Exoplanet Habitability Prediction Model Development: A Comparative Study

Visista Jayanti[1], *Archana Pandita*[2]

[1,2]Amity University Dubai, Dubai, UAE

Abstract: Exoplanets refer to planets existing beyond our solar system. Whether or not life exists on exoplanets is debatable. Studying and predicting habitability on exoplanets helps us learn how these particular systems are formed and how they have evolved. It provides clues to understand whether life exists elsewhere in the universe. This model aims to predict habitability using multiclass classification. After the data pre-processing, feature engineering was performed using three classifiers: Random Forest Classifier, Ada Boost Classifier, and Extra Trees Classifier. Features were selected, and the model was trained on three cases. The intersection of common features from three different classifier sets was obtained, the training of the model was done on features that were obtained from the classifier that gave maximum accuracy, and lastly, the features were taken from all the combined sets. This variation in the selection of features resulted in different confusion matrices, F1 scores, precision, and accuracy scores, which were taken as parameters for comparison. Lastly, the accuracy achieved with the proposed model is compared with K-Nearest Neighbors, Multiclass Decision Tree classification, and Gradient Boosting Classification. It achieved 72.9% accuracy with Decision Tree Classification, 98.2% accuracy with K-Nearest Neighbors Classification, and 96.4% accuracy with Gradient Boosting Classification. These findings highlight the influence of feature selection and classifier choice on prediction performance, offering insights for improving habitability prediction models.

1. Introduction

An extrasolar planet or exoplanet refers to a planet that exists outside the solar system. The first evidence of exoplanets was recorded in 1971. There are multiple methods of detecting exoplanets, namely: transit photometry, Doppler spectroscopy, and so on, but due to clear observational bias, they favor the detection of planets that are located near the sun. The majority of the planets detected so far are situated within the Milky Way galaxy. However, there is evidence that extragalactic planets exist in other galaxies as well. Complex methodologies such as gravitational microlensing have been proposed to detect planets in other galaxies, involving observing the light from a distant star being magnified by the gravitational field of a massive object.[1-2]. The detection of exoplanets has magnified interest in extraterrestrial life. A planet is also known as an exoplanet if it revolves around a star other than the sun. According to NASA, the number of exoplanets as of 2022 is 5220. The measure for analyzing whether an exoplanet supports life is called “Habitability”. To determine an exoplanet's habitability, one must rely on multiple parameters. The Kepler Space Telescope launch has created a potential for the measurement of habitability on exoplanets. To predict habitability on exoplanets, the majority of scientists use the radius of the planet, density, escape velocity, and surface temperature. Machine learning algorithms use stella orbital, planetary, and atmospheric characteristics. Stellar properties like spectral type, luminosity, metallicity, and age play a crucial role in determining the radiation environment and long-term stability of habitable conditions. Additionally, stellar activity influences atmospheric retention and surface radiation levels, factors crucial for supporting life. Orbital parameters, including semi-major axis, eccentricity, and orbital period, define the planet's position within the circumstellar habitable zone and its thermal stability. Planetary features, such as mass, radius, density, and surface gravity, provide insights into surface conditions and potential atmosphere retention. Atmospheric properties, including composition, pressure, and temperature, are essential for assessing greenhouse effects and climate stability. These features collectively enable machine learning models to predict habitability by simulating complex interactions between planetary environments and their host stars.

1 Corresponding author: apandita@amityuniversity.ae

This research paper aims to devise an ML algorithm for calculating the habitability and determining whether it is 0 - not habitable, 1 - conservatively habitable, or 2 - Optimistically habitable based on features being compared to the target tuple "p_habitable". Further, the functioning of this proposed model is compared with conventional supervised machine learning algorithms such as Decision Trees Classification, KNN classification, and Gradient Boosting Classification. The most common machine learning models utilized for the predictability of exoplanet habitation are neural network models, random forest models, linear regression models, and SVM classification models. Support vector machines are supervised learning models that are used to classify exoplanets on habitability characteristics, it is used by cascading several SVM classifiers in the case of multiclass classification. [3-5]. KNN or K-Nearest Neighbors is less common compared to SVM and Random Forest. [6]. Random Forest has been used to classify exoplanets based on habitability, such as Mesoplanets, Psychroplanets, and non-habitable planets. It is viable for habitability assessment [7]. Decision trees, along with feature-building and tree-building methods, are actively used to classify the exoplanets without computing specific habitability scores. This indicates the robustness of tree-based approaches used in classification. [8]. The proposed method also uses gradient boosting machines. This enables global and local interpretation of black-box models [9]. GBM is known for its speed and accuracy. XGBoost is recognized for scalability and efficiency, and Light GBM is used for fast training performance. [10] All of these are used regularly to ensure robust performance. [11].

Machine learning can effectively categorize new exoplanet samples and improve the task of exoplanet classification [12]. ML models can automate exoplanet detection and identification with stacked Gradient boosting models. GBDT models are ML techniques used for optimizing the predictive value of a model through successive steps in the learning process. They state that gradient-boosting models outperform conventional models in transiting exoplanet detection [13]. Different ML algorithms like CART (Classification and Regression Model), Random Forest, Support Vector, and regression models, along with Naïve Bayes, can classify exoplanets based on planet characteristics and stars as potentially habitable. [14] Moreover, ML models can identify exoplanets with the following accuracies as 99.06%, 92.11%, 88.50%, and 99.79% in supervised learning tasks and reasonable clusters in unsupervised learning tasks. [15] Various research studies state that stratified K-fold cross-validation is the best method for habitability prediction, providing 99.95% accuracy in tested datasets.[16] Additionally, deep learning architecture is used for looking at habitable exoplanets based on factors such as planet gravity, eccentricity, mass radius, and other astronomical factors, an physical characteristics. [17]. UAV imagery can also be used for results that give 96% accuracy [25].

Table 1 summarizes various machine learning methodologies for classifying planetary signals, highlighting their limitations, research gaps, and overall performance. Deep learning techniques such as Convolutional Neural Networks (CNNs) trained on both real and synthetic data, achieved an accuracy of 93-95%. CNN is particularly effective because of its ability to process and analyze datasets large and diverse in size and shape. It includes changes such as those found in light intensity, which is common in exoplanet habitation detection and predictability. CNN is relevant because it possesses higher accuracy when used in ensemble models. The Ensemble CNN model outperforms conventional ML models like decision trees, support vector machines, logistic regression, and random forest classifiers in terms of factors such as accuracy and precision. Because of their ability to handle convoluted patterns in data, which is common in astronomical data, they are preferred over traditional models. However, the use of synthetic data introduces biases, necessitating robust validation with real-world datasets. The Random Forest (RF) classifier demonstrated approximately 90% accuracy on Kepler and TESS data, though its interpretability and overfitting in imbalanced datasets remain significant challenges. Self-Organizing Maps (SOM) attained 87% accuracy but struggled with high-dimensional data and lacked probabilistic outputs, indicating a need for improved scalability. Ensemble models combining RF and CNN achieved the highest accuracy of 99.6%, demonstrating the power of hybrid approaches, though at a high computational cost. The k-Nearest Neighbors (KNN) algorithm performed well for less noisy datasets with an accuracy of 86-88%, but faced efficiency issues with larger datasets. Lastly, Support Vector Machines (SVM) using Kepler data showed 90 - 91% accuracy, proving effective for well-curated datasets but struggling with multidimensional and imbalanced data. These findings highlight the need for optimization strategies to enhance generalizability, scalability, and computational efficiency in planetary classification tasks.

Table 1: Summary of Existing Work

Methodology	Limitations	Research Gap	Conclusion	References
Deep Learning (CNNs with real and synthetic data)	Synthetic data introduces potential biases and challenges in validating results with real-world data.	Need for robust validation on unseen, real-world datasets.	Achieved 93-95% Accuracy in distinguishing planetary signals. Shows the potential of combining real and synthetic data for improved model performance.	[18]
Random Forest Classifier	Limited interpretability and potential overfitting with highly imbalanced datasets.	Difficulty in generalizing results across diverse datasets.	Performed well on Kepler and TESS data, achieving approximately 90% accuracy for candidate classification.	[19]
Self-Organizing Maps (SOM)	Does not scale well with high-dimensional data and lacks a probabilistic output Framework for predictions.	Need for Improved scalability and integration with probabilistic models.	Achieved 87% accuracy using Kepler data for classifying planet candidates, though limited by its dimensional handling capabilities.	[20]
Ensemble Models (RF + CNN)	Computationally expensive, particularly for large datasets.	Explore ways to optimize computational efficiency while maintaining high accuracy.	Achieved 99.6% Accuracy for classification tasks, demonstrating the power of ensemble learning techniques.	[21]
k-Nearest Neighbors (kNN)	Sensitive to noisy data and computationally inefficient for large datasets.	Improve noise resilience and computational scalability.	Achieved 86-88% accuracy in signal classification, suitable for datasets with less noise.	[22]
SVM with Kepler Data	Limited performance in multi-dimensional data, especially with Highly imbalanced dataset	Address imbalances and improve hyperparameter optimization strategies.	Achieved 90-91% accuracy; suitable for well-curated datasets, but struggles with more complex or Imbalanced data sources.	[23]

2. Materials and Methods

2.1 Dataset Description

The dataset obtained is from the "NASA Exoplanet Archive". It contains information about astronomical and physical parameters essential for analyzing whether an exoplanet is habitable or not. Some of the example parameters include: p_name, p_status, p_mass, p_radius, s_distance, and so on. The original dataset is partially unclean and contains missing values and redundant values, which require further cleaning. The major data types in this dataset include integer, float, and object. It is a structured dataset organized into tabular rows and columns. The total sample size of this raw dataset is 9564 rows and 49 columns. Being a static dataset, it does not get updated in real-time and hence ensures constant time and space complexity. It contains both categorical values (object, text) and numerical values (integers, floats). This dataset is labeled and hence is suitable for application in supervised learning algorithms such as KNN, Gradient Boosting, and Decision Trees. The algorithm is written and executed in Python. Python libraries such as NumPy, Pandas, Matplotlib, Seaborn, Sklearn, Label Encoder, Counter, Random Forest Classifier, Ada Boost, and Extra Trees Classifier are used.

2.2 Data Preprocessing

Missing values in any dataset are defined as values that do not occur and data that are not stored for specific variables. Missing data can occur because of incomplete entry of data, malfunction of equipment and machines, lost files, and other reasons. These values are identified by using the. null (). any (). sum () function. Parameters having missing values greater than 20% have to be dropped or eliminated. Missing values in each parameter have been visualized using a heatmap visualization in Figure 1. Larger values represent brighter colors, whereas smaller values represent lighter colors. In the x-axis, parameters have been listed, whereas in the y-axis, the number of missing values has been stated. The missing values with greater than 20% have been dropped. The number of columns has been reduced to 28.

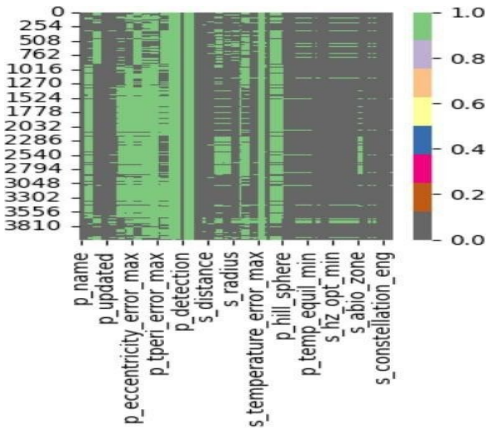

Figure 1: Heatmap representing missing values

When missing values exist in categorical values, there are multiple techniques used to replace them. Data imputation is used for replacing missing values. Data imputation is used to retain the majority of the dataset's data and information by substituting it with different values. This value is based on the orientation and nature of the data. If the distribution of the dataset is normal and missing values are random, mean imputation occurs, whereas median imputation occurs in the case of data skewness. Categorical values are converted to numerical values by using a Label Encoder. Label Encoder is used to normalize labels and to transform non-numerical (categorical) values into numerical labels. The MICE method is a statistical technique that is used to handle missing data in a dataset. It populates the missing data and then combines the results using multiple imputations to produce a final, updated, complete, imputed dataset. The imbalance in the dataset occurs when the majority class is much larger than the minority class. The MICE model imputes missing values by iteratively modelling each feature as a function of others.

A represented in Table 2, this results in no missing values, making it suitable for accurate analysis. It makes the dataset unbiased. Distribution analysis is applied, which checks the count and percentage of each class and shows if there is an imbalance in the dataset, after which SMOTEENN is applied for resampling. Resampling of data consists of taking repeated samples from the original data. This can be done in three ways. Synthetic Minority Oversampling Edited Nearest Neighbor (SMOTE-ENN) is a pre-processing approach for using unbalanced datasets and classification tasks. SMOTEENN is used for resampling in this case to maintain data integrity, and the dataset is split. SMOTE generates synthetic samples for the minority class based on the feature space and creates a better-balanced dataset. Whereas ENN assists in removing external noise in data and eliminates unnecessary points, improving data quality. After SMOTEENN is applied, a final distribution analysis is applied, which rechecks the distribution of the target variable after applying SMOTEENN. This helps in printing the new count and percentage of each class.

Table 2: Percentage of missing values in the imputed dataset

Missing Values in Imputed Dataset		
Feature	**Count**	**Percentage(%)**
p_name	0	0
p_updated	0	0
p_detection	0	0
s_name	0	0
s_alt_names	0	0
p_type	0	0
s_type_temp	0	0
s_ra_t	0	0
s_dec_t	0	0
p_type_temp	0	0
s_constellation	0	0
s_constellation_abr	0	0
s_constellation_eng	0	0

Table 3: Distribution of dataset as inhabitable, conservatively habitable and optimistically habitable

Distribution of dataset into three parameters			
Class	Count(n)	Percentage(%)	Status
Class=0	3828	32.41%	Unhabitable
Class=1	3993	33.80%	Conservatively Habitable
Class=2	3992	33.79%	Optimistically Habitable

As per Table 3, the dataset is distributed with variables 0,1, and 2 based on the prediction of habitability. Non-habitable planets are 32.4%, conservatively habitable are 33.8% and optimistically habitable are 33.79%.

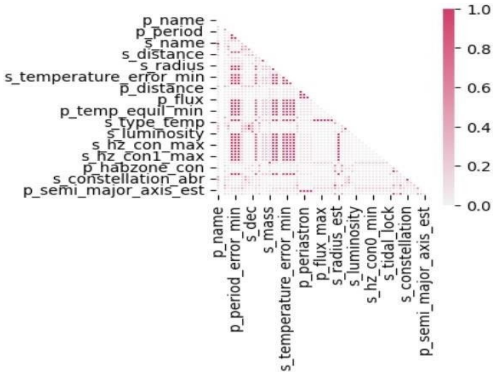

Figure 2: Correlation matrix to identify collinearity

Table 4: First seven columns of the updated dataset

Columns of the updated dataset							
Sr.No	p_name	p_status	**p_year**	**p_updated**	**p_period**	**p_detection**	**s_ra_t**
0	0	3	2007	0	326.03	7	185.179
1	1	3	2009	146	516.21	7	229.274
2	2	3	2008	0	185.84	7	352.822
3	3	3	2002	146	1773.4	7	242.601
4	4	3	1996	146	798.5	7	295.466

2.3 Correlation Matrix

The Pearson Correlation Coefficient is calculated between all pairs of numerical features in the dataset. These coefficient measures the linear relationship between the two variables in three ways based on the value, such as negative correlation (-1), no correlation (0), to positive correlation (+1). The correlation matrix is taken for all features in this dataset. High correlations (close to 1) indicate a strong relation, whereas low correlations (close to 0) indicate a weak relation or no relation. In this dataset, a criterion of 95% correlation is taken into consideration. If any two variables or attributes have more than 95% correlation, they are dropped. This is done to ensure there is reduced redundancy and no multicollinearity. Figure 2 provides a quantitative overview of the relationship between features. Since all the variables with a correlation greater than 95% were dropped. The number of columns was reduced to 28 columns, this is illustrated in Table 4, showing the first five columns of the changed dataset after removing columns.

2.4 Feature Engineering

In this research, features were selected in three cases. After resampling and imputation of the dataset were done, features were selected by using three classifiers (Random Forest Classifier, Extra Trees Classifier, AdaBoost). The features selected by these classifiers were later compared for accuracy using accuracy and F1 score as evaluation metrics. Accuracy will refer to how correctly the model predicts the outcome, whereas the F1 score is a measure of the harmonic mean of two parameters, namely recall and precision. These two metrics are taken as they are helpful for multi-class classification. F1 score joins both precision and recall in a single metric. The dataset is first split into a feature matrix and a target tuple for selecting the features. A feature matrix describes the list of columns containing independent variables that need to be processed. Target tuple, on the other hand, refers to the tuple that will be compared with these features to predict habitability. The target tuple in this research is "p_habitable," which will have a result of either 0,1,2 based on features selected after comparison. Features in the feature matrix are identified by applying three classification algorithms on the dataset to ensure distinct, most accurate features are selected. Each classifier selected a distinct subset of features based on its internal feature importance mechanisms.

2.5 Random Forest Classifier

Random Forest Classifier is a classification algorithm that is used for solving regression and multi-class classification problems. This algorithm consists of a group of decision trees, where each tree consists of a sample of data from a training set called a bootstrap sample. Random Forest Classifier is instantiated with 1000 trees, and a random state of 0 is given. Once the methods train the selector on the feature matrix, a Boolean mask indicating features to be selected is returned. Additionally, a quantitative importance score for each feature is given. This method is applied to ensure reduced dimensionality and computational efficiency. Random forest classifier improves accuracy by maintaining a minimal number of trees. [24], Table 4 shows the importance score for the variables along with a Boolean mask indicating whether these features need to be chosen or not. In this method, p_period is selected because it signifies the duration of a planet's orbit around the star, and it is important for understanding exoplanet potential habitability. s_distance helps in observational bias and detection, s_mass determines energy output of the star which affects the planetary surface conditions, s_temperature affects the habitable boundaries and climate of planet, p_distance determines whether planet falls within habitable zone,

p_type classifies if planet is terrestrial, gas giant or icy planet, thus it is directly related to habitability. p_habzone_opt is the optimal habitable zone classification, which estimates the habitable zone, p_radius_est is the estimated radius, which determines if the planet has a solid surface, and lastly, p_mass_est is the estimated mass, which indicates if there is sufficient gravity to retain an atmosphere or not. Since Random Forests focus on features that have reduced variance, the other features, including host star metallicity, observational metadata, and more, were eliminated.

Table 5: Quantitative Scores and Features Selected by Random Forest Classifier

Features Selected by RandomForestClassifier	
Features Selected	Quantitative Score
p_period	0.014
s_distance	0
s_mass	0.002204
s_temperature	0.0018766
p_distance	0.06022706
p_type	0.00860295
p_habzone_opt	0.00437218
p_radius_est	0.00901657
p_mass_est	0.00843104

2.6 Adaboost Classifier

AdaBoost Classifier is a boosted classifier. This refers to a classifier of the form where each is a weak learner that takes an object input and returns a value indicating the class of the object. AdaBoost selects features based on important weights, which are taken from the model. 50 weak learners are combined in this process, and the random state is kept at 0. It returns a Boolean array indicating which feature is selected in terms of “True” or “False”. Boolean mask “status” is used to retrieve the names of selected features. Table 6 gives a representation of which features to be selected or not, along with only three “Trues” and other “Falses”. The features selected are p_name, which groups planets based on characteristics, p_type distinguishes planets between terrestrial and non-terrestrial, whereas p_habzone_opt is the primary indicator of habitability. Since Adaboost is sensitive to noise and redundancy, a minimal subset of features is selected, consisting of only those that contribute to classification accuracy. Planetary radius, mass, and star properties are omitted in this as they may increase redundancy when “p_habzone_opt” already provides the same information.

Table 6: Representation of features to be chosen with a Boolean array by AdaBoostClassifier

Features Selected by AdaBoostClassifier	
Features Selected	Quantitative Score
p_name	7.21E-02
p_type	4.21E-01
p_habzone_opt	3.60E-01

2.7 Extra Trees Classifier

This algorithm is used for classification and regression tasks. It randomly selects a subset of features and trains them with a decision tree on them. It is then pruned to ensure only the most important features are given to make predictions. 1000 trees are given in the forest by using “estimator”, which improves the robustness of importance scores. It has a random state 0 that ensures reproducibility of results. A Boolean mask “status” is returned, where “True” indicates features that are selected and “False” indicates that they are dropped. The Boolean mask is used to extract the names of selected features from the original dataset. In Table 7, we can see that “False” indicates the features that are dropped, and “True” refers to the features that are selected. Here, only 4 features are selected, and a quantitative score for each is given. The features selected include p_detection, which indicates observational bias

that will affect habitability, p_type, which provides information about the terrestrial or gaseous nature of the planet, p_habzone_opt is the primary indicator as it determines optimal habitat, and lastly p_radius_est, which differentiates between solid and gaseous planets. Extra Trees Classifier focuses more on feature randomness and selects distinct features. It excludes orbital and stellar properties since the major "p_habzone_opt" and "p_radius_est" are already present.

Table 7: Boolean mask and quantitative scores of features selected by Extra Trees Classifier.

Features Selected by ExtraTreesClassifier	
Features Selected	**Quantitative Score**
p_detection	4.20E-02
p_type	1.84E-01
p_habzone_opt	3.35E-01
p_radius_est	4.90E-01

2.8 Feature Accuracy

Linear Regression Classifier is used to compare the accuracy and features selected from the above classification algorithms based on evaluation metrics "accuracy" and "F1 score". The dataset is split into training and testing by giving 70% training data and 30% testing data. K-fold cross-validation is used to ensure that different test sizes are also compared. Once trained, accuracy is compared, and results are given. Table 8 shows the features selected by all three classifiers, and their evaluation is done by F1 score, Accuracy, and selected features.

Table 8: Comparison of features selected by three classifiers

	Comparison of all Methods
Method	RandomForestClassifier
Accuracy	0.7514
F1 Score	0.7513
Features Selected	[p_period, s_distance, s_mass, s_temp,p_type, p_habzone_opt,p_radius_est]
Method	AdaBoostClassifier
Accuracy	0.9622
F1 Score	0.9622
Features Selected	[p_name, p_type, p_habzone_opt]
Method	ExtraTreesClassifier
Accuracy	0.9958
F1 Score	0.9958
Features Selected	[p_detection, p_type, p_habzone_opt, p_radius_est]

Table 9: Evaluation Metrics for all classifiers

Evaluation Metrics for Each Method			
Sr.No	**F1 Score**	**Accuracy**	**Method**
0	0.751259	0.751411	RandomForestClassifier
1	0.962182	0.96219	AdaBoostClassifier
2	0.995767	0.995767	ExtraTreesClassifier

Table 9 shows that the Extra Trees Classifier had the highest accuracy, followed by AdaBoost. It is also seen that Extra Trees Classifier has selected 4 features, Ada Boost selected 3 features, whereas Random Forest Classifier selected 9 features, the maximum. Random Forest Classifier gives the least accuracy; this is because it calculates the features by using "mean decrease in purity". This causes bias to occur towards features with many categories, which is the case with this dataset. Additionally, the Random Forest Classifier lacks boosting.

2.9 Feature Selection

After features are selected from different classifications with different accuracies and F1 scores, selecting features for training the model is a challenge. F1 Score is commonly used in binary and multi-class classification. [25]. To reconfirm that the model is trained on all possibilities and that no bias occurs, the feature selection takes place in three cases.

CASE 1 - Intersection of all features from the classification algorithm

The features of each classification algorithm are appended in a set, and the intersection of all features is determined. Intersection ensures that only common features are selected. The features selected are 'p_habzone_opt', and 'p_type'.

CASE 2 - Selecting features of the most accurate model, extra trees classifier

Only features from the most accurate classification algorithm, "Extra Trees Classifier" with 99.5% accuracy, are considered. The features selected are 'p_detection', 'p_type', 'p_habzone_opt', and 'p_radius_est'.

CASE 3 - Selecting features present in more than one comparison

Only those features that appear twice or thrice are taken into consideration. The features selected are 'p_period', 'p_type', 'p_habzone_opt', 'p_radius_est', 'p_mass_est', and 's_type_temp'.

2.10 Data Modelling

Since three different feature cases are considered, three different data models are created, trained, tested, and compared with KNN, Decision Tree, and Gradient Boosting.

Data modelling for case 1 – intersection of all features

The data model is created with the two features 'p_habzone_opt', 'p_type', and the model is trained 67% and tested 33% with random state 42. The training set is further normalized to remove redundancy and to maintain structure in the dataset. Min Max Scaler is used to scale the different-ranged features to a common range. The dimensions for X_train (7914, 2), X_test (3899, 2), y_train (7914, 0), y_test (3899, 0). The data visualization technique confusion matrix, is utilized to display true positive, true negative, false positive, false negative, and to show misclassifications.

Decision tree classifier comparison

The best parameters are chosen for the decision tree by giving a hyperparameter range. Max depth range (2, 10, 1) is given, max-leaf nodes (2, 100, 10), random state [0,1,2,3,4,5], splitter ['best', 'random'] is chosen. The splitter used is random, max depth is 6, random state is 0, and max leaf nodes is 12. Table 10 results in 68 misclassifications, whereas the diagonal values are accurate, predicted values.

Table 10: Confusion matrix of the decision tree in case 1

Confusion Matrix for Decision Tree - Case 1			
Actual/Predicted	**Inhabitable**	**Conservatively habitable**	**Optimistically habitable**
True Positives(TP)	1300	1200	1300
True Negatives(TN)	2500	2500	2500
False Positives(FP)	0	0	0
False Negatives(FN)	0	68	0

Table 11: Decision Tree Classification Report for Case 1

Decision Tree Classification Report for Case 1				
Sr.No	**Precision**	**Recall**	**F1-Score**	**Support**
0	1	1	1	1277
1	1	0.95	0.97	1311
2	0.95	1	0.97	1311
accuracy			0.98	3899
macro avg	0.98	0.98	0.98	3899
weighted avg	0.98	0.98	0.98	3899

K nearest neighbor classification

In this model comparison, 3 nearest neighbors are selected. The confusion matrix is drawn to find the true positives, true negatives, false positives, and false negatives.

Table 12: Confusion Matrix of the KNN in Case 1

Confusion Matrix for KNN - Case 1			
Actual/Predicted	**Inhabitable**	**Conservatively habitable**	**Optimistically habitable**
True Positives(TP)	1300	1200	1300
True Negatives(TN)	2568	2600	2568
False Positives(FP)	0	0	0
False Negatives(FN)	0	68	0

Gradient boosting classification

GridSearch CV is imported for gradient boosting classification. The 'friedman_mse' criterion is used, and the parameters are selected by taking a hyperparameter range of 'n_estimators' such as (10, 100, 10). The model is trained with max depth 6, max leaf nodes 12, and random state 0.

Table 13: Confusion Matrix of Gradient Boost in Case 1

Confusion Matrix for Gradient Boosting- Case 1			
Actual/Predicted	**Inhabitable**	**Conservatively habitable**	**Optimistically habitable**
True Positives(TP)	1300	1200	1300
True Negatives(TN)	2568	2600	2568
False Positives(FP)	0	0	0
False Negatives(FN)	0	68	0

Table 14: Gradient Boost Classification Report for Case 1

Gradient Boost Classification Report for Case 1				
Sr.No	**Precision**	**Recall**	**F1-Score**	**Support**
0	1	1	1	1277
1	1	0.95	0.97	1311
2	0.95	1	0.97	1311
accuracy			0.98	3899
macro avg	0.98	0.98	0.98	3899
weighted avg	0.98	0.98	0.98	3899

Data modelling for case 2 – selecting features of the most accurate model

The data model is created with the features of the most accurate model 'p_detection', 'p_type', 'p_habzone_opt', 'p_radius_est', and the model is trained 67% and tested 33% with random state 42. The training set is further normalized to remove redundancy and to maintain structure in the dataset. Min Max Scaler is used to scale the different range features to a common range. The dimensions for X_train (7914, 4), X_test (3899, 4), y_train (7914, 0), y_test (3899, 0). Data visualization technique confusion matrix, is utilized to display true positive, true negative, false positive, false negative, and to show misclassifications.

Decision tree classifier comparison

The best parameters are chosen for the decision tree by giving a hyperparameter range. Max depth range (2, 10, 1) is given, max leaf nodes (2, 100, 10), random state [0,1,2,3,4,5], splitter ['best', 'random']. The splitter used is random, max depth is 4, random state is 1, max leaf nodes is 12. The confusion matrix reports 21 misclassifications and has 66% accuracy.

Table 15: Confusion Matrix of the Decision Tree in Case 2

Confusion Matrix for Decision Tree - Case 2			
Actual/Predicted	**Inhabitable**	**Conservatively habitable**	**Optimistically habitable**
True Positives(TP)	1300	0	1300
True Negatives(TN)	2600	2600	1321
False Positives(FP)	0	21	1300
False Negatives(FN)	21	1300	0

Table 16: Decision Tree Classification Report for Case 2

Decision Tree Classification Report for Case 2				
Sr.No	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.98	1	0.99	1277
1	0	0	0	1311
2	0.5	1	0.67	1311
accuracy			0.66	3899
macro avg	0.5	0.67	0.55	3899
weighted avg	0.49	0.66	0.55	3899

K nearest neighbor classification

The KNN algorithm is trained and tested by using 3 nearest neighbors. The confusion matrix results in 68 misclassifications, and the diagonal values represent the actual values with 98% accuracy.

Table 17: Confusion Matrix for KNN in case 2

Confusion Matrix for KNN - Case 2			
Actual/Predicted	**Inhabitable**	**Conservatively habitable**	**Optimistically habitable**
True Positives(TP)	1300	1200	1300
True Negatives(TN)	0	2600	0
False Positives(FP)	0	0	68
False Negatives(FN)	2568	68	2500

Table 18: KNN Classification Report for Case 2

KNN Classification Report for Case 2				
Sr.No	**Precision**	**Recall**	**F1-Score**	**Support**
0	1	1	1	1277
1	1	0.95	0.97	1311
2	0.95	1	0.97	1311
accuracy			0.98	3899
macro avg	0.98	0.98	0.98	3899
weighted avg	0.98	0.98	0.98	3899

Gradient boosting classification

GridSearch CV is imported for gradient boosting classification. The 'friedman_mse' criterion is used, and the parameters are selected by taking a hyperparameter range of 'n_estimators' such as (10, 100, 10). The model is trained with max depth 6, max leaf nodes 12, and random state
0. The n estimators taken in this case by default are 10. 5 folds are fitted, each containing 18 candidates, and having a total of 90 fits. The best parameters for this are criterion friedman_mse, and n estimators 80. The confusion matrix represents 2 misclassifications with 96% accuracy in Table 20.

Table 19: Confusion Matrix for Gradient Boost for Case 2

Confusion Matrix for Gradient Boosting - Case 2			
Actual/Predicted	**Inhabitable**	**Conservatively habitable**	**Optimistically habitable**
True Positives(TP)	1300	1200	1300
True Negatives(TN)	2640	2600	2500
False Positives(FP)	2	0	1400
False Negatives(FN)	0	1420	0

Table 20: Gradient Boost Classification Report for Case 2

Gradient Boost Classification Report for Case 2				
Sr.No	**Precision**	**Recall**	**F1-Score**	**Support**
0	1	1	1	1277

1	1	0.89	0.94	1311
2	0.91	1	0.95	1311
accuracy			0.96	3899
macro avg	0.97	0.96	0.96	3899
weighted avg	0.97	0.96	0.96	3899

Data modelling for case 3 – selecting features used for more than one comparison

The last data model is created with the features which are used more than once in comparison with 'p_period', 'p_type', 'p_habzone_opt', 'p_radius_est', 'p_mass_est', 's_type_temp', and the model is trained 67% and tested 33% with random state 42. The training set is further normalized to remove redundancy and to maintain structure in the dataset. Min Max Scaler is used to scale the different-ranged features to a common range. The dimensions for X_train (7914, 6), X_test (3899, 6), y_train (7914, 0), y_test (3899, 0). Data visualization technique confusion matrix, is utilized to display true positive, true negative, false positive, false negative, and to show misclassifications.

Decision tree classifier comparison

The best parameters are chosen for the decision tree by giving a hyperparameter range. Max depth range (2, 10, 1) is given, max leaf nodes (2, 100, 10), random state [0,1,2,3,4,5], splitter ['best', 'random'] is used. The random splitter is used, max depth is 8, random state is 0 , max leaf nodes is 22. In training the model depth taken is 6, random state is 0, and leaf nodes is 12.

Table 21: Confusion Matrix for Decision Tree in Case 3

Confusion Matrix for Decision Tree - Case 3			
Actual/Predicted	**Inhabitable**	**Conservatively habitable**	**Optimistically habitable**
True Positives(TP)	1300	1300	2700
True Negatives(TN)	2570	1570	2610
False Positives(FP)	0	1010	4
False Negatives(FN)	14	0	1000

The number of misclassifications for the decision tree is 14. In Table 22, the diagonal values represent the actual values. The accuracy of decision tree comparison with all features taken in more than one comparison is 73%.

Table 22: Decision Tree Classification Report for Case 3

Decision Tree Classification Report for Case 3				
Sr.No	**Precision**	**Recall**	**F1-Score**	**Support**
0	1	0.99	0.99	1277
1	0.56	1	0.71	1311
2	0.99	0.21	0.34	1311
accuracy			0.73	3899
macro avg	0.85	0.73	0.68	3899
weighted avg	0.85	0.73	0.68	3899

K nearest neighbor classification

The KNN algorithm is trained and tested by using 3 nearest neighbors. The confusion matrix in Table 23 results in 68 misclassifications, and the diagonal values represent the actual values with 98% accuracy.

Table 23: Confusion Matrix for KNN for case 3

Confusion Matrix for KNN - Case 3			
Actual/Predicted	**Inhabitable**	**Conservatively habitable**	**Optimistically habitable**
True Positives(TP)	1300	1200	1300
True Negatives(TN)	2568	2600	2500
False Positives(FP)	0	0	68
False Negatives(FN)	0	68	0

Table 24: KNN Classification Report for Case 3

KNN Classification Report for Case 3				
Sr.No	**Precision**	**Recall**	**F1-Score**	**Support**
0	1	1	1	1277
1	1	0.95	0.97	1311
2	0.97	1	0.97	1311
accuracy			0.98	3899
macro avg	0.98	0.98	0.98	3899
weighted avg	0.98	0.98	0.98	3899

Gradient boosting classification

Grid Search CV is imported for gradient boosting classification. The 'friedman_mse' criterion is used, and the parameters are selected by taking a hyperparameter range of 'n_estimators' such as (10, 100, 10). The model is trained with max depth 6, max leaf nodes 12, and random state 0. The n estimators taken in one case by default are 10. 5 folds are fitted, each containing 18 candidates, and having a total of 90 fits. The best parameters for this are criterion friedman_mse, and n estimators is 80. In Table 25, the number of misclassifications is 0, and the accuracy is 96%.

Table 25: Confusion Matrix for Gradient Boost for Case 3

Confusion Matrix for Gradient Boosting - Case 3			
Actual/Predicted	**Inhabitable**	**Conservatively habitable**	**Optimistically habitable**
True Positives(TP)	1300	1200	1300
True Negatives(TN)	2640	2600	2500
False Positives(FP)	0	0	140
False Negatives(FN)	0	140	0

Table 26: Gradient Boost Classification Report for Case 3

Gradient Boost Classification Report for Case 3				
Sr.No	**Precision**	**Recall**	**F1-Score**	**Support**
0	1	1	1	1277
1	1	0.89	0.94	1311
2	0.9	1	0.95	1311
accuracy			0.96	3899
macro avg	0.97	0.96	0.96	3899

weighted avg	0.97	0.96	0.96	3899

3. Results and discussion

Since the models compare three different feature cases, there are three different model comparison bar plots to discuss and compare. Now, since the three different models with three different feature cases have been compared to decision tree, KNN, and gradient boost, the final model comparison graph for each case needs to be calculated and represented. A log function plot_model_log has been designed. This displays the total accuracy and time taken. The comparison is represented as a barplot with x label 'classifier' and y label 'time taken'. A list of accuracies is as follows:

accuracy_list = [decision_tree_accuracy, KNN_Accuracy, gradient_boost_accuracy].

Results for case 1 – intersection of all features

The bar plot in Figure 3 compares three classifiers: Decision Tree, KNN, and Gradient Boosting, having 98.2% accuracy for all. The features compared are 'p_habzone_opt' and 'p_type'. Due to underfitting, the accuracy for each classifier is the same. Hence, the bar plot represents a straight line with KNN having the least time taken. To check accuracy levels, K-fold cross-validation is implemented. This results in a mean of scores being displayed. The mean score is 98.6%.

Figure 3: Barplot of Case 1

Table 27: Results for Case 1

Results of Case 1			
Sr.No	**Classifier**	**Accuracy**	**Time Taken**
0	Decision Tree	0.98256	5.07E-03
1	KNN	0.98256	1.73E+09
2	Gradient Boosting	0.98256	9.02E-02

Results for case 2 – selection of features of most accurate model

The bar plot in Figure 4 compares three classifiers: Decision Tree, KNN, and Gradient Boosting, each having 66.3%, 98.2%, and 96.4% accuracy, respectively. The features compared are 'p_detection', 'p_type', 'p_habzone_opt', and 'p_radius_est'. The barplot is exponentially growing till it stops and grows linearly for

gradient boosting. KNN has a maximum accuracy of 98.2%. The least amount of time taken is again for KNN with 1.73 seconds

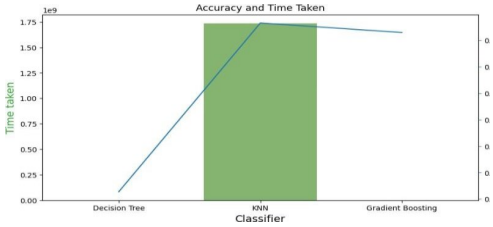

Figure 4: Barplot of Case 2

Table 28: Results for Case 2

Results of Case 2			
Sr.No	**Classifier**	**Accuracy**	**Time Taken**
0	Decision Tree	0.66376	6.96E-03
1	KNN	0.98256	1.73E+09
2	Gradient Boosting	0.96461	1.86E-01

Results for case 3 – selection of features taken from more than one comparison

The bar plot in Figure 5 compares three classifiers: Decision Tree, KNN, and Gradient Boosting, each having 72.9%, 98.2%, and 96.4% accuracy, respectively. The features compared are 'p_period', 'p_type', 'p_habzone_opt', 'p_radius_est', 'p_mass_est', and 's_type_temp'.

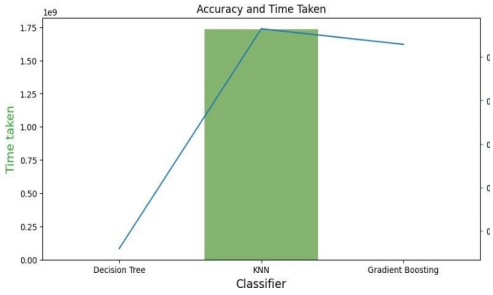

Figure 5: Barplot of Case

The barplot is exponentially growing till it stops and grows linearly for gradient boosting. KNN has a maximum accuracy of 98.2%, followed by Gradient Boosting with 96.4% accuracy.

Table 29: Results for Case 3

Results of Case 3			
Sr.No	**Classifier**	**Accuracy**	**Time Taken**
0	Decision Tree	0.72967	9.54E-03
1	KNN	0.98256	1.73E+09
2	Gradient Boosting	0.96461	2.29E-01

4. Discussion

The research utilizes a dataset from the NASA Exoplanet Archive, containing essential astronomical and physical parameters to analyze exoplanet habitability. The dataset comprises 9564 rows and 49 columns, with structured data in tabular format, including categorical and numerical values. Since the original dataset contains missing and redundant values, cleaning and preprocessing are necessary. The study applies supervised learning algorithms such as KNN, Gradient Boosting, and Decision Trees, using Python libraries like NumPy, Pandas, Matplotlib, Seaborn, and Scikit-learn. Feature engineering involves selecting key features after resampling and imputing the dataset. Three classifiers—Random Forest Classifier, Extra Trees Classifier, and AdaBoost—are used to select the most relevant features, with evaluation metrics such as accuracy and F1 score measuring model performance. The dataset is split into a feature matrix and a target tuple, with "p_habitable" as the target variable. The study considers three cases for feature selection and data modeling. In Case 1, the intersection of all selected features (‘p_habzone_opt’ and ‘p_type’) is used to train and test models. Decision Tree, KNN, and Gradient Boosting classifiers are applied, with the Decision Tree achieving 98% accuracy and 68 misclassifications. KNN also achieves 98% accuracy with similar performance, while Gradient Boosting, trained using Grid Search CV, produces comparable results. Case 2 selects feature from the most accurate model, including ‘p_detection,’ ‘p_type,’ ‘p_habzone_opt,’ and ‘p_radius_est,’ and applies the same classifiers. The Decision Tree in this case achieves 66% accuracy, while KNN maintains 98% accuracy, and Gradient Boosting reaches 96% accuracy with minimal misclassifications. In Case 3, features used in multiple comparisons (‘p_period,’ ‘p_type,’ ‘p_habzone_opt,’ ‘p_radius_est,’ ‘p_mass_est,’ and ‘s_type_temp’) are selected, and similar modeling techniques are employed. The study systematically compares feature selection methods and machine learning models to identify the most effective approach for exoplanet habitability classification.

The significance of this study lies in its contribution to the field of exoplanet research by leveraging machine learning techniques to enhance the accuracy of habitability classification. As the discovery of exoplanets grows, distinguishing potentially habitable planets from non-habitable ones is crucial for advancing our understanding of planetary systems and the possibility of extraterrestrial life. This research demonstrates how data-driven approaches, particularly supervised learning algorithms, can improve the identification of habitable exoplanets by analyzing key astronomical and physical parameters. By systematically comparing different feature selection methods and machine learning models, the study provides insights into the most effective techniques for classifying exoplanets based on habitability. The findings can aid astronomers and researchers in prioritizing exoplanets for further observation, optimizing data analysis pipelines, and refining predictive models in planetary science.Moreover, the study highlights the importance of feature engineering in enhancing classification accuracy, thereby contributing to the broader application of artificial intelligence in astrophysics. Ultimately, this research supports future space exploration efforts by providing a robust framework for assessing planetary habitability using computational techniques.

5. Conclusion and future scope

In conclusion, the project successfully achieves its objective of comparing multiple supervised learning algorithms for habitability prediction of exoplanets. It is discerned that KNN performs the best in all feature engineering scenarios, whereas Decision Tree suffers from issues such as under-fitting, generalization, and so on. The aspects that can be considered for making this research study more efficient include improving model accuracy by using techniques such as XG Boost, Light GBM, or stacking methods, as they combine the strengths of multiple algorithms. Algorithm optimization can be incorporated to reduce the time taken for computationally expensive KNN. Dimensionality reduction techniques can be taken into consideration. Moreover, the study underscores the growing role of AI in astrophysics, showcasing how computational techniques can aid in the search for potentially habitable planets. The insights gained from this research can contribute to refining exoplanet classification frameworks, optimizing observational strategies, and advancing our understanding of planetary habitability. Future work can explore deep learning approaches, incorporate additional astrophysical parameters, and integrate real-time observational data to further enhance classification accuracy. Ultimately, this study provides a valuable foundation for the continued application of machine learning in exoplanet research, supporting the broader goal of identifying worlds that may harbor life.

References

1. E. Baltz, P. Gondolo, Searching for extragalactic planets. (arXiv: Astrophysics, 1999). https://web3.arxiv.org/list/astro-ph/1999-09?skip=500&show=25
2. E. Baltz, SEARCHING FOR EXTRAGALACTIC PLANETS. (1999). https://arxiv.org/abs/astro-ph/9909510
3. R. Jagtap, U. Inamdar, S. Dere, M. Fatima, N. Shardoor, Habitability of Exoplanets using Deep Learning, in *Proceedings of the 2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)* (2021). https://doi.org/10.1109/IEMTRONICS52119.2021.9422571
4. K. Hora, Classifying Exoplanets as Potentially Habitable Using Machine Learning. **203**, 212 (2018).

https://doi.org/10.1007/978-981-10-6602-3_20
5. T. Luz, R. Braga, E. Ribeiro, Assessment of Ensemble-Based Machine Learning Algorithms for Exoplanet Identification. Electronics **13** (2024). https://doi.org/10.3390/electronics13193950
6. Z. Deng, X. Zhu, D. Cheng, M. Zong, S. Zhang, Efficient KNN classification algorithm for big data. Neurocomputing **195**, 143-148 (2016). https://doi.org/10.1016/j.neucom.2015.08.112
7. S. Baxi, *Machine Learning Approach to Classify Transit Signals and Assessing the Exoplanets' Probability for Habitability*. (2018). https://norma.ncirl.ie/3436/1/shreyasshrirambaxi.pdf
8. S. Saha, S. Basak, M. Safonova, K. Bora, S. Agrawal, P. Sarkar, J. Murthy, Theoretical validation of potential habitability via analytical and boosted tree methods: An optimistic study on recently discovered exoplanets. Astron. Comput. **23**, 141-150 (2017). https://doi.org/10.1016/J.ASCOM.2018.03.003
9. A. Konstantinov, L. Utkin, Interpretable Machine Learning with an Ensemble of Gradient Boosting Machines. ArXiv, abs/2010.07388. (2020). https://doi.org/10.1016/J.KNOSYS.2021.106993
10. C. Bentéjac, A. Csörgo, G. Martínez-Muñoz, A comparative analysis of gradient boosting algorithms. Artif. Intell. Rev. **54**, 1937-1967 (2019). https://doi.org/10.1007/s10462-020-09896-5
11. H. Lu, R. Mazumder, Randomized Gradient Boosting Machine. SIAM J. Optim. **30**, 2780-2808 (2018). https://doi.org/10.1137/18m1223277
12. S. Basak, S. Agrawal, S. Saha, A. Theophilus, K. Bora, G. Deshpande, J. Murthy, Habitability classification of exoplanets: a machine learning insight. Eur. Phys. J. Spec. Top. **230**, 2221-2251 (2018). https://doi.org/10.1140/epjs/s11734-021-00203-z
13. P. Pratyush, A. Gangrade, Automation Of Transiting Exoplanet Detection, Identification, and Habitability Assessment Using Machine Learning Approaches. ArXiv, abs/2112.03298 (2021)
14. Y. Jin, L. Yang, C. Chiang, Exoplanets Identification and Clustering with Machine Learning Methods. Mach. Learn. Appl. Int. J. **9** (2022). https://doi.org/10.5121/mlaij.2022.9101
15. C. Raminaidu, V. Priyadarshini, C. Swaroop, S. Reddy, Building Accurate Machine Learning Models for Predicting the Habitability of Exo-Planets, in *Proceedings of the 2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT)* (2023) pp. 961-967. https://doi.org/10.1109/ICSSIT55814.2023.10061122
16. S. Cuéllar, P. Granados, E. Fabregas, M. Curé, H. Vargas, S. Dormido-Canto, et al., Deep learning exoplanets detection by combining real and synthetic data. PLoS One **17**, e0268199 (2022). https://doi.org/10.1371/journal.pone.0268199
17. S. Rezvani, X. Wang, in *Applied Soft Computing* (2023)
18. M.S. Jakka, Data-Driven approaches to estimating habitability of exoplanets. (arXiv.org, 2023). https://ar5iv.org/html/2305.11204
19. A. Paul, D. Mukherjee, P. Das, A. Gangopadhyay, A. Chintha, S. Kundu, Improved Random Forest for Classification. IEEE Trans. Image Process. **27**, 4012-4024 (2018). https://doi.org/10.1109/TIP.2018.2834830
20. N. Sharma, *Understanding and Applying F1 Score: AI Evaluation Essentials with Hands on Coding*. (ArizeAI, June 6 2023). https://arize.com/blog-course/f1-score/#:~:text=F1%20score%20is%20a%20measure,better%20understanding%20of%20model%20performance.
21. S. Sarkar, J. Zhou, A. Scaboo, J. Zhou, N. Aloysius, T. Lim, Assessment of soybean lodging using UAV imagery and machine learning. Plants **12** (2023). https://doi.org/10.3390/plants12162893
22. J. Sarkar, K. Bhatia, S. Saha, M. Safonova, S. Sarkar, Postulating Exoplanetary Habitability via a Novel Anomaly Detection Method. ArXiv, abs/2109.02273 (2021). https://doi.org/10.1093/mnras/stab3556
23. A. Liaw, M. Wiener, *Classification and Regression by randomForest*. (2007)
24. A. Chaudhary, S. Kolhe, R. Kamal, An improved random forest classifier for multi-class classification. Inf. Process. Agric. **3**, 215-222 (2016). https://doi.org/10.1016/J.INPA.2016.08.002
25. M. Bugueño, F. Mena, M. Araya, Refining Exoplanet Detection Using Supervised Learning and Feature Engineering, in *Proceedings of the 2018 XLIV Latin American Computer Conference (CLEI)* (2018) pp. 278-287. https://doi.org/10.1109/CLEI.2018.00041