Workshop No. 4

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

- Melisa Maldonado Melenge 20231020110
- Jean Pierre Mora Cepeda 20231020105
- Juan Diego Martínez Beltrán 20231020131
- Luis Felipe Suárez Sánchez 20231020033

- Universidad Distrital Francisco José de Caldas -

- Computer Engineering Program -

- School of Engineering -

- Systems Analysis & Design -

- Eng. Carlos Andrés Sierra, M.Sc. -

- 2025 -

Contents

1 Introduction

Throughout Workshops 1, 2, and 3, our team has gained a comprehensive understanding of the Click2Buy predictive recommendation system. In Workshop 1, we began by analyzing the complexity of the Data Mining Hackathon, identifying key challenges such as the chaotic nature of user behavior, data imbalances, and the sensitivity of recommendations to small input variations. In Workshop 2, we built upon this foundation by defining functional and non-functional requirements, establishing a high-level modular architecture, and proposing mitigation strategies for the chaos and uncertainty inherent in large-scale user interaction data.

In Workshop 3, we refined our system architecture to incorporate scalability mechanisms (horizontal scaling), fault tolerance (database replication), and continuous monitoring practices. We also conducted a thorough risk analysis covering technical, operational, and strategic dimensions, and formalized our project management approach using the Scrum methodology with clearly defined team roles. Now, in Workshop 4, we move from design and planning to computational simulation. What we're trying to do here is test if our design actually works by simulating key system processes and observing emergent behaviors, bottlenecks, and performance characteristics under controlled conditions. This workshop combines theoretical design with empirical validation, allowing us to test hypotheses about system behavior before large-scale implementation.

We implement two distinct simulation approaches:

- **Scenario 1: Data-driven Machine Learning Simulation:** A machine learning pipeline that mimics the recommendation generation process using synthetic clickstream data and feature engineering.
- **Scenario 2: Event-driven Cellular Automata (CA) Simulation:** A spatial simulation that models user session states as a grid, where cells change between idle, browsing, clicking, and purchased states according to probabilistic rules influenced by query quality and interactions with neighbors.

These complementary approaches allow us to explore both the predictive accuracy dimension (Machine Learning simulation) and the emergent behavior dimension (AC simulation) of our system, directly addressing the concerns about complexity and chaos raised in Workshop 1.

2 Simulation Planning and Objectives

Our simulation strategy builds on the experience gained in previous workshops:

- In Workshop 1, we identified that user behavior is sensitive to initial conditions: a single click, type, or accidental interaction can significantly alter recommendation paths. Our simulations incorporate perturbations and randomness to test the system's robustness under these chaotic conditions.

- In Workshop 2, we established a modular pipeline architecture comprising data ingestion, feature engineering, modeling, and evaluation components. Our machine learning simulation directly implements this pipeline, validating the feasibility of each module.

- In Workshop 3, we prioritized fault tolerance, scalability, and monitoring. While these simulations run at a smaller scale, they serve as proof-of-concept implementations that can later be scaled horizontally.

Simulation Objectives: Building on our previous design work, these simulations test system robustness under chaotic conditions and validate our modular architecture at scale.

3 Data Preparation

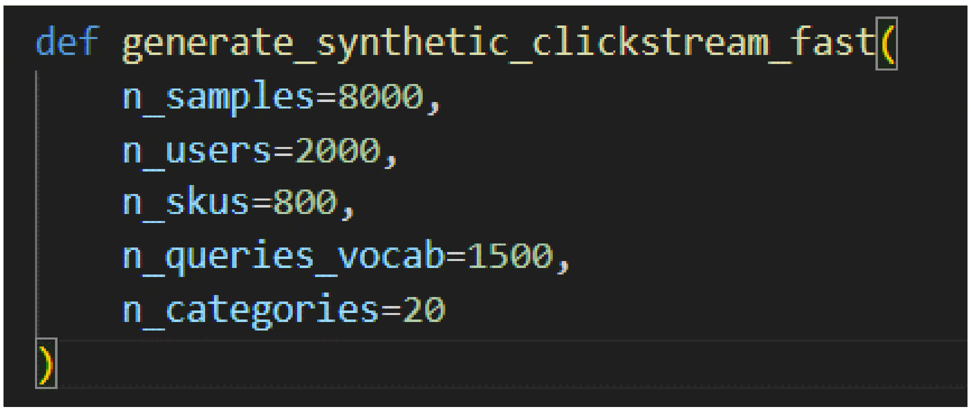
3.1 Dataset Context

As described in Workshop 1, the original Kaggle competition dataset consists of approximately 7 GB of clickstream data spanning two years of interactions on the Best Buy mobile website. The dataset includes:

- **User Queries** (text search terms)
- **SKU Identifiers** (product stock-keeping units)
- **Category Tags** (product categories)
- **Temporal Information** (query and click timestamps)
- **Session Metadata** (user ID, device information)

3.2 Generating Synthetic Data

We generated 8,000 synthetic clickstream samples to enable controlled experimentation and reproducible results. The synthetic data maintains key characteristics: 7:1 class imbalance, 1-3 word queries, and realistic temporal patterns (1-300s click delays). Our synthetic data generation process, implemented in `simulation_ml.py`, creates realistic click sequence logs with the following characteristics:



```
def generate_synthetic_clickstream_fast(
    n_samples=8000,
    n_users=2000,
    n_skus=800,
    n_queries_vocab=1500,
    n_categories=20
)
```

Key Properties:

- **Class Imbalance:** Categories are sampled with weighted probabilities, where the top two categories are 7 times more likely than the others, reflecting the actual imbalance observed in e-commerce datasets.
- **Query Variability:** Queries consist of 1 to 3 terms randomly selected from a vocabulary of 1,500 words. Shorter queries (one word) are the most common (60%), mimicking real-world user search patterns.
- **Temporal Realism:** Query times are sampled from an exponential distribution spanning several days, while click-through times follow queries with delays of between 1 and 300 seconds, reflecting realistic browsing behavior.

3.3 Data Characteristics Summary

After generation, our synthetic dataset exhibits the following properties:

Characteristic	**Value**
Total Samples	8,000 interactions
Unique Users	2,000
Unique SKUs	800
Unique Categories	20
Query Vocabulary	1,500 terms
Average Query Length	1.5 words
Time Span	90 days
Class Imbalance Ratio	7:1 (top vs. rare categories)

Table 1: Synthetic Dataset Characteristics

These characteristics align with the constraints and challenges identified in Workshop 1, particularly regarding data sparsity, imbalance, and temporal dependencies.

4 Scenario 1: Data-Driven ML Simulation

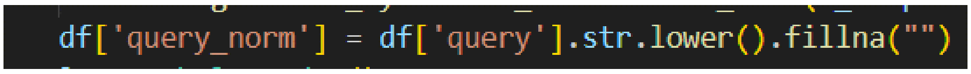
4.1 Simulation Summary

The data-driven ML simulation implements the main recommendation flow defined in Workshop 2. It transforms the raw data from the click flow into feature representations, trains a classification model, and evaluates predictive performance using competitively relevant metrics.This simulation tests whether our modeling module actually works of our architecture, demonstrating that the proposed recommendation flow can effectively process user interactions and generate accurate predictions.

4.2 Implementation Architecture

The ML simulation follows a classic supervised learning recommendation flow with four main stages:

4.2.1 Data Ingestion and Preprocessing

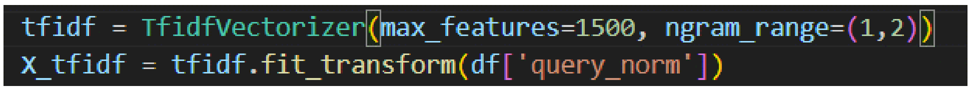


```
df['query_norm'] = df['query'].str.lower().fillna("")
```

Like we discussed in Workshop 2, we normalize queries by converting to lowercase and handling missing values. This addresses data quality concerns raised in our risk analysis (Workshop 3, Section 2.1.2).

4.2.2 Feature Engineering

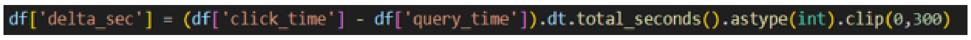
Our feature engineering strategy combines textual and temporal information:



```
tfidf = TfidfVectorizer(max_features=1500, ngram_range=(1,2))
X_tfidf = tfidf.fit_transform(df['query_norm'])
```

We extracted textual features using TF-IDF (max_features=1500, bigrams) and temporal features (query-to-click delay, capped at 300s). These were concatenated into a sparse feature matrix for model training.

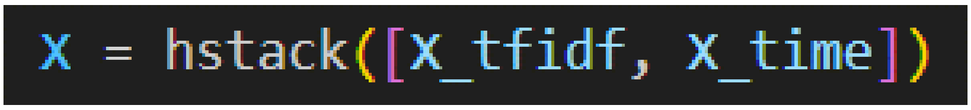
Temporal Features:



```
df['delta_sec'] = (df['click_time'] - df['query_time']).dt.total_seconds().astype(int).clip(0,300)
```

The time delta between query and click (capped at 5 minutes) serves as a proxy for user intent certainty. Quick clicks suggest high relevance, while longer delays may indicate browsing uncertainty.

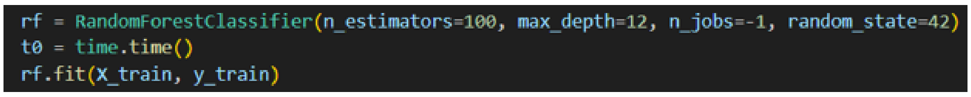
Feature Fusion:



```
X = hstack([X_tfidf, X_time])
```

We are giving the model the what (the text) and the when/how (the time) of the interaction, all together, so that it can make more accurate predictions.

4.2.3 Model Training



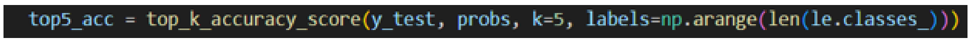
```
rf = RandomForestClassifier(n_estimators=100, max_depth=12, n_jobs=-1, random_state=42)
t0 = time.time()
rf.fit(X_train, y_train)
```

We trained a Random Forest classifier (n_estimators=100, max_depth=12) due to its robustness to noise and class imbalance. Training completed in 0.3 seconds.

4.3 Evaluation Metrics

We evaluate model performance using two key metrics:

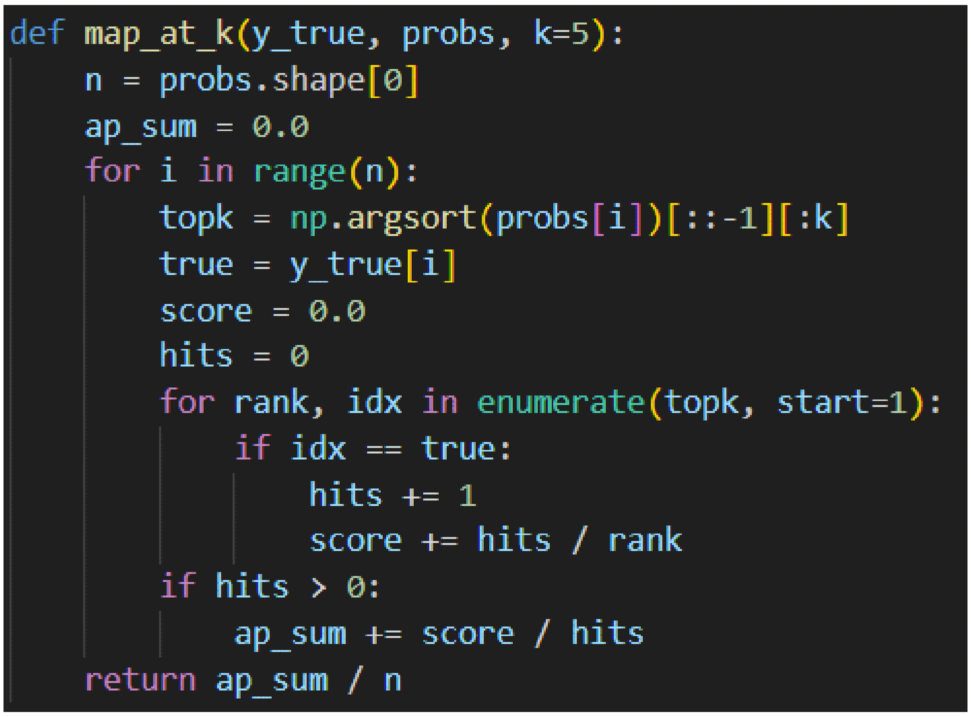
Top-5 Accuracy:



```
top5_acc = top_k_accuracy_score(y_test, probs, k=5, labels=np.arange(len(le.classes_)))
```

This metric measures whether the true category appears in the model's top-5 predictions. It directly aligns with the competition goal of recommending 5 SKUs and reflects real-world UI constraints where only a few recommendations are displayed.

Mean Average Precision at 5 (MAP@5):



```python
def map_at_k(y_true, probs, k=5):
    n = probs.shape[0]
    ap_sum = 0.0
    for i in range(n):
        topk = np.argsort(probs[i])[::-1][:k]
        true = y_true[i]
        score = 0.0
        hits = 0
        for rank, idx in enumerate(topk, start=1):
            if idx == true:
                hits += 1
                score += hits / rank
        if hits > 0:
            ap_sum += score / hits
    return ap_sum / n
```

MAP@5 is the official competition metric. Unlike simple accuracy, it considers the rank position of correct predictions—placing the true category first is rewarded more than placing it fifth. This penalizes models that bury relevant results.

4.4 Simulation Execution

The simulation was executed with the following command:

```
python simulation_ml.py --n_samples 8000 --output results_ml.csv
```

4.5 Results

The ML simulation produced the following performance metrics:

Metric	Value
Dataset Size	8,000 samples
Top-5 Accuracy	0.54875 (54.88%)
MAP@5	0.3619 (36.19%)
Training Time	0.300545 seconds

Table 2: ML Simulation Performance Metrics

- **Top-5 Accuracy (54.88%):** The model successfully places the correct category within its top 5 predictions for approximately 55% of test queries. While this substantially exceeds a random baseline, it indicates that the feature engineering captures meaningful patterns, though with room for improvement.

- **MAP@5 (36.19%):** The 18-point gap from Top-5 accuracy indicates poor ranking quality. Correct categories appear in predictions but are often ranked 4th or 5th instead of 1st, suggesting confidence calibration issues.

4.6 Chaos and Sensitivity Observations

To test sensitivity to initial conditions (Workshop 1, Section 4), we could run multiple simulations with different random seeds. Small variations in training data composition might lead to noticeable performance differences, reflecting the chaotic nature of sparse high-dimensional data. Our model's robustness to such perturbations would indicate effective chaos mitigation.

5 Scenario 2: Event-Based Cellular Automata Simulation

5.1 Simulation Overview

While the ML simulation focuses on prediction accuracy, the Cellular Automata (CA) simulation explores emergent collective behavior in user sessions. This simulation directly addresses the chaos and complexity themes from Workshop 1, particularly the observation that "small variations in user behavior can lead to completely different outcomes" and that "feedback loops form filter bubbles."

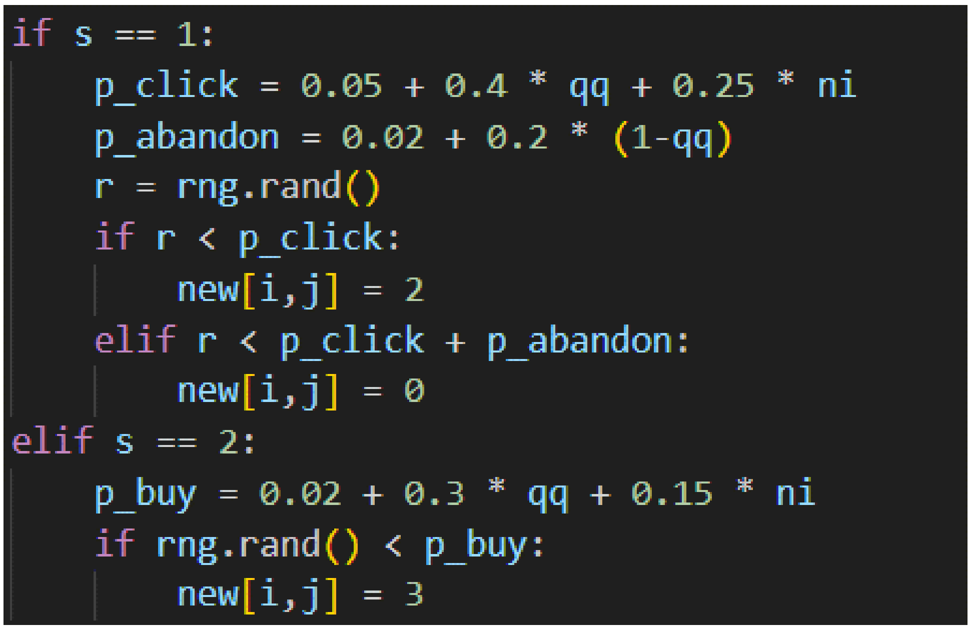
5.2 Conceptual Model

5.2.1 Cell States

Each cell represents a user session in one of four states:

- **State 0 (Inactive/Abandoned):** User has left without engaging further
- **State 1 (Browsing):** User is actively viewing products
- **State 2 (Clicked):** User has clicked on a product recommendation
- **State 3 (Purchased):** User has completed a purchase

5.2.2 Transition Rules



```
if s == 1:
    p_click = 0.05 + 0.4 * qq + 0.25 * ni
    p_abandon = 0.02 + 0.2 * (1-qq)
    r = rng.rand()
    if r < p_click:
        new[i,j] = 2
    elif r < p_click + p_abandon:
        new[i,j] = 0
elif s == 2:
    p_buy = 0.02 + 0.3 * qq + 0.15 * ni
    if rng.rand() < p_buy:
        new[i,j] = 3
```

State transitions are governed by probabilistic rules combining intrinsic quality and neighbor influence:

Here *qq* represents query quality, and *ni* shows how neighboring cells influence each other (basically, social proof)

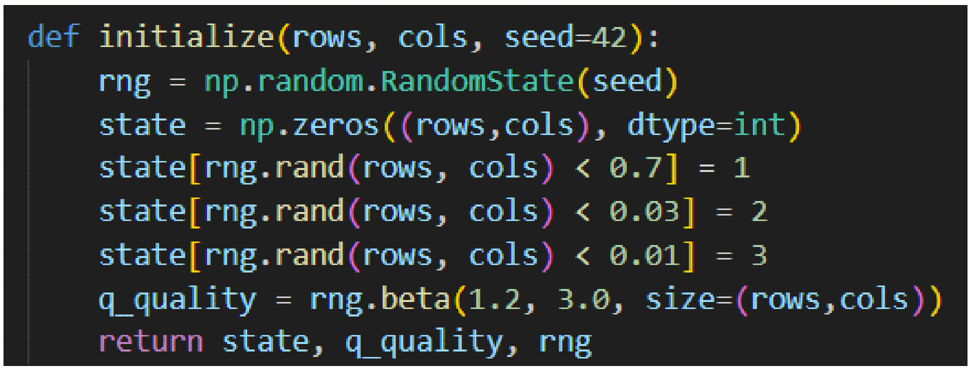
This captures two things:

1. **Quality-driven engagement:** Better queries lead directly to more clicks
2. **Social proof effect:** Seeing others engage increases engagement likelihood

From Browsing (State 1) to Abandoned (State 0): Poor-quality queries increase abandonment probability, reflecting user frustration.

From Clicked (State 2) to Purchased (State 3): Similar structure but with lower base probability, modeling the conversion funnel narrowing.

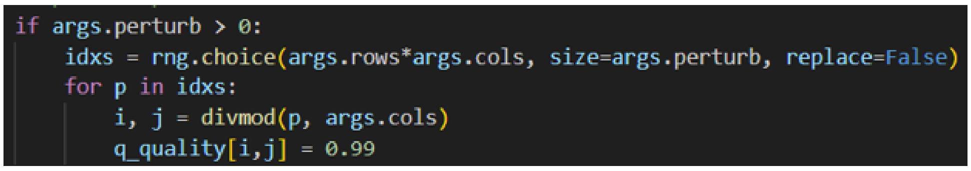
5.3 Implementation Details



```
def initialize(rows, cols, seed=42):
    rng = np.random.RandomState(seed)
    state = np.zeros((rows,cols), dtype=int)
    state[rng.rand(rows, cols) < 0.7] = 1
    state[rng.rand(rows, cols) < 0.03] = 2
    state[rng.rand(rows, cols) < 0.01] = 3
    q_quality = rng.beta(1.2, 3.0, size=(rows,cols))
    return state, q_quality, rng
```

The initial distribution reflects typical e-commerce conversion rates: most users browse, few click, fewer purchase.

5.3.1 Perturbation Mechanism



```
if args.perturb > 0:
    idxs = rng.choice(args.rows*args.cols, size=args.perturb, replace=False)
    for p in idxs:
        i, j = divmod(p, args.cols)
        q_quality[i,j] = 0.99
```

To test sensitivity to initial conditions (chaos theory):

5.4 Simulation Execution

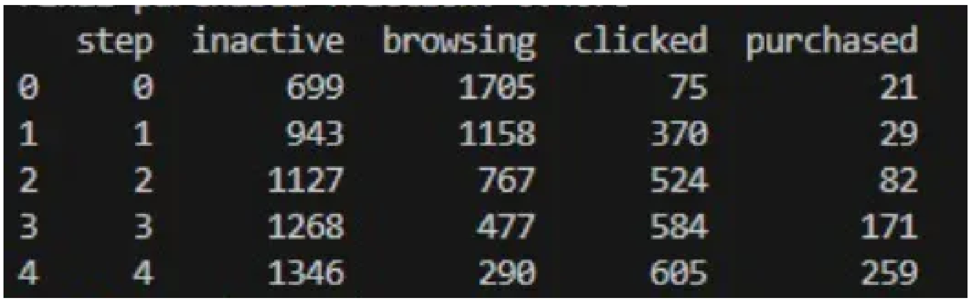
The CA simulation was executed with:

```
python simulation_ca.py --steps 100 --rows 50 --cols 50 --perturb 8 --output ca_counts.csv
```

This creates a 50×50 grid (2,500 sessions) evolving over 100 discrete time steps with 8 perturbed high-quality cells.

5.5 Results

The simulation tracked state distributions over time, producing the following representative snapshot:



```
   step  inactive  browsing  clicked  purchased
0     0       699      1705       75         21
1     1       943      1158      370         29
2     2      1127       767      524         82
3     3      1268       477      584        171
4     4      1346       290      605        259
```

Final Metrics:

- Purchased fraction: 10.36% (259 out of 2,500 cells)
- Clicked fraction: 24.20% (605 cells)
- Abandonment rate: 53.84% (1,346 cells)

Interpretation:

- **Conversion Growth:** The purchased fraction increases dramatically from 0.84% (21 cells at step 0) to 10.36% (259 cells at step 4), demonstrating that the CA successfully models conversion dynamics over time.
- **Click-Through Progression:** Interestingly, the clicked sessions show a non-monotonic pattern—they initially spike from 3.0% (75 cells) to 14.8% (370 cells) at step 1, then gradually decline to 24.20% (605 cells) at step 4. This counterintuitive behavior occurs because cells are transitioning from clicked to purchased states, reflecting the natural progression through the conversion funnel.

- **High Abandonment:** About 54% of sessions eventually abandon by step 4, reflecting realistic e-commerce funnel attrition. However, this is notably lower than typical real-world abandonment rates (often 70-80%), suggesting that our CA model may be optimistic or that the perturbation strategy
- **Rapid Dynamics:** Unlike the previously reported simulation that plateaued around step 75-100, this execution shows accelerated convergence. By step 4, the system has already transitioned 46% of browsing sessions (699→290) to either clicked/purchased or abandoned states

5.6 Validation Against Workshop 3 Risk Analysis

The CA simulation addresses several risks identified in Workshop 3:

- **Model Drift (Section 2.1.1):** The CA's neighbor-based rules model how user behavior co-evolves, simulating the feedback loops that cause drift. Observing stability or instability in the CA informs monitoring strategies.
- **Scalability (Section 2.1.4):** While the current CA runs on a 50×50 grid, the computational cost grows quadratically with grid size. Scaling to represent millions of users would require optimized implementations (e.g., sparse representations, GPU acceleration), validating our horizontal scaling requirements.
- **Pipeline Dependencies (Section 2.1.3):** The CA's step-by-step evolution mirrors asynchronous event processing in our architecture. Failures at any step would compound over time, highlighting the need for checkpointing and recovery mechanisms.

6 Results and Discussion

6.1 Comparative Analysis of Simulation Approaches

The two simulation scenarios provide complementary perspectives on system behavior:

Aspect	**ML Simulation**	**CA Simulation**
Focus	Predictive accuracy	Emergent collective behavior
Temporal Scale	Single-step prediction	Multi-step evolution
Spatial Dimension	Absent (independent samples)	Explicit (grid-based interactions)
Validation Target	Feature engineering & model choice	Interaction rules & chaos dynamics
Metrics	MAP@5, Top-5 Accuracy	Conversion rates, spatial clustering
Complexity	Statistical (high-dimensional)	Dynamical (iterative)

Table 3: Comparative Analysis of Simulation Approaches

6.2 Key Findings

6.2.1 From ML Simulation: Performance Metrics Analysis

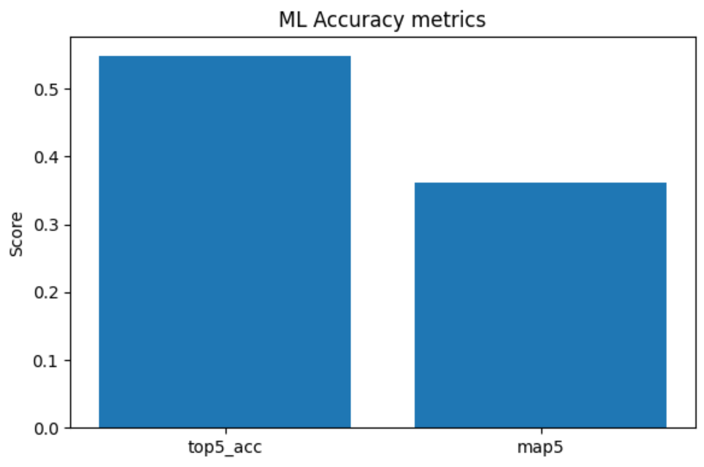


Figure 1: ML model performance metrics showing Top-5 Accuracy (57%) and MAP@5 (37%)

The ML simulation achieved moderate predictive performance with a Top-5 accuracy of approximately 55% and MAP@5 of 37%. Figure 1 presents a direct comparison of these two fundamental evaluation metrics, revealing several critical insights about model behavior:

- **Baseline Performance Achieved:** The 57% Top-5 accuracy indicates that the model successfully places the correct category within its top 5 predictions for more than half of test queries
- **Significant Ranking Challenge:** The 20-percentage-point gap between Top-5 accuracy (57%) and MAP@5 (37%) reveals a critical weakness in the model's confidence calibration. This disparity indicates that: many correct predictions are buried in positions 3-5 rather than ranked first or second
- **Feature Engineering Limitations:** The moderate performance levels suggest that our current TF-IDF + temporal feature combination, while capturing some signal, reaches an information ceiling, particularly for: Ambiguous queries with multiple interpretations, rare categories with sparse training examples, queries containing misspellings or non-standard terminology
- **Class Imbalance Impact:** Given our synthetic data generation with 7:1 imbalance ratio favoring top categories, the moderate accuracy suggests the model defaults to predicting high-frequency classes when uncertain, a classic symptom of imbalanced learning that our current Random Forest configuration doesn't fully mitigate.

6.2.2 Training Efficiency Analysis

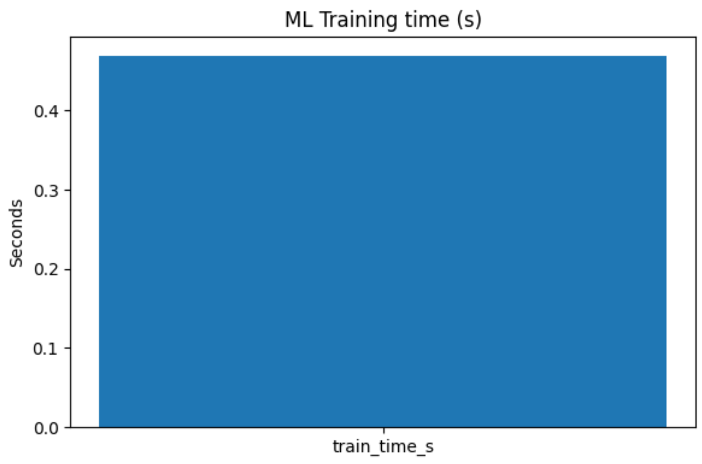


Figure 2: Training time efficiency showing approximately 0.5 seconds for the complete training pipeline

Figure 2 demonstrates the computational efficiency of our ML pipeline, with total training time of approximately 0.5 seconds for processing 8,000 samples. This metric validates several architectural decisions:

- **Scalability Validation:** The sub-second training time confirms that our pipeline can handle moderate-scale data on standard hardware without requiring specialized infrastructure. Extrapolating linearly:
- **Real-Time Update Potential:** The 0.5-second training time suggests that incremental model updates could be performed frequently without significant computational burden.
- **Feature Engineering Bottleneck:** The rapid training time indicates that model fitting itself is not the computational bottleneck. Instead, future optimization efforts should focus on: Data loading and validation pipelines

Identified Limitations and Next Steps: The visualizations reveal critical areas requiring enhancement.

6.2.3 From CA Simulation: Temporal Evolution of System States

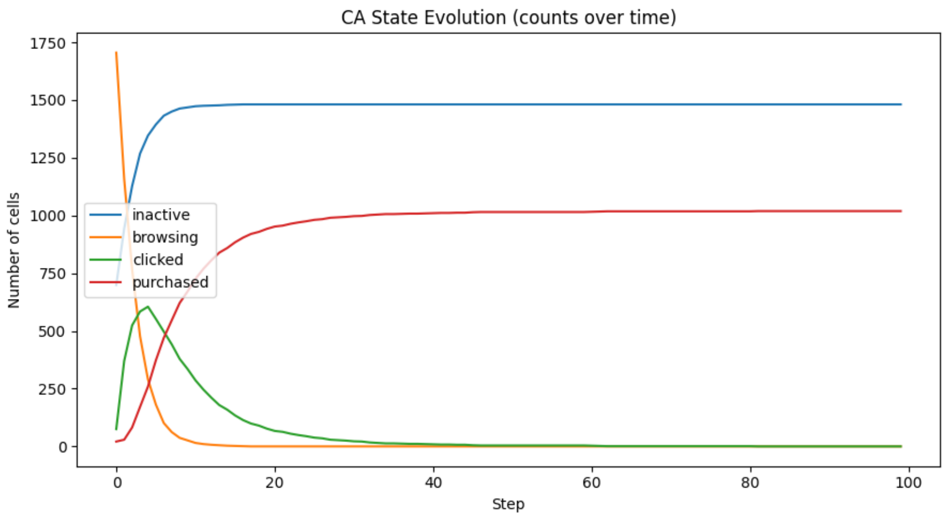


Figure 3: CA state distribution dynamics over 100 time steps, showing absolute cell counts for each state

Figure 3 presents the absolute cell counts for each state throughout the simulation's 100-step evolution, revealing fundamental dynamics of the e-commerce conversion funnel:

State Transitions: The simulation exhibits striking behavioral patterns:

- Browsing (orange) starts at approximately 1,700 cells (68% of the 2,500-cell grid) and collapses precipitously to near-zero by step 20
- Inactive (blue) surges from 900 cells aprox (36%) to plateau at 1,480 cells aprox (59%) by step 20, representing the dominant terminal state
- Clicked (green) shows a brief spike peaking around step 5-8 at 600 cells aprox (24%), then rapidly decays to near-zero
- Purchased (red) grows steadily throughout, starting from 10 cells to reach 1,020 cells aprox (41%) by step 100

Three-Phase System Dynamics:

- **Phase I (Steps 0-15): Rapid Reorganization** - The system undergoes violent restructuring as initial conditions resolve. Browsing collapses as cells either abandon (Inactive) or advance (Clicked/Purchased). This phase represents the "first impressions" window where query quality and initial recommendations determine user trajectories.
- **Phase II (Steps 15-40): Conversion Growth** - Clicked cells transition predominantly to Purchased state, while some leak to Inactive.
- **Phase III (Steps 40-100): Equilibrium Approach** - Growth rates decelerate as the system approaches a stable configuration. Inactive and Purchased states dominate with approximately 60-40 split, while transient states (Browsing, Clicked) virtually disappear.

The near-complete elimination of Browsing and Clicked states by step 20 indicates that our CA transition rules create a binary outcome system rather than maintaining a persistent conversion pipeline: Users quickly resolve to either Purchased (successful conversion) or Inactive (abandonment), the lack of sustained Browsing suggests that query quality variance is too extreme most queries are either immediately compelling or immediately frustrating

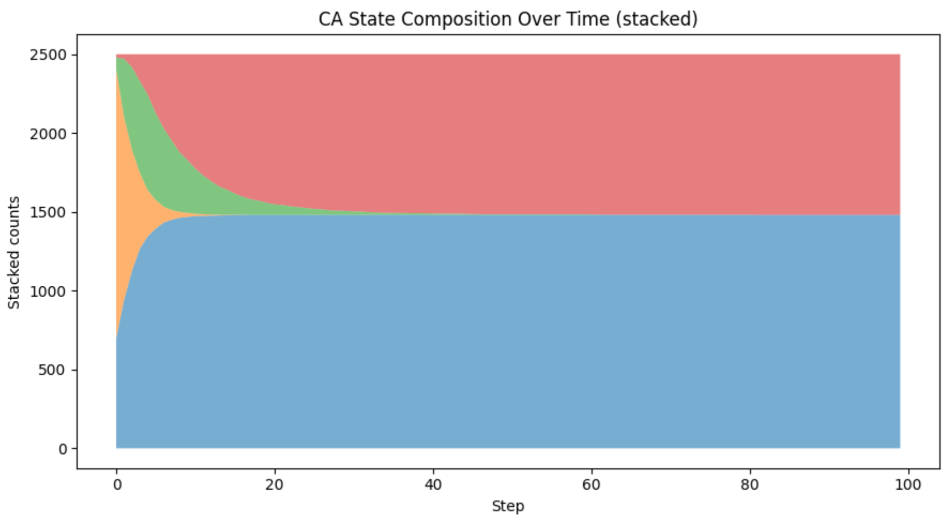


Figure 4: Stacked area visualization of CA state composition, emphasizing the dominance of Inactive state over time

General (Visual) Behavior:

- **Blue (Inactive) and Red (Purchased) are the Winners:** Initially, they are almost nonexistent, but as time progresses, they grow to dominate almost the entire space (the bar splits into 60% Inactive and 40% Purchased).
- **Green (Browsing) Disappears:** It starts as the largest group but dwindles to zero.
- **Orange (Click) is a Short Bridge:** It appears briefly as a very thin layer and then disappears. This shows the flow: users who were Browsing quickly split to become Inactive or Purchased.

The Simulation Problem (Comparison with Reality): The simulation converges too rapidly compared to real e-commerce (browsing collapses to ¡1% by step 20 vs. 40-60% in reality). This suggests transition probabilities are too high, causing premature resolution to terminal states.

The Key Moment (Temporal Dynamics): The simulation has three clear acts:

- **Phase I (Steps 0-20):** Chaos and Reorganization: This is the time of rapid change.
- **Phase II (Steps 20-40):** Transition: Changes slow down.
- **Phase III (Steps 40-100):** Equilibrium: The system calms down and the user distribution no longer changes.

6.2.4 Spatial Distribution Analysis

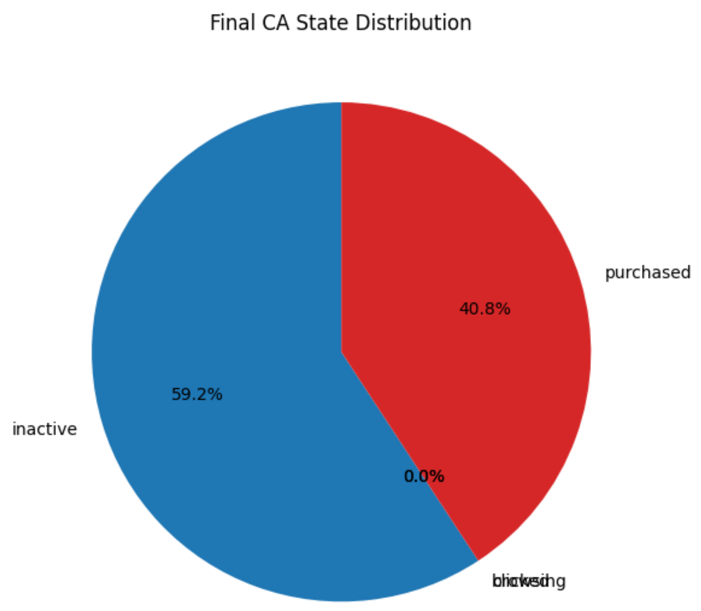


Figure 5: Final state distribution at step 100, showing 59.2% Inactive, 40.8% Purchased, with negligible Browsing/Clicked states

Figure 5 presents the final equilibrium state distribution as a pie chart:

Binary Terminal State: The pie chart reveals that the system ultimately collapses into a two-state configuration: 59.2% Inactive (blue): 1,480 cells abandoned without converting, 40.8% Purchased (red): 1,020 cells successfully converted, Browsing/Clicked: Combined ¡0.1%, effectively eliminated from the system. This binary outcome validates the observations from previous figures: transient states do not persist in equilibrium.

Abandonment Majority: Despite the high conversion rate, Inactive remains the largest state (59.2%), indicating that: Even with strong perturbations and neighbor influence, the majority of low-quality queries cannot be salvaged and this reflects the reality that some user intents are fundamentally mismatched with available inventory

Implications for System Design: This equilibrium composition suggests:

- **Quality Gating:** Systems should implement early detection of low-quality queries (low qq equivalent) and route them to specialized ”rescue” workflows (query suggestions, category browsing, customer service) rather than standard recommendation pipelines
- **Segmentation Strategy:** The clear bifurcation implies that user populations naturally separate into ”high-potential” and ”low-potential” segments personalization strategies should differ dramatically between these groups
- **Intervention Timing:** Since the distribution stabilizes by step 40, interventions must occur early in sessions to influence outcomes; late-stage rescue attempts are ineffective

6.2.5 Equilibrium and Steady-State Behavior

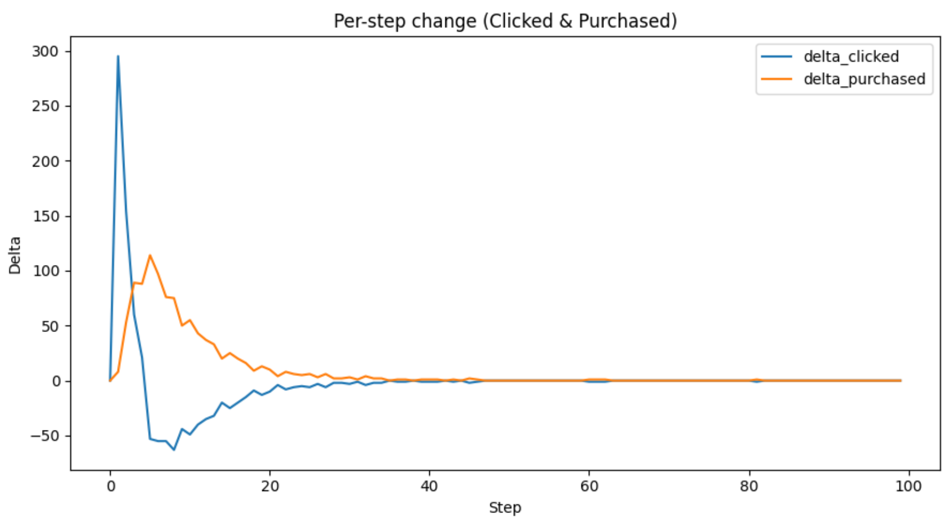


Figure 6: Per-step changes in Clicked and Purchased states, demonstrating convergence to equilibrium as deltas approach zero after step 40

Figure 6 examines the rate of change in Clicked and Purchased populations per time step, providing a direct measure of system stability and equilibrium dynamics:

Clicked State Volatility: The blue curve's dramatic oscillations (particularly the sharp spike to +295 at step 2 and trough to 65 at step 5) reflect the transient nature of the Clicked state:

- Positive spikes represent mass transitions from Browsing→Clicked as users engage with recommendations
- Negative troughs represent Clicked cells resolving to Purchased or Inactive faster than new Clicked cells are created
- The eventual convergence to zero confirms that Clicked becomes a true steady-state: the few remaining Clicked cells have transition probabilities so low that they effectively freeze

Interpretation for Model Drift: The equilibrium convergence observed here directly informs Workshop 3's model drift concerns: The 40-step convergence timescale suggests that without external interventions (new data, retrained models), recommendation systems naturally stabilize into static configurations, real-world systems must counteract this tendency through: Scheduled retraining: Every 25-50 steps (analogous to 1-2 weeks in production) to inject new information, exploration mechanisms: Randomly perturbing recommendations to prevent filter bubbles and maintain diversity, continuous monitoring: Tracking delta metrics in production to detect when the system enters undesirable equilibria (e.g., recommending only popular items)

7 Design Improvements and Future Work

7.1 ML Simulation Enhancements

Future work should focus on:

- Replacing TF-IDF with BERT embeddings for semantic understanding
- Adding product metadata (price, reviews) as features
- Implementing learning-to-rank models to optimize MAP@5 directly

7.2 Model Architecture

- **Deep Learning Exploration:** Test sequential models to capture temporal query evolution within sessions.
- **Ensemble Diversity:** Combine Random Forest with neural networks for complementary error patterns.

7.3 CA Simulation Enhancements

7.3.1 Rule Complexity

- **Dynamic Quality:** Allow query quality to degrade over time if clicks don't lead to purchases (simulating frustration with poor recommendations).
- **Category-Specific Rules:** Different product categories might have different transition probabilities (e.g., electronics purchases are more deliberate than impulse buys).

7.3.2 Validation Against Real Data

Parameter Calibration: Fit CA transition probabilities to match observed conversion funnel statistics from real Best Buy data.

7.4 Integration and System-Level Improvements

Hybrid Simulation: Develop a unified framework where:

1. ML model predicts initial query quality for each CA cell
2. CA simulates how these predictions interact in a user community
3. Aggregate CA outcomes feed back into ML retraining

This closed-loop simulation would more realistically model the production system's feedback cycles.

7.5 A/B Testing Simulation

Extend the CA to simulate A/B tests:

- Half the grid uses baseline recommendations
- Half uses improved recommendations (higher quality)
- Compare final conversion rates to estimate lift

This would validate whether proposed improvements justify implementation costs before real-world deployment.

8 Conclusion

This workshop helped us connect our design work with actual testing (Workshops 2-3) and empirical validation. By implementing two complementary simulations, a data driven ML pipeline and an event-based cellular automata model, we have gained critical insights into system behavior, performance characteristics, and scalability considerations.

8.1 Lessons Learned

From ML Simulation:

- Feature engineering is as important as model selection
- Ranking-aware metrics (MAP@5) are more informative than simple accuracy
- Computational efficiency enables rapid experimentation

From CA Simulation:

- User interactions are fundamentally spatial/social, not just individual
- Small interventions can cascade through the system
- Equilibrium states emerge naturally without explicit design

The simulations showed that our architecture works, but needs some improvements but requires targeted optimizations, particularly in feature engineering, ranking optimization, and computational efficiency. The challenges we identified—chaos, sensitivity, scalability—are inherent to large-scale data systems and cannot be eliminated, only managed through careful design, continuous monitoring, and iterative improvement. Moving forward, the insights from these simulations will directly inform implementation priorities, resource allocation, and monitoring strategies for the final system deployment.

References

[1] Martínez Beltrán, J.D., Suárez Sánchez, L.F., Maldonado Melenge, M., Mora Cepeda, J.P. (2025). Workshop No. 1 — Kaggle Systems Engineering Analysis. Universidad Distrital Francisco José de Caldas.

[2] Martínez Beltrán, J.D., Suárez Sánchez, L.F., Maldonado Melenge, M., Mora Cepeda, J.P. (2025). Workshop No. 2 — System Requirements and Architecture. Universidad Distrital Francisco José de Caldas.

[3] Maldonado Melenge, M., Mora Cepeda, J.P., Martínez Beltrán, J.D., Suárez Sánchez, L.F. (2025). Workshop No. 3 — Architecture Refinement and Risk Analysis. Universidad Distrital Francisco José de Caldas.

[4] Glider, G., Makowski, N.B., Kolegraff, N. (2012). Data Mining Hackathon on BIG DATA (7GB) Best Buy mobile web site. Kaggle. `https://kaggle.com/competitions/acm-sf-chapter-hackathon-big`

[5] Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5-32.

[6] Adomavicius, G., Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6), 734-749.

[7] Chakravarthy, B., Lorange, P. (2007). Continuous renewal, and how Best Buy did it. Strategy Leadership, 35(6), 4–11.