Analysis of sampling strategies to optimize classification of heavily skewed datasets

Nathaniel Johnson
School of Computer Science
University of Guelph
Guelph, Canada
njohns18@uoguelph.ca

Megan Tibbles
School of Computer Science
University of Guelph
Guelph, Canada
mtibbles@uoguelph.ca

***Abstract*—Class imbalance poses a significant challenge for machine learning algorithms, impacting their learning effectiveness. This paper compares four sampling strategies commonly used to address this issue; Synthetic Minority Over-Sampling Technique (SMOTE), Generative Adversarial Networks (GAN), nearest neighbor clustering, and random undersampling, and introduces a novel Naive Bayes oversampling technique. Two stable classifiers are used to evaluate these strategies: Random Forest (RF) classifier and a Linear Support Vector Machine (LSVM). The experiments are conducted on the Covertype dataset and synthetically generated datasets with varying noise levels. SMOTE, random undersampling, and nearest neighbor clustering were the most effective strategies to improve the recall. The novel Naive Bayes strategy was in fact a hindrance to model performance, but when used with the generated noisy data, we found this strategy was uniquely able to mitigate the reduction in performance caused by noise for some classifiers. This paper advances our understanding of sampling strategies, guiding the selection of appropriate techniques for balancing datasets.**

I. Introduction

Generally, machine learning algorithms learn best when the distribution of classes in data is approximately balanced. Unfortunately, in many fields such as fraud detection, diagnosis prediction, and military applications[], the distribution of events is not balanced, but rather is heavily skewed in favor of one of the labels. This overrepresentation of a particular class can make it difficult to train a machine learning model, as the model learns that favoring a particular class minimizes its loss function. This bias gets worse as the imbalance in the labels increases [11]. This is known as the class imbalance problem.

Accuracy, which is the proportion of correctly classified observations, is a metric frequently used to evaluate the performance of a machine learning model. When evaluating a model trained on very skewed data, accuracy can be misleading [11]. Consider a situation where we train a model using a dataset where there is one instance of the positive class for every 99 instances of the negative class. A classifier which only labels observations as being from the negative class would have an accuracy of 99%, but would not be of any use. Instead, metrics such as ROC and F-score are used to assess a classifier's ability to correctly classify observations from both the majority and minority classes [6].

There are many approaches to solving the class imbalance problem, each with their own benefits and drawbacks. One common approach is to transform the training data, either by reducing the number of observations in the majority class (undersampling) or by artificially increasing the number of observations in the minority class (oversampling) [9]. Oversampling has the benefit of not losing information from the majority class, but risks increasing the likelihood of overfitting the model as it introduces copies of the minority data [9], [16]. Undersampling on the other hand, does not risk introducing non-existent patterns into the minority class and can improve the training speed of classifiers, but results in a loss of information in the majority class [9], [16]. There is no consensus whether undersampling or oversampling is better. Lin, Wei-Chao, et al. [9] claims that undersampling is the better strategy, whereas Zhihao, Peng, et al. [16] claims that oversampling is better.

The choice of classifier also plays a role creating an effective model using imbalance data. Ensemble classifiers are a collection of classifiers which work together to make a prediction. Ensemble methods can reduce the risk of overfitting that comes from oversampling, and are generally well equipped for learning from imbalanced data, as their ensemble nature results in a naturally low bias. Ensemble methods are also considered stable classifiers, as small changes to the data used to train them does not result in a significant change in their output. Using stable classifiers for making comparisons between sampling strategies will give us a better idea of the impact of a specific strategy, as small improvements in the quality of the training data as a result of applying the strategies will not be over represented by the performance of the classifier.

In this paper, we compare five sampling strategies, both undersampling and oversampling, by measuring the improvement in F-score of stable classifiers trained on data before and after it has been balanced using that sampling strategy. The five strategies are: SMOTE, undersampling, K-neighbor clustering, Generative Adversarial Network, and a novel Naive-Bayes sampling strategy. The objective of the study is to determine if there exists an optimal strategy for imbalanced classification. Particularly, our project has the following sub-objectives:

Determine if the Naive-Bayes strategy can out-perform the other strategies Determine if the performance of a strategy changes based on the classifier Investigate how strategy per-

formance changes as noise is introduced to the data

We compare the five strategies using two stable classifiers, one an ensemble classifier, and the other a single classifier.

II. Methedology

A. Tools & Libraries

Data preprocessing, implementation of the sampling strategies, training and testing of the Random Forest and Support Vector Machine classifiers, and creation of the graphs and plots were accomplished using Python 3.11.5, numpy v1.24.3, Pandas v2.1.1, Scikit-learn v1.3.0, torchvision v0.15.2, matplotlib v3.8.0, SciPy v1.11.1, and seaborn v0.13.0.

III. Data

The Covertype dataset is available from the UCI repository, and was selected due to its use in Chawla et al. [4] to investigate SMOTE. The dataset contains 581,012 instances, with 52 features, both categorical and numerical, including elevation, wilderness area, and elevation. There are 7 classes, each representing a different type of forest, but for our purposes we will only be using two; the Ponderosa Pine and Cottonwood/Willow labels. The former has 35,754 samples, and the latter just 2,747 samples. This selection is motivated by Chawla et al. [4], who used the same two labels in their study due the imbalance between them. We then applied feature reduction, removing wilderness and soil type columns where all values were 0.

For evaluating the performance of the balancing strategies with respect to noise, we combined samples drawn from two Gaussian distributions to generate several datasets. The Gaussian distributions were generated using the same covariance matrix, but have different means. Each dataset has two classes and 10,000 points, but differs in the amount of skew in the labels, and the amount of noise present. The amount of skew is represented as a ratio of majority to minority, with levels 70:30, 80:20, and 90:10. The amount of noise in the labels is the percentage of perturbed labels, with levels 0%, 5%, 10%, and 20%. Each sample has 3 features, x, y, and z, representing a position in 3-dimensional space, and a label representing the distribution from which the sample was taken. Noise was added by randomly flipping a controlled number of labels to assign the sample the opposite class.

A. Strategies

a) Synthetic Minority Over-Sampling Technique: SMOTE: SMOTE generates new data samples for the minority class based on the existing data. The algorithm finds the 5 nearest neighbours of a data point and randomly selects N of these neighbors to form the basis for the synthetic data point. The value N is based on the number of samples needed to balance the dataset. For example, if the majority class has 20 samples and the minority class has 10 samples. We divide the size of the majority class by the size of the minority class to get 2. Meaning we select 2 of the nearest neighbours to make these additional points. To generate a new sample, we calculate the difference between each feature of the original data point and its nearest neighbor. Then multiply it by a random number between 0 and 1. This creates a new data point that has slightly different features. This algorithm is further outlined by Chawla et al [4].

Generative Adversarial Networks: GANs consist of two neural networks: the generator and the discriminator. The generator creates novel data points based on the minority class data. The discriminator tries to distinguish between the newly generated points and the original dataset. Through an iterative process, these networks train each other. We chose to use 200 epochs and a learning rate of 0.0001 because these parameters seemed to produce the best results. Once the generator has been trained, we use it to augment the minority class data, balancing the overall dataset. Some of our choices for activation functions, loss functions, and optimizers were based on the paper by Tanaka et al. [5].

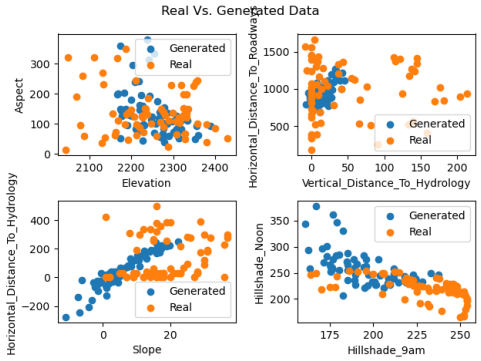

Fig. 1. Synthetic vs. Real data generated by GAN strategy

Nearest Neighbor Clustering: We used the nearest neighbor undersampling strategy described by Lin, Wei-Chao, et al. [9], which was shown to be superior to a strategy where cluster centers were chosen to represent the majority class, rather than their nearest neighbor. Briefly, this strategy pairs the K-means algorithm with the samples from the majority class to place k centroids, where k is the number of samples in the minority class. The algorithm then selects the nearest neighbor to each centroid from the majority samples to use as the undersampled majority in the balanced dataset.

Random Undersampling: The random undersampling strategy selects m points at random from the samples of the majority class. Random undersampling is the most primitive form of undersampling, and while easy to implement, comes with the cost of potentially losing information from the majority class.

Naive Bayes Oversampling: The naive bayes sampling strategy is novel, and makes use of the naive bayes classifier. First, a naive bayes classifier is trained using the full dataset. Then a gaussian distribution is created using the mean and variance for each of the input features. Points are then randomly sampled from the distribution, and classified using the naive bayes model. If the classifier predicts the sample to be from the minority class with a confidence of 95% or greater, the sample

is added to the dataset. This is repeated until the number of samples between both classes are equal.

B. Classifiers

Random Forest Classifier: Random Forest (RF) is an ensemble classifier used for supervised learning. It is a popular method due to its easy implementation and high accuracy [12]. It constructs many individual decision trees that output a prediction. The classifier determines the result based on the class with the highest number of predictions[12]. We used the function from Scikit-learn to implement this classifier. We set the n_estimators parameter to 500, meaning our classifier uses 500 decision trees.

Linear Support Vector Machine: Linear Support Vector Machine (LSVM) is a classifier for supervised learning. This method is often chosen because it is more stable than methods like neural networks[7]. This method works by determining the hyperplane to separate the classes. It works to maximize the separation between these classes. Choosing a linear kernel means the goal is to find a linear hyperplane [7]. We used the LinearSVC function from Scikit-learn with the dual parameter set to auto.

C. Procedure

For the first part of the analysis, the Covertype dataset was partitioned into 5 training and testing sets using the 5-fold cross validation method. Each strategy was then used to balance each of the training folds, creating 25 unique, balanced training folds. A RF classifier and LSVM classifier were trained using the balanced folds, and tested against the original unbalanced testing fold. The F-score, recall, and precision of the classifiers were recorded after each training and testing fold, resulting in 5 values for F-score, recall, and precision per classifier for each of the strategies. The RF and LSVM were then trained using the original unsampled training folds, and had their F-Measure, recall, and precision record after each fold. The ΔF-score was computed for each of the strategies by subtracting the mean F-score for the strategy from the mean F-score from the unsampled classifiers.

The generated noise data was handled differently than the Covertype dataset because we created more than one dataset. We created different versions of the generated data with different majority to minority ratios and different levels of noise. In total we have 12 unique datasets. Since there were multiple sets, 5-fold cross validation would have resulted in a large amount of files that would have been difficult to manage. Instead, we split each dataset into training and testing sets with 30% going towards testing. This resulted in 12 training sets, and 12 testing sets. Each strategy was then used to balance each training set. A RF classifier and LSVM classifier were trained using each balanced training set. The classifiers were then evaluated using the test data with the precision, recall, and F-score being recorded for each. This resulted in 12 F-scores, precision scores, and recall scores for each strategy.

D. Evaluation Metrics

For our evaluation, we used 4 metrics; Precision, Recall, F-score, and ΔF-score. We define a positive event as an event coming from the minority class.

a) Precision: Precision is the ratio of the number of correctly identified positive values to the total number of *identified* positives. A high precision indicates the model is less likely to incorrectly classify a negative event as being positive.

b) Recall: Recall is the ratio of correctly identified positive values to the total number of true positives. A high recall indicates the model is able to correctly identify a majority of the positive instances. In data where the positive class is greatly outnumbered by the negative class, a high recall is important, as it means the model is not being biased by the imbalance of classes in the data.

c) Fscore: F-score is the harmonic mean of precision and recall, and used frequently throughout the literature for comparing model performance, especially when there is an uneven class distribution. [9]

d) ΔF-score: The ΔF-Score for a particular strategy and model is computed by taking the difference in F-Score between a model trained on unsampled data and the same model being trained on the same data that has been balanced using that strategy. A positive ΔF-score indicates that the model performed better when trained on the sampled than on the unsampled data. A negative ΔF-score indicates that the model performed worse when being trained on the sample data, and an F-score of 0 indicates that the model performed no different whether it was trained using the sampled data or the unsampled data.

IV. Results

Classifiers trained using the unbalanced training data were outperformed in precision by classifiers trained on data balanced by the NB strategy, which performed well below expectations overall. Classifiers trained using the unbalanced training data had a lower recall than when trained using SMOTE, clustering, and undersampling balanced data. The NB strategy performed quite low in recall. Only SMOTE had a positive ΔF-score.

The LSVM classifier had the highest precision when trained with the unsampled data. The SMOTE, clustering, and undersampling strategies greatly improved the recall score of both classifiers, compared to when trained using unbalanced training data. All strategies had a ΔF-score very close to zero, with only SMOTE and GAN having positive values.

Across the two classifiers, datasets balanced using SMOTE, clustering, and undersampling resulted in the greatest improvements to correctly classifying observations from the minority class relative to using unbalanced training data. No sampling strategy had a ΔF-score significantly different from zero, except for the Naive Bayes the classifier, which had a ΔF-score of -0.537.

Since we created multiple generated datasets with different levels of noise, we can examine these results in two ways. We

TABLE I
PERFORMANCE BY STRATEGY - RANDOM FOREST

Strategy	Metric			
	Precision	***Recall***	***F-score***	**Δ *F-score***
No Sample	0.942	0.863	0.901	0
SMOTE	0.918	0.907	0.912	0.011
GAN	0.324	0.916	0.479	-0.005
Neighbor	0.625	0.975	0.762	-0.139
Undersampling	0.616	0.978	0.756	-0.145
Naive Bayes	0.970	0.225	0.366	-0.535

TABLE II
PERFORMANCE BY STRATEGY - SVM

Strategy	Metric			
	Precision	***Recall***	***F-score***	**Δ *F-score***
No Sample	0.680	0.376	0.484	0
SMOTE	0.345	0.900	0.499	0.015
GAN	0.940	0.863	0.899	-0.001
Neighbor	0.318	0.917	0.472	-0.012
Undersampling	0.324	0.916	0.479	-0.005
Naive Bayes	0.598	0.357	0.447	-0.037

can examine the effects of noise on each strategy, comparing the metrics for the dataset with no or little noise to the metrics for the dataset with 20% noise. We can also look at how each of the strategies handled the noise in general, comparing overall metrics for each strategy.

To start, we will report the effects that the level of noise had on each strategy. As a baseline, we used RF and LSVM on the data without using any sampling technique. The precision, recall, and F-score were all close to 1 for the data with no noise. With 20% noise, F-Score drops to around 60% and recall drops to around 50% for both classifiers. Precision drops to 70% with random forest and 80% with LSVM. All sampling strategies followed the same pattern with precision, recall, and F-score declining as noise increases.

The ΔF-Score tended to decline as noise increased from 0% - 10%, and all strategies had negative values, indicating the sampling strategy did not improve the classifier results. However, the ΔF-Score for 20% noise data tended to be close to 0 for RF and slightly higher for LSVM. Undersampling, SMOTE, and clustering strategies had very small positive ΔF-Scores for LSVM, indicating the sampling strategy improved the classifier results. Most interestingly was the Naive Bayes strategy results. The ΔF-Score was lowest for the data with no noise and highest for the data with 20% noise, with each increase of noise level corresponding to an increase in ΔF-Score.

We can also report the overall results of each strategy. In terms of precision, none of the tested sampling strategies performed better than no sampling. As for recall, the results were consistent between strategies, no technique significantly out shined the others. F-Scores from the random forest classifier were all mostly even with naive bayes slightly lower than the rest. GAN had the highest ΔF-Score for RF classifier and naive bayes had the lowest. However, for LSVM, GAN had the lowest F-score, with a wider range than the rest and had the lowest ΔF-score.

V. DISCUSSION

We hoped that our novel NB strategy would be able to outperform the other sampling strategies discussed in this paper. Unfortunately, this was not the case. For the Random Forest classifier, although the NB strategy resulted in the best precision scores, the recall and F-scores were very low. For the LSVM, the precision score was similar to GAN and no sampling strategy in the 0.6-0.7 range. For both classifiers, the NB strategy produced a very low recall and F-score. Based on these findings, it appears that classifiers trained using the NB sampling strategy tend to perform worse overall than when trained using any of the other sampling strategies.

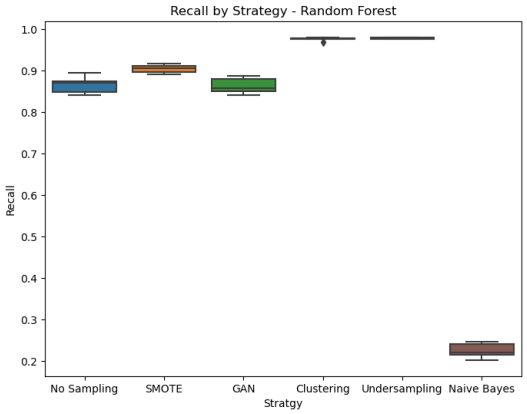

Fig. 2. Recall for all strategies with Covertype dataset with Random Forest Classifier

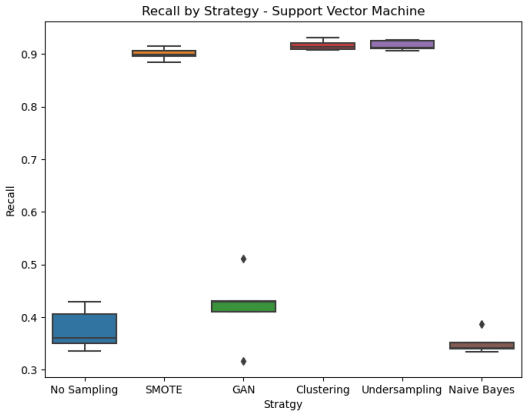

Fig. 3. Recall for all strategies with Covertype dataset with Support Vector Machine

There does not appear to be any interaction between strategy and classifier in the improvements in F-score, recall, and precision. The sampling strategies that most improved the recall for the RF classifier were the same which improved the recall for the SVM classifier. The improvement in recall scores were much greater for SVM, but we cannot say if this

is simply because there was more of an improvement to be made.

Noise is common in many datasets in real life. It is hard to avoid, much like unbalanced classes, and they often come hand in hand [14]. That is why it is important to see how noise affects the results of these sampling strategies. Artificially generating data with different levels of noise allows us to do this. The observed trend of performance declining as noise increases matches expectations. Tantithamthavorn et. al discuss the impact of noise on classification in their article [13]. They found that recall tends to be negatively impacted by noise. This was consistent with our results. However, they found that precision was rarely impacted by noise, whereas in our results, precision declined as noise level increased as well.

Since we have established that noise negatively impacts classifier results, it is important to identify strategies that mitigate these effects. In the literature, random undersampling has been determined as an effective balancing strategy with noisy data [14]. Our results showed that although random undersampling exhibited satisfactory performance, it did not surpass the alternatives. Undersampling was on par with the other sampling strategies having similar precision, recall, and f score.

One of the most intriguing metrics for analysis is the ΔF Score, which reveals the sampling strategy's performance in comparison to the raw data. While using the noise data, this metric informs us about the sampling strategy's impact on noise, whether it exacerbates it or skillfully manages its effects. As previously noted, we saw a pattern where the ΔF Score would decrease as the data changed from 0% noise to 10% noise, only to rise again at 20% noise. This indicates that the sampling strategy exacerbated the impact of noise on the data, resulting in performance deterioration until reaching a 20% noise level. Beyond this threshold, the sampling strategies appeared to yield comparable results to the raw data, occasionally even enhancing performance. This could indicate that some sampling strategies are useful when noise level reaches a certain threshold. Notably, unique results were observed with the Naive Bayes technique and the LSVM. The ΔF-score consistently increased with rising noise level, instead of showing that drop between 0-10%. Although the Δf score is still small at 20% noise, about 0.004, it is still positive. These findings are compelling, suggesting that the Naive Bayes strategy may mitigate the influence of noise. Our study only investigated the performance of the sampling strategies for two class problems. It would be very interesting to see how the strategies perform for different numbers of classes, but it would also lead to a lot more work, as the number of ways a dataset can be skewed increases as the number of classes increases. A future study may consider looking at variations of a 3-class skew. For example, one dataset where there is one majority with two balanced minorities, and another where there is one minority with two balanced majorities. It would be interesting to see which sampling strategies, or even combination of sampling strategies are the most effective for each type of skew.

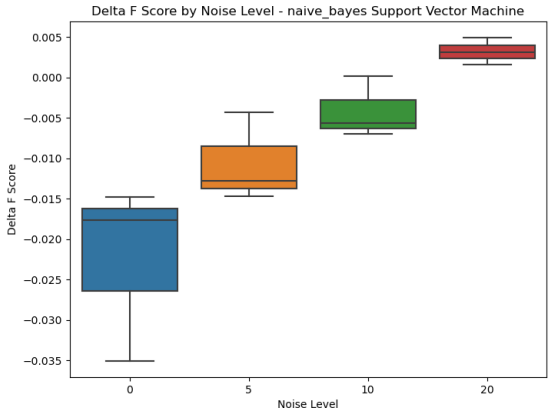

Fig. 4. ΔF Score Naive Bayes sampling strategy on generated data with LSVM

VI. Conclusion

This study compares the effectiveness of five sampling strategies, and introduces a novel Naive Bayes oversampling strategy. The first performance comparison between sampling strategies was performed using the Covertype dataset. SMOTE, random undersampling, and nearest neighbor clustering were found to be the most effective sampling strategies to improve recall relative to using the unbalanced data, but only SMOTE consistently improved the F-score of the classifier. The second performance comparison was performed using generated data, with controlled levels of skew and noise. While skew did not appear to affect the performance of the sampling strategies, performance became increasingly worse as the level of noise in the labels increased. The rate of performance decrease varied between sampling strategies, but for the RF classifier, none of the sampling strategies were able to reduce the rate of performance degradation of the classifier relative to the one trained using unbalanced data. Interestingly, for the LSVM, the NB sampling strategy was able to reduce the rate of model performance degradation as noise increased, eventually climbing to a ΔF-score greater than 0 as noise reached 20%. In future studies, it would be interesting to compare the strategies' performance on multiclass imbalances, as well as look at comparing the performance of combinations of strategies, where first an undersampling technique is applied to reduce the skew, then an oversampling technique is applied to fully balance the data.

References

[1] Aridas, Christos K., et al. "Uncertainty Based Under-Sampling for Learning Naive Bayes Classifiers under Imbalanced Data Sets." IEEE Access, vol. 8, 2020, pp. 2122–2133, doi:10.1109/access.2019.2961784.

[2] Batuwita, Rukshan, and Vasile Palade. "Class Imbalance Learning Methods for Support Vector Machines." Imbalanced Learning, 2013, pp. 83–99, doi:10.1002/9781118646106.ch5.

[3] Blackard, Jock. Covertype. 1998, doi:10.24432/C50K5N.

[4] Chawla, N. V., et al. "Smote: Synthetic Minority over-Sampling Technique." Journal of Artificial Intelligence Research, vol. 16, 2002, pp. 321–357, doi:10.1613/jair.953.

[5] dos Santos Tanaka, Fabio Henrique Kiyoiti, and Claus Aranha. 'Data Augmentation Using GANs'. CoRR, vol. abs/1904.09135, 2019, http://arxiv.org/abs/1904.09135.
[6] He, Haibo. "Introduction." Imbalanced Learning, 2013, pp. 1–12, doi:10.1002/9781118646106.ch1.
[7] Howley, Tom, and Michael G. Madden. "The Genetic Kernel Support Vector Machine: Description and Evaluation." Artificial Intelligence Review, vol. 24, no. 3–4, 2005, pp. 379–395, doi:10.1007/s10462-005-9009-3.
[8] Jeni, Laszlo A., et al. "Facing Imbalanced Data–Recommendations for the Use of Performance Metrics." 2013 Humaine Association Conference on Affective Computing and Intelligent Interaction, 2013, doi:10.1109/acii.2013.47.
[9] Lin, Wei-Chao, et al. "Clustering-Based Undersampling in Class-Imbalanced Data." Information Sciences, vol. 409–410, 2017, pp. 17–26, doi:10.1016/j.ins.2017.05.008.
[10] Liu, Xu-Ying, and Zhi-Hua Zhou. "Ensemble Methods for Class Imbalance Learning." Imbalanced Learning, 2013, pp. 61–82, doi:10.1002/9781118646106.ch4.
[11] Megahed, Fadel M., et al. "The Class Imbalance Problem." Nature Methods, vol. 18, no. 11, 2021, pp. 1270–1272, doi:10.1038/s41592-021-01302-4.
[12] Petkovic, Dragutin, et al. "Improving the Explainability of Random Forest Classifier – User Centered Approach." Biocomputing 2018, 2017, doi:10.1142/9789813235533_0019.
[13] Tantithamthavorn, Chakkrit, et al. "The Impact of Mislabelling on the Performance and Interpretation of Defect Prediction Models." 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, 2015, doi:10.1109/icse.2015.93.
[14] Van Hulse, Jason, and Taghi Khoshgoftaar. "Knowledge Discovery from Imbalanced and Noisy Data." Data & Knowledge Engineering, vol. 68, no. 12, 2009, pp. 1513–1542, doi:10.1016/j.datak.2009.08.005.
[15] Xie, Chenyi, et al. "Effect of Machine Learning Re-Sampling Techniques for Imbalanced Datasets in 18F-FDG PET-Based Radiomics Model on Prognostication Performance in Cohorts of Head and Neck Cancer Patients." European Journal of Nuclear Medicine and Molecular Imaging, vol. 47, no. 12, 2020, pp. 2826–2835, doi:10.1007/s00259-020-04756-4.
[16] Zhihao, Peng, et al. "Comparison of the Different Sampling Techniques for Imbalanced Classification Problems in Machine Learning." 2019 11th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), IEEE, 2019, pp. 431–34, https://doi.org/10.1109/ICMTMA.2019.00101.