E-commerce Session Purchase Prediction

Erica Trofimov

Luana Borma Brugger

Matteo Marello

Yardh Vilgot Göran Berglund

Introduction

Context & Objective

- E-commerce growth increased the volume of digital behavioural data (clicks, views, cart actions, time-on-site.
- Analysing session behaviour helps improve User Experience and targeting, supporting data-driven decisions.
- Practical value: If we can predict purchases, we can spot what behaviours lead to buying and optimise the website around them.

Project objective:

- Build a predictive model that classifies whether an online session ends in a purchase using behavioural features observed during that session

Research Question & Data

Research Question:

- **Can we predict whether an online session will end up in a purchase** based on behavioural features during that session (e.g., events volume, product views, cart additions, time spent)?

Dataset:

- Predictors: event logs capturing view/ cart/ purchase actions and interaction intensity and timing.
- Dataset provides the label of purchase vs. non-purchase.
- Scope: Kaggle *Ecommerce Behavior Data from Multi-Category Store*, ~10GB, 7 months of data, study focused on November 2019.

(Sources: McKinsey, 2021; Chaffey & Ellis-Chadwick, 2000)

Data Storage

Challenge: *Avoid massive downloads on disk while still ensuring accessibility*

STEP 1: initial situation

Location: kaggle

Size: 10 GB

Format: CSV

Accessibility: low

Initially the dataset is stored on kaggle.com, it's in .csv format and is roughly **10GB**. It's not possible to **access** it directly from our local machine **without downloading** it on the disk.

STEP 2: conversion

Location: kaggle

Size: 2.5 GB

Format: parquet

Accessibility: low

Conversion into parquet format **reduces size** by 75%, the new dataset contains the same information and it is **optimised** for analytical operations, being a column-based storage file.

STEP 3: upload to GSC

Location:

Size: 2.5 GB

Format: parquet

Accessibility: high

Uploading the dataset on Google Storage Cloud (GSC) **increases accessibility** because it allows to access the data from local machines **avoiding massive download**.

Data Cleaning

Challenge: *Huge dataset that cannot be loaded into R as a standard data frame to perform operations.*

Solution

Addressing the challenge

- Arrow for lazy evaluation
- Process without loading raw data into memory

Goal: Push collection as late as possible

Out-of-memory Operations

- Performing" heavy" operations to create a compact intermediate dataset before collection

Goal: Grouping, counting events, average price, session start and end time before collecting

In-memory Operations

- Some operations are not supported by Arrow's lazy backend
- Time parsing and date logic to calculate time-related features (e.g. *is_weekend)* needs to be calculated in-memory

Goal: Drop all unnecessary columns and collect

Scalability

- If dataset grows, we again run into the same issue since we need to collect.

Goal: Leverage BigQuery to perform all time transformations

Trade-off: Orchestration complexly vs (unlimited) scalability

2.5 GB (GSC) → **~ 500 MB(Estimated object size)**

Data Analysis

Device

- **16 GB** RAM
- **CPU**-only execution (no GPU, no paralleliz ation)

EDA

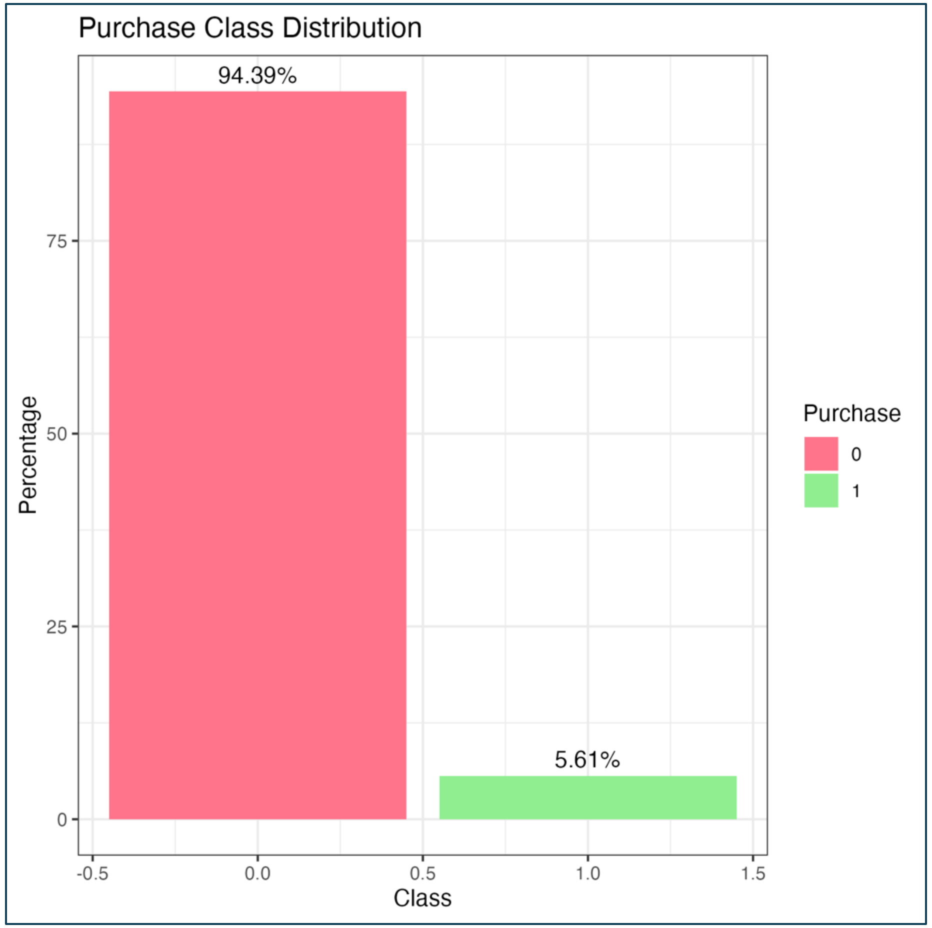
- ‘**Big N**’ Complexity
- Summary of the features
- Target binary variable: n_purchase

Visualizations

- Correlation Matrix → Multicollinearity
- **Barplot** of purchases → high imbalance (5,51% purchases)

Big Data Considerations

- Remove objects not needed
- Plot with a sample if the dataset is huge



Predictive Models

Challenge: Computing models locally without reaching the vector memory limit

Ridge Logistic Regression

Key Aspects
Used **biglasso**: memory-efficient package for Lasso and Ridge
Converted train data to **filebacked.big.matrix** to reduce memory consumption during training.
Memory usage and **training time** recorded to evaluate efficiency
Regular cleanup with **rm()** and **gc()**

Challenging Computations:
Fitting the iterative k-fold **Cross-validation** for tuning parameter λ in the local memory

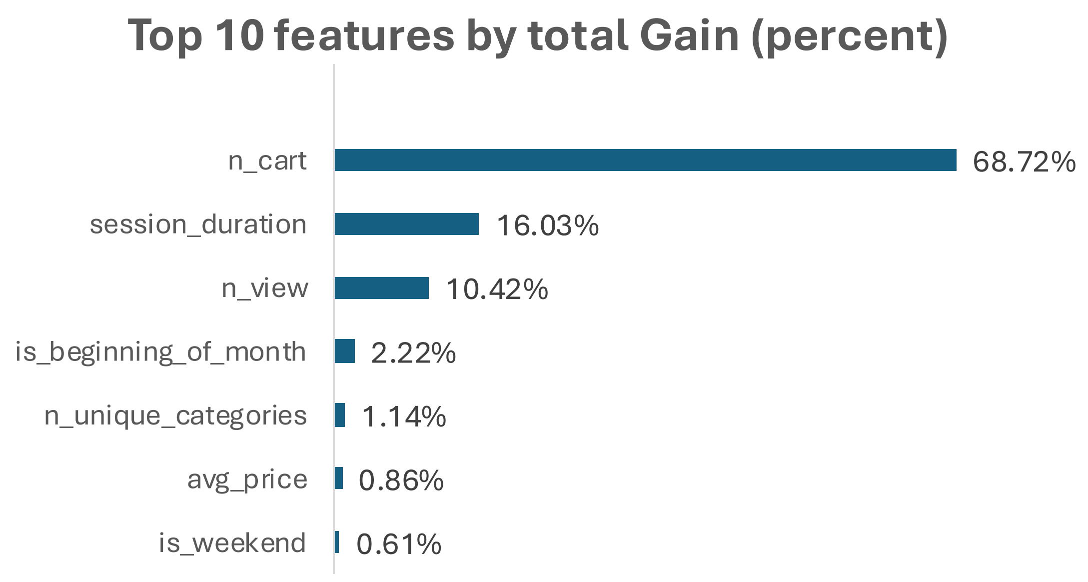
XGBoost

Key Aspects
Fast C++ implementation in R using **xgboost** and **DALEX** packages
Handled unbalanced data via scale_pos_weight
prevented overcomputation by **early stopping** after 20 rounds without AUC improvement

Scalability:
- Might need **cloud computing** (e.g. Spark) for Cross validation and run more optimal models on bigger datasets
- Storing the models will occupy more memory
- Better to **plot** on smaller **samples**

BigLasso documentation: Zeng Y, Breheny P (2021). "The biglasso Package: A Memory- and Computation-Efficient Solver for Lasso Model Fitting with Big Data in R.'

Results

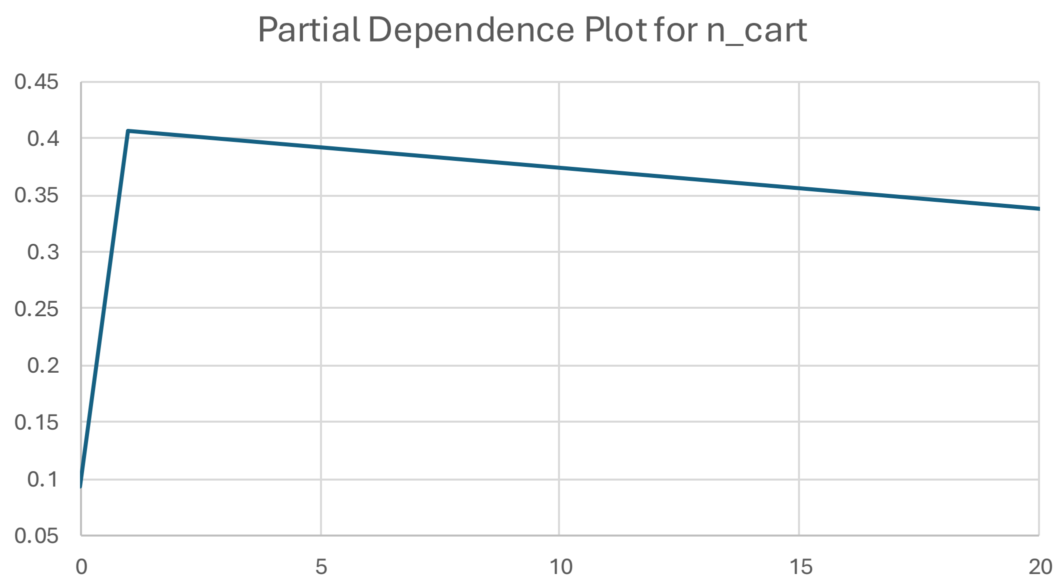


Key drivers (XGBoost)

- Cart activity dominates (~69% of Gain).
- In-session intent signals explain most of the predictive power.

Behavioural pattern:

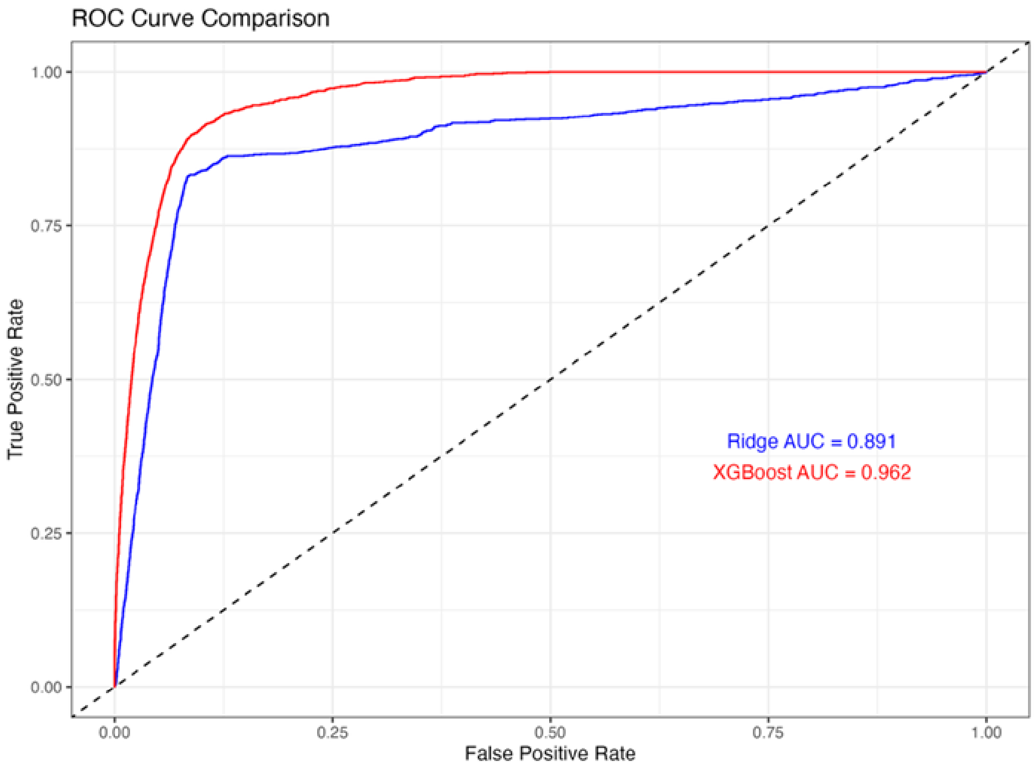
- Purchase probability jumps from 0 to 1 cart event, then declines gradually with more cart events.
- One decisive cart action separates many buyers; multiple cart actions may reflect indecision.



Results

Model performance

- XGBoost AUC ~ 0.96
- Ridge AUC ~ 0.89



Confusion-matrix trade-off

- Ridge: low FP but misses most purchases → very low purchase recall / F1
- XGBoost: captures most purchases, but with more FP → higher recall / F1, better for identifying likely buyers.

Ridge Regression Confusion Matrix

	0	1
0	5161789	279007
1	38915	30948

XGBoost Confusion Matrix

	0	1
0	4714989	27931
1	485716	282024

Thank you!