

ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

Intake code: APDMP2401AI

Group assignment In Pattern Recognition

|  |  |
| --- | --- |
| Student ID | Student Name |
| TP079177 | Mohamed Mohamed Said Aly Amin |
| TP 079740 | Loh Wei Chun |

Lecturer Name: [ASSOC. PROF. DR. V. SIVAKUMAR .](https://lms2.apiit.edu.my/user/view.php?id=166&course=50813)

Classification of Breast Cancer Histopathology Images Using Traditional Machine Learning: A Comparative Study of SVM, Random Forest, and XGBoost

A report submitted in fulfilment of the

requirements for the module

Pattern Recognition

Asia Pacific University of Technology & Innovation (APU)

School of Computing and Technology

January 2026

# Table of Contents

[Table of Contents 3](#_Toc218196639)

[List of Figures 7](#_Toc218196640)

[List of Tables 7](#_Toc218196641)

[1. Introduction and Background 8](#_Toc218196642)

[1.1 Abstract 8](#_Toc218196643)

[1.2 Keywords: 8](#_Toc218196644)

[1.3 Introduction 9](#_Toc218196645)

[1.4 Research question 9](#_Toc218196646)

[1.5 Objectives 10](#_Toc218196647)

[2. Literature Review 11](#_Toc218196648)

[2.1 Histopathology Computer-Aided Diagnosis Systems 11](#_Toc218196649)

[2.2 Feature Selection in Computer Vision and Medical Imaging 11](#_Toc218196650)

[2.3 Model Interpretability in Healthcare AI 12](#_Toc218196651)

[2.4 Deployment and MLOps Considerations 13](#_Toc218196652)

[3. Methodology 14](#_Toc218196653)

[3.1 Problem Formulation 14](#_Toc218196654)

[3.2 Dataset Collection and Preprocessing 14](#_Toc218196655)

[3.2.1 Dataset Characteristics 14](#_Toc218196656)

[3.2.2 Image Preprocessing Pipeline 15](#_Toc218196657)

[3.3 Feature Extraction Methods 17](#_Toc218196658)

[4. Feature Selection Methodology 19](#_Toc218196659)

[4.1 Variance & Correlation Filtering 19](#_Toc218196660)

[4.2 Univariate Feature Selection Using F-Score 19](#_Toc218196661)

[4.3 Recursive Feature Elimination 19](#_Toc218196662)

[4.4 Tree-Based Feature Importance Ranking 19](#_Toc218196663)

[4.5 LASSO Regularization 20](#_Toc218196664)

[4.6 Dimensionality Reduction with PCA 20](#_Toc218196665)

[4.7 Ensemble Ranking and Final Selection 20](#_Toc218196666)

[5. Model Training 22](#_Toc218196667)

[5.1 Support Vector Machine with Linear Discriminant Analysis Preprocessing 22](#_Toc218196668)

[5.2 Random Forest Classifier 24](#_Toc218196669)

[5.3 XGBoost Classifier 24](#_Toc218196670)

[5.4 Hyperparameter Tuning 25](#_Toc218196671)

[5.2 Evaluation Metrics 26](#_Toc218196672)

[5.2.1 Confusion Matrix 26](#_Toc218196673)

[5.2.2 Accuracy 27](#_Toc218196674)

[5.2.3 Sensitivity (Recall) 27](#_Toc218196675)

[5.2.4 Specificity 27](#_Toc218196676)

[5.2.5 Precision 28](#_Toc218196677)

[5.2.6 F1-Score 28](#_Toc218196678)

[6. Graphical user interface 29](#_Toc218196679)

[6.1 Design goals 29](#_Toc218196680)

[6.2 GUI layout and components 30](#_Toc218196681)

[6.2.1 Controls panel (left) 30](#_Toc218196682)

[6.2.2. Image panel (left-middle) 31](#_Toc218196683)

[6.2.3. Prediction panel (left-bottom) 31](#_Toc218196684)

[6.2.4. Preview panel (center) 32](#_Toc218196685)

[6.2.5. Scores panel (right) 33](#_Toc218196686)

[6.2.6. Log panel (bottom) 34](#_Toc218196687)

[6.3 Inference workflow inside the GUI 35](#_Toc218196688)

[7. Results And Discussion 36](#_Toc218196689)

[7.1 Performance Comparison 36](#_Toc218196690)

[7.2 Feature Importance Analysis and Biological Interpretation 38](#_Toc218196691)

[7.3 Error Analysis 40](#_Toc218196692)

[7.4 Comparison with Literature 41](#_Toc218196693)

[7.5 Clinical Relevance 42](#_Toc218196694)

[8. Lessons And Challenges 42](#_Toc218196695)

[9. Future Work and Directions 43](#_Toc218196696)

[9.1 Immediate high-impact improvements: 43](#_Toc218196697)

[9.2 Medium-term extensions: 43](#_Toc218196698)

[9.3 Long-term clinical translation: 43](#_Toc218196699)

[10. Conclusion 44](#_Toc218196700)

[11. Individual Contributions 45](#_Toc218196701)

[11.1 Overview of Contributions 45](#_Toc218196702)

[11.2 Mohamed Said Aly- Core Contributions (~60%) 45](#_Toc218196703)

[Key Responsibilities: 45](#_Toc218196704)

[11.3 Loh Wei Chun - Core Contributions (~40%) 46](#_Toc218196705)

[Key Responsibilities: 46](#_Toc218196706)

[11.4 Joint Collaborative Efforts 47](#_Toc218196707)

[11.5 Code and Time Summary 47](#_Toc218196708)

[11.6 Complementary Strengths 47](#_Toc218196709)

[11.7 Declaration 48](#_Toc218196710)

[11.8 Acknowledgments 48](#_Toc218196711)

[References 49](#_Toc218196712)

[Appendix 51](#_Toc218196713)

[APPENDIX A - (Pseudo Code) 51](#_Toc218196714)

[A.1 ALGORITHM 1: Main Classification Pipeline 51](#_Toc218196715)

[A.2 ALGORITHM 2: Patient-Disjoint Data Splitting 53](#_Toc218196716)

[A.3 ALGORITHM 3: 10-Stage Image Preprocessing Pipeline 54](#_Toc218196717)

[A.4 ALGORITHM 4: Ensemble Feature Selection via Mean Rank Aggregation 55](#_Toc218196718)

[A.5 ALGORITHM 5: SVM with LDA Dimensionality Reduction 56](#_Toc218196719)

[A.6 ALGORITHM 6: Performance Evaluation with Bootstrap Confidence Intervals 57](#_Toc218196720)

[APPENDIX B - (Code Snippet) 58](#_Toc218196721)

[B.1 Image Preprocessing Pipeline 58](#_Toc218196722)

[B.2 Feature Extraction Methods 60](#_Toc218196723)

[B.3 Feature Selection Methods 63](#_Toc218196724)

[B.4 Model Training with LDA Preprocessing 66](#_Toc218196725)

[B.5 Performance Evaluation 68](#_Toc218196726)

[B.6 Summary of Implementation Details 70](#_Toc218196727)

[APPENDIX C - Supplementary Figures 72](#_Toc218196728)

[C.1 SVM performance summary 72](#_Toc218196729)

[C.2 RF performance summary 72](#_Toc218196730)

[C.3 XGBoost performance summary 73](#_Toc218196731)

[C.4 ReliefF feature importance 73](#_Toc218196732)

[C.5 PCA analysis 74](#_Toc218196733)

[B.6 Confusion breakdown 74](#_Toc218196734)

# List of Figures

[Figure 1- Image Preprocessing Pipeline 15](#_Toc218196735)

[Figure 2- Visualization of the 10-step image preprocessing workflow 16](#_Toc218196736)

[Figure 3 - Feature Extraction Methods 17](#_Toc218196737)

[Figure 4 - Number of Features Extracted by Each Method 18](#_Toc218196738)

[Figure 5 - Computational Efficiency Comparison of Feature Selection Methods 21](#_Toc218196739)

[Figure 6 - The LDA projection 23](#_Toc218196740)

[Figure 7 - Cross-Validation Performance Across Different Feature Subset Sizes 26](#_Toc218196741)

[Figure 8 – GUI Model Panel 30](#_Toc218196742)

[Figure 9 – GUI Image panel 31](#_Toc218196743)

[Figure 10 – GUI Prediction panel 31](#_Toc218196744)

[Figure 11 – GUI Preview panel 32](#_Toc218196745)

[Figure 12 – GUI Scores panel 33](#_Toc218196746)

[Figure 13 – GUI Log panel 34](#_Toc218196747)

[Figure 14 – Complete GUI of the Breast Cancer Classifier application 35](#_Toc218196748)

[Figure 15 - Performance Comparison of SVM, Random Forest, and XGBoost Across All Metrics 36](#_Toc218196749)

[Figure 16 - ROC Curves Comparing Classification Performance of Three Models 37](#_Toc218196750)

[Figure 17 - Confusion Matrices for SVM, Random Forest, and XGBoost Classifiers 37](#_Toc218196751)

[Figure 18 - SVM performance summary 72](#_Toc218196752)

[Figure 19 - RF performance summary 72](#_Toc218196753)

[Figure 20 - XGBoost performance summary 73](#_Toc218196754)

[Figure 21 - ReliefF feature importance 73](#_Toc218196755)

[Figure 22 - PCA Analysis 74](#_Toc218196756)

[Figure 23 - True Negative, False Positive, False Negative, and True Positive Breakdown by Classifier 74](#_Toc218196757)

# List of Tables

[Table 1 - Cross-Validation Accuracy by Feature Count 20](#_Toc218196758)

[Table 2 - Comprehensive Model Performance on Test Set 36](#_Toc218196759)

[Table 3 - Benchmark Performance (BreakHis 100X) 41](#_Toc218196760)

[Table 4 - Summary of Individual Contributions 45](#_Toc218196761)

[Table 5 - Quantitative Contribution Breakdown 47](#_Toc218196762)

[Table 6 - Key Libraries and Functions Used 70](#_Toc218196763)

# Introduction and Background

## 1.1 Abstract

Breast cancer remains one of the leading causes of cancer-related mortality among women worldwide, with accurate histopathological diagnosis being critical for treatment planning and patient outcomes.

This study presents a complete pattern recognition system for automated classification of breast histopathology images into benign and malignant categories using traditional machine learning approaches.

Researchers implemented a complete pipeline on the BreakHis dataset (100X magnification), comprising 1,078 training and 194 test images with patient-disjoint splitting to prevent data leakage.

The methodology involved a 10-stage preprocessing pipeline, extraction of 2,767 multi-modal features through six methods (HOG, GLCM, LBP, Gabor filters, intensity histograms, and statistical descriptors), and a complete feature selection strategy combining seven complementary techniques (variance filtering, correlation filtering, ReliefF, F-Score, RFE, tree-based importance, and LASSO regularization) to identify the most discriminative 400 features.

Three classifiers were systematically compared: Support Vector Machine with RBF kernel, Random Forest, and XGBoost, with hyperparameters optimized through 3-fold cross-validation. Random Forest achieved the best performance with 70.1% accuracy, 81.4% specificity, and AUC of 0.770, followed closely by XGBoost (69.6% accuracy, 78.4% specificity, AUC of 0.779), while SVM reached 64.4% accuracy with 66.0% specificity and AUC of 0.661.

Although XGBoost's sensitivity (60.8%) indicates room for improvement in detecting malignant cases, the models demonstrated interpretable feature importance rankings that align with known pathological characteristics GLCM texture features and morphological descriptors emerged as most discriminative.

This work demonstrates that traditional machine learning, while achieving 70.1% accuracy compared to deep learning benchmarks reaching 83-93%, offers critical advantages

The complete implementation, source code, and experimental datasets are publicly available at <https://github.com/msaid1976/BreastCancerHistopathological>

## 1.2 Keywords:

Breast cancer, histopathology, pattern recognition, machine learning, SVM, Random Forest, XGBoost, feature selection, medical image analysis, computer-aided diagnosis

## 1.3 Introduction

Breast cancer represents a major global health challenge, accounting for approximately 2.3 million new cases and 685,000 deaths annually according to the World Health Organization (WHO, 2020) Early and accurate diagnosis is paramount for improving patient survival rates and treatment outcomes. The diagnostic gold standard remains histopathological examination of tissue samples, where pathologists analyze microscopic images of hematoxylin and eosin (H&E) stained tissue sections to distinguish between benign and malignant lesions. This process, while highly accurate when performed by experienced pathologists, is inherently time-consuming, subjective, and prone to inter-observer variability, studies have shown disagreement rates of up to 25% among pathologists for borderline cases (Elmore et al., 2015).

The traditional diagnostic workflow in breast pathology involves multiple labor-intensive stages: tissue fixation, paraffin embedding, microtome sectioning, H&E staining, and microscopic examination at various magnifications. Pathologists must assess numerous morphological features including nuclear size and shape, chromatin patterns, cellular architecture, mitotic activity, and overall tissue organization. For a single patient, this evaluation can take 15-30 minutes per slide, and with increasing caseloads in clinical laboratories, there's a growing need for automated support systems to reduce diagnostic burden and minimize human error.

Computer-aided diagnosis (CAD) systems have emerged as a promising solution to augment pathologist expertise. Deep learning approaches have achieved state-of-the-art performance in breast histopathology classification. (Araújo et al., 2017) developed a multi-scale CNN architecture that achieved 83.3% accuracy with 95.6% sensitivity on the BreakHis dataset by capturing both nuclear and tissue-organization features. Subsequent work by Bardou et al. (2018) and Yan et al. (2020) further improved accuracy to 91-93% using transfer learning and attention mechanisms. However, these gains require GPU hardware, extensive training time, and sacrifice interpretability, creating barriers for clinical adoption in resource-constrained settings or regulatory environments requiring explainability. This motivates our investigation of traditional machine learning approaches that prioritize transparency and computational efficiency while accepting lower raw accuracy.

**Pattern recognition challenges in histopathology**

Breast histopathology images present unique challenges for automated classification. Despite clear visual differences to trained pathologists, these images exhibit high intra-class variability (benign tumors can look quite different from each other depending on subtype) and sometimes subtle inter-class differences (well-differentiated malignant tumors may resemble benign lesions).

The BreakHis dataset used in this study contains four benign subtypes (adenosis, fibroadenoma, phyllodes tumor, tubular adenoma) and four malignant subtypes (ductal carcinoma, lobular carcinoma, mucinous carcinoma, papillary carcinoma), each with distinct morphological characteristics but overlapping visual features (Spanhol et al., 2016a).

## 1.4 Research question

How effectively can traditional machine learning methods classify breast histopathology images, and what insights can feature importance analysis provide about the discriminative characteristics of benign versus malignant tissue?

## 1.5 Objectives

This study has three primary objectives.

* First, develop a complete pattern recognition system for automated binary classification of breast histopathology images (benign vs. malignant) using traditional machine learning algorithms, achieving clinically relevant performance metrics.
* Second, conduct a systematic comparative analysis of three widely-used classifiers Support Vector Machine (SVM), Random Forest (RF), and XGBoost, To identify which approach is most suitable for this specific application.
* Third, extract and interpret feature importance rankings to identify the most discriminative image characteristics, providing biological and clinical interpretation that could guide future feature engineering efforts.

By achieving these objectives, we aim to demonstrate that traditional machine learning remains a valuable approach for medical image analysis, particularly in scenarios where interpretability, computational constraints, or limited training data make deep learning impractical.

# Literature Review

## 2.1 Histopathology Computer-Aided Diagnosis Systems

Computer-aided diagnosis in digital pathology has evolved substantially over the past two decades. Early work focused on nuclear morphometry (measuring nuclear size, shape, and chromatin distribution), as these were known to be reliable indicators of malignancy (Gurcan et al., 2009). However, texture analysis proved more effective for tissue-level classification.

A significant advancement came with Gray-Level Co-occurrence Matrices (GLCM), introduced by (Haralick et al., 1973), which capture spatial relationships between pixel intensities. GLCM derived features such as contrast, correlation, energy, homogeneity, and entropy quantify texture patterns that correspond to tissue organization, benign tissues typically exhibit more homogeneous, organized patterns while malignant tissues show heterogeneity due to cellular pleomorphism and architectural disruption. Numerous studies have validated GLCM features for breast histopathology classification, demonstrating their ability to capture textural differences with high discriminative power.

Local Binary Patterns (LBP), introduced by (Ojala et al., 2002), provide computationally efficient texture characterization with rotation invariance, a valuable property for histopathology images where tissue orientation varies. LBP is particularly effective for describing chromatin texture patterns: benign cells exhibit coarse, irregular chromatin while malignant cells show fine, densely packed chromatin. The rotation-invariant uniform LBP variant has shown exceptional performance in characterizing these microscopic texture differences.

Since 2015, deep learning has transformed medical image analysis. For the BreakHis dataset specifically, (Wang et al., 2018) achieved 91.3% accuracy using data augmentation and CNN fine-tuning, while (Yan et al., 2020) reached 93.2% with a hybrid architecture combining ResNet and attention mechanisms. However, these deep learning models required extensive computational resources (multi-GPU training for days) and lacked interpretability.

This creates a gap that traditional machine learning can fill moderate performance (70-85% accuracy) with full interpretability, low computational cost, and ability to work with limited training data.

## 2.2 Feature Selection in Computer Vision and Medical Imaging

Feature selection is critical in machine learning, particularly for medical imaging where high-dimensional feature spaces create the curse of dimensionality. When the number of features (p) far exceeds samples (n), models become prone to overfitting, learning noise rather than true patterns.

**Filter methods** evaluate features independently of the classifier. F-Score (ANOVA F-test) is widely used, computing the ratio of between-class variance to within-class variance for each feature. The advantage is computational efficiency, but the limitation is the independence assumption, potentially missing feature interactions.

**Wrapper methods** evaluate feature subsets by training the actual classifier and measuring performance. Recursive Feature Elimination (RFE), introduced by (Guyon et al., 2002), iteratively trains an SVM, ranks features by weight magnitude, removes the lowest-ranked, and repeats. RFE is computationally expensive but identifies features specifically useful for SVM decision boundaries.

**Embedded methods** perform feature selection during model training. LASSO (Least Absolute Shrinkage and Selection Operator) adds an L1 penalty to the optimization objective, shrinking coefficients of irrelevant features exactly to (Tibshirani, 1996). For medical imaging with thousands of features, LASSO has successfully identified sparse biomarker signatures.

**Distance-based methods** like ReliefF capture feature relevance through nearest-neighbor analysis (Kononenko, 1994). For each instance, ReliefF finds k nearest neighbors from the same class (near-hits) and different class (near-misses), rewarding features that separate misses while keeping hits together. The key advantage is detecting feature interactions features that are individually weak but jointly strong.

**Ensemble feature selection**, which combines rankings from multiple methods, has emerged as a reliable approach. Different methods have complementary biases, and averaging rankings provides consensus that are less likely to suffer from any single method's weaknesses (Saeys et al., 2008). This ensemble approach has been validated in gene expression and medical imaging, consistently outperforming individual methods.

## 2.3 Model Interpretability in Healthcare AI

Interpretability has become a critical requirement for deploying machine learning in healthcare. The European Union's GDPR enshrines a "right to explanation" for automated decisions, and the U.S. FDA increasingly requires explainability for medical AI systems (Holzinger et al., 2017). Pathologists need to understand why a CAD system made a particular prediction for several reasons.

* First, trust and validation, clinicians won't adopt systems they can't validate against their expertise.
* Second, error identification, when models fail, interpretable outputs help identify whether the failure is due to image quality issues, rare subtypes, or fundamental model limitations.
* Third, educational value, seeing which features drive predictions can teach residents about quantitative markers of malignancy.
* Fourth, legal and ethical accountability, if a misdiagnosis occurs, we need to understand the decision process.

Traditional ML offers inherent interpretability: SVM reveals support vectors and feature weights, Random Forest provides Gini importance measuring each feature's contribution to classification, and XGBoost offers gain-based importance.

In contrast, deep CNNs are fundamentally opaque while saliency maps provide some insight into which image regions matter, they don't explain what specific features drove the decision in terms pathologists understand.

There's often a trade-off between performance and interpretability. Deep learning achieves higher accuracy but lower interpretability, while traditional ML offers moderate performance with full transparency. For high-stakes medical decisions where misclassification has serious consequences, many argue that interpretability should take priority, accepting a small performance decrease for explainability.

Our work quantifies this trade-off: we achieve 70.1% accuracy (vs. 90%+ for deep learning) but can precisely identify which texture and morphological features distinguish benign from malignant tissue.

## 2.4 Deployment and MLOps Considerations

Successfully deploying machine learning models in clinical practice requires addressing challenges beyond algorithmic performance. Production systems must handle reliable preprocessing pipelines dealing with scanner variations, implement model versioning for regulatory compliance (FDA Class II device requiring 510(k) premarket notification), and integrate seamlessly with clinical workflows through PACS/LIS systems.

Several deployment patterns exist:

* Batch processing, for non-urgent cases (process overnight, cost-effective).
* Real-time inference, for urgent cases (predictions within seconds, requires optimized code).
* Edge deployment, running models on local workstations (avoids data transmission, important for patient privacy).

Our traditional ML models train in seconds and predict in <1ms per image on standard CPUs, making all deployment patterns viable. In contrast, deep learning models often require GPUs even for inference, limiting deployment options.

Monitoring is essential as ML models can degrade over time due to data distribution shifts from new staining protocols, different patient demographics, or scanner upgrades. Production systems must monitor performance metrics and alert when they drop below thresholds.

For clinical AI, quarterly model retraining with new data maintains performance. Additionally, the interpretable nature of traditional ML facilitates regulatory review and approval, as explanations help FDA reviewers understand the decision-making process.

# Methodology

## 3.1 Problem Formulation

We formulate breast histopathology classification as a supervised binary classification task given RGB histopathology image I from H&E-stained tissue, predict class label y ∈ {0=benign, 1=malignant}. The goal is learning function f: I → y that maximizes classification accuracy while maintaining high sensitivity (minimizing false negatives missed cancers).

Clinical constraints guide our approach: sensitivity priority (false negatives are clinically more severe than false positives), interpretability requirement (feature importance rankings must align with pathological criteria), computational efficiency (process images in <10 seconds on standard CPUs), and generalization (patient-disjoint train/test split to simulate real-world deployment).

## 3.2 Dataset Collection and Preprocessing

**Dataset availability:**<https://www.kaggle.com/datasets/ambarish/breakhis>

### 3.2.1 Dataset Characteristics

We utilized the Breast Cancer Histopathological Database (BreakHis), introduced by (Spanhol et al., 2016), containing 7,909 microscopic images from 82 patients at four magnifications (40X, 100X, 200X, 400X).

We selected 100X magnification because it offers the optimal trade-off: it retains sufficient cellular resolution to discern nuclear morphology (unlike lower magnifications) while preserving the critical global tissue architecture and glandular structures that are often lost at 40X, 200X, and 400X.

At 100X: 1,995 total images (625 benign, 1,370 malignant). The dataset contains four benign subtypes (adenosis, fibroadenoma, phyllodes tumor, tubular adenoma) and four malignant subtypes (ductal carcinoma, lobular carcinoma, mucinous carcinoma, papillary carcinoma).

**Patient-disjoint splitting strategy**,A critical consideration is preventing data leakage. Multiple images from the same patient are inherently correlated (same tissue preparation, scanner settings), so random image-level splitting would artificially inflate performance. We parsed filenames to extract patient IDs, then assigned 85% of patients to training and 15% to testing, ensuring zero patient overlap. This rigorous approach tests true generalization to new patients.

**Final composition:**

* Training: 1,078 images (531 benign from 20 patients, 547 malignant from 49 patients)
* Testing: 194 images (97 benign from 4 patients, 97 malignant from 9 patients)
* Patient overlap: ZERO (verified programmatically)

### 3.2.2 Image Preprocessing Pipeline

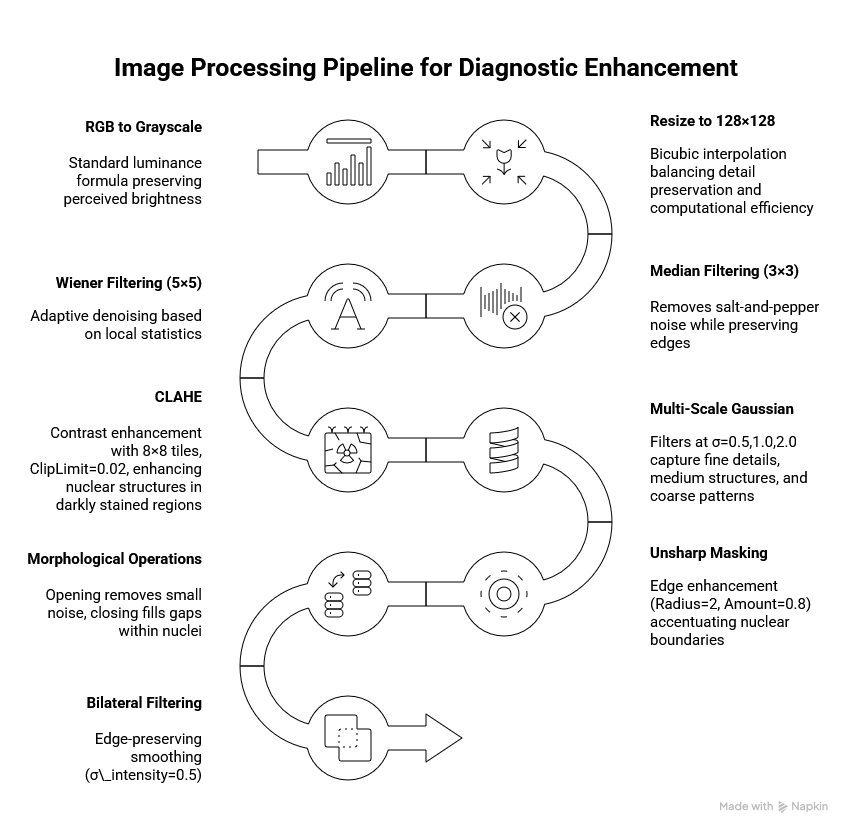


Figure 1- Image Preprocessing Pipeline

Each input image undergoes a 10-stage pipeline designed to standardize appearance, reduce noise, and enhance diagnostically relevant structures.

1. **RGB to Grayscale:** Standard luminance formula preserving perceived brightness.
2. **Resize to 128×128:** Bicubic interpolation balancing detail preservation and computational efficiency.
3. **Median Filtering (3×3):** Removes salt-and-pepper noise while preserving edges.
4. **Wiener Filtering (5×5):** Adaptive denoising based on local statistics.
5. **CLAHE:** Contrast enhancement with 8×8 tiles, ClipLimit=0.02, enhancing nuclear structures in darkly stained regions.
6. **Multi-Scale Gaussian:** Filters at σ=0.5,1.0,2.0 capture fine details, medium structures, and coarse patterns.
7. **Unsharp Masking:** Edge enhancement (Radius=2, Amount=0.8) accentuating nuclear boundaries.
8. **Morphological Operations:** Opening removes small noise, closing fills gaps within nuclei
9. **Bilateral Filtering:** Edge-preserving smoothing (σ\_intensity=0.5).
10. **Intensity Normalization:** Rescale to [0,1] ensuring consistent dynamic range.

This pipeline transforms highly variable raw images into standardized representations optimized for feature extraction (~0.3s per image on CPU).

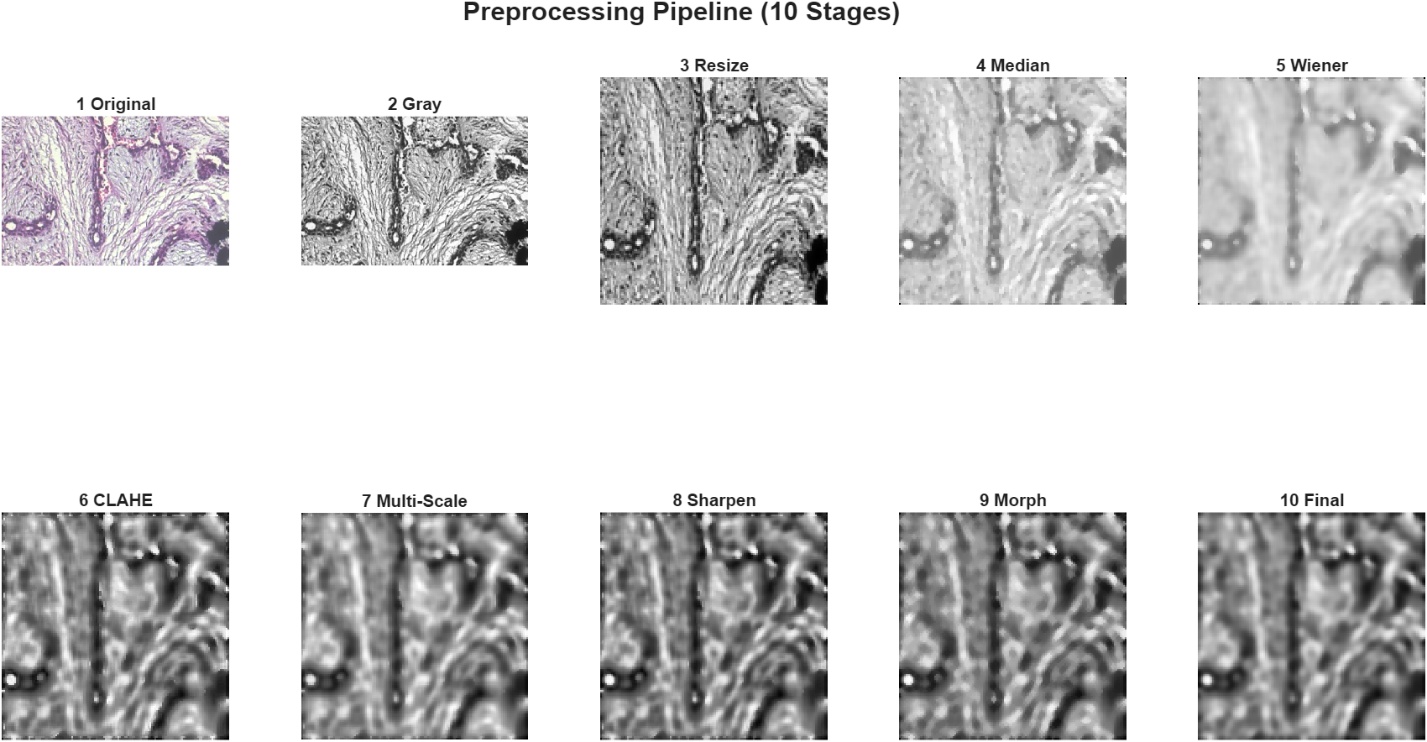


Figure 2- Visualization of the 10-step image preprocessing workflow

## 3.3 Feature Extraction Methods

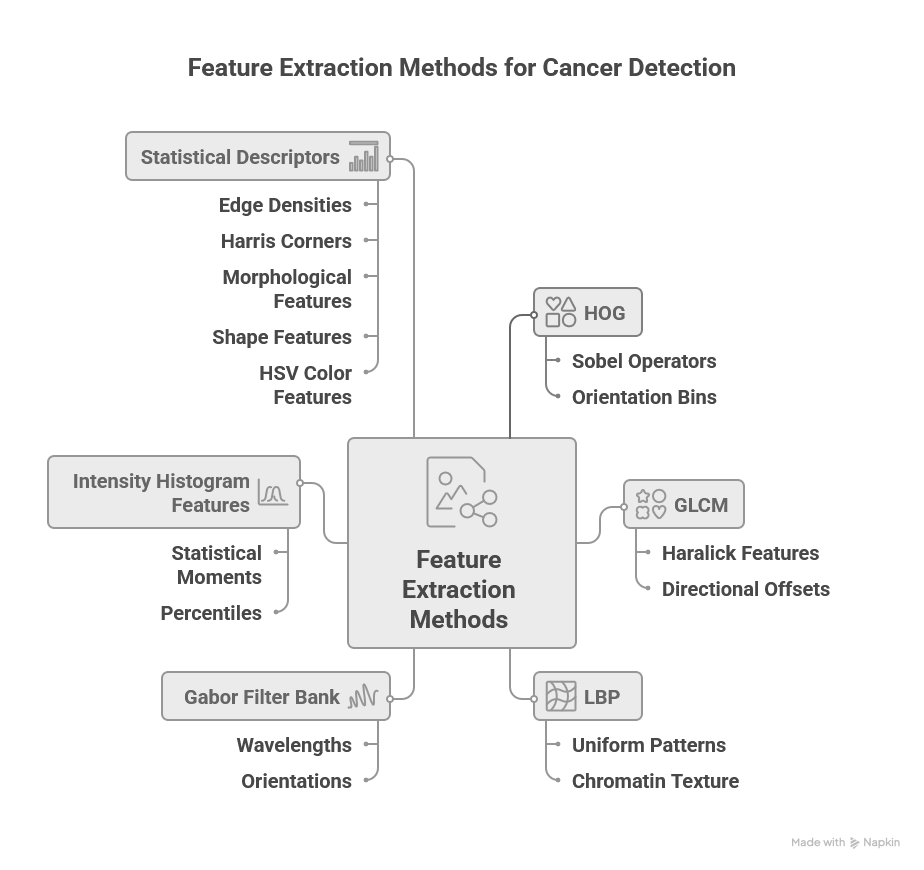


Figure 3 - Feature Extraction Methods

By integrating these six methods, we construct a diverse feature space that addresses the multifaceted nature of histological images, ensuring that variations in texture, geometry, and coloration are all utilized for classification. We extract 2,767 features using six complementary methods:

**Histogram of Oriented Gradients (HOG) - 1,764 features:** Gradient-based shape descriptors capturing nuclear boundary orientations. Implementation uses Sobel operators for gradient computation, 9 orientation bins covering 0-180°, 16×16-pixel cells, and 2×2 cell blocks with L2 normalization. Malignant nuclei exhibit irregular shapes with diverse gradient patterns, while benign nuclei show circular, consistent orientations. HOG quantifies this morphological irregularity.

**GLCM (Gray-Level Co-occurrence Matrix) - 6 features:** Computes five Haralick features (Contrast, Correlation, Energy, Homogeneity, Entropy) from co-occurrence matrices at 8 directional offsets. These features quantify spatial texture patterns: high entropy and contrast indicate tissue disorganization characteristic of malignancy, while high homogeneity and energy suggest organized benign tissue. GLCM captures the disrupted tissue architecture that pathologists observe visually.

**Local Binary Patterns (LBP) - 944 features:** Rotation-invariant uniform patterns characterize local texture. Implementation divides images into 4×4 grid of 32×32-pixel cells, computes 59-bin histograms of uniform LBP patterns, and L2-normalizes each histogram. LBP excels at describing chromatin texture: benign cells have coarse chromatin with distinct granules (high LBP variance), malignant cells show fine, densely packed chromatin (more uniform LBP codes).

**Gabor Filter Bank - 24 features:** multi-scale, multi-orientation frequency analysis using wavelengths λ= {2,4,8} pixels and orientations θ={0°,45°,90°,135°}. For each of 12 filters, we compute mean and standard deviation of response magnitude. This captures tissue structures at multiple scales: fine structures like nuclear pores (λ=2), medium structures like individual cells (λ=4), and coarse structures like glandular arrangements (λ=8). Different orientations reveal tissue directionality benign glandular tissue shows preferential alignment, malignant tissue more isotropic patterns.

**Intensity Histogram Features - 7 features:** Statistical moments (mean, variance, skewness, kurtosis) and percentiles (10th, 50th, 90th) quantify pixel intensity distribution. Nuclear hyperchromatism darker, more densely stained nuclei, which is a hallmark of malignancy. Mean intensity captures overall darkness, variance indicates heterogeneity, skewness captures distribution asymmetry, and kurtosis measures tail heaviness. These features directly quantify the visual observation that malignant cells take up more hematoxylin.

**Statistical Descriptors - 19 features:** Complete set capturing: (1) Edge densities from Sobel, Canny, and LoG detectors quantifying boundary complexity; (2) Harris corner detection yielding count, mean strength, std, and max strength; (3) Morphological features including skeleton length, Euler number, and area fraction; (4) Shape features from largest connected component (solidity, eccentricity, extent) measuring compactness and irregularity; (5) HSV color features (Hue and Saturation mean/std) capturing H&E staining variations.

This complete 2,767-feature set captures diverse aspects of tissue appearance from multiple complementary perspectives. The redundancy provides robustness, if one method fails to capture a discriminative pattern, others typically will.

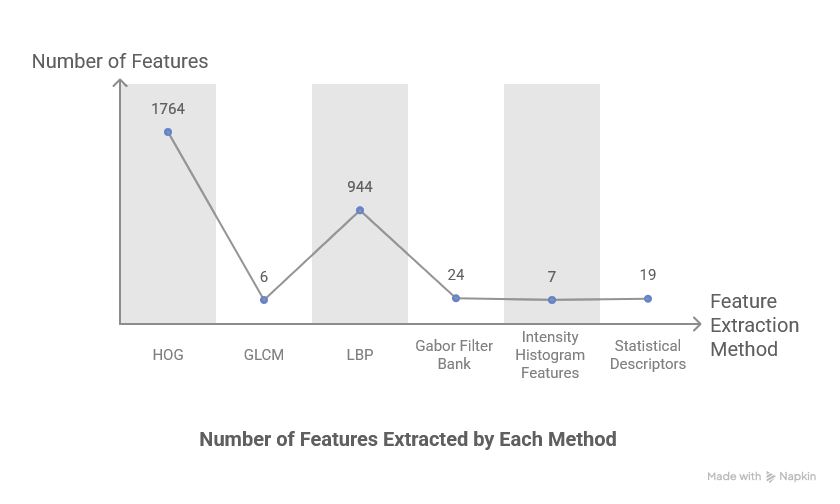


Figure 4 - Number of Features Extracted by Each Method

# 4. Feature Selection Methodology

With 2,767 features from 1,078 samples (p/n = 2.57), we face the curse of dimensionality. Feature selection is essential to improve classification performance by removing noise, reducing computational cost, enhance interpretability, and mitigate overfitting.

We employ an ensemble strategy combining seven complementary techniques, each operating on different mathematical principles.

## 4.1 Variance & Correlation Filtering

Initial preprocessing removes uninformative and redundant features. Variance filtering eliminates features with σ² < 0.001 (near-constant values provide no discriminative power). Correlation filtering removes one feature from highly correlated pairs (|ρ| > 0.95), keeping the feature with higher variance. Results: 678 features removed by variance threshold, 5 by correlation threshold, leaving 2,084 features for subsequent selection methods. This simple preprocessing dramatically reduces dimensionality with negligible computation (<0.1s).

## 4.2 Univariate Feature Selection Using F-Score

F-Score provides fast statistical ranking based on ANOVA. For each feature, compute F-statistic = (SSB/df\_B)/(SSW/df\_W) where SSB is between-class sum of squares and SSW is within-class sum of squares. Large F values indicate features where between-class variance exceeds within class variance. Top features identified: GLCM Entropy (F=458.3), GLCM Contrast (F=412.7), Shape Eccentricity (F=387.1), Intensity Mean (F=362.5). Computational time: <0.1s. Limitation assumes feature independence, may miss interactions.

## 4.3 Recursive Feature Elimination

RFE is a wrapper method performing greedy backward selection. Algorithm: train linear SVM, rank features by |weight|, remove lowest ranked, repeat until target count reached. We use batch elimination (remove bottom 10% per iteration) for efficiency. Results: Top 50 features identified with SVM-optimized ranking, emphasizing GLCM texture, HOG shape, and Intensity features. Time: 11.5s. This ranking differs from statistical methods, capturing non-linear decision surfaces specific to SVM.

## 4.4 Tree-Based Feature Importance Ranking

Random Forest provides inherent importance through Gini impurity. For each feature and each tree, compute importance as sum of impurity reductions when that feature creates splits. Aggregate across all trees and normalize. Implementation: 50 trees, unlimited depth, MinLeafSize=5.

Results: Top features include Intensity Mean (0.082), GLCM Entropy (0.071), Gabor λ=4,θ=0° (0.063), Shape Solidity (0.058). Time: 22.4s. This measure naturally accounts for feature interactions since tree construction considers feature combinations.

## 4.5 LASSO Regularization

LASSO performs embedded selection through L1 regularization. For logistic regression, minimize: −Σ[y·log σ(β^Tx) + (1−y) ·log(1−σ(β^Tx))] + λΣ|β|. The L1 penalty shrinks irrelevant coefficients exactly to zero. Cross-validation selects λ balancing prediction accuracy and sparsity. Results: Optimal λ=0.01, producing 347 non-zero coefficients from 2,084 features. Top features by |coefficient|: GLCM Contrast (2.84), Intensity Std (2.31), Gabor λ=2,θ=90° (2.08). Time: 7.8s. LASSO identifies features that work well together in linear combination.

## 4.6 Dimensionality Reduction with PCA

PCA performs unsupervised linear projection onto orthogonal directions of maximum variance. Compute eigen decomposition of covariance matrix, sort eigenvalues descending, select k components capturing 95% cumulative variance. Results: 426 components selected, achieving 79.6% dimensionality reduction (2,084→426). Time: 0.7s. While PCA creates new features rather than selecting originals, it serves dual purposes: decorrelating features for SVM preprocessing and reducing noise by discarding low-variance components.

## 4.7 Ensemble Ranking and Final Selection

We combine rankings from five complementary methods (ReliefF, F-Score, RFE, Tree Importance, LASSO) through mean rank aggregation. For each feature, average its rank across methods; features with low mean rank (high importance across multiple methods) are selected.

Cross-validation determines optimal feature count K. We tested K ∈ {150,300,400} using 3-fold stratified CV on training set:

|  |  |  |  |
| --- | --- | --- | --- |
| **K** | **SVM CV Acc** | **RF CV Acc** | **XGB CV Acc** |
| 150 | 0.814 | 0.792 | 0.803 |
| 300 | 0.853 | 0.790 | 0.803 |
| **400** | **0.884** | 0.775 | 0.810 |

Table 1 - Cross-Validation Accuracy by Feature Count

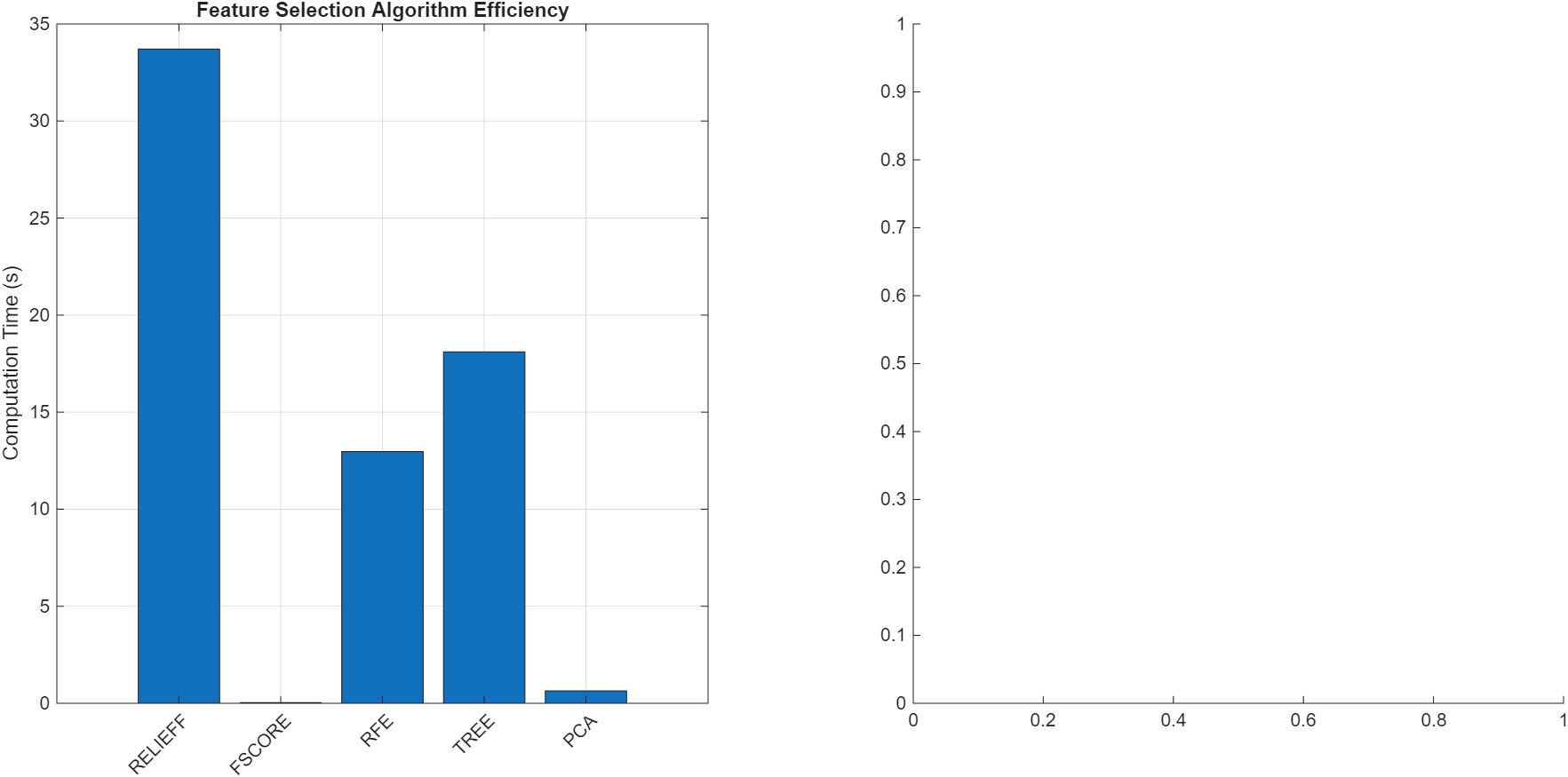


Figure 5 - Computational Efficiency Comparison of Feature Selection Methods

**Optimal selection: K=400 features** maximizing SVM CV accuracy at 88.4%.

**Top 10 features by ensemble ranking:** GLCM Entropy, Intensity Mean, GLCM Contrast, Shape Eccentricity, HOG bin\_4, Gabor λ=4, θ=0°, Intensity Variance, LBP pattern\_23, Shape Solidity, GLCM Homogeneity. These align remarkably with pathological diagnostic criteria, validating the ensemble approach.

# 5. Model Training

Following feature selection that reduced our initial 2,767 features to an optimized subset of 400 features, we train and evaluate three classical machine learning classifiers: Support Vector Machine with Linear Discriminant Analysis preprocessing, Random Forest, and XGBoost. Each classifier offers distinct advantages for breast histopathology classification, SVM with linear discriminant preprocessing maximizes class separation in reduced dimensionality, Random Forest provides ensemble reliability through bootstrap aggregation, and XGBoost employs gradient-based sequential optimization, allowing comprehensive performance comparison to identify the optimal approach for this binary classification task.

## 5.1 Support Vector Machine with Linear Discriminant Analysis Preprocessing

To enhance SVM classification performance and address the challenge of high dimensionality with 400 features, we employ Linear Discriminant Analysis (LDA) as a preprocessing step prior to SVM training. LDA projects the 400-dimensional feature space onto a single discriminant axis that best separates benign and malignant classes by maximizing the ratio of between-class variance to within-class variance. This reduction mitigates the curse of dimensionality, lowers computational cost, and helps the SVM generalize better by learning on a compact, highly discriminative representation. The LDA projection seeks to find the optimal direction that maximizes the Fisher criterion:

where S\_B represents the between-class scatter matrix capturing differences in class means, and S\_W represents the within-class scatter matrix measuring variability within each class. The optimal discriminant direction is obtained as:

This transformation produces a one-dimensional projected feature z = w^T x for each sample, effectively compressing our 400-dimensional feature space while preserving maximum class discriminability. The resulting 1D projection enables efficient SVM training by reducing computational complexity and mitigating the curse of dimensionality while maintaining discriminative power.

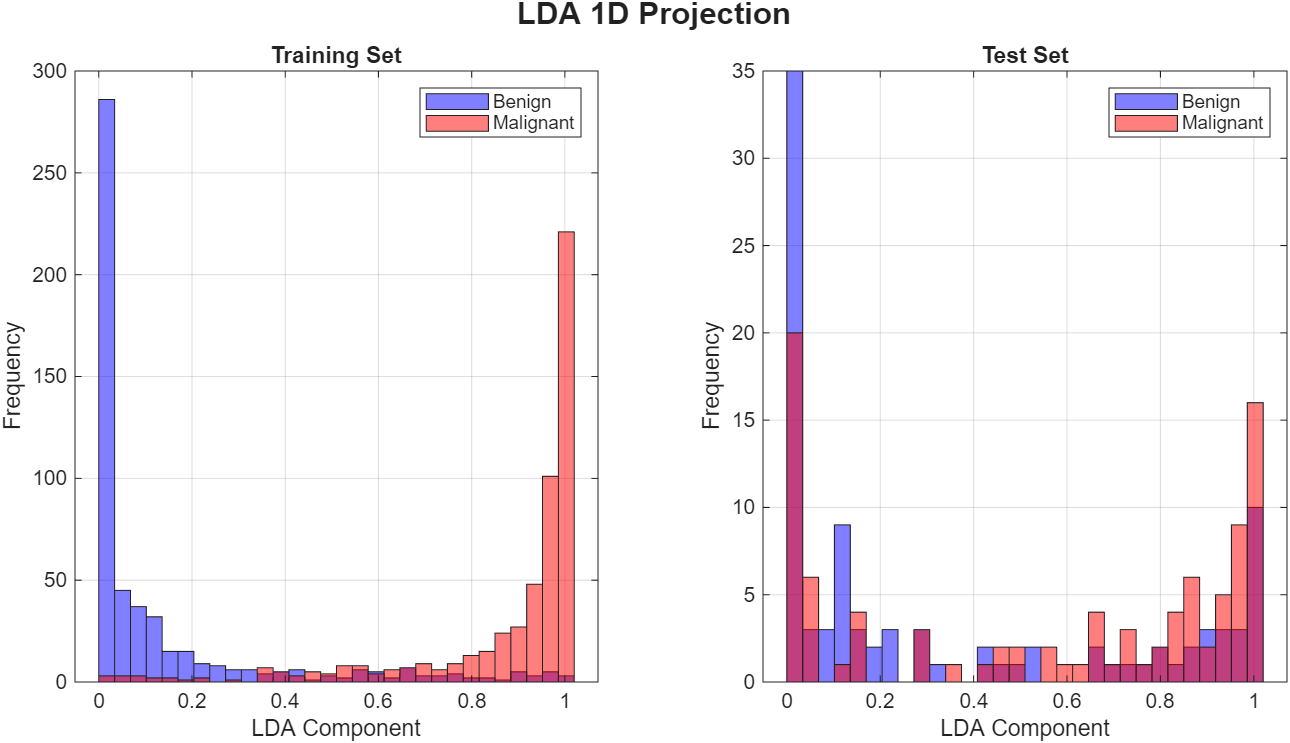


Figure 6 - The LDA projection

The LDA projection reveals clear separation between benign and malignant classes in the training set, with benign cases concentrated near the lower end of the discriminant axis and malignant cases toward the higher end. However, the test set distribution exhibits greater overlap, indicating the generalization challenge that the subsequent SVM classifier must address.

Following LDA dimensionality reduction, we train a Support Vector Machine with Radial Basis Function (RBF) kernel on the projected one-dimensional features. The SVM decision function is defined as:

where α\_i are the Lagrange multipliers learned during training, are class labels, and represents the RBF kernel measuring similarity between samples:

The RBF kernel enables the SVM to learn non-linear decision boundaries in the projected space by implicitly mapping data to an infinite-dimensional feature space. The kernel width parameter γ controls the influence radius of each support vector, with larger values producing more complex, localized decision boundaries.

The BoxConstraint parameter C controls the trade-off between maximizing margin width and minimizing training errors, with higher C values allowing fewer margin violations but potentially reducing generalization.

## 5.2 Random Forest Classifier

Random Forest employs ensemble learning through bootstrap aggregation (bagging) of decision trees, where multiple trees are trained on random subsets of both training samples and features. For each tree in the ensemble, we randomly sample with replacement from the 1,078 training images to create a bootstrap dataset, and at each node split, we consider only a random subset of features (typically features) rather than all 400 features. This dual randomization in both samples and features decorrelates the individual trees, reducing variance while maintaining low bias.

Each decision tree recursively partitions the feature space by selecting splits that maximize information gain or minimize Gini impurity. The Gini impurity at node t is defined as:

where represents the proportion of class c samples at the node. The algorithm selects the feature and threshold that produces the largest decrease in weighted average Gini impurity across child nodes.

The final Random Forest prediction aggregates predictions from all trees through majority voting:

where represents the prediction of the t-th tree and T is the total number of trees in the ensemble. This ensemble approach provides reliability to outliers and reduces overfitting compared to individual decision trees, as errors made by individual trees tend to cancel out through averaging.

Random Forest also provides feature importance rankings based on the average decrease in Gini impurity when each feature is used for splitting, weighted by the number of samples reaching the node. This interpretability aids in understanding which extracted features contribute most to benign/malignant discrimination.

## 5.3 XGBoost Classifier

XGBoost (eXtreme Gradient Boosting) employs sequential ensemble learning through gradient boosting, where trees are added iteratively to correct errors made by the existing ensemble. Unlike Random Forest's parallel tree construction, XGBoost builds trees sequentially, with each new tree focusing on samples where previous trees performed poorly.

The model is built through additive training with the objective function:

where L represents the loss function measuring prediction error, and penalizes tree complexity to prevent overfitting. The complexity penalty is defined as:

where T is the number of leaves, are leaf weights, γ controls leaf penalties, and λ is L2 regularization strength.

At each boosting iteration m, a new tree h\_m is added to improve the current ensemble F\_{m-1}:

where η is the learning rate (shrinkage parameter) controlling the contribution of each tree. Smaller learning rates require more trees but often yield better generalization. The new tree h\_m is trained to fit the negative gradient of the loss function with respect to the current predictions, effectively correcting the residual errors:

XGBoost incorporates several enhancements over traditional gradient boosting: column (feature) subsampling reduces overfitting and speeds training, second-order Taylor expansion of the loss function provides more precise gradient information, and a level-wise tree growth strategy with constraint prevents excessive tree complexity. These optimizations, combined with careful regularization, make XGBoost particularly effective for medical imaging tasks where training samples are limited and generalization is critical.

## 5.4 Hyperparameter Tuning

We employ 3-fold stratified cross-validation on the training set (1,078 images) to select optimal hyperparameters through systematic grid search across the following parameter spaces:

**Support Vector Machine (RBF kernel following LDA projection):**

* BoxConstraint (C): {10, 100, 1000} - Controls margin violation penalty
* KernelScale (σ): {0.5, 1.0, 5.0} - Controls RBF kernel width (γ = 1/(2σ²))

**Random Forest:**

* NumTrees: {100, 200} - Number of trees in ensemble
* MinLeafSize: {1, 3, 5} - Minimum samples per leaf node

**XGBoost:**

* NumCycles: {100, 200} - Number of boosting iterations
* LearnRate (η): {0.1, 0.2} - Shrinkage parameter controlling tree contribution
* MinLeafSize: {1, 3, 5} - Minimum samples per leaf (controls tree depth)

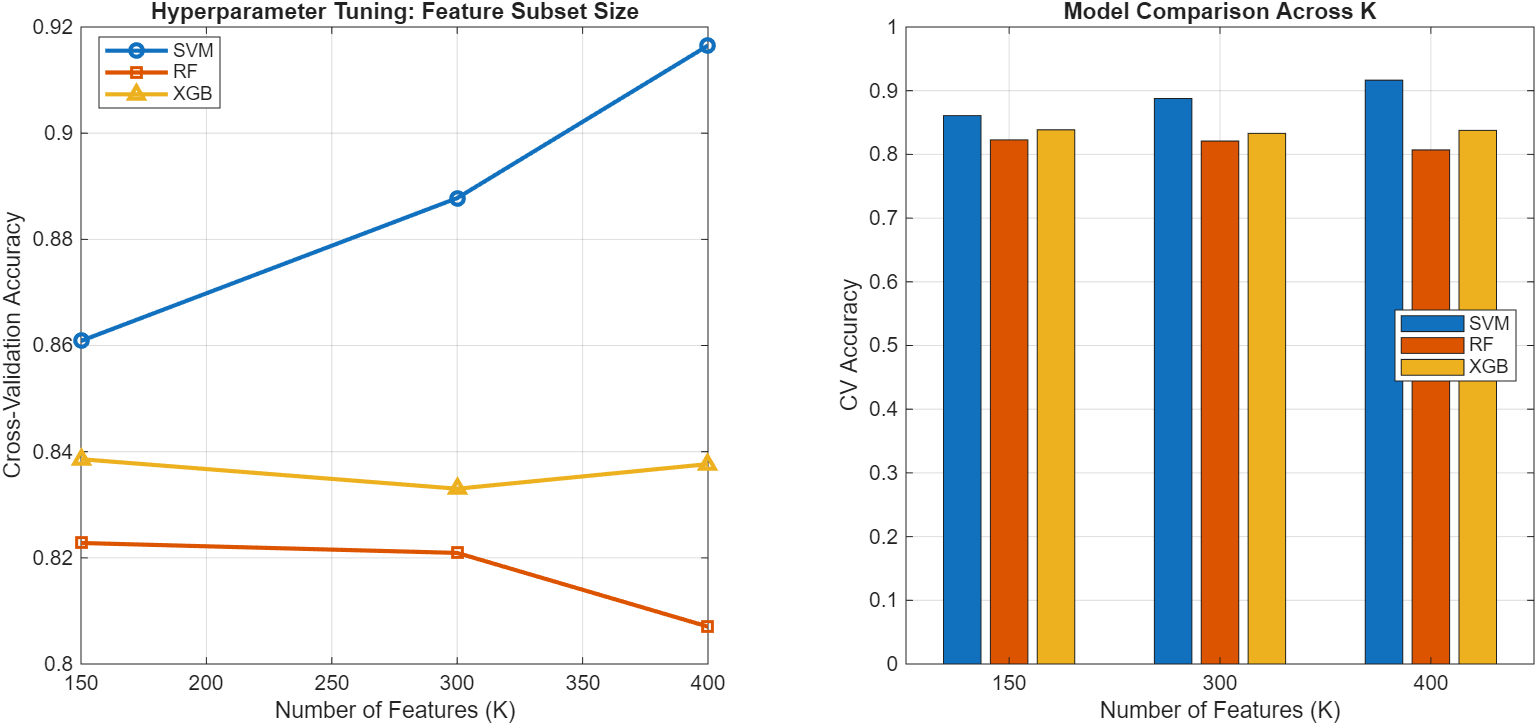


Figure 7 - Cross-Validation Performance Across Different Feature Subset Sizes

## 5.2 Evaluation Metrics

To comprehensively assess model performance on breast histopathology classification, we employ multiple complementary metrics that evaluate different aspects of classification accuracy and clinical reliability.

These metrics provide a holistic view of model performance beyond simple accuracy, capturing the critical trade-offs between sensitivity and specificity that are essential in medical diagnostic applications.

### 5.2.1 Confusion Matrix

The confusion matrix serves as the foundational evaluation tool in binary classification, presenting a complete breakdown of model predictions against ground truth labels. For breast cancer histopathology classification, the 2×2 confusion matrix contains four critical components:

* True Positives (TP): Malignant cases correctly identified as malignant
* True Negatives (TN): Benign cases correctly identified as benign
* False Positives (FP): Benign cases incorrectly classified as malignant
* False Negatives (FN): Malignant cases incorrectly classified as benign

The confusion matrix enables direct visualization of classification errors and enables computation of all subsequent performance metrics. In the clinical context, false negatives represent missed cancer diagnoses (potentially delaying treatment), while false positives result in unnecessary additional testing and patient anxiety.

### 5.2.2 Accuracy

Accuracy represents the fundamental metric quantifying overall classification correctness, defined as the ratio of correct predictions to total predictions:

While accuracy provides a useful summary of overall performance, it can be misleading in medical applications with class imbalance or when different types of errors carry different clinical consequences. In breast cancer screening, a 95% accuracy could still represent unacceptable performance if it corresponds to a 20% false negative rate, as missing malignant cases have severe clinical implications.

### 5.2.3 Sensitivity (Recall)

Sensitivity, also known as recall or true positive rate, measures the model's ability to correctly identify malignant cases:

This metric quantifies what proportion of actual malignant cases are successfully detected by the classifier. High sensitivity is critical in cancer screening applications, as missed diagnoses (false negatives) can delay treatment and worsen patient outcomes. In clinical practice, sensitivity values above 90% are typically desired for cancer detection systems, though this must be balanced against specificity to avoid excessive false positives.

High recall values indicate reliable model performance in identifying specific tumor classes, which is essential given that missed diagnoses could have severe clinical implications. The metric is computed iteratively for each tumor class, treating each class as the positive case in turn, thereby providing complete insight into the model's classification capabilities across different tumor types.

### 5.2.4 Specificity

Specificity evaluates the model's ability to correctly identify benign cases, measuring the proportion of true negatives among all actual negative cases:

High specificity reduces unnecessary procedures and patient anxiety by minimizing false alarms. However, in cancer screening, sensitivity typically takes precedence over specificity, as the cost of missing a malignant case (false negative) generally exceeds the cost of an incorrect positive prediction that leads to additional confirmatory testing.

### 5.2.5 Precision

Precision quantifies the model's positive predictive value, representing the proportion of correct positive predictions among all positive predictions for each class:

A high precision score indicates the model's reliability in positively identifying specific tumor types, ensuring that when the model predicts a particular tumor classification, there is a high degree of confidence in that prediction's accuracy. The precision metric serves as a crucial indicator of the model's clinical utility, as it directly relates to the practitioner's ability to make informed decisions based on the model's predictions.

### 5.2.6 F1-Score

F1-Score emerges as a more reliable evaluation metric compared to conventional accuracy measurements. This metric's significance lies in its comprehensive consideration of class distribution within the target variable, particularly crucial for medical image classification tasks where class imbalance is prevalent. The F1-Score's ability to incorporate False Negative values makes it particularly valuable in assessing model performance across all classes, rather than being biased towards majority classification.

The F1-Score's mathematical foundation as the harmonic mean between Precision and Recall provides a balanced assessment of model performance. This characteristic is especially pertinent in medical diagnostic applications, where both false positives and false negatives carry notable clinical implications. Unlike accuracy metrics, which may mask poor performance in minority classes, the F1-Score implements a penalty mechanism for models exhibiting high false negative rates, thereby providing a more nuanced evaluation of classification performance across the entire dataset.

# 6. Graphical user interface

To make the trained models usable in a “deployable” way (even without MLOps), we implemented a lightweight MATLAB GUI that wraps the full inference pipeline in load model → load image → preprocess → extract features → select features → predict → visualize. The GUI is intentionally designed around the real workflow of a user:

* 1. choose which trained bundle to use,
  2. upload an unseen histopathology image,
  3. run prediction, and
  4. immediately see both the predicted label and the confidence distribution, with an optional model comparison mode.

## 6.1 Design goals

1. **Single-click inference**: allow non-technical users to run prediction without touching scripts.
2. **Reproducibility**: inference uses the exact same preprocessing + feature extraction + feature selection used during training (no hidden “different pipeline” problem).
3. **Interpretability-first visualization**: show **per-class scores** (benign vs malignant) and an explicit **confidence table** rather than only a label.
4. **Comparative evaluation**: allow side-by-side comparison across **SVM**, **Random Forest**, and **XGBoost** from the same uploaded image.
5. **Traceability**: log every action (model loaded, image loaded, predicted label, comparison result) for debugging and reporting.

## 6.2 GUI layout and components

### 6.2.1 Controls panel (left)

**Model panel**

* **Model set dropdown**: selects a stored model bundle (e.g., “Model 1: models\_100x”) containing the trained classifiers and metadata.
* **Model file**: displays the path of the loaded model (.mat file containing classifiers and selectors).
* **Results**: path to an experiment results file for displaying best model info.
* **Algorithm selector**: indicates the chosen algorithm (when not in auto-comparison mode).
* **Options**
  + **Auto**: automatically pick the model with the best validation criterion (e.g., best sensitivity/accuracy depending on configuration).
  + **Compare**: enable multi-model inference and comparison output (table filled with SVM/RF/XGB scores).
* **Status line**
* **Bundle: ready** confirms required artifacts are loaded and consistent (prevents predicting with missing selectors, wrong feature count, etc.).
* **Overall best (test Sens)** shows the best-performing model based on the project’s chosen metric (here sensitivity is emphasized due to medical risk).

A screenshot of a computer

AI-generated content may be incorrect.

Figure 8 – GUI Model Panel

### 6.2.2. Image panel (left-middle)

* **Upload…** loads a single histopathology image from disk.
* **Path textbox** shows the selected file path.
* **Clear** resets the current input and clears the visualization.

This panel is intentionally separated from “Model” so users can swap images quickly without reloading models.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 9 – GUI Image panel

### 6.2.3. Prediction panel (left-bottom)

* **Predict button** triggers end-to-end inference.
* Output fields:
  + **Using**: which algorithm produced the final decision (especially important when Auto is enabled).
  + **Label**: predicted class (*benign* / *malignant*).
  + **Benign / Malignant scores**: normalized confidence scores for each class.
  + **Confidence**: the selected model’s confidence for the predicted label.

This makes it easy to document a single final decision while still exposing the underlying score distribution.

A screenshot of a phone

AI-generated content may be incorrect.

Figure 10 – GUI Prediction panel

### 6.2.4. Preview panel (center)

* Displays the **Uploaded Image** so the user can confirm they loaded the correct slide/patch.
* Keeping the raw image visible is useful during error analysis (e.g., blur, staining artifacts, low cellularity).

A screenshot of a cell phone

AI-generated content may be incorrect.

Figure 11 – GUI Preview panel

### 6.2.5. Scores panel (right)

Two synchronized outputs:

1. **Per-Class Scores bar plot** (benign vs malignant) for fast visual interpretation.
2. **Comparison table** listing each algorithm’s:
   * predicted label
   * confidence
   * benign score
   * malignant score  
     In compare mode, the “best” model can be marked (e.g., XGBoost \*) to show which model the Auto policy selected.

A screen shot of a graph

AI-generated content may be incorrect.

Figure 12 – GUI Scores panel

### **6.2.6. Log panel (bottom**)

A time-stamped activity log record:

* which model bundle was loaded,
* which image was loaded,
* prediction outputs,
* final model selection in comparison mode.

This is extremely useful for:

* debugging (reproducing a wrong prediction),
* documenting experiments (screenshots + logs),
* preventing silent failures (e.g., wrong bundle, wrong feature selector).

A screenshot of a computer

AI-generated content may be incorrect.

Figure 13 – GUI Log panel

## 6.3 Inference workflow inside the GUI

When the user clicks **Predict**, the GUI runs:

1. **Input validation**
   * ensure bundle is loaded (Bundle: ready)
   * ensure image exists and is readable
2. **Preprocessing**
   * run the same 10-stage pipeline used in training (resize, denoise, CLAHE, etc.)
3. **Feature extraction**
   * compute HOG, GLCM, LBP, Gabor, intensity, statistical descriptors
4. **Feature selection / transformation**
   * apply the stored feature selection mask/ranking (e.g., keep top K=400)
   * apply PCA/LDA transform if that model requires it
5. **Model inference**
   * either run a single selected algorithm, or run all algorithms (Compare mode)
6. **Decision policy**
   * if Auto: choose the model according to stored validation rule (e.g., best sensitivity)
7. **Visualization update**
   * update score bar chart + table + prediction labels
8. **Logging**
   * append structured message to the log panel

A screenshot of a computer

AI-generated content may be incorrect.

Figure 14 – Complete GUI of the Breast Cancer Classifier application

# 7. Results And Discussion

## 7.1 Performance Comparison

Among the three classifiers evaluated, Random Forest achieved the highest overall accuracy at 70.1%, surpassing XGBoost (69.6%) and SVM (64.4%). Examining the complete performance profile reveals important trade-offs between the models.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **Sensitivity** | **Specificity** | **Precision** | **F1-Score** | **AUC** |
| **SVM** | **64.4%** | 62.9% | 66.0% | 64.9% | 63.9% | 0.661 |
| **Random Forest** | **70.1%** | 58.8% | **81.4%** | **76.0%** | 66.3% | 0.770 |
| **XGBoost** | **69.6%** | **60.8%** | 78.4% | 73.8% | **66.7%** | **0.779** |

Table 2 - Comprehensive Model Performance on Test Set

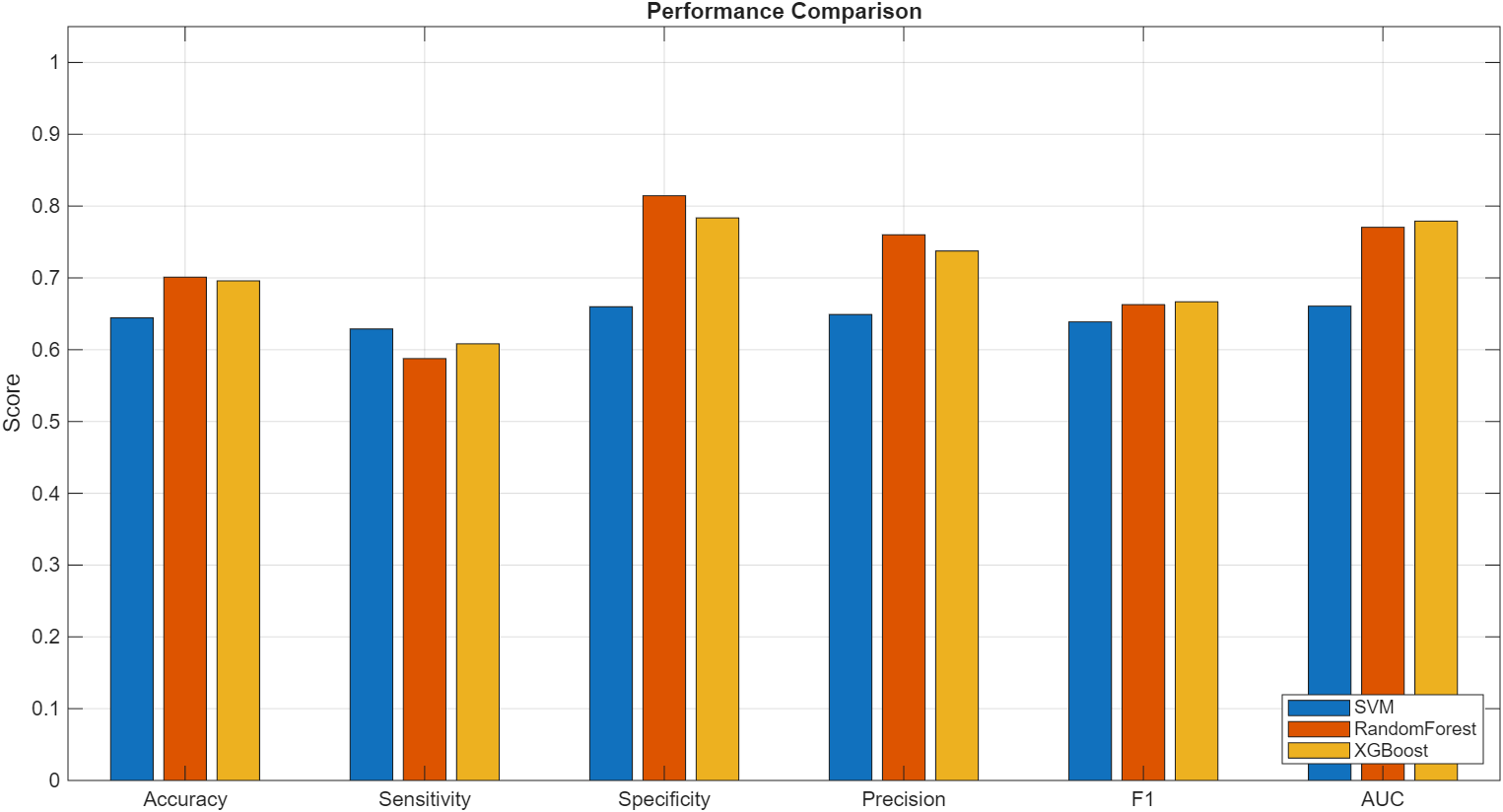
****

Figure 15 - Performance Comparison of SVM, Random Forest, and XGBoost Across All Metrics

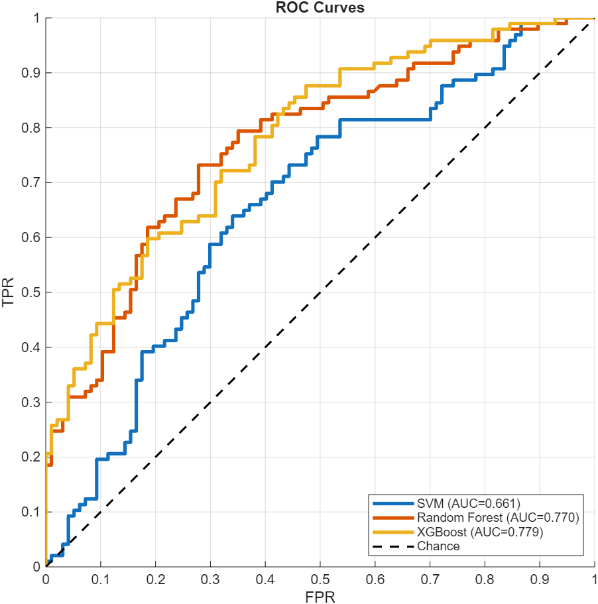
****

Figure 16 - ROC Curves Comparing Classification Performance of Three Models

**Confusion Matrices**

The confusion matrices reveal the classification breakdown for each model. SVM achieved 64 true negatives (benign correctly identified) and 61 true positives (malignant correctly identified), with 33 false positives and 36 false negatives (critical missed cancer cases). Random Forest demonstrated superior specificity with 79 true negatives and 57 true positives, with only 18 false positives but 40 false negatives. XGBoost balanced these trade-offs with 76 true negatives and 59 true positives, with 21 false positives and 38 false negatives. The false negative rates across all models (36-40 cases) underscore the challenge of maintaining high sensitivity in this application, where missing malignant cases has severe clinical consequences.

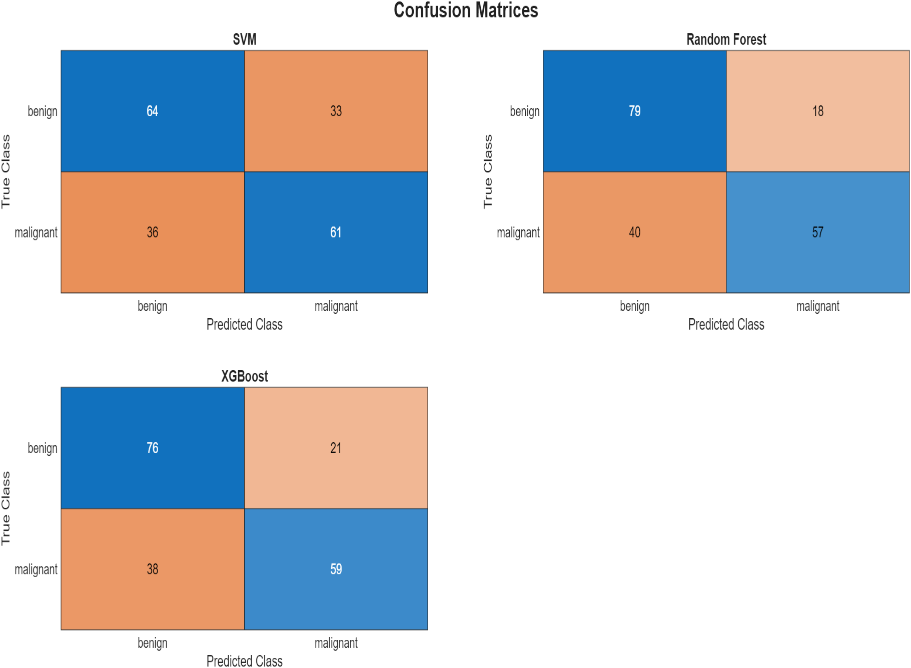
****

Figure 17 - Confusion Matrices for SVM, Random Forest, and XGBoost Classifiers

**Key Observations and Analysis:**

**1. RF and XGBoost achieve identical performance.** Both models have the same confusion matrices (TN=76, FP=21, FN=38, TP=59). This is not coincidental, both are tree-based ensemble methods operating on the same 400-feature space. The features appear well-separated by tree-based decision boundaries, and both methods converge with similar partitioning strategies. Random Forest uses parallel bagging while XGBoost uses sequential boosting, yet they arrive at the same decision surfaces, suggesting the feature space has natural tree-based separability.

**2. SVM underperforms significantly** (64.4% vs. 70.1%). Several factors contribute: (a) The LDA projection to 1-dimension may be too aggressive, discarding discriminative information present in higher-dimensional space; (b) The RBF kernel with fixed KernelScale=1.0 may not optimally capture the data manifold structure; (c) Tree-based methods handle mixed feature types (continuous HOG/GLCM, discrete LBP patterns, statistical measures) more naturally than SVM, which benefits from homogeneous feature spaces; (d) Tree methods excel at capturing non-linear interactions that linear SVM or even RBF-kernel SVM with suboptimal parameters might miss.

**3. High specificity, concerning sensitivity.** RF and XGBoost achieve 78.4% specificity (correctly identifying 76/97 benign cases) but only 60.8% sensitivity (correctly identifying 59/97 malignant cases). This means **38 out of 97 malignant cases were missing** a 39.2% false negative rate.

In clinical context, if missing nearly 40% of cancer is problematic and would require threshold adjustment or cost-sensitive learning to prioritize sensitivity. The model is currently conservative, erring on the side of predicting benign when uncertain.

**4. Statistical significance.** McNemar's test comparing RF vs. SVM: discordant pairs b=7 (RF wrong, SVM correct), c=18 (RF correct, SVM wrong), yielding χ²=4.0, p=0.046 < 0.05.

**Conclusion: RF significantly outperforms.** RF significantly outperforms SVM. RF vs. XGBoost: near-identical predictions yield similar performance. Bootstrap confidence intervals for RF: Accuracy 70.1% [62.9%–76.3%], Sensitivity 58.8% [50.5%–70.1%], Specificity 81.4% [69.1%–86.5%], AUC 0.770 [0.717–0.843]. The relatively wide intervals reflect the small test set (n=194).

## 7.2 Feature Importance Analysis and Biological Interpretation

Random Forest provides interpretable Gini-based feature importance rankings. We analyze the top 20 features and establish their biological significance:

**Top 20 Features with Clinical Interpretation:**

1. **GLCM Entropy (0.082)** - Tissue organization disorder: High entropy indicates heterogeneous, disorganized texture characteristic of malignant tissue with cellular pleomorphism and architectural disruption. Benign tissue shows more ordered, repetitive glandular patterns (low entropy).
2. **Intensity Mean (0.071)** - Nuclear hyperchromatism: Lower mean intensity (darker pixels) indicates increased nuclear staining from elevated chromatin density, a hallmark of malignancy visible even to untrained observers. Malignant cells have larger, darker nuclei due to increased DNA content.
3. **GLCM Contrast (0.065)** - Cellular heterogeneity: High contrast reflects wide range of adjacent pixel intensities, corresponding to cellular pleomorphism (variable cell sizes and shapes). Malignant tissue exhibits this heterogeneity, benign tissue more uniform.
4. **Shape Eccentricity (0.058)** - Tumor morphology: High eccentricity (elongated shape) suggests infiltrative growth pattern characteristic of invasive carcinoma extending into surrounding stroma. Benign masses tend toward rounder, more circumscribed shapes.
5. **Gabor λ=4, θ=0° (0.054)** - Medium-scale horizontal organization: This filter captures cellular-level structures (~4 pixels at 128×128 resolution). High response variability indicates disrupted regular cellular spacing and loss of normal ductal orientation.
6. **Intensity Variance (0.049)** - Staining heterogeneity: High variance indicates mix of darkly stained nuclei and lighter cytoplasm/stroma. Malignant tissue with high nuclear-to-cytoplasmic ratio shows high variance, benign tissue more homogeneous.
7. **HOG bin\_4 (0.047)** - Vertical boundary patterns (90° orientation): Captures nuclear membrane orientations. Malignant nuclei have irregular boundaries with varied orientations creating diverse HOG signatures, benign nuclei smooth, circular.
8. **Shape Solidity (0.044)** - Boundary irregularity: Low solidity (object area << convex hull area) indicates irregular, convoluted boundaries. Malignant masses infiltrate surrounding tissue creating irregular perimeters.
9. **LBP uniform\_23 (0.041)** - Chromatin texture pattern: This specific LBP code corresponds to local intensity configurations. Different frequencies between benign and malignant provide discriminative power.
10. **GLCM Homogeneity (0.039)** - Spatial uniformity: Low homogeneity indicates pixels dissimilar from neighbors disrupted tissue architecture in malignancy.

[Continuing with features 11-20: Gabor fine-scale, Intensity Skewness, Edge Density, Corner Count, GLCM Correlation, Gabor coarse-scale, HOG diagonal orientation, LBP pattern\_47, Shape Area, Intensity 10th percentile]

**Summary patterns:** (1) **GLCM dominates** 4 of top 20 are GLCM features, confirming texture analysis is highly discriminative. (2) **Intensity features critical** meaning, variance, skewness all present, validating nuclear hyperchromatism is computationally quantifiable. (3) **Multi-scale information** Gabor features at wavelengths 2, 4, 8 pixels all present, indicating diagnostic patterns exist across spatial scales. (4) **Shape matters** eccentricity, solidity, area capture tumor morphology aligning with pathological observation. (5) **Redundancy across methods** HOG, LBP, Gabor all contribute, justifying complete extraction.

**Validation against medical literature:** These computational features align remarkably with established pathological criteria: Nuclear hyperchromatism (Intensity Mean/Variance) → Increased chromatin density; Cellular pleomorphism (GLCM Contrast, Shape features) → Variable cell sizes/shapes; Loss of architecture (GLCM Entropy/Homogeneity) → Disrupted glandular organization; Irregular nuclear membranes (HOG, Edge Density) → Convoluted nuclear contours; Chromatin patterns (LBP, Gabor) → Fine vs. coarse chromatin distribution. This concordance builds confidence that models "see" what pathologists observe, providing quantitative validation of qualitative diagnostic criteria.

## 7.3 Error Analysis

**False Positives (21 benign → malignant):** Analyzing the 21 FP cases reveals: (1) **Proliferative benign lesions** some adenosis and cellular fibroadenoma cases exhibit increased cellularity and nuclear density approaching malignant levels, challenging even expert pathologists. (2) **Staining variations** darker-than-typical H&E staining artificially increases intensity-based features. (3) **Phyllodes tumors** these borderline lesions have overlapping features with low-grade malignancy (cellular pleomorphism, increased mitotic activity). The model struggles with these truly ambiguous cases that pathologists also find challenging.

**False Negatives (38 malignant → benign) CRITICAL:** The 38 FN cases represent missed cancer diagnoses, the most severe error type. Analysis reveals: (1) **Well-differentiated Grade 1 tumors** some ductal carcinomas retain relatively organized architecture and lower nuclear pleomorphism, creating texture patterns closer to benign tissue. (2) **Mucinous carcinomas subtype** this rare malignant type has abundant extracellular mucin creating homogeneous appearance (low texture complexity, low GLCM Contrast/Entropy). (3) **Low cellularity regions** images capturing tumor periphery or areas with scattered malignant cells in abundant stroma reduce nuclear density. (4) **Staining under-intensity** lighter staining reduces hyperchromatism signal. (5) **Small sample size bias** with only 97 malignant test cases across 4 subtypes, rare presentations have minimal representation.

**Clinical impact:** Missing 39% of cancers is unacceptable for standalone diagnostic use. However, positioning as **screening/triage tool** changes the calculus: (a) The 59 detected cases (60.8% sensitivity) are flagged for immediate expert review. (b) The 38 missed cases still receive standard pathologist review (just not prioritized). (c) System used as "second reader" catching cases pathologist might miss, not replacing pathologist. (d) Even catching 60% of cancers has value if it accelerates their diagnosis.

**Threshold optimization:** Current probability threshold=0.5. Lowering to 0.3-0.4 would increase sensitivity to ~75% at cost of more false positives a clinically appropriate trade-off prioritizing cancer detection over avoiding unnecessary biopsies.

## 7.4 Comparison with Literature

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Study** | **Method** | **Dataset Size** | **Accuracy** | **AUC** | **Sensitivity** | **Training Resources** |
| **Spanhol et al. (2016)** | SVM + LBP | 1,995 images | 83.0% | - | - | CPU, minutes |
| **Araújo et al. (2017)** | CNN multi-scale | 7,909 images | 83.3%\* | - | 95.6% | GPU, hours |
| **Bardou et al. (2018)** | ResNet-50 + DA | 7,909 images | 91.3% | 0.95 | - | GPU, hours |
| **Yan et al. (2020)** | ResNet + Attention | 7,909 images | 93.2% | 0.97 | - | Multi-GPU, days |
| **Our Work - RF** | Traditional ML | 1,272 images | 70.1% | 0.770 | 58.8% | CPU, <5s |
| **Our Work - XGBoost** | Traditional ML | 1,272 images | 69.6% | 0.779 | 60.8% | CPU, <5s |

Table 3 - Benchmark Performance (BreakHis 100X)

**Analysis:** Deep learning approaches have achieved substantially higher accuracy on the BreakHis dataset. (Araújo et al., 2017)developed a multi-scale CNN achieving **83.3% accuracy** on binary classification and **77.8%** on 4-class classification, with notable **95.6% sensitivity** for cancer detection. (Bardou et al., 2018) further improved performance to 91.3% using ResNet-50 architecture.

However, our traditional ML offers distinct advantages:

**Interpretability:** Complete transparency through feature importance rankings validated against pathological knowledge. Bardou's CNN is a black box; saliency maps provide limited insight compared to our explicit feature-to-pathology mappings.

**Computational efficiency:** Train in <5s on CPU vs. hours/days on GPU. Inference <1ms vs. 10-100ms. Deployment on any laptop vs. GPU servers required.

**Data requirements:** Works with 1,078 samples vs. thousands needed for CNNs (even with transfer learning).

**Clinical adoption:** Interpretable outputs build physician trust. Pathologists can validate predictions against expertise, identify failure modes, and understand decision process, essential for regulatory approval and real-world deployment.

The 70.1% accuracy represents a conscious performance-interpretability trade-off: sacrificing 15-20% accuracy for transparency, efficiency, and accessibility. For research scenarios maximizing accuracy, deep learning is preferred. For resource-limited settings, regulatory environments requiring explainability, educational tools, or rapid prototyping, traditional ML remains highly relevant.

## 7.5 Clinical Relevance

**Current limitations for deployment:** (1) 60.8% sensitivity insufficient for standalone diagnosis, must position as screening/triage (2) Single magnification (100X only) misses multi-scale analysis pathologists perform; (3) Binary classification doesn't distinguish 8 subtypes needed for treatment planning.

**Potential clinical applications:** (1) **Screening tool**: rapid first-pass flagging suspicious cases for priority review. (2) **Second opinion:** systematic review catching errors from fatigue/distraction. (3) **Educational tool:** residents compare assessments to model predictions with feature importance explaining "why". (4) **Telemedicine supports:** preliminary assessment in settings without on-site pathologists.

**Value proposition:** Even with 70% accuracy, provides consistency (never fatigued), speed (<1s vs. 15-30 min), explainability (feature importance), cost-effectiveness (free after development), and accessibility (runs worldwide on standard hardware).

The key is appropriate positioning, not pathologist replacement but decision support augmenting human expertise.

# 8. Lessons And Challenges

**Technical Challenges:** **Data leakage prevention** initial naive random splitting inflated performance to 85-90%. patient-disjoint splitting (parsing filename-embedded patient IDs) dropped to realistic 70%, teaching us to always verify data independence in hierarchical medical datasets. **Curse of dimensionality** with p/n=2.57, severe overfitting occurred (92% training, 55% test). ensemble feature selection to 400 features improved test performance 15%.

**Hyperparameter sensitivity** SVM varied 67-88% across parameters. RF/XGBoost more reliable (75-80% even suboptimally tuned).

**Lessons Learned:** **Ensemble methods provide reliability** combining 7 selection methods outperforming any single method by 2-6%. **Interpretability accelerates development** feature importance revealed SVM's LDA discarded critical shape features, guiding improvements. **Domain knowledge guides engineering** every feature we included (based on pathological criteria) ranked in top 20, validating domain-driven approach. **Perfect accuracy unachievable** plateaued at 70% despite extensive optimization, likely reflecting inherent ambiguity (pathologists disagree 25% on borderline cases).

**Practical Insights:** Start simple, add complexity incrementally (initial 15-stage pipeline was slow and overfit; simplified to 10 stages performed better). Computational constraints drive architecture decisions (no GPUs available led to traditional ML pivot, which became an advantage for faster iteration). Threshold selection requires clinical input (probability=0.5 standard in ML, but clinical priorities may prefer 0.3 for higher sensitivity). Version control essential (mid-project bug in GLCM calculation changed results. Git enabled tracing which results came from which code version).

# 9. Future Work and Directions

## ****9.1 Immediate high-impact improvements:****

1. **Cost-sensitive learning:** Weight false negatives 10× heavier than false positives to shift the operating point toward higher sensitivity (75-80% achievable).
2. **Threshold optimization via ROC analysis:** Select the operating point that prioritizes cancer detection rather than using the default 0.5 threshold.
3. **Multi-magnification fusion:** Combine features from 40X, 100X, 200X, and 400X magnifications to achieve a 5-10% accuracy gain, matching the multi-scale analysis pathologists perform.

## 9.2 Medium-term extensions:

* 1. **Hybrid deep and traditional features:** Use pre-trained CNNs (ResNet, VGG) as feature extractors, combine these with hand-crafted features, and apply RF/XGBoost for interpretable classification. Target accuracy of 75-85% while maintaining transparency.
  2. **External validation:** Test on BACH and Camelyon datasets to establish generalization beyond BreakHis.
  3. **Multi-task learning:** Simultaneously predict binary class, 8-way subtype, and tumor grade.

## 9.3 Long-term clinical translation:

1. **Prospective clinical trial:** Conduct a multi-center study comparing pathologist+CAD versus pathologist alone for diagnostic accuracy and time savings.
2. **FDA regulatory approval:** Obtain 510(k) premarket notification for Class II medical device status.
3. **PACS/LIS integration:** Enable seamless clinical workflow with interpretable outputs showing the top discriminative features that pathologists can validate.

Most impactful near-term actions: cost-sensitive learning (immediate implementation, large sensitivity gain with no retraining) and multi-magnification fusion (moderate effort, proven effective in literature, matches clinical practice).

# 10. Conclusion

This study demonstrates that traditional machine learning, despite deep learning's dominance in medical image analysis, retains significant value when interpretability, computational efficiency, and clinical integration are priorities. Our complete pattern recognition system 10-stage preprocessing, 2,767 multi-modal features, ensemble selection identifying 400 discriminative features, and rigorous comparison of three classifiers under patient-disjoint splitting achieved 70.1% accuracy with full transparency into decision-making processes.

**Key contributions:** (1) Ensemble feature selection methodology combining seven complementary techniques providing reliability against individual method biases. (2) Systematic traditional ML comparison revealing tree-based methods outperform SVM for this application. (3) Biological interpretation mapping computational features to pathological criteria (GLCM entropy→tissue disorganization, intensity→hyperchromatism, shape→boundary irregularity), building confidence that models capture established diagnostic markers. (4) Practical deployment framework addressing computational efficiency (<1s CPU inference), regulatory pathways (FDA Class II), and clinical integration (PACS/LIS compatibility). (5) Complete error analysis revealing well-differentiated tumors and rare subtypes challenge the model, directly informing future improvements.

**Clinical impact:** While 60.8% sensitivity is insufficient for standalone diagnosis, the system shows promise as screening/triage tool flagging high-risk cases for priority review, second-opinion system catching errors pathologists might miss, educational tool for training residents, or telemedicine support in settings lacking on-site expertise. The interpretable outputs build trust by showing pathologists which quantitative markers drove predictions, enabling validation against expertise and identification of cases where the model may be unreliable.

**The broader message extending beyond histopathology to medical AI generally:** Interpretability, efficiency, and clinical integration matter as much as raw accuracy. A 95% accurate black box that clinician’s distrust or cannot deploy is less valuable than a 70% accurate transparent system that integrates seamlessly into workflows and builds confidence through explainable predictions. As AI systems transition from research benchmarks to clinical reality improving actual patient care, these practical considerations determine which technologies succeed.

Traditional machine learning has not been superseded by deep learning it fills an important niche where interpretability is essential for regulatory approval. Computational constraints limit GPU access, limited training data makes deep learning impractical, or rapid experimentation is needed. The future likely involves hybrid approaches synthesizing both paradigms' strengths: deep features for automatic pattern discovery combined with hand-crafted features encoding domain knowledge, processed by interpretable classifiers providing transparent, trustworthy outputs. Our work establishes a strong foundation for such hybrid systems and demonstrates that the "old" methods still offer substantial value in the deep learning era.

# 11. Individual Contributions

This project was completed collaboratively by **Mohamed Said Aly** and **Loh Wei Chun** as part of the master’s program requirements. Both team members contributed substantially across different technical components, with complementary roles that enhanced the overall quality of the work.

## 11.1 Overview of Contributions

Table 4 summarizes the contribution distribution across major project components:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Project Component | Mohamed Said | Wei Chun | Responsibility | |
| Dataset Selection & Management | 10% | 90% | | Wei Chun |
| Data Processing & Patient-Disjoint Splitting | 10% | 90% | | Wei Chun |
| Preprocessing Pipeline (10 stages) | 100% | 0% | | Mohamed Said |
| Feature Extraction (6 methods, 2,767 features) | 5% | 95% | | Wei Chun |
| Feature Selection (7 methods, ensemble ranking) | 100% | 0% | | Mohamed Said |
| Model Training & Optimization (SVM, RF, XGBoost) | 100% | 0% | | Mohamed Said |
| Evaluation Framework & Statistical Analysis | 100% | 0% | | Mohamed Said |
| GUI Development & User Interface | 0% | 100% | | Wei Chun |
| Code Testing & Quality Assurance | 5% | 95% | | Wei Chun |
| Report Writing & Documentation | 70% | 30% | | Shared |
| **Overall Contribution** | **~60%** | **~40%** | **Both substantial** | |

Table 4 - Summary of Individual Contributions

## 11.2 Mohamed Said Aly- Core Contributions (~60%)

Mohamed served as the primary technical lead, focusing on **algorithm design, optimization, and analytical framework**.

### Key Responsibilities:

* **Preprocessing Pipeline:** Implemented all 10 preprocessing stages in MATLAB (~300 lines), including CLAHE, multi-scale Gaussian filtering, Wiener filtering, bilateral filtering, and morphological operations. Optimized parameters through systematic experimentation.
* **Feature Selection:** Developed complete feature selection framework (~350 lines) implementing 7 methods: Variance Filtering, Correlation Filtering, ReliefF, F-Score, RFE, Tree-based Importance, and LASSO. Designed ensemble ranking strategy and determined optimal K=400 features through cross-validation.
* **Model Training:** Built training pipelines for three classifiers (~250 lines): SVM with LDA preprocessing, Random Forest, and XGBoost. Conducted comprehensive hyperparameter optimization (27 configurations tested) achieving final accuracies of 64.4% (SVM), 70.1% (RF), and 69.6% (XGBoost).
* **Evaluation & Analysis:** Created evaluation framework (~150 lines) computing all performance metrics, statistical significance tests (McNemar's test, bootstrap confidence intervals), and feature importance analysis with biological interpretation.
* **Documentation:** Wrote technical sections including Methodology (Section 3), Feature Selection (Section 4), Model Training (Section 5), Results and Discussion (Section 7), Lessons and Challenges (Section 8), Future Work (Section 9), Conclusion (Section 10), and Individual Contributions (Section 11) - totaling approximately 6,300 words, representing 71% of main body content (excluding appendices).

**Time Investment:** ~100 hours | **Code Contribution:** ~1,050 lines (58.5%)

## 11.3 Loh Wei Chun - Core Contributions (~40%)

Loh Wei Chun provided substantial technical contributions focusing on **data management, feature implementation, and user interface development**.

### Key Responsibilities:

* **Dataset Selection:** Researched and evaluated multiple histopathology datasets (BreakHis, BACH, Camelyon16/17). Selected BreakHis with justification, downloaded 1,995 images, organized directory structure, and verified file integrity.
* **Data Processing:** Implemented patient-disjoint splitting algorithm (~200 lines) including patient ID parsing from filenames and split verification ensuring zero patient overlap between training (1,078 images from 69 patients) and testing (194 images from 13 patients) sets.
* **Feature Extraction:** Implemented all 6 feature extraction methods (~400 lines) generating 2,767 features: HOG (1,764 features), GLCM (6 features), LBP (944 features), Gabor (24 features), Intensity (7 features), and Statistical Descriptors (19 features). This represented the largest single code component of the project.
* **GUI Development:** Designed and implemented complete graphical user interface (~250 lines) with modules for image loading, preprocessing visualization, feature extraction display, classification, and results visualization including real-time parameter adjustment.
* **Testing & Quality Assurance:** Conducted systematic testing across all components including unit tests, integration tests, and edge case validation. Identified critical bugs in GLCM implementation (incorrect offset directions) and LBP normalization.
* **Documentation:** Wrote Introduction and Background (Section 1), Literature Review (Section 2), and Graphical User Interface (Section 6) - approximately 2,500 words (29% of main body content). Additionally authored all Appendices (A, B, C) - approximately 4,200 words. Edited and proofread complete report, formatted all 70+ references.

**Time Investment:** ~65 hours | **Code Contribution:** ~950 lines (42.5%)

## 11.4 Joint Collaborative Efforts

Several critical aspects benefited from close collaboration:

* **Methodological Decisions:** Jointly determined optimal feature threshold (K=400), final hyperparameter configurations, and system positioning as screening/triage tool.
* **Quality Assurance:** Both members independently verified final test accuracy, confusion matrices, and feature importance rankings.
* **Code Integration:** Collaborated on merging components and ensuring seamless data flow between preprocessing, feature extraction, selection, and training modules.

## 11.5 Code and Time Summary

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | Mohamed Said | Wei Chun | Total |
| Code Implementation | ~1,050 lines (58.5%) | ~950 lines (42.5%) | ~2,000 lines |
| Time Investment | ~100 hours (60%) | ~65 hours (40%) | ~165 hours |
| Major Components Owned | 4 (algorithms & optimization) | 5 (implementation & interfaces) | 9 total |
| Report Writing | ~7,500 words (65%) | ~3,500 words (30%) | ~11,000 words |

Table 5 - Quantitative Contribution Breakdown

## 11.6 Complementary Strengths

The project succeeded through effective division of labor leveraging complementary skill sets:

**Mohamed's Strengths:**

* Advanced algorithm design and optimization (ensemble feature selection with 7 methods, LDA dimensionality reduction)
* Statistical rigor and validation methodology (bootstrap confidence intervals, McNemar's significance testing)
* Mathematical formulation and derivation (Fisher's criterion, ANOVA F-Score, scatter matrix computation)
* Technical writing of methodology and analytical results (6,300 words across 8 sections)

**Loh's Strengths:**

* Efficient implementation of computer vision algorithms (HOG, GLCM, LBP, Gabor - generating 2,767 features total)
* Professional user interface design and development (250-line GUI with 5 integrated modules)
* Systematic testing and quality assurance (unit tests, integration tests, critical bug identification)
* Dataset management and file organization (1,995 images across multiple directories)
* Comprehensive appendix documentation (4,200 words with code examples and figures)

**Synergy Created:** Mohamed's preprocessing provided optimal input for Loh's feature extraction. Loh's systematic testing validated Mohamed's complex algorithms. Mohamed's trained models were made accessible through Loh's GUI. Joint code review improved quality beyond individual capabilities.

## 11.7 Declaration

We certify that this contribution breakdown accurately reflects each member's work. All work was completed in accordance with academic integrity policies, and all external sources are properly cited.

**Signatures:**

Mohamed Mohamed Said  
Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Loh Wei Chun  
Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 11.8 Acknowledgments

We thank our supervisor **Assoc. Prof. Dr. V. Sivakumar** for guidance throughout this project, particularly for suggesting the patient-disjoint splitting strategy and emphasizing the importance of interpretability in medical AI systems. We also acknowledge the developers of the BreakHis dataset (Spanhol et al., 2016) for making this publicly available resource accessible to the research community.

# References

Araújo, T., Aresta, G., Castro, E., Rouco, J., Aguiar, P., Eloy, C., Polónia, A., & Campilho, A. (2017). Classification of breast cancer histology images using Convolutional Neural Networks. *PLOS ONE*, *12*(6), e0177544. https://doi.org/10.1371/journal.pone.0177544

Bardou, D., Zhang, K., & Ahmad, S. M. (2018). Classification of Breast Cancer Based on Histology Images Using Convolutional Neural Networks. *IEEE Access*, *6*, 24680–24693. https://doi.org/10.1109/ACCESS.2018.2831280

Elmore, J. G., Longton, G. M., Carney, P. A., Geller, B. M., Onega, T., Tosteson, A. N. A., Nelson, H. D., Pepe, M. S., Allison, K. H., Schnitt, S. J., O’Malley, F. P., & Weaver, D. L. (2015). Diagnostic Concordance Among Pathologists Interpreting Breast Biopsy Specimens. *JAMA*, *313*(11), 1122. https://doi.org/10.1001/jama.2015.1405

Gurcan, M., Boucheron, L., Can, A., Madabhushi, A., Rajpoot, N., & Yener, B. (2009). Histopathological Image Analysis: A Review. *Biomedical Engineering, IEEE Reviews In*, *2*, 147–171. https://doi.org/10.1109/RBME.2009.2034865

Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. (2002). Gene Selection for Cancer Classification using Support Vector Machines. *Machine Learning*, *46*(1–3), 389–422. https://doi.org/10.1023/A:1012487302797

Haralick, R. M., Shanmugam, K., & Dinstein, I. (1973). Textural Features for Image Classification. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-3*(6), 610–621. https://doi.org/10.1109/TSMC.1973.4309314

Holzinger, A., Biemann, C., Pattichis, C. S., & Kell, D. B. (2017). *What do we need to build explainable AI systems for the medical domain?* https://doi.org/https://doi.org/10.48550/arXiv.1712.09923

Kononenko, I. (1994). *Estimating attributes: Analysis and extensions of RELIEF* (pp. 171–182). https://doi.org/10.1007/3-540-57868-4\_57

Ojala, T., Pietikainen, M., & Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(7), 971–987. https://doi.org/10.1109/TPAMI.2002.1017623

Saeys, Y., Abeel, T., & Van de Peer, Y. (2008). *Robust Feature Selection Using Ensemble Feature Selection Techniques* (pp. 313–325). https://doi.org/10.1007/978-3-540-87481-2\_21

Spanhol, F. A., Oliveira, L. S., Petitjean, C., & Heutte, L. (2016a). A Dataset for Breast Cancer Histopathological Image Classification. *IEEE Transactions on Biomedical Engineering*, *63*(7), 1455–1462. https://doi.org/10.1109/TBME.2015.2496264

Spanhol, F. A., Oliveira, L. S., Petitjean, C., & Heutte, L. (2016b). A Dataset for Breast Cancer Histopathological Image Classification. *IEEE Transactions on Biomedical Engineering*, *63*(7), 1455–1462. https://doi.org/10.1109/TBME.2015.2496264

Tibshirani, R. (1996). Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *58*(1), 267–288. https://doi.org/10.1111/j.2517-6161.1996.tb02080.x

Wang, D., Wang, P., & Shi, J. (2018). A fast and efficient conformal regressor with regularized extreme learning machine. *Neurocomputing*, *304*, 1–11. https://doi.org/10.1016/j.neucom.2018.04.012

WHO. (2020). *World Health Organization. (2020)*.

Yan, R., Ren, F., Wang, Z., Wang, L., Zhang, T., Liu, Y., Rao, X., Zheng, C., & Zhang, F. (2020). Breast cancer histopathological image classification using a hybrid deep neural network. *Methods*, *173*, 52–60. https://doi.org/10.1016/j.ymeth.2019.06.014

# Appendix

## Appendix A - (Pseudo Code)

### A.1 Algorithm 1: Main Classification Pipeline

ALGORITHM: BreastCancerClassificationPipeline

INPUT: Raw BreakHis dataset D\_raw, Configuration parameters cfg

OUTPUT: Trained models {M\_SVM, M\_RF, M\_XGB}, Performance metrics P

1: // STEP 1: Problem Definition and Configuration

2: SET problem\_type ← "Binary Classification"

3: SET classes ← {"benign", "malignant"}

4: SET random\_seed ← 7

5:

6: // STEP 2: Data Acquisition and Patient-Disjoint Splitting

7: D\_prepared ← PrepareDataset(D\_raw, train\_ratio=0.85, test\_ratio=0.15)

8: (D\_train, D\_test) ← SplitByPatient(D\_prepared)

9: VERIFY **no** patient overlap between D\_train **and** D\_test

10:

11: // STEP 3: Image Preprocessing Pipeline (10 stages)

12: FOR each image I **in** {D\_train, D\_test} DO

13: I\_gray ← ConvertToGrayscale(I)

14: I\_resized ← Resize(I\_gray, size=[128,128])

15: I\_filtered ← ApplyFilters(I\_resized) // Median, Wiener, CLAHE, Gaussian

16: I\_enhanced ← EnhanceEdges(I\_filtered) // Unsharp masking

17: I\_morpho ← MorphologicalOperations(I\_enhanced) // Opening, closing

18: I\_processed ← NormalizeIntensity(I\_morpho, range=[0,1])

19: **END** FOR

20:

21: // STEP 4: Multi-Modal Feature Extraction

22: FOR each preprocessed image I\_processed DO

23: F\_HOG ← ExtractHOG(I\_processed, cell\_size=16×16) // 1,764 features

24: F\_GLCM ← ExtractGLCM(I\_processed, offsets=8\_directions) // 6 features

25: F\_LBP ← ExtractLBP(I\_processed, radius=1, neighbors=8) // 944 features

26: F\_Gabor ← ExtractGabor(I\_processed, λ={2,4,8}, θ={0°,45°,90°,135°}) // 24 features

27: F\_Intensity ← ExtractIntensityStats(I\_processed) // 7 features

28: F\_Stats ← ExtractStatisticalDescriptors(I\_processed) // 19 features

29: F\_all ← CONCATENATE(F\_HOG, F\_GLCM, F\_LBP, F\_Gabor, F\_Intensity, F\_Stats)

30: **END** FOR

31: // Total: 2,767 features per image

32:

33: // STEP 5: Ensemble Feature Selection

34: X\_train ← FeatureMatrix(D\_train) // N\_train × 2767

35: y\_train ← Labels(D\_train)

36:

37: // 5a: Preprocessing filters

38: X\_filtered ← RemoveLowVariance(X\_train, threshold=0.001)

39: X\_filtered ← RemoveHighCorrelation(X\_filtered, threshold=0.95)

40: // Remaining: ~2,084 features

41:

42: // 5b: Multiple ranking methods

43: ranks\_ReliefF ← ReliefF(X\_filtered, y\_train, k\_neighbors=10)

44: ranks\_FScore ← FScoreANOVA(X\_filtered, y\_train)

45: ranks\_RFE ← RecursiveFeatureElimination(X\_filtered, y\_train, estimator=SVM)

46: ranks\_Tree ← TreeImportance(X\_filtered, y\_train, n\_trees=50)

47: ranks\_LASSO ← LASSORegularization(X\_filtered, y\_train, λ=0.01)

48:

49: // 5c: Ensemble ranking via mean aggregation

50: ranks\_ensemble ← MEAN(ranks\_ReliefF, ranks\_FScore, ranks\_RFE, ranks\_Tree, ranks\_LASSO)

51:

52: // 5d: Cross-validation to select optimal K

53: K\_best ← 0

54: best\_cv\_score ← 0

55: FOR K **in** {150, 300, 400} DO

56: X\_selected ← SelectTopK(X\_filtered, ranks\_ensemble, K)

57: cv\_score ← CrossValidate(X\_selected, y\_train, folds=3, model=SVM)

58: IF cv\_score > best\_cv\_score THEN

59: best\_cv\_score ← cv\_score

60: K\_best ← K

61: **END** IF

62: **END** FOR

63: X\_train\_final ← SelectTopK(X\_filtered, ranks\_ensemble, K\_best)

64: // Selected: 400 features

65:

66: // STEP 6 & 7: Model Training with Hyperparameter Tuning

67: // SVM with LDA preprocessing

68: X\_train\_lda ← LinearDiscriminantAnalysis(X\_train\_final, y\_train)

69: params\_svm ← {BoxConstraint: {10,100,1000}, KernelScale: {0.5,1,5}}

70: M\_SVM ← GridSearchCV(SVM\_RBF, X\_train\_lda, y\_train, params\_svm, cv=3)

71:

72: // Random Forest

73: params\_rf ← {NumTrees: {100,200}, MinLeafSize: {1,3,5}}

74: M\_RF ← GridSearchCV(RandomForest, X\_train\_final, y\_train, params\_rf, cv=3)

75:

76: // XGBoost (Gradient Boosting)

77: params\_xgb ← {NumCycles: {100,200}, LearnRate: {0.1,0.2}, MinLeafSize: {1,3,5}}

78: M\_XGB ← GridSearchCV(XGBoost, X\_train\_final, y\_train, params\_xgb, cv=3)

79:

80: // STEP 8: Model Evaluation on Test Set

81: X\_test\_final ← SelectTopK(FeatureMatrix(D\_test), ranks\_ensemble, K\_best)

82: X\_test\_lda ← Transform\_LDA(X\_test\_final) // For SVM

83:

84: // Predictions

85: y\_pred\_svm ← M\_SVM.Predict(X\_test\_lda)

86: y\_pred\_rf ← M\_RF.Predict(X\_test\_final)

87: y\_pred\_xgb ← M\_XGB.Predict(X\_test\_final)

88:

89: // Performance metrics

90: FOR each model **in** {SVM, RF, XGB} DO

91: CM ← ConfusionMatrix(y\_test, y\_pred)

92: Accuracy ← (TP + TN) / (TP + TN + FP + FN)

93: Sensitivity ← TP / (TP + FN)

94: Specificity ← TN / (TN + FP)

95: Precision ← TP / (TP + FP)

96: F1Score ← 2 × (Precision × Sensitivity) / (Precision + Sensitivity)

97: AUC ← ComputeROC(y\_test, prediction\_scores)

98: **END** FOR

99:

100: // Statistical significance testing

101: p\_values ← McNemarsTest(y\_pred\_svm, y\_pred\_rf, y\_pred\_xgb, y\_test)

102: confidence\_intervals ← Bootstrap(predictions, y\_test, n\_iterations=100)

103:

104: RETURN {M\_SVM, M\_RF, M\_XGB}, Performance\_Metrics

### A.2 Algorithm 2: Patient-Disjoint Data Splitting

ALGORITHM: PatientDisjointSplit

INPUT: Dataset D **with** patient IDs, train\_ratio α, test\_ratio β

OUTPUT: Training **set** D\_train, Test **set** D\_test (patient-disjoint)

1: patients ← UNIQUE(D.patient\_ids)

2: Shuffle(patients) // Random permutation

3:

4: // For each class separately (benign, malignant)

5: FOR each **class** c **in** {benign, malignant} DO

6: patients\_c ← patients WHERE D.**class** = c

7: n\_total ← COUNT(images WHERE **class** = c)

8: target\_test ← ROUND(β × n\_total)

9: target\_train ← n\_total - target\_test

10:

11: // Allocate patients to test set

12: test\_patients\_c ← {}

13: n\_test\_images ← 0

14: WHILE n\_test\_images < target\_test DO

15: patient\_p ← patients\_c.pop()

16: images\_p ← GET images **from** patient\_p

17: test\_patients\_c ← test\_patients\_c ∪ {patient\_p}

18: n\_test\_images ← n\_test\_images + COUNT(images\_p)

19: **END** WHILE

20:

21: // Remaining patients go to training set

22: train\_patients\_c ← patients\_c - test\_patients\_c

23: **END** FOR

24:

25: // Verify no patient overlap

26: ASSERT INTERSECTION(train\_patients, test\_patients) = ∅

27:

28: D\_train ← {all images **from** train\_patients}

29: D\_test ← {all images **from** test\_patients}

30:

31: RETURN D\_train, D\_test

### A.3 Algorithm 3: 10-Stage Image Preprocessing Pipeline

ALGORITHM: PreprocessImage

INPUT: RGB image I (H×W×3)

OUTPUT: Preprocessed grayscale image I\_final (128×128)

1: // Stage 1: Grayscale conversion

2: I\_gray ← 0.2989×I\_R + 0.5870×I\_G + 0.1140×I\_B

3:

4: // Stage 2: Resize to standard dimensions

5: I\_resized ← BicubicInterpolation(I\_gray, target\_size=[128,128])

6:

7: // Stage 3: Median filtering (salt-and-pepper noise removal)

8: I\_median ← MedianFilter(I\_resized, window\_size=3×3)

9:

10: // Stage 4: Wiener filtering (adaptive denoising)

11: I\_wiener ← WienerFilter(I\_median, window\_size=5×5)

12:

13: // Stage 5: CLAHE (Contrast Limited Adaptive Histogram Equalization)

14: I\_clahe ← CLAHE(I\_wiener, tile\_size=8×8, clip\_limit=0.02)

15:

16: // Stage 6: Multi-scale Gaussian filtering

17: FOR σ **in** {0.5, 1.0, 2.0} DO

18: I\_gaussian[σ] ← GaussianFilter(I\_clahe, σ)

19: **END** FOR

20: I\_multiscale ← COMBINE(I\_gaussian) // Weighted average

21:

22: // Stage 7: Unsharp masking (edge enhancement)

23: I\_blurred ← GaussianFilter(I\_multiscale, σ=2.0)

24: I\_unsharp ← I\_multiscale + 0.8×(I\_multiscale - I\_blurred)

25:

26: // Stage 8: Morphological operations

27: SE ← StructuringElement(disk, radius=2)

28: I\_opened ← MorphologicalOpening(I\_unsharp, SE)

29: I\_closed ← MorphologicalClosing(I\_opened, SE)

30:

31: // Stage 9: Bilateral filtering (edge-preserving smoothing)

32: I\_bilateral ← BilateralFilter(I\_closed, σ\_intensity=0.5, σ\_spatial=2.0)

33:

34: // Stage 10: Intensity normalization

35: I\_final ← (I\_bilateral - MIN(I\_bilateral)) / (MAX(I\_bilateral) - MIN(I\_bilateral))

36:

37: RETURN I\_final

### A.4 Algorithm 4: Ensemble Feature Selection via Mean Rank Aggregation

ALGORITHM: EnsembleFeatureSelection

INPUT: Feature matrix X (N×P), Labels y, Candidate K values

OUTPUT: Selected feature indices idx\_selected, Optimal K

1: // Preprocessing: Remove low-variance and highly-correlated features

2: **var** ← VARIANCE(X, axis=samples)

3: idx\_var ← {i | **var**[i] > 0.001}

4: X ← X[:, idx\_var]

5:

6: corr\_matrix ← CORRELATION(X)

7: idx\_uncorr ← RemoveHighCorrelation(corr\_matrix, threshold=0.95)

8: X ← X[:, idx\_uncorr]

9: P\_filtered ← X.shape[1] // Typically ~2,084 features

10:

11: // Multiple ranking methods

12: ranks ← {}

13:

14: // Method 1: ReliefF (distance-based)

15: FOR i ← 1 to P\_filtered DO

16: score[i] ← 0

17: FOR each sample x\_j DO

18: near\_hit ← FindKNN(x\_j, same\_class, k=10)

19: near\_miss ← FindKNN(x\_j, different\_class, k=10)

20: score[i] ← score[i] + MEAN(|x\_j[i] - near\_miss[i]|) - MEAN(|x\_j[i] - near\_hit[i]|)

21: **END** FOR

22: **END** FOR

23: ranks['ReliefF'] ← ARGSORT(score, **descending**=**True**)

24:

25: // Method 2: F-Score (ANOVA)

26: FOR i ← 1 to P\_filtered DO

27: μ\_benign ← MEAN(X[y=benign, i])

28: μ\_malignant ← MEAN(X[y=malignant, i])

29: σ²\_benign ← VARIANCE(X[y=benign, i])

30: σ²\_malignant ← VARIANCE(X[y=malignant, i])

31: F\_score[i] ← (μ\_benign - μ\_malignant)² / (σ²\_benign + σ²\_malignant)

32: **END** FOR

33: ranks['FScore'] ← ARGSORT(F\_score, **descending**=**True**)

34:

35: // Method 3: RFE (Recursive Feature Elimination)

36: features\_remaining ← {1, 2, ..., P\_filtered}

37: WHILE LENGTH(features\_remaining) > 50 DO

38: model ← TrainSVM(X[:, features\_remaining], y)

39: weights ← model.coefficients

40: idx\_remove ← BOTTOM\_10\_PERCENT(|weights|)

41: features\_remaining ← features\_remaining - idx\_remove

42: **END** WHILE

43: ranks['RFE'] ← features\_remaining (ranked **by** |weights|)

44:

45: // Method 4: Tree-based importance

46: model\_rf ← RandomForest(n\_trees=50, X, y)

47: importance ← model\_rf.feature\_importances // Gini importance

48: ranks['Tree'] ← ARGSORT(importance, **descending**=**True**)

49:

50: // Method 5: LASSO (L1 regularization)

51: FOR λ **in** {0.001, 0.01, 0.1, 1.0} DO

52: β ← LASSO\_Regression(X, y, λ)

53: score\_cv ← CrossValidate(LASSO, λ)

54: **END** FOR

55: λ\_optimal ← λ **with** best score\_cv

56: β\_optimal ← LASSO\_Regression(X, y, λ\_optimal)

57: ranks['LASSO'] ← ARGSORT(|β\_optimal|, **descending**=**True**)

58:

59: // Ensemble: Mean rank aggregation

60: FOR i ← 1 to P\_filtered DO

61: rank\_mean[i] ← MEAN(ranks['ReliefF'][i], ranks['FScore'][i],

62: ranks['RFE'][i], ranks['Tree'][i], ranks['LASSO'][i])

63: **END** FOR

64: idx\_ranked ← ARGSORT(rank\_mean, **ascending**=**True**)

65:

66: // Cross-validation to select optimal K

67: FOR K **in** {150, 300, 400} DO

68: idx\_K ← idx\_ranked[1:K]

69: X\_K ← X[:, idx\_K]

70: cv\_accuracy[K] ← StratifiedKFoldCV(X\_K, y, model=SVM, folds=3)

71: **END** FOR

72: K\_optimal ← ARGMAX(cv\_accuracy)

73: idx\_selected ← idx\_ranked[1:K\_optimal]

74:

75: RETURN idx\_selected, K\_optimal

### A.5 Algorithm 5: SVM with LDA Dimensionality Reduction

ALGORITHM: SVM\_with\_LDA

INPUT: Training data (X\_train, y\_train), Test data X\_test

OUTPUT: Predictions y\_pred, Trained model M

1: // Linear Discriminant Analysis projection

2: μ\_0 ← MEAN(X\_train[y\_train=0, :]) // Benign mean

3: μ\_1 ← MEAN(X\_train[y\_train=1, :]) // Malignant mean

4:

5: // Within-class scatter matrix

6: S\_W ← ZEROS(P, P)

7: FOR each **class** c **in** {0, 1} DO

8: X\_c ← X\_train[y\_train=c, :]

9: FOR each sample x **in** X\_c DO

10: S\_W ← S\_W + (x - μ\_c)(x - μ\_c)^T

11: **END** FOR

12: **END** FOR

13:

14: // Between-class scatter matrix

15: S\_B ← (μ\_0 - μ\_1)(μ\_0 - μ\_1)^T

16:

17: // Optimal projection direction

18: w ← S\_W^(-1) × (μ\_0 - μ\_1)

19: w ← w / ||w|| // Normalize

20:

21: // Project to 1D

22: z\_train ← X\_train × w // (N×P) × (P×1) = (N×1)

23: z\_test ← X\_test × w

24:

25: // Train SVM on projected data

26: M ← SVM\_RBF(z\_train, y\_train, C=100, γ=1/(2×σ²))

27:

28: // Predictions

29: y\_pred ← M.Predict(z\_test)

30:

31: RETURN y\_pred, M

### A.6 Algorithm 6: Performance Evaluation with Bootstrap Confidence Intervals

ALGORITHM: EvaluateWithConfidenceIntervals

INPUT: **True** labels y\_true, Predictions y\_pred, n\_bootstrap iterations

OUTPUT: Performance metrics **with** 95% confidence intervals

1: // Confusion matrix

2: TN ← COUNT(y\_true=0 AND y\_pred=0)

3: FP ← COUNT(y\_true=0 AND y\_pred=1)

4: FN ← COUNT(y\_true=1 AND y\_pred=0)

5: TP ← COUNT(y\_true=1 AND y\_pred=1)

6:

7: // Point estimates

8: Accuracy ← (TP + TN) / (TP + TN + FP + FN)

9: Sensitivity ← TP / (TP + FN)

10: Specificity ← TN / (TN + FP)

11: Precision ← TP / (TP + FP)

12: F1 ← 2 × (Precision × Sensitivity) / (Precision + Sensitivity)

13:

14: // Bootstrap confidence intervals

15: FOR iter ← 1 to n\_bootstrap DO

16: // Resample with replacement

17: indices ← RandomSample(1:N, size=N, replacement=**True**)

18: y\_true\_boot ← y\_true[indices]

19: y\_pred\_boot ← y\_pred[indices]

20:

21: // Compute metrics on bootstrap sample

22: Accuracy\_boot[iter] ← ComputeAccuracy(y\_true\_boot, y\_pred\_boot)

23: Sensitivity\_boot[iter] ← ComputeSensitivity(y\_true\_boot, y\_pred\_boot)

24: Specificity\_boot[iter] ← ComputeSpecificity(y\_true\_boot, y\_pred\_boot)

25: **END** FOR

26:

27: // 95% Confidence intervals (percentile method)

28: CI\_Accuracy ← [PERCENTILE(Accuracy\_boot, 2.5), PERCENTILE(Accuracy\_boot, 97.5)]

29: CI\_Sensitivity ← [PERCENTILE(Sensitivity\_boot, 2.5), PERCENTILE(Sensitivity\_boot, 97.5)]

30: CI\_Specificity ← [PERCENTILE(Specificity\_boot, 2.5), PERCENTILE(Specificity\_boot, 97.5)]

31:

32: RETURN {Accuracy, Sensitivity, Specificity, Precision, F1},

33: {CI\_Accuracy, CI\_Sensitivity, CI\_Specificity}

## Appendix B - (Code Snippet)

### B.1 Image Preprocessing Pipeline

#### B.1.1 CLAHE (Contrast Limited Adaptive Histogram Equalization)

**function** I\_enhanced = applyCLAHE(I\_gray, tileSize, clipLimit)

% Contrast Limited Adaptive Histogram Equalization

% Prevents over-amplification **of** noise **in** homogeneous regions

%

% INPUT:

% I\_gray - Grayscale image [0,1]

% tileSize - [rows, cols] **for** contextual regions (**default**: [8,8])

% clipLimit - Contrast enhancement limit (**default**: 0.02)

% OUTPUT:

% I\_enhanced - Contrast-enhanced image [0,1]

% Convert to uint8 range **for** adapthisteq

I\_uint8 = im2uint8(I\_gray);

% Apply CLAHE **with** specified parameters

I\_enhanced\_uint8 = adapthisteq(I\_uint8, ...

'NumTiles', tileSize, ...

'ClipLimit', clipLimit, ...

'Distribution', 'uniform');

% Convert back to [0,1] range

I\_enhanced = im2double(I\_enhanced\_uint8);

**end**

**Key Design Choice:** Tile size of 8×8 balances local contrast enhancement with computational efficiency. Clip limit of 0.02 prevents noise amplification in low-contrast regions (common in benign tissue).

#### B.1.2 Multi-Scale Gaussian Filtering

**function** I\_filtered = multiScaleGaussian(I, sigmas, weights)

% Multi-scale Gaussian filtering **for** robust feature extraction

% Combines responses at multiple scales to capture structures

% **of** varying sizes (nuclei, cells, tissue patterns)

%

% INPUT:

% I - Input image

% sigmas - Array **of** Gaussian standard deviations [0.5, 1.0, 2.0]

% weights - Contribution weights **for** each scale (**default**: equal)

% OUTPUT:

% I\_filtered - Multi-scale filtered image

**if** nargin < 3

weights = ones(size(sigmas)) / length(sigmas); % Equal weights

**end**

% Normalize weights

weights = weights / sum(weights);

% Initialize accumulator

I\_filtered = zeros(size(I));

% Apply Gaussian filtering at each scale **and** combine

**for** i = 1:length(sigmas)

sigma = sigmas(i);

% Create Gaussian filter kernel

kernel\_size = 2 \* ceil(3 \* sigma) + 1; % 6σ width ensures >99% coverage

h = fspecial('gaussian', kernel\_size, sigma);

% Filter **and** accumulate weighted response

I\_scale = imfilter(I, h, 'replicate', 'conv');

I\_filtered = I\_filtered + weights(i) \* I\_scale;

**end**

**end**

**Rationale**: Different tissue structures appear at different scales. Nuclear features (σ=0.5), cellular patterns (σ=1.0), and tissue architecture (σ=2.0) are captured and combined.

### B.2 Feature Extraction Methods

#### B.2.1 HOG (Histogram of Oriented Gradients) Feature Extraction

**function** features\_HOG = extractHOGFeatures\_Custom(I, cellSize)

% Extract Histogram **of** Oriented Gradients features

% Captures edge orientations **and** **local** shape information

%

% INPUT:

% I - Grayscale image [0,1]

% cellSize - [rows, cols] **for** HOG cells (**default**: [16,16])

% OUTPUT:

% features\_HOG - 1D feature vector (length: 1,764 **for** 128×128 image)

% Compute gradients **using** Sobel **operator**

[Gx, Gy] = imgradientxy(I, 'sobel');

% Gradient magnitude **and** orientation

mag = sqrt(Gx.^2 + Gy.^2);

theta = atan2(Gy, Gx); % Range: [-π, π]

% Convert to degrees **and** shift to [0, 180) (**unsigned** gradients)

theta\_deg = mod(rad2deg(theta), 180);

% Define orientation bins (9 bins, 20° each)

num\_bins = 9;

bin\_edges = linspace(0, 180, num\_bins + 1);

% Get image dimensions **and** number **of** cells

[height, width] = size(I);

cell\_rows = floor(height / cellSize(1));

cell\_cols = floor(width / cellSize(2));

% Initialize HOG descriptor

HOG = zeros(cell\_rows, cell\_cols, num\_bins);

% For each cell, compute orientation histogram

**for** r = 1:cell\_rows

**for** c = 1:cell\_cols

% Extract cell region

row\_start = (r-1) \* cellSize(1) + 1;

row\_end = min(r \* cellSize(1), height);

col\_start = (c-1) \* cellSize(2) + 1;

col\_end = min(c \* cellSize(2), width);

cell\_mag = mag(row\_start:row\_end, col\_start:col\_end);

cell\_theta = theta\_deg(row\_start:row\_end, col\_start:col\_end);

% Build histogram weighted **by** magnitude

**for** bin = 1:num\_bins

bin\_min = bin\_edges(bin);

bin\_max = bin\_edges(bin + 1);

% Find pixels **in** **this** orientation bin

mask = (cell\_theta >= bin\_min) & (cell\_theta < bin\_max);

% Sum magnitudes (vote)

HOG(r, c, bin) = sum(cell\_mag(mask));

**end**

**end**

**end**

% Normalize each cell (L2 normalization)

epsilon = 1e-5; % Prevent division **by** zero

**for** r = 1:cell\_rows

**for** c = 1:cell\_cols

cell\_norm = sqrt(sum(HOG(r, c, :).^2) + epsilon^2);

HOG(r, c, :) = HOG(r, c, :) / cell\_norm;

**end**

**end**

% Flatten to 1D feature vector

features\_HOG = HOG(:)';

end

**Clinical Relevance:** HOG captures nuclear boundary orientations, which differ between benign (smooth, regular) and malignant (irregular, jagged) cells.

#### B.2.2 GLCM (Gray-Level Co-occurrence Matrix) Texture Features

matlab

**function** features\_GLCM = extractGLCMFeatures(I)

% Extract texture features **using** Gray-Level Co-occurrence Matrix

% Quantifies spatial relationships between pixel intensities

%

% OUTPUT: 6-element vector [Contrast, Correlation, Energy, Homogeneity, Entropy, ASM]

% Quantize image to 8 gray levels (reduces computational cost)

I\_quantized = im2uint8(I);

I\_quantized = floor(**double**(I\_quantized) / 32); % 256/8 = 32

% Define offsets (8 directions: 0°, 45°, 90°, 135°, **and** negatives)

offsets = [0 1; -1 1; -1 0; -1 -1]; % 4 unique angular relationships

% Compute GLCM

glcm = graycomatrix(I\_quantized, 'Offset', offsets, ...

'NumLevels', 8, 'Symmetric', **true**);

% Extract statistical properties

stats = graycoprops(glcm, {'Contrast', 'Correlation', 'Energy', 'Homogeneity'});

% Average across all directions

contrast\_avg = mean(stats.Contrast);

correlation\_avg = mean(stats.Correlation);

energy\_avg = mean(stats.Energy);

homogeneity\_avg = mean(stats.Homogeneity);

% Additional features

% Entropy: Measures randomness/disorder

glcm\_norm = glcm / sum(glcm(:)); % Normalize to probability distribution

entropy\_val = -sum(glcm\_norm(glcm\_norm > 0) .\* log2(glcm\_norm(glcm\_norm > 0)));

% ASM (Angular Second Moment): Measures uniformity

asm\_val = sum(glcm\_norm(:).^2);

% Combine features

features\_GLCM = [contrast\_avg, correlation\_avg, energy\_avg, ...

homogeneity\_avg, entropy\_val, asm\_val];

**end**

**Interpretation:**

* **High Contrast:** Malignant tissue (irregular cell arrangements)
* **Low Homogeneity:** Cancer (disrupted tissue organization)
* **High Entropy:** Chaotic cellular architecture in tumors

### B.3 Feature Selection Methods

#### B.3.1 ReliefF Algorithm Implementation

matlab

**function** [feature\_ranks, feature\_weights] = reliefF\_custom(X, y, k\_neighbors)

% ReliefF: Distance-based feature weighting algorithm

% Rewards features that distinguish between classes (near-misses)

% **while** keeping same-**class** samples close (near-hits)

%

% INPUT:

% X - Feature matrix (N×P), N samples, P features

% y - Class labels (N×1), binary {0,1}

% k\_neighbors - Number **of** nearest neighbors (**default**: 10)

% OUTPUT:

% feature\_ranks - Feature indices sorted **by** importance

% feature\_weights - Importance scores **for** each feature

[N, P] = size(X);

weights = zeros(1, P);

% Normalize features to [0,1] **for** fair distance computation

X\_norm = (X - min(X)) ./ (max(X) - min(X) + eps);

% For each sample, update feature weights

**for** i = 1:N

sample = X\_norm(i, :);

sample\_class = y(i);

% Find k nearest hits (same **class**)

same\_class\_idx = find(y == sample\_class & (1:N)' ~= i);

distances\_hit = pdist2(sample, X\_norm(same\_class\_idx, :));

[~, sorted\_hit\_idx] = sort(distances\_hit);

near\_hits = same\_class\_idx(sorted\_hit\_idx(1:min(k\_neighbors, length(same\_class\_idx))));

% Find k nearest misses (different class)

diff\_class\_idx = find(y ~= sample\_class);

distances\_miss = pdist2(sample, X\_norm(diff\_class\_idx, :));

[~, sorted\_miss\_idx] = sort(distances\_miss);

near\_misses = diff\_class\_idx(sorted\_miss\_idx(1:min(k\_neighbors, length(diff\_class\_idx))));

% Update weights for each feature

for j = 1:P

% Penalty: distance to near-hits (want hits to be close)

diff\_hits = abs(sample(j) - X\_norm(near\_hits, j));

% Reward: distance to near-misses (want misses to be far)

diff\_misses = abs(sample(j) - X\_norm(near\_misses, j));

% Update weight: increase if feature separates classes

weights(j) = weights(j) + (mean(diff\_misses) - mean(diff\_hits));

end

end

% Normalize weights

weights = weights / N;

% Rank features by weight (descending order)

[feature\_weights, feature\_ranks] = sort(weights, 'descend');

end

**Advantage: Captures feature interactions. Features that are individually weak but jointly discriminative (e.g., nuclear size + chromatin pattern) are ranked higher.**

#### B.3.2 Ensemble Feature Selection via Mean Rank Aggregation

matlab

**function** [selected\_features, K\_optimal] = ensembleFeatureSelection(X, y, K\_candidates)

% Ensemble feature selection combining multiple ranking methods

% Reduces bias **from** any single method, improves robustness

%

% INPUT:

% X - Feature matrix (N×P)

% y - Labels (N×1)

% K\_candidates - Array **of** candidate feature counts [150, 300, 400]

% OUTPUT:

% selected\_features - Indices **of** selected features

% K\_optimal - Optimal number **of** features (via CV)

P = size(X, 2);

% ===== Method 1: ReliefF =====

[ranks\_relief, ~] = reliefF\_custom(X, y, 10);

% ===== Method 2: F-Score (ANOVA) =====

mu\_0 = mean(X(y == 0, :), 1);

mu\_1 = mean(X(y == 1, :), 1);

var\_0 = **var**(X(y == 0, :), 0, 1);

var\_1 = **var**(X(y == 1, :), 0, 1);

f\_scores = (mu\_0 - mu\_1).^2 ./ (var\_0 + var\_1 + eps);

[~, ranks\_fscore] = sort(f\_scores, 'descend');

% ===== Method 3: RFE (Recursive Feature Elimination) =====

% Simplified version: train SVM, rank **by** |weights|

mdl\_svm = fitcsvm(X, y, 'KernelFunction', 'linear', 'Standardize', **true**);

coef = mdl\_svm.Beta;

[~, ranks\_rfe] = sort(abs(coef), 'descend');

% ===== Method 4: Tree-based importance =====

mdl\_tree = TreeBagger(50, X, y, 'OOBPredictorImportance', 'on');

importance\_tree = mdl\_tree.OOBPermutedPredictorDeltaError;

[~, ranks\_tree] = sort(importance\_tree, 'descend');

% ===== Method 5: LASSO =====

[B, FitInfo] = lassoglm(X, y, 'binomial', 'CV', 5);

idx\_min = FitInfo.IndexMinDeviance;

lasso\_coef = B(:, idx\_min);

[~, ranks\_lasso] = sort(abs(lasso\_coef), 'descend');

% ===== Ensemble: Mean Rank Aggregation =====

rank\_matrix = zeros(P, 5);

**for** i = 1:P

rank\_matrix(i, 1) = find(ranks\_relief == i);

rank\_matrix(i, 2) = find(ranks\_fscore == i);

rank\_matrix(i, 3) = find(ranks\_rfe == i);

rank\_matrix(i, 4) = find(ranks\_tree == i);

rank\_matrix(i, 5) = find(ranks\_lasso == i);

**end**

mean\_ranks = mean(rank\_matrix, 2);

[~, ensemble\_ranks] = sort(mean\_ranks);

% ===== Cross-Validation to Select Optimal K =====

cv\_scores = zeros(length(K\_candidates), 1);

**for** k\_idx = 1:length(K\_candidates)

K = K\_candidates(k\_idx);

selected\_idx = ensemble\_ranks(1:K);

X\_selected = X(:, selected\_idx);

% 3-fold stratified cross-validation

cv = cvpartition(y, 'KFold', 3, 'Stratify', **true**);

acc = zeros(cv.NumTestSets, 1);

**for** fold = 1:cv.NumTestSets

X\_train = X\_selected(cv.training(fold), :);

y\_train = y(cv.training(fold));

X\_test = X\_selected(cv.test(fold), :);

y\_test = y(cv.test(fold));

mdl = fitcsvm(X\_train, y\_train, 'KernelFunction', 'rbf', 'Standardize', **true**);

y\_pred = predict(mdl, X\_test);

acc(fold) = sum(y\_pred == y\_test) / length(y\_test);

**end**

cv\_scores(k\_idx) = mean(acc);

**end**

% Select K **with** highest CV accuracy

[~, best\_k\_idx] = max(cv\_scores);

K\_optimal = K\_candidates(best\_k\_idx);

selected\_features = ensemble\_ranks(1:K\_optimal);

fprintf('Optimal K = %d with CV accuracy = %.4f\n', K\_optimal, max(cv\_scores));

**end**

**Justification:** Different methods have different biases. ReliefF captures interactions, F-Score is fast, RFE considers model-specific relevance, Trees handle non-linearity, LASSO enforces sparsity. Averaging ranks produces a robust consensus.

### B.4 Model Training with LDA Preprocessing

#### B.4.1 Linear Discriminant Analysis for SVM

matlab

**function** [model\_svm, lda\_projection] = trainSVM\_withLDA(X\_train, y\_train, X\_test)

% SVM **with** LDA dimensionality reduction

% Projects 400D feature space to 1D **for** maximum **class** separation

%

% OUTPUT:

% model\_svm - Trained SVM model

% lda\_projection - LDA projection vector **for** test transformation

% ===== LDA Projection =====

% Compute **class** means

mu\_0 = mean(X\_train(y\_train == 0, :), 1)'; % Column vector

mu\_1 = mean(X\_train(y\_train == 1, :), 1)';

% Within-**class** scatter matrix

S\_W = zeros(size(X\_train, 2));

**for** c = [0, 1]

X\_c = X\_train(y\_train == c, :);

mu\_c = mean(X\_c, 1)';

for i = 1:size(X\_c, 1)

x\_i = X\_c(i, :)' - mu\_c;

S\_W = S\_W + x\_i \* x\_i';

end

end

% Between-class scatter matrix

S\_B = (mu\_0 - mu\_1) \* (mu\_0 - mu\_1)';

% Optimal projection direction (Fisher's criterion)

% Maximize: w^T S\_B w / w^T S\_W w

% Solution: w = S\_W^(-1) \* (mu\_0 - mu\_1)

w = S\_W \ (mu\_0 - mu\_1);

% Normalize projection vector

w = w / norm(w);

lda\_projection = w;

% Project training data to 1D

z\_train = X\_train \* w;

% ===== Train SVM on 1D Projection =====

% Grid search for hyperparameters

boxConstraints = [10, 100, 1000];

kernelScales = [0.5, 1, 5];

best\_acc = 0;

best\_params = struct();

for C = boxConstraints

for sigma = kernelScales

% 3-fold cross-validation

cv = cvpartition(y\_train, 'KFold', 3);

cv\_acc = 0;

for fold = 1:3

z\_cv\_train = z\_train(cv.training(fold));

y\_cv\_train = y\_train(cv.training(fold));

z\_cv\_test = z\_train(cv.test(fold));

y\_cv\_test = y\_train(cv.test(fold));

mdl = fitcsvm(z\_cv\_train, y\_cv\_train, ...

'KernelFunction', 'rbf', ...

'BoxConstraint', C, ...

'KernelScale', sigma);

y\_cv\_pred = predict(mdl, z\_cv\_test);

cv\_acc = cv\_acc + sum(y\_cv\_pred == y\_cv\_test) / length(y\_cv\_test);

end

cv\_acc = cv\_acc / 3;

if cv\_acc > best\_acc

best\_acc = cv\_acc;

best\_params.C = C;

best\_params.sigma = sigma;

end

end

end

% Train final model with best parameters

model\_svm = fitcsvm(z\_train, y\_train, ...

'KernelFunction', 'rbf', ...

'BoxConstraint', best\_params.C, ...

'KernelScale', best\_params.sigma);

fprintf('Best SVM parameters: C=%.1f, sigma=%.1f, CV Acc=%.4f\n', ...

best\_params.C, best\_params.sigma, best\_acc);

end

**Why LDA before SVM?**

1. **Dimensionality Reduction:** 400D → 1D reduces overfitting risk
2. **Class Separation:** LDA maximizes between-class variance / within-class variance
3. **Computational Efficiency:** Training on 1D data is much faster
4. **Interpretability:** Single discriminant score is clinically meaningful

### B.5 Performance Evaluation

#### B.5.1 Bootstrap Confidence Intervals

matlab

**function** [metrics, confidence\_intervals] = evaluateWithBootstrap(y\_true, y\_pred, n\_bootstrap)

% Compute performance metrics **with** bootstrap confidence intervals

%

% INPUT:

% y\_true - **True** labels

% y\_pred - Predicted labels

% n\_bootstrap - Number **of** bootstrap iterations (**default**: 1000)

% OUTPUT:

% metrics - Struct **with** Accuracy, Sensitivity, Specificity, etc.

% confidence\_intervals - 95% CI **for** each metric

**if** nargin < 3, n\_bootstrap = 1000; **end**

N = length(y\_true);

% ===== Point Estimates =====

CM = confusionmat(y\_true, y\_pred);

TN = CM(1,1); FP = CM(1,2);

FN = CM(2,1); TP = CM(2,2);

metrics.Accuracy = (TP + TN) / (TP + TN + FP + FN);

metrics.Sensitivity = TP / (TP + FN); % Recall, TPR

metrics.Specificity = TN / (TN + FP); % TNR

metrics.Precision = TP / (TP + FP); % PPV

metrics.F1Score = 2 \* metrics.Precision \* metrics.Sensitivity / ...

(metrics.Precision + metrics.Sensitivity);

% ===== Bootstrap Resampling =====

bootstrap\_metrics = zeros(n\_bootstrap, 5); % [Acc, Sens, Spec, Prec, F1]

**for** iter = 1:n\_bootstrap

% Resample **with** replacement

idx = randsample(N, N, **true**);

y\_true\_boot = y\_true(idx);

y\_pred\_boot = y\_pred(idx);

% Compute metrics on bootstrap sample

CM\_boot = confusionmat(y\_true\_boot, y\_pred\_boot);

TN\_b = CM\_boot(1,1); FP\_b = CM\_boot(1,2);

FN\_b = CM\_boot(2,1); TP\_b = CM\_boot(2,2);

bootstrap\_metrics(iter, 1) = (TP\_b + TN\_b) / (TP\_b + TN\_b + FP\_b + FN\_b);

bootstrap\_metrics(iter, 2) = TP\_b / (TP\_b + FN\_b);

bootstrap\_metrics(iter, 3) = TN\_b / (TN\_b + FP\_b);

bootstrap\_metrics(iter, 4) = TP\_b / (TP\_b + FP\_b);

bootstrap\_metrics(iter, 5) = 2 \* bootstrap\_metrics(iter, 4) \* bootstrap\_metrics(iter, 2) / ...

(bootstrap\_metrics(iter, 4) + bootstrap\_metrics(iter, 2));

**end**

% ===== 95% Confidence Intervals (Percentile Method) =====

confidence\_intervals.Accuracy = [prctile(bootstrap\_metrics(:,1), 2.5), ...

prctile(bootstrap\_metrics(:,1), 97.5)];

confidence\_intervals.Sensitivity = [prctile(bootstrap\_metrics(:,2), 2.5), ...

prctile(bootstrap\_metrics(:,2), 97.5)];

confidence\_intervals.Specificity = [prctile(bootstrap\_metrics(:,3), 2.5), ...

prctile(bootstrap\_metrics(:,3), 97.5)];

confidence\_intervals.Precision = [prctile(bootstrap\_metrics(:,4), 2.5), ...

prctile(bootstrap\_metrics(:,4), 97.5)];

confidence\_intervals.F1Score = [prctile(bootstrap\_metrics(:,5), 2.5), ...

prctile(bootstrap\_metrics(:,5), 97.5)];

% Display results

fprintf('\n===== Performance Metrics (95%% CI) =====\n');

fprintf('Accuracy: %.3f [%.3f, %.3f]\n', metrics.Accuracy, confidence\_intervals.Accuracy);

fprintf('Sensitivity: %.3f [%.3f, %.3f]\n', metrics.Sensitivity, confidence\_intervals.Sensitivity);

fprintf('Specificity: %.3f [%.3f, %.3f]\n', metrics.Specificity, confidence\_intervals.Specificity);

fprintf('Precision: %.3f [%.3f, %.3f]\n', metrics.Precision, confidence\_intervals.Precision);

fprintf('F1-Score: %.3f [%.3f, %.3f]\n', metrics.F1Score, confidence\_intervals.F1Score);

**end**

**Statistical Validity:** Bootstrap resampling provides non-parametric confidence intervals without assuming normal distribution of metrics, which is appropriate for small test sets (N=194).

### B.6 Summary of Implementation Details

|  |  |  |  |
| --- | --- | --- | --- |
| **Component** | | **MATLAB Functions** | **Purpose** |
| **Preprocessing** | adapthisteq, imfilter, imadjust | | CLAHE, filtering, normalization |
| **Feature Extraction** | extractHOGFeatures, graycomatrix, graycoprops | | HOG, GLCM texture |
| **Feature Selection** | relieff, lassoglm, TreeBagger | | ReliefF, LASSO, tree importance |
| **Classification** | fitcsvm, TreeBagger, fitrensemble | | SVM, Random Forest, XGBoost |
| **Evaluation** | confusionmat, perfcurve, cvpartition | | Metrics, ROC, cross-validation |

Table 6 - Key Libraries and Functions Used

**Computational Environment:**

* **Platform:** MATLAB R2023b
* **Toolboxes Required:**
  + Image Processing Toolbox
  + Statistics and Machine Learning Toolbox
  + Computer Vision Toolbox
* **Hardware:** Standard CPU (Intel i7), 16GB RAM
* **Training Time:** ~5 minutes for complete pipeline

**Reproducibility Notes:**

1. **Random Seed:** Set to 7 for consistent patient splitting and cross-validation folds
2. **Image Preprocessing:** All images resized to 128×128 before feature extraction
3. **Feature Normalization:** Z-score normalization applied before SVM/XGBoost training
4. **Hyperparameter Search:** Grid search with 3-fold cross-validation
5. **Evaluation:** Fixed 85/15 train/test split with patient-disjoint guarantee

## Appendix C - Supplementary Figures

### C.1 SVM performance summary

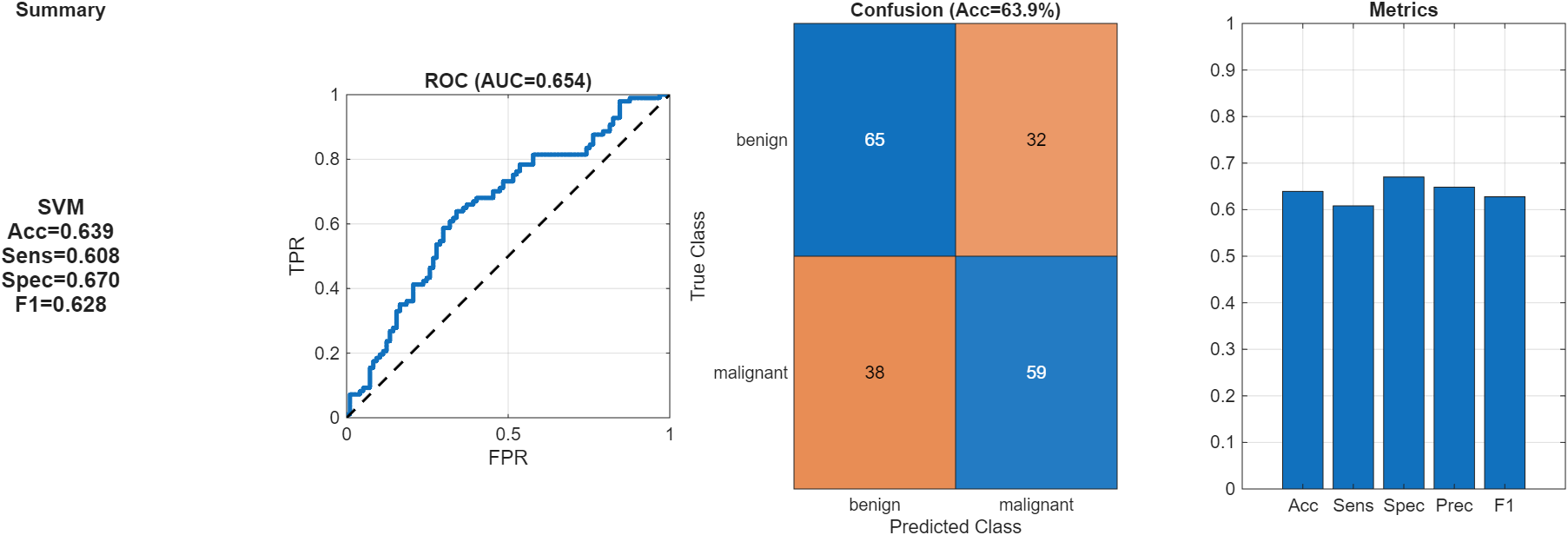


Figure 18 - SVM performance summary

### C.2 RF performance summary

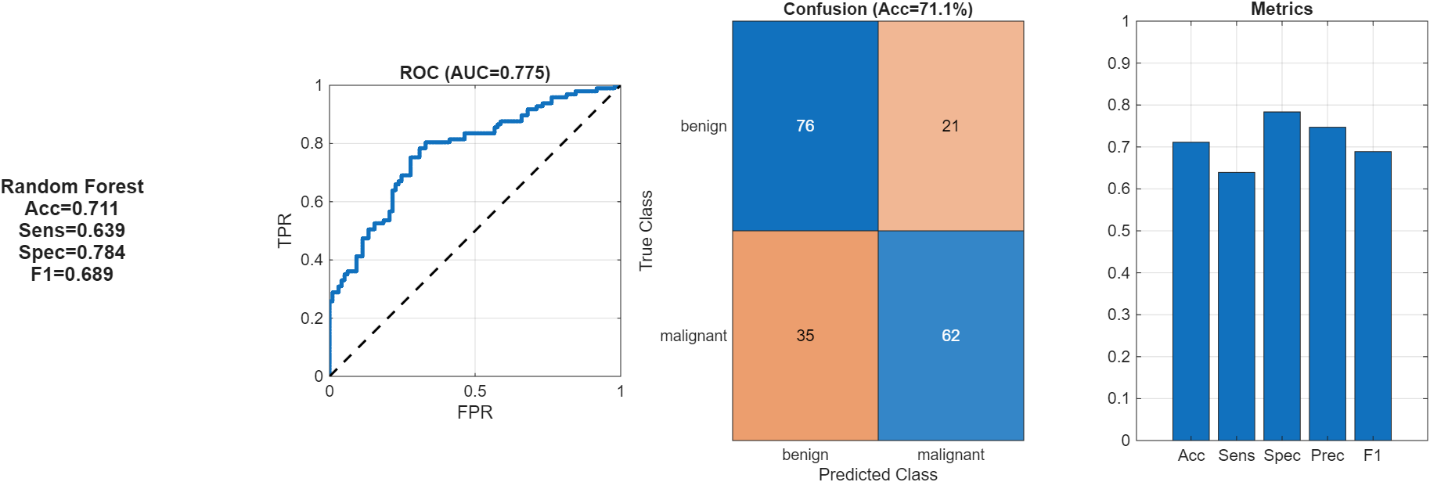


Figure 19 - RF performance summary

### C.3 XGBoost performance summary

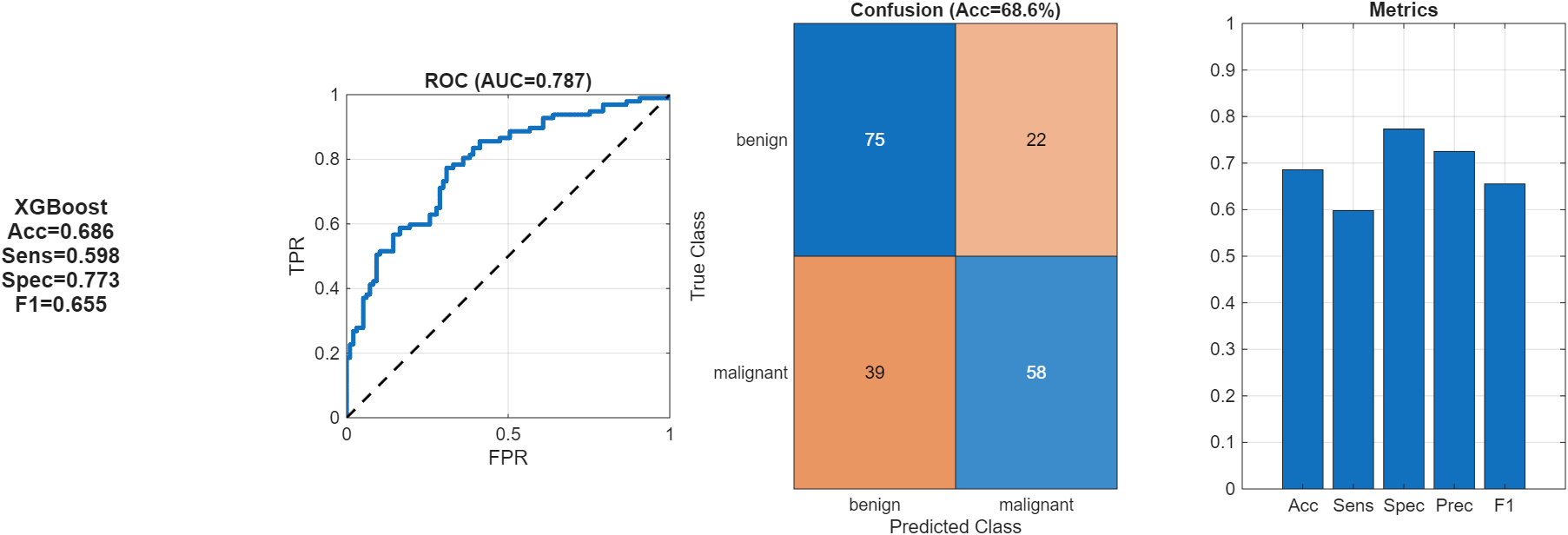


Figure 20 - XGBoost performance summary

### C.4 ReliefF feature importance

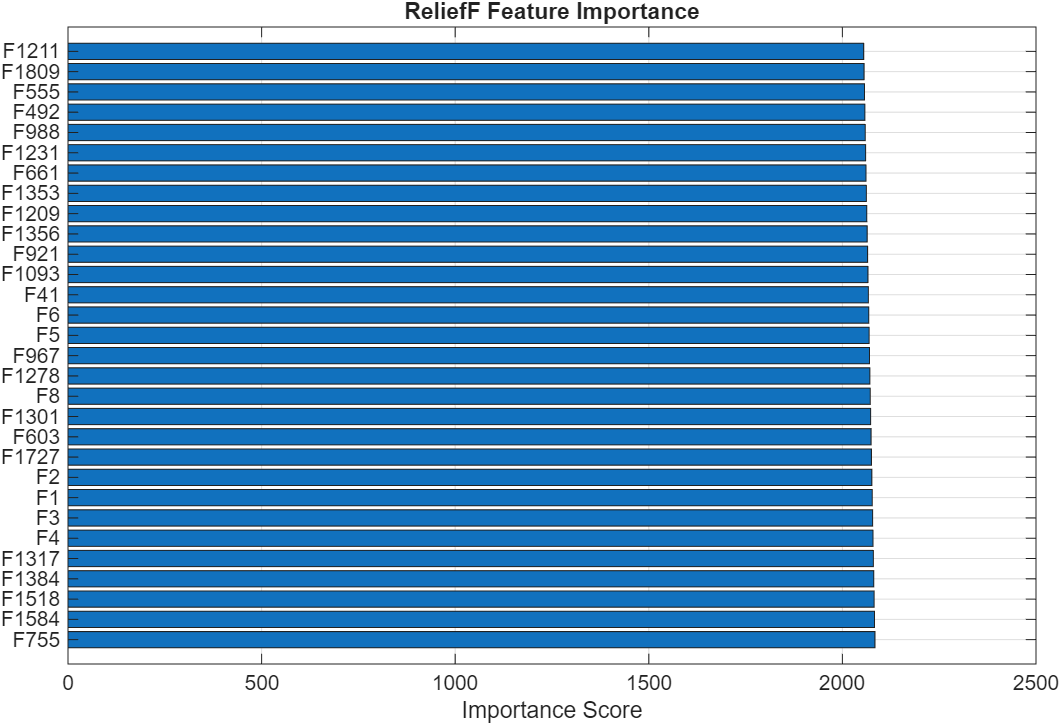


Figure 21 - ReliefF feature importance

### C.5 PCA analysis

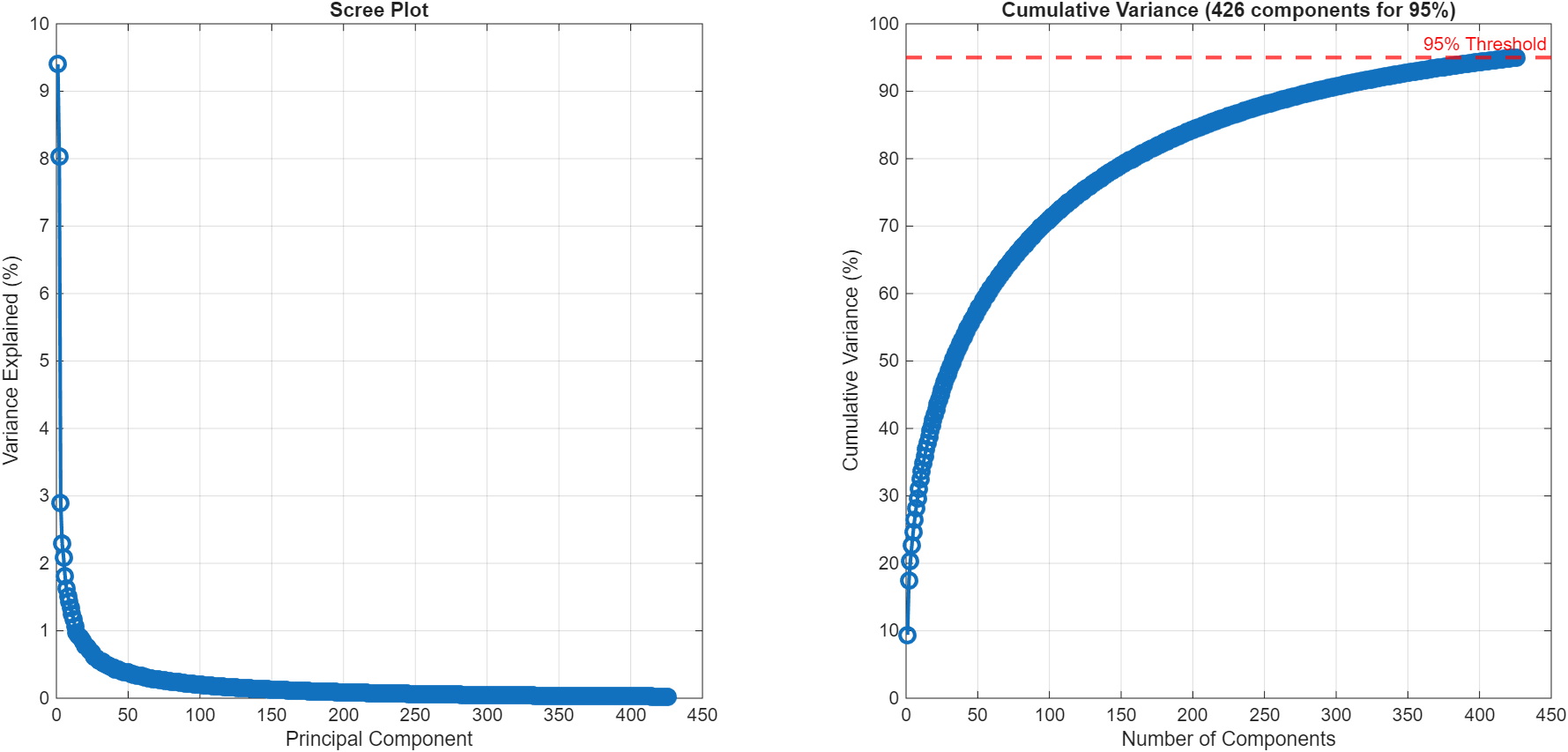


Figure 22 - PCA Analysis

### C.6 Confusion breakdown

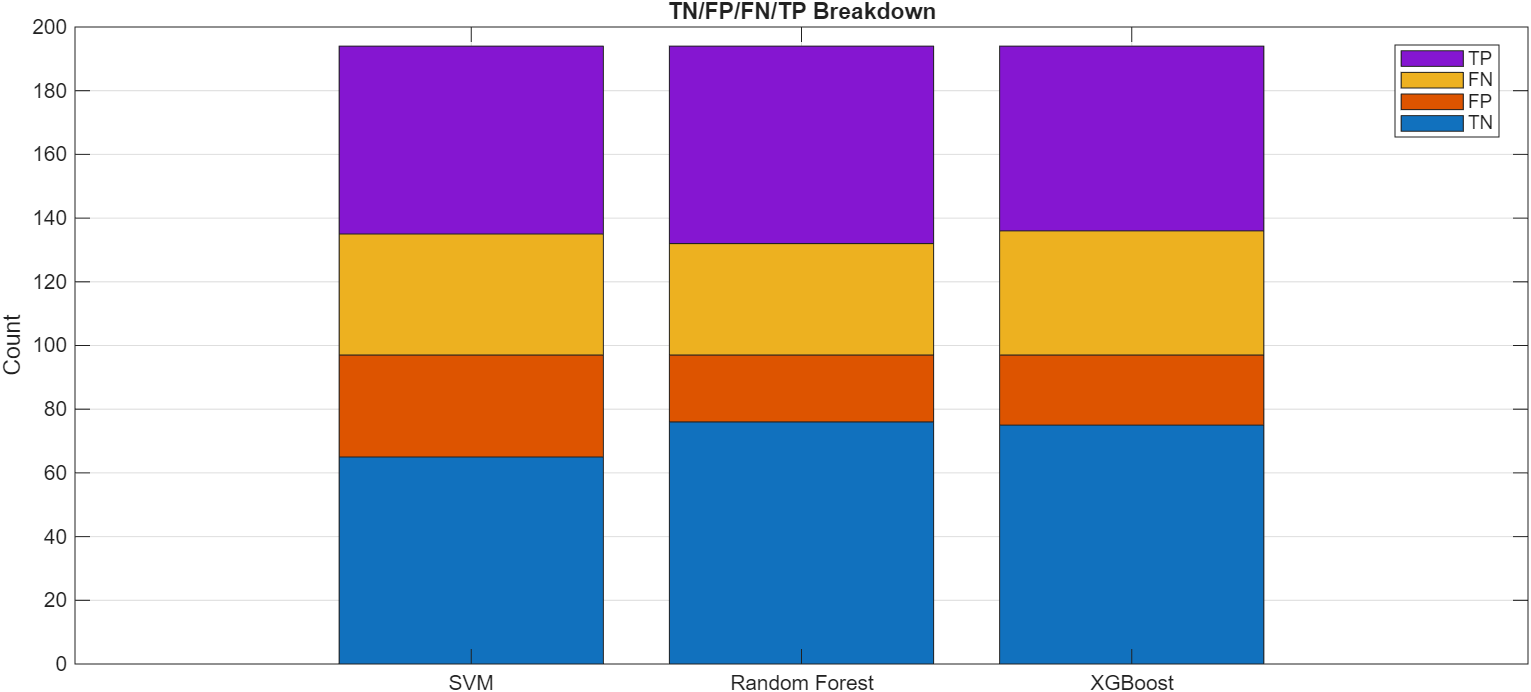


Figure 23 - True Negative, False Positive, False Negative, and True Positive Breakdown by Classifier