System Threat Forecaster

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

Milav Jayeshkumar Dabgar

Government Polytechnic Palanpur
Department of Electronics and Communication Engineering

December 2025

Outline

1. Project Context & Learning Objectives
    - Problem Statement
2. Data & Methodology
    - Dataset Overview
    - Methodology
3. Model Experimentation & Learning
    - Machine Learning
    - Deep Learning
4. Implementation & Deployment
    - System Architecture
5. Conclusion
    - Key Findings
    - Challenges & Limitations

Problem Statement: Objectives & Challenges

Goal

Predict malware infections and compare ML vs DL performance on tabular data

Key Objectives:

1. Kaggle System Threat Forecaster
2. Implement 7 ML algorithms
3. Build 6 DL architectures
4. ML vs DL comparison
5. Full-stack deployment
6. Production web app

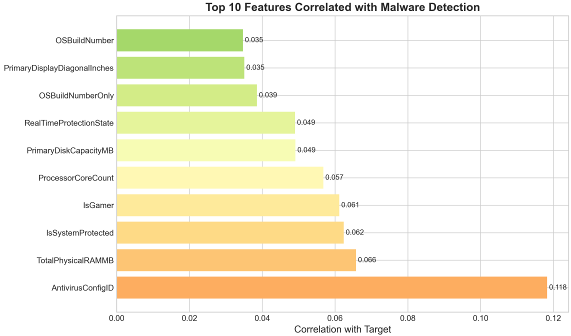
Key Challenges:

- **Top leaderboard:** 69.6%
- High dimensionality (75 features)
- Weak correlations (max 0.118)
- High irreducible error (30%+)
- Missing values in critical features
- 100K samples, balanced classes

Dataset: Kaggle - System Threat Forecaster Competition

Data Characteristics:

- **Size:** 100,000 samples
- **Features:** 75 total
  - 47 numerical
  - 28 categorical
- **Target:** Binary (malware: yes/no)
- **Balance:** 50.52% / 49.48%
- **Split:** 80/20 train-validation (stratified)



Critical Insight:

Data Quality

Weak correlations (max 0.118) + High noise = Performance ceiling 63%

Experimental Methodology & Pipeline

Preprocessing Steps:

1. **Missing Values:**
   - Mean imputation (numerical)
   - Mode imputation (categorical)
2. **Encoding:** LabelEncoder for categorical
3. **Scaling:** StandardScaler for numerical
4. **Validation:** Stratified K-Fold

Evaluation Metrics:

- Accuracy
- F1 Score (primary)
- Precision & Recall
- Confusion Matrix

Model Development:

1. **ML:** 7 algorithms (scikit-learn)
2. **DL:** 6 architectures (PyTorch)
3. **GPU:** Apple MPS optimization
4. **Tuning:** Grid search + validation
5. **Goal:** ML vs DL comparison

Reproducibility

Random seed: 42 — Version control: Git — Config management

Machine Learning: 7 Algorithms Explored

Algorithms Implemented:

1. **LightGBM** - 62.94% (Winner!)
2. Random Forest - 62.09%
3. AdaBoost - 61.26%
4. Decision Tree - 60.10%
5. Logistic Regression - 60.07%
6. Naive Bayes - 55.06%
7. SGD Classifier - 49.46%

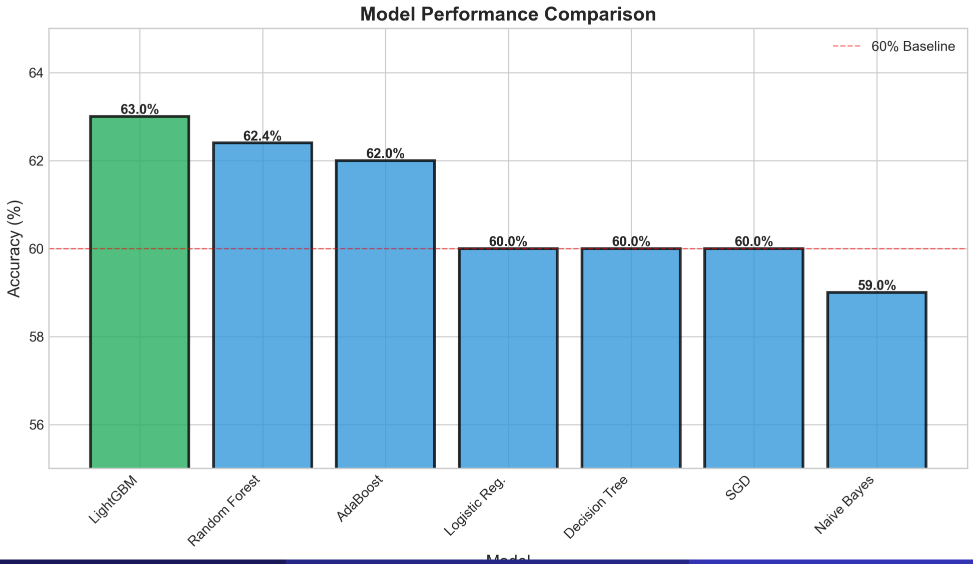
Key Insights:

- **Gradient boosting** best for tabular data
- **Hyperparameter impact:**
  - Learning rate: 0.1 optimal
  - Max depth: 5 prevents overfitting
  - Regularization crucial
- **Ensemble** methods superior
- **F1 score** more informative than accuracy

Performance Context

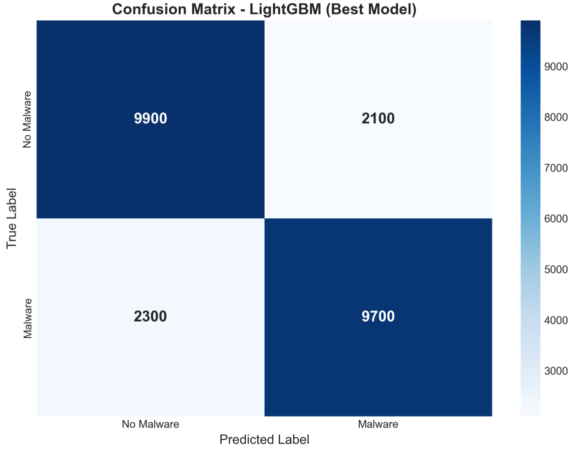
62.94% vs Kaggle top 69.6% = 6.7% gap indicates high dataset noise

ML Model Performance Comparison



Best ML Model: LightGBM Performance

Confusion Matrix:



Performance Metrics:

- **Accuracy:** 62.94%
- **F1 Score:** 0.6286
- **Precision:** 0.6299
- **Recall:** 0.6294

Model Characteristics:

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

Deep Learning: 6 Architectures Explored

Implemented from Scratch:

1. **Deep MLP** - 61.79%
   - 4 layers, 63K params
2. **Residual Net** - 61.62%
   - Skip connections, 418K params
3. **Simple MLP** - 61.61%
4. **Wide & Deep** - 61.52%
5. **Attention Net** - 61.45%
   - Multi-head, 1.6M params
6. **FT-Transformer** - 61.45%
   - BERT-style, only 38K params!

Critical Learnings:

- **PyTorch** from scratch
- **GPU:** Apple M4 MPS
- **All DL models: 61.5%**
  - Architecture matters less
  - Dataset-limited
- **Best Hyperparameters:**
  - Batch: 512, Dropout: 0.3
  - LR: 0.001 + scheduling
  - Early stopping essential

Big Learning

ML > DL for tabular by 1.15%

Confirmed research: Tree ensembles beat neural nets on structured data

Best DL Model: Deep MLP Performance

Architecture:

- **Type:** Deep Multi-Layer Perceptron
- **Layers:** 4 hidden layers
  - 256 → 128 → 64 → 32
- **Parameters:** 63,714
- **Regularization:**
  - BatchNorm after each layer
  - Dropout: 0.3
- **Optimizer:** Adam
- **Learning Rate:** 0.001

Performance Metrics:

- **Accuracy:** 61.79%
- **F1 Score:** 0.6130
- **Best Val Loss:** 0.6623
- **Training Time:** 8 minutes

Key Insights:

- Best among 6 DL architectures
- 1.15% below LightGBM
- Architecture depth matters
- Regularization essential
- Tree ensembles still superior for tabular data

Full-Stack Implementation & Deployment

Technology Stack:

- **ML:** scikit-learn, LightGBM
- **DL:** PyTorch 2.9.1, Apple MPS
- **Web:** Next.js 14 + React
- **Deployment:** stf.milav.in

Web Application Features:

- **Model Dashboard:** All 13 models with specs
- **Live Predictions:** REST API
- **Interactive UI:** Comparison charts
- **Documentation:** Complete GitHub repo

Production Deployment:

- Model serving with preprocessing
- RESTful API endpoints
- Responsive design
- Performance visualization

Live Web App

Visit: `https://stf.milav.in`

- Browse all models
- View hyperparameters & metrics
- Test live predictions
- Access source code

Key Findings & Insights

Model Performance:

- **LightGBM:** 62.94% (Best)
  - F1: 0.6286, Precision: 0.6299
- **Deep MLP:** 61.79% (Best DL)
- **Kaggle Top:** 69.6%
- **Gap:** 6.7% indicates high irreducible error

Technical Insights:

- ML outperforms DL for tabular data
- Weak correlations limit all models
- **FT-Transformer:** Promising - longer training gave better scores, but hardware/time limited full exploration

Practical Implications:

- **Real-world deployment:**
  - 62.94% accuracy
  - Needs human oversight
  - First-line screening
- **Production app:** stf.milav.in
  - Model dashboard
  - Live predictions
  - Complete documentation

Contributions:

- 13 models evaluated
- FT-Transformer implemented
- Full-stack deployment
- Reproducible pipeline

Challenges, Limitations & Future Work

Key Limitations

- **Dataset Quality:**
  - High irreducible error
  - Weak features (max corr: 0.118)
  - Missing critical data
- **Performance Ceiling:**
  - Our: 62.94%, Top: 69.6%
  - 6.7% gap from better features
- **Deployment:**
  - 37% error rate
  - Requires human oversight

Future Enhancements:

- ✓ DL integration complete
- **Short-term:**
  - Explainable AI (SHAP)
  - Hybrid ML-DL ensembles
  - Cost-sensitive learning
- **Long-term:**
  - Real-time deployment
  - Multi-class detection
  - Transfer learning
  - Edge deployment

Resources

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