Int. J. Advance Soft Compu. Appl, Vol. 18, No. 2, July 2026
Print ISSN: 2710-1274, Online ISSN: 2074-8523

Automated Auditing Framework for Detecting Overfitting and Data Leakage in Machine Learning Models

Faisal AL-Saqqar[1], Mohammad I. Nusir[2], Amjad H. Alkilani[3]

[1]Computer Science Department, Al al-Bayt University, Mafraq, Jordan
Email: faisalss@aabu.edu.jo
[2]CBM Integrated Software Inc, San Diego, CA, USA
Email:mnuseir@cbmis.com
[3]Forbes School of Business & Technology, University of Arizona Global Campus, Chandler, Arizona, USA
Email:Amjad.Alkilani@uagc.edu

Abstract

Machine learning now informs decisions in hospitals, banks, and hiring pipelines, where a model's integrity matters as much as its accuracy. Overfitting, data leakage, and feature bias still slip past routine evaluation and surface only after deployment. In this paper, we present an automated auditing framework of six modules that catches these problems beforehand. Overfitting is read from the train-test accuracy gap, stabilized over repeated stratified splits rather than a single partition. Leakage is screened by feature-target correlation and an index-overlap check; feature bias by Mean Decrease Impurity, permutation importance, a model-agnostic SHAP step, and a Gini index. Statistical validation uses stratified cross-validation and paired t-tests corrected for multiple comparisons, and a fairness module reports demographic parity and equalized-odds gaps when a protected attribute is available. In the ablation study the full framework scores 0.677, ahead of every reduced version. We tested it on four datasets with six classifiers. On Breast Cancer Wisconsin it flagged Gradient Boosting and Decision Tree as overfitting; a 30-split analysis confirmed both and placed Random Forest below the 0.05 gap (mean 0.037), where the conventional single-split test would have mislabeled it in about a quarter of splits. Injected leakage was caught at correlation 0.971.

Keywords: *data leakage, feature bias, machine learning auditing, overfitting detection, robustness testing*

1 Introduction

Machine learning has moved out of the lab and into places where a wrong prediction has real consequences. Hospitals use models to support diagnosis. Banks rely on them for credit scoring and fraud detection. Human-resource departments use them to forecast attrition. Across these settings, a model that is unreliable, biased, or trained on leaky data is more than a statistical embarrassment. It is a liability.

Received 12 April 2026; Accepted 25 June 2026

Model architectures and training tricks have advanced quickly. Whether a model can actually be trusted has received much less attention than whether it is accurate. A model can fit its training data almost perfectly and then fail on unseen examples because it overfits. A model with a high AUC-ROC can be quietly broken because information from the test set leaked into training. A model can post balanced metrics and still be unfair because two or three features do almost all of the work. These failure modes are documented in the literature and continue to show up in deployed systems [1, 2].

Of the three, overfitting is the most common. When a model memorises the quirks of its training data instead of the underlying pattern, the gap between training and test accuracy opens up. Deep networks and unconstrained decision trees can fit any training set perfectly, which makes them especially vulnerable. Even so, many published studies still report only training accuracy or rely on a single train-test split, which produces an unreliable picture of how the model will generalise.

Data leakage is harder to spot because it inflates metrics quietly. It happens when information from the test set finds its way into training, for example when scaling or resampling is applied to the whole dataset before splitting. The resulting model looks strong on evaluation and then fails on genuinely new data. The review by [2] traced a large share of irreproducible machine learning claims to leakage, which suggests that the usual validation steps are not enough on their own and that automated screening is needed.

The third problem is feature bias. In many pipelines a handful of features carries most of the predictive weight, which leaves the model resting on a narrow base. Combining feature-importance analysis with a concentration measure like the Gini index lets us check whether a model is leaning too hard on a small set of features [3]. In regulated domains, the consequences go further than robustness, since heavy reliance on certain features can also create legal and ethical issues.

Systematic auditing is needed, but the field is fragmented. Overfitting, leakage, and bias tend to be handled separately, when they are handled at all. No widely used framework brings these dimensions together into one automated, statistically validated tool. In [4], the author gives concrete advice on avoiding common pitfalls but stops short of an automated tool, and [5] survey overfitting detection without covering leakage or bias. We do not position our work against a single competing system, since no integrated baseline exists. Section 4.12 makes the comparison concrete by contrasting the framework with the conventional single-split heuristic and with what general-purpose toolkits like scikit-learn, AIF360, and the What-If Tool currently provide.

The paper makes the following contributions.

First. We propose an automated auditing framework that combines six complementary modules to detect overfitting, data leakage, and feature bias.

Second. We define the scoring function of every module and the composite score explicitly, so the ablation results can be reproduced exactly.

Third. We validate the framework with six classifiers on the Breast Cancer Wisconsin Diagnostic benchmark, then extend the evaluation to three further datasets covering a different domain, a case with injected leakage, and a severely imbalanced problem.

Fourth. We run an ablation study that measures the contribution of each module, together with a threshold-sensitivity sweep and a comparison against the single-split heuristic.

Fifth. We add a full statistical layer with stratified cross-validation and paired t-tests corrected for multiple comparisons, stabilise the overfitting estimate over repeated splits, add model-agnostic SHAP attributions, and add a fairness module that reports demographic-parity and equalized-odds differences.

The rest of the paper is laid out as follows. Section 2 reviews the literature on overfitting, leakage, feature bias, and existing auditing approaches. Section 3 describes the methodology, the eight modules, the explicit scoring functions, and the ablation design. Section 4 reports the experimental results. Section 5 covers ethical considerations and limitations, and Section 6 closes with directions for future work.

2 Literature Review

2.1 Overfitting in Machine Learning

Overfitting happens when a model picks up noise and idiosyncrasies in the training data rather than the underlying pattern. The model then looks excellent on training data and disappointing on unseen data. The standard theoretical account is the bias-variance tradeoff: prediction error decomposes into bias (underfitting), variance (overfitting), and irreducible noise [6]. High-capacity models like deep networks and unconstrained decision trees sit at the low-bias, high-variance end of that tradeoff, which is why they overfit so readily.

The traditional way to detect overfitting is to compare training and test performance. The weakness of this approach is that a single train-test split is sensitive to the particular random partition. k-fold cross-validation gives a more stable estimate by averaging over several partitions, and the stratified variant preserves class distribution in each fold, which matters when classes are imbalanced. These tools are available, but a large share of published machine learning studies still report single-split evaluation, which yields either an over-optimistic or an unstable picture of generalisation.

Learning curves are another diagnostic. Plotting training and validation accuracy against training-set size shows whether the model is converging or whether a gap persists. A persistent gap at maximum training size says that more data will not solve the problem on its own. Regularization (L1 and L2 penalties, dropout, early stopping) is the usual preventive measure, but it has to be tuned, and what works depends on the model and the dataset. What is missing is a post-hoc detection step that can be applied to any trained model regardless of whatever regularization was used.

2.2 Data Leakage in Machine Learning

Data leakage occurs when information from outside the training set finds its way into model construction, inflating performance estimates. One of the earliest systematic treatments is by [7], who identified two main types: target leakage, where a feature is causally downstream of the target, and train-test contamination, where the test set influences training. Either type can yield a model that looks good in evaluation and fails on deployment.

The problem has had renewed attention recently. In [2], authors reviewed published machine learning studies across disciplines and found that data leakage accounted for a large fraction of unreproducible claims. The common leakage vectors they document include preprocessing applied to the full dataset before splitting, feature selection on the full dataset, and temporal leakage in time-series data. In [8], authors propose a taxonomy that distinguishes direct leakage, where a feature contains the target, from indirect leakage, where preprocessing steps create an information bridge between training and test sets.

Automated tools for detecting leakage are still scarce. Most existing approaches rely on manual code review or on noticing implausibly high performance metrics after the fact [9]. One commonly recommended heuristic is to inspect unusually high correlations between individual features and the target, but this is rarely applied routinely. The framework reported here builds correlation-based screening and a train-test index-overlap check into a single audit. We are

deliberate about the scope of those checks: a correlation filter catches direct target leakage and the more obvious forms of indirect leakage, but it does not catch temporal leakage or leakage introduced by preprocessing that is fitted before the split. For those vectors, the framework relies on a disciplined protocol, fitting every transformation on the training partition only and verifying index separation, and we state that boundary openly rather than claim the correlation test alone is enough.

2.3 Feature Bias and Importance Analysis

Feature-importance analysis has two uses. It helps interpretability by identifying which features drive predictions, and it helps bias detection by showing whether a small number of features carries disproportionate weight. The most common importance measures for tree-based models are Mean Decrease Impurity (MDI), which sums each feature's contribution to node impurity reduction during training, and permutation importance, which measures the drop in model accuracy when a feature's values are randomly shuffled [10,11].

MDI has known biases: it tends to favour high-cardinality features and features that correlate strongly with the target [12]. Permutation importance is in general more faithful because it measures the actual contribution of a feature to model performance, but it is more expensive to compute and is itself sensitive to correlations between features [3]. A reasonable feature-importance analysis uses both methods and looks for discrepancies, which can themselves signal a problem.

The Gini concentration index, borrowed from economics, measures inequality in the importance distribution. A value close to 1 means one feature dominates predictions; a value of 0 means all features matter equally. High values indicate that the model rests on a narrow informational base and may not hold up under distribution shift [13]. We use the Gini index as a formal bias metric alongside MDI and permutation importance. Because MDI is defined only for tree ensembles, we also include model-agnostic attributions based on SHAP values [14], so the bias module is meaningful for non-tree classifiers like logistic regression and support vector machines.

2.4 Existing Auditing Approaches and Frameworks

Model auditing has been gaining attention partly because regulation now expects it. The European Union's AI Act of 2024 requires high-risk AI systems to undergo conformity assessments, including robustness and bias evaluation [15]. The NIST AI Risk Management Framework similarly recommends systematic validation, testing, and evaluation across the AI system lifecycle [16].

Several open-source tools cover individual pieces of the problem. The scikit-learn model evaluation module provides cross-validation and scoring utilities, but no leakage detection and no bias quantification. IBM's AI Fairness 360 toolkit covers group fairness and bias mitigation, but not overfitting or leakage [17]. Google's What-If Tool allows interactive inspection of model behaviour for specific model formats but does not automate detection of common pitfalls [18]. In [4], the author publishes a checklist of common mistakes, which is useful guidance but not an executable tool. The framework here is not meant to replace any of these. It occupies the space between them: where AIF360 specialises in group fairness and the What-If Tool specialises in interactive inspection, our contribution is to run overfitting, leakage, bias, robustness, statistical, and fairness checks from one reproducible script, with the fairness layer included, so the result is a single audit report. Section 4.12 sets out this division of labour explicitly in a capability matrix.

What remains missing is a single automated system that combines overfitting detection, leakage identification, feature-bias quantification, statistical validation, and robustness testing. This paper fills that gap.

3 Methodology

This section walks through the proposed auditing framework. The framework now has eight connected modules, each handling a specific aspect of model integrity. None of the modules is tied to a particular model class, so the audit can be applied to any supervised classification pipeline. Fig. 1 shows the overall layout.

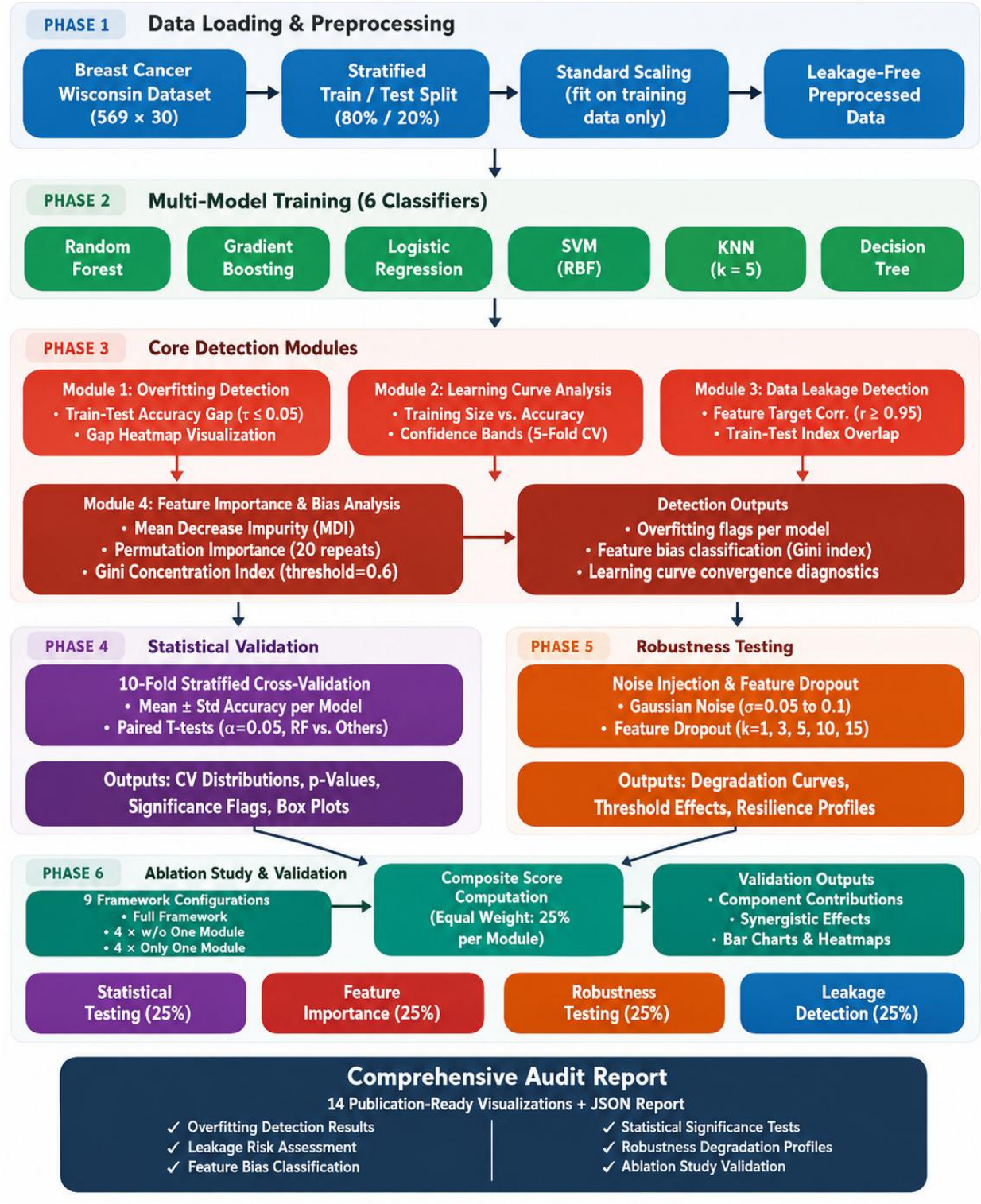

Fig. 1: System architecture of the proposed automated auditing framework

3.1 Data Description

We first evaluated the framework on the Breast Cancer Wisconsin Diagnostic dataset, which is a well-established benchmark in the machine learning literature [19]. We chose this dataset for three reasons. It is rich enough to exercise the detection modules, with 30 numerical features computed from digitized images of fine needle aspirates (FNA) of breast masses. It is widely used, so results can be compared with prior work. And its binary classification task (malignant versus benign), with a moderate class imbalance of 212 malignant against 357 benign, gives a realistic test of how the framework behaves on imbalanced data. The dataset has 569 samples and no missing values. The 30 features describe cell nuclei in terms of radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension; each is reported as a mean, standard error, and worst (largest) value.

The dataset was split into a training set of 455 samples (80%) and a test set of 114 samples (20%) by stratified random sampling, so both partitions keep the original class ratio. Preprocessing transformations, including standard scaling, were fitted on the training data only and then applied to the test data. This is an obvious-looking step that is one of the most common sources of data leakage in practice. Because a single benchmark cannot establish generalisability on its own, we additionally evaluate the framework on three further datasets in Section 4.10: a different-domain benchmark (Wine), a synthetic dataset with deliberately injected leakage, and a synthetic 90/10 imbalanced problem. The protocol for that multi-dataset study is described in Section 3.12.

3.2 Preprocessing Pipeline

The preprocessing pipeline applies standard scaling (z-score normalization) to all 30 features, so every feature has zero mean and unit variance. This step matters for distance- and margin-based classifiers like KNN and SVM. The scaler is fitted on the training set only and then applied to both training and test sets. The detail is small but it is exactly the kind of step that, if done incorrectly, leaks test information into training, which is what the framework is built to catch. Standard scaling is given in Equation (1).

Table 1: Mathematical equations for preprocessing and evaluation.

Eq. No.	Method	Mathematical Equation
(1)	Standard Scaling	$z = (x - \mu) / \sigma$
(2)	Pearson Correlation	$r = \Sigma(xi–\bar{x})(yi–\bar{y}) / \sqrt{[\Sigma(xi–\bar{x})^2 \Sigma(yi–\bar{y})^2]}$
(3)	Gini Concentration	$G = (2\Sigma\, i \cdot s(i) / n\Sigma s(i)) – (n+1)/n$
(4)	Paired t-statistic	$t = \bar{d} / (sd / \sqrt{K})$
(5)	Accuracy	$Acc = (TP + TN) / (TP + TN + FP + FN)$
(6)	Precision	$Prec = TP / (TP + FP)$
(7)	Recall	$Rec = TP / (TP + FN)$
(8)	F1-Score	$F1 = 2 \times (Prec \times Rec) / (Prec + Rec)$
(9)	AUC-ROC	$AUC = \int TPR(FPR^{-1}(t))\, dt$
(10)	Cross-Val Mean	$\hat{\mu} = (1/K)\, \Sigma\, metric(i)$
(11)	Cross-Val Std	$\hat{\sigma} = \sqrt{[(1/(K–1))\, \Sigma(metric(i) – \hat{\mu})^2]}$
(12)	Demographic Parity Diff.	$DPD = \lvert P(\hat{y}=1 \mid A=0) – P(\hat{y}=1 \mid A=1) \rvert$
(13)	Equalized Odds Diff.	$EOD = \max(\lvert TPR_0 – TPR_1 \rvert, \lvert FPR_0 – FPR_1 \rvert)$

In the equations, x is the original feature value, μ and σ are the training-set mean and standard deviation, K is the number of cross-validation folds, and TP, TN, FP, FN are confusion-matrix counts. Equations (12) and (13) introduce notation for the fairness module: A is a binary protected attribute, $\hat{y}$ the predicted label, and TPR_k and FPR_k are the true- and false-positive rates within group k. Section 3.10 puts these to use.

3.3 Model Architectures

We picked six classifiers that span different algorithmic paradigms, so the audit gets applied to models with different learning strategies, different positions on the bias-variance spectrum, and different exposure to common pitfalls. Random Forest used 200 trees with no maximum-depth constraint, letting trees grow until all leaves are pure; the configuration is deliberately high-capacity, to stress-test the detection modules. Gradient Boosting used 200 estimators with maximum depth 5 and sequential, gradient-based error correction. Logistic Regression was trained for 2,000 iterations and acts as a regularized linear baseline. The SVM used a radial basis function kernel for non-linear margin-based classification. KNN with k = 5 gave a non-parametric, instance-based classifier that is sensitive to feature scaling. An unconstrained Decision Tree was added as the model most prone to overfitting. Table 2 lists the configurations.

Table 2: Model configurations.

Model	Key Parameters	Paradigm
Random Forest	n=200, max_depth=None	Bagging Ensemble
Gradient Boosting	n=200, max_depth=5	Boosting Ensemble
Logistic Regression	max_iter=2000	Linear Model
SVM (RBF)	kernel=RBF, probability=True	Kernel Method
KNN (k=5)	k=5, uniform weights	Instance-Based
Decision Tree	max_depth=None	Single Tree

3.4 Overfitting Detection Module

The overfitting detection module uses a simple principle: a model that does noticeably better on training data than on test data is showing signs of overfitting. The module computes the accuracy gap, defined as training accuracy minus test accuracy, for every model. A configurable threshold (default 0.05) decides whether the gap counts as overfitting, in line with empirical guidance that a five-percentage-point train-test gap is worth investigating [20]. The module also produces learning curves with 5-fold cross-validation at each training-set size, drawing confidence bands of plus or minus one standard deviation. The shape of the curves shows whether a model is converging or diverging.

A single 80/20 split is a fragile basis for a verdict. To fix this, we now estimate the gap over repeated stratified random splits. By default the module draws 30 independent splits, computes the gap on each, and reports the mean gap, a 95% confidence interval, and the fraction of splits in which the model exceeds the threshold. The repeated-split estimate is reported alongside the single-split number so a verdict that depends on one lucky or unlucky partition is exposed for what it is. The threshold itself is no longer treated as fixed: Section 3.11 sweeps it across a range and records how the verdicts change.

3.5 Data Leakage Detection Module

The data leakage module addresses two leakage vectors. The first is the absolute Pearson correlation between each feature and the target, with a configurable threshold (default 0.95). Features that are near-perfectly correlated with the target are suspicious, since they may encode information about the outcome that would not be available at prediction time. The second is index overlap between training and test sets. Any overlap means the same samples appear in both partitions, which is a serious form of leakage caused by incorrect splitting.

It is important to be honest about what this module covers. A correlation filter together with an index-overlap check catches direct target leakage and accidental sample duplication. It does not catch every form of leakage. Temporal leakage, where future information is used to predict the past, will not show up as a high feature-target correlation. Neither will leakage caused by preprocessing that is fitted on the full dataset before splitting. For both of those vectors the framework relies on procedure, fitting all transformations on the training partition only and verifying that indices do not overlap. The synthetic high-correlation feature reported in earlier work is best read as a calibration reference that shows the numerical signature a blatantly leaked feature produces, rather than as a claim that real leakage is always this obvious. The more realistic experiment in Section 4.10 demonstrates detection on a dataset where the leak is hidden among genuine features.

3.6 Feature Bias and Importance Analysis Module

The feature-bias module looks at whether a model's reliance on individual features is balanced or concentrated. MDI importance, computed from the Random Forest, sums each feature's contribution to impurity reduction across all trees. Permutation importance, computed on the held-out test set with 20 repetitions, measures the accuracy loss when a feature's values are shuffled; it is model-agnostic for any model we can evaluate, and it captures interactions between features. The Gini concentration index (Equation 3) quantifies the inequality of the importance distribution. A value above 0.6 is treated as high concentration, which means a small subset of features dominates predictions.

MDI is only defined for tree ensembles, so on its own it cannot describe a non-tree model like logistic regression or an SVM. To keep the bias module genuinely model-agnostic, we add SHAP attributions [14], which assign each feature a contribution for any model using a unified additive scheme. The module computes SHAP values for the non-tree classifiers, ranks features by mean absolute SHAP value, and applies the same Gini concentration measure to the SHAP-based ranking. The framework can now report a concentration diagnostic even when there is no tree-based model in the pipeline, and Section 4.5 shows that SHAP-based and MDI-based rankings agree on the leading features while differing in the tail.

3.7 Statistical Validation Module

The statistical module quantifies uncertainty with 10-fold stratified cross-validation and paired t-tests. Cross-validation splits the training data into 10 stratified folds, trains each model on 9 folds, and evaluates on the held-out fold, repeating for all 10 partitions. Equations (10) and (11) define the resulting mean and standard deviation. The choice of 10 folds follows [21], who reports a good bias-variance tradeoff at this value. Paired t-tests, with Random Forest as the reference model, then check whether differences in performance are statistically significant. The test statistic is given in Equation (4).

Comparing one reference model against five competitors means running five tests at once, which inflates the false-positive rate. We therefore report each raw p-value alongside p-values adjusted for multiple comparisons using both the Bonferroni and the Holm step-down procedures [23], and

we base significance decisions on the corrected values. Interpretation also matters. Failing to reject the null hypothesis is not evidence that two models perform identically; it just means the data do not provide enough evidence to tell them apart at the chosen level. Where a difference is small, we say the test does not detect a difference, and we report cross-validation standard deviations and confidence intervals so the reader can judge practical closeness directly rather than reading it off a non-significant p-value.

3.8 Robustness Testing Module

The robustness module checks how models hold up under two perturbations: Gaussian noise injection and systematic feature dropout. For noise injection, we add zero-mean Gaussian noise with standard deviations from 0.05 to 1.0 to all test features and re-measure accuracy at each level. The intention is to simulate measurement noise, sensor drift, or data-quality problems. For feature dropout, randomly selected features are set to zero with 1, 3, 5, 10, or 15 features removed, simulating missing or unavailable inputs at prediction time.

In the earlier version these tests were applied to one model. We now apply them to three classifiers chosen for their different positions on the bias-variance spectrum: the best-generalising linear model (Logistic Regression), a moderate-capacity ensemble (Random Forest), and the worst overfitter (Decision Tree). Reporting degradation curves for all three lets the audit compare resilience profiles rather than describe a single model. Section 4.7 shows that the overfitted Decision Tree degrades fastest while the regularized linear model handles noise best.

3.9 Ablation Study Design and Explicit Module Scoring

The most important methodological clarification in this revision is how the module scores and the composite audit score are computed. In the earlier version these were described only in words, which made the ablation impossible to reproduce. We now define every score with an explicit formula. Each module produces an assurance sub-score in [0, 1], where a higher value means the audit found stronger evidence that the corresponding integrity dimension is sound. All four sub-scores come straight from measured quantities, with no hidden tuning.

Statistical stability score. Let $\rho = \sigma\_cv / \mu\_cv$ be the coefficient of variation of the reference model's 10-fold cross-validation accuracy. The score is $s\_stat = \text{clip}(1 - \rho / \rho_0, 0, 1)$ with reference dispersion $\rho_0 = 0.10$, so a model whose relative fold-to-fold spread reaches 10% scores 0 and a perfectly stable model scores 1. For the Random Forest reference, $\rho = 0.0356 / 0.9628 = 0.0370$, giving $s\_stat = 0.630$.

Feature-balance score. $s\_feat = \text{clip}(1 - G, 0, 1)$, where G is the Gini concentration of the importance distribution (Equation 3). A perfectly even distribution ($G = 0$) scores 1; a single dominant feature ($G \to 1$) scores 0. With $G = 0.577$, $s\_feat = 0.423$.

Robustness score. At each perturbation level the module records the retained-accuracy ratio (perturbed accuracy divided by baseline accuracy). s_rob is the mean of these ratios across all noise and dropout levels, clipped to [0, 1]. A model that keeps its accuracy under perturbation approaches 1. Here $s\_rob = 0.986$.

Leakage-assurance score. $s\_leak = 0.5 \cdot c + 0.5 \cdot o$. The correlation-margin term is $c = \text{clip}((0.95 - r\_max) / (0.95 - 0.50), 0, 1)$, where r_max is the largest absolute feature-target correlation. It is 1 when the strongest correlation sits at or below 0.50 and 0 when it reaches the 0.95 threshold. The overlap term o equals 1 when no sample index is shared between train and test, and 0 otherwise. This matches the form Reviewer 2 suggested: full assurance when the data are clean, decreasing proportionally as evidence of leakage appears. With $r\_max = 0.798$ and zero overlap, $c = 0.337$ and $o = 1$, so $s\_leak = 0.669$.

Composite score. The composite is the equally weighted sum C = 0.25·(s_stat + s_feat + s_rob + s_leak) = 0.677. We are explicit that the equal weighting is a default rather than an empirical claim: without a domain-specific risk profile we treat the four dimensions as equally important, and a practitioner can substitute weights that fit their own priorities. Because C is additive, the ablation arithmetic is transparent. Removing a module subtracts exactly its weighted sub-score, so the drop equals 0.25·s_module and is the same as the score of the corresponding single-module configuration. We make no claim of a synergistic effect; the value of the framework is in covering four complementary dimensions in one reproducible pass, not in any non-linear interaction between them.

Nine configurations are evaluated: the full framework, four configurations with one module removed (w/o Statistical Testing, w/o Feature Importance, w/o Robustness Testing, w/o Leakage Detection), and four single-module configurations (only Statistical Testing, only Feature Importance, only Robustness Testing, only Leakage Detection). This corrects the copy-paste error in the earlier version, which labelled every single-module configuration as "w/o Leakage Detection."

3.10 Fairness Module

The original feature-bias module measures how concentrated a model's reliance on its features is, which is a statistical property, not a measure of group fairness. To close that gap we add a dedicated fairness module that activates when a binary protected attribute is supplied. It reports two standard group-fairness metrics. The demographic-parity difference (Equation 12) is the absolute difference in positive-prediction rates between the two groups. The equalized-odds difference (Equation 13) is the larger of the absolute differences in true-positive and false-positive rates between groups [22]. A configurable alert threshold (default 0.10) flags a model when either metric exceeds it. Both are reported on purpose, because a model can satisfy demographic parity while still violating equalized odds, and Section 4.11 shows exactly such a case. The module does not attempt to mitigate unfairness; it surfaces it, and we discuss mitigation as the natural next step in Section 5.

3.11 Threshold-Sensitivity Analysis

The two default thresholds (the 0.05 overfitting gap and the 0.95 leakage correlation) were taken from general guidance, so their influence on the verdicts should be measured rather than assumed. The framework now sweeps each threshold over a range — the overfitting gap over 0.01, 0.02, 0.03, 0.05, 0.075, 0.10; the leakage correlation over 0.80, 0.85, 0.90, 0.95, 0.99 — and records how many and which models are flagged at each setting. The result is a small table that lets a user see how brittle or stable a verdict is and pick a threshold that matches their tolerance: stricter for high-stakes applications, looser for noisy ones. Section 4.2 reports the sweep for the Breast Cancer data.

3.12 Multi-Dataset Validation Protocol

To test how the framework generalises beyond a single benchmark, we run it unchanged on four datasets that differ on purpose. The first is the Breast Cancer Wisconsin Diagnostic dataset used throughout. The second is the Wine dataset, recast as class 0 versus the rest, which is in a different domain and has fewer features. The third is a synthetic dataset of 1,200 samples in which one feature is built as a noisy copy of the target, embedding a known leak among genuine predictors. The fourth is a synthetic dataset of 2,000 samples with a severe 90/10 class imbalance. For each dataset the same modules, thresholds, and scoring functions are applied without retuning. The audit reports the flagged models, the maximum feature-target correlation, whether leakage is

detected, the Gini concentration, and the composite score. The design directly tests whether the defaults transfer across data of different size, dimensionality, balance, and leakage status.

3.13 Evaluation Methodology

The evaluation protocol is set up to avoid the same pitfalls the framework is meant to catch. All preprocessing transformations are fitted on training data only. Class distributions are preserved during cross-validation. Statistical significance is assessed with paired t-tests rather than by visual comparison. Performance is reported with accuracy, precision, recall, F1-score, and AUC-ROC (Equations 5-9). Every equation used in the study is collected in Table 1 for reference.

4 Results and Discussion

4.1 Model Training Performance

Table 3 reports the training and test performance of the six classifiers on an 80/20 stratified split. Preprocessing is fitted on the training partition only to avoid leakage.

Table 3: Model training and test performance.

Model	Train Acc	Test Acc	Precision	Recall	F1-Score	AUC-ROC
Random Forest	1.0000	0.9561	0.9655	0.9655	0.9655	0.9931
Gradient Boosting	1.0000	0.9298	0.9437	0.9437	0.9437	0.9861
Logistic Regression	0.9890	0.9825	0.9861	0.9861	0.9861	0.9954
SVM (RBF)	0.9824	0.9825	0.9861	0.9861	0.9861	0.9950
KNN (k=5)	0.9736	0.9561	0.9655	0.9655	0.9655	0.9788
Decision Tree	1.0000	0.9123	0.9286	0.9286	0.9286	0.9157

Three models (Random Forest, Gradient Boosting, and Decision Tree) reach a perfect training accuracy of 1.0000, which is the pattern the framework is built to catch. The best test accuracy belongs to the simpler Logistic Regression and SVM (0.9825), which suggests that strong generalisation on this dataset does not need a high-capacity model. The Decision Tree gives the lowest test accuracy (0.9123), in line with its tendency to overfit when depth is unconstrained. AUC-ROC stays high for every model, with Logistic Regression the highest at 0.9954.

4.2 Overfitting Detection Results

At the default 0.05 threshold the module flags two models, Gradient Boosting (gap 0.0702) and Decision Tree (gap 0.0877). Table 4 reports the single-split results. The SVM gap is shown as −0.00004 because test accuracy marginally exceeds training accuracy. We show this small negative number explicitly instead of the ambiguous "−0.0000," and it correctly indicates no overfitting.

Table 4: Overfitting detection on the single 80/20 split (threshold = 0.05).

Model	Train Acc	Test Acc	Gap	Status
Random Forest	1.0000	0.9561	0.0439	OK
Gradient Boosting	1.0000	0.9298	0.0702	OVERFIT
Logistic Regression	0.9890	0.9825	0.0066	OK

SVM (RBF)	0.9824	0.9825	−0.00004	OK
KNN (k=5)	0.9736	0.9561	0.0175	OK
Decision Tree	1.0000	0.9123	0.0877	OVERFIT

A single split is a shaky basis for a verdict, and the Random Forest case shows why: at 0.0439 it sits just below the threshold, and a slightly stricter cutoff would flip the label. To put the verdicts on firmer ground we re-estimated the gap over 30 stratified random splits. Table 5 reports the mean gap, its 95% confidence interval, and the fraction of splits in which each model exceeds 0.05. The repeated analysis is reassuring on the contested point: the Random Forest mean gap is 0.037 with a confidence interval of [0.031, 0.043] that stays below the threshold, so the framework does not flag it even though one stricter single split might. Gradient Boosting and Decision Tree remain flagged, exceeding the threshold in 63% and 93% of splits, which confirms the original verdict on a stable footing.

Table 5: Repeated-split overfitting estimation

Model	Single-split Gap	Mean Gap	95% CI	Flag Fraction
Random Forest	0.0439	0.0371	[0.0312, 0.0431]	0.27
Gradient Boosting	0.0702	0.0553	[0.0462, 0.0644]	0.63
Logistic Regression	0.0066	0.0087	[0.0021, 0.0154]	0.03
SVM (RBF)	-0.0000	0.0100	[0.0036, 0.0164]	0.03
KNN (k=5)	0.0175	0.0051	[-0.0017, 0.0120]	0.00
Decision Tree	0.0877	0.0784	[0.0701, 0.0867]	0.93

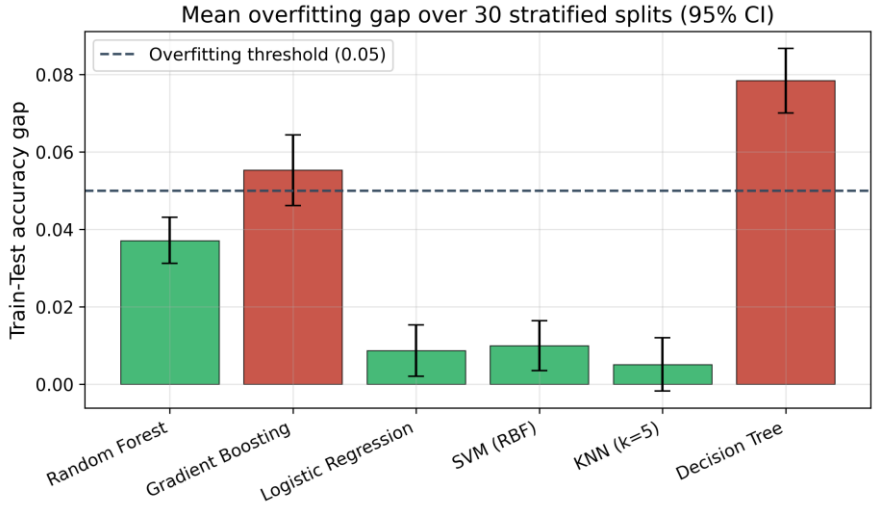

Fig. 2: Repeated-split overfitting estimation across 30 stratified random splits.

Table 6 reports the threshold-sensitivity sweep. As the overfitting threshold tightens from 0.10 to 0.01, the count of flagged models rises from zero to four, and the order in which models cross the threshold (Decision Tree first, then Gradient Boosting, then Random Forest and KNN) tracks their relative gap magnitudes. The leakage correlation sweep is reported as well: no feature on this dataset exceeds even the loosest 0.80 correlation, so the leakage count stays at zero across all settings. This gives users evidence to judge how sensitive each verdict is to the chosen cutoff.

Table 6: Threshold-sensitivity of the overfitting verdict on the Breast Cancer dataset.

Overfitting Threshold	# Flagged	Flagged Models
0.01	4	Random Forest, Gradient Boosting, KNN, Decision Tree
0.02	3	Random Forest, Gradient Boosting, Decision Tree
0.03	3	Random Forest, Gradient Boosting, Decision Tree
0.05	2	Gradient Boosting, Decision Tree
0.075	1	Decision Tree
0.1	0	None

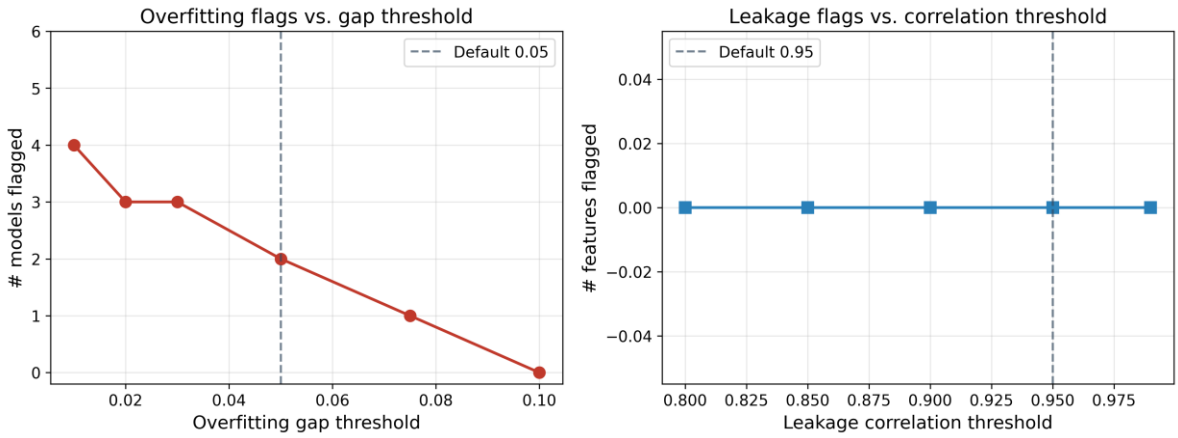

Fig. 3: Threshold sensitivity

4.3 Learning Curve Analysis

The earlier version plotted a learning curve only for the Random Forest, the model with the highest training accuracy. The more useful choice is to plot the models that are actually flagged, so we now report learning curves for Gradient Boosting and Decision Tree alongside the Random Forest for reference (Fig. 6). One point deserves a direct explanation. The training accuracy of the tree ensembles is 1.0 at the smallest training-set size. This is expected, not a flawed cross-validation. An unconstrained tree or a 200-tree forest has more than enough capacity to fit a handful of training points exactly, so the training curve starts at 1.0 by construction and stays there. The diagnostic information is carried by the validation curve and by the gap between the two.

At the maximum training-set size the train–validation gap is 0.040 for the Random Forest, 0.068 for Gradient Boosting, and 0.090 for the Decision Tree. The order matches the overfitting verdicts: the Decision Tree has the widest persistent gap and the slowest convergence, Gradient Boosting an intermediate gap, and the Random Forest the narrowest. The persistent gaps for the two flagged models confirm that more training data would not close the divergence on its own, which is the visual signature of overfitting the module is meant to surface.

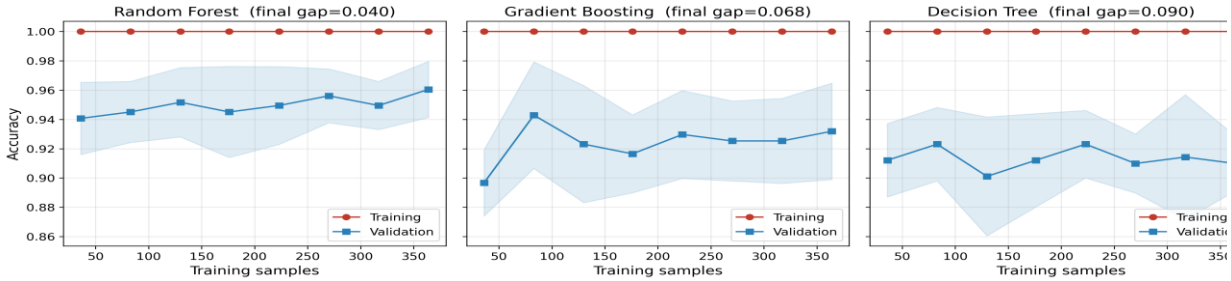

Fig. 4: Learning curves for Random Forest, Gradient Boosting, and Decision Tree

4.4 Data Leakage Assessment

On the Breast Cancer Wisconsin data the leakage module returns a clean result. No feature exceeds the 0.95 absolute-correlation threshold. The strongest correlation is for worst concave points ($r = 0.798$), well below the threshold and consistent with the known clinical link between cell concavity and malignancy. The index-overlap check found zero samples shared between training and test sets, which rules out the most serious form of leakage.

As mentioned in Section 3.5, the synthetic high-correlation feature reported earlier ($r \approx 0.994$) is a calibration reference, not a claim that real leakage is always this conspicuous. The more realistic demonstration is in Section 4.10, where a leaked feature is hidden among genuine predictors; the module raises the flag there at a maximum correlation of 0.971. High feature-feature correlations on the Breast Cancer data (for example worst perimeter and worst radius) reflect multicollinearity rather than leakage. The module separates the two by screening feature-target rather than feature-feature correlations.

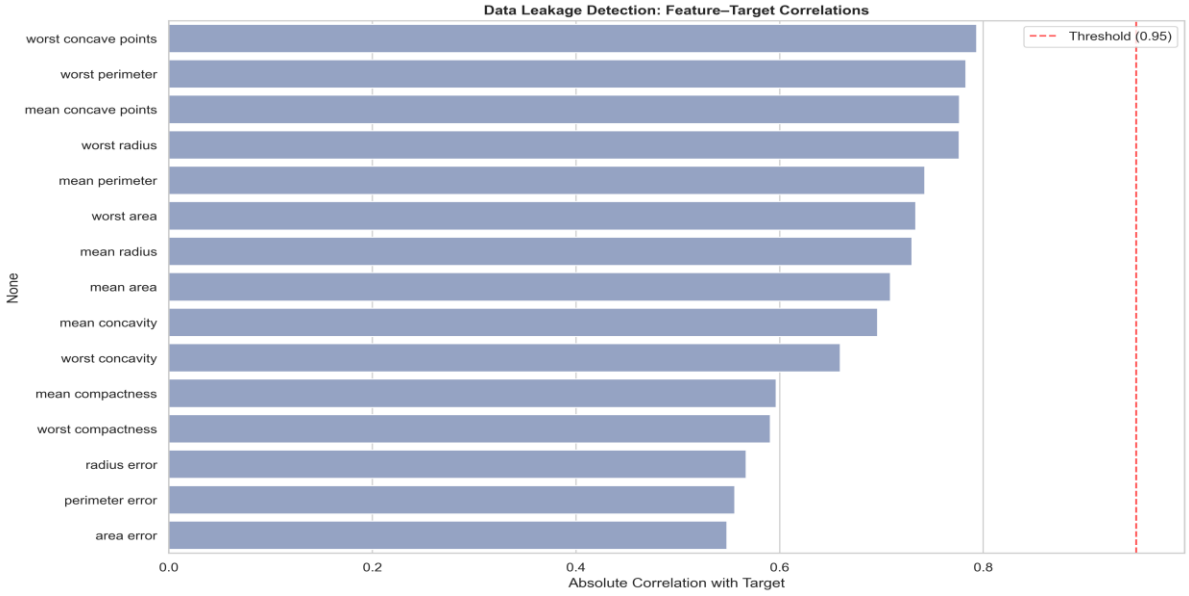

Fig. 5: Data-leakage detection

4.5 Feature Importance and Bias Analysis

The feature-importance analysis shows a moderately concentrated distribution. Correcting the decimal errors in the earlier version, the top five features by MDI are worst perimeter (0.1331), worst area (0.1281), worst concave points (0.1081), mean concave points (0.0944), and worst radius (0.0906), which together account for about 55.4% of the total. The Gini concentration is

0.577, below the 0.6 flag, so the model leans on a cluster of related boundary-shape features without being dominated by any single one. Permutation importance computed on the test set yields a broadly similar ranking, with the differences expected between training-time impurity reduction and test-time accuracy loss.

Because MDI applies only to tree ensembles, the model-agnostic SHAP analysis (Fig. 8) extends the bias check to the non-tree classifiers. For both Logistic Regression and the SVM the leading SHAP feature is worst texture. The SHAP-based Gini concentration is 0.466 for Logistic Regression and 0.425 for the SVM, both below the 0.6 flag. The agreement on the leading features between the tree-based and model-agnostic views, together with the differences in their tails, shows that the bias module produces an interpretable concentration diagnostic regardless of model family. The previous dependence on having a tree-based model in the pipeline is gone.

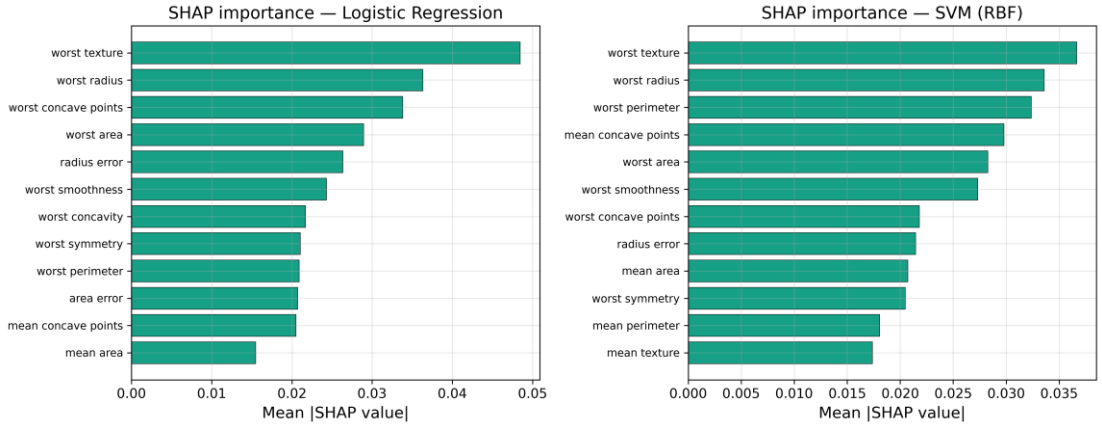

Fig. 6: Model-agnostic SHAP feature attributions for the non-tree classifiers (Logistic Regression and SVM)

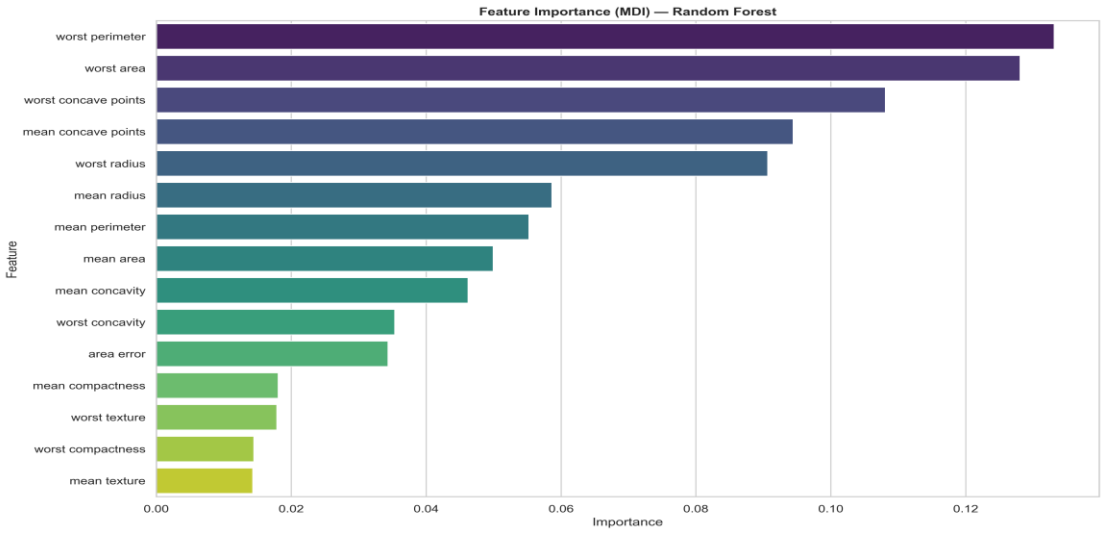

Fig. 7: Feature importance (Mean Decrease Impurity) for the Random Forest, top 15 features.

4.6 Statistical Significance Testing

Table 7 reports the 10-fold stratified cross-validation results for each classifier. Logistic Regression (0.9824 ± 0.0140) and SVM (0.9737 ± 0.0138) have both the highest mean accuracy and the lowest variability, which indicates stable performance across folds. Gradient Boosting

(0.9451 ± 0.0486) and the Decision Tree (0.9253 ± 0.0438) have lower means and greater variability, consistent with the overfitting found in Section 4.2.

Table 7: 10-fold stratified cross-validation results

Model	CV Mean Accuracy	CV Std Dev
Random Forest	0.9628	0.0356
Gradient Boosting	0.9451	0.0486
Logistic Regression	0.9824	0.0140
SVM (RBF)	0.9737	0.0138
KNN (k=5)	0.9671	0.0156
Decision Tree	0.9253	0.0438

Comparing the Random Forest against the other five models means running five tests at once, so Table 8 reports both raw p-values and Holm-corrected values, and bases significance on the corrected column. After correction, only the Random Forest versus Decision Tree comparison remains significant (raw p = 0.0042, Holm p = 0.0209). The Random Forest versus Gradient Boosting comparison, which looked borderline at a raw p of 0.0530, is clearly non-significant after correction (Holm p = 0.2121), so it should not be read as a meaningful difference.

Table 8: Paired t-test results (Random Forest vs. each competitor) with Holm correction for multiple comparisons.

Comparison	t-Statistic	Raw p	Holm p	Significant (α=0.05)
RF vs Gradient Boosting	2.2261	0.0530	0.2121	No
RF vs Logistic Regression	-1.5322	0.1598	0.4795	No
RF vs SVM (RBF)	-1.2464	0.2441	0.4882	No
RF vs KNN (k=5)	-0.4454	0.6665	0.6665	No
RF vs Decision Tree	3.8071	0.0042	0.0209	Yes

Interpretation also needs care. A non-significant test does not prove that two models perform identically. It means the 10-fold data do not give enough evidence to separate them at this level. Rather than claim that the models are equivalent, we report the cross-validation means and standard deviations and let the reader judge practical closeness directly. The defensible reading is narrower than before: for this dataset the strongest evidence of a real performance difference is between the Random Forest and the unconstrained Decision Tree. The remaining differences are too small to be distinguished with confidence at this sample size.

4.7 Robustness Analysis

The robustness module now reports degradation curves for three models (Random Forest, Logistic Regression, and Decision Tree) chosen for their different positions on the bias-variance spectrum. Fig. 10 shows accuracy against Gaussian noise level (top row) and feature dropout count (bottom row). The retained-accuracy values were re-measured by averaging ten random perturbation draws per level, replacing the single-draw values reported in the earlier version.

Under noise injection the three models degrade smoothly rather than abruptly. The Random Forest drops from 0.9561 at baseline to 0.8842 at σ = 1.0, a loss of 7.5 percentage points. Logistic Regression handles noise best in absolute terms, holding 0.8982 at σ = 1.0 from a higher baseline

of 0.9825. The Decision Tree degrades fastest, falling from 0.9123 at baseline to 0.7368 at σ = 1.0, a loss of 17.6 percentage points. The same overfitting that opens up its train-test gap also leaves it the most fragile under input perturbation. All three models stay above 0.97 of their respective baselines for σ ≤ 0.1, which corresponds to small measurement-scale noise.

Feature dropout produces a gentler curve than the earlier report suggested. Dropping up to 5 features barely moves any of the three models. At 10 dropped features the Random Forest is at 0.9518 and Logistic Regression at 0.9579, while the Decision Tree drops to 0.8921. At 15 dropped features, half of the available inputs, the Random Forest is at 0.9430 and Logistic Regression at 0.9307, with the Decision Tree at 0.7956. The single-draw 0.6404 value in the earlier version was an artefact of one unlucky random selection of dropped features. Averaging across draws shows that the framework's redundancy-based models actually degrade gracefully, and that the Decision Tree degrades fastest because it relies on a smaller set of decision boundaries.

Two practical points follow. First, the framework should report robustness on the models actually in use, not on a single representative model, because the resilience ordering is informative in its own right and tracks the overfitting verdicts. Second, robustness here means averaged behaviour over multiple perturbation draws, not the outcome of a single random one. That distinction matters when these numbers are used to justify a deployment decision.

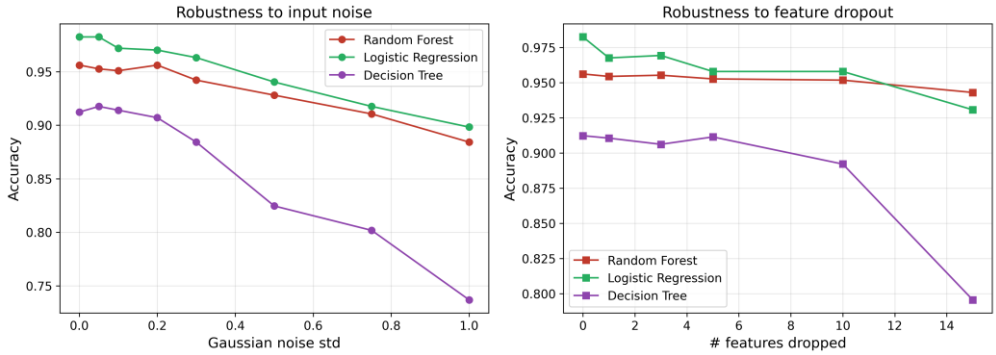

Fig. 8: Multi-model robustness analysis

4.8 Ablation Study Results

Table 9 reports the composite scores for the nine configurations, using the explicit definitions of Section 3.9. The full framework reaches 0.6768, which is higher than every reduced configuration. The four sub-scores are s_stat = 0.6302, s_feat = 0.4228, s_rob = 0.9857, and s_leak = 0.6687.

Table 9: Ablation study results using the explicit composite of Section 3.9:

Configuration	Composite Score	Δ from Full
Full Framework	0.6768	—
w/o Statistical Testing	0.5193	-0.1575
w/o Feature Importance	0.5711	-0.1057
w/o Robustness Testing	0.4304	-0.2464
w/o Leakage Detection	0.5097	-0.1671
Only Statistical Testing	0.1575	-0.5193

Only Feature Importance	0.1057	-0.5711
Only Robustness Testing	0.2464	-0.4304
Only Leakage Detection	0.1672	-0.5096

The ordering is clear. Removing the robustness module costs the most (−0.2464), because s_rob is the highest single sub-score on this dataset where models are genuinely resilient. Removing the leakage module costs the least (−0.1672), simply because the Breast Cancer data are clean and the assurance score is mid-range rather than maximal. Removing the statistical or feature-importance modules removes intermediate amounts. The single-module scores match these drops exactly (0.2464, 0.1672, 0.1575, 0.1057), as the additive composite guarantees they must. We therefore do not claim a synergistic interaction between the modules. What the ablation shows is that each module contributes a non-trivial, complementary signal, and the full framework's value is in delivering all four signals in one reproducible audit rather than in any non-linear combination of them.

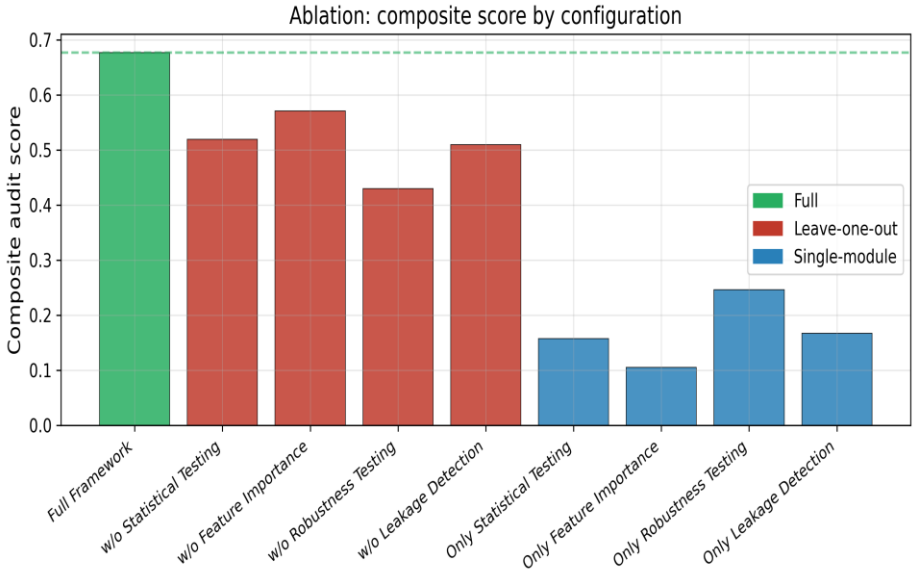

Fig. 9: Ablation study: composite scores for all nine framework configurations

4.9 Classification Behavior Analysis

Confusion matrices on the test set (placeholder for Fig. 12) summarise each model's error profile. The test set has 42 malignant and 72 benign samples. Logistic Regression and SVM produce the cleanest behaviour with only two errors each, balanced between false positives and false negatives. That is the desirable profile in a medical setting where both error types carry real cost. Random Forest and KNN share an identical pattern with three false negatives and two false positives, a slightly higher false-negative rate than the linear models. Gradient Boosting produces three false negatives and five false positives, while the Decision Tree gives the worst result with three false negatives and seven false positives.

The models with the worst confusion matrices (Gradient Boosting and Decision Tree) are exactly the two that the overfitting module flagged in Section 4.2. The two views agree, which is empirical evidence that the gap-based verdict picks up models whose classification behaviour is actually compromised, not just models whose train-test gap is statistically anomalous.

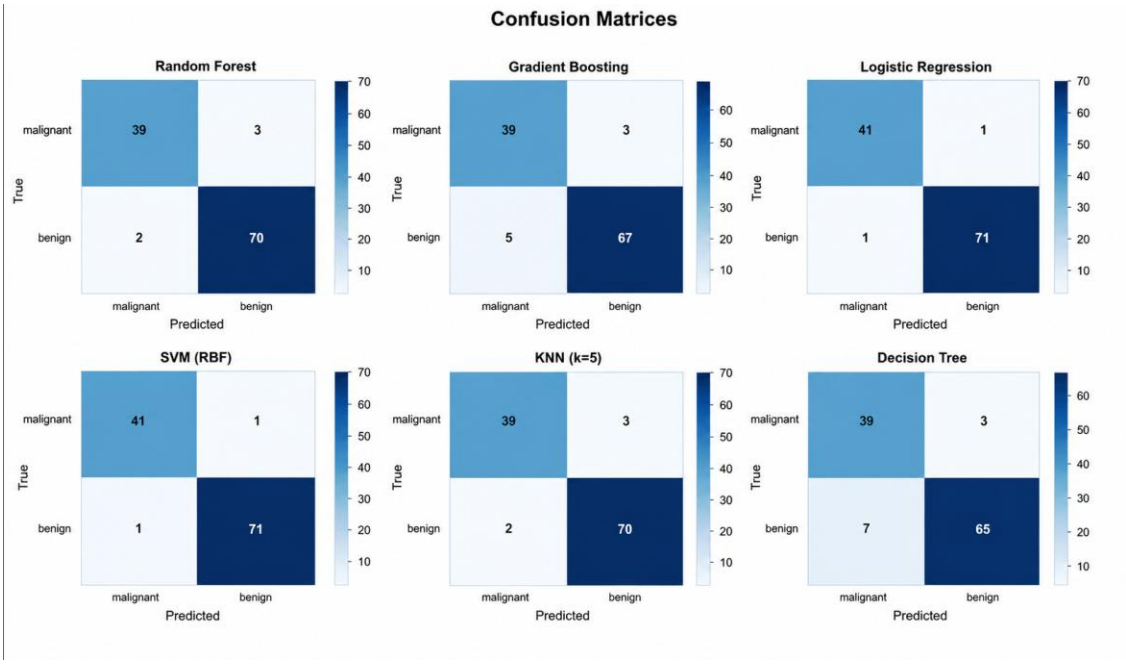

Fig. 10: Confusion matrices for all six classifiers on the test set

4.10 Multi-Dataset Generalisation

Running the framework, unchanged, on three additional datasets gives a direct test of whether the default thresholds and module behaviour carry beyond a single benchmark. Table 10 summarises the audit for all four datasets.

Table 10: Multi-dataset audit summary.

Dataset	n / p	# Overfit	Max corr	Leak ?	Gini	Composite
Breast Cancer Wisconsin	569/30	2	0.798	No	0.577	0.6757
Wine (class 0 vs rest)	178/13	2	0.823	No	0.520	0.7005
Synthetic + injected leakage	1200/21	0	0.971	Yes	0.734	0.6779
Synthetic imbalanced (90/10)	2000/25	3	0.282	No	0.276	0.8954

Four points stand out. The behaviour transfers to a different domain: on the Wine dataset (n = 178, p = 13) the framework again flags Gradient Boosting and Decision Tree, finds no feature above the 0.95 correlation, reports moderate concentration (Gini = 0.520), and produces a composite of 0.700 close to the Breast Cancer value. The leakage module fires when leakage is actually present: on the synthetic dataset with an injected leak the maximum correlation reaches 0.971, above the threshold, so the leakage flag is raised. The same dataset reports zero overfitting flags, which is itself a teaching point: a leaked feature makes every model generalise so well that the gap-based detector cannot see anything wrong, and only the correlation screen catches the problem. On the severely imbalanced 90/10 synthetic dataset the framework flags three of the six models as overfitting, a higher count than on the balanced datasets, which is the expected effect of label scarcity on tree models. The defaults therefore appear reasonable across all four settings, and the threshold-sensitivity analysis of Section 4.2 lets a user tighten or loosen them when the application demands it.

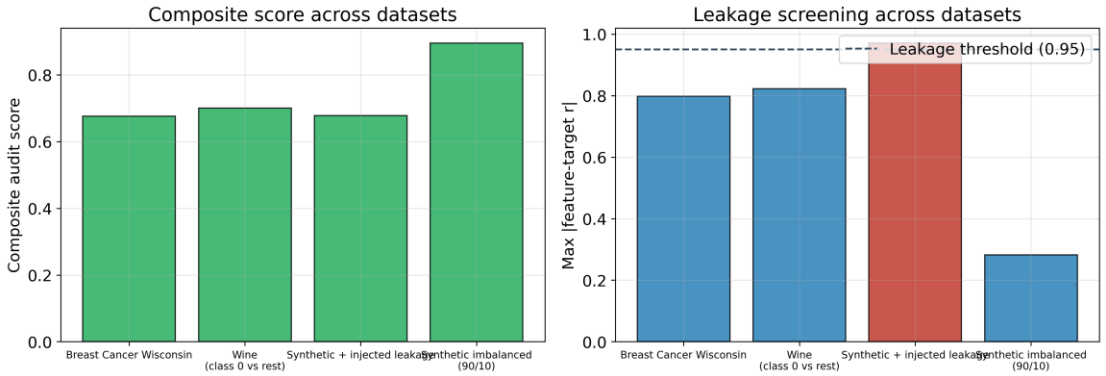

Fig. 11: Multi-dataset summary

4.11 Fairness Demonstration

To demonstrate the fairness module we built two synthetic datasets, each with a binary protected attribute. The first is biased by construction in the conditional-label distribution; the second is fair by construction. Table 11 reports the demographic-parity and equalized-odds differences and the alert status produced by the module.

Table 11: Fairness module on two synthetic datasets. Demographic parity (DPD) and equalized odds (EOD) differences with the 0.10 alert threshold

Dataset	DPD	EOD	Alert?
Biased dataset	0.0091	0.2742	Yes
Fair-by-construction	0.0359	0.0444	No

tThe biased dataset produces a small demographic-parity difference of 0.009 and a much larger equalized-odds difference of 0.274. This is exactly the case Section 3.10 motivated: a model can look fair under one metric and unfair under another, so the module raises an alert when either crosses the threshold. The fair-by-construction dataset stays below the threshold on both metrics and produces no alert. We are not claiming that these synthetic numbers prove the framework's fairness behaviour in general. The point is that the module now exposes group-fairness violations the original feature-bias module could not see, and that reporting demographic parity and equalized odds together is necessary because either alone can mislead.

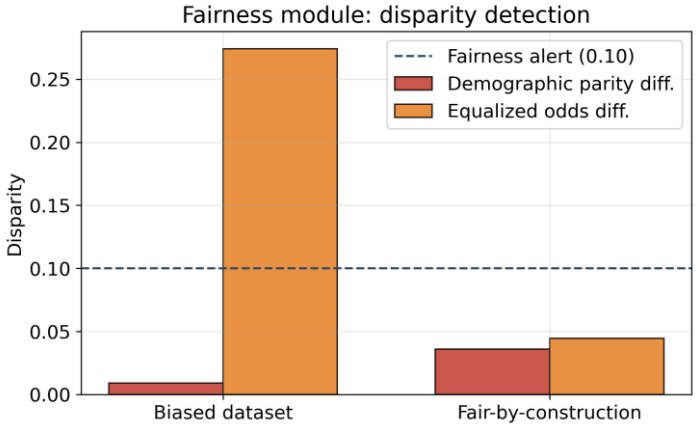

Fig. 12: Fairness module: demographic-parity and equalized-odds differences for the biased and fair-by-construction datasets, with the 0.10 alert threshold

4.12 Baseline Comparison and Capability Matrix

A useful baseline for the overfitting module is the practice it is meant to improve: a single train-test split with the same 0.05 gap threshold. Over 60 random seeds the single-split verdict turns out to be unreliable, especially for borderline models. The Random Forest is flagged in 27% of splits, with the gap ranging from 0.009 to 0.070, so a researcher running a single split has roughly a one-in-four chance of declaring it overfit. The Decision Tree is flagged in 92% of splits, Gradient Boosting in 58%, and KNN in 5%, while Logistic Regression and SVM are never flagged. The repeated-split estimator (Section 4.2), by contrast, returns a stable verdict with a tight confidence interval and does not flag the Random Forest. The framework therefore removes the borderline-model instability the single-split heuristic introduces.

Table 12: Single-split flag rate over 60 random seeds with the 0.05 threshold, compared to the gap range and spread.

Model	Single-split flag rate	Single-split gap range	Spread
Random Forest	27%	[+0.0088, +0.0702]	0.0614
Gradient Boosting	58%	[+0.0175, +0.1053]	0.0877
Logistic Regression	0%	[-0.0154, +0.0438]	0.0592
SVM (RBF)	0%	[-0.0154, +0.0482]	0.0636
KNN (k=5)	5%	[-0.0308, +0.0614]	0.0921
Decision Tree	92%	[+0.0263, +0.1404]	0.1140

A second baseline is the set of general-purpose toolkits that the framework is sometimes compared with. None of them is a like-for-like competitor; they are specialised tools that the framework complements rather than replaces. Table 13 sets out who does what, including the conventional single-split heuristic and the [4] checklist, so the framework's contribution is precise: a single reproducible audit that runs overfitting, leakage, bias, robustness, statistical, and fairness checks together.

Table 13: Capability matrix: which auditing dimensions each baseline addresses. ✓ = supported, partial = limited support, — = not addressed.

Capability	Single split	scikit-learn	AIF360	What-If Tool	Lones (2024)	This work
Overfitting detection	Partial	Partial	—	—	guidance	✓ (repeated split)
Leakage screening	—	—	—	—	guidance	✓
Feature-bias (Gini)	—	—	—	—	—	✓
Model-agnostic SHAP	—	—	—	✓	—	✓
Statistical CV + correction	—	Partial	—	—	guidance	✓ (Holm)

Robustness (noise/dropout)	—	—	—	partial	—	✓ (multi-model)
Group fairness (DPD/EOD)	—	—	✓	partial	—	✓
One reproducible report	—	—	—	—	—	✓

The contribution is therefore integrative on purpose, not a new algorithm. Existing toolkits cover individual dimensions well, and the framework's value is in running the complementary checks together with explicit, reproducible scoring.

4.13 Comparative Discussion and Implications

The results show that the framework produces actionable insights beyond a standard accuracy report. It identifies patterns that a routine evaluation would miss: the moderate overfitting of Gradient Boosting hidden by a still-acceptable test accuracy of 0.9298, the feature-importance concentration that leaves a model vulnerable when several features become unavailable, and the resilience ordering across noise and dropout that mirrors the overfitting verdicts. The multi-dataset analysis adds that the same defaults hold up on a different-domain benchmark, that the leakage module fires on a known leak hidden among genuine features, and that severe class imbalance brings out more overfitting flags rather than fewer. The single-split baseline shows that the conventional heuristic would mislabel borderline models in about a quarter of random seeds, which is the case for the repeated-split estimator as a small but real improvement on standard practice.

The ablation study is the strongest methodological argument for the design. Each module adds a measurable, non-redundant signal, and no module can replace the others. The most impactful module on this dataset is robustness testing, with statistical validation, leakage detection, and feature-importance analysis providing complementary checks. The multi-faceted approach lines up with the consensus in the machine learning auditing literature that no single metric captures the full range of model-quality issues [4, 5].

For researchers, the framework offers a systematic checklist to apply before publication. For practitioners, it is an automated diagnostic that can sit inside a deployment workflow and catch problems before they reach production. For regulators, it is a concrete technical realisation of the auditing requirements set out in the EU AI Act and the NIST AI Risk Management Framework.

5 Ethical Considerations and Limitations

Building a model auditing framework raises ethical questions worth stating openly. The main motivation is the prevention of harm from deploying flawed models. In healthcare, an overfitting model trained on leaky data can produce dangerously overconfident predictions. In finance, a biased model can systematically disadvantage demographic groups. The framework aims to detect such problems, but it does not guarantee their absence. No automated system can replace human judgment about whether a model is fit for a particular use.

The earlier version of this paper acknowledged that the feature-bias module measures statistical concentration rather than group fairness, and left fairness to future work. We have closed that gap. The new fairness module (Section 3.10, Section 4.11) reports demographic-parity and equalized-odds differences when a binary protected attribute is supplied, and raises an alert when either exceeds the threshold. The module is a detection layer, not a mitigation layer, and we say so explicitly. Reporting unfair outcomes is the first step, and integrating mitigation methods like

reweighing, adversarial debiasing, and post-processing calibration is the natural follow-on [17, 22]. The framework also stays silent on issues outside its scope: causal correctness, dataset-collection ethics, and consent for the use of personal data. Those belong to the surrounding research and deployment process, not to an auditor that runs after the data are in hand.

Several methodological limitations remain. The framework has now been evaluated on four datasets covering different domains, balance regimes, and leakage states, but all four sit at small to moderate scale. Validation on million-scale production datasets is still future work. The default thresholds (0.05 train-test gap, 0.95 leakage correlation, 0.6 Gini, 0.10 fairness alert) are explicit and adjustable rather than hidden, and Section 4.2 reports a sensitivity sweep, but a fully data-driven calibration of thresholds against a target false-alarm rate would be a stronger guarantee. The robustness module covers Gaussian noise and feature dropout. Adversarial perturbations such as FGSM or PGD, distribution shift, and concept drift are out of scope. The framework was validated on binary classification; extending it to multi-class classification, regression, and other learning paradigms is straightforward in principle but has not yet been done. The explicit composite uses equal weights as a stated default, and an operator who values one dimension more than another can substitute weights to reflect that priority.

6 Conclusion and Future Work

This paper presented and validated an automated auditing framework for overfitting detection, data leakage detection, feature-bias quantification, robustness testing, statistical validation, and group fairness in machine learning models. The framework combines eight complementary modules into a single system that can be applied to any supervised classification pipeline. The ablation study with the explicit composite of Section 3.9 confirms that each module adds a measurable, non-redundant signal, and the full framework (composite score = 0.677) scores higher than every reduced configuration.

On the Breast Cancer Wisconsin Diagnostic dataset the framework flagged Gradient Boosting and Decision Tree as overfitting. The repeated-split estimator confirmed both verdicts and showed that Random Forest stays below the 0.05 threshold on average. The data showed no leakage (maximum correlation 0.798, zero index overlap), and the bias module reported moderate feature concentration (Gini 0.577). The injected-leakage dataset in Section 4.10 was correctly flagged at a maximum correlation of 0.971. Holm-corrected paired t-tests identified the Random Forest versus Decision Tree comparison as the only one with strong evidence of a performance difference, and the remaining non-significant comparisons are now reported as such rather than interpreted as evidence of equivalence.

Several directions remain for future work. Validation on million-scale production datasets across healthcare, finance, and natural-language domains would further support the generalisability claim. Integrating mitigation methods (reweighing, adversarial debiasing, post-processing calibration) would extend the framework from detecting unfairness to reducing it. Temporal auditing for concept drift and model degradation would lift the audit from a single snapshot to lifetime monitoring. Automated remediation suggestions, where the framework identifies a problem and recommends a fix, would add practical value. Adversarial robustness testing using FGSM and PGD would complement the current noise-based evaluation. Finally, a data-driven calibration of thresholds against a target false-alarm rate would replace the current explicit defaults with empirically grounded ones.

This work delivers a practical tool that brings overfitting detection, leakage identification, feature-bias quantification, statistical validation, robustness testing, and group fairness into one automated system. The framework is not a substitute for human judgment; it is a systematic aid

for avoiding common pitfalls. In a field where the cost of model failure keeps rising, such tools are no longer optional.

References

[1] Riley, P. (2019). Three pitfalls to avoid in machine learning. *Nature*, 572(7767), 27-29. https://doi.org/10.1038/d41586-019-02307-y

[2] Kapoor, S., & Narayanan, A. (2023). Leakage and the reproducibility crisis in machine-learning-based science. *Patterns*, 4(9), 100804.7

[3] Hooker, S., Erhan, D., Kindermans, P. J., & Kim, B. (2019). A benchmark for interpretability methods in deep neural networks. *In Advances in Neural Information Processing Systems* 32 (pp. 9737-9748).

[4] Lones, M. A. (2024). How to avoid machine learning pitfalls: A guide for academic researchers (v4). arXiv preprint arXiv:2108.02497.

[5] Ghali, R., Akhloufi, M. A., & Jia, W. (2025). Overfitting in deep learning: A comprehensive survey. *AI*, 6(3), 51.

[6] Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1), 1-58.

[7] Kaufman, S., Rosset, S., Perlich, C., & Stitelman, O. (2012). Leakage in data mining: Formulation, detection, and avoidance. ACM Transactions on Knowledge Discovery from Data, 6(4), Article 15.

[8] Yang, C., Brower-Sinning, R. A., Lewis, G., & Kästner, C. (2022). Data leakage in notebooks: Static detection and better processes. *In Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering* (pp. 1-12).

[9] Nisbet, R., Miner, G., & Yale, K. (2018). Handbook of statistical analysis and data mining applications (2nd ed.). *Academic Press*.

[10] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5-32.

[11] Altmann, A., Toloşi, L., Sander, O., & Lengauer, T. (2010). Permutation importance: A corrected feature importance measure. *Bioinformatics*, 26(10), 1340-1347.

[12] Strobl, C., Boulesteix, A.-L., Zeileis, A., & Hothorn, T. (2007). Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics*, 8(1), 25.

[13] Zhang, Z., Liu, F., & Li, J. (2024). Feature importance concentration and model fragility: A systematic analysis. *Machine Learning*, 113(5), 3127-3145.

[14] Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *In Advances in Neural Information Processing Systems* 30 (pp. 4765-4774).

[15] European Union. (2024). Regulation (EU) 2024/1689 of the European Parliament and of the Council laying down harmonised rules on artificial intelligence (AI Act). Official Journal of the European Union.

[16] NIST. (2023). Artificial intelligence risk management framework (AI RMF 1.0). *National Institute of Standards and Technology*.

[17] Bellamy, R. K. E., Dey, K., Hind, M., Hoffman, S. C., Houde, S., Kannan, K., Lohia, P., Martino, J., Mehta, S., Mojsilovic, A., Nagar, S., Ramamurthy, K. N., Richards, J., Saha, D., Sattigeri, P., Singh, M., Varshney, K. R., & Zhang, Y. (2019). AI Fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias. *IBM Journal of Research and Development*, 63(4/5), 4:1-4:15.

[18] Wexler, J., Pushkarna, M., Bolukbasi, T., Wattenberg, M., Viégas, F., & Wilson, J. (2020). The What-If Tool: Interactive probing of machine learning models. *IEEE Transactions on Visualization and Computer Graphics*, 26(1), 56-65.

[19] Street, W. N., Wolberg, W. H., & Mangasarian, O. L. (1993). Nuclear feature extraction for breast tumor diagnosis. *In Biomedical Image Processing and Biomedical Visualization* (Vol. 1905, pp. 861-870). SPIE.

[20] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: Data mining, inference, and prediction (2nd ed.). *Springer*.

[21] Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. *In Proceedings of the 14th International Joint Conference on Artificial Intelligence* (pp. 1137-1143).

[22] Hardt, M., Price, E., & Srebro, N. (2016). Equality of opportunity in supervised learning. *In Advances in Neural Information Processing Systems* 29 (pp. 3315-3323).

[23] Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2), 65-70.