# AutoML Analysis Report

Generated on: July 01, 2025 at 23:23
MH-AutoML System

# Executive Summary

This report presents the results of an automated machine learning analysis conducted using the MH-AutoML system. The analysis includes comprehensive data preprocessing, feature engineering, model optimization, and interpretability insights.

# Pipeline Configuration

The AutoML pipeline consists of the following steps:

- 0. Data Info - Dataset validation and information
- 1. Preprocessing - Data cleaning and transformation
- 2. Feature Engineering - Dimensionality reduction and selection
- 3. Model Optimization - Hyperparameter tuning with Optuna
- 4. Interpretability - SHAP and LIME explanations
- 5. Evaluation - Model assessment and reporting

# 0. Data Info

Dataset validation and information analysis:

## System Information:

| Operating System | Windows 10 |
|---|---|
| Python Version | 3.9.7 |
| Architecture | 64bit |
| Total RAM | 31.74 GB |
| Available RAM | 15.40 GB |
| CPU Cores | 8 |
| CPU Usage | 49.6% |

## Dataset Information:

| Info | Value |
|---|---|
| Rows | 15036 |
| Columns | 51 |

## Data Types Analysis:

| Data Type | Count |
|---|---|
| float64 | 51 |

## Class Balance Analysis:

| Label | Percentage |
|---|---|
| 0.0 | 63.01% |
| 1.0 | 36.99% |

## Android Features Analysis:

| Permissions found | API_Calls found |
|---|---|
| 50 | 0 |

# 1. Data Preprocessing - Cleaning and preparation

In this step, the system performs data cleaning, removes missing values, duplicates, and handles outliers, as well as applies necessary encodings. Proper preprocessing ensures the quality and reliability of the data for malware detection.

## Preprocessing Visualizations:

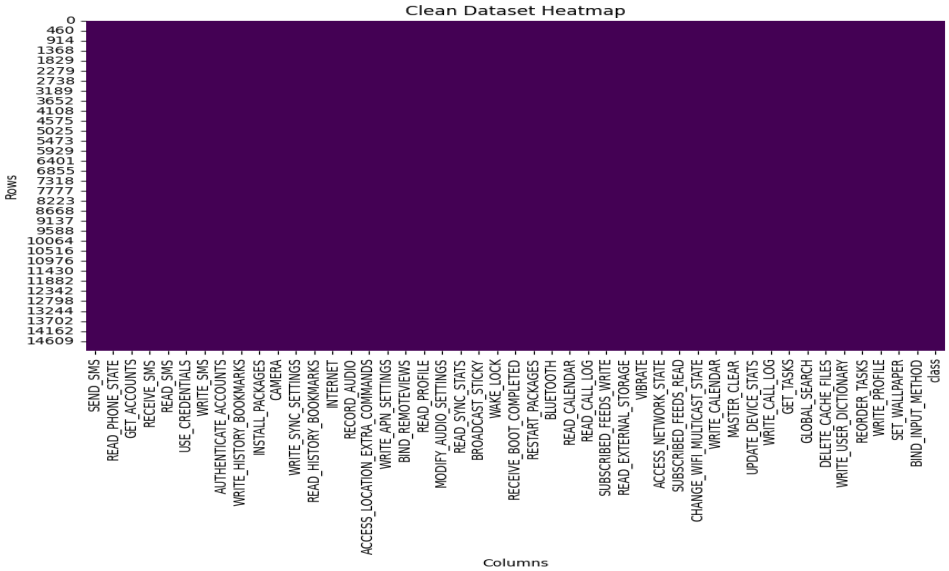


Figure: clean_missing_values_heatmap.png - This heatmap shows the distribution of missing values after data cleaning. It demonstrates the effectiveness of preprocessing in preparing the dataset for modeling.

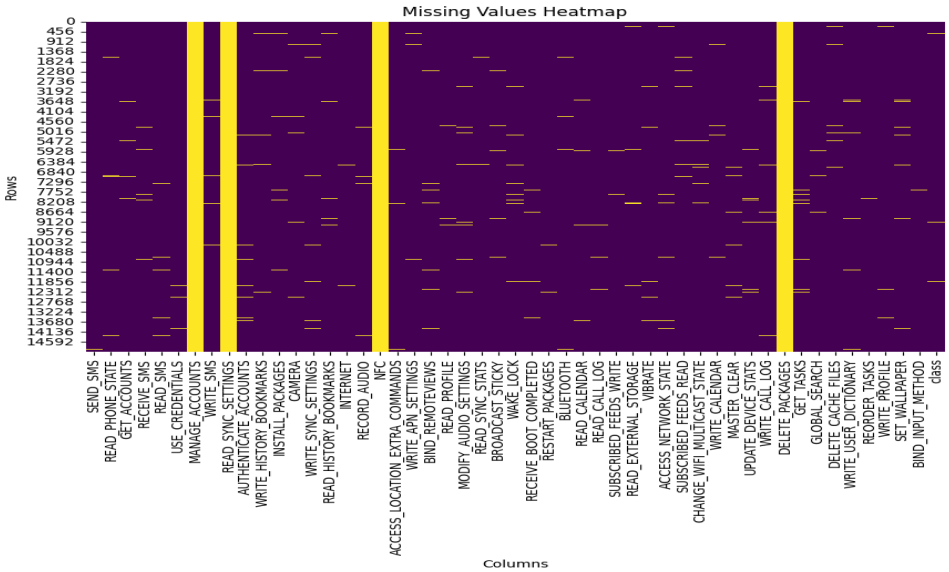


Figure: missing_values_heatmap.png - This heatmap shows the distribution of missing values in the original dataset. It helps identify which features may have incomplete data, which is crucial for reliable malware detection.

# 2. Feature Engineering - Selection and transformation

Feature engineering includes selection and transformation of variables, using techniques such as PCA, LASSO, and ANOVA for dimensionality reduction and selection of the best variables for malware detection.

## Feature Engineering Visualizations:

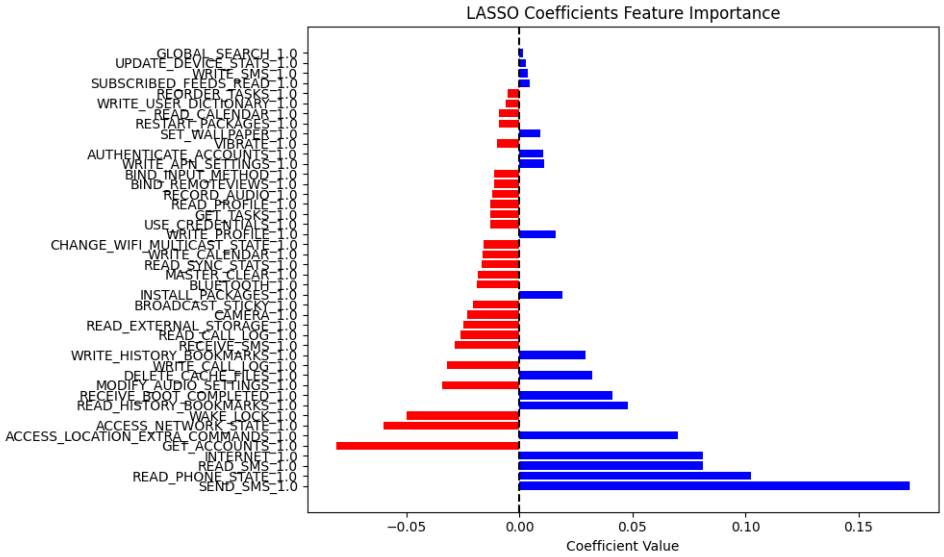


Figure: lasso_feature_importance.png - LASSO feature importance shows which application characteristics are most relevant for malware detection. Features with higher importance are more critical for distinguishing between benign and malicious applications.

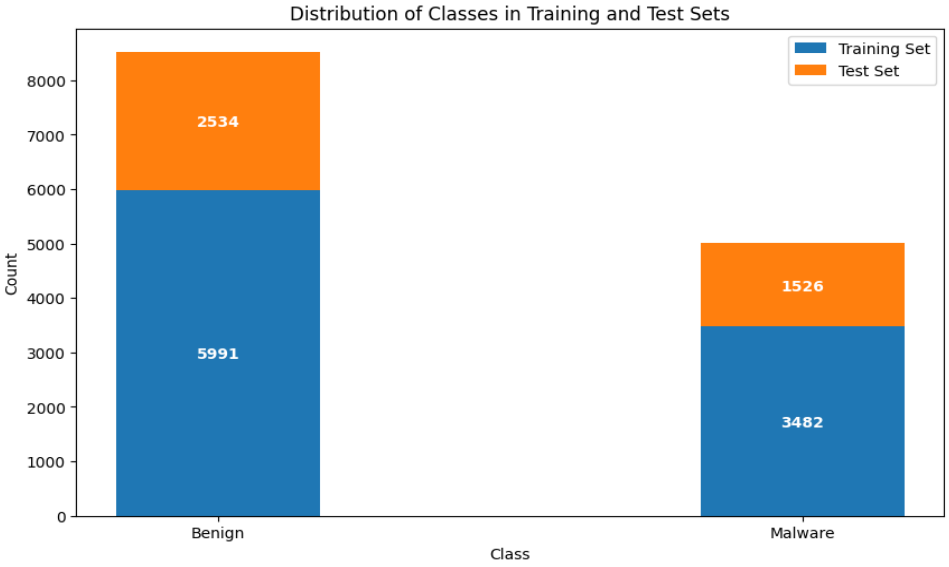


Figure: train_test_distribution.png - Train/test distribution shows the balance of malware and benign samples between training and test sets. This ensures representative data distribution for reliable model evaluation.

# 3. Model Optimization - Hyperparameter tuning

Model optimization through hyperparameter tuning using Optuna, seeking the best possible performance for malware detection accuracy and reliability.

## Model Optimization Visualizations:

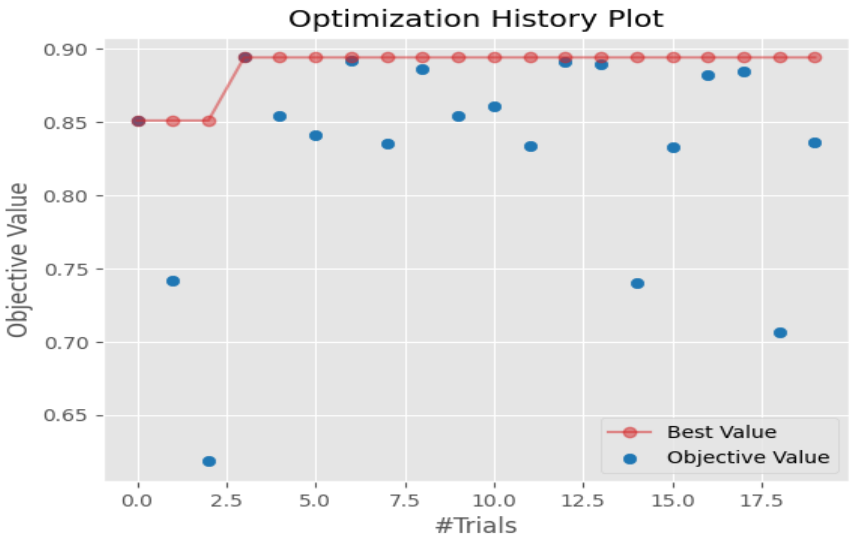


Figure: optuna_optimization_history.png - Optimization history shows how model performance improved during hyperparameter tuning. Each trial represents a different configuration tested for malware detection accuracy.

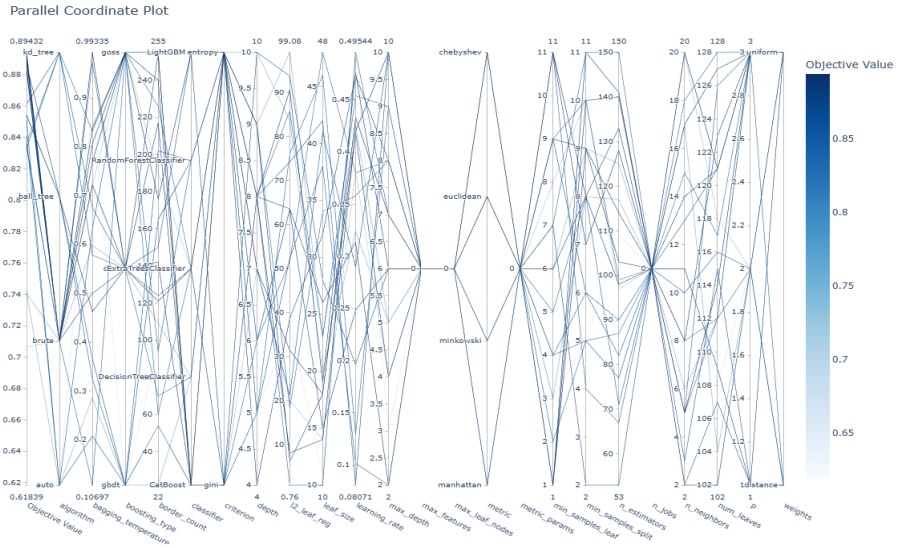


Figure: optuna_parallel_coordinate.png - Parallel coordinates plot shows the relationship between different hyperparameter combinations and their impact on malware detection performance.

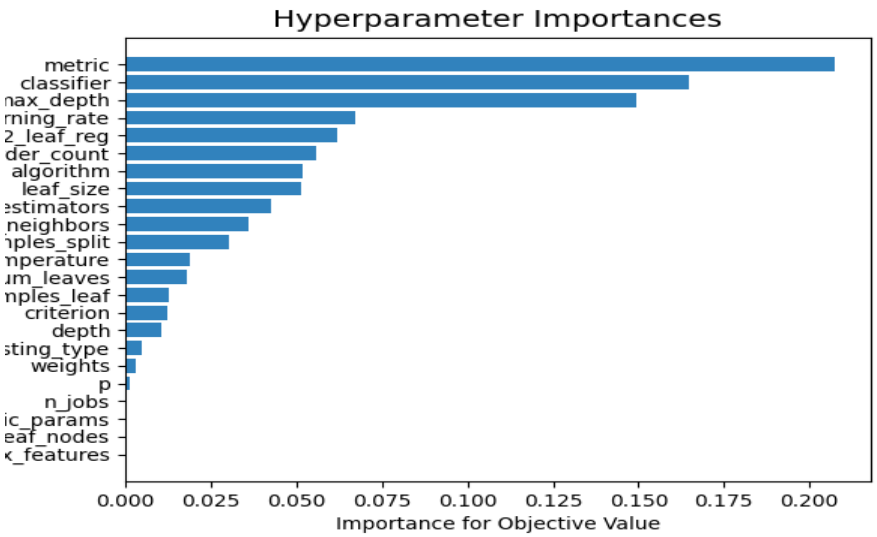


Figure: optuna_param_importance.png - Parameter importance shows which hyperparameters most influence malware detection performance. This helps focus optimization efforts on the most critical parameters.

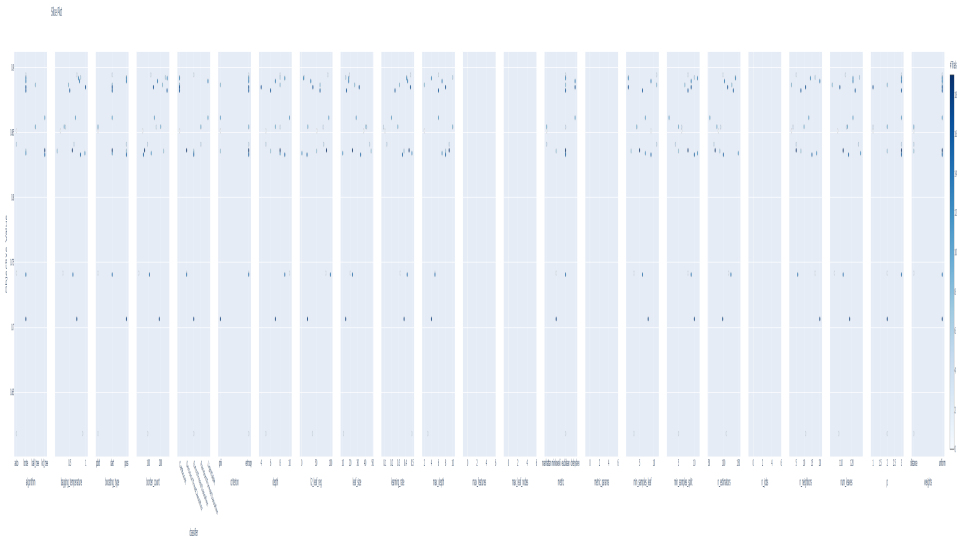

Figure: optuna_slice_plot.png - Slice plot shows the sensitivity of model performance to individual hyperparameter values. This helps identify optimal ranges for each parameter in malware detection.

# 4. Model Evaluation - Performance assessment

Model evaluation with metrics such as accuracy, precision, recall, F1-score, ROC-AUC, and confusion matrix to assess malware detection performance.

## Model Evaluation Visualizations:

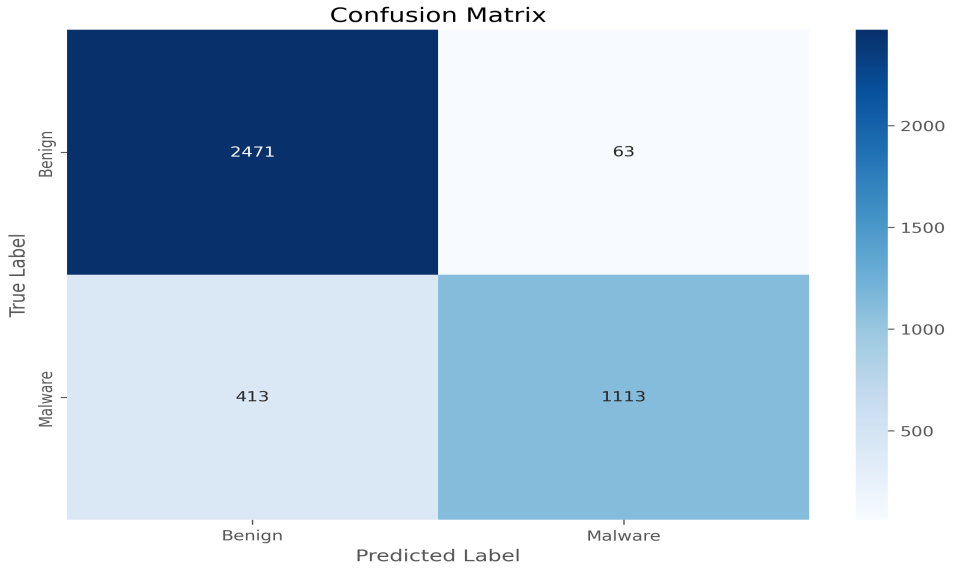


Figure: confusion_matrix.png - Confusion matrix shows the model's classification performance. True positives (correctly identified malware) and true negatives (correctly identified benign apps) are crucial for security. False positives (benign apps flagged as malware) and false negatives (undetected malware) represent different types of security risks.

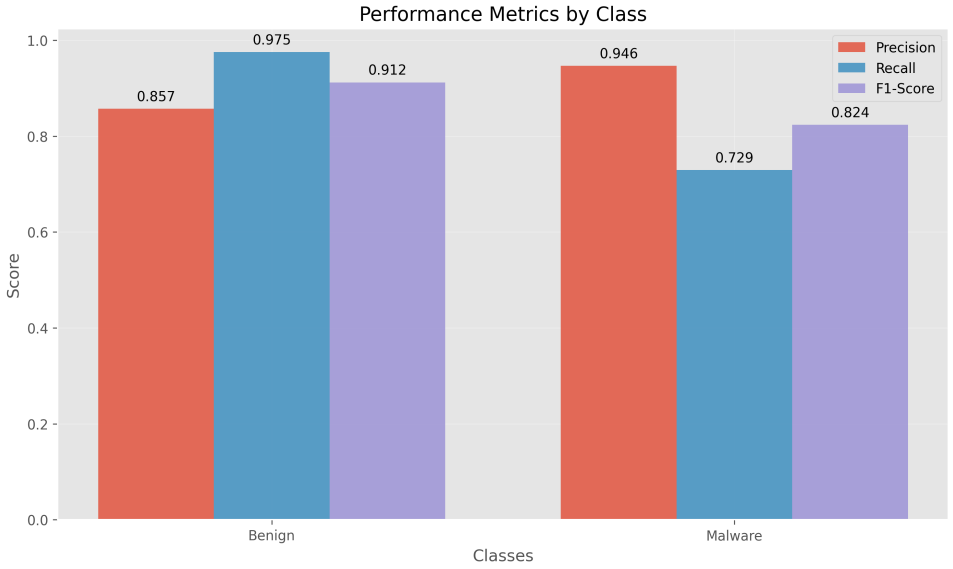


Figure: metrics_by_class.png - Metrics by class compares specific performance for benign vs. malicious applications. This allows adjusting the model to prioritize malware detection or avoid false positives based on security requirements.

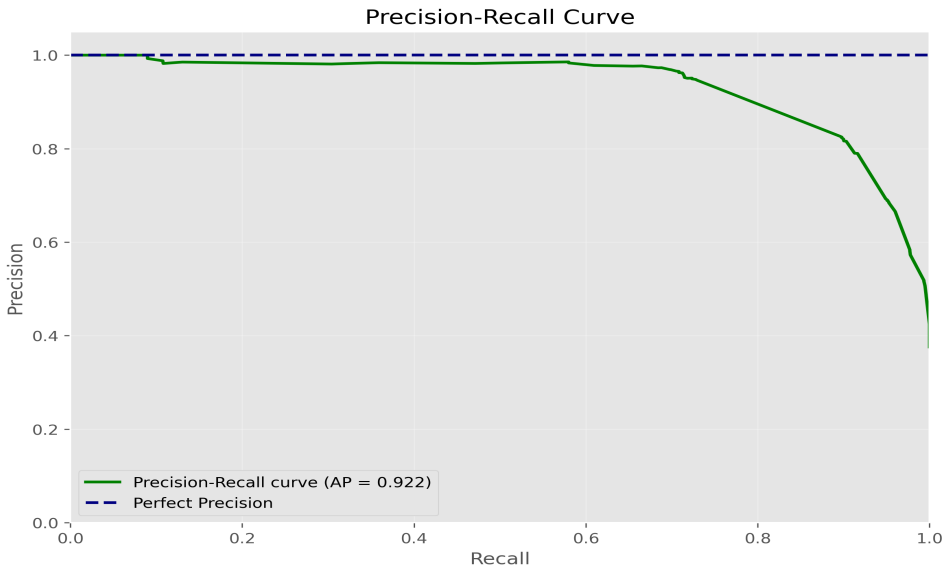


Figure: precision_recall_curve.png - Precision-recall curve is especially important for malware detection where the positive class (malware) is often minority. High precision reduces false positives, while high recall minimizes undetected malware.

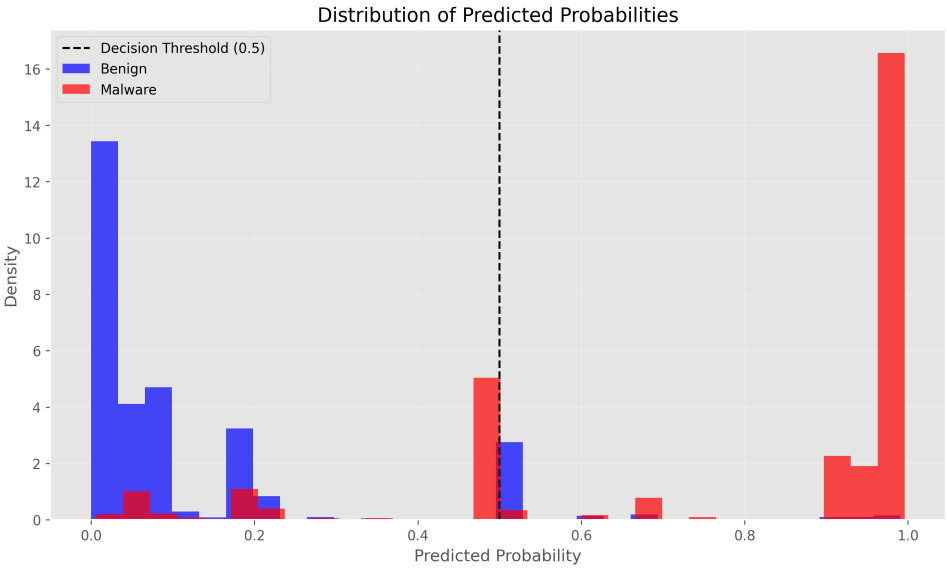


Figure: probability_distribution.png - Probability distribution shows how the model assigns malware probabilities to applications. Good separation between benign and malicious probability distributions indicates confident predictions.

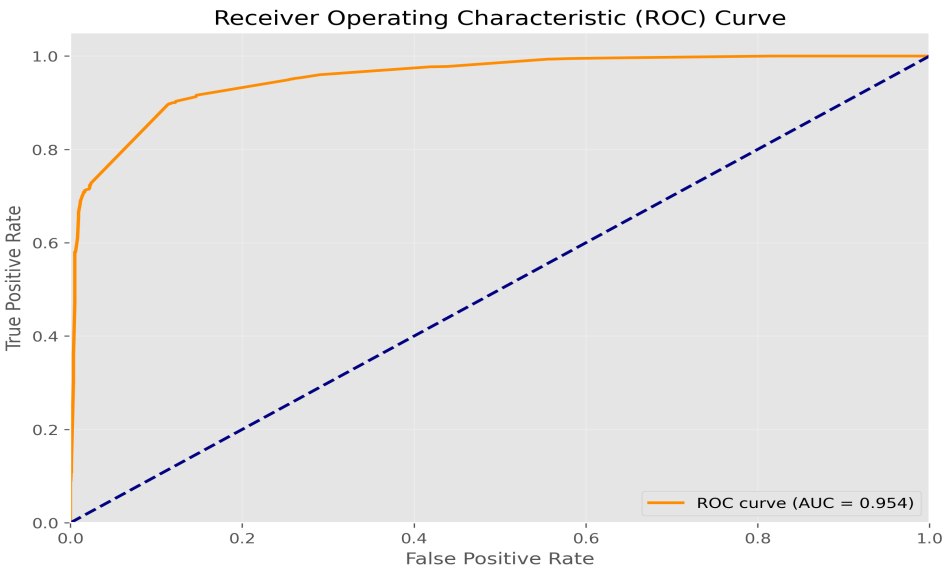


Figure: roc_curve.png - ROC curve shows the model's ability to distinguish between malware and benign applications across different classification thresholds. Higher AUC indicates better discrimination capability.

# 5. Interpretability - SHAP and LIME analysis

Model interpretability analysis using SHAP and LIME to explain model decisions and identify the most important variables for malware detection.

## Interpretability Visualizations:

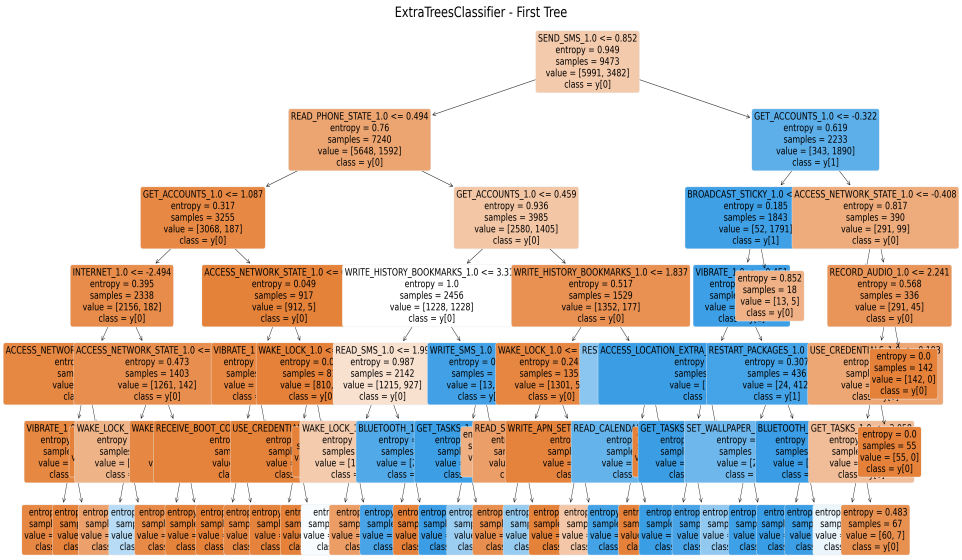


Figure: decision_tree_plot_ExtraTreesClassifier_20250701_232317.png - Decision tree visualization shows the model's decision rules. Each node represents a feature split that helps distinguish between malware and benign applications.

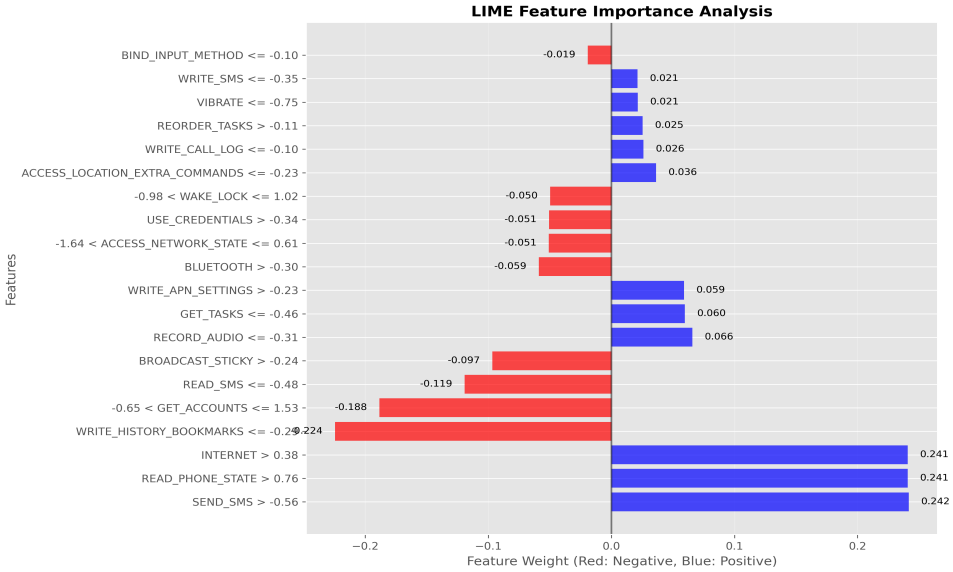


Figure: lime_feature_importance_20250701_232317.png - LIME feature importance shows the weight of each feature in the model's decision for a specific case. Positive values favor malware classification, negative values favor benign classification.

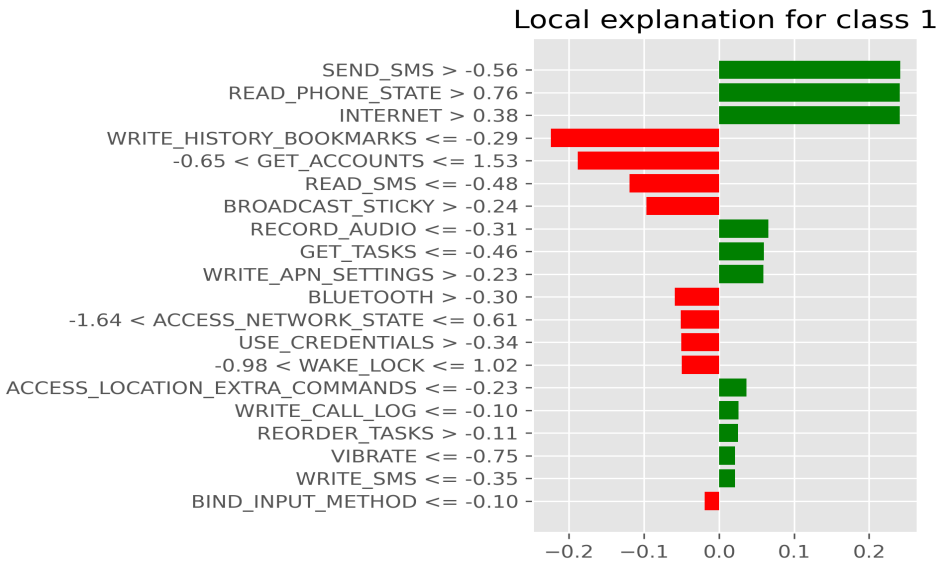


Figure: lime_interpretability_20250701_232317.png - LIME explanation provides local interpretability for a specific prediction. It shows which features contributed most to classifying this particular application as malware or benign.

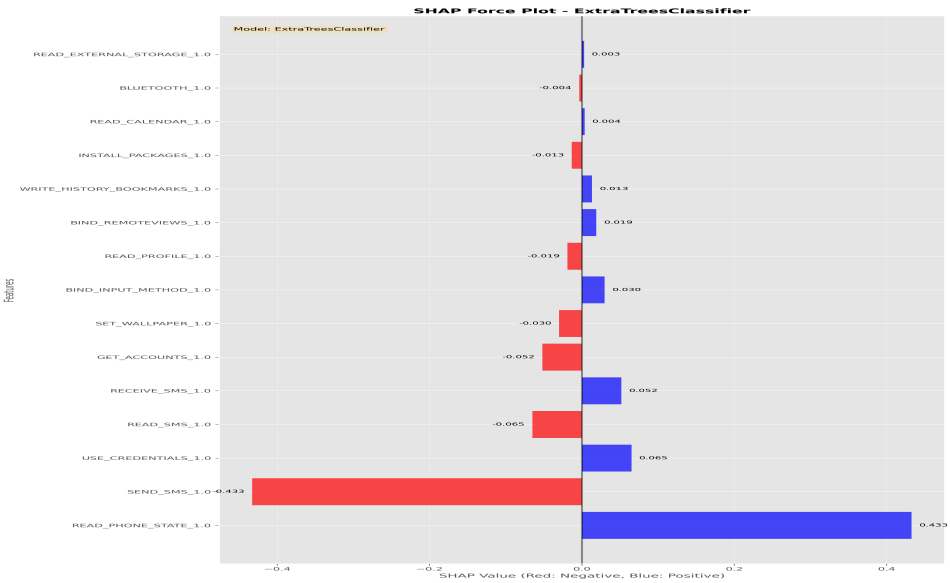


Figure: shap_force_plot_ExtraTreesClassifier_20250701_232317.png - SHAP force plot shows detailed contribution of each feature to a specific malware prediction. Red bars push toward malware classification, blue bars push toward benign classification.

# Generated Artifacts

Complete list of all generated files:

## 1. Data Preprocessing:

■ Visualizations:

• clean_missing_values_heatmap.png

• missing_values_heatmap.png

## 2. Feature Engineering:

■ Visualizations:

• confusion_matrix.png

• decision_tree_plot_ExtraTreesClassifier_20250701_232317.png

• lasso_feature_importance.png

• metrics_by_class.png

• precision_recall_curve.png

• probability_distribution.png

• roc_curve.png

• train_test_distribution.png

■ Data Files:

• Features_Selected_20250701_232221.csv

• treino_20250701_232146.csv

## 3. Model Optimization:

■ Visualizations:

• optuna_optimization_history.png

• optuna_parallel_coordinate.png

• optuna_param_importance.png

• optuna_slice_plot.png

■ Data Files:

• Hyperparameters_Results.csv

• Models_Ranking.csv

• optuna_trials.csv

■ Other Files:

• optuna_optimization_history.html

• optuna_parallel_coordinate.html

• optuna_param_importance.html

• optuna_slice_plot.html

## 4. Model Evaluation:

■ Data Files:

• best_model_20250701_232146.pkl

## 5. Interpretability:

■ Visualizations:

• lime_feature_importance_20250701_232317.png

• lime_interpretability_20250701_232317.png

• shap_force_plot_ExtraTreesClassifier_20250701_232317.png

■ Other Files:

• lime_interpretability_20250701_232317.html

## Reports:

■ Other Files:

• report_20250701_232332.html

Note: This is a simplified PDF version. For the complete interactive report with all visualizations and detailed formatting, please refer to the HTML version.