E-commerce Data Prediction

*Machine Learning Final Project Report

1st Yifei Shang
Khoury College of Computer Sciences
Northeastern University
Seattle, USA
shang.yif@northeastern.edu

***Abstract*—This report presents the motivation, implementation, and analysis of the final project for CS 6140 Machine Learning. The goal of this project is to explore how machine learning classifiers can be applied to predict user purchase behavior from e-commerce clickstream data. We identify the problem of class imbalance—where purchase events are rare—and examine how different classifiers respond to this challenge. The models take aggregated user activity features (such as view count, cart count, and browsing diversity) as input and output the likelihood of a purchase.**

The report provides an overview of our problem formulation, an in-depth explanation of our implementation and tuning of four classifier models—Logistic Regression (SGD), Random Forest, XGBoost, and Naïve Bayes—and a discussion on their performance. The results are evaluated using standard classification metrics with an emphasis on detecting the minority class. We conclude with observations on the strengths and limitations of each approach, and propose ideas for future exploration.

***Index Terms*—Keywords: Machine Learning, Classification, Recommandation system, E-commerce**

I. Introduction

Our paper aims to implement classification method on a 2.7GB Kaggel dataset 'eCommerce behavior data from multi category' in order to predict if a user will make a purchase based on earlier activity in a session or overall.

II. Data

Our dataset encompasses user interactions on an e-commerce platform, comprising the following attributes: event-time, event-type, brand, price, IDs. We decide to analyze the dataset by comparing event-type distribution, brand distribution based on event-type, price distribution based on brand.

Apply boxplot to the dataset (Fig 3). The boxplot reveals that brands like *Apple* and *Sony* have higher median prices and wider interquartile ranges, indicating a broader and more premium product lineup, whereas brands such as *Huawei*, *Oppo*, and *Xiaomi* exhibit lower and more tightly clustered price distributions, suggesting a consistent focus on budget to mid-range offerings.

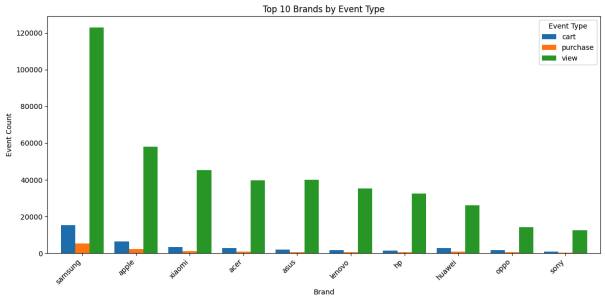
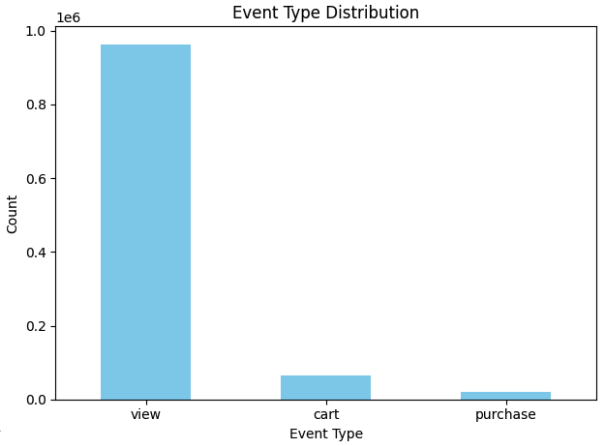


Fig. 1. Brand Type Distribution by event type



0.70.7

Fig. 2. Event Type Distribution

Implementation

When choosing a machine learning classifier for a large dataset, it's important to balance accuracy, interpretability, and computational efficiency. The full implementation and source code are publicly available in the GitHub repository at: `https://github.com/ifffShang/CS6140`. The repository includes detailed documentation, preprocessing steps, and configurable options for running the models on sampled or full datasets.

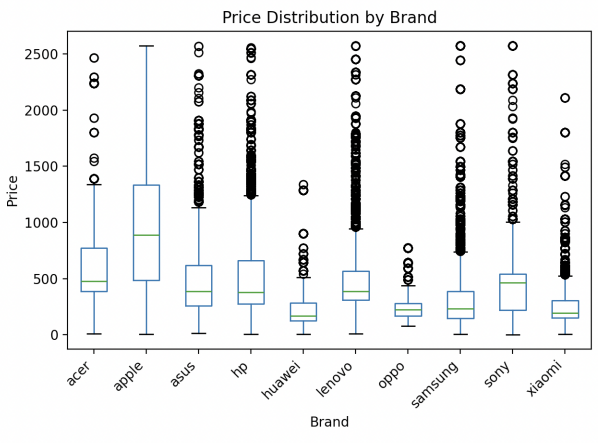


Fig. 3. Price distribution across top 10 brands.

A. Logistic Regression with SGDClassifier

Logistic Regression is often a strong baseline due to its simplicity, speed, and interpretability. It performs well when the relationship between features and the target is linear, and can scale to large datasets using stochastic gradient descent (SGD). Stochastic Gradient Descent (SGD) was chosen due to its scalability and efficiency on large, imbalanced e-commerce behavior datasets. Since we are working with a clickstream dataset where each user has multiple interactions (views, carts, purchases), SGD processes one sample (or a small batch) at a time rather than loading the full dataset into memory. It converges much faster for large and sparse datasets like clickstreams.

B. Random Forest

In this project, Random Forest was employed as a classification model due to its robustness, scalability, and ability to handle nonlinear relationships in user behavior data. Unlike logistic regression, which assumes a linear relationship between features and the target. Instead of using RandomForestClassifier from scikit-learn, this implementation builds the ensemble manually by creating multiple DecisionTreeClassifier models, each trained on a randomly sampled subset of the training data.

For each tree in the forest, a bootstrap sample is drawn from the training data with replacement. A random subset of features is selected for each tree (e.g., square root of total features), ensuring diversity among the individual models. Each tree is trained on this sampled data using scikit-learn's DecisionTreeClassifier with a specified maximum depth and splitting criterion (Gini impurity). During prediction, each tree produces its own prediction, and the final class label is determined by majority voting across all trees.

C. XGBoost

Given the scale and complexity of the e-commerce dataset used in this study, XGBoost presents itself as a highly effective modeling choice. The dataset consists of millions of user interaction events, with features that are often sparse and imbalanced, such as the relatively low proportion of purchase events compared to views or cart additions. XGBoost is specifically optimized for such large-scale, structured data. It supports fast and scalable training through parallel and GPU computation, and is designed to handle missing values natively, which is advantageous when dealing with real-world clickstream data. Moreover, its gradient boosting framework allows it to sequentially correct errors from previous trees, often resulting in higher predictive accuracy than traditional models like logistic regression or random forests. Overall, XGBoost combines computational efficiency with predictive power, making it a strong candidate for this task.

D. Naïve Bayes Classifier

The Gaussian Naïve Bayes Classifier used in this project was adapted to support the dataset and integrate with ensemble methods. The model begins by splitting the training data based on the target labels and computes the mean and standard deviation for each feature within each class. During prediction, it assumes that feature values follow a Gaussian (normal) distribution and uses these class-specific statistics to calculate the likelihood of each feature value given each class. These likelihoods are combined to compute the overall posterior probability for each class under the assumption of feature independence. The final prediction is made by selecting the class with the highest overall probability. Despite the simplicity of this model, its probabilistic foundation and efficiency make it useful for handling continuous data and serving as a baseline in ensemble learning frameworks.

III. Result

Logistic Regression is often a strong baseline due to its simplicity, speed, and interpretability. It performs well when the relationship between features and the target is linear, and can scale to large datasets using stochastic gradient descent (SGD). Stochastic Gradient Descent (SGD) was chosen due to its scalability and efficiency on large, imbalanced e-commerce behavior datasets. Since we are working with a clickstream dataset where each user has multiple interactions (views, carts, purchases), SGD processes one sample (or a small batch) at a time rather than loading the full dataset into memory. It converges much faster for large and sparse datasets like clickstreams.

A. Logistic Regression with SGDClassifier

TABLE I
OGISTIC REGRESSION WITH SGDCLASSIFIER

Class	Precision	Recall	F1-Score	Support
0 (No Purchase)	0.9303	0.7792	0.8480	41283
1 (Purchase)	0.1870	0.4652	0.2668	4508
Accuracy		0.7482		
Macro Avg	0.5586	0.6222	0.5574	45791
Weighted Avg	0.8571	0.7482	0.7908	45791

The logistic regression model trained using stochastic gradient descent achieved strong performance on the majority

class (non-purchase), with a precision of 93.03%. F1-score of 0.2668 for predicting purchases means our model strikes a weak balance between how many buyers it finds (recall = 46.5%), and how accurate it is when it predicts someone will buy (precision = 18.7%). Because precision is low (18.7%), the model makes many false positive predictions (predicts a user will buy when they don't). Even though recall is decent (46.5%), the low precision pulls the F1-score down.

B. Random Forest

As a result, our scratched custom Random Forest is predicting class 0 (No Purchase) for every sample. The reason for it is that the dataset likely has a small number of purchase cases (4508 vs. 41283). That's roughly 1:9 ratio. Many trees never see any class 1 due to bootstrapping — so they learn only class 0. So I updated the bootstrapsample method to force balance sample per tree. I fixed model n-estimators to 50.

The confusion matrix (Fig. 4) indicates that the Random Forest classifier was able to identify a substantial portion of actual purchasers, correctly predicting 3,504 out of 4,508 positive cases. This corresponds to a relatively strong recall for the minority class. However, the model also produced 17,644 false positives, where non-purchasing users were incorrectly predicted as purchasers. While the true negative count (23,639) is high, the large number of false positives indicates that the model sacrifices precision in favor of recall. This behavior is common in imbalanced classification problems, where the model prioritizes detecting rare positive cases. Although this strategy may be acceptable in scenarios where identifying potential buyers is more important than avoiding false alarms, it may not be suitable when marketing resources are limited. Overall, the model demonstrates reasonable capability in recognizing potential buyers but still requires further refinement to improve precision without significantly harming recall.

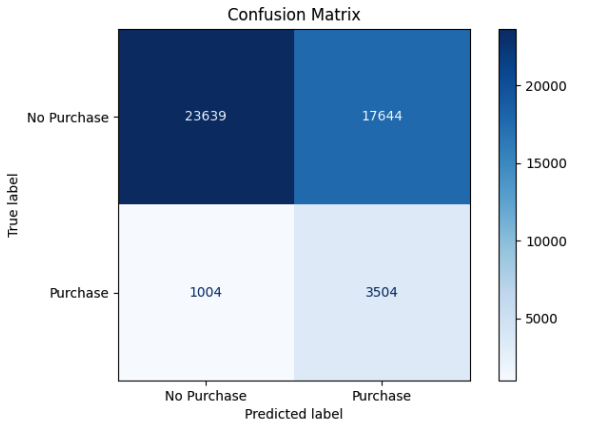


Fig. 4. RandomForest ConfusionMatrix

According to TABLE2, the Random Forest classifier achieved an overall accuracy of 59.28%. The model demonstrated strong performance on the majority class (non-purchase), with a precision of 95.93% and an F1-score of 71.71%. However, for the minority class (purchase), the model achieved an F1-score of 27.32%. This suggests that while the model is effective at identifying actual purchasers, it tends to misclassify many non-buyers as buyers, leading to a high number of false positives. The macro-averaged F1-score of 49.51% reflects this imbalance in class performance, while the weighted average F1-score of 67.34% is skewed by the dominance of the majority class. Overall, the Random Forest model shows potential in capturing actual purchase behavior but would benefit from further tuning to reduce false positives and improve precision for actionable use.

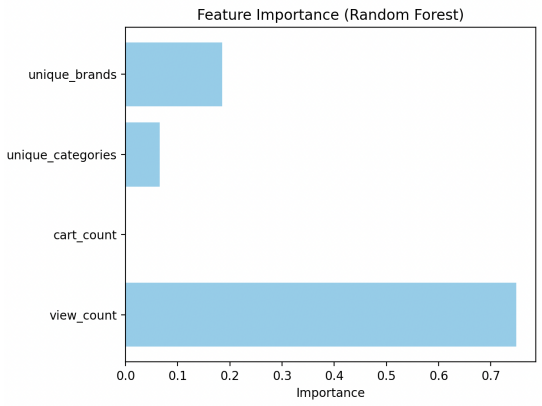


Fig. 5. Feature Importance

TABLE II
RANDOM FOREST

Class	Precision	Recall	F1-Score	Support
0 (No Purchase)	0.9593	0.5726	0.7171	41283
1 (Purchase)	0.1657	0.7773	0.2732	4508
Accuracy		0.5928		
Macro Avg	0.5625	0.6749	0.4951	45791
Weighted Avg	0.8811	0.5928	0.6734	45791

C. XGBoost

I implemented XGBoost from scratch to deepen my understanding of gradient boosting. Initially, the model predicted only class 0 (non-purchase), failing to detect any positive (purchase) cases despite their presence in the test set. This is a common issue when training on imbalanced datasets, where the model minimizes loss by favoring the majority class (in this case, over 90% were non-purchases).

To address this, I introduced a `scale_pos_weight` factor into the gradient calculation and increased both the number of estimators and the tree depth. However, this led to overcompensation, and the model began predicting only class 1 for all users. To resolve this, I reduced the influence of `scale_pos_weight` and limited the number of estimators, which improved balance and generalization.

```
model = SimpleXGBoost(n_estimators=30, learning_rate
    =0.1, max_depth=3)
model.fit(X_train, y_train)
```

TABLE III
CLASSIFICATION REPORT FOR CUSTOM XGBOOST MODEL

Class	Precision	Recall	F1-score	Support
0 (No Purchase)	0.9709	0.3505	0.5151	41283
1 (Purchase)	0.1319	0.9037	0.2302	4508
Accuracy		0.4050		
Macro Avg	0.5514	0.6271	0.3726	45791
Weighted Avg	0.8883	0.4050	0.4870	45791

1) ROC Curve Analysis: To evaluate the performance of the custom XGBoost model, we plotted the Receiver Operating Characteristic (ROC) curve using the predicted probabilities on the test set. The resulting curve, shown in Figure 7, demonstrates the model's ability to distinguish between purchase and non-purchase behavior.

The Area Under the Curve (AUC) was calculated to be 0.71, indicating that the model performs significantly better than a random classifier (AUC = 0.5). This suggests that the model is capable of learning meaningful patterns from the imbalanced data and has moderate discriminative power.

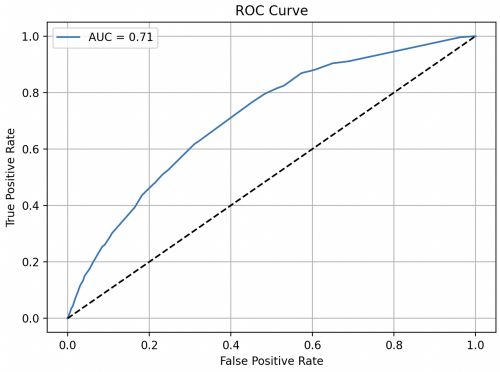


Fig. 6. ROC Curve for Custom XGBoost Model with AUC = 0.71

2) Precision-Recall Curve Analysis: In addition to the ROC curve, we plotted the Precision-Recall (PR) curve to better evaluate performance on the imbalanced purchase prediction task. The curve (Fig. 7) shows that precision is initially very high when recall is near zero, meaning the model can make a few highly confident purchase predictions correctly. However, as recall increases, precision gradually decreases due to the rising number of false positives.

This trade-off is expected in imbalanced datasets and highlights the importance of threshold selection depending on the specific application (e.g., precision-prioritized recommendation vs. recall-prioritized fraud detection).

D. Naïve Bayes Classifier

The Naïve Bayes classifier achieved an overall accuracy of 88.06%, driven largely by excellent performance on the majority class (no purchase). However, performance on the minority class (purchase) was notably weak, with a recall of only 7.9% and an F1-score of 0.1152. This highlights the model's tendency to favor the dominant class, a typical challenge in imbalanced classification scenarios. While the model is efficient and fast, its predictive power for rare events like purchases is limited without further balancing strategies such as resampling or feature scaling.

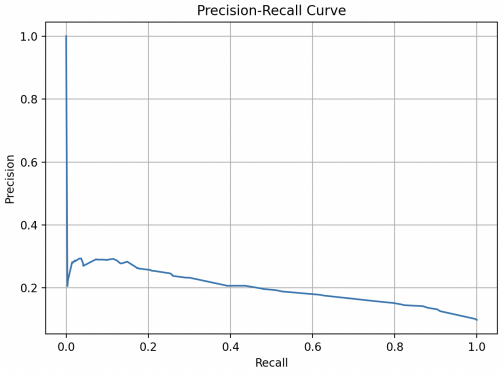


Fig. 7. ROC Curve for Custom XGBoost Model with AUC = 0.71

TABLE IV
CLASSIFICATION REPORT FOR NAÏVE BAYES CLASSIFIER

Class	Precision	Recall	F1-score	Support
0 (No Purchase)	0.9059	0.9681	0.9360	41283
1 (Purchase)	0.2129	0.0790	0.1152	4508
Accuracy		0.8806		
Macro Avg	0.5594	0.5235	0.5256	45791
Weighted Avg	0.8377	0.8806	0.8552	45791

CONCLUSION

IV. CONCLUSION

In this study, we explored four machine learning models—Logistic Regression (with SGD), Random Forest, XGBoost, and Gaussian Naïve Bayes—to predict purchase behavior from a large, imbalanced e-commerce dataset.

Logistic Regression served as a strong baseline with high overall accuracy (74.82%) and reasonable recall for purchases (46.52%), but it suffered from low precision. Random Forest improved recall to 77.73% while maintaining a more balanced F1-score, though at the cost of many false positives. XGBoost, after tuning with `scale_pos_weight`, achieved the highest recall (90.37%) on the minority class and an AUC of 0.71, showing strong discriminative ability but lower overall accuracy (40.5%) due to its overprediction of the positive class. Naïve Bayes was computationally efficient and accurate on the majority class, but failed to detect purchases effectively, with only 7.9% recall.

These results highlight the challenge of predicting rare events in imbalanced datasets. While XGBoost and Random Forest showed superior capability in identifying potential buyers, they require careful threshold and parameter tuning to avoid overwhelming false positives. Future work may explore ensemble techniques, cost-sensitive learning, or hybrid architectures to further optimize both recall and precision for real-world deployment in targeted marketing or recommendation systems.

REFERENCES

[1] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer, 2009.

[2] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[3] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016, pp. 785–794.

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[5] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[6] M. Kechinov, "E-Commerce Behavior Data from Multi-Category Store," Kaggle, 2019. [Online]. Available: https://www.kaggle.com/datasets/mkechinov/ecommerce-behavior-data-from-multi-category-store

[7] H. Zhang, "The Optimality of Naive Bayes," in *Proc. 17th International FLAIRS Conference*, 2004, pp. 562–567.