Prediction of Online Shoppers' Purchasing Intention

Lisa Lyu (zl879), Yuanqi Sun (ys2274), Wenxuan Zhang (wz443)

GitHub Link: https://github.com/YuanqiSun0102/ORIE5741Project.git

1 Introduction

The rapid expansion of e-commerce has a significant impact on marketing strategies. Over the past years, the explosion of online shopping formats has changed the interaction between sales platforms and customers thus imposing more complicated strategies for marketing. This shift emphasizes the important role of marketing analysis teams to accurately perform data analysis and data-driven decisions. By tracking user behaviors based on their attributes, the company can provide personal recommendations and time the promotion to improve user experience, engagement, and user purchase intention rate, so that the overall revenue can be increased as well. Hence, we would determine whether the customer will finalize his/her transaction or not based on a series of behaviors, which is a binary classification problem.

The remaining of the report is structured as follows: section 2 will introduce the dataset and exploratory analysis; Section 3 will provide a detailed description of modeling techniques, including methods for dealing with imbalanced data and 4 models; Section 4 will discuss the final conclusion. Moreover, in this project, Lisa contributed on Data preprocessing and EDA, Yuanqi implemented and analyzed different imbalanced data handling methods, and Wenxuan built and analyzed further classifiers based on the chosen data handling methods.

2 Dataset

2.1 Dataset Introduction and Preprocessing

The dataset we utilized is from the UCI Machine Learning Repository, containing 12330 sessions, where each session represents a different customer in a 1-year period. There are 18 columns in the dataset, including the binary output column 'Revenue', 10 numerical features, and 7 categorical variables. In this data, 10422 customers are in the 0 class, which means they didn't finalize their transaction, showing the imbalanced distribution of the data. Table 1 shows several features and their corresponding meaning, the full table can be found in the appendix. Since the data contains user behavior information, it is conducted based on Google Analytics.

Feature Name	Categorical or Numerical	Feature Description
Product related	Numerical	Number of pages viewed related to the product
Product related duration	Numerical	Amount of time (seconds) spent by the customer in viewing the product-related pages
Exit Rate	Numerical	Average exit rate for the user while viewing the page
Special Day	Numerical	Closeness of customer visiting time and special days

Visitor Type	Categorical	If a visitor is new, returning, or other
Weekend	Categorical	If the visit date is on the weekend
Brower	Categorical	Type of browser each customer used

Table 1: Features in the dataset and corresponding introduction

The dataset contains no missing values However, there are 125 duplicated values. The duplicates may be due to data entry errors, where a single session is accidentally entered several times, a data combining process, where muti-sources data does not consider the repetition session, or the user's own behavior, where he/she clicks the website multiple times and all clicks are detected by the tracking sensor. Thus, we decided to delete these values but kept the first occurrence of the record, ensuring that each session would only appear once. The outliers were also checked in the dataset. We mainly focused on columns with the name "duration" since these columns account for time values in seconds. It shows that even though there are big values, such as 63973 in the "product-related duration" column, since the unit is in seconds, and the session records a 1-year period of data, we consider the large values are still reasonable in the given context. Hence, we made no changes on the comparable larger values.

Due to the presence of categorical values, it is crucial for us to perform the one-hot encoding. This technique helps us to create additional columns represented by boolean values, which is useful when we try to build machine learning algorithms. The columns that need to be encoded are "Visitor type", "Browser", "Region", "Operating System", "Traffic Type", and "Weekend". We also used a map function to convert the "Month" column (original string type) into integers.

2.2 Exploratory Data Analysis

To further understand the distribution of the data and characteristics of each feature, several histograms are plotted. Figure 1 shows the histogram of all features.

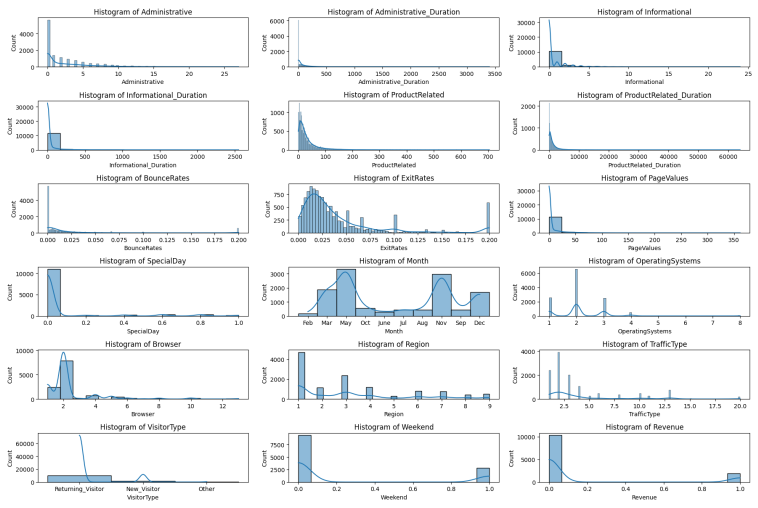

Figure 1: Histogram of Features

From the histogram, we can see the general pattern of several features. For example, the histogram of product-related features shows a right skew pattern, indicating that for most customers, the number of pages viewed related to the product is low. We can also notice the count for each month is also different, which can provide some intuitions on marketing strategy such as sales events or promotion.

We also generated the correlation matrix between each feature. Since our data has categorical values, even though they are in int or float type, the number is not ordinal and thus cannot provide a meaningful sense if we directly plot the correlation matrix. Hence, we plot the correlation heat map after encoding. While the size of the matrix is big, we can still understand the internal correlation of each feature so that we can better set up the following model selection procedure.

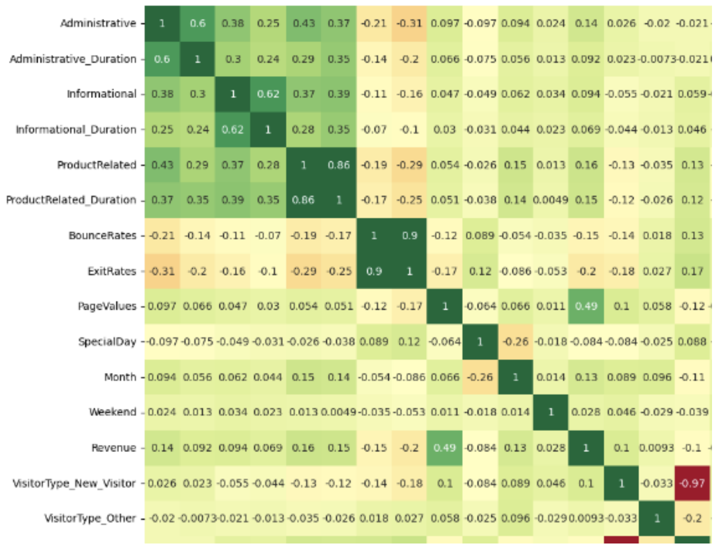

Figure 2: Part of the correlation matrix

3 Modeling

3.1 Data Preparation

To ensure robust training and evaluation of our models, we divided our dataset into three subsets following the 80-10-10 rule. These include the training set (80% of the data), a validation set (10%), and a test set (10%). All model comparisons are conducted on the validation set to provide an unbiased environment for testing the model's performance with real, unseen testing data. Prior to fitting the models, we standardized all features in the data by removing the mean and scaling to unit variance to prevent inaccurate predictions from models that are sensitive to the scale of data.

3.2 Imbalance Handling Methods and Comparison

In Section 1, data preprocessing revealed a significant class imbalance, which could potentially lead to biased predictions and reduced performance, especially for minority class instances. Our dataset consists of 12,330 samples with a notable imbalance: 10,297 instances represent non-completed transactions, and only 1,908 instances represent completed transactions after dropping duplicates. To counteract the adverse effects of this imbalance and enhance model performance, we explored three commonly used data handling methods: Stratified Shuffle Split, Class Weighting, and Synthetic Minority Over-sampling Technique (SMOTE).

We selected the Decision Tree Classifier as our baseline model, configured with *criterion='entropy', max_depth=10, min_samples_leaf=50, random_state=42*. Decision trees, known for their sensitivity to class distribution, serve as an excellent candidate for assessing the impact of different imbalance-handling techniques (Liu et al., 2022). We trained the baseline model on the training data after applying each of the three techniques and compared their performance by predicting the validation set using confusion matrices. Additionally, we established a baseline case by fitting the model on the original training data without any imbalance handling, predicting the validation set, and calculating the confusion matrix for comparison.

3.2.1 Stratified Shuffle Split

Stratified Shuffle Split is designed to ensure that each train and validation split of the data contains approximately the same percentage of samples of each target class as the original dataset. It prevents any training subset from being overwhelmed by the majority class, which could further exacerbate the imbalance. It ensures that the minority class is adequately represented in every split, which is vital for training models that can accurately recognize and classify examples from underrepresented classes.

Stratified Shuffle Split, as its name implies, contains three steps, stratification, shuffling, and data splitting. 1. Stratification: The primary goal of stratification is to ensure that each split is representative of the underlying class proportions. We divide the dataset into homogeneous subgroups based on the target classes before sampling. 2. Shuffling: After stratifying the data, we randomly shuffle the data each time it generates a new split. This shuffling is what differentiates it from standard stratified methods that may simply slice the data without shuffling, leading to less randomness in split assignments. 3. Data Splitting: after we first split 10% test data randomly from the original dataset, we then allocate 80% data to the training set. The process ensures that each class is split according to the specified proportions, thereby preserving the overall structure of the dataset.

3.2.2 Class Weighting

Class Weighting is a technique used to adjust the importance attributed to each class during the model training process. In this method, different weights are assigned to classes, influencing the model's loss function to penalize misclassifications of the minority class more severely than those of the majority class. To implement this, we adjusted our Decision Tree Classifier by adding the parameter *class_weight='balanced'*. This setting automatically adjusts the weights of each class inversely proportional to their frequencies in the input data. The formula used to calculate the weights is as follows:

$$weight_{class} = \frac{total\ number\ of\ samples}{number\ of\ classes\ \times\ number\ of\ samples\ in\ class}$$

This adjustment encourages the model to pay more attention to samples from underrepresented classes, aiming to mitigate the impact of class imbalance on model training.

3.2.3 SMOTE

SMOTE is an oversampling technique designed to increase the number of cases in the dataset in a balanced way. It was introduced by N. V. Chawla et al. in 2002 and is primarily used to address the imbalance in datasets when one class significantly outnumbers the other. Different from traditional oversampling methods, SMOTE generates synthetic samples from the minority class instead of creating copies. This approach helps in overcoming the overfitting problem, which arises when exact duplicates of the minority class examples are added to the dataset. SMOTE first identifies the minority class in the dataset. For each minority class instance, SMOTE selects one of its k-nearest neighbors and generates synthetic samples along the line segment joining them in the feature space. This is done by randomly choosing a point along the line to create a new instance that combines features of both original instances but in a new data point. It then adds the new data point to the training data so that we maintain an equal number of samples between the two classes.

3.2.4 Performance Analysis of Imbalance Handling Methods

After implementing each imbalance handling technique and predicting on the validation set, we performed an in-depth analysis of the confusion matrices to evaluate their effectiveness. Our validation set comprised 1,042 samples with non-finalized transactions and 179 positive (completed transaction) samples. Given the significant imbalance in our dataset, our primary focus was on the effectiveness of each method in predicting the minority class while also maintaining high accuracy for the majority class.

In the base case, where no specific imbalance handling techniques were applied, the model demonstrated reasonable accuracy in predicting majority class instances but showed limitations in identifying minority class instances, predicting only 104 out of 179 positive samples correctly. The Stratified Shuffle Split method maintained an equally high accuracy for the majority class, comparable to the base case. However, it correctly predicted only 98 positive samples, resulting in a higher Type II error rate, making its performance inferior to the base case. Both the Class Weighting and SMOTE techniques showed significant improvements in predicting minority class instances compared to the base case, with correct predictions for 149 and 135 positive samples, respectively. Despite this improvement, both methods exhibited a higher false positive rate. Notably, while the Class Weighting method showed some potential, the F1-scores for false/true and weighted categories were lower compared to those achieved by the SMOTE method. Additionally, SMOTE maintained a high accuracy rate for the majority class, predicting 941 out of 1,042 correctly. Detailed results of the confusion matrices can be found in Figure 3.

Based on the analysis, the SMOTE technique proved to be the most effective method for handling the class imbalance in our dataset. It not only improved the prediction accuracy for the minority class but also maintained a satisfactory accuracy level for the majority class. Considering these findings, we recommend adopting the SMOTE method for training other classifiers to ensure robust and equitable predictions across all classes.

3.3 Classification Models and Comparison

Following the preprocessing of data and addressing the imbalance issue with SMOTE, we embarked on the evaluation of various classification models. Our analysis encompassed both fundamental models, namely Support Vector Machine (SVM) and Logistic Regression, and advanced ensemble learning techniques, specifically Random Forest and Gradient Boosting Decision Tree.

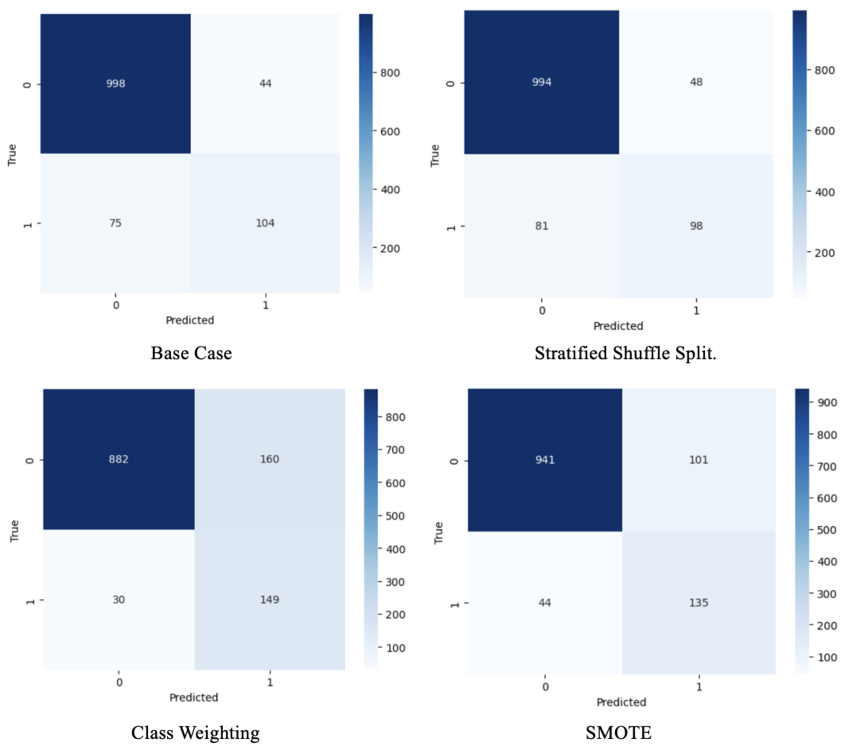

Figure 3: Confusion Matrices of the 4 Methods

3.3.1 Support Vector Machine (SVM)

SVM seeks to identify the optimal decision boundary to maximize the margin, thereby enhancing robustness and beneficial in predicting customer purchase intention where distinct patterns may exist. Its adaptability allows for the selection of diverse non-linear margins, enabling the capture of intricate patterns within the dataset.

3.3.2 Logistic Regression

Logistic Regression is selected for its simplicity and interpretability and often used as a baseline model due to its ease of implementation and understanding. It provides insights into the likelihood of a customer making a purchase based on various features.

3.3.3 Random Forest

Random Forest exhibits resilience to overfitting and outliers through the technique of bagging weak learners and introducing randomness. The utilization of decision trees as base learners enables the model to effectively capture complex and non-linear relationships between the features and shoppers' purchasing intentions.

3.3.4 Gradient Boosting Decision Tree (GBDT)

GBDT constructs a sequence of decision trees, proficiently capturing complex relationships and outliers in the dataset. Its sequential methodology facilitates high accuracy by prioritizing challenging instances. For this project, we opted for the XGBoost implementation of GBDT, as it offers efficient and scalable gradient boosting with regularization, alongside the capability for feature importance analysis.

3.3.5 Performance Analysis of Classification Models

Table 2 presents the performance of the models on the validation dataset. Throughout the experimentation process, all models consistently employed SMOTE as the oversampling technique, and hyperparameters were optimized via grid search to maximize the Area Under the Curve (AUC), a metric indicative of the model's ability to effectively discriminate between classes. Notably, the results indicate that XGBoost consistently outperforms the other models across all metrics evaluated. Therefore, based on its superior performance, XGBoost was selected as the final model for predicting customer purchase intention.

	SVM	Logistic Regression	Random Forest	XGBoost
AUC	0.902	0.912	0.942	0.946
Accuracy	0.888	0.876	0.894	0.902
Weighted F1 Score	0.896	0.886	0.898	0.904
Hyperparameter	Linear kernel Regularization param = 10	Regularization param = 0.1	Max depth = None N estimators = 200 Min sample leaf = 5	Learning rate=0.1 Max depth = 5 N estimators = 100

Table 2: Performance and Hyperparameters of Classification Models

We proceeded to evaluate the XGBoost model with the best hyperparameters *(learning_rate=0.1, max_depth = 5, n_estimators=100)* on the testing dataset. The selected model demonstrated strong performance metrics on the testing data, achieving **AUC of 0.933**, **accuracy of 0.901,** and **F1-score of 0.903**. Although the testing results exhibited a slight decrease compared to the validation performance, they remained consistent. This consistency indicates the robustness of our model and its ability to generalize well to unseen data. Remarkably, the confusion matrix in Figure 4 demonstrated clear discrimination between labels, affirming the successful handling of the imbalance issue inherent in the dataset.

Furthermore, our analysis delved deeper into the factors influencing customer behavior. The feature importance bar chart revealed that "PageValue" emerged as the most significant feature. This aligns with our intuition, suggesting that when previous customers tend to make purchases after visiting a particular page, it is probable that new customers will exhibit similar behavior. Following closely as the second most important feature is "TrafficType," indicating that customers originating from different channels or sources exhibit varying degrees of intention to make a purchase. This insight sheds light on the heterogeneous nature of customer behaviors and underscores the importance of considering diverse factors in predictive modeling.

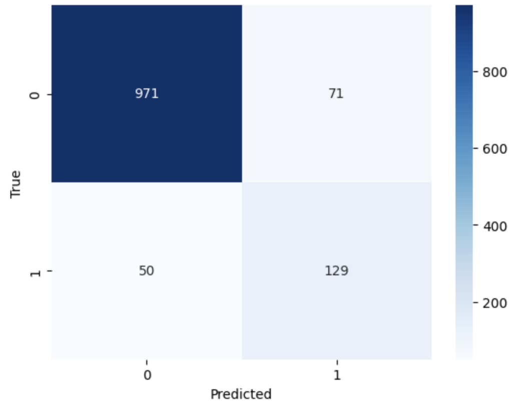

Figure 4: Confusion Matrices of XGBoost

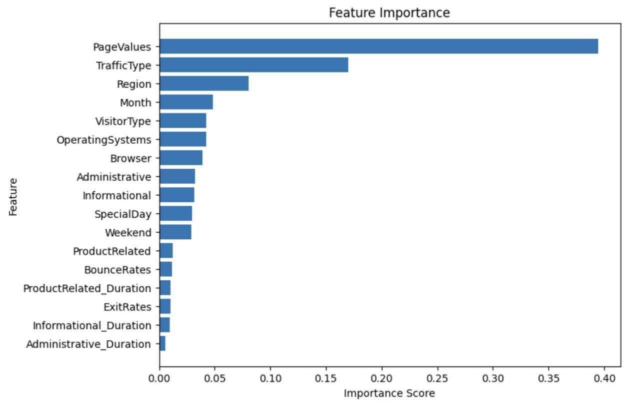

Figure 5: Feature Importance

4 Conclusion and Future Steps

Our approach, involving feature engineering, SMOTE for imbalance handling, and XGBoost with optimized hyperparameters, yielded an outstanding result of 0.901 accuracy in predicting customer purchase intention. "PageValues" and "TrafficType" emerged as key features linked to purchasing behavior. Despite our model's impressive performance, opportunities for improvement remain.

While the SMOTE technique has proven effective, continual enhancement and testing of imbalance handling methods remain crucial. Future research could explore hybrid approaches that combine the strengths of SMOTE with other oversampling or undersampling techniques to further reduce false positive rates without compromising the detection of minority class instances. Additionally, experimenting with different parameters for existing methods, such as adjusting the number of nearest neighbors in SMOTE, could lead to more optimized models.

Expanding our feature set to include demographic data like age, gender, and purchase history can enhance our model's predictive capacity by capturing a broader range of factors influencing purchase decisions, thereby improving accuracy. Additionally, adopting dynamic data testing methods, such as using timestamps or real-time data APIs, allows us to evaluate the model's adaptability to evolving trends and real-world scenarios, ensuring its relevance and robustness in practical applications.

5 Reference

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research, 16*, 321–357.

Liu, L., Wu, X., Li, S. et al. Solving the class imbalance problem using ensemble algorithm: application of screening for aortic dissection. *BMC Med Inform Decis Mak 22*, 82 (2022).

Sakar, Cemal Okan et al. “Real-time prediction of online shoppers’ purchasing intention using multilayer perceptron and LSTM recurrent neural networks.” *Neural Computing and Applications* 31 (2018): 6893 - 6908.

Dataset Source:

https://archive.ics.uci.edu/dataset/468/online+shoppers+purchasing+intention+dataset

Appendix

Feature Name	Categorical or Numerical	Feature Description
Product related	Numerical	Number of pages viewed related to the product
Product related duration	Numerical	Amount of time spent by the customer in viewing the product-related pages
Exit Rate	Numerical	Average exit rate for the user while viewing the page
Special Day	Numerical	Closeness of customer visiting time and special days
Visitor Type	Categorical	If a visitor is new, returning, or other
Weekend	Categorical	If the visit date is on the weekend
Brower	Categorical	Type of browser each customer used
Administrative	Numerical	Number of pages visited about accounts related
Administrative duration	Numerical	Amount of time spent by the customer on viewing the accounts-related page
Informational	Numerical	Number of pages visited about information related
Informational duration	Numerical	Amount of time spent by customers on pages about information related
Bounce rate	Numerical	Average bounce rate
Page value	Numerical	Average page value
OperatingSystems	Categorical	Operating system used by the customer
Region	Categorical	Region of each customer

TrafficType	Categorical	Traffic sources such as direct, SMS
Month	Categorical	Visited month of each customer