Machine Learning Techniques and Custom Loss Functions for Enhanced Fraud and Anomaly Detection in Highly Imbalanced Datasets: A Comprehensive Survey

ARSHIA RAFIEIOSKOUEI, Michigan State University, MI
AYAAN SHAIK, Michigan State University, MI

Fraud and anomaly detection are critical applications of machine learning in finance and security domains, where identifying suspicious activities or transactions can prevent significant financial losses and protect sensitive information. These datasets are often imbalanced, making it challenging for machine learning models to perform effectively. This project surveys various machine learning techniques applied to both fraud and anomaly detection, with a focus on finance and cybersecurity. We compare traditional methods and advanced models and explore custom loss functions such as Binary Cross-Entropy (BCE), Focal Loss, and ROC-Star to improve model performance on skewed datasets.

Additional Key Words and Phrases: fraud detection, anomaly detection, imbalanced datasets, skewed data, machine learning, custom loss functions, BCE, Focal Loss, ROC-Star, finance

1 Introduction

Fraud detection and anomaly detection are crucial in sectors like finance and cybersecurity, where identifying suspicious or abnormal behavior can prevent severe financial losses, breaches, and other forms of cyberattacks [22, 27]. Both tasks involve highly imbalanced datasets , where fraudulent or anomalous activities are rare compared to the majority of legitimate ones[6, 12]. The challenge of detecting such rare events makes machine learning an essential but vulnerable tool in these fields.

Machine learning is vulnerable to imbalanced datasets because the algorithms are often biased towards the majority class, leading to poor performance in identifying minority class instances, which are often the most critical. Standard ML models are designed to optimize overall accuracy, which can result in ignoring the minority class altogether in heavily imbalanced datasets. This bias can cause problems such as high false negatives, where important events like fraud or rare diseases go undetected [10], or high false positives, which can lead to inefficiencies and user frustration. Furthermore, the minority class may not provide sufficient data for the model to learn meaningful patterns, leading to overfitting or underfitting. Evaluation metrics like accuracy can also be misleading, as they fail to reflect the poor performance on the minority class. The imbalance amplifies the risk of biased decision-making and reduces the generalizability of the model to real-world scenarios, particularly in high-stakes domains like healthcare, finance, and cybersecurity[13].

In addition, Fig. 1 illustrates the publication trend of papers in the field of imbalanced learning from January 2013 to July 2023 [3]. This trend highlights the enduring relevance of imbalanced learning as a research topic and the steady growth in its research hotspots. These observations motivate us to pursue a comprehensive study in this field.

This study aims to investigate various machine learning approaches for detecting fraud and anomalies. It evaluates both traditional and advanced models while analyzing loss functions tailored for imbalanced datasets, including Binary Cross-Entropy (BCE), Focal Loss, and ROC-Star.

Authors' Contact Information: Arshia Rafieioskouei, Michigan State University, MI, East Lansing, rafieios@msu.edu; Ayaan Shaik, Michigan State University, MI, East Lansing, shaikaya@msu.edu.

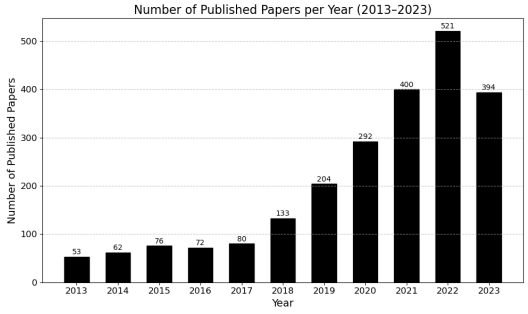


Fig. 1. Published papers over working with imbalanced dataset

1.1 Machine Learning Models

According to Tom M. Mitchell, a pioneer in the field of machine learning, a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. This definition, introduced in his influential book [16], encapsulates the essence of machine learning: the ability of a system to enhance its performance on specific tasks by learning from data or experience, without requiring explicit programming for every scenario.

Machine Learning (ML), a subfield of artificial intelligence, focuses on developing algorithms and statistical models that enable computers to learn patterns from data and make predictions or decisions without being explicitly programmed. ML can be broadly categorized into:

- **Supervised Learning**: The training data consists of input features and corresponding output labels. The goal is to develop a model $f(.)$ that accurately maps inputs to outputs and generalizes well to unseen samples during testing.
- **Unsupervised Learning**: The training data includes only input features without any output labels. The objective is to identify underlying patterns or structures, such as clusters, in the data by learning a model $f(.)$.

This study focuses on popular machine learning models within the field of classification, which involves assigning data points to predefined categories based on learned patterns.

1.1.1 Logistic Regression. This technique [11, 14, 28] is a supervised learning algorithm widely used for binary classification tasks, where the objective is to predict the probability of an outcome belonging to one of two possible classes. It works by modeling the relationship between the input features (independent variables) and the target variable (dependent variable) using a logistic function, also known as the sigmoid function. This function maps the predicted values to a range between 0 and 1, which represents the probability of the positive class. The model is trained by finding the best-fit coefficients (weights) for the input features that maximize the likelihood of correctly classifying the training data. This is achieved using an optimization method like gradient descent or gradient ascent, which updates the weights iteratively based on a loss function, such as the log-loss or binary cross-entropy. The coefficients determine the decision boundary, which separates the two classes in the feature space.

In Logistic Regression, we train the model by optimizing the weights w to maximize the log-likelihood function, which measures how well the model predicts the target labels. Logistic regression prediction function:

$$P(y = 1 \mid x) = \frac{1}{1 + e^{-z}}, \quad \text{where } z = w^T x + b \tag{1}$$

The log-likelihood function for n training samples is:

$$L(w) = \sum_{i=1}^{n} [y_i \log(P(y = 1 \mid x_i)) + (1 - y_i) \log(1 - P(y = 1 \mid x_i))] \tag{2}$$

To find the optimal w, we maximize $L(w)$ using gradient ascent, an iterative algorithm that updates the weights in the direction of the gradient of $L(w)$. The gradient is:

$$\nabla_w L(w) = \sum_{i=1}^{n} (y_i - P(y = 1 \mid x_i))\, x_i \tag{3}$$

which indicates how changes in w affect $L(w)$. The weight update (gradient ascent) in each iteration is:

$$w \leftarrow w + \eta \cdot \nabla_w L(w), \tag{4}$$

where η is the learning rate controlling the step size. The process repeats until $L(w)$ converges or a maximum number of iterations is reached. Because Eq. 3 is concave, gradient ascent Eq. 4 ensures convergence to a global maximum. This results in a set of weights w that best separates the classes, allowing the model to predict probabilities or classify new data effectively.

1.1.2 Random Forest. This model [2] is an ensemble learning algorithm primarily used for classification and regression tasks. It builds multiple decision trees during training and combines their outputs to improve accuracy and robustness while reducing overfitting. Each tree in the forest is constructed using a random subset of the training data, a process known as bootstrap aggregation or bagging. Additionally, at each split in a tree, a random subset of features is selected, further introducing diversity among the trees.

The algorithm (See Fig. 2) works by averaging the predictions of all the trees in the case of regression or using a majority vote in the case of classification. This ensemble approach ensures that the model generalizes well to unseen data, as the combination of diverse decision trees reduces the variance compared to a single tree. The randomness in both data sampling and feature selection makes Random Forest resistant to overfitting, especially when the number of trees in the forest is large.

The Random Forest algorithm can be summarized as follows:

- A subset of the training data is randomly sampled with replacement (bootstrap sampling) to create multiple datasets for training individual decision trees.
- The predictions from all trees in the forest are aggregated to make the final decision (e.g., majority voting).

Random Forests employ bagging, where each tree is trained on a randomly sampled subset of the data with replacement. This ensures that each tree is trained on a slightly different dataset, increasing the likelihood of including minority class instances in some of the samples. This enhances the model's ability to learn patterns associated with the minority class. Furthermore, Random Forests compute feature importance scores, which help identify the features most relevant for distinguishing between the majority and minority classes. These characteristics make Random Forests well-suited for achieving robust performance on imbalanced datasets [17].

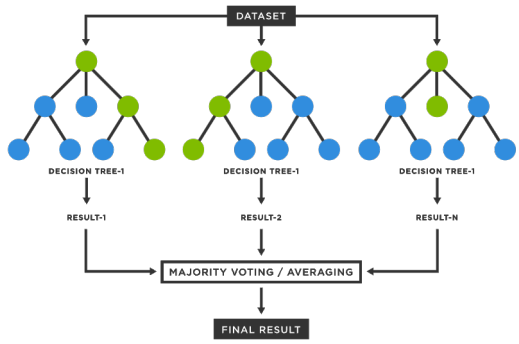


Fig. 2. Random Forest Algorithm From (Figure Reference Link)

1.1.3 Gradient Boosting (LightGBM). LightGBM [9] is an advanced implementation of Gradient Boosting Decision Trees (GBDT) that is designed to handle large-scale datasets efficiently while maintaining high accuracy. The algorithm builds decision trees sequentially, where each new tree corrects the errors of the previous ones by minimizing a differentiable loss function, typically by fitting to the negative gradient of the loss. The algorithm optimizes splits using a histogram-based approach, which discretizes continuous features into bins, reducing memory usage and computation time.

LightGBM introduces two key innovations: Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). GOSS prioritizes data instances with large gradients, as these are under-trained and more critical to improving model performance. Instances with smaller gradients are randomly sampled and re-weighted to preserve the overall data distribution. The adjusted information gain during training with GOSS is computed as:

$$\tilde{V}_j(d) = \frac{1}{n}\left(\frac{\left(\sum_{x_i \in A_l} g_i + \frac{1-a}{b}\sum_{x_i \in B_l} g_i\right)^2}{n_{j_l}(d)} + \frac{\left(\sum_{x_i \in A_r} g_i + \frac{1-a}{b}\sum_{x_i \in B_r} g_i\right)^2}{n_{j_r}(d)}\right) \tag{5}$$

In Eq. 5, $\tilde{V}_j(d)$ represents the estimated information gain for feature j at the split point d, calculated using a subset of the training data. Here, n is the total number of data instances, A_l and A_r are subsets of data instances with large gradients (under-trained instances retained in full), while B_l and B_r are subsets of data instances with small gradients (randomly sampled to reduce computational cost). The gradients of the loss function for the i-th data instance are denoted as g_i, and a, b are sampling ratios, where a is the proportion of large-gradient data retained and b is the proportion of small-gradient data sampled. $n_{j_l}(d)$ and $n_{j_r}(d)$ are the numbers of data instances in the left (l) and right (r) child nodes at the split d, respectively. The factor $\frac{1-a}{b}$ scales the small-gradient data to preserve the distribution.

EFB reduces the effective number of features by grouping mutually exclusive features, which rarely take nonzero values simultaneously, into bundles. This bundling minimizes redundancy without losing critical information, enabling faster training and lower memory usage.

The algorithm proceeds iteratively: in each iteration, LightGBM builds a new tree that minimizes the residual errors from the previous iteration. By incorporating GOSS and EFB, LightGBM achieves significant improvements in training speed and scalability while maintaining high predictive

accuracy. This makes it particularly effective for tasks such as classification, regression, and ranking in large-scale machine learning problems.

These features allow LightGBM to handle high-dimensional datasets and focus on the most influential data points, enhancing predictive performance. Moreover, its compatibility with custom loss functions and ability to adapt to different machine learning tasks makes it particularly relevant for our problem.

1.1.4 Neural Network. These days, neural networks are widely used in various fields, including image recognition, natural language processing, and predictive analytics, due to their ability to solve complex problems and adapt to different types of data. A Multilayer Perceptron (MLP) is a type of artificial neural network designed to learn hierarchical features and handle data that is not linearly separable. Unlike traditional linear models, MLPs use multiple layers of neurons to transform input data into higher-dimensional feature spaces, enabling the modeling of complex patterns and relationships [8, 15, 23, 25].

MLPs rely on feature learning through hidden layers. The network takes an input vector $\mathbf{x} = [x_1, x_2, \ldots, x_m]$ and transforms it into a feature vector $\mathbf{z} = [z_1, z_2, \ldots, z_n]$ in the hidden layer. The feature transformation is computed as:

$$z_j = \sigma\left(\sum_{i=1}^{m} w_{ij}^{(0)} x_i + w_j^{(0)}\right) = \sigma\left(w_j^{(0)^T} x\right), \quad \text{for } j = 1, \ldots, m \tag{6}$$

where $w_{ij}^{(0)}$ are the weights connecting input x_i to hidden unit z_j, $w_j^{(0)}$ are the biases for the hidden layer, and σ is a activation function. Some commonly known activation functions are, hard threshold, sigmoid, linear, Rectified Linear Unit (ReLU), SoftPlus, Hyperbolic tangent (tanh).

The output of the hidden layer is then processed by the output layer to compute the final prediction:

$$h(\mathbf{x}) = \sigma\left(\sum_{i=1}^{n} w_i^{(1)} z_i + w_0^{(1)}\right) = \sigma\left(w^{(1)^T} z\right) \tag{7}$$

where $w_i^{(1)}$ are the weights connecting hidden unit z_i to the output, $w_0^{(1)}$ is the bias term for the output layer, and σ is the activation function of the output layer. MLPs are trained using backpropagation and gradient descent to minimize a loss function $L(y, \hat{y})$, where y is the true label and $\hat{y}$ is the predicted output. The training process involves updating the weights and biases across all layers to reduce the prediction error.

1.2 Loss Functions

To work with the machine learning models discussed in previous sections, a loss function is required. A loss function quantifies the error or difference between the model's predictions and the actual target values. It acts as the objective the model minimizes during training, helping the model improve its predictions and better capture the relationship between input features and target outputs.To handle imbalanced data, we will experiment with the following loss functions.

1.2.1 Binary Cross-Entropy (BCE). Cross-Entropy (CE) is a widely used loss function in machine learning that measures the difference between two probability distributions. When the softmax function in CE is replaced with the sigmoid function, the resulting loss function is known as Binary Cross-Entropy (BCE), specifically designed for binary classification tasks. The BCE loss for a single data instance is defined as:

$$\mathcal{L}(y, \hat{y}) = -\left[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\right] \tag{8}$$

where $y \in \{0, 1\}$ is the true label (binary target), and $\hat{y} \in [0, 1]$ is the predicted probability of the positive class. For a dataset with n examples, the total BCE loss is the average over all instances:

$$\text{BCE Loss Function} = -\frac{1}{n} \sum_{i=1}^{n} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \tag{9}$$

The terms $y \log(\hat{y})$ and $(1 - y) \log(1 - \hat{y})$ penalize the model when the predicted probabilities deviate from the true labels. Specifically, the first term penalizes incorrect predictions for the positive class ($y = 1$), while the second term penalizes incorrect predictions for the negative class ($y = 0$). The loss is minimized when $\hat{y} = y$, meaning the predicted probability matches the true label.

Binary Cross-Entropy (BCE) can perform poorly on imbalanced datasets because it minimizes the average loss across all examples. In such datasets, the majority class dominates the loss calculation, leading the model to disproportionately favor predicting the majority class while neglecting the minority class. However, Weighted Binary Cross-Entropy (Weighted BCE) can address this issue by assigning different weights to the classes, ensuring that the minority class contributes more significantly to the loss. This adjustment helps the model focus on both classes more effectively, improving performance on imbalanced datasets. [19]

1.2.2 Focal Loss. Focal Loss is a loss function designed to address class imbalance issues in classification tasks by focusing more on hard-to-classify examples while reducing the influence of well-classified ones. It builds on Binary Cross-Entropy (BCE) by adding a modulating factor to down-weight the loss contribution of easy examples, making it particularly effective in imbalanced datasets. The Focal Loss for binary classification is given as:

$$\mathcal{L}(y, \hat{y}) = -\alpha \cdot (1 - \hat{y})^{\gamma} \cdot y \log(\hat{y}) - \alpha \cdot \hat{y}^{\gamma} \cdot (1 - y) \log(1 - \hat{y}) \tag{10}$$

Where, in Eq. 10 $y \in \{0, 1\}$ is the true label, $\hat{y} \in [0, 1]$ is the predicted probability of the positive class, $\alpha \in [0, 1]$ is a balancing factor to account for class imbalance, $\gamma \geq 0$ is the focusing parameter that controls the strength of the modulating factor. The term $(1 - \hat{y})^{\gamma}$ reduces the loss contribution from well-classified examples, where $\hat{y}$ is close to the true label. This allows the model to focus on hard-to-classify examples, which often belong to minority or difficult classes. Focal Loss is widely used in classification tasks where class imbalance is a significant challenge [18, 30, 31].

1.2.3 ROC-STAR. This loss function [32] is specifically designed to optimize binary classifiers, particularly in the context of imbalanced datasets, by targeting the maximization of the Area Under the Receiver Operating Characteristic Curve (AUC-ROC). Unlike traditional loss functions, which typically optimize for metrics such as accuracy or cross-entropy, the proposed ROC-Star loss function aligns directly with the AUC-ROC metric. This makes it particularly effective in scenarios where the positive class is underrepresented.

The AUC-ROC metric quantifies the trade-off between the true positive rate (TPR) and the false positive rate (FPR) across various classification thresholds. Mathematically, it is defined as:

$$\text{AUC} = \frac{1}{|N^-| \cdot |N^+|} \sum_{(x,y) \in N^+ \times N^-} \begin{cases} 1 & \text{if } x < y, \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

In Eq. 11, N^+ and N^- represent the sets of positive and negative instances, respectively. Intuitively, this measures the proportion of correctly ranked positive-negative instance pairs, normalized by the total number of such pairs, $|N^-| \cdot |N^+|$.

To derive a loss function, we invert the ranking condition, penalizing wrong-ordered pairs where $x > y$. The primary challenge is to handle the discontinuity introduced when x crosses y. To address this, we introduce a smoothed loss function:

$$\mathcal{L} = \sum_{(x,y)\in N^- \times N^+} \begin{cases} (x + \Gamma - y)^p & \text{if } x + \Gamma > y, \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

In Eq. 12, $x < y$ is flipped to $y < x$ to emphasize wrong-ordered pairs, making the loss higher when ranking is incorrect. Additionally, the loss is weighted by the magnitude of the violation, $(x + \Gamma - y)$, raised to the power p. The padding term Γ provides a buffer, penalizing not only wrong-ordered pairs but also right-ordered pairs that are too close. This ensures that even correctly ranked pairs maintain a sufficient margin, reducing the risk of them being misranked in future iterations.By focusing on pairwise ranking violations and introducing a mechanism to enforce separability, this loss function directly optimizes for AUC-ROC while addressing challenges inherent in imbalanced datasets.

ROC-Star is particularly effective for imbalanced datasets because it focuses on ranking rather than absolute predictions. It not only penalizes wrong-ordered pairs but also encourages a safe margin between positive and negative scores to reduce the risk of errors due to small perturbations.

1.3 Evaluation Metrics

To assess the performance of our model, we will use a variety of evaluation metrics that provide a comprehensive understanding of how well the model performs. Below are the metrics, their formulas, and their practical meanings:

- Precision measures how many of the positive predictions made by the model are actually correct. It focuses on the quality of positive predictions and is calculated as:

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}.$$

  A high precision indicates that the model makes fewer false alarms (incorrect positive predictions).
- Recall, also known as sensitivity or true positive rate, tells us how many of the actual positive cases the model identifies correctly. It is defined as:

$$\text{Recall} = \text{ True positive rate} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}.$$

  A high recall means the model rarely misses positive cases, which is critical in tasks like fraud detection or medical diagnosis.
- The F1 score combines precision and recall into a single metric by taking their harmonic mean. It is particularly useful for imbalanced datasets where both precision and recall are important. The formula is:

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

  A higher F1 score indicates a good balance between precision and recall.
- The Area Under the Receiver Operating Characteristic Curve (AUC-ROC) measures the ability of the model to distinguish between positive and negative classes. It is a threshold-independent metric and is particularly relevant for imbalanced datasets. AUC is computed as the area under the curve where the true positive rate (TPR) is plotted against the false positive rate (FPR).

- False positive rate measures the proportion of actual negatives incorrectly classified as positives. It is defined as:

$$\text{FPR} = \frac{\text{False Positives (FP)}}{\text{False Positives (FP)} + \text{True Negatives (TN)}}.$$

  A lower FPR indicates fewer false alarms.
- True negative rate, also known as specificity, measures the proportion of actual negatives correctly identified by the model. It is the complement of FPR and is calculated as:

$$\text{TNR} = \frac{\text{True Negatives (TN)}}{\text{True Negatives (TN)} + \text{False Positives (FP)}}.$$

  A high TNR indicates the model is effective at correctly identifying negative cases.
- False negative rate measures how often the model misses positive cases. It is the complement of TPR and is given by:

$$\text{FNR} = \frac{\text{False Negatives (FN)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}.$$

  A lower FNR is crucial in applications like medical screening, where missing a positive case can have severe consequences.

1.4 Problem Statement

The goal of this project is to conduct a comprehensive survey of existing machine learning techniques and loss functions specifically designed for skewed datasets in fraud and anomaly detection. By systematically evaluating both traditional and advanced models, as well as custom loss functions like Binary Cross-Entropy (BCE), Focal Loss, and ROC-Star, we aim to determine the most effective methods for handling imbalanced data and improving model performance in these critical applications.

2 Methodology

The overall approach, illustrated in Fig. 3, involved several key steps: selecting a dataset aligned with our objectives, preprocessing the data, and employing widely used machine learning techniques to develop the model. We utilized tailored loss functions to train the models and conducted a comparative analysis of the results across different methodologies. This section provides a concise overview of each step in the process.

2.1 Dataset

We sought a dataset with a skewed label distribution, representing an imbalanced dataset. To this end, we selected the publicly available Credit Card Fraud Detection dataset[1], which is notably imbalanced, as shown in Fig. 4.

2.2 Preprocessing

2.2.1 Credit Card Fraud Dataset. The preprocessing for the Credit Card Fraud Dataset involved minimal steps due to its clean and consistent nature. The dataset was free of null values and exhibited a well-structured format. As the primary preprocessing step, the features were scaled using a Standard Scaler to normalize the data and ensure uniformity across different feature magnitudes. This scaling was crucial for improving the performance and convergence of the model during training.

[1]https://kaggle.com/datasets/mlg-ulb/creditcardfraud

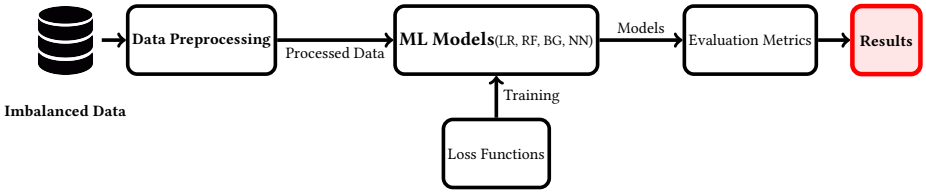


Fig. 3. Overview of the Methodology. The framework starts with imbalanced datasets undergoing preprocessing. Preprocessed data is fed into machine learning models (e.g., Logistic Regression (LR), Random Forest (RF), Bagging (BG), Neural Networks (NN)). Loss functions are used during training, and the models are evaluated based on comparison metrics, leading to the final results.

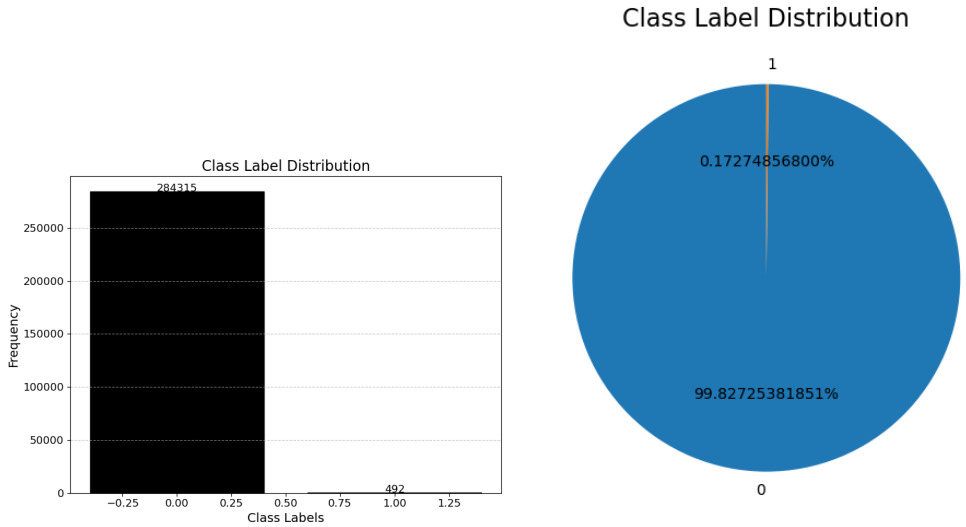


Fig. 4. Dataset imbalanced class demonstration

2.2.2 Loan Default Prediction Dataset. The Loan Default Prediction Dataset also exhibited high data quality, with no null values present. However, preprocessing required additional steps to handle categorical variables present in the dataset. Encoding techniques, such as one-hot encoding or label encoding, were applied to convert categorical features into numerical representations suitable for machine learning algorithms. Additionally, numerical features were standardized using a Standard Scaler to normalize the range of values, ensuring that the dataset was optimized for model training and improving the stability of the optimization process.

2.3 Training Machine Learning Models and Evaluation

In this section, we apply the models introduced in Section 1.1, aligning them with the loss functions described in Section 1.2. We implement logistic regression and random forest using the `scikit-learn` library functions [21]. For neural networks, we utilize the `PyTorch` library [20], configuring two settings: a *light setting* with one hidden layer (16 perceptrons, ReLU activation, and a sigmoid output layer) and a *dense setting* with two hidden layers (64 and 32 perceptrons, ReLU activations, and a sigmoid output layer). To address the inherent class imbalance in datasets such as those used for fraud detection, we employ Synthetic Minority Oversampling Technique

Table 1. Performance Metrics for Various Models (Precision, Recall, and F1-Score)

Model	Precision	Recall	F1-Score	Accuracy
LightGBM BCE	0.961	0.755	0.846	0.999
Lightgbm Focal Loss	0.961	0.755	0.846	0.999
Lightgbm ROC-Star	0.688	0.847	0.755	0.999
Dense NN	0	0	0	0.998
Dense NN SMOTE	0.001	0.956	0.003	0.197
Light NN	0.001	0.793	0.002	0.021
Light NN SMOTE	0.001	0.641	0.002	0.01
Logistic Regression	0.88	0.641	0.742	0.999
Random Forest	0.944	0.739	0.829	0.999

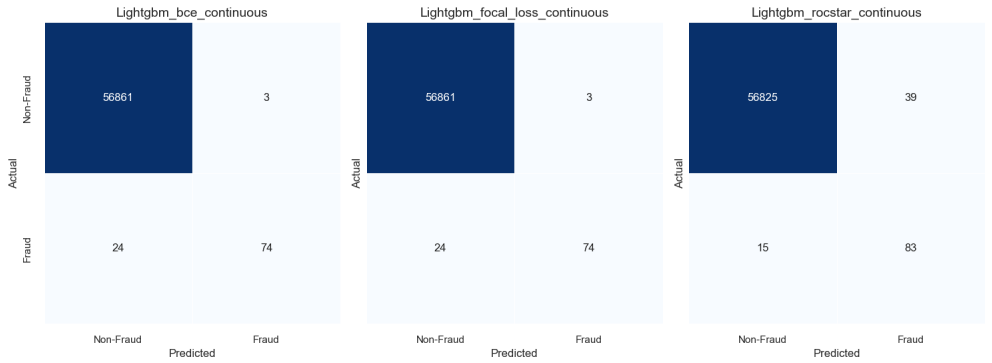


Fig. 5. Confusion Matrices for LightGBM: (Left) Binary Cross-Entropy (BCE) Loss, (Middle) Focal Loss, (Right) ROC-Star Loss on the Credit Card Fraud dataset.

(SMOTE) during the training of neural networks. By generating synthetic samples for the minority class, SMOTE helps improve the network's ability to generalize and reduces the bias toward the majority class, ultimately enhancing Recall and True Positive Rate (TPR). In the case of LightGBM, we develop and implement the custom loss algorithm from scratch.

In the evaluation step, we assess our models using the metrics introduced in Section 2, with a particular emphasis on Recall and True Positive Rate (TPR). These metrics are critical in scenarios such as fraud detection or disease diagnosis, where minimizing false negatives is paramount. Misclassifying a positive case as negative can incur significant consequences, making the reduction of false negatives a priority.

3 Result and Discussion

In this section, we present the results obtained from Section 2. Table 1 summarizes the key evaluation metrics. It is evident that LightGBM outperforms other models in terms of the F1-Score. However, for this specific problem, our primary focus is on a model that excels in Recall (True Positive Rate), as minimizing false negatives is crucial. Table 1 also demonstrates that LightGBM achieves superior Recall compared to the other models.

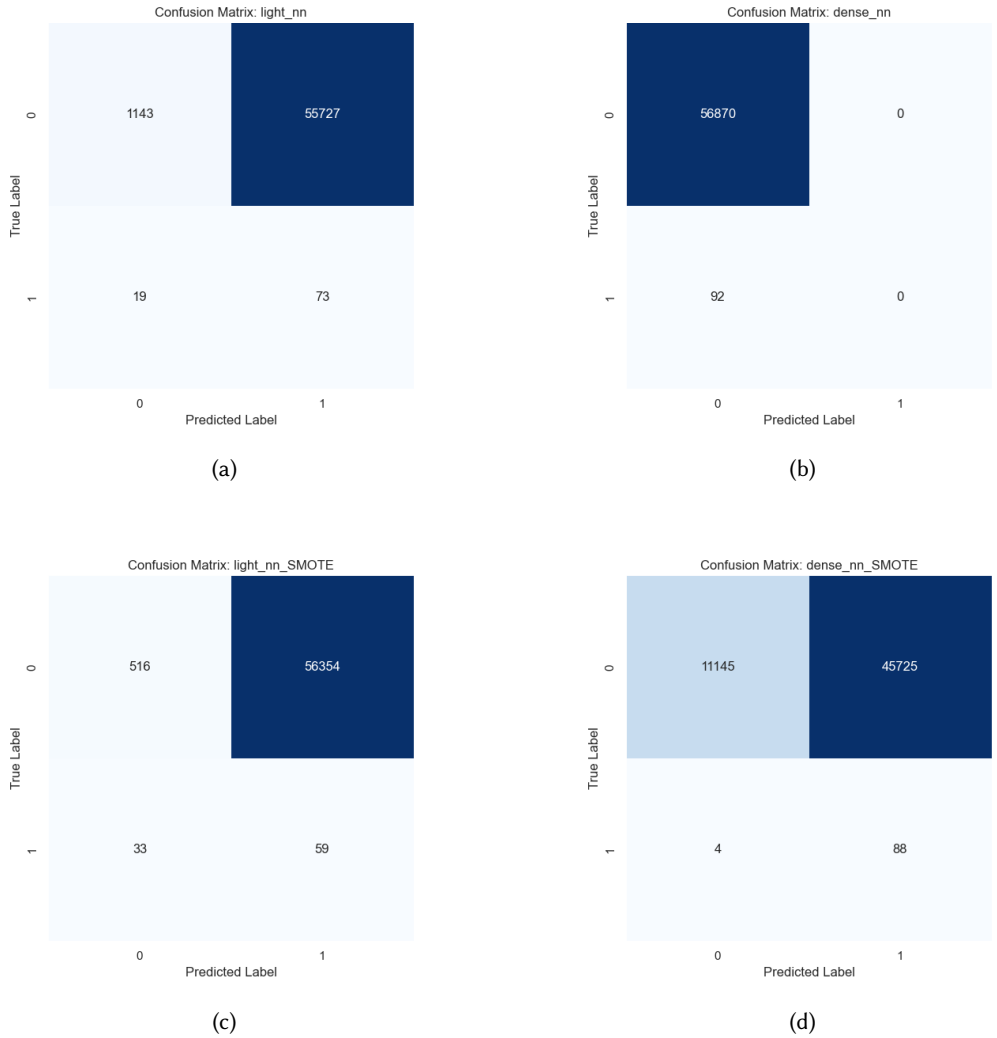
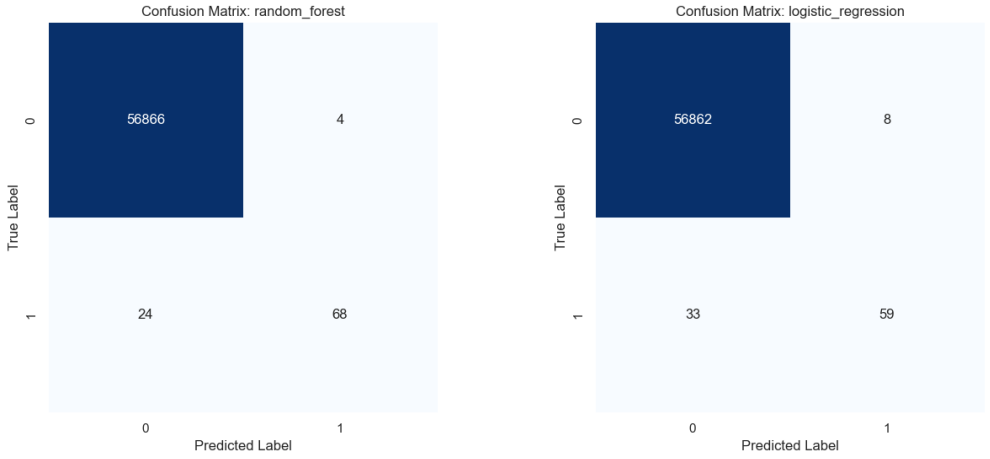


Fig. 6. Confusion Matrices: (a) Light Neural Network, (b) Dense Neural Network, (c) Light NN with SMOTE, and (d) Dense NN with SMOTE.

3.1 Model Performance Comparison

LightGBM Models: The LightGBM models consistently outperformed other approaches, achieving the highest Recall (0.846–0.853) and F1-Score (0.885). Among them:

- *BCE Loss:* Achieved the highest Recall (0.853), indicating its superior ability to minimize false negatives, critical for fraud detection tasks.
- *ROC-Star Loss:* Despite a slightly lower AUC compared to BCE, ROC-Star Loss minimized false negatives even further (Figure 8), aligning with its design to prioritize True Positive Rate (TPR) over global metrics like AUC. This makes it particularly suitable for highly imbalanced datasets.



(a) (b)

Fig. 7. Confusion Matrices: (a) Random Forest, (b) Logistic Regression.

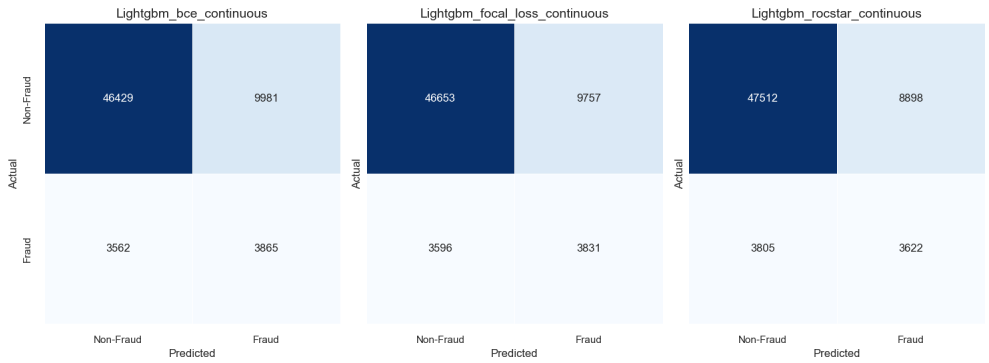


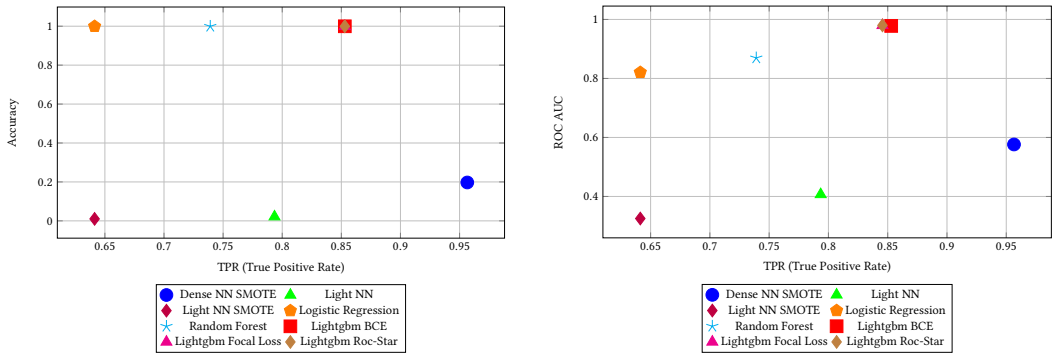
Fig. 8. Confusion Matrices for LightGBM: (Left) Binary Cross-Entropy (BCE) Loss, (Middle) Focal Loss, (Right) ROC-Star Loss on the Loan Default dataset.

- *Focal Loss:* Performed similarly to ROC-Star in terms of Recall but offered marginally higher Precision, making it useful for scenarios where false positives also carry a cost.

Neural Networks: Both Light and Dense Neural Networks, even when augmented with SMOTE, failed to effectively classify minority class instances. The Recall values remained near zero, and the F1-Scores were negligible (Figure 6). This failure highlights the limitations of these architectures in handling extreme class imbalances without advanced preprocessing or specialized loss functions.

Traditional Models: Logistic Regression and Random Forest demonstrated respectable performance:

- Logistic Regression achieved moderate Recall (0.641) and F1-Score (0.742), but its higher false negative rate limits its practical use in high-stakes applications like fraud detection.



(a) (b)

Fig. 9. Accuracy vs True Positive Rate (left), ROC-AUC vs True Positive Rate (right)

- Random Forest excelled in Precision (0.944) but lagged behind LightGBM in Recall (0.739), indicating a bias toward the majority class, as evident in its confusion matrix (Figure 9).

3.2 Trade-Off Analysis

Figures 9a and 9b illustrate the trade-offs between Accuracy, TPR, and AUC:

- LightGBM ROC-Star Loss achieved the best trade-off, minimizing false negatives while maintaining competitive Accuracy and AUC. This aligns with the goal of maximizing TPR in imbalanced datasets.
- Random Forest, while achieving high Precision, sacrificed Recall, making it less suitable for detecting rare events like fraud.

4 Related Work

Several studies have focused on fraud detection and anomaly detection in finance and cybersecurity. Traditional machine learning methods like logistic regression and decision trees have been used in fraud detection for financial transactions [5, 24]. Meanwhile, anomaly detection in cybersecurity focuses on identifying unusual network behavior or system logs [1, 4, 29].

Machine Learning popular techniques, including neural networks, random forest [2] , and gradient boosting [7], have demonstrated improved performance in both fraud and anomaly detection tasks. However, dealing with imbalanced datasets remains a challenge. Custom loss functions, such as Focal Loss and ROC-Star, have been proposed to better address this issue [26, 32].

5 Conclusion and Future Work

This project will deliver a comprehensive survey of machine learning models and custom loss functions for fraud and anomaly detection in finance and security. This study evaluated multiple models for fraud detection on imbalanced datasets, with a focus on minimizing false negatives. The key takeaways are as follows:

- LightGBM models, particularly with BCE Loss, demonstrated the best performance in Recall (0.853) and F1-Score (0.885), making it the most suitable choice for this problem.
- Neural network models, even when enhanced with SMOTE, were unable to effectively classify the minority class, highlighting their limitations in handling imbalanced datasets without further optimization.

- While Random Forest and Logistic Regression showed respectable performance, their higher false negatives compared to LightGBM reduce their applicability for this task.

Future work could focus on optimizing LightGBM through hyperparameter tuning or exploring ensemble methods to further enhance Recall and overall performance. Additionally, advanced techniques like cost-sensitive learning or dynamic thresholding could be investigated to better handle imbalanced datasets.

References

[1] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. 2016. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications* 60 (2016), 19–31.

[2] Leo Breiman. 2001. Random Forests. *Machine Learning* 45, 1 (Oct. 2001), 5–32. https://doi.org/10.1023/A:1010933404324

[3] Wuxing Chen, Kaixiang Yang, Zhiwen Yu, Yifan Shi, and C. L. Philip Chen. 2024. A survey on imbalanced learning: latest research, applications and future directions. *Artificial Intelligence Review* 57, 6 (May 2024), 137. https://doi.org/10.1007/s10462-024-10759-6

[4] Marina Evangelou and Niall M Adams. 2020. An anomaly detection framework for cyber-security data. *Computers & Security* 97 (2020), 101941.

[5] Vaishali Ganganwar. 2012. An overview of classification algorithms for imbalanced datasets. *International Journal of Emerging Technology and Advanced Engineering* 2, 4 (2012), 42–47.

[6] Swati Ganguly and Samira Sadaoui. 2017. Classification of imbalanced auction fraud data. In *Advances in Artificial Intelligence: 30th Canadian Conference on Artificial Intelligence, Canadian AI 2017, Edmonton, AB, Canada, May 16-19, 2017, Proceedings 30*. Springer, 84–89.

[7] Zhiyuan He, Danchen Lin, Thomas Lau, and Mike Wu. 2019. Gradient Boosting Machine: A Survey. *arXiv e-prints*, Article arXiv:1908.06951 (Aug. 2019), arXiv:1908.06951 pages. https://doi.org/10.48550/arXiv.1908.06951 arXiv:1908.06951 [stat.ML]

[8] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks* 2, 5 (1989), 359–366.

[9] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf

[10] Mohammed Khalilia, Sounak Chakraborty, and Mihail Popescu. 2011. Predicting disease risks from highly imbalanced data using random forest. *BMC Medical Informatics and Decision Making* 11, 1 (July 2011), 51. https://doi.org/10.1186/1472-6947-11-51

[11] Gary King and Langche Zeng. 2001. Logistic Regression in Rare Events Data. *Political Analysis* 9, 2 (2001), 137–163. https://doi.org/10.1093/oxfordjournals.pan.a004868

[12] Sara Makki, Zainab Assaghir, Yehia Taher, Rafiqul Haque, Mohand-Said Hacid, and Hassan Zeineddine. 2019. An experimental study with imbalanced classification approaches for credit card fraud detection. *IEEE Access* 7 (2019), 93010–93022.

[13] Koppula Manasa and L. M. I. Leo Joseph. 2023. A Machine Learning-Based Vulnerability Detection Approach for the Imbalanced Dataset UNSW-NB15. In *Communication and Intelligent Systems*, Harish Sharma, Vivek Shrivastava, Kusum Kumari Bharti, and Lipo Wang (Eds.). Springer Nature Singapore, Singapore, 279–297.

[14] P. McCullagh and J. A. Nelder. 1989. *Generalized Linear Models*. Vol. 37. http://dx.doi.org/10.1007/978-1-4899-3242-6

[15] Marvin Minsky and Seymour Papert. 1969. *Perceptrons: An Introduction to Computational Geometry*. MIT Press.

[16] Tom M Mitchell. 1997. *Machine learning*. Vol. 1. McGraw-hill New York.

[17] A. S. More and Dipti P. Rana. 2017. Review of random forest classification techniques to resolve data imbalance. In *2017 1st International Conference on Intelligent Systems and Information Management (ICISIM)*. 72–78. https://doi.org/10.1109/ICISIM.2017.8122151

[18] Mulyanto Mulyanto, Muhamad Faisal, Setya Widyawan Prakosa, and Jenq-Shiou Leu. 2021. Effectiveness of Focal Loss for Minority Classification in Network Intrusion Detection Systems. *Symmetry* 13, 1 (2021). https://doi.org/10.3390/sym13010004

[19] Quang Du Nguyen and Huu-Tai Thai. 2023. Crack segmentation of imbalanced data: The role of loss functions. *Engineering Structures* 297 (2023), 116988. https://doi.org/10.1016/j.engstruct.2023.116988

[20] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[22] Nadipuram R Prasad, Salvador Almanza-Garcia, and Thomas T Lu. 2010. Anomaly detection. *Computers, Materials, & Continua* 14, 1 (2010), 1–22.

[23] Frank Rosenblatt. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review* 65, 6 (1958), 386–408.

[24] Neelam Rout, Debahuti Mishra, and Manas Kumar Mallick. 2018. Handling imbalanced data: a survey. In *International Proceedings on Advances in Soft Computing, Intelligent Systems and Applications: ASISA 2016*. Springer, 431–443.

[25] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *Nature* 323, 6088 (1986), 533–536.

[26] Takaya Saito and Marc Rehmsmeier. 2015. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PloS one* 10, 3 (2015), e0118432.

[27] Philip Olaseni Shoetan, Adedoyin Tolulope Oyewole, Chinwe Chinazo Okoye, and Onyeka Chrisanctus Ofodile. 2024. Reviewing the role of big data analytics in financial fraud detection. *Finance & Accounting Research Journal* 6, 3 (2024), 384–394.

[28] Craig Starbuck. 2023. *Logistic Regression.* Springer International Publishing, Cham, 223–238. https://doi.org/10.1007/978-3-031-28674-2_12

[29] Chee-Wooi Ten, Junho Hong, and Chen-Ching Liu. 2011. Anomaly detection for cybersecurity of the substations. *IEEE Transactions on Smart Grid* 2, 4 (2011), 865–873.

[30] Cheng Wang, Jorge Balazs, György Szarvas, Patrick Ernst, Lahari Poddar, and Pavel Danchenko. 2022. Calibrating Imbalanced Classifiers with Focal Loss: An Empirical Study. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track*, Yunyao Li and Angeliki Lazaridou (Eds.). Association for Computational Linguistics, Abu Dhabi, UAE, 145–153. https://doi.org/10.18653/v1/2022.emnlp-industry.14

[31] Chen Wang, Chengyuan Deng, and Suzhen Wang. 2020. Imbalance-XGBoost: leveraging weighted and focal losses for binary label-imbalanced classification with XGBoost. *Pattern Recognition Letters* 136 (2020), 190–197. https://doi.org/10.1016/j.patrec.2020.05.035

[32] Lian Yan, Robert H Dodier, Michael Mozer, and Richard H Wolniewicz. 2003. Optimizing classifier performance via an approximation to the Wilcoxon-Mann-Whitney statistic. In *Proceedings of the 20th international conference on machine learning (icml-03)*. 848–855.