Regularization oversampling for classification tasks : to exploit what you do not know

Regularization Oversampling for Classification Tasks: To Exploit What You Do Not Know

Van der Schraelen Lennert[a,b], Stouthuysen Kristof[a,b], Vanden Broucke Seppe[c,d], Verdonck Tim[e,f,1]

[a]Area Accounting and Finance, Vlerick Business School, Vlamingenstraat 83, 3000 Leuven, Belgium
[b]Department of Accounting, Finance and Insurance, Faculty of Economics and Business, KU Leuven, Naamsestraat 69, 3000 Leuven, Belgium
[c]Department of Business Informatics and Operations Management, Faculty of Economics and Business Administration, UGhent, Tweekerkenstraat 2, 9000 Ghent, Belgium
[d]Research Center for Information Systems Engineering (LIRIS), Faculty of Economics and Business, KU Leuven, Naamsestraat 69, 3000 Leuven, Belgium
[e]Department of Mathematics, Faculty of Science, UAntwerpen, Middelheimlaan 1, 2020 Antwerp, Belgium
[f]Department of Mathematics, Faculty of Science, KU Leuven, Celestijnenlaan 200B, 3001 Leuven, Belgium

Abstract

In numerous binary classification tasks, the two groups of instances are not equally represented, which often implies that the training data lack sufficient information to model the minority class correctly. Furthermore, many traditional classification models make arbitrarily overconfident predictions outside the range of the training data. These issues severely impact the deployment and usefulness of these models in real life. In this paper, we propose the boundary regularizing out-of-distribution (BROOD) sampler, which adds artificial data points on the edge of the training data. By exploiting these artificial samples, we are able to regularize the decision surface of discriminative machine learning models and make more prudent predictions. Next, it is crucial to correctly classify many positive instances in a limited pool of instances that can be investigated with the available resources. By smartly assigning predetermined nonuniform class probabilities outside the training data, we can emphasize certain data regions and improve classifier performance on various material classification metrics. The good performance of the proposed methodology is illustrated in a case study that consists of both benchmark balanced and imbalanced classification data sets.

Keywords: Binary classification, Regularization, Sampling, Data imbalance

1. Introduction

The classification of instances into two groups is one of the core concepts of data mining. Often, a major challenge arises because the two classes are not equally represented. For instance, the imbalance is often present in fraud [1], customer churn [49], credit scoring [32] and bankruptcy [42] data sets. Many traditional machine learning models try to minimize the overall error, which causes the machine learning model to focus on the majority (negative) class which usually results in a poor capability to find minority (positive) class samples. Hence, machine learning models could achieve excellent accuracy without detecting any positive cases by training on imbalanced data sets. Unfortunately, the detection of minority samples is often of utmost importance. For example, in fraud applications, severe financial and credibility losses are incurred if fraudulent instances cannot be detected. The authors of [14] and [49] sorted the methods that can handle these imbalanced data sets into several categories. First, new algorithms can be designed, or existing algorithms can be modified to handle the class imbalance issue. For instance, algorithms can be modified to accept costs to emphasize instance importance or in an attempt to mimic reality.

Furthermore, the class distribution of the provided training data can be rebalanced as a preprocessing step by decreasing the number of majority samples or increasing the number of minority samples. Moreover, ensemble solutions that exploit the previously mentioned algorithmic or data-level solutions can also handle the imbalance issue.

In this paper, we develop an oversampling algorithm. However, we do not create artificial samples that belong to the minority class. Instead, our method samples artificial data points on the boundary of the data set that can be exploited during the training phase.

Machine learning models have demonstrated that they can achieve satisfactory performance in predicting test samples similar to the training data. However, discriminative machine learning models tend to be overconfident in data regions they have never seen before [21], [20], and [25]. This behavior in the presence of dissimilar or unseen data implies that developing and deploying machine learning models in real life is very challenging. For example, dissimilar or unseen data can be encountered in imbalanced classification tasks where the training data lack sufficient information to model the minority class correctly.

We propose the boundary regularizing out-of-distribution (BROOD) sampler, which adds artificial data points on the data distribution's boundary to aid the machine learning model in

Email addresses: lennert.vanderschraelen@vlerick.com (Van der Schraelen Lennert), kristof.stouthuysen@vlerick.com (Stouthuysen Kristof), seppe.vandenbroucke@ugent.be (Vanden Broucke Seppe), tim.verdonck@uantwerpen.be (Verdonck Tim)

[1]Corresponding author at: Middelheimlaan 1, 2020 Antwerp, Belgium

learning a suitable decision surface. By exploiting these artificial samples, we are able to regularize the decision surface of discriminative machine learning models and make more reliable predictions. This yields more interpretable machine learning models that exploit the assignment of high or low probabilities in out-of-distribution (OOD) regions.

Using the discounted cumulative gain (DCG) metric, we show that our approach has a substantial positive effect on detecting minority instances when only limited sources are available. Moreover, we show that our approach positively affects other performance metrics, such as the average precision (AP). In particular, we

- propose the BROOD sampler that samples artificial data points that lie on the data distribution's boundary;
- investigate whether artificial samples can be used to regularize the decision surface of various machine learning models;
- elucidate the difference between existing oversampling approaches and show how our methodology can complement minority oversampling methods;
- illustrate the excellent performance of the proposed sampler in an extensive data study by using the XGBoost [8] algorithm.

The data study consists of various balanced and imbalanced classification data sets. Furthermore, we focus on the XGBoost algorithm, which is considered one of the best algorithms to learn from tabular data.

2. Literature Review

2.1. Minority Oversampling

If the data set is imbalanced, traditional machine learning models will only learn the specific data regions where the minority samples are present, which may have a negative impact on the predictive power. Moreover, due to this class imbalance, traditional classification models tend to be focused on the majority class. An extensive summary of methods mitigating the class imbalance problem can be found in [49].

One possible procedure is to assign additional weights to minority class instances (or to replicate the instance multiple times) to enhance their importance. However, this approach can result in severe overfitting. A way to mitigate this issue is to apply an oversampling technique. For instance, the conventional SMOTE [7] generates similar synthetic samples of the minority class from a minority instance by sampling artificial data points on a straight line between the minority instance's minority class nearest neighbors. This aids the classifier in building larger decision regions enclosing nearby actual and artificial minority class points and reduces the classifier's focus toward the majority class.

A well-known extension of SMOTE is ADASYN [18]. ADASYN generates synthetic samples from minority instances proportional to the impurity of the instance's neighborhood. This approach aims to shift the decision boundary by letting the classifier focus more on difficult-to-learn minority instances. A weakness of ADASYN is that this method emphasizes the importance of noisy minority instances and outliers.

Next, ROSE [26] is a popular oversampling technique that generates data from a class-dependent kernel density estimate [6] with a certain smoothing matrix. By considering the extensive kernel density estimation literature, suitable smoothing matrices can be chosen. Furthermore, ROSE can be seen as a smoothed bootstrap resampling technique and thus has applications in bootstrap aggregating (bagging) [4] procedures.

Although synthetic oversampling techniques reduce overfitting by creating similar artificial minority samples, many standard oversampling techniques introduce noise into the data set and enhance class overlap.

Recently, new methods have been proposed to mitigate the introduction of noise and class overlap. For instance, the authors of [10] used K-means clustering to discover clusters with a high ratio of minority instances. Artificial data points are sampled using SMOTE in only these data areas because it avoids noise generation and data overlap. Moreover, within-class imbalance can be improved by assigning more synthetic samples to clusters with a high average distance among the cluster's minority samples. The authors have shown that their method outperforms various minority oversamplers. Likewise, to reduce data overlap while generating artificial samples, the authors of [39] trained a support vector data description (SVDD) model [40] to define overlapping instances that will be removed during the generation of minority samples. Furthermore, the authors address hard-to-learn and within-class imbalance issues by constructing boundary and density factors. These factors enable the creation of more artificial samples from minority instances in low-density regions and regions close to the SVDD class boundary. The authors have shown that their sampler, which is a weighted SMOTE scheme, achieves superior performance compared to various other oversamplers. These samplers, mitigating the introduction of noise, contain an additional layer of complexity as an initial machine learning model must be trained, such as K-means or an SVDD model.

We give an overview of existing OOD samplers in the next subsection. Although the purpose of OOD sampling and minority oversampling is different, properly generated OOD samples should be able to aid the classifier in building suitable decision boundaries and should not introduce noise and data overlap.

2.2. Out-Of-Distribution Sampling

Let us introduce some notation. We denote with $\boldsymbol{x}$ the data attributes and with y the actual response of an instance. The symbol ∂ represents a boundary, and $\boldsymbol{\theta}$ represents the model's hyperparameters.

Most discriminative machine learning models are overconfident in unseen data regions, as no data are available to guide the model. Generative classifiers can be trained to solve this issue, as these models also incorporate data distributions. Generative classifiers model the joint probability $P(X, Y)$ from which we can derive $P(Y|X)$. However, these classifiers often make

stronger modeling assumptions and are overtaken by discriminative models for large sample sizes [30].

Another approach to bypass this problem is by first detecting the OOD data points and only using the trained model to predict the in-distribution (ID) data points. To detect these OOD samples, one can train anomaly detection models such as isolation forests (IF) [22] or one-class classification models such as SVDD on the training data set. However, this method is unsatisfactory because predictions of the OOD data's label cannot be obtained.

In the fields of open set recognition [16] (training classifiers that can classify seen classes and deal with unseen classes) and OOD detection [47] (detecting test samples drawn from a distribution that is different from the training distribution), various oversampling techniques are proposed to tackle issues that arise due to unseen data. In [20], deep neural networks (NNs) are trained more conservatively by exposing the neural classification networks to OOD samples. Denote the ID and OOD data distributions with f_{id} and with f_{ood} respectively. The authors of [20] propose to minimize

$$L(X, \boldsymbol{y}, \boldsymbol{\theta}) = \mathbb{E}_{f_{id}}\left[P_{\boldsymbol{\theta}}(y|\boldsymbol{x})\right] + \lambda \mathbb{E}_{f_{ood}}\left[KL(\mathcal{U}(y)||P_{\boldsymbol{\theta}}(y|\boldsymbol{x}))\right] \quad (1)$$

with $\mathcal{U}(y)$ being the uniform distribution. This causes the classifier to assign the base value (the class prior) to the OOD samples. As it is infeasible to sample the whole OOD space, the OOD samples are generated relatively close to the ID data points. A generative adversarial network (GAN) [17] is constructed to generate OOD samples. The discriminator is trained to distinguish between artificiality-generated samples and ID data points. The generator is trained to fool the discriminator (generate samples close to the ID data points) and to generate low-density artificial samples. The classifier is trained by minimizing Equation 1. The three models are trained in an alternating manner. A limitation is that the same class prior is imposed to all the OOD samples. The authors have shown their method's effectiveness using deep convolutional NNs on image data sets by comparing their method to a baseline detector without artificially generated samples. Furthermore, the authors have shown that the OOD samples can also aid in handling unseen classes by learning an OOD class. Similarly, the authors of [48] generated ID and OOD samples in an unsupervised manner using a GAN. These samples are exploited to distinguish between all seen and potential unseen classes employing support vector machines with a radial basis function kernel. The authors have shown effectiveness on toy, image and document classification data sets by comparing their method to various methods for open-set recognition. Likewise, in [15], OOD samples are generated using a GAN, incorporating the ability to deal with unknown classes using extreme value theory. The authors used NNs and have shown the model's usefulness in image classification tasks by comparing their method to baseline detectors without artificially generated samples.

In [43], a conditional variational auto encoder (CVAE) [19], is used to distinguish between two types of OOD samples surrounding the latent encoding and exploiting the OOD samples for OOD detection. The authors compared their model to other classifier-based OOD detectors on image classification data sets using NNs. Although GANs and (V)AEs are good tools for generating data, they are often challenging to train.

In [28], an approach is proposed to sample data points close to the ID data set without the explicit need to use a GAN or (V)AE. The authors state that the data sampled by this method can be used to safeguard NNs and validate generalization performance. A suitable artificial sample is created by iteratively moving in a random direction from a randomly selected ID data point. If the minimum distance between the new location and the ID data is satisfactory, the iteration stops, and one OOD sample has been generated. The authors also include a 'softness' possibility to stop iterating even if the minimum distance is not achieved. Especially for high-dimensional data, it is difficult to assign a suitable minimum distance and step size. The authors used a VAE to construct a low-dimensional data embedding in their experiments. Furthermore, the approach is relatively slow and very local in nature, as the algorithm samples one artificial instance at a time by creating random directions from a random ID data point. Next, fewer artificial OOD samples will be sampled in sparse data regions as the OOD sampling procedure starts from a random ID data point. The authors have shown that their method is able to improve OOD detection on image data and provide a case study on the synthetic generation of OOD trajectories for automated driving. Nevertheless, to our knowledge, no previous research has been done that compares existing OOD samplers and discusses their limitations.

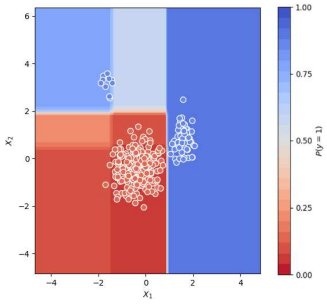
We propose a computationally attractive OOD sampler that mitigates the previously mentioned problems without needing GANs or (V)AEs, enabling us to assign class-specific probabilities outside the training data.

2.3. Research Questions

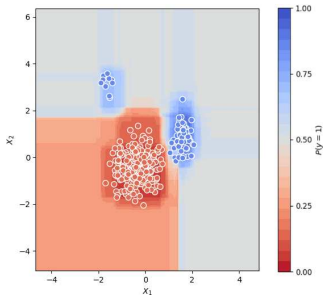
As no data are available to guide the model due to insufficient training data or concept drift, discriminative machine learning models assign arbitrarily overconfident predictions to these data. This is an important problem, as it causes people to lose confidence in the model, as the predictions are not substantiated.

Hence, a machine learning model should be able to assign a predetermined probability to data regions it has never seen before. Consider the case where resources are only available to investigate a limited number of data instances. If low probabilities are assigned to unseen data regions, these regions will not be important, and one will exploit the current knowledge of training data. Conversely, if high probabilities are assigned to unseen data regions, these regions are important, and one will be encouraged to investigate these new data regions.

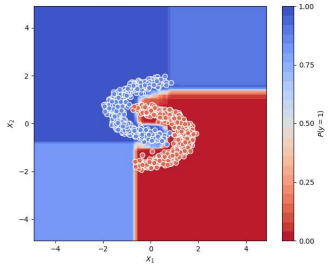
Figure 1 visualizes the decision boundary of XGBoost [8] classifiers on a toy data set created by the author and the well-known moons data set. On the left-hand side, we plot the decision boundary of the XGBoost classifier, which is trained using the binary cross entropy loss. One can deduce that this classifier is arbitrarily confident in certain data regions. For instance, the model assigns an unfounded high probability in the lower right corner for the toy data set.



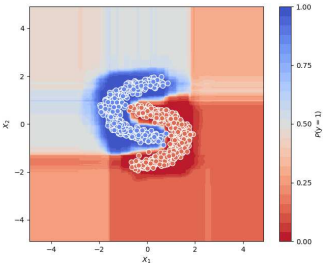
Visual of a toy data set which is generated by the author. The decision surface is trained with an XGBoost classifier using the binary cross entropy loss.



Visual of a toy data set which is generated by the author. The decision surface is trained with an XGBoost classifier using a regularizing loss.



Visual of the moons data set. The decision surface is trained with an XGBoost classifier using the binary cross entropy loss.



Visual of the moons data set. The decision surface is trained with with an XGBoost classifier using a regularizing loss.

Figure 1: Visual comparison of various decision surfaces. On the left-hand side we plot the decision surfaces that are trained with the XGBoost classifier using the standard binary cross-entropy loss. On the right-hand side we plot the decision surfaces that are trained with the XGBoost classifier using a regularizing loss.

We aim to construct decision surfaces such as those on the right-hand side. The classifier should assign, in general, a high probability to the ID minority sample regions and a low probability to the ID majority sample regions. Next, predetermined probabilities should be assigned to the OOD data regions 'close' to the ID minority data points and 'far' from the ID majority data points. Furthermore, one should be able to assign predetermined probabilities to the OOD regions 'far' from the ID minority samples and 'close' to the ID majority samples. For the plots on the right-hand side of Figure 1, OOD data regions closer to the ID minority samples have a higher probability relative to the data regions closer to the ID majority samples.

We aim to create artificial samples on the boundary of the training data distribution and the minority/majority class that will guide our model in unseen data regions. More rigorously, for all responses y, we create samples on $\partial\hat{P}(X|Y = y) \cap \partial\hat{P}(X)$, defining the class-specific boundary between ID and OOD data. Thus, we investigate the following research questions:

RQ1: Are artificial OOD samples able to suitably regularize the machine learning model's probability boundary?

RQ2: Are artificial OOD samples able to enhance the machine learning model's performance? Does our approach complement existing minority oversampling methods?

RQ3: What are the additional advantages of artificial OOD samples for imbalanced classification tasks?

From the previous discussion, if new data points lie in a data region where there is no or only a little information available, we believe that the inclusion of artificial OOD samples positively impacts the predictive power of machine learning models. Furthermore, suppose the training data captures all data regions present in the test data. In that case, we expect that the inclusion of artificial OOD samples will not have a negative impact on the performance measures as these samples will not interfere with the ID data points.

3. Methodology

In this section, we first develop a new oversampling technique that can efficiently generate artificial OOD samples. Next, we elucidate how we can use these artificial samples to regularize machine learning models in unseen data regions. Finally, we discuss how the sampler's hyperparameters affect the samples generated.

3.1. Boundary Regularizing Out-Of-Distribution (BROOD) Sampling

To answer the research questions, we need to construct a suitable OOD sampler. We name our sampler the boundary regularising out-of-distribution (BROOD) sampler since we construct a sampler that creates artificial data points at the boundary of the training data to regularize the decision surfaces of machine learning models. As properly generated BROOD samples are situated on the edge of the training data, they should not introduce noise. We divide the BROOD sampler's pseudo-code 1 into six steps. In the following paragraphs, we substantiate the BROOD sampler's different steps, discuss its helper functions, and elaborate on its five input parameters.

In the first three steps, we calculate preliminary results that we exploit in the last three steps. In the first step, we quantify data dispersion, enabling us to construct artificial samples at a suitable distance from the ID points in the last three steps. Moreover, we determine the data points from which we will sample in the first step. In the second step, we construct directions that we address in the future to create artificial OOD samples. In the third step, we determine the relevant neighborhoods of the ID data points from which we will sample. We will use these neighborhoods to efficiently determine potential artificial samples that could interfere with the ID data points. The first three steps are all executed once.

In the fourth step, we query a data point and find the attributes of the instances that belong to the relevant neighborhoods of this data point. In the fifth step, we create artificial OOD samples using the previously constructed directions and delete OOD samples that lie too close to the ID data points and OOD samples of the same label. The artificial OOD samples receive the same label as the point from which they are sampled. In the last step, we update the neighborhoods with the retained OOD samples to enhance sparseness and reduce duplicates.

The BROOD sampler makes use of several helper functions. In Algorithm 2, we present the pseudocode for constructing

scalers related to data dispersion. Next, Algorithm 3 contains the pseudocode of the function for sampling points on a sphere and creating directions. In Algorithm 4, we present the pseudocode to calculate Mahalanobis distances. Finally, we present the pseudocode for appropriately scaling the directions in Algorithm 5.

The first prominent input parameter of the BROOD sampler is h_{strat}. It is infeasible to sample data that cover the whole potential OOD data space. Hence, we aim to generate artificial samples that are far enough (but not too far) from the original ID samples. Thus, one must choose a suitable distance between the original data and the artificial samples. We use the extensive literature on kernel density estimation to achieve this [6]. Kernel density estimation is a nonparametric way to estimate the probability density function of a particular random variable. For this task, one chooses a suitable bandwidth that balances both bias and variance of the density estimate. Let us introduce some notation. We denote with h_i and σ_i the bandwidth and the standard deviation that correspond with dimension i, respectively. Next, we denote with d the number of dimensions and with $H = \text{diag}(h_1^2, \ldots, h_d^2)$ a matrix of squared bandwidths. By optimizing the asymptotic mean integrated squared error, under normality of the data and a Gaussian kernel function, the rule of thumb selector [6] can be derived, given by

$$\hat{h}_i = \left(\frac{4}{d+2}\right)^{\frac{1}{d+4}} n^{\frac{-1}{d+4}} \sigma_i. \tag{2}$$

We use this bandwidth to obtain a grasp of the dispersion of the data over different dimensions. Recall that we are not interested in estimating the actual densities accurately. One can also use a label-dependent version of the rule of thumb selector such as in ROSE [26], which did not cause a substantial performance increase in our experiments.

Although the (class-dependent) rule of thumb selector is often satisfactory, this choice can suffer from non-constant density regions in the feature space, as $\hat{\boldsymbol{h}}$ is not instance dependent. Hence, we provide a dispersion measure with an instance-dependent $\hat{\boldsymbol{h}}$ according to the sparseness of data regions. Denote with d_{j_k} the Euclidean distance from an instance $\boldsymbol{x}_j$ to its k-nearest neighbor and with n_{j_k} the set of k-nearest neighbors of instance $\boldsymbol{x}_j$. In [5], d_{j_k} is used as the bandwidth for kernel density estimation. Since outliers have a substantially higher d_{j_k}, we decided to use local density information of the k-nearest neighbors. Hence, instead of using the k-nearest neighbor distance of the instance itself, we take the mean of the k-nearest neighbor distance of the instance's neighbors. Notice the strong connection with (simple) local factor outlying [34]. In this case, we define $\hat{\boldsymbol{h}}$ as

$$\hat{\boldsymbol{h}}(\boldsymbol{x}_j) = \sum_{\boldsymbol{x}_i \in n_{j\_k}} d_{i\_k}. \tag{3}$$

One disadvantage of this approach is that we need to define one additional hyperparameter k. We do not claim that we propose an optimal $\hat{\boldsymbol{h}}$, and many other choices are possible.

Under the assumption that $\hat{H}$ is positive definite, we find that

$$\hat{H}^{-1} = \text{diag}\left(1/\hat{h}_1^2, \ldots, 1/\hat{h}_d^2\right). \tag{4}$$

The Mahalanobis distance between two points p_1 and p_2 with covariance matrix $\hat{H}$ equals

$$d_{(\text{mahal},\hat{H})}(p_1, p_2) = \sqrt{(p_1 - p_2)\hat{H}^{-1}(p_1 - p_2)}. \tag{5}$$

The next crucial input parameter of the BROOD sampler is $\text{query}_{\text{strat}}$. We use this parameter in the first step, as this parameter determines the ID data points we will utilize to sample artificial data points. The queried data points can be chosen randomly, by their class, or by an OOD detector such as IF [22] or SVDD [40].

The parameter $n_{\text{dir\_max}}$ is used in the second step and represents the maximal number of directions (and thus potential new artificial data points) that we will sample at each queried ID data point. We create directions by sampling points on the surface of a d-dimensional hypersphere. To generate one direction, we sample a vector $\boldsymbol{v}$ from a standard normal distribution ($\boldsymbol{v} \sim \mathcal{N}(\boldsymbol{0}, I_d)$) [36] and scale this vector accordingly such that the norm of $\boldsymbol{v}$ is equal to one. By executing this procedure multiple times, we can randomly sample points on the surface of a d-dimensional hypersphere. Note that we use these directions for every point we will sample from, as this has a significant positive effect on computation time.

Next, we want the directions to be approximately evenly spaced to cover all data regions as well as possible. To achieve this, we use the input parameter $n_{\text{dir\_samp}}$ in the second step of the BROOD sampler. We employ Mitchell's Best-Candidate algorithm [27] to ensure that the 'randomly' sampled points are approximately evenly spaced. In every iteration, we sample several potential directions and store the direction with the highest angle between all the previously stored directions. We continue iterating until the number of stored directions equals $n_{\text{dir\_max}}$. A higher value of $n_{\text{dir\_samp}}$ will in general improve the sampling policy but will increase the computation time. For every ID data point from which we sample, we accordingly scale the directions that we created. [2]

The parameter $\text{dist}_{\text{id\_ood}}$ is used in the BROOD sampler's steps $3-6$ and captures the distance between the artificial OOD samples and the ID data. This value must have the same meaning over multiple data sets. This prerequisite is met as we use Equation (5), exploiting data-dependent scalers, to calculate the distance between two data points.

Furthermore, one can impose that the distance between artificial OOD samples of the same class should be greater than a constant dist_{ood}. Using this, redundant artificial OOD samples, such as duplicates, are avoided. However, a dist_{ood} that is too high can cause fewer adjacent artificial OOD samples.

Finally, when generating artificial OOD samples, we assign the same label to the artificial OOD samples as the ID data point from which they are sampled. In the appendix, we

[2] Define $\tilde{\boldsymbol{v}} = \text{diag}(\hat{h}_1, \ldots, \hat{h}_d) \cdot \boldsymbol{v}$. One can deduce that $\|\boldsymbol{v}\|_{euclid} = \|\tilde{\boldsymbol{v}}\|_{(\text{mahal},\hat{H})}$.

provide an extension of the BROOD sampler that uses the ID data point's neighborhood to assign a label to the artificial OOD samples. This extension did not result in a substantial performance increase in our experiments. Although we assign class-specific labels to the BROOD samples, notice that they do not affect class ratios since BROOD samples are inherently different from the actual ID data. Next, our current implementation of the BROOD sampler cannot handle nominal (=categorical) features. We mitigate this problem by encoding nominal features in our data study. An extension of the BROOD sampler that can handle fully categorical or mixed data sets without encoding is a path for future research.

3.2. *Out-Of-Distribution Regularized Supervised Machine Learning Models*

Let us introduce some additional notation. We denote with v whether the data point is artificial ($v = 1$) or not ($v = 0$). We aim to minimize, with $g(\boldsymbol{x}, \boldsymbol{\theta})$ the machine learning's prediction function, the loss

$$L(X, \mathbf{y}, \boldsymbol{\theta}) = \frac{1}{n}\Big(\sum_{\boldsymbol{x}:y=1,v=0} l_{00}(g(\boldsymbol{x}, \boldsymbol{\theta})) + \sum_{\boldsymbol{x}:y=0,v=0} l_{10}(g(\boldsymbol{x}, \boldsymbol{\theta})) + \beta_t\Big(\sum_{\boldsymbol{x}:y=0,v=1} l_{01}(g(\boldsymbol{x}, \boldsymbol{\theta})) + \sum_{\boldsymbol{x}:y=1,v=1} l_{11}(g(\boldsymbol{x}, \boldsymbol{\theta}))\Big)\Big). \quad (6)$$

Although the hyperparameter β_t can be optimized during cross-validation, we always choose $\beta_t = 1$ in our experiments. Furthermore, we make use of the logistic loss that is given by

$$\begin{aligned} l_{00}(g(\boldsymbol{x}, \boldsymbol{\theta})) &= \log\left(1 + \exp\left(g(\boldsymbol{x}, \boldsymbol{\theta})\right)\right) \\ l_{10}(g(\boldsymbol{x}, \boldsymbol{\theta})) &= \log\left(1 + \exp\left(-g(\boldsymbol{x}, \boldsymbol{\theta})\right)\right). \end{aligned} \quad (7)$$

Note that in the case of a binary response $Y \in (0, 1)$ one finds that

$$\begin{aligned} \mathrm{KL}(P(Y)\|P(Y|X)) = P(Y = 1) \log\left(\frac{P(Y = 1)}{P(Y = 1|X)}\right) \\ + (1 - P(Y = 1)) \log\left(\frac{1 - P(Y = 1)}{1 - P(Y = 1|X)}\right). \end{aligned} \quad (8)$$

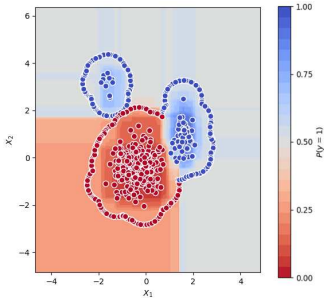
Next, we denote

$$P_{\boldsymbol{\theta}}(Y = 1|X = \boldsymbol{x}) = \frac{1}{1 + \exp\left(-g(\boldsymbol{x}, \boldsymbol{\theta})\right)}. \quad (9)$$

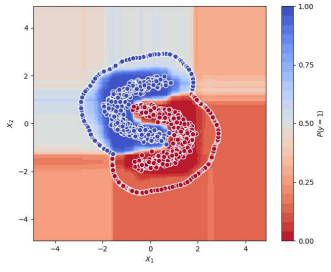
Moreover, let us denote with α the prior of the minority class and with c_{ood_0} and c_{ood_1} constants to scale the prior. These class-dependent different scales enable us to customize the decision surface in data sparse regions. By using Equation (8), we determine the loss functions $l_{01}(g(\boldsymbol{x}, \boldsymbol{\theta}))$ and $l_{11}(g(\boldsymbol{x}, \boldsymbol{\theta}))$, which are given by

$$\begin{aligned} l_{01}(g(\boldsymbol{x}, \boldsymbol{\theta})) &= c_{ood_0}\alpha \log\left(\frac{c_{ood_0}\alpha}{P_{\boldsymbol{\theta}}(Y = 1|X = \boldsymbol{x})}\right) \\ &\quad + (1 - c_{ood_0}\alpha) \log\left(\frac{(1 - c_{ood_0}\alpha))}{(1 - P_{\boldsymbol{\theta}}(Y = 1|X = \boldsymbol{x}))}\right) \\ l_{11}(g(\boldsymbol{x}, \boldsymbol{\theta})) &= c_{ood_1}\alpha \log\left(\frac{c_{ood_1}\alpha}{P_{\boldsymbol{\theta}}(Y = 1|X = \boldsymbol{x})}\right) \\ &\quad + (1 - c_{ood_1}\alpha) \log\left(\frac{(1 - c_{ood_1}\alpha))}{(1 - P_{\boldsymbol{\theta}}(Y = 1|X = \boldsymbol{x}))}\right). \end{aligned} \quad (10)$$

By using these loss functions, objective function 6 is always convex if $g(\boldsymbol{x}, \boldsymbol{\theta})$ is linear. We impose that $c_{ood_1}\alpha$ and $c_{ood_0}\alpha$ are bounded by 0 and 1. Furthermore, note that at these bounds, the loss functions (10) are the same as the logistic loss functions (7). In Figure 2, we again visualize our toy data sets with the decision surfaces trained with XGBoost. We also include the artificially generated OOD samples that we used to train the decision surfaces. One can indeed deduce that the OOD samples can regularize the decision surface and that the class-dependent constants can emphasize certain data regions.



Visual of the toy data set with BROOD samples.



Visual of the moons data set with BROOD samples.

Figure 2: Visual of decision surfaces and generated BROOD samples. The XGBoost classifier is trained using a regularizing loss. For the toy data set, we use $c_{ood_1} = 2$ and $c_{ood_0} = 1$. For the moons data set, we set $c_{ood_1} = 1$ and $c_{ood_0} = 0.2$.

3.3. *The Effect Of The BROOD Sampler's Hyperparameters*

Before we investigate our methodology in a real-life setting, we discuss the effect of the BROOD sampler's hyperparameters on both the generation of BROOD samples and the training of decision surfaces.

First, let us discuss $n_{\text{dir\_samp}}$ and $\text{query}_{\text{strat}}$. As previously mentioned, a high value of $n_{\text{dir\_samp}}$ will cause the directions to be more uniformly distributed. Concerning $\text{query}_{\text{strat}}$, we regularize the OOD spaces close to the instances from which we sample. For example, sampling from minority instances results in regularized OOD spaces close to the minority instances. We denote $\text{query}_{\text{strat}} = \text{ALL}$ if we sample from all instances and we denote $\text{query}_{\text{strat}} = \text{MIN}$ if we sample from the minority instances.

The parameter h_{strat} determines how we grasp the dispersion of the data. In subsection 3.1, we proposed multiple ways to achieve this goal. For now, we denote $h_{\text{strat}} = \text{ROT}$ if we use Equation 2 to grasp data dispersion. The rule of thumb

Algorithm 1: Boundary Regularizing Out-Of-distribution (BROOD) Sampler.

Input: X: Covariates, $\mathbf{y}$: Responses, h_{strat}: Captures how to define $\hat{H}$, $\text{query}_{\text{strat}}$: Captures the query strategy, $n_{\text{dir\_max}}$: Maximal number of directions, $n_{\text{dir\_samp}}$: Captures the number of potential directions, $\text{dist}_{\text{id\_ood}}$: The distance between artificial samples and the data point from which they are sampled

Output: X_{brood}: BROOD samples' attributes, y_{brood}: BROOD samples' labels

$\hat{H}_X = \text{scaler}(X, h_{\text{strat}})$ # $\hat{H}_X$ represents the scalers for all instances
Denote with d the dimension of X
Determine the ID data points X_{query} (and corresponding $\boldsymbol{y}_{\text{query}}$) we will use to create artificial samples. This strategy is captured in the parameter $\text{query}_{\text{strat}}$

Step 1: Capture data dispersion for every instance in X, using h_{strat}. Determine the ID data points that we will use to create artificial OOD data using $\text{query}_{\text{strat}}$.

$V := \text{dir\_samp}(n_{\text{dir\_samp}}, n_{\text{dir\_max}}, d)$

Step 2: Use $n_{\text{dir\_max}}$ and $n_{\text{dir\_samp}}$ to create and store approximately evenly spaced directions.

Determine dist_{ood} †
$D_{id} = \text{mahal}(X, X_{\text{query}}, \hat{H}_X)$
$D_{\text{id\_query}} = \text{mahal}(X_{\text{query}}, X_{\text{query}}, \hat{H}_{X_{\text{query}}})$
$\text{NGBH}_{\text{id}} := \text{where}(D_{\text{id}} < 2 \cdot \text{dist}_{\text{id\_ood}}, 1, 0)$ # element wise, 1 if true, else 0
$\text{NGBH2}_{\text{id}} := \text{where}(D_{\text{id\_query}} < 2 \cdot \text{dist}_{\text{id\_ood}} + \text{dist}_{\text{ood}}, 1, 0)$
Keep all data from X that has at least one 1 in NGBH_{id} (in the ID ngbh of X_{query}). We denote with X_{data} the retained data with label $\boldsymbol{y}_{\text{data}}$
$\text{NGBH}_{\text{id\_ood}} = \text{NGBH}_{\text{id}}$

Step 3: To enhance sparseness, define the minimum distance between artificial OOD samples (dist_{ood}). Use $\text{dist}_{\text{id\_ood}}$ and dist_{ood} to determine the relevant neighborhoods of the ID data points from which we will sample. All the ID data points that are not in at least a single neighborhood can be deleted.

for $\boldsymbol{x}_{id}$ *in* X_{query} **do**
  Get label y_{id} of $\boldsymbol{x}_{\text{id}}$
  Use $\text{NGBH}_{\text{id\_ood}}$ to get $X_{\text{NGBH\_id}}$, that is the ID data ngbh of $\boldsymbol{x}_{\text{id}}$ (distance less than $2 \cdot \text{dist}_{\text{id\_ood}}$ and ID data) and corresponding labels $\boldsymbol{y}_{\text{NGBH\_id}}$
  Use $\text{NGBH}_{\text{id\_ood}}$ to get $X_{\text{NGBH\_ood}}$, that is the OOD data ngbh of $\boldsymbol{x}_{\text{id}}$ (distance less than $\text{dist}_{\text{id\_ood}} + \text{dist}_{\text{ood}}$ and artificial OOD data) and the corresponding labels $\boldsymbol{y}_{\text{NGBH\_ood}}$
  Use NGBH2_{id} to get $X_{\text{NGBH2\_id}}$, that is the ID data 2 step ngbh of $\boldsymbol{x}_{\text{id}}$ (distance less than $2 \cdot \text{dist}_{\text{id\_ood}} + \text{dist}_{\text{ood}}$ and of the same class) and the corresponding labels $\boldsymbol{y}_{\text{NGBH2\_id}}$

Step 4: Query an ID data point from which we will create artificial OOD samples and get the attributes of the data points that are contained in the relevant neighborhoods of the queried ID data point.

  Determine matrix of scaled directions $S_{x_{id}}$ for instance $\boldsymbol{x}_{id}$ using $\text{dir\_scale}(V, \hat{H}_{x_{id}})$
  $X_{\text{samp}} = \boldsymbol{x}_{\text{id}} + S_{x_{id}}$
  $D_{\text{id\_samp}} := \text{mahal}(X_{\text{NGBH\_id}}, X_{\text{samp}}, \hat{H}_{X_{\text{NGBH\_id}}})$
  Use $D_{\text{id\_samp}}$ to determine the minimum distance between each of the samples in X_{samp} and $X_{\text{NGBH\_id}}$
  Delete the samples in X_{samp} with minimum distance from $X_{\text{NGBH\_id}}$ less than $\text{dist}_{\text{id\_ood}}$
  $D_{\text{ood\_samp}} := \text{mahal}(X_{\text{NGBH\_ood}}, X_{\text{samp}}, \min(\hat{H}_{X_{\text{NGBH\_ood}}}, \hat{H}_{x_{id}}))$
  Use $D_{\text{ood\_samp}}$ to determine the minimum distance between each of the samples in X_{samp} and $X_{\text{NGBH\_ood}}$
  Delete the samples in X_{samp} with minimum distance from $X_{\text{NGBH\_ood}}$ less than dist_{ood}
  Create $\boldsymbol{y}_{\text{samp}}$ by assigning the label y_{id} to X_{samp}
  $X_{\text{data}} = X_{\text{data}} \cup X_{\text{samp}}$
  $\boldsymbol{y}_{\text{data}} = \boldsymbol{y}_{\text{data}} \cup \boldsymbol{y}_{\text{samp}}$

Step 5: Investigate the possible directions and keep allowed artificial OOD samples. We delete artificial OOD samples that lie too close to ID data points and that lie too cloe to OOD samples of the same label. The artificial OOD samples get the same label as the point from which they are sampled.

  $D_{\text{id2\_samp}} := \text{mahal}(X_{\text{NGBH2\_id}}, X_{\text{samp}}, \hat{H}_{X_{\text{NGBH2\_id}}})$
  Append a column of zeros in $\text{NGBH}_{\text{id\_ood}}$ for every data point in X_{samp}
  Use $D_{\text{id2\_samp}}$ to determine the data points in $X_{\text{NGBH2\_id}}$ which are closer than $\text{dist}_{\text{id\_ood}} + \text{dist}_{\text{ood}}$ to at least on instance of X_{samp} and of the same class. Store this information in $\text{NGBH}_{\text{id\_ood}}$

Step 6: If new artificial OOD samples with the same label will be constructed in the future, ensure that the distances toward these new artificial OOD samples will be sufficient.

end
Get X_{brood} by extracting all artificially sampled data points from X_{data}
$\boldsymbol{y}_{\text{brood}}$ equals the artificially sampled data points' labels

† The value of dist_{ood} should be between 0 and the minimum distance between artificial OOD samples generated from the same ID data point. In our experiments, we take dist_{ood} equal to one tenth of the minimum distance between artificial OOD samples generated from the same ID data point.

Algorithm 2: Scaler (scaler(·))

Input: X: Covariates with shape (n, d), h_{strat}: Captures the query strategy
Output: $(H_0, \ldots, H_n)$: vector of scalers
if $h_{strat} = ROT$ **then**
  $\boldsymbol{\sigma} := \text{std}(X)$
  $\hat{\boldsymbol{h}} := \left(\frac{4}{d+2}\right)^{\frac{1}{d+4}} n^{\frac{-1}{d+4}} \boldsymbol{\sigma}$
  Calculate $H_i := \text{diag}\left(1/\hat{\boldsymbol{h}}^2\right)$ for every instance in X # instance independent
end
if $h_{strat} = KNN$ **then**
  Calculate $\boldsymbol{n}_k$ and $\boldsymbol{d}_k$, which respectively equals the k-nearest neighbors and the distance from its k-nearest neighbor from each instance in X
  Calculate $H_i := \text{diag}\left(1/(\sum_{\boldsymbol{x}_i \in n_{j\_k}} d_{i\_k})^2, \ldots, 1/(\sum_{\boldsymbol{x}_i \in n_{j\_k}} d_{i\_k})^2\right)$ for every instance in X # instance dependent
end

Algorithm 3: Direction Sample (dir_samp(·))

Input: $n_{\text{dir\_samp}}$: Captures the number of initial directions, $n_{\text{dir\_max}}$: Maximal number of directions, d: Data dimension
Output: V_{fi} : Matrix of directions
Initialize the matrix of final directions V_{fi} by sampling one sample from a standard normal distribution with d dimensions
while *The number of directions in V_{fi} is strictly smaller than n_{dir_max}* **do**
  Denote with $n_{\text{i\_samp}}$ the number of directions already sampled. Obtain a matrix of directions V by sampling $n_{\text{dir\_samp}} \cdot n_{\text{i\_samp}}$ samples from a standard normal distribution with d dimensions
  Construct a matrix of angles Θ between the directions in V_{fi} and V
  Use Θ to find the direction in V with the maximal minimal angle between the directions in V_{fi}. Include this direction in V_{fi}
end

Algorithm 4: Mahalanobis Distance (mahal(·))

Input: X_a (shape (n_a, d)), X_b (shape (n_b, d)), $(S_0, \ldots, S_{n\_a})$ (components have shape (d, d))
Output: D: Distance matrix (shape (n_a, n_b))
$Z := X_a[:, None, :] - X_b$ (shape (n_a, n_b, d)) # we denote with *None* a dimensionality increase
for *Each row i in D* **do**
  $A[i, :] = Z[i, :, :] \cdot S_i$
  $D[i, :] = \sqrt{A[i, :] \cdot Z[i, :, :]^T}$
end

Algorithm 5: Direction Scale (dir_scale(·))

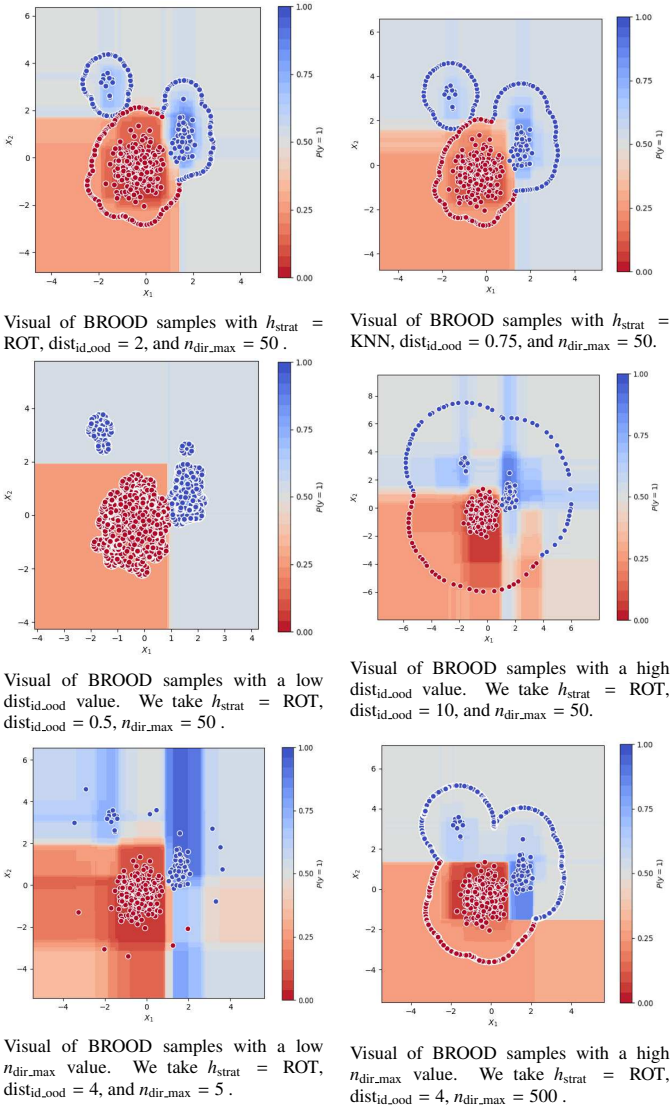
Input: V: Matrix of Directions S: Scaler
Output: V_{sc}: Matrix of scaled directions
$V_s := \sqrt{S^{-1}V}$
$D := \sqrt{V_s S V_s^T}$
$V_{sc} := V_s / D$

selector differs in every dimension and is the same for every instance. Hence, we actually create directions on a multidimensional instance-independent ellipse. Next, we denote h_{strat} = KNN if we use Equation 3 to grasp data dispersion. In this case, $\hat{\boldsymbol{h}}$ is constant but instance-dependent. Hence, we create directions on a multidimensional sphere with an instance-dependent radius. The top images of Figure 3 visualize the effect of this hyperparameter.

The parameter $\text{dist}_{\text{id\_ood}}$ determines the distance from the ID data points to the BROOD samples. A too low $\text{dist}_{\text{id\_ood}}$ will cause the BROOD samples to interfere with the ID data (i.e., introduce noise) and will negatively influence computation time due to the construction of an excessive number of BROOD samples. However, a $\text{dist}_{\text{id\_ood}}$ that is too high will attenuate the regularizing effect of the BROOD samples. As the distance of the BROOD samples from the ID data increases, the distance between the BROOD samples will also increase, and more BROOD samples will be needed to fully regularize the decision surface. Furthermore, if the BROOD samples are too far away from the ID data points, the decision surface will be more chaotic between the ID data and BROOD samples. The images in the middle of Figure 3 visualize the effect of $\text{dist}_{\text{id\_ood}}$ on the generation of BROOD samples and the decision surface trained by XGBoost in these extreme cases.

Next, the parameter $n_{\text{dir\_max}}$ determines the maximum number of artificial samples created from one ID sample. Thus, on the one hand, a $n_{\text{dir\_max}}$ that is too low will not provide sufficient BROOD samples to fully regularize the decision surface. On the other hand, a too high $n_{\text{dir\_max}}$ will have a negative influence on computation time due to an excessive number of training points. Additionally, without a proper weighting strategy, an excessively high $n_{\text{dir\_max}}$ diminishes the importance of the existing ID data points. Visualizations can be found on the bottom side of Figure 3.

Note that the number of dimensions can affect the optimal value of the $n_{\text{dir\_max}}$ parameter. Similar to other existing oversampling techniques, our method suffers from the curse of dimensionality, as it is more challenging to cover multidimensional data regions with only a few data points. Hence, for high-dimensional problems, an optimal OOD sampling strategy will generally require more OOD samples and thus require more computational effort.



Visual of BROOD samples with h_{strat} = ROT, $\text{dist}_{\text{id\_ood}} = 2$, and $n_{\text{dir\_max}} = 50$.

Visual of BROOD samples with h_{strat} = KNN, $\text{dist}_{\text{id\_ood}} = 0.75$, and $n_{\text{dir\_max}} = 50$.

Visual of BROOD samples with a low $\text{dist}_{\text{id\_ood}}$ value. We take h_{strat} = ROT, $\text{dist}_{\text{id\_ood}} = 0.5$, $n_{\text{dir\_max}} = 50$.

Visual of BROOD samples with a high $\text{dist}_{\text{id\_ood}}$ value. We take h_{strat} = ROT, $\text{dist}_{\text{id\_ood}} = 10$, and $n_{\text{dir\_max}} = 50$.

Visual of BROOD samples with a low $n_{\text{dir\_max}}$ value. We take h_{strat} = ROT, $\text{dist}_{\text{id\_ood}} = 4$, and $n_{\text{dir\_max}} = 5$.

Visual of BROOD samples with a high $n_{\text{dir\_max}}$ value. We take h_{strat} = ROT, $\text{dist}_{\text{id\_ood}} = 4$, $n_{\text{dir\_max}} = 500$.

Figure 3: Visual comparison of the effect of the BROOD sampler's hyperparameters. We visualise the effect of h_{strat}, $n_{\text{dir\_max}}$, and $\text{dist}_{\text{id\_ood}}$ on the generation of BROOD samples and the decision surface trained on a toy data set. We always take $\text{query}_{\text{strat}}$ = ALL and $n_{\text{dir\_samp}} = 50$. We impose that $c_{ood_1} = 2$ and $c_{ood_0} = 1$ and keep all XGBoost's hyperparameters constant.

Hence, these hyperparameters should be optimized using the validation set to achieve optimal performance, which can be time-consuming. Thus, there is a serious need to propose some initial rules or best practices.

Using h_{strat} = ROT, the bandwidth actually represents the standard deviation parameter of the Gaussian kernel. It is a well-known heuristic that, for a Gaussian distribution, almost all data points have a distance less than three times the standard deviation from the mean. Hence, in this case, we reason that $\text{dist}_{\text{id\_ood}}$ should be greater than three in most cases. Furthermore, as discussed previously, $\text{dist}_{\text{id\_ood}}$ should not be too large, as this has several negative implications. In our analysis, we noticed that $\text{dist}_{\text{id\_ood}}$ should be smaller than seven. As the rule of thumb estimator accounts for the number of dimensions, we believe the effect of the number of dimensions on this parameter is minimal. We found that good results can be achieved by

taking $\text{dist}_{\text{id\_ood}} = 4$ if $h_{\text{strat}} = \text{ROT}$.

Next, we derive a similar rule if one takes $h_{\text{strat}} = \text{KNN}$. Denote by S_d^r the surface area of a d-dimensional hypersphere with radius r. The k-nearest neighbor density estimator proposed in [23] equals

$$\hat{P}(\boldsymbol{x}_j) = \frac{k}{nS_d^1(d_{j_k})^d} \tag{11}$$

for $\boldsymbol{x}_j$. Hence, only the k instances with a distance smaller than or equal to d_{j_k} contribute to this nearest neighbor estimate. Hence, following this reasoning, $\text{dist}_{\text{id\_ood}}$ should be greater than one. Furthermore, from our experiments, we recommend that $\text{dist}_{\text{id\_ood}}$ should be smaller than 2. We obtain good results on many data sets by taking $\text{dist}_{\text{id\_ood}} = 1.5$. The authors of [23] reason that the optimal $k = O\left(n^{4/(d+4)}\right)$. By taking $k = 2n^{4/(d+4)}$, we are able to achieve satisfactory performance.

In our experiments, we always take $n_{\text{dir\_max}} = 30$ and $n_{\text{dir\_samp}} = 50$.

4. Experiments

In the next section, we perform an extensive data study to evaluate the performance of our proposed machine learning models. First, we discuss the balanced and imbalanced classification data sets we will use in our experiments. Next, we elaborate on our experimental methodology and introduce various performance metrics. Finally, we discuss whether BROOD samples are able to enhance the machine learning model's performance and deduce whether our approach complements existing oversampling techniques.

4.1. Experimental Design

4.1.1. Data Sets

First, we compare BROOD-enhanced models with their standard counterparts on 15 real-world benchmark data sets for binary classification. We test our models on the UCI [dataset][11] E.coli, mammographic masses, sonar, ionosphere, breast cancer Wisconsin, heart, Indian liver patient, liver disorders, breast cancer, vehicle, contraceptive method, German credit card, credit approval, bank marketing, and madelon data set. All these data sets, and corresponding code, can be found on GitHub[3]. The term 'benchmark data sets' encapsulates all the previous data sets for conciseness. A summary of the benchmark data sets can be found in Table 1. The data sets can consist of numerical (N) and categorical (C) features.

Next, we test our BROOD-enhanced machine learning models on multiple imbalanced data sets. We test our models on data from the ESA Polish bankruptcy data set [dataset][11], financial distress data set [dataset][12], UCI German credit card, UCI credit approval and UCI bank marketing data set [dataset][11], IBM Telco customer churn data set [dataset][3], ETL credit card approval data set [dataset][24], VUB credit scoring data set [dataset][44], KDD Cup 1998 data set [dataset][31], (private) car insurance fraud data set, (private) APATE credit card fraud data set, JAR financial statement fraud data set [dataset][46], IEEE-CIS customer transaction fraud data set [dataset][38], synthetic mobile transaction fraud data set [dataset][33], and synthetic credit card fraud data set [dataset][13]. The ESA Polish bankruptcy data set contains financial rates of Polish companies from a forecasting period of a year and bankruptcy status after five years. The data set consists of operating companies between 2000 and 2012 and bankrupt companies between 2007 and 2013 [50]. The financial distress data set involves information concerning financial distress for a sample of companies. The UCI German credit card data set classifies people as good or bad credit risks, and the UCI credit approval data contain credit card applications [11]. The UCI bank marketing data set contains information from a direct marketing campaign from May 2008 to November 2010 of a Portuguese banking institution [29]. The IBM Telco customer churn data set contains data from a fictional telco company's customers and their churn behavior. The ETL credit card approval data set involves information concerning credit card applicants. The VUB credit scoring data set consists of accepted loans given to individual customers from January 2016 to December 2016 by a Romanian nonbanking financial institution [32]. The KDD data set contains information on a direct mailing campaign and donations and has been used for the second international knowledge discovery and data mining tools competition. Next, we received a private car insurance fraud data set from an international insurer. The data set consists of a sample of fraudulent and nonfraudulent Belgian automobile claims from 2012 until 2021. The APATE credit card fraud data set consists of fraudulent and nonfraudulent transactions from a large Belgian credit card issuer [41]. We use an already preprocessed version of this data set in our study. The JAR financial statement fraud data set is a manually gathered data set [2] that contains accounting fraud samples from the SEC's Accounting and Auditing Enforcement Releases (AAERs), enhanced with financial accounting data. The data set consists of fraud and legitimate instances between 1990 and 2014. The IEEE-CIS customer transaction fraud data set consists of real-world e-commerce transactions. The synthetic mobile transaction fraud data set is a synthetic data set based on a sample of real transactions from a mobile money service in an African country over multiple periods. Finally, the synthetic credit card fraud data set is a synthetic data set that contains transactions of synthetic consumers who reside in the United States.

We execute some additional preprocessing steps for some imbalanced data sets. We randomly undersample certain imbalanced data sets to ensure empirical tractability. Furthermore, if the data set is not severely imbalanced, we randomly undersample its minority class. We use the provided summary variables for the KDD data set to include the promotion and donation histories. The JAR financial statement fraud data set spans several periods [2]. The fact that the training and testing sets could contain the same firm (and that the data contain firm-specific characteristics) can seriously enhance the performance of machine learning methods. To ensure that no serial correlation is present in the financial statement fraud data set, we keep

[3] `https://github.com/CFLDT/Regularization-Oversampling-for-Classification-Tasks-To-Exploit-What-You-Do-Not-Know`

	Class 1	Class 0	# Samples	Ratio	# Dimensions
UCI E.coli (N)	pp	all other	366	0.155	7
UCI Mammographic Masses (C)	1	0	961	0.463	4
UCI Sonar (N)	M	R	208	0.536	60
UCI Ionosphere (N)	b	g	351	0.359	34
UCI Breast Cancer Wisconsin (N)	M	B	569	0.373	30
UCI Heart (N & C)	2	1	270	0.444	13
UCI Indian Liver Patient (N & C)	2	1	583	0.286	10
UCI Liver Disorders (N)	consumption > 0.5	consumption ≤ 0.5	345	0.339	5
UCI Breast Cancer (C)	recurrence -events	no-recurrence -events	286	0.297	9
UCI Vehicle (N)	VAN	all others	964	0.235	18
UCI Contraceptive method (C)	long term	all others	1,473	0.226	9
UCI German Credit Card (N & C)	2	1	1,000	0.300	20
UCI Credit Approval (N & C)	+	−	690	0.445	15
UCI Bank Marketing (N & C)	yes	no	4,119	0.109	20
UCI Madelon (N)	1	0	2,600	0.500	501

Table 1: Summary of the benchmark data sets. The data sets can consist of numerical (N) and categorical (C) features.

only the first fraud case of a particular firm in the data set. For the synthetic mobile transaction fraud data set, we extract the first eight periods of the simulation. Next, we create aggregated features that, for instance, capture the number of transactions the recipient already received during this period. For the synthetic credit card fraud data set, we extract the first three customers of the simulation. For this data set, we also create aggregate features that, for instance, capture the number of transactions the customer has done in the last 24 hours. A summary of the imbalanced data sets after these general preprocessing steps can be found in Table 2. Despite the private car insurance fraud data set and private APATE credit card fraud data set, all preprocessed imbalanced data sets used in this paper can be found on GitHub.

4.1.2. Methodology

We use random stratified sampling to obtain an initial training set consisting of 75% of all instances and a test set consisting of 25% of all instances. Furthermore, we split the initial training set (in a stratified way) into the final training set (50% of all instances) and a validation set (25% of all instances). We train the model on the final training set, use the validation set to optimize the models' hyperparameters, and assess performance on the test set. We execute this procedure four times in a row for every data set, each time with a completely different test set.

We train logistic regression benchmark models and XGBoost [8] models with and without artificially generated BROOD samples. Logistic regression is the typical benchmark model and is widely used in various experimental studies. The XGBoost algorithm is a boosted tree algorithm that involves combining many trees and uses gradient information to choose the direction it needs to improve the fit to the data. The XGBoost algorithm is known to be computationally efficient and can control for overfitting. More details about our pipeline and the set of hyperparameters can be found in the appendix.

We use three evaluation metrics to assess the performance of our algorithms. First, we compare the area under the receiving operating curve (ROC-AUC) of the different models to compare the aggregate performance (over different thresholds) and minimize misclassification errors [37]. In short, the ROC-AUC measures how well a model can distinguish the two classes. We denote with $\text{TPR}(\phi) = \mathbb{E}_{y_1}[g_\phi(\boldsymbol{x}, \boldsymbol{\theta})]$ the true positive rate for threshold ϕ and with $\text{FPR}(\phi) = \mathbb{E}_{y_0}[g_\phi(\boldsymbol{x}, \boldsymbol{\theta})]$ the false positive rate for threshold ϕ. The ROC-AUC is equal to

$$\text{ROC-AUC} = \int_0^1 \text{TPR}(\phi) d\text{FPR}(\phi). \tag{12}$$

Next, we compare the average precision (AP). The AP equals the sum of precisions, with the increase in TPR (also known as recall) over different thresholds as weights. This measure is suitable for imbalanced learning because it does not incorporate the algorithm's ability to detect majority instances. The precision equals $\text{PR}(\phi) = \mathbb{E}_{y_1}[g_\phi(\boldsymbol{x}, \boldsymbol{\theta})] / \mathbb{E}_y[g_\phi(\boldsymbol{x}, \boldsymbol{\theta})]$ for threshold ϕ. Over thresholds $\phi_1, \ldots, \phi_n$, the AP equals

	Class 1	Class 0	# Samples	Ratio	# Dimensions
ESA Polish Bankruptcy (N)	1	0	7,027	0.0386	64
Financial Distress (N)	financial distress < -0.5	financial distress ≥ -0.5	3,672	0.0370	84
UCI German Credit Card IMB (N & C)	2	1	760	0.0789	20
UCI Credit Approval IMB (N & C)	+	-	414	0.0749	15
UCI Bank Marketing IMB (N & C)	yes	no	3,758	0.0239	20
IBM Telco Customer Churn (N & C)	Yes	No	5,548	0.0674	19
ETL Credit Card Approval (N & C)	1	0	12,654	0.0043	17
VUB Credit Scoring (N & C)	1	0	16,352	0.0392	18
KDD Cup 1998 (N & C)	1	0	14,312	0.0519	394
(Private) Car Insurance Fraud (N & C)	1	0	16,308	0.0210	23
(Private) APATA Credit Card Fraud (N & C)	True	False	7,279	0.0076	7
JAR Financial Statement Fraud (N & C)	1	0	14,910	0.0270	42
IEEE-CIS Customer Transaction Fraud (N & C)	1	0	11,811	0.0328	391
Synthetic Mobile Transaction Fraud (N & C)	1	0	3,591	0.0251	13
Synthetic Credit Card Fraud (N & C)	Yes	No	5,784	0.0214	15

Table 2: Summary of the severely imbalanced data sets. The data sets can consist of numerical (N) and categorical (C) features. To ensure severe imbalance and analytical tractability, certain data sets are undersampled versions of the original data sets.

$$\text{AP} = \sum_{i\in(2,\ldots,n)} \Big(\text{TPR}(\phi_i) - \text{TPR}(\phi_{i-1})\Big)\text{PR}(\phi_i). \tag{13}$$

Finally, we use the discounted cumulative gain (DCG) that was also used in [2] to assess financial statement fraud. This metric is material for fraud detection algorithms, as there are often insufficient resources to investigate all suspicious cases. Similarly, in churn and marketing data sets, there is only a limited budget to target certain customers. Hence, detecting fraud, and targeting valuable customers can be seen as a ranking problem. The formula of discounted cumulative gain that we use in our analysis is given by

$$\text{DCG}(b) = \sum_{i=1}^{b} \frac{2^{\text{rel}_i} - 1}{\log_2(i+1)}. \tag{14}$$

The relevance (rel) equals one if the instance is part of the positive class and zero otherwise. We can limit the performance evaluation to only a small number (i.e., b as defined above) of data points with the highest predicted probability. We take the parameter b equal to the number of instances in the positive class in the test set.

To compare all classifiers to each other, we use the non-parametric Nemenyi test and construct critical difference plots [9]. The Nemenyi test for pairwise comparisons of k classifiers on n data sets with critical value q_α (confidence level α) states that if their rank differs by at least the critical difference (CD)

$$\text{CD} = q_\alpha \sqrt{\frac{k(k+1)}{6n}}, \tag{15}$$

the classifiers are significantly different. In the figures that we construct, the classifiers connected by the bold horizontal line represent the not significantly different classifiers with a confidence level of 0.90.

Let us discuss the BROOD sampler's hyperparameters used in our experiments and introduce some new notations to enhance conciseness. First, we always take $n_{\text{dir\_max}} = 30$ and $n_{\text{dir\_samp}} = 50$. We denote with 'ALL' if $\text{query}_{\text{strat}} = \text{ALL}$ and we denote with 'MIN' if $\text{query}_{\text{strat}} = \text{MIN}$. We use the notation 'ROT' if we generate BROOD samples by taking $h_{\text{strat}} = \text{ROT}$ and $\text{dist}_{\text{id\_ood}} = 4$. Furthermore, we use the notation 'KNN' if we generate BROOD samples by taking $h_{\text{strat}} = \text{KNN}$ and $\text{dist}_{\text{id\_ood}} = 1.5$. For example, BROOD(ALL_ROT) uses all instances to generate OOD samples with $n_{\text{dir\_max}} = 30$, $n_{\text{dir\_samp}} = 50$, $h_{\text{strat}} = \text{ROT}$, and $\text{dist}_{\text{id\_ood}} = 4$. We never optimize the BROOD sampler's hyperparameters to substantiate its real-life practicability. We do not claim that any of our proposed hyperparameter combinations are optimal.

Next, we impose that β_t (see Equation 6) equals one[4]. Finally, we find that logistic regression models are not able to

[4]Nevertheless, one should tune β_t to obtain the best performance. If including artificial samples (SMOTE, ADASYN, ROSE, BROOD, ...) reduces the model's performance on the validation set, using them for training the model would not be optimal. Second, if the number of BROOD samples is low or high

handle the nonlinearities that can arise from the BROOD samples in particular data sets. Hence, we decided not to discuss BROOD-enhanced logistic regression models.

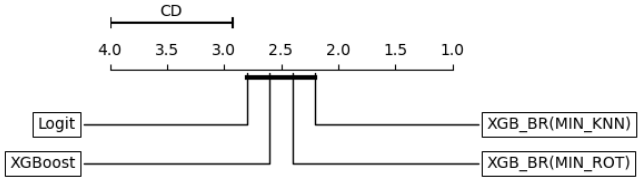
4.2. Experimental Results

4.2.1. Benchmark Data Sets

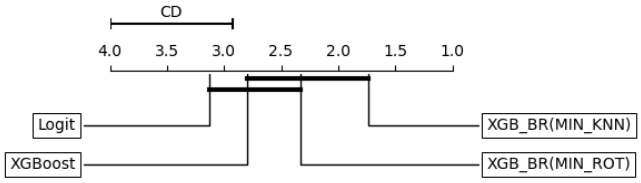
We impose a maximum of 10,000 BROOD samples on the benchmark data sets. If this bound is reached, we only create BROOD samples from the most outlying data points. We create critical difference diagrams to assess the difference in the predictive power of the machine learning models with a significance level of 0.10. We always include logistic regression as a benchmark model.

Figure 4 visualizes the Nemenyi test on the ROC-AUC, AP and DCG of the different classifiers on the benchmark data sets. First, we find that the XGBoost model achieves a higher average rank than the logistic regression model on all the performance metrics. This can be explained by the fact that XGBoost can construct nonlinear decision boundaries. Next, due to the simplicity of the benchmark data sets, there are likely none (or a limited number) of OOD data points in the test sets. Hence, it is essential to show that using BROOD samples does not result in overfitting or a performance decrease (or both) on the test set. For the ROC-AUC, on which we cross-validate, we can deduce that the normal and BROOD-enhanced XGBoost models achieve similar performance. Although we did optimize the model's hyperparameters on the ROC-AUC, the other performance metrics shed some light on the power of BROOD samples. We find that the BROOD-enhanced XGBoost models in general perform at least as well as the regular XGBoost model on the AP and DCG metrics. Furthermore, the BROOD(ALL_KNN)-enhanced XGBoost classifier performs better than the BROOD(ALL_ROT)-enhanced XGBoost classifier on all performance metrics on the benchmark data sets. Table 3 contains the average number of artificial BROOD samples created during training. On average, the BROOD(ALL_KNN) sampler samples fewer artificial points than the BROOD(ALL_ROT) sampler with the current choice of hyperparameters.

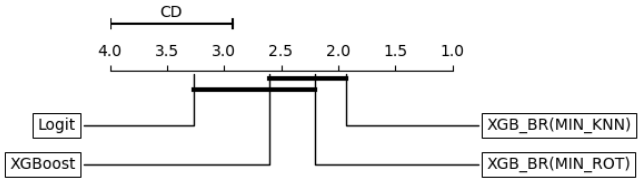
compared to the number of ID samples, it may be appropriate to weigh them appropriately. Finally, one can interpret the terms that account for BROOD samples in the objective function as regularization terms, and one often optimizes the weight of such terms.



Critical difference diagram for the ROC-AUC metric of the models that are trained on the benchmark data sets.



Critical difference diagram for the AP metric of the models that are trained on the benchmark data sets.



Critical difference diagram for the DCG metric of the models that are trained on the benchmark data sets.

Figure 4: Performance metrics: critical difference diagrams with significance level 0.10 of the models that are trained on the benchmark data sets. We use BROOD(ALL_ROT) ($n_{\text{dir\_max}} = 30$, $n_{\text{dir\_samp}} = 50$, $\text{dist}_{\text{id\_ood}} = 4$) and BROOD(ALL_KNN) ($n_{\text{dir\_max}} = 30$, $n_{\text{dir\_samp}} = 50$, $\text{dist}_{\text{id\_ood}} = 1.5$) to generate BROOD samples. We abbreviate 'XGBoost' as ' XGB' and 'BROOD' as 'BR'.

4.2.2. Severely Imbalanced Data Sets

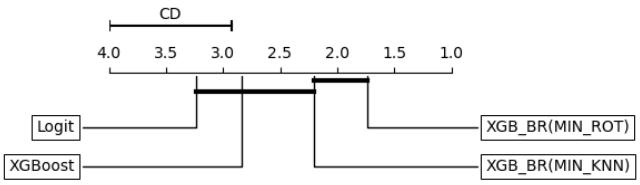
This section examines the usefulness of the BROOD sampler for developing machine learning models on severely imbalanced data sets. We investigate whether BROOD samples can aid machine learning models in learning a suitable decision surface to classify positive instances correctly.

Hence, both AP and DCG are suitable metrics, as one is more interested in the model's ability to detect minority instances in many applications. Furthermore, in many applications, there is only a limited pool of resources to target positive instances. In this case, the DCG is even more appropriate because it is only affected by the most likely positive instances in the test set. Hence, we use the DCG to optimize the model's hyperparameters during cross-validation in this subsection.

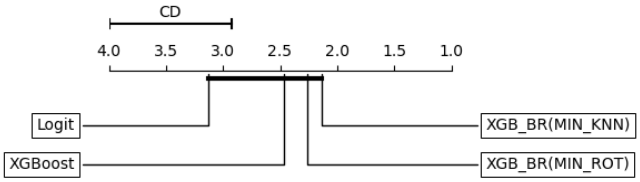
The query strategies' decision (this captures the ID data points from which we will sample) is related to the previous paragraph. We are less interested in the model's ability to classify majority instances correctly. Hence, we decide to sample BROOD samples from the minority class to indicate the OOD spaces closer to the minority instances. Furthermore, this allows us to compare BROOD sampling with existing minority oversampling techniques. As the number of positive instances is limited, the BROOD sampler will be quite fast, and the number of BROOD samples will be relatively low. This implies that the negative impact of sampling on the computation time is

limited.

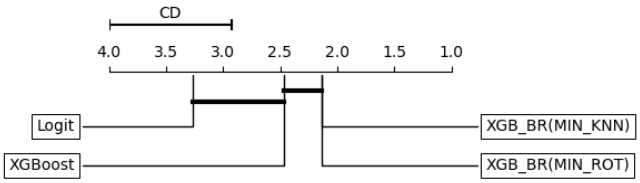
By creating BROOD samples from a limited number of data points (only the minority instances), we find that the BROOD-enhanced XGBoost models achieve a higher average rank than the regular XGBoost model on all performance metrics on the severely imbalanced data sets. We find that the XGBoost model that exploits BROOD samples using the rule of thumb selector with the standard hyperparameters outperforms the regular counterpart on the DCG metric. Moreover, we find that the BROOD-enhanced XGBoost classifiers are at least able to attain a similar performance as the standard XGBoost model on the ROC-AUC and AP. One can find the numerical values of the various performance metrics in Table 10 in the appendix.



Critical difference diagram for the DCG metric of the models that are trained on the severely imbalanced data sets.



Critical difference diagram for the AP metric of the models that are trained on the severely imbalanced data sets.



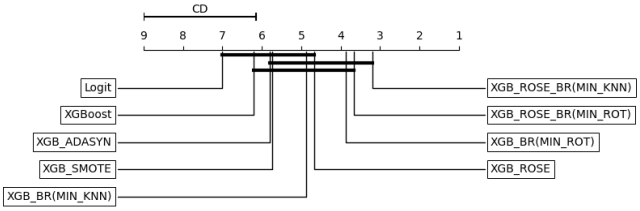
Critical difference diagram for the ROC-AUC metric of the models that are trained on the severely imbalanced data sets.

Figure 5: Performance metrics: critical difference diagrams with significance level 0.10 of the models that are trained on the severely imbalanced data sets. We use BROOD(MIN_ROT) ($n_{\text{dir\_max}} = 30$, $n_{\text{dir\_samp}} = 50$, $\text{dist}_{\text{id\_ood}} = 4$) and BROOD(MIN_KNN) ($n_{\text{dir\_max}} = 30$, $n_{\text{dir\_samp}} = 50$, $\text{dist}_{\text{id\_ood}} = 1.5$) to generate BROOD samples. The best-performing model is in bold. We abbreviate 'XGBoost' as ' XGB' and 'BROOD' as 'BR'.

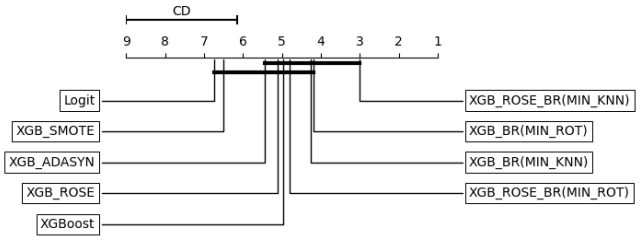
Remember that existing minority oversampling techniques aim to cover more broad data regions to aid machine learning models in learning less specific data regions. Our oversampling technique aids in regularizing the machine learning model's decision surface by assigning predetermined probabilities in previously unseen data regions. Although the purpose of BROOD sampling is different, it is interesting to assess the performance differences on severely imbalanced data sets.

We find that ROSE is the best-performing minority oversampling technique. We include the combination of both ROSE and BROOD sampling to investigate whether BROOD samples complement existing minority oversampling methods. We first execute the ROSE sampler on the training set and execute the BROOD sampler on real minority data points and synthetic ROSE minority samples with the previously mentioned hyperparameters.

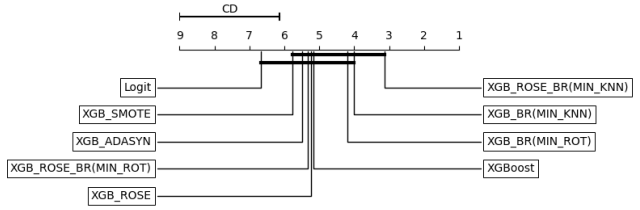
The performance of all our models on the imbalanced data sets can be found in Figure 6. The BROOD-enhanced XGBoost models are at least able to attain a similar performance as the regular XGBoost model on all performance metrics on the imbalanced data sets. Next, we see that all XGBoost models that exploit artificial samples achieve a higher average rank than the standard XGBoost model on the DCG metric, on which we cross-validate. We find that BROOD-enhanced XGBoost models are at least able to attain a similar performance as the XGBoost models exploiting artificial minority samples. Moreover, we find that the XGBoost models exploiting both ROSE and BROOD samples achieve a higher average rank than all models on the DCG metric on the imbalanced data sets. The DCG of the XGBoost model exploiting both ROSE and BROOD(MIN_KNN) samples is significantly different compared to the standard XGBoost model. The numerical values of the various performance metrics can be found in Table 10 in the appendix. Next, Table 4 contains the average number of artificial BROOD samples created during training.



Critical difference diagram for the DCG metric of the models that are trained on the severely imbalanced data sets.



Critical difference diagram for the AP metric of the models that are trained on the severely imbalanced data sets.



Critical difference diagram for the ROC-AUC metric of the models that are trained on the severely imbalanced data sets.

Figure 6: Performance metrics: critical difference diagrams with significance level 0.10 of the models that are trained on the severely imbalanced data sets. We use BROOD(MIN_ROT) ($n_{\text{dir\_max}} = 30$, $n_{\text{dir\_samp}} = 50$, $\text{dist}_{\text{id\_ood}} = 4$) and BROOD(MIN_KNN) ($n_{\text{dir\_max}} = 30$, $n_{\text{dir\_samp}} = 50$, $\text{dist}_{\text{id\_ood}} = 1.5$) to generate BROOD samples. We use the targeted imbalance ratio that on average performs best on all data sets. The stand alone SMOTE, ADASYN, and ROSE samplers aim to achieve an imbalance ratio of 0.20. The ROSE sampler along with the BROOD sampler aim to achieve an imbalance ratio of 0.10. The best-performing model is in bold. We abbreviate 'XGBoost' as ' XGB' and 'BROOD' as 'BR'.

5. Discussion

We wrote this manuscript to address the arbitrary overconfidence of discriminative machine learning methods in unseen data regions. We believe that this issue has not yet received enough attention, and many methodologies tackling this problem can still be explored. We mitigate this problem by sampling artificial points on the edge of the training data using the BROOD sampler.

On the other hand, minority oversampling methods for imbalanced classification are very popular and well-documented. Imbalanced classification is also an application where BROOD samples can improve performance. Of course, discriminative machine learning models tend to be overconfident in unseen data regions, and imbalanced data sets lack sufficient information to model the minority class correctly. As the goal of minority oversampling and OOD sampling is inherently different, their combination has not been discussed in previous literature. In presenting this manuscript, we aim to inspire researchers to investigate new usages of artificial samples in machine learning.

Our methodology and research have several limitations. First, BROOD samples have little value in applications where one does not encounter data points in unseen data spaces during the test/production phase. Next, similar to all oversampling methods, our method increases the learning time. Furthermore, BROOD samples can diminish the importance of the original data points and improperly generated BROOD samples can add little value or even introduce noise. As discussed before, our current implementation of the BROOD sampler cannot handle nominal (=categorical) features.

We did compare our model in an extensive data study with benchmark models and existing popular minority oversamplers. However, comparisons with existing OOD samplers result in some incompatibility issues. Existing research generally uses GANs or VAEs, which are often too complex for tabular and low-dimensional data. Moreover, existing OOD samplers usually do not distinguish between class labels, which gives comparability issues for the imbalanced data set experiment, where we only sample from minority instances. Furthermore, OOD samples are often used for other goals, such as OOD detection or open-set recognition. This is inherently different from our goal in the experiments, which is to improve classifier performance (without any unseen classes). Finally, existing OOD samplers differ in choosing/bounding the number of OOD samples, which is an important parameter and can seriously affect classifier performance. A survey paper discussing and comparing existing OOD samplers in different fields and applications would be an exciting path for further research.

	ALL_ROT		ALL_KNN	
UCI E.coli	440.00	(1) 61.00 (0) 379.00	368.00	(1) 43.25 (0) 324.75
UCI Mammographic Masses	153.25	(1) 101.25 (0) 52.00	169.25	(1) 101.25 (0) 68.00
UCI Sonar	3,059.50	(1) 1,660.75 (0) 1,398.75	1,470.75	(1) 803.00 (0) 667.75
UCI Ionosphere	3,772.25	(1) 1,806.25 (0) 1,966.00	3,118.00	(1) 1,585.50 (0) 1,532.50
UCI Breast Cancer Wisconsin	7,501.00	(1) 3,055.25 (0) 4,446.25	4,603.50	(1) 1,966.50 (0) 2,637.00
UCI Heart	2,372.75	(1) 1,140.00 (0) 1,232.75	940.75	(1) 464.00 (0) 476.75
UCI Indian Liver Patient	1,614.50	(1) 203.00 (0) 1,411.5	1,001.75	(1) 212.50 (0) 789.25
UCI Liver Disorders	559.75	(1) 159.00 (0) 400.75	397.75	(1) 137.25 (0) 260.50
UCI Breast Cancer	958.25	(1) 386.00 (0) 572.25	401.75	(1) 168.50 (0) 233.25
UCI Vehicle	5,483.50	(1) 1,241.50 (0) 4,242.00	2,341.25	(1) 571.25 (0) 1,770.00
UCI Contraceptive method	1,473.75	(1) 200.75 (0) 1,273.00	576.00	(1) 105.25 (0) 470.75
UCI German Credit Card	4,583.00	(1) 1,632.25 (0) 2,950.75	1,256.00	(1) 458.75 (0) 797.25
UCI Credit Approval	3,909.75	(1) 2,079.50 (0) 1,830.25	1,897.50	(1) 1,047.75 (0) 849.75
UCI Bank Marketing	9,724.25	(1) 3,391.50 (0) 6,332.75	4,298.25	(1) 1,116.00 (0) 3,183.75
UCI Madelon	9,985.75	(1) 4,436.00 (0) 5,549.75	9,994.75	(1) 4,726.75 (0) 5,268.00

Table 3: The average number of BROOD samples generated from the benchmark data sets. We use BROOD(ALL_ROT) ($n_{\text{dir\_max}} = 30$, $n_{\text{dir\_samp}} = 50$, $\text{dist}_{\text{id\_ood}} = 4$) and BROOD(ALL_KNN) ($n_{\text{dir\_max}} = 30$, $n_{\text{dir\_samp}} = 50$, $\text{dist}_{\text{id\_ood}} = 1.5$) to generate BROOD samples. We impose a maximum bound of $10,000$ BROOD samples. The sum of the average number of BROOD samples sampled from class 1 (denoted with (1)) and class 0 (denoted with (0)) equals the total number of average BROOD samples.

	MIN_ROT	MIN_KNN	ROSE_MIN_ROT	ROSE_MIN_KNN
ESA Polish Bankruptcy	404.75	404.25	5,980.00	729.25
Financial Distress	1,504.00	84.50	4,645.25	496.25
UCI German Credit Card IMB	558.50	165.50	655.75	191.50
UCI Credit Approval IMB	379.75	206.50	474.50	218.00
UCI Bank Marketing IMB	954.75	370.00	4,740.50	1,657.00
IBM Telco Customer Churn	2,961.50	1,161.75	4,360.00	1,836.25
ETL Credit Card Approval	242.75	71.75	6,649.00	831.75
VUB Credit Scoring	2,521.75	925.50	10,990.00	5,319.75
KDD Cup 1998	9,128.25	2,323.50	17,257.50	3,957.75
(Priv.) Car Insurance Fraud	2,372.75	977.50	19,713.75	6,476.25
(Priv.) APATA Credit Card Fraud	425.00	50.00	1,564.25	558.00
JAR Financial Statement Fraud	2,003.50	1,580.50	17,424.75	13,338.00
IEEE-CIS Cus. Trans. Fraud	2,728.50	142.00	13,944.00	267.00
Synth. Mobile Trans. Fraud	159.50	136.25	1,444.75	528.25
Synthetic Credit Card Fraud	1,623.50	466.75	7,428.75	1,711.75

Table 4: The average number of BROOD samples generated from the severely imbalanced data sets. We use BROOD(MIN_ROT) ($n_{\text{dir\_max}} = 30$, $n_{\text{dir\_samp}} = 50$, $\text{dist}_{\text{id\_ood}} = 4$) and BROOD(MIN_KNN) ($n_{\text{dir\_max}} = 30$, $n_{\text{dir\_samp}} = 50$, $\text{dist}_{\text{id\_ood}} = 1.5$) to generate BROOD samples. We use the targeted imbalance ratio that on average performs best on all data sets. The ROSE sampler along with the BROOD sampler aim to achieve an imbalance ratio of 0.10.

6. Conclusion and Future Research

In this paper, we develop a methodology that mitigates the excessive confidence of discriminative machine learning models in regions where little information is known. We propose the easily customizable BROOD sampler that generates artificial data points at the boundary of the training data.

This sampler allows assigning a customized probability to OOD regions, mitigating the unfounded randomness and overconfidence of discriminative classifiers in unseen data regions. We show that the BROOD-enhanced XGBoost models perform at least as well as the regular XGBoost model on balanced and imbalanced data sets. We show that with a suitable choice of hyperparameters for the BROOD sampler, the BROOD-enhanced XGBoost model outperforms the regular XGBoost model on the DCG on severely imbalanced data sets.

We discuss the similarities and differences with existing oversampling techniques. Although the purpose of BROOD sampling is different compared to existing minority oversampling techniques, we compare the performance of models that exploit BROOD samples sampled from the minority instances with those that exploit SMOTE, ADASYN and ROSE samples. We find that XGBoost models exploiting BROOD samples are at least able to attain a similar performance as these oversampling methods. Next, we discover that the performance of XGBoost models can be further enhanced by exploiting the combination of artificial BROOD and ROSE samples on severely imbalanced data sets. With a suitable choice of hyperparameters for the ROSE and the BROOD sampler, we find that the XGBoost model exploiting both ROSE and BROOD samples outperforms the regular XGoost model on the DCG in severely imbalanced classification tasks.

The current implementation of the BROOD sampler is able to handle multiple classes. Hence, for further research, we will investigate the use of unlabeled samples for multiclass classification tasks. Furthermore, we plan to develop a suitable BROOD sampler that can handle categorical features without encoding and to apply our methodology in a cost-sensitive setting.

7. CRediT Authorship Contribution Statement

Van der Schraelen Lennert: Conceptualization, Methodology, Software, Validation, Formal analysis, Writing – original draft. **Stouthuysen Kristof**: Conceptualization, Writing – review & editing, Supervision. **Vanden Broucke Seppe**: Conceptualization, Writing – review & editing. **Verdonck Tim**: Conceptualization, Writing – review & editing, Supervision.

8. Declaration of Interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

9. Funding

This research was supported by the partners of the Centre for Financial Leadership and Digital Transformation at Vlerick Business School.

References

[1] Abdallah, A., Maarof, M.A., Zainal, A., 2016. Fraud detection system: A survey. Journal of Network and Computer Applications 68, 90–113. doi:https://doi.org/10.1016/j.jnca.2016.04.007.

[2] Bao, Y., Ke, B., Li, B., Yu, Y.J., Zhang, J., 2020. Detecting accounting fraud in publicly traded us firms using a machine learning approach. Journal of Accounting Research 58, 199–235. doi:https://doi.org/10.1111/1475-679X.12292.

[3] blastchar, n.d. Telco customer churn. https://www.kaggle.com/datasets/blastchar/telco-customer-churn. Retrieved on Sep 02, 2022.

[4] Breiman, L., 1996. Bagging predictors. Machine learning 24, 123–140. doi:https://doi.org/10.1023/A:1018054314350.

[5] Breiman, L., Meisel, W., Purcell, E., 1977. Variable kernel estimates of probability density estimates. Technometrics 19, 135–144. doi:https://doi.org/10.2307/1268623.

[6] Chacón, J.E., Duong, T., 2018. Multivariate kernel smoothing and its applications. Chapman and Hall/CRC. doi:https://doi.org/10.1201/9780429485572.

[7] Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P., 2002. Smote: synthetic minority over-sampling technique. Journal of artificial intelligence research 16, 321–357. doi:https://doi.org/10.1613/jair.953.

[8] Chen, T., Guestrin, C., 2016. Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, pp. 785–794. doi:https://doi.org/10.1145/2939672.2939785.

[9] Demšar, J., 2006. Statistical comparisons of classifiers over multiple data sets. The Journal of Machine Learning Research 7, 1–30. doi:https://dl.acm.org/doi/10.5555/1248547.1248548.

[10] Douzas, G., Bacao, F., Last, F., 2018. Improving imbalanced learning through a heuristic oversampling method based on k-means and smote. Information Sciences 465, 1–20. doi:https://doi.org/10.1016/j.ins.2018.06.056.

[11] Dua, D., Graff, C., 2017. UCI machine learning repository. http://archive.ics.uci.edu/ml.

[12] Ebrahimi, n.d. Financial distress prediction. https://www.kaggle.com/datasets/shebrahimi/financial-distress. Retrieved on Nov 23, 2022.

[13] Erik, A., n.d. Credit card transactions. https://www.kaggle.com/datasets/ealtman2019/credit-card-transactions. Retrieved on June 6, 2022.

[14] Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., Herrera, F., 2011. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42, 463–484. doi:https://doi.org/10.1109/TSMCC.2011.2161285.

[15] Ge, Z., Demyanov, S., Chen, Z., Garnavi, R., 2017. Generative openmax for multi-class open set classification. arXiv preprint arXiv:1707.07418 doi:https://doi.org/10.48550/arXiv.1707.07418.

[16] Geng, C., Huang, S.j., Chen, S., 2020. Recent advances in open set recognition: A survey. IEEE transactions on pattern analysis and machine intelligence 43, 3614–3631. doi:https://doi.org/10.1109/TPAMI.2020.2981604.

[17] Goodfellow, I., 2016. Nips 2016 tutorial: Generative adversarial networks. Conference on Neural Information Processing Systems doi:https://doi.org/10.48550/arXiv.1701.00160.

[18] He, H., Bai, Y., Garcia, E.A., Li, S., 2008. Adasyn: Adaptive synthetic sampling approach for imbalanced learning, in: 2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence), IEEE. pp. 1322–1328. doi:https://doi.org/10.1109/IJCNN.2008.4633969.

[19] Kingma, D.P., Welling, M., et al., 2019. An introduction to variational autoencoders. Foundations and Trends® in Machine Learning 12, 307–392.

[20] Lee, K., Lee, H., Lee, K., Shin, J., 2018. Training confidence-calibrated classifiers for detecting out-of-distribution samples. International Conference on Learning Representations doi:https://doi.org/10.48550/arXiv.1711.09325.

[21] Liang, S., Li, Y., Srikant, R., 2018. Enhancing the reliability of out-of-distribution image detection in neural networks. International Conference on Learning Representations doi:https://doi.org/10.48550/arXiv.1706.02690.

[22] Liu, F.T., Ting, K.M., Zhou, Z.H., 2008. Isolation forest, in: 2008 eighth IEEE international conference on data mining, IEEE. pp. 413–422. doi:https://doi.org/10.1109/ICDM.2008.17.

[23] Mack, Y., Rosenblatt, M., 1979. Multivariate k-nearest neighbor density estimates. Journal of Multivariate Analysis 9, 1–15. doi:https://doi.org/10.1016/0047-259X(79)90065-4.

[24] Mario, C., 2022. ETL credit card data set. https://github.com/caesarmario/etl-credit-card-dataset-using-pentaho. Retrieved on Sep 09, 2022.

[25] Meinke, A., Hein, M., 2019. Towards neural networks that provably know when they don't know. International Conference on Learning Representations doi:https://doi.org/10.48550/arXiv.1909.12180.

[26] Menardi, G., Torelli, N., 2014. Training and assessing classification rules with imbalanced data. Data mining and knowledge discovery 28, 92–122. doi:https://doi.org/10.1007/s10618-012-0295-5.

[27] Mitchell, D.P., 1991. Spectrally optimal sampling for distribution ray tracing, in: Proceedings of the 18th annual conference on Computer graphics and interactive techniques, pp. 157–164. doi:https://doi.org/10.1145/127719.122736.

[28] Moller, F., Botache, D., Huseljic, D., Heidecker, F., Bieshaar, M., Sick, B., 2021. Out-of-distribution detection and generation using soft brownian offset sampling and autoencoders, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 46–55. doi:https://doi.org/10.48550/arXiv.2105.02965.

[29] Moro, S., Cortez, P., Rita, P., 2014. A data-driven approach to predict the success of bank telemarketing. Decision Support Systems 62, 22–31. doi:https://doi.org/10.1016/j.dss.2014.03.001.

[30] Ng, A., Jordan, M., 2001. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. Advances in neural information processing systems. 14. doi:https://doi.org/10.1007/s11063-008-9088-7.

[31] Parsa, I., Howes, K., 1998. Kdd cup 1998 data. https://kdd.ics.uci.edu/databases/kddcup98/kddcup98.html. Retrieved on Oct 04, 2022.

[32] Petrides, G., Moldovan, D., Coenen, L., Guns, T., Verbeke, W., 2022. Cost-sensitive learning for profit-driven credit scoring. Journal of the Operational Research Society 73, 338–350. doi:https://doi.org/10.1080/01605682.2020.1843975.

[33] Rojas, E.L., n.d. Synthetic financial datasets for fraud detection. https://www.kaggle.com/datasets/ealaxi/paysim1. Retrieved on May 5, 2022.

[34] Schubert, E., Zimek, A., Kriegel, H.P., 2014. Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection. Data mining and knowledge discovery 28, 190–237. doi:https://doi.org/10.1007/s10618-012-0300-z.

[35] Siddiqi, N., 2012. Credit risk scorecards: developing and implementing intelligent credit scoring. volume 3. John Wiley & Sons. doi:https://doi.org/10.1002/9781119282396.

[36] Simon, C., 2015. Generating uniformly distributed numbers on a sphere. http://corysimon.github.io/articles/uniformdistn-on-sphere/. Last checked on Apr 04, 2022.

[37] Sinha, A.P., May, J.H., 2004. Evaluating and tuning predictive data mining models using receiver operating characteristic curves. Journal of Management Information Systems 21, 249–280. doi:https://doi.org/10.1080/07421222.2004.11045815.

[38] Society, I.C.I., 2019. IEEE-CIS fraud detection. https://www.kaggle.com/competitions/ieee-fraud-detection/overview. Retrieved on Apr 12, 2022.

[39] Tao, X., Zheng, Y., Chen, W., Zhang, X., Qi, L., Fan, Z., Huang, S., 2022. Svdd-based weighted oversampling technique for imbalanced and overlapped dataset learning. Information Sciences 588, 13–51. doi:https://doi.org/10.1016/j.ins.2021.12.066.

[40] Tax, D.M., Duin, R.P., 2004. Support vector data description. Machine learning 54, 45–66. doi:https://doi.org/10.1023/B:MACH.0000008084.60811.49.

[41] Van Vlasselaer, V., Bravo, C., Caelen, O., Eliassi-Rad, T., Akoglu, L., Snoeck, M., Baesens, B., 2015. Apate: A novel approach for automated credit card transaction fraud detection using network-based extensions. Decision Support Systems 75, 38–48. doi:https://doi.org/10.1016/j.dss.2015.04.013.

[42] Veganzones, D., Séverin, E., 2018. An investigation of bankruptcy prediction in imbalanced datasets. Decision Support Systems 112, 111–124. doi:https://doi.org/10.1016/j.dss.2018.06.011.

[43] Vernekar, S., Gaurav, A., Abdelzad, V., Denouden, T., Salay, R., Czarnecki, K., 2019. Out-of-distribution detection in classifiers via generation. arXiv preprint arXiv:1910.04241 doi:https://doi.org/10.48550/arXiv.1910.04241.

[44] VUB Data Analytics Laboratory, 2020. Cost-sensitive learning for profit-driven credit-scoring. https://github.com/vub-dl/data-csl-pdcs. Retrieved on Apr 11, 2022.

[45] Weisstein, E.W., 2002. Hypersphere. https://mathworld.wolfram.com/.

[46] Yang, B., Julia, Y., 2019. Jarfraud fraud detection. https://github.com/JarFraud/FraudDetection. Retrieved on Apr 11, 2022.

[47] Yang, J., Zhou, K., Li, Y., Liu, Z., 2021. Generalized out-of-distribution detection: A survey. arXiv preprint arXiv:2110.11334 doi:https://doi.org/10.48550/arXiv.2110.11334.

[48] Yu, Y., Qu, W.Y., Li, N., Guo, Z., 2017. Open-category classification by adversarial sample generation. arXiv preprint arXiv:1705.08722 doi:https://doi.org/10.48550/arXiv.1705.08722.

[49] Zhu, B., Baesens, B., vanden Broucke, S.K., 2017. An empirical comparison of techniques for the class imbalance problem in churn prediction. Information sciences 408, 84–99. doi:https://doi.org/10.1016/j.ins.2017.04.015.

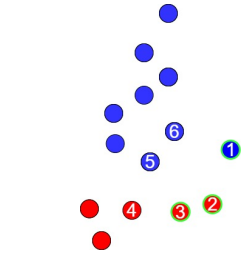
[50] Zięba, M., Tomczak, S.K., Tomczak, J.M., 2016. Ensemble boosted trees with synthetic features generation in application to bankruptcy prediction. Expert systems with applications 58, 93–101. doi:https://doi.org/10.1016/j.eswa.2016.04.001.

10. Appendix

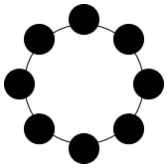
10.1. The BROOD Sampler

10.1.1. Visualization of the BROOD Sampler's Steps

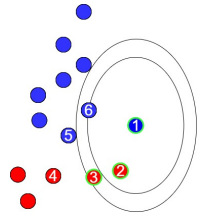
Figure 8 visualizes the different steps of the BROOD sampler on an artificial data set. In this example, $n_{\text{dir\_max}}$ is equal to 8, and we will sample from data points 1, 2, and 3.



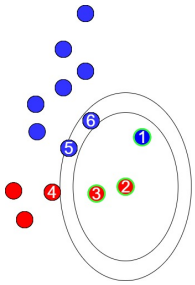
Step 1: Determine the data points from which we will sample. These data points have a green border in the plot.



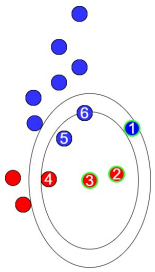
Step 2: We employ Mitchell's Best-Candidate algorithm to ensure that the sampled points are approximately evenly spaced on the surface.



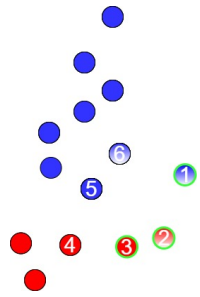
Step 3: We determine the neighborhoods of data point 1. The distance to point $(2, 6)$ is smaller than $2 \cdot \text{dist}_{\text{id\_ood}}$. The distance to the data point 3 is smaller than $2 \cdot \text{dist}_{\text{id\_ood}} + \text{dist}_{\text{ood}}$.



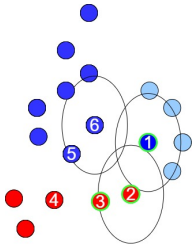
Step 3: We determine the neighborhoods of data point 2. The distance to the data points $(1, 3)$ is less than $2 \cdot \text{dist}_{\text{id\_ood}}$. The distance to points $(5, 6)$ is less than $2 \cdot \text{dist}_{\text{id\_ood}} + \text{dist}_{\text{ood}}$.



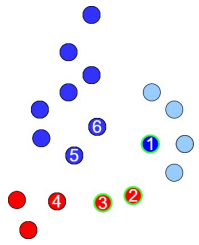
Step 3: We determine the neighborhoods of data point 3. The distance to the data points $(2, 4, 5, 6)$ is less than $2 \cdot \text{dist}_{\text{id\_ood}}$. The distance to point 1 is less than $2 \cdot \text{dist}_{\text{id\_ood}} + \text{dist}_{\text{ood}}$.



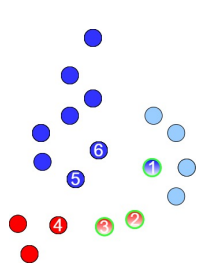
Step 4: We found that the data points $(2, 6)$ have a distance less than $2 \cdot \text{dist}_{\text{id\_ood}}$ from data point 1. We extract their data attributes.



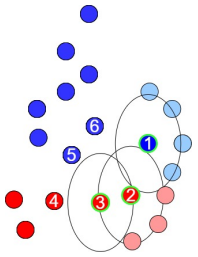
Step 5: We create artificial data points by sampling from data point 1. We keep the artificial data point if the distance from the data points $(2, 6)$ to the artificial data point is greater than $\text{dist}_{\text{id\_ood}}$.



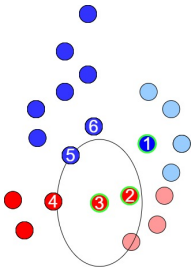
Step 6: We did not find any data points for this step.



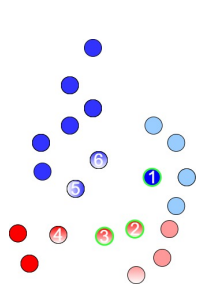
Step 4: We found that the data points $(1, 3)$ have a distance less than $2 \cdot \text{dist}_{\text{id\_ood}}$ from data point 2. We found that data point 3 has a distance less than $2 \cdot \text{dist}_{\text{id\_ood}} + \text{dist}_{\text{ood}}$, and we will sample from this data point in the future. We extract their data attributes.



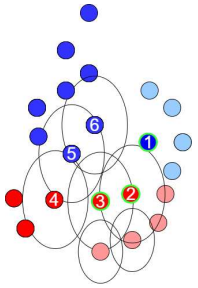
Step 5: We create artificial data points by sampling from data point 2. We keep the artificial data point if the distance from the data points $(1, 3)$ to the artificial data point is greater than $\text{dist}_{\text{id\_ood}}$.



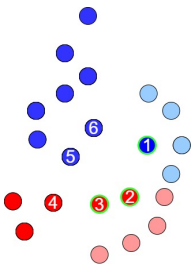
Step 6: The distance from data point 2 to data point 3 (same class, and we will sample from it in the future) is less than $2 \cdot \text{dist}_{\text{id\_ood}} + \text{dist}_{\text{ood}}$. The distance from data point 3 to one artificial data point is less than $\text{dist}_{\text{id\_ood}} + \text{dist}_{\text{ood}}$.



Step 4: We found that the data points $(2, 4, 5, 6)$ have a distance less than $2 \cdot \text{dist}_{\text{id\_ood}}$ from data point 3. Further, we found that the distance to one OOD sample is less than $\text{dist}_{\text{id\_ood}} + \text{dist}_{\text{ood}}$. We extract their data attributes.



Step 5: We create artificial data points by sampling from data point 3. We keep the artificial data point if the distance from the data points $(2, 4, 5, 6)$ to the artificial data point is greater than $\text{dist}_{\text{id\_ood}}$ and if the distance from the concerned artificial data point sampled from point 2 is less than dist_{ood}.



Step 6: We did not find any data points for this step.

Figure 8: Visualisation of the BROOD sampler's steps. The author created this toy example that visualises the BROOD sampler's working.

10.1.2. Modifications

The Python code of the BROOD sampler can be found on Github[5]. We wrote a modification of the BROOD sampler that checks the class distribution of the ID data points with Mahalanobis smaller than one from the ID point from which we sample. In this case, we compare the local class distribution and the global class prior. The OOD samples receive the label for which the quotient between the local class distribution and the global class prior is the greatest.

Next, in the case that $\hat{\boldsymbol{h}}$ is class or instance dependent, the Euclidean distance between the OOD samples that are sampled from different points can differ. Hence, we propose a method that ensures that the Euclidean distance between the OOD samples remains the same. Focusing on BROOD(KNN), the surface area S_d^r of a d-dimensional hypersphere with radius r is given by [45]

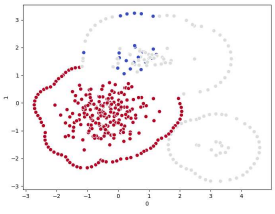
[5] https://github.com/CFLDT/Regularization-Oversampling-for-Classification-Tasks-To-Exploit-What-You-Do-Not-Know

$$S_d^r = \frac{2\pi^{d/2}}{\Gamma(d/2)} r^{d-1}. \tag{16}$$

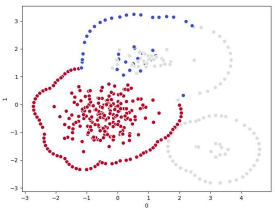
Denote with $S_d^{r\_\max}$ the surface of the hypersphere on which we sample directions from the instance with the highest d_{j_k}. We ensure that the number of directions that are created from an instance i equals

$$\frac{S_d^{r\_i}}{S_d^{\max\_r}} n_{\text{dir\_max}}. \tag{17}$$

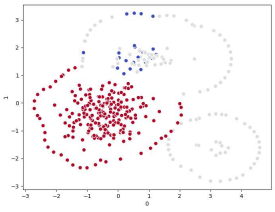
In Figure 9, one finds a visualization of the two modifications. The BROOD(KNN) sampler is found on the top left. In this case, the artificially generated samples have the same label as the instance from which they are sampled. The local counterpart considers the labels of data points in the neighborhood of the data point from which they are sampled. The bottom pictures visualize the case where we use the BROOD(KNN) sampler with more equal distances between OOD samples.



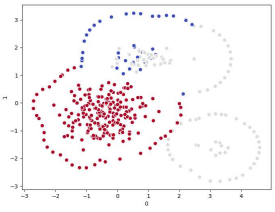
Visual of the standard BROOD(KNN) sampler.



Visual of local BROOD(KNN) sampler.



Visual of the standard BROOD(KNN) sampler with more equal distance between OOD samples.



Visual of local BROOD(KNN) sampler with more equal distance between OOD samples.

Figure 9: Visual comparison of the BROOD sampler's modifications. Comparison of the BROOD sampler's modifications in a multi-class setting on a toy data set.

10.2. General Overview of the Pipeline

As explained in the paper, we use random stratified sampling to obtain an initial training set consisting of 75% of all instances and a test set consisting of 25% of all instances. Furthermore, we split the initial training set (in a stratified way) into the final training set (50% of all instances) and a validation set (25% of all instances). We train the model on the final training set, use the validation set to optimize the models' hyperparameters, and assess performance on the test set. We execute this procedure four times in a row for every data set, each time with a completely different test set. A visual representation of the procedure is presented in Figure 10.

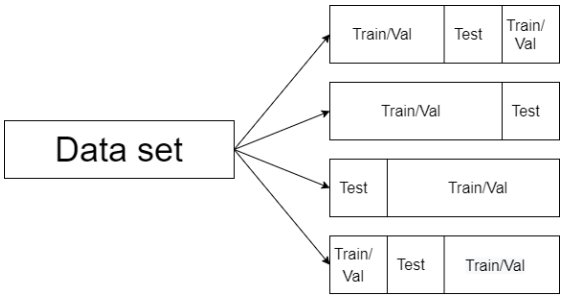


Figure 10: Visual representation of our cross-validation structure. We create four folds, each time with a completely different test set.

We apply a clever pipeline structure to ensure that the machine learning models can extract a maximal amount of information from the provided data sets. First, we create an additional dummy column for every numerical feature. If the numerical feature contains NaN values, the dummy column equals true at these positions. Further, in the numerical column, the NaN values are imputed by the median of the corresponding column. We treat the NaN values in categorical columns as an additional class.

Next, we use weight of evidence (WOE) encoding to quantify the predictive power of each attribute [35]). The WOE value for attribute a in column c is given by

$$\text{WOE}_a^c = \log\left(\frac{((\text{ratio non-event in } a) + 0.5)/(\text{ratio non-event in } c)}{((\text{ratio event in } a) + 0.5)/(\text{ratio event in } c)}\right) \tag{18}$$

where we add 0.5 to handle missing events. The information value (IV) for column c is given by

$$\text{IV}^c = \sum_a \Big(\frac{((\text{ratio non-event in } a) + 0.5)}{(\text{ratio non-event in } c)} - \frac{((\text{ratio event in } a) + 0.5)}{(\text{ratio event in } c)}\Big)\text{WOE}_a^c. \tag{19}$$

From [35], if an attribute has an IV value lower than 0.10, it is considered weak. To calculate the IV value for numerical features, we bin the numerical feature into 20 bins that contain approximately the same number of instances. We only keep informative features by deleting features with an IV value lower than 0.10. Furthermore, if the number of features is greater than 30, we only keep the 30 most informative features with the highest IV value. Of course, since WOE encodings require label information, these encodings are fitted on the training set. Unseen categorical attributes in the test or validation set have a WOE value of 0.

Subsequently, we standardize all features of the training set, and we apply the same scaler to the validation and test sets. We apply the BROOD sampler, with predetermined parameters, to the training set.

Finally, we iterate over every possible combination of hyperparameters for logistic regression and XGBoost models with

and without artificially generated OOD samples. We test the performance of a model with a specific combination of hyperparameters on the validation set. For instance, the ROC-AUC can be compared for every combination of hyperparameters. We use the best model to predict the test set's instances and calculate the corresponding performance measures.

10.3. Overview of the Results on the Benchmark Data Sets

We use a validation set to select an optimal combination of the set of hyperparameters. The set of hyperparameters can be found in Table 5. The performance metrics of our models on the benchmark data sets are presented in Table 6. These values were used to create Figure 4.

10.4. Overview of the Results on the Severely Imbalanced Data Sets

We use a validation set to select an optimal combination of the set of hyperparameters. The set of hyperparameters can be found in Table 7. The performance metrics of our models on the severely imbalanced data sets are presented in Table 10. These values were used to create Figures 5 and 6.

	Logit	XGBoost	XGBoost_BROOD
β_t	/	/	[1]
c_{ood_1}	/	/	[0, 0.50, 1, 2, 10]
c_{ood_0}	/	/	[0, 0.50, 1]
L2-Regularisation	[0, 0.10, 1, 10]	[0, 10, 100]	[0, 10, 100]
Max Tree Depth	/	[5, 10]	[5, 10]
# Estimators	/	[50, 200]	[50, 200]
Subsample	/	[0.25, 0.50, 0.75]	[0.25, 0.50, 0.75]
Learning Rate	[0.10]	[0.10]	[0.10]

Table 5: Lists of hyperparameters of the machine learning models that are trained on the benchmark data sets. We iterate over these lists during cross-validation to find the optimal combination.

	Logit	XGBoost	XGB_BR(ALL_ROT)	XGB_BR(ALL_KNN)
UCI E.coli	0.932	0.953	**0.967**	0.959
UCI Mammographic Masses	**0.870**	0.865	0.854	0.857
UCI Sonar	0.776	0.859	**0.912**	0.878
UCI Ionosphere	0.899	0.969	0.976	**0.984**
UCI Breast Cancer Wisconsin	**0.993**	0.991	0.993	0.987
UCI Heart	0.893	0.881	0.886	**0.895**
UCI Indian Liver Patient	0.749	0.731	**0.753**	0.746
UCI Liver Disorders	**0.673**	0.593	0.644	0.627
UCI Breast Cancer	**0.692**	0.653	0.655	0.684
UCI Vehicle	0.988	**0.995**	0.985	0.992
UCI Contraceptive method	0.732	**0.740**	0.737	0.739
UCI German Credit Card	0.778	**0.785**	0.779	0.765
UCI Credit Approval	0.928	0.937	0.935	**0.938**
UCI Bank Marketing	0.934	0.938	0.938	**0.940**
UCI Madelon	0.629	**0.857**	0.835	0.850

ROC-AUC of the machine learning models that are trained on the benchmark data sets.

	Logit	XGBoost	XGB_BR(ALL_ROT)	XGB_BR(ALL_KNN)
UCI E.coli	0.798	0.851	**0.863**	0.852
UCI Mammographic Masses	**0.831**	0.810	0.807	0.808
UCI Sonar	0.772	0.868	**0.920**	0.885
UCI Ionosphere	0.890	0.961	0.970	**0.976**
UCI Breast Cancer Wisconsin	0.991	0.989	**0.991**	0.987
UCI Heart	0.885	0.875	0.878	**0.888**
UCI Indian Liver Patient	0.495	0.483	0.521	**0.526**
UCI Liver Disorders	**0.505**	0.443	0.473	0.492
UCI Breast Cancer	0.489	0.443	0.451	**0.491**
UCI Vehicle	0.961	**0.984**	0.971	0.974
UCI Contraceptive method	0.434	0.467	0.469	**0.476**
UCI German Credit Card	0.585	0.591	**0.607**	0.596
UCI Credit Approval	0.914	0.921	0.916	**0.924**
UCI Bank Marketing	0.609	0.638	0.635	**0.647**
UCI Madelon	0.579	**0.849**	0.825	0.848

AP of the machine learning models that are trained on the benchmark data sets.

	Logit	XGBoost	XGB_BR(MIN_ROT)	XGB_BR(MIN_KNN)
UCI E.coli	4.087	4.408	**4.529**	4.504
UCI Mammographic Masses	**18.248**	18.239	18.070	18.041
UCI Sonar	6.496	7.228	**7.607**	7.429
UCI Ionosphere	8.067	8.787	8.794	**8.896**
UCI Breast Cancer Wisconsin	12.939	12.894	12.983	**13.025**
UCI Heart	7.746	7.489	**7.792**	7.761
UCI Indian Liver Patient	5.299	5.458	6.062	**6.133**
UCI Liver Disorders	**4.487**	3.823	4.046	4.336
UCI Breast Cancer	3.511	3.137	3.218	**3.566**
UCI Vehicle	12.075	**12.256**	12.237	12.060
UCI Contraceptive method	8.637	**9.379**	9.283	9.246
UCI German Credit Card	10.353	10.552	10.422	**10.565**
UCI Credit Approval	15.231	**15.398**	15.277	15.314
UCI Bank Marketing	14.209	14.674	14.866	**14.873**
UCI Madelon	29.449	**40.607**	40.011	40.394

DCG of the machine learning models that are trained on the benchmark data sets.

Table 6: Performance metrics of the machine learning models on the benchmark data sets. We use BROOD(ALL_ROT) ($n_{\text{dir\_max}}$ = 30, $n_{\text{dir\_samp}}$ = 50, $\text{dist}_{\text{id\_ood}}$ = 4) and BROOD(ALL_KNN) ($n_{\text{dir\_max}}$ = 30, $n_{\text{dir\_samp}}$ = 50, $\text{dist}_{\text{id\_ood}}$ = 1.5) to generate BROOD samples. The best-performing model is in bold. The best-performing model is in bold. We abbreviate 'XGBoost' as ' XGB' and 'BROOD' as 'BR'.

	Logit	XGBoost	XGBoost_BROOD
β_t	/	/	[1]
c_{ood_1}	/	/	[0, 1, 2, 5, 10, 100, 1,000]
c_{ood_0}	/	/	/
L2-Regularisation	[0, 0.10, 1, 10]	[0, 10, 100]	[0, 10, 100]
Max Tree Depth	/	[5, 10]	[5, 10]
# Estimators	/	[50, 200]	[50, 200]
Subsample	/	[0.25, 0.50, 0.75]	[0.25, 0.50, 0.75]
Learning Rate	[0.10]	[0.10]	[0.10]

Table 7: Lists of hyperparameters of the machine learning models that are trained on the severely imbalanced data sets. We iterate over these lists during cross-validation to find the optimal combination.

	Logit	XGBoost	XGBoost_SMOTE	XGBoost_ADASYN	XGBoost_ROSE
ESA Polish Bankruptcy	0.852	0.892	0.906	0.902	0.900
Financial Distress	0.913	0.912	0.922	0.920	0.911
UCI German Credit Card IMB	0.774	**0.805**	0.773	0.763	0.783
UCI Credit Approval IMB	0.908	0.930	0.915	0.941	0.936
UCI Bank Marketing IMB	0.897	0.917	0.925	**0.926**	0.910
IBM Telco Customer Churn	0.817	0.591	0.629	0.638	0.642
ETL Credit Card Approval	0.618	0.597	0.592	0.586	0.635
VUB Credit Scoring	0.775	**0.784**	0.753	0.759	0.763
KDD Cup 1998	0.528	0.517	0.526	0.532	0.519
(Priv.) Car Insurance Fraud	0.677	0.708	0.707	0.717	0.718
(Priv.) APATA Credit Card Fraud	0.810	0.500	0.500	0.500	0.500
JAR Financial Statement Fraud	0.708	0.728	0.718	0.716	0.714
IEEE-CIS Cus. Trans. Fraud	0.768	0.828	0.770	0.753	0.783
Synt. Mobile Trans. Fraud	0.948	0.993	0.990	0.984	0.993
Synthetic Credit Card Fraud	0.980	0.981	0.981	**0.987**	0.983

	XGB_BR(MIN_ROT)	XGB_BR(MIN_KNN)	XGB_ROSE_BR(MIN_ROT)	XGB_ROSE_BR(MIN_KNN)
ESA Polish Bankruptcy	0.895	**0.908**	0.891	0.897
Financial Distress	0.915	0.915	0.921	**0.931**
UCI German Credit Card IMB	0.766	0.764	0.734	0.782
UCI Credit Approval IMB	0.928	**0.947**	0.929	0.945
UCI Bank Marketing IMB	0.912	0.923	0.911	0.918
IBM Telco Customer Churn	**0.820**	0.820	0.814	0.819
ETL Credit Card Approval	0.599	0.581	0.589	**0.629**
VUB Credit Scoring	0.777	0.776	0.773	0.771
KDD Cup 1998	0.536	0.527	0.533	**0.541**
(Priv.) Car Insurance Fraud	0.699	0.690	**0.761**	0.751
(Priv.) APATA Credit Card Fr.	**0.853**	0.708	0.655	0.808
JAR Financial Statement Fr.	0.719	0.730	0.729	**0.737**
IEEE-CIS Cus. Trans. Fraud	0.825	**0.839**	0.778	0.769
Synth. Mobile Trans. Fr.	**0.992**	0.989	0.989	0.991
Synthetic Credit Card Fraud	0.983	0.985	0.982	0.985

ROC-AUC of the machine learning models that are trained on the severely imbalanced data sets.

	Logit	XGBoost	XGBoost_SMOTE	XGBoost_ADASYN	XGBoost_ROSE
ESA Polish Bankruptcy	0.476	0.630	0.632	0.619	0.618
Financial Distress	0.313	**0.400**	0.384	0.366	0.389
UCI German Credit Card IMB	0.271	0.299	0.286	0.282	**0.321**
UCI Credit Approval IMB	0.561	0.571	0.576	**0.629**	0.616
UCI Bank Marketing IMB	0.239	**0.294**	0.278	0.268	0.292
IBM Telco Customer Churn	0.226	0.110	0.126	0.137	0.132
ETL Credit Card Approval	0.053	0.041	0.022	0.030	0.052
VUB Credit Scoring	0.129	0.127	0.121	0.128	0.125
KDD Cup 1998	0.059	0.058	0.058	**0.061**	0.055
(Priv.) Car Insurance Fraud	0.068	0.064	0.069	0.075	0.073
(Priv.) APATA Credit Card Fraud	0.052	0.008	0.008	0.008	0.008
JAR Financial Statement Fraud	0.072	0.084	0.079	0.073	0.071
IEEE-CIS Cus. Trans. Fraud	0.289	0.322	0.237	0.245	0.310
Synt. Mobile Trans. Fraud	0.678	0.880	0.856	0.881	0.898
Synthetic Credit Card Fraud	0.797	0.839	0.845	0.853	0.866

	XGB_BR(MIN_ROT)	XGB_BR(MIN_KNN)	XGB_ROSE_BR(MIN_ROT)	XGB_ROSE_BR(MIN_KNN)
ESA Polish Bankruptcy	0.623	**0.648**	0.622	0.629
Financial Distress	0.392	0.383	0.400	0.388
UCI German Credit Card IMB	0.275	0.282	0.272	0.273
UCI Credit Approval IMB	0.608	0.609	0.559	0.613
UCI Bank Marketing IMB	0.211	0.265	0.248	0.275
IBM Telco Customer Churn	0.232	**0.233**	0.228	0.228
ETL Credit Card Approval	0.051	0.037	0.050	**0.078**
VUB Credit Scoring	0.124	0.126	0.123	**0.131**
KDD Cup 1998	0.058	0.057	0.058	0.060
(Priv.) Car Insurance Fraud	0.087	0.074	**0.111**	0.086
(Priv.) APATA Credit Card Fr.	**0.256**	0.178	0.147	0.171
JAR Financial Statement Fr.	0.084	0.083	0.095	**0.096**
IEEE-CIS Cus. Trans. Fraud	0.325	**0.343**	0.297	0.298
Synth. Mobile Trans. Fr.	0.878	0.890	0.907	**0.910**
Synthetic Credit Card Fraud	**0.884**	0.875	0.880	0.869

AP of the machine learning models that are trained on the severely imbalanced data sets.

	Logit	XGBoost	XGBoost_SMOTE	XGBoost_ADASYN	XGBoost_ROSE
ESA Polish Bankruptcy	9.312	10.788	10.803	10.795	10.796
Financial Distress	3.363	4.480	4.497	4.174	4.404
UCI German Credit Card IMB	1.679	2.028	2.049	1.779	**2.104**
UCI Credit Approval IMB	2.097	1.966	2.128	2.126	**2.232**
UCI Bank Marketing IMB	2.039	2.605	2.486	2.486	**2.779**
IBM Telco Customer Churn	5.293	4.005	4.035	4.589	3.654
ETL Credit Card Approval	0.408	0.520	0.322	0.398	0.733
VUB Credit Scoring	5.431	4.992	5.140	5.228	5.435
KDD Cup 1998	2.104	1.984	2.003	**2.221**	1.705
(Priv.) Car Insurance Fraud	2.541	2.288	2.400	2.885	2.527
(Priv.) APATA Credit Card Fraud	0.000	0.000	0.000	0.000	0.000
JAR Financial Statement Fraud	2.023	2.703	2.384	2.210	2.329
IEEE-CIS Cus. Trans. Fraud	8.463	9.191	7.698	7.939	8.889
Synth. Mobile Trans. Fraud	5.282	6.519	6.610	6.693	6.720
Synthetic Credit Card Fraud	7.600	7.628	7.644	7.803	7.964

	XGB_BR(MIN_ROT)	XGB_BR(MIN_KNN)	XGB_ROSE_BR(MIN_ROT)	XGB_ROSE_BR(MIN_KNN)
ESA Polish Bankruptcy	10.851	10.936	10.846	**10.989**
Financial Distress	4.562	4.464	**4.696**	4.255
UCI German Credit Card IMB	1.774	1.707	1.906	1.830
UCI Credit Approval IMB	2.151	2.072	2.049	2.072
UCI Bank Marketing IMB	1.902	2.241	2.321	2.592
IBM Telco Customer Churn	5.344	5.359	**5.674**	5.662
ETL Credit Card Approval	0.590	0.465	0.677	**0.955**
VUB Credit Scoring	5.050	5.116	5.109	**5.771**
KDD Cup 1998	1.949	1.698	2.218	1.967
(Priv.) Car Insurance Fraud	3.217	2.764	**4.002**	2.911
(Priv.) APATA Credit Card Fr.	**2.086**	1.572	1.562	1.580
JAR Financial Statement Fr.	3.073	2.842	**3.258**	3.130
IEEE-CIS Cus. Trans. Fraud	9.290	**9.560**	8.888	8.971
Synth. Mobile Trans. Fr.	6.720	6.705	6.670	**6.794**
Synthetic Credit Card Fraud	8.104	8.034	**8.155**	8.019

DCG of the machine learning models that are trained on the severely imbalanced data sets.

Table 10: Performance metrics of the machine learning models that are trained on the severely imbalanced data sets. We use BROOD(MIN_ROT) ($n_{\text{dir\_max}}$ = 30, $n_{\text{dir\_samp}}$ = 50, $\text{dist}_{\text{id\_ood}}$ = 4) and BROOD(MIN_KNN) ($n_{\text{dir\_max}}$ = 30, $n_{\text{dir\_samp}}$ = 50, $\text{dist}_{\text{id\_ood}}$ = 1.5) to generate BROOD samples. We use the targeted imbalance ratio that on average performs best on all data sets. The stand alone SMOTE, ADASYN, and ROSE samplers aim to achieve an imbalance ratio of 0.20. The ROSE sampler along with the BROOD sampler aim to achieve an imbalance ratio of 0.10. The best-performing model is in bold. We abbreviate 'XGBoost' as ' XGB' and 'BROOD' as 'BR'.