model, one must consider the following approaches:

- Input manipulation: Each base inductor is fitted using different subsets of the data.
- Manipulated learning: The use of each inductor is constantly changed. The way the base inductor searches the optimal solution under the hypotheses spaces is manipulated.
- Partitioning: The dataset can be split into smaller subsets, where each of them will be used to train a specific inductor.

- Hybridization: To join at least two strategies during the construction of the Ensemble algorithm.

### 3.1. Adaboost

Adaboost is the most well-known Ensemble classification algorithm. The Adaboost structure relies on dependency. Therefore, the result of each inductor prediction consequently influences the construction of the next one. The central concept of Adaboost is to focus on previously misclassified instances during the new inducer training.

The first iteration of the algorithm applies an equal weight value to all instances. In the following iterations, each misclassified instance will receive a higher weight value than the correctly classified instances. Furthermore, weights are assigned to the inductors according to their overall predictive performance [7].

### 3.2. Random Forest

The most popular Ensemble algorithm was presented and independently developed by Ho, T. [15] and Amit, Y. and Geman, D. [16]. This algorithm was named Random Forest. Later, this algorithm was modified and became popular via the work of [17]. Random Forest's popularity continues to increase due to its simplicity and predictive performance. Random Forest is also considered an easy method to modify compared to other methods [7].

The fundamental concept of the Random Forest model is to classify the data using many uncorrelated decision trees because of their randomness. Predictions from decision trees have higher performance when they are connected rather than individually. Besides being trained on different datasets, decision trees use specific variables for decision making [18].

### 3.3. Stacking

Stacking, or stacked generalization, uses a meta-learning algorithm to learn the best approach to merge the prediction from three or more Machine Learning algorithms. Stacking's primary characteristic is to use the main capacities of various algorithms. The predictions of each algorithm become a new training dataset and are added to a new layer named meta-model in the Stacking process [19].

### 3.4. Random Subspace Method

Ho, T. [20] introduced the Random Subspace Method algorithm and developed it to build decision trees. This algorithm analyzes the entire dataset with a high dimension of variables, randomly creates sets of samples of low dimensionality (i.e., a low number of variables, which, in the literature, is called subspace), and builds a classifier. In the next step, a combination rule is applied for the final decision. The Random Subspace Method is a simple and popular Ensemble method. However, using random samples from a high-dimensional dataset is impractical. Subspace random sampling is time consuming as it must ensure a sufficient and exact number of samples from a high-dimensional dataset, leading to considerable computational effort [21].

### 3.5. Extremely Randomized Trees

The Extremely Randomized Trees or Extra Trees algorithm was proposed by [22]. The algorithm training process is conducted randomly. Besides adding the best attribute among a random subset of variables, variable split points are also randomly created during training [7]. Likewise, for each variable (randomly selected in each interior of a node), a maximum discretization point (cut point) is randomly selected for splitting instead of choosing the best-cut point based on the local sample [23].

## 4. Methodology for Evaluating the Algorithms

All algorithms were trained and tested on the online platform Kaggle, which provides hardware with the following configuration: 30 Gb of RAM and CPU with four cores.

The algorithms were evaluated according to some performance measurement metrics mentioned by Tharwat, A. [24]. A confusion matrix was created with the following values: TP (true positive) exoplanets confirmed and classified as confirmed, TN (true negative) exoplanet candidates and classified as candidates, FP (false positive) exoplanet candidates but classified as confirmed, and FN (false negative) exoplanets confirmed but classified as candidates. The result from the confusion matrix allowed the following metrics to be calculated and analyzed, where the results of Equations (1) to (5) are multiplied by 100 and are shown in percentages (%):

Accuracy (Acc): This is the percentage of data predicted correctly. Equation (1) is the ratio between the samples correctly classified and the total number of data samples.

$$Acc = \frac{(TP + TN)}{(TP + TN + FP + FN)} \quad (1)$$

Sensitivity (Sen): This is the percentage of data predicted correctly that belongs to positive classes. It is also called recall. Equation (2) is the ratio between positive samples classified correctly and the total number of positive samples.

$$Sen = \frac{(TP)}{(TP + FN)} \quad (2)$$

Specificity (Spec): This is the percentage of data predicted correctly that belongs to negative classes. Equation (3) is the ratio between negative samples classified correctly and the sum of negative samples classified correctly and incorrectly. It is similar to sensitivity but with a focus on negative classes.

$$Spec = \frac{(TN)}{(TN + FP)} \quad (3)$$

Precision (Prec): This is the percentage of correct positive predictions made. Equation (4) is the ratio between correct classified positive data and the sum of correct and incorrect classified positive data.

$$Prec = \frac{(TP)}{(TP + FP)} \quad (4)$$

F1 score (F1): This is the harmonic mean between precision and specificity. The F1 Score is expressed by Equation (5).

$$F1 = \frac{2TP}{(2TP + FP + FN)} \quad (5)$$

Time consumption: Experimentally, this is the time consumed by each algorithm execution and is measured in seconds. Equation (6) is the difference between the end and start of the algorithm execution.

$$TimeElapsed = EndofExecution - StartofExecution. \quad (6)$$

#### 4.1. Data Preprocessing

The KOI file “cumulative.csv” was retrieved from the electronic site [25]. The untreated dataset contained 9564 rows and 50 columns. For data preprocessing, the following steps were followed.

1. Removing columns: Six columns were selectively removed due to their lack of substantive contribution to the predictive modeling. These columns were found to serve solely as identifiers, devoid of significant relevance to the prediction task at hand.
  - rowid: A sequential number that identifies each row. A primary key of the dataset;
  - kepid: A unique random number that identifies the possible exoplanet;

- *kepoi\_name*: A second unique random number that identifies the possible exoplanet;
- *kepler\_name*: Textual data naming the exoplanet;
- *koi\_pdisposition*: Textual data that provide the best possible explanation of the data origin. Since the KOI had another column named *koi\_disposition*, it was necessary to remove this column so as to not intervene in the prediction;
- *koi\_score*: A value between 0 and 1 that indicates the confidence in the KOI disposition.

The following columns “*koi\_teq\_err1*” and “*koi\_teq\_err2*” were also removed because they were completely empty.

2. Limiting the values of the target column: Only the values *candidate* and *confirmed* were left. Given the presence of false-positive data within the column, it was deemed essential to focus solely on instances classified as either candidate or confirmed exoplanets. Accordingly, rows containing false-positive values were removed from the dataset to ensure the accuracy and integrity of the assessment.
3. Transforming the target column “*koi\_disposition*” into binary values: The target column underwent transformation into binary values. Given that this column originally comprised two distinct string values, i.e., *candidate* or *confirmed*, a binary encoding was applied. Specifically, a value of 1 was assigned to instances denoted as candidate exoplanets, while a value of 0 corresponded to confirmed exoplanets. This approach ensured the suitability of the data for subsequent analytical procedures.
4. Replacing missing values in the “*koi\_tce\_delivname*” column: The missing entries were replaced with the mean of all values present within this column. Given that this column primarily consisted of categorical variables, a transformation utilizing dummy variables was undertaken to transform the data into numerical form.
5. Splitting the dataset into two parts: X (independent variables) and y (which was the only target column “*koi\_disposition*”) (dependent variable).
6. Submitting the training and test data to a scaling function *StandardScaler*, which normalized the data within the same range of values [26].
7. Creating two sets of data: 70% of the data were randomly split for training and 30% for testing.
8. Creating a seed to generate the same random number every time to ensure the algorithm always ran on the same random numbers.

After data treatment, the training dataset had 3178 rows and 43 columns. The target column “*koi\_disposition*” contained 1589 confirmed exoplanets and 1589 candidates.

#### 4.2. Dataset Training and Testing

To mitigate overfitting, this study utilized the cross-validation k-fold technique outlined by Raschka, S. [27]. This approach involves dividing the dataset into 10 subsets and iteratively selecting each subset as a validation set while using the remaining subsets for training. This process was iterated 5 times to ensure reliability. By averaging the performance metrics from each fold, including accuracy, sensitivity, precision, specificity, and F1 score, a comprehensive assessment of the model’s performance was obtained.

#### 4.3. Implementation of the Ensemble Algorithms

Each Ensemble algorithm was implemented at least two times. During the first implementation, the hyperparameter values were kept as the initial. However, in the second implementation, the Grid Search function in Python was responsible for tuning the algorithm’s hyperparameters. For each algorithm, some important hyperparameters were chosen; they are described in Section 5. The Grid Search tested a configuration that yielded the highest accuracy on the prediction process [28]. Afterward, these new values were passed to their respective hyperparameters to predict the exoplanets. A confusion matrix demonstrated at each implementation the number of correct and incorrect predictions of the Ensemble algorithms.



## 5. Results

### 5.1. Adaboost Implementation

In the first implementation, the algorithm Adaboost managed to correctly predict 502 confirmed exoplanets (TP) and 599 candidate exoplanets (TN). The algorithm achieved an accuracy value of 81.37%. Adaboost has only two hyperparameters: estimator number and learning rate. Figure 2a shows the confusion matrix with the obtained results. These hyperparameters had, respectively, the following initial values: 50 and 1.0. The algorithm consumed 88 seconds to predict the result.

		Predicted Value	
		Positive	Negative
Actual Value	Positive	TP 502 36.83%	FN 157 11.52%
	Negative	FP 105 7.70%	TN 599 43.95%

(a)

		Predicted Value	
		Positive	Negative
Actual Value	Positive	TP 516 37.86%	FN 143 10.49%
	Negative	FP 105 7.70%	TN 599 43.95%

(b)

**Figure 2.** Confusion matrix with the (a) initial and (b) improved hyperparameter values for Adaboost.

In the second implementation, the hyperparameter values were modified in the following way: the number of estimators was changed from 50 to 974, and the learning rate was changed from 1.0 to 0.1. After changing the values for the hyperparameters, the percentage of Adaboost's accuracy increased, reaching 82.52% when compared to the first implementation. Therefore, the second implementation prediction accuracy was 1.15% higher than the first Adaboost implementation.

The Adaboost algorithm correctly predicted 516 confirmed exoplanets (TP) and 599 candidate exoplanets (TN). This result shows that the algorithm is more efficient when the value of the hyperparameter numbers of the estimators is greater than the initial value. However, regarding the learning rate hyperparameter, the algorithm obtained a better result when it was lower than its initial value. The algorithm took 1627 seconds to predict the result. When compared with previous executions, it took eighteen times more time.

All other performance metrics, sensitivity, specificity precision, and F1 score, had a better result. For the improved hyperparameter values, Figure 2b shows the confusion matrix.

### 5.2. Random Forest Implementation

In the first implementation, the algorithm Random Forest correctly predicted 514 confirmed exoplanets (TP) and 593 candidate exoplanets (TN). The algorithm achieved an accuracy value of 82.25%. The algorithm took 176 seconds. Figure 3a shows the confusion matrix with the obtained results.

Random Forest has 18 hyperparameters and, arbitrarily, only four were selected due to their values changing. The elected hyperparameters were as follows: the number of estimators, maximum number of variables, maximum depth of trees, and *criterion*. The hyperparameter initial values were as follows: number of estimators = 100, maximum number of variables = *sqrt*, maximum depth of trees = *none*, and *criterion* = *gini*.

In the second implementation, the hyperparameter values were changed in the following way: number of estimators = 1600 and *criterion* = *entropy*. The other following hyperparameters were kept their initial values: the maximum number of variables and maximum depth of trees. The algorithm's accuracy achieved 82.64%. This meant there was

an increase of 0.34% regarding the first implementation. The results showed that increasing the number of trees contributed to a higher accuracy performance. The algorithm took 2916 s to predict the result. When compared with previous executions, it took sixteen times more time.

The sensitivity of the algorithm increased from 79.45% to 76.64% and specificity from 84.68% to 85.72%. The other performance metrics also increased when compared to the first implementation. The confusion matrix in Figure 3b shows an increase of 4 confirmed exoplanets (TP) and 12 candidate exoplanets (TN) that were correctly predicted.

		Predicted Value	
		Positive	Negative
Actual Value	Positive	TP 514 37.71%	FN 145 10.64%
	Negative	FP 111 8.14%	TN 593 43.51%

(a)

		Predicted Value	
		Positive	Negative
Actual Value	Positive	TP 518 38.00%	FN 141 10.34%
	Negative	FP 99 7.26%	TN 605 44.39%

(b)

**Figure 3.** Confusion matrix with the (a) initial and (b) improved hyperparameter values for Random Forest.

### 5.3. Stacking Implementation

In the first implementation, it was necessary to select values for the hyperparameter estimators. The reason for this was that this hyperparameter did not have initial values. Arbitrarily, two estimators (Ensemble algorithms) were chosen: Random Forest and Gradient Boosting. These two estimators were added with their hyperparameter initial values. The algorithm LogisticRegression, which is the meta modeling default, was kept. The Stacking algorithm precisely predicted 527 confirmed exoplanets (TP) and 599 candidates exoplanets (TN). The accuracy performance reached 82.72%, and it was superior to Adaboost and Random Forest. Figure 4a shows the confusion matrix with the obtained results. The algorithm consumed 2772 s.

		Predicted Value	
		Positive	Negative
Actual Value	Positive	TP 527 38.66%	FN 132 9.68%
	Negative	FP 105 7.70%	TN 599 43.95%

(a)

		Predicted Value	
		Positive	Negative
Actual Value	Positive	TP 527 38.66%	FN 132 9.68%
	Negative	FP 104 7.63%	TN 600 44.02%

(b)

**Figure 4.** Confusion matrix for the (a) first and (b) second implementation of Stacking.

In the second implementation, the LogisticRegression algorithm was still kept as a meta-model. The values of some of the hyperparameters of the estimator algorithms, i.e., Random Forest and Gradient Boosting, were changed. The hyperparameters changed



in Random Forest were the same as those configured previously, that is, the number of estimators = 1600, maximum number of variables = *sqrt*, maximum depth of trees = *none* and *criterion* = *entropy*. The hyperparameters that were changed in Gradient Boosting were as follows: number of estimators = 1600 and learning rate = 0.1. The accuracy achieved was 83.03%, which was an increase of 0.31% compared to the first implementation. Furthermore, all performance metrics achieved a greater result except sensitivity, which remained at 80.05%. The algorithm took 10,856 s to predict the result. When compared with previous executions, it took three times more time. The confusion matrix with the obtained results is shown in Figure 4b.

As previously mentioned, the Stacking algorithm can use two or more estimators as a base algorithm to make predictions. Thus, nine Ensemble and non-Ensemble algorithms were trained to compare their accuracy performances. This training process only happened without a cross-validation process because a faster analysis of the proposed algorithms could then be performed.

The prediction results of the trained algorithms are represented in Figure 5 in ascending order of accuracy performance. The Gradient Boosting algorithm had the best performance as an estimator among all of the other algorithms as it classified the exoplanets with an accuracy of 82.53%. Note that the non-Ensemble algorithms, Naive Bayes, K Nearest Neighbor, and Decision Tree, had the three worst results. This evidence shows that the Ensemble algorithms produced superior performances in predicting the exoplanet data even with the initial hyperparameter values.

Stacking achieved the best performance among all of the algorithms. This result may be due to Stacking's ability to utilize a meta-model layer that works with the data previously generated by other algorithms when using Stacking. Consequently, Stacking used its second meta-model layer to best combine the capabilities of each chosen algorithm in its structure.

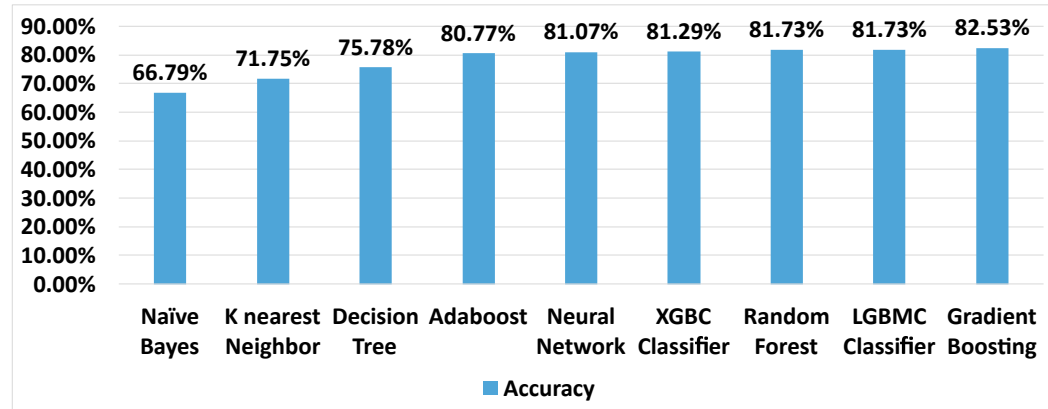


Figure 5. Accuracy results for the algorithms tested as estimators for Stacking.

From these nine tested algorithms, five combinations of estimators were created. They were based on the algorithms' accuracy performances and variation in the union of Ensemble and non-Ensemble algorithms. The proposed estimators were as follows.

1. LGBMC Classifier + Gradient Boosting: Two Ensemble algorithms with high accuracy performance;
2. Neural Network + K Nearest Neighbor + Decision Tree: Three non-Ensemble algorithms;
3. Random Forest + Gradient Boosting + LGBMC Classifier + XGBC Classifier + Adaboost: Five Ensemble algorithms;
4. Random Forest + Naive Bayes: An Ensemble and non-Ensemble algorithm;
5. Random Forest + Adaboost: Two Ensemble algorithms.

Each combination of estimators was passed to the Stacking algorithm as an estimator. Table 2 presents the performance of each estimator in the exoplanet prediction. Stacking with Estimator 1 achieved the best performance among all in terms of accuracy with 83.08%.

Also, it reached a sensitivity, specificity, precision, and F1 score that was higher than the first implementation.

**Table 2.** Prediction results of the five estimators for Stacking.

Metric	Estimator 1	Estimator 2	Estimator 3	Estimator 4	Estimator 5
Accuracy	83.08%	75.54%	82.29%	81.73%	82.24%
Sensitivity	80.18%	74.53%	79.94%	79.97%	80.13%
Specificity	85.96%	78.40%	85.79%	84.54%	84.67%
Precision	84.88%	77.21%	84.51%	83.57%	83.67%
F1 Score	82.41%	74.25%	82.12%	81.76%	81.77%

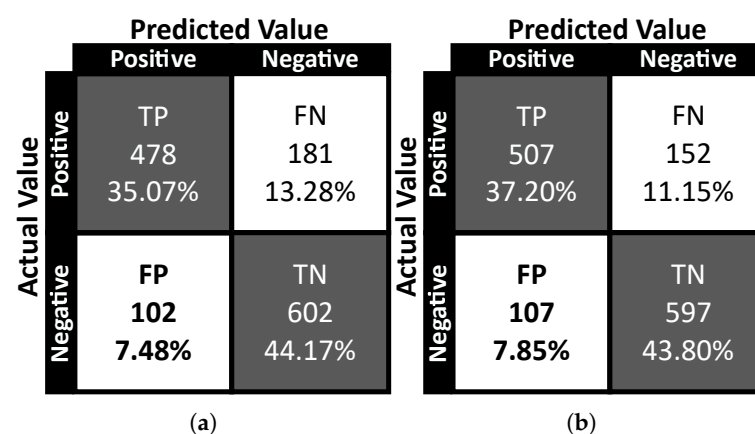
#### 5.4. Random Subspace Method Implementation

In the first implementation, the Random Subspace algorithm correctly predicted 478 confirmed exoplanets (TP) and 602 candidate exoplanets (TN). The algorithm achieved an accuracy of 80.55%. Figure 6a shows the confusion matrix with the obtained results. The algorithm took 217 s to complete its predictions.

The Random Subspace algorithm has eleven hyperparameters. Three hyperparameters were arbitrarily selected for value change. The selected hyperparameters were as follows: the number of estimators, the maximum number of samples, and the maximum number of variables. Their initial values were, respectively, 10, 1, and 1.

In the second implementation, the hyperparameter values that yielded the best results with a 81.91% accuracy were as follows: number of estimators = 1000, maximum number of samples = 0.1, and the maximum number of variables remained at 1. To increase the performance of the algorithm on data prediction, more estimators, a low sample rate, and variables were needed. The algorithm took 1312 s to predict the result. When compared with previous execution, it took six times more time.

Considering the results of all the metrics, a confusion matrix was generated, as presented in Figure 6b. The performance of the algorithm in predicting confirmed exoplanets (TP) was superior to the previous one with initial hyperparameter values. The algorithm correctly predicted 29 more confirmed exoplanets (TP). In addition, it also incorrectly predicted 5 less candidate exoplanets (TN) compared to the first implementation. The algorithm achieved a better result on all performance metrics, except for specificity, when compared to the first implementation.



**Figure 6.** Confusion matrix with the (a) initial and (b) improved hyperparameter values for the Random Subspace Method.

The Random Subspace algorithm allows for alteration of its hyperparameter estimator, which is a decision tree as its initial value, by other algorithms. Therefore, considering this aspect, the algorithm was implemented five times, each time with a different algorithm acting as an estimator. The algorithms used for this objective were Random Forest, Adaboost,

Gradient Boosting, Logistic Regression, and Stacking. The results of the Random Subspace Method estimator prediction are presented in Table 3.

**Table 3.** Prediction results with five estimators for the Random Subspace Method.

Metric	Random Forest	Adaboost	Gradient Boosting	Logistic Regression	Stacking
Accuracy	81.88%	81.18%	81.61%	76.30%	81.77%
Sensitivity	79.35%	77.54%	73.33%	66.00%	78.97%
Specificity	84.12%	84.44%	85.15%	85.93%	84.34%
Precision	83.33%	83.29%	83.71%	81.46%	83.22%
F1 Score	81.25%	80.46%	80.92%	72.92%	81.04%

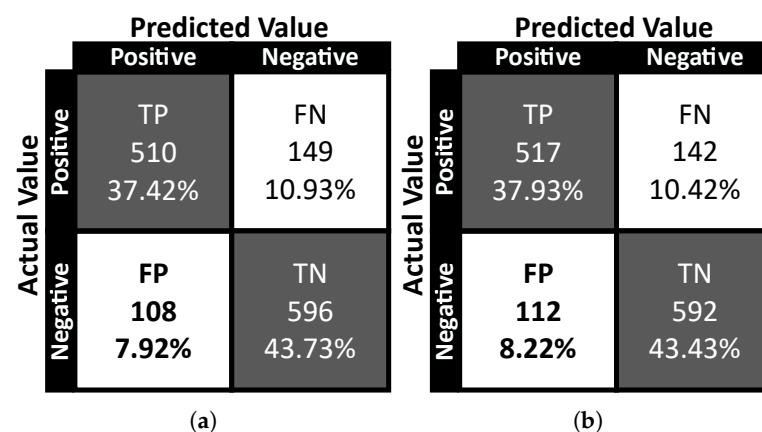
The algorithm Random Subspace Method with Random Forest acting as initial estimator obtained the best performance for accuracy with 81.88%, sensitivity with 79.35%, and F1 score with 81.25%. However, using the non-Ensemble Logistic Regression algorithm as an estimator achieved the best specificity with 85.93%. The best precision, with 83.71%, was achieved by the Random Subspace Method, with the Ensemble algorithm Gradient Boosting as an estimator. Table 4 shows the confusion matrix results of each algorithm when acting as an estimator of the Random Subspace Method.

**Table 4.** Confusion matrix results with different estimators for the Random Subspace Method.

Metric	Random Forest	Adaboost	Gradient Boosting	Logistic Regression	Stacking
True Positive (TP)	584	589	596	605	587
True Negative (TN)	514	525	510	435	522
False Positive (FP)	120	115	108	99	117
False Negative (FN)	145	134	149	224	137

### 5.5. Extremely Randomized Trees Implementations

In the first implementation, the Extremely Randomized Trees algorithm correctly predicted 510 confirmed exoplanets (TP) and 596 candidate exoplanets (TN). The algorithm achieved an 82.06% accuracy. Figure 7a shows the confusion matrix with the obtained results. The algorithm took 67 s.



**Figure 7.** Confusion matrix with the (a) initial and (b) improved hyperparameter values for Extremely Randomized Trees.

This algorithm has 18 hyperparameters. Four of them were selected arbitrarily for value modification. The selected hyperparameters were as follows: the number of estimators, maximum number of variables, maximum tree depth, and *criterion*. These hyperparameters had the following initial values 100, *sqrt*, *none*, and *gini*, respectively.

In the second implementation, the algorithm achieved a higher accuracy of 82.36% compared to the first implementation. The hyperparameter values that changed were as follows: number of estimators = 200 and *criterion* = *entropy*. The hyperparameter maximum number of variables and maximum tree depth remained at their initial values. All metrics, except precision, had a slight increase. Figure 7b exhibits the confusion matrix. Extremely Randomized Trees correctly predicted five more confirmed exoplanets (TP). However, it also predicted four less candidate exoplanets (TN) than the first implementation. The algorithm took 155 s to predict the results. When compared with the previous execution, it took two times more time.

### 5.6. Comments

Table 5 displays the results obtained by the Ensemble algorithms with their initial and improved hyperparameter values. It was observed that Stacking achieved the best accuracy and F1 score performance. Based on the previously shown results, this algorithm was the most versatile and had the highest predictive power of all. Its versatility is because of the option of using several algorithms as estimators, such as Ensemble and non-Ensemble algorithms. In contrast, the same did not happen with the other algorithms such as Random Forest, which used only decision trees as an estimator.

**Table 5.** Comparison of the Ensemble algorithms' prediction results.

	Accuracy		Sensitivity		Specificity		Precision		F1 Score	
	Initial	Improved	Initial	Improved	Initial	Improved	Initial	Improved	Initial	Improved
Adaboost	81.37%	82.52%	77.54%	78.44%	85.14%	86.51%	83.69%	85.12%	80.46%	81.61%
Random Forest	82.25%	82.64%	79.45%	79.64%	84.68%	85.72%	83.64%	84.54%	81.73%	81.91%
Stacking	82.72%	83.08%	80.05%	80.18%	85.20%	85.96%	84.31%	84.88%	82.04%	82.41%
Random Subspace Method	80.55%	81.91%	74.87%	78.60%	85.39%	85.14%	83.36%	84.04%	78.73%	81.07%
Extremely Randomized Trees	82.06%	82.36%	78.64%	78.83%	85.41%	85.68%	84.54%	84.40%	81.32%	81.51%

## 6. Discussion

In summary, the choice of the five Ensemble algorithms was based on the diversity of their operating models, which were as follows: fusion method, dependency, and training approach. This choice allowed for a result that integrated the different structures of these algorithms.

Adaboost, the first algorithm evaluated, obtained an accuracy of 81.37% with its initial hyperparameter values. In its second implementation, the technique called *hyperparameter tuning* was used. Using this technique, different values of the algorithm's hyperparameters were tested one hundred times. This process resulted in a combination of hyperparameters that produced the highest accuracy. As a result, Adaboost had its number of estimators and its learning rate altered, and, with this new configuration, the accuracy increased by over 1.1% and reached 82.52%. This result shows that the Adaboost algorithm can predict more accurately when it has more estimators in its structure.

The Random Forest algorithm, which joins decision trees, obtained the third-best accuracy among the five algorithms when implemented with its initial hyperparameter values. Random Forest achieved 82.25% of accuracy. This algorithm is simple to implement and has good predictive performance. Therefore, this algorithm is one of the most used among Ensemble algorithms. After the combination of hyperparameter values, the accuracy improved by 0.39%. Among the altered hyperparameters, the number of estimators that had their value significantly increased from 100 to 1600 decision trees were highlighted.

The Stacking algorithm, in contrast to the Random Forest algorithm, unites not only trees, but also different algorithms. Stacking achieved the best performance among all algorithms using initial hyperparameter values, reaching an accuracy of 82.72%. This result may be due to Stacking's ability to utilize a meta-model layer that works with the data previously generated by the algorithms included in Stacking. Consequently, Stacking uses its second meta-model layer to best combine the capabilities of each chosen algorithm in its

structure. In the second implementation, the value of some of the hyperparameters from the two chosen estimators of Random Forest and Gradient Boosting were changed. The Stacking algorithm reached an accuracy of 83.03%. Because of the capacity of Stacking to join two or more algorithms, five different algorithm combinations, acting as estimators, were implemented. The combination of LGBMC and Gradient Boosting algorithms, both Ensemble, produced an accuracy of 83.08%. This result was the best obtained by all of the algorithms, demonstrating that the Stacking algorithm is the most effective for exoplanet prediction.

The Random Subspace Method algorithm performed similarly to Adaboost and showed an improvement of over 1% of accuracy. This algorithm, when using the combination of hyperparameter values, achieved an accuracy of 81.91%. This algorithm has a simple structure, even though it uses decision trees as estimators and creates random samples of variables. The algorithm could not achieve 82% of accuracy even with its hyperparameter values changed.

The Extremely Randomized Trees algorithm achieved an accuracy of 82.06% in the first implementation. This algorithm had a high accuracy, and it was only lower than Stacking. In the second implementation, the accuracy only increased 0.3%. This algorithm achieved a minor improvement in accuracy. This algorithm has a random process that occurs throughout its execution stage, and this characteristic can explain this small positive result.

This study demonstrated that Ensemble algorithms have a great capacity to identify exoplanets. All algorithms managed to reach an accuracy greater than 80%. However, it is necessary to adjust hyperparameter values for a better result. The astronomy and computer science fields could use the greater predictive power of these algorithms to improve exoplanet identification. In this paper, all of the exoplanet predictions were conducted using the KOI dataset.

For future work, other datasets, such as TESS, K2, and Corot from the NASA exoplanet archive, could be used to determine if the performance of Ensemble algorithms is different on other data sources. The interpretability of the Ensemble algorithms could be deeper explored in future research to enhance the reproducibility and understanding of the decision-making process.

**Author Contributions:** Conceptualization, T.S.F.L., R.A.S.B. and E.R.R.; methodology, T.S.F.L., R.A.S.B. and E.R.R.; software, T.S.F.L. and R.A.S.B.; validation, T.S.F.L., R.A.S.B. and E.R.R.; formal analysis, T.S.F.L. and R.A.S.B.; investigation, T.S.F.L.; resources, T.S.F.L. and E.R.R.; data curation, T.S.F.L.; writing—original draft preparation, T.S.F.L.; writing—review and editing, T.S.F.L., R.A.S.B. and E.R.R.; visualization, T.S.F.L.; supervision, R.A.S.B. and E.R.R.; project administration, T.S.F.L. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received support from the Office of Research and Graduate Studies (PRPPG) and the Postgraduate Program in Electrical Engineering of the Federal University of Itajubá.

**Data Availability Statement:** The exoplanet dataset used in this research is available via the Kaggle website at <https://www.kaggle.com/datasets/nasa/kepler-exoplanet-search-results> (accessed on 15 May 2024). All the of the notebooks created to assess the Ensemble algorithms are available at <https://www.kaggle.com/thiagosalesfreireluz/code> (accessed on 15 May 2024).

**Acknowledgments:** The authors wish to thank CNPq, CAPES, Fapemig, and Unifei for their support.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Wolszczan, A.; Frail, D. A planetary system around the millisecond pulsar PSR1257+ 12. *Nature* **1992**, *355*, 145–147. [CrossRef]
2. Mason, J. *Exoplanets: Detection, Formation, Properties, Habitability*; Springer: Berlin/Heidelberg, Germany, 2008. Available online: <https://books.google.com.br/books?id=p4-BHI3tRl8C> (accessed on 1 June 2024).
3. Ofman, L.; Averbuch, A.; Shliselberg, A.; Benaun, I.; Segev, D.; Rissman, A. Automated identification of transiting exoplanet candidates in NASA Transiting Exoplanets Survey Satellite (TESS) data with machine learning methods. *New Astron.* **2022**, *91*, 101693. [CrossRef]



4. Priyadarshini, I.; Puri, V. A convolutional neural network (CNN) based ensemble algorithm for exoplanet detection. *Earth Sci. Inform.* **2021**, *14*, 735–747. [\[CrossRef\]](#)
5. Mitchell, T.M. *Machine Learning*, 1st ed.; McGraw-Hill Education: New York, NY, USA, 1997.
6. Soofi, A.; Awan, A. Classification techniques in machine learning: Applications and issues. *J. Basic Appl. Sci.* **2017**, *13*, 459–465. [\[CrossRef\]](#)
7. Sagi, O.; Rokach, L. Ensemble learning: A survey. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2018**, *8*, e1249. [\[CrossRef\]](#)
8. Farooq, U.; Ademola, M.; Shaalan, A. Comparative Analysis of Machine Learning Models for Predictive Maintenance of Ball Bearing Systems. *Electronics* **2024**, *13*, 438. [\[CrossRef\]](#)
9. Nigri, E.; Arandjelovic, O. Light Curve Analysis From Kepler Spacecraft Collected Data. In Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval (ICMR '17), Bucharest, Romania, 6–9 June 2017; pp. 93–98.
10. Fluke, C.; Jacobs, C. Surveying the reach and maturity of machine learning and artificial intelligence in astronomy. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2020**, *10*, e1349. [\[CrossRef\]](#)
11. Schanche, N.; Cameron, A.C.; Hébrard, G.; Nielsen, L.; Triaud, A.H.M.J.; Almenara, J.M.; Alsubai, K.A.; Anderson, D.R.; Armstrong, D.J.; Barros, S.C.C.; et al. Machine-learning approaches to exoplanet transit detection and candidate validation in wide-field ground-based surveys. *Mon. Not. R. Astron. Soc.* **2019**, *483*, 5534–5547. [\[CrossRef\]](#)
12. Bhamare, A.R.; Baral, A.; Agarwal, S. Analysis of kepler objects of interest using machine learning for exoplanet identification. In Proceedings of the 2021 International Conference on Intelligent Technologies (CONIT), Hubli, India, 25–27 June 2021; pp. 1–8.
13. Akeson, R.; Chen, X.; Ciardi, D.; Crane, M.; Good, J.; Harbut, M.; Jackson, E.; Kane, S.; Laity, A.; Leifer, S.; et al. The NASA exoplanet archive: Data and tools for exoplanet research. *Publ. Astron. Soc. Pac.* **2013**, *125*, 989. [\[CrossRef\]](#)
14. Dong, X.; Yu, Z.; Cao, W.; Shi, Y.; Ma, Q. A survey on ensemble learning. *Front. Comput. Sci.* **2020**, *14*, 241–258. [\[CrossRef\]](#)
15. Ho, T.K. Random Decision Forests. In Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, Canada, 14–16 August 1995; Volume 1, pp. 278–282.
16. Amit, Y.; Geman, D. Shape Quantization and Recognition with Randomized Trees. *Neural Comput.* **1997**, *9*, 1545–1588. [\[CrossRef\]](#)
17. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [\[CrossRef\]](#)
18. Tony, Y. Understanding Random Forest. Available online: <https://towardsdatascience.com/understanding-random-forest-58381e0602d2,2021> (accessed on 20 October 2023).
19. Brownlee, J. *Ensemble Learning Algorithms with Python: Make Better Predictions with Bagging, Boosting, and Stacking*, 1st ed.; Machine Learning Mastery: San Jose, Costa Rica, 2021.
20. Ho, T. The random subspace method for constructing decision forests. *IEEE Trans. Pattern Anal. Mach. Intell.* **1998**, *20*, 832–844.
21. Zhu, Y.; Liu, J.; Chen, S. Semi-random subspace method for face recognition. *Image Vis. Comput.* **2009**, *27*, 1358–1370. [\[CrossRef\]](#)
22. Geurts, P.; Ernst, D.; Wehenkel, L. Extremely randomized trees. *Mach. Learn.* **2006**, *63*, 3–42. [\[CrossRef\]](#)
23. Geurts, P.; Louppe, G. Learning to rank with extremely randomized trees. *Proc. Learn. Rank. Chall.* **2011**, PMLR *14*, 49–61.
24. Tharwat, A. Classification assessment methods. *Appl. Comput. Inform.* **2020**, *17*, 168–192. [\[CrossRef\]](#)
25. NASA Kepler Exoplanet Search Results. Available online: <https://kaggle.com/datasets/nasa/kepler-exoplanet-search-results,2023> (accessed on 17 August 2023).
26. StandarScaler Documentation. Available online: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html> (accessed on 6 September 2024).
27. Raschka, S. Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. The Computing and Research Repository. *arXiv* **2018**, arXiv:1811.12808.
28. Pearson, K.; Palafox, L.; Griffith, C. Searching for exoplanets using artificial intelligence. *Mon. Not. R. Astron. Soc.* **2017**, *474*, 478–491. [\[CrossRef\]](#)

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.