MACHINE LEARNING · Application

E-Commerce Purchase Behavior Prediction

Predicting user purchase intent with XGBoost & behavioral sequence features

Henry Wong · April 2026

Agenda

01

Problem Statement

Research problem and economic value.

02

Data Processing

~700K records · 5 raw fields · 4 behavior types.

03

Feature Engineering

3 factor groups: Behavioral · Category · Temporal.

04

Modeling & Tuning

XGBoost + scale_pos_weight + Optuna Bayesian tuning.

05

Results & Insights

AUC-ROC 0.9994 · AUC-PR 0.9760.

06

Limitations & Future Improvement

Lessons learned and conclusion.

Problem Statement

The Problem Statement

Problem

Predict purchase intent by engineering behavioral features, such as frequency, recency, and engagement intensity that is derived from raw user interaction data, including behavior types, item categories, and timestamps.

Data set

Real-world user behavior data from Taobao (2014), processed for anonymization

Economic Value

Personalized Recommendations

Serve the right product at the right moment, boosting conversion rates and revenue.

Behavioral Dynamics Mapping

Identifying which sequences of views and adds-to-cart signify a high-probability buy.

Inventory & Supply Chain

Anticipate demand spikes before they occur, reducing stock outs and overstock costs.

Dataset Overview

~700K
Raw behavior records

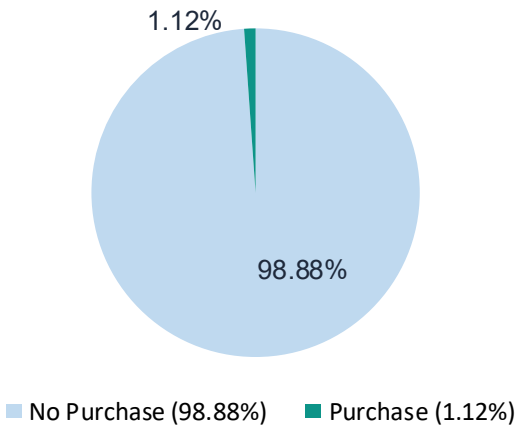
599,116
(User, Item) pairs modeled

6697
purchase events among pairs

16
Features entering model

Field Name	Description
user_id	Unique identifier for each user
item_id	Unique identifier for each product
behavior_type	1=View, 2=Favorite, 3=Add to Cart, 4=Purchase
item_category	Category ID of the product
time	Timestamp of the action (Format: YYYY-MM-DD HH)

Class Distribution



Data Processing

Goal: Analyze the complete behavioral history of a specific user for a specific item

Step1: **Group by (User, Item)**

Step2: **Sequence all actions into a time-ordered string**

Example behavior string: "1 1 3 1 | 4" → [browse, browse, add-to-cart, browse] | purchase

1	1	3	1	|	4
Browse	Browse	Add to Cart	Browse	CUT HERE	Purchase (excluded)

Step3: **Sample labeling**

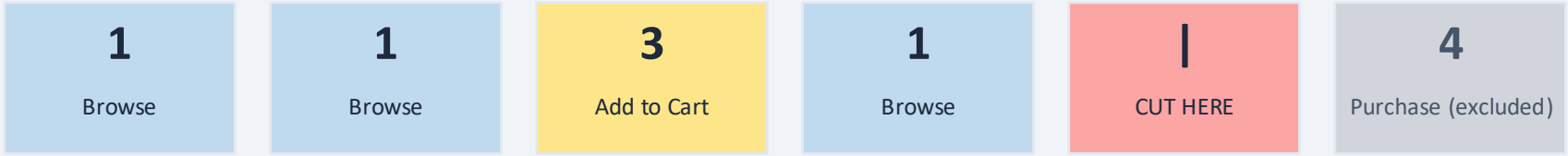
(User, Item)
- contains a “Purchase” action → target variable as 1
- doesn’t contain “Purchase” → target variable as 0

Step4: **Generate features for model**

- behavior_type → **Behavioral Sequence Features**
- item_category → **Category Level Features**
- time → **Temporal Features**

Factor 1 — Behavioral Sequence Features

Example behavior string: "1 1 3 1 | 4" → [browse, browse, add-to-cart, browse] | purchase



`seq_len`

Sequence length before purchase.
Longer = more engaged user.

`count_browse`

Browse count (type 1).
Reflects awareness & exploration depth.

`count_fav`

Favorite count (type 2).
Saved items → elevated interest.

`count_cart`

Add-to-cart count (type 3).
Strongest single purchase signal.

`last_action`

Final action type in prefix.

`time_diff_hours`

Hours from first browse to first add-to-cart.
No path → -1.

Factor 2 — Category-Level Features

2.1 cat_sample_count

Total behavioral records in this product category.
Large categories are more statistically reliable — a signal of market depth and maturity.

2.1 cat_sample_ratio

Category share of total dataset activity.
Helps the model distinguish niche vs. mainstream categories, which have very different conversion dynamics.

2.2 (Target Encoding) cat_purchase_rate

Historical purchase rate for this category.
Captures category-level willingness-to-buy.
Computed only on training fold, then mapped to val/test.

2.2 cat_avg_time_buy

Average hours from first action to purchase in this category.
Fast categories differ from considered purchases.

2.3 z_score

User's activity level in this category relative to all other users, z-normalized.
Positive = power user; Negative = casual visitor.

Factor 3 — Temporal Features

`click_density_1h`

Based on the last action, action count in the last 1 hour.

`click_density_6h`

Action count over the last 6 hours.

`click_density_24h`

Action count over the last 24 hours.

`last_active_hour`

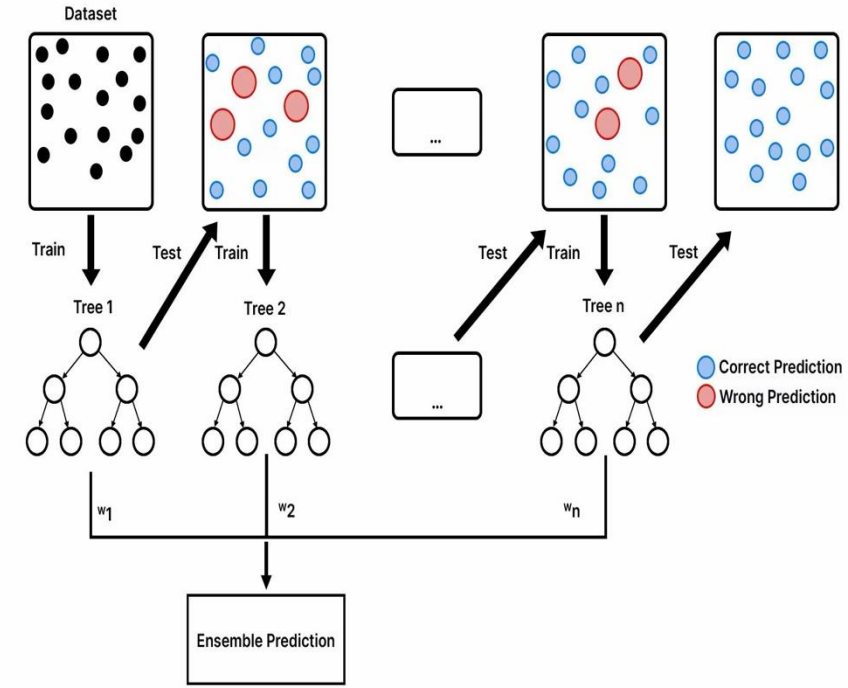
The hour of the day (0–23) when the last interaction occurred.

`time_span_hours`

The total duration of the user's behavioral sequence.
A longer span suggests deliberative shopping.

Model: XGBoost

How XGBoost works:



Why XGBoost?

- **Mixed Feature Types**
  Can handle different types of data such as counts, ratios, z-scores, binary flags, and so on, so no standardization needed
- **Built-in Imbalance Weight**
  Its parameter “scale_pos_weight” add weights to the minority data
- **Good SHAP Compatibility**
  Because XGBoost is a tree model, it is inherently capable of working seamlessly with an explanation tool like SHAP so that I can know both
- **L1 + L2 Regularization**
  reg_alpha and reg_lambda prevent overfitting on the small positive class.
- **High Speed**
  It trains very quickly. 479,000 pieces of data and 17 features can be completed in just 2 minutes.

Action before training

1. Division of data

Before training the model, we must divide the data into three parts, each to be used for a different purpose:

Data set	Proportion
Training set	72%
Validation set	8%
Test set	20%

2. Handle sample imbalance

1.12% of our samples are positive

98.88% are negative

Solution: We set **scale_pos_weight** = number of negative samples / number of positive samples ≈ 88.46.
Making a wrong prediction on one buyer is as costly as making 88 wrong predictions on regular browsers, forcing the model to pay close attention to the rare positive class.

3.Baseline Configuration

Parameter	Value
objective	binary:logistic
eval_metric	aucpr
n_estimators	500
scale_pos_weight	88.46
max_depth	6
learning_rate	0.05
subsample	0.8
colsample_bytree	0.8
early_stopping	30 rounds

Results

0.9994

AUC-ROC

0.9760

AUC-PR

0.9639

F1 Score

Confusion Matrix (Test Set — 119,824 samples)

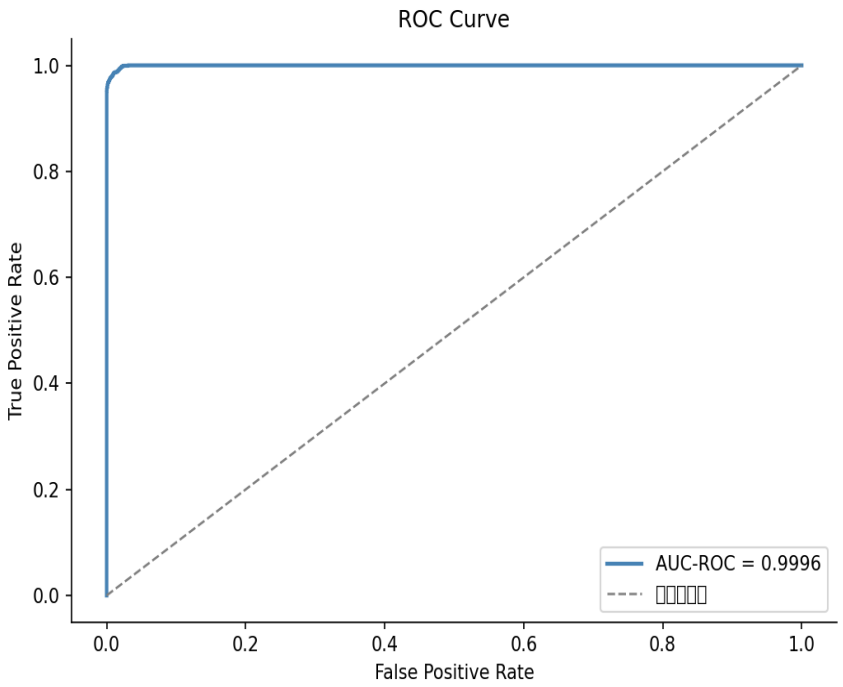
	Predicted: No Buy	Predicted: Buy
Actual: No Buy	118,475 ✓ TN	10 FP
Actual: Buy	84 FN	1,255 ✓ TP

Key Insight

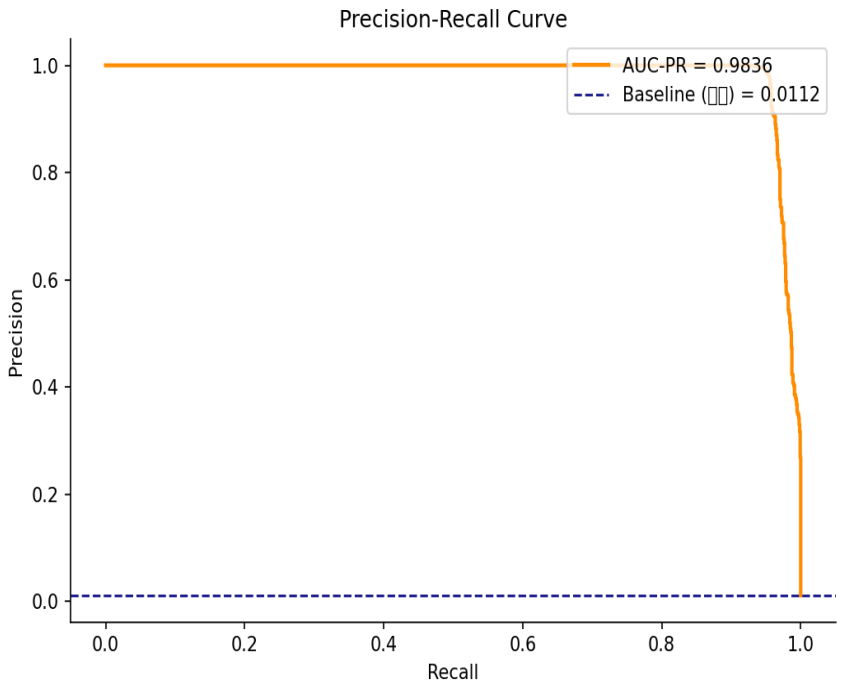
Only 10 false alarms out of 118,485 non-buyers (Precision: 99.2%). Only 84 missed purchases out of 1,339 buyers (Recall: 93.7%). The model is both precise and reliable.

Evaluation Curves

ROC Curve (AUC = 0.9994)



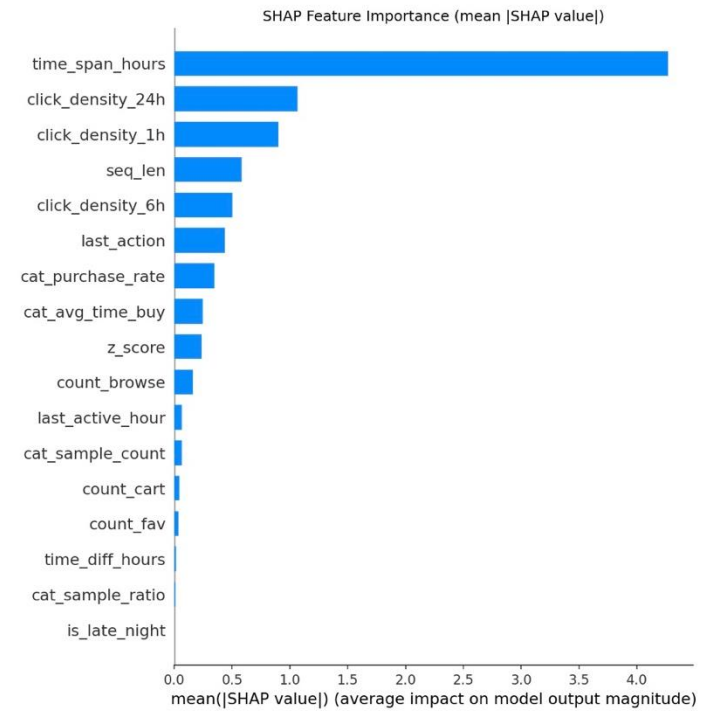
Precision-Recall Curve (AUC-PR = 0.9760)



ROC: near-perfect separation across all thresholds. PR curve: a random classifier on this data scores AUC-PR ≈ 0.011; our model scores 0.976 — an 89× improvement.

SHAP Feature Importance

SHAP Feature Importance



Top1: time span hours

The time interval between the user's first visit and their last interaction is the key indicator for predicting whether they will make a purchase.

Top2&3: click density within a short time

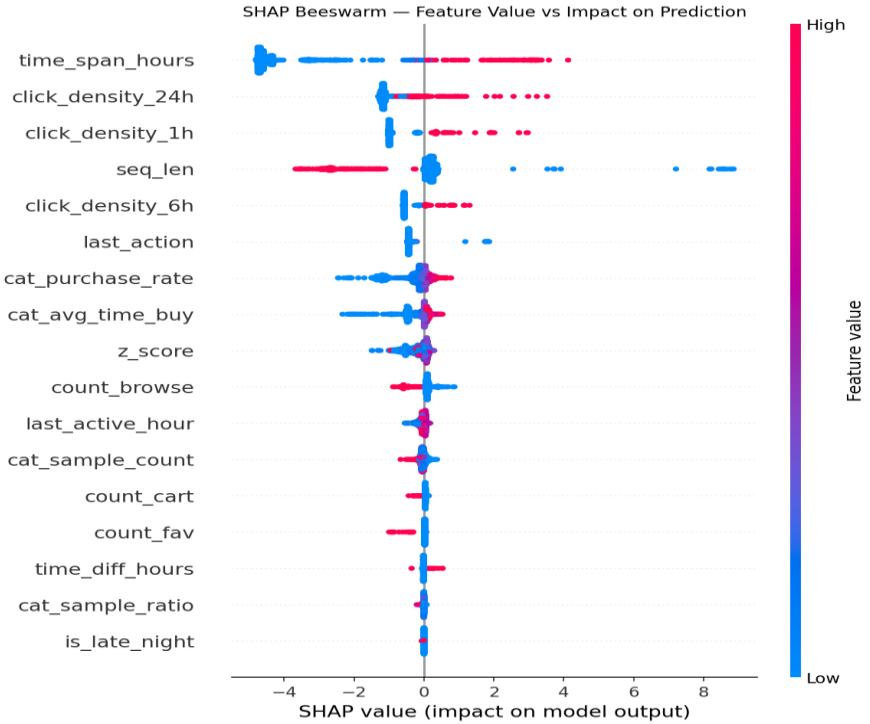
The click, browse frequency of users within 1 hour and 24 hours are important signals. This indicates that high-frequency activity within a short period of time is more indicative of purchase intentions than scattered browsing.

Not important: click density within a short time

Number of things added to cart, number of favorites, whether it is late at night, etc., have very small contribution to the model's prediction.

SHAP Beeswarm

SHAP Beeswarm (Global Feature Impact)



1. Time span hours

The longer the time between the user's first visit and their last interaction, the more likely the model is to predict that they will make a purchase

2. Clicking density

The higher the user's click browsing frequency within a short period of time, the higher the predicted probability of purchase

3. Length of the sequence

The shorter the sequence of the user's actions (such as directly placing an order without any additional browsing), the more likely the model considers that the user will make a purchase

Limitations

1. Sample size

The original dataset was randomly sampled to 1/16 of its size due to computational resource constraints, which may impact the statistical robustness of rare categories.

2. Target Encoding

A more rigorous approach would be to calculate target encoding per fold during K-fold cross-validation, therefore further eliminating potential bias.

3. Temporal Data Handling

In this project, all behavior data was flattened for analysis, without strictly separating the training and prediction periods by time windows. In a production environment, it is critical to ensure that all training data predates the prediction timestamp.

4. Feature Multicollinearity

Potential multicollinearity exists between the features about clicking during late night or the ones about the latest active hour. While XGBoost is robust to multicollinearity, these features would need to be cleaned if the model were migrated to a linear framework.

Future Improvement

Incorporate Sequence Models:
Utilize LSTM or Transformer architectures to directly model the behavior sequences, capturing the sequential dependencies between user actions

Add User-level Features:
Integrate broader user characteristics such as overall purchase history, number of active days, and account tenure to better characterize the user's overall consumption capacity

Online Learning:
User behavior evolves in real-time. Deploy incremental learning models to update prediction scores in real-time based on new behaviors

Conclusion

Strong Predictive Performance

AUC-ROC 0.9994 & AUC-PR 0.9760 — behavioral sequences carry rich purchase signals even at a 1.12% positive rate.

Imbalance Requires Careful Design

scale_pos_weight, threshold optimization, and AUC-PR (not accuracy) are essential. These lessons transfer directly to credit risk and fraud detection.

Feature Engineering is the Key Lever

The 3-factor system (behavioral, category, temporal) substantially outperforms raw behavior counts alone. Target encoding and z-scores add market context.

Explainability Builds Trust

SHAP confirms count_cart dominates — consistent with intuition. XGBoost + SHAP bridges ML performance and economic interpretability.

Thank you · Questions welcome