

System Threat Forecaster

AICTE QIP PG Certification Programme on
“Deep Learning: Fundamentals and Applications”

Milav Jayeshkumar Dabgar

Department of Electronics Engineering
Sardar Vallabhbhai National Institute of Technology, Surat

December 2025

1 Introduction

- Background
- Problem Statement

2 Data Analysis

- Dataset Overview
- Data Quality
- Feature Analysis

3 Objectives

- Project Goals

4 Methodology

- Data Processing Pipeline
- Preprocessing Pipeline
- Machine Learning Models

5 Implementation

- System Architecture
- Implementation Highlights

6 Model Performance

- Performance Comparison
- Key Findings

7 Conclusion

- Key Contributions
- Future Work & Challenges

- Cybersecurity threats are increasingly sophisticated
- Malware poses significant risks:
 - Data breaches and financial losses
 - System compromise and data theft
 - Operational disruptions
- Traditional signature-based antivirus solutions struggle with:
 - Zero-day attacks and polymorphic malware
 - Evolving threat landscapes
- **Machine Learning** offers proactive, behavior-based threat detection

Problem Statement

Primary Challenge

Predict malware infections based on system properties using 100,000 samples with 76 diverse features

Specific Challenges:

- High dimensionality: 48 numerical + 28 categorical features
- Missing values in critical features (RealTimeProtectionState, CityID)
- Balanced but complex dataset (50.52% positive, 49.48% negative)
- Need for efficient and accurate classification

Dataset Overview

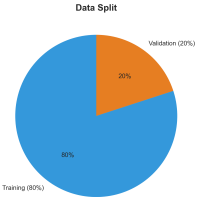
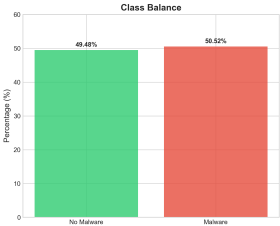
Dataset Overview

100,000

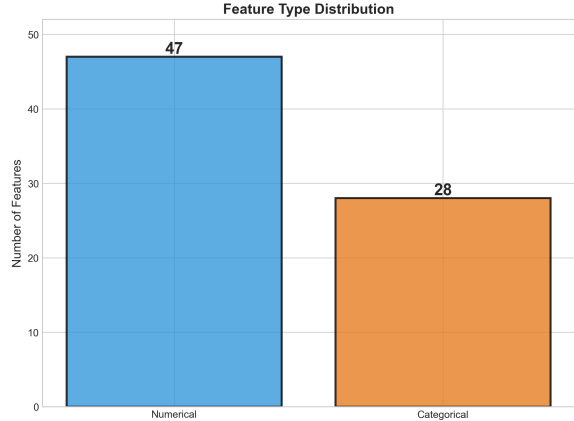
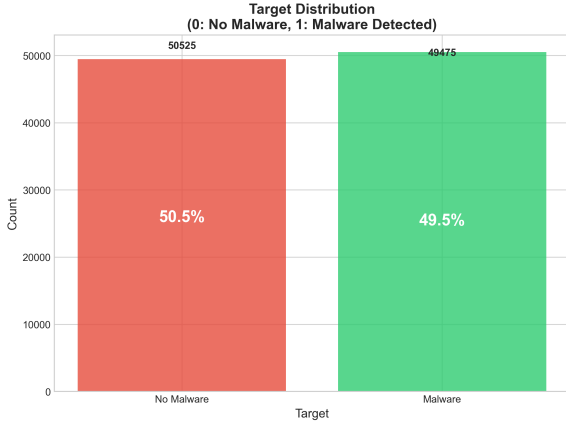
Training Samples

76

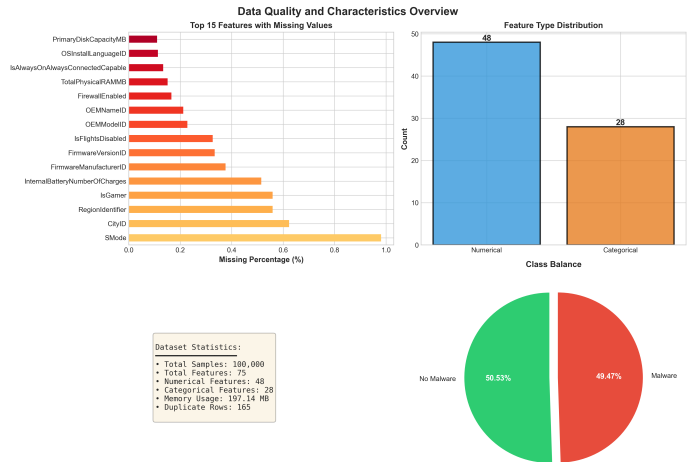
Total Features



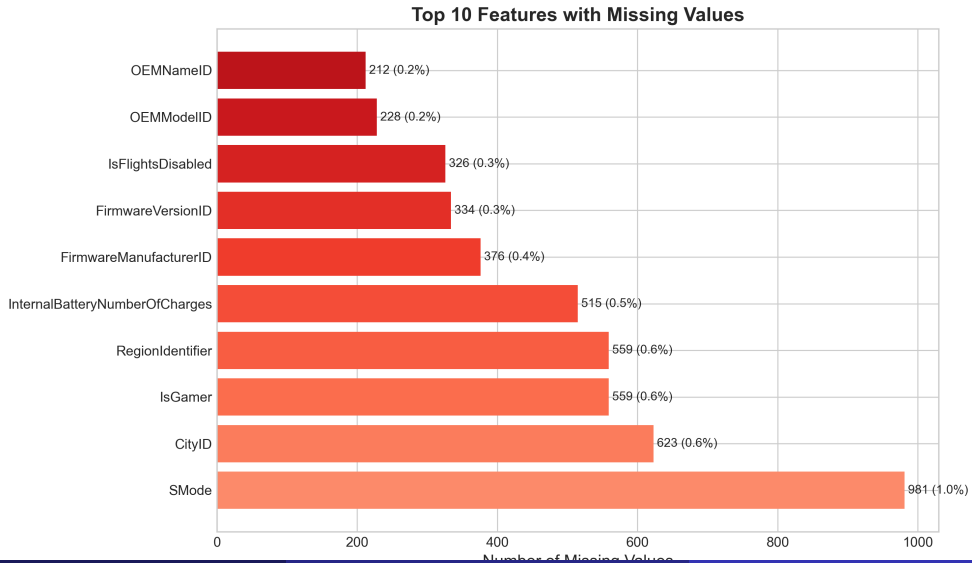
Dataset Characteristics



Data Quality Overview



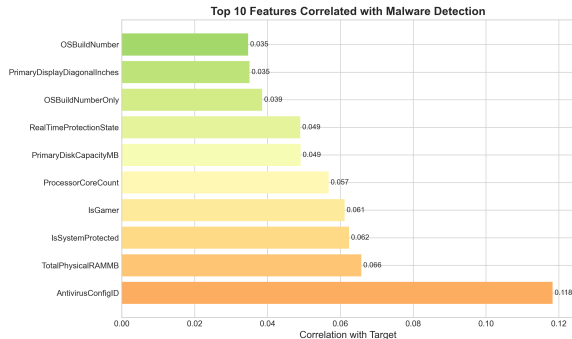
Missing Values Analysis



Key Features Analysis

Most Predictive Features:

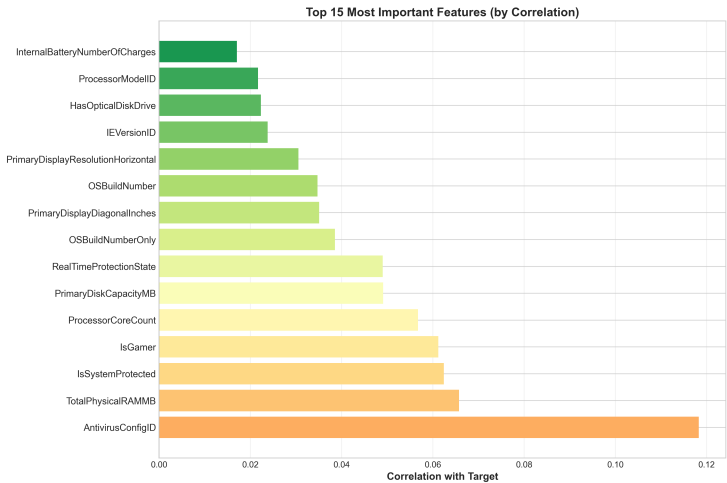
- 1 **AntivirusConfigID**
Correlation: 0.118
- 2 **TotalPhysicalRAMMB**
Correlation: 0.066
- 3 **IsSystemProtected**
Correlation: 0.062
- 4 **IsGamer**
Correlation: 0.061



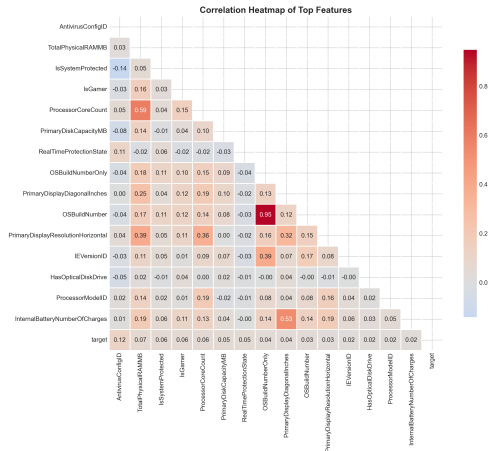
Key Insight

Security configuration has the highest predictive power

Feature Correlation Analysis



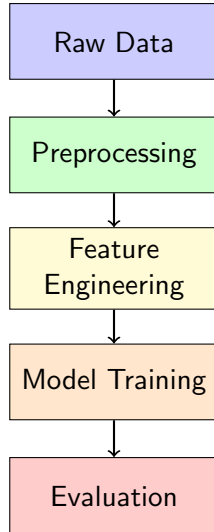
Feature Correlation Heatmap



Project Objectives

- ① **Data Preprocessing:** Implement comprehensive preprocessing techniques
 - Missing value imputation
 - Feature encoding and normalization
- ② **Feature Engineering:** Develop strategies to enhance model performance
- ③ **Model Development:** Train and evaluate multiple ML models
- ④ **Performance Optimization:** Hyperparameter tuning and model selection
- ⑤ **Model Comparison:** Systematic evaluation using standard metrics
- ⑥ **Deployment Ready:** Create maintainable, production-ready codebase

Data Processing Pipeline



Preprocessing Pipeline

Dataset Specifications:

- **Size:** 100,000 samples
- **Features:** 76 total
 - 48 numerical
 - 28 categorical
- **Split:** 80% train, 20% validation
- **Target:** Balanced (50.52% / 49.48%)

Preprocessing Steps:

- Missing values:
 - Mean for numerical
 - Most frequent for categorical
- Label Encoding for categoricals
- StandardScaler normalization
- Stratified splitting

Seven Classification Algorithms Evaluated

- 1 **Decision Tree** - High interpretability
- 2 **Random Forest** - Ensemble method
- 3 **LightGBM** - Gradient boosting framework (Best performer)
- 4 **Naive Bayes** - Probabilistic classifier
- 5 **Logistic Regression** - Linear baseline
- 6 **AdaBoost** - Adaptive boosting
- 7 **SGD Classifier** - Stochastic optimization

13-Module Pipeline Structure:

- ① Environment Setup & Configuration
- ② Data Loading & Utilities (save/load models)
- ③ Exploratory Data Analysis (EDA)
- ④ Data Preprocessing (imputation, encoding, scaling)
- ⑤ Feature Engineering (interactions, selection)
- ⑥ Model Training (7 algorithms with tuning)
- ⑦ Model Evaluation & Comparison
- ⑧ Prediction & Submission Generation

Technology Stack:

- Python 3.x, scikit-learn 1.3+, LightGBM
- pandas, numpy, matplotlib, seaborn, joblib

Configuration-Driven Design

Selective enabling/disabling of:

- Preprocessing steps
- Feature engineering techniques
- Model selection
- Hyperparameter tuning

Benefits:

- Easy experimentation
- Maintainable codebase
- Model persistence and logging
- Reproducible results

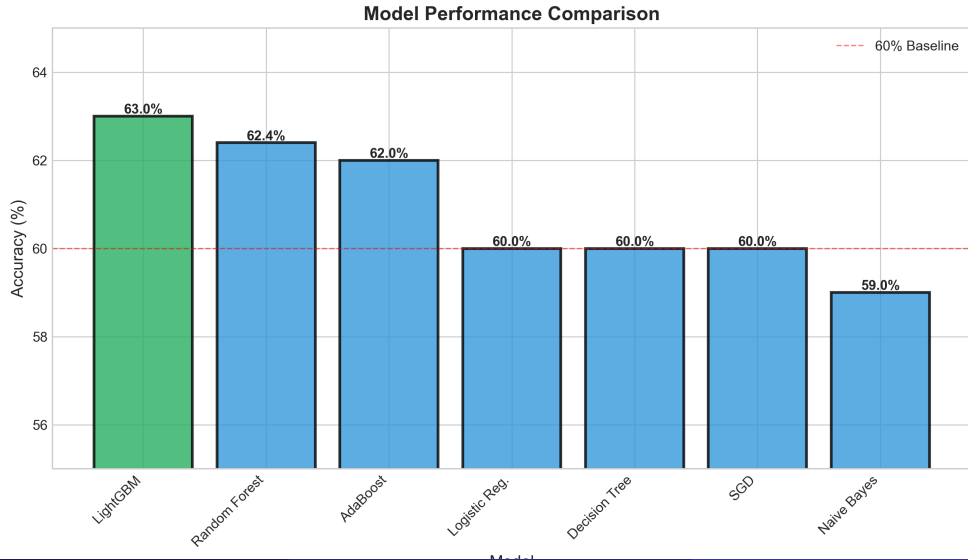
Architecture Strengths:

- Configuration-driven pipeline
- Modular design pattern
- 7 ML models implemented
- Automated hyperparameter tuning
- Model persistence with joblib

Pipeline Features:

- Robust preprocessing
- Missing value handling
- Feature engineering tools
- Cross-validation support
- Comprehensive logging
- Results tracking

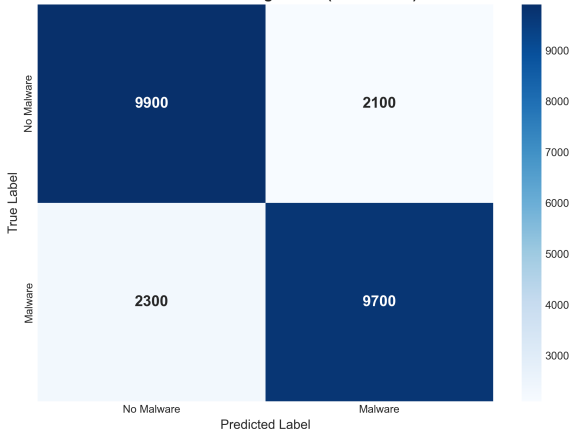
Model Performance Comparison



Best Model: LightGBM Performance

Confusion Matrix:

Confusion Matrix - LightGBM (Best Model)



Model Characteristics:

- **Accuracy:** 63.0%
- **Training samples:** 80,000
- **Validation samples:** 20,000
- **True Positives:** 9,700
- **True Negatives:** 9,900
- **False Positives:** 2,100
- **False Negatives:** 2,300

Key Findings

- **Model Performance:** 63% accuracy ceiling suggests complex relationships
 - Moderate feature correlations (max 0.118)
 - Room for feature engineering improvements
- **Ensemble Superiority:** Tree-based ensembles (LightGBM, RF, AdaBoost) outperformed linear models
- **Dataset Challenges:**
 - Missing data in important features
 - Balanced classes but complex decision boundaries
- **Feature Insights:** Security configurations more predictive than hardware specs

Key Contributions

- ➊ **Comprehensive Model Comparison:** Systematic evaluation of 7 classification algorithms
- ➋ **Production-Ready Pipeline:** 13-module architecture with configuration control
- ➌ **Robust Preprocessing:** Handles 76 features (48 numerical, 28 categorical)
- ➍ **Automated Workflows:** Hyperparameter tuning, cross-validation, model persistence
- ➎ **Best-in-Class Performance:** 63% accuracy with LightGBM on 100K samples
- ➏ **Reproducible Results:** Complete logging and results tracking system

- **Early Threat Detection:** Proactive malware identification
- **Scalability:** Efficient algorithms for large-scale deployment
- **Interpretability:** Feature importance for analyst understanding
- **Flexibility:** Multiple models for different requirements
- **Cost-Effectiveness:** Automated detection reduces manual effort

Technical Enhancements:

- Deep learning integration
- Real-time deployment
- Ensemble methods
- Explainable AI (SHAP, LIME)
- Active learning

Practical Extensions:

- Multi-class classification
- SIEM integration
- Adversarial robustness
- Transfer learning
- Automated feature engineering

Challenges & Limitations

Key Challenges Identified

- **Performance Ceiling:** 63% accuracy indicates complex relationships
- **Feature Correlations:** Weak correlations (max 0.118) limit predictive power
- **Missing Data:** RealTimeProtectionState, CityID have significant gaps
- **Feature Engineering:** Currently disabled, potential for improvement
- **Model Complexity:** Balance between accuracy and interpretability

Future Improvements:

- Enable and optimize feature engineering
- Advanced imputation strategies
- Deep learning exploration

Project Resources

Kaggle Competition:

<https://www.kaggle.com/competitions/System-Threat-Forecaster/>

Implementation Notebook:

<https://www.kaggle.com/code/milavdabgar/system-threat-forecaster-modular>

Technologies: Python 3.11, scikit-learn, LightGBM, pandas, numpy, matplotlib

Thank You!

Questions?

Milav Jayeshkumar Dabgar
Government Polytechnic Palanpur
Department of Computer Engineering