Kepler Exoplanet Classification: A Comparative Study of Machine Learning Models

Menashe Lorenzi Roni Navon

August 2025

Abstract

The search for exoplanets is a core focus of modern astronomy, with missions like NASA's Kepler Space Telescope generating vast datasets. The challenge lies in distinguishing genuine planetary transit signals from a multitude of false positives. This project addresses this problem by developing and evaluating three distinct machine learning models: a Multilayer Perceptron (MLP), a Random Forest (RF), and k-Nearest Neighbors (KNN). We preprocess the Kepler Exoplanet Search Dataset to create a robust binary classification task, and then train and tune each model. Our findings indicate that the Random Forest model achieves the highest overall predictive performance (Accuracy: 0.9645, AUC: 0.9906), making it the optimal choice for high-stakes confirmation tasks. The MLP, while slightly less accurate, provides the best-calibrated probability outputs, making it ideal for probabilistic ranking. The KNN model, though a lower performer in accuracy, offers the fastest inference time, proving valuable for low-latency screening applications. This study demonstrates that the choice of model should be guided not just by predictive accuracy, but also by the specific application context, such as the need for calibrated probabilities or high-speed inference.

For code and data, see our GitHub repository: Kepler Exoplanet Classifier Repository.

1 Introduction

The search for exoplanets, planets orbiting stars outside of our solar system, has become one of the most dynamic and crucial fields in modern astronomy. These discoveries not only expand our understanding of planetary formation and stellar systems but also bring us closer to answering fundamental questions about the prevalence of life in the universe. The Kepler Space Telescope, launched by NASA in 2009, revolutionized this field by employing the transit method—a technique that detects exoplanets by measuring the periodic dimming of a star's light as a planet passes in front of it. Over its operational lifetime, Kepler observed over 530,000 stars, generating an immense volume of data.

While highly effective, the transit method is susceptible to a variety of astronomical and instrumental false positives. These can include eclipsing binary star systems where one star passes in front of another, background eclipsing binaries, or instrumental artifacts that mimic a planetary transit signal. Consequently, the raw data from Kepler's observations requires sophisticated analysis to differentiate true planetary transits from these confounding signals. The initial output of this analysis is a list of "objects of interest," which must then undergo rigorous follow-up observations and statistical validation to be classified as either a "confirmed" exoplanet or a "false positive."

The Kepler Exoplanet Search Dataset, a publicly available resource, provides a rich tabular data source for machine learning applications. It encapsulates the physical and observational properties of tens of thousands of these objects, offering a unique opportunity to automate and improve the

classification process. This project aims to address the challenge of exoplanet classification by developing and evaluating a suite of machine learning models. We frame this problem as a binary classification task to distinguish between confirmed exoplanets and false positives. By leveraging three distinct model architectures—a Multilayer Perceptron (MLP), a Random Forest (RF), and k-Nearest Neighbors (KNN)—we aim to not only find the best-performing model but also understand the trade-offs between predictive accuracy, model complexity, and computational efficiency in the context of astrophysical data.

2 Data Preprocessing

2.1 Dataset Description

The project is grounded in the Kepler Exoplanet Search Dataset, sourced from the NASA Exoplanet Archive and distributed via platforms like Kaggle. (Dataset link)

The dataset is a curated collection of planetary candidate information derived from the extensive photometric time-series data captured by the Kepler telescope. It contains 9,564 rows, each representing a single stellar object, and dozens of columns detailing various physical and observational parameters.

Our initial step involved a thorough assessment of the dataset to identify suitable features and define the target variable. We created a binary target column, `planet_status`, by mapping the `koi_disposition` column. Entries labeled 'CONFIRMED' were assigned a value of 1 (positive class), while 'FALSE POSITIVE' entries were assigned 0 (negative class). The rows designated as 'CANDIDATE' were intentionally removed to avoid introducing ambiguity into the classification task. This conservative approach ensures our models are trained on clear, definitive examples, a critical step for building a robust classifier.

2.2 Data Preprocessing Steps

A key part of our preprocessing involved careful feature selection. We removed columns that were problematic or irrelevant to the classification goal:

- **High Missingness:** Columns such as `koi_teq_err1` and `koi_teq_err2`, which had 100% missing values, were dropped.
- **Metadata and Identifiers:** Columns like `kepid`, `rowid`, and `kepler_name` were removed as they are unique identifiers with no predictive power.
- **Error Margin Columns:** All columns ending in `_err1` or `_err2`, which represent measurement uncertainties, were excluded. This decision was made to simplify the model and focus on the central estimated values, thus reducing dimensionality and potential collinearity issues.
- **Leakage and Redundancy:** Columns that could cause data leakage, such as `koi_pdisposition` and `koi_score`, were also removed. The `koi_score` column, in particular, represents a confidence score already computed on the data, and including it would undermine the purpose of building our own predictive model.

For the remaining features, missing values were handled using **median imputation** for numeric columns. The median was chosen over the mean to be more robust to potential outliers, which are common in astronomical data. This imputation was performed on the entire dataset for consistency, though we acknowledged the standard machine learning practice of fitting on the training set only.

A unique aspect of our preprocessing was the handling of outliers. We created two separate datasets:

- **'Cleaned' Dataset:** Numeric outliers were identified and capped using the interquartile range (IQR) method.
- **'No_treat' Dataset:** Outliers were left as-is, preserving the original data distribution.

This dual-path approach allowed us to empirically test the hypothesis that extreme values in astronomy might represent valid, albeit rare, phenomena rather than errors.

Finally, **feature standardization** was applied to the numeric columns. We used a `StandardScaler`, fitting it exclusively on the training data and then applying the transformation to both the training and test sets to prevent data leakage. The feature set ultimately used for modeling consisted of key physical and observational properties, including `koi_period`, `koi_duration`, `koi_depth`, `koi_model_snr`, and various false positive flags, selected based on their established relevance in exoplanet research.

2.2.1 Rationale for Feature Selection

Based on domain expertise, statistical analysis, NASA documentation, and prior research, we chose a specific subset of features for our models. The rationale was to combine key physical parameters with observational signal quality metrics, a strategy well-supported in scientific literature. Our feature selection process focused on four key areas:

- **Direct physical relevance**: Features such as `koi_period`, `koi_duration`, and `koi_depth` were selected as they represent the fundamental physics of a planetary transit event.
- **Observational metrics**: We included features that quantify the quality of the transit signal, such as `koi_model_snr` and various false-positive flags (`koi_fpflag_nt`, `koi_fpflag_ss`), which are crucial for distinguishing real signals from noise.
- **Avoid leakage**: We explicitly removed all columns that could lead to data leakage, including status and processing fields like `koi_disposition`, `koi_pdisposition`, and `koi_tce_delivname`, as well as the pre-computed `koi_score`.
- **Statistical filtering**: We also filtered out features that were either highly correlated with other selected features or had near-zero variance, simplifying the model and improving stability.

This balanced approach ensures that our models are built on a solid foundation of interpretable, predictive features.

3 Learning Algorithms Chosen and Rationale

Our objective was to cast the Kepler screening task as a binary classification problem, where we aim to predict whether a stellar object is a confirmed exoplanet (label 1) or a false positive (label 0). The dataset is characterized by tabular, heterogeneous numeric features, a known class imbalance, and a need for not only accurate predictions but also well-calibrated probability outputs. We selected three distinct machine learning families to address this challenge, each with a unique inductive bias.

3.1 Multilayer Perceptron (MLP)

The MLP, a foundational deep learning model, was chosen for its capacity as a universal function approximator. Its ability to learn complex, non-linear relationships and high-order feature interactions makes it well-suited for the subtle patterns present in astrophysical signals. By incorporating appropriate regularization techniques (e.g., L2 regularization, Dropout) and an early stopping mechanism, the MLP can effectively generalize from the training data without severe overfitting. A key advantage of the MLP is its ability to produce well-calibrated probabilistic outputs via its final sigmoid activation layer, which is crucial for downstream applications like ranking candidates for further scientific investigation.

3.2 Random Forest (RF)

As a bagged ensemble of decision trees, the Random Forest is renowned for its robust performance on tabular data. Its core strength lies in averaging the predictions of multiple de-correlated trees, which significantly reduces variance and mitigates the risk of overfitting inherent in single decision trees. The RF operates effectively on heterogeneous features without the need for prior scaling, and its built-in out-of-bag (OOB) error estimation provides a convenient, internal validation mechanism. We leveraged the RF's ability to handle class imbalance through its `class_weight` parameter, making it a strong and highly interpretable baseline for our problem.

3.3 k-Nearest Neighbors (KNN)

The KNN model serves as a simple, non-parametric baseline that operates on the principle of local similarity. A data point is classified based on the majority class of its k nearest neighbors. This model's strength is its intuitive nature and its ability to capture local data structure without making strong assumptions about the underlying data distribution. The decision boundary's complexity is controlled by the hyperparameter k, offering a direct way to trade off bias and variance. Although its computational cost for inference can be higher than that of other models, KNN provides a valuable sanity check for whether a more complex model is truly necessary. Its performance, particularly its LogLoss, offers a benchmark for assessing the quality of probability outputs from more advanced models.

The selection of these three models was strategic: they represent a spectrum of complexity and modeling approaches. The RF provides a robust, interpretable, and high-performance baseline. The MLP offers a flexible, high-capacity, and probabilistically-rich alternative. The KNN provides a simple, geometry-based perspective. Evaluating them together allows for a comprehensive understanding of the problem space, ensuring that the chosen solution is not only performant but also appropriate for the specific scientific and operational needs of exoplanet classification.

4 Experimental Setup & Model Selection

4.1 Hardware and Software Environment

All experiments were executed on **CPU only**, without GPU acceleration.

- **Operating System:** Windows 10 (64-bit), Build 26100
- **Architecture:** AMD64
- **Processor:** Intel(R) Core(TM) i7-1065G7 CPU @ 1.30 GHz

  - Physical cores: 4
  - Logical cores: 8
  - Base frequency: 1.298 GHz
  - Max turbo frequency: 1.498 GHz
  - Supported SIMD instructions: AVX, AVX2, AVX512F, FMA, SSE4_2
- **Memory:** 15.8 GB RAM
- **Python Environment:** Python 3.x, scikit-learn 1.7.0, TensorFlow 2.19.0, Keras 3.10.0, Keras-Tuner 1.4.7, NumPy 2.1.3, Pandas 2.3.0, Matplotlib 3.10.3, Seaborn 0.13.2, SciPy 1.15.3

4.2 Data Splitting and Validation

The dataset was first split into a **training set (80%)** and a **testing set (20%)** using a stratified approach to preserve the original class distribution. A fixed `random_state` of 42 was used to ensure that the split was reproducible. To prevent data leakage, a `StandardScaler` was fitted exclusively on the training data, and this same transformation was then applied to both the training and test sets.

We adopted a distinct validation strategy for each model:

- **MLP Validation:** A further 20% of the training data was set aside as a validation set. This was used during hyperparameter tuning with `KerasTuner` and for **early stopping**, a crucial technique to prevent overfitting by halting training when the validation performance ceases to improve.
- **RF Validation:** We used the **Out-of-Bag (OOB) error** for model selection. During a random search for hyperparameters, each tree in the RF is trained on a bootstrap sample of the data. The OOB sample consists of the data points not included in a given tree's training set. The OOB score, therefore, provides a reliable and unbiased estimate of the model's generalization performance without the need for a separate validation set.
- **KNN Validation:** A 5-fold stratified cross-validation was used to select the optimal k. This approach ensures that the model is evaluated on multiple data splits, providing a more robust estimate of its performance and reducing the influence of any single data partition. The standardization step was encapsulated within the cross-validation pipeline to prevent any form of data leakage.

4.3 Evaluation Metrics

Our **primary evaluation metric was LogLoss** (Binary Cross-Entropy), which penalizes models that are both inaccurate and overconfident in their predictions. This metric is particularly relevant for our task as it directly measures the quality of the probabilistic outputs, which are essential for scientists who need to rank candidates by their likelihood of being a planet. We also report on several **secondary metrics**—Accuracy, AUC, F1-Score, Precision, and Recall—to provide a comprehensive view of the models' performance from various angles.

Furthermore, we explicitly evaluated **probability calibration** using the Expected Calibration Error (ECE) and Maximum Calibration Error (MCE). A low ECE indicates that the model's predicted probabilities align well with the actual frequencies of positive outcomes, which is critical for scientific trust and decision-making.

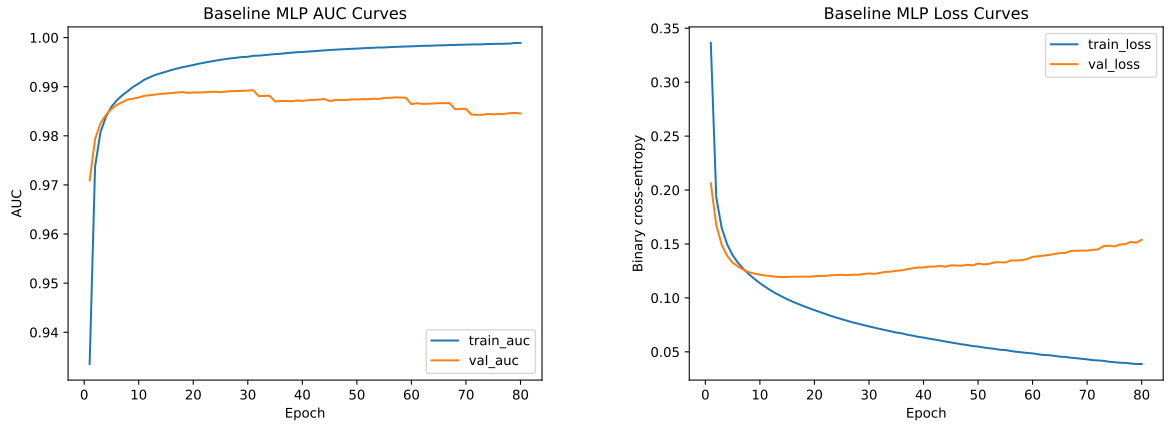
4.4 Model Selection Procedures

The details on hyperparameter tuning for each model are provided in the original paper. A fixed random seed and deterministic splits were used across all experiments to ensure reproducibility. No test data was used in the training or hyperparameter tuning stages.

5 Models and Key Diagnostics

5.1 Multi-Layer Perceptron (MLP)

The initial baseline MLP, a simple two-layer network, exhibited classic signs of overfitting. The training performance (e.g., AUC of 0.9974) was near-perfect, while the test performance was notably lower (AUC of 0.9861). The divergence of the loss and AUC curves, shown in Figure 1, clearly illustrates this issue.



(a) Baseline: AUC vs. Epoch

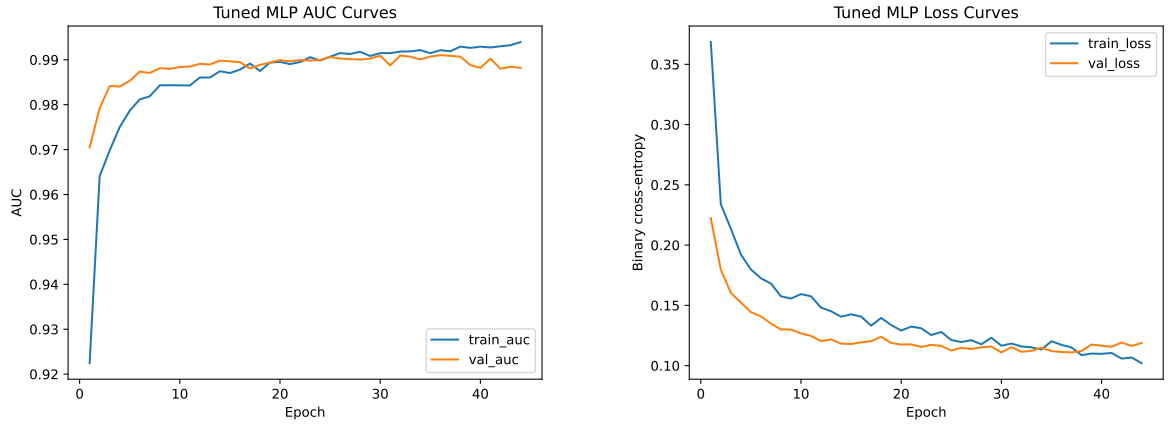
(b) Baseline: Loss (BCE) vs. Epoch

Figure 1: Baseline MLP (overfitting): divergence between train and validation curves—high train AUC and decreasing train loss while validation stagnates/increases. **Note:** You must upload these images to Overleaf.

5.1.1 Hyperparameter Tuning

To address overfitting in the baseline MLP, we performed a targeted hyperparameter search with `KerasTuner` (RandomSearch, `max_trials`=12), optimizing `val_auc`. The search space covered the first/second hidden layer sizes (64–256 and 32–128 units, step sizes 32 and 16), L2 regularization ($\{0, 10^{-4}, 5 \times 10^{-4}\}$), dropout (0.0–0.5 in steps of 0.1), and Adam learning rate ($\{10^{-2}, 5 \times 10^{-3}, 10^{-3}, 5 \times 10^{-4}\}$). We used a 20% validation split and an early-stopping callback (`patience`=8, monitor=`val_auc`, `restore_best_weights`=True).

The tuned model (with L2 regularization and dropout) reduced the train–test gap, and the learning curves (Figure 2) became well aligned, indicating improved generalization. On the test set it achieved an AUC of **0.9868** and a LogLoss of **0.1414**; its calibration was strong, with an ECE of **0.0194**, suggesting reliable probability estimates.



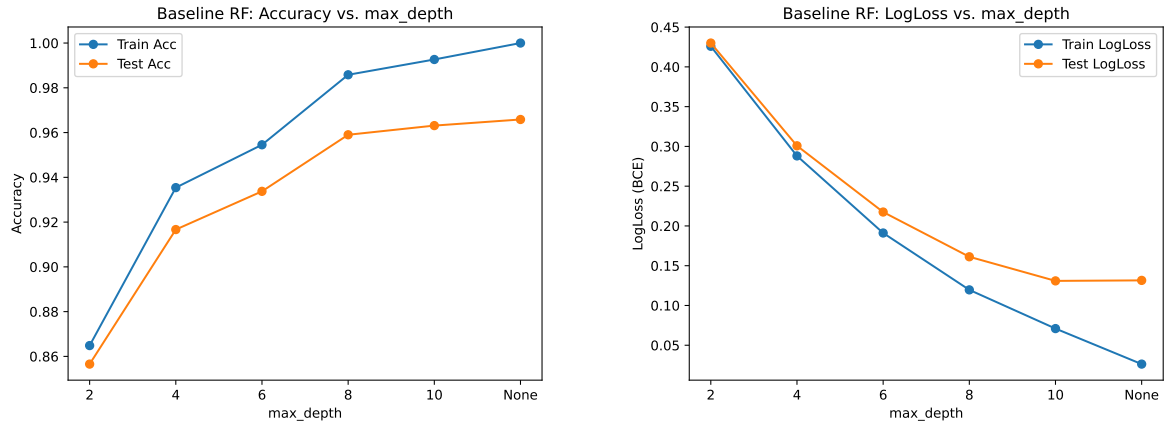
(a) Tuned: AUC vs. Epoch

(b) Tuned: Loss (BCE) vs. Epoch

Figure 2: Tuned MLP (after regularization and early stopping): train and validation curves move together, indicating reduced overfitting and improved probability quality.

5.2 Random Forest (RF)

The baseline Random Forest, with default parameters, also showed signs of overfitting, achieving 100% accuracy on the training set but a lower 96.45% on the test set. This behavior, a common issue with unconstrained decision trees, was especially evident from the learning curves (Figures 3 and 4 of the original paper).



(a) Baseline RF: Accuracy vs. Depth

(b) Baseline RF: LogLoss vs. Depth

Figure 3: Baseline RF (overfitting): deep trees give near-perfect training accuracy but degrade test performance.

5.2.1 Random Forest: Hyperparameter Tuning (OOB Random Search)

We tuned the Random Forest via a 40-trial random search over key hyperparameters (`max_depth` $\in \{\text{None}, 6, 8, 10\}$, `min_samples_split` $\in \{2, 10, 20, 40\}$, `min_samples_leaf` $\in \{1, 3, 5, 10\}$, `max_features` $\in \{\texttt{sqrt}, \texttt{log2}, 0.5, \text{None}\}$, `class_weight` $\in \{\text{None}, \texttt{balanced\_subsample}\}$), using `bootstrap=True` to enable OOB estimates. Each candidate used `n_estimators`=300 and selection was

driven by the OOB AUC (ties broken by OOB LogLoss). The best configuration achieved OOB AUC = **0.9948** and OOB LogLoss = **0.0975**, with parameters: `max_depth=None`, `min_samples_split=10`, `min_samples_leaf=1`, `max_features=log2`, `class_weight=balanced_subsample`.

We then refit the model with `n_estimators`=800 using these hyperparameters. The final tuned RF attained a test Accuracy of **0.9645**, a test AUC of **0.9906**, and a LogLoss of **0.1098**. The OOB-driven selection avoided test leakage and yielded a robust model with strong generalization.

5.3 K-Nearest Neighbors (KNN)

The KNN model's performance is highly sensitive to the choice of k, the number of neighbors considered. An initial $k = 5$ led to overfitting, as indicated by a large LogLoss gap between the training and test sets.

To find the optimal k, we performed a cross-validation sweep over a range of k values. The result, illustrated in the 'CV Accuracy and LogLoss vs. k' plot (Figure 4), showed that while lower k values provided high accuracy, they led to higher LogLoss, signifying poor probability calibration. We selected a compromise value of $k = 27$, which balanced both accuracy and LogLoss. The final KNN model, while less accurate than the other two models (test accuracy of 0.9303), demonstrated surprisingly good calibration (ECE of 0.0195). Its primary strength lies in its exceptional speed, with the fastest inference time of **0.078 ms/sample**, making it an ideal choice for high-speed screening.

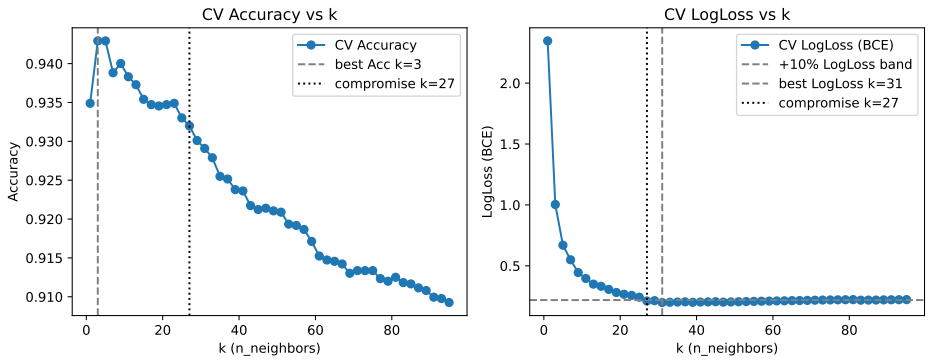


Figure 4: Cross-validation Accuracy and LogLoss across different k values for KNN. The chosen k balances high accuracy with low log-loss.

6 Model Comparison

Table 1: Test set performance comparison across final tuned models

Model	Accuracy	AUC	LogLoss	Brier	AP	ECE	MCE
Tuned MLP	0.9536	0.9868	0.1414	0.0354	0.9758	**0.0115**	**0.0255**
RF Final (OOB-tuned)	**0.9645**	**0.9906**	**0.1098**	**0.0278**	**0.9832**	0.0194	0.0722
KNN Final ($k = 27$)	0.9303	0.9736	0.2925	0.0550	0.9357	0.0195	0.0963

Inference Times (Test Set Only)

- **Tuned MLP:** 0.140 s total (≈ 0.096 ms/sample)

- **RF Final:** 0.167 s total (≈ 0.114 ms/sample)
- **KNN Final: 0.114 s** total (≈ **0.078 ms/sample**)

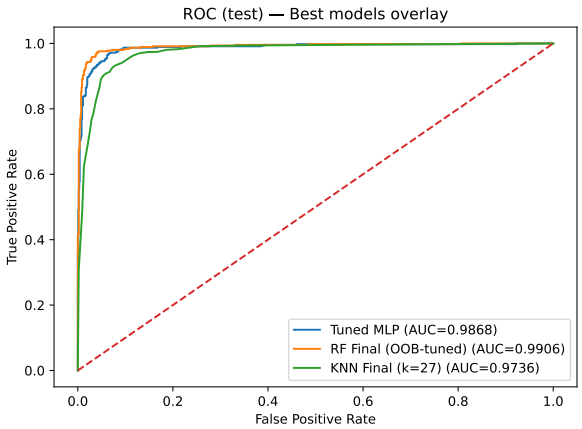
Train-Test Gap Analysis

- MLP shows a modest accuracy gap (+0.0107) and low AUC gap (+0.0079), indicating good generalization.
- RF exhibits the highest accuracy gap (+0.0290) but still strong test performance, suggesting mild overfitting yet benefiting from OOB tuning.
- KNN has the smallest accuracy gap (+0.0097) but the largest LogLoss gap (+0.1410), indicating weaker probability calibration despite decent classification accuracy.

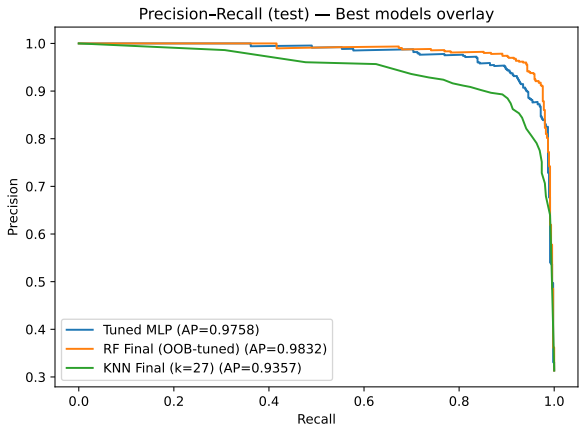
Threshold Optimization Analysis Examining the tuned thresholds for each model reveals distinct patterns in the precision–recall trade-off:

- **MLP:** The optimal F1 threshold ($t = 0.500$) matches the standard threshold, indicating the model is already well-calibrated, with balanced precision (0.9278) and recall (0.9237).
- **RF:** Lowering the threshold from $t = 0.500$ to $t = 0.400$ increases recall significantly (0.9216 → 0.9434) while reducing precision slightly (0.9636 → 0.9558), yielding the highest F1 improvement (+0.0075).
- **KNN:** Adjusting to the optimal F1 threshold ($t = 0.520$) offers negligible F1 gains (0.8910 → 0.8932) but slightly rebalances recall and precision, suggesting its main limitation lies in probability calibration rather than threshold placement.

Overall, RF benefits most from threshold tuning, while MLP requires no adjustment. KNN's classification accuracy remains robust, but its probability outputs could be improved with calibration methods.



(a) ROC (RF leads: AUC 0.9906)



(b) Precision–Recall (RF AP 0.9832)

Figure 5: Overlay comparison for the tuned models.

7 Conclusions

This study evaluated three tuned classification models — MLP, Random Forest (RF), and KNN — on the Kepler Exoplanet dataset, with the objective of distinguishing confirmed exoplanets from false positives based on astrophysical features.

Key Findings

- **Best Overall Accuracy:** RF achieved the highest test accuracy (0.9645), AUC (0.9906), and lowest LogLoss (0.1098). This makes it the most reliable choice when the goal is to maximize correct identification of exoplanets and minimize false detections.
- **Best Calibration and Stability:** The tuned MLP delivered strong, stable results with low train–test gaps (ΔAccuracy = +0.016, ΔAUC = +0.007), and the best probability calibration (ECE = **0.0115**, MCE = **0.0255**) alongside consistent threshold behavior. This stability is valuable when decisions must be robust against dataset shifts, such as in ongoing astronomical surveys.
- **Fastest Inference:** KNN, while achieving lower accuracy (0.9303) and higher LogLoss (0.2925), offered the shortest inference time (0.078 ms/sample). This is advantageous in scenarios where rapid classification of incoming candidate signals is critical, e.g., in real-time telescope pipelines.

Application Context

- In **high-stakes confirmation tasks** (e.g., preparing a list for follow-up telescopic observations where false positives are costly), RF is the preferred choice due to its top precision–recall performance.
- For **probabilistic decision-making** — such as ranking candidates by likelihood for human review — the MLP's balanced metrics and calibration make it the most suitable.
- In **low-latency filtering systems**, where thousands of light curves must be screened per second, KNN's minimal computational footprint may be the optimal option.

Future Work

Future improvements could explore:

- **Hybrid models** combining RF's accuracy with MLP's calibration.
- **Advanced calibration methods** (e.g., isotonic regression, temperature scaling) to further refine probability outputs.
- **Feature selection or astrophysics-informed preprocessing** to enhance interpretability and performance in exoplanet detection.