

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

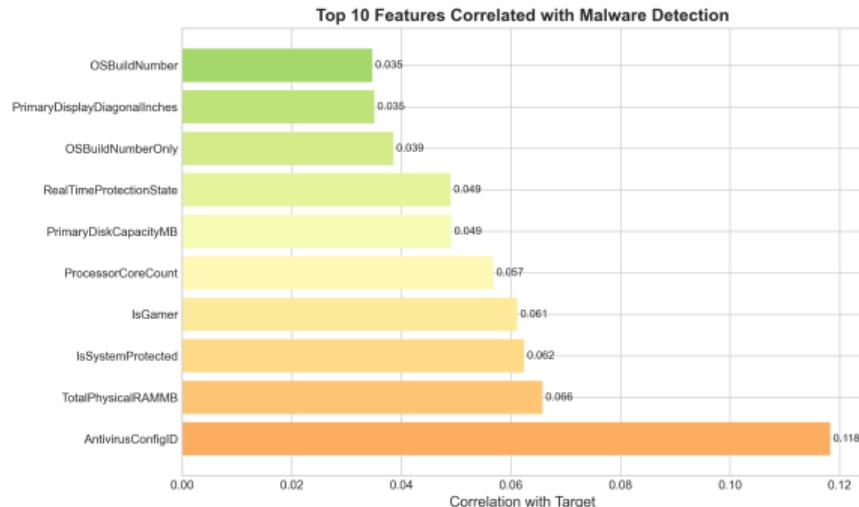
Key Challenges:

- **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%

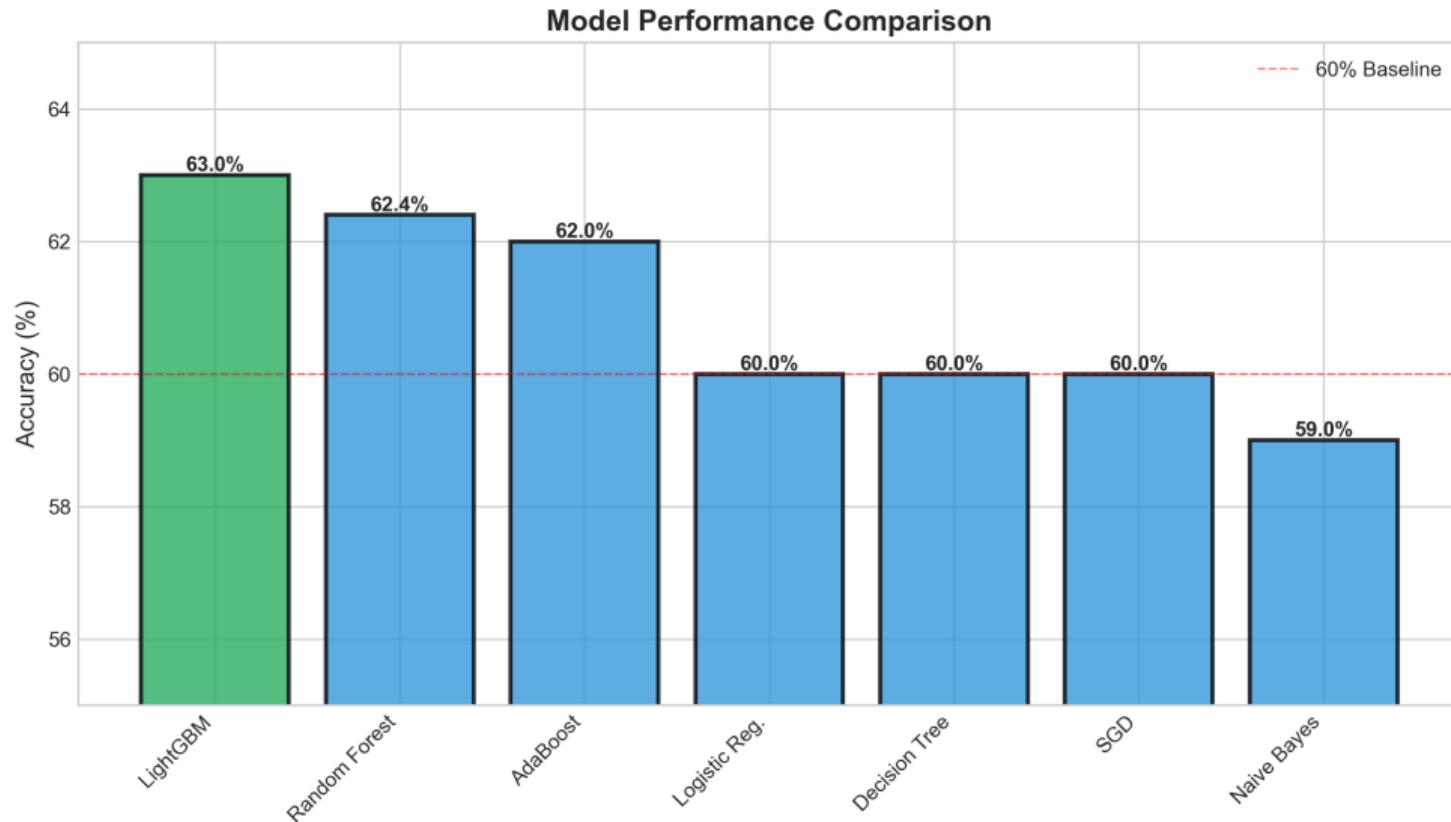
Key Insights:

- **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

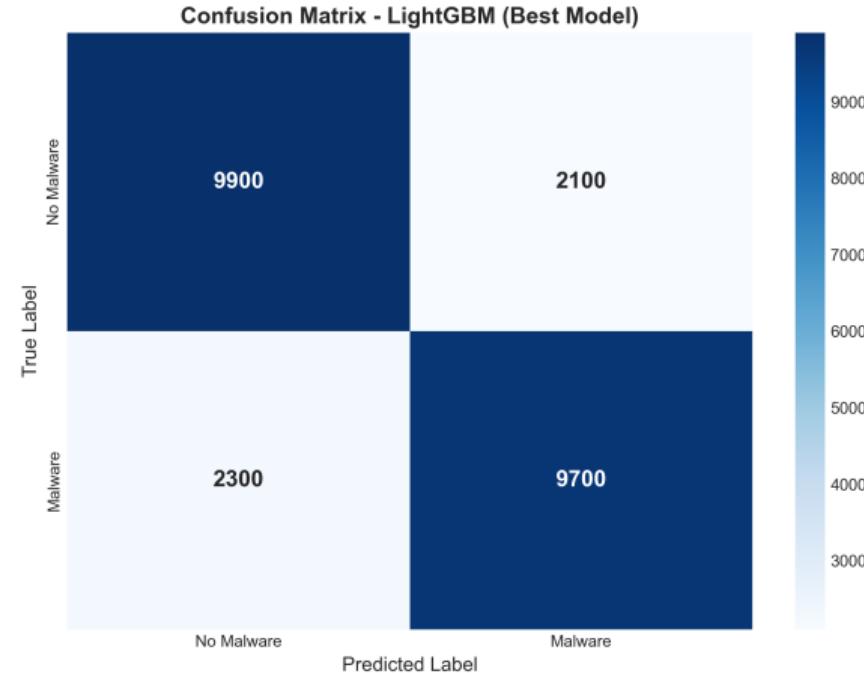
62.94% vs Kaggle top 69.6% = 6.7% gap
indicates high dataset noise

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
 - $256 \rightarrow 128 \rightarrow 64 \rightarrow 32$
- **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

Production Deployment:

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

Web Application Features:

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

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

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