IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 27 May 2022, accepted 19 June 2022, date of publication 22 June 2022, date of current version 30 June 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3185227

A Sampling-Based Stack Framework for Imbalanced Learning in Churn Prediction

SOUMI DE[1] (Member, IEEE), AND P. PRABU[2]
[1]Department of Data Science, CHRIST (Deemed to be University), Bengaluru 560029, India
[2]Department of Computer Science, CHRIST (Deemed to be University), Bengaluru 560029, India

Corresponding authors: Soumi De (soumi.de@res.christuniversity.in) and P. Prabu (prabu.p@christuniversity.in)

ABSTRACT Churn prediction is gaining popularity in the research community as a powerful paradigm that supports data-driven operational decisions. Datasets related to churn prediction are often skewed with imbalanced class distribution. Data-level solutions, like over-sampling and under-sampling, have been commonly used by researchers to address this problem. There are limited number of case studies that attempt to evolve these data-level solutions by integrating them with computationally advanced frameworks, like ensembles. Ensembles primarily employ algorithmic diversity using a fixed set of training instances to achieve superior performance. This study aims to introduce algorithmic diversity in ensembles by modifying the fixed set of training instances using diverse sampling strategies to increase predictive performance in imbalanced learning. Data is acquired from the world's largest open hotel commerce platform company. A four-part series of experiments is conducted to analyze the effectiveness of sampling techniques and ensemble solutions on model performance. A new sampling-based stack framework called ''Stacking of Samplers for Imbalanced Learning'' is proposed. The framework combines the prediction capabilities of sampling solutions to stimulate the information gain of the meta features in ensemble. It is observed that the proposed framework leads to improvement in model performance with AUC of 86.4% and top-decile lift of 4.7 for customers of the hotel technology provider. Additionally, results show that the framework records a higher information gain for meta features used in a stack, compared to commonly used stack frameworks.

INDEX TERMS Churn prediction, ensemble classifiers, over-sampling, under-sampling, ensemble stack.

I. INTRODUCTION

Churn prediction aims to identify the consumers who are likely to terminate their service contract with a service provider. The intention behind a customer's churn decision may be involuntary or voluntary in nature. A consumer in financial distress may not have a reputable payment history, forcing the service provider to cancel the service contract. This is an example of involuntary churn. When a consumer actively chooses to leave a brand because of a price advantage from a competitor, or poor customer service standards, this is called voluntary churn.

A typical churn prediction workflow that uses machine learning techniques follows the process of data collection and cleansing, feature selection, model application, and finally, churn prediction. Literature on churn prediction focuses mainly on one or more of the above-mentioned processes.

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Liu.

Churn datasets mostly use demographic and product-usage related features. However, recent studies have incorporated a new set of attributes that are derived from social network of a consumer. Predicting churn, like any other discipline, is associated with a few challenges. Class imbalance is one such challenge that has not received much attention to clear ground for academic innovation. An imbalanced dataset, unfortunately, is not ideal for many learning systems. Application of data-level solutions, like sampling strategies, provides no conclusive evidence on the superiority of one technique over the other. Effectiveness of these techniques primarily depends on use-case under consideration, and further studies are required to evolve these data-level solutions into a more generic form.

The paper proposes a new framework called Stacking of Samplers for Imbalanced Learning (SS-IL) that aims to incorporate and integrate the prediction capabilities of sampling methods, into a single ensemble framework via stacking. The study performs a comparative analysis of classifier

performance to establish superiority of SS-IL by applying sampling strategies to an imbalanced churn dataset. The dataset is sourced from a hotel commerce platform company. Results are encouraging and show that the framework leads to enhanced performance of churn classifiers. The remaining paper is organized as follows. Section 2 presents related literature on sampling techniques and ensemble methods that are applied to imbalanced datasets. Section 3 describes the methodology used in the study. This is followed by section 4, where details on the model evaluation measures are discussed. Section 5 provides an overview on the experimental setup. Results are presented in section 6. Finally, section 7 contains a brief conclusion.

II. RELATED WORK

Improving the performance of classifiers has always been the main objective of churn prediction-related experiments. However, most churn data sets are class imbalanced. Imbalanced datasets are defined as those in which there is a significant difference between distribution of positive class and negative class. As a result, sampling techniques are used to balance the target class while training the model. Sampling strategies help to balance the training data to facilitate seamless integration with machine learning classifiers. Additionally, researchers have also used ensemble techniques to improve predictive capability of models. These techniques are discussed in the following section.

A. SAMPLING TECHNIQUES

There are 2 main types of sampling strategies implemented for balancing training data – over-sampling and under-sampling. Over-sampling methods are a group of techniques for balancing the class distribution by either creating copies of the minority class, or by generating synthetic samples of the minority class. Random Over-Sampling (ROS) is the simplest over-sampling technique to balance skewed datasets. However, ROS often leads to overfitting of classifiers. Synthetic Minority Oversampling Technique (SMOTE) is another over-sampling strategy that creates synthetic samples of minority class based on k nearest neighbor principle [1], [2]. A drawback of SMOTE is that the synthetically generated samples may lead to noise in the balanced dataset. A variation of SMOTE is k-means SMOTE. This algorithm over-samples the minority instances in clusters that have the highest number of minority samples. K-means SMOTE addresses the problem of noise in a balanced dataset. In another algorithm, called Adaptive Synthetic Sampling (ADASYN), areas with low density of minority samples are given higher priority and synthetic minority class instances are generated in these areas [3]. A detailed study conducted in [4] presents a comparison of 85 versions of over-sampling strategies using 104 imbalanced datasets for evaluation. The findings reveal that the type of evaluation metric utilized has an impact on a model's performance. Additionally, the study helps to establish a baseline for defining key principles that lead to best performance of a model.

Under-sampling methods are another set of techniques that aim to balance the class distribution by reducing the number of samples from the majority class. Random Under-Sampling (RUS) is the simplest method in this group that randomly selects and removes instances from the majority class. The main disadvantage of RUS is loss of information due to random instance selection. Additionally, there are heuristic methods of under-sampling that are based on sample significance and information content. Near Miss Under-sampling [5] is one of the heuristic sampling strategies that selects samples of majority class on the basis of its average Euclidean distance from samples of minority class. Another algorithm, called Condensed Nearest Neighbor (CNN), seeks to create a subset of the main dataset to preserve maximum information of the data [6], [7]. Tomek link, on the other hand, is a modified version of CNN. It identifies pairs of samples belonging to opposite classes that have the minimum Euclidean distance in the defined feature space. These pairs are called Tomek-links [8], [9]. The majority class samples are then removed from the identified Tomek- links. One sided selection (OSS) [10], Neighborhood Cleaning Rule (NCL) [11], are few other popular under-sampling methods. Main drawback of these heuristic techniques is the lack of control over the number of majority instances to retain or delete in the dataset. In cases where the majority class is not diversely represented in feature space, application of heuristic under-sampling methods becomes a challenge.

Sampling strategies have been widely applied for churn prediction. In a study, researchers proposed an improved version of SMOTE to address the problem of class imbalance in a telecom dataset [12]. They utilized a multi-objective rain optimization algorithm to determine the best sampling rate for SMOTE. On the other hand, an under-sampling balancing technique was adopted in [13] where majority class is under-sampled in each iteration of cross validation. Each iteration used different independent groups of under-sampled majority class.

Although application of sampling technique is common in churn prediction, their effectiveness mostly depends on the use-case under consideration. There is no conclusive evidence on the superiority of one technique over the other. As a result, there is a need to evolve these data-level solutions into a more integrated and generic form.

B. ENSEMBLE METHODS

Ensemble methods are very popular for enhancing a classifier's prediction capability. These methods are based on the paradigm of combining multiple classifiers for optimizing quality of prediction [14]. The framework utilizes the strength of each classifier in the ensemble, and minimizes their weaknesses, for better classification results. A general ensemble framework is presented in Fig. 1 (a). An ensemble consists of several base learners (BL) in the first level i.e., level 0. The base learner decisions are further aggregated using aggregation rules like majority voting, followed by classification decision. Bagging and boosting are common examples of

ensemble techniques. There are numerous studies that provide evidence of improved performance in classification task by using ensemble framework, as compared to standalone models [15], [16].

Stacking is an advanced and more refined version of ensemble framework, where output of base learners is combined and used as features for aggregation and classification. It incorporates a multilevel hierarchical approach where the number of levels, number of models, the selected set of base learners (BL), can vary and are design-dependent [17]. Fig. 1 (b) is an example of a stacking framework that comprises of n base learners in level 0. The base learner decisions are further aggregated at level 1 and utilized by another learner, called meta classifier, as attributes.

Ensemble methods and stacking have been deployed in few churn-related studies. Results are indicative of enhanced predictive capability of the framework [18]. However, in order to optimize the performance of an ensemble or stacking framework, it is important to focus on maximizing the ensemble diversity. This is a fundamental problem of ensemble methods. While ensemble diversity is not easy to quantify, information gain serves as an alternative that can be utilized to measure ensemble effectiveness. This study is an attempt to promote information gain in stacking by increasing span of attributes, using sampling strategies.

III. METHODOLOGY

A. DATASET

The dataset is obtained from world's largest open hotel commerce platform company and is private in nature. It consists of 78 independent variables and 1 dependent variable that represents a binary target class. The attributes in this dataset comprise of a customer's demographic and product usage variables. Features that represent geographic location of a customer are one-hot encoded. Values present in numeric attributes are discretized to tackle outliers. There are 19,542 customer instances, with the minority class representing 14% of the dataset. The study is the first to utilize this dataset for churn analysis.

A time window technique is followed for deriving the target variable. Customer characteristics are tracked and used as features for a 9-month period in year 2021. In case a customer is active throughout this period, the instance is classified as a non-churner, or belongs to the negative class. If the customer discontinues service contract in this period, the instance is classified as a churner i.e., it belongs to the positive class. Independent variables related to product usage are derived based on the month of discontinuation, in case of churners; or the last day of the 9-month window, in case of non-churners.

B. PROPOSED FRAMEWORK SS-IL

A sampling-based stack framework called SS-IL is proposed for churn prediction. The framework exploits the ensemble paradigm for improving classifier performance. As discussed before, ensemble learning primarily fuses the decisions of several base classifiers to finally classify instances. Stacking is a special case of ensemble learning in which there are several base learners, or level 0 learners. The base learners are trained using the same training set [19]. However, in the proposed framework, varied training sets are used for level-0 classifiers, with an aim to increase the span of attributes and promote information gain in the ensemble through sampling. Stack framework is further characterized by the presence of an additional meta learner that is trained using the predictions of the level 0 learners. The meta learner learns the combination weights for all base level decision probabilities, and classifies instances. For a stack ensemble to perform well, it is important to promote information gain of features used for training the meta learner through level 0 base learners [19], [20]. The proposed framework is motivated by this rationale.

1) BASE LEARNERS

There are 6 models that are selected in this framework as base learners, namely, Random Forest (RF), k nearest neighbor (KNN), AdaBoost, Support Vector Machine (SVM-rbf kernel), Decision Tree (DT), and Logistic Regression (LR). The justification behind selection of these models as base learners is mainly to incorporate maximum diversity in the ensemble keeping in mind the varied nature of the algorithms. Each model is briefly described in the following section.

a: RANDOM FOREST (RF)

RF is an ensemble-centric classifier that fuses the decisions of individual decision trees [21]. Classification is based on the following steps:

Step 1: The algorithm uses bootstrap sampling to generate n training subsets

$$D_{train} = \{T_1, T_2, \ldots, T_n\}$$

Step 2: An out-of-bag (OOB) dataset is generated for each training subset

$$D_{OOB} = \{OOB_1, OOB_2, \ldots, OOB_n\}$$

such that $T_i \cap OOB_i = \phi$

Step 3: For each training subset T_i, decision trees are formed using splits of smaller set of independent variables. The process of splitting is repeated until a leaf node is reached. This leads to n decision trees that constitute the random forest.

Step 4: For a new unseen sample X, each decision tree casts a vote predicting its class. Finally, RF assigns a class to the unseen instance for which it received maximum votes from decision trees.

b: K NEAREST NEIGHBOR (KNN)

KNN is a distance-based supervised learning algorithm. It assigns a class to an unseen instance X, by calculating the Euclidean distance between X and its k nearest training neighbors [22]. The formula for calculating Euclidean distance between two vectors $P = (x_1, x_2, \ldots x_m)$ and

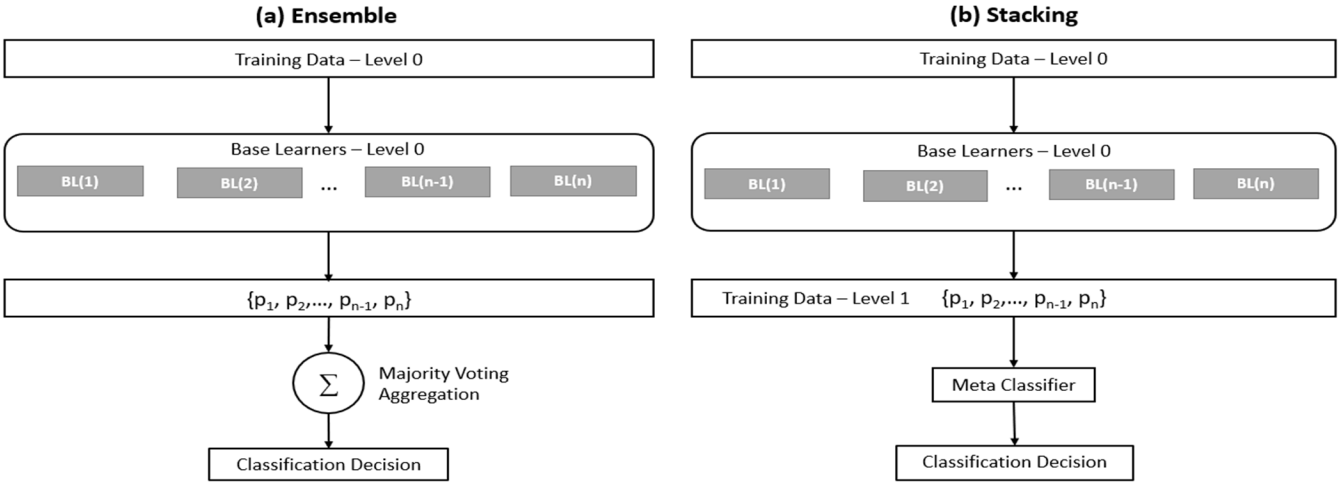

FIGURE 1. (a) Ensemble (b) Stacking.

$Q = (y_1, y_2, \ldots y_m)$ is given by (1):

$$dist(P,Q) = [(\sum (x_i - y_i)^2) \div m]^{1/2} \quad (1)$$

Here, m denotes the number of features that represent the sample and $i = \{1, 2, \ldots, m\}$. X is assigned the majority class of its k nearest neighbors.

c: ADAPTING BOOSTING (AdaBoost)

AdaBoost is an ensemble learner that uses the boosting principle to classify instances. The model continuously learns from mis-classified instances by assigning them more weight during training iterations to improve predictive power [23]. The final class for unseen instance X is calculated as a weighted majority vote of base classifiers

$$F(X) = \sum \alpha_t c_t(X) \quad (2)$$

where $t = \{1, 2, \ldots, T\}$, denotes identifier of base classifier of the algorithm, and $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_T\}$ represents the weight of base classifiers satisfying condition $\sum_{i=1}^{T} \alpha = 1$. The decision of t^{th} base classifier for instance X is represented by $c_t(X)$.

d: SUPPORT VECTOR MACHINE (SVM)

SVM is a learning algorithm that classifies data points using a hyperplane defined by a set of support vectors forming the best possible decision boundary separating classes with maximal margin [24]. The hyperplane is represented in (3):

$$w^T x + b = 0 \quad (3)$$

where w is a vector that is orthogonal to the hyperplane, and x is feature vector of a sample defined in n-dimensional space. In (3), b is a constant that minimizes the error on training set.

e: DECISION TREE (DT)

DT is a tree-based classifier that is composed of nodes, branches and leaves [25]. Nodes represent the variables, branches are the rules applied on variables, and leaves represent the classification outcome. The algorithm is guided by optimization of either Gini impurity or entropy, for each split of the growing phase of tree. Gini impurity measures the amount of randomness in samples. It is calculated as

$$Gini\ impurity = 1 - \sum p_i^2 \quad (4)$$

Entropy is the information required to describe an entity and is defined as:

$$Entropy = -\sum p_i \times log(p_i) \quad (5)$$

where i denotes the number of classes, and p_i represents the probability distribution of i^{th} class. An unseen sample $X = (x_1, x_2, \ldots, x_n)$, is assigned a class that is indicated at the leaf node of the decision tree following the decision rules satisfied by the features of sample.

f: LOGISTIC REGRESSION (LR)

LR uses a statistical approach for classification based on regression equation derived from learning the training data. For an unseen instance $X = (x_1, x_2, \ldots, x_n)$, the algorithm computes the probability of churn, denoted by P, using the formula:

$$P = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_n x_n \quad (6)$$

where $\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_n$ are the coefficients learnt by classifier during training phase.

2) META LEARNER

RF is used as a meta learner in SS-IL framework. The meta learner takes predictions of base learners as input, and utilizes it for classification task.

3) SS-IL FRAMEWORK

The data set is split into a training set and test set following 90% - 10% distribution. Z-score transformation rule

is applied to the numerical features and merged with the one-hot encoded geographic location-related demographic variables to form the complete training dataset. Thereafter, numerical variables in the test set are transformed using transformation parameters from the training set. The training set is used as an input to 3 systems, namely, an under-sampling cluster, an over-sampling cluster, and lastly, a sampling-free cluster. Each cluster contains 6 base learners.

The under-sampling cluster uses Tomek link algorithm with sampling strategy set as ''not minority'' for removing the majority class instances. The under-sampled training set is thereafter used as an input to the 6 base learners in the first cluster. Similarly, the over-sampling cluster, over-samples the training set using the SMOTE algorithm with sampling strategy set as ''not majority'' with k = 5. The new over-sampled training set is further used as input to 6 base learners in the second cluster. The final cluster uses the original training set with imbalanced class distribution. The transformation of original training set into 2 modified training sets, and 1 original training set, is in stark contrast with the normal practice of employing a single training set as input to all base learners in stack frameworks.

The test set is classified by 6 trained base learners in each of the 3 clusters. Hence, there are 18 classification decisions received for test set. The out-of-sample predictions from level 0 learners are saved for each split of the 10-fold cross-validation process. After completion of 1 cycle of cross-validation for the full data set, a consolidated meta data set comprising of 18 features obtained from out of sample level-0 predictions, is created. This set is denoted by $P = \{p_{1(1)}, p_{2(2)}, \ldots, p_{m(18)}\}$, where $p_{i(j)}$ denotes the classification decision for i^{th} instance and j^{th} base learner; m denotes the total number of instances in the dataset. The set P is further used to train the meta classifier. Performance is evaluated via a second 10-fold cross validation process. Fig. 2 is a graphical representation of this framework.

IV. EVALUATION MEASURE

The study uses 5 parameters to evaluate the performance of the proposed framework.

A. PRECISION

Precision is defined as the proportion of predicted churners that are classified correctly. The rationale behind selecting this metric is related to the use case of this study. The hotel technology provider company has limited resources to focus on nurturing possible churners. Hence, optimization of precision is critical. It is calculated using (7), where True Positive (TP) is the total number of churners that are identified correctly by the classifier. False Positive (FP) is the number of non-churners that are incorrectly classified as churners.

$$Precision = \frac{TP}{(TP + FP)} \tag{7}$$

B. F1-SCORE

F1-score is the geometric mean of precision and recall. Recall is defined as the proportion of true churners that are classified correctly. In the case of an imbalanced dataset, F1-score helps to rule out presence of model bias. F1-score and recall can be derived using (8).

$$Recall = \frac{TP}{(TP + FN)};$$
$$F1 - score = 2 \times \frac{Precision \times Recall}{(Precision + Recall)} \tag{8}$$

C. AREA UNDER CURVE (AUC)

AUC represents the area under the Receiver Operating Characteristic (ROC) curve. The ROC curve graphically illustrates the prediction capability of a classifier at various decision thresholds. It is a plot of True Positive Rate (TPR) vs. False Positive Rate (FPR). AUC is the definite integral of this curve and summarizes ROC as a single numerical measure as shown in (9). It is a metric that evaluates a classifier independent of any decision threshold. AUC can take values between 0 and 1; a value close to 1 indicates a good classifier. Hence, the metric is selected for an unbiased and threshold-independent evaluation of the framework.

$$AUC = \int_0^1 TPRd(FPR) \tag{9}$$

D. TOP DECILE LIFT

Top decile lift (TDL) is a meaningful evaluation parameter, particularly in the case of churn. TDL represents number of times a classifier is better at predicting churn, compared to a random guess. The measure is particularly useful in cases where focus is on the top 10% of the predicted churners. Considering ''limited resources'' constraint for our existing use-case, TDL is appropriate as it enables an organization to effectively utilize scarce resources that are deployed to nurture the top 10% of the predicted churners [15]. Mathematically, it is derived using the formula below:

$$TDL = \frac{Perc\_P_{dec}}{Perc\_P} \tag{10}$$

where $Perc\_P_{dec}$ represents the percentage of true churners in the top decile of predicted churners, and Perc_P denotes the percentage of true churners in the dataset.

E. INFORMATION GAIN IN ENSEMBLE

Information Gain has its roots in information theory and is closely related to the notion of entropy. Entropy is defined as the extent of randomness in a dataset. Let D denote a dataset with n samples, $D = \{(x_1, c_1), (x_2, c_2), \ldots, (x_n, c_n)\}$. Here x_i represents the feature vector of the i^{th} sample, and c_i is the corresponding class, $c_i \in \{0, 1\}$, for binary classification. For a randomly selected instance, the probability that it belongs to a class c_i is given by $p(c_i) = \|c_i\|/n$, where $||c_i||$ is the number of samples in D with class c_i [26]. The entropy of

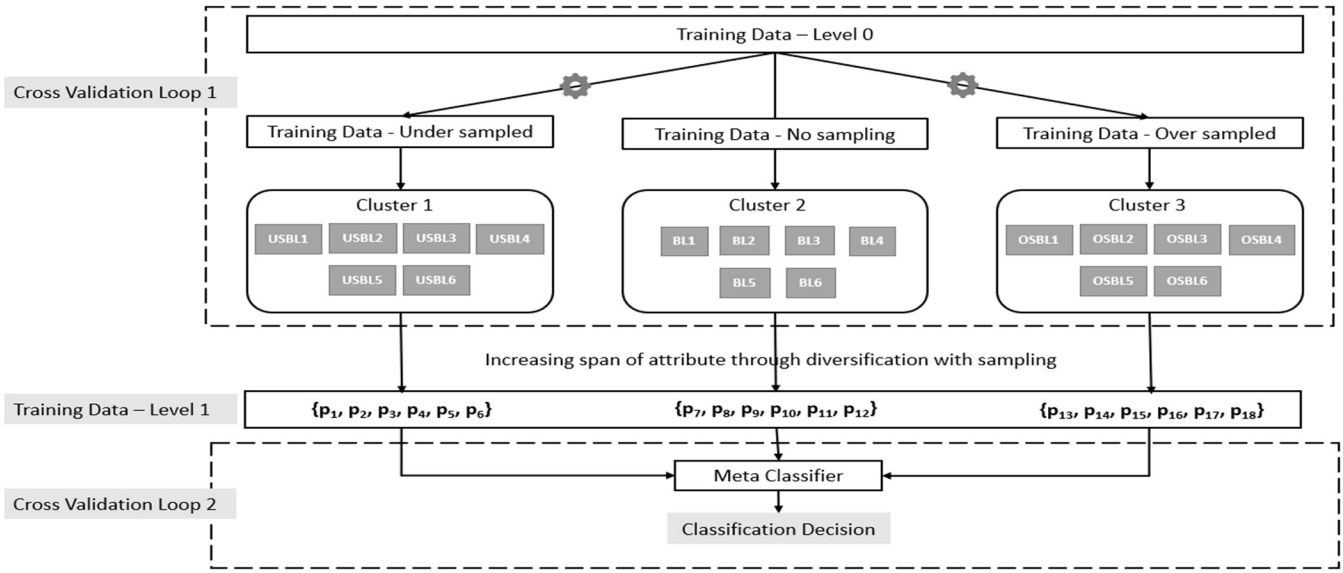

FIGURE 2. SS-IL framework.

dataset D with respect to class variable c_i is given by (11):

$$E(c) = -\sum_{i=1}^{k} p(c_i) * log_2(p(c_i)) \tag{11}$$

where k = 2 for binary classification.

The entropy of dataset D with respect to class variable c_i, given a discrete variable f is as follows:

$$E(c|f) = -\sum_{i=1}^{k} \{p(c_i|f) \times log_2(p(c_i|f)) \times \frac{||f||}{n}\} \tag{12}$$

where ||f|| denotes the frequency of values in feature f.

Information gain (IG) of variable f can then be numerically derived by the reduction in the entropy:

$$IG = E(c) - E(c|f) \tag{13}$$

This study utilizes the IG criterion of meta features for evaluating commonly used stack ensemble approaches. Since there are multiple variables in the training set of a meta learner in stack, the overall information gain for all features of the training set is derived as follows. The probabilistic output of each level 0 base learner is transformed into a binary output using a threshold of 0.5. Thereafter, a new feature β is derived by concatenating the binary output of the level 0 classifiers for each sample. Information Gain for the training dataset is then calculated with respect to this new feature β.

Algorithm: Model evaluation for standalone base learners BL_{ns}

Input:
Data set D = {(x_1, y_1), (x_2, y_2),…, (x_n, y_n)}
Cross validation splits: k
Classification algorithms A = {A_1, A_2,…, A_t}
Evaluation metrics: precision, f1_score, AUC, TDL

Process:
Step 1: Split D into k subsets D_1, D_2,…,D_k
Step 2: Evaluate each algorithm in BL_{ns} following k-fold cross validation
for i = 1…k:
  Select D_i as test set
  Training set T = (D – D_i)
  for j = 1…t:
    $BL_{ns(j)} = A_j(T)$ %Train classifier with training data T
    $h_{ij} = BL_{ns(j)}(D_i^{test})$
  end for
  Calculate Precision(i), f1_score(i), AUC(i) and TDL(i)
end for

Output:

$$Precision = \frac{1}{k}\sum_{i=1}^{k} precision(i); f1\_score = \frac{1}{k}\sum_{i=1}^{k} f1\_score(i);$$

$$AUC = \frac{1}{k}\sum_{i=1}^{k} AUC(i); TDL = \frac{1}{k}\sum_{i=1}^{k} TDL(i)$$

FIGURE 3. Model evaluation for standalone classifiers.

V. EXPERIMENTAL SETUP

In order to assess the lift in the performance achieved by SS-IL, a set of 4 independent experiments are conducted. The algorithms and framework devised in this study are coded in Python using Spyder, which is a free integrated development environment available in Anaconda. A Windows 11 machine with Intel Core i7-10510U, 2.3 GHz processor and 16GB RAM is used for the experiments. Cross validation is a robust methodology to evaluate a classifier's effectiveness. The proposed framework SS-IL utilizes 10-fold cross validation to evaluate classifier.

A. EXPERIMENT 1

The first experiment records the performance of a set of individual standalone base learners that are trained on original imbalanced data. The trained base learners are denoted

Algorithm: Model evaluation for standalone sampled base learners BL_{os} and BL_{us}

Input:
Data set D = {(x_1, y_1), (x_2, y_2),…, (x_n, y_n)}
Cross validation splits: k
Classification algorithms A = {A_1, A_2,…, A_t}
Sampling strategy = {O, U}
Evaluation metrics: Precision, f1_score, AUC, TDL

Process:
Step 1: Split D into k subsets D_1, D_2,…,D_k
Step 2: Evaluate each algorithm following k-fold cross validation
for i = 1…k:
  Select D_i as test set
  Training set T = (D – D_i)
  for j = 1…t:
    $BL_{os(j)} = A_j(O(T))$ %Train classifier with oversampled training set T
    $h_{ij}^{o} = BL_{os(j)}(D_i)$
    $BL_{us(j)} = A_j(U(T))$ %Train classifier with under sampled training set T
    $h_{ij}^{u} = BL_{us(j)}(D_i)$
  end for
  Calculate precision_o(i), f1_score_o(i), AUC_o(i) and TDL_o(i) for sampling strategy O
  Calculate precision_u(i), f1_score_u(i), AUC_u(i) and TDL_u(i) for sampling strategy U
end for

Output:
$$Precision_o = \frac{1}{k}\sum_{i=1}^{k} precision_{o(i)}; f1\_score\_o = \frac{1}{k}\sum_{i=1}^{k} f1\_score\_o(i); AUC\_o = \frac{1}{k}\sum_{i=1}^{k} AUC\_o(i); TDL\_o = \frac{1}{k}\sum_{i=1}^{k} TDL\_o(i)$$
$$Precision\_u = \frac{1}{k}\sum_{i=1}^{k} precision_{u(i)}; f1\_score\_u = \frac{1}{k}\sum_{i=1}^{k} f1\_score\_u(i); AUC\_u = \frac{1}{k}\sum_{i=1}^{k} AUC\_u(i); TDL\_u = \frac{1}{k}\sum_{i=1}^{k} TDL\_u(i)$$

FIGURE 4. Model evaluation for classifiers trained on sampled data.

by BL_{ns}. No sampling is applied to train these classifiers. Performance is recorded following 10-fold cross validation process. The pseudo- code of this experiment is presented in Fig. 3.

The experiment uses data set D with n samples. A 10-fold cross validation process is followed for each train-test split. Classifiers BL_{ns} are trained using algorithm A and training set T. Test set predictions are generated on test set D_i. Precision, F1-score, AUC and TDL are recorded for each train-test split, and for each BL_{ns}. The results of this experiment are discussed in the next section.

B. EXPERIMENT 2

The second set of experiments aim to analyze the impact of 2 sampling strategies on performance of the base learners. The pseudo- code of this experiment is presented in Fig. 4. The experiment uses dataset D with n samples. A 10-fold cross validation process is followed to split the dataset into 10 subsets. For each split, the training data T is first over-sampled to train base learners BL_{os} using algorithm A. Similarly, training data T is under-sampled to train base learners BL_{us}. The predictions on test set Di are recorded for BL_{os} and BL_{us}. Precision, F1-score, AUC and TDL are recorded for each train-test split.

Experiments 1 and 2 are aimed to understand the impact of over-sampling and under-sampling on performance of base learners. The results are plotted in Fig. 5.

As can be seen in Fig. 5, application of over-sampling or under-sampling leads to a reduction in precision for almost all base learners. This implies that BL_{ns} classifiers capture maximum true churners from those that are predicted as churners. However, at the same time, these models record a low F1-score. This is indicative of a low recall value which implies that these classifiers predict majority of the instances as non-churners. Hence, even though precision is high for BL_{ns}, the total number of true churners that can be identified using these models, is debatable. AUC, on the other hand, either improves slightly or remains same for most models, when trained with sampled data. It is observed that TDL improved in 50% of the base learners when trained with under-sampled data. Over-sampling is not effective in this case. However, overall, BL_{ns}(RF) records the highest TDL of 4.5. This means that out of top 10% of customers that are predicted as churners by this model, 63% are correctly classified. Additionally, RF emerges as the best classifier in the second experiment.

C. EXPERIMENT 3

The third set of experiments aim to understand the effect of implementing a stacked framework on model performance. Here, BL_{os} and BL_{us} models are arranged in 2 independent and separate stacked ensembles, denoted by "Stacked OSBL" and "Stacked USBL", respectively. RF model is used as a meta learner in both the stack ensembles. This experiment follows the steps of a general stack framework as depicted in Fig. 6.

A stacking process begins with a dataset D consisting of n samples. It is used to train a given set of base learners BL. Let z_{ij} denote the classification decision of j^{th} BL on a test set for i^{th} sample. A new dataset D_{meta} is created using test set predictions for each split of a k fold cross validation. Finally, meta learner ML is trained using D_{meta}. Classification

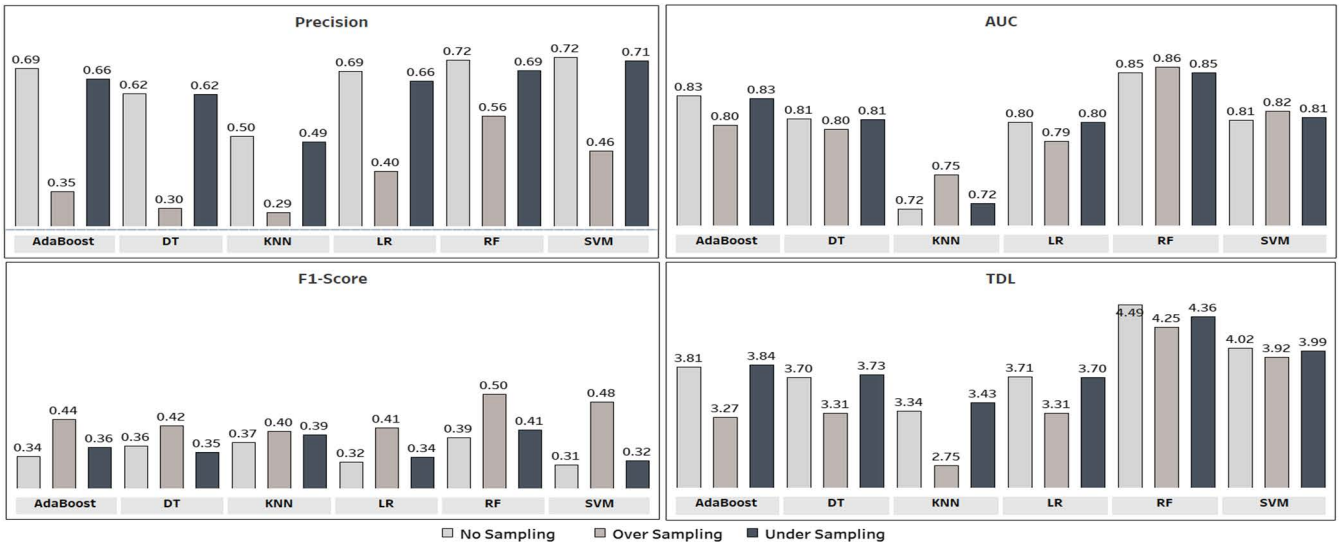

FIGURE 5. Summary of performance of BL_{ns}, BL_{os} and BL_{us}.

decision of ML is denoted by p_{meta} which is further used to measure precision, F1-score, AUC and TDL.

In experiment 3, there are 2 stack frameworks, called "Stacked OSBL" and "Stacked USBL", that are implemented following the guidelines described above. Performance of meta learner ML is recorded for evaluation. A 10-fold cross validation process is used to calculate average performance for both the experimental settings. Additionally, the training sets used for meta learners in Stacked OSBL and Stacked USBL, is analyzed to derive the corresponding information gain achieved using meta-attributes. The results of this experiment are discussed in the next section.

D. EXPERIMENT 4

The final experiment represents our proposed framework SS-IL. The algorithm consists of base learners BL_{ns}, BL_{os} and BL_{us}, combined together in a single stacked ensemble. RF model is used as a meta learner. The algorithm is described in Fig. 7.

SS-IL uses dataset D with n samples. The dataset D is split into k subsets. The training sets O(T), U(T) and T are used to train base learners BL_{os}, BL_{us} and BL_{ns} respectively. O(T), U(T) and T denote over-sampled, under-sampled, and non-sampled training set. Predictions for test set Di for each train-test split are cumulatively consolidated to produce a new dataset D_{meta} for training the meta classifier ML. D_{meta} is characterized by an increased span of attributes that contribute to the information gain in base learners through sampling. The meta learner ML is finally trained and evaluated using k-fold cross validation. In order to compute the information gain achieved by the increased span of attributes in SS-IL, the training set of ML is analyzed. Information Gain calculation for the training set of this meta learner follows the same process as described in Experiment 3.

Algorithm: General Stacking Process

Input:
Data set $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$
Level 0 learners: $BL_1, BL_2, \ldots, BL_t$
Meta learner: ML

Process:
Step 1: Train the base learners:
for i = 1,…,t:
  $p_i = BL_i(D)$
end for
Step 2: Build new data set of predictions from base learners:
$D_{meta} = \phi$ %Initialize new data set to null set
for i = 1,…,n:
  for j = 1,…,t:
    $z_{ij} = p_j(x_i)$
  end for
  $D_{meta} = D_{meta} \cup \{(z_{i1}, \ldots, z_{it}), y_i)\}$
end for
Step 3: Train meta learner with new dataset D_{meta}

Output:
$p_{meta} = ML(D_{meta})$

FIGURE 6. General process of stacking.

VI. RESULTS AND DISCUSSION

To summarize, the study aims to investigate improvement in performance through sampling in a stacking framework. Hence, a series of 4 experiments is conducted in a phased manner. Experiment 1 investigates the performance when no sampling is used in training phase. Experiment 2 analyzes the impact of introducing 2 sampling strategies in training data – SMOTE and Tomek link. Experiment 3 integrates sampling strategies with stacking to study performance variation – Stacked OSBL and Stacked USBL. Experiment 4 finally incorporates sampling strategies to increase span of

Algorithm: SS-IL

Input:
Data set D = $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$
Cross validation splits: k
Classification algorithms A = $\{A_1, A_2, \ldots, A_t\}$
Meta learner: ML
Sampling strategy = {O, U}
Evaluation metrics: precision, f1_score, AUC, TDL

Process:
Step 1: Split D into k subsets $D_1, D_2, \ldots, D_k$
Step 2: Train the base learners and build new data set of predictions from base learners :
$D_{meta} = \phi$ %Initialize new data set to null set
for i = 1…k:
  Select D_i as test set
  Training set T = $(D - D_i)$
  for j = 1…t:
    $BL_{os(j)} = A_j(O(T))$ %Train classifier with oversampled training set T
    $h_{ij}^{o} = BL_{os(j)}(D_i)$
    $BL_{us(j)} = A_j(U(T))$ %Train classifier with under sampled training set T
    $h_{ij}^{u} = BL_{us(j)}(D_i)$
    $BL_{ns(j)} = A_j(T)$ %Train classifier with original training set T
    $h_{ij}^{n} = BL_{ns(j)}(D_i)$
    $h_{ij} = (h_{ij}^{o} \cup h_{ij}^{u} \cup h_{ij}^{n})$ %Increasing span of attribute through sampling diversification
  end for
  $D_{meta} = D_{meta} \cup \{(h_{ij}, y_i)\}$
end for
Step 3: Train meta learner with new dataset D_{meta}
$p_{meta} = ML(D_{meta})$
Calculate precision, f1-score, AUC and TDL using k-fold cross validation

Output:
$$Precision = \frac{1}{k}\sum_{i=1}^{k} precision(i); f1\_score = \frac{1}{k}\sum_{i=1}^{k} f1\_score(i); AUC = \frac{1}{k}\sum_{i=1}^{k} AUC(i); TDL = \frac{1}{k}\sum_{i=1}^{k} TDL(i)$$

FIGURE 7. SS-IL framework.

attributes for meta classifier in a single stack. This framework is called SS-IL. The experimental results are consolidated and graphically presented in Fig. 8 and the following conclusions are drawn.

Results show that the sampling strategies in isolation, or in combination with stack framework, have no significant impact on AUC. This is in agreement with claims of [27] that AUC is not sensitive to class distribution. In the experiments, SS-IL achieves the highest AUC of 86.4% that is indicative of an improved TPR, independent of class distribution.

Another observation is related to precision. RF as a classifier achieves the highest precision of 0.72 with no sampling applied to training data. This is a case of a biased classifier that predicts majority of instances as non-churners, leading to high precision at the cost of low recall. The claim is further supported by a low F1-score of 0.39 for the same classifier with no sampling applied for training. While application of sampling techniques leads to a reduction in precision of RF for the use case under study, F1-score has the opposite effect. This indicates reduction in bias of a classifier towards majority class. Sampling when combined with stacking, has a positive impact on both precision and F1-score. Proposed framework, SS-IL, further records an improvement in precision and F1-score at 0.69 and 0.49 respectively, and emerges as second-best classifier. This can be attributed to SS-IL's ability to reduce model bias by expanding and diversifying the size of the training set.

Results for TDL show a performance of 4.49 for RF when no sampling is applied during training phase. Application of sampling techniques independently, or combined with stacking, do not show positive impact on performance. However, SS-IL achieves the best TDL of 4.7 in the experiment. This essentially means that out of the top 10% of customers that are predicted as churners by SS-IL, 65% are identified correctly. As discussed before, for our use case, TDL is an

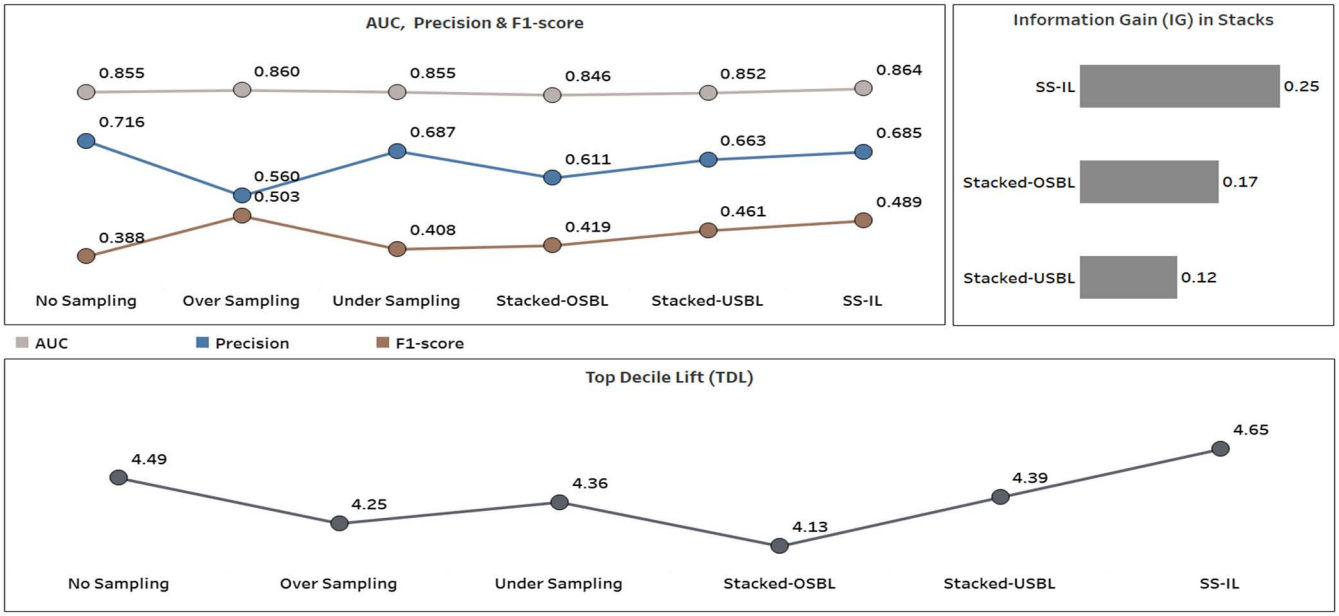

FIGURE 8. Comparison of performance with SS-IL.

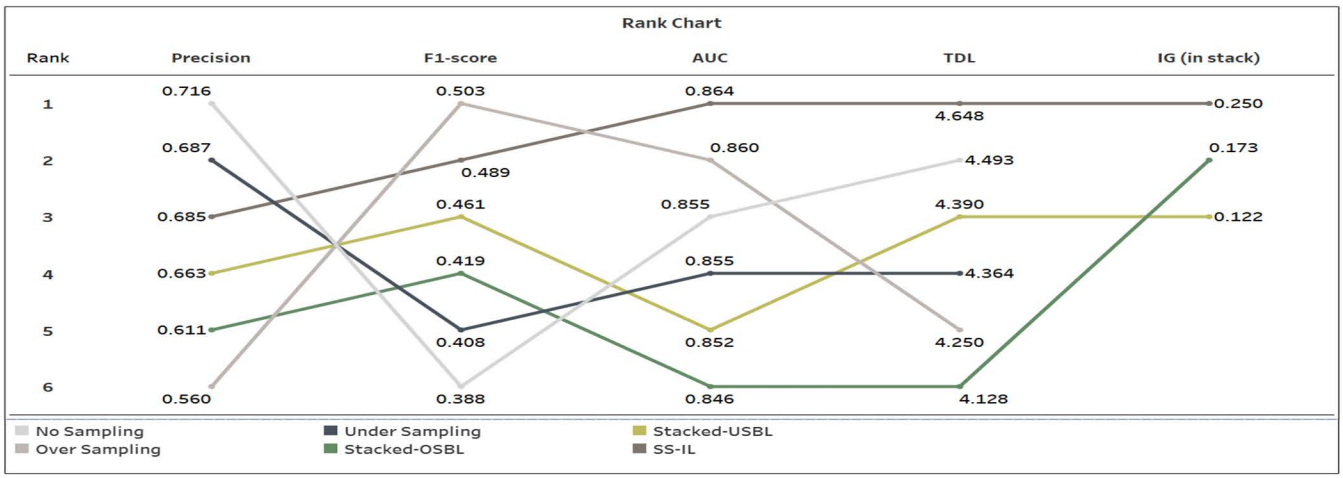

FIGURE 9. Rank chart of applied techniques.

Time (in sec)	AdaBoost	DT	KNN	LR	RF	SVM	Stacked	SS-IL
Under sample	1.00	0.18	0.01	0.43	2.00	94.00	114.00	692.00
Over sample	3.00	0.37	0.01	1.00	6.00	371.00	459.00	

FIGURE 10. Computational time for training models.

important measure of performance evaluation due to ''limited resources'' criterion that is previously described in section 4.

Information Gain for meta features in SS-IL is the highest, compared to Stacked OSBL and Stacked USBL approaches. The expanded set of meta variables employed in the training of meta learner leads to maximum reduction in the randomness of the dataset with respect to the target variable. The observation further substantiates the

relevance of the attributes used for training the meta learner in SS-IL.

The competence of SS-IL is supported by a rank chart in Fig. 9. Proposed framework emerges as the best classifier in 3 out of the 5 evaluation parameters, proving its effectiveness in imbalance learning for the use case under study. As the framework attempts to utilize the prediction capabilities of over-sampled, under-sampled and standalone models, it cancels out overfitting through the increased span of attributes in the ensemble aggregation process and emerges as a balanced classifier with maximum IG.

A. COMPUTATIONAL TIME

The computational training time taken for each experiment is recorded and presented in Fig. 10. As can be seen, SS-IL is not computationally efficient recording the longest time for training base learners. There are 2 main reasons behind this. Firstly, the framework, in its current state, is sequential in nature. Due to limited computational resources for the conducted experiments, the base learners in the 3 clusters are trained sequentially. Secondly, SS-IL includes a computationally expensive base learner, SVM. Hence, although time taken by SS-IL is of the order of a few minutes, the chosen base learners in the framework play a significant role in the computational time spent in training phase. Parallel and distributed processing of 3 clusters in SS-IL can make it computationally efficient.

VII. CONCLUSION

Imbalanced learning is a key challenge in churn prediction. Datasets related to churn are often skewed, and over-sampling and under-sampling methods are applied to balance the training data for effective imbalanced learning. Ensemble methods have shown promising results in many studies proving their effectiveness in improving classifier performance. The new framework SS-IL aims to stimulate diversity with sampling-based base models, and cancel out overfitting by increasing span of attributes used to train a meta classifier in a stack ensemble. A four-part series of experiments is conducted on a churn dataset obtained from a leading hotel commerce platform company. A set of 5 evaluation measures are used for assessing performance, namely, precision, F1-score, AUC and TDL. Information gain is computed for the training sets used to train meta learner. Results show that the framework positively contributes to the information gain of the training set used for the meta learner of the stack. SS-IL, although computationally expensive, is effective in predicting churn achieving highest AUC and TDL of 86.4% and 4.7, respectively.

The study has few limitations. Firstly, the framework utilizes Tomek link and SMOTE as sampling strategies. Investigating the impact of other sampling methods on the performance of SS-IL is an interesting direction for future studies. Secondly, the framework has six models that are trained sequentially in each of the 3 clusters. This leads to relatively high computational time spent in training. Incorporating efficient base learners in the framework, and using parallel and distributed processing in the training phase, are a few ways to address the limitation in future. Thirdly, the study validates the framework for a specific use case pertaining to a hotel commerce platform company. Future investigations that use imbalanced datasets from other domains will help to establish the effectiveness of the framework.

REFERENCES

[1] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002, doi: 10.1613/jair.953.

[2] A. Bernardo and E. D. Valle, "An extensive study of C-SMOTE, a continuous synthetic minority oversampling technique for evolving data streams," *Expert Syst. Appl.*, vol. 196, Jun. 2022, Art. no. 116630, doi: 10.1016/j.eswa.2022.116630.

[3] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IEEE World Congr. Comput. Intelligence)*, Jun. 2008, pp. 1322–1328, doi: 10.1109/IJCNN.2008.4633969.

[4] G. Kovács, "An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets," *Appl. Soft Comput.*, vol. 83, Oct. 2019, Art. no. 105662, doi: 10.1016/j.asoc.2019.105662.

[5] Y. M. Chyi, "Classification analysis techniques for skewed class distribution problems," M.S. thesis, Dept. Inf. Manage., Nat. Sun Yat-Sen Univ., Taiwan, 2003.

[6] F. Z. Belgrana, N. Benamrane, M. A. Hamaida, A. M. Chaabani, and A. Taleb-Ahmed, "Network intrusion detection system using neural network and condensed nearest neighbors with selection of NSL-KDD influencing features," in *Proc. IEEE Int. Conf. Internet Things Intell. Syst. (IoTaIS)*, Jan. 2021, pp. 23–29, doi: 10.1109/IoTaIS50849.2021.9359689.

[7] P. Hart, "The condensed nearest neighbor rule (corresp.)," *IEEE Trans. Inf. Theory*, vol. IT-14, no. 3, pp. 515–516, May 1968, doi: 10.1109/TIT.1968.1054155.

[8] E. F. Swana, W. Doorsamy, and P. Bokoro, "Tomek link and SMOTE approaches for machine fault classification with an imbalanced dataset," *Sensors*, vol. 22, no. 9, p. 3246, Apr. 2022, doi: 10.3390/s22093246.

[9] I. Tomek, "Two modifications of CNN," *IEEE Trans. Syst., Man Cybern.*, vol. SMC-6, no. 11, pp. 769–772, Nov. 1976. [Online]. Available: http://ci.nii.ac.jp/naid/80013575533/en/

[10] M. Kubat, "Addressing the curse of imbalanced training sets: One-sided selection," in *Proc. 14th Int. Conf. Mach. Learn.*, 2000, p. 179.

[11] J. Laurikkala, "Improving identification of difficult small classes by balancing class distribution," in *Artificial Intelligence in Medicine* (Lecture Notes in Computer Science). Berlin, Germany: Springer, 2001, doi: 10.1007/3-540-48229-6_9.

[12] I. V. Pustokhina, D. A. Pustokhin, P. T. Nguyen, M. Elhoseny, and K. Shankar, "Multi-objective rain optimization algorithm with WELM model for customer churn prediction in telecommunication sector," *Complex Intell. Syst.*, pp. 1–13, Apr. 2021, doi: 10.1007/s40747-021-00353-6.

[13] M. Al-Mashraie, S. H. Chung, and H. W. Jeon, "Customer switching behavior analysis in the telecommunication industry via push-pull-mooring framework: A machine learning approach," *Comput. Ind. Eng.*, vol. 144, Jun. 2020, Art. no. 106476, doi: 10.1016/j.cie.2020.106476.

[14] P. Cichosz. (2015). *Data Mining Algorithms*. [Online]. Available: https://www.wiley.com

[15] K. W. De Bock and A. De Caigny, "Spline-rule ensemble classifiers with structured sparsity regularization for interpretable customer churn modeling," *Decis. Support Syst.*, vol. 150, Nov. 2021, Art. no. 113523, doi: 10.1016/j.dss.2021.113523.

[16] M. Ahmed, H. Afzal, I. Siddiqi, M. F. Amjad, and K. Khurshid, "Exploring nested ensemble learners using overproduction and choose approach for churn prediction in telecom industry," *Neural Comput. Appl.*, vol. 32, no. 8, pp. 3237–3251, Apr. 2020, doi: 10.1007/s00521-018-3678-8.

[17] I. Syarif, E. Zaluska, A. Prugel-Bennett, and G. Wills, "Application of bagging, boosting and stacking to intrusion detection, in *Machine Learning and Data Mining in Pattern Recognition*. Berlin, Germany: Springer, 2012, pp. 593–602, doi: 10.1007/978-3-642-31537-4_46.

[18] M. J. Shabankareh, M. A. Shabankareh, A. Nazarian, A. Ranjbaran, and N. Seyyedamiri, "A stacking-based data mining solution to customer churn prediction," *J. Relationship Marketing*, vol. 21, no. 2, pp. 124–147, Apr. 2022, doi: 10.1080/15332667.2021.1889743.
[19] Z. H. Zhou, *Ensemble Methods: Foundations and Algorithms*. Boca Raton, FL, USA: CRC Press, 2012, doi: 10.1201/b12207.
[20] M. Samieinasab, S. A. Torabzadeh, A. Behnam, A. Aghsami, and F. Jolai, "Meta-health stack: A new approach for breast cancer prediction," *Healthcare Anal.*, vol. 2, Nov. 2022, Art. no. 100010, doi: 10.1016/j.health.2021.100010.
[21] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.
[22] M. A. Ganaie and M. Tanveer, "KNN weighted reduced universum twin SVM for class imbalance learning," *Knowl.-Based Syst.*, vol. 245, Jun. 2022, Art. no. 108578, doi: 10.1016/j.knosys.2022.108578.
[23] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. 13th Int. Conf. Mach. Learn.*, 1996, pp. 148–156, doi: 10.1.1.133.1040.
[24] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995, doi: 10.1023/A:1022627411411.
[25] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986, doi: 10.1023/A:1022643204877.
[26] L. E. Raileanu and K. Stoffel, "Theoretical comparison between the Gini index and information gain criteria," *Ann. Math. Artif. Intell.*, vol. 41, no. 1, pp. 77–93, May 2004, doi: 10.1023/B:AMAI.0000018580.96245.c6.
[27] G. M. Weiss and F. Provost, "Learning when training data are costly: The effect of class distribution on tree induction," *J. Artif. Intell. Res.*, vol. 19, pp. 315–354, Oct. 2003, doi: 10.1613/jair.1199.

SOUMI DE (Member, IEEE) received the B.Sc. degree (Hons.) in physics from Calcutta University, West Bengal, India, in 2000, and the M.C.A degree in computer applications from Visva Bharati University, West Bengal, in 2003. She is currently pursuing the Ph.D. degree in data science with CHRIST (Deemed to be University), Karnataka, India. Since 2004, she has been with several multi-national corporations in the domain of business intelligence and analytics. Her research interests include decision science, natural language processing, and predictive analytics. She received the Top Performer Award, in 2015 for her contribution to automating analytical solutions in a multi-national company.

P. PRABU received the Ph.D. degree in computer applications from Anna University, India. He is currently working as an Assistant Professor with the Department of Computer Science, CHRIST (Deemed to be University), Karnataka, India. He has more than 14 years' experience in teaching and industry. His research interests include software engineering, web service, deep learning, the IoT, and mobile application.