

Systems Analysis & Design – Workshop 4

Data-Driven and Event-Based Simulation for Fraud Detection Systems

Leonardo Rodríguez Salas- 20231020150
Santiago Marín - 20231020159
Davidson Sanchez Gordillo- 20231020183
Luis Mario Ramirez- 20231020166

November 2025

Contents

1	Introduction	2
2	Dataset Preparation	2
2.1	Preprocessing Summary	2
3	Simulation Planning	2
3.1	Scenario 1: Data-Driven ML Simulation	2
3.2	Scenario 2: Event-Based Cellular Automata	3
4	Base Simulation Implementation	3
4.1	Scenario 1: ML Pipeline	3
4.2	Scenario 2: Cellular Automata	3
5	Simulation Execution	4
5.1	ML Sensitivity Analysis	4
5.2	Cellular Automata Evolution	9
6	Results and Discussion	15
6.1	ML Simulation	15
6.2	Event-Based Automata	16
7	System Workflow Validation	17
8	Complexity and Chaos Exploration	18
9	Conclusions and Future Work	18

1 Introduction

This workshop explores system validation through two complementary simulations:

- **Scenario 1: Data-driven simulation** using a classical Machine Learning pipeline for fraud detection.
- **Scenario 2: Event-driven simulation** using Cellular Automata to model spatial and emergent fraud patterns.

Both simulations were aligned with the proposed system architecture from Workshop #2 to analyze:

- Learning dynamics
- Sensitivity to perturbations
- Emergent behavior
- Effects of chaotic parameters

2 Dataset Preparation

The dataset used corresponds to the Kaggle fraud detection competition.

2.1 Preprocessing Summary

- Total samples used: **[200,000-590540]**
- Fraud rate: **[3.50 - 3.55%]**
- Numeric features selected: **401**
- Missing values: replaced with median values.
- Scaling: StandardScaler normalization.

3 Simulation Planning

3.1 Scenario 1: Data-Driven ML Simulation

A Random Forest classifier was selected in order to simulate a learning agent capable of detecting fraudulent events.

Key metrics

- Accuracy
- ROC-AUC
- Precision
- Recall

3.2 Scenario 2: Event-Based Cellular Automata

A spatial cellular automaton was implemented where:

- Each cell represents a transaction event state:
 - Normal
 - Suspicious
 - Fraudulent
 - Flagged
- Local neighborhood interactions model fraud propagation.
- Random mutations simulate unknown or chaotic fraud emergence.

Key metrics

- Fraud density
- System entropy
- Cluster size and cluster count

4 Base Simulation Implementation

4.1 Scenario 1: ML Pipeline

Model:

- RandomForestClassifier
- n_estimators = 100-400
- max_depth = 15-25
- class_weight=balanced
- min_samples_split = 20

Chaos Perturbation Analysis

Noise was injected into feature space at the following levels:

[0.0, 0.01, 0.05, 0.10, 0.15, 0.20]

This perturbation simulated chaotic sensitivity to initial conditions.

4.2 Scenario 2: Cellular Automata

Spatial Parameters

- Grid size: 100×100
- Neighborhood radius: $[1 - 3]$
- Mutation rate: $[0.003 - 0.007]$
- Simulation duration: 70 - 100 generations

Transition Rules

- Normal \rightarrow Suspicious: $\geq [2+]$ fraudulent neighbors
- Suspicious \rightarrow Fraudulent: $\geq [3+]$ fraudulent neighbors
- Fraudulent \rightarrow Flagged: $\geq [5+]$ normal neighbors
- Chaos injection: random mutation at rate $[0.003]$

5 Simulation Execution

5.1 ML Sensitivity Analysis

- Sample Size:200000, max_depth:15, n_estimators:100

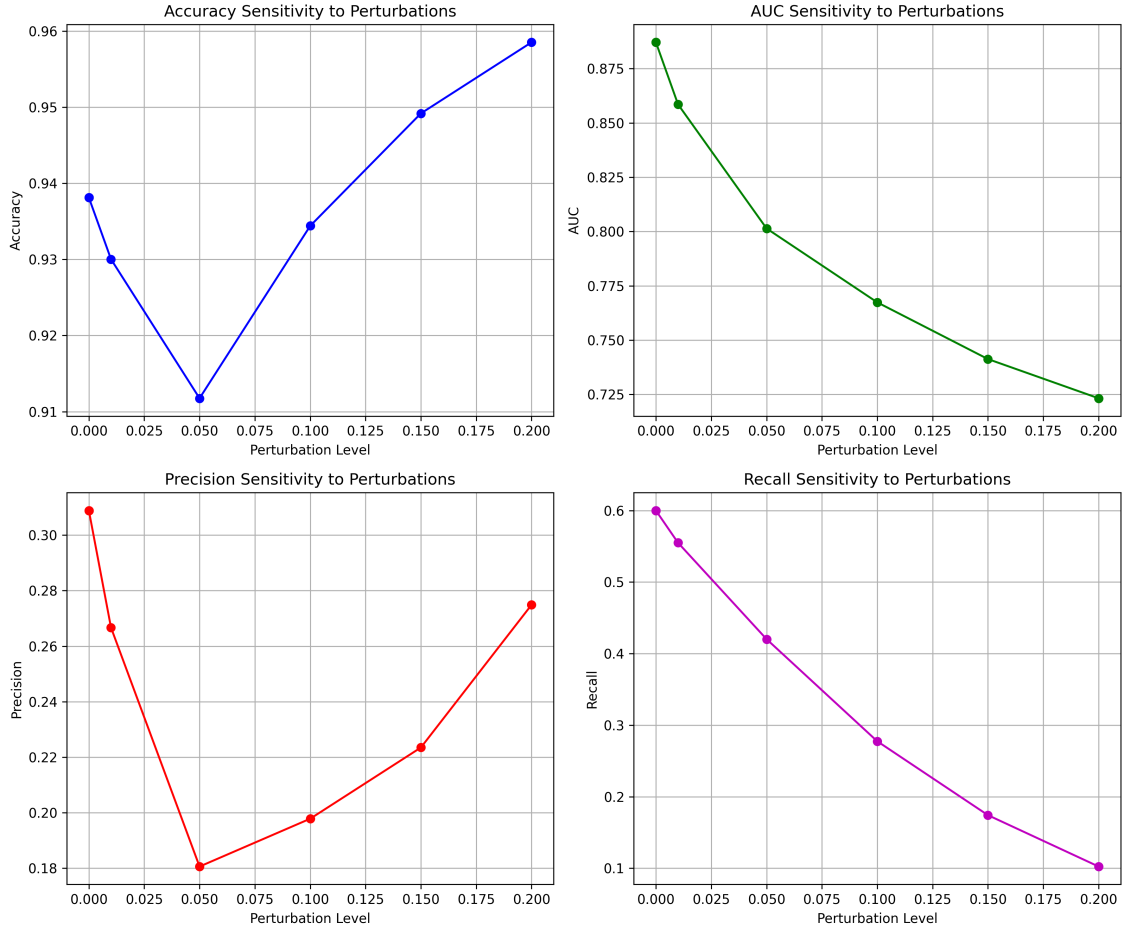


Figure 1: ML sensitivity to chaotic perturbations

Sample Size:200000, max_depth:15, n_estimators:100

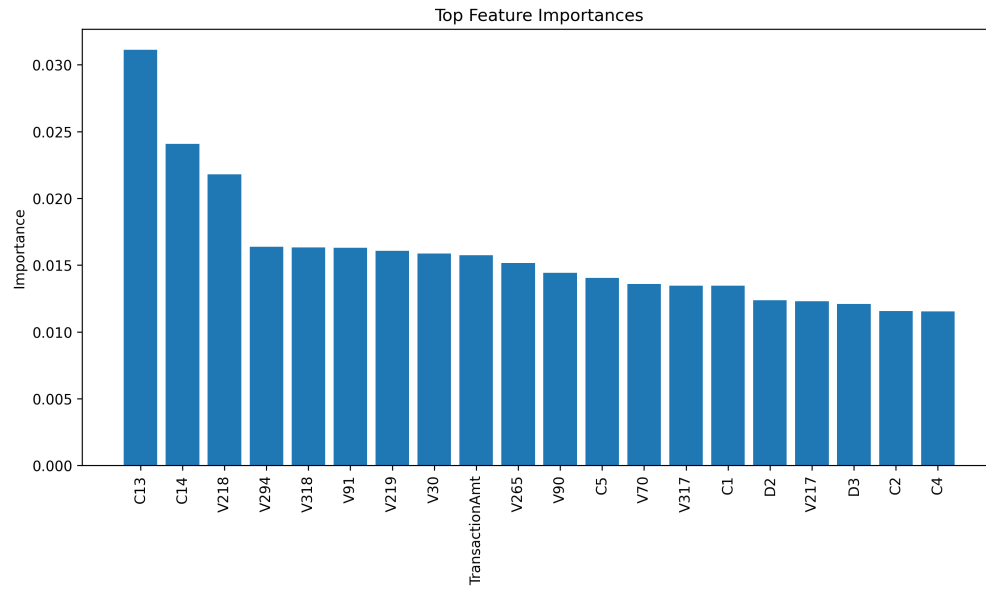


Figure 2: Top 20 Most Influential Features

Model: Random Forest (100 estimators)

- Sample Size:400000, max_depth:20, n_estimators:100

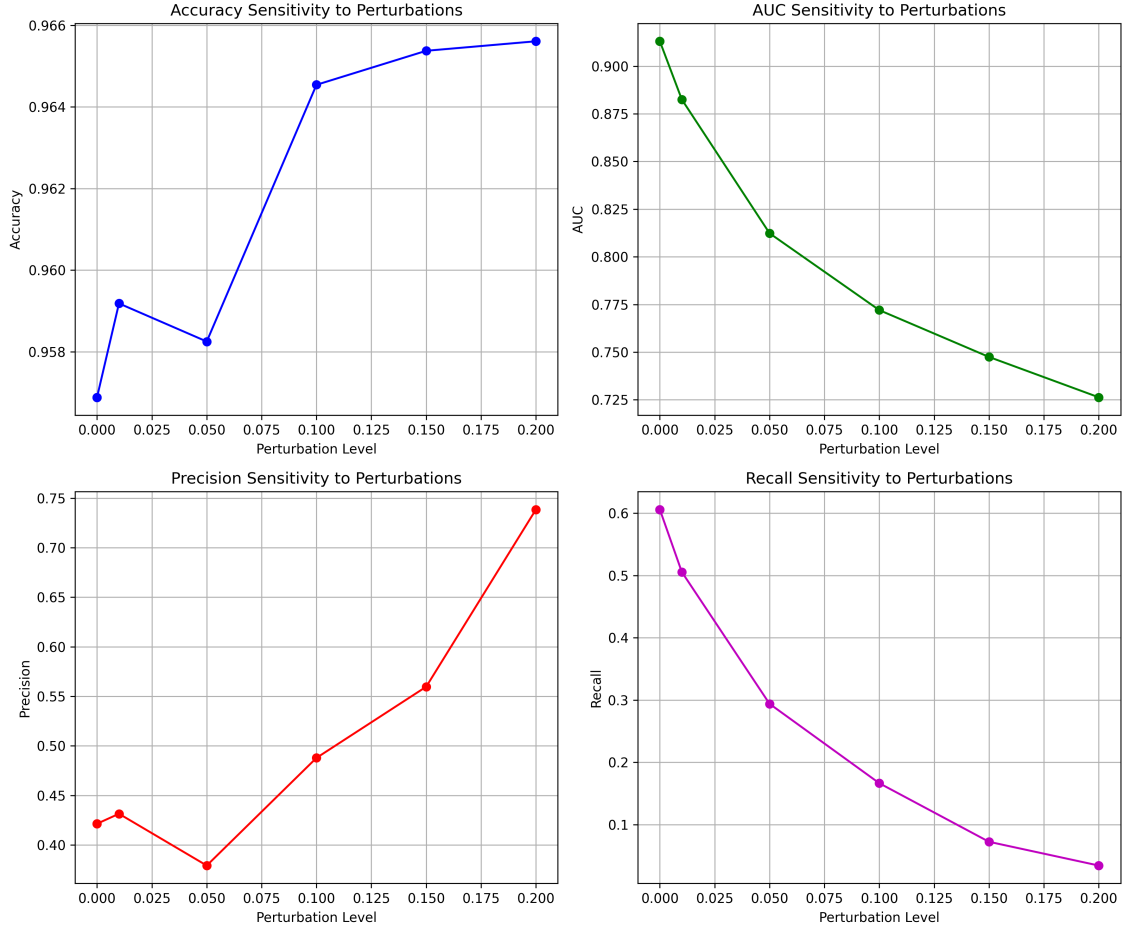


Figure 3: ML sensitivity to chaotic perturbations

Sample Size:400000, max_depth:20, n_estimators:100

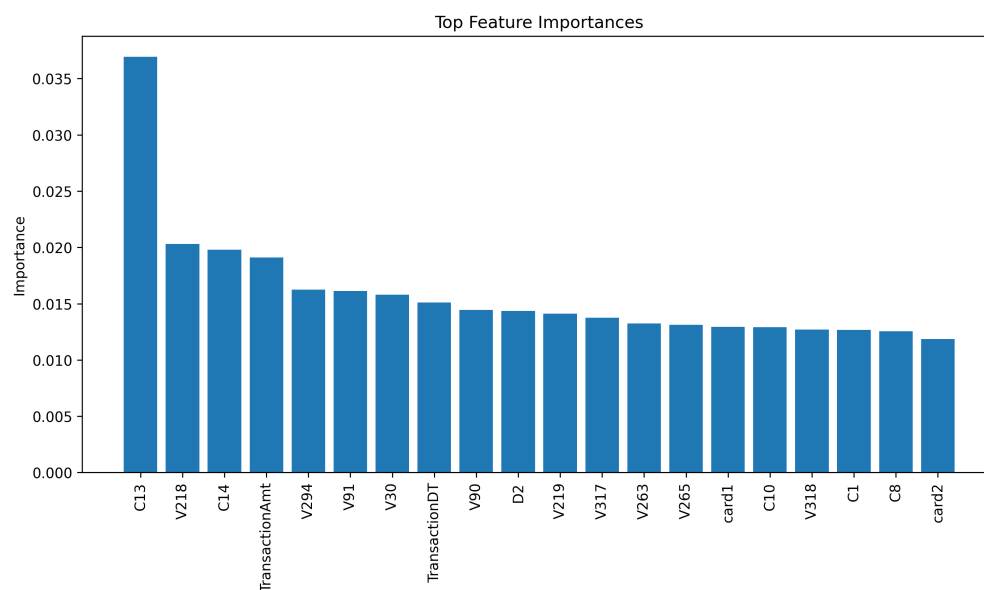


Figure 4: Top 20 Most Influential Features

Model: Random Forest (100 estimators)

- Sample Size:590540, max_depth:25, n_estimators:400

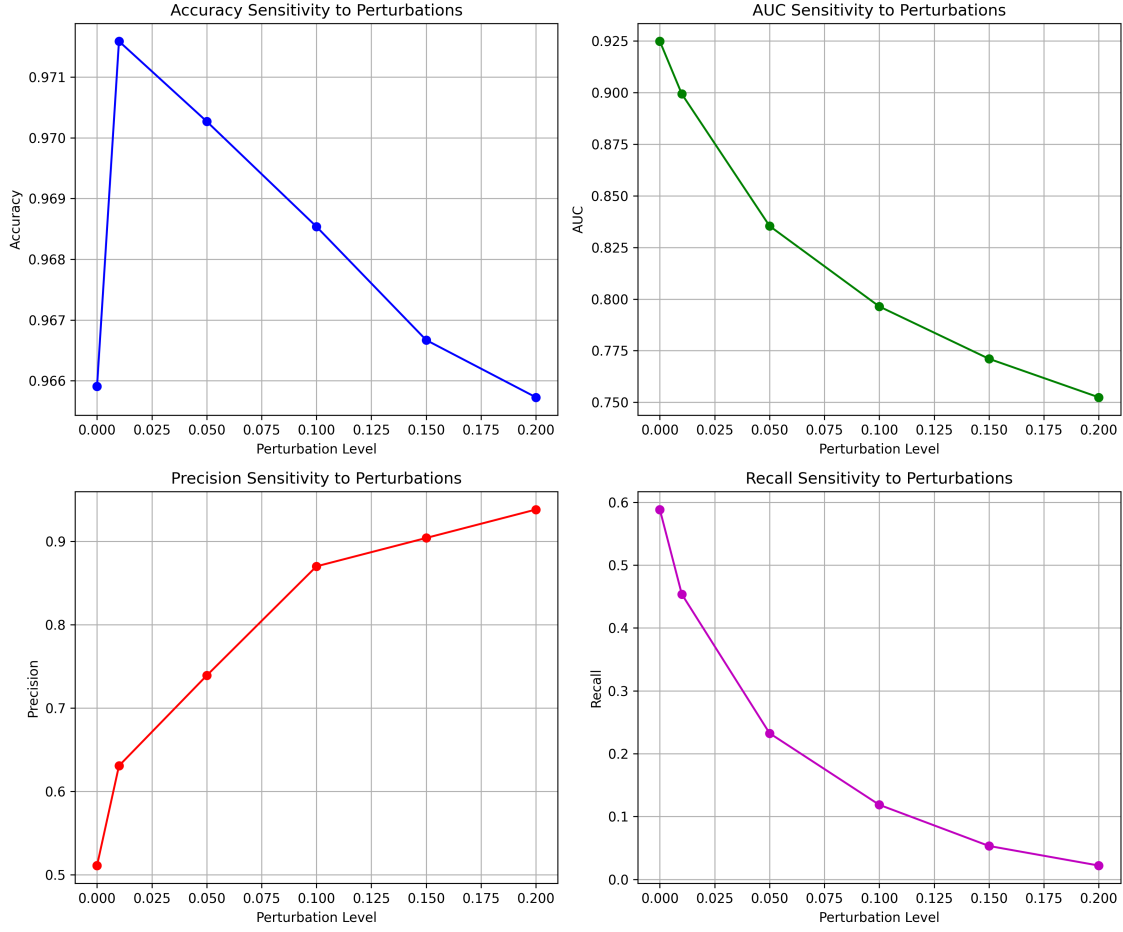


Figure 5: ML sensitivity to chaotic perturbations

Sample Size:590540, max_depth:25, n_estimators:400

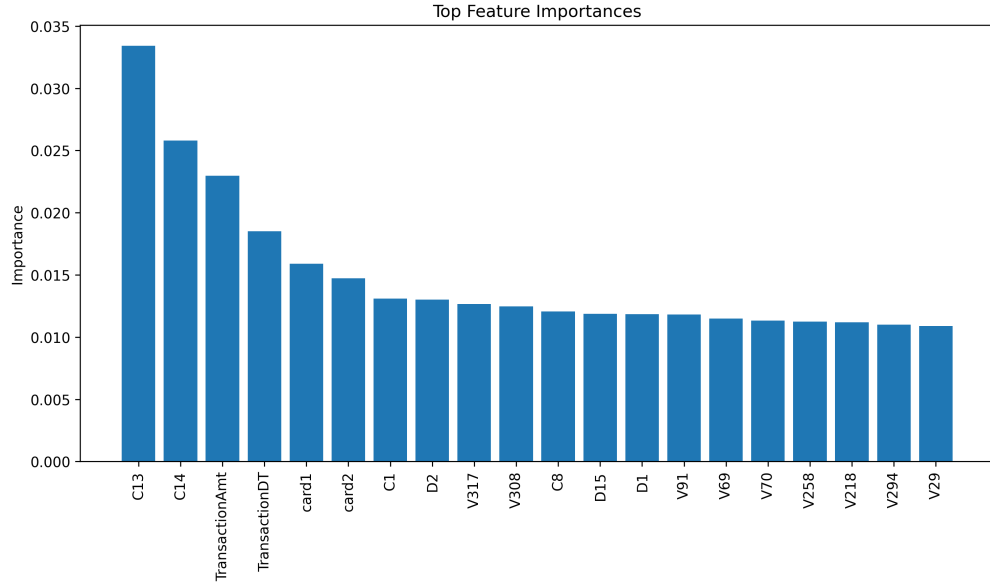


Figure 6: Top 20 Most Influential Features

Model: Random Forest (400 estimators)

5.2 Cellular Automata Evolution

- 70 generations, mutation rate = $[0.003]$, radius = $[1]$

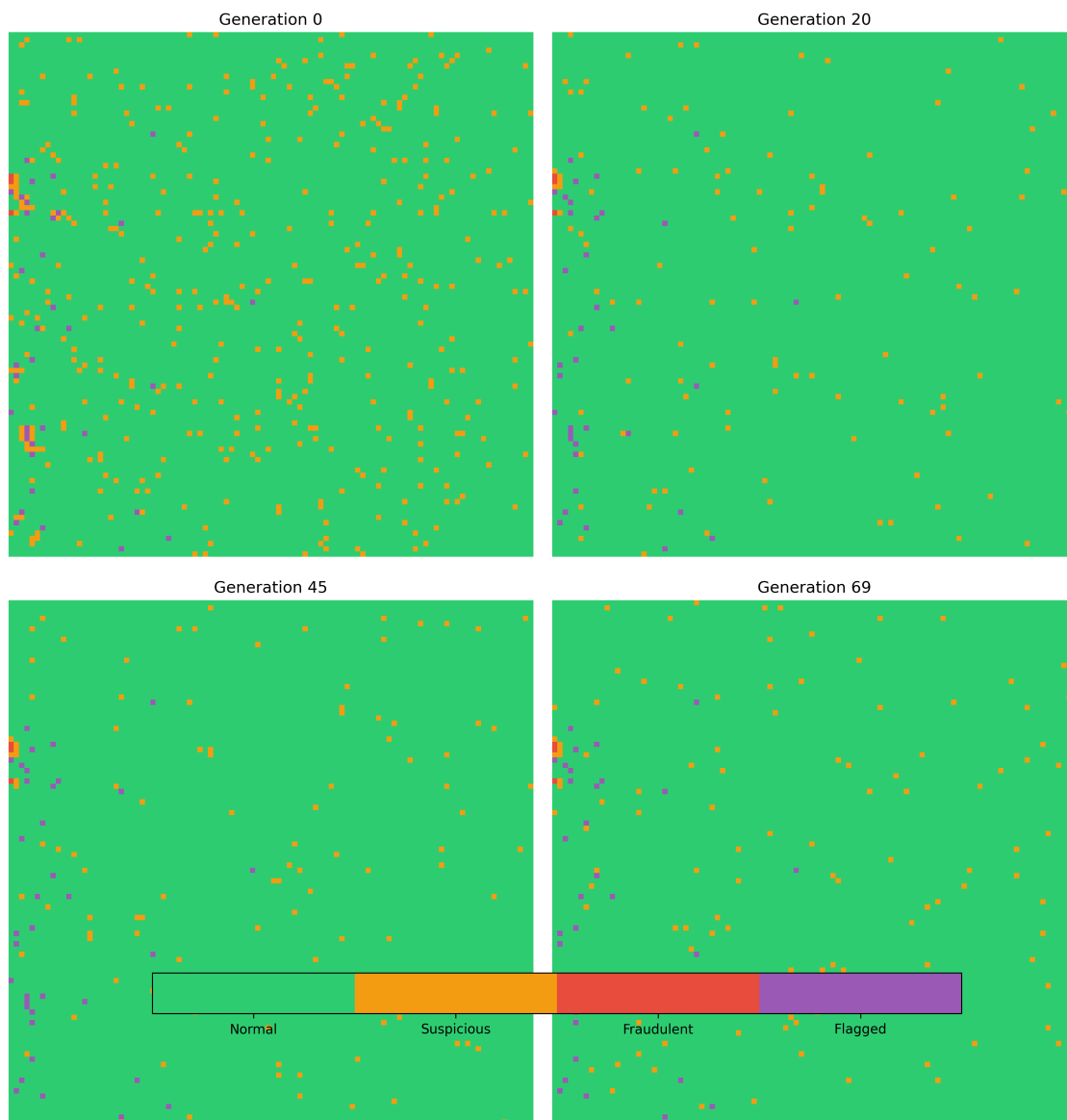


Figure 7: Spatial evolution of Cellular Automata

Grid 100x100, 70 generations, mutation rate = [0.003], radius = [1]

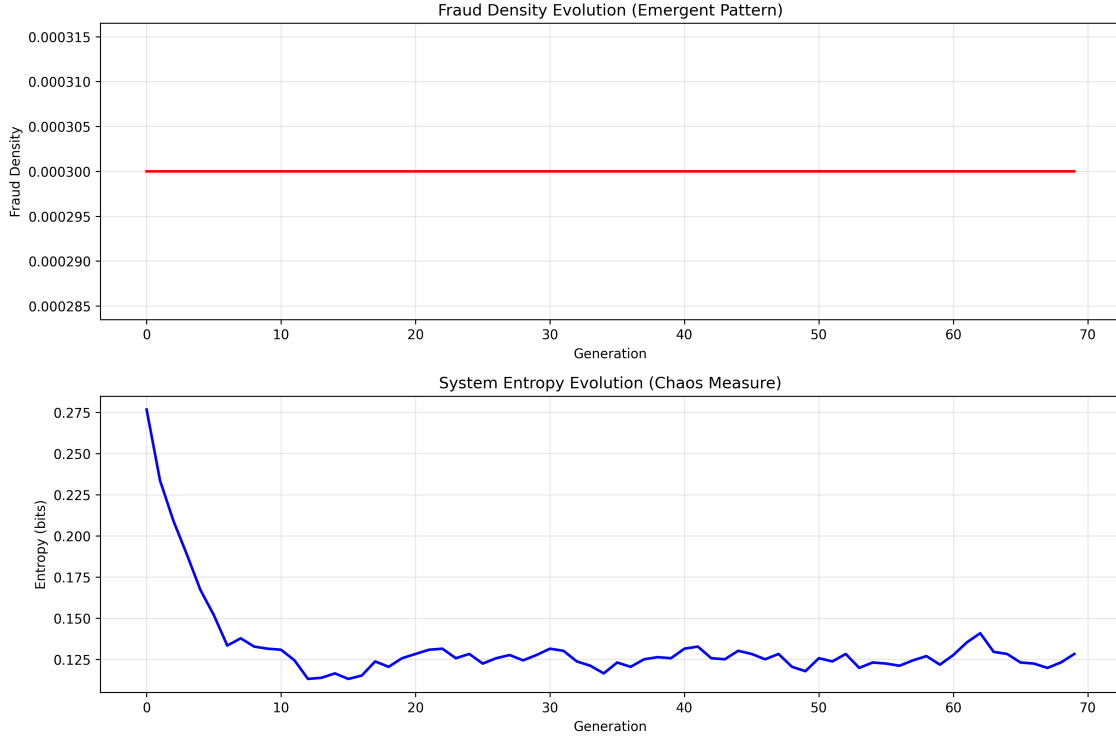


Figure 8: Fraud Density and Entropy Over Time

Entropy trends observed during 70-generation simulation

- 70 generations, mutation rate = $[0.007]$, radius = $[2]$

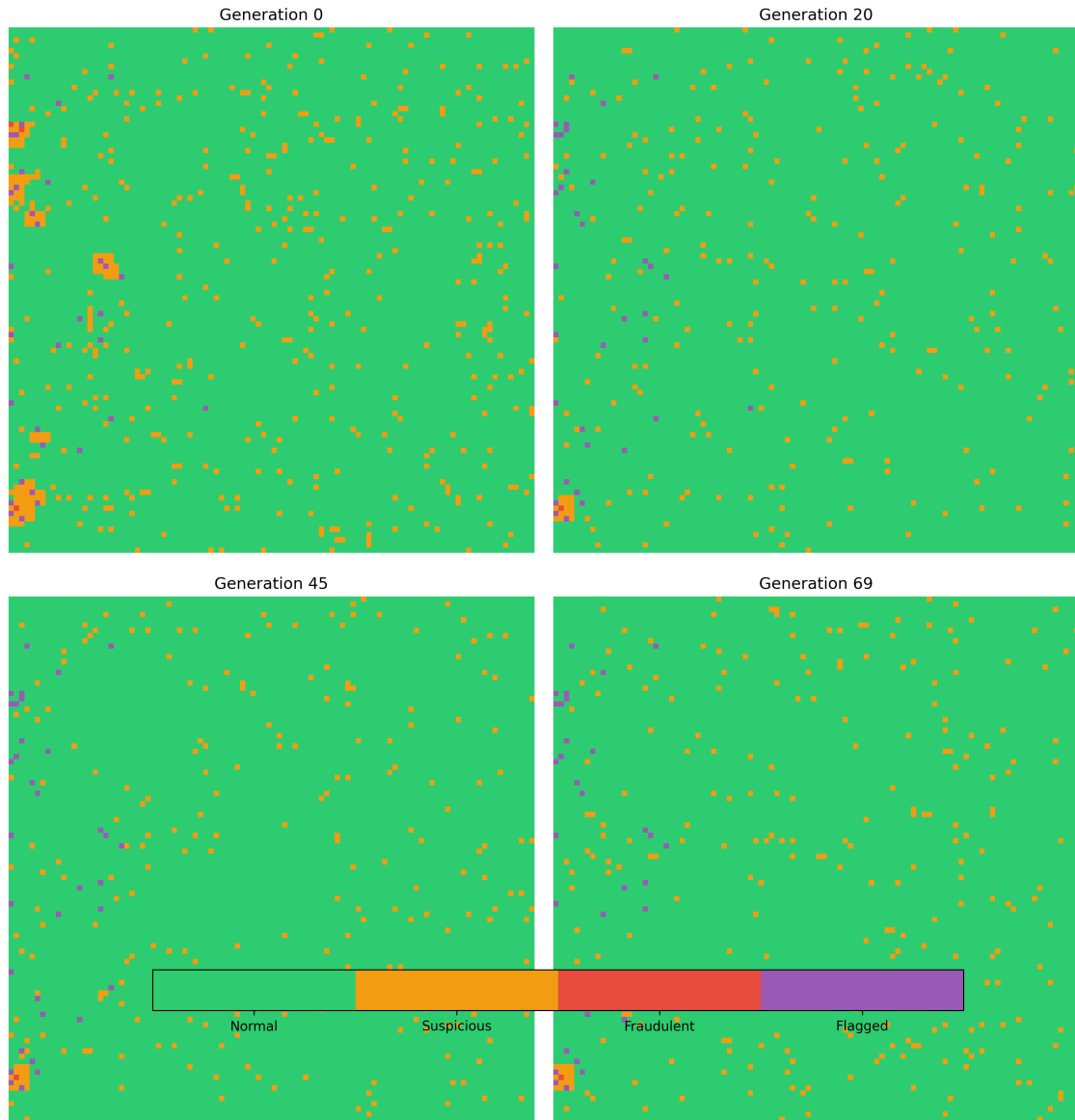


Figure 9: Spatial evolution of Cellular Automata

Grid 100x100, 70 generations, mutation rate = [0.007], radius = [2]

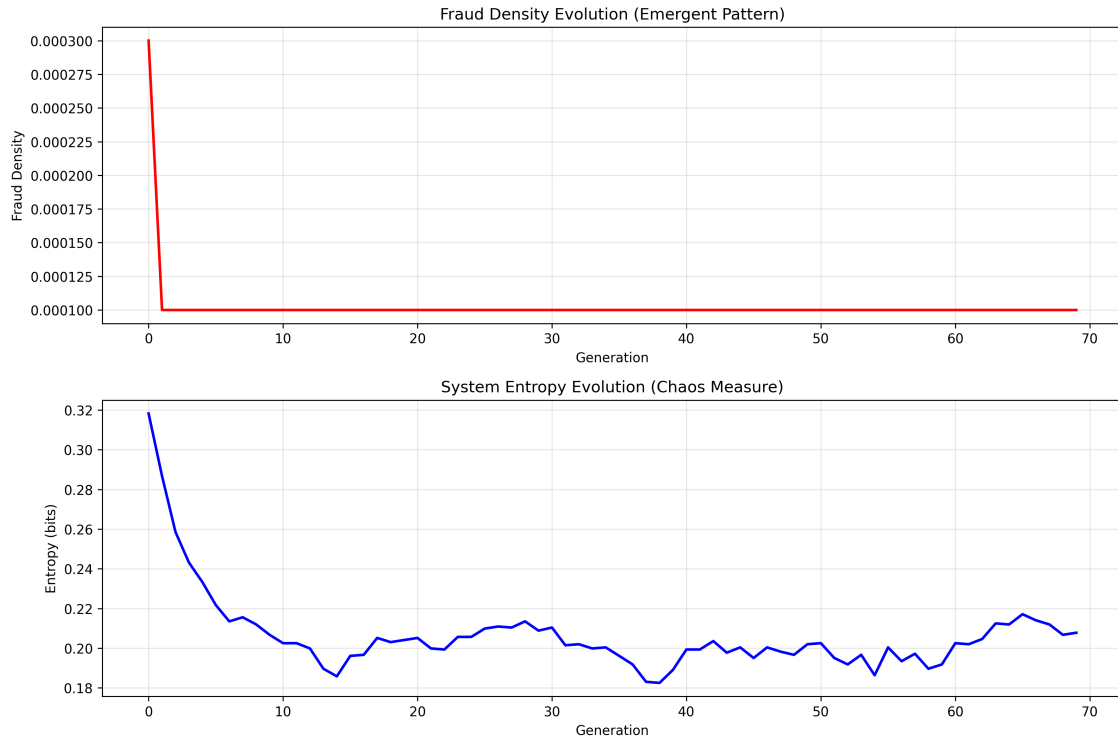


Figure 10: Fraud Density and Entropy Over Time

Entropy trends observed during 70-generation simulation

- 100 generations, mutation rate = $[0.007]$, radius = $[3]$

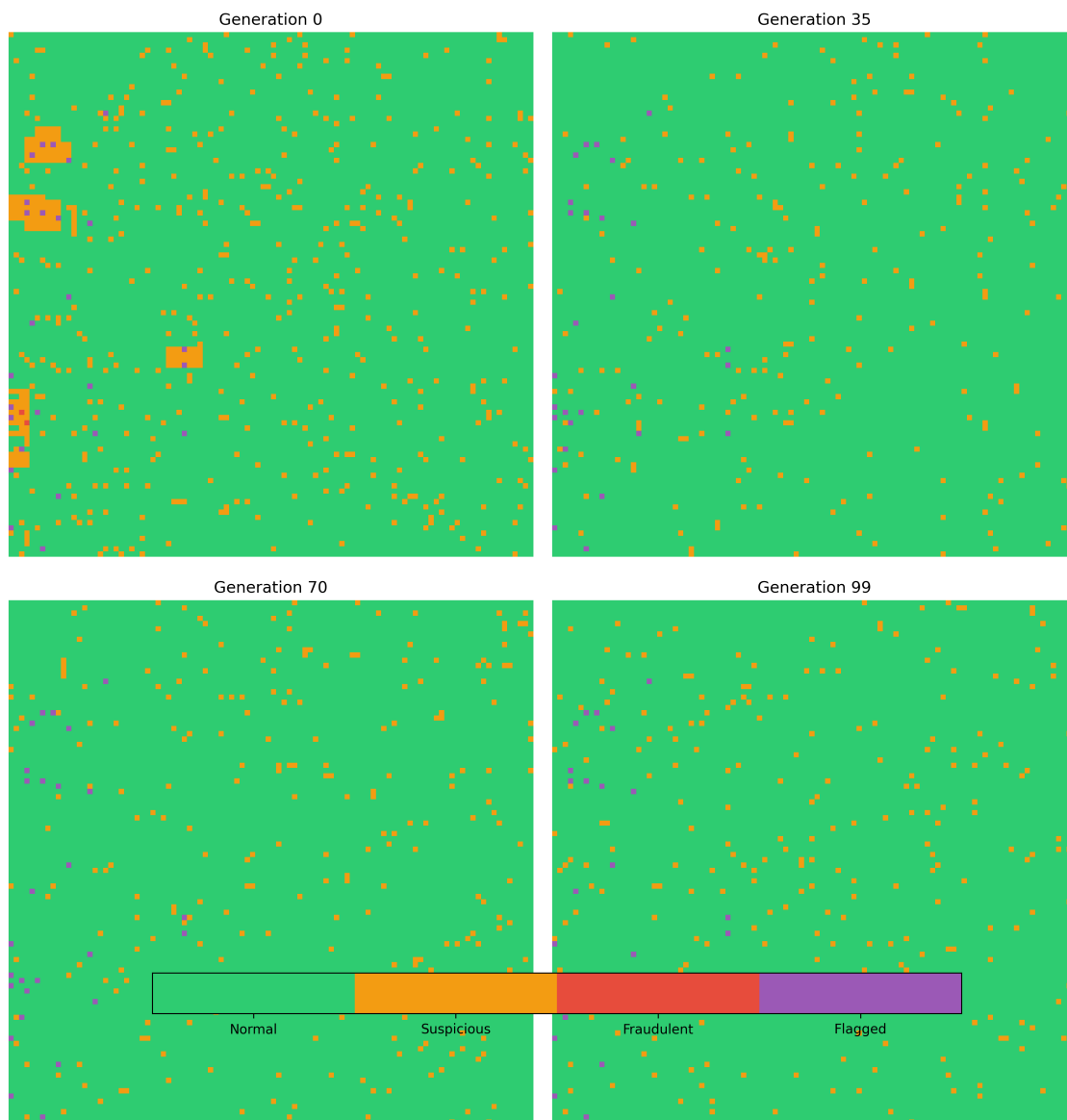


Figure 11: Spatial evolution of Cellular Automata

Grid 100x100, 100 generations, mutation rate = [0.007], radius = [3]

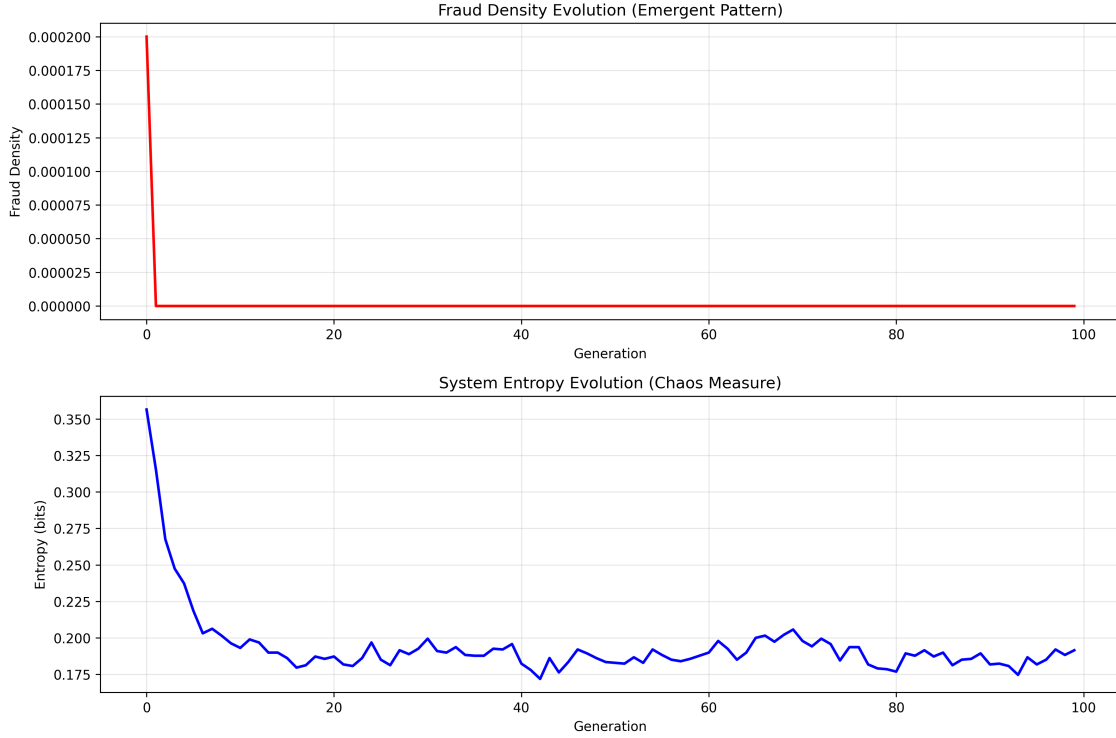


Figure 12: Fraud Density and Entropy Over Time

Entropy trends observed during 100-generation simulation

6 Results and Discussion

6.1 ML Simulation

First Baseline results with 200k transactions:

- Accuracy: **0.938**
- AUC: **0.887**
- Recall (fraud): **0.60**
- Precision (fraud): **0.31**

Observations:

- The system shows clear sensitivity to perturbations above noise = 0.10.
- Recall drops rapidly under feature noise, validating chaotic sensitivity concepts.
- Tradeoff observed:

High recall \Rightarrow Lower precision

Second Baseline results with 400k transactions:

- Accuracy: **0.9569**

- AUC: **0.9130**
- Recall (fraud): **0.61**
- Precision (fraud): **0.42**

Observations:

- The system improves overall performance, increasing AUC, accuracy, and notably fraud precision.
- Despite the stronger baseline, the model still degrades significantly when perturbations exceed $\text{noise} = 0.10$.
- The prior tradeoff is reduced: recall remains stable while precision improves, indicating a more balanced detection profile.

Third Baseline results with 600k transactions:

- Accuracy: **0.9659**
- AUC: **0.9248**
- Recall (fraud): **0.59**
- Precision (fraud): **0.51**

Observations:

- The model reaches its strongest baseline so far, improving accuracy, AUC, and fraud precision compared to previous iterations.
- Despite the improved baseline, the system continues showing instability for perturbations > 0.10 , consistent with chaotic sensitivity behavior.
- Precision–recall balance shows significant enhancement: precision surpasses 0.50 while maintaining recall near previous levels, indicating better fraud discrimination with fewer false positives.

6.2 Event-Based Automata

First Simulation:

- Fraud density: 0.0003
- System entropy: 0.1282
- Detected clusters: 2

Observed patterns:

- The system quickly stabilizes: entropy falls in the first generations and then remains almost constant, indicating that the automaton reaches a fixed state.

- Fraud detection is weak: the system identifies very few real fraudulent cells and generates many false positives, which is reflected in very low accuracy and recall.
- The fraud density is so low that no significant clusters are formed, limiting the automaton's ability to detect clear spatial patterns

Second Simulation:

- Fraud density: 0.0001
- System entropy: 0.2078
- Detected clusters: 1

Observed patterns:

- Detection performance decreased: both precision and recall dropped significantly, with fewer true fraud cells detected than in the previous run.
- The system evolved into a higher-entropy state, showing more chaotic behavior and less stable pattern formation than the earlier experiment.
- Spatial clustering weakened: only one isolated fraud cell was detected, whereas the previous run produced two clusters, reducing the automaton's ability to amplify weak fraud signals.

Third Simulation:

- Fraud density: 0.0000
- System entropy: 0.1916
- Detected clusters: 0

Observed patterns:

- Fraud disappears entirely early in the simulation, indicating stronger suppression/decay behavior in the automaton rules.
- Entropy increased, so the final configuration is more disordered even though fraudulent cells were eliminated. This suggests ongoing fluctuations between suspicious and normal states
- No spatial consolidation of fraud occurs in the current run; fraud collapses before clusters can form.

7 System Workflow Validation

Integration of both simulations validated the conceptual system architecture:

- **ML Pipeline:** Detects fraud at transaction-level.
- **Automata Simulation:** Models spatial and group dynamics of fraudulent activities.

Data flow was observed as:

$$Data \rightarrow ML \rightarrow EventPatterns \rightarrow SystemMonitoring$$

8 Complexity and Chaos Exploration

Both simulations demonstrate:

- Sensitivity to initial conditions.
- Non-linear dynamics.
- Emergent spatial patterns.
- Parameter-controlled chaos escalation.

Key chaotic parameter groups:

- ML perturbation noise σ
- Automata mutation rate m
- Neighborhood radius r

9 Conclusions and Future Work

This workshop successfully demonstrated how:

- The Kaggle IEEE-CIS fraud detection algorithm relies on classical ML models like LightGBM. These models iteratively adjust parameters to minimize classification error on transaction data. The learning process simulates adaptive dynamics, refining weights through gradient boosting. Each iteration mirrors trial-and-error, improving fraud detection accuracy over time. Patterns in anonymized features are extracted, reinforcing predictive power with feedback loops. This simulation captures essential adaptive mechanisms of fraud recognition in financial systems. Thus, classical ML becomes a computational lens for studying fraud learning dynamics.
- Fraudulent transactions often display emergent complexity across anonymized categorical features. Cellular automata analogies help explain how local transaction rules scale into global fraud patterns. Interactions between device IDs, card usage, and time variables generate intricate structures. These emergent behaviors resemble diffusion or clustering seen in automata simulations. Complexity arises not from single variables but from their collective interactions. The Kaggle algorithm leverages feature engineering to capture these emergent signals. In essence, fraud detection reveals hidden order within seemingly random transaction streams.
- Fraud detection models are highly sensitive to chaotic variables like time lags or device entropy. Small perturbations in anonymized features can cascade into large shifts in prediction outcomes. This sensitivity destabilizes performance, amplifying noise in certain feature distributions. Metrics such as AUC fluctuate when chaotic variables dominate the training set. Optimization strategies must account for unpredictability in these dynamic inputs. Yet, chaos introduces diversity, forcing models to generalize beyond rigid patterns. Ultimately, managing chaos is key to balancing stability with fraud detection innovation.