UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

UNIVERSITAT POLITÈCNICA DE CATALUNYA

FACULTAT D'INFORMÀTICA DE BARCELONA

Machine Learning Techniques for Exoplanet Detection with NASA Kepler Data

Machine Learning Course Final Project, June 2023

Authors:
León Villapún, Luis Alfredo, *email*: luis.alfredo.leon@estudiantat.upc.edu
Ogbuchi, Chidibiere, *email*: chidiebere.ogbuchi@estudiantat.upc.edu
Supervised by:
Prof. Marta Arias Vicente, *email*: marias@cs.upc.edu
Prof. Bernat Coma Puig, *email*: bernat.coma@upc.edu

Contents

List of Figures

List of Tables

1 Abstract

The interpretation and characterization of exoplanets from astronomical observations pose significant challenges due to the presence of false positives which arise from instrumental noise, stellar activity, and other kinds of astronomical phenomena. In this study, we explore the utilization of various machine learning classification methods to distinguish whether an observation can be considered a candidate exoplanet or should be labeled as a false positive. Furthermore, our aim is to propose a model that can effectively help researchers reduce or validate future observations. For this matter, we are utilizing labeled data from the NASA Exoplanets dataset, extracted from the Kepler Space Telescope. This work compares 4 Machine Learning Classification techniques, namely Support Vector Machines, Random Forests, MultiLayer Perceptron and XGBoost. The findings suggest that while the four techniques show promising results, the Random Forest approach is able to better extrapolate the patterns on the data both with and without precomputed features. This study is an effort to contribute to the ever growing field of exoplanet detection, and future work could explore the possibilities of implementing deep-learning methods to enhance the abilities of the existing approaches.

2 Introduction

NASA's Kepler Mission includes as one of its main objectives the detection and monitoring of exoplanets capable or not of hosting extraterrestrial life. Launched in 2009 and retired in 2018, the Kepler telescope was capable of discovering more than 2600 exoplanets, generating tons of data with its multiple instruments. Missions of this kind will be more common in the future, and it is necessary to count with preliminary analysis of what are the main characteristics that constitute a candidate exoplanet, enhancing the accuracy of the detectors and allowing scientists to focus on more promising observations.

The goal of this study is to find a machine learning model that serves as either filter or "second opinion" for candidate exoplanets among proposed observations. The dataset picked for this work was initially provided by NASA[1], and it comes with a set of both raw and preprocessed data that we will further evaluate in the next section. Data generation and formalities can be reviewed at the Kepler Data Processing Handbook, where all definitions on the processes to obtain the data are placed [Jen17]. Other valuable repositories to extract this sort of dataset include those of Caltech and the Mikulski Archive for Space Telescopes[2]. For this work, however, we will stick to the provided Kaggle NASA dataset, as it provides an intuitive introductory structure to the problem at hand: how can we, provided with a repository of preprocessed observations, better estimate their adequacy to label them as candidate exoplanets or just as false positives?

This paper discusses the overall findings of the experimentation performed, however it is also suggested to review the annotated Jupyter Notebook[3] for the project.

3 Implementing Submission

The submission consists of two files: The python notebook and the exoplanet.csv file.

[1] To download the original dataset.

[2] Caltech and Mikulski.

[3] To run the notebook, please go to our Google Colab notebook, you can attach a copy on your personal Google Drive and run it there. To extract the dataset, you will need a Kaggle API user and token, or download manualy the dataset and save it inside your Google drive folder.

The notebook is divided into several section for easy execution. It is more efficient to run the notebook on a Jupyter environment, preferably Google colab.

3.1 Requirements

Ensure the notebook is properly active. Then, Import all the required libraries as seen in the notebook.

3.2 Loading the Exoplanets Dataset

The dataset is sourced from Kaggle and can be accessed at the following URL: "https://www.kaggle.com/dataset or can be found attached in this submission.

To load the dataset, you need to create a directory on your googles drive and upload the exoplanets file to directory "/content/drive/My Drive/ML/sources/exoplanets.csv". otherwise, the data can be loaded from Kaggle by inputting your Kaggle credentials when prompted to.

3.3 Execution

After confirmation that the file has been successfully loaded, you can now execute the rest of the notebook which includes Pre-processing, Modeling and Comparison of Results.

4 Previous Work

There are multiple online projects related to this particular dataset[4], as NASA itself encourages students, researchers, and analysts to thoroughly study this data in an effort to extract the best insights and methods. However, most of these projects are nothing but introductory analysis, lacking a more formal approach to the dataset at hand. For instance, the mentioned online projects are including features that can lead to bias. Their use is subject for discussion, but as we will review on this study, it is better to deal with these features carefully.

In this context, we will consider any precomputed values as potentially leading to bias, as any human-labeled feature is subject of external observation. This is the case for values such as *koi score* and the numerous *fp flag* on the dataset.

Multiple more rigorous and formal approaches have been published as well. It is worth mentioning the work from [JYC22] and [Vis+22]. Both studies evaluate the use of different supervised ML techniques, arriving to interesting results. Another point to highlight is that one of the studies includes the precomputed values as valid features, while the other one discards them.

5 Data Exploration

Upon analysis of the original dataset encompassing 49 distinct columns, it has been determined that three of these constitute identifiers (*kepid*, *kepoi name*, *kepler name*). Furthermore, two variables have been identified that can effectively operate as target classes, specifically *koi disposition* and *koi pdisposition*. The remaining columns are primarily descriptive features pertinent to each observation, as well as flags and scores that we will discuss later. It is also important to highlight that the dataset is composed of 9564 observations, which makes it a small to medium-sized dataset, suitable for many different approaches to discuss in the next sections.

[4]Namely, arashnik's Kaggle notebook, Garza's analysis, and Araujo's approach.

5.1 Target classes

It is important to note that the variable *koi disposition* presents with three potential outcomes: CONFIRMED, CANDIDATE, and FALSE POSITIVE. In contrast, the variable *koi pdisposition* is binary in nature, offering only two potential results: CANDIDATE and FALSE POSITIVE.

Given the scope and objectives of this investigation, it has been resolved that *koi pdisposition* will be utilized as our target variable. This decision is backed by the consideration that the models derived from this variable will be more efficient in determining whether to categorize a candidate as a FALSE POSITIVE, as well as in the logic that a CONFIRMED attribute can be also considered a former CANDIDATE. Hence, we are going to deal with a case of Binary Classification.

5.2 Feature analysis and preprocessing

5.2.1 Initial Data Inspection and Features Distribution

An important part of the initial analysis task was to evaluate the distributions of the features on the dataset. This was primarily achieved through the visual representation of these distributions, allowing us to draw significant inferences which were subsequently researched. Our investigation revealed that the columns labeled with 'flag' follow a Bernoulli binary distribution. Further consultation with the original data dictionary drew us to the understanding that these columns should also be excluded from our analysis. The logic for this exclusion stems from the fact that these columns are preannotated values and, as such, their inclusion could potentially introduce bias into our analysis. The same occurs with the *koi score* feature, which is, indeed, a pre-annotated value on the dataset. Curiously enough, the work from [JYC22] includes these variables as part of their analysis. On the other hand, [Vis+22] seems to have excluded them. We have explored both approaches for this study.

5.2.2 Splitting the dataset

In an effort to avoid data leakages, we split the dataset into train and test sets before doing any further modification to the data. This preemptive measure ensures the preservation of the testing set's integrity, keeping it free from any unattended influence that might compromise its validity. The split was done via the standard method from scikit learn[5]. The partition was made with a fraction of 0.8 for the training set and 0.2 for the test set, with a random seed of 42 and specifying stratification to preserve the distributions of the target feature labels.

5.2.3 Missing Values

Following the preliminary exploration of the dataset, our initial analytical step was to assess the extent of missing values, as depicted in Figure 1. Subsequently, we eliminated the columns wherein the proportion of missing data exceeded 50 percent. This decision was made on the consideration that such a high proportion of missing values presents significant challenges for data imputation and could potentially result in misleading conclusions.

As for the remaining features, we first tried to employ the K-Nearest Neighbors Imputation strategy, utilizing the KNNImputer. The KNN imputer estimates missing values based on the values of neighboring samples.[6] module from the Scikit-learn library, with k set to 10.

[5] train, test, split
[6] knn imputer

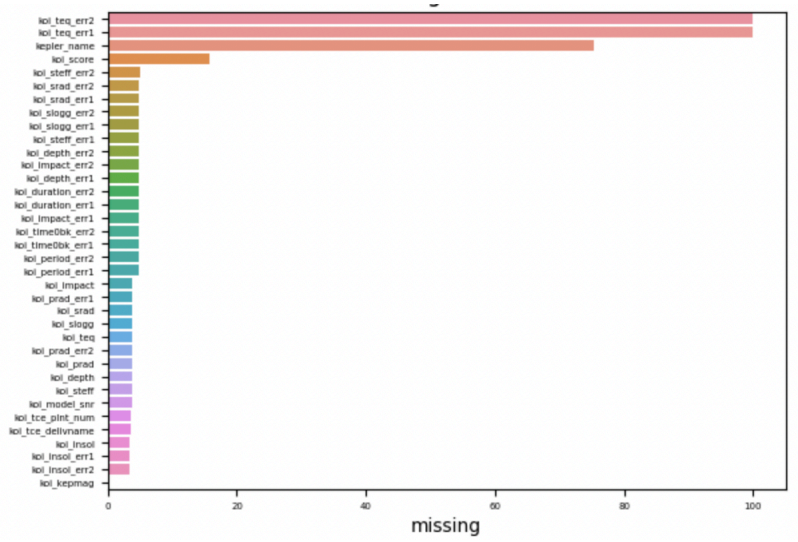


Figure 1: Missing value percentage per column

However, due to the high percentage of null values in the dataset, the imputation was generating too much noise, which led us to decide to simply drop the null values from the dataset.

5.2.4 Outlier Detection

Once we have imputed the values, we proceeded to detect outliers. To filter them out, two strategies were followed.

First, we performed Z-score outlier detection on the dataset, where we considered, per feature, the values whose z-score was below 4. The Z-score measures the number of standard deviations a data point is away from the mean. By setting a threshold of 4 for the Z-score, we consider any data point that is more than 4 standard deviations away from the mean as an extreme outlier. Removing outliers with a Z-score of 4 helps to eliminate data points that are significantly different from the majority of the dataset and may have a disproportionate impact on the classification model's performance..

Subsequently, we implemented a LocalOutlierFactor[7] [Bre+00] with five neighbors to detect additional anomalies. Observations bellow the 5th percentile were considered as outliers and these were filtered out as well from the training data.

5.2.5 Feature Engineering

Feature engineering involves creating new features derived from existing ones to capture additional information relevant to the classification task. In this preprocessing pipeline, several derived features are created from numerical variables, such as the ratio of planet radius to stellar radius, period-to-effective temperature ratio, depth-to-stellar radius ratio, and a combined metric involving insolation and planet radius. To further enhance the classification task, categorical features are created based on numerical features. The original numerical features are binned into categories using appropriate thresholds. This conversion allows the classification models to capture non-linear relationships between the features and the target variable. We categorized the duration, planet's radius, temperature equilibrium and model signal to noise ratio.

[7] local outlier factor

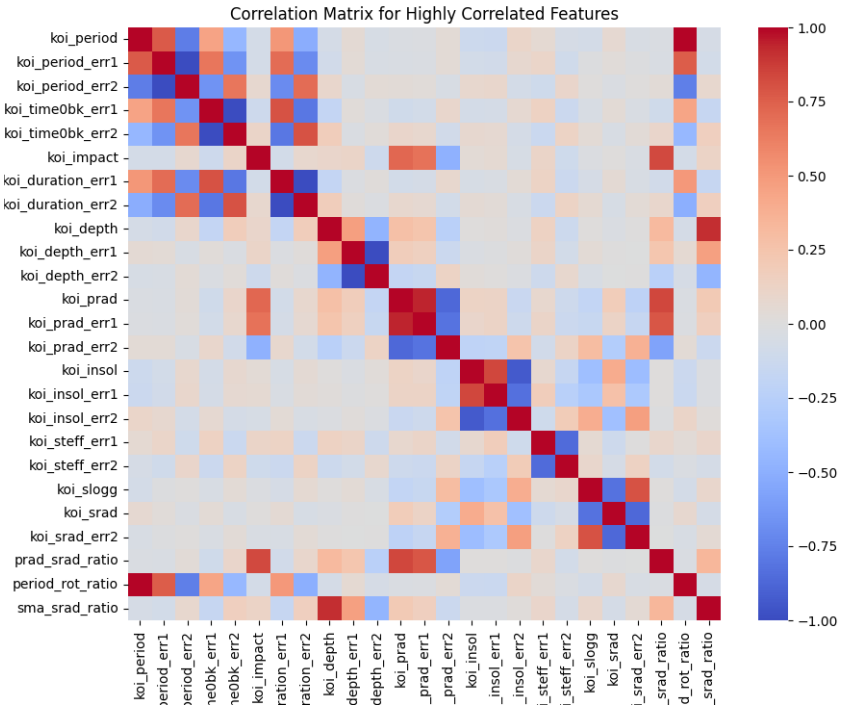


Figure 2: Correlation Matrix for Highly Correlated Features

5.2.6 Normalization

Normalization is applied to ensure that all features are on a similar scale and have equal importance during model training. The Min-Max scaling technique is employed to normalize numerical features, except for a few categorical features that are not subjected to normalization. We normalize at this point to stabilize model convergence, preserve relative relationships as well as the interpretability of the features during classification.

5.2.7 Dimensionality Reduction

To address multicollinearity and improve model interpretability, irrelevant features are dropped from the dataset. The decision to remove these features is based on correlation matrix, domain knowledge and feature importance analysis.

6 Methodology

6.1 After Pre-process

After preprocessing the dataset aforementioned, We selected select models based on their effectiveness in handling classification problems and their potential to provide accurate predictions for distinguishing between exoplanet candidates and false positives.

6.2 Model Selection

In the model selection and training phase, we choose between different classification models, including Random Forest, Support Vector Classifier (SVC), Multi-layer Perceptron (MLP), and

XGBoost, to predict the disposition of exoplanets as either candidates or false positives. Each model selection are justified based on the following specific criteria and considerations.

Random Forest:

1. **Ensemble Learning:** Random Forest is an ensemble learning method that combines multiple decision trees to make predictions. This ensemble approach helps capture complex relationships between features and the target variable, which can be beneficial for predicting the disposition of exoplanets [Bre01].

2. **Handling Imbalanced Classes:** Random Forest has the capability to handle imbalanced datasets effectively. In the case of exoplanet classification, where the number of candidate planets and false positives are imbalanced, Random Forest can mitigate the impact of class imbalance and provide reliable predictions for both classes [CLB04].

3. **Feature Importance Analysis:** Random Forest provides a measure of feature importance[8] based on the mean gain of each feature across the ensemble of trees. This analysis can help identify the most influential features in determining the disposition of exoplanets and provide insights for feature selection and engineering [Bre01].

Support Vector Classifier (SVC):

1. **Non-linear Classification:** SVC is a powerful model for non-linear classification tasks. It uses kernel functions to transform the input data into a higher-dimensional space, enabling the separation of complex patterns. This flexibility makes it suitable for capturing non-linear relationships between features and the target variable in the exoplanet classification task [CV95].

2. **Margin Maximization:** SVC aims to maximize the margin between the decision boundary and the support vectors, which are the data points closest to the boundary. This margin maximization approach helps improve the model's generalization ability and can enhance its performance in accurately classifying exoplanet candidates and false positives [CV95].

3. **Predictive Accuracy:** SVC has demonstrated high predictive accuracy in various domains, including astronomy and astrophysics. By selecting appropriate kernel functions and tuning hyperparameters, SVC can effectively classify exoplanets and provide reliable predictions [CS08].

Multi-layer Perceptron (MLP):

1. **Non-linear Relationships:** MLP is a type of artificial neural network that can capture non-linear relationships between features and the target variable. It consists of multiple layers of interconnected neurons, allowing it to learn complex patterns in the data. This ability is advantageous for the prediction of exoplanet dispositions, as the underlying relationships can be highly non-linear [RHW86].

2. **Flexibility and Adaptability:** MLP is a flexible model that can handle various data types and adapt to different feature types and distributions. It can learn and adjust its weights and biases during training, making it capable of capturing intricate patterns and improving classification performance [HTF09].

[8] forest importances

3. **Universal Approximation Theorem:** MLP is theoretically proven to be a universal function approximator, meaning it can approximate any continuous function with arbitrary accuracy given enough hidden units. This property makes MLP suitable for modeling complex relationships in the exoplanet classification task [HSW89].

XGBoost:

1. **Gradient Boosting:** XGBoost is an optimized implementation of gradient boosting, which combines multiple weak learners (decision trees) to create a strong predictive model. It sequentially builds new trees to correct the mistakes made by previous trees, leading to improved model accuracy. This boosting technique can enhance the prediction of exoplanet dispositions by effectively learning from previous model errors [CG16].

2. **Handling Imbalanced Classes:** XGBoost includes techniques such as weighted loss functions and subsampling of the training set to address class imbalance. These techniques can improve the model's performance when dealing with imbalanced datasets, ensuring reliable predictions for both exoplanet candidates and false positives [CG16].

3. **Predictive Power:** XGBoost has demonstrated excellent predictive power in various machine learning competitions and real-world applications. It is known for its ability to capture complex patterns in the data and provide accurate predictions, making it a suitable choice for predicting the disposition of exoplanets [CG16].

These selected classification models (Random Forest, SVC, MLP, and XGBoost) offer a combination of ensemble learning, non-linear modeling capabilities, handling of imbalanced classes, and feature importance analysis. By leveraging these models, we aim to improve the accuracy and reliability of predicting the disposition of exoplanets as either candidates or false positives.

6.3 Feature Encoding

Ordinal Encoding is used to transform categorical features into numerical representations. This encoding maintains the ordinal relationships between categories and allows the models to interpret the encoded features appropriately. One-hot encoding is then applied to convert the ordinal encoded features into binary representations. This step ensures compatibility with models that require numerical inputs.

6.4 Cross-Validation and Evaluation

Cross-validation is performed using the `cross_val_score`[9] function with 8 folds. This technique estimates the model's performance by evaluating it on multiple subsets of the training data, helping to assess its generalization ability and potential overfitting. The mean accuracy is calculated as the average of the cross-validation scores, providing an overall measure of model performance.

6.5 Performance Evaluation

The classification report[10] is generated using the predicted and actual target values. This report provides metrics such as precision, recall, and F1-score for each class. It helps evaluate the model's performance in terms of its ability to correctly classify exoplanet candidates and false positives.

[9] cross validation
[10] classification report

6.6 Feature Importance Analysis

To gain insights into the significance of different features, the code provides an option to analyze feature importance using the mean gain method. The importance of each feature is calculated based on the model's `feature_importances_` attribute. Visualizing the feature importance helps identify the most influential features in the classification task, aiding in feature selection or engineering.

6.7 Testing and Validation Report

This section of the code focuses on testing and validating the trained classification model using the test dataset. The evaluation includes generating a confusion matrix[11], calculating accuracy, recall, and precision scores, and storing the results in a DataFrame.

1. Confusion Matrix Visualization: The `make_confusion_matrix` function is used to create a heatmap visualization of the confusion matrix. The confusion matrix provides a comprehensive overview of the model's performance by showing the true positive, true negative, false positive, and false negative predictions. Visualizing the confusion matrix helps in understanding the distribution of predicted and actual labels and identifying any imbalances or misclassifications.

2. Evaluation Metrics Calculation: The `validate_model` function calculates various evaluation metrics based on the predicted and actual labels. The metrics calculated include accuracy, recall, and precision. Accuracy measures the overall correctness of the model's predictions. Recall (sensitivity) calculates the proportion of actual positive cases correctly classified as positive. Precision calculates the proportion of positive predictions that are actually positive. By calculating these metrics[12], we gain insights into the model's performance in correctly identifying exoplanet candidates and false positives.

3. Results Storage: The results of the evaluation metrics (accuracy, recall, and precision) are stored in a DataFrame called `results_table`.

6.8 Pipeline

Following the explanation of the above steps, the diagram in figure 3 gives a thorough overview of the pipeline from the Pre-processing to the Model Running with the final parameter selection.

7 Results

7.1 Training Cross-Validation

Based on the training cross-validation results, here is a summary of the performance of the classification models:

Based on these metrics, we can ascertain the best performing model.

- **Random Forest:** This model shows high precision, recall, and F1-score for both classes. It achieves an overall accuracy of 96%, indicating its effectiveness in distinguishing between "CANDIDATE" and "FALSE POSITIVE" instances.

[11] confusion matrix
[12] accuracy, recall, and precision

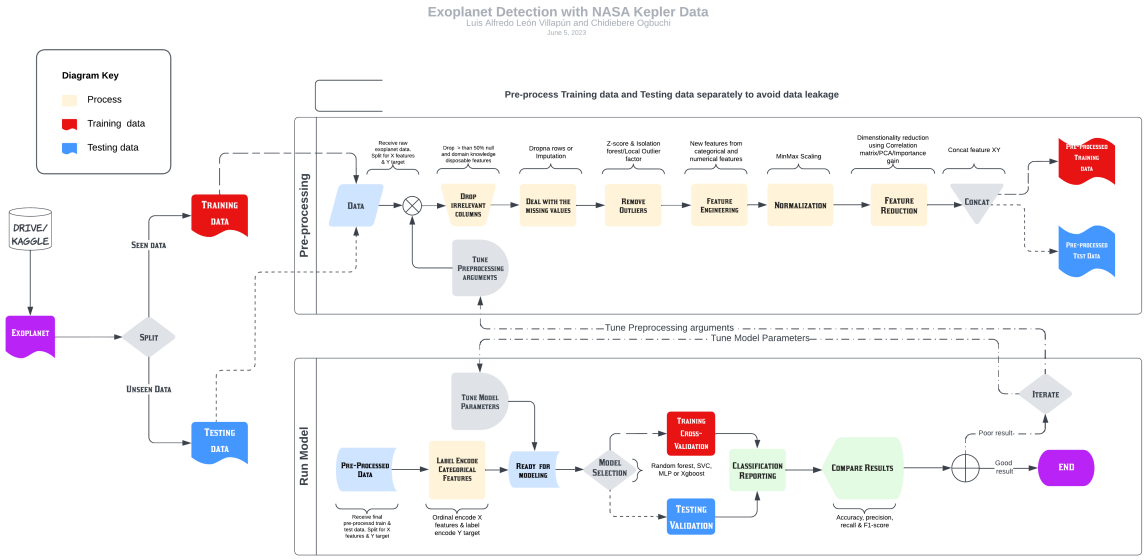


Figure 3: Pipeline of Machine Learning Modelling

Table 1: Classification Performance with and without *Koi_score* feature

Model	Acc	Prc (C)	Rc (C)	F1 (C)	Prc (FP)	Rc (FP)	F1 (FP)
RF	0.96	0.98	0.96	0.97	0.94	0.97	0.96
MLP	0.96	0.97	0.95	0.96	0.94	0.96	0.95
SVC	0.96	0.97	0.95	0.96	0.94	0.97	0.95
XGB	0.96	0.97	0.96	0.97	0.95	0.96	0.96
RF (No *Ks*)	0.81	0.76	0.94	0.84	0.89	0.64	0.75
MLP (No *Ks*)	0.81	0.77	0.92	0.84	0.87	0.67	0.76
SVC (No *Ks*)	0.73	0.68	0.96	0.79	0.89	0.45	0.59
XGB (No *Ks*)	0.80	0.76	0.92	0.83	0.86	0.65	0.74

- **XGBoost:** Similar to Random Forest, XGBoost demonstrates consistently high precision, recall, and F1-score for both classes. It achieves an accuracy of 96%, making it a strong performer.
- **ML Perceptron:** This model also performs well, with high precision, recall, and F1-score for both classes. It has an accuracy of 96%, matching the top models.
- **Support Vector Classifier (SVC):** The SVC model exhibits high precision, recall, and F1-score for both classes, achieving an accuracy of 96%.
- **ML Perceptron (No KOI Score):** This model performs slightly worse than the previous ones but still maintains reasonably high precision, recall, and F1-score for both classes. It has an accuracy of 81%, which is lower than the top models.
- **Random Forest (No KOI Score):** This model shows a drop in performance compared to the others, with lower precision, recall, and F1-score for both classes. It achieves an accuracy of 81%.

- **XGBoost (No KOI Score):** Similar to the previous models without KOI scores, this model exhibits lower precision, recall, and F1-score for both classes. It has an accuracy of 80%.
- **SVC (No KOI Score):** This model performs the poorest among the evaluated models. It has lower precision, recall, and F1-score for both classes, with an accuracy of 73%.

Overall, the Random Forest, XGBoost, and ML Perceptron models perform the best, consistently achieving high precision, recall, F1-score, and accuracy. The models without KOI scores show a decline in performance, indicating that the KOI score feature is important for accurate classification. The SVC (No KOI Score) model performs the worst, with significantly lower precision, recall, and F1-score.

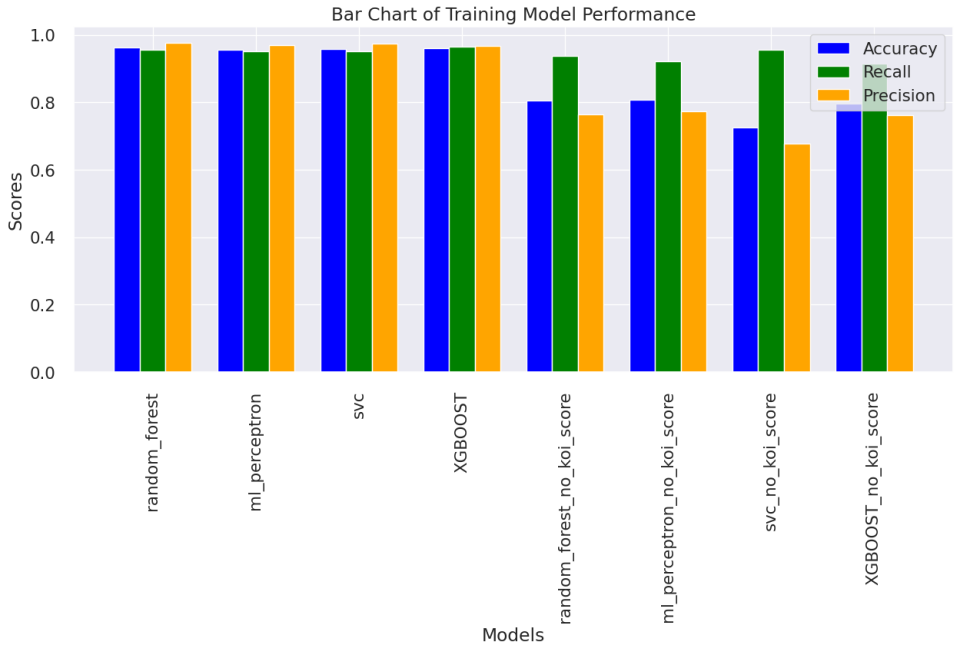


Figure 4: Bar Chart showing Training Validation Results

7.2 Testing Validation

The test results from the application of various machine learning models to detect exoplanets are summarised in the table provided. The models' performance was evaluated primarily based on their recall, as the research question focused on maximizing the detection of candidate exoplanets over minimizing false positives. However, accuracy and precision metrics were also considered to provide a comprehensive view of the models' performances.

Models utilizing the koi_score feature were compared with their counterparts that did not use the koi_score. Koi_score is a precomputed value that could potentially introduce bias, but the incorporation of this feature might also enhance model performance.

Among the models utilizing koi_score, the XGBOOST model yielded the best performance with an overall accuracy of 96.13%. In terms of recall, it achieved an excellent result, with recall rates of 96.50% and 95.64% for detecting candidate exoplanets and false positives, respectively. This high recall rate means the XGBOOST model was extremely proficient in identifying true positive results. Its precision was also high at 96.62% and 95.48%, indicating that the model was also

Table 2: Results Test Table

	Model	Accuracy	Recall	Precision
0	random_forest	0.962	[0.956 0.970]	[0.976 0.945]
1	random_forest_no_koi_score	0.806	[0.937 0.645]	[0.764 0.894]
2	ml_perceptron	0.955	[0.951 0.961]	[0.970 0.938]
3	ml_perceptron_no_koi_score	0.807	[0.921 0.668]	[0.772 0.874]
4	svc	0.958	[0.952 0.967]	[0.973 0.940]
5	svc_no_koi_score	0.727	[0.956 0.446]	[0.679 0.892]
6	XGBOOST	0.961	[0.965 0.956]	[0.966 0.955]
7	XGBOOST_no_koi_score	0.797	[0.915 0.652]	[0.763 0.863]

effective in minimizing false positive results.

The random forest and svc models also yielded good results, both achieving overall accuracy scores of 96.20% and 95.83%, respectively. Their recall and precision scores were also high, indicating strong performance in both correctly identifying true positives and minimizing false positives.

Contrastingly, among the models that did not use koi_score, performance understandably decreased. The random forest model without koi_score still performed best among these with an accuracy of 80.59%. Its recall rates were 93.74% for detecting candidate exoplanets and 64.49% for false positives. The high recall rate for candidate exoplanets is promising, but the low recall rate for false positives means that this model might be missing a significant number of false alarms.

Other models not utilizing koi_score, such as ml_perceptron and svc, achieved similar performances. However, the svc_no_koi_score model notably had the lowest overall accuracy at 72.67% and the lowest recall rate for false positives at 44.60%, suggesting it struggled more with the detection of false positives.

While the incorporation of the koi_score may introduce bias, models using this feature achieved superior results in our study. The XGBOOST model utilizing the koi_score was the best performer overall, suggesting its high suitability for detecting candidate exoplanets with minimal false positives. However, the high recall rate of candidate exoplanets in the random forest model without the koi_score feature highlights its potential value when bias reduction is a key consideration. We suspect that the random forest is robust to overfitting, sensitive to feature interaction when creating decision trees and better at handling noise than the rest of the models. Thus we propose using Random forest or Xgboost above the others for future works.

8 Discussion

Our investigation culminates with several crucial implications and points for further deliberation brought forth by our findings.

Firstly, the developed models show potential for application as third-party screening mechanisms to discern between candidate and non-candidate observations. Notably, under these conditions, the use of the koi_score metric is permissible since the model would only be employed post computation of the koi_score. On the other hand, should the model be intended for use as an additional predictive tool, it becomes essential to negate any inherent dataset bias. This would

involve the removal of not only the koi_score but also flag columns.

The aspect of model reusability with similar datasets, albeit with differing measurements, forms another pertinent area for discussion. The specificity of the model to Kepler's data raises questions about its applicability to observations from other missions, which will invariably employ different instruments and data measurement techniques. This consideration should be central to future investigations seeking to adapt the model for datasets beyond Kepler's purview.

In concluding, we underscore the benefit of utilizing machine learning models to filter such data. The inherent generalization power of ensemble methods like Random Forests and XGBoost proves especially valuable in these circumstances, providing a consensus-based approach to highlight new and intriguing observations. Such applications of machine learning could dramatically enhance our ability to detect and study exoplanets, thereby expanding our understanding of the universe.

References

[Bre01] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[Bre+00] Markus M. Breunig et al. "LOF: Identifying Density-Based Local Outliers". In: *SIGMOD Rec.* 29.2 (May 2000), pp. 93–104. ISSN: 0163-5808. DOI: 10.1145/335191.335388. URL: https://doi.org/10.1145/335191.335388.

[CS08] Juan Cabrera and Jean Schneider. "Support vector machines for binary classification of exoplanet candidates". In: *Astronomy & Astrophysics* 491.3 (2008), pp. 855–861.

[CLB04] Chao Chen, Andy Liaw, and Leo Breiman. "Using random forest to learn imbalanced data". In: *University of California, Berkeley* 110.1-2 (2004), pp. 24–31.

[CG16] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2016, pp. 785–794.

[CV95] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297.

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural networks* 2.5 (1989), pp. 359–366.

[Jen17] Jon M Jenkins. *KEPLER DATA PROCESSING HANDBOOK*. 2017. URL: https://www.hubblesite.org/files/live/sites/mast/files/home/missions-and-data/kepler/_documents/KSCI-19081-002-KDPH.pdf.

[JYC22] Yucheng Jin, Lanyi Yang, and Chia-En Chiang. "Identifying Exoplanets with Machine Learning Methods: A Preliminary Study". In: *International Journal on Cybernetics and Informatics* 11.2 (Apr. 2022), pp. 31–42. DOI: 10.5121/ijci.2022.110203. URL: https://doi.org/10.5121%2Fijci.2022.110203.

[RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (1986), pp. 533–536.

[Vis+22] Varad Vishwarupe et al. "Comparative Analysis of Machine Learning Algorithms for Analyzing NASA Kepler Mission Data". In: *Procedia Computer Science* 204 (2022). International Conference on Industry Sciences and Computer Science Innovation, pp. 945–951. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2022.08.115. URL: https://www.sciencedirect.com/science/article/pii/S1877050922008535.

Appendix A

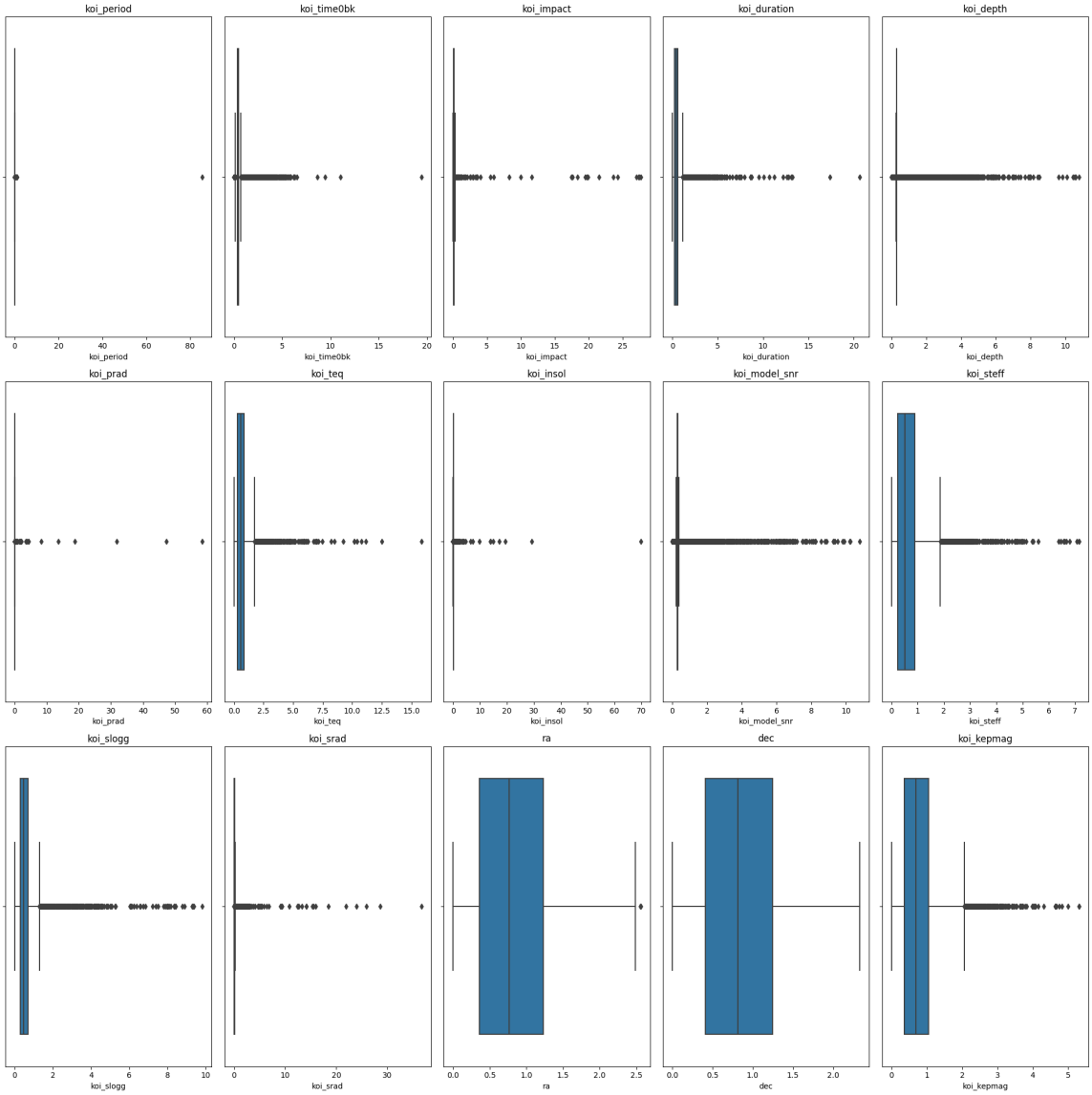


Figure 5: Bar Chart showing Training Validation Results

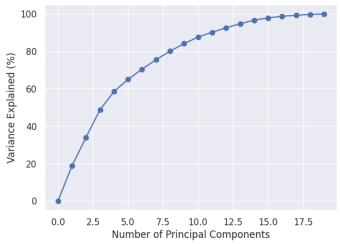
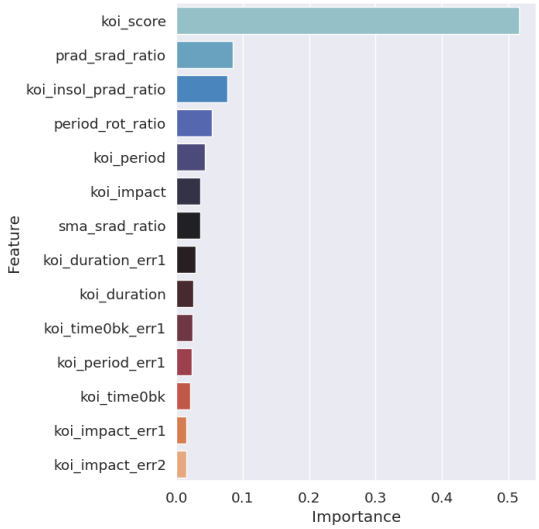
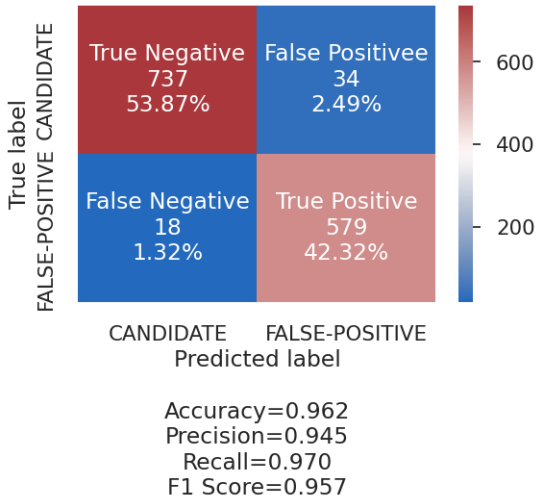


Figure 6: PCA

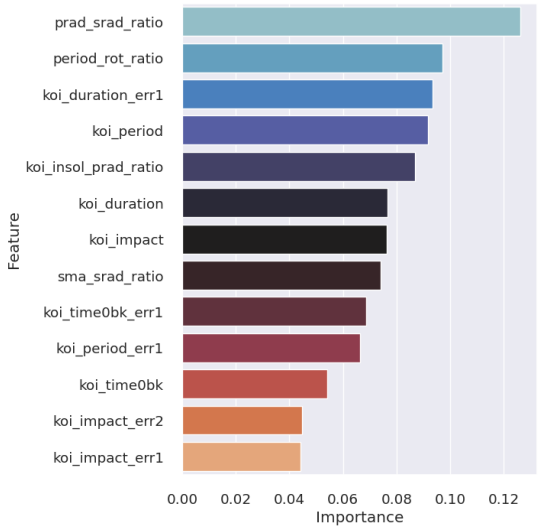


(a) Training Validation Mean Gain

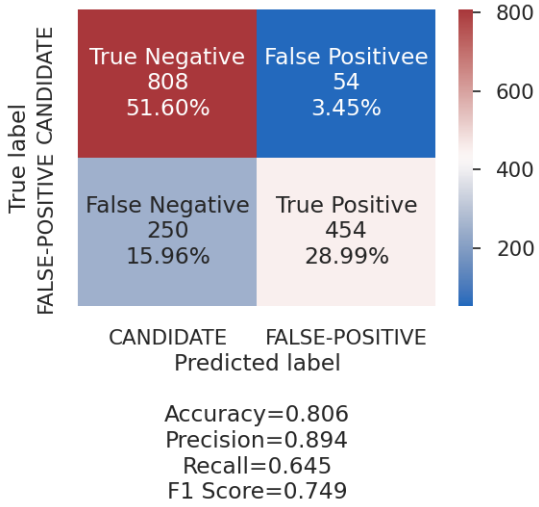


(b) Test Validation

Figure 7: Random Forest

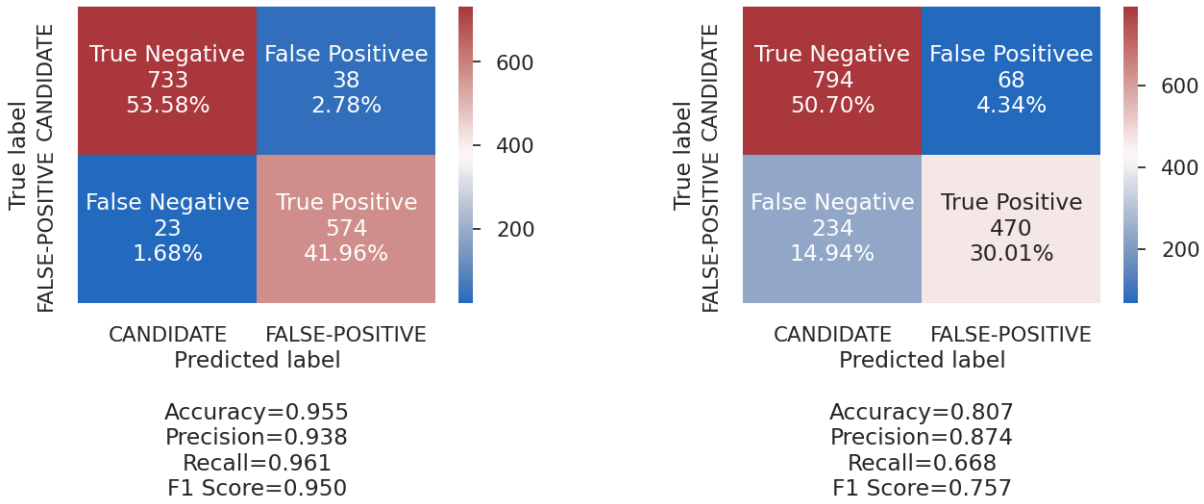


(a) Training Validation Mean $\text{Gain}_{NoKoiscore}$



(b) Test Validation NoKoiscore

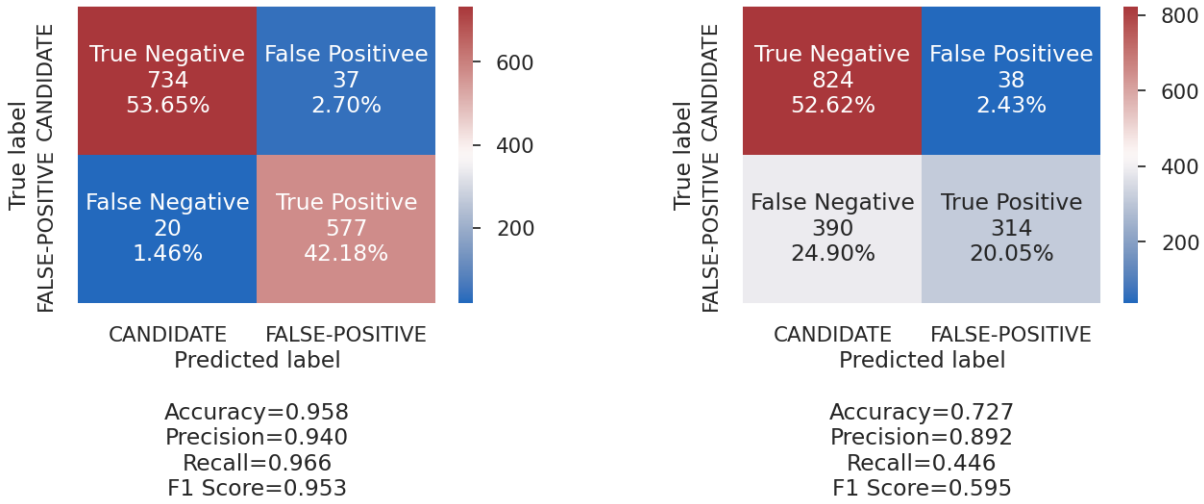
Figure 8: Random Forest Nokoiscore



(a) Test Validation Results

(b) Test Validation Nokoiscore

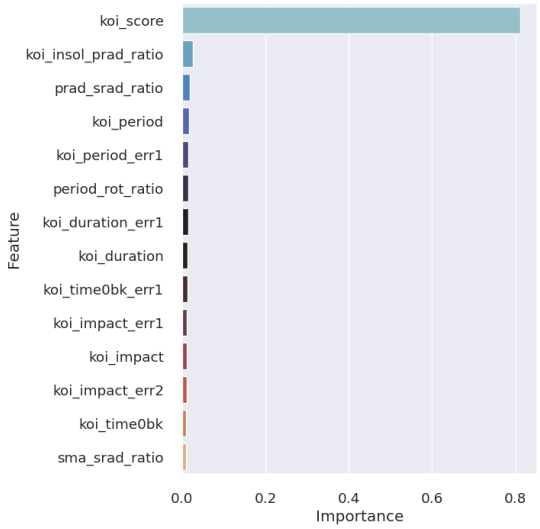
Figure 9: Multi-layer Perceptron



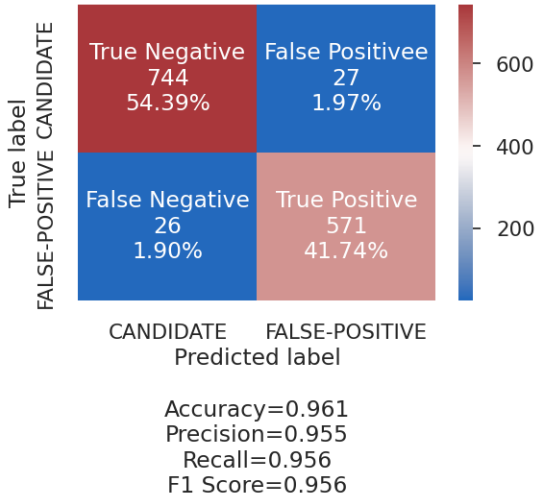
(a) Test Validation Results

(b) Test Validation Nokoiscore

Figure 10: Support Vector Classifier

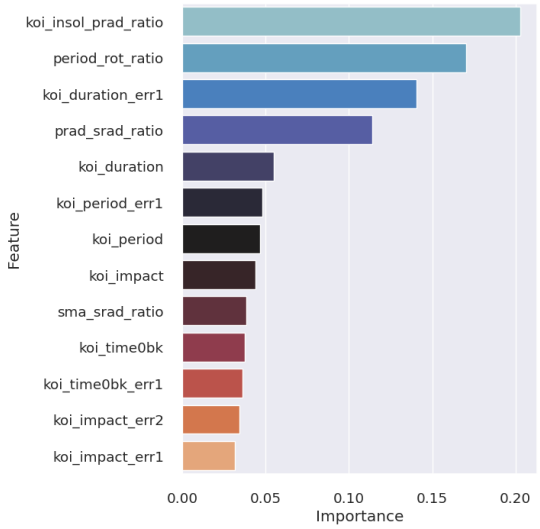


(a) Training Validation Mean Gain

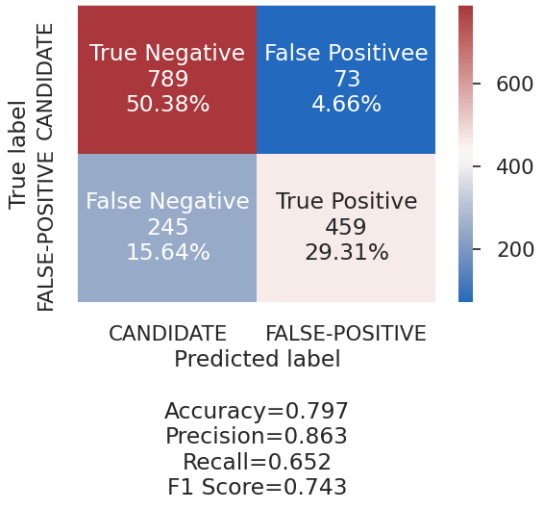


(b) Test Validation Results

Figure 11: XGBOOST



(a) Training Validation Mean Gain NoKoiscore



(b) Test Validation Results NoKoiscore

Figure 12: XGBOOST