Simulation Report

Daniel David Cuellar, Henry Jhonmarcos Osorio Sánchez,
Juan Andrés Jimenez Palomino, Miguel Alejandro Chavez Porras

Universidad Distrital Francisco Jose de Caldas

Abstract. In this document, we analyze the proposed dataset (KKBox Music Recommendation Challenge) and prepare the data—performing the necessary cleaning steps—to enable simulations that will help us solve our problem.

Keywords: Dataset · Analysis · Simulation

1 Data Preparation

The dataset was fetched from the same closed-priced Kaggle competitio, the KKBox Music Recommendation Challenge. Raw competition files included user profiles, song metadata, and extended music information, which serve as the primary sources to simulate learning behavior and user interaction flows in the system. The fetched data sources are confirmed as: members.csv, songs.csv, song_extra_info.csv, and train.csv, all obtained from the competition dataset.

Data cleaning and preprocessing were performed with a domain-driven data preparation script (membersclean.py) that merges all relevant files and applies attribution-aware sanitization to avoid unrealistic or unstable learning inputs. The members table was cleaned, with invalid age values replaced by NaN when outside the human range (≤ 0 or > 100) and missing gender values imputed as "unknown", reducing potential noise that could generate unpredictable simulation divergence.

Temporal fields (registration_init_time, expiration_date) were converted to proper datetime formats to stabilize time-dependent learning and downstream evaluation. Song attributes such as genre_ids, artist_name, composer, and lyricist were also filled with "unknown", ensuring encoding feasibility. Duplicated user–song interactions were removed to prevent circular learning loops. All data were merged into a unified dataset and saved as clean_kkbox.csv (22 features, 7,377,417 records).

To keep simulations computationally feasible, a reduced subset was later used for modular inference validation (accuracy.py), loading 50,000 rows with 22 columns, demonstrating system scalability control and workflow stability when working with smaller data slices.

Data characteristics were summarized directly during pipeline execution. The final cleaned dataset contains:

Total features (columns): 22 Total samples: 7,377,417 (full ML simulation) Reduced simulation slice: 50,000 samples (saved and validated) Feature types include: demographic user variables, temporal validity ranges, categorical source context, song numerical length, genre density, and ISRC-derived country/year features.

Main data characteristics were automatically inspected at model load time:

Full dataset shape: (7,377,417, 22) Columns detected: msno, song_id, source_system_tab, source_screen_name, source_type, target, city, bd, gender, registered_via, registration_init_time, expiration_date, song_length, genre_ids, artist_name, composer, lyricist, language, song_length_min, num_genres, isrc_country, isrc_year.

Missing value handling summary:

Unrealistic age values were replaced with NaN. Core categorical variables were safely imputed as "unknown" before encoding (gender, genre_ids, composer, source attributes). Dataset rows were filtered using a sparsity-threshold rule requiring at least 50% valid data per record.

Distribution summary (domain observation):

The target replay variable shows an approximately balanced binary distribution. Categorical source attributes present high cardinality and long-tail sparsity, which was considered in the design as a potential source of sensitivity. Numerical features such as bd (age), song_length, and language encoded as numerical categories show a wide range but were preserved to allow classical learning behavior simulation without normalization collapse.

These preprocessing, inspection, and reduction steps ensure that both simulation modes required in the workshop (data-driven ML learning + event-based automata) remain feasible and stable within resource limits, while maintaining fidelity to real competition data to validate the designed system architecture.

2 Simulation Planning

Scenario 1: Data-Driven ML

Scenario 1 Goal: Simulate system learning and replay-prediction behavior using a classical supervised ML model, exercising the Prediction Service module defined in Workshop #2.

Planned ML predictor: A Random Forest classifier was selected because the system requires probabilistic ML-based prediction, aligning with the model types proposed in Workshop #2.

Planning guided by prior data-analysis: All feature-engineering, cleaning, and encoding decisions were planned and executed by directly replicating the same data-driven analytical rationale applied by the Medium author, whose pipeline focuses on demographic validation, categorical imputation, interaction-source modeling, and feature relevance filtering.

Dataset planned for use: clean_kkbox.csv — generated from merging raw competition sources, validated for structural consistency before simulation. The full dataset contains 22 attributes and 7.4M rows, while smaller slices were planned for feasibility monitoring.

Feature set planned for learning simulation: 8 predictors selected after relevance filtering, exactly matching the features used in the prototype training script.

Encoding strategy planned: Label-based categorical encoding fitted on the combined feature space to avoid unseen-category disruption, ensuring modular stability when consumed by other architectural components.

Train/Test execution plan: Controlled, stratified 80/20 split, with fixed random seed to guarantee reproducibility when sweeping parameters or re-running simulations.

Expected model outputs:

Probabilistic replay predictions exported in submission structure
Model artifacts serialized for independent modular consumption
Performance metrics logged into the Metrics Store

Success evaluation plan: The simulations were planned to track ranking-quality and pipeline stability using ROC-AUC as the primary metric and accuracy as a secondary validation, comparing results across full and reduced inference slices to verify sensitivity to data integrity.

3 Simulation Implementation

3.1 Scenario 1: Data-Driven ML

For Scenario 1, a classic ML prediction workflow was implemented using a Random Forest classifier to simulate replay learning and song-replay probability estimation, exercising the system's prediction component defined in Workshop #2.

Training process:

The cleaned analytical dataset clean_kkbox.csv (22 features, 7,377,417 records) was loaded into the modeling module.

Eight core predictors were selected and encoded consistently across train/test feature space.

The data were divided into an 80/20 stratified train–test split, preserving target proportions.

Model sensitivity observation: initial experiments performed directly on test.csv produced poor approximations and unstable categorical representations. Performance only improved once the datasets were cleaned, irrelevant noise was mitigated, and consistent category labels were enforced, enabling the model to learn meaningful patterns.

Model training outcome: the Random Forest model was constrained to moderate complexity and randomness was controlled using a fixed seed for reproducible learning behavior.

Evaluation metrics computed:

Full test partition (1,475,484 samples):
$ROC - AUC = 0.6634, Accuracy \approx 0.63$

Reduced 50,000-row inference validation slice:
$ROC - AUC = 0.7345, Accuracy = 0.7435$

These results confirm that the data-driven ML predictor successfully exercised and validated ingestion → encoding → learning → probabilistic replay inference, while demonstrating the pipeline's sensitivity to data quality and categorical consistency.

The performance trend supports the design hypothesis: feature relevance and data integrity directly drive learning stability, while raw unfiltered inputs amplify chaos and degrade approximation quality.

3.2 Scenario 2: Event-Driven Cellular Automata

For the event-based prototype, a Cellular Automata (CA) model using Moore neighborhood (8-adjacency without periodic wrap-around) was implemented to represent local behavioral propagation of replay tendency.

Simulation space and initialization constraints:

A 50×50 two-dimensional grid was used, providing a feasible state space (2,500 simulation nodes) while keeping computational cost and visualization interpretable.

The grid was initialized in a fully unbiased inactive state (0s), representing a system with no prior replay activation, allowing event dynamics to dominate the observed behavior.

The simulation evolved for 300 discrete iterations, each representing a bounded learning window in which events act asynchronously on local cells rather than refreshing the entire grid.

Event-based transition design:
At each iteration a stochastic number of localized events (1 to 8) were generated, exercising only the affected cells

Triggered Event	Intention
Song Play (0.14)	observation event, no direct state change
Replay Activation (0.10)	local activation to state 1
Neighbor Influence (0.35)	conditional activation if neighbors ≥ 3
Random Noise (0.01)	state flip perturbation

Table 1. Event-Intention Table

A local influence threshold of 3 active neighbors was selected to ensure that state adoption requires meaningful adjacency pressure, preventing immediate activation by single random plays, approximating a realistic multi-user feedback influence chain.

Chaos-sensitivity insight:
During execution, the CA displayed high sensitivity to minor probability changes. This indicates that the rule set behaves as a perturbation-sensitive discrete dynamic system, where small local noise or influence-probability shifts sharply alter the global number of activated cells.

This sensitivity generated inverted outcomes across runs (large activation patches turning inactive and vice-versa), making the system appear locally ordered but globally unstable under small random fluctuations. Such behavior is consistent with bounded stochastic CA systems near phase transition thresholds, where replay propagation is neither purely random nor permanently stable, but easily inverted when damping or noise slightly dominates.

4 Executing the simulations

The data-driven CA was executed multiple times with fixed seeds and varied event rates. Runs without prior cleaning or categorical selection produced poor replay activation approximation. Only after datasets were cleaned in earlier stages did the system's architecture show predictable patterns in ML, while CA demonstrated stability limits.
This, system performance was examined by modulating probabilities rather than data slices, revealing:
No permanent bottleneck, but Sharp activation inversions when noise slightly dominated, showing high sensitivity rather than catastrophic failure.

5 Results and Discussion

Using the Machine Learning Simulation we see a stable and predictable behavior, strongly dependent on data quality, categorical consistency, and the level of preprocessing applied.These results indicate that the model captures meaningful replay patterns but is highly sensitive to noise and uncleaned categories.
In contrast, the Cellular Automata model showed high sensitivity to small fluctuations, exhibiting chaotic-like behavior. Minor changes in event probabilities produced drastically different outcomes.
Unlike the ML predictor, the CA does not learn from real data; instead, it simulates local propagation dynamics of replay tendencies based on stochastic rules. Now some **similarities** Between Both Approaches.

Despite their methodological differences, both models share important properties:

- **Dependence on Initial Conditions:** ML depends on preprocessing quality; CA depends on initialization and event probabilities.

- **Sensitivity to Noise:** ML performance decreases with noisy categories; CA drastically shifts global behavior with small perturbations.
- **Need for Calibration:** Both models require careful parameter tuning to produce realistic outcomes.
- **Fragility When Misconfigured:** Incorrect data cleaning or unbalanced probabilities lead to misleading results in both simulations.

Recommended improvements for the ML simulation:

- Reduce categorical cardinality by grouping rare values.
- Explore embedding-based representations for songs, artists, and genres.
- Test gradient boosting models such as LightGBM or XGBoost.
- Apply noise-reduction strategies to the full dataset.
- Evaluate feature importance to remove irrelevant predictors.

Recommended improvements for the CA simulation:

- Introduce activation decay to reduce instability.
- Add intermediate activation states to model gradual replay tendency.
- Experiment with alternative neighborhoods (e.g., Von Neumann).
- Implement partially synchronous updates to mitigate volatility.
- Calibrate event probabilities based on expected replay distributions.

References

1. https://www.kaggle.com/competitions/kkbox-music-recommendation-challenge/overview
2. https://github.com/EngAndres/ud-public/tree/main/courses/systems-analysis
3. Systems Analysis and Design, by Alan Dennis, Barbara Haley Wixom, and Roberta M. Roth