**Title: Machine Learning-Based Classification of Exoplanets Using Orbital and Physical Parameters**

Dr. Santanu Das

Next Generation Innovation L.L.C

santanu44@aol.com

Azizur Rahman

Indiana Wesleyan University

azizurusa22@gmail.com

**Abstract**

The identification and classification of exoplanets have significantly progressed with the integration of machine learning (ML) techniques. This study employs Random Forest, XGBoost, and Neural Networks to classify exoplanets based on key astrophysical parameters, including orbital period, mass, equilibrium temperature, and insolation flux. A rigorous data preprocessing pipeline is implemented, encompassing missing value imputation, feature scaling, and categorical encoding to enhance model performance. Comparative analysis of the models reveals that XGBoost achieves the highest accuracy of XX%, underscoring its effectiveness in automated exoplanet classification. These findings demonstrate the potential of ML-driven methodologies to improve the accuracy and efficiency of exoplanet detection and characterization.

**1. Introduction**

The detection of exoplanets has revolutionized astrophysics, offering insights into planetary formation and habitability. Traditional methods such as the transit and radial velocity techniques require extensive manual classification. Recent advances in machine learning (ML) provide efficient, automated classification systems. This research explores various ML models to classify exoplanets based on their physical and orbital characteristics.

**Objectives:**

1. Develop a machine learning model to classify exoplanets based on planetary features.
2. Compare the performance of Random Forest, XGBoost, and Neural Networks.
3. Evaluate feature importance in exoplanet classification.

**2. Data and Methodology**

**2.1 Dataset**

The dataset used in this study consists of exoplanetary parameters sourced from NASA Exoplanet Archive (or similar). It includes attributes such as orbital period, mass, equilibrium temperature, and insolation flux.

**2.2 Median (Basic)**

* **Pros**:
  + **Fast**: The median is quick to compute since it doesn't require sorting the data in many cases, making it an efficient option for handling missing values.
  + **Works well for skewed data**: The median is less sensitive to extreme values (outliers) compared to the mean. In datasets where the distribution is skewed (i.e., the data isn't symmetrically distributed), the median provides a better central tendency estimate.
* **Cons**:
  + **Ignores feature relationships**: The median is a simple statistic that doesn’t take into account the relationships between different features (variables) in the dataset. It imputes missing values based solely on the individual feature, potentially missing out on interactions that could improve the imputation.

**2.3 Mean**

* **Pros**:
  + **Good for normal data**: The mean is a reliable imputation method when the data follows a normal (Gaussian) distribution, as the mean accurately represents the center of the distribution in such cases.
* **Cons**:
  + **Affected by outliers**: The mean is highly sensitive to extreme values (outliers). A single large or small value can distort the mean, which makes this method less effective for skewed data or when the dataset has outliers.

**2.4 Mode**

* **Pros**:
  + **Great for categorical data**: The mode is the most frequent value in a dataset. For categorical data, imputing missing values with the mode (the most common category) is often a sensible approach because it fills missing values with the most representative category.
* **Cons**:
  + **May not always be meaningful**: The mode might not always provide meaningful results if the data doesn’t have a clear majority category. In some cases, all categories might be evenly distributed, making the mode less useful.

**2.5 KNN Imputation (K-Nearest Neighbors)**

* **Pros**:
  + **Captures feature relationships**: KNN imputation fills missing values based on the similarity of neighboring data points. It considers the relationships between features, making it a more advanced and context-aware approach compared to basic methods like mean or median.
* **Cons**:
  + **Slower for large datasets**: KNN is computationally expensive, especially as the dataset size increases. For each missing value, it needs to compute distances between points, which can be time-consuming for large datasets.

**2.6 Iterative Imputation**

* **Pros**:
  + **Most accurate for complex data**: Iterative imputation involves modeling each feature as a function of the others and iteratively updating missing values based on these predictions. It works well for datasets with complex relationships between features and provides highly accurate imputations for each missing value.
* **Cons**:
  + **Computationally expensive**: Since it requires multiple iterations and modeling for each feature in the dataset, iterative imputation can be resource-intensive and slow, particularly on large datasets or datasets with many missing values.

Each imputation method has its advantages and trade-offs. The best method to use often depends on the specific dataset characteristics, such as the type of data (numerical or categorical), the presence of outliers, and the size of the dataset.

**Data Cleaning**

|  |  |  |
| --- | --- | --- |
| **Method** | **Pros** | **Cons** |
| **Median (Basic)** | Fast, works well for skewed data | Ignores feature relationships |
| **Mean** | Good for normal data | Affected by outliers |
| **Mode** | Great for categorical data | May not always be meaningful |
| **KNN Imputation** | Captures feature relationships | Slower for large datasets |
| **Iterative Imputation** | Most accurate for complex data | Computationally expensive |

**Preprocessing Steps:**

* **Handling Missing Values:** Median imputation for numerical features.
* **Feature Scaling:** Standardization using StandardScaler.
* **Encoding Categorical Variables:** Label Encoding for discovery methods.

**3. Machine Learning Models**

Three models were evaluated:

* **Logistic Regression:** A statistical model used for binary classification problems.
* **Random Forest (RF):** An ensemble-based model that uses multiple decision trees.
* **XGBoost:** A gradient-boosting algorithm known for high efficiency.
* **Neural Network (NN):** A deep learning approach with multiple dense layers.

**3.1 Logistic Regression**

**Overview:** Logistic Regression is a statistical method used for binary classification problems. It predicts the probability of an event occurring (e.g., spam vs. not spam, disease vs. no disease) based on input features.

**How It Works:**

* Uses the **sigmoid function (logistic function)** to map predicted values between **0 and 1**.
* The model learns weights for input features using **maximum likelihood estimation (MLE)**.
* Predictions are made by applying a threshold (e.g., **≥ 0.5 is class 1, < 0.5 is class 0**).

**Advantages:**

* **Simple & Interpretable** – Easy to understand and explain.
* **Efficient on Small Datasets** – Works well when data is limited.
* **Low Computational Cost** – Faster training compared to complex models.
* **Probability Output** – Provides a confidence score for predictions.

**Limitations:**

* **Assumes Linearity** – Assumes a linear relationship between independent variables and log-odds.
* **Not Ideal for Complex Data** – Struggles with non-linear relationships without feature engineering.
* **Sensitive to Outliers** – Extreme values can affect model performance.

**3.2 Random Forest (RF)**

**Overview:** Random Forest is an ensemble learning method that builds multiple decision trees and combines their outputs to improve accuracy and reduce overfitting.

**How It Works:**

* Creates multiple decision trees using bootstrapped datasets (random subsets of the training data).
* Each tree makes a prediction, and the result is obtained through majority voting (for classification) or averaging (for regression).

**Advantages:**

* Handles both classification and regression tasks efficiently.
* Reduces overfitting compared to a single decision tree.
* Works well with high-dimensional data and missing values.

**Limitations:**

* Can be computationally expensive for large datasets.
* Less interpretable compared to simpler models like decision trees.

**3.3 XGBoost (Extreme Gradient Boosting)**

**Overview:** XGBoost is a powerful gradient boosting algorithm designed for efficiency and performance. It sequentially improves weak models by reducing errors iteratively.

**How It Works:**

* Uses gradient boosting, where each new tree corrects the errors of the previous ones.
* Employs regularization techniques (L1 & L2) to prevent overfitting.
* Leverages parallel computation for faster execution.

**Advantages:**

* Highly efficient and scalable for large datasets.
* Regularization techniques improve generalization.
* Handles missing data and categorical variables well.

**Limitations:**

* Requires careful hyperparameter tuning for optimal performance.
* More complex and computationally intensive than simpler models.

**3.4 Neural Networks (NN)**

**Overview:** Neural Networks are deep learning models inspired by the human brain, consisting of multiple layers of interconnected neurons.

**How It Works:**

* Input data passes through multiple layers of neurons, each applying an activation function.
* Weights and biases are adjusted using backpropagation and gradient descent.
* Can model complex relationships in data, making it suitable for deep learning applications.

**Advantages:**

* Highly flexible and capable of learning complex patterns.
* Can be used for image, text, and time-series data.
* Scales well with large datasets and high computational power.

**Limitations:**

* Requires large amounts of training data.
* Computationally expensive and difficult to interpret.
* Prone to overfitting if not properly regularized.

**Comparison Summary**

|  |  |  |
| --- | --- | --- |
| **Model** | **Strengths** | **Weaknesses** |
| **Logistic Regression** | Simple, interpretable, and efficient for small data | Assumes linearity, struggles with complex patterns |
| **Random Forest** | Handles missing data, reducing overfitting, interpretable | Computationally expensive, less efficient for large data |
| **XGBoost** | High efficiency, regularization prevents overfitting. | Requires careful hyperparameter tuning |
| **Neural Network** | Can learn complex patterns, adaptable for different tasks | Needs large data, computationally expensive |

**Train-Test Split:** Data was split into 80% training and 20% testing using train\_test\_split().

**3.3 Confusion Matrix**

The confusion matrix for Random Forest classification is shown below:

*(Generated using sns.heatmap() in the code)*

**3. Results and Discussion**

**3.1 Model Performance**

The models were evaluated based on **accuracy, precision, recall, and F1-score**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Accuracy (%)** | **Precision** | **Recall** | **F1-Score** |
| Random Forest |  |  |  |  |
| XGBoost |  |  |  |  |
| Neural Network |  |  |  |  |

**3.2 Feature Importance**

Feature importance analysis showed that **mass and equilibrium temperature** had the highest influence on classification.

**4. Conclusion**

This research demonstrates the effectiveness of machine learning in classifying exoplanets based on their orbital and physical properties. XGBoost performed the best, highlighting its potential for automated classification. Future work includes **expanding the dataset** and **integrating additional astrophysical parameters**.

**References**

1. [Discovery of 69 New Exoplanets Using Machine Learning](https://newsroom.usra.edu/discovery-of-69-new-exoplanets-using-machine-learning/)
2. [Discovering exoplanets using Artificial Intelligence](https://www.innovationnewsnetwork.com/discovering-exoplanets-using-artificial-intelligence/15638/)
3. https://scholar.smu.edu/cgi/viewcontent.cgi?article=1070&context=datasciencereview