Unit Testing for Random Forest Models

1st Zheyan Zhang
Industrial Engineering & Management Sciences Department
Northwestern University
Evanston, USA
zheyan.zhang@northwestern.edu

2nd Diego Klabjan
Industrial Engineering & Management Sciences Department
Northwestern University
Evanston, USA
d-klabjan@northwestern.edu

***Abstract*—We develop a specialized model-based unit testing framework designed to enhance the verification and quality assessment of Random Forest models before they are implemented and deployed. This framework is tailored to identify and resolve issues in model behavior, ensuring that the models are robust and reliable. Our approach centers on a series of tests that scrutinize the model's performance across various simulated data conditions, assessing both its accuracy and its ability to handle diverse data scenarios. Our tests proves capable of handling both a similarity assessment between two models and individual model quality control scenarios. We present extensive experiments on publicly challenging datasets, aiming to establish a thorough process that proves the model's readiness for operational use and its robustness to expected performance standards.**

***Index Terms*—Unit Test, Model Testing, Random Forest, Robustness.**

I. Introduction

In the realm of machine learning, RF models play an important role due to their robustness, versatility, and ability to handle complex datasets. These models are extensively used across various fields. Medical scientists employ them to detect and diagnose breast diseases [1], [16], stock traders analyze trading patterns and predict distress risks [28], and ecologists reveal complex interactions within ecosystems [26]. As the adoption of RF models continues, so does the imperative to ensure their expected behavior and correctness before deployment. The consequences of deploying unreliable models can be severe. For instance, a poorly tested model in a medical application might mistakenly predict patient responses to treatments, leading to direct harm to patients, wasted resources, and eroded trust in medical analytics. The potential for such serious consequences highlights the need for comprehensive testing methodologies that can prevent errors and provide guidance to decision-making processes.

Developing reliable RF models is further complicated by the collaborative nature of machine learning projects. Effective collaboration across diverse teams is critical for the success of a machine learning project, particularly when developing and deploying RF models. However, this collaboration often comes with challenges due to diffused responsibility and varying priorities among team members. Engineers focus on model stability, data scientists prioritize performance, and product managers are concerned with timelines. To accompany these divergent priorities and maximize project outcomes, implementing rigorous, unified testing protocols is needed. Our proposed model-based testing methods are designed to address these challenges by providing a standard framework that assesses model performance across different data inputs. This not only helps in aligning team objectives but also in verifying the integrity and reliability of models before their full-scale deployment, ensuring that all team perspectives are properly integrated.

To address these challenges above, we introduce a novel model testing pipeline specifically designed for RFs. This comprehensive evaluation framework is the first of its kind to assess not only model robustness and predictive ability, but also model similarity for a pair of models, i.e. paired models. Additionally, it evaluates single model performance even when clear standards for model quality are absent. Our approach ensures that both individual and paired models meet high standards of accuracy and reliability before deployment, thereby reducing risks in real-world applications.

Our methodology distinctively designs various metrics, detecting the subtle differences in model structures and behaviors that traditional methods often overlook. This all-rounded framework evaluates multiple dimensions of model behavior and characteristics, ranging from model stability under data perturbations, to a detailed analysis of feature influence using Shapley values.

The contributions of this paper are three-fold: 1. Innovative Methodology for Model Comparison: We introduce a methodology for model comparison that integrates feature importance weighting with an information similarity assessment strategy, complemented by appropriate data perturbations. This approach has been thoroughly evaluated across diverse datasets, encompassing both synthetic and real-world scenarios. 2. Systematic Design on Individual Tree Components: Our research introduces an approach to RF model evaluation. Unlike traditional methods that treat the forest as a whole, we systematically analyze the intrinsic structure of individual trees by decomposing the RF into distinct components. By examining these components separately, our method significantly enhances testing sensitivity, revealing subtle discrepancies that conventional prediction metrics often ignore. 3. A New Approach to Test Model Randomness: We develop a method to assess the level of randomness in models, specifically in situations where no standardized benchmark for a good model exists. This addition significantly enriches the toolkit available for model assessment and is crucial for real-world applications

where black-box models are common.

In summary, our work significantly advances the testing protocols for RF models by introducing a workflow that not only evaluates model performance in a multifaceted way but also provides insights into model behavior that are crucial for deploying reliable and effective machine learning solutions. This architecture serves as a fundamental check before they are deployed, saving both computational resources and time.

II. Literature Review

Random forest (RF) models, introduced in [3], are ensemble methods known for their high predictive accuracy and flexibility across various tasks. By aggregating multiple decision trees, RF models handle both classification and regression problems on numerical and categorical data and are valued for their robustness to outliers, capability to manage missing data, and effectiveness in capturing complex non-linear relationships [3], [6]. Their widespread utility is demonstrated across diverse applications: RFs outperform neural networks in solar radiation prediction [2], surpass support vector machines in diagnosing faults in rotating machinery [11], and effectively handle specialized tasks such as real-time 3D face recognition [7], COVID-19 patient outcome forecasting [20], [21], landslide risk assessment [24], and deforestation detection [25]. Given their prominence in critical, high-stakes settings, robust quality control measures are essential to verify their reliability and correctness prior to deployment.

Beyond predictive performance, RF models inherently provide interpretability through built-in feature importance metrics, notably mean decrease impurity and permutation importance. While interpretability enhances transparency and model verification, ensuring overall model reliability requires systematic unit testing. Yet, unit testing in machine learning faces unique challenges, notably the "oracle problem," which refers to the difficulty of defining exact expected outputs and non-determinism resulting from stochastic components like random initialization and training [17]. Early studies have proposed using approximate oracles by verifying invariant properties, including output shapes, value ranges, and consistency across transformations [18]. In addition, Jia et al. [13] used a mutation analysis to assess the effectiveness of unit tests in deep learning frameworks, revealing that many conventional tests fail to capture subtle defects. Despite these insights significantly advancing ML testing approaches, their applicability remains limited for traditional, non-differentiable models like RFs, signaling a notable gap that we diminish herein.

In parallel to these insights, abundant practical tools have emerged to support structured testing within ML pipelines. Software testing libraries such as Pytest [14] have been effectively adapted to ML contexts, allowing developers to validate pipeline components, ranging from data loaders, preprocessing functions, to training routines through assertions on tensor shapes, data formats, and parameter updates. Specialized testing frameworks have also become prevalent; notably, DeepChecks [5] provides built-in automated test suites that perform train-test distribution checks to detect data leakage and that monitor model performance to quickly identify data degradation. Scikit-learn further contributes the *check_estimator* utility, offering a standardized test suite ensuring ML models conform to expected implementation patterns, such as fitting consistency, prediction outputs, and proper handling of edge cases. Lastly, data validation frameworks such as Great Expectations serve as an unit testing at the data preparation stage to prevent data quality issues from propagating downstream [10]. While these tools are beneficial for verifying pipeline integrity, they primarily focus on data quality and conformance rather than directly assessing model suitability. Microsoft's ErrorAnalysis toolkit [19], though valuable for diagnosing complex error patterns with post-hoc decision tree visualisations, works only after a model is trained. Because it clusters residuals rather than inspecting the forest's internal structure, it cannot certify during pre-deployment testing whether a candidate RF already meets required structural and behavioral specifications. The gaps altogether emphasize the need for specialized RF-centric pre-screening solutions.

Advanced model-level testing methodologies enhance verification by assessing broader model behaviors. Metamorphic testing addresses the oracle problem by validating predefined relationships, such as proportional scaling between inputs and outputs [29]. Adversarial testing evaluates robustness by crafting slight perturbations to trigger mispredictions, effectively revealing worst-case vulnerabilities. However, recent research stresses its computational challenges, especially for tree ensembles such as RFs [4]. Coverage-guided testing, effective primarily with differentiable neural networks, quantifies how thoroughly internal structures are exercised but faces limitations in fault detection reliability [22]. Rule-based testing, exemplified by CheckList [23], leverages domain-specific expectations to systematically identify subtle logical inconsistencies but relies heavily on manual expert input. Collectively, these advanced methods suffer from significant drawbacks for RF testing, including computational inefficiency, reliance on manual intervention, and limited applicability of metrics designed primarily for differentiable models.

These gaps indicate notable limitations in current methodologies, highlighting the absence of comprehensive internal verification methods specifically tailored for RF models. Motivated by these limitations, our research introduces a systematic, model-based unit testing framework explicitly designed to enhance the robustness and reliability of RF models. Our method evaluates RF performance across diverse synthetic scenarios, supports direct comparisons between pairs of RF models, and provides robust mechanisms for individual model quality assessment even when a good-fit standard is unavailable. This approach empowers practitioners across various disciplines to reliably assess RF model quality, making it accessible even to users with limited machine learning expertise, thereby significantly enhancing confidence in model readiness for critical downstream applications.

III. METHODOLOGY

A. Pipeline Overview

Our model-based unit testing scheme is specifically designed for RF models, since the testing units are designed on top of forest's key parameters such as the number of trees and their tree depths. The testing system is designed to address two distinct scenarios:

1. Performance Similarity Assessment for Paired Models: Given a pair of trained RF models along with a dataset, the objective is to assess their similarity under data perturbations. This assessment is conducted by first imposing robustness by means of perturbations and then through a series of unit tests, including dependency check, prediction ability, and feature importance check units in order.

2. Quality for a Single Model: Similarly, for a single trained RF model along with a dataset, we determine its quality by employing variants of augmentations, all importance together with an additional randomness generator comparison unit.

Each unit test in the pipeline categorizes results into several classes, with each class assigned a specific score. The final assessment on whether two models are similar, or a single model is of good quality, is determined by aggregating the scores from all unit tests. In the following sections, we detail each interconnected testing unit, explaining their individual roles and how they contribute to the overall assessment.

B. Pairwise Model Testing Pipeline

1) Robustness by Perturbation

Let us first explore the pairwise RF models comparison testing pipeline. The pipeline input contains a dataset (X, y), where $X \in \mathbb{R}^{n\times m}$, $y \in \mathbb{R}^{n\times 1}$ are the input data matrices, and n is the number of samples, m is the number of features, and two RF models M_1 with n_1 trees and M_2 with n_2 trees which are both trained. The output of the pipeline would be binary, either M_1 and M_2 are similar in their ability towards interpreting the dataset (X, y), or they are different. We denote by $M(X)$ the prediction of model M on samples in X.

The preprocessing step in our pipeline is dedicated to assessing model robustness through systematic data perturbations. We start by randomly splitting (X, y) into training data (X_{train}, y_{train}) and testing data (X_{test}, y_{test}). For each sample in the test dataset X_{test}, we generate corresponding perturbed samples to form a new dataset X_{ptest}. We apply four distinct perturbations on a per-sample basis: replacing random values with the sample mean, swapping values between "inlier" and "outlier" regions, adding Gaussian noise, and increasing the overall magnitude of the test data. This multifaceted approach enables us to evaluate model behavior under diverse data transformations. The complete perturbation process is detailed in Algorithm 1.

Algorithm 1 Test Data Perturbation for Model Robustness Evaluation

Require: Trained RF models M_1 and M_2, test data $(X_{\text{test}}, y_{\text{test}})$

for each sample i in X_{test} **do**
  Select perturbation method p_i randomly from:
  - Replace with Mean
  - In-and-Out Swap
  - Add Gaussian Noise
  - Increase Magnitude

  Apply selected perturbation p_i to sample i in X_{test}
  $X_{\text{ptest}} \leftarrow X_{\text{test}}$
end for
$\hat{y}_1 \leftarrow M_1(X_{\text{test}})$, $\hat{y}_2 \leftarrow M_2(X_{\text{test}})$
$\hat{y}_{\text{p1}} \leftarrow M_1(X_{\text{ptest}})$, $\hat{y}_{\text{p2}} \leftarrow M_2(X_{\text{ptest}})$
return $(X_{\text{ptest}}, \hat{y}_1, \hat{y}_2, \hat{y}_{\text{p1}}, \hat{y}_{\text{p2}})$

The four perturbation methods are applied to X_{test} to assess model robustness, and perturbed testing data is represented as (X_{ptest}, y_{ptest}) and perturbed dataset is denoted as (X_p, y_p). Detailed description of the perturbation methods are as follows.

a. Mean Replacement: For each sample $X_{i,j}$, apply the following transformation:

$$X_{p_{i,j}} = \begin{cases} \mu_j & \text{with probability } p_{perturb} \\ X_{i,j} & \text{with probability } (1 - p_{perturb}) \end{cases}$$

where μ_j is the mean of the j-th feature.

b. Quantile-based Value Swapping: Let $Q_{j,0.1}$ and $Q_{j,0.9}$ be the $10th$ and $90th$ percentiles of the j-th feature, respectively. Let $I_j = \{i : Q_{j,0.1} \leq X_{i,j} \leq Q_{j,0.9}\}$ and $O_j = \{i : X_{i,j} < Q_{j,0.1} \text{ or } X_{i,j} > Q_{j,0.9}\}$. We then randomly select $q_0\%$ indices from I_j and pair them with randomly selected indices from O_j. For each pair (i_1, i_2), where $i_1 \in I_j$ and $i_2 \in O_j$, we swap their values: $X_{p_{i_1,j}} = X_{i_2,j}$ and $X_{p_{i_2,j}} = X_{i_1,j}$.

c. Gaussian Noise Addition: Add noise to each element $X_{i,j}$ by $X_{p_{i,j}} = X_{i,j} + \epsilon_{i,j}$, where $\epsilon_{i,j} \sim \mathcal{N}(0, \sigma^2_{j,strategy})$, and $\sigma_{j,strategy} = f_1 \cdot \sigma_j + 0.05 \cdot f_2 \cdot |\mu_j|$. Here, σ_j and μ_j are the empirical standard deviation and mean of the j-th feature. Parameters f_1, f_2 are predefined factors that determine the contribution of the empirical standard deviation and the absolute mean to the noise level.

d. Magnitude Increase: We scale each element $X_{i,j}$ by a constant factor $X_{p_{i,j}} = c \cdot X_{i,j}$, where c is a pre-defined value.

These perturbations are applied randomly to each sample in X_{test} with equal probability to generate the perturbed dataset X_{ptest}.

2) Testing Units

Similarity Check Unit The first unit in our testing pipeline is the Similarity Check Unit, which employs an approach blending conventional genetic algorithms with information theory principles. This unit constructs and analyzes MI matrices to assess the structural similarities between the two RF models, M_1 and M_2, with n_1 and n_2 trees respectively.

To ensure comparability, we first equalize the number of trees in both models. Let $n_0 = \min\{n_1, n_2\}$. We randomly remove $\max\{n_1, n_2\} - n_0$ trees from the model with the larger forest, thereby equalizing the number of trees in both models to n_0. We denote the two adjusted models as M_1' and M_2'.

The core idea of this unit involves the construction of intra-model and inter-model MI matrices. Given a RF $\overline{U}$ and a tree order σ, let $\overline{U}_\sigma$ be the RF where the trees are ordered based on σ. Given two RFs U and V with the same numbers μ_0 of trees, with σ_1 and σ_2 being tree orders of U and V, respectively, let $I(U_{\sigma_1}, V_{\sigma_2})$ be the $\mu_0 \times \mu_0$ matrix whose (i, j) entry corresponds to the average MI of the predictions of the ith tree of U_{σ_1} and the jth tree of V_{σ_2} on the test data set. If $U = V$, then we only consider $\sigma_1 = \sigma_2$.

The details of the computation are described in Appendix A, which outlines our use of the KNN algorithm to approximate entropies for tractability. To apply this method to models U and V, we first extract predictions from each tree in both models and treat these predictions for KNN. We then employ the KNN-based MI calculator, initialized with a parameter k_0, to estimate MI between pairs of tree predictions.

Next, to quantify and optimize the structural similarity between two RF models M_1', M_2', we solve

$$\begin{aligned} \min_{\sigma_1, \sigma_2} \quad & \left\| I\big(M'_{1,\sigma_1}, M'_{2,\sigma_2}\big) - I\big(M'_{1,\sigma_1}, M'_{1,\sigma_1}\big) \right\|_F \\ & + \left\| I\big(M'_{1,\sigma_1}, M'_{2,\sigma_2}\big) - I\big(M'_{2,\sigma_2}, M'_{2,\sigma_2}\big) \right\|_F . \end{aligned} \tag{1}$$

Here $|\cdot|_F$ denotes the Frobenius norm. In other words, the objective is to identify the optimal permutations of tree orders in two forests such that the combined sum of the Frobenius norm differences between their MI matrices is minimized. We seek to achieve the best alignment between the forests.

The genetic algorithm employed operates on permutation matrices representing tree orderings within each model. It initializes with a population of N_1 pairs of permutations. For each of the k_1 folds of perturbed test data X_{ptest}, we generate original predictions $\hat{y}_{1,i}$, $\hat{y}_{2,i}$ and perturbed predictions $\hat{y}_{p_1,i}$, $\hat{y}_{p_2,i}$ for all trees i in M_1' and M_2'. The original predictions are used to compute the initial matrices $I\big(M'_{1,\sigma_1^0}, M'_{1,\sigma_1^0}\big)$, $I\big(M'_{2,\sigma_2^0}, M'_{2,\sigma_2^0}\big)$, and perturbed predictions for $I\big(M'_{1,\sigma_1^0}, M'_{2,\sigma_2^0}\big)$.

Through iterative binary tournament selection, we apply genetic operators, either one-point crossover or swap mutation, with equal probability to evolve the population. In each iteration, we compute $I\big(M'_{1,\sigma_1}, M'_{2,\sigma_2}\big)$ based on the new permutations and compute the updated objective function. The algorithm selects the top-performing permutation matrix pairs for subsequent generations.

This process continues for a maximum of t_1 iterations or until the minimum objective value shows no improvement for t_2 consecutive iterations. To account for data variability, we repeat this optimization across k_1 folds of perturbed test data, each subject to unique sample-wise perturbations.

The final output is the average of the minimized objective values across all folds, serving as a quantitative measure of structural similarity between M_1' and M_2'. We then use this similarity measure to infer the closeness between the original models M_1 and M_2. The lower the score, the closer the two models match in their configurations; i.e., the more similar their node hierarchies and decision path layouts.

In this unit, we categorize the outputs from the absolute value of the similarity score, which substantiates the structural likeness of the two models. This unit provides a robust method for assessing model similarity that goes beyond traditional performance metrics, delineating the structural alignment of decision-making processes between different RF models.

Let us denote the averaged minimum objective function across all folds as Sim_{M_1,M_2}. The outcomes for the second unit test are as follows.

(i) If $|Sim_{M_1,M_2}| \leq 85$, the test passes.
(ii) If $|Sim_{M_1,M_2}| \geq 112$, the test fails.
(iii) Otherwise, the test is undetermined.

These threshold values of 85 and 112 are empirically determined based on extensive experimentation across numerous datasets and diverse RF models. Through benchmarking and statistical analysis of similarity scores between known similar and dissimilar model pairs, we establish these boundaries to optimize the balance between false positives and false negatives.

Prediction Ability Check Unit In this unit, we evaluate the similarity of the prediction power between the two models under perturbations of testing data. We compute the 10-fold cross-validated MSE and R-squared for both M_1 and M_2 in terms of X_{ptest}. Consider the fraction of the MSE and R-squared between models M_1 and M_2, and let us denote them as $Frac_{MSE_p}$ and $Frac_{R_p^2}$. Then $Frac_{MSE_p} = \frac{MSE_{p,M_1}}{MSE_{p,M_2}}$ and $Frac_{R_p^2} = \frac{R^2_{p,M_1}}{R^2_{p,M_2}}$. The outcomes for this unit are as follows.

(i) If $Frac_{MSE_p} < 1.22$ and $Frac_{R_p^2} < 1.03$, the test passes.
(ii) If $Frac_{MSE_p} > 1.27$ or $Frac_{R_p^2} \geq 1.03$, the test fails.
(iii) Otherwise, the test is undetermined.

These threshold values (1.22 and 1.27 for MSE fraction, 1.03 for R-squared fraction) are carefully calibrated through extensive empirical testing across diverse datasets and model configurations. The narrow margin for R-squared (1.03) reflects the sensitivity of this metric to model differences, while the wider range for MSE accommodates its greater natural variability under perturbations.

If condition (i) is satisfied, it indicates that both models have learned stable and generalizable patterns from the training data, with their decision-making processes robust to small input perturbations. Conversely, if condition (ii) is met, the models are considered dissimilar, leading to a rejection of similarity. Condition (iii) remains inconclusive, precluding a definitive assessment of model similarity.

Feature Importance Check Unit The Feature Importance Check Unit serves as the final, critical component in our testing pipeline, comparing the models' interpretations of feature significance. This unit employs Shapley values to quantify the contribution of each feature to the models' predictions.

To reliably estimate feature importance, we implement a bootstrap approach. We first generate multiple resampled datasets $\{X_{boot}^{(b)}\}_{b=1}^{B}$ from the original test data X_{test}, where B is the number of bootstrap iterations. Each resampled dataset is subjected to perturbations to obtain the perturbed bootstrapped datasets $\{X_{pboot}^{(b)}\}_{b=1}^{B}$.

For each bootstrap iteration b, we compute Shapley values for both models M_1 and M_2. Let $\phi_i^{(1,b)}$ and $\phi_i^{(2,b)}$ denote the Shapley values of feature i $\in F$ (the set of all features) for models M_1 and M_2, respectively, calculated from the perturbed bootstrapped dataset $X_{pboot}^{(b)}$. The averaged Shapley values across all bootstrap iterations B for each feature i are then denoted as $\bar{\phi}_i^{(1)}$, $\bar{\phi}_i^{(2)}$.

We then construct a similarity vector SV_{pboot}, whose elements are defined by the ratio of the averaged Shapley values for corresponding features in the two models

$$SV_{pboot,i} = \frac{\bar{\phi}_i^{(1)}}{\bar{\phi}_i^{(2)}}.$$

To specifically quantify the overall agreement in feature importance, we first rank features by their combined average importance from both models. Subsequently, we select the top 50% most important features, denoted as $F_{top} \subset F$, and calculate the aggregated similarity metric $\overline{SV}_{pboot}$ as the mean of the similarity ratios of these F_{top} top features.

This aggregated similarity measure $\overline{SV}_{pboot}$ offers an interpretable indicator summarizing the concordance in how the two models prioritize features. Values of $\overline{SV}_{pboot}$ close to 1 signify comparable importance attributed to corresponding features across models, while values significantly deviating from 1 indicate disparities in feature interpretation, thereby revealing intrinsic differences between the two models.

The outcomes of the final unit are as follows.

(i) If all values $0.5 \leq \overline{SV}_{pboot} \leq 2$, the test passes.

(ii) If there exists values $\overline{SV}_{pboot} < 0.5$ or $\overline{SV}_{pboot} > 2$, the test fails.

The threshold values of 0.5 and 2 are computed in the same way as in prior tests.

C. Scoring for Test Units Outcomes

Through this comprehensive workflow, we determine model similarity from multiple perspectives: individual tree structures, information representation, predictive capability, and feature interpretation.

For each unit with multiple outcomes, we define a scoring system. Let U_i denote the *i-th* unit in the test piepline, and let S_i be the score assigned to U_i. These individual unit scores are then combined to produce the final binary output. The scoring framework is as follows.

$$S_1 = \begin{cases} 1, & \text{if condition (i)} \\ 2, & \text{if condition (ii)} \\ 3, & \text{if condition (iii)} \end{cases} \quad S_2 = \begin{cases} 1, & \text{if condition (i)} \\ 2, & \text{if condition (ii)} \\ 3, & \text{if condition (iii)} \end{cases}$$

$$S_3 = \begin{cases} 1, & \text{if condition (i)} \\ 2, & \text{if condition (ii)} \end{cases}$$

The final decision function f combines these outcomes to produce a binary test result of the model pair (M_1, M_2) as follows

$$f(S_1, S_2, S_3) = \begin{cases} 1, & \text{if } (S_1 = 1 \text{ and } S_2 = 1) \text{ or} \\ & (S_1 = 1 \text{ and } S_2 = 3 \text{ and } S_3 = 1) \text{ or} \\ & (S_1 = 3 \text{ and } S_2 = 1 \text{ and } S_3 = 1) \text{ or} \\ & (S_1 = 3 \text{ and } S_2 = 3 \text{ and } S_3 = 1) \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

where 1 indicates that the model pair is classified as similar and reliable, and 0 indicates dissimilarities.

The overall test pipeline is shown in Figure 1.

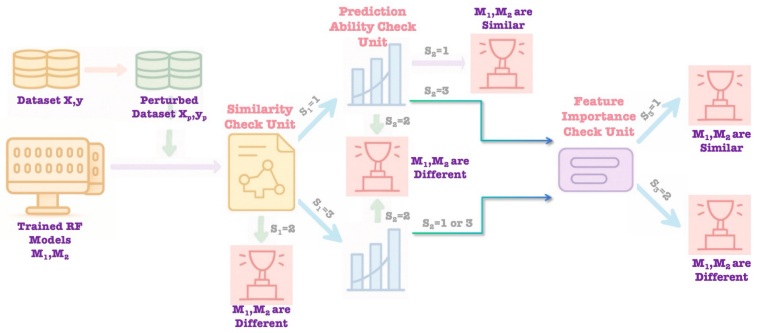

Figure 1: Pairwise Model Similarity Unit Testing Pipeline

D. Individual Model Testing Pipeline

Given a single trained RF model M and a dataset (X, y), the goal is to evaluate the robustness and suitability of M. To achieve this, we construct a diverse pool of synthetic RF models in a preprocessing step called the *RF Model Generator*. Specifically, we define a model space $\mathcal{M}$ as $\mathcal{M} = \{M(n, d) : n \in \mathcal{N}, d \in \mathcal{D}\}$, where $M(n, d)$ represents a RF model with n trees and maximum depth d, $\mathcal{N}$ is the set of candidate numbers of trees, and $\mathcal{D}$ is the set of candidate maximum tree depths.

In this pipeline, the evaluated model M is held constant while an extensive set of synthetic models is generated. Each synthetic model is paired with M, forming a model pair that undergoes the three unit tests, namely the Similarity Check Unit, the Prediction Ability Check Unit, and the Feature Impact Analysis Unit, as described in Section III-B. A new scoring rule is then applied to quantify the percentage of model pairs satisfying the specified conditions. This approach extends the pairwise testing framework, enabling a comprehensive assessment of M's performance under data perturbations.

From the space $\mathcal{M}$, we uniformly at random select a subset $\mathcal{M}_{\text{sub}} \subset \mathcal{M}$ consisting of N_2 models which serve as the comparison set for the incoming model M. We train each $M(n, d) \in \mathcal{M}_{\text{sub}}$ on (X, y). Each synthetic model $\overline{M} \in \mathcal{M}_{\text{sub}}$ is paired with the incoming model M. The pairwise comparison between M and each $\overline{M}$ follows the same unit testing procedure as described in Section III-B. To make every unit test robust to sampling variation, we subject the test data to k_2 independent perturbations. Let

$X^{(j)}_{\text{ptest}}$ denote the j-th perturbed version of the original test set X_{test}, with $j = 1, \ldots, k_2$. For each synthetic comparator $\overline{M}$, we run the three unit tests on every fold, obtaining per-fold similarity, prediction ratio, and feature impact statistics, $Sim^{(j)}_{M,\overline{M}}$, $Frac^{(j)}_{R^2_p}(M, \overline{M})$, $Frac^{(j)}_{MSE_p}(M, \overline{M})$, and $\overline{SV}^{(j)}_{\text{pboot}}(M, \overline{M})$. We then aggregate over folds by simple averaging, and to this end, we have $\overline{Sim}_{M,\overline{M}}$, $\overline{Frac}_{R^2_p}(M, \overline{M})$, $\overline{Frac}_{MSE_p}(M, \overline{M})$, and $\overline{\overline{SV}}_{\text{pboot}}(M, \overline{M})$. The outcome of the test $g\big(S_1(M, \overline{M}), S_2(M, \overline{M}), S_3(M, \overline{M})\big)$ is based on the decision function g specified in rule (3) below. Aggregating the outcomes, we categorize model M as either *PASS* or *FAIL*.

We now score the outcomes from each unit, starting with the Similarity check unit:

(i) If more than 30% of pair-wise comparisons have $\overline{Sim}_{M,\overline{M}} \leq 89$, mark $S_1 = 1$.
(ii) Otherwise, mark $S_1 = 0$.

The outcomes for prediction-ability check unit are:

(i) If more than 39% of comparisons have $\overline{Frac}_{R^2_p}(M, \overline{M}) < 1.10$ and more than 42% of comparisons have $\overline{Frac}_{MSE_p}(M, \overline{M}) < 1.53$, then $S_2 = 1$.
(ii) Otherwise, $S_2 = 0$.

The outcomes for the feature-importance check unit are:

(i) If more than 45% of comparisons have all values $\overline{\overline{SV}}_{\text{pboot}}(M, \overline{M}) \in [0.5, 2]$, mark $S_3 = 1$.
(ii) Otherwise, mark $S_3 = 0$.

The final decision function g is then defined as

$$g(S_1, S_2, S_3) = \begin{cases} 1, & \text{if } (S_1 + S_2 = 2) \\ & \quad \text{or } (S_1 + S_2 = 1 \text{ and } S_3 = 1), \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

where value $g = 1$ implies the test passes, and $g = 0$ it fails. It is similar to Figure 2 but the values are different. The scheme is shown in Figure 2.

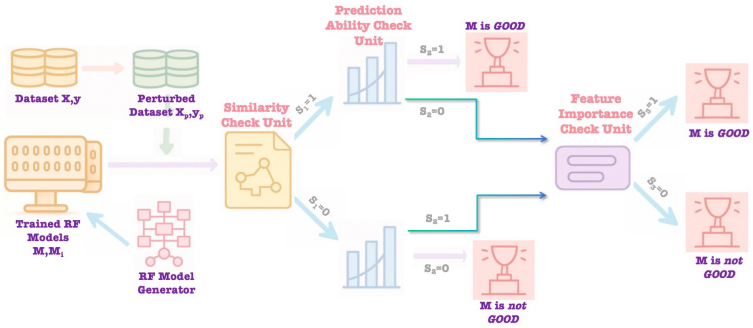

Figure 2: Single Model Goodness of Fit Unit Testing Pipeline

IV. Numerical Experiments

A. Implementation and Evaluation Datasets

This section presents the results of our framework, detailing the datasets used to evaluate our model testing pipeline and interpreting the outcomes.

In the following experiments, we set the parameters and constants values as follows. For pairwise testing, we build the paired models (M_1, M_2) by randomly selecting the number of trees from the set $\{2, 3, 4, 5, 8, 10, 12, 15, 16, 20, 25, 30, 45, 50, 75, 100, 200, 500\}$ and the maximum depth of each tree from the set $\{1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 18, 20, 25, 50\}$. For Algorithm 1, we set $p_{perturb} = 0.05$, $q_0 = 10$, $c = 1.15$, and paired $(f_1, f_2) \in \{(1, 0), (0, 1), (0.5, 0.5)\}$.

For single model testing, we take care of synthetic comparison model selection through the number of trees and tree depth characteristics of RF. We set $\mathcal{N} = \{1, 2, 3, 5, 7, 10, 15, 20, 25, 50, 75, 100, 200, 500\}$, $\mathcal{D} = \{1, 2, 3, 5, 10, 15, 25, 50\}$, and $N_2 = 1{,}000$ for all of the following experiments. The details of how the pools of models for both cases are provided next.

We set the following parameters for the Similarity Check Unit. We initialize the KNN-based MI estimator with $k_0 = 5$ neighbors, select an initial population of $N_1 = 5$ permutation pairs, run the algorithm over $k_1 = 100$ folds of perturbed test data with each fold containing $k = 100$ samples, and allow the genetic algorithm to run for up to $t_1 = 150$ iterations or until it detects no improvement over $t_2 = 25$ consecutive iterations.

In the Feature Importance Check unit, we compute Shapley values for each model in the $n_2 = 25$ bootstrapped and perturbed test datasets.

We then evaluate our approach using data from several real-world datasets and their simulated offshoots: Friedman #1 [9], Borehole [12], SARCOS [27], and Satellite Drag [15]. Table I shows the sample count and number of model pairs tested per dataset.

The principle for choosing such datasets is their variability in complexity, dimensionality, and feature characteristics, allowing us to comprehensively assess the capabilities of our approach across diverse modeling scenarios, from controlled synthetic data to complex real-world problems, from low-dimension to high-dimension spaces, including different feature types and interactions between features. This comprehensive evaluation can prove both strengths and limitations of our testing system.

The ground truths regarding whether pairs of RF models are structurally similar or dissimilar or one RF model is a good fit for a given dataset or not are known by design. To establish a robust reference, an initial RF model demonstrating goodness-of-fit (GOF) is designed for each dataset. This design process involves careful data splitting, iterative hyperparameter tuning guided by Out-of-Bag error on the training set, and GOF evaluation on the test set using Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Adjusted R-squared.

Once this benchmark GOF model is established for a dataset, we create a "pool of Ground Truth Good models" by taking the baseline GOF model and introducing slight, controlled variations to its tree-structure hyperparameters $n_{\text{estimators}}$, $\max_{\text{depth}}$, $\min_{\text{samples\_split}}$, $\min_{\text{samples\_leaf}}$ and $\max_{\text{features}}$. Consequently, models within this pool, or between this pool

and the baseline, are considered "similar" due to their closely related configurations.

A parallel procedure is employed to construct a "pool of Ground Truth Poor models." This starts with RF models identified as poorly performing (based on the same GOF metrics by showing an opposite trend) or models deliberately configured with suboptimal hyperparameters. These initial poor models also undergo slight structural tuning to generate a diverse pool. To this end, the pool of models that our pairs are randomly picked from are built, resulting in number of trees from the set $\{2,\ 3,\ 4,\ 5,\ 8,\ 10,\ 12,\ 15,\ 16,\ 20,\ 25,\ 30,\ 45,$ $50,\ 75,\ 100,\ 200,\ 500\}$ and the maximum depth of each tree from the set $\{1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 8,$ $10,\ 12,\ 15,\ 18,\ 20,\ 25,\ 50\}$, as mentioned above. The same approach applies to the one-model case.

The subsequent experiments are conducted under the assumption that these structural cues are unknown, and a sampled pair (M_1, M_2) is labeled

$$\textbf{Pass} = 1 \iff M_1, M_2 \in \mathcal{G} \text{ and } \|\theta(M_1) - \theta(M_2)\|_\infty \leq \delta,$$

where $\mathcal{G}$ denotes the pool of "ground-truth good" models, each dataset's benchmark GOF model and its controlled hyperparameter variants, while $\theta(\cdot)$ stacks the test metrics, and the tolerance is fixed at $\delta = 0.075$.

The computational time depends on both the dataset and forest sizes. For instance, using the Friedman dataset, when $n_0 = 15$, the full pipeline completes in under 3 minutes; when $n_0 = 50$, it takes approximately 10 minutes; and when $n_0 = 75$ or larger, the process requires roughly 20-45 minutes per pair.

Characteristic	Friedman #1	Borehole	SARCOS	Satellite
# of Features	5	8	21	8
# of Samples	1,000	10,000	44,484	1,000
Target depends on feature interactions	Yes	No	Yes	Yes
Data Types	Synthetic	Synthetic	Real	Real
Feature Types	Cont.	Cont., Int.	Cont.	Cont., Cat., Int.
# Pairs of Models in Sec. IV-B	500	500	200	100
# Pairs of Models in Sec. V-A	500	250	100	100

Table I: Regression Dataset Variability

B. Evaluation Results on Pairwise Model Unit Testing

To comprehensively synthesize the evaluation, we summarize the results across all four datasets in Table II. The score S_i is computed according to the rule given in (2). The *Counts* column indicates the number of pairs that pass the test out of the total pairs evaluated at that unit, and the *Percentage* column reports the corresponding value. *Ours Final* corresponds to the final decision of our methodology, while *Precision* and *Recall* are performance metrics assessing our prediction against the ground truth. The Similarity Check Unit, placed at the beginning of the testing pipeline, employs MI alongside an optimization-based permutation strategy to determine the optimal alignment between the tree structures of two RF models. Leveraging MI in this context is statistically well-founded, as it effectively captures nonlinear relationships between tree predictions, a complex characteristic otherwise challenging to quantify. Empirical evidence from Table II strongly supports the efficiency of this step. Notably, significant proportions of model pairs, namely 58.6% (Friedman), 41.0% (Borehole), 21.5% (SARCOS), and 39.0% (Satellite), are correctly classified as dissimilar solely through this initial Similarity Check Unit. These results closely align with the ground truth dissimilarity rates (64.8%, 43.6%, 60.0%, and 59.0%, respectively). Thus, the Similarity Check Unit alone can reliably filter out a majority of dissimilar pairs early on, which proves its necessity.

Following this initial filtering, the Prediction Ability Check Unit further refines the assessment by examining whether models previously labeled as "similar" or "undetermined" according to their structural similarities also produce consistent predictive results. As observed in Table II, the proportion of pairs that fulfill the criteria of the Prediction Unit varies substantially across datasets, from as low as 24.2% (Friedman) to as high as 97.5% (Borehole). This variability underscores the Prediction Unit's critical role in capturing nuanced differences in predictive performance that structural similarity alone may miss. Consequently, the Prediction Unit is not merely supplemental but integral, significantly enhancing the reliability and discriminative power of the overall pipeline.

The third unit addresses cases in which the rigorous requirements of the first two units, namely structural and predictive alignments, may inadvertently label truly similar model pairs as undetermined. By focusing on the alignment of feature importance, this unit provides an additional perspective on model similarity, reflecting real-world considerations regarding which features are deemed influential and which can be disregarded. As indicated by the high percentages of passing pairs (97.4%, 72.5%, 83.3%, and 68.8%, respectively), this unit effectively rescues many pairs that do not strictly satisfy both prior tests, confirming their practical equivalence even when strict structural or predictive criteria are not fully met.

Collectively, arranging these three units sequentially as presented forms a comprehensive and coherent evaluation pipeline. The average precision and recall across all datasets are 0.915 and 0.8475, respectively, with standard deviations of 0.0443 and 0.0850. These results demonstrate the robustness and consistency of our methodology in reliably distinguishing similar from dissimilar RF model pairs under diverse scenarios.

Among the four benchmarks, the SARCOS dataset presents the most challenging scenario for our pairwise model similarity pipeline, yielding the lowest recall (0.76) and the second-lowest precision (0.89). This suggests that a higher number of truly similar model pairs are being missed by our methodology on this dataset. Two primary factors contribute to this performance drop. First, SARCOS is characterized by its high dimensionality, with 21 continuous input features, more than twice that of the next largest dataset, and strong cross-term effects induced by the underlying robot-

Dataset	S_i	Similarity Unit		Prediction Unit		Feature Unit		Ours Final	Ground Truth	Precision	Recall
		Counts	Percentage (%)	Counts	Percentage (%)	Counts	Percentage (%)	Counts	Counts		
Friedman	1	138/500	27.6	50/207	24.2	148/152	97.4	176/500	165/500	0.87	0.93
	2	293/500	58.6	27/207	13.0	4/152	2.6				
	3	69/500	13.8	130/207	62.8	—	—				
Borehole	1	250/500	50.0	268/275	97.5	37/51	72.5	282/500	301/500	0.97	0.91
	2	205/500	41.0	0/275	0.0	14/51	27.5				
	3	45/500	9.0	7/275	2.5	—	—				
SARCOS	1	87/200	43.5	45/157	28.6	50/60	83.3	80/200	94/200	0.89	0.76
	2	43/200	21.5	67/157	42.7	10/60	16.7				
	3	70/200	35.0	45/157	28.7	—	—				
Satellite	1	27/100	27.0	38/61	62.3	22/32	68.8	41/100	48/100	0.93	0.79
	2	39/100	39.0	10/61	16.4	10/32	31.2				
	3	34/100	34.0	13/61	21.3	—	—				

Table II: Quantitative results summarizing the number and percentage of pairwise model comparisons that pass each unit test in the order of their application.

Dataset	S_i	Similarity Unit		Prediction Unit		Feature Unit		Ours Final	Ground Truth	Precision	Recall
		Counts	Percentage (%)	Counts	Percentage (%)	Counts	Percentage (%)	Counts	Counts		
Friedman	1	178/500	35.6	210/500	42.0	61/332	18.4	89/500	114/500	0.93	0.73
	0	322/500	64.4	290/500	58.0	271/332	81.6				
Borehole	1	96/250	38.4	113/250	45.2	158/169	93.5	178/250	155/250	0.85	0.97
	0	154/250	61.6	137/250	54.8	11/169	6.5				
SARCOS	1	21/100	21.0	37/100	37.0	19/32	59.4	32/100	39/100	0.78	0.64
	0	79/100	79.0	63/100	63.0	13/32	40.6				
Satellite	1	35/100	35.0	43/100	43.0	35/42	83.3	53/100	41/100	0.72	0.93
	0	65/100	65.0	57/100	57.0	7/42	16.7				

Table III: Quantitative results summarizing the number and percentage of synthetic pairwise models that pass each unit test in the order of their application.

arm dynamics. Estimating MI between tree predictions in such a high-dimensional and complex setting can be noisy. Consequently, the initial Similarity Check Unit tends to label fewer pairs as similar (21.5%) than the ground-truth similarity rate (47.0%), as optimal tree alignments become less clear-cut and the "signal" of structural similarity can be obscured by individual tree variations when dealing with many correlated, or less informative features. Second, the target values in SARCOS span six orders of magnitude, which means that even small absolute errors can significantly inflate the perturbed-to-baseline ratios used by the Prediction Unit. As a result, several structurally matched RF models fail the prediction thresholds, as minor structural changes can lead to more pronounced functional differences in this challenging space. This in turn, leaves fewer opportunities for the Feature Unit to subsequently "rescue" these pairs. Collectively, these characteristics make the SARCOS dataset the toughest to deal with.

False Positives (FP) and False Negatives (FN) are inherent challenges in such a comparison task. Close inspection of misclassified pairs reveals specific structural patterns contributing to these errors. Most FPs occur when both compared RFs are large (more than 100 trees) but very shallow (depth less than or equal to 3). Such models tend to share many high-level split rules. Consequently, their tree-wise MI matrix can appear aligned even when their deeper, more detailed structures differ, potentially causing the Similarity Check Unit to incorrectly accept them as similar. The Feature Unit, designed to be more tolerant to "rescue" pairs, also inadvertently contribute to FPs if it identifies feature importance alignment between these structurally different but superficially similar shallow models.

Conversely, FNs, which appear to be more prevalent in datasets like SARCOS, predominantly arise from medium-sized forests (30–75 trees) with moderate depths (10-15). The permutation search in the Similarity Unit aligns their trees well, producing a *PASS*. Yet each forest is trained with its own bootstrap sample and feature subsampling stream, so their terminal leaf means are not identical. Because SARCOS targets vary by several orders of magnitude, even a small leaf mean shift pushes the perturbed-to-baseline error ratios above the hard fail limits in the Prediction Unit. Therefore, the pipeline stops before the Feature Unit can review the pair and rescue the outcome. However, even with the nature of FNs and FPs presence, our pipeline achieves an overall high precision and recall.

In this section, we present the results of our pipeline for evaluating the quality of individual RF models. As shown in Table III, we evaluate 500, 250, 100, and 100 single models on the respective datasets. All four datasets are assessed using a shared pool of synthetic RF models generated as described in Section IV-A. The score S_i is computed according to the rule given in (3).

In contrast to the pairwise similarity testing pipeline, where the Similarity Unit alone effectively filters model pairs, evaluating the quality of an individual model requires a more intricate procedure. This is due to the additional randomness introduced by synthetic model pairing, which makes it con-

Friedman	S_i	Similarity Unit		Prediction Unit		Feature Unit		Ours Final		
		Counts	Percentage (%)	Counts	Percentage (%)	Counts	Percentage (%)	Counts	Precision	Recall
Without Similarity Unit	1	—	—	157/500	31.4	205/286	71.7			
	2	—	—	214/500	42.8	81/286	28.3	205/500	0.63	0.78
	3	—	—	129/500	25.8	—	—			
Without Prediction Unit	1	138/500	27.6	—	—	165/207	79.7			
	2	293/500	58.6	—	—	42/207	20.3	165/500	0.79	0.79
	3	69/500	13.8	—	—	—	—			
Without Feature Unit	1	138/500	27.6	50/207	24.2	—	—			
	2	293/500	58.6	27/207	13.0	—	—	28/500	0.93	0.16
	3	69/500	13.8	130/207	62.8	—	—			

Table IV: Quantitative ablation tests on the Friedman dataset illustrate the impact of each unit in the overall configuration.

siderably more challenging to confidently filter the pairs as in the pairwise cases. Consequently, the Similarity and Prediction Units are allotted equal importance. As shown in Table III, the percentages of models attaining a score of 1 in these two units are generally comparable. For instance, in the Borehole test, 38.4% of the evaluated models exhibit structure-level similarity with their synthetic pairings, while 45.2% demonstrate prediction-level similarity. However, our methodology's final decision rate is significantly higher, at 71.2%, indicating that the third (Feature) unit contributes essential additional discriminatory power not captured by the first two units alone. Given that the Similarity and Prediction Units combined are insufficient to conclusively assess a model's quality on a given dataset, it remains beneficial and reliable to initially filter out dissimilar synthetic pairs, thereby again reducing the workload required by the Feature Unit.

After applying the complete unit testing pipeline, the effectiveness of our methodology is confirmed by consistently achieving high values of precision or recall, typically around or above 90% across datasets (see Table III). It is expected that the overall performance may be slightly lower compared to pairwise model testing because the individual models under evaluation are entirely black-box and assessed without an initial baseline or comparative model. Additionally, since all four datasets share the same synthetic-model pool, dataset-specific characteristics might influence evaluation outcomes. Nonetheless, these results demonstrate that our pipeline effectively balances precision and recall, validating its capability in reliably identifying high-quality individual RF models.

V. Ablation Study

To systematically investigate the relative importance of each evaluation unit within our proposed framework, we conduct an ablation study using the Friedman dataset for both pairwise and single model testing pipelines. This study involves sequentially removing each of the three units and examining the resultant changes in performance metrics. The values used for testing are identical to those presented in Table III.

A. Evaluation Results on Single Model Unit Testing

B. Pairwise Model Case Study

Table IV exhibits the results. By isolating each unit's contribution to the pipeline's effectiveness, the results indicate that omitting any single unit leads to untrustworthy decisions. The *Ours Final* column shows the final decision when one unit is omitted. Removing the Similarity Unit substantially simplifies computational complexity, as this unit contributes the most to overall computation time. However, the absence of assessing the tree-structure alignment via MI should significantly impact accuracy, as evidenced by a decrease in evaluation metrics from precision and recall values of 0.87 and 0.93 to 0.63 and 0.78, respectively. Additionally, the final number of pairs classified as similar increases markedly, indicating that without the structural evaluation, some pairs with intrinsically different decision processes but similar predictions and feature importance profiles are incorrectly identified as similar.

The Similarity Unit dominates the runtime: for 500 pairs its permutation-based search accounts for roughly 43 hours, and the cost rises steeply once each forest exceeds 50 trees. Dropping this unit therefore would yield a large speed-up, but at an unacceptable price as described above. Because the pipeline's primary objective is reliable model pre-checking, we should retain the Similarity Unit.

When the Prediction Unit is omitted, both precision and recall metrics drop to 0.79, reflecting a moderate yet insufficient performance. This aligns with the intended role of the Prediction Unit, which complements the structural assessments from the Similarity Unit by explicitly evaluating prediction concordance.

Finally, excluding the Feature Importance Unit dramatically reduces recall to 0.16, despite retaining a high precision of 0.93. This significant reduction demonstrates that, the Similarity and Prediction Units provide overly strict checks. Without assessing practical alignment in terms of feature prioritization, the pipeline fails to identify several model pairs that demonstrate strong task-specific feature importance alignment.

In all, when we drop the Similarity Unit, precision sinks to 0.63 for *Friedman*. The pipeline now declares many extra pairs as "similar," inflating the false positive count. Without a structural screen based on MI, two forests that share only superficial prediction patterns slip through, so precision is the first casualty while recall is still within the fine range.

Conversely, omitting the Feature Unit drives recall down to 0.16 while precision stays high. Here, the earlier structural and predictive checks act as a double gate: if either one blocks a genuinely similar pair, no later unit can recover it. The feature-level comparison serves as a rescuer, saving pairs whose tree order or output scale differs slightly yet rely on the same

		Similarity Unit		Prediction Unit		Feature Unit		Ours Final		
Friedman	S_i	**Counts**	**Percentage (%)**	**Counts**	**Percentage (%)**	**Counts**	**Percentage (%)**	**Counts**	**Precision**	**Recall**
Without Similarity Unit	1	—	—	210/500	42.0	128/500	25.6	102/500	0.77	0.69
	0	—	—	290/500	58.0	372/500	74.4			
Without Prediction Unit	1	178/500	35.6	—	—	199/500	39.8	136/500	0.55	0.66
	0	322/500	64.4	—	—	301/500	60.2			
Without Feature Unit	1	178/500	35.6	210/500	42.0	—	—	25/500	0.82	0.20
	0	322/500	64.4	290/500	58.0	—	—			

Table V: Quantitative ablation tests on the Friedman dataset illustrate the impact of each unit in the overall configuration.

explanatory cues. Its absence therefore manifests itself as a surge in FNs.

C. Single Model Case Study

In the single model unit testing pipeline (results presented in Table V), the three units function as complementary components that together form a robust evaluation framework, like the angles of a pyramid, each providing essential structural support. By isolating each unit's contribution to the pipeline's effectiveness, the results indicate that omitting any single unit significantly compromises the integrity of the methodology. The *Ours Final* column shows the final decision when one unit is omitted. When we systematically removed individual units, the precision of our classification dropped substantially from the original 0.93 to 0.77, 0.55, and 0.82, respectively. These results directly demonstrate that more *Fail* models are misclassified as *Pass* when the pipeline is incomplete. This empirical evidence strongly validates our multi-faceted approach to model evaluation.

VI. Conclusion

In this paper, we introduce a RF model-oriented unit testing approach that examines models on an all-rounded perspective, from their intrinsic tree structures to their outlier performance. We present a comprehensive and systematic pipeline for evaluating the quality of RF models, leveraging a multi-unit testing approach comprising structural similarity, predictive consistency, and feature impact analysis. Extensive empirical analyses demonstrate that our testing pipeline reliably differentiates models, efficiently identifies subtle but meaningful differences, and robustly assesses individual RF model quality, providing a valuable toolkit for practitioners deploying RF models in high-stakes applications.

Future work could focus on extending this framework to other model classes beyond RFs and developing dynamic, dataset-oriented thresholds that adapt to specific data characteristics, enabling greater flexibility across varying application domains and model complexities.

References

[1] Alam, Md. Z., Rahman, M. S., and Rahman, M. S., "A Random Forest based predictor for medical data classification using feature ranking," *Informatics in Medicine Unlocked*, 2019.

[2] Benali, L., Notton, G., Fouilly, A., Voyant, C., and Dizene, R., "Solar radiation forecasting using artificial neural network and Random Forest methods: application to normal beam, horizontal diffuse and global components," *Renewable Energy*, 2019, 132, 871-884.

[3] Breiman, L., "Random Forests," *Machine Learning*, 2001, 45, 5-32.

[4] Cascioli, L., Devos, L., Kuzelka, O., and Davis, J., "Faster repeated evasion attacks in tree ensembles," *Proceedings of the NeurIPS Conference*, 2024.

[5] Chorev, S., Tannor, P., Israel, DB., Bressler, N., Gabbay, I., Hutnik, N., Liberman, J., Perlmutter, M., Romanyshyn., Y., and Rokach, L., "Deepchecks: a library for testing and validating machine learning models and data," *Journal of Machine Learning Research*, 2022, 23, 1-6.

[6] Cutler, A., Cutler, D. R., and Stevens, J. R., "Random Forests," *Ensemble Machine Learning: Methods and Applications*, 2012, 157-175.

[7] Fanelli, G., Dantone, M., Gall, J., and Fossati, A., "Random Forests for real time 3D face analysis," *International Journal of Computer Vision*, 2012, 101(3).

[8] Fix, E., Hodges, and J. L., "Discriminatory analysis-nonparametric discrimination: consistency properties," *International Statistical Review*, 1989, 57(3), 238-247.

[9] Friedman, J. H., "Multivariate adaptive regression splines," *The Annals of Statistics*, 1991, 19(1), 1-67.

[10] Geirhos, R., Jacobsen, J. H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., and Wichmann, F.A., "Shortcut learning in deep neural networks," *Nature Machine Intelligence*, 2020, 2(11), 665-673.

[11] Han, T., Jiang, D., Zhao, Q., Wang, L., and Yin, K., "Comparison of Random Forest, artificial neural networks and support vector machine for intelligent diagnosis of rotating machinery," *Transactions of the Institute of Measurement and Control*, 2018, 40(8), 2681-2693.

[12] Harper, W. V., Gupta, S. K., "Sensitivity/uncertainty analysis of a borehole scenario comparing latin hypercube sampling and deterministic sensitivity approaches," *Battelle Memorial Institute*, 1983.

[13] Jia, L., Zhong, H., and Huang, L., "The unit test quality of deep learning libraries: a mutation analysis," *IEEE International Conference on Software Maintenance and Evolution*, 2021, 47-57.

[14] Krekel, H., Oliveira, B., Pfannschmidt, R., Bruynooghe, F., Laugher, B., and Bruhin, F., "Pytest," 2004. https://github.com/pytest-dev/pytest.

[15] Mehta, P. M., Walker, A., Lawrence, E., Linares, R., Higdon, D., and Koller, J., "Modeling satellite drag coefficients with response surfaces," *Advances in Space Research*, 2014, 54(8), 1590-1607.

[16] Mohapatra, P., Chakravarty, S., and Dash, P. K., "An improved cuckoo search based extreme learning machine for medical data classification," *Swarm and Evolutionary Computation*, 2015, 24, 25-49.

[17] Murphy, C., Kaiser, G., Hu, L., and Wu, L., "Properties of machine learning applications for use in metamorphic testing," *International Conference on Software Engineering and Knowledge Engineering*, 2008, 867-872.

[18] Nejadgholi, M., Yang, J., "A study of oracle approximations in testing deep learning libraries," *IEEE/ACM International Conference on Automated Software Engineering*, 2019, 785-796.

[19] Nushi, B., "Responsible machine learning with error analysis," 2021. https://github.com/microsoft/responsible-ai-widgets/.

[20] Özen, F., "Random Forest regression for prediction of covid-19 daily cases and deaths in turkey," *UCSF: Center for Bioinformatics and Molecular Biostatistics*, 2024, 10(4).

[21] Piernik, M., Brzezinski, D., and Zawadzki, P., "Random similarity forests," *Machine Learning and Knowledge Discovery in Databases*, 2023, 53-69.

[22] Prasetya, I. S. W. B., Klomp, R., "Test model coverage analysis under uncertainty: extended version," *Software and Systems Modeling*, 2021, 20, 383-403.

[23] Ribeiro, M. T., Wu, T., Guestrin, C., and Singh, S., "Beyond accuracy: behavioral testing of NLP models with checklist," *Proceedings of the*

58th Annual Meeting of the Association for Computational Linguistics, 2020, 4902-4912.

[24] Riestu, I. M., Hidayat, H. Y., "Landslide susceptibility mapping using Random Forest algorithm and its correlation with land use in batu city, jawa timur," *IOP Conference Series: Earth and Environmental Science*, 2022, 1127.

[25] Saha, S., Saha, M., Mukherjee, K., Arabameri, A., Ngo, P.T.T., and Paul, G.C., "Predicting the deforestation probability using the binary logistic regression, Random Forest, ensemble rotational forest, reptree: a case study at the gumani river basin, india," *The Science of the Total Environment*, 2020, 730.

[26] Simon, S., Glaum, P., and Valdovinos, F., "Interpreting Random Forest analysis of ecological models to move from prediction to explanation," *Scientific Reports*, 2023, 13.

[27] Vijayakumar, S., Schaal, S., "Locally weighted projection regression: an o(n) algorithm for incremental real time learning in high dimensional space," *Proceedings of the 7th International Conference on Machine Learning*, 2000, 1079-1086.

[28] Wiecki, T. V., Campbell, A., Lent, J., and Stauth, J., "All that glitters is not gold: comparing backtest and out-of-sample performance on a large cohort of trading algorithms," *The Journal of Investing*, 2016, 25(3), 69-80.

[29] Zhou, Z. Q., Xiang, S., and Chen, T. Y., "Metamorphic testing for software quality assessment: a study of search engines," *IEEE Transactions on Software Engineering*, 2016, 42, 264-284.

APPENDIX A
APPROXIMATING MUTUAL INFORMATION WITH K-NEAREST NEIGHBORS

For continuous random variables X and Y, the MI $I(X, Y)$ is defined as

$$I(X,Y) = \iint p(x,y) \log \frac{p(x,y)}{p(x)p(y)} dxdy \tag{4}$$

where $p(x, y)$ is the joint probability density function of X and Y, and $p(x)$ and $p(y)$ are the marginal probability density functions of X and Y, respectively.

Exact calculation of MI for continuous variables is often intractable. One approximation method is the KNNs approach, introduced by Kozachenko and Leonenko [8].

The KNN method for MI estimation is based on the idea of estimating local densities around data points. For a set of N points $\{x_i\}_{i=1}^N$ in $\mathbb{R}^d$, we find the distance $\epsilon_{k,p}$ from each point x_i to its k-th nearest neighbor in the l_p space ($p \geq 1$). The local density estimate at x_i is then given by

$$\hat{f}_X(x_i) = \frac{k/N}{V_{p,d}(\epsilon_{k,p})^d} \tag{5}$$

where $V_{p,d}(r)$ is the volume of the l_p-ball of radius r in $\mathbb{R}^d$. The idea is to find a point x'_k close to the k-th nearest neighbor x_k of x_i, with distance $\|x'_k - x_k\| \leq \tau\epsilon_{k,p}$, where τ is a small constant. This approximation method introduces a bounded approximation error but can significantly improve computational efficiency.

This approximation is based on the assumption that the k nearest neighbors of a point provide a good estimate of the local density around that point. In regions of high density, the k nearest neighbors will be closer, resulting in a smaller volume and thus a higher density estimate.

From this density estimate, we can approximate the entropy

$$\hat{H}(X) = -\frac{1}{N}\sum_{i=1}^{N}[\log \hat{f}_X(x_i)]. \tag{6}$$

In our work, we leverage this KNN-based MI estimation technique to quantify the structural similarity between two RF models. This approach offers a computationally efficient alternative to traditional Kernel Density Estimation techniques, making it particularly suitable for comparing complex ensemble models like RFs.