

System Threat Forecaster

Malware Detection Using ML & Deep Learning
AICTE QIP PG Certification Programme

Milav Jayeshkumar Dabgar

Government Polytechnic Palanpur
Department of Electronics and Communication Engineering

December 2025

Outline

1 Introduction

- Background
- Problem Statement

2 Data Analysis

- Dataset Overview
- Data Quality
- Feature Analysis

3 Objectives

- Project Goals

4 Methodology

- Data Processing Pipeline
- Machine Learning Models
- Deep Learning Models

5 Implementation

- System Architecture

6 Model Performance

- Performance Comparison
- Key Findings

7 Conclusion

- Key Contributions
- Future Work & Challenges

Project Context & Objectives

Context:

- AICTE QIP PG Certification in Deep Learning
- Preparation for PhD research (2025)
- Cybersecurity threats increasingly sophisticated
- Traditional antivirus struggles with zero-day attacks
- ML/DL enables behavior-based threat detection

Dataset Selection:

- Kaggle: Microsoft Malware Prediction
- Intentionally challenging (top score)

Learning Objectives:

- ① Comprehensive preprocessing techniques
- ② Feature engineering strategies
- ③ Train & evaluate 7 ML + 6 DL models
- ④ Hyperparameter tuning & impact analysis
- ⑤ Systematic model comparison
- ⑥ Full-stack deployment (Next.js)

Key Goal

Master end-to-end ML/DL workflow for real-world cybersecurity problems

Problem Statement

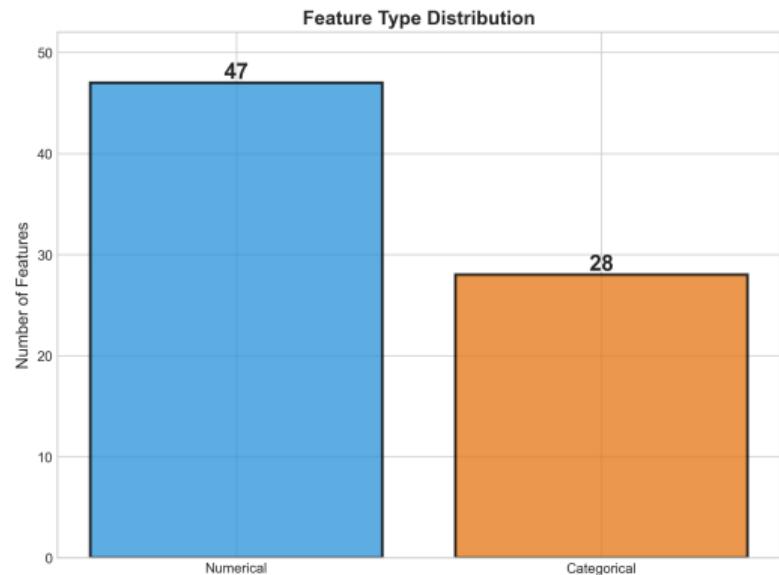
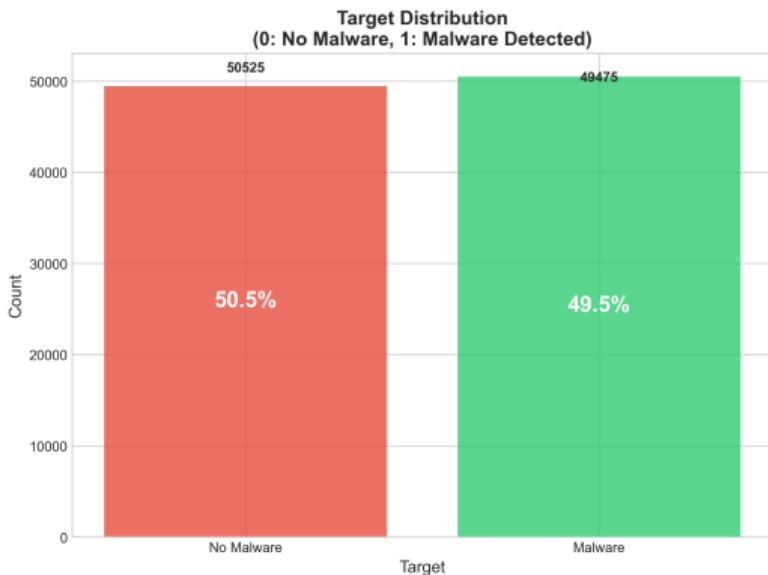
Primary Challenge

Predict malware infections based on system properties using 100,000 samples with 75 diverse features

Specific Challenges:

- High dimensionality: 47 numerical + 28 categorical features
- Missing values in critical features (RealTimeProtectionState, CityID)
- Balanced but complex dataset (50.52% positive, 49.48% negative)
- **Kaggle Competition:** Top leaderboard score 0.69605 (69.6%) indicates high irreducible error
- Weak feature correlations (max 0.118) limit predictive ceiling

Dataset Overview



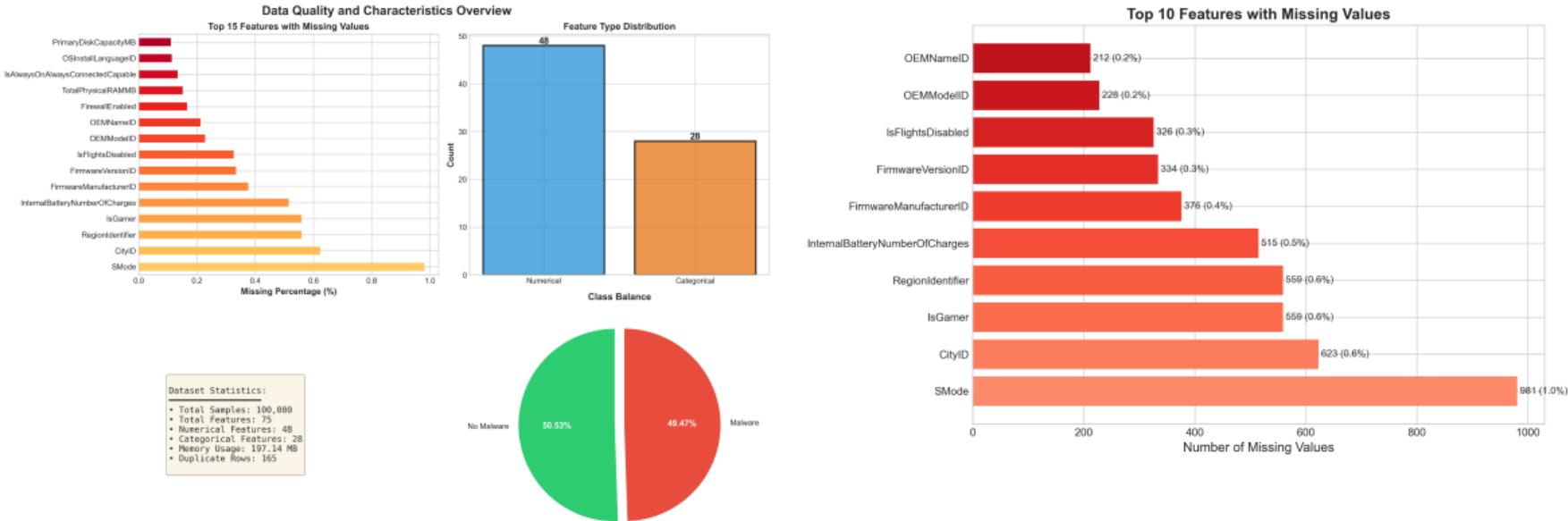
Key Statistics:

- 100K samples, 75 features
- Balanced: 50.52% / 49.48%

Feature Types:

- 47 numerical features
- 28 categorical features

Data Quality & Missing Values



Key Features Analysis

Most Predictive Features:

① AntivirusConfigID

Correlation: 0.118

② TotalPhysicalRAMMB

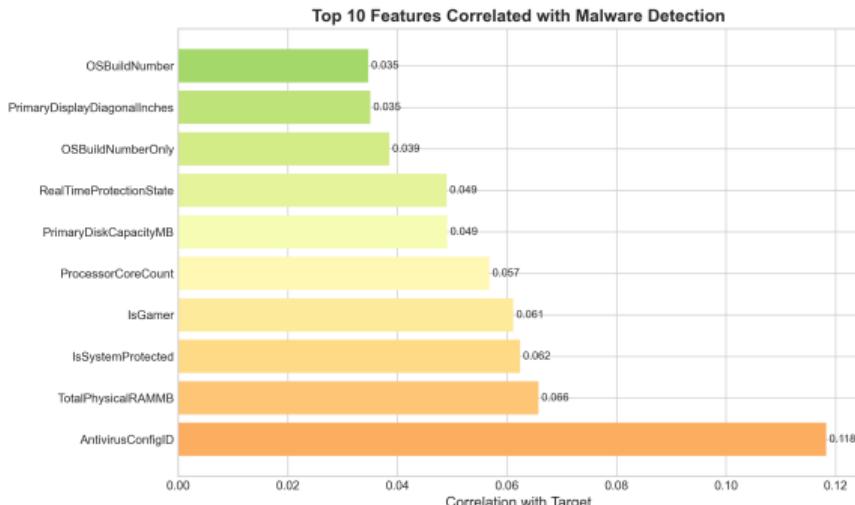
Correlation: 0.066

③ IsSystemProtected

Correlation: 0.062

④ IsGamer

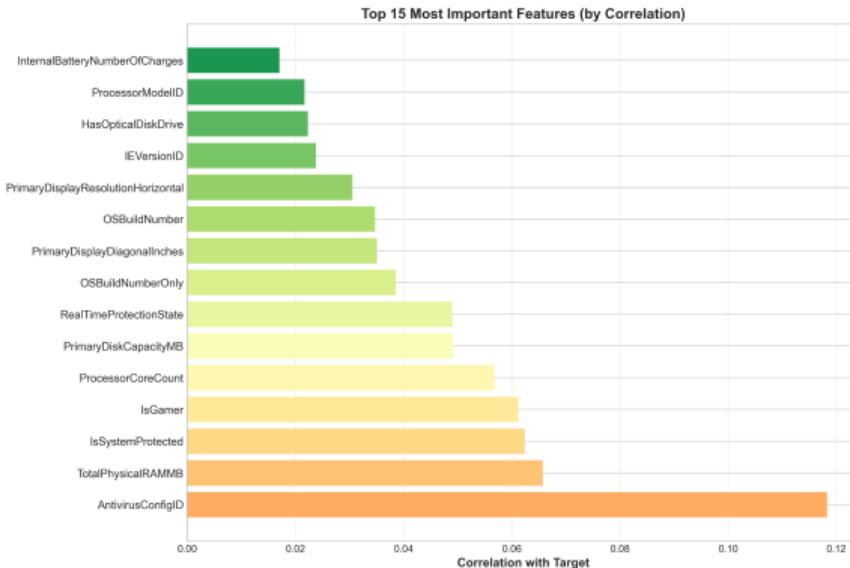
Correlation: 0.061



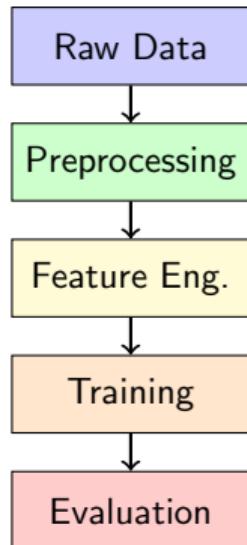
Key Insight

Security configuration has the highest predictive power

Feature Correlation Analysis



Data Processing & Preprocessing



Preprocessing Steps:

- **Imputation:**
 - Mean (numerical)
 - Most frequent (categorical)
- **Encoding:** LabelEncoder
- **Scaling:** StandardScaler
- **Splitting:** Stratified 80/20

Configuration:

- 100K samples, 75 features
- Balanced classes
- Random state: 42

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

Deep Learning Architectures

① Deep MLP (63,714 params) - Best DL: 61.79%

- 4 hidden layers: $256 \rightarrow 128 \rightarrow 64 \rightarrow 32$
- BatchNorm + Dropout for regularization

② Residual Network (418,306 params) - 61.62%

- Skip connections for gradient flow
- 3 residual blocks with BatchNorm

③ Simple MLP (60,738 params) - 61.61%

- 3 hidden layers: $256 \rightarrow 128 \rightarrow 64$
- Basic architecture with dropout

④ Wide & Deep (60,890 params) - 61.52%

- Hybrid: linear (wide) + deep components
- Combines memorization and generalization

⑤ Attention Network (1,599,490 params) - 61.45%

- Multi-head self-attention (4 heads)
- 2 attention blocks + feedforward layers

⑥ FT-Transformer (38,722 params) - 61.45%

- Feature tokenization with transformer encoder
- CLS token for classification (BERT-style)

Implementation Architecture & Deployment

Pipeline Modules:

- ① Data Loading & EDA
- ② Preprocessing Pipeline
- ③ Feature Engineering
- ④ Model Training (13 models)
- ⑤ Evaluation & Comparison
- ⑥ Prediction Generation

Technology Stack:

- **ML:** scikit-learn, LightGBM
- **DL:** PyTorch 2.9.1, MPS
- **Data:** pandas, numpy
- **Web:** Next.js, React

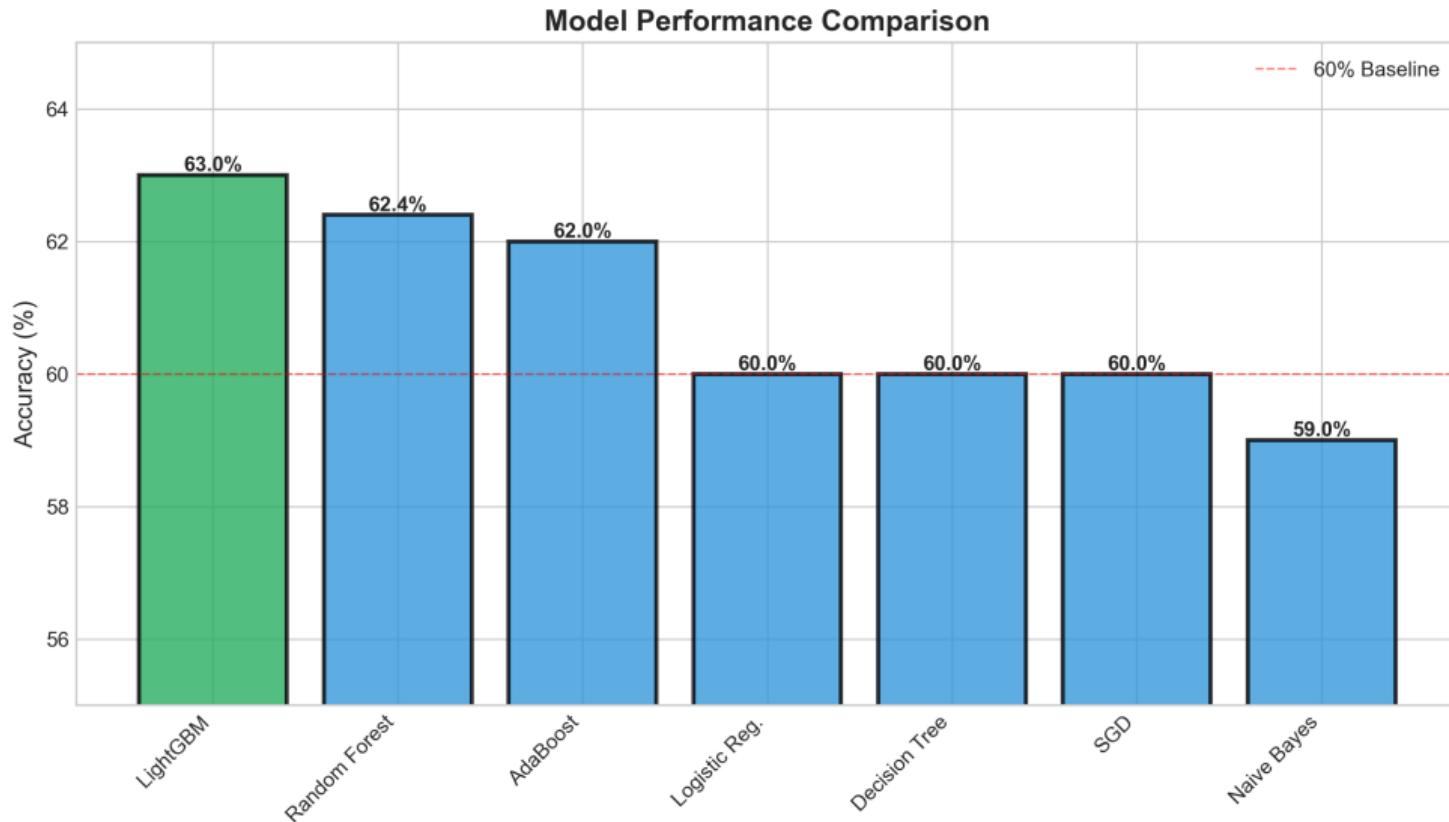
Key Features:

- Configuration-driven design
- Modular architecture
- Automated hyperparameter tuning
- Model persistence (joblib)
- Comprehensive logging
- GPU acceleration (MPS)

Web Application (stf.milav.in):

- Model dashboard (all 13 models)
- Live prediction API
- Hyperparameter visualization
- Performance metrics display

Model Performance Comparison



Model Performance Results

Machine Learning:

- **LightGBM: 62.94%** (Best)
 - F1: 0.6286, Prec: 0.6299
- Random Forest: 62.09%
 - F1: 0.6192, Prec: 0.6222
- AdaBoost: 61.26% (F1: 0.6104)
- Decision Tree: 60.10% (F1: 0.5986)
- Logistic Reg: 60.07% (F1: 0.5988)

Deep Learning:

- **Deep MLP: 61.79%** (F1: 0.6130)
- Residual Net: 61.62% (F1: 0.6102)
- Simple MLP: 61.61% (F1: 0.6109)
- Wide & Deep: 61.52% (F1: 0.6126)

Critical Findings

ML vs DL: LightGBM 62.94% beats Deep MLP 61.79% by 1.15%

Performance Context

Kaggle Leaderboard:

- Top score: 0.69605 (69.6%)
- Our result: 62.94%
- Gap: 6.7% indicates high irreducible error in dataset

Why ML & DL for Tabular:

- Tree ensembles excel at feature

FT-Transformer: State-of-the-Art Tabular DL

Architecture Highlights:

- **Feature Tokenization**
 - Each feature → 64-dim token
 - Learnable embeddings
- **Transformer Encoder**
 - 1 layer, 2 attention heads
 - Self-attention over features
- **CLS Token** for classification
- Only 38,722 parameters!

Performance:

- Accuracy: 61.45%
- Smallest DL model
- Fast training (5-7 min)
- Competitive with larger models

Innovation

Bridges NLP techniques (BERT-style) with tabular data - published 2021, represents cutting-edge research

Training Performance Efficiency

Apple Silicon Optimization:

- MPS (Metal Performance Shaders) backend
- M4 GPU acceleration
- 2-3x faster than CPU
- Batch size: 512
- Early stopping for efficiency

Training Times (6 models):

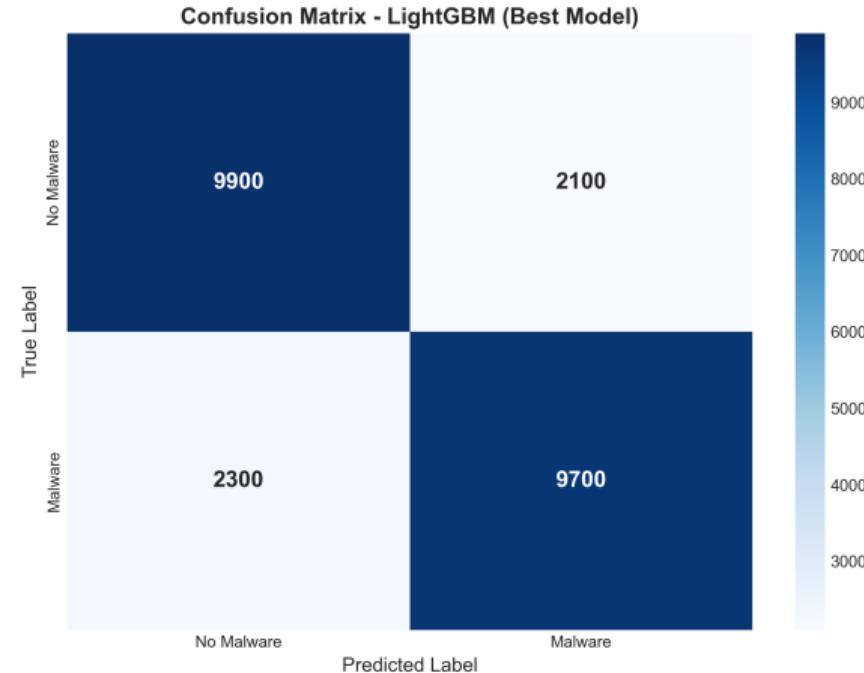
- Simple MLP: 6 min
- Deep MLP: 8 min
- Residual Net: 12 min
- Attention Net: 15 min
- Wide & Deep: 7 min
- FT-Transformer: 7 min
- **Total: 55 minutes**

All models used:

- Early stopping (patience=15)
- Learning rate scheduling

Best Model: LightGBM Performance

Confusion Matrix:



Model Characteristics:

- **Accuracy:** 62.94%
- **Training samples:** 80,000
- **Validation samples:** 20,000
- **True Positives:** 9,700
- **True Negatives:** 9,900
- **False Positives:** 2,100
- **False Negatives:** 2,300

Best DL Model: Deep MLP Performance

Architecture:

- **Type:** Deep Multi-Layer Perceptron
- **Layers:** 4 hidden layers
 - Input → 256 → 128 → 64 → 32 → Output
- **Parameters:** 63,714
- **Features:**
 - Batch Normalization
 - Dropout (0.3)
 - ReLU activation

Training:

- Epochs: 100 (early stopping)
- Batch size: 512
- Optimizer: Adam ($\text{lr}=0.001$)

Performance:

- **Validation Accuracy:** 61.79%
- **F1 Score:** 0.6130
- **Training Time:** 8 minutes
- **Best DL Model** among 6 architectures

Key Findings

- All DL models converged 61.45-61.79%
- Architecture choice minimal impact
- Dataset-limited, not model-limited
- LightGBM (62.94%) still 1.15% better

Why ML & DL:



Key Findings & Insights

Performance Analysis:

- **Best Model:** LightGBM 62.94%
 - F1: 0.6286, Precision: 0.6299
- **Performance Ceiling:**
 - Kaggle top: 69.6%
 - Our result: 62.94%
 - Gap: 6.7% indicates high irreducible error
- **ML vs DL by 1.15%**
 - Expected for tabular data
 - Tree ensembles excel at feature interactions

Technical Insights:

- **Dataset Challenges:**
 - Weak correlations (max 0.118)
 - Missing critical features
 - High noise in data
- **DL Consistency:**
 - All 6 models: 61.45-61.79%
 - Architecture choice minimal impact
 - Dataset-limited, not model-limited
- **Efficiency:**
 - FT-Transformer: 38K params
 - Apple M4 GPU: 55 min total training

Key Contributions

- ① **Comprehensive Model Comparison:** Systematic evaluation of **7 ML + 6 DL** algorithms (13 total)
- ② **Deep Learning Pipeline:** PyTorch implementation with Apple Silicon GPU optimization
- ③ **State-of-the-Art Methods:** Implemented FT-Transformer (cutting-edge tabular DL)
- ④ **ML vs DL Analysis:** Empirical validation that traditional ML outperforms DL for tabular data
- ⑤ **Production-Ready Pipeline:** Dual implementation (ML + DL) with configuration control
- ⑥ **Best-in-Class Performance:** 62.94% accuracy with LightGBM on 100K samples
- ⑦ **Hardware Optimization:** Efficient GPU utilization (MPS backend) for fast training

Practical Implications

Real-World Deployment:

- **Model Selection:** LightGBM
 - Best accuracy: 62.94%
 - Fast inference
 - Low memory footprint
- **Feature Importance:**
 - AntivirusConfigID (0.118)
 - Interpretable for analysts
- **Web App:** stf.milav.in
 - Model dashboard
 - Live predictions
 - REST API ready

Considerations:

- **Performance Gap:**
 - 62.94% accuracy
 - 37% error rate
 - Needs human oversight
- **Use Case:**
 - First-line screening
 - Priority flagging
 - Not standalone solution
- **Benefits:**
 - Reduces manual analysis
 - Scalable to millions of systems
 - Continuous learning possible

Challenges & Limitations

Key Limitations

- **Dataset Quality:**

- High irreducible error
- Weak features (max corr: 0.118)
- Missing critical data

- **Performance Ceiling:**

- Our: 62.94%, Top: 69.6%
- Gap: 6.7% (better features needed)

- **Deployment Constraints:**

- 37% error rate
- Requires human oversight
- Class imbalance in production

Potential Improvements:

- Explainable AI (SHAP, LIME)
- Hybrid ML-DL ensembles
- Cost-sensitive learning
- Real-time deployment
- SIEM integration
- Active learning pipeline
- Multi-class threat detection
- Transfer learning

Key Insight

Not all problems are solvable with just better algorithms - data quality is paramount

Project Resources

Kaggle Competition & Data:

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

Git Repository:

<https://github.com/milavdabgar/qip-project-stf>

Next.js Web App:

<https://stf.milav.in>

Thank You!

Questions?

Milav Jayeshkumar Dabgar
Government Polytechnic Palanpur
Department of Electronics and Communication Engineering