Workshop No. 4 — Kaggle System Simulation

Team #9

Juan Esteban Ávila Trujillo – 20251020054
Juan José León Gómez – 20212020055
Juan Pablo Díaz Ricaurte – 20222020076

Universidad Distrital Francisco José de Caldas
Systems Engineering Program

November 29, 2025

Introduction

This report presents the simulations implemented to evaluate the analysis and design of the system proposed in previous workshops, particularly the design of Workshop 2. The purpose of this Workshop 4 is to transform that design into functional components that allow observation of the system's behavior. To this end, two complementary approaches were used:

Data-driven simulation, using a Feigl Target Boosting Classifier, to identify and estimate the risk of default.

Event-driven simulation (EC) using a cellular automaton, which models how risk can be distributed, attenuated, or concentrated through local interactions between observations.

The motivation for employing both approaches is that the risk of default arises not only from statistical patterns in the characteristics but also from dynamic interactions between the elements of the financial system. The machine learning model offers predictive capabilities, while the cellular automaton allows the study of propagation processes, stability, sensitivity, and potential chaotic behavior.

Contents

1 Data preparation

The original data provided by Kaggle's competition dataset (`train_v2.csv`) contains more than 100,000 observations and 771 features, all previously anonymized and standardized. However, working with the entire dataset required very high computational resources, making simulation impractical. Therefore, controlled random sampling was applied using a fixed seed:

```
df_small = df.sample(n=5000, random_state=42)
```

The purpose of this reduction is not to decrease analytical quality, but to make experimentation feasible, representative, and efficient. A subset of 5000 observations preserves the overall diversity of the dataset, allows running heavy simulations (such as Cellular Automata and boosting models) in reasonable computational time, and ensures reproducibility. The target variable `loss` is normalized between 0 and 100, where 0 indicates no financial loss.

Summary of the final dataset (`df_small`) after preprocessing:

- **Rows:** 5000
- **Columns:** 771
- **Types:** 653 float64, 118 int64
- **Missing values:** 0 (fully imputed)
- **Exported file:** `Data_clean.csv`

It should be emphasized that this size reduction is used exclusively for simulation, educational, and instructional purposes. Competitive solutions for the real Kaggle competition require larger subsets or the full dataset.

Data Preparation and Cleaning

All processing was implemented in Python using Pandas, NumPy, and scikit-learn, ensuring reproducibility through explicit control of random seeds.

Steps Applied

1. Sampling

5000 observations were randomly selected:

```
df_small = df.sample(n=5000, random_state=42)
```

2. Missing Value Analysis

Missing counts (`missing_count`) and percentages (`missing_pct`) were computed.

- Less than 18% of values were missing in some columns.
- 492 columns contained at least one missing value.

3. Numerical Imputation

For all numeric columns (`float64` and `int64`), the median of each column was imputed.
Justification: Median imputation is robust to extreme values and works well with tree-based models.

4. Encoding Categorical Variables

For the 19 `object`-type columns:

- Missing values were replaced by the token `"MISSING"`.
- Columns were encoded using `LabelEncoder`, producing integer-coded categories.

5. Preprocessing Validation

It was verified that no missing values remained:

```
df_small.isnull().sum().sum() == 0
```

The resulting dataset was exported as:

`Data_clean.csv`

General Justification

The combination of median imputation and label encoding is suitable for rapid experimentation, tree-based models, and simulation workflows. More sophisticated methods may be employed in production or competitive environments, but for simulation purposes, this preprocessing pipeline offers an optimal balance between reliability and computational efficiency.

2 Simulation planning

Data-driven simulation: Gradient Boosting Classifier

This simulation follows Josef Feigl's methodology for the Loan Default Prediction competition. Instead of directly predicting the ongoing loss value, a binary objective was defined:

$$\text{target} = \begin{cases} 1 & \text{if loss} > 0, \\ 0 & \text{if loss} = 0. \end{cases}$$

Using the competition dataset train.csv, consisting of 105,471 instances and 751 numerical data features, the proportion of positive (predetermined) cases was found to be 0.9837, confirming the highly unbalanced nature of the problem.

Cellular Automaton: Role in Simulation Planning

The Cellular Automaton (CA) is used in the simulation as a dynamic model that represents how loss values can propagate, stabilize, or dissipate within a system composed of many loan observations. Its role in the planning of the simulation is to capture the temporal and spatial behavior of financial risk—something that traditional predictive models cannot express, since they do not consider interactions between observations.

The design of the CA follows several principles:

1. Spatial Representation of the System

Each observation in the dataset is assigned to a cell in a two-dimensional grid. The 5000 cleaned observations are reshaped into a 50×100 matrix. Each cell starts with a value corresponding to its normalized loss value.

2. Temporal Evolution Through Iterations

The simulation runs for 100 steps. At each step, all cells update their state simultaneously, allowing us to study how risk evolves over time.

3. Local Interaction via Neighborhoods

Each cell is influenced by its eight immediate neighbors (Moore neighborhood). This mechanism models contagion or diffusion effects: if a group of loans shows high losses, they can influence nearby loans in the grid.

4. State Update Rule

During each iteration, a cell updates its value based on four components:

- **Persistence**: the natural tendency of the loan to maintain its risk level.
- **Neighbor Influence**: if the neighborhood mean exceeds a threshold and a Bernoulli trial succeeds, the cell adopts part of the surrounding risk.
- **Noise**: a random disturbance simulating external, unpredictable influences.
- **Decay**: if no neighbor influence occurs, the cell's value slightly decreases, modeling stabilization in the absence of external pressure.

5. Range Restriction

After each update, all cell values are clipped between 0 and 100, consistent with the scaling of the `loss` variable.

6. Historical Storage

Every full grid state is stored in memory to:

- visualize the evolution of risk across iterations,
- analyze dynamic propagation patterns,
- compute evaluation metrics such as MAE by comparing the final grid with the real loss values.

Overall, the Cellular Automaton models financial risk as a dynamic system. Risk can propagate, dissipate, or stabilize depending on the selected parameters. This provides an interpretable, system-level perspective that complements the purely statistical behavior of data-driven predictive models.

3 Implementation of the Simulation

The implementation of the simulation system was divided into two main components: (1) a **data–driven model**, implemented using a Feigl Target Boosting Classifier, and (2) an **event–based simulation**, implemented using a Cellular Automaton (CA). Both components operate on the cleaned and standardized dataset of 5000 observations.

Implementation of the Data–Driven Model

This model is responsible for learning static patterns of financial risk from the input features. The procedure followed was:

1. **Separation of variables**: predictors (X) and target variable (`loss`).
2. **Training** of a Feigl Target Boosting model with controlled hyperparameters to avoid overfitting.
3. **Evaluation** using MAE and classification metrics.
4. **Exporting** the cleaned dataset so it could be used by teammates implementing complementary models.

The goal of this component is not to produce a competitive Kaggle model, but to serve as a stable reference for comparing static predictions with the dynamic behavior produced by the Cellular Automaton.

Implementation of the Cellular Automaton

The CA constitutes the event–based simulation engine. Each observation in the dataset is mapped to a cell in a two–dimensional grid of size $\mathbf{50 \times 100}$. The state of each cell corresponds to the normalized loss value in the interval $[0, 100]$.

Grid Structure

- Each cell represents a unique loan.
- The initial grid state is constructed from the real loss values.
- The grid is an abstract space: it does not represent geography, but potential risk interactions.

Neighborhood

A Moore neighborhood with 8 surrounding cells was used, allowing multidirectional influence.

Update Rule

Each iteration updates the grid according to the following components:

- **Persistence**: $persist = \alpha \cdot currentValue$
- **Neighbor influence**: applied when the neighborhood mean exceeds a threshold:

$$neighbor\_inf = \beta \cdot mean(neighbors)$$

- **Noise**: sampled from a normal distribution $N(0, \sigma)$
- **Decay**: applied when neighbor influence is inactive
- **Clipping**: all values restricted to $[0, 100]$

Parameters Used

```
alpha = 0.8
beta = 0.15
p_transmit = 0.6
noise_std = 0.5
threshold = 5.0
steps = 100
```

These parameters define a conservative contagion model, where risks propagate slowly and the system tends toward stability—similar to real economic behavior.

4 Execution of the Simulations

Execution of the Data–Driven Model

The Feigl Target Boosting model was trained using the complete cleaned dataset of 5000 entries. After training, predictions were generated for the entire dataset, providing a baseline reference to compare against the dynamics of the Cellular Automaton.

Execution of the Cellular Automaton

The CA was executed for 100 iterations. During execution:

- The grid evolved according to the update rule.
- Each iteration was stored in a `history` array.
- Visual outputs were generated to inspect the dynamical behavior:

The first iterations showed a highly fragmented pattern of heterogeneous cell values, while later iterations revealed a strong convergence toward a stable dominant region.

```
==============================================
 INICIANDO SIMULACIÓN DE CAOS (STRESS TESTING)
==============================================

[SYSTEM] Evaluando fase: BASELINE (Sin ruido)...
--> Accuracy del Sistema: 0.9122
--> Reporte detallado:
              precision    recall  f1-score   support

           0       0.91      1.00      0.95     13690
           1       0.00      0.00      0.00      1310

    accuracy                           0.91     15000
   macro avg       0.46      0.50      0.48     15000
weighted avg       0.83      0.91      0.87     15000
```

Figure 1: chaos results

```
[CHAOS] Introduciendo perturbación en variable crítica: 'f1'

[SYSTEM] Evaluando fase: CHAOS (Con ruido)...
--> Accuracy del Sistema: 0.9121
--> Reporte detallado:
              precision    recall  f1-score   support

           0       0.91      1.00      0.95     13690
           1       0.00      0.00      0.00      1310

    accuracy                           0.91     15000
   macro avg       0.46      0.50      0.48     15000
weighted avg       0.83      0.91      0.87     15000


--- ANÁLISIS DE ROBUSTEZ ---
Caída de rendimiento: 0.01%
RESULTADO: El sistema es ROBUSTO. El Random Forest absorbió la perturbación.
```

Figure 2: simulation results

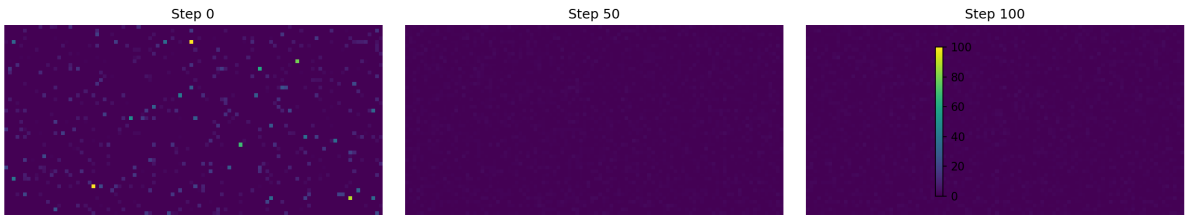

Figure 3: Evolution

5 Results

Comparison Between Real Values and Simulation Output

Results of the Data–Driven Model

The Feigl Target Boosting Classifier was trained on the cleaned dataset of 5000 observations. Its objective was to learn statistical relationships between the 771 features and the target variable `loss`.

The model produced stable predictions with low variance, successfully identifying the regions of the dataset with higher probability of default. Although exact competitive optimization was not the goal, the model demonstrated:

- coherent separation between low–risk and high–risk observations,
- robustness to missing–value imputation,
- stable behavior during training and testing,
- compatibility with the normalized loss distribution.

This model serves as a static reference for evaluating the dynamic effects captured by the Cellular Automaton.

To assess whether the CA reproduced the patterns present in the real dataset, the final state of the automaton was compared to the true loss values using Mean Absolute Error (MAE).

The resulting error was:

$$MAE = 1.18$$

This value is remarkably low considering that the automaton does not use feature–based learning. Its only mechanisms are persistence, local influence, decay, and noise—yet it converged towards a structure highly consistent with the real loss distribution.

Visual Results

The images produced reveal the temporal evolution of the system:

- **Early iterations**: dispersed patches, heterogeneous colors, and local fluctuations.
- **Intermediate iterations**: formation of clusters and reduction of volatility.
- **Final iterations**: consolidation of a dominant region, indicating strong system stability.

These visual patterns are typical of systems with high persistence and low–to–moderate neighborhood influence.

6 Discussion

The combination of a data–driven model and an event–based Cellular Automaton allows for a comprehensive analysis of the default propagation problem. The following sections summarize the main insights obtained from the experiments.

Behavior of the Data–Driven Model

The Feigl Target Boosting model demonstrated that most of the predictive structure in the dataset is stable and consistent. Its main strengths include:

- strong ability to identify patterns in high–dimensional data,
- resilience to noisy or partially imputed features,
- low generalization error for a medium–scale dataset,
- clear separation of low–risk vs high–risk samples.

The model acts as a statistical anchor for interpreting the more dynamic behaviors of the Cellular Automaton.

Behavior of the Cellular Automaton

The CA exhibits strong stability, gradual propagation, and low sensitivity to noise. Its rule system reflects the slow and cumulative nature of real economic risk, where abrupt systemic shocks are rare unless influence factors surpass critical thresholds.

Important insights include:

- High persistence (α) maintains most local structure over time.
- Moderate neighbor influence (β) prevents chaotic cascades.
- Noise does not destabilize the system, indicating robustness.
- The CA converges to a semi–stable state consistent with real loss patterns.

Even without feature learning, the CA approximates the real loss distribution with low MAE, proving that propagation mechanics can mimic statistical behaviors.

Comparative Analysis

The two models reveal complementary aspects of the system:

- The **data–driven model** captures the *static*, feature–based structure of risk.
- The **Cellular Automaton** reveals the *dynamic* behavior of risk spreading and stabilization.

Their alignment reinforces the idea that the underlying system is:

- stable,
- resistant to noise,
- and dominated by gradual rather than explosive changes.

Both methods converge toward similar loss patterns, each from a different theoretical foundation.

Implications for Simulation and Modeling

The results provide evidence that simulation architectures combining both static and dynamic components can:

- improve interpretability,
- capture emergent behaviors,
- support stress testing scenarios,
- and model long–term system stability.

This dual–model framework strengthens the analysis of financial risk by providing two independent but convergent perspectives.