YZV411E Big Data Analytics
Progress Report
E-Commerce User Behaviour Analysis

Abdulkadir Külçe (150210322) Berkay Türk (150220320)
Umut Çalıkkasap (150210721)
Dept. of AI and Data Engineering, Istanbul Technical University

December 18, 2025

Abstract

This project addresses the challenge of processing massive e-commerce clickstream data to predict real-time purchase intent using distributed computing paradigms. Utilizing a dataset of **42.4 million interaction events**, we developed two primary models: a **Purchase Intent Predictor** using Random Forest and a **Recommender System** using Alternating Least Squares (ALS). A central component of this study is the benchmarking of vertical scaling (Pandas) versus horizontal scaling (Apache Spark). While Pandas offered lower latency for the 6GB dataset due to in-memory C-array optimizations, Spark demonstrated superior fault tolerance by leveraging RDD-based lineage tracking and disk spilling mechanisms [1], proving its viability for production-grade pipelines.

1 Introduction

The ability to predict user intent in real-time creates a significant competitive advantage for e-retailers. Unlike traditional churn prediction which relies on long-term history, session-based intent prediction requires processing high-velocity clickstream data. This project processes user interaction data collected throughout October 2019 to identify high-value sessions. The primary objectives are:

1. **Big Data Processing:** To handle the "Volume" aspect of Big Data by processing 42M rows efficiently.
2. **Intent Prediction:** To classify session outcomes (Purchase vs. No-Purchase) using behavioral features.
3. **Benchmarking:** To perform a rigorous comparison between single-node (Pandas) and distributed (Spark) frameworks.

This approach aligns with prior studies on real-time clickstream-based intent prediction in large-scale e-commerce systems [2].

2 Exploratory Data Analysis (EDA)

We conducted a comprehensive EDA to uncover behavioral patterns that inform feature engineering.

2.1 Dataset Overview & Brand Dynamics

The dataset contains 42,448,764 events, occupying ∼13.94 GB in raw memory. It involves 3M unique users and 166k products.

- **Funnel Analysis:** The event distribution reveals a highly distinct funnel: View (96.1%), Cart (2.2%), and Purchase (1.7%).
- **Brand Dominance:** Interactions follow a long-tail distribution dominated by major tech manufacturers. **Samsung** (5.28M interactions), **Apple** (4.12M), and **Xiaomi** (3.08M) account for a significant portion of the traffic.

2.2 Temporal Dynamics (The "Night" Shift)

User activity is heavily concentrated in daytime hours (06:00–22:00, ∼81% of events). However, a critical insight emerged regarding conversion efficiency. The purchase rate during night hours (22:00–06:00) is **19.91%** (relative to night events), which is significantly higher than the daytime rate. This implies that nighttime users exhibit more decisive, goal-oriented buying behavior.

2.3 Behavioral Segmentation

We identified two distinct purchasing journeys that guided our feature engineering:

- **Abandoned Carts:** Defined as sessions with repeated views and at least one cart addition but no purchase. These represent "hesitation".
- **Direct Buys:** Surprisingly, **53.5% of purchases** occur without an explicit "cart" event log. This indicates "Buy Now" behavior. Consequently, our model does not rely solely on cart events but infers intent from view patterns (frequency, duration) as well.

3 Predictive Modeling

3.1 Purchase Intent Prediction

The goal is to classify whether an active session will result in a purchase (Binary Classification).

3.1.1 Leakage-Free Feature Engineering

A major risk in session prediction is "Look-Ahead Bias" (Data Leakage). If the model is exposed to user actions occurring *after* a purchase (e.g., viewing receipts), it learns trivial patterns. We implemented a strict **Time-Travel Logic** using Spark Window functions:

1. Identify the timestamp of the *first purchase* in a session.
2. Filter out all events occurring *after* this timestamp.
3. Aggregate pre-purchase events to derive features: `view_count`, `cart_count`, `session_duration`, `avg_price`.

3.1.2 Handling Class Imbalance

With a session conversion rate of only 6.8%, baseline models were biased towards the negative class (Accuracy: 93%, Recall: 0%). We applied **Random Undersampling** [4]:

- **Effectiveness:** While aggressive, this strategy maximized the model's sensitivity to buying signals, raising the F1-Score from 0.51 to 0.84.
- **Minority Class (Buyers):** Retained 100%.
- **Majority Class (Non-Buyers):** Randomly sampled to match the minority count (1:1 ratio).

This reduced the training dataset from 9M to ∼1.2M sessions, significantly improving sensitivity.

3.1.3 Model Results (Spark Distributed)

We trained a Random Forest Classifier (Trees=50, Depth=10). Random Forest [3] was chosen for its robustness against noise and ability to handle non-linear relationships in behavioral data [3].

Table 1: Intent Prediction Metrics (Spark)

Metric	Value
Area Under ROC (AUC)	**0.9276**
F1-Score	**0.8366**
Recall (Weighted)	**0.8385**
Accuracy	0.8385

Feature Importance: `cart_count` (0.42) is the dominant predictor, followed by `session_duration` (0.21). This confirms that "time spent" is a strong proxy for intent in the absence of cart actions.

3.2 Recommender System (ALS)

We implemented Collaborative Filtering using **Alternating Least Squares (ALS)** [5]. Since explicit ratings are absent, we used the "Implicit Feedback" approach [5], where confidence c_{ui} is calculated as $c_{ui} = 1 + \alpha r_{ui}$. The Pandas implementation achieved a Recall@10 of **0.3108**, whereas Spark achieved **0.0999**, highlighting the difficulty of rapid hyperparameter tuning on distributed clusters.

4 Scalability & Performance Analysis

We benchmarked Vertical (Pandas) vs. Horizontal (Spark) Scaling.

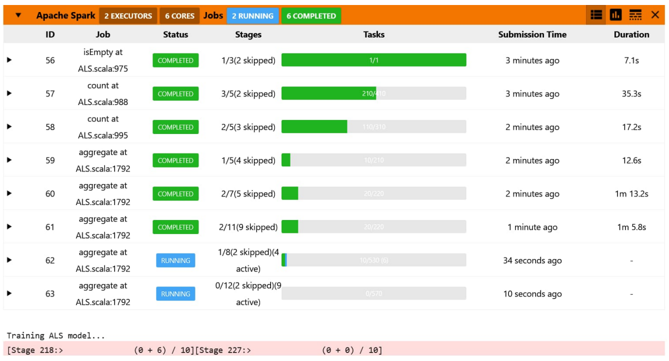
4.1 The "Small Data" Paradox

Contrary to initial expectations, **Pandas outperformed Spark in raw latency** for this 6 GB dataset.

Table 2: Framework Benchmark (Data Size: ∼6 GB)

Task / Metric	Pandas (Single-Node)	Spark (Dist.)
Architecture	In-Memory	Lazy Eval + JVM
Intent Prep.	∼13 min	∼0.5 min (Cluster)
Intent Train	1.84 sec	485.49 sec
Fault Tolerance	**Low (Risk of Crash)**	**High (Disk Spill)**

- **Pandas Efficiency:** Pandas leverages zero-copy memory access and highly optimized C/Cython routines. Since the dataset fit entirely within the 16 GB RAM, there was no swapping overhead.
- **Spark Overhead:** As visualized in Figure 1, distributed computing incurs fixed costs. Even simple aggregations take >10 seconds due to JVM startup, DAG scheduling, and network shuffle.



ID	Job	Status	Stages	Tasks	Submission Time	Duration
56	isEmpty at ALS.scala:975	COMPLETED	1/3(2 skipped)	1/1	3 minutes ago	7.1s
57	count at ALS.scala:988	COMPLETED	3/5(2 skipped)	210/410	3 minutes ago	35.3s
58	count at ALS.scala:995	COMPLETED	2/5(3 skipped)	110/310	2 minutes ago	17.2s
59	aggregate at ALS.scala:1792	COMPLETED	1/5(4 skipped)	10/210	2 minutes ago	12.6s
60	aggregate at ALS.scala:1792	COMPLETED	2/7(5 skipped)	20/220	2 minutes ago	1m 13.2s
61	aggregate at ALS.scala:1792	COMPLETED	2/11(9 skipped)	20/220	1 minute ago	1m 5.8s
62	aggregate at ALS.scala:1792	RUNNING	1/8(2 skipped)(4 active)	10/530 (6)	34 seconds ago	-
63	aggregate at ALS.scala:1792	RUNNING	0/12(2 skipped)(9 active)	0/570	10 seconds ago	-

```
Training ALS model...
[Stage 218:>              (0 + 6) / 10][Stage 227:>              (0 + 0) / 10]
```

Figure 1: Spark UI visualization. The "Job Duration" reflects the overhead of scheduling distributed tasks and managing RDD partitions (35.3s for a simple count).

4.2 Why Spark is Necessary?

Despite the latency gap, Spark demonstrated architectural superiority [1] required for "Big Data":

1. **Robustness via RDD Lineage and Disk Spilling:** Spark's Resilient Distributed Dataset (RDD) abstraction maintains lineage information, allowing lost partitions to be recomputed in case of failure [1]. During the complex "Time-Travel" aggregations, the single-node Spark run utilized disk spilling, demonstrating Spark's ability to handle datasets exceeding available RAM, whereas Pandas would raise a *MemoryError*.
2. **Scalability:** Spark allows parallel training of multiple models (Grid Search) across nodes, linearizing the optimization cost which is exponential on a single node.

5 Conclusion

In this project, we successfully implemented a scalable e-commerce analytics pipeline.

- We developed a **Leakage-Free Purchase Intent Model** (F1: 0.84), successfully identifying high-value users.
- We demonstrated that while Pandas provides low latency for RAM-fitting data, it lacks the **fault tolerance** of Apache Spark.
- The final architecture on Google Cloud Dataproc is ready to scale to terabyte-sized datasets.

References

[1] M. Zaharia et al., "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," *USENIX NSDI*, 2012.

[2] J. Rubin et al., "Real-time clickstream analysis for e-commerce purchase prediction," *IEEE Big Data*, 2019.

[3] L. Breiman, "Random forests," *Machine Learning*, 45(1), 2001.

[4] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[5] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," *ICDM*, 2008.