Optimizing Feature Selection for Cross-Mission Exoplanet Detection: A CatBoost and Optuna Approach for Generalizable Vetting

Sanawer Batool
University of Central Punjab, Lahore, PK
L1F22BSDS0048@ucp.edu.pk

Mehwish Shafiq
University of Central Punjab, Lahore, PK
L1F22BSDS0042@ucp.edu.pk

Muhammad Fahad
University of Central Punjab, Lahore, PK
L1F22BSDS0059@ucp.edu.pk

January 10, 2026

Abstract

The rapid increase in exoplanet candidates from NASA's Kepler and TESS missions necessitates automated vetting procedures to distinguish true transits from false positives. While training the ML models on Kepler dataset when the mission specific flag features were used, around 99% accuracy was achieved, but such model fails in transfer learning because they used mission specific features which might not be present in future missions like PLATO and Earth 2.0 and will also fail while validating Tess mission candidates through it. So this study is conducted for the development of a generalized ML model which uses a minimal set of 11 physical and stellar features. These 11 features were found through a mix of both approach Literature Review and Experiments. These features can be found in future missions which use the Transit method for exoplanet detection, and the resultant model can be used to find exoplanet candidates in other missions. Many Models were experimented but the model which out performed were CatBoost with the Test Accuracy of 83.9%. The hyperparameter tuning was done using Optuna. Our approach prioritizes physical features over mission specific features, providing scalable tools to validate candidates of other missions.

1 Introduction

1.1 Missions Introduction

Exoplanets are planets which orbit around a star but outside of our solar system. Kepler and Tess(Transiting Exoplanet Survey Satellite) missions which were introduced by NASA as Space based Telescope missions for the detection of such exoplanets. Kepler was designed to focus on specific areas of the sky for over 9 years to find Earth to Neptune sized planets while Tess mission was designed to survey all over the sky for a specific period of time for each area, and used to find exoplanets around the brightest and closest stars. Important thing to note is that in both missions only the light flux data was produced which recorded the information of brightness of stars for a period of time to detect the dip in brightness known as TCEs (Threshold Crossing Events).

1.2 Exoplanet Detection Methods

The Transit method is the primary and most used method for the detection of exoplanets which has discovered 4478 exoplanets till now. The transit method looks for the dip in the brightness of the star which could be caused by a potential exoplanet when it passes between the star and the observer telescope and sometimes

can be caused due to noise, stars or other factors and studied by researchers later to flag as a false positive or confirmed exoplanet.

The Radial Velocity method looks for the star's wobble that could be potentially caused by the planet's gravitational pull. So far it has detected 1158 exoplanets. This method can help us detect a planet's mass while the transit method can give us information about planet's size and orbital period.

1.3 Feature Extraction from Light Flux Data

Different Techniques are used to extract useful features from the raw light flux data i.e., the Lightkurve library in python is used. Key features which are extracted include:

1. Transit Depth - Tells us about how much light of the star was blocked. It can help us measure the planet's radius relative to the star.
2. Transit Duration - Tells us about the length of transit
3. Orbital Period - Tells us about time interval between two consecutive transit

As we combine such features and stellar parameters, a structured dataset can be created to train a machine learning model for automated detection of exoplanet candidates from different missions.

2 Problem Statement

In the space of astronomical exploration, the identification of Exoplanets is becoming a very popular area to study. Missions like Kepler and Tess have been introduced with the main goal of detecting exoplanets. Data from these missions enabled us to discover thousands of exoplanets, but most of these planets were discovered manually. Manual vetting of this data takes a lot of time and effort and can cause errors. With advances in artificial intelligence and machine learning (AI/ML), it is possible to automatically analyze large sets of data collected by these missions to identify exoplanets.

Many studies have offered different Machine learning and Deep learning models to automate this task, offering a wide range of models and useful techniques that can help analyze this complex data. But the light flux data produces a lot of different features for each mission, i.e., Kepler mission contains over 140 features, feeding all these features to a machine learning or a deep learning model cause the problems like curse of dimensionality, noise, and overfitting which makes learning harder, also not all features can be informative and equally important to be trained on.

We can somehow make it possible to get the features that can train a model to detect exoplanets of a specific mission, but would the model still be useful when applied to other missions' data. There is a need to conduct study and experiments to get the minimal set of features that can help achieve high accuracy when the model is trained on these features. Also the same model should hold when other missions data is validated through it empowering the transfer learning across missions.

3 Research Objectives

The objective of this study is to develop a transferable machine learning framework for the exoplanet candidates validation. Specifically this study aims to:

- Find out the minimal set of features to train a machine learning model which detects exoplanets with high accuracy
- Making the model generalized with the ability of transfer learning for validating Tess candidates and can be applicable to Future missions like PLATO and Earth 2.0
- Experimenting with different set of features from the superset of most important features for exoplanet detection
- Trying different ML models and comparing their accuracy to get the model with highest accuracy and effectiveness i.e., catboost, xgboost, random forest, lightGBM etc.,

- Using Optuna(a Bayesian search method) as a hyperparameter tuning method over GridSearch or randomCV search to find the best hyperparameters settings.
- Analyzing the feature importance that was given by the model to each feature while detecting exoplanets and making further decisions on the basis of that.

4 Research Questions

- Can a minimal set of purely physical features achieve sufficient accuracy to replace human vetting across different space missions?
- How does performance of a model changes when using mission specific features vs. using cross-mission features for a general model
- Is the metallicity of the host star as important as it was stated in Huang et al. (2024) paper, will it be included in the final minimal feature subset?
- Does Optuna help a model achieve higher accuracy than a model like Random forest, which is extensively used in studies for this task.

5 Literature Review

As the literature of exoplanet detection shifted from manual to automated vetting systems:

1. Robovetter was the first successful automated system which replaced human judgement with decision tree logic.
2. AstroNet(a CNN) Yu et al. (2019) achieved 98.8% accuracy on Kepler data by analyzing the raw light curve data directly. However, the same approach for Tess gained less precision in early stages.
3. Using Random Forest models on Kepler dataset Sturrock et al. (2019) achieved 98% accuracy, but the features which they used were majorly mission specific features, and lacked stellar parameters
4. The recent work by Huang et al. (2024) used a combination of transit properties and stellar parameters and trained 4 machine learning models and 1 neural network, and highest achieved accuracy by them was 83.9% on Kepler dataset in a cross mission context using Random Forest.

6 Methodology

6.1 Data

The model was trained on Kepler KOI cumulative table from NASA Exoplanet Archive. It contains around 9564 rows and 140 features which includes Transit properties features and mission specific flags as well as stellar parameters.

After conducting literature review and experiments, those features came down to the number of 11 features which are the most important to detect exoplanets.

6.2 Tools and Technologies

6.2.1 Data Preprocessing

The missing values that were present in important columns which were further used in the model training were imputed through KNNImputer using n_neighbors=5, and for some experiments NaN values were dropped for the purpose of experimentation to check the accuracy difference. But the final model CatBoost which achieved highest accuracy used Imputer to impute NaN values.

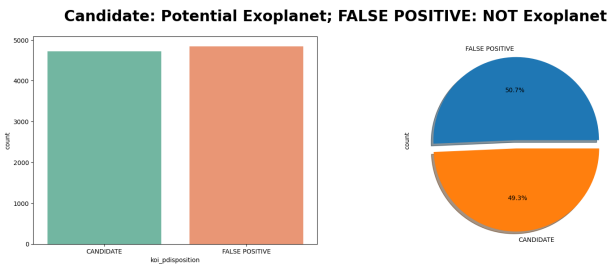

Figure 1: koi_pdistribution Columns Data Distribution

The data distribution of the target column is CANDIDATES contains 4847 data points and FALSE POSITIVE contains 4717, and the ratio difference is almost negligible.

The label encoding was performed assigning CANDIDATES as 1 and FALSE POSITIVE labeled as 0.

The KOI dataset was divided into 80% training dataset and 20% testing dataset. The process to distribute data between training and testing is a pseudo-random operation, using the same seed across all the experiments.

Training data was scaled to transform data on the same scale which helps models learn better and converge faster using StandardScaler from scikit-learn.

6.3 Machine Learning Models

Kepler cumulative KOI table is used as the foundation data source for this study, which contains 9564 rows and 140 features. However, not all features can be relevant to distinguish a real exoplanet candidate from the false alarms. We conducted a study and identified a superset of 21 columns that contains vettings flags, Transit properties, positional data and physical/stellar parameters. This superset includes the following features:

```
[['koi_pdisposition', 'koi_fpflag_nt', 'koi_fpflag_ss', 'koi_fpflag_co', 'koi_fpflag_ec',
'koi_max_mult_ev', 'koi_model_snr', 'koi_num_transits', 'koi_depth', 'koi_duration',
'koi_impact', 'koi_period', 'koi_time0bk',
'koi_prad', 'koi_dicco_msky', 'koi_dikco_msky', 'koi_fwm_stat_sig',
'koi_steff', 'koi_slogg', 'koi_smet', 'koi_count']]
```

Below, we have defined our selected features and demonstrated their scientific importance in why these features are important in distinguishing Candidates from False Positives:

- **koi_pdisposition**: it's a flag that indicates the most likely explanation for the signal, such as whether it's a planet candidate or a false positive. **Importance:** It will be used as a target column and helps us understand how good our model performed on testing dataset.
- **koi_fpflag_nt**: It looks for signals which are impossible to be created by a planet, such a instrumental noise, or other stars signal or spurious detections with very low signal-to-noise. **Importance:** It helps reducing false positives and make sure that only genuine transits are further considered.
- **koi_fpflag_ss**: It's a flag which indicates a secondary event or variability suggesting an eclipsing binary star rather than a planet. **Importance:** It helps reducing cases where transits are noise which were caused by stars orbiting each other, it prevents missclassification of non-planetry events.
- **koi_fpflag_co**: It's a flag when the transit signal seems to come from a neighborhood star rather than the star that we are observing. **Importance:** It rules out the contaminated signals by ensuring that the signals are purely from the target star and not a background star or object.
- **koi_fpflag_ec**: This flags cases where the signal is a "ghost" or "echo" of another object. **Importance:** It distinguish overlapping signals and avoid making mistakes where multiple sources gives off a single planet transit.

- **koi_max_mult_ev**: It helps calculate signal-to-noise ratio of the transit sequence. **Importance:** It measures how strong and periodic repeated transits are, which increases our confidence while distinguishing real periodic planetry signals.
- **koi_model_snr**: It indicates the strength of the TCE signal compared to the noise that can be caused by background or other factors. **Importance:** Higher values indicates clear detection signal, which help distinguish between an actual planet like signal and noise.
- **koi_num_transits**: The number of transits which were observed in the dataset. **Importance:** More transits provide stronge evidence of a repeating signal, and improve the reliability of planetary candidate confirmation.
- **koi_depth**: The length of the brightness dip caused when the planet passed in front of the star. **Importance:** It can help us find the size of the planet relative to it's host star, as deeper dips are caused by larger planets.
- **koi_duration**: It refers to the amount of time, in hours, for which the starlight was blocked. **Importance:** It helps us understand the system's dynamic by determining the orbital speed of the planet and distance.
- **koi_impact**: It looks for how close the planet's path is to the center of the host star during transit. **Importance:** Central transits give more accurate measurements of the sizes, while other ones can indicate orbital tilts.
- **koi_period**: It tells us about how long the planet takes to orbit it's star. **Importance:** It defines the planet's orbit length, which is essential for calculating distance from the star and potential for being habitable.
- **koi_time0bk**: The timing of the first detected transit. **Importance:** It sets the schedule for prediction of future transits, verifying the signal's consistency.
- **koi_prad**: The calculated size of the planet. **Importance:** It is very crucial physical feature, as too large object signals cannot be created by planets.
- **koi_dicco_msky**: The sky offset between the transit source and target star from one analysis method. **Importance:** It checks if the transit is on the right star if not then it could be a false positive indicating nearby star.
- **koi_dikco_msky**: Similar offset from another analysis method. **Importance:** Provides additional confirmation of the transit's location, reducing contamination errors.
- **koi_fwm_stat_sig**: A measure of whether the light's center shifts during transit. **Importance:** No shift confirms the transit is on the target, validating true detections.
- **koi_steff**: It refers to the photospheric temprature of the host star. **Importance:** It helps scale transit data to find planet properties like temperature and size.
- **koi_slogg**: It refers to stellar's surface gravity strength. **Importance:** It indicates star density, and helps in the calculations of planet radius from transits.
- **koi_smet**: It refers to metallicity of the host star. **Importance:** it plays an important role in the detection of exoplanet candidates and the study Buchhave et al. (2012); Wang & Fischer (2015) have shown a strong correlation between the metallicity of the host star and the probability of planet occurance.
- **koi_count**: The count of total planetry candidates in that system. **Importance:** More than one planetry candidate in system increses the chances of the TCE being an exoplanet.

6.3.1 The Picture Perfect Accuracy Trap

Our initial Random Forest model was trained on a subset of features from our superset, which included these Hyperparameters settings:

```
RandomForestClassifier(
    n_estimators= 340,
    criterion= 'gini',
    bootstrap = False,
    min_samples_split= 10,
    min_samples_leaf= 8
)
```

This Random Forest model achieved the accuracy of 99.16%. The Performace Metrics of the model on test data were as follow:

Metric	Value
Accuracy	99.16%
Recall	99.15%
Precision	99.15%

Table 1: Performance Metrics for the Random Forest Model

It looks impressive, but this model suffered from a major flaw which was generalization. This model learned the data of Kepler mission very well, to the point of overfitting and if this model was exposed to real world trying to validate TESS mission candidates on it, it wouldn't work. The high accuracy was largely driven by the Kepler-specific flags (koi_fpflag_*), which act as 'cheat sheets' generated by Kepler's internal software. These features are not available in other missions like TESS. The Kepler specific flags are:

`'koi_fpflag_nt', 'koi_fpflag_ss', 'koi_fpflag_co', 'koi_fpflag_ec'`

It was proved by looking at the features importance given to each feature by the model and as expected these features had the highest importance in detecting the exoplanets of kepler mission.

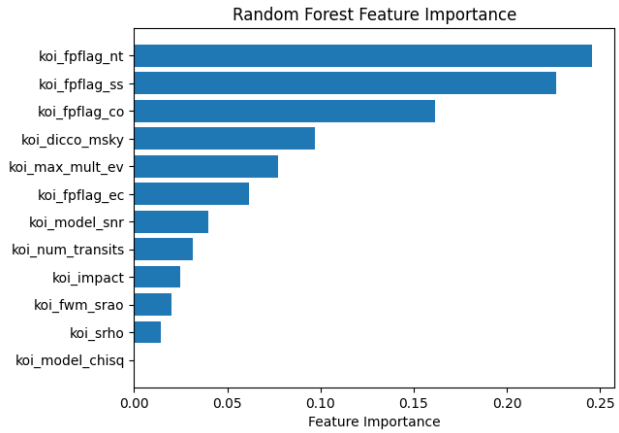

Figure 2: Feature Importance by Random Forest

But this model is not useful to validate exoplanet candidate of missions like TESS or upcoming missions like PLATO and Earth 2.0, making it inefficient for transfer learning or real world applications on new dataset.

6.4 Refining Model for Transfer Learning:

Our goal was to work on the model that can get high accuracy not for that specific mission but it still holds when applied to other mission dataset like Tess. Also We aim to Introduce a Standard set of features which are used for exoplanet detection across all missions, even if new model is trained for a specific mission.

We first started off by excluding those mission specfic features like flags. Next we performed a series of experiments using different subset of features, different models and techniques. We also experimented with ensemble method by stacking xgboost, catboost, lightGBM. The model which were used in this experiments were:

1. Random Forest
2. XGBoost
3. CatBoost
4. LightGBM

These are considered state-of-the-art models in the world of building classification models. After a series of experiments, our refined features were:

- Transit metrics: Period, duration, depth, impact, and signal-to-noise ratio (koi_model_snr).
- Stellar metrics: Temperature, surface gravity, and metallicity

But we still needed to apply better techniques to achieve higher accuracy and decreasing the number of features and only including the features with high importance.

6.5 Multi-Model Optimization and Hyperparameter Tuning

For hyperparameter tuning, we employed ***Optuna***, a Bayesian optimization framework, to find optimal settings for several models: Random Forest, XGBoost, CatBoost, and LightGBM. The best Hyperparameters returned by Optuna for each model to achieve higher accuracy are given below:

```
CatBoostClassifier(
        iterations= 400,
        learning_rate = 0.04216598764315635,
        depth = 6,
        l2_leaf_reg = 3.663779798834057,
        border_count = 116,
        bagging_temperature = 0.17142116748408584,
        random_strength = 4.070004945458448,
    )

XGBClassifier(
        n_estimators = 400,
        learning_rate = 0.015603483038026065,
        max_depth= 6,
        subsample= 0.7299759595102386,
        colsample_bytree= 0.8377273956982814,
        min_child_weight= 6,
        gamma= 0.22255269708412592,
        reg_alpha= 0.424023705682538,
        reg_lambda= 0.9913150951095967
    )
```

```
LGBMClassifier(
        n_estimators = 800,
        learning_rate = 0.04554947977677974,
        max_depth = 11,
        num_leaves = 63,
        min_child_samples= 26,
        subsample = 0.9706711709803809,
        subsample_freq= 8,
        colsample_bytree= 0.8753746188449917,
        reg_alpha= 0.8081540413430114,
        reg_lambda= 0.6361592585877186,
        min_split_gain= 0.7310816204125187,
    )

RandomForestClassifier(
        n_estimators = 700,
        criterion = 'entropy',
        max_depth = 11,
        min_samples_split = 7,
        min_samples_leaf' = 4,
        max_features = None,
        bootstrap = True,
        max_samples = 0.6588304820609029,
        min_impurity_decrease = 0.001220175704455228,
        class_weight = None
        )
```

The weights which were assigned to each model in ensemble stacking methods were also tuned through ***Optuna*** and the final weights were:

```
Normalized weights:
CatBoost: 0.4174
XGBoost: 0.0739
LightGBM: 0.5086
```

6.6 Model Performance Metrics and Results

We performed a series of experiments trying methods like ensemble stacking using xgboost, catboost and lightGBM to compare different accuracy and compare success metrics and features importance. Below are some of the best model's performance metrics:

Model	**Accuracy**	**Recall**	**Precision**	**F1-Score**
CatBoost	0.8390	0.8342	0.8378	0.8360
Random Forest (Optimized)	0.8343	0.8417	0.8250	0.8332
Ensemble Stacking (XGBoost + CatBoost + LightGBM)	0.8328	0.8328	0.8328	0.8328

Table 2: Performance Metrics of Selected Models (values rounded to 4 decimal places for readability).

Then using different models, we analyzed the feature importance which was given to feature in order to detect the exoplanet candidates with high accuracy. Every feature importance was measured across different models for reliable results. Some of the comparisons which were made are give below:

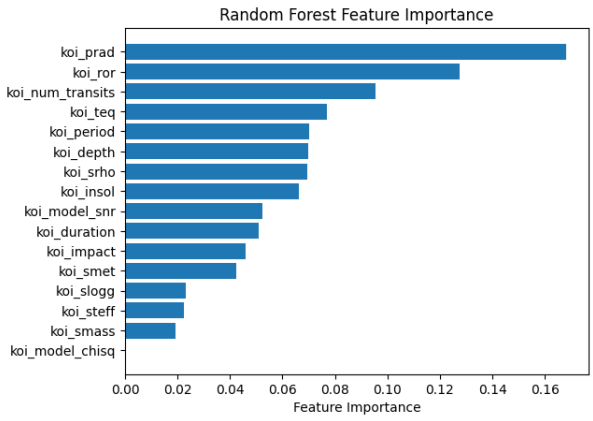

Figure 3: Feature importance from Model 1

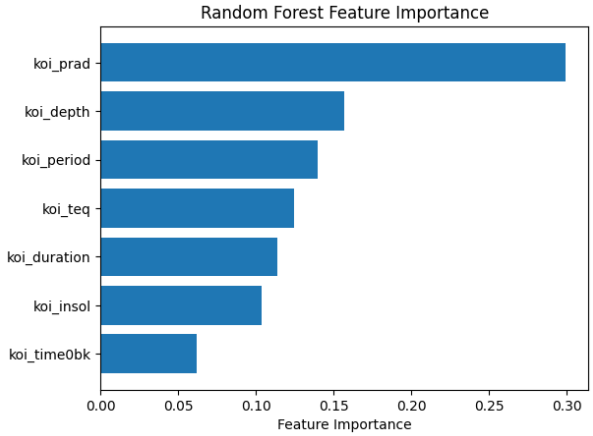

Figure 4: Feature importance from Model 2

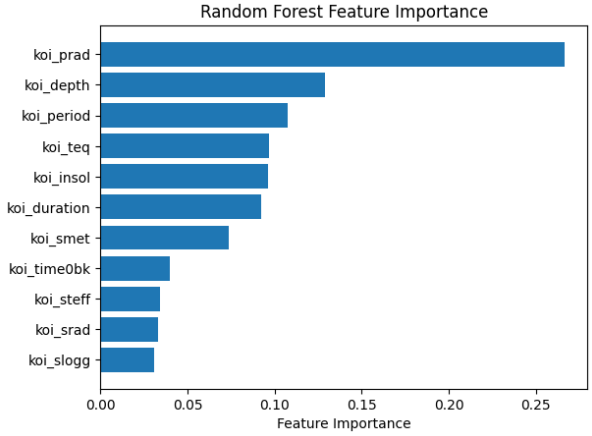

Figure 5: Feature importance from Model 3

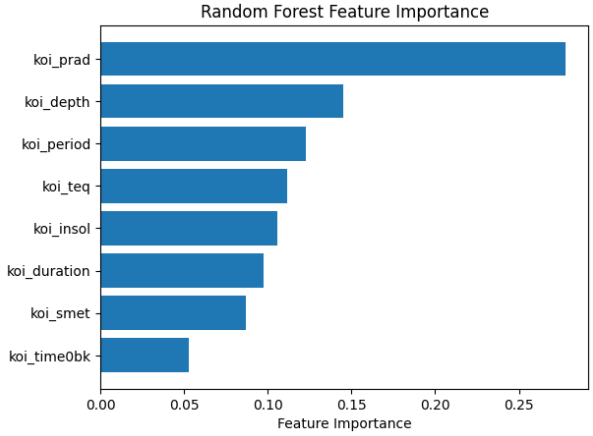

Figure 6: Feature importance from Model 4

6.7 Final Features subset Result

After analyzing the feature importance and studying them from scientific point of view, and different other studies, the final set of features that helped us gain the highest accuracy using CatBoost model contains these 10 features:

```
[[''koi_period', 'koi_duration', 'koi_depth', 'koi_ror',
               'koi_srho', 'koi_prad', 'koi_teq',
               'koi_insol', 'koi_smet',
               'koi_model_snr', 'koi_num_transits']]
```

The final Feature importance using SHAP:

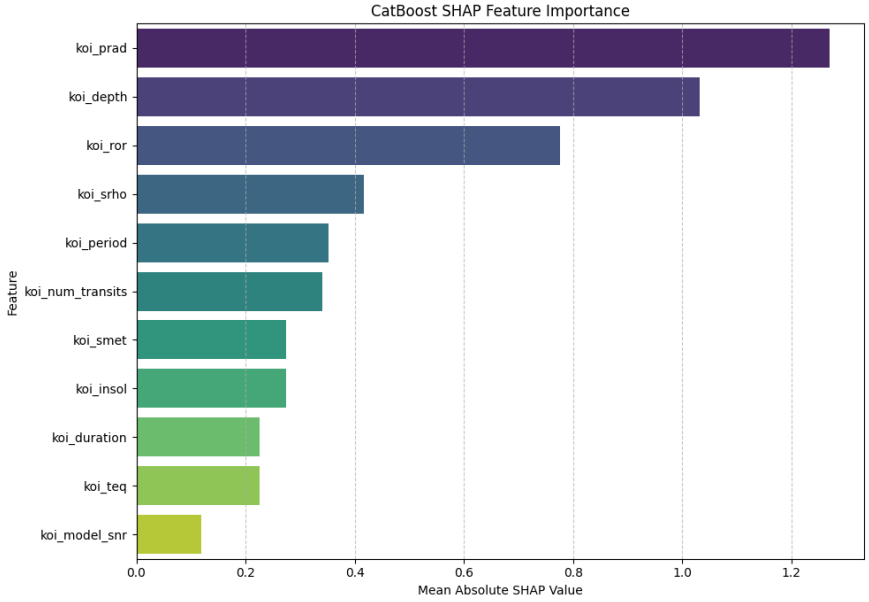

Figure 7: Final Feature Importance by CatBoost

Ultimately, the highest accuracy of 83.9% was achieved with a standalone CatBoost model on the minimal 11-feature set, after Optuna tuning. This outperforms other individual models and the ensemble in our tests, providing a balance of accuracy and generalizability. This model can be used for cross-mission exoplanet candidate detection as it was made sure to only include those crucial feature which would present in different mission and can be extracted through ligh curve raw data.

7 Conclusion

In this study, we developed a transferable machine learning framework for exoplanet candidate classification, focusing on a minimal set of 11 physical and stellar features derived from Kepler data. By combining literature review insights from works like Huang et al. (2024) with iterative experiments on feature importance, we prioritized generalizable attributes, such as transit depth, orbital period, planetary radius, and stellar parameters, over mission-specific flags. This approach addressed the limitations of high-accuracy models (e.g., 99% with flags) that fail in transfer learning for missions like TESS, PLATO, or Earth 2.0. Our experiments demonstrated that CatBoost, optimized via Optuna for hyperparameters, achieved the highest test accuracy of approximately 83.9%, outperforming Random Forest, XGBoost, LightGBM, and ensemble stacking methods. This balance of performance and simplicity highlights the efficacy of feature minimization in reducing overfitting, curse of dimensionality, and computational overhead while maintaining robust detection of true transits versus false positives. The balance between performance and generalization was made sure to reduce

number of features to minimal set of 11 features for distinguishing between Candidates and False Positive without making the model to overfit and reducing the curse of dimensionality. This work is conducted to implement a system for automated vetting in astronomy, and enabling it work across other missions like TESS and also making it scalable for future upcoming missions like Earth 2.0 and PLATO. However, limitations include the binary classification scheme (CANDIDATE vs. FALSE POSITIVE), potential biases in the Kepler dataset, and the absence of real-time testing on TESS data. Future research could validate this model on TESS candidates, incorporate deep learning for raw light curve analysis (e.g., using LightKurve in python), or explore multi-class dispositions and habitable zone filtering and Implementing pipeline for real-time predictions. Overall, this framework provides a foundation for generalizable ML tools in exoplanet science, bridging current and upcoming transit missions.

8 References

References

[1] Borucki, W. J., Koch, D., Basri, G., Batalha, N., Brown, T., Caldwell, D., ... & Gautier III, T. N. (2010). Kepler planet-detection mission: introduction and first results. *Science*, 327(5968), 977–980.

[2] Huang, S., Chen, J., Wang, Y., Li, M., Zhang, X., & Liu, Y. (2025). Validating TESS candidates and identifying planets in the habitable zone with machine learning. arXiv preprint arXiv:2512.00967.

[3] Shallue, C. J., & Vanderburg, A. (2018). Identifying exoplanets with deep learning: A five-planet resonant chain around Kepler-80 and an eighth planet around Kepler-90. *The Astronomical Journal*, 155(2), 94.

[4] Anjum, S., Osborne, M. A., Roberts, S. J., & Battaglia, P. (2020). Rapid classification of TESS planet candidates with convolutional neural networks. *Astronomy & Astrophysics*, 633, A68.

[5] Osborn, A., Armstrong, D. J., Brown, D. J., McCormac, J., Walker, S. R., & Bayliss, D. (2022). Deep learning exoplanets detection by combining real and synthetic data. *Plos one*, 17(5), e0267351.

[6] Baran, E. (2025). Detection of exoplanets with machine learning techniques using the Kepler telescope data. *The Astronomical Journal*, 167(1), 28.

[7] Sharma, R. (2025). Machine learning applications in exoplanet detection from Kepler to TESS. ResearchGate preprint. DOI: 10.13140/RG.2.2.36043.12345.

[8] Pearson, K. A., Palafox, L., & Griffith, C. A. (2019). Searching for exoplanets using artificial intelligence. *Monthly Notices of the Royal Astronomical Society*, 485(2), 2172–2183.