LUND UNIVERSITY
School of Economics and Management

Department of Economics

Classification of Premium and Non-Premium Products using XGBoost and Logistic Regression

An empirical study in the Food and Beverage Industry

Master's Essay in Data Analytics and Business Economics (DABN01, 15 credits)

Authors: Francisco Erazo and Stephany Rojas Gerena

Supervisor: Antonio Marañon

May 2022

Abstract

In the past few years, many industries have become interested in premium product segmentation to achieve higher unit margins. In this paper, we applied machine learning algorithms to predict whether a product is premium or non-premium. This product is manufactured by a food and beverage company that considers the incorrect classification of products as their primary concern, especially when incorrectly predicting premium products (False Positives). Therefore, the focus of this study is to minimize the misclassification of premium products. We selected Logistic Regression (LR) and XGBoost (XGB) and applied balancing methods, feature selection, and tuning parameters. The main contribution of this research is the application of a Cost-Sensitive (CS) analysis for addressing misclassification with a highly imbalanced dataset. According to our results, the model with the best performance was CS-XGB-SMOTE achieving a False Positive Rate (FPR) of 2.7%. A more robust way to assign the costs for the CS analysis and a direct modification of the loss function for XGB can be explored for future research and may improve the performance of this algorithm.

Keywords: XGBoost, Logistic Regression, Classification Algorithms, Food and Beverage, Cost-Sensitive Analysis, SMOTE.

Disclaimer: *Due to confidentiality reasons, it is not possible to present or divulge all aspects or characteristics of the cooperating company and its dataset.*

Acknowledgments

Firstly, we would like to express our sincere gratitude to our supervisor Antonio Marañon for his support and commitment throughout the process. We genuinely value all the time he dedicated to supervising our thesis.

Special Thanks

My special gratitude to my parents and sister for motivating me to achieve my personal and professional goals, and Yael, for her unconditional support and for being there when I needed it.

Francisco Erazo

My genuine gratitude to my beloved parents for their constant support and motivation to dream big. Thank you for backing me up in every goal I decide to pursue and constantly to remind me what I am capable of.

Stephany Rojas

Table of Contents

List of Figures

List of Tables

Abbreviations

AUC	Area under the ROC Curve
COR	Correct Instances
FN	False Negative
FP	False Positive
FPR	False Positive Rate
INCOR	Incorrect Instances
KNN	K-Nearest Neighbors
LR	Logistic Regression
ML	Machine Learning
N	Total Instances
NEG	Actual Negative Instances
PNEG	Predicted Negative Instances
POS	Actual Positive Instances
PPOS	Predicted Positive Instances
PPV	Positive Predictive Value
PR	Precision-Recall
RF	Random Forest
ROC	Receiver Operating Characteristic Curve
SMOTE	Synthetic Minority Oversampling Technique
SMOTEENN	SMOTE and Edited Nearest Neighbors
SMOTETomek	SMOTE and Tomek Links
SVM	Support Vector Machine
TN	True Negative
TNR	True Negative Rate
TP	True Positive
XGB	XGBoost

1. Introduction

Many industries have become interested in premium product segmentation in the past few years. The term premium refers to a product with superior quality, higher price, and selectively distributed with the highest quality channels. There are two main reasons why this kind of product is interesting for the companies. The first is the possibility of achieving higher unit margins, and the second is the potential to address consumer needs closely (Quelch, 1987).

Several researchers have applied machine learning to categorize food and beverage goods. Some consider supervised learning (Ma et al., 2021; Kumar, Agrawal and Mandan, 2020; Monforte, Martins and Silva Ferreira, 2021; Shaw, Suman and Chakraborty, 2019) and unsupervised learning Gómez-Meire et al., 2014, Mahima et al., 2020; Teye, Amuah, McGrath and Elliott, 2019). There is, however, not much research that has previously studied the classification of premium and non-premium products in the food and beverage industry. Furthermore, we have not encountered any published study about Cost-Sensitive analysis applied to classification problems. In that sense, the contribution of this paper is twofold, apply machine learning for the classification of premium and non-premium products in the food and beverage industry, and use Cost-Sensitive analysis to address misclassification costs.

The primary goal of this study is to predict whether the product is premium or non-premium, minimizing the misclassification of premium products (False Positives). This classification problem has a significant economic impact on the company as incorrectly classifying products is very costly. For that purpose, we used supervised machine learning classifiers to create a premium product prediction model. In Figure 1, a workflow diagram is presented. The selected algorithms for this study were Logistic Regression and XGBoost, and the results of each method were evaluated according to confusion matrix, accuracy, log-loss, Area Under the ROC Curve (AUC), sensitivity (recall), False Positive Rate (FPR) and precision.

The paper is organized as follows: Section 2 reviews previous literature regarding the implementation of classification machine learning algorithms in the food and beverage industry.

The theory behind the models (Logistic Regression and XGBoost) is explained in Section 3. Section 4 describes the details of the dataset, balancing and feature selection methods, tuning parameters, and optimal threshold approaches. It also includes the results for each of the practices described. Finally, Section 5 contains our conclusions, primary findings, and recommendations.

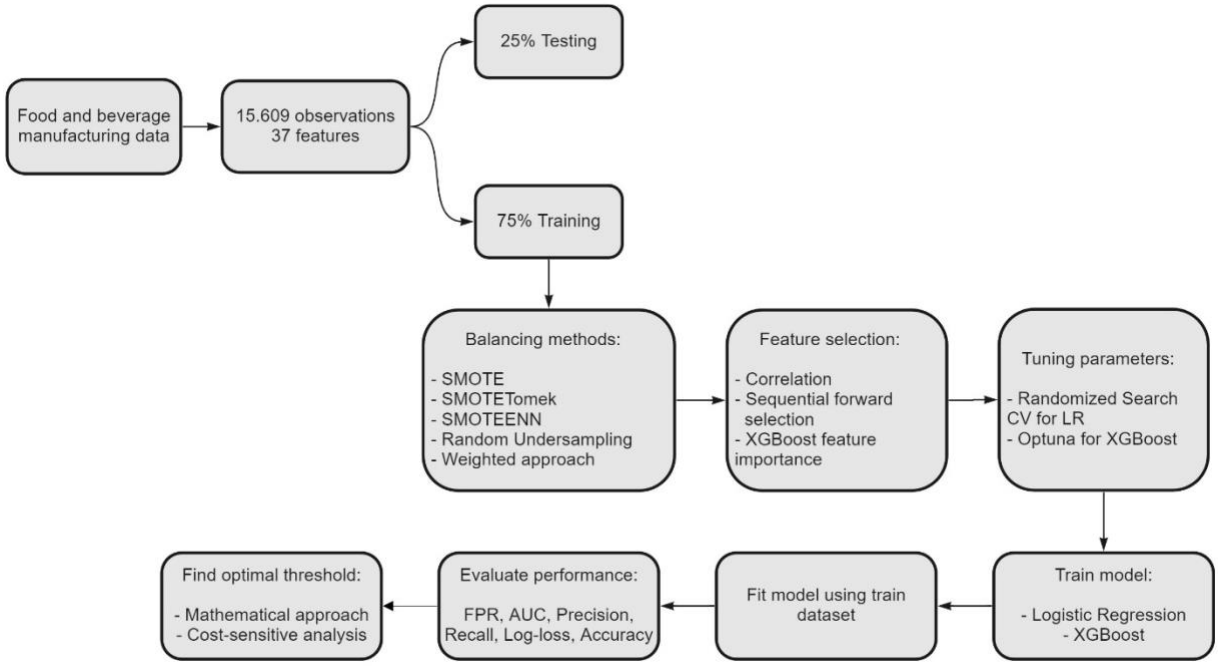


Figure 1. Workflow Diagram

2. Literature Review

Due to their proven success and practical applicability in various fields, the application of machine learning algorithms has been increasing within the food science domain in recent times (Wang, Bouzembrak, Lansink and Fels-Klerx, 2021a; Oliveira Chaves et al., 2021). Two of these methods are logistic regression (LR) and XGBoost (XGB). For years, the former has been considered one of the most commonly used statistical procedures for studying the classification of biological reactions (Granato, de Araújo Calado and Jarvis, 2014) and bacterial food growth (Hajmeer and Basheer, 2003; Ratkowsky and Ross, 1995). The latter, developed in 2016, increased in popularity since its introduction, implemented in food classification (He et al., 2021a; Yao et al., 2022) and beverage classification problems (Bhardwaj et al., 2022; Mu, Gu, Zhang and Zhang, 2020).

There is exhaustive research related to classification problems in the food industry. Rodríguez-Saavedra et al. (2021) implemented LR to help brewers predict whether microorganisms that produced spoilage in craft beer would grow or not. Bhardwaj et al. (2021) implemented an XGB algorithm to predict the wine quality of New Zealand pinot noir. Similarly, Trivedi and Sehrawat (2018) also predicted wine quality using LR and Random Forest (RF), where the classification was binary: good or bad quality wine. Jiang et al. (2020) used multinomial LR to classify the tomato according to their level of maturity. Saberioon et al. (2018) categorized rainbow trouts into fish-meal-based or plant-based diets to analyze the consequences of diet on the fish skin (since consumers perceive it as related to quality). De Andrade et al. (2022) applied ML classification algorithms to categorize artisanal cheeses into ten types and four producing areas in Brazil. Some of the supervised methods they used were Artificial Neural Networks, KNN, Learning Vector Quantization, LR, RF, and SVM. Koranga, Pandey, Joshi and Kumar. (2021) use LR, RF, SVM, Naïve Bayes, J48, and Multilayer Perceptron to classify white wine according to quality level. While there are several classification methods, this paper aims to structure binary classification using LR and XGB algorithms.

Wang et al. (2021b) used three feature selection methods: filter using correlation, recursive elimination, and Elastic-Net for penalization. The filter technique removes highly correlated variables, recursive elimination involves removing one feature at a time, and the Elastic-Net is a penalization that combines Lasso and Ridge penalties. For tunning the XGB parameters, Bayesian Optimization was utilized. This method uses the Bayes Theorem for maximizing or minimizing a given metric. Wang et al. (2021b) worked on a classification problem with imbalanced data, where there is a significant disparity in proportions between the binary categories. Motivated by them, we utilized an optimization method for tuning XGB parameters and three strategies for feature selection: correlation-based, recursive elimination with forward stepwise, and algorithm-level tuning.

Most research for imbalanced classification problems focuses on oversampling or undersampling methods for addressing the imbalance. Bhardwaj et al. (2021) used the Synthetic Minority Over Sampling Technique (SMOTE) to create synthetic observations due to the lack of information on wine quality variables. Abdulghani (2021) used SMOTE as a balancing technique for legitimate

and fraudulent transactions on credit cards. Following this approach, we used SMOTE as one of our balancing techniques.

There are diverse approaches for which metrics should be used to evaluate the model's performance with imbalanced data. Wang et al. (2021b) argue that using the traditional accuracy performance measure is incorrect, and instead, they use AUC. In the same line, Trivedi and Sehrawat (2018) and Koranga et al. (2021) consider accuracy insufficient as the only metric, so they employed f1-score, accuracy, 10-fold accuracy, precision, recall, and specificity. Saberioon et al. (2018) use the correct classification rate (CCR), Cohen's Kappa coefficient, sensitivity, specificity, and AUC ROC. On the other hand, Abdulghani (2021) uses accuracy as one of the primary metrics to evaluate and compare models. This paper does not focus only on accuracy as the primary metric since it can be misleading when one misclassification error is more costly. We also consider AUC, precision, recall, log-loss, and FPR as our main metrics to evaluate the algorithm's performance.

Different authors in the food and beverage industry have used different splits for training and testing the data, and in most cases, the datasets do not have many observations. Jiang et al. (2020) have 300 instances, and the dataset is split into 60% for training and 40% for testing. Rodríguez-Saavedra et al. (2021) used twenty craft beers and modified their features to create 331 instances, and for the validation data, they used ten craft beers. However, one drawback was the small number of instances. According to Beleites et al. (2013), for classification models, a sample size of 5 to 25 observations per category helps to achieve acceptable performance. However, a range between 75 to 100 observations is necessary to test or validate a fair classifier. De Andrade et al. (2021) have 422 observations that were split into 75% for the training dataset and 25% for the test dataset. The dataset used in this paper contains a significantly higher number of observations compared to previous research in the food and beverage industry. In that sense, we can have a smaller percentage of testing data and still cover many observations. Following De Andrade et al. (2021), we split the data into 75% for the training dataset and 25% for the test dataset.

Some authors in other industries have addressed the classification problem by combining algorithms. Dong et al. (2021) created a classification model for a system to predict whether a person had coronavirus or not. With this aim, they used an XGB algorithm combined with LR to

have better performance, achieving an AUC of 98.8%. They used the XGB as feature selection, and with those features, they built the LR. Adapting this to the domain of our paper, we used feature selection of XGB as input for LR, but unfortunately, it did not improve our results.

A cost-sensitive analysis has been suggested for cases where misclassification represents different costs. He et al. (2021b) applied a Cost-Sensitive analysis for an XGB model for classification for Malicious URL detection. The idea of the Cost-Sensitive approach is that the model is trained to consider that each misclassification error represents a cost, and some might be more expensive than others. In their paper, they created a new loss function for XGB that included the misclassification costs, which was also their way of balancing the data. Inspired by this idea, we explored the Cost-Sensitive analysis in XGB, but we took a slightly different approach. Instead of using Cost-Sensitive analysis as the balancing method, we used our balanced algorithm and applied the Cost-Sensitive approach to achieve better performance.

3. Theoretical Analysis

3.1 Machine Learning for Classification

The main goal of classification problems in machine learning is optimally categorizing objects in a set of classes. The output is a classification rule that determines which category the object should be classified. This classification rule is also called a classifier. We use some existing data, referred to as the training data, to train the algorithm and estimate the necessary parameters. The idea is to create a classifier that can perform adequately in unseen or test data (Schapire and Freund, 2014).

As mentioned earlier, there are multiple algorithms for classification purposes, and this paper focuses on LR and XGB.

3.2 Boosting

Gradient boosting is one of the most powerful techniques for prediction. It was created to make a weak learner a better predictor by using the learner multiple times (Brownlee, 2018). Those weak learners can refer to as classification trees. As a classification model by itself, trees produce too

accurate models and tend to overfit when generalizing to new data. Assemble methods such as boosting help to address this problem and make classification models better predictors (Wade and Glynn, 2020).

In boosting algorithms, multiple models are built sequentially. It means that each model is built to correct the errors of the previous model. The final prediction is a weighted average of the models, which gives more importance to the more accurate classifiers (Hastie, Tibshirani and Friedman, 2017). The process is as illustrated in Figure 2, where every model is trained based on the errors found in the previous model, and the prediction is the weighted average.

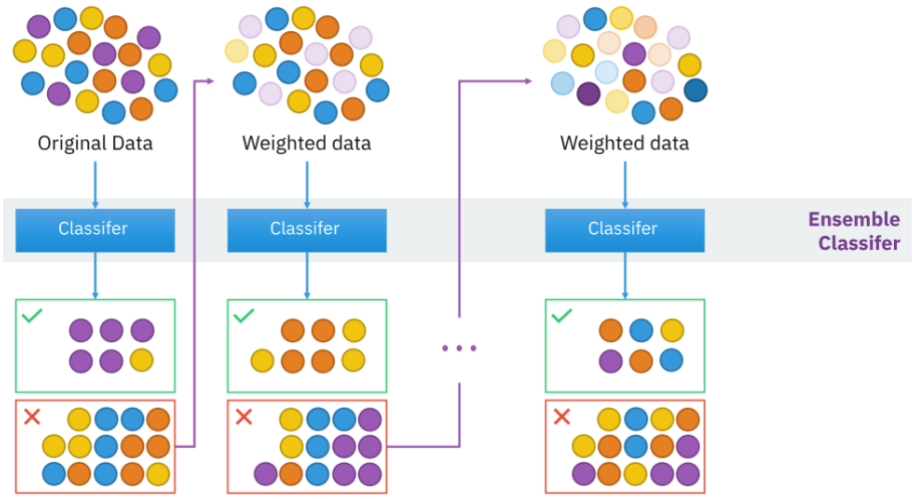


Figure 2. Boosting illustration. Taken from Wikimedia [File: Ensemble Boosting.svg - Wikimedia Commons]

3.2.1 XGBoost

Tianqi Chen from the University of Washington went deeper into the boosting methods and created a gradient boosting method that included regulation and was computationally more efficient. This method is called XGBoost, which stands for eXtreme Gradient Boosting (Wade and Glynn, 2020). It implements gradient boosting with trees as base learners. According to Wade and Glynn (2020) formally, the objective of XGBoost is to minimize the loss function that has an added regularization term, as follows:

$$obj^t = \sum_{i=1}^{n} l(y_i, \hat{y_i}^t) + \sum_{i=1}^{t} \Omega(f_t)$$

Where $l(y, \hat{y})$ is the loss function of the t-th boosted tree, and $\Omega(\theta)$ a regularization term. Since XGBoost is a boosting algorithm where the current prediction considers the previous one, the loss function part can be rewritten as follows:

$$obj^t = \sum_{i=1}^{n} l(y_i, \hat{y_i}^{t-1} + f_t(x_i)) + \sum_{i=1}^{t} \Omega(f_t)$$

Where $\hat{y_i}^{t-1}$ is the previous prediction and $f_t\,(x_i)$ is the current prediction. We explain the two components of this objective in greater detail.

- Loss function:

The loss function used for classification problems is log-loss. We can include the log-loss in the objective to obtain:

$$obj^t = \sum_{i=1}^{n} l(y_i - (\hat{y_i}^{t-1} + f_t(x_i))) + \sum_{i=1}^{t} \Omega(f_t)$$

Here, expanding the terms and using the fact that a second-degree polynomial can be rewritten using Taylor polynomial (further details in Wade and Glynn (2020)), we arrive at:

$$obj^t = \sum_{i=1}^{n} g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)^2 + \sum_{i=1}^{t} \Omega(f_t)$$

Where g_i and h_i can be written as partial derivative as follows:

$$g_i = \partial_{\hat{y_i}^{t-1}} l\left(y_i, \hat{y_i}^{t-1}\right)$$
$$h_i = \partial^2_{\hat{y_i}^{t-1}} l\left(y_i, \hat{y_i}^{t-1}\right)$$

The XGBoost algorithm is a solver that maximizes the objective function taking g_i and h_i as inputs.

- Regularization term:

This regularization term is what makes XGBoost different from other boosting algorithms. It is used to prevent overfitting as a penalty that is added to the loss function. Formally it is:

$$\Omega(f_t) = \gamma T + \frac{1}{2}\,\lambda\,\left|\left|w\right|\right|^2$$

Where T is the number of leaf nodes, ||w|| is the module of the leaf node vector, Y is the difficulty of node segmentation, and λ is the L2 regularization coefficient (Dong et.al, 2021).

3.3 Logistic Regression

LR is a statistical procedure that has been adopted in the field of ML due to its historical application in binary classification problems. It works similarly to other regression techniques, LR finds a model with optimal coefficients or weights that maximize the log-odds and can explain the relationship between a response variable and a group of explanatory variables. Its name comes from the base function it uses, the logistic or sigmoid function. Moreover, LR differentiates from the linear regression model since the outcome, or dependent variable, is binary (Hosmer, Lemeshow and Surdivant, 2013).

LR provides a probability of the default class (e.g., Y=1), wherein these probabilities are transformed into binary values using the sigmoid function. The log odds function used in LR is as follows:

$$\log\left(\frac{P(Y=1|X)}{1-P(Y=1|X)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_N X_N$$

Here, β_0 corresponds to the intercept, while $\beta_1 \ldots \beta_n$ are the weights related to the independent variables $x_1,\ldots,x_n$ These are not necessarily dichotomous, and they can be discrete or continuous, or a combination (Hajmeer and Basheer, 2002). The ratio on the left of the equation is the log-

odds. To calculate the odds, we take the probability that the default class will occur divided by the likelihood it will not happen. After that, the proportion is transformed using the logarithm, which results in the log-odds.

3.4 Confusion Matrix

The confusion matrix is a cross-tabulation used to summarize machine learning classifiers' performance by representing the counts of various classes (Larner, 2021). It is a variant of the well-known contingency table that has been broadly applied within the medical field (Matthews, 1995; Stigler, 2002).

The confusion matrix, also known as the error matrix, tabulates all instances into actual and predicted classes by presenting the number of observations of a real category in the rows against the observations of a predicted category in the columns. It helps graphically identify if a ML algorithm is confusing two or more categories when predicting (Larner, 2021).

		Predicted Class		
		Negative	Positive	**Total**
Actual Class	Negative	TN	FP	NEG
	Positive	FN	TP	POS
	Total	PNEG	PPOS	N

Figure 3. Confusion Matrix

This 2x2 table divides the instances into True Negative (TN), False Positive (FP), False Negative (FN), and True Positive (TP). The TN ("correct rejections") and TP ("hits") represent the correct classification in the classes of a binary variable, while FP ("false alarms") and FP ("misses") represent the misclassified observations between the same classes (Larner, 2021; Fernández et al., 2018).

Additionally, as specified in Table 1, we can obtain marginal totals from the confusion matrix by adding the values within the respective columns and rows, such as the total number of instances

(N), actual positive instances (POS), actual negative instances (NEG), predicted positive instances (PPOS), predicted negative instances (PNEG), correct instances (COR), and incorrect instances (INCOR) (Larner, 2021).

Table 1. Confusion Matrix Marginal Totals

Marginal Totals	**Formula**
Number of Instances (N)	TP + FP + FN + TN
Actual Positive Instances (POS)	TP +FN
Actual Negative Instances (NEG)	FP + TN
Predicted Positive Instances (PPOS)	TP + FP
Predicted Negative Instances (PNEG)	FN + TN
Correct Instances (COR)	TP + TN
Incorrect Instances (INCOR)	FP + FN

Other relevant measures can be derived from the confusion matrix, such as sensitivity and specificity, which are terms introduced by Yerushalmy (1947). Sensitivity is also called true positive rate (TPR) or recall, whereas specificity is also known as true negative rate (TNR). The former indicates the probability of a TP instance in the presence of the actual positive instances; the latter suggests the likelihood of a TN instance given the actual negative instances. There is a tradeoff between these two metrics. If the ratio of one increases, the other decreases (Larner, 2021).

$$Sensitivity\ (TPR\ or\ recall)\ = \frac{TP}{(TP + FN)}$$

$$Specificity\ (TNR)\ = \frac{TN}{(TN\ +\ FP)}$$

Similarly, we can also determine the False Positive Rate (FPR) and False Negative Rate (FNR). They are both measures of misclassification. FPR indicates the probability of an incorrect positive prediction given the actual negative instances, while FNR expresses the likelihood of an incorrect negative prediction given the actual positive instances (Larner, 2021).

$$FPR\ (1 -\ Specificity)\ = \frac{FP}{(TN\ +\ FP)}$$

$$FNR\ (1 -\ Sensitivity)\ = \frac{FN}{(TP\ +\ FN)}$$

The accuracy and inaccuracy (error rate) metrics are defined as follows. Accuracy is the sum of the TP and TN divided by the total number of observations from the confusion matrix, where higher ratio values imply better accuracy of the ML model. On the contrary, inaccuracy is the sum of the FP and FN divided by the total number of observations. It is desirable to obtain lower values for this proportion, which means lower error (Larner, 2021).

$$Accuracy\ =\ \frac{TP\ +\ TN}{N}$$

$$Inaccuracy\ (error)\ = \frac{FP\ +\ FN}{N}$$

Finally, the Positive Predictive Value (PPV) and Negative Predictive Value (NPV). The PPV or precision corresponds to the probability of a TP instance in the presence of the predicted positive instances. On the other hand, NPV is the probability of a TN instance given the predicted negative instances. Higher values for these ratios imply better predictive results (Larner, 2021).

$$PPV\ (precision)\ = \frac{TP}{(TP + FP)}$$

$$NPV\ = \frac{TN}{(FN\ +\ TN)}$$

All the metrics mentioned above range between 0 and 1, where 1 is the highest score and 0 is the lowest.

In this paper we pursued to reduce false positives (FP), meaning we want to achieve a low FPR. Given the highly imbalanced dataset, accuracy can be a misleading metric, since we could have a high accuracy by only predicting the majority class (only non-premium products), but we do care about predicting premium products. Therefore, we focus more on metrics such as precision, recall, log-loss, and FPR.

3.5 ROC Curve

One of the most common ways to measure the predictive power and performance of a classifier is by finding the misclassification errors. TPR and FPR can be understood as conditional probabilities of having a particular predicted class given the true class (Krzanowski and Hand, 2009). If t is the threshold, and s is the classification score. We can say that an observation is classified in class 1 or class 0, according to the following:

- True Positive Rate $p(s > t|Y = 1)$ is the probability that an observation in class 1 is correctly classified.
- False Positive Rate $p(s > t|Y = 0)$ is the probability that an observation in class 0 is incorrectly classified.

Given that definition, the class in which an observation is assigned depends on the definition of the threshold (t), which is the point that determines the cutoff of the class.

The Receiving Operating Characteristic (ROC) Curve is a graph that illustrates the TPR on the vertical axis and FPR on the horizontal axis as the classification threshold changes. The threshold takes values between 0 and 1, so the ROC curve lies between those values (Krzanowski and Hand, 2009). The ROC curve is helpful in ML because it illustrates the tradeoff between two types of error. In Figure 4, a typical ROC curve is shown. The ideal classifier would have TPR=1 and FPR=0 at the top left of the graph, with coordinates (0,1) (Brownlee, 2022).

The main index for the ROC curve is the Area Under the Curve (AUC). It refers to the probability that the scores given to a classifier will rank a positive class higher than the negative one. The AUC takes values from 0 to 1, where a value of 1 indicates a perfect classifier (Brownlee, 2022). AUC values of acceptable classification models are above 0.5. An AUC value of 0.5 suggests that the classification model exhibits no effective discrimination.

However, for cases of highly imbalanced data, ROC is not the most accurate measurement. ROC is insensitive to class distribution; therefore, it is very likely that using a default threshold (0.5), the predicted class corresponds to the majority class in most cases (Brownlee, 2022). One way to address this problem is to find a different threshold. This option is explored in this paper.

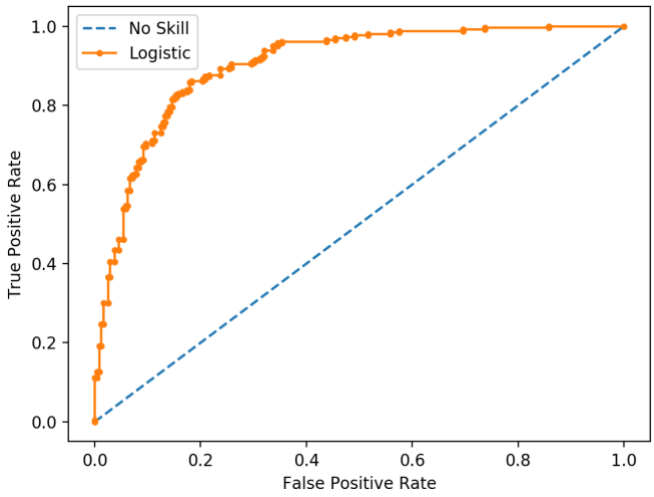


Figure 4. ROC Curve (Brownlee, 2022)

3.6 Precision-Recall Curve

A less frequent method for a graphical description of a model's performance, which is similar to ROC, is the Precision-Recall (PR) Curve. It illustrates the precision (PPV) against the recall (sensitivity). The curves within the PR plot could also be considered "threshold-free". Because they represent all the possible combinations of precision and recall ratios for different threshold values illustrating the evolution of the algorithm performance (Larner, 2021). Figure 5 shows the PR curve, where the orange line represents the curve among all possible combinations of thresholds.

For imbalanced datasets, PR reports the performance of an algorithm better than ROC curves since standard measures such as TPR and FPR are not affected by the weight ratios of the classes. On the contrary, precision or PPV can identify these differences on an imbalanced dataset (Cook & Ramadas, 2020; Ozenne, Subtil and Maucort-Boulch, 2015, Saito and Rehmsmeier, 2015).

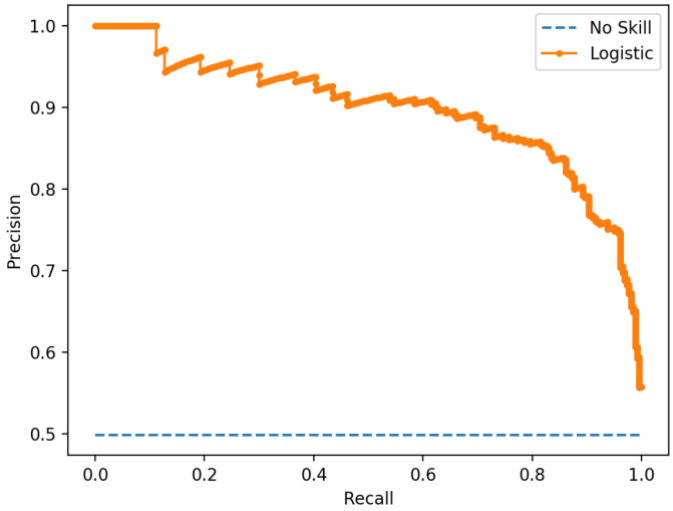


Figure 5. Precision-Recall Curve (Brownlee, 2022)

4. Empirical Analysis

4.1 Dataset Description

The dataset used for this study was provided by a food and beverage manufacturer. Considering that confidentiality is an important concern for the company and the authors, it is not possible to present and divulge all the aspects of the data.

The data file contains information about a manufactured product that can be classified as premium or non-premium. According to the company, the term “premium” defines a product with specific prominent characteristics and, consequently, a superior margin. For this case, the margin of a premium product is five SEK, while for a non-premium one, the margin is minimal, just one SEK.

The production process is described in Figure 6. We have two types of inputs: catalyzers and chemicals. These inputs are processed in machines for a given time and a specific temperature. This transforms the inputs into a final product (batch), which can be classified as premium or non-premium.

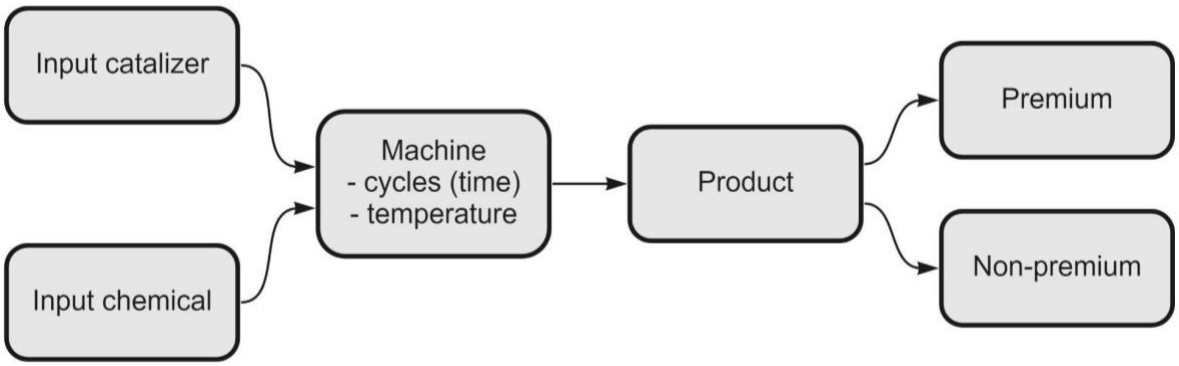


Figure 6. Production Process

The dataset contains 15609 observations and 37 features. The features are divided into three groups:

- Time: They represent a machine's time to do steps in the production process.
- Temperature: The temperature readings needed for steps in the process.
- Input: They represent chemicals or catalyzers needed as inputs for production.

The outcome variable is a categorical variable that indicates whether the product/observation is premium or not. The corresponding category proportions can be seen in Figure 7, where 0 is a non-premium product and 1 is a premium product. As can be seen, there are much more observations for non-premium (88% of total observations) than for premium products (12% of total observations), which means that we have an imbalanced dataset with a ratio of 7.2 non-premium products for each premium product.

From the 37 features, 4 are categorical variables and 33 quantitative variables. We exclude two variables from the analysis, one for possible data leakage and another related to the time. The manufacturer did not consider the latter relevant for addressing the classification problem.

The goal of the models is twofold, on one side, predict whether the product is premium or non-premium, and on the other side, have the least possible amount of False Positive (FP). This means that we should target the minimum possible incorrect predictions of premium products since this is the company's primary concern.

Nowadays, the manufacturer can only identify whether a product is premium or non-premium after its commercialization in the market for some time. They consider that predicting whether a product is a premium or not can affect the planning, storage, and logistical processes of that product. And as a result, it can lead to costs and expenses optimization.

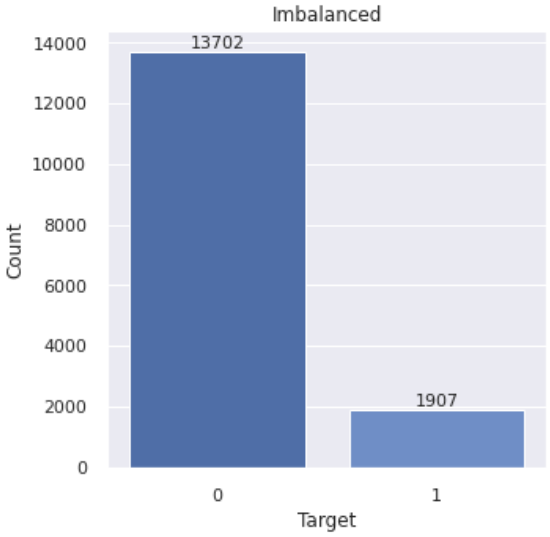


Figure 7. Outcome Variable Class Distribution

4.2 Data Cleaning

4.2.1 Encoding Variables: One Hot Encoder

XGBoost, and in general machine learning algorithms, may assume that encoded numerical variables have an ordinal relationship. If the categories are 0, 1, 2, or 3, the algorithm would assume that 3 is somehow more important or higher than 0, which is not the case (Brownlee, 2018).

We created dummy variables for each category to solve this technical predicament. This is done for the three categorical variables and increases the feature space from 35 features to 148 features since several levels or categories exist.

4.3 Train and Test Datasets

An effective way to evaluate an ML model performance is to separate the data into two different parts, one for training and the other for testing. The model is fitted using the first subset, and the second makes it possible to assess its predictions against the actual results or evaluate its performance on new unseen data (Hastie, Tibshirana, & Friedman, 2009).

There is no consensus about an optimal train and test split in ML due to the difficulty or high cost some fields face in obtaining data (Tan et al., 2021). For that reason, various studies employ different proportions for splitting. In our research, we proceed with 75% for the training set and 25% for the testing set, following the approach of De Andrade et al. (2021).

4.4 Balancing Data

Most machine learning algorithms operate in classification data where the two classes have a comparatively equivalent number of observations. In imbalanced datasets, the minority class would be considered unimportant and could be ignored to achieve high performance (Brownlee, 2021). In other words, if the imbalance problem is not considered, the model would predict the majority class and ignore the minority class, still giving high accuracy.

In our case, it is essential to consider both premium and non-premium categories for the prediction, so we balance the imbalanced dataset so that the algorithms consider both classes for the predictions. However, we do this only for the training dataset since this is the one that is used for training or structuring the algorithm. It is crucial to keep the test dataset as close to reality as possible. We use two approaches for balancing the training data: data-level and algorithm-level solutions.

4.4.1 Data-Level Solution

It is considered one of the first solutions for imbalanced datasets. Essentially, the solution consists of altering the data using sampling procedures, such as oversampling and undersampling, to address the imbalance problem (generate a balance between the classes).

The former originates a greater dataset by cloning some observations or producing new observations using the current ones. The latter creates a smaller dataset by removing observations from the majority class (Fernández et al., 2018).

4.4.1.1 SMOTE (Synthetic Minority Oversampling Technique)

SMOTE is an oversampling technique that works by creating synthetic samples that are neighbors in the feature space. The algorithm chooses a point in the minority class, finds the KNN in the feature space, draws a line between them, and creates a point inside that line, which is the synthetic sample (Chawla, Bowyer, Hall and Kegelmeyer, 2002; Brownlee, 2022). In Figure 8, the resulting dataset after using the SMOTE technique can be observed.

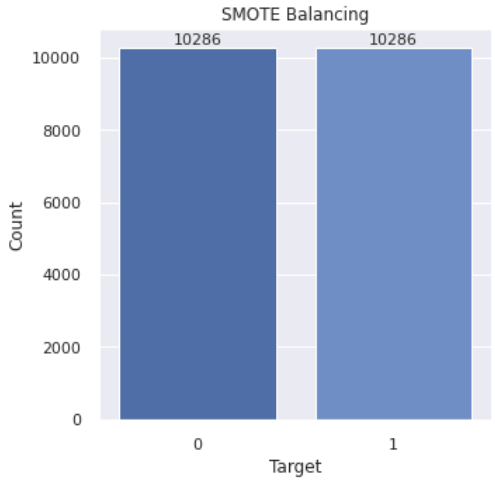


Figure 8. Class Distribution with SMOTE Balancing Method

4.4.1.2 SMOTETomek (SMOTE and Tomek Links)

SMOTE + Tomek Links is a hybrid-sampling minority technique. It deals with the problem of interpolating one of the classes in the space of the other class after applying SMOTE oversampling. Training a model in this kind of scenario usually leads to overfitting. Tomek Links cleans the oversampled dataset by removing the majority or minority class instances invading the other class's space, generating a balanced training dataset with distinct class clusters (Fernández et al., 2018; Batista, Prati and Monard, 2004). In Figure 9, the dataset after using the SMOTETomek technique can be observed.

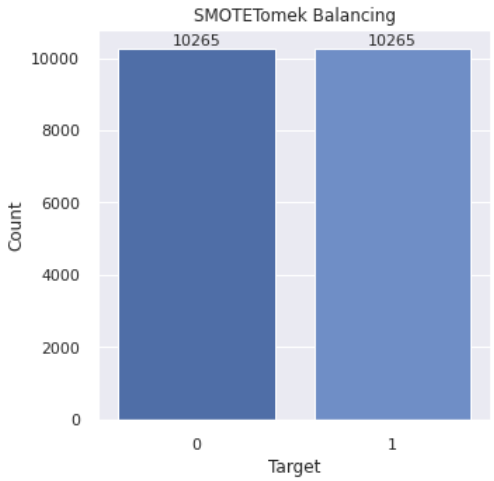


Figure 9. Class Distribution with SMOTETomek Balancing Method

4.4.1.2 SMOTEENN (SMOTE and Edited Nearest Neighbors)

SMOTEENN has a similar motivation as SMOTETomek since this technique combines the creation of synthetic samples using SMOTE with a removal technique. Edited Nearest Neighbors (ENN) is a technique used to remove samples from both classes (the majority and minority class) when the observation's class is different from the class of the KNN in the majority class. Compared to the Tomek Links, ENN removes more observations, providing a deeper data cleaning (Batista, Prati and Monard, 2004). In Figure 10, the dataset after using the SMOTEENN technique can be observed.

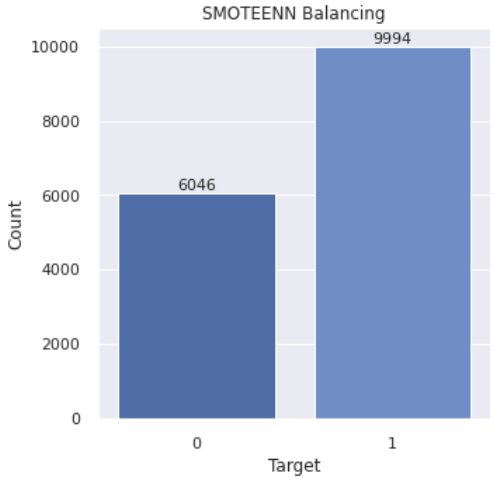


Figure 10. Class Distribution with SMOTEENN Balancing Method

4.4.1.3 Random Undersampling

Random Undersampling is a method that balances the class distribution by randomly removing instances of the majority class. The main disadvantages of this technique are twofold. Firstly, in a highly imbalanced dataset, a significant part of the majority class is deleted, causing a substantial loss in data. Secondly, potentially valuable data for addressing the problem could be removed (Fernández et al., 2018; Batista, Prati and Monard, 2004).

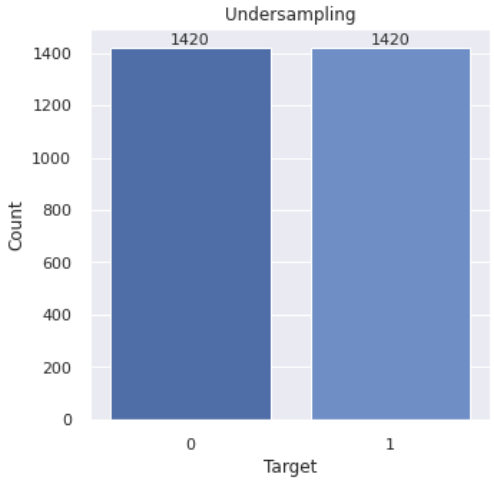


Figure 11. Class Distribution with Random Undersampling Method

4.4.2 Algorithm-Level Solution

Algorithm-level solutions are alternative methods to data-level solutions for handling imbalanced datasets. This perspective directly modifies the training process of the algorithm instead of altering the training dataset to address the imbalance (Fernández et al., 2018).

4.4.2.1 Weighted Approach

This method consists of handling the imbalanced data by assigning the weight ratio of the majority class to the minority class (sum of the majority instances divided by the sum of the minority instances). The weighted approach increases the minority class influence in the training process of the ML method (Fernández et al., 2018).

Both XGBoost and Scikit-Learn libraries for Python programming language allow specifying this weight ratio in the parameter "scale_pos_weight" in the former and the "class_weight" parameter for the latter.

In XGBoost, the "scale_pos_weight" parameter is a way to control the balance of positive and negative weights. The usual value to consider is the total number of negative instances divided by the number of positive instances (XGBoost Parameters — xgboost 2.0.0-dev documentation, 2022). The negative instances refer to the majority class, which in our case is non-premium. So, taking the imbalanced dataset, we tune this parameter as follows

$$scale\_pos\_weight = \frac{\sum\ negative\ instances}{\sum\ positive\ \ instances} = \frac{10286}{1420} = 7.2436$$

In LR, the "class_weight" parameter is used to assign the weights related to the classes, and by default, it is specified that both classes have the same weight. The ratios are calculated through the following formula:

$$class\_weight = \frac{\#\ of\ instances}{\#\ of\ classes \times \#\ of\ instances\ in\ each\ class}$$

$$class\_weight\ (positive) = \frac{11706}{2 \times 1420} = 4.1218$$

$$class\_weight\ (negative) = \frac{11706}{2 \times 10286} = 0.5690$$

For the positive class (premium), the weight is 4.1218, while for the negative class (non-premium), the weight is 0.5690.

4.4.3 Balancing Methods Results

We implement XGB (Table 2) and LR (Table 3) with five balanced training datasets (SMOTE, SMOTETomek, SMOTEENN, Random Undersampling, and Weighted). Each dataset considers 147 parameters for predicting whether a product is premium or non-premium (1 or 0). A total of 3903 instances were used to test the models. The testing instances were obtained before applying

any balancing method. Additionally, we evaluated each model's performance on test data using precision, recall, FPR, AUC, log-loss, and accuracy.

XGB metrics are available in Table 2. For this case, a training dataset balanced with SMOTE provides the highest accuracy (88.1%), AUC (0.8575), and FPR (6.5%). XGB-SMOTETomek ranks as the second-best model with an accuracy of 87.3%, an AUC of 0.8571, slightly below XGB-SMOTE, and an FPR of 6.7%, lightly above XGB-SMOTE. On the other hand, XGB-Undersampling registers the highest recall with 72.4%, followed by XGB-SMOTEENN with 69.4%. The remaining models present a ratio of less or equal to 50%. Regarding precision, XGB-SMOTE has a ratio of 52.5%, while the other models are under 50%, especially XGB-Undersampling, which registers the lowest precision with a value of 25.1%.

Table 2. XGBoost Metrics According to Balancing Method

Metrics	SMOTE	SMOTETomek	SMOTEENN	Undersampling	Weighted
accuracy	0.8811	0.8734	0.8175	0.6956	0.8642
precision	0.5247	0.4922	0.3751	0.2509	0.4573
recall	0.5010	0.4559	0.6940	0.7248	0.4723
FPR	0.0647	0.0670	0.1648	0.3085	0.0799
AUC	0.8575	0.8571	0.8446	0.7892	0.8405
log-loss	0.2895	0.2914	0.4109	0.6152	0.3177

According to Table 3, the results demonstrate that training the LR algorithm using a SMOTETomek dataset provides the highest accuracy (84.6%) when evaluating it with the test data, just slightly above LR-SMOTE (84.5%) and LR-SMOTEENN (81.3%). On the other hand, undersampling and weighting the training dataset decrease the accuracy of LR on test data to less than 70%.

LR-SMOTETomek reduces log-loss and FPR the most, with 0.3763 and 8.3%, respectively. LR-SMOTE performs similarly with 0.3775 and 8.3% on the same metrics. The remaining models present a log-loss over 0.40 and FPR above 14%. For AUC, LR-Weighted performs better with a metric of 0.78. The rest of the models perform similarly, with an AUC around 0.77.

LR-SMOTEENN received the highest precision score with 49%, followed by LR-SMOTETomek (37.4%) and LR-SMOTE (36.8%). The other model's score was below 25%. However, LR-

Undersampling and LR-Weighted performed better in recall than LR with oversampling methods, both with a ratio of 74.9% above the average of 34% of the oversampling models.

Table 3. Logistic Regression Metrics According to Balancing Method

Metrics	SMOTE	SMOTETomek	SMOTEENN	Undersampling	Weighted
accuracy	0.8447	0.8460	0.8127	0.6777	0.6882
precision	0.3681	0.3744	0.4949	0.2432	0.2500
recall	0.3409	0.3491	0.332	0.7495	0.7495
FPR	0.0834	0.0831	0.1420	0.3326	0.3206
AUC	0.7715	0.7744	0.7774	0.7726	0.7812
log-loss	0.3775	0.3763	0.4099	0.5843	0.5677

Creating synthetic data in the training dataset through oversampling methods (SMOTE, SMOTETomek and SMOTEENN) is relevant to preparing algorithms for addressing the imbalance between classes, for this case, premium, and non-premium products. The undersampling method is not the way to address this kind of problem. The results in either XGB or LR are not satisfactory. The reason is the tremendous amount of lost data, which could be relevant in the learning of the model. This is one of the disadvantages of this technique.

Regarding the weighted approach, it is a method that works best with XGB than LR. It provides better accuracy, log-loss, FPR, and precision results than the XGB-SMOTEENN model. LR-Weighted performs just slightly better than LR-Undersampling, and unlike XGB-Weighted, it cannot provide superior results to oversampling methods.

Finally, considering the metrics detailed in Table 2 and 3, XGB-SMOTE and LR-SMOTETomek are the top candidates for predicting premium and non-premium products.

4.5 Feature Selection

Multiple factors may affect machine learning algorithms' performance. Some studies have shown that algorithms can be adversely affected by irrelevant attributes. Therefore, selecting relevant features is critical to feed the model with pertinent information to get better performance or reduce dimensionality to allow a more straightforward interpretation (Hall, 1999). We compared three

different feature selection techniques, and we discussed the performance of each one of them in both LR and XGB algorithms.

4.5.1 Filter Method: Correlation-Based

This technique uses statistical measures as a filter to generate a subset with the essential features. Most of these statistical measures are univariate. In other words, they evaluate every feature on its own. A critical difference between the filter model and the wrapper model is that it is low computation demanding; however, its disadvantage is that we can lose relevant associations among variables due to the selection of redundant features.

To apply the filter method is vital to recall two concepts, relevance and redundancy. The first evaluates the predictor towards the class; the second evaluates the predictor against other predictors (Cai, Luo, Wang and Yang, 2018; Xu, Tang, He and Man, 2017). One of these methods is the Pearson correlation. The logic is to specify a threshold to remove the highly correlated features among themselves and with a low correlation regarding the outcome variable, avoiding collinearity problems. There is no specific threshold to argue whether a predictor is relevant enough for the model (Kuhn and Johnson, 2013). However, most researchers consider a cutoff greater than 0.9 as a strong relationship (Schober, Boer and Schwarte, 2018). In this research, we consider a threshold of 0.8.

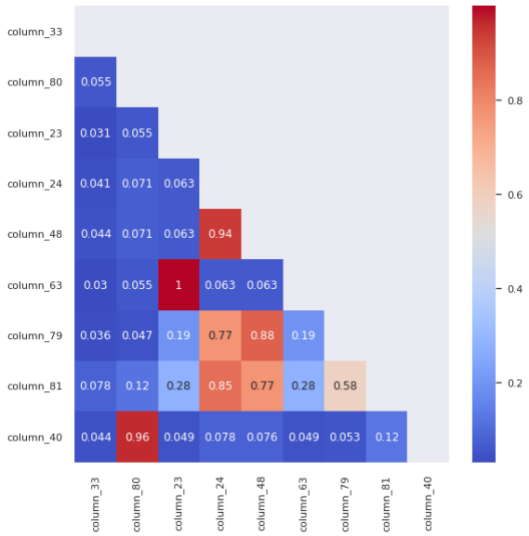


Figure 12. Correlation Matrix using Pearson's R (Variables with a Coefficient > 0.8)

Table 4 summarizes the Pearson correlation coefficients above the established threshold (0.8), for five variable interactions, at the same time, it shows which variable is discarded (not relevant) and held (relevant). On that account, "column_63", "column_80", "column_24", "column_79", "column_24" were removed. Conversely, "column_23", "column_40", "column_48", "column_81", along with other variables (with correlation coefficients <0.8) were retained.

Table 4. Variables with a Correlation Coefficient > 0.8

Features	Pearson Corr.	Relevant	Not Relevant
column_23 & column_63	1	column_23	column_63
column_40 & column_80	0.96	column_40	column_80
column_24 & column_48	0.94	column_48	column_24
column_48 & column_79	0.88	column_48	column_79
column_24 & column_81	0.85	column_81	column_24

4.5.2 Wrapper Method: Sequential Forward Selection (SFS)

Wrapper methods use classification error or accuracy as the performance metric to find a better performance by adding or removing features. Compared to filter methods, wrapper methods tend to have better performance. However, they imply poor generalization capacity and higher computational cost (Cai, Luo, Wang and Yang, 2018).

4.5.3 Embedded Method: XGBoost

In XGBoost, a variable registers higher importance when it is used more frequently in tree splits. Variable importance is calculated by the amount that each split improves the performance measure weighted by the number of observations in the node (Brownlee, 2018).

4.5.4 Feature Selection Methods Results

We implemented three methods (Filter Method-Correlation, Sequential Forward Selection, and Feature Importance) in our best models (XGB-SMOTE and LR-SMOTETomek) to obtain the relevant features from the 147 independent variables.

For both algorithms XGB and LR, four features were irrelevant for the model in the filter method (using Pearson Correlation); therefore, 143 features were selected. In the case of XGB with SFS, the 146 best features were considered, and 141 features were chosen with feature importance. On the other hand, 146 variables were deemed to be relevant for LR using SFS. Finally, with feature importance, 141 parameters were chosen.

For the XGB model, as observed in Table 5, correlation, SFS, and feature selection slightly increased the FPR, which is undesirable for our model, having the highest increase with SFS. Precision, recall, AUC, and accuracy decrease with all feature selection methods. The log-loss increased for the three methods. Therefore, we kept the XGB model without the feature selection method since none of the methods improved the metrics.

Table 5. XGB-SMOTE Metrics According to Feature Selection Method

Metrics	**SMOTE**	**P. Correlation**	**SFS**	**F. Importance**
accuracy	0.8811	0.8734	0.8352	0.8760
precision	0.5247	0.4922	0.3097	0.5032
recall	0.5010	0.4579	0.2607	0.4825
FPR	0.0647	0.0673	0.0828	0.0679
AUC	0.8575	0.8583	0.7374	0.8550
log-loss	0.2895	0.2920	0.3805	0.2964

As specified in Table 6, for LR, SFS can remove the less important feature from the model while maintaining similar metrics as the original one (LR-SMOTETomek). The Pearson correlation filter is the second alternative with an accuracy of 84.4%, slightly below SFS. However, it reported the lowest FPR (8.23%). The precision, recall, and AUC decreased for Pearson correlation, SFS, and feature importance, while the log-loss increased in all methods. In particular, the feature importance method metrics were the most affected.

Table 6. LR-SMOTETomek Metrics According to Feature Selection Method

Metrics	**SMOTETomek**	**P. Correlation**	**SFS**	**F. Importance**
accuracy	0.8460	0.8442	0.8458	0.8419
precision	0.3744	0.3628	0.3731	0.3587
recall	0.3491	0.3285	0.347	0.3388
FPR	0.0831	0.0823	0.0831	0.0864
AUC	0.7744	0.7716	0.7744	0.7688
log-loss	0.3763	0.378	0.3764	0.3805

We showed that relevant features were chosen using three methods for feature selection. However, we did not observe an advantageous effect on the model's performance in either of these cases. Considering this, we remained with the original model XGB-SMOTE and LR-SMOTETomek.

4.6 Parameter Tuning

One way of improving the performance of an algorithm is through parameter tuning. Considering that ML algorithms are parametrized, their performance can enhance using an optimal combination of parameters for a specific problem. Finding this combination is an iterative process. It starts with some random estimate aiming to land on a superior or optimum explanation for the problem (Yang, Deb, Loomes and Karamanoglu, 2013).

4.6.1 Tuning Parameters in XGBoost

XGBoost has many parameters that can be configured. Most parameters are created to address the bias and variance tradeoff. In other words, balancing model complexity and predictive power. A complete explanation of the XGBoost parameters can be found on their documentation (XGBoost Parameters — xgboost 2.0.0-dev documentation, 2022).

- **objective:** we selected "binary:logistic", which is used for logistic regression for binary classification.
- **colsample_bytree:** it represents the fraction of columns to be randomly sampled for each tree. This parameter helps to reduce the influence of the columns and reduce variance. The value must be between 0 and 1.
- **gamma:** it is a regularization parameter used to control model complexity. Formally, this is the minimum loss reduction required to make a further partition on a leaf node of the tree.
- **max_depth:** it refers to the maximum depth of a tree. A deeper tree might increase the performance and make the model more complex and more likely to overfit. The value must be an integer.
- **learning_rate:** it determines the step size at each iteration while your model optimizes toward its objective. A low learning rate makes computation slower and requires more

rounds to achieve the same reduction in residual error as a model with a high learning rate. However, it optimizes the chances of reaching the best optimum. The value must be between 0 and 1.

- **n_estimators:** it represents the number of trees in our ensemble. Equivalent to the number of boosting rounds. The value must be an integer.
- **subsample:** it refers to the fraction of observations to be sampled for each tree. Lower values prevent overfitting but might lead to under-fitting. The value must be between 0 and 1.

We used Optuna optimization as the hyperparameter tuning method. This optimization method allows the user to create the feature space dynamically. Compared to the most widely used techniques, such as “Grid Search CV” and “Randomized Search CV”, Optuna is efficient computationally and is scalable and versatile (Akiba et al., 2019).

In most of the literature, the objective of the Optuna optimization is to maximize accuracy. However, for this paper, the objective is to maximize precision since this metric cares about the prediction of premium products. In Table 7, we summarize the parameter values evaluated and the optimal value that was found.

Table 7. XGB Hyperparameter Tuning Values

Parameter	Default value	Parameter values	Best Value
max_depth	6	Range between [1, 9]	1
gamma	0	Range between [1e-8, 9]	4.0700
n_estimators	100	Range between [100, 1000]	642
learning_rate	0.3	Range between [1e-8, 1]	0.9300
colsample_bytre	1	Range between [0.3, 1]	0.3000
subsample	1	Range between [0.4, 1]	0.6100

4.6.2 Tuning Parameters for Logistic Regression

In LR, there are no parameters that can be tuned. However, Scikit-learn allows to iterate LR using "Randomized Search CV" or "Grid Search CV" with three arguments that can lead to an enhancement of the model performance, solver, penalty, and penalty strength.

- **solver:** it corresponds to the algorithm used for the optimization. There are five options:
    - "newton-cg": A Newton method that computes the inverse of Hessian matrix explicitly, so, for large datasets, it can be high computational demanding (see more at Royer, O'Neill and Wright, 2019).
    - "lbfgs": It stands for Limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm. It computes an estimation of the inverse Hessian matrix (see more at Liu and Nocedal, 1989)
    - "liblinear": It stands for Library for Large Linear Classification; it employs a coordinate descent method to advance towards the minimum in every iteration (see more at Fan et al. 2008).
    - "sag": SAG or Stochastic Average Gradient is a variant of gradient descent. It uses an earlier gradient value to obtain a speedier convergence than the Stochastic Gradient (SG) (see more at Schmidt, Le Roux and Bach, 2016).
    - "saga": It is derived from SAG. It allows more regularization method.
- **penalty:** It refers to the regularization methods, L1 (Lasso regularization), L2 (Ridge regularization), Elastic-Net (combination of L1 and L2), and none. It is important to reiterate that not all solver methods work with all the penalty options, see Table 8.
- **C:** It represents the inverse of penalty strength; lower values lead to more robust regularization.

Table 8. Solver Options for LR and their Respective Penalizations

Solver	L1 penalty	L2 penalty	Elastic-Net	None
newton-cg	no	yes	no	yes
lbfgs	no	yes	no	yes
liblinear	yes	yes	no	no
sag	no	yes	no	yes
saga	yes	yes	yes	yes

In Table 9, we summarized the parameter values evaluated with LR and the optimal value found.

Table 9. LR Parameter Tuning Values

Parameter	Default value	Parameter values	Best Value
solver	lbfgs	lbfgs, newton-cg, liblinear, sag, saga	newton-cg
penalty	l2	l1, l2, elasticnet, none	l2
C	1	np.logspace(-4, 4, 20)	1438.4498

4.6.3 Parameter Tuning Results

Having implemented the feature selection methods for both algorithms, we proceeded with parameter tuning for the top models so far (XGB-SMOTE and LR-SMOTETomek). The methods used for tunning parameters were Optuna for XGB-SMOTE and Randomized Search CV for LR-SMOTETomek.

On the one hand, for XGB-SMOTE there was no improvement in metrics after tuning parameters using Optuna optimization. In fact, in Table 10, we noted that all the metrics performed worst. The AUC decreased, same with precision and recall. The FPR increased, which is undesirable since we care about the FP.

Table 10. XGB Metrics with Parameter Tuning

Metrics	XGB SMOTE	XGB - Tuned SMOTE
accuracy	0.8811	0.8178
precision	0.5247	0.3108
recall	0.5010	0.3778
FPR	0.0647	0.1194
AUC	0.8575	0.6797
log-loss	0.2895	0.6931

In the case of the LR-SMOTETomek model, there was an improvement in its metrics using parameter tuning. The model was enhanced with newton-CG (solver), l2 (regularization or penalty), and a value of 1438 (C: inverse of penalty strength). In Table 11, it is possible to analyze a summary of their results.

After implementing SFS and parameter tuning in LR, we observed that accuracy, log-loss, and FPR enhanced compared to the original model, while metrics such as precision, recall, and AUC slightly declined.

Table 11. LR Metrics with Parameter Tuning

Metrics	LR SMOTETomek	LR - Tuned SMOTETomek
accuracy	0.8460	0.8475
precision	0.3744	0.3714
recall	0.3491	0.3203
FPR	0.0831	0.0773
AUC	0.7744	0.7712
log-loss	0.3763	0.3737

To conclude, after tunning parameters, LR-SMOTETomek was enhanced. We noticed some improvement specifically in FPR, which is our primary goal for this classification problem. In the case of XGB-SMOTE, there was no enhancement.

4.7 Models Comparison

We proceeded with three techniques for both algorithms: balancing methods, feature selection, and tuning parameters. Here we summarized those findings and selected the best algorithm for this paper.

We tried five balancing methods for each algorithm: SMOTE, SMOTEEN, SMOTETomek, Random Undersampling, and Weighted Approach. For LR, we found that the balancing method that performed best was SMOTETomek, and for XGB, it was SMOTE. In both XGB and LR, we discovered that feature selection methods did not improve the model's performance. Therefore, the models without feature selection were kept. Lastly, we tuned parameters for each algorithm, finding that tunning did not improve XGB, but it did for LR.

We compared the best models for each algorithm: XGB-SMOTE and LR-SMOTETomek (Tuned). In Figure 14, the ROC curve for both models can be observed. For the default threshold (red point in the curve), XGB achieved a significantly higher true positive rate (TPR) than LR, having a similar FPR. XGB is more aligned with the "perfect classifier curve" compared to LR. This is closer to the coordinate (0,1), which is desirable for a classifier.

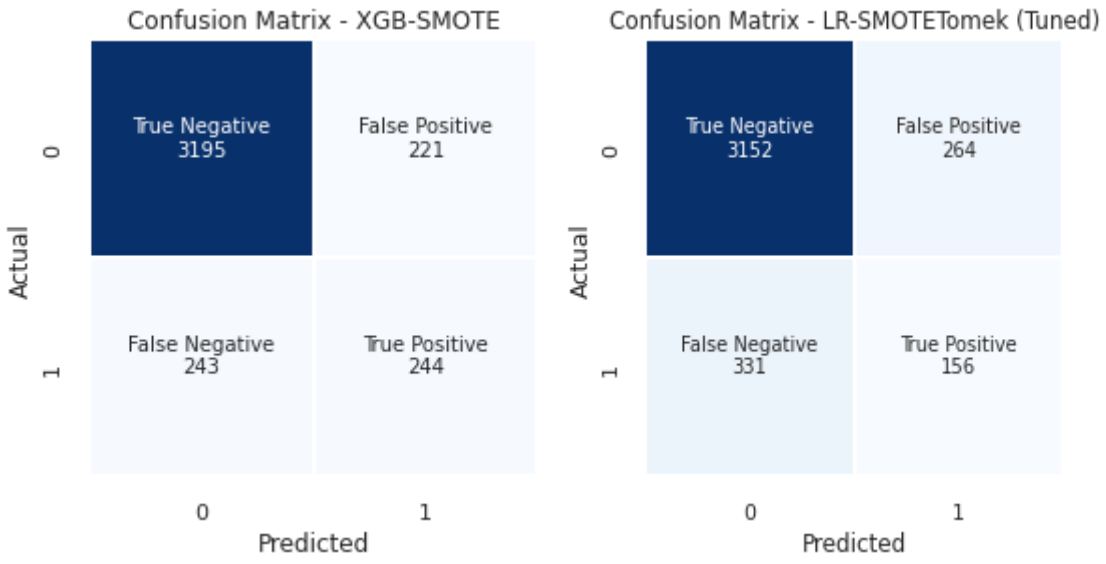


Figure 13. Confusion Matrix Comparison Between LR-SMOTETomek (Tuned) and XGB-SMOTE

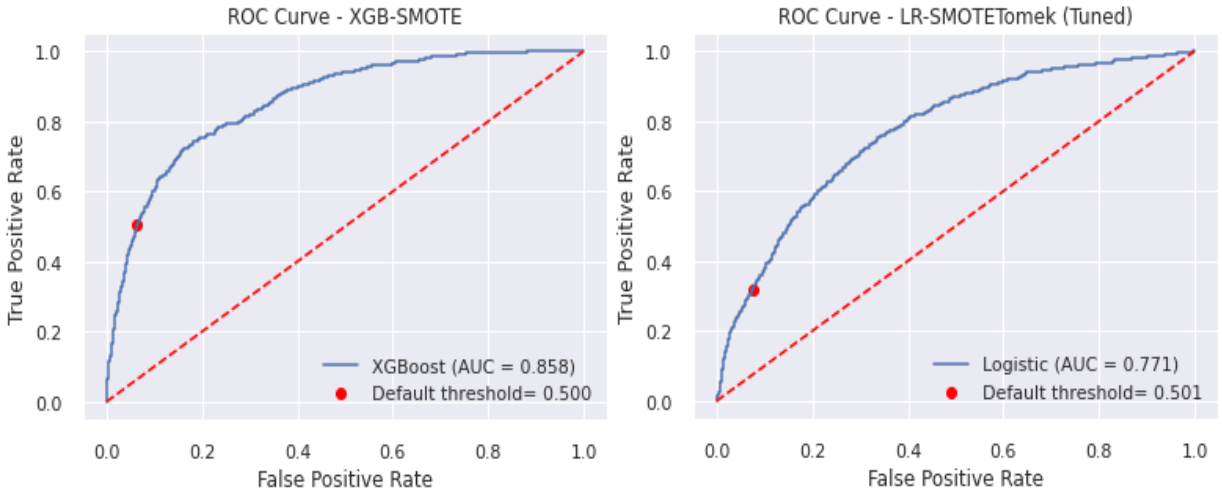


Figure 14. ROC Curve Comparison Between LR-SMOTETomek (Tuned) and XGB-SMOTE

In Figure 15, the precision-recall curve for both models are shown. It can be observed that with the default threshold (red point in the graph), XGB registered a higher recall and higher precision compared to LR. Therefore, XGB showed the best tradeoff between quality (precision) and quantity (recall).

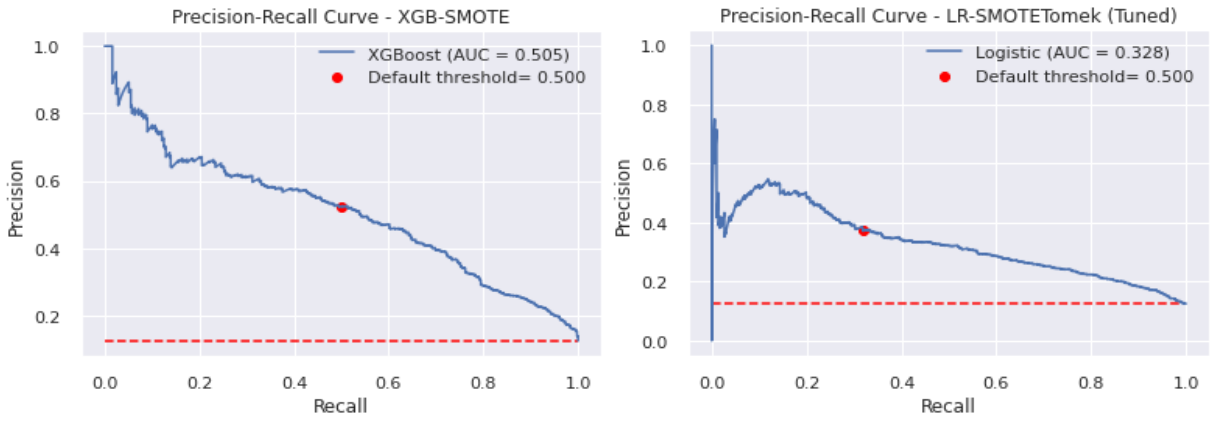


Figure 15. PR Curve Comparison Between LR-SMOTETomek (Tuned) and XGB-SMOTE

Furthermore, Table 12 presents a summary of the metrics for both models. It is noticeable from the metrics and the ROC curve itself that the AUC of XGB-SMOTE is greater than LR-SMOTETomek. Additionally, XGB achieves higher precision and recall while maintaining the lowest FPR possible.

Table 12. Metrics Comparison for Best Models in XGB and LR

Metrics	**XGB SMOTE**	**LR - Tuned SMOTETomek**
accuracy	0.8811	0.8475
precision	0.5247	0.3714
recall	0.501	0.3203
FPR	0.0647	0.0773
AUC	0.8575	0.7712
log-loss	0.2895	0.3737

As a result, we identified that the XGB-SMOTE classifier outperforms LR-SMOTETomek (Tuned). Therefore, XGB was a better model for its accurate prediction of premium products, reducing false positives (FP) and considering the prediction of the minority class (premium). In essence, XGB-SMOTE can be viewed as a better model for predicting whether an item is a premium or non-premium product.

4.8 Optimal Threshold

In this section, we used two methods to find the optimal threshold. First, mathematically with G-mean and F1-Score, and second with Cost-Sensitive Analysis. We evaluated both methods for our best model, which is XGB-SMOTE.

4.8.1 Mathematical Approach

We employed two approaches to finding the optimal threshold: G-mean and F1-Score. These are plotted in the ROC and Precision-Recall Curve (Figure 16).

The ROC curve can be used to find the optimal balance between TPR (sensitivity) and FPR (1-specificity). The geometric mean (G-mean) is a metric used for imbalanced classification. This metric can be used to find the point in the ROC curve that maximizes both sensitivity and specificity (Brownlee, 2022)

$$G - mean = \sqrt{Sensitivity \times Specificity}$$

On the other hand, the F1-Score is a metric that integrates the precision and the recall index into one criterion by using the Harmonic Mean (HM). F Measure overlooks the negative observations and illustrates the tradeoff between quality (precision) and quantity (recall) when classifying positive instances (Fernández et al., 2018, Larner, 2021). This metric can be used to find the point in the precision-recall curve that better integrates both metrics. In algebraic notion:

$$F1 - Score = \frac{2 \times (Precision \times Recall)}{Precision + Recall}$$

Following the G-mean approach, we can see in Figure 16 that the optimal threshold for XGB is 0.255. The TPR and FPR rise from 50% and 6.7% to 74.5% and 18.5%, respectively. While FP increases from 221 to 632 observations, FN decreases from 243 to 124 observations. This optimal threshold obtained by G-mean caused precision to decrease from 52% to 36% in XGB.

In the F1-Score, the TPR and FPR rose from 50% and 6.7% to 63% and 10.7%, accordingly. FP increases from 221 to 366 samples, whereas FN decreases from 243 to 180 samples. Additionally,

this optimum threshold obtained by F Measure induced precision to reduce from 52% to 45% in XGB.

Even though we obtained the optimal threshold using G-mean and F1-Score, which are considered suitable measures for imbalance classification, there is an increase in the recall or TPR but at the same time an increase in FPR in both scenarios. Our test data was not processed (it is still imbalanced between classes), so the increase is notorious when we analyzed FP.

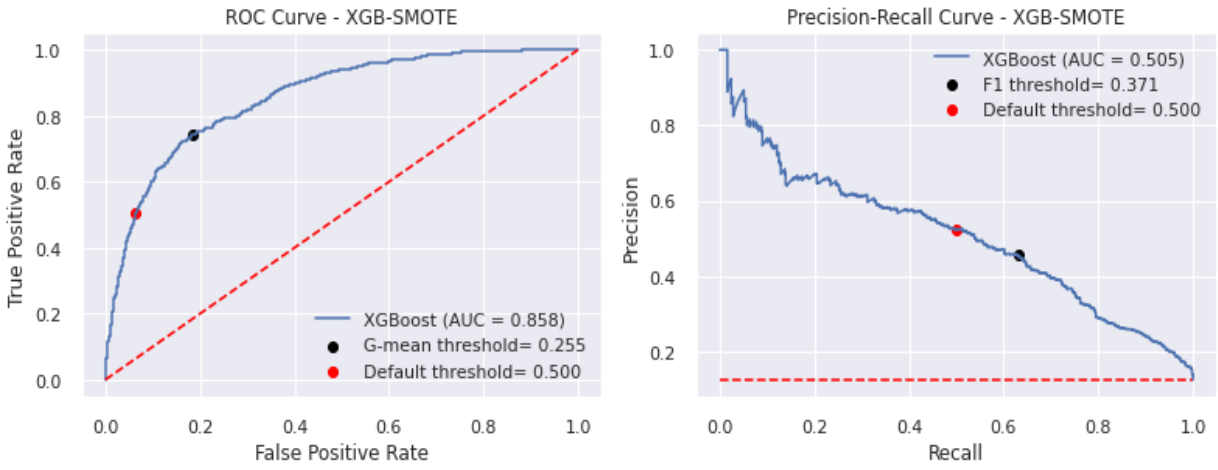


Figure 16. ROC and PR Curve with Default, G-mean, and F1 Thresholds

4.8.2 Cost-Sensitive Analysis

In most classification problems, there is no difference in the cost associated with different misclassification errors (He et al., 2021). However, as mentioned earlier, the deepest concern in this classification problem is the FP. Therefore, we faced a situation in which misclassification errors represent different costs. This can be addressed by using Cost-Sensitive Analysis.

The core idea is to assign different costs/benefits to the different misclassification errors. There are more robust ways to specify the cost that each category should have, e.g., costs can be provided by domain experts who asses the most accurate values. In our case, we used the imbalance ratio obtained in Section 4.1, considering that we do not have more information about the opportunity cost in each scenario (Fernández et al., 2018). As Table 13 illustrates, we assigned a cost of -7 to

classifying a product as premium when it is not (FP) because this is our main concern and a cost of -1. Also, we assigned a benefit 8 to correctly classify products as premium when they are (TP).

Table 13. Confusion Matrix with Cost-Sensitive Analysis

		Predicted Class	
		Negative	Positive
Actual Class	Negative	TN (2)	FP (-7)
	Positive	FN (-1)	TP (8)

We wanted to find the threshold that maximizes the earnings with these costs and benefits. For that, we generated an earnings function for every possible threshold and then found the optimal one. The earnings were calculated as follows

$$Earnings = \frac{(2TN + 8TP - 1FN - 7FP)}{N}$$

The threshold values were taken from the ROC Curve of our best model (XGB-SMOTE), and for each threshold, we considered the corresponding quadrants of the confusion matrix (TN, TP, FN, FP) and calculated the corresponding earnings function. After this procedure, we observed the earnings results against the threshold values in Figure 17. The optimal threshold is placed at the point where the curve reaches its maximum, a threshold of 0.6612, as derived computationally. This is the point that gives us the maximum earnings possible without unnecessarily increasing the FPR.

The results are shown in Table 14 and the corresponding confusion matrix in Figure 18. There was a significant reduction (-57%) in the number of FP compared to the XGB-SMOTE model. If we observe the metrics there was also a remarkable improvement compared to the default threshold. The FPR presents a reduction of 57%, recalled was reduced by 37%, whereas precision increased by 17%.

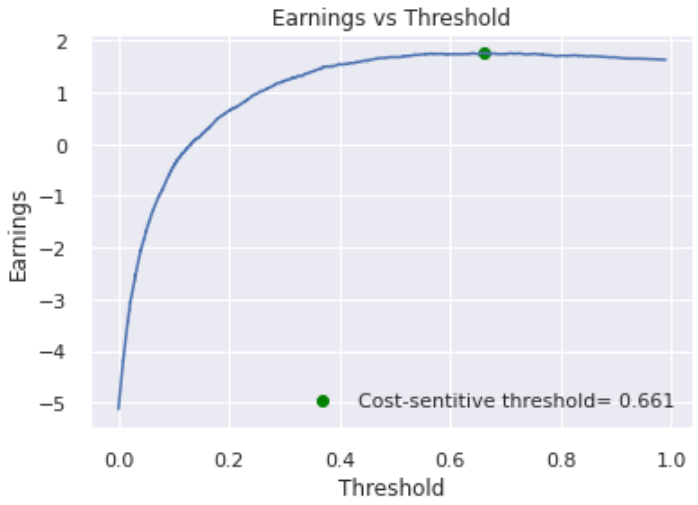


Figure 17. Cost-Sensitive Analysis: Earnings Curve and Thresholds

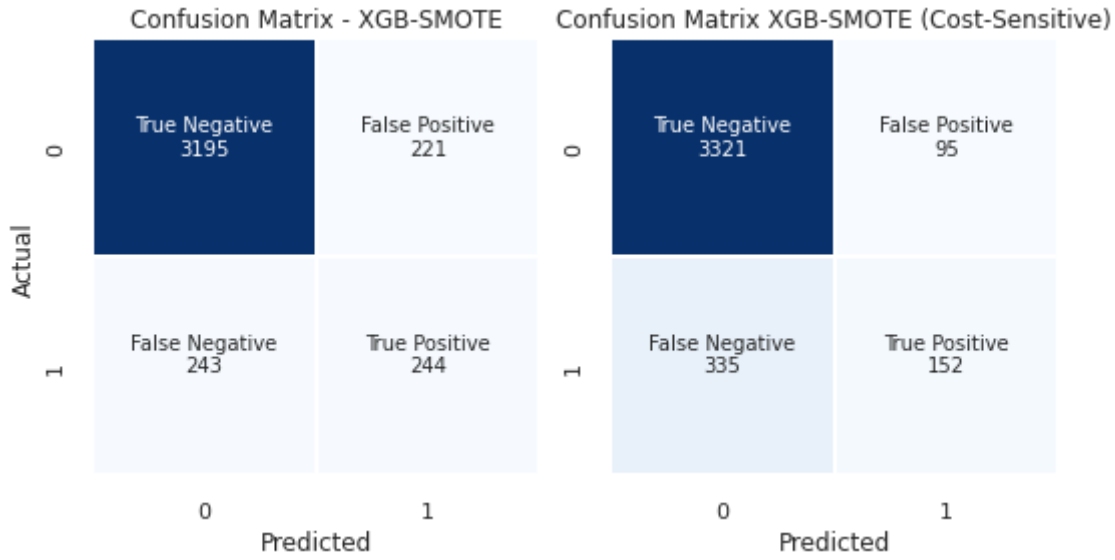


Figure 18. Confusion Matrix Comparison Between XGB-SMOTE and XGB-SMOTE (Cost-Sensitive)

Table 14. Metrics with Cost-Sensitive Analysis

Metrics	XGB-SMOTE Default-Threshold	XGB-SMOTE CS-Threshold
accuracy	0.8811	0.8898
precision	0.5247	0.6154
recall	0.5010	0.3121
FPR	0.0647	0.0278
AUC	0.8575	0.8575
log-loss	0.2895	0.2895

Therefore, we concluded that our best model is XGB-SMOTE with an optimal threshold of 0.6612. The reason for the metrics to improve drastically is that we are considering what our model should care about, reducing FP, by penalizing those, and incentivizing the prediction of TP.

5. Conclusions

The current study provides the application of ML algorithms to predict whether a product is premium or non-premium. We faced a high category imbalance and costly misclassification (especially for premium products).

We introduced the Logistic Regression (LR) and XGBoost (XGB) models as machine learning classifiers. We evaluated the performance of the classification using precision, recall, FPR, AUC, log-loss, and accuracy. However, accuracy was the least relevant since it can be misleading for the highly imbalanced dataset.

The dataset used for this paper had a relatively large amount of data compared to other studies in the food and beverage industry. For building the algorithms, first, we split the dataset into 25% of the data for testing and 75% for training. Then, we tried balancing methods, feature selection, and tuning parameters to find the best possible model for each algorithm. We tested five balancing methods: SMOTE, SMOTEEN, SMOTETomek, Random Undersampling, and Weighted Approach. Afterward, we used three feature selection methods: Correlation Based, Sequential Forward Selection (SFS), and XGB Feature Importance. After that, we tuned LR using Randomized Search CV and XGB using Optuna optimization. We then chose the best model to train and finally evaluated its performance.

Overall, feature selection did not improve the performance of the algorithms, and tuning parameters only improved LR. For LR, the best possible model was found using SMOTETomek as the balancing method, with tuning parameters but without feature selection. For XGB, the best viable model was found with SMOTE as a balancing method, without tuning parameters and using all the variables.

The reduction of incorrectly classified premium products (False Positives) was the critical feature in this study. XGB-SMOTE classifier outperformed LR-SMOTETomek. XGB predicted premium products with higher precision and lower FPR compared to LR.

This study applied the Cost-Sensitive Analysis in XGB to find a model that minimized the False Positives despite the significant class imbalance. Cost-Sensitive XGB was inspired by similar research in other industries, with suitable modifications for the application in this paper's food and beverage problem. This method remarkably improved the results, achieving a precision, recall, and FPR of 61.5%, 31.2%, and 2.7%, respectively.

For future studies, a Cost-Sensitive Analysis could be done by modifying the loss function inside the XGB algorithm itself as a more effective way to address the imbalance of the dataset accurately. Furthermore, one can explore a more robust way to assign the costs for the Cost-Sensitive Analysis.

6. References

Abdulghani, A., UCAN, O. and Alheeti, K., 2021. Credit Card Fraud Detection Using XGBoost Algorithm. *2021 14th International Conference on Developments in eSystems Engineering (DeSE)*.

Akiba, T., Sano, S., Yanase, T., Ohta, T. and Koyama, M., 2019. Optuna. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Batista, G., Prati, R. and Monard, M., 2004. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, 6(1), pp.20-29.

Beleites, C., Neugebauer, U., Bocklitz, T., Krafft, C. and Popp, J., 2013. Sample size planning for classification models. *Analytica Chimica Acta*, 760, pp.25-33.

Bhardwaj, P., Tiwari, P., Olejar, K., Parr, W. and Kulasiri, D., 2022. A machine learning application in wine quality prediction. *Machine Learning with Applications*, 8, p.100261.

Brownlee, J., 2018. *Xgboost with python. Gradient boosted trees with xgboost and scikit-learn*. 1st ed.

Brownlee, J., 2022. *Imbalanced Classification with Python: Better Metrics, Balance Skewed classes, and apply cost-sensitive learning*. 1st ed.

Cai, J., Luo, J., Wang, S. and Yang, S., 2018. Feature selection in machine learning: A new perspective. Neurocomputing, 300, pp.70-79. Available online: https://www.sciencedirect.com/science/article/pii/S092523121830291

Chawla, N., Bowyer, K., Hall, L. and Kegelmeyer, W., 2002. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16, pp.321-357.

Cook, J. and Ramadas, V., 2020. When to consult precision-recall curves. *The Stata Journal: Promoting communications on statistics and Stata*, 20(1), pp.131-148.

De Andrade, B., Margalho, L., Batista, D., Lucena, I., Kamimura, B., Balthazar, C., Brexó, R., Pia, A., Costa, R., Cruz, A., Granato, D., Sant'Ana, A., Luna, A. and de Gois, J., 2022. Chemometric classification of Brazilian artisanal cheeses from different regions according to major and trace elements by ICP-OES. *Journal of Food Composition and Analysis*, 109, p.104519.

Dong, C., Qiao, Y., Shang, C., Liao, X., Yuan, X., Cheng, Q., Li, Y., Zhang, J., Wang, Y., Chen, Y., Ge, Q. and Bao, Y., 2022. Non-contact screening system based for COVID-19 on XGBoost and logistic regression. *Computers in Biology and Medicine*, 141, p.105003.

Fan, R., Chang, K., Hsieh, C., Wang, X., Lin, C., 2008 LIBLINEAR: A library for large linear classification Journal of Machine Learning Research 9, 1871-1874.

FernándezA., GarcíaS., Galar, M., Prati, R.C., Bartosz Krawczyk and Herrera, F. (2018). *Learning from Imbalanced Data Sets*. Cham Springer International Publishing.

Gómez-Meire, S., Campos, C., Falqué, E., Díaz, F. and Fdez-Riverola, F., 2014. Assuring the authenticity of northwest Spain white wine varieties using machine learning techniques. *Food Research International*, 60, pp.230-240.

Granato, D., de Araújo Calado, V. and Jarvis, B., 2014. Observations on the use of statistical methods in Food Science and Technology. *Food Research International*, 55, pp.137-149.

Hajmeer, M. and Basheer, I., 2003. Comparison of logistic regression and neural network-based classifiers for bacterial growth. *Food Microbiology*, 20(1), pp.43-55.

Hall, M., 1999. *Correlation-based Feature Selection for Machine Learning*. Doctor of Philosophy. University of Waikato.

Hastie, T., Tibshirani, R. and Friedman, J., 2017. *The elements of statistical learning*. 2nd ed. Springer.

He, W., He, H., Wang, F., Wang, S., Li, R., Chang, J. and Li, C., 2021a. Rapid and Uninvasive Characterization of Bananas by Hyperspectral Imaging with Extreme Gradient Boosting (XGBoost). *Analytical Letters*, 55(4), pp.620-633.

He, S., Li, B., Peng, H., Xin, J. and Zhang, E., 2021b. An Effective Cost-Sensitive XGBoost Method for Malicious URLs Detection in Imbalanced Dataset. *IEEE Access*, 9, pp.93089-93096.

Hosmer, D., Lemeshow, S. and Sturdivant, R. 2013. *Applied Logistic Regression*. [online] New York, Etc.: John Wiley And Sons, Cop. Available at: https://www.wiley.com/en-us/Applied+Logistic+Regression%2C+3rd+Edition-p-9780470582473.

Jiang, Y., Bian, B., Wang, X., Chen, S., Li, Y. and Sun, Y., 2020. Identification of tomato maturity based on multinomial logistic regression with kernel clustering by integrating color moments and physicochemical indices. *Journal of Food Process Engineering*, 43(10).

Matthews, 1995. *Quantification and the quest for medical certainty*. Princeton, N.J.: Princeton Univ. Press, Cop.

J. Tan, J. Yang, S. Wu, G. Chen, J. Zhao, 2021. A critical look at the current train/test split in machine learning. arXiv -Artificial Intelligence. https://doi.org/arxiv-2106.04525

Krzanowski, W. and Hand, D., 2009. *ROC Curves for Continuous Data*. CRC Press.

Kumar, S., Agrawal, K. and Mandan, N., 2020. Red Wine Quality Prediction Using Machine Learning Techniques. *2020 International Conference on Computer Communication and Informatics (ICCCI)*.

Kuhn, M & Johnson, K 2016, *Applied predictive modeling*, Springer, New York.

Koranga, M., Pandey, R., Joshi, M. and Kumar, M., 2021. Analysis of white wine using machine learning algorithms. *Materials Today: Proceedings*, 46, pp.11087-11093.

Larner, A. 2021. *2X2 MATRIX : contingency, confusion and the metrics of binary classification.* S.L.: Springer Nature.

Liu, D. and Nocedal, J., 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1-3), pp.503-528.

Ma, P., Li, A., Yu, N., Li, Y., Bahadur, R., Wang, Q. and Ahuja, J., 2021. Application of machine learning for estimating label nutrients using USDA Global Branded Food Products Database, (BFPD). *Journal of Food Composition and Analysis*, 100, p.103857.

Mahima, Gupta, U., Patidar, Y., Agarwal, A. and Singh, K., 2020. Wine Quality Analysis Using Machine Learning Algorithms. *Micro-Electronics and Telecommunication Engineering*, pp.11-18.

Monforte, A., Martins, S. and Silva Ferreira, A., 2021. Discrimination of white wine ageing based on untarget peak picking approach with multi-class target coupled with machine learning algorithms. *Food Chemistry*, 352, p.129288.

Mu, F., Gu, Y., Zhang, J. and Zhang, L., 2020. Milk Source Identification and Milk Quality Estimation Using an Electronic Nose and Machine Learning Techniques. *Sensors*, 20(15), p.4238.

Oliveira Chaves, L., Gomes Domingos, A., Louzada Fernandes, D., Ribeiro Cerqueira, F., Siqueira-Batista, R. and Bressan, J., 2021. Applicability of machine learning techniques in

food intake assessment: A systematic review. *Critical Reviews in Food Science and Nutrition*, pp.1-18.

Ozenne, B., Subtil, F. and Maucort-Boulch, D., 2015. The precision–recall curve overcame the optimism of the receiver operating characteristic curve in rare diseases. *Journal of Clinical Epidemiology*, 68(8), pp.855-859.

Quelch, J., 1987. Marketing the premium product. *Business Horizons*, 30(3), pp.38-45. Available online:https://www.sciencedirect.com/science/article/pii/0007681387900358

Ratkowsky, D. and Ross, T., 1995. Modelling the bacterial growth/no growth interface. *Letters in Applied Microbiology*, 20(1), pp.29-33.

Rodríguez-Saavedra, M., Pérez-Revelo, K., Valero, A., Moreno-Arribas, M. and González de Llano, D., 2021. A Binary Logistic Regression Model as a Tool to Predict Craft Beer Susceptibility to Microbial Spoilage. *Foods*, 10(8), p.1926.

Royer, C., O'Neill, M. and Wright, S., 2019. A Newton-CG algorithm with complexity guarantees for smooth unconstrained optimization. *Mathematical Programming*, 180(1-2), pp.451-488.

Saberioon, M., Císař, P., Labbé, L., Souček, P., Pelissier, P. and Kerneis, T., 2018. Comparative Performance Analysis of Support Vector Machine, Random Forest, Logistic Regression and k-Nearest Neighbours in Rainbow Trout (Oncorhynchus Mykiss) Classification Using Image-Based Features. *Sensors*, 18(4), p.1027.

Saito, T. and Rehmsmeier, M. 2015. The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *PLOS ONE*, 10(3), p.e0118432. doi:10.1371/journal.pone.0118432

Schapire, R. and Freund, Y., 2014. *Boosting: foundations and algorithms*. Cambridge: The MIT Press.

Schober, P., Boer, C. and Schwarte, L., 2018. Correlation Coefficients. *Anesthesia & Analgesia*, 126(5), pp.1763-1768.

Schmidt, M., Le Roux, N. and Bach, F., 2016. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2), pp.83-112.

Stigler, S., 2002. The missing early history of contingency tables. *Annales de la faculté des sciences de Toulouse Mathématiques*, 11(4), pp.563-573.

Teye, E., Amuah, C., McGrath, T. and Elliott, C., 2019. Innovative and rapid analysis for rice authenticity using hand-held NIR spectrometry and chemometrics. *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy*, 217, pp.147-154.

Trivedi, A. and Sehrawat, R., 2018. Wine Quality Detection through Machine Learning Algorithms. *2018 International Conference on Recent Innovations in Electrical, Electronics & Communication Engineering (ICRIEECE)*.

Wang, X., Bouzembrak, Y., Lansink, A. and Fels-Klerx, H., 2021a. Application of machine learning to the monitoring and prediction of food safety: A review. *Comprehensive Reviews in Food Science and Food Safety*, 21(1), pp.416-434.

Wang, Z., He, Y., Zhang, A., Zhang, J., Liu, H., Shi, P. and Han, R., 2021b. Product Key Reliability Characteristics Identification Method Based on XGBoost in Manufacturing Process. *2021 Global Reliability and Prognostics and Health Management (PHM-Nanjing)*.

Wade, C. and Glynn, K., 2020. *Hands-On Gradient Boosting with XGBoost and scikit-learn*. Packt publishing, pp.221-241.

Xgboost.readthedocs.io. 2022. *XGBoost Parameters — xgboost 2.0.0-dev documentation*. [online] Available at: <https://xgboost.readthedocs.io/en/latest/parameter.html> [Accessed 24 May 2022].

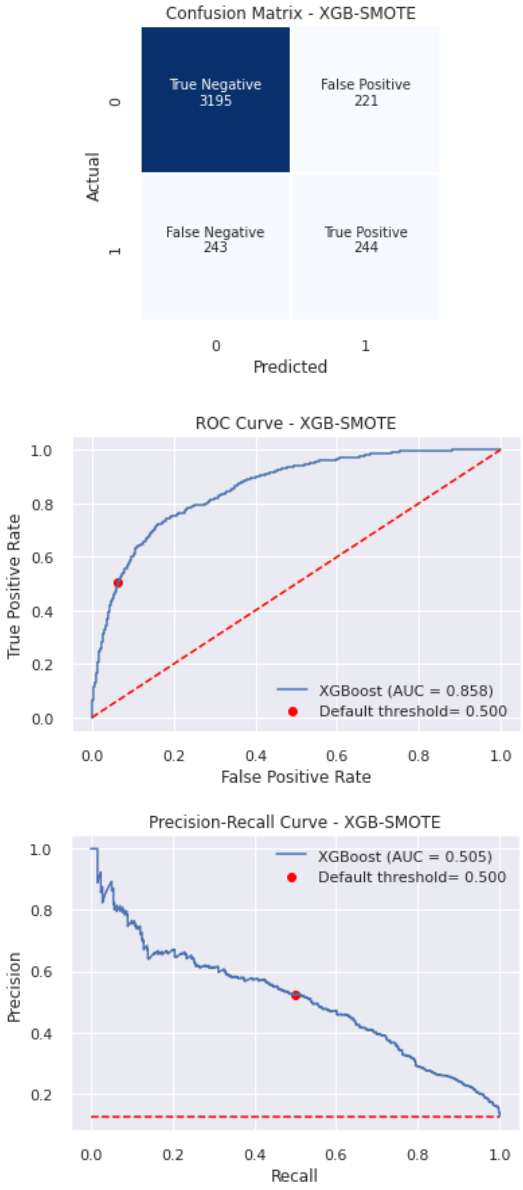
Xu, J., Tang, B., He, H. and Man, H., 2017. Semisupervised Feature Selection Based on Relevance and Redundancy Criteria. *IEEE Transactions on Neural Networks and Learning Systems*, 28(9), pp.1974-1984.

Yao, K., Sun, J., Chen, C., Xu, M., Zhou, X., Cao, Y. and Tian, Y., 2022. Non-destructive detection of egg qualities based on hyperspectral imaging. *Journal of Food Engineering*, 325, p.111024.

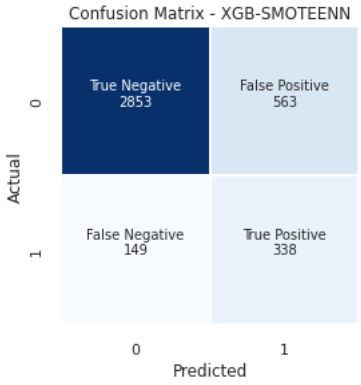
Yang, X., Deb, S., Loomes, M. and Karamanoglu, M., 2013. A framework for self-tuning optimization algorithm. *Neural Computing and Applications*, 23(7-8), pp.2051-2057.

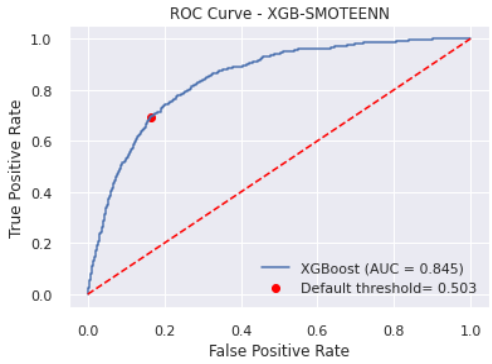
7. Appendices

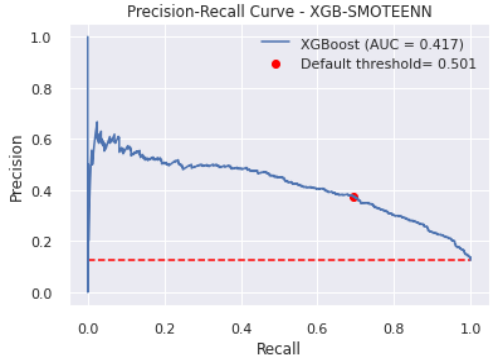
Appendix 1. Confusion Matrix, ROC, and PR Curve for XGB-SMOTE.



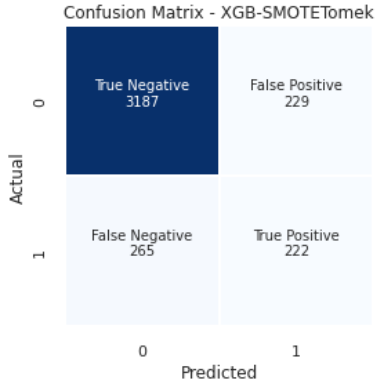
Appendix 2. Confusion Matrix, ROC, and PR Curve for XGB-SMOTEENN

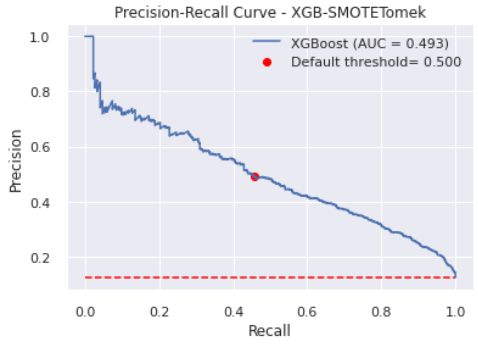




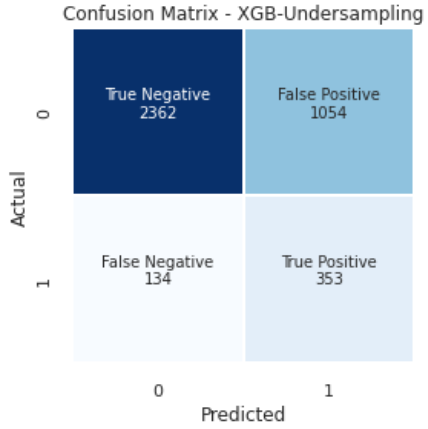


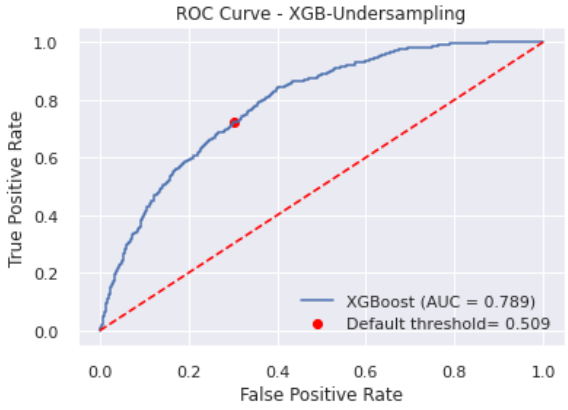
Appendix 3. Confusion Matrix, ROC, and PR Curve for XGB-SMOTETomek.

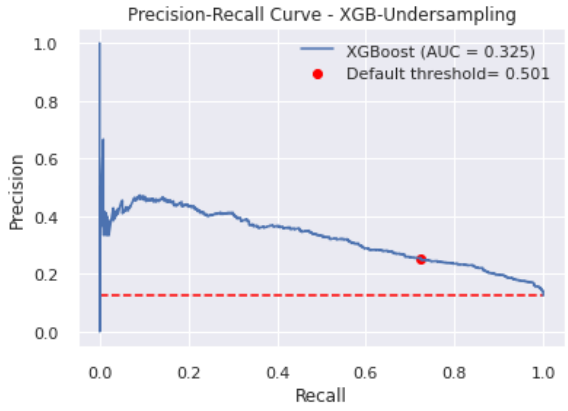




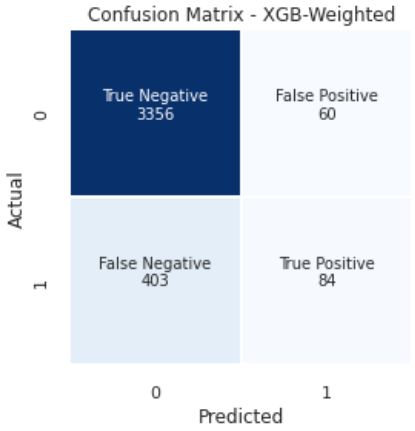
Appendix 4. Confusion Matrix, ROC, and PR Curve for XGB-Undersampling.

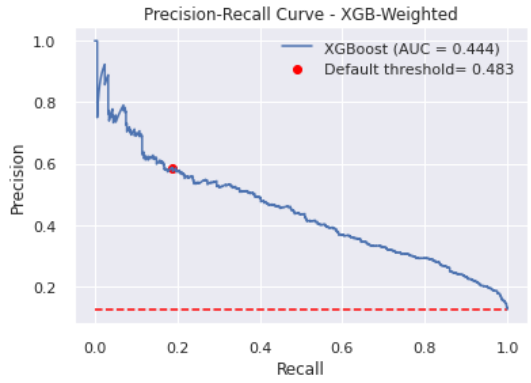




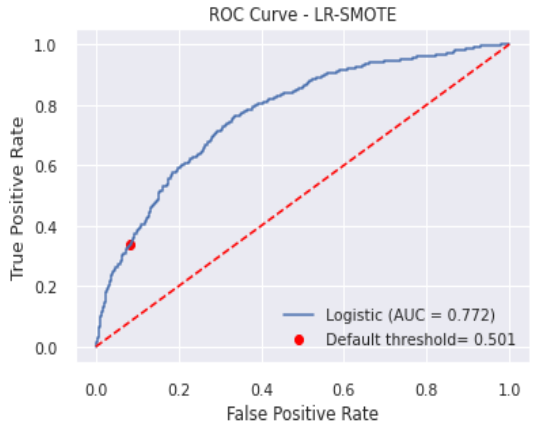


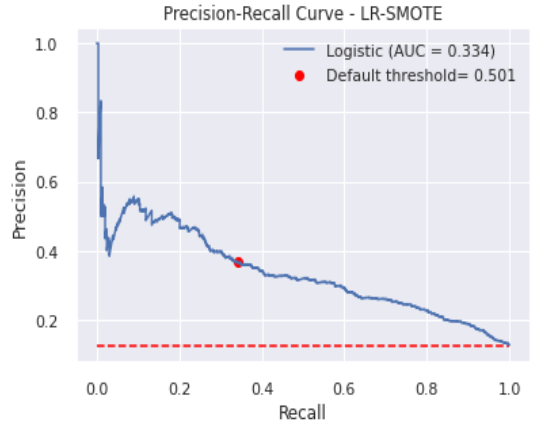
Appendix 5. Confusion Matrix, ROC, and PR Curve for XGB-Weighted.



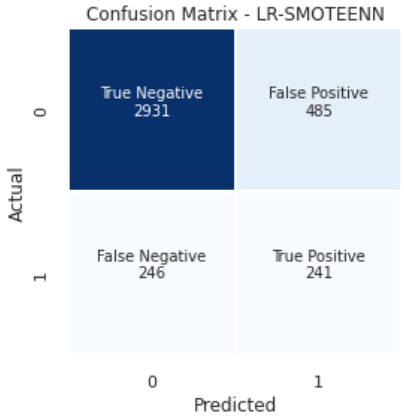


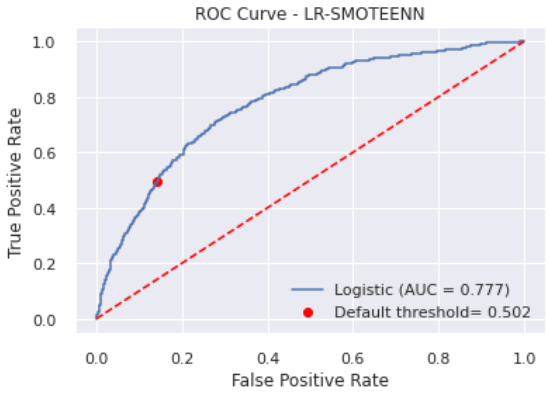
Appendix 6. Confusion Matrix, ROC, and PR Curve for LR-SMOTE.

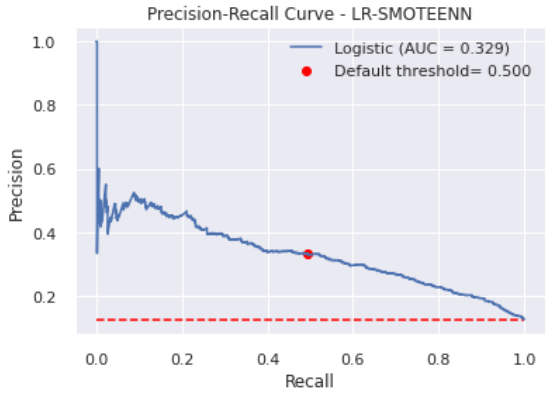




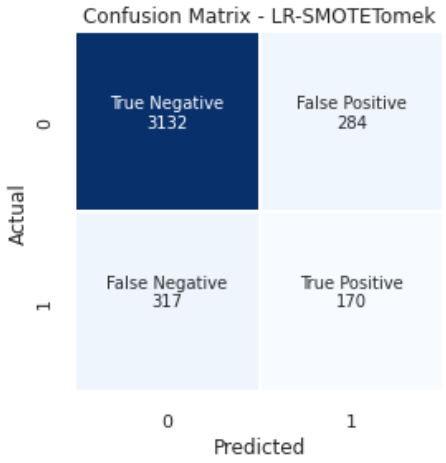
Appendix 7. Confusion Matrix, ROC, and PR Curve for LR-SMOTEENN.

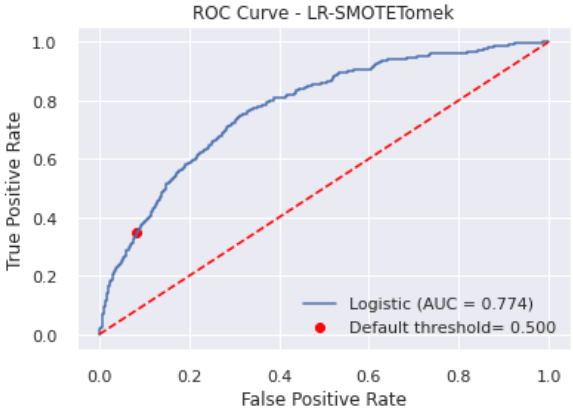


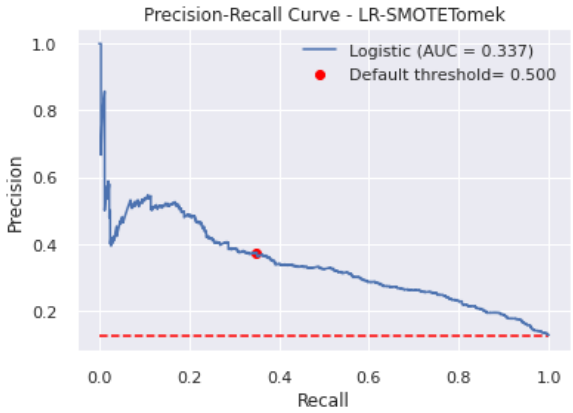




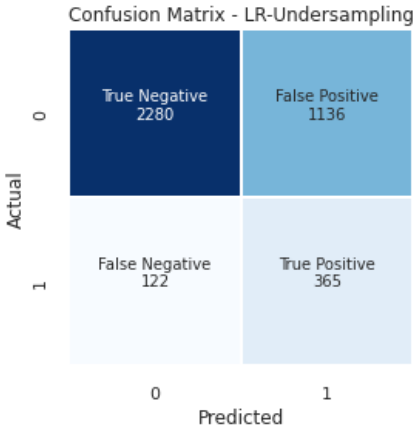
Appendix 8. Confusion Matrix, ROC, and PR Curve for LR-SMOTETomek.

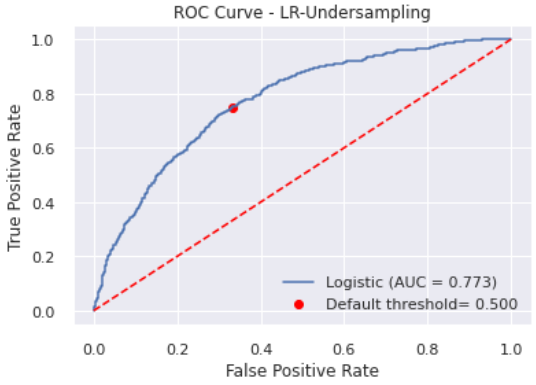


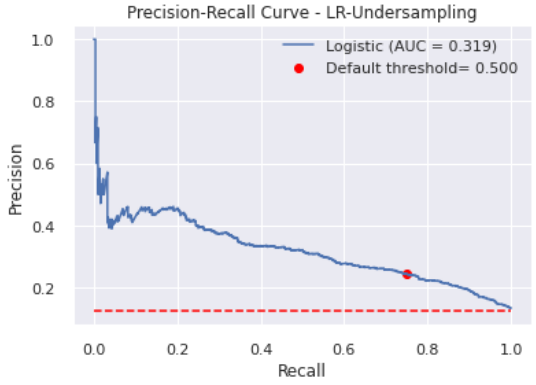




Appendix 10. Confusion Matrix, ROC, and PR Curve for LR-Undersampling.







Appendix 10. Confusion Matrix, ROC, and PR Curve for LR-Weighted.