Review

Not peer-reviewed version

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

Mehdi Imani * and Hamid Reza Arabnia

Posted Date: 21 August 2023

doi: 10.20944/preprints202308.1478.v1

Keywords: Machine learning, Churn Prediction, Imbalanced Data, Combined Data Sampling Techniques, Hyperparameter Optimization.

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] Mehdi Imani, Department of Computer and System Sciences, Stockholm University, Stockholm, Sweden, m.imani@gmail.com

[2] Hamid R. Arabnia, School of Computing, University of Georgia, Athens, Georgia, United States, hra@uga.edu

* Correspondence: m.imani@gmail.com.

Abstract: In this paper, a variety of machine learning techniques, including Artificial Neural Networks, Decision Trees, Support Vector Machines, Random Forests, Logistic Regression, and three gradient boosting techniques (XGBoost, LightGBM, and CatBoost), were employed to predict customer churn in the telecommunications industry using a publicly available dataset. To address the issue of imbalanced data, various data sampling techniques, such as SMOTE, the combination of SMOTE with Tomek Links, and the combination of SMOTE with Edited Nearest Neighbors, were implemented. Additionally, hyperparameter tuning was utilized to optimize the performance of the machine learning models. The models were evaluated and compared using commonly used metrics, including Precision, Recall, F1-Score, and the Receiver Operating Characteristic Area Under Curve (ROC AUC). The results revealed that the performance of the models was enhanced by the application of hyperparameter tuning and the combined data sampling methods on the training data. Overall, LightGBM demonstrated superior performance compared to the other machine learning techniques examined. The findings indicate that LightGBM exhibited a superior performance both prior to and following the application of these techniques.

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

1. Introduction

The implementation of Customer Relationship Management (CRM) is a strategic approach to managing and enhancing relationships between businesses and their customers. 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 [6–15], to name a few. 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]. As a result, it is imperative for businesses to develop practical tools to achieve this goal. In recent years, various Machine Learning (ML) methods have been proposed for constructing a churn model, including Artificial Neural Networks (ANN) [8,9,16–18], Decision Trees [8,9,11,13,16,17], Random Forests [19,20], Logistic Regression (LR) [9,13], Support Vector Machines (SVM) [17], and Rough Set Approach [21], among others.

In the following paragraphs, an overview is provided of five of the most commonly utilized techniques. Additionally, three prominent boosting algorithms, namely eXtreme Gradient Boosting (XGBoost), Categorical Boosting (CatBoost), and Light Gradient Boosting Machine (LightGBM), were selected for use. Ensemble techniques [22], specifically boosting and bagging algorithms, have become the prevalent choice for addressing classification problems [23,24], particularly in the realm of churn prediction [25,26], due to their demonstrated high effectiveness.

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

2. Machine Learning Techniques

In the following, an overview is provided of the most frequently utilized techniques for addressing the issue of churn prediction, including Artificial Neural Network, Decision Tree, Support Vector Machine, Random Forest, Logistic Regression, and three advanced gradient boosting techniques, specifically XGBoost, LightGBM, and CatBoost.

A. Artificial Neural Network

Artificial Neural Network (ANN) is a widely employed technique for addressing complex issues such as the churn prediction problem [27]. 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) [27], and have been shown to exhibit improved performance when compared to Logistic Regression (LR) and DTs in the context of churn prediction [28].

B. Support Vector Machine

The technique of Support Vector Machine (SVM) was first introduced by authors in [29]. 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 [17,30].

For this study, we utilize 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:

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

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:

$$K(x.x') = (x^T x' + c)^d \tag{2}$$

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

C. Decision Tree

A Decision Tree (DT) is a representation of all potential decision pathways in the form of a tree structure [31,32]. 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" [33]. 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.

D. 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 (DTs) [10], it is often beneficial to apply pre-processing and transformation techniques to the original data prior to utilizing LR.

3. 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 [34]. In ensemble learning, it is possible to combine various weak machine learning models (base learners) to construct a stronger model with more accurate predictions [35,36]. Currently, ensemble learning methods are widely accepted as a standard choice for enhancing the accuracy of machine learning predictors [35]. Bagging and boosting are two distinct types of ensemble learning techniques that can be utilized to improve the accuracy of machine learning predictors [36].

A. Bagging

As depicted in Figure 1, in the bagging technique, the training data is 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) [36–38].

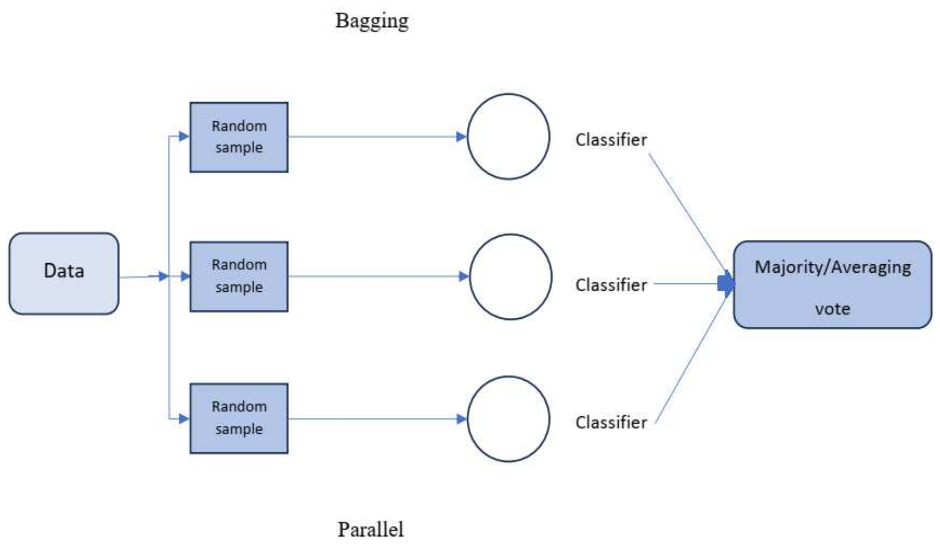

Figure 1. Visualization of the bagging approach.

B. *Random Forest*

The concept of Random Forest was first introduced by Ho in 1995 [19] and has been the subject of ongoing improvement by various researchers. One notable advancement in this field was made by Leo Breiman in 2001 [20]. 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. In general, Random Forests exhibit superior performance compared to Decision Trees (DTs), however, the performance can be influenced by the characteristics of the data.

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 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 $x^{'}$.

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

C. 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 2. Like the bagging technique, boosting improves the performance of machine learning predictors and in addition, it reduces the bias of the model [36].

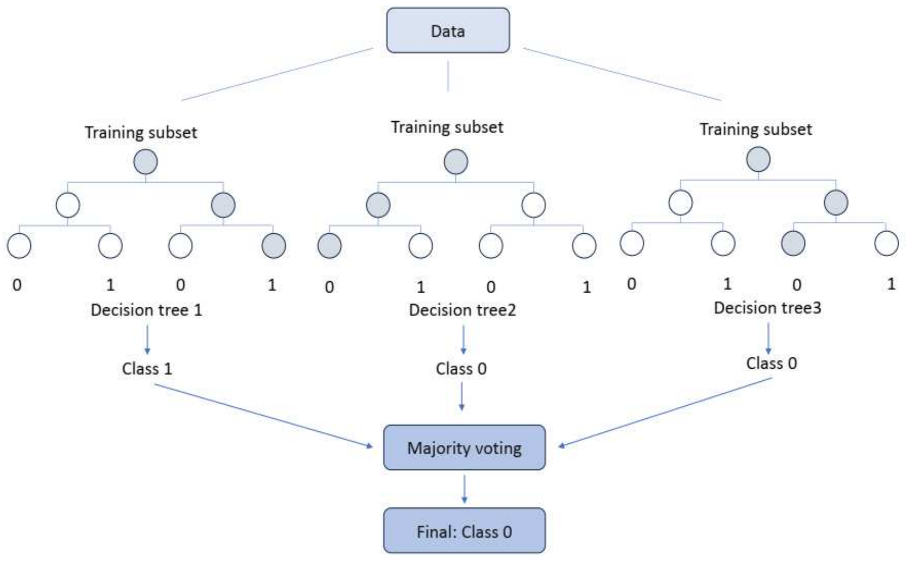

Figure 2. Visualization of the Random Forest classifier.

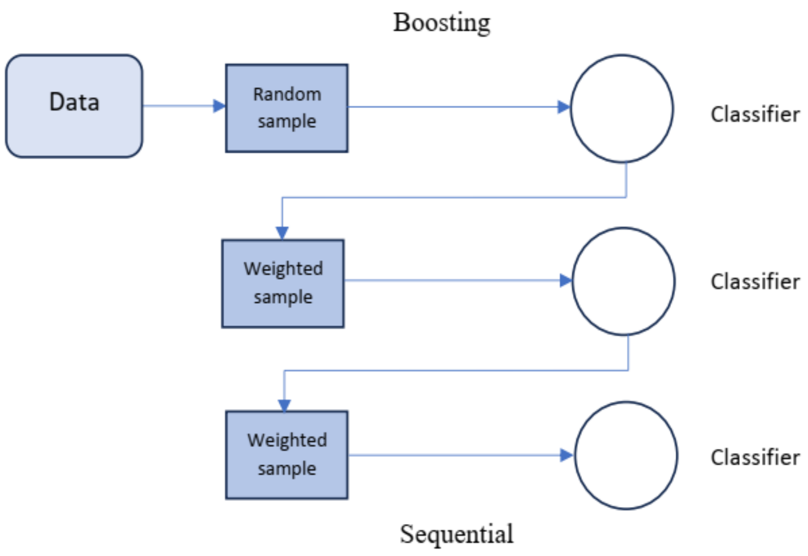

Figure 3. Visualization of the boosting approach.

D. The Famous Trio: XGBoost, LightGBM, 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 [39]. 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, θ)* (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, θt)*, where the increment is directed by:

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

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

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

The 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 $\hbar(x, \theta_i)$
        c. Find $p_i$ using eq()
        d. Update the function
               $\hat{f}_i = \hat{f}_{i-1} + p_i \hbar(x, \theta_i)$
3 − End
```

After initiating the algorithm with a single leaf, the learning rate is optimized for each record and each node [40–42]. 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:

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

with

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

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

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

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

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

where

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

and

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

The *Gain* represents the score of no new child case, *H* indicates the score of the left child, and *G* denotes the score of the right child [43].

To decrease the implementation time, the LightGBM method was developed by a team from Microsoft in April 2017 [8]. 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 faster training speed, higher accuracy, as well as the ability to handle large scale data and support GPU learning.

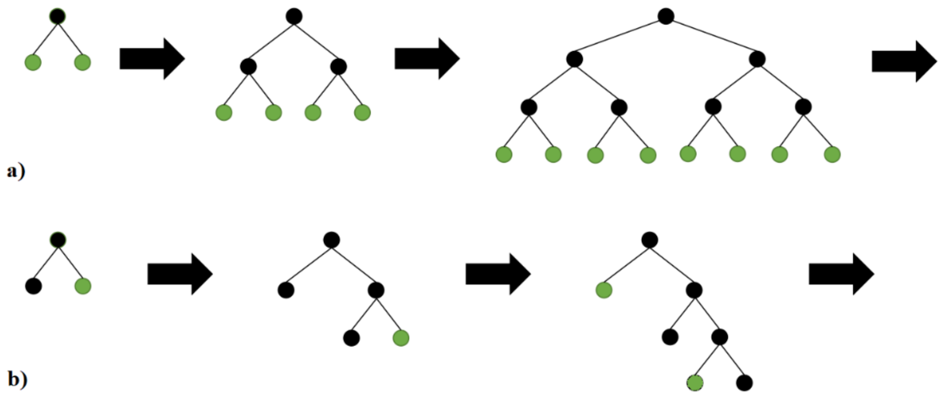

Figure 4. (**a**) XGBoost Level-wise tree growth and (**b**) LightGBM Leaf-wise tree growth.

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:

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

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* [44–46].

4. Handling Imbalanced Data

Imbalanced data is 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. This characteristic of the 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 and oversampling [47].

A. Sampling Techniques

Synthetic Minority Over-Sampling Technique (SMOTE) [48] 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.

Tomek Links is 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") [49]. The Tomek links method identifies pairs of examples (each from a different class) that have the minimum Euclidean distance to each other.

Edited Nearest Neighbors (ENN) is another undersampling method, proposed by Wilson (in his 1972 paper titled "Asymptotic Properties of Nearest Neighbor Rules Using Edited Data") [50]. 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.

Applying just one undersampling or oversampling method to the training data can effectively handle imbalanced data, but to achieve the best results, it is advisable to use combination techniques. 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.

B. 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 is 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 was 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 a 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 at 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.

A. 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 (TP) 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 (TN) 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 [51]. Therefore, other metrics such as recall, precision, and F-measure 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 1.

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.

Table 1. The confusion matrix for evaluating methods.

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

In Table 1, 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:

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

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

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

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 F-measure metric as a single metric to evaluate the accuracy of our models. F-measure 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. F-measure is defined as follows:

$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{12}$$

The more the value of the F-measure is closer to 1, the better combination of Precision and Recall is achieved by the model [52].

B. 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:

$$TruePositiveRate = \frac{TP}{TP + FN} \qquad (13)$$

And the false positive rate is calculated as follows:

$$FalsePositiveRate = \frac{FP}{FP + TN} \qquad (14)$$

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.

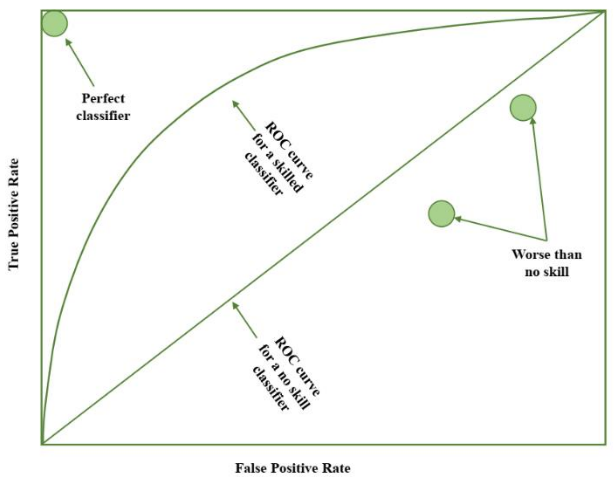

Figure 5. The ROC curve.

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. 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, as it focuses on the performance of the classifier on the minority class.

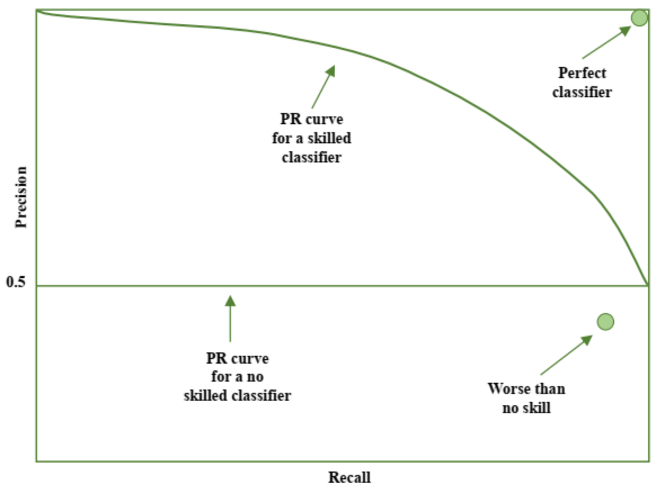

Figure 6. The Precision-Recall curve.

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 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.

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 and in a way that takes into account the specific characteristics of the data.

C. ROC AUC Benchmark

It is clear that a 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 2, the worst possible AUC is 50%, which is similar to the result of a coin flip for prediction.

Table 2. ROC AUC benchmark for predicting churn.

ROC AUC<= 50%	Something Is Wrong *
50%<= ROC AUC <60%	Similar to flipping a coin
60%<= ROC AUC <70%	Weak prediction
70%<= ROC AUC <80%	Good Prediction
80%<= ROC AUC <90%	Very Good Prediction
ROC AUC >= 90%	Excellent Prediction

* Check the data and the AUC calculation.

6. Simulation

A. 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 3. 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 precision, recall, F-measure, 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 3. 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*

7. 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) on unseen data using a range of metrics including 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 on unseen data.

A. Applying 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 4, with the highest values highlighted in bold.

Table 4. Evaluation metrics for the different models after applying 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
CatB	90	90	90	90
LGBM	94	**91**	**92**	**91**
XGB	**96**	87	91	87

As depicted in Table 4, the Random Forest and XGBoost models exhibit superior performance in terms of precision compared to other machine learning algorithms. However, in terms of recall, F1-Score, and ROC AUC, the LightGBM model outperforms the other methods. Figure 7 shows the diagram of the ROC Curve for the different models after feature selection.

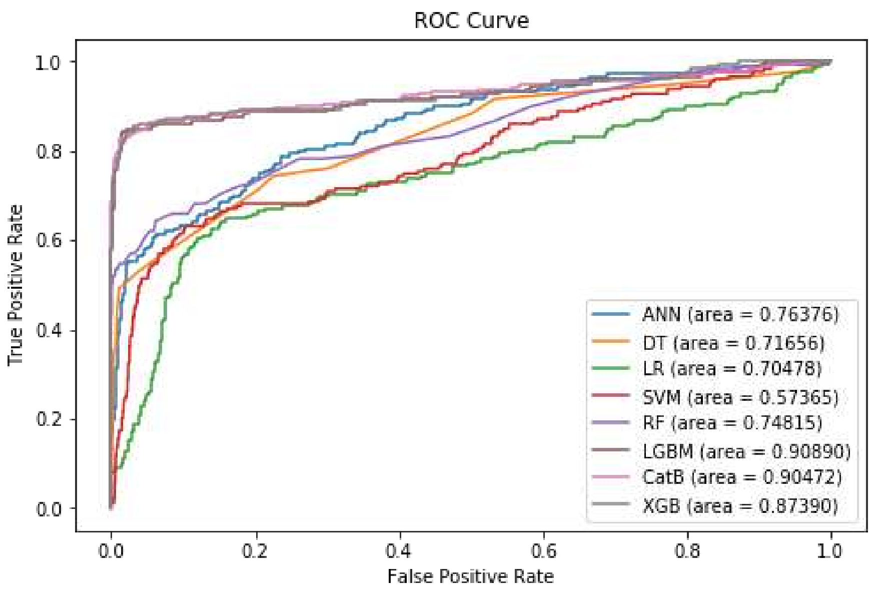

Figure 7. ROC Curve after applying feature selection.

B. 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 5 shows, LightGBM and XGBoost outperform other ML techniques in all evaluation metrics. Figure 8 shows the diagram of the ROC Curve for the different models after applying SMOTE.

Table 5. 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
CatB	79	88	83	88
LGBM	87	**90**	88	**90**
XGB	**95**	**90**	**92**	**90**

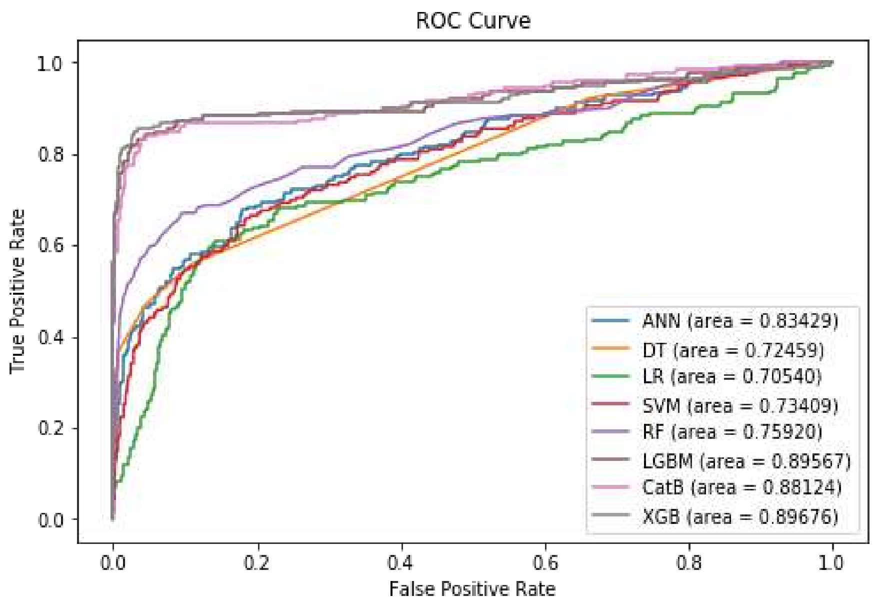

Figure 8. ROC Curve after applying SMOTE.

C. 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 have 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 6. As evidenced by the table, both LightGBM and XGBoost outperform the other machine learning methods and demonstrate slightly improved results when compared to utilizing SMOTE alone, as shown in Table 5. 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 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
CatB	80	88	83	88
LGBM	89	**91**	90	**91**
XGB	**94**	89	**91**	89

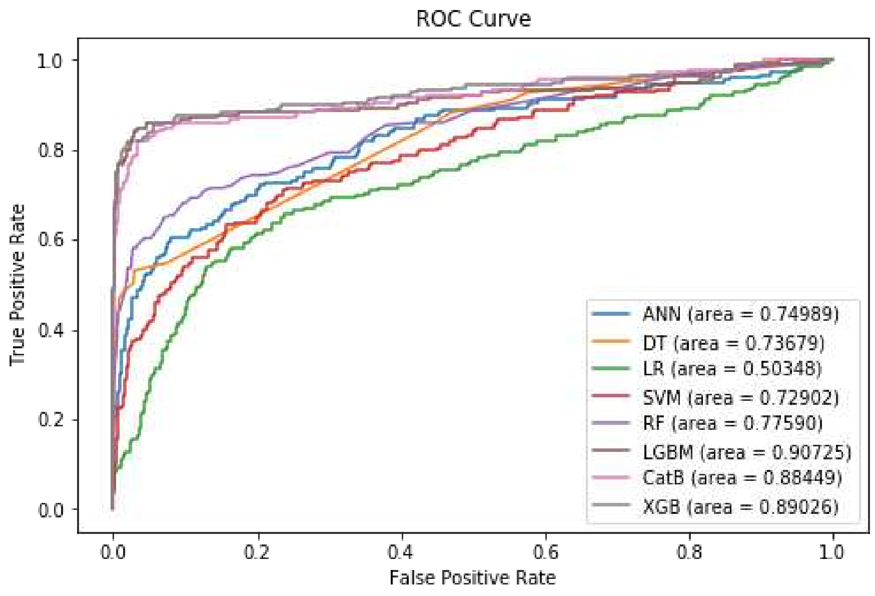

Figure 9. ROC Curve after applying SMOTE with Tomek Links.

D. Applying SMOTE with ENN

As previously discussed in Section IV, 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 7 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 LightGBM and XGBoost models outperform other machine learning techniques in all evaluation metrics once more. Figure 10 shows the diagram of the ROC Curve for the different models after applying SMOTE with Tomek Links.

Table 7. 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	13	50
SVM	60	70	58	70
RF	67	76	69	76
CatB	70	83	72	83
LGBM	80	**89**	84	87
XGB	**88**	**89**	**88**	**89**

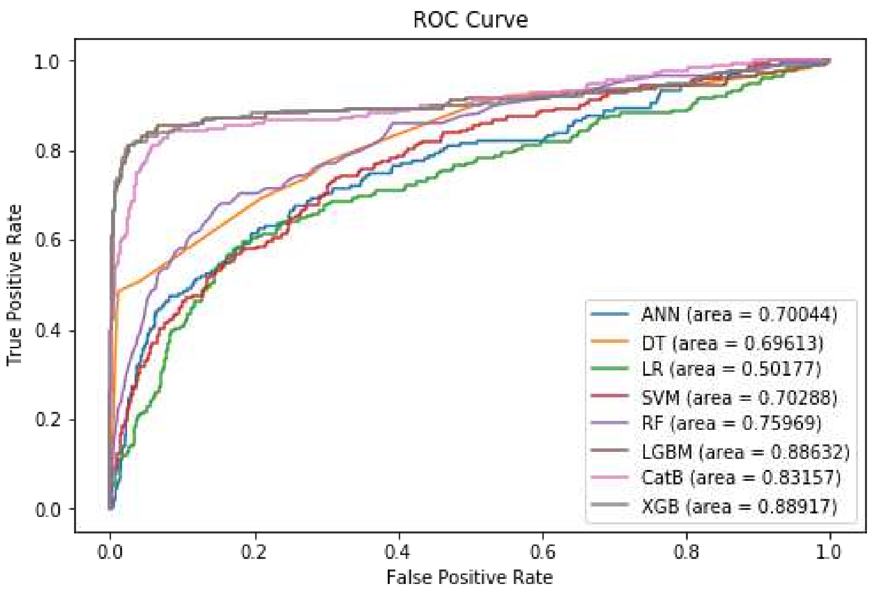

Figure 10. ROC Curve after applying SMOTE with ENN.

E. Applying OPTUNA Hyperparameter Optimizer

Takuya Akiba et al. (2019) [53] 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 of Parzen Estimator (TPE) [54,55] for independent parameter sampling, Covariance Matrix Adaptation (CMA) [56], and Gaussian Processes (GP) [55] for relational parameter sampling. The library also utilizes a variant of the Asynchronous Successive Halving (ASHA) algorithm [57] to prune search spaces. In this study, we applied the Optuna library to the popular machine learning models, CatBoost, XGBoost, and LightGBM. The results, as presented in Table 8, indicate that CatBoost outperforms XGBoost and LightGBM when utilizing Optuna for hyperparameter optimization. Figure 11 shows the diagram of the ROC Curve for the different models after applying OPTUNA hyperparameter tuning.

Table 8. Evaluation metrics for various models after the application of Optuna hyperparameter optimization.

Models	Precision%	Recall%	F1-Score%	ROC AUC%
CatB	89	**91**	90	**91**
CatB-Optuna	**95**	**91**	**93**	**91**
LGBM	92	90	91	90
LGBM-Optuna	93	89	90	89
XGB	93	88	90	88
XGB-Optuna	94	88	91	88

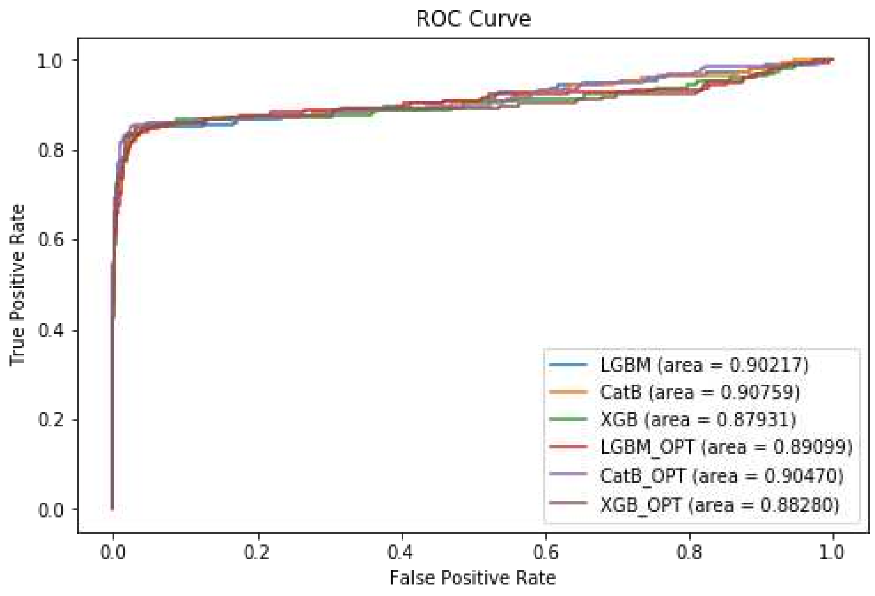

Figure 11. ROC Curve after OPTUNA hyperparameter tuning.

8. Conclusion

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 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 on unseen data. LightGBM and XGBoost consistently exhibit superior performance across various evaluation metrics, including 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 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, 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: Not applicable.

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

References

1. The Chartered Institute of Marketing, "Cost of Customer Acquisition versus Customer Retention", 2010.
2. F. Eichinger, D.D. Nauck, F. Klawonn, "Sequence mining for customer behaviour predictions in telecommunications", in: Proceedings of the Workshop on Practical Data Mining at ECML/PKDD, 2006, pp. 3–10.
3. U.D. Prasad, S. Madhavi, "Prediction of churn behaviour of bank customers using data mining tools", Indian J. Market. 42 (9) (2011) 25–30.
4. Keramati, Abbas, Hajar Ghaneei, and Seyed Mohammad Mirmohammadi. "Developing a prediction model for customer churn from electronic banking services using data mining." Financial Innovation 2.1 (2016): 1-13.
5. Scriney, Michael, Dongyun Nie, and Mark Roantree. "Predicting customer churn for insurance data." International Conference on Big Data Analytics and Knowledge Discovery. Springer, Cham, 2020.
6. De Caigny, Arno, Kristof Coussement, and Koen W. De Bock. "A new hybrid classification algorithm for customer churn prediction based on logistic regression and decision trees." European Journal of Operational Research 269.2 (2018): 760-772.
7. K. Kim, C.-H. Jun, J. Lee, "Improved churn prediction in telecommunication industry by analyzing a large network", Expert Syst. Appl. 41 (15) (2014) 6575–6584.
8. Ahmad, Abdelrahim Kasem, Assef Jafar, and Kadan Aljoumaa. "Customer churn prediction in telecom using machine learning in big data platform." Journal of Big Data 6.1 (2019): 1-24.
9. De Caigny, Arno, Kristof Coussement, and Koen W. De Bock. "A new hybrid classification algorithm for customer churn prediction based on logistic regression and decision trees." European Journal of Operational Research 269.2 (2018): 760-772.
10. R.J. Jadhav, U.T. Pawar, "Churn prediction in telecommunication using data mining technology", IJACSA Edit. 2 (2) (2011) 17–19.
11. D. Radosavljevik, P. van der Putten, K.K. Larsen, "The impact of experimental setup in prepaid churn prediction for mobile telecommunications: what to predict, for whom and does the customer experience matter?", Trans MLDM 3 (2) (2010) 80–99.
12. Y. Richter, E. Yom-Tov, N. Slonim, "Predicting customer churn in mobile networks through analysis of social groups", SDM, vol. 2010, SIAM, 2010, pp. 732–741.
13. Amin, Adnan, et al. "Cross-company customer churn prediction in telecommunication: A comparison of data transformation methods." International Journal of Information Management 46 (2019): 304-319.
14. K. Tsiptsis, A. Chorianopoulos, "Data Mining Techniques in CRM: Inside Customer Segmentation", John Wiley & Sons, 2011.
15. Joudaki, Majid, et al. "Presenting a New Approach for Predicting and Preventing Active/Deliberate Customer Churn in Telecommunication Industry." Proceedings of the International Conference on Security and Management (SAM). The Steering Committee of the World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2011.
16. Amin, Adnan, et al. "Customer churn prediction in telecommunication industry using data certainty." Journal of Business Research 94 (2019): 290-301.
17. E. Shaaban, Y. Helmy, A. Khedr, M. Nasr, "A proposed churn prediction model", J. Eng. Res. Appl. 2 (4) (2012) 693–697.
18. Khan, Yasser, et al. "Customers churn prediction using artificial neural networks (ANN) in telecom industry." Editorial Preface From the Desk of Managing Editor 10.9 (2019).
19. Ho, Tin Kam. "Random decision forests." Proceedings of 3rd international conference on document analysis and recognition. Vol. 1. IEEE, 1995.
20. Breiman, Leo. "Random forests." Machine learning 45.1 (2001): 5-32.
21. A. Amin, S. Shehzad, C. Khan, I. Ali, S. Anwar, "Churn prediction in telecommunication industry using rough set approach, in: New Trends in Computational Collective Intelligence", Springer, 2015, pp. 83–95.
22. I. H. Witten, E. Frank, M. A. Hall and C. J. Pal, Data Mining : Practical Machine Learning Tools and Techniques, San Francisco: Elsevier Science & Technology, 2016.
23. A. Kumar and M. Jain, Ensemble Learning for AI Developers: Learn Bagging, Stacking, and Boosting Methods with Use Cases, Apress, 2020.
24. M. Van Wezel and R. Potharst, "Improved customer choice predictions using ensemble methods," European Journal of Operational Research, vol. 181, no. 1, pp. 436-452, 2007.
25. I. Ullah, B. Raza, A. K. Malik, M. Imran, S. U. Islam and S. W. Kim, "A churn prediction model using random forest: analysis of machine learning techniques for churn prediction and factor identification in telecom sector," IEEE Access, pp. 60134-60149, 2019

26. P. Lalwani, M. M. Kumar, J. Singh Chadha and P. Sethi, "Customer churn prediction system: a machine learning approach," Computing, pp. 1-24, 2021.
27. Tarekegn, Adane, et al. "Predictive modeling for frailty conditions in elderly people: machine learning approaches." JMIR medical informatics 8.6 (2020): e16678.
28. Ahmed, Mahreen, et al. "Exploring nested ensemble learners using overproduction and choose approach for churn prediction in telecom industry." Neural Computing and Applications 32.8 (2020): 3237-3251.
29. B.E. Boser, I.M. Guyon, V.N. Vapnik, "A training algorithm for optimal margin classifiers", in Proceedings of the Fifth Annual Workshop on Computational Learning Theory", ACM, 1992, pp. 144–152.
30. Y. Hur, S. Lim, "Customer churning prediction using support vector machines in online auto insurance service, in: Advances in Neural Networks" – ISNN 2005, Springer, 2005, pp. 928–933.
31. S.J. Lee, K. Siau, A review of data mining techniques, Ind. Manage. Data Syst. 101 (1) (2001) 41–46.
32. Mazhari, N.,Imani, M., Joudaki, M. and Ghelichpour, A.,"An overview of classification and its algorithms" 3rd Data Mining Conference (IDMC'09): Tehran, 2009.
33. G.S. Linoff, M.J. Berry, "Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management", John Wiley & Sons, 2011.
34. Z.-H. Zhou, Ensemble Methods - Foundations and Algorithms, Taylor & Francis group, LLC, 2012.
35. A. Kumar and M. Jain, Ensemble Learning for AI Developers: Learn Bagging, Stacking, and Boosting Methods with Use Cases, Apress, 2020.
36. I. H. Witten, E. Frank, M. A. Hall and C. J. Pal, Data Mining : Practical Machine Learning Tools and Techniques, San Francisco: Elsevier Science & Technology, 2016.
37. J. Karlberg and M. Axen, "Binary Classification for Predicting Customer Churn," Umeå University, Umeå, 2020.
38. D. Windridge and R. Nagarajan, "Quantum Bootstrap Aggregation," in *International Symposium on Quantum Interaction*, 2017.
39. J. C. Wang, T. Hastie, "Boosted varying-coefficient regression models for product demand prediction," *Journal of Computational and Graphical Statistics*, vol. 23, no. 2, pp 361–382, 2014.
40. E Al Daoud, "Intrusion Detection Using a New Particle Swarm Method and Support Vector Machines," *World Academy of Science, Engineering and Technology*, vol. 77, 59-62, 2013.
41. E. Al Daoud, H Turabieh, "New empirical nonparametric kernels for support vector machine classification," *Applied Soft Computing*, vol. 13, no. 4, 1759-1765, 2013.
42. E. Al Daoud, "An Efficient Algorithm for Finding a Fuzzy Rough Set Reduct Using an Improved Harmony Search," *I.J. Modern Education and Computer Science*, vol. 7, no. 2, pp16-23, 2015.
43. Y. Zhang, A. Haghani. "A gradient boosting method to improve travel time prediction. Transportation Research Part C," *Emerging Technologies*, vol. 58,308–324,2015.
44. A. Dorogush, V. Ershov, A. Gulin "CatBoost: gradient boosting with categorical features support," *NIPS*, p1-7, 2017.
M. Qi, K. Guolin, W. Taifeng, C. Wei, Y. Qiwei, M. Weidong, L. TieYan, "A Communication-Efficient Parallel Algorithm for Decision Tree," *Advances in Neural Information Processing Systems*, vol. 29, pp. 1279-1287, 2016.
45. A. Klein, S. Falkner, S. Bartels, P. Hennig, F. Hutter, "Fast Bayesian optimization of machine learning hyperparameters on large datasets," *In Proceedings of Machine Learning Research PMLR*, vol. 54, pp 528-536,2017.
46. Kubat, Miroslav, and Stan Matwin. "Addressing the curse of imbalanced training sets: one-sided selection." *Icml*. Vol. 97. No. 1. 1997.
47. Chawla, Nitesh V., et al. "SMOTE: synthetic minority over-sampling technique." Journal of artificial intelligence research 16 (2002): 321-357.
48. Tomek, Ivan. "Two modifications of CNN." (1976).
49. Wilson, Dennis L. "Asymptotic properties of nearest neighbor rules using edited data." *IEEE Transactions on Systems, Man, and Cybernetics* 3 (1972): 408-421.
50. S. Tyagi and S. Mittal, "Sampling Approaches for Imbalanced Data Classification Problem in Machine Learning," in Proceedings of ICRIC 2019, 2020.
51. T. Fawcett, "An introduction to roc analysis", Pattern Recogn. Lett. 27 (8) (2006) 861–874.
52. Akiba, Takuya, et al. "Optuna: A next-generation hyperparameter optimization framework." *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019.
53. Bergstra, James, Daniel Yamins, and David Cox. "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures." Proceedings of The 30th International Conference on Machine Learning. 2013.
54. Bergstra, James S., et al. "Algorithms for hyper-parameter optimization." Advances in Neural Information Processing Systems. 2011.
55. Hansen, Nikolaus, and Andreas Ostermeier. "Completely derandomized self-adaptation in evolution strategies." *Evolutionary computation* 9.2 (2001): 159-195.

56. Li, Liam, et al. "A system for massively parallel hyperparameter tuning." Proceedings of Machine Learning and Systems 2 (2020): 230-246.