Mitigating Default Risks in Banking: Ensemble-Based Approaches for Addressing Class Imbalance in Credit Scoring

Vittoria Fiocchi (3149211), Edoardo Putignano (3195342)
Tommaso Ravasio (3192281), Leonardo Tonelli (3216378)

13th May, 2024 - Project for "30412 - Machine Learning" - Instructor: Andrea Celli

Contents

1. Introduction

The objective of the assignment was to create a reliable credit scoring algorithm suitable for banks to assess potential default risks in issuing loans to new customers, thereby minimizing defaults and supporting financial stability. We were provided with a dataset containing 11 variables to build a robust binary classification model capable of predicting the binary variable **"SeriousDlqin2yrs"**, which indicates whether an individual has experienced a 90-day delinquency or worse.

1.1 Methodology

The primary challenge of our project lies in addressing the issue of class **imbalance** within the dataset. Our approach to tackle this issue begins with a comprehensive data analysis, including feature engineering and missing value handling through various strategies. We proceed with a univariate analysis to detect the presence of outliers and a bivariate analysis to understand the correlations within the different variables; all these search operations have been carried out on the train dataset in order to prevent any data leakage. Moving on, we established a baseline model that provides a reference point for further improvement and given the complexity of our task, we opted to also train a series of ensemble methods. To address the imbalance, we employed three strategies: data resampling, cost-sensitive learning, and algorithm-level techniques. After recording the performances of all our strategies, we selected the best three performing methods and fine tuned their parameters. Finally, we conclude with a comparison among them and with the final submission. In all our projects, our primary reliance was on the scikit-learn library, adhering strictly to default parameters for all its pre-implemented functions. Across all our models, we opted for decision trees with a depth of 1. The only instance where we deviated from default parameters was to ensure that decision trees had a depth of 1, in case the default depth was different.

1.2 Feature Engineering

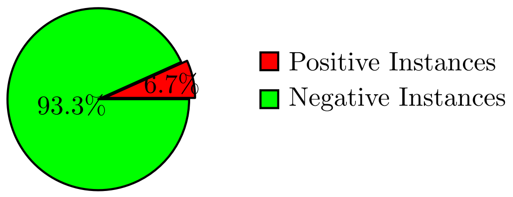
We began by applying Feature Engineering to boost the performance of our models. Feature engineering involves creating new features that aim to capture crucial information within the dataset, thereby enhancing model performance. These additional features play a vital role in improving the model's capability to differentiate between minority and majority classes, particularly within imbalanced datasets. To this end, we introduced three new variables:

- **CreditUtilizationRatio**: This variable is derived by dividing the 'RevolvingUtilizationOfUnsecuredLines' column by one plus the 'DebtRatio' column. This calculation likely aims to standardize or adjust the credit utilization ratio based on the borrower's overall debt burden.
- **Young**: This binary variable is determined by evaluating if the 'age' column falls below or equals a specified age threshold. If so, the corresponding 'Young' value is set to 1 (indicating 'True'); otherwise, it is set to 0 (indicating 'False'). This feature may capture the potential variance in credit behaviors between younger and older individuals.
- **LatePaymentsFrequency**: This feature is computed by summing the values from the columns 'NumberOfTime30-59DaysPastDueNotWorse', 'NumberOfTimes90DaysLate', and 'NumberOfTime60-89DaysPastDueNotWorse'. By aggregating the counts of late payments across different time frames, we obtain a comprehensive measure of the frequency of delinquencies.

These newly engineered features aim to provide richer insights into the dataset and potentially enhance the model's performance in addressing the challenges posed by class imbalance.

1.3 Exploratory Data Analysis

We initiated our analysis with a thorough Exploratory Data Analysis (EDA) of the training dataset. Initially, we examined the summary statistics and distributions of variables to gain insights into their behavior. Identifying missing values in "Number Of Dependents" (2945, 2.6% of total) and "Monthly Income" (19.72%), we introduced temporary binomial variables to track missing entries. Correlation analysis revealed random distribution of missing values. We imputed median values for "Monthly Income" and zeros for "Number of Dependents". We imagined that if customers had missing values for the number of dependents, it likely meant they had no dependents. We noticed a common trend among most of the variables of majority of observation clustering around small values with some extreme outliers. Due to the absence of contextual information regarding data collection, some of these outliers appeared so extreme that they raised concerns about potential errors in data collection. To mitigate any adverse impact on our models, we chose to eliminate all outliers deemed excessively extreme and significantly distant from the bulk of the data. In most instances, these outliers were found above the 0.995 quantile. Furthermore, we investigated the relationship between the target variable and independent variables using a Correlation Matrix. This examination revealed no significant correlations between the variables, indicating a lack of strong linear relationships. Lastly, we observed a high degree of **imbalance**, the minority class comprised only 6.7% of all observations. Unbalancedness can be an issue since most learning algorithms do not account for class imbalance and often prioritize the majority class and neglect the minority class. To address this issue, we devised a strategy to utilize evaluation metrics beyond accuracy, acknowledging its inherent limitations in imbalanced scenarios.



1.4 Metrics

To assess model performance comprehensively, we adopted a range of evaluation strategies, including Confusion Matrix, Recall, F1 score, and AUC. By leveraging alternative metrics that focus on effectively capturing the predictive performance of both classes, we aimed to ensure a more comprehensive assessment of our model's effectiveness.

- **Confusion Matrix**: This table provides a visualization of a classification model's performance by depicting counts of true positives, true negatives, false positives, and false negatives. It offers insights into the model's ability to correctly classify instances.

$$\begin{bmatrix} \text{True Positive} & \text{False Positive} \\ \text{False Negative} & \text{True Negative} \end{bmatrix}$$

- **Recall**: Also known as sensitivity or true positive rate, recall measures a model's capability to identify all relevant instances among the total number of actual positive instances. Mathematically, it is calculated as the ratio of true positives to the sum of true positives and false negatives.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

- **F1 Score**: The F1 score offers a balanced evaluation of a classification model's performance, particularly valuable in dealing with imbalanced datasets. Combining precision and recall, it represents the harmonic mean of these two metrics, providing a single measure of a model's effectiveness.

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

- **AUC (Area Under the Curve)**: AUC assesses the discriminative ability of a binary classification model, especially in scenarios requiring assessment across different threshold values. It measures the area under the Receiver Operating Characteristic (ROC) which plots the true positive rate (sensitivity) against the false positive rate (1 - specificity) for different threshold values.

$$\text{AUC} = \int_0^1 \text{TPR}(\text{FPR}^{-1}(u))\, du$$

1.5 Baseline Model

For our baseline modeling approach, we selected **Logistic Regression** due to its simplicity and widespread applicability in predictive tasks similar to ours. Logistic Regression estimates the probability of a binary outcome by fitting a sigmoid-shaped curve to the relationship between input variables and the log-odds of the outcome. Serving as our baseline model, it establishes a reference point for comparison with more complex models. Despite the baseline model's lower performance in recall and F1 score metrics, it achieved a commendable AUC score of 0.8077. This difference in performance can be attributed to the highly unbalanced nature of our target variable. While the model demonstrates strong discriminatory ability overall, it encounters challenges in accurately capturing instances of the minority class.

2. Ensemble Methods

To increase our performance with respect to the baseline, we explored ensemble techniques. These models are designed to improve prediction accuracy by combining multiple learning algorithms, thus benefiting from the strengths of each. In our analyses, we used three ensemble methods: Bagging Decision Tree, Boosted Decision Tree and Random Forest. Most of these show a better handling of the imbalanced data. As a matter of fact, ensemble learning techniques can achieve better performance than a single classifier when the dataset is imbalanced. This is because they reduce the risk of overfitting to the majority class and enhancing the overall prediction accuracy.

Classifier	Acc.	Recall	F1	AUC
Boosted Decision Tree	0.936	0.235	0.325	0.854
Random Forest	0.934	0.184	0.267	0.824
Bagging Decision Tree	0.932	0.187	0.265	0.817
Logistic Regression*	0.935	0.181	0.268	0.808

As expected the ensemble methods perform slightly better than the baseline model. The Boosted Decision Tree has the highest AUC score of **0.853982**, indicating the best performance among the three. The Random Forest follows with an AUC of **0.824225**, and the Bagging Decision Tree has an AUC of **0.817106**, both showing good classification capabilities as well. Given these results, it would be advantageous to consider additional strategies to enhance the performance of the ensemble methods.

3. Resampling

One of the most popular approaches to deal with imbalance data is balance resampling. Balancing the dataset allows the model to learn from both classes equally, thereby enhancing its ability to make accurate predictions and improving its interpretability. We tried three different resampling techniques explained in the following sections. In the following section we are going to refer to the class of "0"s as the majority class, while we are going to name the class of "1"s as the minority class.

3.1 Random undersampling (RUS)

Our first approach was to randomly undersample the majority class. With this method we are selecting and dropping random observations from the majority class until we reach the same number of observations in the two classes. As we are reducing the dataset size there is a higher probability that our model overfit, therefore it is important to be aware that this method can artificially increase the variance of our model. With this procedure our three models from the previous section, bagging decision tree, boosted decision tree and random forest, score in the AUC metric respectively **0.8177**, **0.8564** and **0.8472**.

3.2 Random oversampling (ROS)

Then, we tried the opposite approach: we oversampled our minority class randomly duplicating observations until the two classes were balanced. Opposite to the RUS, with this technique we are artificially reducing the variance of the dataset. From the results we see that for the same three models we get respectively 0.7720, 0.8546, 0.8280 for AUC. Noteworthy is that random oversampling performs worse than random undersampling in all the three models.

3.3 SMOTE oversampling

We have tried to randomly duplicate or remove observations, but let's try now with a more deterministic oversampling method. The SMOTE (Synthetic Minority Oversampling Technique) is a technique that generates new observations by interpolating , for a given observation x_i, between one of the k-nearest neighbors x_{zi}. In formula:

$$x_{\text{new}} = x_i + \lambda \cdot (x_{zi} - x_i)$$

Where λ is a random number in the range [0,1], this creates a new synthetic observation in the line between x_i and x_{zi}, one of the k-th nearest neighbors. We are aware that there are many versions of this algorithm, but we are going to stick to the standard regular version. With this method we get 0.7657, 0.8540, 0.8343.

Classifier	Accuracy	Recall	F1	AUC
SMOTE + Boosted	0.936	0.234	0.324	0.853
SMOTE + RF	0.934	0.191	0.278	0.834
Logistic Regression*	0.935	0.181	0.267	0.807
SMOTE + Bagging	0.931	0.185	0.262	0.765

It is important to specify that the SMOTE and the other resampling method, does not guarantee a better performance. Indeed, for example, we see that the Bagging decision tree with SMOTE records a lower AUC score than the vanilla version.

4. Class Weighting

Another solution provided in the literature ([7][8][9]) is the class weighting procedure. This is a way to give different importance to given sample classes for a target variable, by constructing a **weighted loss function** and optimizing for it. In our specific case, we would like to increase the weight of the class '1' compared to '0' to attenuate the problem of unbalancedness of the dataset. We can do it in a very simple way with the library scikit-learn, that allows the "class_weight" argument in their ensemble (and not) classifiers. In this argument we pass a dictionary with the **weight given to each class**, or even use the "balanced" standardized argument given by scikit-learn that computes the weight of the class i based on its proportion inside the dataset, using the following formula inspired by King, Zen [10]:

$$w_i = \frac{N}{C \times f_i}$$

With w_i being the weight of class i, N as the number of samples, C number of classes (2 in our case), and f_i the frequency of class i. This so-called "**cost-sensitive learning**" is tailored to each classifier since it acts on the model specific loss function. For the ensemble methods, it is applied at the base estimator level, then, since all our ensembles have Decision Tree classifiers as weak learners, we can explain in detail how class_weights is implemented in DT, which will apply for all our ensembles.

4.1 Weighted Decision Tree

The class weighted decision tree modifies the way the GINI index is computed by weighting the proportions of a class by the respective class weights, reaching a **weight balance** for both the classes. In our case the classes are only two and the weights given by the formula previously given, defined by the default "balanced" argument. For each decision tree in the ensemble the **weighted GINI index** of a node c is defined in the following way (in scikit-learn and other applications [9]):

$$GINI_c = 1 - \sum_i \left(\frac{w_i n_{c,i}}{\sum_i w_i n_{c,i}} \right)^2$$

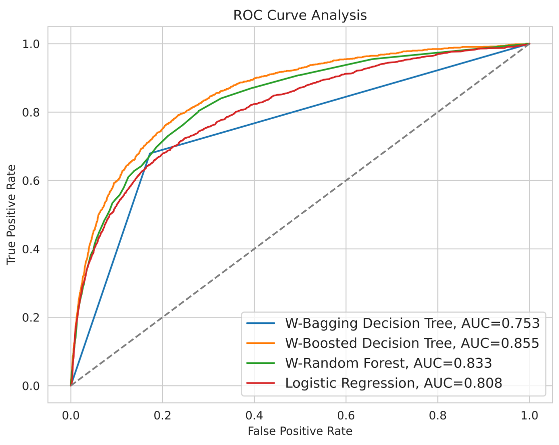
$n_{c,i}$ is the number of observations of class i in node c, and w_i is the weight assigned to class i. We can observe how by equating the weights to one, these formulas become the standard GINI index. For the weighted GINI of the **overall potential split**, it is computed as a weighted average of the indices of the potential children: weighted either by the proportions of observations in the node over the parent node's (as the standard decision tree algorithm does) or by the fraction of the parent node's total weight that is in each child node, that looks like the following:

$$GINI_{split} = \sum_c \left(\frac{\sum_i w_i n_{c,i}}{\sum_i w_i n_{p,i}} \times GINI_c \right)$$

Where the $n_{p,i}$ is the number of observations of class i in the parent node.
We could not find any reliable reference (other than a sketchy stack overflow answer) that explains which of these two weighted averages is used by scikit-learn in the class_weight argument. We couldn't figure it out even by looking at the source code of the library on GitHub. Nonetheless the results should not be much different in either case. Finally, the single decision trees are trained by **optimizing for the GINI index of each potential split**, to reach a classification prediction or a probability prediction, depending on the purpose of the study. In our case probability prediction will do the job to measure the AUC and variable predictions to calculate the other metrics.

4.2 Weighted Random Forest

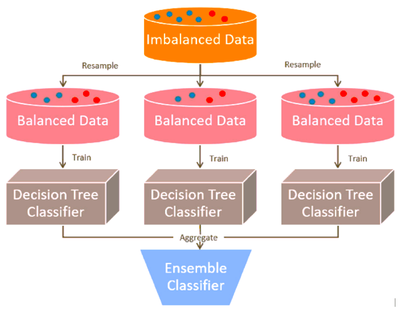
The weighted random forest (WRF) is instead another technique used by Chen, Liaw, Breiman[9], which tries to overcome the unbalance problem using the weighted decision trees described above for base estimators and **aggregate their results weighting the votes** of each tree by the average weights in the terminal nodes of all trees. Again, scikit-learn does not provide additional reference on the "class_weight" argument, therefore we assume that they use just a weighted decision tree classifier with standard aggregation of tree outputs.



In the graph are displayed the ROC curves of each of the weighted techniques, and we can see that there is a very good performance for the weighted boosted (with an AUC of 0.855) while a decrease in predictability of the bagging model. This **underperformance** could be caused by an overfitting of the model or derived by an overpenalization of the negative class of our target variable, that could make our algorithm converge into a **sub-optimal solution**.

5. Algorithmic level solutions

One last resolution to the unbalance problem can be brought by some techniques that act at the algorithmic level of the ensemble methods. These are usually defined with an **internal resampling** before the training of each base estimator inside the ensemble. This methodology can **improve the sensitivity** of a predictor to the minority class, especially when its proportion in the data is much smaller. We will see three specific examples of such a solution, based on the three ensembles that we are studying. Follows a simple illustration depicting the common architecture of these techniques.



5.1 Balanced Random Forest

For random forest (solution brought by Chen, Liaw, Breiman [9]) the strategy is to **bootstrap and under-sample** the majority class for each decision tree and train the estimator with a balanced dataset. This helps to improve the recall and the AUC measure but at the same time negatively impacts the overall accuracy, since now the frequency of the majority class is not learned to the fullest. For this, imbalanced-learn constructs a different classifier called BalancedRandomForestClassifier, which explicitly takes reference to the paper previously cited.

5.2 Easy Ensemble

The Easy Ensemble involves creating **balanced sub-samples** of the training dataset by selecting all examples from the minority class and a random subset from the majority class, in order to train its ensemble units with a balanced dataset. Rather than using pruned decision trees, boosted decision trees are used on each subset, specifically the **AdaBoost algorithm**. A pseudocode and the full description can be found in the paper by Liu, Wu, and Zhou [11]. The EasyEnsembleClassifier class from the imbalanced-learn library provides an implementation of the easy ensemble technique.

5.3 Balanced Bagging

This instead applies the same principles to **bagging**. There are different papers that suggest different types of resampling used to take care of unbalancedness combined with bagging (i.e. exactly balanced bagging [12], roughly balanced bagging [13], etc..). We decided to report the standard solution of **UnderBagging** (provided by [18]), which applies data random undersampling on the bootstrap sample prior to fitting the weak learner model.
The imbalanced-learn library provides an implementation of UnderBagging, standard methodology of the BalancedBaggingClassifier.

Classifier	Accuracy	Recall	F1	AUC
Easy Ensemble	0.935	0.742	0.327	0.855
Balanced Random Forest	0.935	0.772	0.319	0.852
Balanced Bagging	0.935	0.619	0.332	0.821
LogisticRegression*	0.935	0.181	0.267	0.807

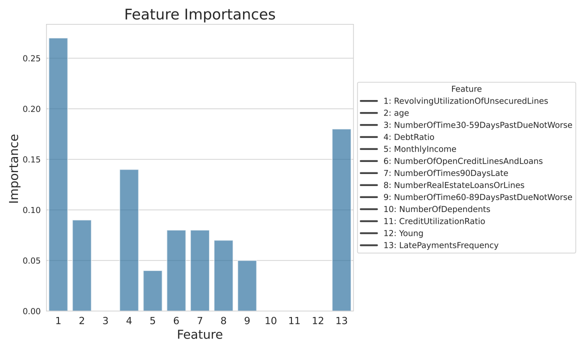
Here we can see that the performances of such methods are the best among the strategies provided, especially looking at how the **tradeoff** between accuracy and recall has been well calibrated. These are methods that successfully help us in our cause, they are in fact very much used in the field to solve the unbalance issue.

5.4 Paper reference

In our search for solutions to unbalanced binary classification, we found an **intriguing paper** [15] examining resampling+ensembling combination methods. It provides performance data on various combinations for datasets like Glass, Yeast, and Ecoli. We identified similarities between our dataset and one present in the study, particularly with Glass-6, where ROS+Adaboost and SMOTE+RandomForest showed excellent performance. We had already run the same methods but with different hyperparameters. Despite implementing these combinations with the same arguments, our results didn't match. This could be due to dataset complexity, sample size diversity, or implementation issues. Nonetheless, exploring these insights was valuable.

6. Comparison

After an ineffective finetuning of the best three models, we obtain similar and relatively high AUC scores (at least 0,85) indicating strong classification capabilities. However, we have scores in accuracy, recall, and F1 score that might affect their ability of avoiding false negatives (recall) or maintaining a balance between precision and recall (F1 score). RUS + Boosted Decision Tree is the model with the highest AUC score. Hence, being its other metrics relatively similar to the ones in the other models , we use RUS + Boosted Decision Tree as our "best model".



The bar chart of feature importances done for the best model shows that "RevolvingUtilizationOfUnsecuredLines" is the most significant predictor. "DebtRatio" and "NumberOFTimes90DaysLate" are also notably significant. Also, among the features created with feature engineering, the "LatePaymentsFrequency" shows a high level of importance, comparable to some of the key features like "DebtRatio" and "NumberOfTimes90DaysLate". The relatively high importance of "LatePaymentsFrequency" highlights the value of this engineered feature in predicting outcomes.

7. Conclusion

The study tries to seek and implement a good solution for modeling the target variable in the dataset given in the assignment, exploring different methodologies evaluated in the current literature on unbalanced binary classification. The research presents **various limitations** indeed. On the data cleaning side, we did not evaluate other imputation procedures that could have benefited our models, and made some small assumptions when dealing with outliers that could have decreased the informativeness of our training data. Another limitation is the evaluation of just the three ensemble methods, a choice guided both by theory and **limited computational resources** (SVM and Neural Networks were too expensive).

Final Results

Classifier	Acc.	Recall	F1	AUC
RUS + Boosted	0.795	0.757	0.326	**0.856**
Easy Ensemble	0.800	0.742	0.327	0.855
ROS + Boosted	0.802	0.746	0.331	0.854
W-Boosted Decision Tree	0.804	0.740	**0.332**	0.854
Boosted Decision Tree	**0.936**	0.234	0.324	0.853
SMOTE + Boosted	0.936	0.234	0.324	0.853
Balanced Random Forest	0.784	**0.772**	0.319	0.852
RUS + Random Forest	0.782	0.763	0.315	0.847
SMOTE + Random Forest	0.934	0.191	0.278	0.834
W-Random Forest	0.934	0.124	0.199	0.833
ROS + Random Forest	0.930	0.244	0.314	0.825
Random Forest	0.934	0.183	0.267	0.824
Balanced Bagging	0.836	0.619	0.332	0.821
RUS + Bagging	0.783	0.690	0.294	0.817
Bagging Decision Tree	0.932	0.186	0.265	0.817
LogisticRegression*	0.935	0.181	0.267	0.807
ROS + Bagging	0.924	0.219	0.274	0.772
SMOTE + Bagging	0.931	0.185	0.262	0.765
W-Bagging Decision Tree	0.817	0.679	0.328	0.753

One last point is instead the lack of deep tuning of hyperparameters in all the strategies used, that could've resulted in a boost in performances, again a choice guided by our limited resources in time and computing power. Could also be asserted a critique towards the performance metric given in the assignment, as the paper [17] describe, also the AUC metric could be biased in case of unbalancedness. Then some more **carefulness** in that direction could have increased the confidence of our results. In conclusion, various techniques were explored to address the issue of dataset imbalance. Initially, a basic logistic regression model achieved an AUC score of 0.8077. Subsequently, "vanilla" ensemble methods were experimented with, with Boosted Decision Tree emerging as the most successful, achieving an AUC score of 0.8539, significantly surpassing the baseline. Integration with data resampling techniques led to the top-performing model, rus+Boosted Decision Tree, achieving an AUC score of 0.8564. Attempts to construct models using weighted loss functions yielded comparable results, although falling short of the top model's AUC score. Algorithm-level techniques, such as Easy Ensemble, also produced robust models, with an AUC score of 0.8556, closely aligned with rus+Boosted Decision Tree. Shallow fine-tuning was conducted on the top three models, resulting in marginal performance improvements. Ultimately, the RUS + Boosted Decision Tree model, with parameters obtained from fine-tuning, proved to be the most effective, achieving an AUC score of 0.8582. Although only slight improvements were made to the AUC score of the initial "vanilla" Boosted Decision Tree, differences between models are minimal, and another run with a different random state could yield slightly different results. However, we are confident in having developed several reliable credit scoring algorithms capable of accurately predicting the variable "SeriousDlqin2yrs".

REFERENCES

[1] Surendra Kuma, "How to Handle/Detect Outliers for machine learning?" jul 25, 2021 https://medium.com/@surendraprjapat/how-to-handle-outliers-for-machine-learning-fef864e30c0b

[2] Kartik Chaudhary, "How to deal with Imbalanced data in classification?", 23 sept. 2023, https://medium.com/game-of-bits/how-to-deal-with-imbalanced-data-in-classification-bd03cfc66066

[3] J.Osborne, "Dealing with Missing or Incomplete Data: Debunking the Myth of Emptiness"

[4] "Learning from Imbalanced Data." Jeremy Jordan, Jeremy Jordan, 5 Mar. 2023, www.jeremyjordan.me/imbalanced-data/.

[5] Haldar, Supratim, et al. "How Does Class_weight Work in Decision Tree." Data Science Stack Exchange, 1 Apr. 1965, datascience.stackexchange.com/questions/56250/how-does-class-weight-work-in-decision-tree.

[6] "Trees - Weights and Feature Importance (Theory + Code)." YouTube, YouTube, 3 Jan. 2024, www.youtube.com/watch?v=Zy4-CYoc-MY.

[7] Gajowniczek, Krzysztof, Iga Grzegorczyk, Tomasz Zabkowski, and Chandrajit Bajaj. 2020. "Weighted Random Forests to Improve Arrhythmia Classification" Electronics 9, no. 1: 99. https://doi.org/10.3390/electronics9010099

[8] Yu, Q. (2011). Weighted bagging: a modification of AdaBoost from the perspective of importance sampling. Journal of Applied Statistics, 38(3), 451–463. https://doi.org/10.1080/02664760903456418

[9] Chen, Chao & Breiman, Leo. (2004). Using Random Forest to Learn Imbalanced Data. University of California, Berkeley.

[10] King, Gary and Langche Zeng. "Logistic Regression in Rare Events Data." Political Analysis 9 (2001): 137 - 163.

[11] Liu XY, Wu J, Zhou ZH. Exploratory undersampling for class-imbalance learning. IEEE Trans Syst Man Cybern B Cybern. 2009 Apr;39(2):539-50. doi: 10.1109/TSMCB.2008.2007853. Epub 2008 Dec 16. PMID: 19095540.

[12] R. Maclin, and D. Opitz. "An empirical evaluation of bagging and boosting." AAAI/IAAI 1997 (1997): 546-551.

[13] S. Hido, H. Kashima, and Y. Takahashi. "Roughly balanced bagging for imbalanced data." Statistical Analysis and Data Mining: The ASA Data Science Journal 2.5-6 (2009): 412-426.

[14] H. He and E. A. Garcia, "Learning from Imbalanced Data," in IEEE Transactions on Knowledge and Data Engineering, vol. 21, no. 9, pp. 1263-1284, Sept. 2009, doi: 10.1109/TKDE.2008.239

[15] Azal Ahmad Khan, Omkar Chaudhari, Rohitash Chandra, "A review of ensemble learning and data augmentation models for class imbalanced problems: Combination, implementation and evaluation", Expert Systems with Applications, Volume 244, 2024, 122778, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2023.122778.

[16] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011

[17] Brabec, Jan & Machlica, Lukas. (2018). Bad practices in evaluation methodology relevant to class-imbalanced problems.

[18] Barandela, R., Sanchez, J., and Valdovinos, R. 2003. New Applications of Ensembles of Classifiers. Pattern Analysis and Applications. 6(3), pp. 245-256.