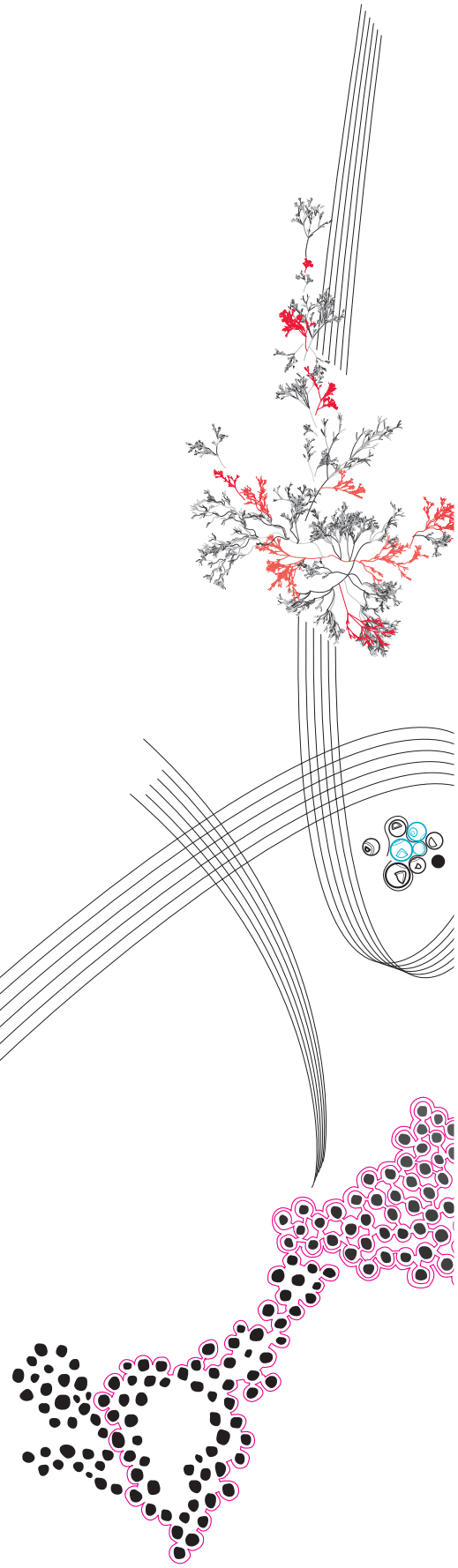

MSc Cybersecurity
Final Project

Cybersecurity
Detecting fake accounts on TikTok

Mahsa Heydari

Supervisor: Dr. Dipti K. Sarmah, Lecturer, University of Twente,
Dr. Alessandro Galeazzi, Assistant Professor, University of Padova

July, 2025

Department of Computer Science
Faculty of Electrical Engineering,
Mathematics and Computer Science,
University of Twente

UNIVERSITY OF TWENTE.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Research Questions . . . 3
- 1.3 Outline and Summary . . . 4

2 Background and Related Work **5**
- 2.1 Machine Learning Algorithms . . . 5
- 2.2 Rationale for Semi-Supervised Learning . . . 7
- 2.3 Data Preprocessing . . . 7
  - 2.3.1 Normalization Methods . . . 8
  - 2.3.2 Resampling Methods . . . 9
- 2.4 Feature Selection Algorithm . . . 11
  - 2.4.1 Feature Comparison Across Platforms . . . 12
  - 2.4.2 Recursive Feature Elimination with a Support Vector Machine classifier . 16
- 2.5 Evaluation Metrics for Classifier Performance . . . 17
- 2.6 Overview of Applied Classification Techniques . . . 18
- 2.7 Self-training semi-supervised learning algorithm based on resampling . . . 19

3 Proposed solution **21**
- 3.1 Proposed Self-Training Method for Fake Account Detection . . . 21
  - 3.1.1 Workflow of the proposed SSSTR Method . . . 22
  - 3.1.2 Parameter h in Self-Training . . . 24
- 3.2 Data Collection . . . 26
  - 3.2.1 TikTok API Overview . . . 27
- 3.3 TikTok Feature Selection . . . 27

4 Results and Discussion **30**
- 4.1 Evaluation of Feature Subsets and Preprocessing Strategies . . . 30
- 4.2 Visual Analysis and Discussion of Experimental Results . . . 37

5 Conclusion and Future Work **43**

Abstract

Online social networks (OSNs) are essential for modern communication, enabling users to connect, share content, access news, and engage with trending topics. However, the proliferation of fake accounts on these platforms raises concerns about misinformation, fraud, and malicious activities. While prior research has extensively explored fake account detection on platforms like Twitter, Facebook, and Instagram using supervised and semi-supervised machine learning techniques, TikTok remains underexplored. This study bridges this gap by adapting a semi-supervised self-training framework, originally developed for Twitter, to detect fake accounts on TikTok, using its unique feature set and evaluating performance with metrics such as AUC, Recall, and G-Mean. We use a labeled dataset of 1,699 TikTok accounts (758 fake, 941 real) sourced from GitHub, enriched with features extracted via the TikTok API and Apify.com. Robust preprocessing, including Z-Score and Min-Max normalization, ensures consistent feature scaling, while resampling techniques like Synthetic Minority Oversampling Technique (SMOTE) and Cluster-Based Undersampling Technique (CBUTE) address class imbalance to enhance model performance. The proposed Self-Training Semi-Supervised Resampling (SSSTR) framework, tested with six classifiers including Random Forest, CART, Gradient Boosting, AdaBoost, K-Nearest Neighbor, and Naïve Bayes, demonstrates strong results. Among the classifiers evaluated, Random Forest consistently performed best when paired with Z-Score normalization and a carefully selected set of features. CART and Gradient Boosting also yielded strong outcomes under similar conditions. A reduced feature set containing only four attributes maintained competitive performance, showing that efficient detection is possible even with a compact model. Using SMOTE to balance class distribution improved performance across classifiers, particularly compared to CBUTE. By integrating resampling within the self-training process and analyzing feature subsets, this research provides novel insights into fake account detection on TikTok. The methodology and findings offer a foundation for future studies on TikTok feature extraction and platform security.
Keywords: Fake accounts, social media, TikTok, machine learning, semi-supervised learning

Chapter 1

Introduction

In this chapter we outline the main focus of the research and the steps taken throughout the study. It discusses the interests that motivated the research, which helped formulate research questions and objectives. The chapter also addresses the importance of the research, emphasizing its significance both in academic contexts and practical applications. Finally, it shows how the research is organized and explains how each chapter helps to build the main argument and reach the research goals.

1.1 Motivation

In today's digital age, online social networks (OSNs) have become an integral part of society, connecting individuals across the globe. As shown in FIGURE 1.1, platforms such as Facebook, Instagram, TikTok, and Twitter have experienced rapid growth in recent years, fundamentally transforming how we interact and maintain social connections[65].

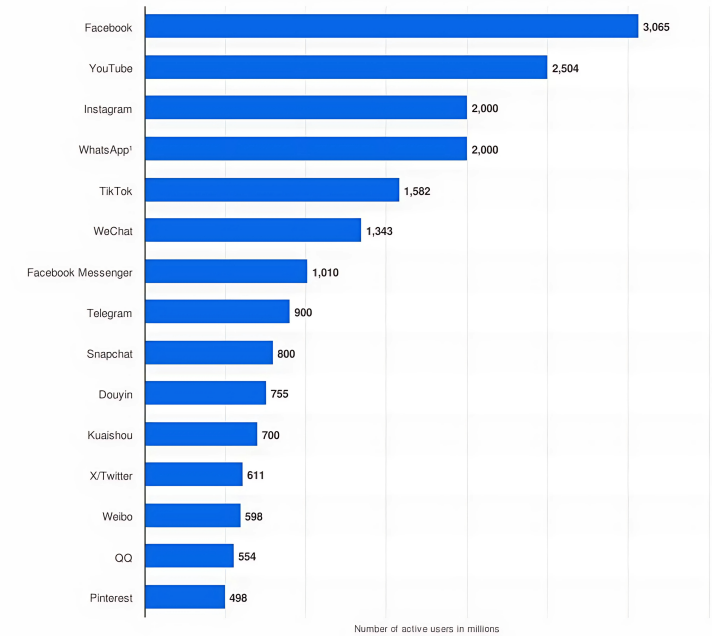

FIGURE 1.1: Most popular social networks worldwide, by number of monthly active users(in millions)-April 2024[77]

These platforms make it easy to meet new people, stay in touch with friends, and keep up with their updates[26]. However, this rapid expansion has also given rise to serious issues, such as fake profiles and online impersonation[29]. As OSNs continue to grow and accumulate personal data, they have become attractive targets for malicious activities. Attackers exploit these platforms to steal sensitive information, spread misinformation, and carry out fraudulent schemes[76]. A common tactic involves using fake profiles to initiate unsolicited connections, which can escalate into manipulative interactions and financial scams under false pretenses[3]. Unfortunately, effective solutions to counter fake accounts remain limited. Traditional rule-based detection systems are too rigid and often fail to adapt to new fraud tactics. For example, fraudsters now create AI-generated deepfake profiles on platforms like LinkedIn to impersonate real users and deceive others, making detection more challenging [80]. Crowdsourced [81][16] reporting is another approach, defined as gathering user reports to identify suspicious activities. However, it is often slow because it depends on manual user participation. For example, users need to take the initiative to report issues, which may not happen promptly. Moreover, it is prone to biases, since reports may reflect subjective opinions rather than objective facts. For instance, a user might label content as harmful simply because it contradicts their personal beliefs. As a result, the identification of fake accounts is delayed, emphasizing the need for more adaptive and automated detection techniques.

Several machine learning techniques have been developed to detect fake accounts in OSNs, such as Support Vector Machines (SVM) [35], Random Forest (RF) [27], k-Nearest Neighbors (KNN) [8], and Decision Trees (DT) [52]. These methods leverage content features (e.g., URLs in tweets), behavioral features (e.g., clickstreams, likes, photo uploads), and social graph features (e.g., friendship graphs, follower–following graphs) generated by fake accounts. While several studies have addressed fake account detection on platforms such as Facebook, Twitter, and Instagram, research on TikTok remains limited, partly due to its recent emergence and challenges in accessing user data [9][90]. Given its rapid growth, unique features, and the popularity of its short-form video content, we chose TikTok as the focus of our study. Unlike other platforms, TikTok's short video format offers distinctive user behaviors and engagement patterns, making it an interesting environment to apply machine learning algorithms for fake account detection [9][90]. According to recent data, TikTok has emerged as one of the world's most popular applications, more than 1.5 billion monthly active users[77]. TikTok's widespread popularity can be attributed to its engaging content across various topics, including dance trends, comedy skits, and DIY (Do It Yourself) tutorials [4]. A key factor in TikTok's success is its sophisticated recommendation system, known as the "For You Page" (FYP) algorithm, which personalizes content to maintain user engagement while efficiently incorporating music and e-commerce features [86].

In this research, we aim to detect fake accounts within an unlabeled dataset collected from the followers of 10 Dutch influencers using the TikTok API [78] and Apify platform [6], using a TikTok labeled dataset from GitHub [7] and a semi-supervised self-training method that has previously been applied to Twitter [83], adapted here for TikTok.

The research explores the effectiveness of semi-supervised learning in environments where labeled data is scarce, examines the impact of different feature selection techniques on classification performance, and addresses class imbalance challenges through resampling strategies. This research addresses the challenge of identifying fake accounts on TikTok, where studies remain limited, by utilizing both labeled and unlabeled data. It aims to contribute to academic knowledge and offer practical insights that could support future efforts to enhance platform security and foster user trust in the growing social media environment. To achieve these objectives, appropriate classification metrics are essential for evaluating the performance of fake account detection models, particularly when dealing with imbalanced datasets [42]. Metrics such as recall, F1-score, G-Mean, and AUC provide a more comprehensive and reliable assessment compared to accuracy alone, which can be misleading in highly skewed class distributions [75].

In imbalanced datasets, models often become biased toward the majority class, resulting in poor detection performance for minority classes such as fake accounts [42]. Therefore, the application of data balancing techniques is necessary to ensure that models treat both classes appropriately. Similarly, feature scaling is important when working with features that have large variations in numerical range, such as follower count or number of likes. Without appropriate scaling, algorithms sensitive to feature magnitudes, including k-Nearest Neighbors (KNN) and Support Vector Machines (SVM), may perform suboptimally [39]. Addressing these challenges is critical for building robust detection models and lays the foundation for the formulation of the research questions in this research.

1.2 Research Questions

This study investigates the detection of fake accounts on TikTok using a self-training semi-supervised machine learning algorithm. To guide the research process and structure the analysis, we define the following research questions (RQs):

RQ1. What are the most effective features for distinguishing fake TikTok accounts from real ones, and how do their interactions influence the accuracy of fake account detection?

RQ2. How can machine learning be applied to detect fake accounts on TikTok with both labeled and unlabeled datasets?

To better address this research question, we divide it into three subquestions that explore model selection, performance interpretation, and limitations:

- **RQ2.1** Which machine learning algorithm is implemented in this study, and why was it selected for use in a semi-supervised learning context?
- **RQ2.2** How can the performance of this algorithm be evaluated using standard classification metrics such as accuracy, recall, F1-score, G-Mean, and AUC, given the limited availability of labeled data?
- **RQ2.3** What are the main challenges and limitations of applying a semi-supervised machine learning algorithm in practice?

RQ3. What strategies or improvements, such as data balancing and feature scaling, can be implemented to enhance fake account detection on TikTok?

The first research question is addressed in Section 2.4, which outlines the feature selection process, and is further explored in Section 3.3, where TikTok-specific features are identified and evaluated. The second research question, along with its sub-questions (RQ2.1, RQ2.2, and RQ2.3), is discussed extensively throughout the research. Chapter 2 introduces the theoretical background and justifies the use of a semi-supervised learning approach. More specifically, Section 2.7 presents the self-training algorithm with resampling, while Section 3.1 explains the implementation details of the proposed method. The results and evaluation related to these questions are analyzed in Section 4.2, where the performance of different classifiers, normalization techniques, and sampling methods is assessed. The third research question is primarily explored in Chapter 2, where various strategies such as normalization, resampling techniques, such as Synthetic Minority Oversampling Technique, which generates synthetic samples for the minority class (SMOTE), and Cluster-Based Undersampling Technique, which selects representative samples from the majority class using clustering (CBUTE), are introduced to enhance the performance of classification models.

1.3 Outline and Summary

The main contributions of the research are summarized as follows, providing a foundation for the structure of the remaining chapters:

- **Review of Existing Fake Account Detection Techniques:** Chapter 2 provides a comprehensive review of fake account detection approaches across major social media platforms, including Facebook, Twitter, and Instagram. It discusses the machine learning algorithms commonly employed for such classification tasks and presents the theoretical foundation for adopting semi-supervised learning in contexts where both labeled and unlabeled data are available.

- **Data Preprocessing and Feature Selection Algorithm:** Chapter 2 further outlines the significance of data preprocessing, including the application of Min-Max and Z-Score normalization techniques. To address the issue of class imbalance, Synthetic Minority Oversampling Technique (SMOTE) and Cluster-Based Undersampling Technique (CBUTE) are employed. The chapter also introduces the Recursive Feature Elimination method with a Support Vector Machine classifier (RFE-SVM)[37], which is further enhanced by integrating a Bit-Flip-based local search strategy to optimize feature subset selection.

- **Proposed Semi-Supervised Learning Framework:** Chapter 3 presents the proposed Self-Training Semi-Supervised Resampling (SSSTR) framework, which combines labeled and unlabeled data for the detection of fake accounts on TikTok. This chapter also details the data collection methodology and elaborates on the selected feature set specific to the TikTok.

- **Evaluation and Experimental Analysis:** Chapter 4 reports on the experimental results obtained from applying the proposed framework. A series of evaluations are conducted using multiple classifiers. Performance is assessed using standard metrics such as Accuracy, Recall, G-Mean, AUC, and F1-Score. The chapter presents a comparative analysis through figures and tables to validate the effectiveness of the approach.

- **Discussion and Conclusion:** Chapter 4 and Chapter 5 discuss the interpretation of experimental results in the context of the research questions. They reflect on the strengths and limitations of the proposed SSSTR framework and suggest directions for future research to improve fake account detection methods on TikTok and similar platforms.

Chapter 2

Background and Related Work

This chapter discusses the background knowledge and related work relevant to fake account detection using machine learning techniques. It reviews algorithms applied across platforms such as Facebook, Twitter, Instagram, and TikTok, and introduces key machine learning methods used for fake account detection. Semi-supervised learning approaches for limited labeled data are reviewed, along with preprocessing methods such as Z-Score and Min-Max normalization, resampling techniques like SMOTE and CBUTE, and feature selection strategies including RFE-SVM. The chapter also describes the evaluation metrics used to assess classifier performance, such as Accuracy, Recall, G-Mean, AUC, and F1-Score. Finally, the self-training algorithm and classifiers implemented in this study, including CART, KNN, Naïve Bayes, Gradient Boosting, AdaBoost, and Random Forest, are introduced.

2.1 Machine Learning Algorithms

Machine learning provides the foundation for many classification tasks, including fake account detection on social media platforms. It enables systems to learn from data patterns and improve decision-making without explicit programming. The effectiveness of a machine learning model largely depends on the type of learning strategy it employs and the quality of the available data. Machine learning techniques can be broadly categorized into three types: supervised, unsupervised, and semi-supervised learning [61]. Each type serves a different purpose depending on the availability of labeled data and the complexity of the classification problem.

- **Supervised learning** is a machine learning approach where models are trained on datasets consisting of input-output pairs, using known labels to learn the mapping between features and their corresponding target classes [10]. For example, Kondeti et al. [52] used supervised classifiers such as Support Vector Machines (SVM), Random Forests (RF), and Logistic Regression (LR) to distinguish between fake and genuine Twitter accounts. They trained their models on user-level features like follower count, status count, and language code, achieving a maximum accuracy of 98%.

- **Unsupervised learning** refers to methods that analyze unlabeled data to uncover hidden patterns, groupings, or structures within the dataset [41]. As an example, Zouzou et al. [89] applied an unsupervised clustering method to detect fake followers by examining behavioral patterns, such as users' positions in the follower graph, follow-back ratios, and timing of follow actions. Their approach effectively identified coordinated fake-follower campaigns without requiring labeled data.

- **Semi-supervised learning** is a hybrid approach that leverages a labeled data along with unlabeled data to build predictive models [88]. Predictive models are machine learning algorithms that learn patterns from labeled data and make predictions on new or unseen instances, such as classifying accounts as fake or real. For example, Zeng et al. [83] proposed a semi-supervised self-training framework for detecting bot accounts on Twitter. Their method began with a labeled dataset and iteratively expanded it by adding the most confidently predicted unlabeled instances, which significantly improved the overall classification performance.

As shown in Figure 2.1, supervised learning uses fully labeled datasets, while unsupervised learning operates entirely without labeled data. Semi-supervised learning lies between these two, typically leveraging a small labeled subset (e.g., 1–10%) alongside a much larger pool of unlabeled data [15][88]. This setting allows the model to benefit from limited labeled information while exploiting the structure of the unlabeled data. In this research, we adopt a feature-based method to detect fake accounts [8][83]. Rather than analyzing network connections or content, our approach focuses on account-based features available from the TikTok platform, which include follower count, following count, number of posted videos, and profile-related information, as detailed in Section 2.4.1. We assume that fake accounts tend to show different behavioural patterns than real users. By identifying these differences through selected features, we train a classification model to distinguish between genuine and fraudulent accounts. The next section reviews earlier studies that support this approach.

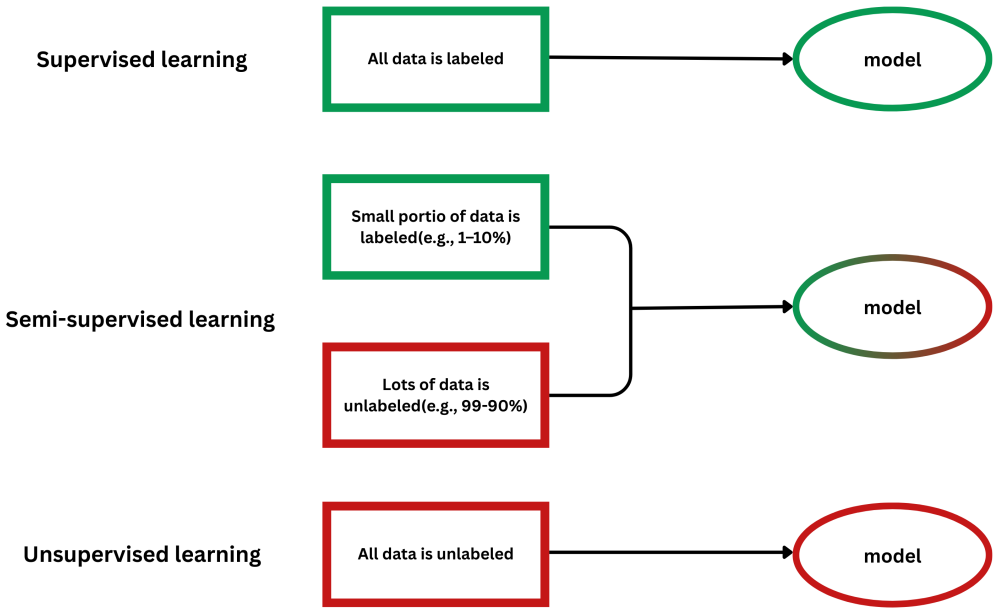

FIGURE 2.1: Supervised vs unsupervised vs semi-supervised machine learning

2.2 Rationale for Semi-Supervised Learning

There are various methods for semi-supervised learning (SSL), with some of the most notable being self-training[54], co-training[59], graph-based approaches[32], and semi-supervised support vector machines(S3VMs)[84]. These methods can effectively use a small amount of labeled data (typically around 5% to 10% of the total data) together with a large amount of unlabeled data to build accurate models. Self-training and co-training have been widely applied in tasks such as text classification and web mining [87], while graph-based methods and Semi-supervised support vector machines (S3VMs) are particularly useful when the structure of the data can support better learning [15]. S3VMs enhance classification by refining the decision boundary using both labeled and unlabeled data[15].

The self-training method, in particular, offers greater flexibility than other SSL techniques because it does not require specific assumptions and relies solely on self-predicted instances for learning. Self-training can be viewed as a self-guided algorithm[55], where the classifier is initially trained on a labeled dataset L to predict labels for an unlabeled set U. The most reliable instances and their predicted labels are then added to L, and the classifier is retrained with the updated set. This process repeats until a predefined stopping condition is satisfied. The iterative expansion of L draws from the instance selection methodology, as instances with more accurate structural information tend to enhance classifier performance[66]. As a result, self-training is an effective way to improve classifier accuracy and address the challenge of labeling training instances[53].

A study on semi-supervised learning for fake account detection[8] addressed the challenge of limited labeled data by introducing a framework that integrates both labeled and unlabeled datasets to improve model performance. The SSL framework defines a labeled instance set $L = \{(l_i, y_i)\}$, where $y_i \in \{\omega_1, \omega_2, ..., \omega_s\}$, and an unlabeled instance set $U = \{u_1, u_2, ..., u_m\}$, where $m \gg n$. By combining L and U into a single training set, SSL leverages information from unlabeled instances to enhance model robustness. This study[8] proposed an enhanced graph-based semi-supervised learning algorithm (EGLSA) to detect fake users on Twitter. The results demonstrated that EGLSA achieved an impressive accuracy of 90.3%, significantly outperforming traditional methods such as k-Nearest Neighbors (KNN, 63.0%), Support Vector Machines (SVM, 63.2%), and Decision Trees (63.1%). For clarity, k-NN classifies data points based on the majority class of their closest neighbors[21], SVM finds the optimal boundary to separate classes by maximizing the margin[20], and Decision Trees split data into branches based on feature values to make decisions [68]. These findings underscore the effectiveness of semi-supervised learning in addressing data imbalance and improving fake user detection in large-scale social networks.

Knauth et al. [50] proposed a self-training framework for detecting crowdturfing accounts on Instagram. Crowdturfing refers to the practice of using coordinated groups of real users, often paid or incentivized, to artificially boost engagement metrics such as likes, follows, and comments. The method starts with a small set of labeled accounts purchased from crowdturfing service providers and uses logistic regression to iteratively add high-confidence pseudo-labeled instances from the unlabeled set. This iterative process continues until a convergence condition is met. Their approach, using only 1–9% labeled data, demonstrated the effectiveness of self-training in detecting coordinated fake engagement behavior in social media environments.

2.3 Data Preprocessing

Data preprocessing is a fundamental step in machine learning workflows, aimed at transforming raw data into a format suitable for model training and evaluation [56]. It addresses common challenges such as inconsistent feature scales, missing values, and class imbalance, all of which can significantly impact the performance of machine learning algorithms [40]. For instance, differ-

ences in feature scales can negatively affect distance-based algorithms, such as K-Nearest Neighbors (KNN), which rely on calculating distances between data points and are sensitive to variations in feature magnitudes. Similarly, optimization-based algorithms like Support Vector Machines (SVMs), which attempt to find the optimal hyperplane that separates classes, can also be disrupted by inconsistent feature scaling.
Additionally, class imbalance, where one class significantly outnumbers the other, can bias classifiers toward the majority class, reducing their ability to correctly predict instances from the minority class [43]. To mitigate these issues, this study employs normalization to standardize feature scales and resampling to balance class distributions. The following subsections provide an overview of the normalization and resampling methods used, specifically Z-score and MinMax normalization, and Synthetic Minority Oversampling Technique (SMOTE)[18] and Cluster-based undersampling techniques (CBUTE)[57].

2.3.1 Normalization Methods

In the data preprocessing stage, all non-numerical features, including textual and Boolean variables, were converted into numerical representations to ensure compatibility with machine learning algorithms. This transformation allows categorical data to be used effectively during model training and evaluation[69].
To make sure that all features contribute equally and to prevent bias from varying value ranges, two normalization techniques were applied: Z Score normalization and Min Max normalization[45]. These approaches scale the feature values to a common range or distribution, which is especially important for algorithms that are sensitive to the magnitude of input values, such as Support Vector Machines and K Nearest Neighbors[64].
The mathematical formulas used for each method are as follows.
Z-Score Normalization: This method transforms each feature by subtracting the mean and dividing by the standard deviation, standardizing the feature so that it has a mean of 0 and a standard deviation of 1. The mean and standard deviation are computed across all data points in the feature, ensuring a consistent scale across the dataset. [47].

$$Z = \frac{X - \mu}{\sigma} \tag{4}$$

where:

- X is the data point,
- μ is the mean of the dataset,
- σ is the standard deviation of the dataset.

Example: Suppose the dataset contains the values 70, 80, 90, 100, and 110. The mean $\mu = 90$ and standard deviation $\sigma = 15$.
For a value $X = 100$:

$$Z = \frac{100 - 90}{15} = \frac{10}{15} \approx 0.67$$

This means the value is 0.67 standard deviations above the mean.
Min-Max Normalization: MinMax normalization linearly transforms each feature to a specified range, typically [0, 1], by scaling the data based on its minimum and maximum values. The

transformation ensures that the minimum value maps to 0 and the maximum to 1, preserving the relative distances between data points. [48].

$$X_{\text{scaled}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \tag{5}$$

where:

- X is the data point,
- X_{min} is the minimum value in the dataset,
- X_{max} is the maximum value in the dataset.

Example: Using the same dataset (70, 80, 90, 100, 110), where $X_{\text{min}} = 70$ and $X_{\text{max}} = 110$, for $X = 100$:

$$X_{\text{scaled}} = \frac{100 - 70}{110 - 70} = \frac{30}{40} = 0.75$$

The value is now scaled to 0.75 in the range [0, 1].

2.3.2 Resampling Methods

In machine learning, classification problems involve assigning input data to predefined categories or classes[10]. Common examples include determining whether an email is spam, recognizing handwritten digits, or diagnosing a medical condition based on patient data. These problems often suffer from imbalanced class distributions, where one class has significantly more instances than others[62].
Class imbalance can adversely affect the performance of many classifiers. For instance, support vector machines may generate biased hyperplanes that favor the majority class[1], decision trees often produce splits that prioritize the majority class, reducing sensitivity to the minority class[17], and neural networks trained on imbalanced data may achieve high overall accuracy while failing to correctly identify minority class instances[14]. A typical example arises in medical diagnosis tasks, where data related to rare diseases is often scarce. In such cases, models may perform well overall but poorly detect the minority class, leading to high false-negative rates and overlooked diagnoses[38].
To address this issue, we applied synthetic minority oversampling techniques (SMOTE)[18] and cluster-based undersampling techniques (CBUTE)[85]. These methods address class imbalance, where the majority class has far more instances than the minority class, by boosting the minority class's representation and reducing the majority class's influence, thus improving model performance [18, 79].

Synthetic minority oversampling technique(SMOTE)

SMOTE is a widely used resampling technique that generates synthetic examples for the minority class to address class imbalance in machine learning datasets [18]. Synthetic examples are artificially created data points formed by interpolating between a minority class instance and its nearest neighbors, producing new samples that lie along the line segments connecting these points in the feature space [28].
By generating more diverse examples of the minority class, this approach helps the model build a clearer boundary between classes and reduces the chance of memorizing specific training examples, which can harm performance on new data[18]. For example, in the Twitter bot dataset

by [22], SMOTE can generate synthetic fake accounts by interpolating features like high posting frequency and repetitive content, improving the classifier's ability to detect rare bot accounts.
The oversampling procedure[43] using SMOTE is outlined as follows[18]:

1. Let $\mathcal{S}$ be the set of all samples from the minority class.
2. For each sample $p \in \mathcal{S}$, compute its k nearest neighbors within $\mathcal{S}$.
3. Randomly choose $r \leq k$ neighbors (with replacement) for each sample p, where r is determined based on the desired oversampling ratio.
4. For each selected neighbor l_N, generate a new synthetic instance l_{new} using the following formula:

   $$l_{\text{new}} = l + \text{rand}(0, 1) \times |l - l_N|$$

   where l is the feature vector of the current sample p, l_N is one of its neighbors, and $\text{rand}(0, 1)$ is a random number between 0 and 1.
5. Add the synthetic instances l_{new} to the dataset, labeling them as minority class samples.

Cluster-based undersampling techniques(CBUTE)

CBUTE removes a subset of majority class instances to balance class distribution, but this can result in significant information loss, particularly when the minority class is small. To mitigate this, K-means clustering is employed to guide the undersampling process. K-means is a widely used unsupervised learning algorithm that partitions data into k clusters by minimizing the variance within each cluster[57]. It iteratively assigns each data point to the nearest cluster centroid and updates the centroids based on the mean of the assigned points.
This clustering approach helps preserve the representative structure of the majority class while reducing its size. Its effectiveness in preserving diversity and reducing bias makes it a suitable choice for undersampling in imbalanced classification tasks[85].
The undersampling procedure[43] using CBUTE is outlined as follows[57]:

1. Let $L_{\text{maj}} = \{l_1, l_2, \ldots, l_p\}$ denote the set of instances in the majority class, where p is the total number of majority instances, and $L_{\text{min}} = \{l_1, l_2, \ldots, l_q\}$ denote the set of instances in the minority class, where q is the total number of minority instances.
2. Then, randomly select q instances from L_{maj} to form the initial set of centroid vectors $\mu = \{\mu_1, \mu_2, \ldots, \mu_q\}$.
3. Initialize each cluster as empty:

   $$\text{Cluster}_t = \emptyset, \quad t = 1, 2, \ldots, q.$$

4. For each instance $l_p \in L_{\text{maj}}$ and for each centroid μ_q, compute the squared Euclidean distance:

   $$d_{pq} = \|l_p - \mu_q\|_2^2.$$

5. Then, assign l_p to the cluster corresponding to the centroid that minimizes d_{pq}; that is, add l_p to the appropriate cluster:

   $$\text{Cluster}_t = \text{Cluster}_t \cup \{l_p\}.$$

6. Recalculate the centroid for each cluster by taking the mean of the instances assigned to that cluster:

$$\mu_q = \frac{1}{|\text{Cluster}_t|} \sum_{l \in \text{Cluster}_t} l.$$

7. Repeat steps 3 and 4 until the centroids converge (i.e., the centroid vector μ no longer changes).

8. Finally, construct the undersampled training set by combining the final centroids with the original minority class instances:

$$L_{\text{new}} = \mu \cup L_{\text{min}}.$$

In summary, addressing class imbalance is essential for improving the performance of classification models, especially in domains where minority classes are underrepresented. SMOTE and CBUTE provide effective solutions by either enriching the minority class with synthetic examples or reducing the dominance of the majority class through representative clustering. By incorporating both oversampling and undersampling strategies, we aim to construct a more balanced dataset that supports better generalization and more reliable classification, particularly in scenarios involving rare or subtle patterns such as fake account detection.

2.4 Feature Selection Algorithm

Feature selection is a process in machine learning that involves choosing the most relevant characteristics, or features, of the data to use in building a model [40]. In fake account detection, features are measurable details from user actions or profile info that help a model spot real accounts from fake ones. For example, the average number of daily posts, how old the account is, the follower-to-following ratio, or the timing of user activity [22]. The goal of feature selection is to improve the model's performance by focusing on the most important features while removing those that are less useful, which can reduce complexity, prevent the model from learning unnecessary patterns, and avoid issues like overfitting, where the model learns irrelevant details and performs poorly on new data [40, 36].

In the context of detecting fake accounts on social networks, selecting appropriate features is crucial, as each platform may exhibit unique behavioral patterns. For instance, a high posting frequency might signal suspicious activity on Twitter, while limited profile information could be a stronger indicator on Instagram [22].

Feature selection methods are generally categorized into three types: filter methods, embedded methods, and wrapper methods [36]. Filter methods evaluate features independently of the model, often by measuring how much each feature correlates with the outcome, such as whether an account is fake or genuine [51]. Embedded methods combine feature selection with the model training process, such as using algorithms that automatically prioritize important features during training [11]. Wrapper methods test different combinations of features by training a model and assessing its performance, selecting the set that produces the best results. An example of this approach is Recursive Feature Elimination (RFE), which iteratively removes the least important features based on the model's performance [34]. By selecting the right features, models can better identify fake accounts across various platforms.

2.4.1 Feature Comparison Across Platforms

In this section, we address RQ1 and provide a brief overview of features used on other platforms, including Instagram, Facebook, and Twitter[63][5][2]. The discussion draws on two key studies for each platform to illustrate the types of features employed and to inform our feature selection for TikTok. These papers and their associated feature tables are referenced to provide insight into effective features for distinguishing fake accounts from genuine ones, and we have adapted some of these features for our TikTok dataset. By analyzing the approaches and features used in prior work, we aim to identify relevant characteristics that can be applied or modified for TikTok's unique environment.

A. Instagram

The researches [63][5] focus on detecting fake profiles on Instagram by applying various machine learning techniques, including supervised learning, unsupervised clustering, and hybrid ensemble methods. Both researches aimed to distinguish real accounts from fake ones based on profile features.

To build datasets for detecting fake Instagram profiles, Muñoz and Pinto [63] and Anklesaria et al. [5] collected public Instagram data via automated web scraping, using custom scripts or external platforms, with manual validation for accuracy. Muñoz and Pinto gathered 936 real profiles from verified sources, such as official programming league pages, and 150 fake profiles from forums, employing Python, Selenium, and the Google Vision API. Anklesaria et al. collected 1,000 profiles (500 real, 500 fake), likely from public figures and celebrities, using scraping tools.

Muñoz and Pinto [63] applied supervised models, unsupervised clustering, and hybrid ensemble techniques, with Random Forest achieving the highest accuracy of 96%. For clarity, Random Forest combines multiple decision trees to enhance accuracy [13], Support Vector Machines (SVM) separate classes by finding an optimal boundary [20], K-Means groups similar data points into clusters [60], and hybrid ensembles integrate multiple models for improved performance [71]. They suggested that larger datasets could further improve results.

Anklesaria et al. [5] split their dataset into 67% training and 33% testing, evaluating several supervised classifiers. Random Forest led with 98% accuracy, outperforming AdaBoost, Multi-Layer Perceptron (MLP), Artificial Neural Networks (ANN), and Stochastic Gradient Descent (SGD). Briefly, AdaBoost weights misclassified data to boost performance [30], MLP and ANN model complex patterns via layered node networks [72], and SGD iteratively optimizes model parameters [12].

Both studies underscore the value of robust data collection and the superior performance of Random Forest in identifying fake Instagram profiles, highlighting the potential of machine learning in social media analysis. The features extracted from Instagram profiles in both studies are reported in Table 2.1, which lists the features used for fake account detection.

The three features listed below require further explanation, as two of them are also included in the TikTok feature set used in this research.

- "Percentage Completed Profile" quantifies how complete a user's profile is by assigning weighted percentages to elements such as profile picture, number of posts, followers, bio, and language detection. A higher percentage indicates a more legitimate-looking account.

- "Nickname Contains Name" measures the similarity between the display name and username using the Jaro distance formula. A higher similarity suggests that the username is more likely to be genuine rather than randomly generated.

Specifically, the Jaro distance formula is used, as shown in equation (1):

$$d_j = \begin{cases} 0 & \text{if } m = 0, \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) & \text{otherwise,} \end{cases} \tag{1}$$

where:

- $|s_i|$ represents the length of the string s_i,
- m is the number of matching characters, and
- t is half the number of transpositions.

- "Nickname Complexity" evaluates the structure of a username by assigning scores to character types (digits, uppercase letters, or others). The final score is adjusted based on how closely the nickname resembles the display name, helping to distinguish between real and fake profiles.
  The score is determined using a function $f(x)$, adapted from the feature scoring approach proposed by Muñoz et al. [63], where:

$$f(x) = \begin{cases} 3, & \text{if } x \text{ is a digit (0-9)}, \\ 2, & \text{if } x \text{ is an uppercase letter (A-Z)}, \\ 1, & \text{if } x \text{ is neither a digit nor an uppercase letter.} \end{cases} \tag{2}$$

  The sum of these scores is first divided by the result of "Nickname contains name". This result is then divided again by the total number of characters in the username.
  The final computation is represented by:

$$\text{Nickname complexity} = \frac{\left(\sum_{i=1}^{n} f(x_i)\right) / \text{Nickname contains name}}{n}, \tag{3}$$

  where n represents the total number of characters in the username.

B. Facebook

Several studies have explore machine learning techniques for detecting fake accounts on Facebook. Two notable works by Albayati and Altamimi [2] and Sallah et al. [74] applied distinct algorithms and datasets to tackle this challenge.
Albayati and Altamimi [2] analyzed a dataset of 982 Facebook profiles (781 real, 201 fake) using supervised and unsupervised learning models. Their supervised methods included K-Nearest Neighbors, which classifies profiles based on their similarity to the closest labeled examples [21]; Support Vector Machines, which separate classes by identifying an optimal boundary [20]; and the ID3 Decision Tree, which builds rules by splitting data based on information gain [68]. The ID3 algorithm achieved the highest accuracy, reaching 97.76% in the first experiment and 97.66% in the second. In contrast, unsupervised clustering methods, such as k-means and k-medoids, which group similar profiles without labeled data [60], yielded lower accuracies between 66.41% and 68.08%.
Sallah et al. [74] evaluated seven supervised models on a dataset of 889 Facebook profiles. These

Features from [63]	Features from [5]
location	Profile Pic
followers	Nums/Length Username
following	Full Name Words
total publications	Bio Length
first publication date	External URL
last publication date	Verified
publications per week	Business
tagging publications	Post
lang	Followers
nickname complexity	Following
nickname contains name	Last Post Recent
percentage completed profile	Post Single Day
user has photo	Index of Activity
private account	Average of Likes
profile photo person	
photos similarity	
photos similarity internet	

TABLE 2.1: Comparison of Instagram user features

included Gradient Boosting, which builds sequential models to correct previous errors [31]; XGBoost, an optimized gradient boosting method with regularization [19]; CatBoost, designed to handle categorical variables efficiently [67]; LightGBM, a fast gradient boosting framework [49]; Random Forest, which aggregates predictions from multiple decision trees [13]; AdaBoost, which improves accuracy by reweighting misclassified examples [30]; and Extra Trees, which enhances generalization through randomized tree construction [33]. CatBoost and Random Forest achieved the highest accuracy of 99.52%. To optimize efficiency, Sallah et al. applied NSGA-II, a genetic algorithm that selects key features by balancing accuracy and model simplicity [24], reducing features by 70% while maintaining high performance.
Both studies demonstrate the superiority of supervised learning, particularly ensemble methods like Random Forest and CatBoost, for detecting fake Facebook accounts. Feature selection techniques, such as NSGA-II, further enhance model efficiency without compromising accuracy. The key features extracted from the Facebook profiles in these studies are summarized in Table 2.2.

C. Twitter

In their study, Ersahin et al. [27] collected a dataset of 1,000 Twitter accounts (real and fake) using the Twitter API. They preprocessed the data by applying Entropy Minimization Discretization (EMD), which converts numerical features into discrete categories to enhance classifier performance[25]. Specifically, EMD was applied to sixteen user-related numerical features, such as tweet frequency or follower count. The Naive Bayes classifier, which predicts account authenticity based on probabilistic relationships between features [70], combined with EMD, achieved the highest accuracy of 90.41%.
Zeng et al.[83] utilized a dataset of 37,438 Twitter accounts from Kaggle, including 12,425 bots and 25,013 humans. They introduced the ”Reputation” feature, the ratio of followers to the sum of friends and followers, to distinguish bots from humans. Normalized and analyzed through cumulative distribution plots, reputation effectively highlighted behavioural differences, with humans scoring higher. Integrated into their semi-supervised framework, this feature achieved a classification accuracy of 96.8%, demonstrating its strong discriminative power.

Features from Paper[2]	**Features from Paper[74]**
Profile Picture	Number of Friends
Work Place	Phototag (identifying individuals in uploaded photos)
Education	Photopost (total count of uploaded photos)
Living Place	Video (number of shared videos)
Check-In	Check-in (user location announcements)
No. of Posts	Sport (number of favorite sports)
No. of Tags	Player (number of favorite players)
Introduction (Bio)	Music (number of favorite music tracks)
No. of Mutual Friends	Film (number of favorite films)
No. of Pages	Series (number of favorite series)
No. of Groups	Book (number of favorite books)
Family/Relationship	Game (number of favorite games)
	Restaurant (number of favorite restaurants)
	Like (number of likes given)
	Group (number of groups joined)
	Post Shared/Post Posted Rate (engagement or virality of posts)
	Education Level (e.g., secondary school, university)
	About Me (information filled in the "About Me" section)
	Family Info (details about family members)
	Gender (e.g., male or female)
	Relationship Status (e.g., single, married, complicated)
	Note (indicates if the user writes and publishes longer-form content)

TABLE 2.2: Comparison of Facebook user features

The specific features used in each study are presented in Table 2.3, allowing for a direct comparison of the characteristics considered in detecting fake Facebook profiles.

Features from Paper[27]	**Features from Paper[83]**
Description	Default profile
Protected	Default profile image
followers_count	Favorites count
friends_count	Followers count
statuses_count	Friends count
favourites_count	Geographic enabled
listed_count	Account ID
Verified	Language
profile_use_background_image	Location
contributors_enabled	Statuses count
default_profile	Verified
default_profile_image	Average tweets per day
is_translator	Account age days
hashtags_average	Account type
mentions_average	
urls_average	

TABLE 2.3: Comparison of Twitter user features

2.4.2 Recursive Feature Elimination with a Support Vector Machine classifier

All features listed in Tables 2.1, 2.2, and 2.3 for the three different social media platforms were selected based on their significance and relevance to each platform. Since there is no existing feature set specifically designed for detecting fake accounts on TikTok, we used these features as references to compile a new feature table tailored to TikTok, which is presented in Table 3.6. To formalize the feature selection process, the Recursive Feature Elimination with Support Vector Machine (RFE-SVM) algorithm is employed[37]. The training data, denoted as X, consists of a collection of social media accounts (e.g., from the TikTok dataset), where each account is described by a set of features such as posting frequency, account age, or number of followers. Let y represent the labels for these accounts, where a label of 1 indicates a fake account and 0 indicates a genuine account. The algorithm iteratively removes features from the initial feature set F, which includes all features in X, based on their importance as determined by a linear Support Vector Machine (SVM) classifier. The SVM computes a weight vector w, where each w_i represents the importance of a feature i, and features are ranked by the absolute values $|w_i|$. The process continues until a predefined number of features, denoted as k, is reached, resulting in a final feature subset S. The RFE-SVM algorithm is detailed in Algorithm 1.

Algorithm 1 Recursive Feature Elimination with SVM (RFE-SVM)

Require: Training data X, labels y, target number of features k
Ensure: Selected feature subset S
Initialize feature set $F \leftarrow$ all features in X
while $|F| > k$ **do**
    Train a linear SVM classifier on X using features in F and labels y
    Compute the weight vector w from the trained SVM model
    Rank features in F by their absolute weights $|w_i|$ in ascending order
    Remove the feature with the smallest $|w_i|$ from F
end while
Set $S \leftarrow F$ **return** S

To evaluate the trade-off between model simplicity and predictive performance, subsets containing 1, 2, 3, 5, and 7 features were selected using Recursive Feature Elimination (RFE) with a linear Support Vector Machine (SVM)[37]. These subset sizes were chosen to assess classification performance across a spectrum of small to moderately large feature sets, drawn from the 14 predictive attributes (excluding the non-informative *username* and the binary target label). The 3, 5, and 7 feature subsets were selected as primary subsets to represent increasing levels of model complexity, corresponding to roughly 21%, 36%, and 50% of the total 14 predictive features, respectively. This approach reflects a commonly used strategy in feature selection to balance computational efficiency with predictive informativeness [73]. Although subsets of 1 and 2 features were evaluated during preliminary experiments, they consistently yielded lower performance, indicating that such minimal configurations were insufficient to capture the complexity required for effective fake account detection. The use of three subset sizes with two-feature increments enabled structured comparisons across varying complexity levels, while avoiding the computational burden of exhaustively testing all possible subset sizes. Even-numbered sizes such as 4 and 6 were not directly selected via RFE, as they were instead explored through the Bit-Flip local search applied around the 5-feature subset. This strategy helped reduce redundancy in the experimental setup.
This stepwise method was supported by performance trends, where AUC scores improved with increasing feature count, from 3 features (e.g., Gradient Boosting, AUC approximately 70%) to 5 features (e.g., K-Nearest Neighbors, AUC 83.36%) to 7 features (e.g., Random Forest, AUC 92.17%) Section 4.1. To refine the subsets produced by RFE, a Bit-Flip local search algorithm was

applied. This method iteratively added or removed a single feature at a time to explore neighboring combinations [37]. The search identified a 4 feature subset (Face Detected, Bio, Jaro Similarity, Video Count) that achieved an AUC of 80.77%, demonstrating that local optimization can yield compact, interpretable feature sets with competitive performance Section 3.2. The selected feature subsets and their corresponding classification accuracies are summarized as follows:

- **RFE with 3 Features:**
  Accuracy: 70.59%
  Selected Features: `Bio`, `Jaro Similarity`, `Nickname Complexity`
- **RFE with 5 Features:**
  Accuracy: 79.02%
  Selected Features: `Face Detected`, `Bio`, `Jaro Similarity`, `Nickname Complexity`, `Video Count`
- **RFE with 7 Features:**
  Accuracy: 93.33%
  Selected Features: `Face Detected`, `Follower Count`, `Bio`, `Jaro Similarity`, `Nickname Complexity`, `Create Time weekday`, `Video Count`
- **Best Selected Features After Local Search (Bit-Flip):**
  Accuracy: 80.59%
  Selected Features: `Face Detected`, `Bio`, `Jaro Similarity`, `Video Count`

These results indicate that increasing the number of features generally enhances classification performance. For example, accuracy improved from 70.59% with 3 features to 93.33% when using 7 features. However, employing the local search method (Bit-Flip) enabled us to balance model complexity and predictive accuracy effectively. Specifically, we achieved an accuracy of 80.59% with just 4 features, surpassing the 70.59% obtained with 3 features using standard Recursive Feature Elimination (RFE). This highlights the advantage of thoughtfully choosing a concise set of features to optimize performance, while ensuring the model remains efficient and easier to interpret.

2.5 Evaluation Metrics for Classifier Performance

To evaluate the generalization performance of our trained classifier on unseen and imbalanced data, we employed several metrics specifically suited for fake account detection [44]. These metrics including Accuracy, Recall, Geometric Mean, AUC (Area Under the Receiver Operating Characteristic Curve), and F1 Score were used to assess and quantify the classifier's effectiveness, following established practices [23].
The evaluation metrics used in this research are designed to measure how well the classifier distinguishes between fake and real TikTok accounts. These metrics are derived from the model's prediction outcomes, using four fundamental values: True Positives (TP), False Negatives (FN), False Positives (FP), and True Negatives (TN). These values are summarized in a confusion matrix, which indicates the number of correct and incorrect predictions made by the model, as shown in Table 2.4. All evaluation metrics are calculated as follows [44, 23]:

- **Recall** (or Sensitivity) measures the proportion of actual fake accounts that were correctly identified:

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

- **True Negative Rate (TNR)** indicates the proportion of real accounts correctly identified:

$$\text{TNR} = \frac{TN}{TN + FP}$$

- **G-Mean** combines Recall and TNR to reflect balanced classification performance across both classes:

$$\text{G-Mean} = \sqrt{\text{Recall} \times \text{TNR}}$$

- **AUC (Area Under the ROC Curve)** measures the classifier's ability to distinguish between the positive (fake) and negative (real) classes across all possible classification thresholds. AUC values range from 0 to 1, where:
  - AUC = 1.0 indicates perfect separation
  - AUC = 0.5 indicates no discriminative ability (random guessing)
  - AUC < 0.5 suggests the model performs worse than random

  Higher AUC values reflect better classifier performance across varying decision boundaries, making it especially useful in imbalanced classification tasks.

Class	**Predicted value**	
	Positive	**Negative**
Actual value	True Positive (TP)	False Negative (FN)
	False Positive (FP)	True Negative (TN)

TABLE 2.4: Confusion matrix for binary classification

2.6 Overview of Applied Classification Techniques

In this section, we present the machine learning classifiers employed in our research. These models were selected to evaluate the effectiveness of the proposed self-training framework using widely accepted classification performance metrics.

- **Classification and Regression Tree (CART):** CART is a decision tree algorithm that splits the dataset based on feature values to create a hierarchical tree structure [58]. Each internal node represents a decision based on a feature, and each leaf node represents a class label. CART is valued for its interpretability and ability to model non-linear relationships.

- **K-Nearest Neighbor (KNN):** KNN is a straightforward algorithm that classifies a new data point by comparing it to the most similar examples in the dataset, known as its nearest neighbors [21]. It looks at the k closest points and assigns the new point to the class that appears most frequently among them. KNN is effective for smaller datasets but its accuracy can be influenced by how distance is measured and the choice of k.

- **Naïve Bayes (NB):** Naïve Bayes is a probabilistic classifier based on Bayes' theorem, assuming conditional independence between features [70]. The predicted class $\hat{y}$ is given by:

$$\hat{y} = \arg\max_{y \in Y} P(y) \prod_{i=1}^{n} P(x_i|y)$$

  This means the algorithm chooses the class y that maximizes the product of the prior probability $P(y)$ and the likelihoods $P(x_i \mid y)$ of each feature. It is particularly effective for high-dimensional data such as text classification.

- **Gradient Boosting (GB):** Gradient Boosting is an ensemble learning technique that builds a strong predictive model by combining several weaker models in a sequential manner [31]. These weaker models, known as *weak learners*, are simple classifiers that perform slightly better than random guessing. Each new model is trained to correct the errors made by the previous one. The overall model is updated step-by-step using the following formula:

  $$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

  Here, $F_m(x)$ is the updated model at stage m, $h_m(x)$ is the newly added weak learner, and γ_m is the learning rate that controls how much influence the new learner has.

- **Adaptive Boosting (AdaBoost):** AdaBoost is an ensemble learning algorithm that combines several weak learners to create a stronger classifier. In each iteration, it pays more attention to the training examples that were previously misclassified by increasing their weights. This way, the next weak learner focuses more on the difficult cases. Over time, AdaBoost builds a sequence of models that collectively improve prediction accuracy and become more robust to noise in the data [30].

- **Random Forest (RF):** Random Forest is an ensemble learning method that builds multiple decision trees using randomly selected subsets of the data and features [13]. Each tree makes a prediction, and the final result is determined by majority voting across all trees. By introducing randomness and combining diverse models, Random Forest reduces overfitting and improves the model's ability to generalize to new data.

2.7 Self-training semi-supervised learning algorithm based on resampling

In this section, we explain the Self-training Semi-Supervised Learning algorithm (SSSTR) from the original paper [83]. This method was initially designed to detect bot accounts on Twitter, where class imbalance and limited labeled data pose significant challenges. Zeng et al. proposed a semi-supervised self-training framework that integrates resampling techniques (SMOTE for oversampling and CBUTE for undersampling) into the iterative learning process to improve classification performance. Although the original study focused on distinguishing bots from humans using Twitter-specific behavioral and content features, the core idea of the SSSTR framework applies to other platforms.
In this research, we adapt the SSSTR algorithm for fake account detection on TikTok, using TikTok-specific features and applying modified preprocessing and feature engineering techniques to better fit our dataset. Our research uses this algorithm as a base and introduces some changes in the preprocessing and the algorithm itself to make it suitable for our features and dataset. The changes and the specific version of the algorithm used in our research are explained in chapter 3. The framework is shown in FIGURE 2.2 and the algorithm consists of four steps:

- First, the raw Twitter dataset is preprocessed:
  - Missing values are handled,
  - Duplicate entries are removed,
  - Relevant features are selected.
- The cleaned dataset is randomly split into two parts:

- 80% is used for training,
- 20% is used for testing (denoted as T).

- From the training set:
  - 10% of the instances are randomly selected to form the labeled dataset (L),
  - The remaining 90% are kept as the unlabeled dataset (U).

- To fix the class imbalance in L, resampling techniques are applied:
  - SMOTE (Synthetic Minority Over-sampling Technique),
  - CBUTE (Cluster-Based Under-sampling Technique),

  (Refer to Section 2.3.2.) After resampling, the balanced labeled dataset is called L-new.

- The SSSTR method is used to expand the labeled data:
  - A classifier is trained on L-new,
  - It is used to predict labels for instances in U,
  - Predictions are ranked by confidence,
  - The top high-confidence instances are selected and added to L-new,
  - These instances are removed from U,
  - This process repeats until U becomes empty.

- Finally, classifiers mentioned in Section 2.6 are trained on the complete labeled dataset and used to make predictions on the test set T.

In the next chapter, we present the algorithm used in our research, based on the SSSTR approach described in this section.

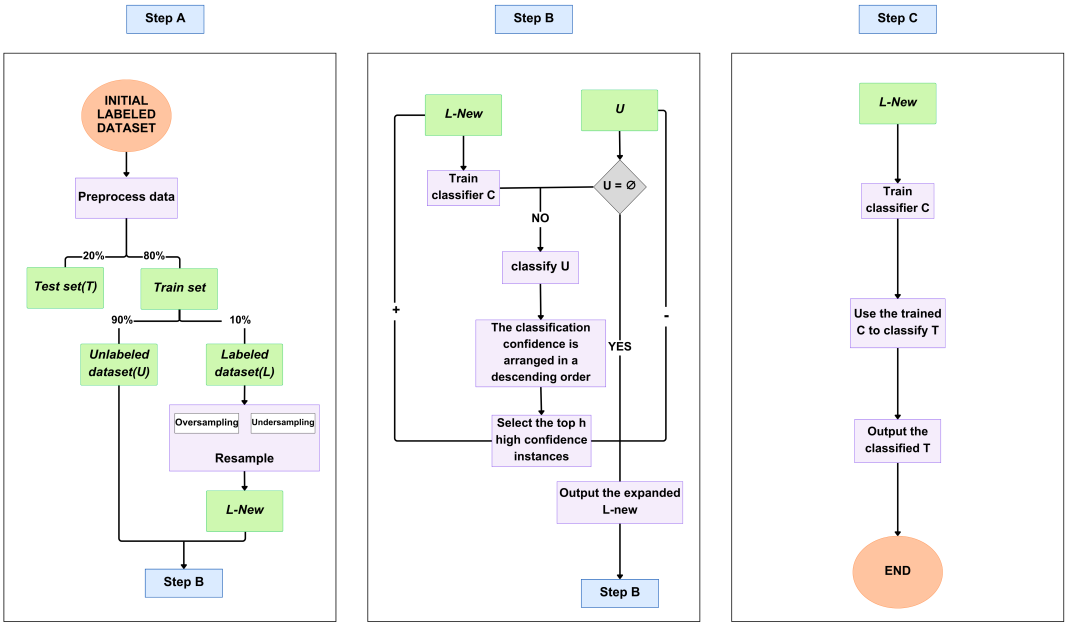

FIGURE 2.2: The classification process of SSSTR based on resampling based on paper[83]

Chapter 3

Proposed solution

This chapter describes the machine learning algorithm called self-training semi-supervised learning, which is used to conduct the experiments in this research. We start by explaining the models and procedures applied to the dataset to distinguish between genuine and fake TikTok accounts. The next sections of this chapter cover how the features were selected, how we handled the class imbalance issue, and what scaling techniques were applied to the data. We also explain the logic behind using semi-supervised learning and how it was implemented in our workflow. The chapter concludes with a summary of how the experiments were organized and how performance was measured throughout the study.

3.1 Proposed Self-Training Method for Fake Account Detection

To address RQ2.1 and RQ3 outlined in this research, we developed a machine learning algorithm aimed at detecting fake accounts on TikTok using a combination of labeled and unlabeled data. The proposed method is inspired by a semi-supervised self-training approach originally applied to Twitter account classification[83].

In the original study, the semi-supervised method was evaluated by splitting a fully labeled dataset into labeled and unlabeled subsets, as explained in Section 2.7. In contrast, our work uses two separate datasets: one that is already labeled and another that contains only unlabeled instances. We then further divide the labeled dataset into a training set and a test set. This approach reflects a common real-world situation, where only a portion of the available data is labeled while the remainder is unlabeled, requiring a learning strategy that can effectively utilize both sources. Following the implementation of the SSSTR, this section outlines the step-by-step process applied to the datasets in this study.

The goal is to evaluate the performance of multiple classifiers in detecting fake TikTok accounts using a semi-supervised self-training strategy, adapted from prior work on Twitter [83]. The classification process is shown in FIGURE 3.1. The process begins with only the labeled dataset and follows the steps described in [83]. Different values of h (ranging from 0 to 0.5) are evaluated (discussed in Subsection 3.1.2), and the value yielding the best performance according to evaluation metrics such as Recall, G-mean, and AUC is selected as the optimal h. After determining the optimal h, the SSSTR process is applied to the full labeled set L and the unlabeled set U, using the selected h.

The details of the following steps are as follows:

First, the collected data are divided into two subsets: a *Labeled set* (L) and an *Unlabeled set* (U). As a preprocessing step, feature normalization is performed using both Min-Max and Z-score scaling techniques to ensure consistent feature ranges. This standardization enhances model stability

and classification performance. Further details on the normalization procedures are provided in Section 2.3.1. The labeled set L is then further split into 80% *Training subset* and 20% *Test subset*, denoted as L_{Train} and L_{Test}, respectively.
In the next step, the two resampling techniques, *SMOTE* and *CBUTE*, are applied independently to balance the class distribution (see Section 2.3.2). This results in a balanced labeled dataset, denoted as L_{new}. Next, a classifier C is trained on L_{new} and used to predict labels for the unlabeled set U. The instances in U that receive the highest prediction confidence (top h) are selected and added to L_{new}, while being simultaneously removed from U.
This self-training process is performed iteratively: the classifier C is retrained on the updated L_{new}, and then reapplied to the remaining unlabeled data. The loop continues until $U = \emptyset$. Once all unlabeled instances have been used and the final labeled set L_{new} is obtained, the classifier C is retrained one last time. It is then applied to the test set L_{Test}, and the predicted labels are produced as the final classification output. The detailed procedure of the SSSTR algorithm is presented in the following subsection.

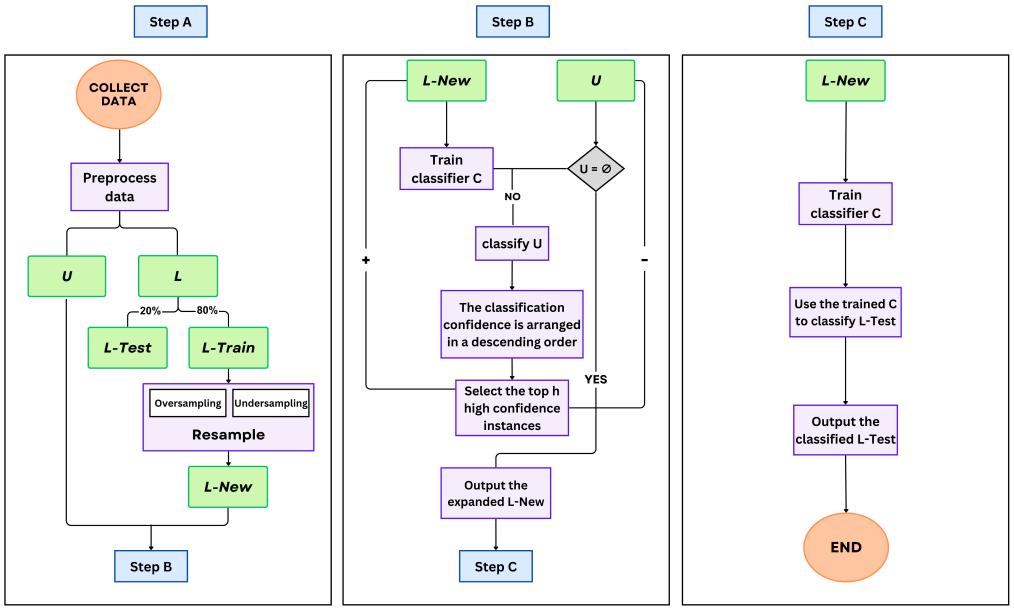

FIGURE 3.1: The classification process of SSSTR based on resampling

Different values of h (ranging from 0 to 0.5) are evaluated (widely discussed in SubSection 3.1.2), and the values with the best performance, according to evaluation metrics such as Recall, G-mean, and AUC, are selected as optimal. This range is chosen to ensure that only high-confidence predictions are added during each iteration of self-training. Limiting h to 0.5 or less helps prevent the inclusion of low-confidence instances, thereby minimizing noise and stabilizing the learning process. After determining the optimal h, the SSSTR process is applied to the full labeled set L and the real unlabeled set U, using the selected h. Finally, the classifier C is retrained on the expanded labeled set and evaluated on the test subset L_{Test}.

3.1.1 Workflow of the proposed SSSTR Method

Algorithm 2 presents the overall workflow of the proposed SSSTR method. The process begins with a normalization step, where feature values in both the labeled dataset L and the unlabeled

dataset U are scaled using Min-Max and Z-score techniques, as described in Section 2.3.1. After normalization, the labeled dataset L is split into a training set L_{Train} (80% of the data), a test set L_{Test} (20%), and an unlabeled dataset U. The training set is used in the resampling and self-training phases, while the test set is reserved for final evaluation.

The proposed method is a self-training semi-supervised learning algorithm that integrates resampling techniques to address both the scarcity of labeled data and the imbalance between class distributions. The algorithm is structured into three main components: resampling the labeled data, iterative self-training, and final classification.

Resampling:

Since the labeled training set L_{Train} in our dataset exhibits class imbalance, resampling techniques are applied to address this issue. The set is first divided into two subsets: L_{maj} and L_{min}, representing the majority and minority class instances, respectively. To balance the class distribution, oversampling is applied to L_{min}, generating synthetic samples based on a sampling ratio N derived from the degree of imbalance. Concurrently, undersampling is applied to L_{maj} using a centroid-based clustering approach (CBUTE), where initial centroids are randomly selected and iteratively refined based on cluster membership until convergence. The resulting centroids form the reduced majority set. The combination of the oversampled minority and undersampled majority subsets yields a balanced labeled training set L_{new}. Detailed descriptions of the SMOTE and CBUTE resampling methods are provided in Section 2.3.2.

Self-Training:

The algorithm proceeds with a semi-supervised self-training phase. A classifier C is trained on the initial labeled set L_{time}, initialized as L_{new}, and then used to predict labels for the unlabeled set U_{time}, initialized as U. The prediction results are sorted by classification confidence, and the top h most confident instances are selected and added to L_{time}, while being simultaneously removed from U_{time}. The classifier is retrained on the updated labeled set, and the process is repeated iteratively until $U_{\text{time}} = \emptyset$. As a result, the final labeled set consists of both manually labeled and confidently pseudo-labeled instances.

Final Classification:

After completing the self-training loop, the classifier C is retrained one final time on the expanded labeled set L_{time} and applied to the held-out test set L_{Test}. The predictions obtained from this stage represent the final classification results of the proposed SSSTR framework. In the identification framework, we employ six commonly used classification models to evaluate the effectiveness of SSSTR. These models include Classification and Regression Tree(CART), K-Nearest Neighbor(KNN), Naïve Bayes(NB), Gradient Boosting(GB), Adaptive Boost(AB), and Random Forest(RF). The experimental results and analysis are presented in Chapter 4.

Algorithm 2 Self-Training Semi-Supervised Learning with Resampling

Require: L, U, C, h

Ensure: The labels of L_{Test}

Normalize features in both labeled set L and unlabeled set U using Min-Max and Z-score scaling techniques.
Split the labeled set L into a training set L_{Train} and a test set L_{Test}.
$L_{\text{Train}} = \{l_1, l_2, ..., l_n\}$; $L_{\text{maj}} = \{l_1, ..., l_p\}$; $L_{\text{min}} = \{l_1, ..., l_q\}$ where $p > q$ and $p + q = n$
Oversampling minority class:
    • For $i = 1, 2, ..., q$, calculate distances between l_i and other L_{min} instances, find K-nearest neighbors.
    • Set sampling ratio N based on class imbalance in L.
    • Construct synthetic instances using Formula (1).
    • Obtain oversampled set L_{over}.
Undersampling majority class via clustering:
    • Randomly choose q instances from L_{maj} as centroids $\mu = \{\mu_1, ..., \mu_q\}$.
    • Initialize empty clusters, assign each l_p to the nearest centroid using Euclidean distance.
    • Recalculate centroids until convergence.
    • Output centroid vectors as L_{under}.
Set $L_{\text{new}} = L_{\text{over}}$ or L_{under}.
Initialize $time = 0$, $L_{\text{time}} = L_{\text{new}}$, $U_{\text{time}} = U$.
while $U_{\text{time}} \neq \emptyset$ **do**
    Train C on L_{time}, classify U_{time}.
    Sort classification confidences and select top-h instances.
    Let S_{time} be the selected samples and labels.
    $L_{\text{time}} = L_{\text{time}} \cup S_{\text{time}}$
    $U_{\text{time}} = U_{\text{time}} \setminus S_{\text{time}}$
    $time = time + 1$
end while
Train C on extended set L_{time}.
Use C to predict labels of L_{Test}.
return The predicted labels of L_{Test}.

3.1.2 Parameter h in Self-Training

Definition and Function:
In the self-training framework, the parameter h, which was introduced and empirically evaluated by Zeng et al. [83], determines the proportion of unlabeled instances added to the labeled dataset at each iteration. Specifically, the model selects the top $h \times |U|$ instances from the unlabeled set U, based on prediction confidence, and incorporates them into the labeled set with their predicted labels. This iterative process gradually expands the training set and improves the classifier.

Significance of h:
Selecting an appropriate value of h is essential to balance learning speed and label quality. A small value of h adds only the most confident predictions, ensuring higher precision but slower progress. In contrast, a large h speeds up learning by labeling more instances, but may introduce noisy or incorrect labels, degrading model performance.

Experimental Range and Reference:
In their original study, Zeng et al. [83] evaluated h at six discrete values within the interval $[0, 0.5]$, specifically:

$$h \in \{0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$$

These values were chosen to explore how different proportions of high-confidence instances affect performance across multiple classifiers. The authors assessed the impact of each setting using standard evaluation metrics such as Recall, AUC, and G-mean, and reported the optimal value of h for each classifier based on empirical performance. The original study used the SSSTR algorithm, described in detail in Section 2.7, to evaluate each candidate h and select the optimal value based on classifier performance.
The selected values are summarized below:

TABLE 3.1: Optimal values of h for each classifier (adapted from [83])

Classifier	**Optimal** h
Classification and Regression Tree (CART)	0.20
K-Nearest Neighbor (KNN)	0.05
Naïve Bayes (NB)	0.40
Gradient Boosting (GB)	0.50
Adaptive Boosting (AdaBoost)	0.50
Random Forest (RF)	0.30

Adopted Configuration:
In this research, we did not adopt the h values reported in the original work by Zeng et al. [83], as their experiments were conducted on a Twitter dataset, which differs significantly from the TikTok dataset used in our research in terms of features and account behavior. Instead, we empirically determined the optimal value of h for each classifier using our own labeled dataset. Specifically, we tested the same set of discrete values within the interval $[0, 0.5]$, as suggested in the original study:

$$h \in \{0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$$

Each value was evaluated based on classifier performance on the labeled data using standard metrics such as Recall, G-Mean, and AUC.
Reported h:
Once the optimal value of h was identified for each classifier, it was used as a fixed parameter during the self-training process. The same value of h was applied when training the model on the full labeled dataset L and when iteratively labeling the unlabeled dataset U in the SSSTR algorithm.

The final selected h values for each classifier, under different combinations of resampling and normalization techniques, are presented in the tables below.

TABLE 3.2: Optimal values of h for each classifier under CBUTE with minmax normalization

Classifier	**Optimal** h
Classification and Regression Tree (CART)	0.20
K-Nearest Neighbor (KNN)	0.10
Naïve Bayes (NB)	0.40
Gradient Boosting (GB)	0.45
Adaptive Boosting (AdaBoost)	0.45
Random Forest (RF)	0.30

TABLE 3.3: Optimal values of h for each classifier under CBUTE with z-score normalization

Classifier	**Optimal** h
Classification and Regression Tree (CART)	0.20
K-Nearest Neighbor (KNN)	0.10
Naïve Bayes (NB)	0.40
Gradient Boosting (GB)	0.40
Adaptive Boosting (AdaBoost)	0.45
Random Forest (RF)	0.30

TABLE 3.4: Optimal values of h for each classifier under SMOTE with minmax normalization

Classifier	**Optimal** h
Classification and Regression Tree (CART)	0.20
K-Nearest Neighbor (KNN)	0.45
Naïve Bayes (NB)	0.40
Gradient Boosting (GB)	0.45
Adaptive Boosting (AdaBoost)	0.45
Random Forest (RF)	0.30

TABLE 3.5: Optimal values of h for each classifier under SMOTE with z-score normalization

Classifier	**Optimal** h
Classification and Regression Tree (CART)	0.20
K-Nearest Neighbor (KNN)	0.05
Naïve Bayes (NB)	0.40
Gradient Boosting (GB)	0.45
Adaptive Boosting (AdaBoost)	0.45
Random Forest (RF)	0.30

3.2 Data Collection

This section outlines the methodology for collecting the datasets used in our study and the limitations encountered due to TikTok API restrictions. We required a labelled dataset to train our model using a semi-supervised self-training approach. Since a labeled TikTok dataset specific to the Netherlands was unavailable, we sourced a labeled dataset not particular to any location from GitHub [7], which originally contained 10,043 accounts (5,029 fake and 5,014 real). However, this dataset included features different from those needed for our research, prompting us to extract usernames and retrieve new attributes via the TikTok API[78] and Apify platform[6].
We encountered two challenges with the TikTok API: first, it did not return features for all usernames, and second, some required features were unavailable through the API. To address the missing features, we attempted to use a secondary API from a website. However, this also failed

```python
client_id = '[redacted]'
client_secret = '[redacted]'
client_key = '[redacted]'

auth_url = 'https://open.tiktokapis.com/v2/oauth/token/'
auth_payload = {
    'client_key': client_key,
    'client_secret': client_secret,
    'grant_type': 'client_credentials'
}
auth_response = requests.post(auth_url, data=auth_payload)
```

FIGURE 3.2: TikTok API authentication using client credentials and a POST request

to return all usernames and only provided a limited number of them. As a result of these combined limitations, the dataset was reduced to 1,699 accounts (758 fake and 941 real), which we used to train our model for testing on a dataset specific to the Netherlands.
Additionally, we collected an unlabeled dataset to test our model, focusing on the followers of popular Dutch TikTok influencers, as identified by a top influencer list [46]. While the original goal was to retrieve the complete follower list of a single influencer, technical constraints of the TikTok API limited access to full follower lists for any username. On November 11, 2024, we collected 23,950 followers across all influencers, limited by the maximum number of usernames the API would return. Following the constraints discussed for the labeled dataset, we narrowed this down to 5,914 unique usernames to use as the unlabeled dataset.

3.2.1 TikTok API Overview

The TikTok API is a programmatic interface provided by TikTok to access publicly available user data, requiring authentication via API keys or OAuth tokens and enforcing rate limits to comply with privacy regulations[82]. The "Query User Info" endpoint retrieves profile details, including fields like `display_name`, `bio_description`, `avatar_url`, `is_verified`, `following_count`, `follower_count`, `video_count`, and `bio_URL`, as listed in Table 3.6.
In this study, these fields were used to extract features for both labeled and unlabeled datasets. However, the API's restrictive nature limited access to complete data; for instance, the "Query User Followers" endpoint could not retrieve full follower lists, and some usernames lacked data. Supplementary tools like Apify [6] provided additional features, including `Create Time Hour`, `Create Time Weekday`, `Friend Count`, while features like `Jaro Similarity`, `Face Detected`, `Nickname Time Gap`, `Nickname Complexity`, `Has Contact Info`, and `Heart Count` were computed manually. To authenticate with the TikTok Research API, client credentials (client ID, secret, and key) were used to obtain a bearer token via a POST request, as shown below. The access token was later used to authorize further user profile requests. This script sends a POST request with the necessary credentials to the token endpoint. Upon success, the response includes a bearer token used for accessing TikTok's protected endpoints.

3.3 TikTok Feature Selection

This section addresses RQ1 by identifying and evaluating the key features that distinguish fake TikTok accounts from real ones, building on the comparisons across platforms discussed in Section 2.4.1. Since there is no existing feature set specifically designed for detecting fake accounts on TikTok, we compiled a new feature set for TikTok, presented in Table 3.6, by extracting data directly from the TikTok API and APIWebsite [6], as well as calculating additional features based on prior work and our own analysis. From the TikTok API, we obtained

the features `Following Count`, `Follower Count`, `Verified`, `Bio`, `Video Count`, `Display Name`, and `Avatar Status`, which capture user profile and activity information. From APIWebsite, we extracted `Heart Count`, `Friend Count`, `Create Time Hour`, and `nickName Modify Time`, providing additional metrics on user engagement and account creation details.

We also calculated several features: `Jaro Similarity` and `Nickname Complexity` were computed following the methodology from a study about Instagram by Anklesaria et al. [5], discussed earlier in Section 2.4.1, who used these to measure the similarity and complexity of usernames for fake account detection. Additionally, we introduced `Face Detected`, `Nickname Time Gap`, and `Create Time Weekday` as novel features derived from our dataset to capture visual, temporal, and behavioral patterns. The features `Create Time Hour` and `Nickname Time Gap` originate from users in different countries; due to varying time zones, these features were excluded to improve the reliability and consistency of our analysis. Figure 3.3 illustrates the relative importance of features extracted from TikTok user data, based on their contribution to classification accuracy using a Random Forest model.

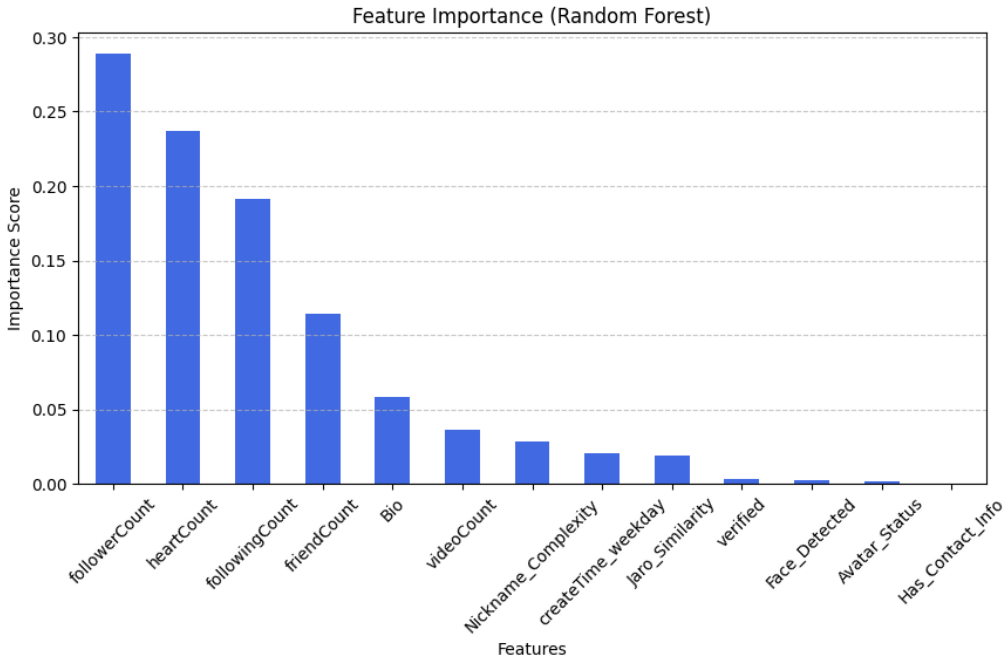

FIGURE 3.3: Feature Importance from Random Forest

To improve the accuracy of fake account detection and reduce unnecessary information, we applied a wrapper-based feature selection method called *Recursive Feature Elimination (RFE)* with a linear Support Vector Machine (SVM) classifier, as originally proposed by Guyon et al. [37] and introduced earlier in Section 2.4.2. This method starts with all available features and removes the least important one at each step, based on the absolute values of the SVM's weight coefficients. The process continues until only a specified number of features remains.

Although Recursive Feature Elimination (RFE) is effective, its greedy nature may lead to the premature removal of important features. To address this limitation, [37] proposed an enhancement by integrating the *Bit-Flip* local search strategy, as described in Section 2.4.2. This approach iteratively adds or removes individual features from the subset, evaluates the resulting combinations, and selects those that improve classification accuracy. By exploring alternative feature sets beyond the greedy path, it helps avoid convergence to local optima and increases the likelihood of

identifying a more optimal solution.
Based on RFE-SVM, four subsets of features were selected as follows:

- `Bio`, `Jaro Similarity`, `Nickname Complexity`
- `Face Detected`, `Bio`, `Jaro Similarity`, `Nickname Complexity`, `Video Count`
- `Face Detected`, `Follower Count`, `Bio`, `Jaro Similarity`, `Nickname Complexity`, `Create Time weekday`, `Video Count`
- `Face Detected`, `Bio`, `Jaro Similarity`, `Video Count`

Further discussion of classifier performance using these selected features is provided in Chapter 4.
We can summarize this chapter in four main points:

- First, we introduced the SSSTR algorithm, a self-training semi-supervised learning method adapted from prior work on Twitter, designed to detect fake TikTok accounts using both labeled and unlabeled data.
- Second, we described the step-by-step implementation of the SSSTR framework, including data normalization, class balancing through SMOTE and CBUTE, iterative self-training, and final classification using six machine learning classifiers.
- Third, we collected and prepared the datasets by using a labeled dataset from GitHub with additional attributes via the TikTok API and Apify, and constructed an unlabeled dataset from followers of Dutch influencers, addressing API limitations and feature availability.

Name	**Description**
Username	Unique identifier for the user account
Face Detected	Indicates if a face was detected in the profile picture
Avatar Status	Indicates the presence and status of the user's avatar picture
Follower Count	The number of followers that the user currently has
Following Count	The number of other TikTok users this user is following
Friend Count	The number of friends (mutual connections) the user has
Heart Count	The total number of likes (hearts) that all of the user's videos have received
Verified	Indicates if the account is verified
Bio	The user's bio description, which may contain information about them
Has Contact Info	Indicates if the user has any contact information listed on their profile
Jaro Similarity	Measures the similarity between the username and the actual name of the user
Nickname Complexity	Complexity of the user's nickname or username (e.g., length, special characters)
Create Time Weekday	The weekday on which the account was created
Nickname Time Gap	The time difference between the creation of the username and the actual name
Video Count	The total number of videos published by the user

TABLE 3.6: Selected Features of TikTok

Chapter 4

Results and Discussion

This chapter presents the results of our study on detecting fake TikTok accounts using the Self-Training Semi-Supervised Learning algorithm (SSST) introduced in Chapter 3. The performance of six classifiers, including Random Forest, Classification And Regression Tree(CART), Gradient Boosting, AdaBoost, K-Nearest Neighbors, and Naïve Bayes, is evaluated across four feature subsets, two normalization methods (Z-score and Min-Max), and two resampling techniques: Synthetic Minority Oversampling Technique (SMOTE) and Cluster Based Undersampling Technique (CBUTE), and a baseline without any resampling, referred to as NORS (No Resampling Strategy). The findings are presented concerning the research objectives, and performance is assessed using five classification metrics: Accuracy, Recall, F1-score, G-Mean, and AUC.

4.1 Evaluation of Feature Subsets and Preprocessing Strategies

This chapter presents the experimental findings of our study on fake account detection on TikTok. Building on the SSSTR framework introduced in Chapter 3, we explore how different combinations of feature subsets, data normalization techniques, and resampling methods affect classifier performance. The goal is to examine patterns, identify high-performing configurations, and interpret results in the context of the research questions introduced in Section 1.2. We focused on three main evaluation metrics: Recall (the ability to correctly identify fake accounts), G-Mean (the balance between correctly identifying both fake and real accounts), and AUC (a measure of how well the model separates fake from real accounts), as these were shown to be effective in the study by Zeng et al. [83] and are formally defined in Section 2.5. To ensure consistency and enable meaningful comparison in a similar social media context, we adopted the same metrics for evaluating our research about TikTok. The following discussion explores how different evaluation metrics, feature subsets, and classifiers influence model behavior.

• Performance Using the 3 Feature Subset

Table 4.1 and Table 4.2 show the classifier performance using only three features (`Bio`, `Jaro Similarity`, `Nickname Complexity`) under Min-Max and Z-Score normalization, respectively. These three features were selected using Recursive Feature Elimination (RFE) with a linear Support Vector Machine (SVM), which ranks features based on their contribution to a linear decision boundary(see 2.4.2). Although these features are not among the top-ranked in the Random Forest importance plot, as shown in Figure 3.3, they were selected by RFE using a linear SVM because the two models evaluate features differently. Random Forest ranks features based on how much they help split data in its decision trees, which allows it to capture complex and nonlinear patterns. In contrast, a linear SVM selects features that help create the clearest linear boundary

between classes. As a result, some features that appear less important to Random Forest can still be useful for a linear classifier like SVM. Additionally, some of the top-ranked features in the Random Forest plot, such as `followerCount` and `heartCount`, show strong correlation. In such cases, RFE with a linear SVM tends to select a smaller, less redundant set of features that still contribute to classification performance. This promotes a more compact and diverse subset by avoiding overlapping information among highly correlated variables.

In Table 4.1, although Recall values were very high, above 95% for most classifiers and sampling methods, G-Mean and AUC were significantly lower. For example, Gradient Boosting and AdaBoost achieved Recall scores of 97.22%, but their G-Mean values dropped below 5%, and AUC scores remained below 52%. This indicates that while these models effectively detected fake accounts, they struggled to correctly classify real users, resulting in poor overall class balance. In comparison, KNN achieved the highest G-Mean of 51.16% without resampling(NORS), and Random Forest recorded a higher AUC of 55.68%, showing a more balanced classification performance.

Table 4.2 shows similar trends, with slight improvements when using Z-Score normalization. KNN again achieved the best AUC, reaching 61.63% when combined with SMOTE. This suggests that Z-Score normalization helped by scaling feature values more evenly, which supports better learning in distance-based models like KNN. SMOTE also contributed by generating synthetic examples of real user accounts, which are underrepresented in the dataset. This helped balance the class distribution during training and allowed the model to better distinguish between fake and real accounts, improving its ability to generalize to unseen data. Despite these improvements, the overall performance remained limited due to the small number of features, which restricted the models' ability to fully capture the complexity of fake account behavior.

In summary, the evaluation with three features highlights the limitations of a minimal feature subset. Although some classifiers, particularly KNN with Z-Score normalization and SMOTE, performed relatively better, the overall results show that such a small feature set does not provide sufficient balance or discrimination. The consistently low G-Mean and AUC values across most configurations indicate that the models were unable to effectively distinguish between fake and real accounts. These outcomes reinforce the need to explore larger and more informative feature subsets in order to improve classification reliability and address the complexity of fake account detection.

Evaluation Index	Sampling Method	CART	RF	GB	AB	KNN	Naive Bayes
Recall	NORS	98.98	99.30	99.97	100.00	93.03	100.00
	SMOTE	98.91	99.46	99.97	100.00	92.15	100.00
	CBUTE	98.88	99.21	99.96	100.00	56.47	99.92
G-Mean	NORS	39.96	34.47	8.53	0.00	51.16	0.00
	SMOTE	39.67	35.05	8.35	0.00	50.66	0.00
	CBUTE	38.53	31.65	7.19	0.00	41.81	4.54
AUC	NORS	57.61	55.68	50.47	50.00	60.85	50.00
	SMOTE	57.45	55.96	50.49	50.00	60.23	50.00
	CBUTE	56.98	54.71	50.39	50.00	43.79	50.26

TABLE 4.1: Performance Metrics of Classifier Performance with 3 Features and Min-Max Normalization under SMOTE and CBUTE and without resampling(NORS)

Note: All values are percentages (%).

Evaluation Index	Sampling Method	CART	RF	GB	AB	KNN	Naive Bayes
Recall	NORS	98.72	99.40	99.96	100.00	93.36	100.00
	SMOTE	99.08	99.30	99.95	100.00	94.48	100.00
	CBUTE	99.13	99.47	100.00	100.00	64.73	99.95
G-Mean	NORS	40.76	35.01	7.02	0.00	51.24	0.00
	SMOTE	40.49	35.79	9.87	0.00	51.80	0.00
	CBUTE	38.56	32.53	8.71	0.00	35.08	5.28
AUC	NORS	57.82	55.91	50.35	50.00	61.03	50.00
	SMOTE	57.86	56.16	50.55	50.00	61.63	50.00
	CBUTE	57.10	55.10	50.50	50.00	44.87	50.28

TABLE 4.2: Performance Metrics of Classifier Performance with 3 Features and Z-Score Normalization under SMOTE and CBUTE and without resampling(NORS)

Note: All values are percentages (%).

• Performance Using the 5 Feature Subset

The results for the 5 feature subset (`Face Detected`, `Bio`, `Jaro Similarity`, `Nickname Complexity`, `Video Count`) are presented in Table 4.3 and Table 4.4, using Min-Max and Z-Score normalization, respectively. As shown in Table 4.3, classifier performance improved substantially compared to the 3-feature subset. For instance, Random Forest with SMOTE achieved 95.79% Recall and 82.08% G-Mean. This corresponds to a 1.31% increase in Recall and a significant 30.28% improvement in G-Mean over the best result obtained with three features. These gains indicate that adding just two more features enhanced the model's ability to identify fake accounts more accurately while also improving class balance, which is critical for developing robust and generalizable detection models.

Gradient Boosting and CART also demonstrated strong and balanced performance, with G-Mean values exceeding 85% in several configurations. These results suggest that both classifiers can effectively distinguish between fake and real accounts when supported by a moderate feature set. KNN also maintained consistent performance, achieving over 82% G-Mean across all resampling methods. This result confirms its robustness across different preprocessing configurations In contrast, Naïve Bayes remained relatively weak, with Recall below 66% and AUC around 82%. This suggests its limited capacity to model the underlying complexity of the data. AdaBoost achieved very high Recall (up to 97.94%) but struggled with overall balance, as indicated by its lower G-Mean and AUC. This implies that while it was highly sensitive to detecting fake accounts, it was more prone to misclassifying real ones.

Table 4.4 shows further performance improvements when using Z-Score normalization. Gradient Boosting combined with SMOTE reached a G-Mean of 85.48% and an AUC of 85.95%, outperforming all corresponding results based on three features. This improvement shows that standardizing the feature variance helped classifiers better separate classes. CART, Random Forest, and KNN also continued to deliver strong and balanced results with Z-Score normalization, supporting the idea that this scaling technique is more effective for the chosen classifiers and dataset. In nearly all settings, SMOTE resulted in higher G-Mean and AUC than CBUTE, reinforcing the benefit of oversampling in handling class imbalance. This suggests that the synthetic examples provided by SMOTE contributed to better generalization and improved detection of minority class instances (i.e., the less frequent class, such as fake accounts). Naïve Bayes showed only minor improvement compared to its performance with three features and continued to lag behind the other classifiers.

Overall, the 5-feature subset significantly improved classification performance. Most classifiers showed better balance in both G-Mean and AUC while maintaining high Recall. These results demonstrate that even a modest increase in feature count enables the model to capture a wider

range of distinguishing patterns between fake and genuine accounts. The combination of SMOTE and Z-Score normalization proved particularly effective, and the consistent performance improvements across classifiers highlight the 5 feature subset as a strong middle ground between model simplicity and detection accuracy.

Evaluation Index	Sampling Method	CART	RF	GB	AB	KNN	Naive Bayes
Recall	NORS	94.15	95.05	95.18	93.82	90.95	65.16
	SMOTE	94.71	95.79	95.04	97.94	92.61	64.68
	CBUTE	95.03	96.12	95.89	97.62	86.36	65.76
G-Mean	NORS	82.68	82.61	85.72	76.01	82.93	80.65
	SMOTE	82.28	82.08	85.18	71.40	82.29	80.32
	CBUTE	81.58	80.40	77.98	73.27	78.64	81.04
AUC	NORS	83.40	83.45	86.21	78.85	83.36	82.52
	SMOTE	83.16	83.10	85.71	75.33	82.95	82.26
	CBUTE	82.59	81.72	79.70	76.61	79.05	82.85

TABLE 4.3: Performance Metrics of Classifier Performance with 5 Features and Min-Max Normalization under SMOTE and CBUTE and without resampling(NORS)

Note: All values are percentages (%).

Evaluation Index	Sampling Method	CART	RF	GB	AB	KNN	Naive Bayes
Recall	NORS	94.23	95.15	94.41	99.29	90.43	65.27
	SMOTE	94.00	95.62	94.78	98.90	90.26	65.60
	CBUTE	95.13	95.88	96.05	98.70	93.18	65.62
G-Mean	NORS	82.94	82.35	85.34	69.14	82.34	80.70
	SMOTE	82.73	82.35	85.48	68.42	82.35	80.86
	CBUTE	82.20	80.81	79.96	69.10	75.29	80.97
AUC	NORS	83.64	83.23	85.79	73.86	82.82	82.55
	SMOTE	83.44	83.33	85.95	73.39	82.77	82.67
	CBUTE	83.12	82.02	81.35	73.71	77.39	82.80

TABLE 4.4: Performance Metrics of Classifier Performance with 5 Features and Z-Score Normalization under SMOTE and CBUTE and without resampling(NORS)

Note: All values are percentages (%).

• Performance Using the 7 Feature Subset

The performance results for the 7 feature subset (`Face Detected`, `Follower Count`, `Bio`, `Jaro Similarity`, `Nickname Complexity`, `Create Time weekday`, `Video Count`) are reported in Table 4.5 and Table 4.6, using Min-Max and Z-Score normalization respectively. This is the most extensive feature configuration explored in this study. As shown in Table 4.5, classifier performance improved across all metrics compared to the 3 and 5 feature subsets. Random Forest with SMOTE achieved 91.74% G-Mean and 92.07% AUC. Gradient Boosting and AdaBoost also delivered strong Recall (above 99%) and competitive G-Mean values, indicating robust performance. KNN continued to lag slightly in balance but maintained strong Recall, while Naïve Bayes showed improved results compared to the 3 and 5 feature settings, particularly in AUC and G-Mean. These improvements reflect the benefit of incorporating additional informative features such as `Follower Count` and `Create Time weekday`, which help distinguish user activity patterns more clearly.

Table 4.6 shows the strongest results overall. Random Forest combined with Z-Score normalization and SMOTE achieved the highest AUC (92.17%) and a G-Mean of 91.85%. Both AdaBoost and Gradient Boosting maintained excellent Recall (above 99%) and improved G-Mean and AUC values compared to their performance with fewer features. Particularly, SMOTE consistently outperformed CBUTE, highlighting its advantage in handling class imbalance by generating diverse synthetic instances of the minority class. This allowed classifiers to generalize more effectively and reduced the risk of overfitting to the majority class(i.e., the more common class, such as real accounts).

Comparing performance improvements across subsets, the jump from 5 to 7 features resulted in substantial gains, up to 30.28% in G-Mean and over 24% in AUC in the best configurations. However, the improvement from 5 to 7 features, while still significant, was more moderate: for example, G-Mean increased from 85.48% (Gradient Boosting, 5 features) to 91.85% (Random Forest, 7 features), an improvement of 7.37%. This shows that while adding more features still improves detection, the improvement becomes smaller and starts to slow down. This slowing improvement suggests that choosing features carefully can create a model that remains simple and interpretable without significantly reducing performance in key metrics such as Recall, G-Mean, or AUC.

In summary, the 7 feature subset produced the highest overall classification performance, particularly when combined with Z-Score normalization and SMOTE. The results confirm that richer feature sets enhance the model's ability to capture complex behavioral patterns associated with fake accounts. However, since performance improvements become less significant beyond 5 features, further increases in model complexity should be carefully weighed against practical considerations such as computational cost and how easily the model can be understood.

Evaluation Index	Sampling Method	CART	RF	GB	AB	KNN	Naive Bayes
Recall	NORS	98.41	99.67	99.37	99.50	99.06	65.91
	SMOTE	98.27	99.64	99.63	99.36	99.10	67.16
	CBUTE	98.80	99.84	99.75	99.61	99.02	65.59
G-Mean	NORS	90.01	91.74	90.56	90.91	72.83	80.27
	SMOTE	90.62	91.45	90.44	91.01	73.11	80.99
	CBUTE	89.68	89.79	88.52	90.09	68.24	80.28
AUC	NORS	90.38	92.07	90.97	91.29	76.32	81.88
	SMOTE	90.93	91.80	90.88	91.37	76.54	82.45
	CBUTE	90.12	90.31	89.16	90.56	73.06	81.97

TABLE 4.5: Performance Metrics of Classifier Performance with 7 Features and Min-Max Normalization under SMOTE and CBUTE and without resampling(NORS)

Note: All values are percentages (%).

Evaluation Index	Sampling Method	CART	RF	GB	AB	KNN	Naive Bayes
Recall	NORS	98.46	99.76	99.55	99.21	98.96	66.38
	SMOTE	98.42	99.80	99.69	99.31	98.87	66.28
	CBUTE	98.74	99.84	99.65	99.64	98.96	66.10
G-Mean	NORS	90.42	91.85	90.62	90.78	74.14	80.40
	SMOTE	90.17	91.58	90.68	91.10	73.99	80.34
	CBUTE	89.13	89.97	88.65	90.47	69.90	80.48
AUC	NORS	90.76	92.17	91.03	91.15	77.28	81.94
	SMOTE	90.54	91.93	91.10	91.45	77.15	81.87
	CBUTE	89.62	90.47	89.27	90.91	74.20	82.07

TABLE 4.6: Performance Metrics of Classifier Performance with 7 Features and Z-Score Normalization under SMOTE and CBUTE and without resampling(NORS)

Note: All values are percentages (%).

• Performance Using the Best Feature Subset

Table 4.7 and Table 4.8 show results using the best feature subset (`Face Detected`, `Bio`, `Jaro Similarity`, `Video Count`), identified through Bit-Flip local search applied after RFE. This subset was selected because it achieved high classification performance while using fewer features than both the 5 and 7 feature configurations. In Table 4.7, Random Forest reached 93.29% Recall and an AUC of 80.50%. Compared to the 5 feature configuration, this shows similar recall (down just 2.5 points from 95.79%) but requires one less feature and offers a more compact representation. In particular, its AUC outperformed the 3 feature setting by more than 25 percentage points. This demonstrates that a smaller, carefully selected subset can offer a strong trade-off between predictive power and model simplicity.

Table 4.8 confirms these findings under Z-Score normalization. Random Forest again performed strongly, achieving 93.47% Recall and 80.62% AUC. Gradient Boosting and AdaBoost maintained high Recall, while Naïve Bayes recorded its best G-Mean (81.38%) in this configuration. These consistent gains across classifiers indicate that the selected four features effectively capture key signals for classification. Moreover, SMOTE continued to outperform CBUTE in nearly all cases, highlighting the advantage of oversampling the minority class.

Overall, this feature set is considered the most effective because it was identified through local optimization (Bit-Flip) and achieved strong performance across all three evaluation metrics: Recall, G-Mean, and AUC, while requiring fewer features than other high-performing subsets. This makes it particularly well suited for practical use cases where reducing computational overhead and maintaining model interpretability are important considerations.

Evaluation Index	Sampling Method	CART	RF	GB	AB	KNN	Naive Bayes
Recall	NORS	93.28	93.29	95.33	98.56	84.15	64.86
	SMOTE	93.71	93.05	95.53	98.62	86.57	65.74
	CBUTE	94.70	93.87	96.16	96.56	79.64	65.74
G-Mean	NORS	80.50	79.44	80.55	69.30	78.05	80.50
	SMOTE	80.59	79.80	80.35	69.44	78.37	81.05
	CBUTE	77.68	75.46	75.17	66.80	75.53	81.00
AUC	NORS	81.39	80.50	81.73	73.83	78.55	82.43
	SMOTE	81.53	80.77	81.57	73.96	79.05	82.87
	CBUTE	79.30	77.36	77.49	72.18	75.69	82.80

TABLE 4.7: Performance Metrics of Classifier Performance with Best Features and Min-Max Normalization under SMOTE and CBUTE and without resampling(NORS)

Note: All values are percentages (%).

Evaluation Index	Sampling Method	CART	RF	GB	AB	KNN	Naive Bayes
Recall	NORS	93.79	93.47	95.42	97.91	86.48	66.26
	SMOTE	93.73	93.91	95.83	99.42	85.20	65.83
	CBUTE	93.12	94.46	95.71	96.61	89.32	64.65
G-Mean	NORS	80.63	79.57	80.49	69.40	77.95	81.38
	SMOTE	80.64	79.43	79.96	68.19	78.49	81.10
	CBUTE	78.29	76.81	75.90	64.74	65.36	80.32
AUC	NORS	81.58	80.62	81.68	73.81	78.61	83.13
	SMOTE	81.57	80.57	81.30	73.18	79.08	82.91
	CBUTE	79.51	78.52	78.00	71.79	69.96	82.26

TABLE 4.8: Performance Metrics of Classifier Performance with Best Features and Z-Score Normalization under SMOTE and CBUTE and without resampling(NORS)

Note: All values are percentages (%).

4.2 Visual Analysis and Discussion of Experimental Results

The effect of feature subset size, sampling strategy, and normalization technique on classification performance is illustrated in Figure 4.1 and Figure 4.2.

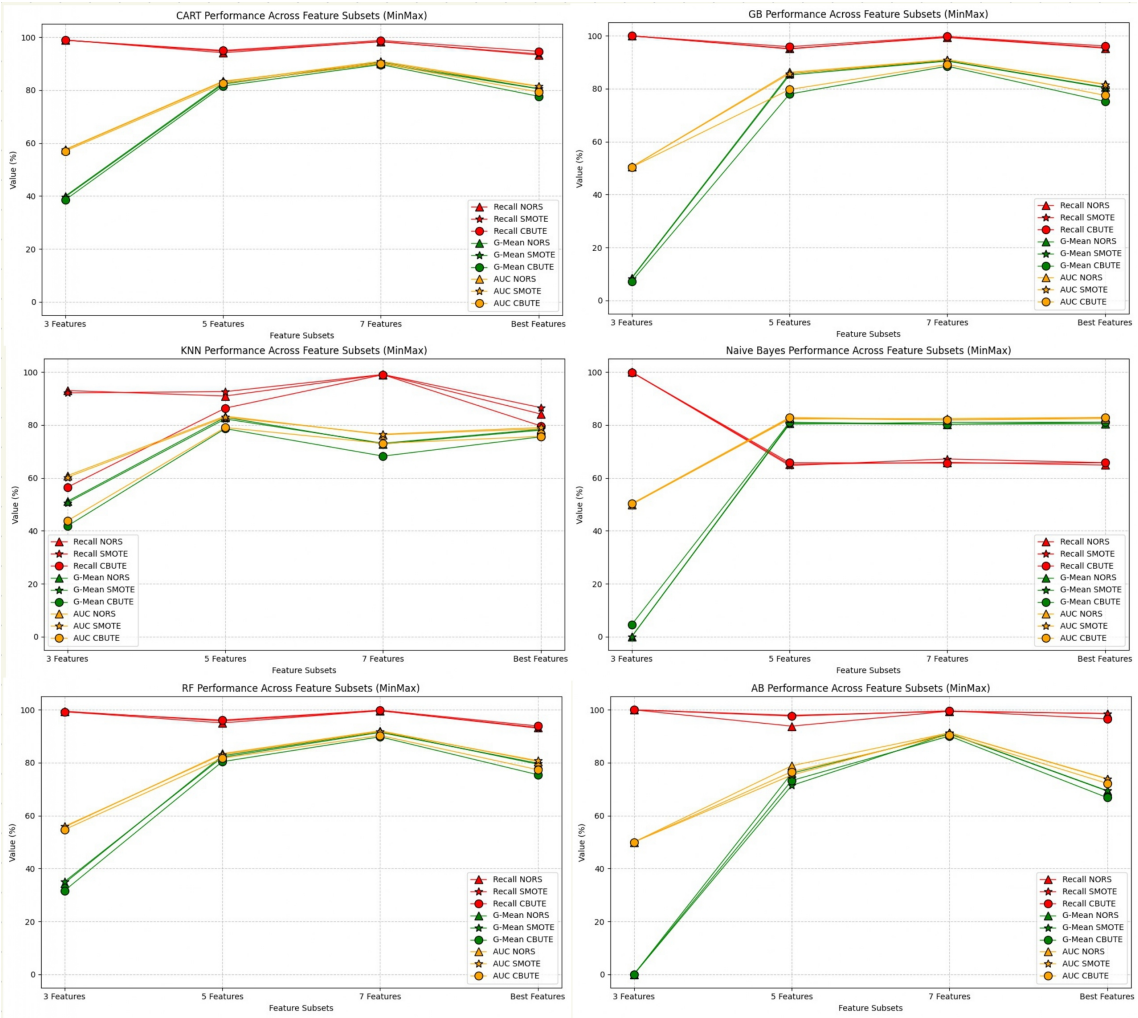

FIGURE 4.1: Comparison of classification performance across different feature subset sizes using six classifiers (CART, RF, GB, AB, KNN, NB) under Min-Max normalization, with using SMOTE and CBUTE and without resampling(NORS).

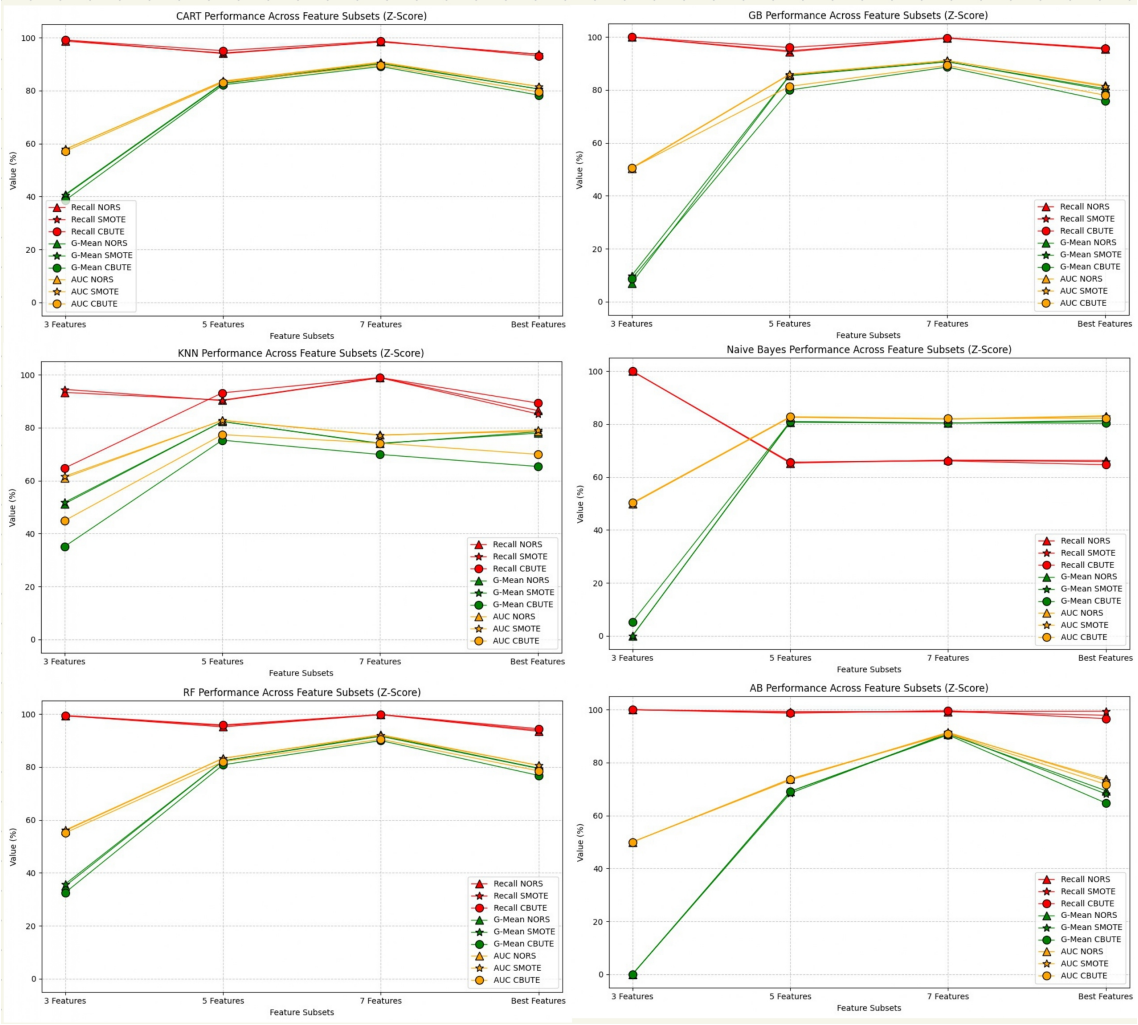

FIGURE 4.2: Comparison of classification performance across different feature subset sizes using six classifiers (CART, RF, GB, AB, KNN, NB) under Z-score normalization, with using SMOTE and CBUTE and without resampling(NORS).

These figures summarize model behavior across six classifiers (CART, RF, GB, AB, KNN, and NB) using three evaluation metrics (Recall, G-Mean, AUC) and four feature subsets. Each classifier is tested under both Min-Max and Z-Score normalization and evaluated with using SMOTE and CBUTE and without resampling. In the legend, different shapes represent the resampling strategies (triangles for NORS, stars for SMOTE, and circles for CBUTE), while colors distinguish the evaluation metrics (red for Recall, green for G-Mean, and yellow for AUC). This visual comparison explains how preprocessing choices and feature richness affect detection effectiveness.

In Figure 4.1, most classifiers demonstrate improved AUC, G-Mean, and Recall as the number of features increases. Random Forest, Gradient Boosting, and AdaBoost show the clearest improvements when combined with SMOTE, particularly when using five or seven features. In contrast, performance with only three features is noticeably weaker, especially under CBUTE. SMOTE contributes to more stable and higher results across most models.

Figure 4.2 displays a similar pattern, but with consistently higher values across all classifiers. Z-Score normalization enhances classifier robustness, particularly when used alongside SMOTE and larger feature sets. Random Forest again outperforms the other classifiers, achieving the highest AUC and G-Mean in nearly all cases. CBUTE continues to underperform relative to SMOTE, while simpler models such as Naïve Bayes and KNN remain more sensitive to changes in sampling and normalization.

The experimental results and comparative evaluations presented throughout this chapter provide a detailed assessment of the SSSTR framework in the context of fake account detection on TikTok. Across all configurations, Random Forest consistently outperformed other classifiers in terms of AUC and G-Mean, particularly when paired with Z-Score normalization and SMOTE resampling. In addition to SMOTE and CBUTE, the framework was also evaluated under a no-resampling condition, referred to as NORS (No Resampling Strategy), which served as a baseline to better assess the impact of resampling techniques. These results highlight the importance of robust normalization, effective handling of class imbalance, and strategic feature selection. The transition from 3 to 7 features demonstrated how additional behavioral and profile-based indicators substantially improve detection performance, while the optimized best-feature subset confirms that high accuracy can still be achieved with a reduced, interpretable set of features.

Together, these results validate the adaptability of the SSSTR method for the TikTok platform and underline the relevance of careful pre-processing and feature design. The transition from 3 to 7 features demonstrated how additional behavioral and profile-based indicators substantially improve detection performance, while the optimized best-feature subset confirms that strong classification performance can still be achieved with a reduced, interpretable set of features, as measured by AUC and G-Mean.

A summary of the best configurations, limitations, and conclusions under Z-Score and Min-Max normalization is provided in Table 4.9 and Table 4.10.

Feature Subset (Z-Score)	Best Classifier + Config	Limitations	Conclusion
3 Features	KNN + SMOTE	G-Mean and AUC remained low across classifiers; even with SMOTE, real users were hard to identify.	Not reliable; performs poorly in class balance despite high recall.
5 Features	Gradient Boosting + SMOTE	Some variation in results across classifiers and resampling methods.	Strong and stable performance; good compromise between model size and accuracy.
7 Features	Random Forest + SMOTE	Performance improved but gains were moderate compared to 5 features. Complexity increased slightly.	Best results overall; strong choice when accuracy is the main goal.
Best (4 Features)	Random Forest + SMOTE	Slight drop in recall and AUC compared to 7 features.	Compact and strong; fewer features but still good performance.

TABLE 4.9: Best Configurations, Limitations, and Conclusions – Z-Score Normalization

Feature Subset (Min-Max)	Best Classifier + Config	Limitations	Conclusion
3 Features	KNN + SMOTE	Class balance was weak; models often failed to correctly classify real users.	Not useful alone; lacks enough information for reliable detection.
5 Features	Random Forest + SMOTE	Misses subtle user behavior features; limited flexibility in some classifiers.	Strong overall; consistent improvement over 3 features.
7 Features	Random Forest + SMOTE	Small performance gain over 5 features; training becomes heavier.	Highest metrics; preferred when maximum performance is needed.
Best (4 Features)	Random Forest + SMOTE	Slight performance decrease in AUC and G-Mean compared to 7 features.	Small and efficient; good results with fewer features.

TABLE 4.10: Best Configurations, Limitations, and Conclusions – Min-Max Normalization

While this study includes detailed performance evaluations for detecting fake accounts on TikTok, its primary goal was not solely to maximize classification metrics. Instead, the focus was on exploring how the SSSTR framework performs when applied to TikTok, a platform with unique behavioral patterns and data access constraints. This involved adapting a method originally developed for Twitter [83] and evaluating its applicability in a different social media environment. Throughout this process, several challenges emerged, including limited access to labeled data, incomplete or missing user attributes due to API restrictions, and sensitivity of classifier performance to normalization methods and feature selection strategies.

These findings provide a clearer understanding of the challenges involved in applying semi-supervised learning in settings where acquiring reliable labeled data is constrained by platform-specific limitations or data availability. The results also highlight the importance of robust preprocessing and design choices. Specifically, the study shows that performance is influenced by the selection of informative features, the choice of normalization method (with Z-Score consistently outperforming Min-Max), and the use of classifiers that maintain stable performance under different sampling and feature configurations. Classifiers such as Random Forest and Gradient Boosting consistently achieved high G-Mean and AUC scores, demonstrating resilience across experimental conditions. These observations suggest that careful integration of preprocessing, resampling, and model selection is critical for effective fake account detection under semi-supervised constraints. Although the findings are specific to TikTok, they may be relevant to other platforms that exhibit similar behavioral dynamics and data access challenges.

Chapter 5

Conclusion and Future Work

This chapter summarizes the main findings and contributions of the research, reflecting on how the proposed approach addressed the identified research questions. It outlines the key insights gained from developing and evaluating the SSSTR framework for fake account detection on TikTok and highlights the significance of the selected features, data preparation strategies, and classification models. Finally, the chapter presents the overall conclusions drawn from the study and suggests directions for future work.

This study explored detecting fake TikTok accounts using a semi-supervised learning method adapted from Twitter's SSSTR framework. The results demonstrate that these accounts can be effectively identified using tree-based classifiers, appropriate feature selection, normalization, and resampling techniques during model training. Random Forest achieved the best performance in terms of AUC and G-Mean, particularly when paired with Z-Score normalization and SMOTE. Among the classifiers tested, Random Forest consistently delivered the best overall performance under the same preprocessing conditions. For example, when using seven features with Z-Score normalization and SMOTE, Random Forest achieved the highest G-Mean (91.58%) and AUC (91.93%) compared to Gradient Boosting (G-Mean: 90.68%, AUC: 91.10%) and CART (G-Mean: 90.17%, AUC: 90.54%). While other models like AdaBoost also performed well in terms of Recall, Random Forest maintained a better balance across all metrics, indicating stronger generalization. These results support the suitability of Random Forest as a robust and effective classifier for fake account detection when combined with proper preprocessing and resampling. Strong preprocessing, maintaining balanced training data, and careful classifier selection proved crucial for high classification performance.

Although the TikTok dataset used in this study was only moderately imbalanced, the application of resampling techniques such as SMOTE and CBUTE still led to consistent improvements in G-Mean and AUC across most classifiers. For instance, with seven features and Z-Score normalization, Random Forest achieved a G-Mean of 91.80% and AUC of 91.93% using SMOTE, compared to 89.97% and 90.47% with CBUTE. While these differences are moderate in this context, they become increasingly critical when working with more severely imbalanced datasets. Thus, maintaining class balance remains an essential step in building robust fake account detection models, especially in domains where minority classes are underrepresented. Using seven features instead of three led to better results, indicating that incorporating richer feature sets can enhance detection. However, a smaller, optimized subset of features also yielded strong outcomes, confirming that accurate and efficient detection is still achievable with fewer, carefully chosen inputs.

The results directly address the research questions presented in Section 1.2. Features such as nickname complexity, face detection, and Jaro similarity supported the findings for RQ1. RQ2 was explored through the adaptation of the SSSTR framework to TikTok, which showed encouraging

semi-supervised performance using both labeled and unlabeled data. However, challenges noted in RQ2.3, highlight the need for better confidence filtering and higher-quality initial labels. RQ3 was addressed through experiments showing that Z-Score normalization and SMOTE significantly improved class balance and detection metrics, especially for tree-based classifiers.
While these results are promising, they do not conclusively establish the SSSTR framework's effectiveness for fake account detection on TikTok. The observed performance indicates potential for semi-supervised learning, particularly when combined with appropriate feature subsets and resampling strategies, although further validation is needed. Given the dynamic and adversarial environment of social media platforms, detection systems must adapt to evolving patterns in account creation, content manipulation, and user interaction aimed at evading detection. To strengthen the proposed framework and address its current limitations, the following directions are suggested for future research:

- **Expanding and Diversifying the Labeled Dataset**
  Future research should collect a much larger and more diverse set of TikTok accounts. This should include accounts from different countries, languages, and content types (such as dance, education, or comedy).

- **Adding TikTok-Specific Behavioral and Content Features**
  This study focused on profile-based features such as nickname complexity, face detection, and video count. However, TikTok is highly dynamic and video-centric. Future studies should incorporate behavioral signals such as engagement metrics (likes, comments, shares), use of popular music, hashtag activity, and temporal behaviors like sudden activity bursts to better capture fake account patterns.

- **Using More Advanced Machine Learning Models**
  While this study relied on classic models like Random Forest and Gradient Boosting, future research could explore deep learning approaches. Convolutional Neural Networks (CNNs) could be used for image or video analysis, while Graph Neural Networks (GNNs) may be effective for modeling user interactions and network structures. These methods may uncover more complex patterns that traditional models cannot detect.

- **Applying the Framework to Other Platforms**
  The SSSTR framework was originally designed for Twitter and adapted here for TikTok. Future work could extend the framework to other short-form video platforms, such as Instagram Reels and YouTube Shorts. Cross-platform evaluation could help determine whether the approach generalizes and reveal both shared and platform-specific patterns in fake account behavior.

In conclusion, this work investigated model performance and the challenges of applying semi-supervised learning to TikTok fake account detection. The results show that factors such as the amount of labeled data, classifier sensitivity to settings, and the quality of feature selection significantly influence outcomes. In particular, models trained with limited labeled data or minimal feature sets often showed imbalanced performance, even when resampling techniques were applied. These insights are valuable for developing more reliable fake account detection systems and can guide future research toward improving model robustness under limited supervision.

Bibliography

[1] Rehan Akbani, Stephen Kwek, and Nathalie Japkowicz. Applying support vector machines to imbalanced datasets. *European conference on machine learning*, pages 39–50, 2004.

[2] M. Albayati and A. Altamimi. MDFP: A Machine Learning Model for Detecting Fake Facebook Profiles Using Supervised and Unsupervised Mining Techniques. *International Journal of Simulation: Systems, Science & Technology*, 20(1):1–10, 2019. URL: https://www.researchgate.net/profile/Ahmad-Altamimi/publication/334194679_MDFP_A_Machine_Learning_Model_for_Detecting_Fake_Facebook_Profiles_Using_Supervised_and_Unsupervised_Mining_Techniques/links/5d1c88e1a6fdcc2462bb448c/MDFP-A-Machine-Learning-Model-for-Detecting-Fake-Facebook-Profiles-Using-S pdf.

[3] A. Alharbi, A. Alotaibi, L. Alghofaili, M. Alsalamah, N. Alwasil, and S. Elkhediri. Security in Social-Media: Awareness of Phishing Attacks Techniques and Countermeasures. In *Proceedings of the 2022 2nd International Conference on Computing and Information Technology (ICCIT)*, pages 10–16. IEEE, January 2022.

[4] K. Anderson. How tiktok is rewriting the world. *The New York Times*, 2020. URL: https://www.nytimes.com/2020/03/10/style/what-is-tiktok.html.

[5] K. Anklesaria, Z. Desai, V. Kulkarni, and H. Balasubramaniam. A survey on machine learning algorithms for detecting fake instagram accounts. In *Proceedings of the 2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, pages 141–144. IEEE, December 2021. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9725724.

[6] Apify. Apify console, 2025. Accessed: 2025-03-26. URL: https://console.apify.com.

[7] R. Aurucci. Fake-TikTok-Account-Detection, 2024. GitHub repository, [Accessed: 2 December 2024]. URL: https://github.com/raffaele-aurucci/Fake-TikTok-Account-Detection.

[8] M. BalaAnand, N. Karthikeyan, S. Karthik, and et al. An Enhanced Graph-Based Semi-Supervised Learning Algorithm to Detect Fake Users on Twitter. *Journal of Supercomputing*, 75:6085–6105, 2019. doi:10.1007/s11227-019-02948-w.

[9] Alazzawi Basel. Detecting fake tiktok accounts: Challenges and research directions. *International Journal of Computer Applications*, 183(25):30–35, 2021.

[10] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[11] Avrim L. Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997. doi:10.1016/S0004-3702(97)00063-5.

[12] L. Bottou. Large-scale machine learning with stochastic gradient descent. *Proceedings of COMPSTAT'2010*, pages 177–186, 2010. doi:10.1007/978-3-7908-2604-3_16.

[13] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[14] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, 2018.

[15] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-supervised learning*. MIT press, 2006.

[16] D. Chatzakou, K. Krombholz, K. A. B., and T. G. Detecting fake accounts in social media platforms: Challenges and solutions. In *Proceedings of the 2017 ACM Conference on Security and Privacy*, pages 23–32, 2017.

[17] Nitesh V Chawla. Data mining for imbalanced datasets: An overview. *Data mining and knowledge discovery handbook*, pages 853–867, 2005.

[18] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[19] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016. doi:10.1145/2939672.2939785.

[20] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. doi:10.1007/BF00994018.

[21] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[22] Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 963–972, 2017.

[23] I. M. De Diego, A. R. Redondo, R. R. Fernández, J. Navarro, and J. M. Moguerza. General performance score for classification problems. *Applied Intelligence*, 52(10):12049–12063, 2022.

[24] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002. doi:10.1109/4235.996017.

[25] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. *Proceedings of the Twelfth International Conference on Machine Learning*, pages 194–202, 1995. doi:10.1016/B978-1-55860-377-6.50031-3.

[26] Nicole B. Ellison, Jessica Vitak, Rebecca Gray, and Cliff Lampe. Cultivating social resources on social network sites: Facebook relationship maintenance behaviors and their role in social capital processes. *Journal of Computer-Mediated Communication*, 19(4):855–870, 2014.

[27] B. Erşahin, Ö. Aktaş, D. Kılınç, and C. Akyol. Twitter Fake Account Detection. In *Proceedings of the 2017 International Conference on Computer Science and Engineering (UBMK)*, pages 388–392. IEEE, October 2017. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8093420.

[28] Alberto Fernández, Salvador García, Mikel Galar, Ronaldo C. Prati, Bartosz Krawczyk, and Francisco Herrera. *Learning from Imbalanced Data Sets*. Springer, 2018. doi:10.1007/978-3-319-98074-4.

[29] B. Foley, S. Sundar, and Z. Liu. The spread of fake profiles and online impersonation on social media. *Journal of Digital Communication*, 45(2):121–135, 2019.

[30] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[31] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.

[32] H. Gan, Z. Li, and W. Wu. Safety-aware graph-based semi-supervised learning. *Expert Systems with Applications*, 107:243–254, 2018. doi:10.1016/j.eswa.2018.04.031.

[33] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006. doi:10.1007/s10994-006-6226-1.

[34] Pablo M. Granitto, Cesare Furlanello, Franco Biasioli, and Flavia Gasperi. Recursive feature elimination for nonlinear svm. *Chemometrics and Intelligent Laboratory Systems*, 83(2):83–90, 2006. doi:10.1016/j.chemolab.2006.01.008.

[35] A. Gupta and R. Kaushal. Towards Detecting Fake User Accounts in Facebook. In *Proceedings of the 2017 ISEA Asia Security and Privacy (ISEASP)*, pages 1–6. IEEE, January 2017. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7976996.

[36] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003. URL: http://jmlr.org/papers/v3/guyon03a.html.

[37] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1):389–422, 2002. doi:10.1023/A:1012487302797.

[38] Guo Haixiang, Gong Yijing, Wang Haoyi, Zhang Yunfei, Wang Xitong, and Liu Bing. Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, 73:220–239, 2017.

[39] J. Han et al. The impact of feature scaling on machine learning algorithms. *Journal of Artificial Intelligence Research*, 73:573–600, 2022.

[40] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 3rd edition, 2011.

[41] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The elements of statistical learning: Data mining, inference, and prediction. *Springer Series in Statistics*, 2009.

[42] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

[43] Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009. doi:10.1109/TKDE.2008.239.

[44] M. Hossin and M. N. Sulaiman. A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining & Knowledge Management Process*, 5(2):1, 2015.

[45] Liangjian Huang, Jing Qin, Yang Zhou, Fei Zhu, Li Liu, and Ling Shao. Normalization techniques in training dnns: Methodology, analysis and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(8):10173–10196, 2023.

[46] HypeAuditor. Top 1000 tiktok influencers in the netherlands, 2024. [Accessed: 11 November 2024]. URL: https://hypeauditor.com/top-tiktok-netherlands/.

[47] T. Jayalakshmi and A. Santhakumaran. Statistical normalization and back propagation for classification. *International Journal of Computer Theory and Engineering*, 3(1):89–93, 2011. doi:10.7763/IJCTE.2011.V3.288.

[48] Sunil Kappal. Data normalization using median & median absolute deviation (mmad) based z-score for robust predictions vs. min-max normalization. *London Journal of Research in Science: Natural and Formal*, 19(4):39–44, 2019.

[49] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, 30:3146–3154, 2017.

[50] Timo Knauth, Christian Reuter, Lukas Brechmann, Florian Hessel, and Philip Zeidler. Are we all in a truman show? spotting instagram crowdturfing through self-training. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management (CIKM)*, pages 2063–2072. ACM, 2022.

[51] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997. doi:10.1016/S0004-3702(97)00043-X.

[52] P. Kondeti, L. P. Yerramreddy, A. Pradhan, and G. Swain. Fake Account Detection Using Machine Learning. In *Proceedings of the 2020 International Conference on Evolutionary Computing and Mobile Sustainable Networks (ICECMSN)*, pages 791–802, Singapore, 2021.

[53] T.T. Li and J. Lu. Improved naive bayes self-training algorithm based on weighted k-nearest neighbor. *Wuhan University Journal of Natural Sciences*, 65(5):465–471, 2019. doi:10.14188/j.1671-8836.2019.05.007.

[54] T.T. Li and J. Lu. Divide-and-conquer ensemble self-training method based on probability difference. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–13, 2020. doi:10.1007/s12652-020-01971-7.

[55] T.T. Li, J. Lu, and W.Y. Fan. Semi-supervised self-training pu learning based on novel spy technology. *Journal of Computer Applications*, 10(39):2822–2828, 2019. doi:10.11772/j.issn.1001-9081.2019040606.

[56] Huan Liu and Lei Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):491–502, 2005. doi:10.1109/TKDE.2005.66.

[57] Stuart Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[58] Wei-Yin Loh. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23, 2011.

[59] J. Lu and Y. Gong. A co-training method based on entropy and multi-criteria. *Applied Intelligence*, 51(6):3212–3225, 2021. doi:10.1007/s10489-020-02014-6.

[60] J. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1:281–297, 1967.

[61] Batta Mahesh. Machine learning algorithms - a review. *International Journal of Science and Research (IJSR)*, 9(1):381–386, 2020. doi:10.21275/ART20203995.

[62] A. More. Survey of resampling techniques for improving classification performance in unbalanced datasets. *arXiv preprint arXiv:1608.06048*, 2016.

[63] S. D. Muñoz and E. P. G. Pinto. A Dataset for the Detection of Fake Profiles on Social Networking Services. In *Proceedings of the 2020 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 230–237. IEEE, December 2020. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9458194.

[64] Babajide Oluleye and Khaled Elleithy. The effect of data scaling and normalization on machine learning algorithms. In *2014 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, pages 1–5. IEEE, 2014.

[65] A. S. Omar, K. O. Ondimu, and A. S. Omar. The impact of social media on society: A systematic literature review. *The International Journal of Engineering and Science*, 13(6):96–106, 2024.

[66] A. Onan and S. Korukoglu. Exploring performance of instance selection methods in text sentiment classification. In *Artificial Intelligence Perspectives in Intelligent Systems*, pages 167–179. Springer, Cham, 2016. doi:10.1007/978-3-319-33625-1_16.

[67] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. Catboost: Unbiased boosting with categorical features. *Advances in Neural Information Processing Systems*, 31:6638–6648, 2018.

[68] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986. doi:10.1007/BF00116251.

[69] Sebastian Raschka. *Python Machine Learning*. Packt Publishing Ltd, 2015.

[70] Irina Rish. An empirical study of the naïve bayes classifier. In *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, pages 41–46, 2001.

[71] L. Rokach. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2):1–39, 2010. doi:10.1007/s10462-009-9124-7.

[72] F. Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, Washington, DC, 1961.

[73] Yvan Saeys, Iñaki Inza, and Pedro Larrañaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, 2007.

[74] A. Sallah, E. A. Abdellaoui Alaoui, A. Hessane, S. Agoujil, and A. Nayyar. An efficient fake account identification in social media networks: Facebook and instagram using nsga-ii algorithm. *Neural Computing and Applications*, pages 1–29, 2024. URL: https://link.springer.com/article/10.1007/s00521-024-10350-8.

[75] Maria A Santos et al. A comprehensive review of class imbalance problem: Handling and learning techniques. *Intelligent Data Analysis*, 22(3):593–627, 2018.

[76] Akash Shah, Sapna Varshney, and Monica Mehrotra. Threats on online social network platforms: classification, detection, and prevention techniques. *Multimedia Tools and Applications*, pages 1–33, 2024.

[77] Statista. Most popular social networks worldwide as of April 2024, by number of monthly active users (in millions), 2024. Accessed: March 4, 2025. URL: https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/.

[78] TikTok. Tiktok for developers. https://developers.tiktok.com/, 2025. Accessed: April 17, 2025.

[79] Yanan Wan, Hong Wang, and Huiguang He. Cluster-based under-sampling technique to handle class imbalance. *Pattern Recognition Letters*, 152:191–197, 2022.

[80] Mika Westerlund. The emergence of deepfake technology: A review. *Technology Innovation Management Review*, 9(11):40–53, 2019.

[81] M. Zafar, S. Ali, and S. A. Shah. Crowdsourced detection of fake accounts: User reporting and moderator systems. *Social Media Research Journal*, 12(3):151–162, 2018.

[82] X. Zeng et al. Data privacy and ethics in social media research: The case of tiktok. In *Proceedings of the 2020 International Conference on Social Media, Wearable, and Web Computing (SMWWC)*, pages 51–59, 2020.

[83] Z. Zeng, T. Li, S. Sun, J. Sun, and J. Yin. A Novel Semi-Supervised Self-Training Method Based on Resampling for Twitter Fake Account Identification. *Data Technologies and Applications*, 56(3):409–428, 2022. URL: https://www.emerald.com/insight/content/doi/10.1108/dta-07-2021-0196/full/html.

[84] Y. Zhan, Y. Bai, and W. Zhang. A p-admm for sparse quadratic kernel-free least squares semi-supervised support vector machine. *Neurocomputing*, 306:37–50, 2018. doi:10.1016/j.neucom.2018.03.069.

[85] Q. Zhou and B. Sun. Adaptive k-means clustering based under-sampling methods to solve the class imbalance problem. *Data and Information Management*, 8(3):100064, 2024.

[86] R. Zhou. Understanding the Impact of TikTok's Recommendation Algorithm on User Engagement. *International Journal of Computer Science and Information Technology*, 3(2):201–208, 2024.

[87] Zhi-Hua Zhou and Ming Li. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1529–1541, 2005.

[88] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, University of Wisconsin-Madison, 2005. URL: http://pages.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf.

[89] Yasser Zouzou and Onur Varol. Unsupervised detection of coordinated fake-follower campaigns on social media. *EPJ Data Science*, 13(1):62, 2024. URL: https://epjdatascience.springeropen.com/articles/10.1140/epjds/s13688-024-00499-6, doi:10.1140/epjds/s13688-024-00499-6.

[90] Lin Zuo and Krishna Jayakar. The landscape of tiktok research: A systematic review. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 1236–1245. IEEE, 2022.