School of Information Technology and Engineering at the ADA University

School of Engineering and Applied Science at the George Washington University

EXPLORING CLASS IMBALANCE SOLUTIONS: INVESTIGATING THE EFFECTIVENESS OF DATA BALANCING TECHNIQUES ON MODEL PERFORMANCE

A Thesis

Presented to the Graduate Program of Computer Science and Data Analytics

of the School of Information Technology and Engineering

ADA University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science and Data Analytics

ADA University

By

Shukurov Shamil

April 2024

THESIS ACCEPTANCE

This Thesis by: Shamil Shukurov

Entitled: *Exploring Class Imbalance Solutions: Investigating the Effectiveness of Data Balancing Techniques on Model Performance*

has been approved as meeting the requirement for the Degree of Master of Science in Computer Science and Data Analytics of the School of Information Technology and Engineering, ADA University.

Approved:

Adviser	(Date)
Program Director	(Date)
Dean	(Date)

Abstract

The class Imbalance problem is one of the common challenges in machine learning. Despite its common nature, there is no best solution to handle this problem. The purpose of this thesis, ”Exploring Class Imbalance Solutions: Investigating the Effectiveness of Data Balancing Techniques on Model Performance,” is to investigate the topic of class imbalance and provide insight into the efficiency of different imbalance solutions. The ultimate objective of this thesis is to give practitioners practical guidance on handling the challenges brought about by class inequality, which will help to produce more accurate and equitable machine learning models.

Contents

List of Figures

List of Tables

Abbreviations

ML	Machine Learning
PTB	Propensity to Buy
SMOTE	Synthetic Minority Over-sampling Technique
ADASYN	Adaptive Synthetic Sampling Approach for Imbalanced Learning
RAMO	Ranked Minority Oversampling
FPR	False Positive Rate
TPR	True Positive Rate
AUC	Area Under Curve
ROC	Receiver Operator Carachteristic
ROS	Random Over Sampling
RUS	Random Under Sampling
XGBoost	eXtreme Gradient Boosting
XGB	XGBoost
PSM	Parallel Stacked Models
SSM	Sequential Stacked Models

Chapter 1

Introduction

The class imbalance issue is a widespread problem in machine learning and data science, ultimately having a negative impact on the predictions of the models designed and developed. An imbalance of classes in a dataset appears when each class is not evenly represented, causing the majority class to be larger than one or more minority classes. Most common algorithms assume that distributed classes equal misclassification costs or not. Thus, models using these algorithms address data set imbalance issues by presenting unequal representatives; hence, they do not provide fair accuracy gains across all classes of data [1].

All kinds of machine learning applications including financial fraud detection or rare disease identification in medical diagnostics are significantly affected by class imbalance. These contexts, however, demonstrate a tendency for the minority class to be the subject of the event, which makes its accurate identification a precondition for the practical usability of the model. Fraud detection is just one application where fraudulent transactions are just a small fraction of all transactions, making the minority class. On the other hand, the datasets for the rare diseases in medical diagnosis are mostly overwhelmed by vast data of healthy cases or more common diseases.

Class imbalance exerts several deleterious effects on the accuracy of machine learning models in a number of ways. First of all, traditional algorithms tend to be biased towards the majority class and do not distinguish well for the minority class[2]. This overfitting to the majority class means that while the model may accurately predict expected outcomes, it fails to capture the nuances and patterns specific to the rare events or conditions, which are often of greater interest [2]. Thus, the second problem arising from the minority class under-representation is insufficient learning about its properties, which causes the lack of the model's generalizability

and lower performance.

In addition, if class imbalance is not solved, social disparities can arise. In the criminal justice, finance, and employment sectors, biased model predictions can make things more unequal and unfair. This is particularly relevant for credit score systems or models through which police act because the majority opinion could cause people in the minority class to be abused.

The work of studies on these issues has pointed to the vital part of creating the techniques to manage class imbalance within the ML community. Some of these practitioners believe that the models derived from the marginal class distribution, which is the same as that of the training data, are best suited for learning, and so the new data will be classified by the model built from the data with the same underlying distribution. Other experts consider that the data set should be enrolled with more minority class samples than normal because the classifier will not correctly label minority class examples otherwise [3].

1.1 Application Domains

Imbalance in class distribution is widespread not only in non-domains but also in business, medical insurance, and public health domains where data mining is used. This problem behaves itself at a structural level and in some application areas [4].

Banking: Fraud Detection In banking, as a sector class, imbalance is one big problem experienced in fraud detection. Fraudulent transactions are relatively infrequent compared to normal business transactions. Consequently the costs associated with this rare type of error are very high both in terms of finances and customer confidence [5]. Traditional models trained on such imbalanced datasets tend to predict most transactions as legitimate, resulting in a high rate of false negatives for fraudulent activities.

Medical Diagnosis: Rare Disease Identification The medical industry has been known to have its fair share of discrepancies in diagnoses of rare diseases. In such a context, datasets are greatly biased towards healthy patients, with examples of rare diseases forming a small minority. This imbalance could cause the models to be able to identify healthy cases well but not be able to detect diseases enough, which is problematic in medical diagnostics where the costs of misdiagnosis are high.[2].

Banking: PTB Models In banking, PTB models or propensity-to-buy models are of great importance due to the variety of products and services offered to customers: loans, credit cards, investment products, and insurance services. The broad customer base that is characteristic of banks may find that only a small part of their customers would be interested or qualify for a particular product. This case points to the class imbalance, where the number of buyers of the specific product (the minority class) is largely out of proportion to the number who are not (the majority class).

Social Media: Hate Speech Detection The proliferation of social media platforms has ushered in challenges related to monitoring and filtering undesirable content, such as hate speech. The vast majority of content on these platforms is neutral or non-offensive, making hate speech a minority class. The risk here is twofold: models may overlook hate speech due to its rarity, or they may overfit to the minority class and incorrectly classify neutral content as offensive [6]. Addressing class imbalance through methods like focal loss or extended isolation forest can help finely tune the balance between sensitivity and specificity, ensuring that content moderation tools are both practical and fair.

Network Intrusion Detection Sun et al. (2011) emphasize the role of class imbalance in network intrusion detection [4]:
As network-based computer systems play increasingly vital roles in modern society, attacks on computer systems and computer networks grow more commonplace. Learning prediction rules from network data is an effective anomaly detection approach to automate and simplify the manual development of intrusion signatures. It is found that different types of network attacks are present, some overwhelming, others rare in the collection of network connection records.

Financial Management The class imbalance problem in financial management is just as significant, and it comes with its own set of issues that affect how decisions are made, risks are assessed, and strategies are planned. Financial management includes a lot of different tasks, such as choosing investments, keeping track of stocks, figuring out credit risk, and finding financial crimes. Inequality between classes can significantly affect all of these areas, making it harder to trust the predictive models used to make decisions about money and how to spend it. Sanz et al. tested imbalanced data classification methods in 11 financial problems, including stock market prediction, credit card/loans approval application system, and fraud detection [7].

Manufacturing: Defect Detection The quality control process in production plays a crucial role in maintaining product properties. The main problem with detecting defects can be a class imbalance, as defects are often rare compared to samples of non-defective items. Unrecognition of such defects will result in great economic losses and damage to brand equity. Utilizing class imbalance techniques, e.g., oversampling techniques and center-out anomaly detection like isolation forest, defects of low prevalence would be detected with higher accuracy and better customer satisfaction.

1.2 Problem Statement

Class imbalance is a significant issue in predictive modeling that makes it difficult to make fair and accurate predictions in many different areas of application. Even though this issue is widely known, there needs to be a complete comparison of class imbalance solutions considering various aspects. This gap between the research and reality shows how important it is to carefully look into how different solutions work in different situations where classes are imbalanced. How well these solutions work may depend on aspects like the data type, how imbalanced it is, and the specific needs of the prediction task.

This thesis aims to fill a crucial gap by thoroughly assessing different class imbalance solutions across various imbalance percentages. Initially, this research had two main objectives:

1. Assessment of Solutions for Class Imbalance.
2. Creating Practical Guidelines.

Assessment of Solutions for Class Imbalance: Comparing results of each model for each dataset to understand the effect of balancing.

Creating Practical Guidelines: Creating practical guidelines for practitioners to address class imbalance. The guidelines are based on empirical evidence collected from evaluating the solutions mentioned, offering insights into choosing the most effective techniques depending on problem characteristics. This contribution is anticipated to have substantial implications for academic research and practical applications, improving machine learning models' accuracy, fairness, and effectiveness in addressing class imbalance.

Lastly, we also proposed our new approaches using ensemble methods when we conducted the study. Therefore, our research now has a third main objective: to develop and test new class imbalance solutions.

1.3 Limitations of the study

This study has several limitations. The first limitation is the quantity and variety of the datasets used. To make results more significant, we need more than 30 datasets with different minority class percentages. However, we only used 13 datasets, of which five were resampled from the same data source.

Additionally, we used the same set of hyperparameters in baseline models for each task. There are two reasons for this. First, we wanted to compare only the effect of the balancing technique across all tasks. Additionally, applying a searching algorithm for hyperparameter tuning is very time-consuming. Having the same set of hyperparameters can cause the following problem: There could be a better baseline model with the best hyperparameters, for which we would not need any balancing technique.

Furthermore, this study only focuses on binary classification. The class imbalance problem occurs in binary and multiclass classification tasks, but we only analyzed binary classification cases in this study. However, applying the techniques discussed here to multiclass cases can be the next future work.

Lastly, in some datasets, we modified the multiclass target to obtain a binary classification problem, which might also affect the nature of the problem.

Chapter 2

Literature Review

A number of algorithms and techniques were proposed as remedies for classification problems with unbalanced data sets. Some algorithms de-biased the data to reach the desirable class balance. In another scenario, they adjust machine learning algorithms to give more focus to the minority class.

This chapter will study such techniques and investigate their impact on the performance of machine learning models. We will investigate multiple studies involving their methodologies, datasets, and findings.

Haixiang et al. thoroughly examines methods and applications for dealing with unbalanced data in their study titled ”Learning from class imbalanced data: Review of methods and applications” [8]. The study provides methods for tackling imbalanced datasets, including preprocessing and cost-sensitive learning. It seeks to analyze classification algorithms like ensemble classifiers and modified algorithms. Notably, the paper outlines the critical role of multiclass classification in advancing binary classification algorithms. The research technique included a systemic search to assemble necessary papers to refer to trends in imbalanced learning during the last decade. The field of study can be applied to multiple domains, ranging from financial management to medical diagnosis and telecommunications, thereby illustrating the far-reaching practicality of class imbalance mitigation in data analysis.

In their paper titled ”SMOTE: Synthetic Minority Over-sampling Technique,” Chawla et al. have proposed an innovative technique for handling the class imbalance problem in the case of imbalanced datasets in classification tasks [9]. SMOTE method combines under-sampling of the majority class with a specific over-sampling of the minority class to show the classifier performance.

Alternative methods are presented side by side with a methodology and various performance indicators for data sets and classifiers are provided to show the efficacy of the approach. The work serves an example of resolving class imbalance in machine learning and demonstrate several approaches of enhancing the efficiency of such technologies in practice.

"Adaptive Synthetic Sampling Approach for Imbalanced Learning", authored by He et al., proposes a new method of dealing with imbalanced data classification issues through the ADASYN algorithm [10]. Through ADASYN, synthetic data samples are automatically generated to give the minority class the expected performance level, and bias is reduced. This ensures that the model can handle imbalanced data sets by doing the weight distribution of minority class instances based on their difficulties of learning. It supplies artificial data for examples from the minority class, which are harder to learn and identify, unlike the majority examples. This Adaptive Synthetic Sampling approach intends to reduce the bias created by disproportionate number of instances from the majority class and the boundary of classification of the classifier to the boundary of complex examples, which contributes to the overall improved recognition performance. Not only do the simulation results and evaluation metrics reveal how the algorithm operates with the real-world data gathered from the UCI Machine Learning Repository, but also they shed light on how the algorithm is working with data from the real world. The message holds that unfair learning is prominent in machine learning study. Its concept implies the fact that ADASYN may play an instrumental role in developing the comprehension for dwindling classes and incremental imbalanced learning.

In their research paper titled "Boosting Support Vector Machines for Imbalanced Data Sets," Wang and Japkowicz describe a novel approach that makes use of support vector machines and boosting algorithms to improve the accuracy of predictions made for both majority and minority classes [11]. To avoid the limitations of under- and oversampling, they emphasize the distribution of data and the change of classifier modifications. The study demonstrates that their ensemble classifier approach is capable of solving skewed vector spaces and overfitting in an effective manner, which has the promise of providing a solution for imbalanced data sets in applications that are used in the real world.

The research paper titled "Balancing Training Data for Automated Annotation of Keywords: a Case Study" by Batista, Bazzan, and Monard discusses the process of learning from imbalanced data sets in SWISS-PROT keyword annotation [12]. The authors highlight the difficulties faced when dealing with skewed class distributions. They discuss the impact of imbalanced data on

machine learning algorithms such as k-nearest Neighbor and decision trees, emphasizing the need for data balancing techniques to improve model performance. To address the class imbalance issue, the research suggests using random oversampling and SMOTE oversampling combined with Tomek Links. The findings indicate a reduction in the number of false negatives and a slight increase in the number of false positives.

Dal Pozzolo, Caelen, and Bontempi (2015) comprehensively analyze the effectiveness of undersampling as a solution for class imbalance in their paper titled "When is Undersampling Effective in Unbalanced Classification Tasks?" [13]. The study delves into the impact of undersampling on classification accuracy by considering the increase in variance due to sample reduction and the warping of the posterior distribution caused by changes in prior probabilities. The findings suggest that the effectiveness of undersampling is influenced by factors such as the number of samples, classifier variance, class imbalance degree, and posterior probability value. The research highlights the need for a nuanced approach to undersampling, recommending adaptive selection techniques like racing for tailored and calibrated undersampling strategies. This paper contributes valuable insights to exploring class imbalance solutions, emphasizing the importance of considering various factors and adopting customized approaches to enhance classification effectiveness in unbalanced datasets. The authors recommend a cautious and tailored approach to rebalancing classes before learning a classifier in unbalanced classification scenarios. Specifically, they suggest the following strategies:

1. Avoid Naive Undersampling: The authors caution against a simplistic or unsupervised application of undersampling, as it may not always lead to improved classification performance. Instead, they advocate for a more informed and adaptive selection of undersampling techniques.

2. Adopt Specific Adaptive Selection Techniques: The authors suggest using specific adaptive selection strategies, such as racing, to alleviate class imbalance successfully. With the help of these methods, the undersampling strategy can be evaluated and calibrated individually depending on the needs of the learning job and the dataset's properties.

3. Consider Prior Probabilities and Variances: The authors talk about how crucial it is to take into account how undersampling may affect the dataset's prior probabilities and variances. Making decisions about rebalancing strategies and their possible impact on classification accuracy can be aided by thoroughly understanding these effects.

By recommending a thoughtful and adaptive approach to rebalancing classes in unbalanced classification scenarios, the authors aim to enhance the effectiveness of undersampling techniques and improve the overall performance of classifiers trained on imbalanced datasets.

An innovative method to class imbalance in machine learning is introduced in a paper, embodied by Chen et al. referred to as the RAMOBoost approach [14]. RAMOBoost unifies with two boosting and oversampling approaches to specifically create instances of oversampled minority class with the ratios of nearest neighbors. This dissimilitude from the conventional methods is the AI-based algorithm continuously changing sample weights and directing on the majority as well as minority class instances especially difficult to learn. The study articulates that dealing with the magnitude of discrepancy itself in a learning situation that has intrinsic biases poses serious challenges. On the contrary, the assessments are done on two-class imbalanced scenarios, but the authors submit the possible adjustments to multiclass extreme situations. The work is specified as the use of Euclidean distance to get the measure, and then this can be explored in other distance measures. This part of the paper points to the problem of parameter tuning and offers research implications for improvements of that. Interestingly, the presented RAMOBoost may turn out to be a powerful tool to be used causing a paradigm shift in the imbalanced learning domain for exploring new ideas and finding applications accordingly.

The paper of Krawczyk is a complete summary of the challenges regarding learning from the imbalanced data and the future directions [2]. It talks about the turning point from the application of biased learning beyond the two tasks that are imbalanced, and what is the essence of the true imbalanced data in the contemporary applications. Data-level and algorithm-level techniques will be the focus of the author who emphasizes the fact that solutions for the complex dataset imbalance require both methods. This work argues that the main issues are elaborated on the seven topics such as classification, regression, clustering, streams data, and big data as well as proposing ideas about the open obstacles and solutions. In particular, the paper considers in-depth investigation of non-majority class hierarchies, hybrid multi-class and multi-label learning algorithms, preliminary clustering procedures, and improved metrics for challenging regression problems.

The article ”Learning from Imbalanced Data” by He and Garcia presents, in a broad manner, the issues and developments in handling imbalanced problems [1]. The series of the constitutive role of learning from an unbalanced data in different important applications such as surveillance, security, Internet, and finance is discussed. The paper pinpoints the new learning laws,

paradigms, algorithms, and mechanisms which must be put in place to deal successfully with data scarcity and extreme class imbalance. It concerns the recent technologies and assement metrics employed with the imbalanced situation to quantify learning achievements.

In their publication titled "SMOTEBoost: Improving Prediction of the Minority Class in Boosting," Chawla et al. propose a new technique for mitigating imbalance based on the SMOTE technique integration with the boosting algorithms, "SMOTEBoost," [15]. SMOTEBoost applies the approach by generating synthetic samples from the minority class using the recursively via SMOTE after each boosting round to dichotomize the distribution and expand the discrimination accuracy. The performance of SMOTEBoost based on the experiments on diverse datasets has been found to be higher than the conventional boosting algorithms and capable standalone SMOTE applications that attain the higher F-values and results in the better accuracy of the minority class predictions. Hence, this methodology is especially benefiting in domains where the presence of rare events plays a crucial role, such as network intrusion detection and medical diagnostics, where classifying minority classes is crucial. One of its main contributions is an argument that underlines the importance of class imbalance in machine learning and a strategy for boosting the learning performance in minority classes.

Liu et al. provide new algorithms in their work titled "Exploratory Undersampling for Class-Imbalance Learning," EasyEnsemble and BalanceCascade, which try to tackle the problems associated with traditional undersampling methods handling imbalanced data sets [16]. Contrasting to conventional class-imbalance learning approaches, such algorithms have manifested higher outcomes which are precise by AUC, F-score and G-mean values. Moreover, they offer faster training times, making them efficient solutions for dealing with imbalanced data sets.

The paper "RUSBoost: Improving Classification Performance when Training Data is Skewed" by Seiffert et al. introduces RUSBoost. This novel algorithm combines random undersampling with boosting to enhance classification performance in the presence of skewed training data [17]. The study compares RUSBoost to SMOTEBoost, highlighting RUSBoost's simplicity, computational efficiency, and faster model training times. Results demonstrate that RUSBoost outperforms SMOTEBoost on most datasets, offering a promising alternative for addressing class imbalance. Future research directions include evaluating RUSBoost with additional learners and datasets and comparing its performance with other boosting algorithms tailored for class imbalance.

Chapter 3

Research Methodology

In this thesis, we tested 21 different imabalance techniques on 13 real-world datasets. Since some of the methods are trained with three different baseline models, we have 42 models for each task. We can divide tested techniques into four main groups:

1. **Resampling Methods** are used to balance the data before training the model. Each of these methods is tried on three different baseline models: XGBoost, RandomForest, and Logistic Regression

2. **Algorithm Level Methods** are techniques designed to optimize algorithms for handling imbalanced datasets.

3. **Anomaly Detection Algorithms** are designed for detecting anomalies, but we also test them in this study to see if they help class imbalance

4. **Novel approaches using Ensemble Learning** We propose new approaches for class imbalance data using Ensemble Learning.

In the following sections, we will discuss baseline models and imbalance techniques.

3.1 Datasets Used

To conduct the research, 13 real-world imbalanced datasets were collected from 9 sources. In Table 3.1, you can see the overview of used datasets.
The "Credit Card Fraud Detection" dataset contains transactions made by credit cards in September 2013 by European cardholders [18]. The original Dataset contains 492 frauds out of

Table 3.1: Overview of Datasets

Dataset	#Rows	#Features	Minority Class Percent	Target Name
cc_fraud_05 [18]	98,400	30	0.5	Class
cc_fraud_1 [18]	49,200	30	1	Class
cc_fraud_2 [18]	24,600	30	2	Class
Nursery [19]	12,960	25	2.55	class
cc_fraud_3 [18]	16,400	30	3	Class
CoverType [20]	50,000	54	3.47	target
cc_fraud_4 [18]	12,300	30	4	Class
GMC_Credit_Scoring[21]	150,000	10	6.7	SeriousDlqin2yrs
Churn_Ecom [22]	49,358	47	11.44	target_class
PTB_Online [23]	12,330	26	15.47	Revenue
Wine Quality [24]	6,497	11	19.65	quality
Taiwan_Credit_Scoring[25]	30,000	23	22.12	Y
Adult [26]	47,621	44	24.2	income_class

284,807 transactions. We resampled this dataset to obtain five different datasets with different minority percentages. Besides resampling, no other preprocessing technique was applied to the dataset.

The ”Nursery” dataset was derived from a hierarchical decision model originally developed to rank applications for nursery schools. It was used in the 1980s when there was excessive enrollment to these schools in Ljubljana, Slovenia, and the rejected applications frequently needed an objective explanation [19]. In this thesis, we predict whether or not an application is recommended. We used One-hot encoding to encode categorical variables. No other preprocessing techniques are applied.

The “Covertype” dataset is used to predict forest cover type from cartographic variables only (no remotely sensed data) [20]. Our model predicts whether the cover type is ’Krummholz’ or not. The original dataset contains 581012 rows; we took a sample of 50000 due to the extensive training size.

The ”Give Me Some Credit” dataset aims to improve on the State of the art in credit scoring by predicting the probability that somebody will experience financial distress in the next two years [21].

The “User Churn” dataset is built to be used in the research paper on the user churn model in e-commerce retail & simulation-based technique for set importance estimations [22]. The goal is to predict customer churn.

The ”Online Shoppers Purchasing Intention” dataset consists of feature vectors belonging to

12,330 sessions. The dataset was formed so that each session would belong to a different user in 1 year to avoid any tendency to a specific campaign, special day, user profile, or period. The goal is to predict a customer's propensity to buy. Of the 12,330 sessions in the dataset, 84.5% (10,422) were negative class samples that did not end with shopping, and the rest (1908) were positive class samples ending with shopping [23].

The "Wine Quality" dataset concerns red and white Vinho Verde wine samples from northern Portugal. The goal is to model wine quality based on physicochemical tests [24]. Our experiments predict whether wine is high quality (quality¿=7) or not.

The "Default of Credit Card Clients" dataset is used to predict the probability of default of credit card users in Taiwan [25].

The "Adult" dataset contains demographic information, and the goal is to predict whether income exceeds $50K/yr based on census data [26]. This dataset is also known as the "Census Income" dataset. The column "native-country" is dropped in preprocessing due to the high number of unique values, and the "education" column is also dropped because we can get the same information by using the "education-num" column. One-hot encoding is used to encode the remaining categorical columns.

3.2 Evaluation Metrics

In this thesis, we have tested 42 different models on 13 different datasets, resulting in 546 models. We have collected experiment results in the format of Table 3.2.

Table 3.2: First ten results of the experiments

Model	Sample	Accuracy	F1 Score	Precision	Recall	AUC Score	Dataset
SelfBoostingXGB	Train	0.868	0.690	0.798	0.608	0.926	Adult
SelfBoostingXGB	Test	0.862	0.676	0.803	0.583	0.920	Adult
PSM	Train	0.866	0.703	0.758	0.655	0.925	Adult
PSM	Test	0.857	0.686	0.752	0.631	0.918	Adult
SSM	Train	0.866	0.700	0.763	0.647	0.924	Adult
SSM	Test	0.858	0.685	0.761	0.623	0.917	Adult
SelfBoostingXGB	Train	0.999	0.939	0.992	0.891	0.999	cc_fraud_1
SelfBoostingXGB	Test	0.997	0.847	0.949	0.765	0.969	cc_fraud_1
PSM	Train	1.000	0.985	0.987	0.982	1.000	cc_fraud_1
PSM	Test	0.997	0.840	0.916	0.776	0.941	cc_fraud_1

Analyzing and deriving insights from such results with diverse datasets and models of minority percentages can be challenging without special metrics and approaches. We will discuss these approaches in 3.2.1; let's now focus on the widely used metrics in classification.

Accuracy is the most common metric for assessing a binary classification model. It quantifies the model's accuracy by calculating the correct and total prediction. A high accuracy score implies that the model is making a significant proportion of accurate predictions, whereas a low accuracy score shows that the model is making an excessive number of wrong predictions. The calculation of accuracy is determined by employing the subsequent formula:

$$Accuracy = \frac{\#correct\_predictions}{\#all\_predictions}$$

Accuracy can be misleading in imbalance classification tasks. Consider the following scenario: You are building a model with a dataset with 98% negative classes. Now, if we build a very simple model that only predicts negative classes all of the time, we would get 98% accuracy. Therefore, we need some other metrics to evaluate the classification model.

A confusion matrix is often used to represent the results of a binary classification model. As we can see from Figure 3.1, it consists of four components: True Positive (TP): The number of positive cases the model accurately classifies as positive. False Positive (FP): The number of negative cases the model mistakenly classifies as positive. True Negative (TN): The number of negative examples the model accurately classifies as negative. False Negative (FN): The

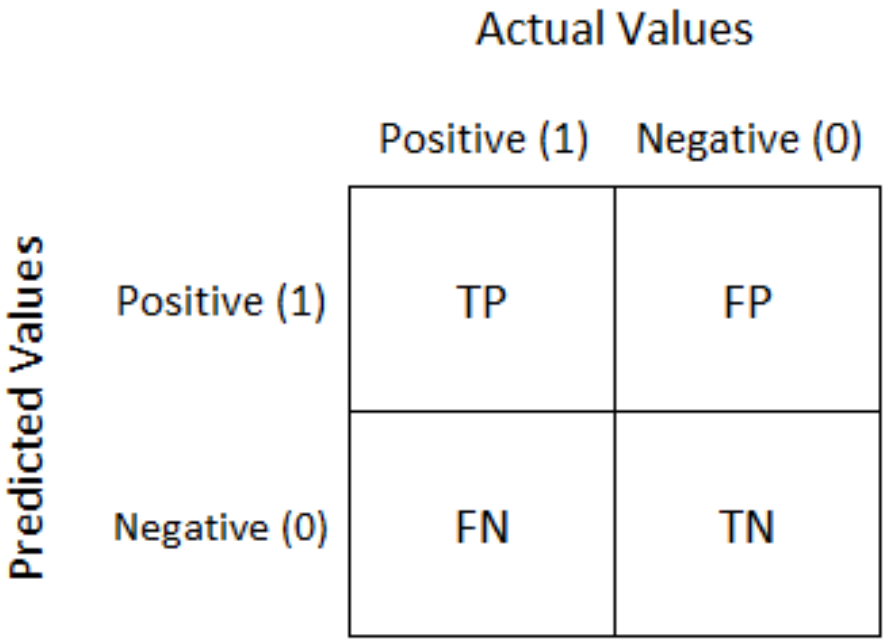

Figure 3.1: Confusion Matrix

number of positive cases the model mistakenly classifies as negative.

You can find a lot of information in the confusion matrix, but often, you might want a shorter measure. The precision of the classifier, or the accuracy of the positive predictions, is an important one to examine [27].

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

Precision is typically used along with another metric named recall, also called sensitivity or true positive rate:

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

A commonly used approach to evaluate classifiers is to merge precision and recall into a single measure known as the F1 score. This is especially useful when a straightforward method is required to compare two classifiers. The F1 score is calculated as the harmonic mean of precision and Recall. While the regular mean considers all values with equal importance, the harmonic mean assigns significantly greater significance to low numbers.

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

The classifier will only achieve a high F1 score if Recall and precision are good.

The AUC-ROC curve is a metric used to evaluate the performance of classification models across different threshold settings. The ROC curve is a graphical representation of the probability distribution. In contrast, the AUC (Area Under the Curve) quantifies the degree to which the classes may be distinguished. It quantifies the model's ability to differentiate between different classes. A higher AUC indicates that the model is more effective in correctly predicting 0 classes as 0 and 1 classes as 1. The Receiver Operating Characteristic (ROC) curve is graphed by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR), with TPR shown on the y-axis and FPR on the x-axis.

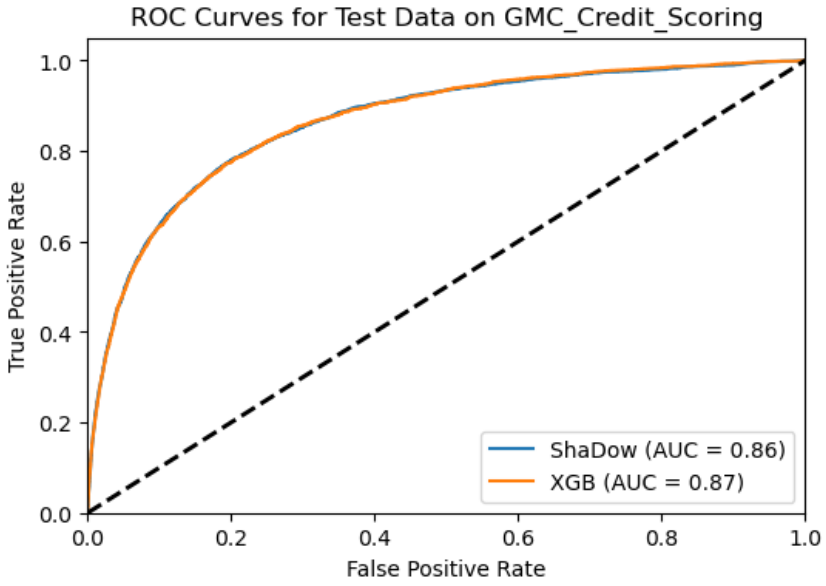

Figure 3.2: ROC Curves of XGB and ShaDow models on GMC_credit_scoring

In Figure 3.2, you can see the example roc curve used in this study.

3.2.1 Custom Evaluation Metrics

As discussed, we need custom approaches and metrics to analyze the results.

When we evaluate classification models, most of the time, we check more than one metric; for example, even if our primary metric is F1-Score, we still check the AUC Score; if the model with the highest F1 results in poor AUC, we can consider taking another model as the best model. In our case, we can't check 559 models one by one. We should have some overall score to evaluate all models simultaneously. To achieve this, we are using a weighted average

of accuracy, F1-Score, and AUC Score:

$$OverallScore = 0.05 * Accuracy + 0.475 * F1 + 0.475 * AUC$$

As accuracy is not very informative in imbalance classification cases, we gave the smallest weight to it but didn't completely disregard it. Therefore, we give it a minimal weight in the formula and equally divide the weights of F1-Score and AUC.

Some classification tasks are naturally easy, even if they are imbalanced, and classification baseline models show good results. Since the baseline model is enough to do the task, we don't need to analyze these cases for different balancing techniques. Therefore, based on baseline model performance, we need a flag variable to decide whether the task is easy or hard. We used previously defined OverallScore to achieve the goal:

$$\text{Easy Natural Flag} = \begin{cases} 1 & \text{if OverallScore} > 0.9 \\ 0 & \text{otherwise} \end{cases}$$

Next, we should define overfitting cases to spot overfitted models because even if a model has excellent test results, we can't use it if it is overfitted. Thus, we defined the overfitting flag as follows: If the percentage difference between the train and test overall score is greater than or equal to 10%, then the model is overfitted.

$$\text{Overfitting Flag} = \begin{cases} 1 & \text{if TrainScore} >= \text{TestScore} * 1.1 \\ 0 & \text{otherwise} \end{cases}$$

Since we have many models to compare on multiple datasets, it would be better to rank models for each dataset based on each evaluation metric: AUC, F1, precision, Recall, and OverallScore. With ranks, we can do a couple of analyses, like how many times a model is ranked first, what the median rank of each model is, etc.

Furthermore, from each rank, we can generate a rank score as the inverse of rank to assign a score to each model in each task:

$$rank\_score = \frac{1}{rank}$$

3.3 Classifiers without any imbalance technique

3.3.1 XGBoost

XGBoost is an open-source library that implements gradient-boosted decision trees in a scalable and optimized manner.

Boosting involves iteratively training a weak learner, incorporating it into the ensemble model, and updating the training dataset to more effectively consider the strengths and weaknesses of the current ensemble model while training the next base model.

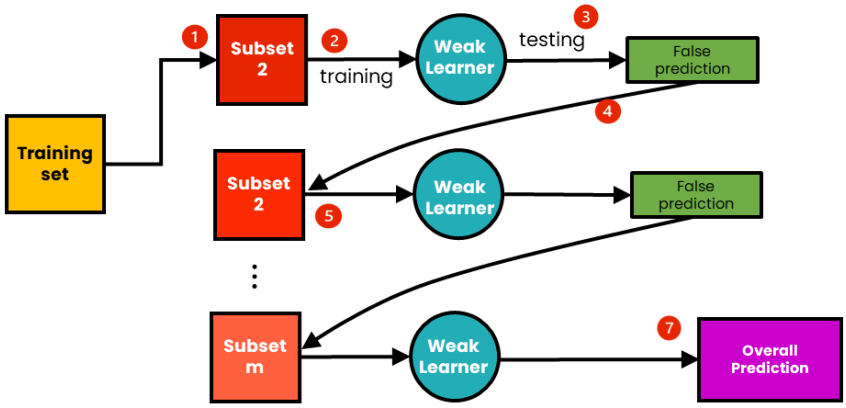

Figure 3.3: The Process of Boosting

XGBoost was first created as a research project 2016 by Tianqi Chen and Carlos Guestrin [28]. Since then, it has become the standard method for traditional supervised tasks on structured data. For many common regression and classification problems, it yields state-of-the-art results.

3.3.2 Random Forest

As its name suggests, random forests comprise numerous separate decision trees that work together as a group [29]. To create an ensemble model that is more robust than the individual models that make it up, Random Forest uses ”bagging” (Bootstrap Aggregation). The final prediction of the model is derived from the aggregation of the class predictions produced by each tree in the random forest. The basic idea behind bagging is to ”average” the predictions of multiple independent models to produce a model with a smaller variance. In actuality, though, we cannot construct completely independent models due to the large amount of data needed.

Therefore, to fit almost independent models, we rely on bootstrap samples' representativity and independence properties.

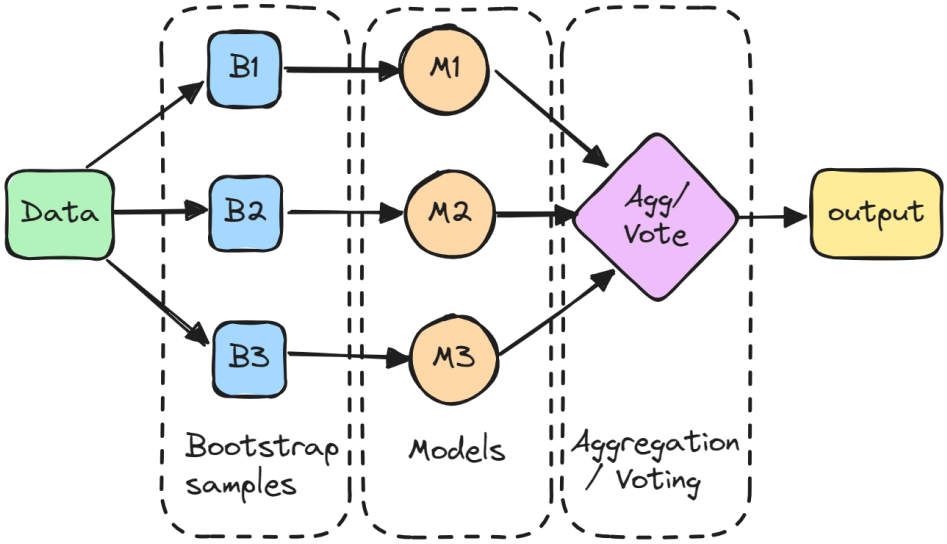

Figure 3.4: The Process of Bagging

3.3.3 Logistic Regression

Logistic regression is one of the most common algorithms used to solve binary classification problems [30]. The logistic regression model can be expressed as:

$$\hat{y} = \sigma(w^T x + b),$$

where x is an n dimensional vector of features, w is weight vector, b is bias and $\sigma(z)$ is sigmoid function:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

3.4 Resampling Methods

Resampling strategies are employed to readjust the sample space for an unbalanced dataset to mitigate the impact of the skewed class distribution during the learning process. Resampling approaches offer greater versatility as the choice of classifier does not influence them. Resampling approaches can be categorized into three groups based on the approach used to balance the distribution of classes [8].

1. Over-sampling methods: Addressing the negative effects of imbalanced distribution by generating additional samples for the minority class. There are two commonly used techniques for generating synthetic minority samples: randomly replicating the minority samples and using SMOTE [9].

2. Under-sampling methods: Addressing the negative effects of an imbalanced distribution by removing the intrinsic samples in the dominant class. The most straightforward and efficient approach is Random UnderSampling (RUS), which randomly removes samples from the majority class [31].

3. Hybrid methods: These are a combination of the over-sampling method and the under-sampling method [8].

3.4.1 Random Oversampling

Random oversampling duplicates existing examples from the minority class with replacement. Each data point in the minority class is equally likely to be duplicated [1].
It is easy to implement random oversampling, and it can lead to better model performance for the minority class, reducing bias towards the majority class and potentially improving overall accuracy. However, there are some risks of using this technique. First, replicating instances of the minority class can lead to overfitting, where the model learns the specific details of the oversampled instances rather than generalizing from the broader dataset [32]. Furthermore, the natural class distribution is changed by oversampling, which could cause the model to make predictions based on incorrect assumptions about how frequently minority classes occur.

3.4.2 Random Undersampling

Random Undersampling is the opposite of Random Oversampling. This method seeks to select and remove samples from the majority class randomly, consequently reducing the number of examples in the majority class in the transformed data [33].
In random under-sampling, vast quantities of data are discarded. This can be problematic since it can be more challenging to learn the decision border between the minority and majority occurrences when such data is lost, which would lower classification performance [34].

3.4.3 SMOTE

The SMOTE generates synthetic data for the minority class. This technique was introduced by Chawla et al. in 2002 [9]. It works by randomly picking a point from the minority class and computing the k-nearest neighbors for this point. The synthetic points are added between the chosen point and its neighbors.

SMOTE reduces the risk of overfitting because it generates synthetic samples that are not identical to the original samples. However, synthetic samples can introduce noise and outliers in the data. Furthermore, applying SMOTE can be costly if the dataset is large.

3.4.4 ADASYN

ADASYN (Adaptive Synthetic Sampling) [10] like SMOTE [9] generates new samples by interpolation. The significant difference is that ADASYN generates samples next to the original ones wrongly classified by a KNN classifier. In other words, minority samples with more majority class samples in the neighborhood have greater weight and more synthetic samples are generated for those points.

The dataset obtained after applying ADASYN will not only ensure a balanced representation of the data distribution. Still, it will also compel the learning algorithm to prioritize the complex examples that are challenging to learn. Contrary to the SMOTE technique, this represents a significant distinction where an equal amount of synthetic samples are produced for every minority data instance [10].

However, due to the adaptability nature of ADASYN, the precision of the models may be lowered. This results from increased data generation in neighborhoods with a large concentration of instances from the majority class. This could lead to many false positives as the synthetic data produced could resemble the majority class data quite a bit [35].

3.4.5 Tomek Links and SMOTE-Tomek

Given two examples E_i and E_j belonging to different classes, and $d(E_i, E_j)$ is the distance between E_i and E_j. A (E_i, E_j) pair is called a Tomek link if there is not an example E_l, such that $d(Ei, E_l) < d(E_i, E_j)$ or $d(E_j, E_l) < d(E_i, E_j)$ [36].

Removing the occurrences of the majority class in each Tomek link increases the gap between the two classes, making the classification process easier.

Although Tomek links might reduce the imbalance in the dataset, they may not consistently achieve an exact balance. One drawback of utilizing Tomek links is that excessively deleting instances from the majority class can result in underfitting and a decline in the overall classification performance.

SMOTE-Tomek [12] is a hybrid method in which data is oversampled with SMOTE and then undersampled using Tomek Links.

3.5 Algorithm-level Methods

3.5.1 Cost-Sensitive Learning

Many machine learning algorithms can be adjusted by modifying the existing training loss function to consider the imbalanced distribution of the classes. This can be accomplished by assigning distinct weights to both the majority and minority classes. The weight disparity will impact the classification of the classes during the training stage. The primary objective is to apply a penalty for the incorrect classification of the minority class by assigning a greater weight to that class while simultaneously decreasing the weight assigned to the majority class. For example, consider the following formula of log loss for Logistic Regression [37]:

$$L(y, p) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i)\right],$$

where N is the number of observations, $y_i$is the actual label of the ith observation, p_i is the predicted probability that the ith observation is of the positive class. Adjusted log loss formula with class weights given below:

$$L(y, p, w) = -\frac{1}{N}\sum_{i=1}^{N}\left[w_0 \cdot (1 - y_i) \cdot \log(1 - p_i) + w_1 \cdot y_i \cdot \log(p_i)\right]$$

where N is the number of observations, $y_i$is the actual label of the ith observation, p_i is the predicted probability that the ith observation is of the positive class, w_0 is the weight for class 0, $w1$ is the weight for class 1.

In our cases, cost optimization is achieved by setting `class_weight = ́balanced´` for Logistic Regression and Random Forest. For XGBoost, we used `scale_pos_weight` argument.

3.5.2 EasyEnsemble

The EasyEnsemble approach is specifically developed to address the limitations of conventional undersampling methods when dealing with class-imbalance learning. EasyEnsemble independently selects multiple subsets from the majority class that are commonly disregarded in undersampling. These subsets include potentially valuable information that can enhance the model's performance. A classifier is trained by employing the minority class examples and a selected subset from the majority class for each subset sampled. The methodology uses AdaBoost to train the classifier. By training classifiers on various subsets, the outputs of these classifiers are merged to create a final ensemble classifier. This ensemble classifier utilizes the information derived from multiple subgroups of the majority class to enhance the accuracy of predictions.

3.5.3 BalancedBagging

The Balanced Bagging Classifier trains multiple classifiers on different subsets of the training data. It combines their predictions like the Bagging Classifier and uses random undersampling in each subgroup to achieve balanced learning [38]. This approach reduces bias towards the majority class and enhances the model's performance on the minority class. The method utilizes bagging and resampling techniques to distribute evenly across the minority and majority classes. However, the Balanced Bagging Classifier does have certain constraints. One inherent constraint is that it requires a substantial quantity of minority-class samples to be effective. If the minority-class sample quantity is insufficient, resampling strategies may fail to produce a sufficient number of different examples for training the classifiers.

3.5.4 SMOTEBagging

SMOTEBagging is similar to BalancedBagging with undersampling; the difference is that instead of undersampling, we use SMOTE to balance each subset of the data [38].

3.5.5 SMOTEBoost, RUSBoost, and RAMOBoost

RAMOBoost [14], SMOTEBoost [15], and RUSBoost [17] are hybrid algorithms that integrate sampling approaches with boosting to tackle class imbalance in machine learning. SMOTEBoost uses the SMOTE technique during each boosting step to achieve data balancing [15].

The benefit of this strategy is that while regular boosting assigns the same weight to all misclassified data, SMOTE results in additional instances of the minority class at each boosting iteration.

Similarly, RUSBoost achieves the same goal by performing random undersampling at each boosting iteration instead of SMOTE [17].

RAMOBoost uses an adaptive synthetic data generation method with a ranked minority oversampling strategy [14]. The Ranked Minority Oversampling (RAMO) technique in RAMOBoost functions dynamically, creating synthetic instances for the minority class, considering the class ratios of the nearest neighbors in the underlying data distribution. RAMOBoost seeks to alter the decision boundary to include difficult-to-learn instances from the majority and minority classes. The approach enhances the model's capacity to grasp complex decision boundaries and improve classification accuracy by producing additional synthetic examples for these problematic scenarios.

3.6 Anomaly Detection Algorithms

Anomaly detection, also known as outlier detection, identifies observations, occurrences, or data points that are different from the typical, standard, or expected patterns and inconsistent with the remainder of a dataset. We can conclude that anomaly detection problems are naturally imbalanced classification, but can we treat minority classes as outliers in all types of imbalanced classification tasks? In this thesis, we also tested the effectiveness of 2 anomaly detection algorithms in classifying imbalanced datasets. The idea is simple: we enhance the data by adding the results of the anomaly detection algorithm to the original dataset as a feature. We used this stacking method with three versions:

1. Using LOF (Local Outlier Factor) results as a new feature.
2. Using IF (Isolation Forest) results as a new feature.
3. Using both LOF and IF results as new features.

The following subsections will discuss LOF and IF and the main ideas behind these algorithms.

3.6.1 Isolation Forest

The Isolation Forest [39] is a type of unsupervised machine learning algorithm specifically designed to detect anomalies. As its name suggests, Isolation Forest is an ensemble approach that shares similarities with random forest. Put simply, it calculates the average of the predictions made by many decision trees to determine the final anomaly score for a particular data point. Isolation Forest differs from conventional anomaly detection algorithms by immediately attempting to isolate abnormal data points rather than first defining what is considered ”normal” and then identifying everything else as anomalous.

3.6.2 Local Outlier Factor

The local Outlier Factor (LOF) is an unsupervised technique to detect and identify outliers within a dataset [40]. Samples with a significantly lower density than their neighbors are classified as outliers. The ratios between the surrounding area's local densities and a point's neighbors determine its LOF score [40]. It considers the relative density of data points.
The point may be an outlier if the ratio is significantly less than 1, which would signal that it is in a less dense area than its neighbors. In contrast, a point is seen as deep within a cluster and not an outlier if the ratio is near 1. By examining this ratio, LOF can determine an object's relative isolation level within its local neighborhood, hence offering a more sophisticated comprehension of dataset outliers. LOF functions well when there is variation in the data density across the dataset.

3.7 Novel Approaches using Ensemble Learning

In this section, we will introduce new approaches to imbalanced classification using ensemble learning.

3.7.1 ShaDow: Downsampling with Shallow Decision Trees

When working with an imbalanced dataset, sometimes we can pre-filter some of the majority class examples in the Data Understanding step. Consider the following example: We are building a churn model where we predict whether customers will churn within the next month or not. We have a 5% percent churn rate among all of our customers. However, when we analyze

the data, it turns out that we can pre-filter customers who have made transactions in the current month because there is no chance that they will churn in the next month. After this filter, the churn rate among customers is 20%, which is not a severe imbalance.
What if we can detect patterns like this automatically before training our model and then train it on a filtered and more balanced dataset? This is what ShaDow tries to achieve with Shallow Decision Trees. The idea is to filter the cases where Shallow Tree is sure they are from the majority class.
However, this approach could lead to overfiltering and, therefore, false negatives since it highly depends on the minority threshold.

3.7.2 DCBoost: Double-Check Boosting

We are introducing a new approach to handle the class imbalance problem, DCBoost: Double-Check Boosting. The algorithm works as follows: First, we build a biased boosting model towards the minority class by setting class weights. Since this model is biased towards the minority class, Recall of this model will be higher; we will have more false positives and fewer false negatives. That means we can rely on this model, whose predictions are majority class (negative class). After training the first model, we filter the cases with predicted positive (minority) classes; then we feed them to our next, fair model. Thus, for the cases where the biased model predicted a minority class, the second model makes the final decision. This approach resembles diagnosing a positive diagnosed patient a second time to make sure of the result. That's where the name Double-Check comes from.
The DCBoost algorithm has two main limitations. First, the prediction probabilities of the final model will be biased since half of them come from the biased model and the other half from the other model, which is trained on differently distributed data. To overcome this issue, we can adjust the probabilities of two models or apply calibration. Additionally, if there are very few minority class examples in data and the first model performs well, the second model is forced to train on the with few rows. This can result in overfitting. This issue can be solved by checking if there is enough data in the second model. If there is not, we can augment the dataset from the original dataset where the first model is not very confident

3.7.3 SelfBoosting

When we analyze the models with cost optimization, we can observe that these models increase Recall dramatically, but the F1 score does not improve significantly. The idea of SelfBoosting is to train two boosting models; the first model uses class weights to balance classes, the predictions of this model are stacked to the original dataset as a new column, and enriched data is fed to the second model to train the final model. The goal of SelfBoosting is to learn from the predictions of the first model and improve them.

3.7.4 PSM: Parallel Stacked Models

In SelfBoosting, we used the predictions of the cost-optimized model as features in the second model. What if we train more than one model that handles class imbalance separately and then use their predictions as features in the final model? This approach of combining the predictions of multiple models is known as stacking. In this thesis, we will use stacking with multiple class imbalance algorithms.

PSM aims to integrate diverse imbalance solutions by combining multiple models that handle imbalance differently. One disadvantage may be the performance of the sub-models, as the performance of the primary model heavily depends on them. Furthermore, this model needs careful tuning and sub-model selection. In our experiments, we used XGBoost with class weights, EasyEnsemble, SMOTEBagging, RUSBoost, and BalancedBagging models.

3.7.5 SSM: Sequential Stacked Models

The SSM (Sequential Stacked Model) is a stacked model like the PSM, but unlike PSM, SSM trains models sequentially, meaning that the predictions of each model are fed to the next model as input.

Like PSM, SSM needs careful selection models; another disadvantage of SSM is error propagation. If one of the models performs poorly or learns incorrect patterns, this error can propagate to the subsequent models.

Chapter 4

Results and Analysis

In this section, we will analyze the results and try to derive useful insights. To achieve this goal, let's start by asking questions. The first question that comes to mind about the imbalanced classification task is whether it is worth trying balancing techniques. Do they help to achieve better results?

Table 4.1: Comparison of Best Baseline and Best Model Performances

Best Baseline	Dataset	Easy	Best Model	Score Increase %	F1 Increase %
XGB	Adult	0	DCBoost	1.519	6.844
XGB	Churn_Ecom	0	DCBoost	12.623	78.791
XGB	CoverType	0	DCBoost	5.342	14.028
XGB	GMC	0	DCBoost	4.350	36.274
XGB	Nursery	1	SSM	7.986	18.611
XGB	PTB_Online	0	ROS_XGB	2.518	7.079
XGB	Taiwan	0	RUS_XGB	3.949	12.987
XGB	Wine	0	ROS_XGB	4.315	12.873
XGB	cc_fraud_05	1	SelfBoostingXGB	0.025	0.000
XGB	cc_fraud_1	1	IF_XGB	0.004	0.000
XGB	cc_fraud_2	1	IF_RandomForest	0.796	2.597
XGB	cc_fraud_3	1	SMOTEBoost	0.703	2.126
RandomForest	cc_fraud_4	1	LOF_IF_XGB	0.208	0.148

To answer the question, let's compare the results of the best baseline model and the best model for each task. You can see this information on the Table 4.1. The first thing we no-

ticed in the table is that nearly half of the tasks were easy. Therefore, we can't see significant improvement in those cases. Of course, we also consider that we have nine different datasets, and 2 of them were easy to predict naturally. In our following analyses, we will always filter out these easy cases. Another observation is that XGB is the best baseline in every task except 'cc_fraud_4'. In Figure4.1, we can see that the results of the AUC scores of the XGB and RandomForest are the same.

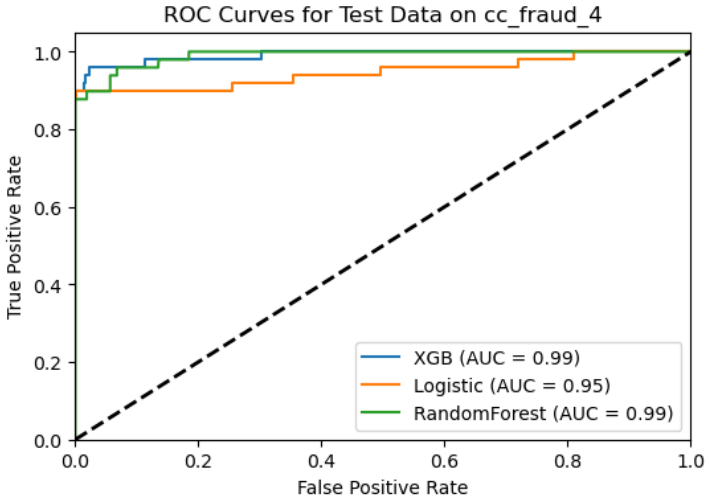

Figure 4.1: ROC Curves of baseline models on cc_fraud_4

In Table 4.2, we can see that there is only a 0.04 point difference between an overall score of XGB and RandomForest. These results suggest that boosting models can give good results

Test Score	XGB	RandomForest	Logistic
AUC	0.990	0.989	0.946
Accuracy	0.994	0.995	0.994
F1	0.926	0.935	0.926
Overall Score	0.960	0.964	0.939

Table 4.2: Baseline Model Test Scores for cc_fraud_4

even if the classes are imbalanced. If we go back to the question to see the improvement of balancing techniques, we first filter out the naturally easy problems. We can see that, on average, balancing techniques show a 4.95 improvement in the overall score and a 24.13% improvement in the F1 Score. These results show that balancing techniques can improve our model results, which is worth trying.

Now, let's analyze the relationship between minority percent and score increase. In Figure4.2, the trend line indicates the negative correlation between minority class percent and score increase, which is intuitive.

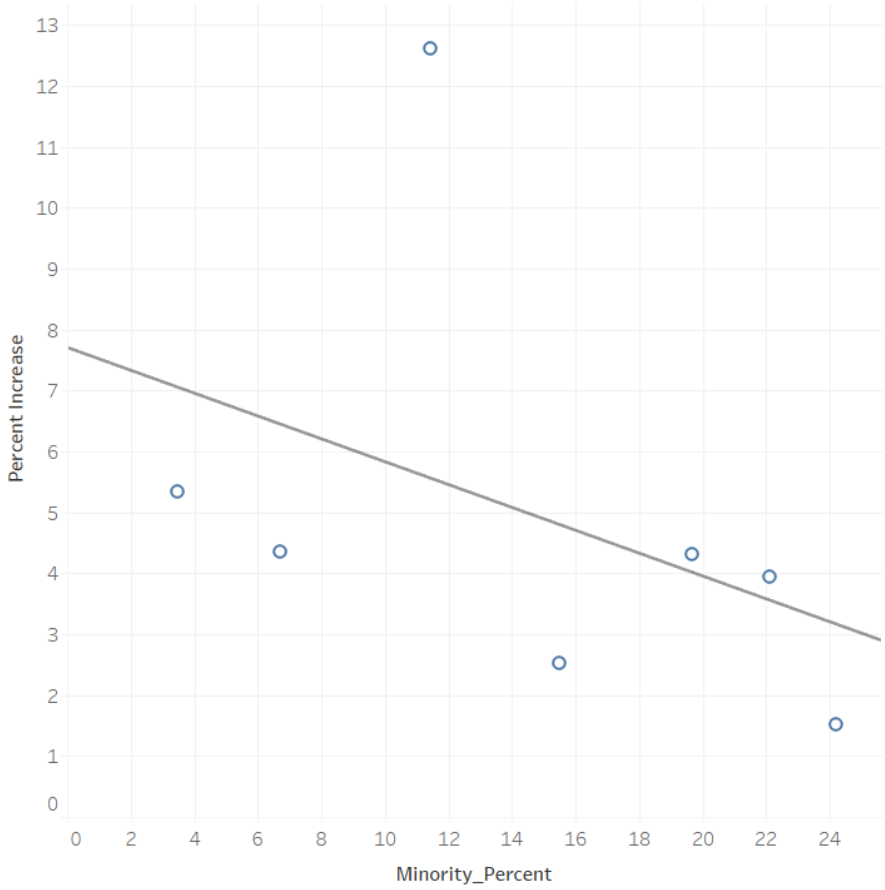

Figure 4.2: Scatterplot of Minority Percent vs Score Increase

Note that we only have seven samples here; more accurate results could be achieved with more samples. In that case, we could also check how the performance of each of the individual models changes when the minority class percentage changes.

Now, let's analyze the best models for different metrics (Overall Score, F1, AUC, and Recall).

4.1 Best models based on Overall Score

Out of 7 problems, DCBoost showed the best results in 4 based on the Overall Score we defined in 3.2.1. You can see these results in Figure 4.3

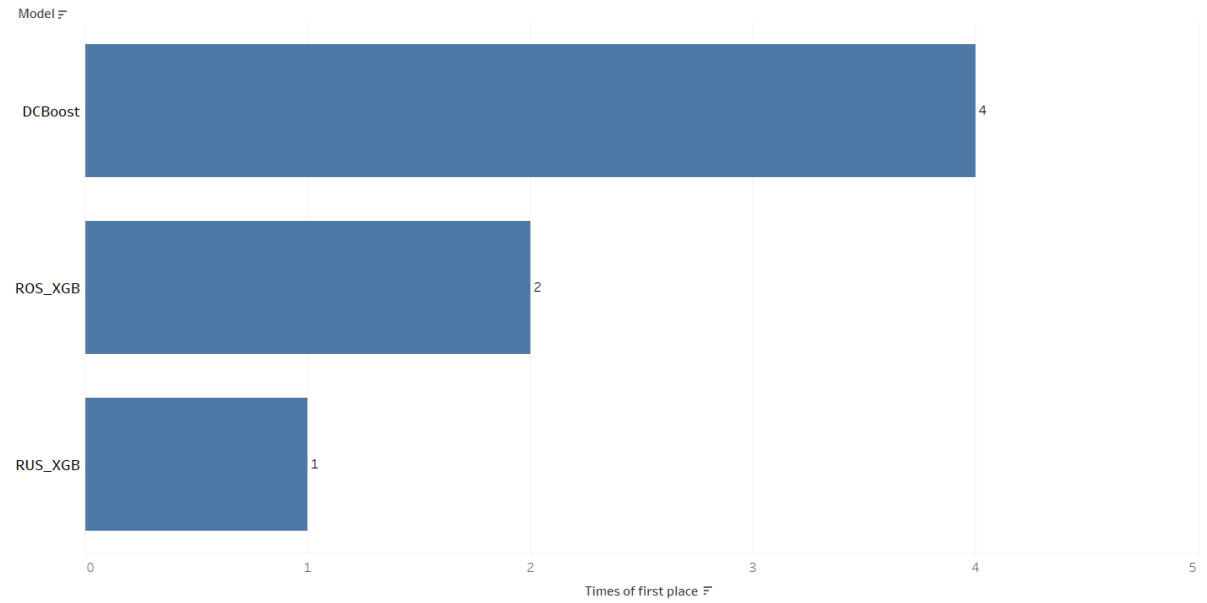

Figure 4.3: How many times each model took the first place based on Overall Score

We observe that DCBoost is the best if we compare the count of first places; we can also compare the average Rank Scores of the models based on Overall Score. In Figure 4.4 we can see that DCBoost has the greatest result, which is nearly 50% more compared to the second model in the graph (ROS_XGB).

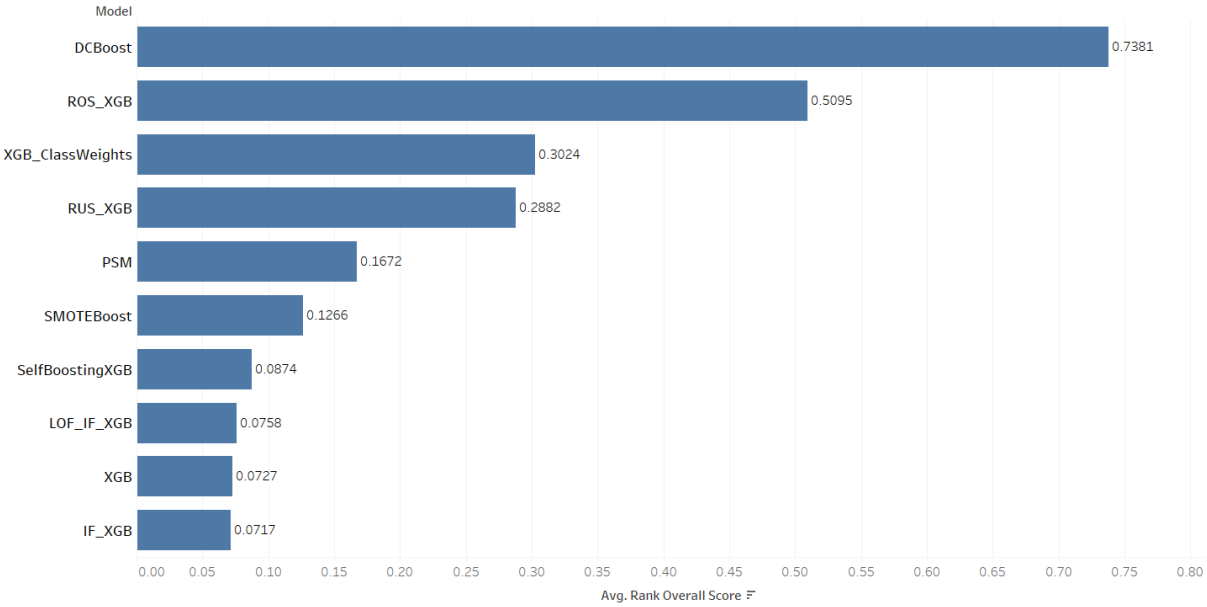

Figure 4.4: Top 10 models with the highest Average Rank Overall Score

Furthermore, let's compare each model's result with the best baseline model result in each task and see how much improvement each model made. In Figure 4.5, we can see the top 20 models

with the highest increase in Overall Score.

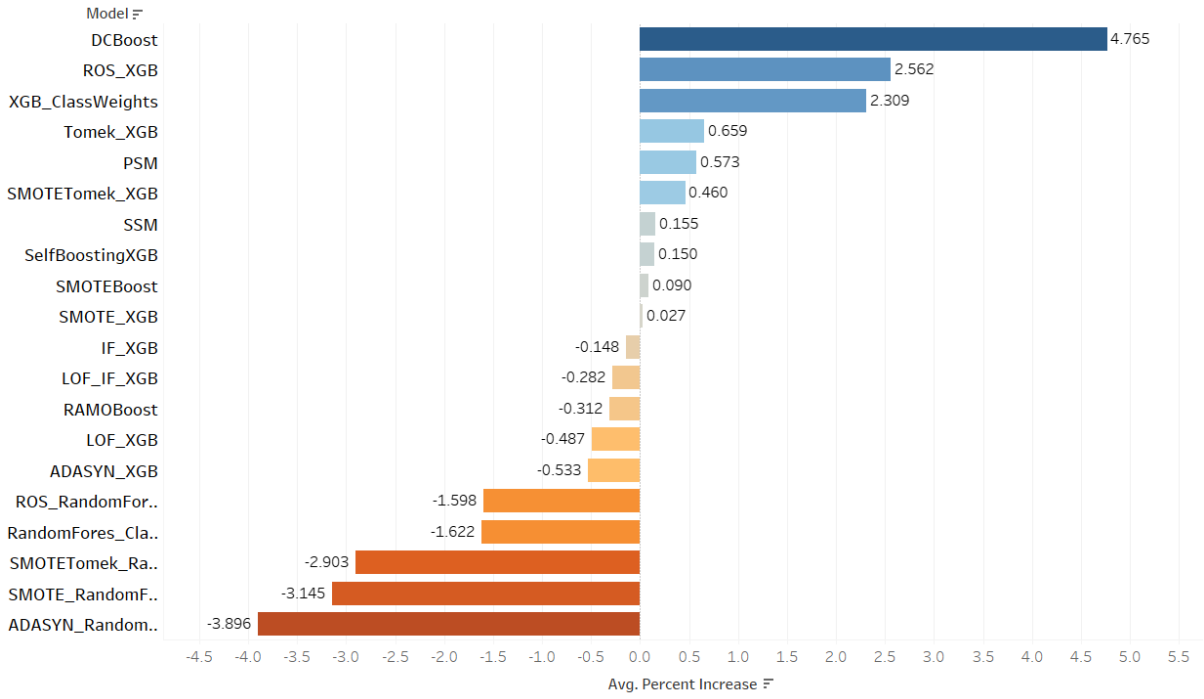

Figure 4.5: Top 20 models with the highest percentage increase in Overall score.

We can derive two important conclusions from Figure 4.4. Firstly, not all imbalance techniques improve the performance; they may also decrease it. Secondly, DCBoost shows the best results regarding increase in overall score.

In summary, DCBoost performed best among 42 models based on the Overall Score.

4.2 Best models based on F1 Score

As we discussed in 3.2, the F1 metric might be the best metric choice for dealing with class imbalance. In this section, we will analyze the best models based on the F1 Score. In Figure 4.6, each model is compared based on how many times they took first place according to the F1 Score. It is the same as Figure 4.3, but ranks are given based on the F1 Score this time.

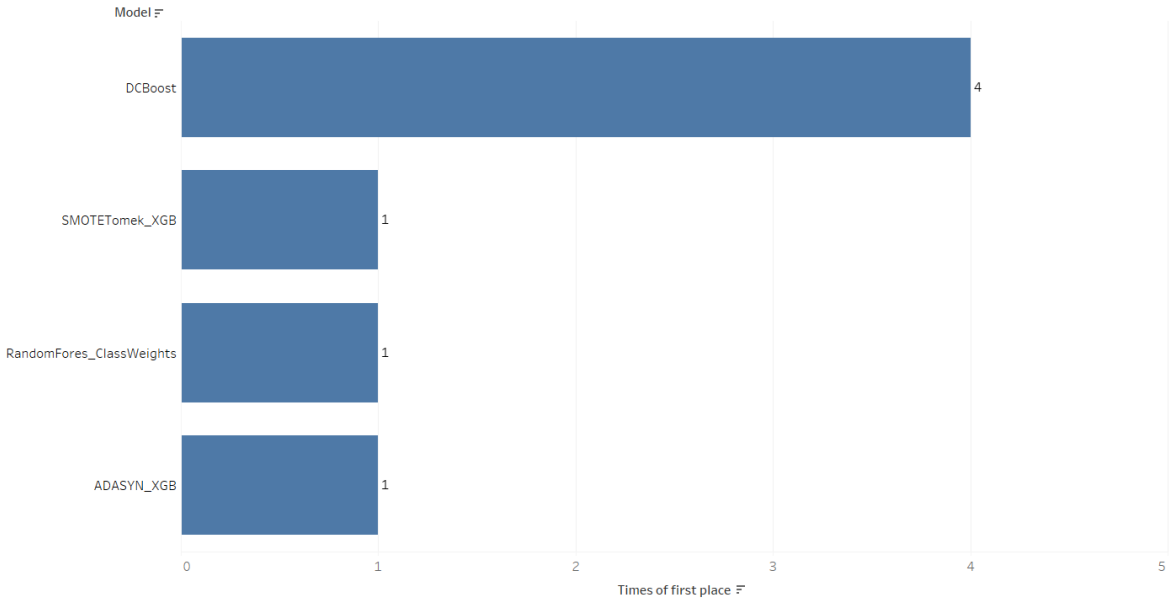

Figure 4.6: How many times each model took the first place based on F1 Score

We see that DCBoost still shows the best results in 4 of the tasks out of 7. Let's compare the results with other analyses as we did in the previous section.

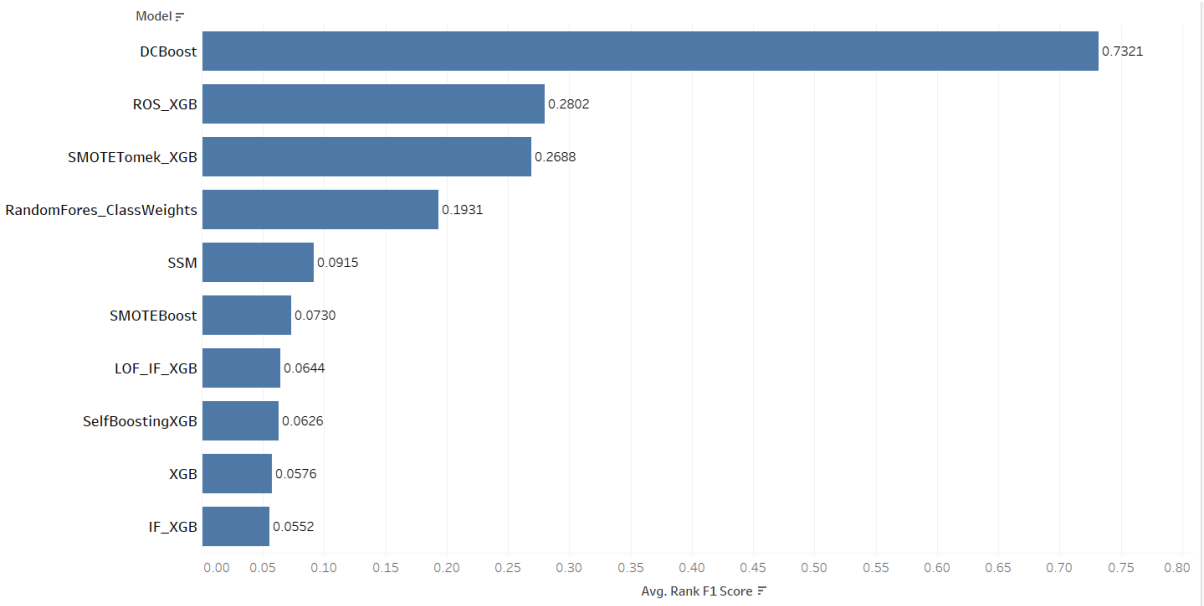

Figure 4.7: Top 10 models with the highest Average Rank Score F1

In Figure 4.7, we can see the top models with the highest rank scores based on the F1 score. The performance of DCBoost is even better here compared to the Overall Score (see 4.4).
In Figure 4.8, we can see that, on average, DCBoost has shown a 24.22% percent increase in F1 score compared to the best baseline model.

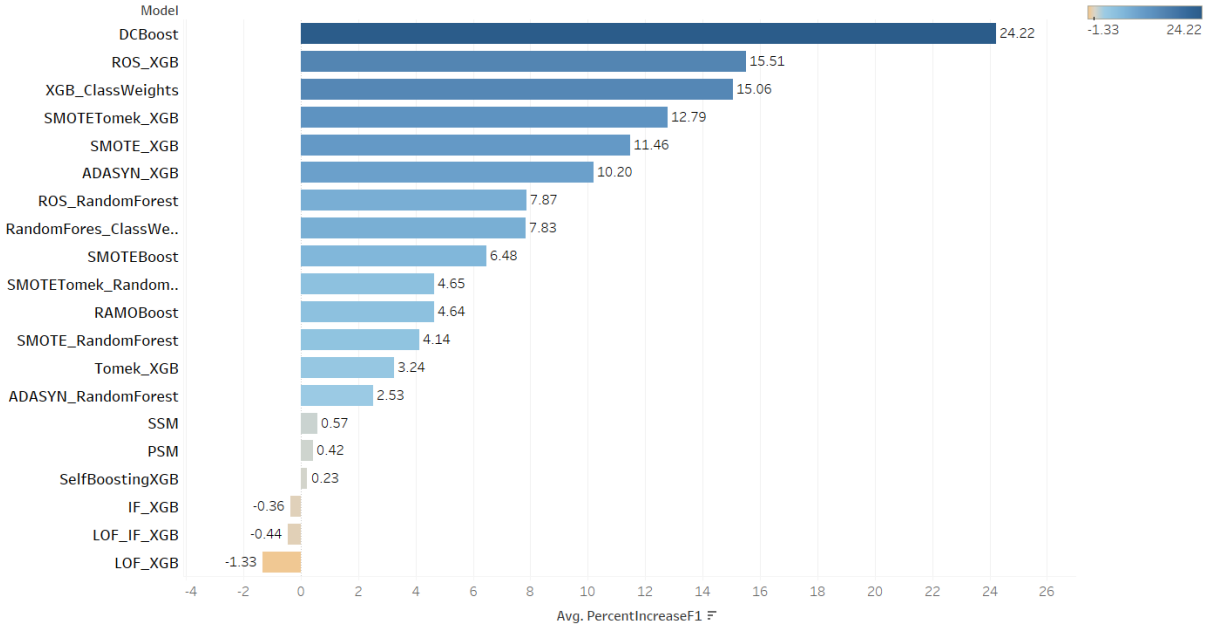

Figure 4.8: Top 20 models with the highest percentage increase in F1 score.

4.3 Best models based on AUC Score

To analyze the best models based on AUC, let's visualize the average rank score of each model based on AUC. In Figure 4.9, cost-optimized XGBoost showed the best result.

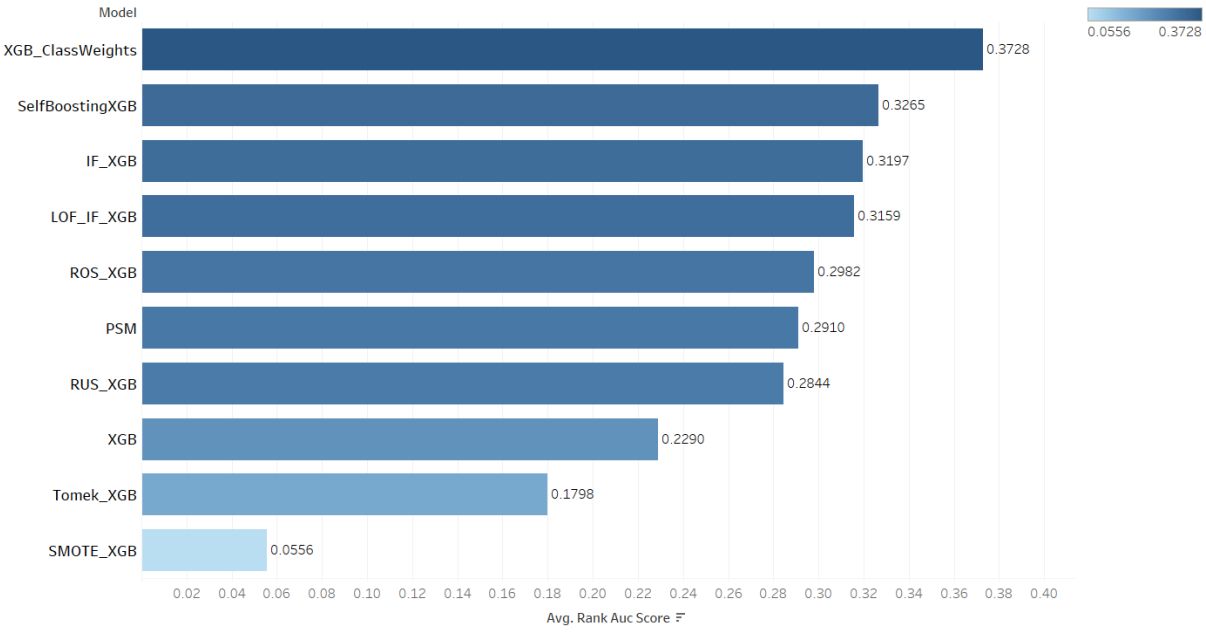

Figure 4.9: Top 10 models with the highest Average Rank Score AUC

In the previous chapter, we talked about the limitations of the DCBoost, and we underlined

that the probability predictions of DCBoost are biased and, therefore, can result in poor AUC results. Figure 4.9 also confirms this statement since DCBoost is not among the best models regarding the AUC score. Furthermore, we can observe that the results of individual models are very close to each other; let's plot the average percent increase of AUC for each model to see if the improvements are consistent.

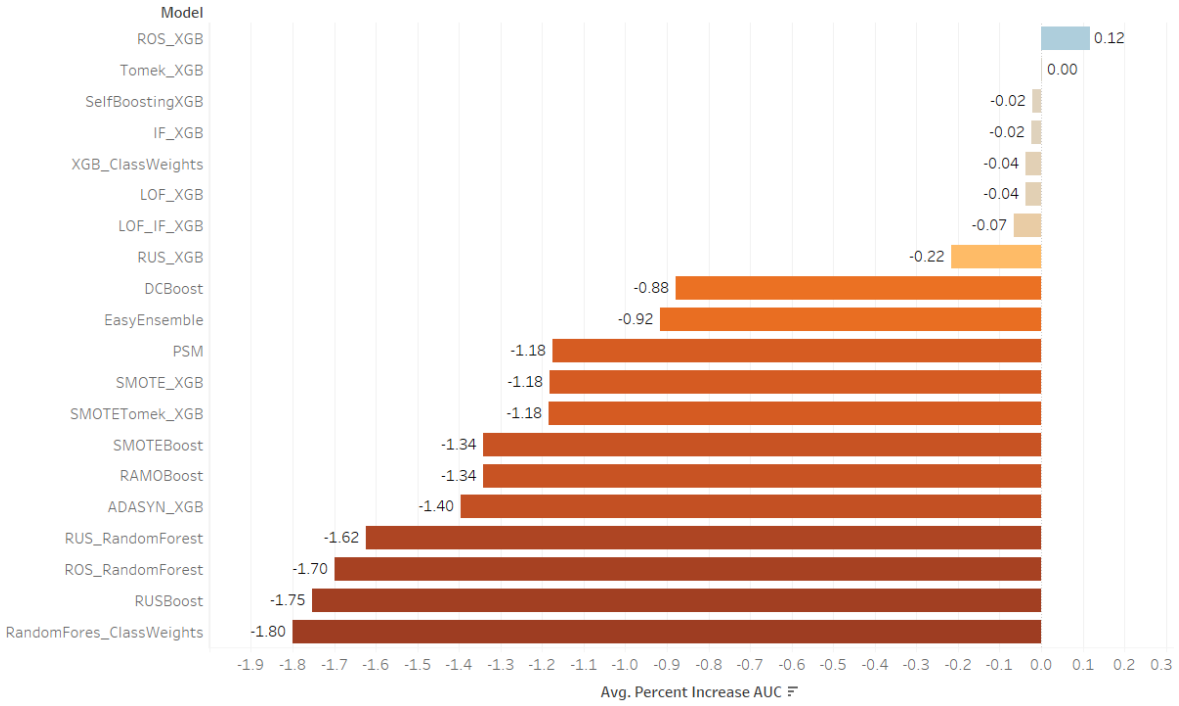

Figure 4.10: Top 20 models with the highest percentage increase in AUC

From Figure 4.10, we can see that none of the models showed significant improvement to AUC on average. This result suggests that the imbalanced techniques we analyzed are not generally effective on AUC.

4.4 Best models based on Recall

In Figure 4.11, we can see the average rank score of each type of model based on Recall. Intuitively, the Cost Optimization technique resulted in the best recall score since we make the model biased towards the minority class when we assign greater weight to it.

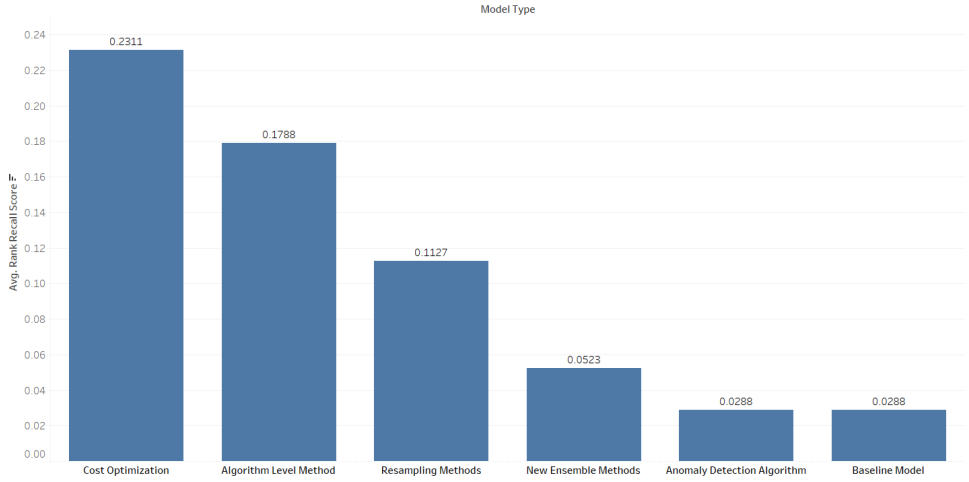

Figure 4.11: Average Rank Scores of technique types

4.5 Analysis of DCBoost

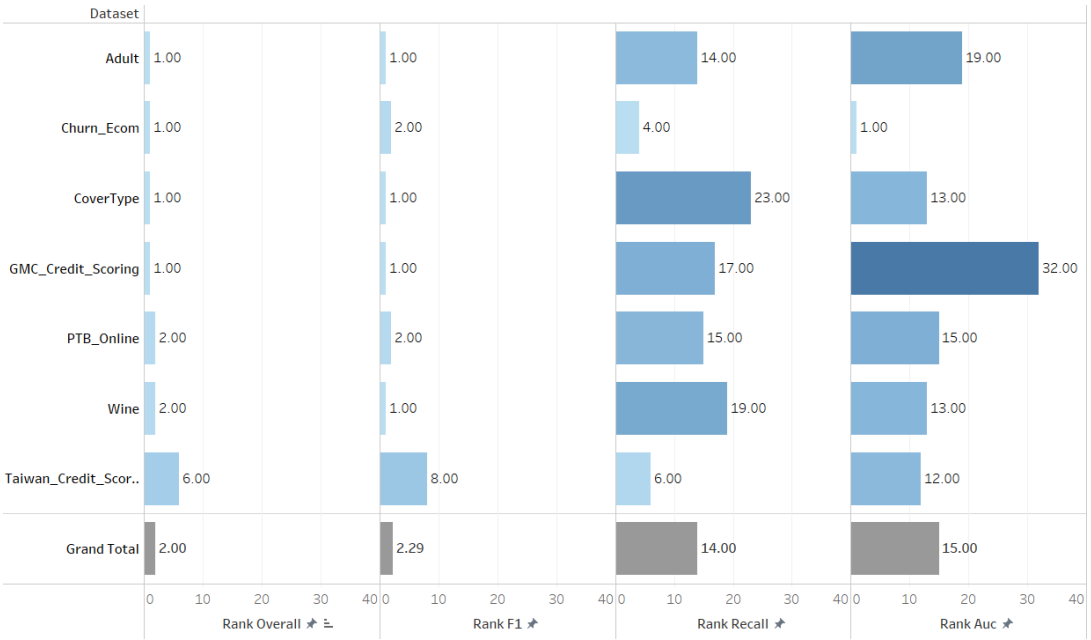

Figure 4.12: DCBoost Ranks on each dataset for main metrics

Let's analyze the results of DCBoost deeply since it is a new approach with good results. In Figure 4.12, we can see the rank of DCBoost for each problem and each metric. The average rank is also shown as 'Grand Total.' Based on the figure, we can make the following conclusions:

1. DCBoost outperforms other algorithms in most of the tasks in terms of F1 and Overall Score.

2. As discussed, DCBoost doesn't perform well regarding AUC because of its biased predictions. This issue can be fixed by adjusting the prediction probabilities of the model.

3. DCBoost doesn't perform well in terms of Recall.

4.6 Analysis of ShaDow

As discussed in the previous chapter, the results of the ShaDow are highly dependent on the nature of the dataset and the minority threshold. Therefore, we decided to analyze this model separately on one of the datasets where the results of the best baseline model and best models' results are not very high. The idea is to see if we can build a ShaDow model that can outperform the baseline and the best model. Therefore, we selected the GMC_credit_scoring dataset where the best baseline model is XGBoost, and the best performer is DCBoost. For the ShaDow model, we used the DecisionTree Classifier with a max depth of 6 as a tree classifier and the Cost Optimized XGBoost model as a main model. We set the minority threshold to 99%; in Figure 4.13, you can see the ROC curve for the XGB and Shadow model.

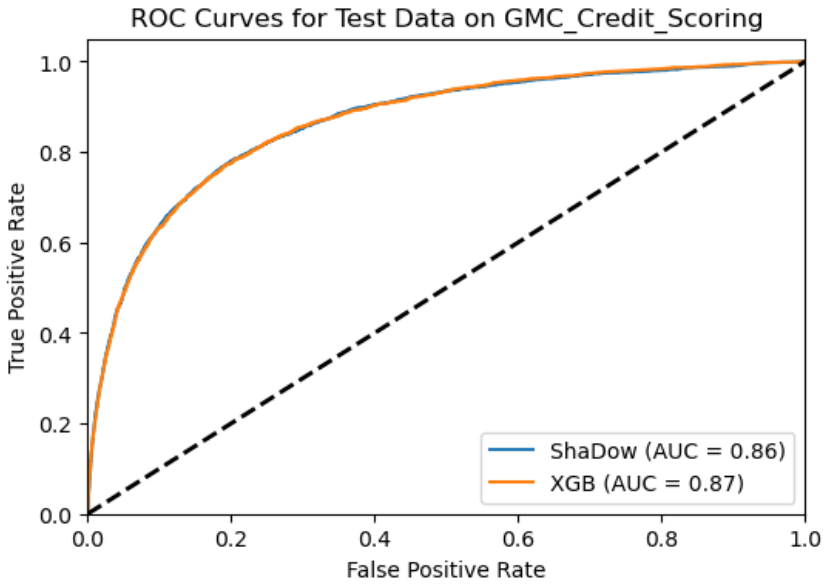

Figure 4.13: ROC Curves of XGB and ShaDow models on GMC_credit_scoring

In Table 4.3, the performances of ShaDow, DCBoost, and XGB are compared. ShaDow outperforms DCBoost and XGB in terms of F1-Score and Overall Score. The successful results of ShaDow also suggest that if we have enough domain-specific knowledge about the dataset and

	DCBoost	XGB	Shadow
AUC Score	0.824	**0.865**	**0.864**
Accuracy	0.859	**0.938**	0.877
F1 Score	0.392	0.288	**0.411**
Precision	0.274	**0.597**	0.301
Recall	**0.689**	0.190	0.645
Overall_Score	0.620	0.595	**0.650**

Table 4.3: Results of ShaDow, XGB, and DCBoost on GMC_credit_scoring

the prediction task, we can pre-filter our dataset to keep only hard-to-classify observations by removing already known majority class examples, resulting in more balanced data.

Chapter 5

Conclusion and Discussion

In this thesis, we analyzed and compared 21 class imbalance solutions, 5 of which are proposed in this work. We tested these techniques on 13 real-world datasets and compared the results. In this section, we will discuss the main findings and practical guidelines for practitioners facing the class imbalance problem.

If you have an imbalanced dataset, do not rush into using class imbalance techniques. Before applying imbalance solutions, it would be good to train some baseline models without any balancing. In some tasks, the baseline model performed perfectly, so we didn't use any balancing technique. Furthermore, even if your data can not be predicted very accurately by your baseline model, imbalance techniques do not always guarantee better results; they may even decrease the results of your baseline model.

If you don't get good results from your baseline model, try to analyze and understand your data better. Before applying class imbalance solutions, we can analyze the data to find some patterns of the majority class so that we can pre-filter these cases before training our model and achieving a more balanced dataset. The successful results of the ShaDow model also suggest that this pre-filter is worth spending time on.

If you decided to use imbalance techniques and your primary metric is F1, you can try DCBoost In our experiments, DCBoost outperformed other techniques based on F1 scores in most tasks.

If your primary metric is Recall, try the Cost Optimization technique. With the class weights, true positives have a more positive effect on cost function than true negatives, so these models will be biased towards positive class, resulting in better Recall.

If your main metric is AUC, there is a high chance that none of the tested imbalance solutions will make significant improvement. Our experiment showed that, on average, none of the imbalance solutions significantly affect the AUC.

If you want a well-performed model in general, you can try DCBoost. DCBoost outperformed other imbalance techniques in terms of custom-defined Overall Score.

5.1 Future Work

The results of this study provide the framework for several fascinating future studies. First, improvements to DCBoost can be made to achieve more reliable prediction probabilities and a more robust model for overfitting.
Additionally, the ShaDow model and other novel approaches can be analyzed deeply, and improvements can be made.
As we discussed before, this study only focused on binary classification tasks. In future work, multiclass classification problems with imbalanced problems can also be analyzed. Additionally, newly proposed methods can be improved to handle multiclass problems.
Lastly, future work could involve more real-world datasets with various minority percentages.

Bibliography

[1] Haibo He and Edwardo A. Garcia. "Learning from Imbalanced Data". In: *IEEE Transactions on Knowledge and Data Engineering* 21.9 (2009), pp. 1263–1284. DOI: 10.1109/TKDE.2008.239.

[2] Bartosz Krawczyk. "Learning from imbalanced data: Open Challenges and Future Directions". In: *Progress in Artificial Intelligence* 5.4 (Apr. 2016), pp. 221–232. DOI: 10.1007/s13748-016-0094-0.

[3] G. M. Weiss and F. Provost. "Learning When Training Data are Costly: The Effect of Class Distribution on Tree Induction". In: *Journal of Artificial Intelligence Research* 19 (Oct. 2003), pp. 315–354. ISSN: 1076-9757. DOI: 10.1613/jair.1199. URL: http://dx.doi.org/10.1613/jair.1199.

[4] Yanminsun, Andrew Wong, and Mohamed S. Kamel. "Classification of imbalanced data: a review". In: *International Journal of Pattern Recognition and Artificial Intelligence* 23 (Nov. 2011). DOI: 10.1142/S0218001409007326.

[5] Andrea Dal Pozzolo et al. "Learned lessons in credit card fraud detection from a practitioner perspective". In: *Expert Systems with Applications* 41 (Aug. 2014), pp. 4915–4928. DOI: 10.1016/j.eswa.2014.02.026.

[6] Thomas Davidson et al. "Automated Hate Speech Detection and the Problem of Offensive Language". In: *Proceedings of the International AAAI Conference on Web and Social Media* 11 (Mar. 2017). DOI: 10.1609/icwsm.v11i1.14955.

[7] José Antonio Sanz, Francisco Herrera Triguero, and Humberto Bustince. "A Compact Evolutionary Interval-Valued Fuzzy Rule-Based Classification System for the Modeling and Prediction of Real-World Financial Applications with Imbalanced Data". In: (2015). DOI: 10.1109/TFUZZ.2014.2336263. URL: http://hdl.handle.net/10481/64939.

[8] Guo Haixiang et al. “Learning from class-imbalanced data: Review of methods and applications”. In: *Expert Systems with Applications* 73 (2017), pp. 220–239. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2016.12.035. URL: https://www.sciencedirect.com/science/article/pii/S0957417416307175.

[9] Nitesh Chawla et al. “SMOTE: Synthetic Minority Over-sampling Technique”. In: *J. Artif. Intell. Res. (JAIR)* 16 (June 2002), pp. 321–357. DOI: 10.1613/jair.953.

[10] Haibo He et al. “ADASYN: Adaptive synthetic sampling approach for imbalanced learning”. In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. 2008, pp. 1322–1328. DOI: 10.1109/IJCNN.2008.4633969.

[11] Benjamin X. Wang and Nathalie Japkowicz. “Boosting Support Vector Machines for imbalanced data sets”. In: *Knowledge and Information Systems* 25.1 (Mar. 2009), pp. 1–20. DOI: 10.1007/s10115-009-0198-y.

[12] Gustavo Batista, Ana Bazzan, and Maria-Carolina Monard. “Balancing Training Data for Automated Annotation of Keywords: a Case Study.” In: Jan. 2003, pp. 10–18.

[13] Andrea Dal Pozzolo, Olivier Caelen, and Gianluca Bontempi. “When is Undersampling Effective in Unbalanced Classification Tasks?” In: Sept. 2015. ISBN: 978-3-319-23527-1. DOI: 10.1007/978-3-319-23528-8_13.

[14] Sheng Chen, Haibo He, and Edwardo A. Garcia. “RAMOBoost: ranked minority oversampling in boosting”. In: *Trans. Neur. Netw.* 21.10 (Oct. 2010), pp. 1624–1642. ISSN: 1045-9227. DOI: 10.1109/TNN.2010.2066988. URL: https://doi.org/10.1109/TNN.2010.2066988.

[15] Nitesh Chawla et al. “SMOTEBoost: Improving Prediction of the Minority Class in Boosting”. In: vol. 2838. Jan. 2003, pp. 107–119. ISBN: 978-3-540-20085-7. DOI: 10.1007/978-3-540-39804-2_12.

[16] Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. “Exploratory undersampling for class-imbalance learning”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39.2 (Apr. 2009), pp. 539–550. DOI: 10.1109/tsmcb.2008.2007853.

[17] Chris Seiffert et al. “RUSBoost: Improving Classification Performance when Training Data is Skewed”. In: Dec. 2008, pp. 1–4. DOI: 10.1109/ICPR.2008.4761297.

[18] Andrea Dal Pozzolo and Gianluca Bontempi. “Adaptive Machine Learning for Credit Card Fraud Detection”. In: 2015. URL: https://api.semanticscholar.org/CorpusID:111359889.

[19] Vladislav Rajkovic. *Nursery*. https://doi.org/10.24432/C5P88W. 1997.

[20] Jock Blackard. *Covertype*. UCI Machine Learning Repository. 1998. URL: https://doi.org/10.24432/C50K5N.

[21] Will Cukierski. *Give Me Some Credit*. https://kaggle.com/competitions/GiveMeSomeCredit. 2011.

[22] M. Fridrich and P. Dostál. “User Churn Model in E-Commerce Retail”. In: *Scientific Papers of the University of Pardubice, Series D: Faculty of Economics and Administration* 30.1 (2022), p. 1. DOI: 10.46585/sp28031105. URL: https://doi.org/10.46585/sp28031105.

[23] C. Sakar and Yomi Kastro. *Online Shoppers Purchasing Intention Dataset*. UCI Machine Learning Repository. https://doi.org/10.24432/C5F88Q. 2018.

[24] Paulo Cortez et al. *Wine Quality*. https://doi.org/10.24432/C56S3T. 2009.

[25] I-Cheng Yeh. *Default of Credit Card Clients*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C55S3H. 2016.

[26] Barry Becker and Ronny Kohavi. *Adult*. UCI Machine Learning Repository. DOI: https://doi.org/10.244 1996.

[27] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O’Reilly Media, Sept. 2019. URL: http://books.google.ie/books?id=HHetDwAAQBAJ&pg=PR2&dq=9781492032649&hl=&cd=1&source=gbs_api.

[28] Tianqi Chen and Carlos Guestrin. “XGBoost: A Scalable Tree Boosting System”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’16. San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 785–794. ISBN: 9781450342322. DOI: 10.1145/2939672.2939785. URL: https://doi.org/10.1145/2939672.2939785.

[29] L Breiman. “Random Forests”. In: *Machine Learning* 45 (Oct. 2001), pp. 5–32. DOI: 10.1023/A:1010950718922.

[30] Kuk Lida Lee Chao-Ying Joanne Peng and Gary M. Ingersoll. “An Introduction to Logistic Regression Analysis and Reporting”. In: *The Journal of Educational Research* 96.1 (2002), pp. 3–14. DOI: 10.1080/00220670209598786. eprint: https://doi.org/10.1080/00220670209598786. URL: https://doi.org/10.1080/00220670209598786.

[31] Muhammad Tahir et al. “A Multiple Expert Approach to the Class Imbalance Problem Using Inverse Random under Sampling”. In: June 2009, pp. 82–91. ISBN: 978-3-642-02325-5. DOI: 10.1007/978-3-642-02326-2_9.

[32] György Kovács. “An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets”. In: *Applied Soft Computing* (July 2019). DOI: 10.1016/j.asoc.2019.105662.

[33] Dedy Trisanto et al. “Effectiveness Undersampling Method and Feature Reduction in Credit Card Fraud Detection”. In: *International Journal of Intelligent Engineering and Systems* (2020). DOI: 10.22266/ijies2020.0430.17.

[34] Haibo He and Yunqian Ma. *Imbalanced Learning*. John Wiley and Sons, June 2013. URL: http://books.google.ie/books?id=CVHx-Gp9jzUC&printsec=frontcover&dq=1118074629&hl=&cd=1&source=gbs_api.

[35] J. A. Stanosheck et al. “Oversampling methods for machine learning model data training to improve model capabilities to predict the presence of escherichia coli mg1655 in spinach wash water”. In: *Journal of Food Science* 89 (1 2023), pp. 150–173. DOI: 10.1111/1750-3841.16850.

[36] Uma R. Salunkhe and Suresh N. Mali. “Classifier Ensemble Design for Imbalanced Data Classification: A Hybrid Approach”. In: *Procedia Computer Science* 85 (2016). International Conference on Computational Modelling and Security (CMS 2016), pp. 725–732. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2016.05.259. URL: https://www.sciencedirect.com/science/article/pii/S1877050916306093.

[37] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. Springer Series in Statistics, 2009. ISBN: 978-0-387-84857-0.

[38] Shuo Wang and Xin Yao. “Diversity analysis on imbalanced data sets by using ensemble models”. In: *2009 IEEE Symposium on Computational Intelligence and Data Mining*. 2009, pp. 324–331. DOI: 10.1109/CIDM.2009.4938667.

[39] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. “Isolation Forest”. In: *2008 Eighth IEEE International Conference on Data Mining*. 2008, pp. 413–422. DOI: 10.1109/ICDM.2008.17.

[40] Markus M. Breunig et al. “LOF: identifying density-based local outliers”. In: *SIGMOD Rec.* 29.2 (May 2000), pp. 93–104. ISSN: 0163-5808. DOI: 10.1145/335191.335388. URL: https://doi.org/10.1145/335191.335388.