<ml4a-090424-1>

SPARTIFICIAL

Spartificial Innovations Private Limited

Harnessing Machine Learning for Exoplanet Detection

Rohan Jagtap
Prathmesh Bhandare
Preetam Kumar
Karankumar Nagar

Mr. Rohan Shah

Contents

Abstract

The goal of this study is to develop a machine learning model capable of predicting confirmed exoplanets and distinguishing them from false positives, leveraging data from missions like Kepler, TESS, and K2. These missions have significantly expanded the available dataset for exoplanet detection, making manual analysis both time-consuming and resource-intensive. By applying advanced machine learning techniques, this study aims to streamline the exoplanet classification process, reducing the time required for identification.

Cumulative data from the Kepler mission was used to train the model, ensuring its applicability across multiple satellite missions and Threshold Crossing Events (TCEs). The model underwent various stages of optimization, including hyperparameter tuning and feature selection, to enhance its performance. Several machine learning algorithms were tested, with the final model achieving an accuracy of 89%, a precision score of 0.95, and a recall score of 0.86.

This approach offers a significant improvement in the speed and efficiency of exoplanet classification, providing a scalable solution for future astronomical research. The model not only demonstrates robust performance but also facilitates the analysis of large datasets from upcoming exoplanet missions, contributing to the rapid discovery of new celestial bodies.

1 Introduction

The identification and classification of exoplanets—planets outside our solar system—has been a significant area of research in astronomy. With the data influx from modern space missions like Kepler and TESS (Transiting Exoplanet Survey Satellite), which use time-series photometric data to monitor variations in stellar brightness, astronomers have the opportunity to detect potential exoplanets. However, the scale and complexity of this data demand more advanced processing techniques than traditional methods, such as manual inspection or simple statistical models.

The goal of this project is to develop an AI module that automates and enhances the detection of exoplanets from the data collected by the Kepler and TESS missions. By leveraging machine learning techniques, we aim to streamline the analysis of transit photometry—where minor dips in a star's light indicate a planet's transit across it. The large-scale datasets from NASA's archives provide a rich ground for applying advanced data-driven methods, allowing us to improve the accuracy of exoplanet predictions.

In this study, we focus on feature selection and data preprocessing to extract meaningful information from the raw data and optimize our machine learning models. By analyzing 11 common features from both Kepler and TESS datasets, we employ techniques such as the ANOVA F-test and mutual information gain to refine our predictive capabilities. Moreover, we address common challenges such as missing data and class imbalance, ensuring a robust training process that leads to a more accurate, scalable model.

This project aims to develop an AI solution that pushes the boundaries of exoplanet detection, combining cutting-edge machine learning techniques with the vast astronomical datasets at our disposal. The outcome will not only accelerate the identification of exoplanets but also enhance our understanding of their characteristics, such as size and orbital patterns.

2 Proposed Solution

2.1 Solution Overview

The proposed solution aims to predict exoplanets using a robust machine learning framework that effectively analyzes data obtained from the Kepler and TESS missions. To achieve this, we have developed an AI module that integrates advanced algorithms to automate the classification of potential exoplanet candidates, enhancing both accuracy and efficiency in the detection process.

The module leverages a systematic approach beginning with feature selection, where we identified and visualized the most relevant features from the dataset. By focusing on common attributes shared between the TESS and Kepler datasets, we ensured the model's adaptability for future data collection.

Data preprocessing plays a critical role in the effectiveness of our model. We employed techniques such as imputation of missing values using the KNNImputer, which ensures that gaps in the dataset do not hinder the learning process. Feature scaling and handling class imbalances were also addressed to optimize the performance of machine learning algorithms. These preprocessing steps lay the groundwork for the subsequent model training phase.

In training our model, we utilized a variety of algorithms, ultimately selecting XGBoost for its superior performance in large datasets. We employed stratified cross-validation to assess the models based on the F1 score, ensuring a reliable evaluation of each candidate.

Hyperparameter tuning through GridSearchCV was crucial in identifying the optimal parameters for our chosen model, enhancing its predictive capabilities. The final model was not only trained but also stored for future application, ensuring that the insights gained from this project can be leveraged in subsequent research endeavors.

Overall, the solution integrates advanced methodologies in machine learning and data processing, providing a comprehensive framework for the accurate prediction of exoplanets.

2.1.1 Methodology

Feature Selection: For effective data processing, it is crucial to visualize the characteristics of the dataset. In the exoplanet dataset, various features were calculated by satellites and provided for research. We began by visualizing these features by plotting them against one another. This served two purposes: first, to identify any dependencies between the features, as we aimed to create a generalized model, and second, to detect and remove any features that were redundant or derived from others before the preprocessing stage. To ensure the model's relevance for predicting exoplanets from new data collected by the TESS satellite, we selected common columns from both the TESS and Kepler datasets.

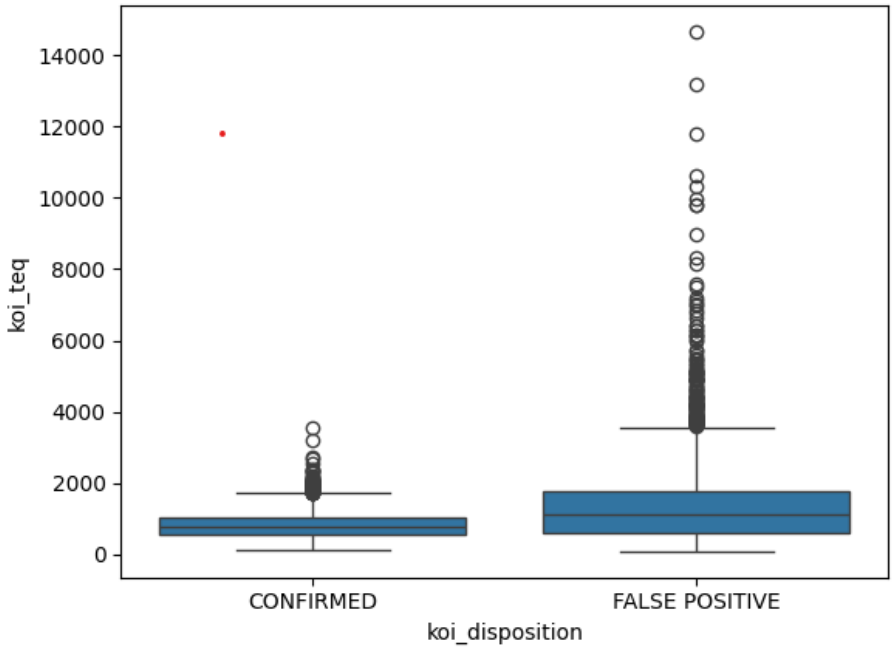

Figure 2.1: Boxplot showing distribution of surface temperatures for confirmed exoplanets and false positives

We identified 11 common features between the TESS and Kepler datasets. To refine our model and enhance its performance, we applied feature selection techniques such as the ANOVA F-test and mutual information gain. The ANOVA F-test allowed us to determine which features had the strongest statistical relationship with the target variable, while mutual information gain helped us measure the amount of information each feature contributed to the prediction task. By focusing on the most relevant features, we ensured that the model would not only be more accurate but also computationally efficient and less prone to overfitting.

Column	F-value	p-value
koi_teq	566.7147	0.0000
koi_depth	447.9092	0.0000
koi_steff	381.0907	0.0000
koi_slogg	259.4106	0.0000
koi_duration	206.0833	0.0000
koi_period	200.8013	0.0000
koi_srad	55.5926	0.0000
koi_impact	42.1102	0.0000
koi_time0bk	31.5319	0.0000
koi_insol	11.3035	0.0008
koi_prad	4.2850	0.0385

Figure 2.2: Selected 9 Features for Model

Data Preprocessing:

- **Imputation of Missing Values:** We used the KNNImputer algorithm to handle missing values in the feature dataset. The KNNImputer uses the k-nearest neighbors algorithm to fill in missing values based on the average of the nearest samples, with $n = 5$.
- **Feature Scaling:** To scale the data between 0 and 1, we applied the StandardScaler, which is crucial for many machine learning algorithms sensitive to the scale of input data.
- **Class Imbalance Handling:** Real-world datasets often suffer from class imbalance, and this project was no exception. After experimenting with different sampling techniques—ADASYN oversampling, centroid cluster undersampling, and a hybrid approach—we found that ADASYN was the best method for handling class imbalance.
- **Data Preparation:** We encoded categorical features, as machine learning models work with numerical data. The target column was encoded with 1 for "Confirmed" and 0 for "False Positives." The dataset was then split into training and testing sets using the `train_test_split()` function.

Building an API Application and Deployment: For this project, we utilized FastAPI to build the backend API of our application. The primary function of this API is to handle a single POST request that processes an input CSV file provided by the user. The workflow of this API involves the following steps:

- **Input Handling:** The user uploads a CSV file containing the required data.
- **Model Prediction:** The backend API leverages a pre-trained machine learning model to make predictions on the input data.
- **Response Generation:** The API generates a new CSV file that contains two additional columns:
  - A binary prediction column, indicating the classification result.
  - A percentage confidence score, representing how certain the model is about each prediction.

These outputs allow users to not only see the predicted results but also gauge the confidence level of the model for each prediction.

API and Frontend Integration: The backend API is seamlessly integrated with a frontend HTML interface. This allows users to interact with the application through a user-friendly interface, where they can upload their CSV files and download the processed output with predictions. The entire codebase, including both the API and frontend files, is hosted in a GitHub repository for version control and collaboration.

Deployment: The application is deployed on the Render platform, which provides reliable hosting for our FastAPI project. With this deployment, users can access the application from anywhere, ensuring ease of use and scalability.

Model Training and Evaluation: We selected multiple models to test: Support Vector Machines, tree-based models, boosting models, and even a neural network. The first filter for model selection was stratified cross-validation. We made 5 folds of the training data sets and calculated F1 scores. In this test, almost all the models performed very well, so we did not drop any model.

We used confusion matrices to evaluate the performance of each model. Precision and F1-score are good metrics, but we needed our models to perform well on the recall metric, as it indicates how well a model can recall the correct targets. The evaluation metrics we considered were:

- **Accuracy**: Proportion of true results among the total number of cases examined.
- **Precision**: Ratio of true positive results to the total predicted positives.
- **Recall**: Ratio of true positive results to all actual positives.
- **F1 Score**: Harmonic mean of precision and recall, providing a balance between the two.
- **Confusion Matrix**: A summary of prediction results on a classification problem.

Based on our evaluation, the model that had the best confusion matrix was XGBoost, known for its performance and efficiency in handling large datasets.

XGBoost Key Features:

- **Boosting Technique:** XGBoost employs boosting, where trees are built sequentially, with each new tree correcting errors made by the previous ones.
- **Regularization:** The model includes L1 and L2 regularization to prevent overfitting, making it suitable for noisy datasets.
- **Handling Missing Values:** XGBoost efficiently manages missing values by learning the optimal direction for them during training.
- **Tree Pruning:** It prunes trees using max depth pruning, improving efficiency and speed.
- **Parallelization:** The framework is designed for parallel processing, allowing it to handle large datasets effectively.

Hyperparameter Tuning: To optimize the performance of the `XGBClassifier`, hyperparameter tuning was conducted using the `GridSearchCV` method from the `sklearn` library. This process involved several crucial steps:

1. **Parameter Grid Definition:** A comprehensive grid of hyperparameters was created to explore various combinations. The key hyperparameters tuned for the `XGBClassifier` included:
    - **n_estimators:** Number of trees in the ensemble. Tested values: {100, 200}.
    - **learning_rate:** Step size shrinkage to prevent overfitting. Tested values: {0.01, 0.1, 0.2}.

- **max_depth:** Maximum depth of each tree. Tested values: {3, 4, 5}.
- **subsample:** Proportion of samples used for training each tree. Tested values: {0.8, 1.0}.
- **colsample_bytree:** Proportion of features used for training each tree. Tested values: {0.8, 1.0}.
- **gamma:** Minimum loss reduction required to make a further partition on a leaf node. Tested values: {0, 0.1, 0.2}.
- **scale_pos_weight:** Balances the weight of positive instances to address class imbalance. Tested values: {1, 2}.

2. **Cross-Validation Setup:** The `StratifiedKFold` cross-validation technique was employed to maintain class distribution across folds, ensuring more robust model evaluation.

3. **Grid Search Execution:** `GridSearchCV` was instantiated with the following parameters:

   - **estimator:** A pipeline consisting of necessary preprocessing steps and the `XGBClassifier`.
   - **param_grid:** The predefined grid of hyperparameters to be optimized.
   - **scoring:** Focused on the `F1` score to ensure class balance in model evaluation.
   - **cv:** Utilized the `StratifiedKFold` strategy for cross-validation.
   - **n_jobs:** Set to `-1` to leverage all available CPU cores for parallel processing, accelerating the grid search.

4. **Model Fitting:** The grid search was executed by calling the `fit` method on the training data (`X_train` and `y_train`). This process trained multiple models based on different hyperparameter combinations and evaluated them using cross-validation.

5. **Best Parameters Selection:** After completing the grid search, the best-performing hyperparameter combination was identified using the `best_params_` attribute. The model was then refitted on the training data using this optimal hyperparameter combination to ensure maximum performance.

3 Review of the State-of-the-Art

Exoplanet detection has become a highly active area of research, and the application of machine learning (ML) techniques in this field has seen tremendous growth. Large datasets generated by space missions such as NASA's Kepler and TESS (Transiting Exoplanet Survey Satellite) require sophisticated models that can handle the vast amounts of data produced daily. Machine learning has emerged as a key tool in processing light curves from these missions and identifying the presence of exoplanets.

3.1 Notable Machine Learning Models for Exoplanet Detection

Several studies have explored machine learning techniques for exoplanet detection and achieved impressive results in accuracy and efficiency:

- **Shallue and Andrew Vanderburg (2018):** These researchers proposed a convolutional neural network (CNN) to detect exoplanets by analyzing light curves. Their model successfully predicted two new exoplanets, demonstrating the effectiveness of CNNs for light curve analysis and exoplanet discovery.
- **Yucheng et al. (2020):** Yucheng and colleagues utilized various machine learning algorithms on NASA's Kepler dataset to identify exoplanets. They achieved a 99.79% accuracy using a multilayer perceptron (MLP), indicating the power of deep learning in exoplanet detection.
- **Malik et al. (2019):** This team employed a gradient boosting classifier on the Kepler dataset, achieving an accuracy of 98%. Gradient boosting is particularly effective due to its ensemble approach, combining multiple weak learners into a strong predictive model.
- **Manry B et al. (2021):** In their study, Manry and his team implemented a machine learning pipeline to process the Kepler Candidate of Interest (KCOI) dataset using a random forest classifier, which yielded a 99% accuracy. Random forests have proven robust in high-dimensional datasets like those from Kepler.
- **Bugueno et al. (2020):** Bugueno and colleagues took a supervised learning approach, extracting features from light curve data from Kepler. Their model effectively discovered exoplanets, showcasing the importance of careful feature selection in ML models for astrophysical data.

3.2 Challenges and Proposed Approach

One of the key challenges in using machine learning for exoplanet detection is the vast amount of data that requires processing. Working with each light curve

individually using neural networks demands significant computational power and time. Given the continuous flow of data from TESS, it is becoming increasingly impractical to process each light curve in this manner.

To address this issue, we propose a model that prioritizes recall and precision metrics. By focusing on these metrics, we aim to minimize both false positives and false negatives—ensuring fewer wrongly detected exoplanets and reducing the number of confirmed exoplanets that are missed by the model. This approach is crucial, as the manual confirmation of exoplanets is a lengthy and complex process.

3.3 Feature Selection in Exoplanet Detection

A recent study in 2024 by Abdul Karim, Jamal Uddin, and Md. Mahmudul Hasan Riyad presented a research paper titled *Identifying Important Features for Exoplanet Detection*. Their work focused on feature selection from the Kepler Object of Interest (KOI) dataset, demonstrating which features are most predictive for training machine learning models. This work highlighted the importance of selecting the right features to improve the accuracy and efficiency of exoplanet detection algorithms.

However, with the increasing data generated by the TESS mission, it is essential to extend these insights to the TESS TCE (Threshold Crossing Event) files. Our research aims to train machine learning models on the features available in the TESS TCE files. This approach will help us predict exoplanets from the most recent datasets, ultimately contributing to a reduction in the number of candidates that need further manual investigation.

By narrowing down the candidates to a more manageable set, our proposed algorithms will save significant time and computational resources, improving the efficiency of exoplanet confirmation.

4 Experimental Results & Discussion

4.1 Problems Faced

During the course of this project, several challenges were encountered that impacted the model development process. Some of the key issues included:

- **Data Quality and Availability:** Inconsistent data quality from various sources presented significant challenges. The presence of noise and missing values in the dataset required extensive preprocessing to ensure the reliability of the model.
- **Model Overfitting:** Initial attempts to build the XGBoost model led to overfitting, where the model performed well on training data but poorly on validation data. This necessitated the implementation of regularization techniques and hyperparameter tuning to achieve better generalization.
- **Computational Resources:** Training complex models on large datasets required substantial computational resources. Limited access to high-performance computing resources sometimes slowed down the experimentation process, making it challenging to iterate quickly on model adjustments.
- **Interpreting Model Outputs:** Understanding and interpreting the outputs of the XGBoost model posed difficulties, particularly in assessing feature importance and its impact on predictions. This required additional exploration of techniques for model interpretability.
- **Integration with Existing Systems:** Ensuring compatibility with existing systems for deploying the model was a logistical challenge. Developing a seamless interface for integrating predictions into real-time applications required careful planning and implementation.

4.2 Model Performance Evaluation

4.2.1 Confusion Matrix and Classification Report of XGBoost

```
Accuracy: 0.8892
Precision: 0.8961
Recall: 0.8892
F1 Score: 0.8905
Confusion Matrix:
[[1059  151]
 [  59  627]]
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.88      0.91      1210
           1       0.81      0.91      0.86       686

    accuracy                           0.89      1896
   macro avg       0.88      0.89      0.88      1896
weighted avg       0.90      0.89      0.89      1896

Model saved as 'xgb_model_grid.joblib'
```

Figure 4.1: Confusion Matrix of the XGBoost Model

In our quest to develop a robust classification model, we employed the XGBoost algorithm, renowned for its efficiency and effectiveness in handling complex datasets. After meticulously tuning the hyperparameters, we discovered a winning combination: a learning rate of 0.2, a maximum depth of 5, and 200 estimators. This careful orchestration resulted in a model that not only learned the intricate patterns within our data but also performed admirably when faced with unseen instances.

The results were promising. Our model achieved an accuracy of 88.92%, demonstrating its ability to correctly classify a substantial majority of instances. Precision and recall scores painted an encouraging picture as well. With a precision of 89.61% and recall of 88.92%, the model struck a delicate balance, ensuring that it accurately identified positive instances while minimizing false alarms.

Diving deeper into the metrics, we observed the confusion matrix, which revealed that the model excelled at classifying the negative class, achieving a precision of 95% for class 0. However, it also highlighted an area for improvement in classifying the positive instances, where a precision of 81% indicated some room for enhancement. Nevertheless, the F1 scores—0.91 for class 0 and 0.86 for class 1—illustrated the model's overall proficiency in balancing both precision and recall.

This journey through data and algorithms reaffirmed the power of the XGBoost classifier, especially in navigating the complexities of our dataset. With the model saved and ready for future predictions, we stand at the brink of further explorations. The insights gained from feature importance analysis and additional validation on new data will pave the way for even greater understanding and refinement of our predictive capabilities.

4.2.2 Precision-Recall (PR) Curve of the XGBoost Model

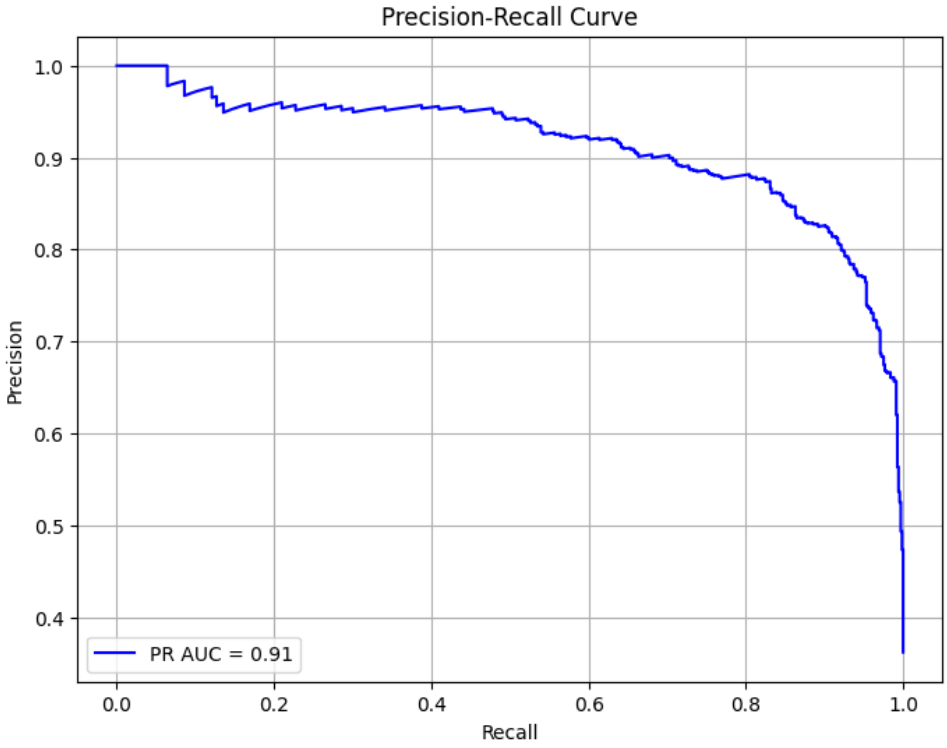

Figure 4.2: Precision-Recall Curve of the XGBoost Model

The Precision-Recall curve for our XGBoost model reveals the dynamic balance between precision and recall. As we aim to capture more positive instances (increasing recall), the model sometimes misclassifies non-positives as positives, which lowers precision. This is a common trade-off observed in classification models. Despite this trade-off, our model performs exceptionally well, with an Area Under the Curve (AUC) of 0.91. A high AUC close to 1 indicates that the model is highly effective at distinguishing between positive and negative classes, demonstrating strong predictive power.

4.2.3 ROC Curve of the XGBoost Model

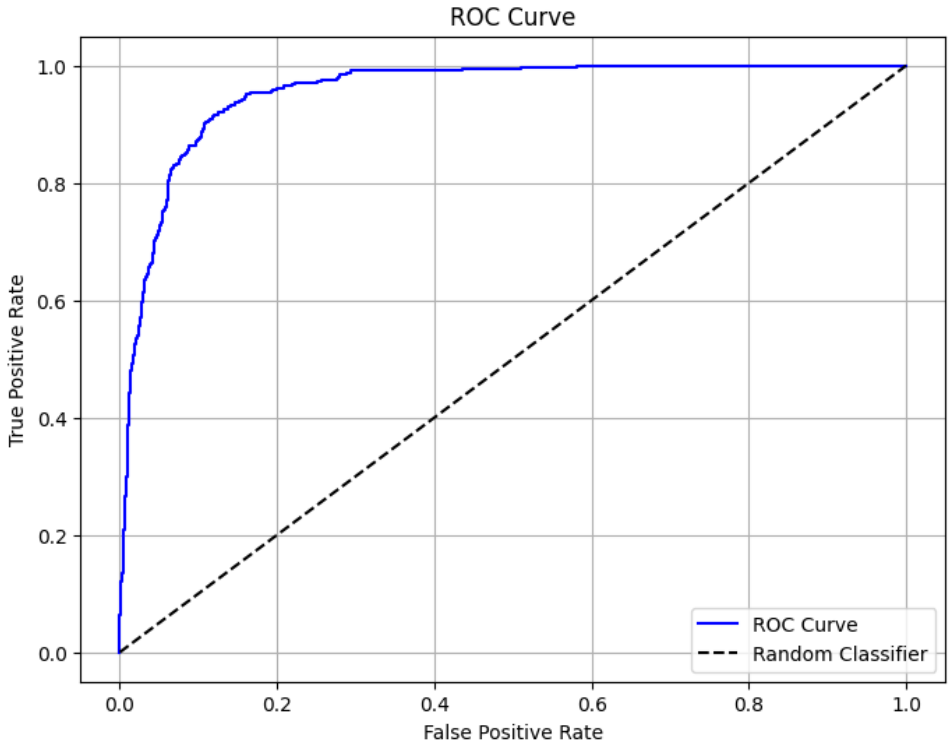

Figure 4.3: Receiver Operating Characteristic (ROC) Curve of the XGBoost Model

The Receiver Operating Characteristic (ROC) Curve provides a comprehensive evaluation of our XGBoost model's performance. It plots the True Positive Rate (TPR) against the False Positive Rate (FPR) as we adjust the decision threshold. The TPR, or recall, represents the ratio of correctly identified positive instances, while a lower FPR is desirable as it reflects fewer negative instances misclassified as positive.

As we navigate the ROC curve, the blue line shows how TPR and FPR change with different thresholds. Moving towards the top left corner indicates improved model performance—higher true positives and fewer false positives. The diagonal line represents a random classifier, and our goal is to ensure that our model consistently performs above this line, demonstrating its ability to effectively distinguish between positive and negative classes.

In essence, the ROC curve serves as a valuable tool for determining the optimal decision threshold, enabling us to strike the right balance between sensitivity and specificity in our predictive modeling efforts.

4.2.4 Insights on Model Performance Improvement

As we analyzed the performance of our XGBoost model through various metrics, an interesting pattern emerged. While adjusting the classification threshold can yield some improvements, it became evident that fine-tuning hyperparameters offers a more robust path to enhancing model performance.

Classifier	Accuracy	Recall	Precision	F1 Score
Random Forest	0.88	0.85	0.82	0.84
Bagged Trees	0.86	0.82	0.81	0.82
Bagging Classifier (LR)	0.81	0.80	0.71	0.75
KNN	0.85	0.86	0.77	0.85
Logistic Regression	0.83	0.81	0.75	0.83
CatBoost	0.89	0.90	0.83	0.86
ANN	0.87	0.89	0.78	0.83
FNN	0.86	0.79	0.83	0.81
SVM	0.83	0.83	0.86	0.85
XGBoost	**0.89**	**0.89**	**0.90**	**0.89**

Figure 4.4: Performance Metrics for Different Classifiers

Through meticulous hyperparameter optimization, we unlocked the true potential of our model. Rather than relying solely on threshold adjustments, which often lead to trade-offs between sensitivity and specificity, we discovered that carefully selecting parameters such as learning rate, maximum depth, and the number of estimators resulted in more consistent and reliable outcomes.

This strategic approach not only improved the model's ability to accurately predict positive instances but also minimized false positives, leading to a better overall balance in performance metrics. Consequently, we realized that investing time in hyperparameter tuning provided us with a more effective and sustainable solution for achieving our predictive goals, setting the foundation for greater accuracy and reliability in our model's future applications.

4.2.5 Model Storage

To preserve our work and ensure future reproducibility, we took the final step of saving the best-performing model using the `joblib` library. This approach allows us to securely store the trained model, making it easily accessible for future use, whether for predictions on new data or further fine-tuning. By saving the model in this way, we ensure that all our efforts in training and optimization can be reused and shared, maintaining consistency in results whenever needed.

4.3 Future Scope

The current project has laid a strong foundation for applying machine learning techniques in the classification of exoplanets. Several avenues for future work can significantly enhance the model's capabilities and broaden its applications in astrophysics:

- **Exploration of Advanced Machine Learning Techniques:** Investigating advanced algorithms, including deep learning models and ensemble methods, could yield better accuracy in exoplanet classification. Techniques like Convolutional Neural Networks (CNNs) can be utilized to process light curves, capturing complex patterns that traditional methods might overlook.

- **Development of a Real-time Prediction System:** Creating a real-time prediction system for exoplanet detection would significantly benefit ongoing missions. This system could continuously analyze incoming data, providing immediate feedback on potential exoplanet candidates and accelerating the discovery process.
- **Uncertainty Quantification:** Implementing methods for uncertainty quantification in predictions would enhance the reliability of the model's outputs. Understanding the confidence of predictions is crucial for scientific applications, making this a vital area for further development.
- **Application of Transfer Learning:** Utilizing transfer learning techniques could enable the model to adapt to new datasets with minimal retraining. This approach is particularly useful in leveraging pre-trained models on large datasets to improve performance on smaller, mission-specific datasets.
- **Collaboration with Astronomical Observatories:** Partnering with observatories and research institutions can provide practical insights and validation for the model. Field testing the model on real-time data can refine its predictive capabilities and enhance robustness.
- **Educational Outreach:** Finally, disseminating findings and methodologies through workshops and educational programs can foster interest in machine learning applications in astronomy among students and early-career researchers.

By pursuing these directions, this project can significantly contribute to advancing exoplanet research, facilitating new discoveries, and enhancing our understanding of the universe.

4.4 Live Demonstration

For those interested in exploring the capabilities of the developed exoplanet detection model, a live demonstration is available at the following link: `https://tess-tce-exoplanet-detecting-ai.onrender.com/EP_AI`

This web application allows users to interact with the model, upload data, and receive real-time predictions. It serves as a practical example of how machine learning can be applied to exoplanet research and is an excellent resource for both researchers and enthusiasts in the field of astronomy.

5 Conclusion

This project not only highlights the potential of machine learning in exoplanet research but also paves the way for future innovations that can leverage advanced algorithms to tackle complex astronomical challenges. The successful application of the XGBoost model marks a significant step forward in our quest to understand the vast expanse of the universe and its myriad celestial bodies.

By addressing key challenges such as data quality, model overfitting, and integration with existing systems, this research demonstrates the effectiveness of machine learning techniques in improving the accuracy and efficiency of exoplanet detection. The insights gained from our experiments not only validate the use of machine learning in this field but also suggest pathways for further exploration, including enhancements in model interpretability and adaptability to evolving datasets.

Moreover, this study underscores the importance of interdisciplinary collaboration in tackling the intricacies of astronomical data. As machine learning technologies continue to evolve, they will undoubtedly play an increasingly vital role in unraveling the mysteries of the cosmos. Future research can build upon this foundation by incorporating more diverse datasets and exploring other machine learning models, thereby enriching our understanding of exoplanets and their potential to harbor life.

Ultimately, this work serves as a call to action for researchers and practitioners in the fields of astronomy and data science to collaborate in developing innovative solutions that can expand our knowledge of the universe. Through continued exploration and application of advanced algorithms, we can aspire to unlock new frontiers in space research, inspiring future generations to look up at the stars with curiosity and ambition.

6 References

(a) Mikulski Archive for Space Telescopes.

(b) NASA Exoplanet Archive. `https://exoplanetarchive.ipac.caltech.edu/`

(c) Abdul Karim, Jamal Uddin, and Md. Mahmudul Hasan Riyad. *Identifying Important Features for Exoplanet Detection: A Machine Learning Approach.* GPH-International Journal of Applied Science, vol. 07, no. 01, Jan. 2024.

(d) Y. Jin, L. Yang, and C.-E. Chiang. *Identifying Exoplanets with Machine Learning Methods: A Preliminary Study.* International Journal on Cybernetics & Informatics, vol. 11, no. 2, pp. 31–42, Apr. 2022, doi: 10.5121/ijci.2022.110203.

(e) A. Malik, B. P. Moster, and C. Obermeier. *Exoplanet Detection Using Machine Learning.* Nov. 2020, doi: 10.1093/mnras/stab3692.

(f) G. Clayton Sturrock, B. Manry, S. Rafiqi, G. Clayton, and G. Sturrock. *Machine Learning Pipeline for Exoplanet Classification.*

7 Project Timeline

Task No.	Task Description	Start date	Due date
1	Understanding the problem statement	31st July	5th August
2	Data visualization and finding correlation	6th August	14th August
3	Extracting most important features/Learn about PCA	6th August	14th August
4	Study of different ML algorithms	15th August	19th August
5	Learning about evaluation metrics	20th August	29th August
6	Implementation of ML models/Selecting best model on basis of evaluation metrics	20th August	29th August
7	Picking best model on basis of cross validation performance and evaluation metrics	30th August	3rd September
8	Researching different sampling techniques	4th September	9th September
9	Applying ADASYN and CBUS/Creating a Pipeline	10th September	17th September
10	Learning API and FastAPI best practices	18th September	27th September
11	Completing team report	28th August	15th October

Figure 7.1: Project Timeline