technologies

MDPI

Review

Hyperparameter Optimization and Combined Data Sampling Techniques in Machine Learning for Customer Churn Prediction: A Comparative Analysis

Mehdi Imani [1,*] and Hamid Reza Arabnia [2,*]

[1] Department of Computer and System Sciences, Stockholm University, 10691 Stockholm, Sweden
[2] School of Computing, University of Georgia, Athens, GA 30602, USA
* Correspondence: m.imani@gmail.com (M.I.); hra@uga.edu (H.R.A.)

Abstract: This paper explores the application of various machine learning techniques for predicting customer churn in the telecommunications sector. We utilized a publicly accessible dataset and implemented several models, including Artificial Neural Networks, Decision Trees, Support Vector Machines, Random Forests, Logistic Regression, and gradient boosting techniques (XGBoost, LightGBM, and CatBoost). To mitigate the challenges posed by imbalanced datasets, we adopted different data sampling strategies, namely SMOTE, SMOTE combined with Tomek Links, and SMOTE combined with Edited Nearest Neighbors. Moreover, hyperparameter tuning was employed to enhance model performance. Our evaluation employed standard metrics, such as Precision, Recall, F1-score, and the Receiver Operating Characteristic Area Under Curve (ROC AUC). In terms of the F1-score metric, CatBoost demonstrates superior performance compared to other machine learning models, achieving an outstanding 93% following the application of Optuna hyperparameter optimization. In the context of the ROC AUC metric, both XGBoost and CatBoost exhibit exceptional performance, recording remarkable scores of 91%. This achievement for XGBoost is attained after implementing a combination of SMOTE with Tomek Links, while CatBoost reaches this level of performance after the application of Optuna hyperparameter optimization.

Keywords: machine learning; churn prediction; imbalanced data; combined data sampling techniques; hyperparameter optimization

Citation: Imani, M.; Arabnia, H.R. Hyperparameter Optimization and Combined Data Sampling Techniques in Machine Learning for Customer Churn Prediction: A Comparative Analysis. *Technologies* **2023**, *11*, 167. https://doi.org/10.3390/technologies11060167

Academic Editors: Mohammed Mahmoud and Sikha Bagui

Received: 17 August 2023
Revised: 17 November 2023
Accepted: 21 November 2023
Published: 26 November 2023

1. Introduction

The implementation of Customer Relationship Management (CRM) is a strategic approach to managing and enhancing relationships between businesses and their customers. CRM is a tool employed to gain deeper insights into the requirements and behaviors of consumers, specifically end users, with the aim of fostering a more robust and meaningful relationship with them. Through the utilization of CRM, businesses can establish an infrastructure that fosters long-term and loyal customers. This concept is relevant across various industries, such as banking [1–4], insurance companies [5], and telecommunications [6–14], to name a few.

The telecommunications sector assumes a prominent role as a leading industry in revenue generation and a crucial driver of socioeconomic advancement in numerous countries globally. It is estimated that this sector incurs expenditures of approximately 4.7 trillion dollars annually [1,2]. Within the sector, there exists a high degree of competition among companies, driven by their pursuit of augmenting revenue streams and expanding the market influence through the acquisition of an expanded customer base. A key objective of CRM is customer retention, as studies have demonstrated that the cost of acquiring new customers can be 20 times higher than retaining existing ones [1]. Therefore, maintaining existing customers in the telecommunications industry is crucial for increasing revenue and reducing marketing and advertising costs.

The telecommunications sector is grappling with the substantial issue of customer attrition, commonly referred to as churn. This escalating issue has prompted service providers to shift their emphasis from acquiring new customers to retaining existing ones, considering the significant costs associated with customer acquisition. In recent years, service providers have been progressively emphasizing the establishment of enduring relationships with their customers. Consequently, these providers uphold CRM databases wherein every customer-specific interaction is systematically documented [5]. CRM databases serve as valuable resources for proactively predicting and addressing customer requirements by leveraging a combination of business processes and machine learning (ML) methodologies to analyze and understand customer behavior.

The primary goal of ML models is to predict and categorize customers into one of two groups: churn or non-churn, representing a binary classification problem. As a result, it is imperative for businesses to develop practical tools to achieve this goal. In recent years, various ML methods have been proposed for constructing a churn model, including Decision Trees (DTs) [8–16], Artificial Neural Networks (ANNs) [8,9,15–17], Random Forests (RFs) [18,19], Logistic Regression (LR) [9,12], Support Vector Machines (SVMs) [16], and a Rough Set Approach [20], among others.

In the following, an overview is provided of the most frequently utilized techniques for addressing the issue of churn prediction, including Artificial Neural Networks, Decision Trees, Support Vector Machines, Random Forests, Logistic Regression, and three advanced gradient boosting techniques, namely eXtreme Gradient Boosting (XGBoost), Categorical Boosting (CatBoost) and Light Gradient Boosting Machine (LightGBM).

Ensemble techniques [21], specifically boosting and bagging algorithms, have become the prevalent choice for addressing classification problems [22,23], particularly in the realm of churn prediction [24,25], due to their demonstrated high effectiveness. While many studies have explored the field of churn prediction, our research distinguishes itself by offering a comprehensive examination of how machine learning techniques, imbalanced data, and predictive accuracy intersect.

We carefully investigate a wide range of machine learning algorithms, along with innovative data sampling methods and precise hyperparameter optimization techniques. The objective is to offer subscription-based companies a comprehensive framework for effectively tackling the complex task of predicting customer churn. In the current data-centric business environment, the relevance of this study is not only significant but also imperative. It equips subscription-based businesses with the tools to retain customers, optimize revenue, and develop lasting relationships with their customers in the face of evolving industry dynamics. This study makes several significant contributions, including the following:

1. Providing a comprehensive definition of binary classification machine learning techniques tailored for imbalanced data.
2. Conducting an extensive review of diverse sampling techniques designed to address imbalanced data.
3. Offering a detailed account of the training and validation procedures within imbalanced domains.
4. Explaining the key evaluation metrics that are well-suited for imbalanced data scenarios.
5. Employing various machine learning models and conducting a thorough assessment, comparing their performance using commonly employed metrics across three distinct phases: after applying feature selection, after applying SMOTE, after applying SMOTE combined with Tomek Links, after applying SMOTE combined with ENN, and after applying Optuna hyperparameter tuning.

Table 1, below, shows a summary of the important acronyms used throughout this paper.

Table 1. Summary of important acronyms.

Acronym	Meaning
ANN	Artificial Neural Network
AUC	Area Under the Curve
BPN	Back-Propagation Network
CatBoost	Categorical Boosting
CNN	Condensed Nearest Neighbor
DT	Decision Tree
ENN	Edited Nearest Neighbor
LightGBM	Light Gradient Boosting Machine
LR	Logistic Regression
ML	Machine Learning
RF	Random Forest
ROC	Receiver Operating Characteristic
SMOTE	Synthetic Minority Over-Sampling Technique
SVM	Support Vector Machine
XGBoost	eXtreme Gradient Boosting

The remainder of the paper is organized as follows: Section 2 presents an introduction to classification machine learning techniques, Section 3 delves into the examination of sampling methods, Section 4 explains the training and validation process, Section 5 defines evaluation metrics, simulation results are presented in Section 6, and the paper concludes in Section 7.

2. Classification of Machine Learning Techniques

2.1. Artificial Neural Network

An Artificial Neural Network (ANN) is a widely employed technique for addressing complex issues, such as the churn-prediction problem [26]. ANNs are structures composed of interconnected units that are modeled after the human brain. They can be utilized with various learning algorithms to enhance the machine learning process and can take both hardware and software forms. One of the most widely utilized models is the Multi-Layer Perceptron, which is trained using the Back-Propagation Network (BPN) algorithm. Research has demonstrated that ANNs possess superior performance compared to Decision Trees (DTs) [26] and have been shown to exhibit improved performance when compared to Logistic Regression (LR) and DTs in the context of churn prediction [27].

2.2. Support Vector Machine

The technique of Support Vector Machine (SVM) was first introduced by the authors in [28]. It is classified as a supervised learning technique that utilizes learning algorithms to uncover latent patterns within data. A popular method for improving the performance of SVMs is the utilization of kernel functions [8]. In addressing customer churn problems, SVM may exhibit superior performance in comparison to Artificial Neural Networks (ANNs) and Decision Trees (DTs) based on the specific characteristics of the data [16,29].

For this study, we utilized both the Gaussian Radial Basis kernel function (RBF-SVM) and the Polynomial kernel function (Poly-SVM) for the Support Vector Machine (SVM) technique. These kernel functions are among the various options available for use with SVM.

For two samples x and x′, the RBF kernel is defined as follows:

$$K(x.\, x') = exp\left(-\frac{\|x - x'\|^2}{2\delta^2}\right) \quad (1)$$

where $\|x - x'\|^2$ can be the squared Euclidean distance, and δ is a free parameter.

For two samples x and x′, the d-degree polynomial kernel is defined as follows:

$$K(x.x') = \left(x^T x' + c\right)^d \quad (2)$$

where $c \geq 0$ and $d \geq 1$ is the polynomial degree.

2.3. Decision Tree

A Decision Tree is a representation of all potential decision pathways in the form of a tree structure [30,31]. As Berry and Linoff stated, "a Decision Tree is a structure that can be used to divide up a large collection of records into successively smaller sets of records by applying a sequence of simple decision rules" [32]. Though they may not be as efficient in uncovering complex patterns or detecting intricate relationships within data, DTs may be used to address the customer churn problem, depending on the characteristics of the data. In DTs, class labels are indicated by leaves, and the conjunctions between various features are represented by branches.

2.4. Logistic Regression

Logistic Regression (LR) is a classification method that falls under the category of probabilistic statistics. It can be employed to address the churn-prediction problem by making predictions based on multiple predictor variables. In order to obtain high accuracy, which can sometimes be comparable to that of Decision Trees [9], it is often beneficial to apply pre-processing and transformation techniques to the original data prior to utilizing LR.

2.5. Ensemble Learning

Ensemble learning is one of the widely utilized techniques in machine learning for combining the outputs of multiple learning models (often referred to as base learners) into a single classifier [33]. In ensemble learning, it is possible to combine various weak machine learning models (base learners) to construct a stronger model with more accurate predictions [21,22]. Currently, ensemble learning methods are widely accepted as a standard choice for enhancing the accuracy of machine learning predictors [22]. Bagging and boosting are two distinct types of ensemble learning techniques that can be utilized to improve the accuracy of machine learning predictors [21].

2.5.1. Bagging

As depicted in Figure 1, in the bagging technique, the training data are partitioned into multiple subset sets, and the model is trained on each subset. The final prediction is then obtained by combining all individual outputs through majority voting (in classification problems) or average voting (in regression problems) [21,34–36].

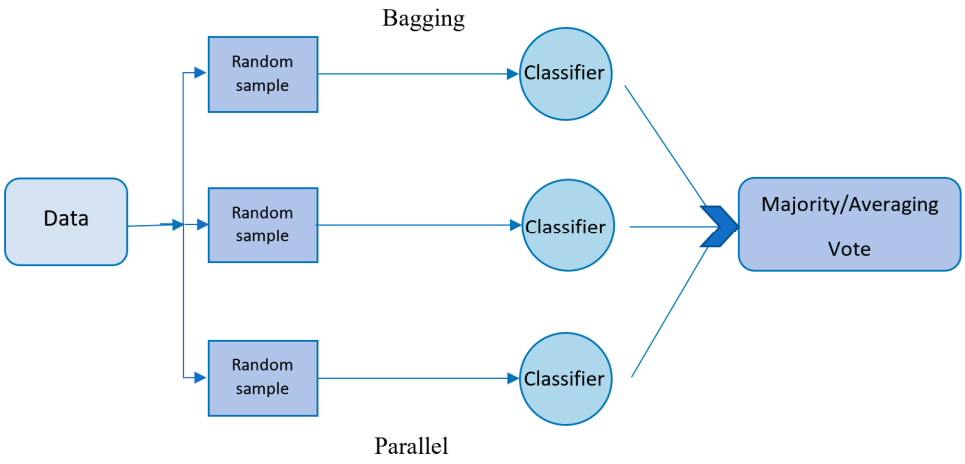

Figure 1. Visualization of the bagging approach.

Random Forest

The concept of Random Forest was first introduced by Ho in 1995 [18] and has been the subject of ongoing improvements by various researchers. One notable advancement in this field was made by Leo Breiman in 2001 [19]. Random Forests are an ensemble learning technique for classification tasks that employs a large number of Decision Trees in the training model. The output of Random Forests is a class that is selected by the majority of the trees, as shown in Figure 2. In general, Random Forests exhibit superior performance compared to Decision Trees. However, the performance can be influenced by the characteristics of the data.

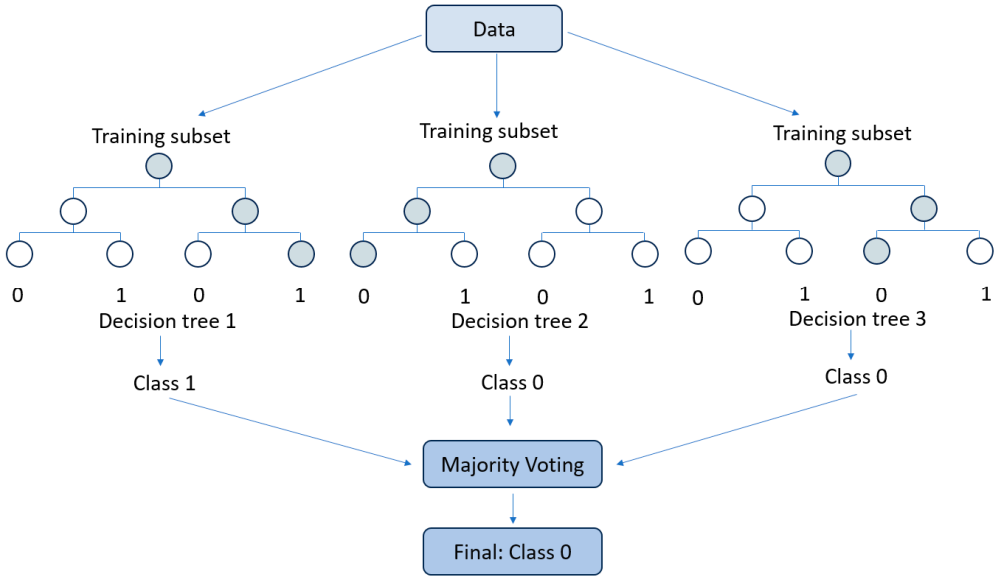

Figure 2. Visualization of the Random Forest classifier.

Random Forests utilize the bagging technique for their training algorithm. In greater detail, the Random Forests operate as follows: for a training set $TS_n = \{(x_1.y_1). \cdots .(x_n.y_n)\}$, bagging is repeated B times, and each iteration selects a random sample with a replacement from TS_n and fits trees to the samples:

1. Sample n training examples, $X_b.Y_b$.
2. Train a classification tree (in the case of churn problems) f_b on the samples $X_b.Y_b$.

After the training phase, Random Forests can predict unseen samples x' by taking the majority vote from all the individual classification trees x'.

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x') \quad (3)$$

2.5.2. Boosting

Boosting is another method for combining multiple base learners to construct a stronger model with more accurate predictions. The key distinction between bagging and boosting is that bagging uses a parallel approach to combine weak learners, while boosting methods utilize a sequential approach to combine weak learners and derive the final prediction, as shown in Figure 3. Like the bagging technique, boosting improves the performance of machine learning predictors, and in addition, it reduces the bias of the model [21].

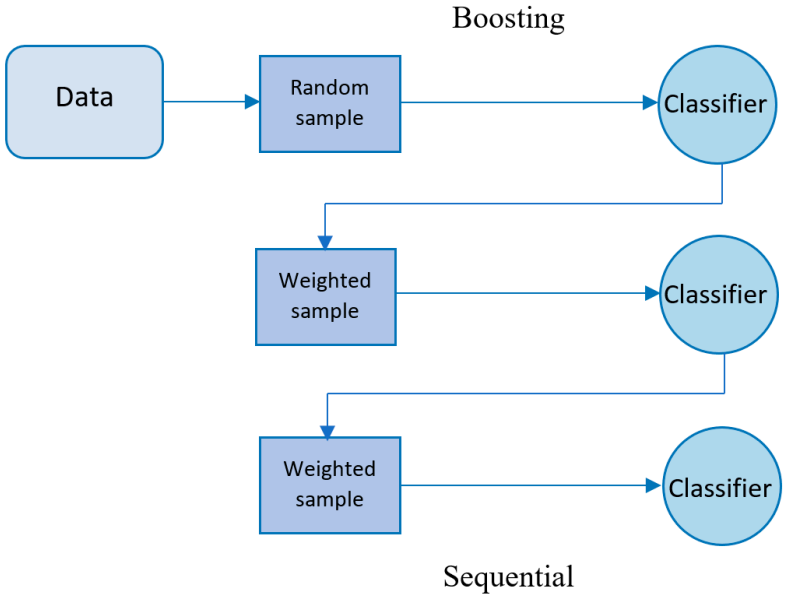

Figure 3. Visualization of the boosting approach.

The Famous Trio: XGBoost, LightGBM, and CatBoost

Recently, researchers have presented three effective gradient-based approaches using Decision Trees: CatBoost, LightGBM, and XGBoost. These new approaches have demonstrated successful applications in academia, industry, and competitive machine learning [37]. Utilizing gradient boosting techniques, solutions can be constructed in a stagewise manner, and the over-fitting problem can be addressed through the optimization of loss functions. For example, given a loss function $\psi(y, f(x))$ and a custom base-learner $h(x, \theta)$ (e.g., Decision Tree), the direct estimation of parameters can be challenging. Thus, an iterative model is proposed, which is updated at each iteration with the selection of a new base-learner function $h(x, \theta t)$, where the increment is directed by the following:

$$g_t(x) = E_y\left[\frac{\partial \psi(y, f(x))}{\partial f(x)}|x\right]_{f(x)=\tilde{f}^{t-1}(x)} \quad (4)$$

Hence, the hard optimization problem is substituted with the typical least-squares optimization problem:

$$(p_t, \theta_t) = \arg\min_{p,\theta} \sum_{i=1}^{N} [-g_t(x_i) + ph(x_i, \theta)]^2 \quad (5)$$

Friedman's gradient boost algorithm is summarized by Algorithm 1.

Algorithm 1 Gradient Boost

1: *Let* $\hat{f}_0$ *be a constant*
2: *For* $i = 1$ *to* M
a. Compute $g_i(x)$ using eq()
b. Train the function $h(x, \theta_i)$
c. Find p_i using eq()
d. Update the function

$$\hat{f}_i = \hat{f}_{i-1} + p_i h(x, \theta_i)$$

3: *End*

After initiating the algorithm with a single leaf, the learning rate is optimized for each record and each node [38–40]. The XGBoost method is a highly flexible, versatile,

and scalable tool that has been developed to effectively utilize resources and overcome the limitations of previous gradient boosting methods. The primary distinction between other gradient boosting methods and XGBoost is that XGBoost utilizes a new regularization approach for controlling overfitting, making it more robust and efficient when the model is fine-tuned. To regularize this approach, a new term is added to the loss function as follows:

$$L(f) = \sum_{i=1}^{n} L(\hat{y}_i, y_i) + \sum_{m=1}^{M} \Omega(\delta_m) \quad (6)$$

with

$$\Omega(\delta) = \alpha|\delta| + 0.5\beta||w||^2$$

where w represents the value of each leaf, Ω indicates the regularization function, and $|\delta|$ denotes the number of branches. A new gain function is used by XGBoost, as follows:

$$G_j = \sum_{i \in I_j} g_i \quad (7)$$

$$H_j = \sum_{i \in I_j} h_i$$

$$\text{Gain} = \frac{1}{2}\left[\frac{G_L^2}{H_L + \beta} + \frac{G_R^2}{H_R + \beta} - \frac{(G_R + G_L)^2}{H_R + H_L + \beta}\right] - \alpha$$

where

$$g_i = \partial_{\hat{y}_i} L(\hat{y}_i + y_i)$$

and

$$h_i = \partial^2_{\hat{y}_i} L(\hat{y}_i + y_i)$$

The Gain represents the score of the no new child case, H indicates the score of the left child, and G denotes the score of the right child [41].

To decrease the implementation time, the LightGBM method was developed by a team from Microsoft in April 2017 [42]. The primary difference is that LightGBM Decision Trees are constructed in a leaf-wise manner, rather than evaluating all previous leaves for each new leaf (Figure 4a,b). The attributes are grouped and sorted into bins, known as the histogram implementation. LightGBM offers several benefits, including a faster training speed, higher accuracy, and the ability to handle large scale data and support GPU learning.

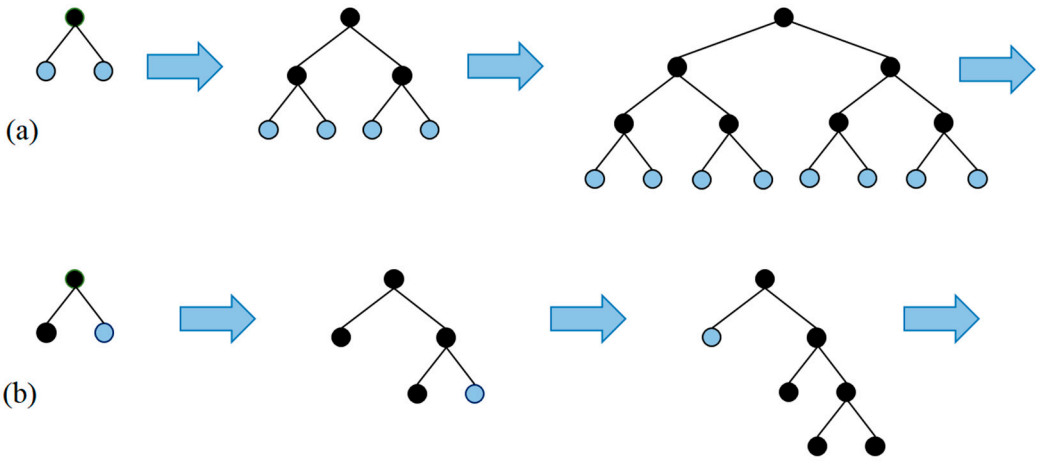

Figure 4. Comparison of tree growth methods. (**a**) XGBoost Level-wise tree growth. (**b**) LightGBM Leaf-wise tree growth.

The focus of CatBoost is on categorical columns through the use of permutation methods, target-based statistics, and one_hot_max_size (OHMS). By using a greedy technique at each new split of the current tree, CatBoost has the capability to address the exponential

growth of feature combinations. The steps described below are employed by CatBoost for each feature with more categories than the OHMS (an input parameter):

1. To randomly divide the records into subsets,
2. To convert the labels to integer numbers,
3. To transform the categorical features to numerical features, as follows:

$$\text{avgTarget} = \frac{\text{countInClass} + \text{prior}}{\text{totalCount} + 1} \quad (8)$$

where totalCount denotes the number of previous objects, countInClass represents the number of ones in the target for a specific categorical feature, and the starting parameters specify prior [43].

3. Handling Imbalanced Data

Imbalanced data are a prevalent problem in data mining. For instance, in binary classifications, the number of instances in the majority class may be significantly higher than the number of instances in the minority class. As a result, the ratio of instances in the minority class to instances in the majority class (imbalanced ratio) may vary from 1:2 to 1:1000. The dataset used in this study is imbalanced, with the distribution of majority class (non-churned) instances being six times that of the minority class (churned) instances [44].

3.1. The Challenge of Imbalanced Data

While imbalanced datasets can skew overall model performance metrics towards the majority class, the more nuanced challenge lies in how specific algorithms inherently respond to this imbalance. For example, Support Vector Machines (SVMs) inherently aim to find a hyperplane that delineates classes by maximizing the margin. However, with imbalanced datasets, the sheer volume of majority class instances can push this hyperplane in a way that does not genuinely represent the optimal boundary, especially from the perspective of the minority class.

In a similar vein, Decision Tree algorithms, which seek to achieve node purity through recursive partitioning, can end up favoring the majority class. In imbalanced contexts, the tree's terminal nodes might predominantly represent the majority class, leading to compromised predictive accuracy for the minority instances.

Addressing these algorithmic biases necessitates approaches beyond mere accuracy metrics. Techniques, like sampling, which adaptively adjust the class distribution, emerge as pivotal to mitigate such biases, ensuring that algorithms do not just superficially perform well but genuinely understand and predict minority class instances.

3.2. Sampling Techniques

This characteristic of the imbalanced data leads to the construction of a biased classifier that has high accuracy for the majority class (non-churned) but low accuracy for the minority class (churned). Several sampling methods have been proposed to address this issue. Sampling techniques are applied to imbalanced data to alter the class distribution and create balanced data. Generally, sampling techniques are divided into two categories: undersampling, where instances from the majority class are removed, and oversampling, where instances from the minority class are artificially increased [45]. These methods aim to adjust the class distribution to enable the classifier to make better-informed decisions.

3.2.1. Synthetic Minority Over-Sampling Technique (SMOTE)

The Synthetic Minority Over-Sampling Technique (SMOTE) [46] is an oversampling technique that aims to balance the data by replicating instances of the minority class and is widely utilized to address this issue. Unlike simplistic methods that merely replicate minority instances, SMOTE innovatively crafts synthetic samples through an interpolation process between existing minority instances. This nuanced augmentation not only enhances the representation of the minority class but also fosters a more diverse and expansive

decision boundary. Such an enriched decision space proves particularly beneficial for algorithms, like Support Vector Machines (SVMs), which are inherently sensitive to the distribution of instances in their modeling process.

3.2.2. Tomek Links

Tomek Links are an undersampling method and an extension to the Condensed Nearest Neighbor (CNN) method, proposed by Ivan Tomek (in his 1976 paper titled "Two modifications of CNN") [47]. The Tomek links method identifies pairs of examples (each from a different class) that have the minimum Euclidean distance from each other. By removing such instances, especially from the majority class, decision boundaries can become clearer and less prone to overlap. This enhanced delineation of decision spaces proves notably advantageous for classifiers, such as Decision Trees and k-Nearest Neighbors (k-NNs), which rely heavily on a clear distinction between classes for optimal performance.

3.2.3. Edited Nearest Neighbors (ENNs)

Edited Nearest Neighbors (ENN) are another undersampling method proposed by Wilson (in his 1972 paper titled "Asymptotic Properties of Nearest Neighbor Rules Using Edited Data") [48]. This method computes the three nearest neighbors for each instance in the dataset. If the instance belongs to the majority class and is misclassified by its three nearest neighbors, then it is removed from the dataset. Alternatively, if the instance belongs to the minority class and is misclassified by its three nearest neighbors, then the three majority-class instances are removed. This method often results in smoother decision boundaries, particularly benefiting algorithms sensitive to noisy data.

3.3. Combined Data Sampling Techniques

While individual sampling techniques can offer improvements, combining methods often yields superior results. This is because a combination captures the benefits of both oversampling and undersampling, refining decision boundaries and enhancing classifier robustness. In this study, to address imbalanced data, we use two of the most popular combinations of sampling techniques, such as the combination of SMOTE and Tomek Links and the combination of SMOTE and ENN.

4. Training and Validation Process

For evaluating our classifiers, we employ the k-fold cross-validation technique. However, there is a limitation when using this technique with imbalanced data. The issue is that, with this technique, the data are split into k-folds with a uniform probability distribution, and in imbalanced data, some folds may have no or few examples from the minority class. To address this issue, we can use a stratified sampling technique when performing train-test split or k-fold cross-validation. Using stratification ensures that each split of the data has an equal number of instances from the minority class.

We utilize an out-of-sample testing approach to evaluate the performance of the models. This approach demonstrates the performance of the models on unseen data that were not used to train the models.

When working with imbalanced data, it is essential to up-sample or down-sample only after splitting the data into train and test sets (and validate if desired). If the dataset is up-sampled prior to splitting it into test and train, it is likely that the model experiences data leakage. This way, we may wrongly assume that our machine learning model is performing well. After building a machine learning model, it is recommended to test the metric on the not-up-sampled train dataset. When the metric is tested on the not-up-sampled dataset, the model's performance can be more realistically estimated compared to when it is tested on the up-sampled dataset.

5. Evaluation Metrics

We employ two types of metrics to evaluate our models. (1) Threshold metrics: these metrics are designed to minimize the error rate and assist in calculating the exact number of predicted values that do not match the actual values. (2) Ranking metrics: these metrics are designed to evaluate the effectiveness of classifiers in separating classes. These metrics require classifiers to predict a probability or a score of class membership. By applying different thresholds, we can test the effectiveness of classifiers, and those classifiers that maintain a good score across a range of thresholds will have better class separation and, as a result, will have a higher rank.

5.1. Threshold Metrics

Normally, we use the standard accuracy metric (Equation (6)) for measuring the performance of ML models. However, for imbalanced data, classification ML models may achieve high accuracy, as this metric only considers the majority class. In an imbalanced dataset, instances of the minority class (churned) are rare, and thus, True Positives (TPs) do not have a significant impact on the standard accuracy metric. This metric, therefore, cannot accurately represent the performance of the models. For example, if the model correctly predicts all data points in the majority class (non-churned), it will result in high True Negatives (TNs) and a high standard of accuracy without accurately predicting anything about the minority class (churned). In the case of imbalanced data, this metric is not sufficient as a benchmark criterion measure [49]. Therefore, other metrics, such as recall, precision, and F1-score, are commonly used to evaluate the performance of ML models in minority classes and can be extracted from the confusion matrix, as shown in Table 2.

Table 2. The confusion matrix for evaluating methods.

		Predicted Class	
		Churners	**Non-Churners**
Actual Class	Churners	TP	FN
	Non-churners	FP	TN

The confusion matrix helps us to understand the performance of ML models by showing which class is being predicted correctly and which one is being predicted incorrectly.

In Table 2, TP and FP stand for True Positive and False Positive, and FN and TN stand for False Negative and True Negative, respectively. Precision, Recall, and Accuracy can be calculated using the following formulas:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{9}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{10}$$

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{11}$$

But Precision and Recall are not sufficient for evaluating the accuracy of the mentioned methods, since they do not provide enough information and can be misleading. Therefore, we usually use the F1-score metric as a single metric to evaluate the accuracy of our models. The F1-score is a combination of Precision and Recall metrics and balances both precision and recall and provides a single metric that represents the overall performance of the model. The F1-score is defined as follows:

$$\text{F1} - \text{score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{12}$$

The more the value of the F1-score is closer to 1, the better combination of Precision and Recall is achieved by the model [50].

5.2. Ranking Metrics

In the field of churn prediction, the Receiver Operating Characteristic (ROC) Curve is widely recognized as a prominent ranking metric for evaluating the performance of classifiers. This metric enables the assessment of a classifier's ability to differentiate between classes by providing a visual representation of the True Positive rate and False Positive rate of predicted values, as calculated under various threshold values.

The True Positive rate (recall or sensitivity) is calculated as follows:

$$\text{TruePositiveRate} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{13}$$

And the False Positive rate is calculated as follows:

$$\text{FalsePositiveRate} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{14}$$

Each point on the plot represents a prediction made by the model, with the curve being formed by connecting all points. A line running diagonally from the bottom left to the top right on the plot represents a model with no skill, and any point located below this line represents a model that performs worse than one with no skill. Conversely, a point in the top left corner of the plot symbolizes a perfect model.

The Area Under the ROC curve can be calculated and utilized as a single score to evaluate the performance of models. A classifier with no skill has a score of 0.5, and a perfect classifier has a score of 1.0, as shown in Figure 5. However, it should be noted that the ROC curve can be effective for classification problems with a low imbalanced ratio and can be optimistic for classification problems with a high imbalanced ratio. In such cases, the Precision–Recall curve is a more appropriate metric because it focuses on the performance of the classifier on the minority class, as depicted in Figure 6.

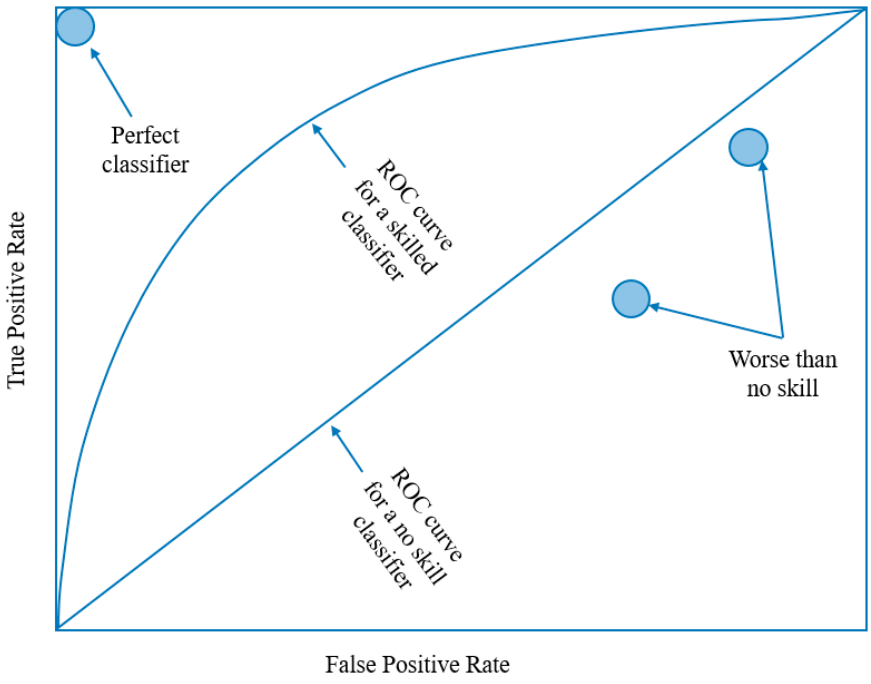

Figure 5. The ROC curve.

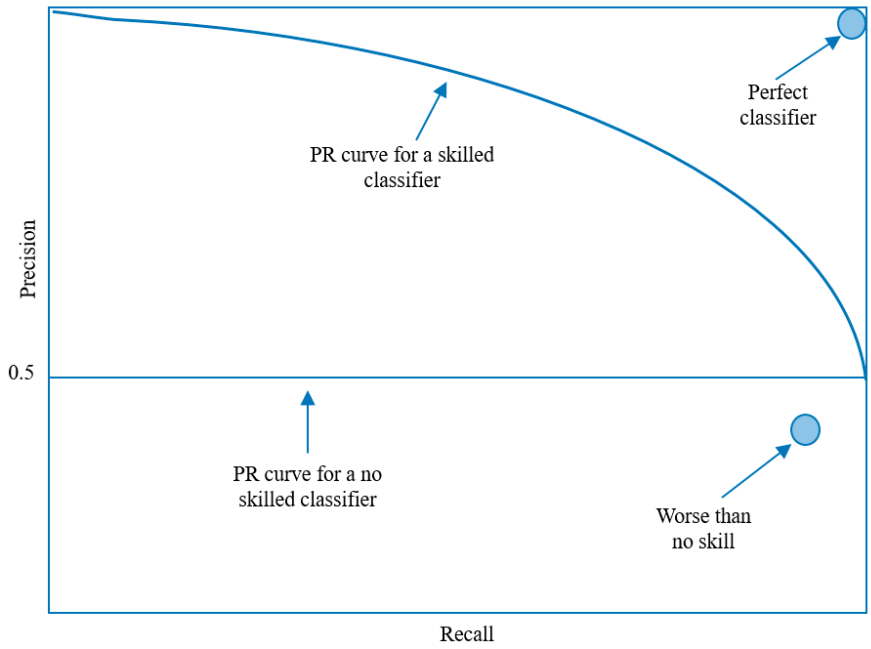

Figure 6. The Precision–Recall curve.

The ROC curve is a widely used method for evaluating the performance of machine learning models. The ROC curve plots the True Positive rate against the False Positive rate at various threshold settings, with each point on the curve representing a predicted value by the model.

A horizontal line on the plot signifies a model with no skill, while points below the diagonal line indicate a model that performs worse than random chance. Conversely, a point located in the top left quadrant of the plot represents a model with perfect performance.

In datasets with a balanced distribution of positive and negative examples, the horizontal line on the ROC plot is typically set at 0.5. However, when the dataset is imbalanced, such as with an imbalanced ratio of 1:10, the horizontal line is adjusted to 0.1 to reflect the imbalanced nature of the data.

In addition to the ROC curve, the Area Under the ROC curve (AUC) is also a commonly used metric for evaluating the performance of machine learning models. The AUC provides a single score for comparing the performance of different models. In cases where the dataset has a high imbalanced ratio, the Precision–Recall AUC (PR AUC) may be more informative as it specifically focuses on the performance of the minority class. However, if the imbalanced ratio of the dataset is not excessively high, such as the dataset utilized in this study, the use of PR AUC may not be necessary for the evaluation.

In this paper, we employ a comprehensive set of metrics to evaluate the performance of machine learning models, including Recall, Precision, F1-score, and Receiver Operating Characteristic (ROC) AUC. These metrics provide a comprehensive evaluation of the model's performance, including its ability to accurately identify positive examples, balance False Positives and False Negatives, and handle imbalanced datasets.

Among these four metrics, we primarily focused on the F1-score and ROC AUC metrics for the following reasons:

- F1-score: Given the imbalance in our dataset, the F1-score is particularly useful as it does not inflate the performance of the model due to the high number of True Negatives, which is a common issue with accuracy in such datasets.
- ROC AUC: Unlike the standard accuracy metric, ROC AUC places a particular emphasis on the performance of the minority class, and the accurate prediction of minority class instances is central to its calculation. This is particularly useful in situations where the dataset is imbalanced, as it ensures that the model's performance is evaluated fairly. This metric is less sensitive to class imbalance and provides insight into

the model's ability to distinguish between classes, making it a robust measure for comparing the performance of different models.

5.3. ROC AUC Benchmark

It is clear that an ROC Area Under the Curve (AUC) of 100% represents the optimal performance that a machine learning model can achieve, as it indicates that all instances of the positive class (e.g., churns in the case of customer retention) are ranked higher in risk than all instances of the negative class (e.g., non-churns). However, it is highly unlikely that any model will achieve this level of performance in real-world problems.

As such, when comparing the performance of different machine learning models using ROC AUC, it is necessary to have a benchmark to determine whether the model's performance is acceptable. The ROC AUC ranges from 50% to 100%, with 50% being equivalent to random guessing and 100% representing perfect performance. As can be seen in Table 3, the worst possible AUC is 50%, which is similar to the result of a coin flip for prediction. If the percentages are less than 50%, it indicates an issue with the model. Consider the worst-case scenario of obtaining a zero percent accuracy. While this might seem problematic, it actually means that the model ranked all non-churn customers as higher risk than churn customers. Surprisingly, this result could be considered good because it implies your model can perfectly predict customer retention. However, most likely, there was an error in your model setup causing it to predict in the opposite direction.

Table 3. ROC AUC benchmark for predicting churn.

ROC AUC Threshold	Description
ROC AUC < 50%	Something is wrong *
50% $\leq$ ROC AUC < 60%	Similar to flipping a coin
60% $\leq$ ROC AUC < 70%	Weak prediction
70% $\leq$ ROC AUC < 80%	Good Prediction
80% $\leq$ ROC AUC < 90%	Very Good Prediction
ROC AUC $\geq$ 90%	Excellent Prediction

* Check the data and the AUC calculation.

In Table 3, the categorization of the ROC AUC metric follows empirical norms and established methodologies within machine learning to yield a discernible evaluation of model efficacy. This discretization strategy is intended to furnish a pragmatic benchmark for evaluating churn-prediction models, facilitating a straightforward appraisal for both researchers and practitioners.

6. Simulation

6.1. Simulation Setup

The primary objective of this study is to evaluate and compare the performance of several popular classification techniques in solving the problem of customer churn prediction. The classifiers under examination include Decision Tree, Logistic Regression, Random Forest, Support Vector Machine, XGBoost, LightGBM, and CatBoost. To achieve this goal, simulations were conducted using the Python programming language and various libraries, such as Pandas, NumPy, and Scikit-learn.

A real-world dataset was used for this study, which was obtained from Kaggle and is outlined in Table 4 [44]. The training dataset consists of 20 attributes and 4250 instances, while the testing dataset has 20 attributes and 750 instances. The training dataset features a churn rate of 14.1% and an active subscriber rate of 85.9%. The performance of the models was evaluated using a variety of metrics, including the Precision, Recall, F1-score, and ROC AUC as defined previously. After undergoing pre-processing steps, such as handling categorical variables, feature selection, and removing outliers, these metrics were evaluated using both the training and testing datasets. Additionally, the SMOTE technique was used to handle imbalanced data, and the effect on the performance of the models was examined.

Table 4. The names and types of different variables in the churn dataset.

Variable Name	Type
state, (the US state of customers)	*string*
account_length (number of active months)	*numerical*
area_code, (area code of customers)	*string*
international_plan, (whether customers have international plans)	*yes/no*
voice_mail_plan, (whether customers have voice mail plans)	*yes/no*
number_vmail_messages, (number of voice-mail messages)	*numerical*
total_day_minutes, (total minutes of day calls)	*numerical*
total_day_calls, (total number of day calls)	*numerical*
total_day_charge, (total charge of day calls)	*numerical*
total_eve_minutes, (total minutes of evening calls)	*numerical*
total_eve_calls, (total number of evening calls)	*numerical*
total_eve_charge, (total charge of evening calls)	*numerical*
total_night_minutes, (total minutes of night calls)	*numerical*
total_night_calls, (total number of night calls)	*numerical*
total_night_charge, (total charge of night calls)	*numerical*
total_intl_minutes, (total minutes of international calls)	*numerical*
total_intl_calls, (total number of international calls)	*numerical*
total_intl_charge, (total charge of international calls)	*numerical*
number_customer_service_calls, (number of calls to customer service)	*numerical*
churn, (customer churn—the target variable)	*yes/no*

6.2. Simulation Results

In this study, we evaluate the performance of several machine learning models (Decision Tree, Logistic Regression, Artificial Neural Network, Support Vector Machine, Random Forest, XGBoost, LightGBM, and CatBoost) based on unseen data using a range of metrics, including the Precision, Recall, F1-score, Receiver Operating Characteristic (ROC) Area Under the Curve (AUC), and Precision–Recall (PR) AUC. The evaluation is carried out on the testing dataset to assess the generalization ability of the models and to determine their performance based on unseen data.

6.2.1. After Pre-Processing and Feature Selection

After undergoing several pre-processing steps, such as handling categorical features and feature selection, the aforementioned models were applied to the data, and their performance was evaluated. The results of this evaluation are presented in Table 5, with the highest values highlighted in bold and marked with an asterisk.

Table 5. Evaluation metrics for the different models after pre-processing and feature selection.

Models	Precision%	Recall%	F1-score%	ROC AUC%
DT	91	72	77	72
ANN	85	76	80	77
LR	61	70	62	70
SVM	81	57	59	57
RF	96	75	81	75
CatBoost	90	90	90	90
LightGBM	94	91	**92***	**91***
XGBoost	96	87	91	87

As depicted in Table 5, the boosting models demonstrate superior performance, particularly in relation to F1-score and ROC AUC metrics. Notably, LightGBM surpasses

the performance of other methods, achieving an impressive F1-score of 92% and an ROC AUC of 91%. Figure 7 shows the diagram of the ROC curve for the different models after pre-processing and feature selection.

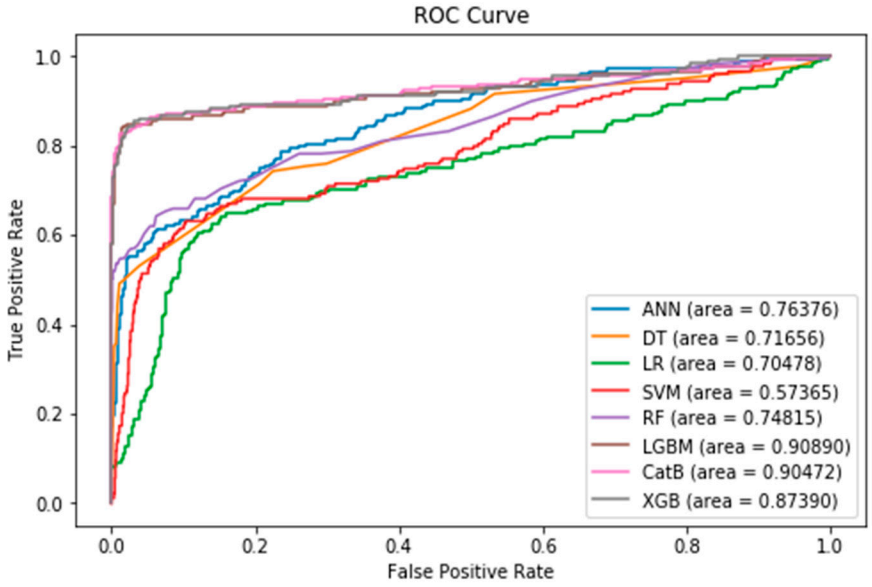

Figure 7. ROC curve after pre-processing and feature selection.

6.2.2. Applying SMOTE

To address the issue of class imbalance in the training data, where the number of instances of class-0 is 3652 and the number of instances of class-1 is 598, we have applied the SMOTE technique to the training dataset. This technique was used to create synthetic instances of the minority class in order to achieve a balanced training dataset. As a result of the application of SMOTE, the number of instances for both class-0 and class-1 is now equal to 2125.

As Table 6 shows, LightGBM and XGBoost outperform other ML techniques in all evaluation metrics. Notably, LightGBM and XGBoost surpass the performance of other methods, with both achieving an impressive ROC AUC of 90%, and XGBoost outperforms the other methods, achieving an impressive F1-score of 92%. Figure 8 shows the diagram of the ROC curve for the different models after applying SMOTE.

6.2.3. Applying SMOTE with Tomek Links

As previously discussed in Section IV, the Tomek Links method is an undersampling technique that is used to identify pairs of examples, where each example belongs to a different class that has the minimum Euclidean distance to each other. Additionally, as noted in the section, it is beneficial to utilize a combination of both oversampling and undersampling techniques to achieve optimal results. The results of the evaluation metrics for the various models after applying the SMOTE technique in conjunction with Tomek Links are presented in Table 7. Notably, LightGBM outperforms the other methods, achieving an impressive ROC AUC of 91%, and XGBoost surpasses the other methods, achieving an impressive F1-score of 91%. As indicated in Table 7, XGBoost demonstrates a marginal performance improvement, with a modest 2% enhancement in the ROC AUC compared to the pre-processing and feature selection stage (initial state), as shown in Table 6. Figure 9 shows the diagram of the ROC curve for the different models after applying SMOTE with Tomek Links.

Table 6. Evaluation metrics for the different models after applying SMOTE.

Models	Precision%	Recall%	F1-score%	ROC AUC%
DT	69	72	70	72
ANN	70	73	71	83
LR	61	71	61	70
SVM	65	73	68	73
RF	83	76	79	76
CatBoost	79	88	83	88
LightGBM	87	90	88	**90***
XGBoost	95	90	**92***	**90***

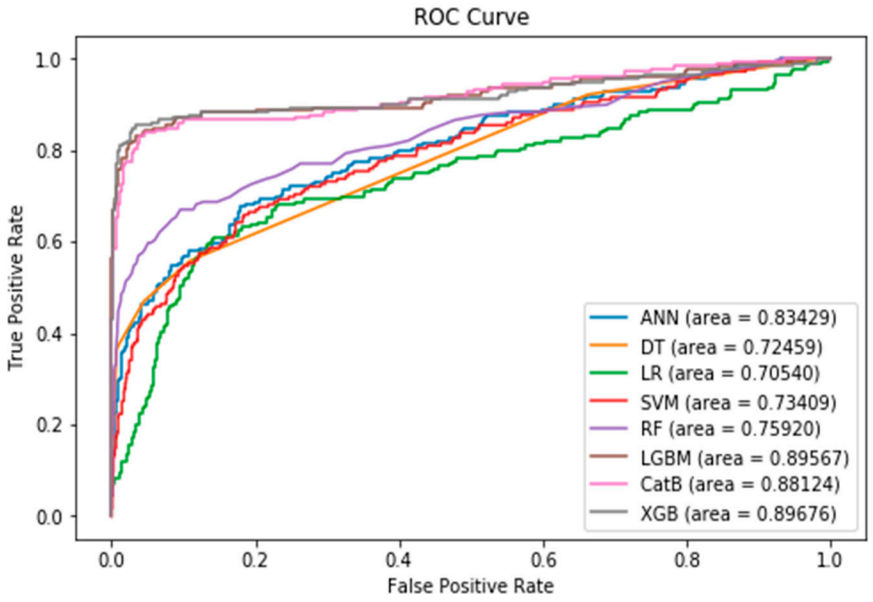

Figure 8. ROC curve after applying SMOTE.

Table 7. Evaluation metrics for the different models after applying SMOTE with Tomek Links.

Models	Precision%	Recall%	F1-score%	ROC AUC%
DT	74	74	74	74
ANN	69	75	71	75
LR	61	70	61	69
SVM	65	73	67	73
RF	85	78	81	78
CatBoost	80	88	83	88
LightGBM	89	91	90	**91***
XGBoost	94	89	**91***	89

6.2.4. Applying SMOTE with ENN

As previously discussed in Section 3 the ENN method is employed to compute the three nearest neighbors for each instance within the dataset. In instances where the sample belongs to the majority class and is misclassified by its three nearest neighbors, the instance is removed from the dataset. Conversely, if the instance belongs to the minority class and is misclassified by its three nearest neighbors, the three majority class instances are removed. Furthermore, as previously stated, it has been shown to be beneficial to utilize a combination of undersampling and oversampling techniques in order to achieve optimal results. Table 8 illustrates the evaluation metrics for the various models following the application of the SMOTE technique in conjunction with the ENN method. The results

indicate that XGBoost outperforms the other machine learning techniques, achieving an F1-score of 88% and an ROC AUC of 89%. As indicated in Table 8, XGBoost exhibits a performance decline, experiencing a 3% reduction in F1-score compared to the pre-processing and feature selection stage (initial state), as shown in Table 6. Figure 10 shows the diagram of the ROC curve for the different models after applying SMOTE with ENN.

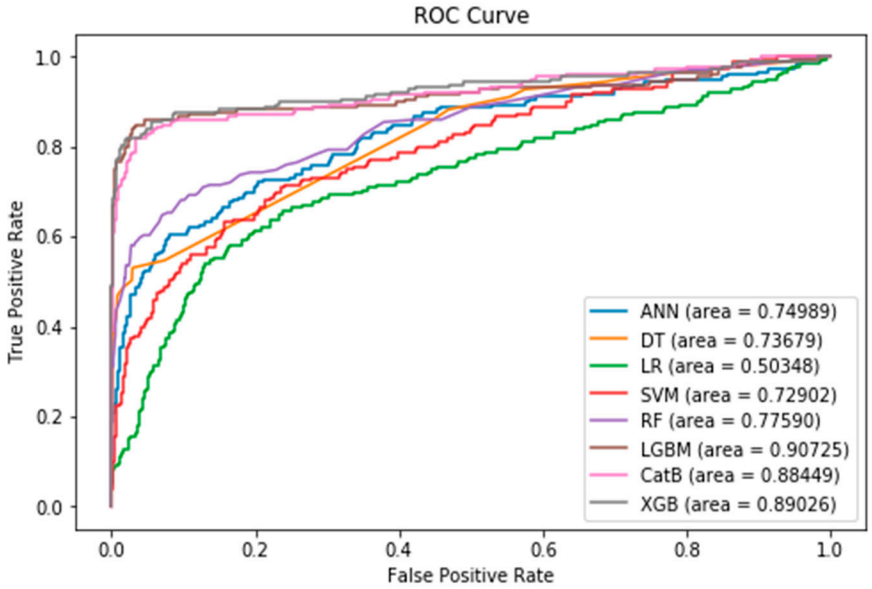

Figure 9. ROC curve after applying SMOTE with Tomek Links.

Table 8. Evaluation metrics for the different models after applying SMOTE with ENN.

Models	Precision%	Recall%	F1-Score%	ROC AUC%
DT	60	70	50	70
ANN	61	70	60	70
LR	52	50	50	50
SVM	60	70	58	70
RF	67	76	69	76
CatBoost	70	83	72	83
LightGBM	80	89	84	87
XGBoost	88	89	**88***	**89***

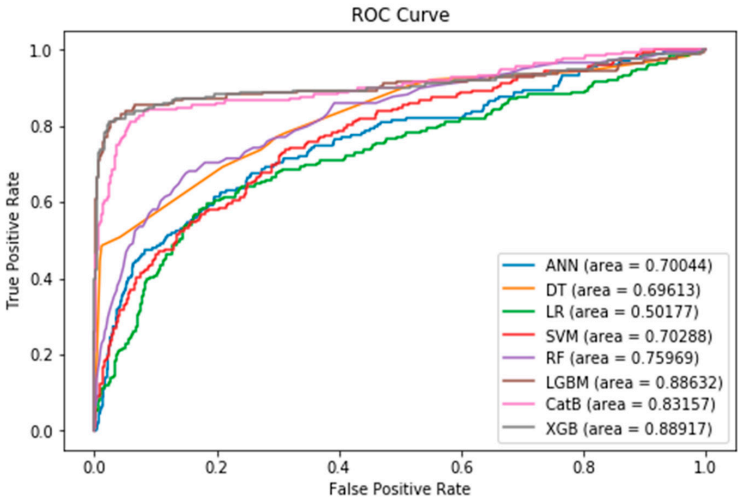

Figure 10. ROC curve after applying SMOTE with ENN.

6.2.5. The Impact of Sampling Techniques

F1-Score

Table 9 and Figure 11 show the impact of three distinct sampling techniques (SMOTE, SMOTE with Tomek Links, and SMOTE with ENN) on the F1-score metric of various machine learning models. These comparisons offer insights into the effectiveness of each technique in handling imbalanced datasets.

1. Impact of SMOTE Sampling Technique:
   - Most models saw a decrease in the F1-score after applying SMOTE compared to the pre-processing and feature selection stage (initial state).
   - CatBoost and LightGBM experienced a reduction in F1-scores, but XGBoost showed slight improvements.
   - Support Vector Machine (SVM) exhibits an enhanced F1-score.
2. Impact of SMOTE with Tomek Links Sampling Technique:
   - SMOTE with Tomek Links demonstrates further enhancements in F1-scores for several models compared to SMOTE alone.
   - Support Vector Machine (SVM) showed improvements.
   - CatBoost experienced a reduction in F1-scores compared to the pre-processing and feature selection stage (initial state).
   - LightGBM showed a slight reduction in F1-scores by 2%.
   - XGBoost remained consistent with an F1-score of 91.
3. Impact of SMOTE with ENN Sampling Technique:
   - SMOTE with ENN leads to varied impacts on F1-scores across models.
   - Some models, like the Decision Tree (DT), Logistic Regression (LR), and CatBoost, experience significant drops in F1-scores compared to the pre-processing and feature selection stage (initial state).
   - LightGBM maintains relatively high F1-scores, with LightGBM achieving 84%.
   - XGBoost remains strong with an F1-score of 88% despite the decline.
   - SMOTE with ENN may not consistently enhance performance and should be chosen carefully based on the specific model and dataset characteristics.

Table 9. F1-score of different ML models after applying different sampling techniques.

	DT	ANN	LR	SVM	RF	CatBoost	XGBoost	LightGBM
Initial	**77***	**80***	**62***	59	**81***	**90***	**92***	91
SMOTE	70	71	61	**68 ***	79	83	88	**92***
SMOTE-TOMEK	74	71	61	67	**81***	83	90	91
SMOTE-ENN	50	60	50	58	69	72	84	88

In summary, the impact of different sampling techniques on F1-scores varied across models. SMOTE generally led to reduced F1-scores, with CatBoost and LightGBM experiencing declines and XGBoost showing slight improvements. SMOTE with Tomek Links enhanced F1-scores for several models, particularly benefiting SVM, but CatBoost and LightGBM saw reductions. SMOTE with ENN had mixed effects on F1-scores, significantly decreasing scores for some models but maintaining higher scores for LightGBM and XGBoost. Among the sampling techniques, SMOTE-ENN yields the least favorable results for all machine learning models when contrasted with methods, such as SMOTE and SMOTE-TOMEK. Choosing the appropriate sampling technique should consider specific model and dataset characteristics.

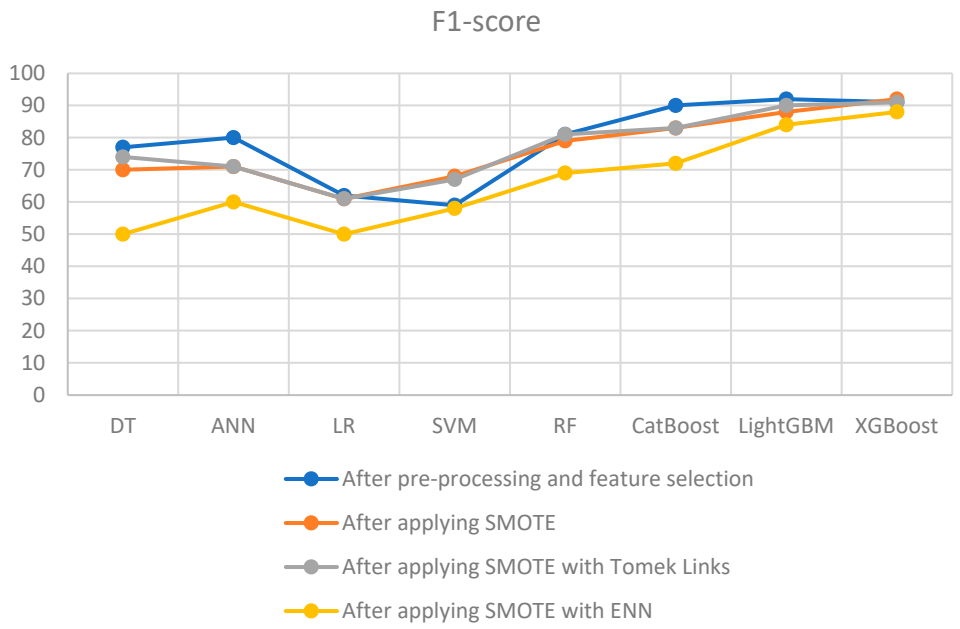

Figure 11. The impact of sampling techniques on the F1-score of different ML models.

ROC AUC

Table 10 and Figure 12 show the impact of three distinct sampling techniques (SMOTE, SMOTE with Tomek Links, and SMOTE with ENN) on the ROC AUC metric of various machine learning models. These comparisons offer insights into the effectiveness of each technique in handling imbalanced datasets.

1. Impact of SMOTE Sampling Technique:
    - After applying SMOTE, there are noticeable improvements in ROC AUC metrics for some models.
    - ANN, SVM, RF, and XGBoost experience ROC AUC enhancements, but CatBoost and LightGBM showed a slight reduction compared to the pre-processing and feature selection stage (initial state).
    - Models, like ANN and SVM, see substantial improvements, with ROC AUC scores reaching 83% and 73%, respectively.
2. Impact of SMOTE with Tomek Links Sampling Technique:
    - SMOTE combined with Tomek Links maintains or enhances ROC AUC metrics for most models.
    - DT, SVM, and RF observe improved ROC AUC metrics.
    - LightGBM and CatBoost maintain high ROC AUC scores of 91% and 88%, respectively.
    - This technique's combination of class balancing (SMOTE) and the removal of borderline instances (Tomek Links) continues to prove effective.
3. Impact of SMOTE with ENN Sampling Technique:
    - SMOTE with ENN produces mixed results for ROC AUC metrics.
    - While some models, like RF and XGBoost, and SVM showed improvements in ROC AUC metrics, others experienced drops.
    - Logistic Regression (LR) encounters a significant reduction in the ROC AUC.
    - LightGBM maintains a respectable ROC AUC metric of 87%.
    - Researchers should exercise caution when applying SMOTE with ENN, as its impact varies across models.

Table 10. ROC AUC of different ML models after applying different sampling techniques.

	DT	ANN	LR	SVM	RF	CatBoost	XGBoost	LightGBM
Initial	72	77	**70***	57	75	**90***	**91***	87
SMOTE	72	**83***	**70***	**73***	76	88	90	**90***
SMOTE-TOMEK	**74***	75	69	**73***	**78***	88	**91***	89
SMOTE-ENN	70	70	50	70	76	83	87	89

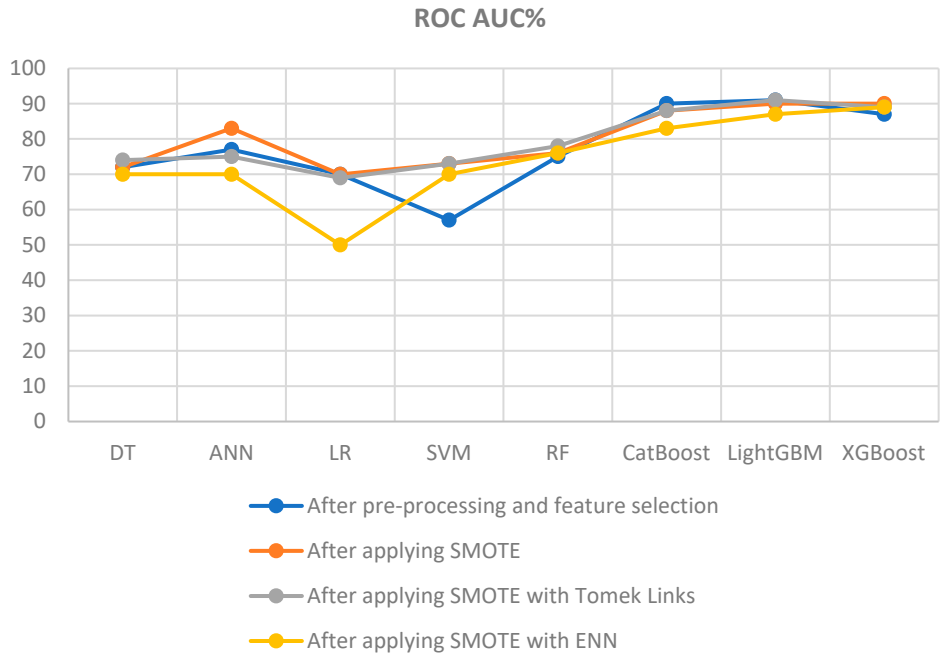

Figure 12. The impact of sampling techniques on ROC AUC of different ML models.

In summary, the impact of different sampling techniques on ROC AUC metrics varied among models. SMOTE led to improvements for ANN, SVM, RF, and XGBoost but slight reductions for CatBoost and LightGBM. Notably, ANN and SVM achieved substantial ROC AUC scores of 83% and 73%, respectively. SMOTE with Tomek Links generally maintained or improved ROC AUC metrics, benefiting models, like DT, SVM, RF, LightGBM, and CatBoost, with the latter two maintaining high scores. SMOTE with ENN produced mixed results, improving the ROC AUC for some models, such as RF, XGBoost, and SVM, while causing a significant reduction in Logistic Regression. LightGBM maintained a respectable ROC AUC of 87%. Similar to the F1-score metric, SMOTE-ENN demonstrates lower performance in terms of the ROC AUC for all machine learning models compared to techniques, such as SMOTE and SMOTE-TOMEK. Researchers should select the most appropriate sampling technique based on their dataset and model to achieve optimal ROC AUC results.

Sampling Techniques vs. Boosting Techniques

Several factors contribute to the relatively modest impact of sampling techniques on the performance of boosting algorithms:

- Iterative Nature: Boosting methods iteratively train a sequence of weak models, typically Decision Trees. Each subsequent model focuses on the errors made by the previous ones. Boosting is adaptive in the sense that it can adjust to the errors and potentially correct them in subsequent iterations.

- Adaptive Nature: While oversampling introduces more instances of the minority class, boosting models, given their adaptive nature, can sometimes already compensate for the imbalance to some degree. As a result, oversampling might not always result in significant performance improvements.
- Weighted Loss Function: Many boosting algorithms, like XGBoost, offer a weighted loss function where instances from different classes can be assigned different weights. This built-in mechanism can help in addressing class imbalance, reducing the need for external sampling methods.

In summary, while data sampling can rectify the decision boundary in models like SVM that are sensitive to class distributions, boosting techniques, due to their adaptive and iterative nature, might already have mechanisms to handle an imbalance to a certain extent. However, the actual impact of sampling can vary based on the dataset, the degree of imbalance, the specific boosting algorithm used, and its hyperparameters.

6.2.6. Applying Optuna Hyperparameter Optimizer

Hyperparameter optimization is pivotal in machine learning for enhancing the performance of models. While models come with default hyperparameters, fine-tuning them to a specific dataset can substantially boost their efficacy. One prominent tool in this space is Optuna. Takuya Akiba et al. (2019) [51] introduced Optuna, an open-source Python library for hyperparameter optimization. Optuna aims to balance the pruning and sampling algorithms through the execution of various techniques, such as the Tree-Structured Parzen Estimator (TPE) [52,53] for independent parameter sampling, Covariance Matrix Adaptation (CMA) [54], and Gaussian Processes (GPs) [53] for relational parameter sampling. The library also utilizes a variant of the Asynchronous Successive Halving (ASHA) algorithm [55] to prune search spaces.

TPE is a Bayesian optimization technique. Unlike grid or random search, which treats hyperparameters as isolated, TPE considers the relationship between hyperparameters and the objective function. The advantage of TPE over other methods lies in its efficiency. By constructing a probabilistic model of the objective function, it can suggest hyperparameters that are more likely to yield better results, hence reducing the number of trials [52,53].

Both CMA and GP are methodologies used in Optuna for relational parameter sampling. CMA captures the interdependencies between parameters, optimizing the sampling process, while GP uses the kernel trick to project data into higher dimensions, capturing complex relationships in the hyperparameter space [53,54].

The goal of ASHA is efficiency. It is an early stopping strategy to prune trials that do not show promise, which allows for a more efficient hyperparameter search. By identifying and halting unpromising trials early, computational resources are channeled more effectively [55].

In this study, we applied the Optuna library to the popular machine learning models, CatBoost, XGBoost, and LightGBM. The results, as presented in Table 11, indicate that CatBoost outperforms XGBoost and LightGBM when utilizing Optuna for hyperparameter optimization, achieving an impressive F1-score of 93% and an ROC AUC of 91%. The improved F1-score and ROC AUC results observed after employing Optuna hyperparameter tuning for CatBoost likely result from the enhanced hyperparameter settings. Optuna fine-tunes these settings more effectively for your specific data, reducing overfitting and enhancing the models' generalization to new data. This ultimately leads to improved overall model performance, as hyperparameters play a significant role in how effectively these algorithms operate with the dataset. Figure 13 shows the diagram of the ROC curve for the different models after applying Optunal hyperparameter tuning.

Table 12 includes all the parameters that were used in the XGBoost, LightGBM, and CatBoost models after applying Optuna hyperparameter tuning. The table provides a clear and concise summary of the parameter values that were selected for the models.

Table 11. Evaluation metrics for various models after applying Optuna hyperparameter optimization.

Models	Precision%	Recall%	F1-Score%	ROC AUC%
CatBoost	89	91	90	**91***
CatBoost-Optuna	95	91	**93***	**91***
LightGBM	92	90	91	90
LightGBM-Optuna	93	89	90	89
XGBoost	93	88	90	88
XGBoost-Optuna	94	88	91	88

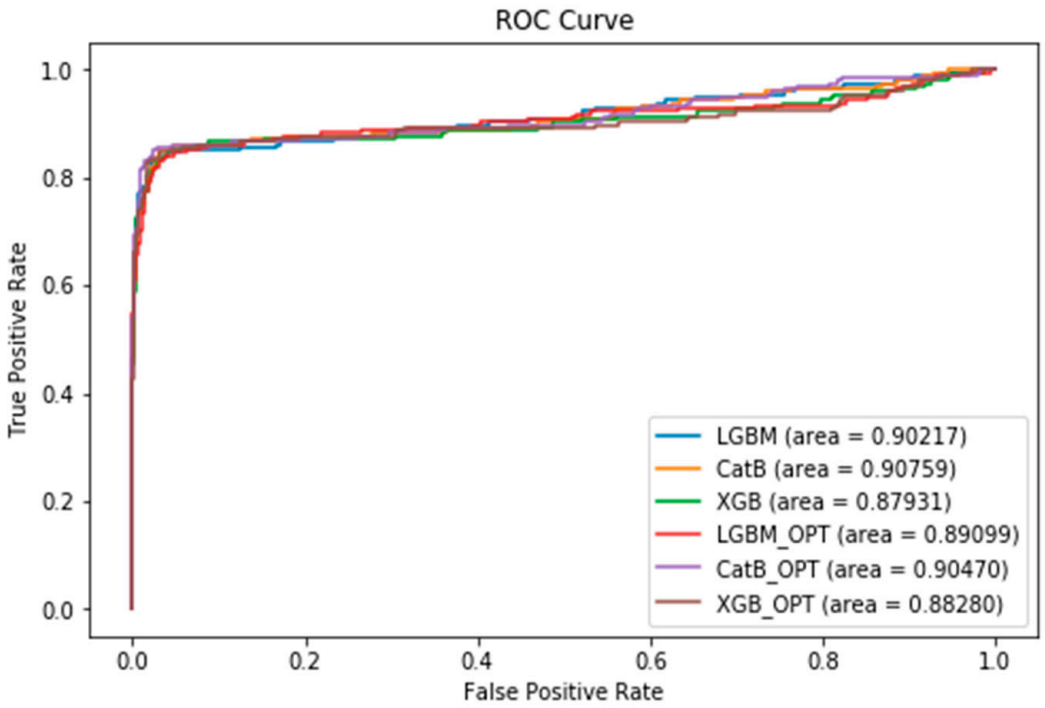

Figure 13. ROC curve after Optuna hyperparameter tuning.

Table 12. Optuna hyperparameter optimization parameters.

Parameter	Description	Value
XGBoost Tuning Parameters		
verbosity	Verbosity of printing messages	0
objective	Objective function	binary:logistic
tree_method	Tree construction method	exact
booster	Type of booster	dart
lambda	L2 regularization weight	0.010281489790562261
alpha	L1 regularization weight	0.0008440304772889829
subsample	Sampling ratio for training data	0.8298281841818362
colsample_bytree	Sampling according to each tree	0.9985902928710126
max_depth	Maximum depth of the tree	7
min_child_weight	Minimum child weight	2
eta	Learning rate	0.12406825365082062
gamma	Minimum loss reduction required to make a further partition on a leaf node of the tree	0.0004490383815764321
grow_policy	Controls the way new nodes are added to the tree	depthwise

Table 12. *Cont.*

Parameter	Description	Value
LightGBM Tuning Parameters		
objective	Objective function	binary
metric	Metric for binary classification	binary_logloss
verbosity	Verbosity of printing messages	−1
boosting_type	Type of booster	dart
num_leaves	Maximum number of leaves in one tree	1169
max_depth	Maximum depth of the tree	10
lambda_l1	L1 regularization weight	$2.689492421801289 \times 10^{-7}$
lambda_l2	L2 regularization weight	$7.2387875465462 \times 10^{-8}$
feature_fraction	LightGBM will randomly select part of features on each iteration	0.870805980078817
bagging_fraction	LightGBM will randomly select part of data without resampling	0.6280893693081118
bagging_freq	Frequency for bagging	7
min_child_samples	Minimum amount of data in one leaf	8
CatBoost Tuning Parameters		
Objective	Objective function	Logloss
colsample_bylevel	Subsampling rate per level for each tree	0.07760972009427407
depth	Depth of the tree	12
boosting_type	Type of booster	Ordered
bootstrap_type	Sampling method for bagging	Bayesian
bagging_temperature	Controls the similarity of samples in each bag	0.0

7. Conclusions

In this study, we employed various machine learning (ML) models, including Artificial Neural Networks, Decision Trees, Support Vector Machines, Random Forests, Logistic Regression, and three modern gradient boosting techniques, namely XGBoost, LightGBM, and CatBoost, to predict customer churn in the telecommunications industry using a real-world imbalanced dataset. We evaluated the impact of different sampling techniques, such as SMOTE, SMOTE with Tomek Links, and SMOTE with ENN, to handle the imbalanced data. We then assessed the performance of the ML models using various metrics, including the Precision, Recall, F1-score, and Receiver Operating Characteristic Area Under the Curve (ROC AUC). Finally, we utilized the Optuna hyperparameter optimization technique on CatBoost, LightGBM, and XGBoost to determine the effect of optimization on the performance of the models. We compared the results of all the steps and presented them in tabular form.

The simulation results demonstrate the performance of different models based on unseen data. LightGBM and XGBoost consistently exhibit superior performance across various evaluation metrics, including the Precision, Recall, F1-score, and ROC AUC. The performance of these models is further improved when applying techniques, such as SMOTE with Tomek Links or SMOTE with ENN, to handle imbalanced data. Additionally, the use of Optuna hyperparameter optimization for CatBoost, XGBoost, and LightGBM models shows further improvements in performance.

In summary, the key findings of the study are as follows:

- ❖ Impact of SMOTE: After applying SMOTE, both LightGBM and XGBoost achieved impressive ROC AUC scores of 90%. Additionally, XGBoost outperformed other methods with an impressive F1-score of 92%. SMOTE effectively balanced class distribution, leading to enhanced recall and ROC AUC for most models.
- ❖ SMOTE with Tomek Links: After applying SMOTE with Tomek Links, LightGBM excelled among the methods with an impressive ROC AUC of 91%. XGBoost also outperformed other methods with an impressive F1-score of 91%. LightGBM demonstrates a slight performance boost, with a modest 2% improvement in the F1-score and a 1% increase in ROC AUC compared to when using SMOTE alone. Conversely,

XGBoost showed a slight performance decline, experiencing a corresponding 1% reduction in the F1-score and ROC AUC compared to exclusive SMOTE utilization.

- ❖ SMOTE with ENN: After applying SMOTE with ENN, XGBoost surpassed other ML techniques, achieving an F1-score of 88% and an ROC AUC of 89%. However, XGBoost exhibited a performance decline, with a 4% reduction in the F1-score and a 1% decrease in ROC AUC compared to exclusive SMOTE utilization.

The best results for the F1-score and ROC AUC across different ML models after applying various sampling techniques are summarized in Table 13.

Table 13. The best result of the F1-score and ROC AUC for different ML models.

Metrics/ Methods	F1-Score	ROC AUC
DT	Initial = 77%	SMOTE-TOMEK = 74%
ANN	Initial = 80%	SMOTE = 83%
LR	Initial = 62%	Initial and SMOTE = 70%
SVM	SMOTE = 68%	SMOTE and SMOTE-TOMEK = 73%
RF	Initial and SMOTE-TOMEK = 81%	SMOTE-TOMEK = 78%
CatBoost	Initial = 90%	Initial = 90%
XGBoost	Initial = 92%	Initial and SMOTE-TOMEK = 91%
LightGBM	SMOTE = 92%	SMOTE = 90%

- ❖ Impact of Optuna Hyperparameter Tuning: After applying Optuna Hyperparameter Tuning, Cat-Boost outperformed XGBoost and LightGBM when Optuna was utilized for hyperparameter optimization, achieving an impressive F1-score of 93% and an ROC AUC of 91%. The enhanced F1-score and ROC AUC results observed after applying Optuna hyperparameter tuning to CatBoost, XGBoost, and LightGBM are likely attributable to improved hyperparameter configurations. Optuna fine-tuned these settings more effectively for the specific dataset, reducing overfitting and enhancing the models' capacity to generalize to new data. This ultimately resulted in improved overall model performance, as hyperparameters significantly influence the performance of these algorithms with your dataset.

In future work, several avenues can be explored. Firstly, other machine learning techniques, such as deep learning models, like Long Short-Term Memory (LSTM) or Transformer-based models, can be evaluated for churn prediction. These models have shown promise in various domains and may provide further insights into churn behavior. Secondly, we suggest exploring the use of the AdaSyn technique to handle imbalanced data and compare the results. Lastly, we recommend applying the above techniques to a highly imbalanced dataset to evaluate their performance in such conditions. Furthermore, employing the learning curve method to determine whether the models are overfitting could also be a valuable avenue of research.

Author Contributions: Writing and original draft preparation, subsequent revisions, coding, analysis, and interpretation of the results, M.I.; supervision, review, and editing, H.R.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: This study makes use of a publicly accessible dataset sourced from Kaggle [44].

Conflicts of Interest: The authors declare no conflict of interest.

References

1. *Cost of Customer Acquisition versus Customer Retention*; The Chartered Institute of Marketing: Cookham, UK, 2010.
2. Eichinger, F.; Nauck, D.D.; Klawonn, F. Sequence mining for customer behaviour predictions in telecommunications. In Proceedings of the Workshop on Practical Data Mining at ECML/PKDD, Berlin, Germany, 18–22 September 2006; pp. 3–10.
3. Prasad, U.D.; Madhavi, S. Prediction of churn behaviour of bank customers using data mining tools. *Indian J. Market.* **2011**, *42*, 25–30.
4. Keramati, A.; Ghaneei, H.; Mirmohammadi, S.M. Developing a prediction model for customer churn from electronic banking services using data mining. *Financ. Innov.* **2016**, *2*, 10. [CrossRef]
5. Scriney, M.; Dongyun, N.; Mark, R. Predicting customer churn for insurance data. In *International Conference on Big Data Analytics and Knowledge Discovery*; Springer: Cham, Switzerland, 2020.
6. De Caigny, A.; Coussement, K.; De Bock, K.W. A new hybrid classification algorithm for customer churn prediction based on logistic regression and decision trees. *Eur. J. Oper. Res.* **2018**, *269*, 760–772. [CrossRef]
7. Kim, K.; Jun, C.-H.; Lee, J. Improved churn prediction in telecommunication industry by analyzing a large network. *Expert Syst. Appl.* **2014**, *41*, 6575–6584. [CrossRef]
8. Ahmad, A.K.; Jafar, A.; Aljoumaa, K. Customer churn prediction in telecom using machine learning in big data platform. *J. Big Data* **2019**, *6*, 28. [CrossRef]
9. Jadhav, R.J.; Pawar, U.T. Churn prediction in telecommunication using data mining technology. *IJACSA Edit.* **2011**, *2*, 17–19.
10. Radosavljevik, D.; van der Putten, P.; Larsen, K.K. The impact of experimental setup in prepaid churn prediction for mobile telecommunications: What to predict, for whom and does the customer experience matter? *Trans. Mach. Learn. Data Min.* **2010**, *3*, 80–99.
11. Richter, Y.; Yom-Tov, E.; Slonim, N. Predicting customer churn in mobile networks through analysis of social groups. In Proceedings of the 2010 SIAM International Conference on Data Mining, Columbus, OH, USA, 29 April–1 May 2010; Volume 2010, pp. 732–741.
12. Amin, A.; Shah, B.; Khattak, A.M.; Moreira, F.J.L.; Ali, G.; Rocha, A.; Anwar, S. Cross-company customer churn prediction in telecommunication: A comparison of data transformation methods. *Int. J. Inf. Manag.* **2018**, *46*, 304–319. [CrossRef]
13. Tsiptsis, K.; Chorianopoulos, A. *Data Mining Techniques in CRM: Inside Customer Segmentation*; John Wiley & Sons: Hoboken, NJ, USA, 2011.
14. Joudaki, M.; Imani, M.; Esmaeili, M.; Mahmoodi, M.; Mazhari, N. Presenting a New Approach for Predicting and Preventing Active/Deliberate Customer Churn in Tel-ecommunication Industry. In Proceedings of the International Conference on Security and Management (SAM), Las Vegas, NV, USA, 18–21 July 2011; The Steering Committee of the World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp): Athens, GA, USA, 2011.
15. Amin, A.; Al-Obeidat, F.; Shah, B.; Adnan, A.; Loo, J.; Anwar, S. Customer churn prediction in telecommunication industry using data certainty. *J. Bus. Res.* **2019**, *94*, 290–301. [CrossRef]
16. Shaaban, E.; Helmy, Y.; Khedr, A.; Nasr, M. A proposed churn prediction model. *J. Eng. Res. Appl.* **2012**, *2*, 693–697.
17. Khan, Y.; Shafiq, S.; Naeem, A.; Ahmed, S.; Safwan, N.; Hussain, S. Customers Churn Prediction using Artificial Neural Networks (ANN) in Telecom Industry. *Int. J. Adv. Comput. Sci. Appl.* **2019**, *10*. [CrossRef]
18. Ho, T.K. Random decision forests. In Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, Canada, 14–16 August 1995; Volume 1.
19. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
20. Amin, A.; Shehzad, S.; Khan, C.; Ali, I.; Anwar, S. Churn Prediction in Telecommunication Industry Using Rough Set Approach. In *New Trends in Computational Collective Intelligence*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 83–95.
21. Witten, I.H.; Frank, E.; Hall, M.A. *Data Mining: Practical Machine Learning Tools and Techniques*; Elsevier Science & Technology: San Francisco, CA, USA, 2016.
22. Alok, K.; Mayank, J. *Ensemble Learning for AI Developers*; BApress: Berkeley, CA, USA, 2020.
23. van Wezel, M.; Potharst, R. Improved customer choice predictions using ensemble methods. *Eur. J. Oper. Res.* **2007**, *181*, 436–452. [CrossRef]
24. Ullah, I.; Raza, B.; Malik, A.K.; Imran, M.; Islam, S.U.; Kim, S.W. A Churn Prediction Model Using Random Forest: Analysis of Machine Learning Techniques for Churn Prediction and Factor Identification in Telecom Sector. *IEEE Access* **2019**, *7*, 60134–60149. [CrossRef]
25. Lalwani, P.; Mishra, M.K.; Chadha, J.S.; Sethi, P. Customer churn prediction system: A machine learning approach. *Computing* **2021**, *104*, 271–294. [CrossRef]
26. Tarekegn, A.; Ricceri, F.; Costa, G.; Ferracin, E.; Giacobini, M. Predictive Modeling for Frailty Conditions in Elderly People: Machine Learning Approaches. *Psychopharmacol.* **2020**, *8*, e16678. [CrossRef] [PubMed]
27. Ahmed, M.; Afzal, H.; Siddiqi, I.; Amjad, M.F.; Khurshid, K. Exploring nested ensemble learners using overproduction and choose approach for churn prediction in telecom industry. *Neural Comput. Appl.* **2018**, *32*, 3237–3251. [CrossRef]
28. Boser, B.E.; Guyon, I.M.; Vapnik, V.N. A training algorithm for optimal margin classifiers. In Proceedings of the Fifth Annual Workshop on Computational Learning Theory, Pittsburgh, PA, USA, 27–29 July 1992; ACM: New York, NY, USA, 1992; pp. 144–152.

29. Hur, Y.; Lim, S. Customer churning prediction using support vector machines in online auto insurance service. In *Advances in Neural Networks, Proceedings of the ISNN 2005, Chongqing, China, 30 May–1 June 2005*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 928–933.
30. Lee, S.J.; Siau, K. A review of data mining techniques. *Ind. Manag. Data Syst.* **2001**, *101*, 41–46. [CrossRef]
31. Mazhari, N.; Imani, M.; Joudaki, M.; Ghelichpour, A. An overview of classification and its algorithms. In Proceedings of the 3rd Data Mining Conference (IDMC'09), Tehran, Iran, 15–16 December 2009.
32. Linoff, G.S.; Berry, M.J. *Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management*; John Wiley & Sons: Hoboken, NJ, USA, 2011.
33. Zhou, Z.-H. *Ensemble Methods—Foundations and Algorithms*; CRC press: Boca Raton, FL, USA, 2012.
34. Karlberg, J.; Axen, M. *Binary Classification for Predicting Customer Churn*; Umeå University: Umeå, Sweden, 2020.
35. Windridge, D.; Nagarajan, R. Quantum Bootstrap Aggregation. In Proceedings of the International Symposium on Quantum Interaction, San Francisco, CA, USA, 20–22 July 2016; Springer: Berlin/Heidelberg, Germany, 2017.
36. Wang, J.C.; Hastie, T. Boosted Varying-Coefficient Regression Models for Product Demand Prediction. *J. Comput. Graph. Stat.* **2014**, *23*, 361–382. [CrossRef]
37. Al Daoud, E. Intrusion Detection Using a New Particle Swarm Method and Support Vector Machines. *World Acad. Sci. Eng. Technol.* **2013**, *77*, 59–62.
38. Al Daoud, E.; Turabieh, H. New empirical nonparametric kernels for support vector machine classification. *Appl. Soft Comput.* **2013**, *13*, 1759–1765. [CrossRef]
39. Al Daoud, E. An Efficient Algorithm for Finding a Fuzzy Rough Set Reduct Using an Improved Harmony Search. *Int. J. Mod. Educ. Comput. Sci. (IJMECS)* **2015**, *7*, 16–23. [CrossRef]
40. Zhang, Y.; Haghani, A. A gradient boosting method to improve travel time prediction. *Transp. Res. Part C Emerg. Technol.* **2015**, *58*, 308–324. [CrossRef]
41. Dorogush, A.; Ershov, V.; Gulin, A. CatBoost: Gradient boosting with categorical features support. In Proceedings of the Thirty-first Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 1–7.
42. Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; Liu, T.Y. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2017; Volume 30.
43. Klein, A.; Falkner, S.; Bartels, S.; Hennig, P.; Hutter, F. Fast Bayesian optimization of machine learning hyperparameters on large datasets. In Proceedings of the Machine Learning Research PMLR, Sydney, NSW, Australia, 6–11 August 2017; Volume 54, pp. 528–536.
44. Christy, R. Customer Churn Prediction 2020, Version 1. 2020. Available online: https://www.kaggle.com/code/rinichristy/customer-churn-prediction-2020 (accessed on 20 January 2022).
45. Kubat, M.; Matwin, S. Addressing the curse of imbalanced training sets: One-sided selection. In Proceedings of the 14th International Conference on Machine Learning, Nashville, TN, USA, 8–12 July 1997; Volume 97, p. 179.
46. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
47. Tomek, I. Two Modifications of CNN. *IEEE Trans. Syst. Man Cybern.* **1976**, *SMC-6*, 769–772. [CrossRef]
48. Wilson, D.L. Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. *IEEE Trans. Syst. Man Cybern.* **1972**, *2*, 408–421. [CrossRef]
49. Tyagi, S.; Mittal, S. Sampling Approaches for Imbalanced Data Classification Problem in Machine Learning. In Proceedings of the ICRIC 2019: Recent Innovations in Computing, Jammu, India, 8–9 March 2019; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; pp. 209–221.
50. Fawcett, T. An Introduction to ROC analysis. *Pattern Recogn. Lett.* **2006**, *27*, 861–874. [CrossRef]
51. Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; Koyama, M. Optuna: A next-generation hyperparameter optimization framework. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019.
52. Bergstra, J.; Yamins, D.; Cox, D. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In Proceedings of the 30th International Conference on Machine Learning, Atlanta, GA, USA, 17–19 June 2013.
53. Bergstra, J.; Bardenet, R.; Bengio, Y.; Kégl, B. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2011; Volume 24.
54. Hansen, N.; Ostermeier, A. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evol. Comput.* **2001**, *9*, 159–195. [CrossRef]
55. Li, L.; Jamieson, K.; Rostamizadeh, A.; Gonina, E.; Ben-Tzur, J.; Hardt, M.; Recht, B.; Talwalkar, A. A system for massively parallel hyperparameter tuning. *Proc. Mach. Learn. Syst.* **2020**, *2*, 230–246.