electronics

MDPI

Article

Enhancing Keystroke Dynamics Authentication with Ensemble Learning and Data Resampling Techniques

Xiaofei Wang [1] and Daqing Hou [2,*]

[1] Department of Electrical Computer Engineering, Clarkson University, Potsdam, NY 13676, USA; wangx4@clarkson.edu
[2] Department of Software Engineering, Rochester Institute of Technology, Rochester, NY 14623, USA
* Correspondence: dqvse@rit.edu

Abstract: Background: Keystroke dynamics authentication is a behavioral biometric method that verifies user identity by analyzing typing patterns. While traditional machine learning methods (e.g., decision trees, SVM) have shown potential in this field, their performance suffers in real-world scenarios due to data imbalance and limited recognition of certain classes, undermining system security and reliability. Methods: To address these issues, this study combines the Synthetic Minority Over-sampling Technique (SMOTE) with ensemble learning methods to improve classification accuracy. A Django-based platform was developed for standardized keystroke data collection, generating a balanced dataset to evaluate various classifiers. Experiments: Experiments were conducted using both the Django-collected dataset and the CMU benchmark dataset to compare traditional classifiers with SMOTE-enhanced ensemble learning models, such as Random Forest, XGBoost, and Bagging, on metrics like accuracy, recall, G-mean, and F1-score. Conclusions: The results show that SMOTE-enhanced ensemble learning models significantly outperform traditional classifiers, particularly in detecting minority classes. This approach effectively addresses data imbalance, improving model robustness and security, and provides a practical reference for building more reliable keystroke dynamics authentication systems.

Keywords: keystroke dynamics authentication; ensemble learning; resampling techniques

check for updates

Citation: Wang, X.; Hou, D. Enhancing Keystroke Dynamics Authentication with Ensemble Learning and Data Resampling Techniques. *Electronics* **2024**, *13*, 4559. https://doi.org/10.3390/electronics13224559

Academic Editor: Ahmed Abu-Siada

Received: 16 October 2024
Revised: 13 November 2024
Accepted: 18 November 2024
Published: 20 November 2024

1. Introduction and Motivation

1.1. Introduction

In today's information society, digital security has become a critical issue. With the rapid development of the internet and digital technologies, users frequently transmit and store sensitive information across various online platforms and systems, imposing high demands on information security [1]. Both individual users and organizations must ensure that their digital identities and data are protected from unauthorized access or tampering [2]. Traditional user authentication methods, such as passwords and PINs, are widely used in various systems. However, these methods have significant limitations [3]. Firstly, passwords and PINs are prone to cracking or guessing. Users often choose simple, easy-to-remember passwords, making it easier for attackers to break in through brute force or social engineering. Additionally, passwords are susceptible to being stolen through phishing attacks, keyloggers, or data breaches. Even if users choose complex passwords, they still need to remember numerous different passwords, increasing inconvenience and the likelihood of errors [4].

With technological advancements, biometric recognition technologies have been introduced into the field of user authentication, providing higher security and convenience. Common biometric recognition technologies include fingerprint recognition, facial recognition, and iris recognition. These technologies authenticate users based on unique biological characteristics, making them difficult to forge or replicate, significantly enhancing security.

However, biometric recognition technologies also present some challenges. Firstly, these technologies typically require specialized hardware devices, such as fingerprint scanners, cameras, and iris scanners, adding to the cost and complexity of deployment [5]. Secondly, once biometric data are compromised, they are difficult to change or replace, posing challenges to user privacy protection. In this context, keystroke dynamics, an emerging behavioral biometric authentication method, has drawn the attention of researchers and developers. Keystroke dynamics refers to the behavioral patterns of users when typing on a keyboard, including parameters such as key press time, key intervals, and keystroke force. These behavioral patterns are unique and difficult to mimic, making them a viable basis for identity verification. Compared to traditional passwords and PINs, keystroke dynamics do not require users to remember complex strings; compared to biometric recognition technologies, keystroke dynamics do not require specialized hardware and can be implemented using a standard keyboard. Moreover, keystroke dynamics have several potential advantages. For instance, data collection and verification can occur seamlessly during user input without interrupting normal operations. The authentication process can be conducted in the background, providing a more natural and convenient user experience [6,7].

In summary, the purpose of this research is to address the growing need for more secure and convenient user authentication methods in the context of increasing digital security threats. Keystroke dynamics as an emerging behavioral biometric authentication method, offers unique advantages over traditional methods, such as not requiring users to remember complex strings or relying on specialized hardware. This research focuses on exploring the potential of keystroke dynamics for enhancing authentication systems by leveraging machine learning and data resampling techniques to overcome current challenges, such as data imbalance and limited performance in real-world applications. Given its potential to provide seamless background authentication with minimal user intervention, keystroke dynamics show promise for broad applications, making them a valuable focus for further investigation.

1.2. Motivation

The motivation for utilizing keystroke dynamics in user authentication arises from the urgent need to address several critical challenges in the digital security landscape. Traditional authentication methods, such as passwords and biometric technologies, exhibit significant limitations, including vulnerability to theft, phishing, and the need for specialized hardware [8]. Keystroke dynamics offer a promising alternative, leveraging unique and difficult-to-replicate behavioral characteristics. This section outlines the key motivations for adopting keystroke dynamics, highlights current challenges in the field, and introduces the core contributions of this paper.

Keystroke dynamics analyze how an individual types on a keyboard by measuring parameters such as key press duration, key intervals, and overall typing rhythm. These behavioral patterns are unique to each individual, making keystroke dynamics a viable option for user authentication. The primary motivations for using keystroke dynamics include the following:

(1) Enhanced Security: Unlike passwords, which can be easily guessed, stolen, or shared, keystroke dynamics rely on unique typing behaviors that are significantly harder for attackers to imitate or forge. This characteristic makes keystroke dynamics a promising solution for enhancing security in environments that require continuous and unobtrusive authentication.

(2) Improved User Experience: Keystroke dynamics operate seamlessly in the background as users type naturally, eliminating the need for memorizing complex passwords or carrying additional hardware. This not only simplifies the authentication process but also enhances the overall user experience by minimizing disruptions during authentication.

(3) Cost-Effective and Scalable Implementation: Keystroke dynamics utilize standard keyboards, avoiding the need for expensive biometric hardware. This makes them

an affordable and scalable solution for a wide range of applications, from personal computing to enterprise-level security systems.

Despite its potential, keystroke dynamics research faces two significant challenges. The first is data scarcity and lack of standardized collection methods. One of the primary issues in keystroke dynamics research is the scarcity of large, high-quality datasets. Existing datasets are often small and vary significantly in terms of the data collection methods used, making it difficult to compare results across different studies. The lack of standardized protocols for data collection further exacerbates this issue, limiting the reliability and reproducibility of research findings. The second is high variance in High-Dimensional data with small sample sizes. Traditional machine learning methods, such as Decision Trees (DT) and Support Vector Machines (SVM), often struggle with high-dimensional data, especially when the sample size is small. These methods tend to have high variance, leading to overfitting and poor generalization of new data. This is a critical concern in keystroke dynamics research, where the number of features (dimensions) can be quite large, but the available data points (samples) are limited.

To address these challenges, this paper makes three core contributions:

(1) Development of a Keystroke Data Collection Platform Using Django Framework: We developed a Django-based web platform for standardized keystroke data collection. This platform captures input timings and generates balanced datasets, addressing the issue of data scarcity and variability by providing a consistent, scalable means of collecting keystroke dynamics data from diverse user populations.
(2) Implementation of Resampling Techniques to Handle Class Imbalance: To tackle the issue of class imbalance, we employed resampling methods such as SMOTE and Random Under-sampling. These techniques help to balance the dataset, particularly addressing the underrepresentation of minority keystroke patterns, which is critical for improving model training and enhancing generalization.
(3) Application of Ensemble Learning Methods for Improved Classification: We propose the use of ensemble learning methods, including Random Forest and XGBoost, to enhance keystroke dynamics classification. These methods outperform traditional machine learning algorithms by reducing variance and improving generalization, particularly when dealing with high-dimensional data. This paper demonstrates how ensemble learning can effectively mitigate the risks of overfitting in small, high-dimensional datasets.

The rest of the paper is organized as follows: Section 2 briefly reviews existing literature on traditional user authentication methods, keystroke dynamics authentication, and the application of ensemble learning in behavioral biometric authentication. In Section 3, we introduce the keystroke data collection website developed using the Django framework, describe the feature extraction process, and discuss the classification algorithms and resampling techniques employed. Section 4 details the experimental setup, defines the evaluation metrics, and presents and analyzes the experimental results comparing ensemble learning methods with traditional machine learning methods. The paper is concluded in Section 5.

2. Related Work

2.1. Traditional User Authentication Methods

Traditional user authentication methods have been the cornerstone of digital security for decades. These methods include passwords, Personal Identification Numbers (PINs), and various biometric technologies. Each of these methods has its strengths and limitations, which have been extensively studied and documented in the literature.

Passwords and PINs remain the most prevalent forms of authentication due to their simplicity and ease of implementation. However, these methods are fraught with security vulnerabilities. For example:

Passwords: Users frequently select weak and easily guessable passwords, such as “123456” or “password”, making systems vulnerable to brute force attacks and credential stuffing [3]. Even when users create strong passwords, these can be compromised through phishing attacks, keylogging, or data breaches. The 2012 LinkedIn breach, which exposed 6.5 million hashed passwords, is a notorious example of the risks associated with password-based security [4].

PINs: Similar to passwords, PINs (typically 4–6 digits) are used to secure mobile devices and ATMs. Their short length makes them highly susceptible to guessing attacks, with common combinations like "1234" and "0000" being frequently used [9]. While PINs may be easier to remember, this convenience comes at the cost of security.

To mitigate the weaknesses of passwords and PINs, biometric authentication technologies have been developed. Biometric methods use unique physiological or behavioral traits, such as fingerprints, facial recognition, and iris scans, to verify a user's identity. These methods offer several advantages, including higher security and reduced reliance on memorization. However, biometric systems are not without their challenges:

Hardware Dependency: Biometric systems require specialized hardware, such as fingerprint scanners or facial recognition cameras, which can be costly and difficult to deploy at scale. Vulnerability to Spoofing: Despite their security benefits, biometric systems can be bypassed with sophisticated spoofing techniques. For instance, fingerprint recognition systems have been fooled using high-quality replicas.

Privacy Concerns: Biometric data are highly sensitive. Unlike passwords, which can be easily changed after a breach, compromised biometric data are immutable. The 2015 Office of Personnel Management (OPM) data breach, where the fingerprint data of 5.6 million federal employees were stolen, underscores the severity of biometric data breaches.

In conclusion, while traditional authentication methods like passwords, PINs, and biometrics have played a crucial role in securing digital systems, they each face inherent limitations. Passwords and PINs are prone to various attacks, such as brute force and phishing, while biometric systems, though more secure, introduce challenges related to hardware costs and privacy risks.

These limitations highlight the need for alternative authentication methods that can provide stronger security, better usability, and enhanced privacy protections. Keystroke dynamics, which will be discussed in the following sections, present a promising solution by addressing several of these issues through their use of unique, hard-to-replicate behavioral traits.

2.2. Application of Machine Learning in Keystroke Dynamics Authentication

Traditional machine learning algorithms have been widely applied to keystroke dynamics for distinguishing between legitimate users and impostors based on typing patterns [10,11]. The most commonly used algorithms in this field include Decision Trees (DT), Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), and Neural Networks (NN) [12–14]. While these methods have advanced the field, each has its own strengths and limitations when applied to keystroke dynamics.

Decision Trees (DT) are favored for their simplicity and interpretability [15]. In keystroke dynamics, features such as keypress duration and typing speed are used to build decision rules that help classify users. However, Decision Trees are prone to high variance and overfitting, especially when dealing with high-dimensional data. Techniques such as pruning are commonly applied to reduce overfitting, but these solutions are not always sufficient for complex data [16].

Support Vector Machines (SVM) are well-suited for binary classification tasks and are particularly effective with high-dimensional data. SVMs work by identifying an optimal hyperplane that maximizes the margin between different classes. This makes them a strong candidate for handling the complex feature relationships in keystroke dynamics, such as the interaction between keystroke durations and key intervals. However, SVMs can be computationally expensive, especially with large datasets, and they often struggle with imbalanced data where one class is underrepresented [17,18].

k-Nearest Neighbors (k-NN) is a simple and intuitive non-parametric method used for classification [19–21]. In keystroke dynamics, k-NN compares a user's typing pattern to the stored patterns of known users and assigns a class based on the majority of the k-nearest neighbors. While easy to implement, k-NN becomes computationally intensive with larger datasets and performs poorly in high-dimensional feature spaces due to the curse of dimensionality.

Neural Networks (NN) capture complex, non-linear patterns in data, making them suitable for keystroke dynamics authentication, where intricate relationships between typing behaviors need to be modeled [22,23]. However, they require large amounts of data to avoid overfitting and can be computationally expensive to train.

While traditional machine learning algorithms have laid a solid foundation for keystroke dynamics authentication, they each face inherent challenges. These include high variance (especially in high-dimensional data), sensitivity to imbalanced datasets, and computational complexity.

To address these challenges, our work introduces two key approaches: resampling methods to handle class imbalance and ensemble learning techniques to enhance model robustness. Resampling methods, such as SMOTE, improve the performance of models on underrepresented classes, while ensemble learning, such as Random Forest and XGBoost, helps reduce variance and improve generalization, particularly in high-dimensional datasets. By integrating these approaches, we aim to build more reliable and scalable keystroke dynamics authentication systems.

3. Proposed Method

3.1. Data Collection

To address the challenges of data scarcity and variability in keystroke dynamics research, we developed a comprehensive keystroke data collection platform using the Django framework. This platform provides a standardized and scalable solution for collecting, processing, and organizing keystroke data, ensuring consistency and reliability across datasets, which is critical for advancing keystroke dynamics authentication research.

The platform incorporates several key functionalities to streamline the process of capturing keystroke dynamics:

User Input Collection: Users input their personal information on simulated web interfaces (e.g., airline booking forms, shopping websites, and car rental forms), designed to mimic real-world data entry tasks. This realistic context ensures the collection of natural typing behavior. Each user repeats the input process multiple times to account for natural variations, capturing robust datasets of genuine keystroke patterns.

Simulated Impostor Input Generation: In addition to genuine user data, the platform generates simulated impostor behavior. After completing the genuine input tasks, the system automatically creates variations of the user's previous entries, mimicking potential impostor behavior. This automated process ensures consistency and reduces user burden, providing a comprehensive dataset for training and testing machine learning models.

Keystroke Data Capture: As users input information, the platform records key metrics, such as key press durations and intervals between keystrokes. JavaScript is used to capture these events in real time, with data sent to the Django backend for processing and storage. This ensures that detailed and accurate keystroke data are immediately available for analysis.

Data Processing: The platform automatically processes raw keystroke data into structured features suitable for machine learning. Keypress durations and intervals between consecutive keystrokes are calculated in real time. These features are critical for characterizing typing behavior and distinguishing between legitimate users and impostors.

Dataset Generation: The platform generates both positive and negative class datasets. The positive class consists of genuine keystroke patterns collected from users, while the negative class comprises simulated impostor data, where variations of the original inputs are generated to reflect potential malicious behavior. This balanced dataset is crucial for training and evaluating classification algorithms.

Data Storage and Organization: All collected data, including keystroke metrics and relevant metadata (e.g., user ID, input context, timestamp), are stored in a structured format in the Django database. This organization facilitates efficient data retrieval for both real-time and offline analysis, ensuring scalability and flexibility for machine learning tasks.

Privacy and Security Measures: The platform strictly complies with data protection regulations, including GDPR. All personally identifiable information (PII) is encrypted before storage, ensuring that even in the event of unauthorized access, the data remain secure. Data collection adheres to the principles of data minimization and purpose limitation, ensuring that only necessary information is collected and used. Users are fully informed about how their data will be processed, and explicit consent is obtained. Additionally, users can request the deletion of their data at any time, in accordance with GDPR's "right to be forgotten".

For example, when a user logs into the platform, they are presented with simulated web forms (e.g., an airline booking page) where they enter information such as name, address, and payment details. They repeat this process several times, capturing multiple samples of their typing behavior. After genuine data collection, the platform automatically generates impostor samples by replaying the user's original entries with slight variations. This process ensures that the negative samples are consistent and realistically reflect potential impostor behavior.

By leveraging the Django framework, the platform provides a robust scalable solution for capturing and processing keystroke dynamics data. It addresses key challenges in the field by generating high-quality datasets with both genuine and impostor samples. Furthermore, strict compliance with data privacy regulations ensures that user data are protected throughout the process, meeting the highest standards of data security and privacy.

3.2. Data Resampling

Keystroke dynamics authentication relies on the analysis of typing patterns to distinguish between genuine users and impostors. One of the primary challenges in developing effective authentication systems is the class imbalance in the dataset, where genuine user inputs (positive samples) significantly outnumber impostor inputs (negative samples). This imbalance can lead to biased models that perform well on the majority class but poorly on the minority class. To address this issue, the Synthetic Minority Over-sampling Technique (SMOTE) is applied [24].

SMOTE is an over-sampling technique that addresses class imbalance by generating synthetic samples for the minority class. Instead of simply duplicating minority class samples, SMOTE creates new samples by interpolating between existing ones. The workflow of this algorithm mainly consists of two steps. The first is, for each minority class sample x_i, SMOTE selects k nearest neighbors from the same class. The second is a new synthetic sample x_{new} is generated by randomly selecting one of the k nearest neighbors x_{nn} and interpolating between x_i and x_{nn} as follows:

$$x_{new} = x_i + \epsilon \cdot (x_{nn} - x_i) \tag{1}$$

where ϵ is a random number between 0 and 1. This process is repeated until a balanced distribution of classes is achieved. By generating synthetic samples, SMOTE helps expand the decision space of the classifier, which in turn, leads to improved performance, especially in identifying impostor inputs. Unlike simple duplication, SMOTE produces more diverse samples, which helps reduce overfitting by encouraging the classifier to generalize better to unseen data.

The SMOTE Algorithm 1 process is shown in the following pseudocode and the flow chart of SMOTE Algorithm is shown in Figure 1:

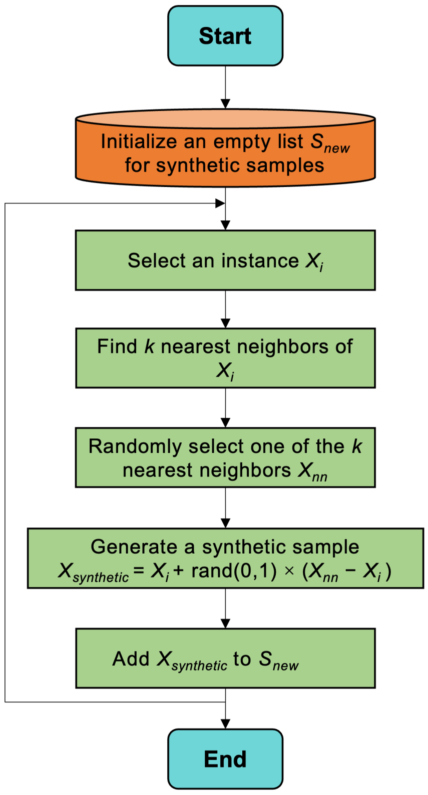
Start
Initialize an empty list S_{new} for synthetic samples
Select an instance X_i
Find k nearest neighbors of X_i
Randomly select one of the k nearest neighbors X_{nn}
Generate a synthetic sample $X_{synthetic} = X_i + \text{rand}(0,1) \times (X_{nn} - X_i)$
Add $X_{synthetic}$ to S_{new}
End

Figure 1. Flow chart of SMOTE Algorithm.

Algorithm 1 SMOTE Algorithm.

```
1:  function SMOTE($X_{min}$, $N$, $k$)
2:      Initialize an empty list $S_{new}$ for synthetic samples
3:      for each sample $x_i$ in $X_{min}$ do
4:          Find $k$ nearest neighbors of $x_i$
5:          for $N$ times do
6:              Randomly select one of the $k$ nearest neighbors $x_{nn}$
7:              Generate a synthetic sample $x_{synthetic} = x_i + \text{rand}(0,1) \times (x_{nn} - x_i)$
8:              Add $x_{synthetic}$ to $S_{new}$
9:          end for
10:     end for
11:     return $X_{min} \cup S_{new}$
12: end function
```

SMOTE is particularly advantageous for keystroke dynamics authentication because it generates synthetic samples that represent plausible impostor behaviors by interpolating between existing negative class samples. This helps create a more balanced and diverse dataset, improving the classifier's ability to detect impostor inputs. By addressing the class imbalance, SMOTE enhances the overall robustness of the system, allowing for more accurate detection of both genuine and impostor users.

In addition to SMOTE, combining this resampling technique with ensemble learning methods can further improve classification performance. This combination leverages the strengths of both approaches—SMOTE's ability to balance the dataset and ensemble learning's ability to reduce variance and improve model generalization.

3.3. Ensemble Learning

Ensemble learning is a powerful technique in machine learning that combines multiple models to improve overall performance. By aggregating the predictions of several base models, ensemble methods can achieve better accuracy, robustness, and generalization compared to individual models. Ensemble learning methods have shown significant promise in the field of keystroke dynamics authentication. These methods combine multiple base models to produce a stronger overall model that often performs better than any single model alone. In our method, the main ensemble learning techniques used include random forest, XGBoost, and bagging. These methods help improve the accuracy and robustness of classification by leveraging the strengths of various underlying models and mitigating their respective weaknesses.

3.3.1. Random Forest

Random forest is an ensemble learning method that constructs multiple decision trees during the training process and outputs the classification results of a single tree [25]. It is particularly useful for handling high-dimensional data and reducing model variance. Random forest first performs guided sampling, creating multiple guided samples from the original training data. For each guiding sample, grow a decision tree. At each node, select a random subset of features and choose the best segmentation from that subset. In the end, each tree votes for a category, with a majority vote serving as the final prediction.

The mathematical formulation for the Random Forest can be expressed as follows:

$$\hat{f}(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} T_b(\mathbf{x}) \tag{2}$$

where $\hat{f}(\mathbf{x})$ is the final prediction, B is the number of trees, and $T_b(\mathbf{x})$ is the prediction from the b-th tree.

3.3.2. XGBoost

XGBoost (Extreme Gradient Boosting) is an optimized implementation of gradient boosting that enhances performance and speed [26]. It builds an ensemble of weak learners (usually decision trees) in a sequential manner, where each subsequent model corrects the errors of the previous ones. The XGBoost algorithm first starts with an initial prediction and then adds new trees sequentially to minimize the loss function. Each tree is trained to predict the residual errors of the current model. Finally, regularization is applied to avoid overfitting. The objective function in XGBoost includes a loss function and a regularization term:

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{k=1}^{t} \Omega(f_k) \tag{3}$$

where l is the loss function, y_i is the true value, $\hat{y}_i^{(t)}$ is the prediction at iteration t, and $\Omega(f_k)$ is the regularization term for the k-th tree.

3.3.3. Bagging

Bagging is an ensemble technique that aims to reduce the variance of a model by training multiple models on different subsets of the training data and then averaging their predictions [27]. Each subset is created using bootstrap sampling. The Bagging algorithm can be summarized as follows: (1) Create multiple bootstrap samples from the original training data. (2) Train a separate model on each bootstrap sample. (3) For classification, aggregate the predictions by majority voting. For regression, average the predictions. The mathematical formulation for Bagging can be expressed as follows:

$$\hat{f}(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}_b(\mathbf{x}) \tag{4}$$

where $\hat{f}(\mathbf{x})$ is the final prediction, B is the number of bootstrap samples/models, and $\hat{f}_b(\mathbf{x})$ is the prediction from the b-th model.

3.4. Combining SMOTE with Ensemble Learning Methods in Keystroke Dynamics Authentication

SMOTE is a powerful method for addressing class imbalance by generating synthetic samples for the minority class, which is a common issue in keystroke dynamics datasets. When combined with ensemble learning methods such as Random Forest, XGBoost, and Bagging, SMOTE significantly enhances classifier performance by providing a balanced training dataset that enables the model to learn effectively from both majority and minority classes.

In the context of keystroke dynamics authentication, this combination provides several distinct advantages, i.e., Improved Detection of Minority Class Instances: By generating synthetic samples for the minority class (e.g., impostor typing patterns), SMOTE ensures that the ensemble models can better learn the distinctions between genuine users and impostors. This reduces the risk of biased models that might otherwise fail to detect impostor attempts due to insufficient minority class data. Enhanced Model Robustness: Ensemble learning methods inherently provide robustness by aggregating predictions from multiple base models, which makes the final model less susceptible to noise and outliers. When combined with the balanced dataset created by SMOTE, the robustness is further enhanced, leading to more reliable and consistent predictions. Synergistic Benefits of Ensemble Methods: The ensemble methods utilized—Random Forest, XGBoost, and Bagging—leverage the diversity in base models to achieve better generalization. SMOTE complements this by ensuring that each base model has access to a well-balanced dataset, which helps in accurately learning complex typing behaviors and identifying impostor patterns. Reduced Overfitting and Enhanced Generalization: SMOTE's synthetic sample generation helps prevent models from overfitting to the majority class by providing varied examples for training. The combination with ensemble methods that already reduce variance (such as Bagging and Random Forest) results in models that generalize better to unseen data, which is crucial for the real-world applications of keystroke dynamics authentication.

The overall workflow of combining SMOTE with ensemble learning methods is illustrated in Figure 2. The flowchart shows the major steps in our approach, from setting up the Django web application to extracting user input, applying SMOTE for data augmentation, training ensemble learning models, and evaluating their performance.

The entire algorithm process is shown in the following pseudocode:

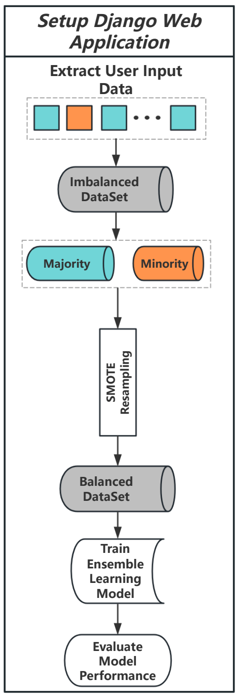

Figure 2. Flow chart of Algorithm 2.

Algorithm 2 Data Classification using Django, SMOTE, and Ensemble Learning.

```
1:  Step 1: Setup Django Web Application
2:  Initialize Django project and create necessary models and forms
3:  Define a view to handle user input and data processing
4:  Step 2: Extract User Input Data
5:  function GET_USER_DATA(request)
6:      Extract data from Django form
7:      Validate and preprocess the input data
8:      return preprocessed data
9:  end function
10: Step 3: Apply SMOTE for Data Augmentation
11: function APPLY_SMOTE(data)
12:     Import SMOTE from imbalanced-learn library
13:     Apply SMOTE to balance the dataset
14:     return augmented data
15: end function
16: Step 4: Train Ensemble Learning Model
17: function TRAIN_ENSEMBLE_MODEL(data)
18:     Import necessary libraries
19:     Split data into training and testing sets
20:     Define ensemble model
21:     Train the model on the training data
22:     return trained model
23: end function
24: Step 5: Evaluate Model Performance
25: function EVALUATE_MODEL(model, test_data)
26:     Predict on test data
27:     Calculate evaluation metrics
28:     return metrics
29: end function
30: Step 6: Output Evaluation Metrics
31: function OUTPUT_METRICS(metrics)
32:     Display metrics on Django web page
33: end function
34: Main Execution
35: function MAIN(request)
36:     user_data ← GET_USER_DATA(request)
37:     balanced_data ← APPLY_SMOTE(user_data)
38:     model ← TRAIN_ENSEMBLE_MODEL(balanced_data)
39:     metrics ← EVALUATE_MODEL(model, balanced_data)
40:     OUTPUT_METRICS(metrics)
41: end function
```

3.5. Computational Complexity Analysis

To thoroughly understand the computational time consumption of the proposed algorithm for data classification using Django, SMOTE, and ensemble learning, a detailed complexity analysis is provided. The computational complexity can be divided into several key components: the extraction of user input data, the application of SMOTE for data augmentation, the training phase of the ensemble learning models, and the evaluation of the model's performance.

The process begins with the extraction of user input data. During this step, data from Django forms are retrieved and preprocessed. Assuming that the input size is denoted by n, where n is the number of samples collected, the computational complexity of this step is linear, as each input data point is processed once. Therefore, the complexity of this step is $\mathcal{O}(n)$.

Next, the SMOTE algorithm is applied for data augmentation. The SMOTE algorithm generates synthetic samples by finding k-nearest neighbors for each minority class sample. This involves calculating the distances between each data point and every other point, resulting in a computational complexity of $\mathcal{O}(n^2)$ for a brute-force approach. Once the nearest neighbors are identified, synthetic samples are generated with an additional linear cost of $\mathcal{O}(n)$. Thus, the overall complexity of the SMOTE application is dominated by the nearest neighbor search, yielding a total complexity of $\mathcal{O}(n^2)$.

Following the SMOTE step, the ensemble learning model is trained. The training phase includes splitting the dataset into training and testing subsets, which is a linear operation with complexity $\mathcal{O}(n)$. For training the ensemble models, such as Random Forest or XGBoost, the time complexity depends on both the size of the dataset and the number of trees T. Random Forest training, for instance, has a complexity of $\mathcal{O}(T \cdot n \log n)$, where T is the number of trees and n is the number of training samples. XGBoost follows a similar complexity due to the sequential nature of boosting. Thus, the training phase has a complexity of approximately $\mathcal{O}(n \log n)$.

Once the model is trained, the evaluation of model performance occurs where predictions are made on the test set. Assuming the size of the test set is proportional to the training set, the complexity of making predictions is linear, or $\mathcal{O}(n)$. Calculating the evaluation metrics, such as accuracy, precision, recall, and F1-score also requires iterating over the predictions, making this step linear as well, with complexity $\mathcal{O}(n)$.

In summary, the overall computational complexity of the proposed algorithm is dominated by the SMOTE application step, which has the highest time complexity of $\mathcal{O}(n^2)$. The training phase and data extraction contribute lower-order terms with complexities of $\mathcal{O}(n \log n)$ and $\mathcal{O}(n)$, respectively. Therefore, the final computational complexity of the entire algorithm is as follows: $\mathcal{O}(n^2)$.

This quadratic complexity is primarily due to the SMOTE algorithm's nearest neighbor search, which can be further optimized using more efficient nearest neighbor algorithms if necessary.

4. Experimental

In this section, we present the experimental setup, the evaluation metrics used, and the comparative analysis of different machine learning methods applied to keystroke dynamics authentication. The experiments are designed to evaluate the effectiveness of combining SMOTE with ensemble learning methods.

4.1. Experimental Setup

4.1.1. Datasets

For our experiments, we used two datasets with distinct characteristics to evaluate the generalizability of our proposed method:

1. Django Collected Dataset: This dataset was collected using our custom keystroke data collection platform built on the Django framework. Users were asked to input their personal information on simulated web interfaces (e.g., airline booking pages, shopping websites, and car rental forms). Each user repeated the input process multiple times to generate genuine keystroke patterns. Simulated impostor inputs were also created to represent the minority class. Specifically, each dataset is composed of an equal proportion of genuine user inputs and impostor user inputs. The impostor users are divided into several groups, generating the same amount of forged data. The diversity of impostor users produces fake class labels with different characteristics, which helps train a more robust classifier.

2. CMU [28] Keystroke Dynamics Benchmark Dataset: This well-known dataset contains keystroke dynamics data collected from users typing predefined text. It should be noted that the original CMU dataset does not contain labels. For classification purposes, we generate a sub-dataset with the same number of subjects for each subject as genuine and the rest as impostors.

Although both the Django and CMU datasets are generated from user-simulated inputs, they have fundamental differences, which lead to distinct uses. Specifically, the CMU dataset consists of unlabeled inputs from multiple users for a single password, where each user records their input, and the key information lies in the varying keystroke characteristics among different users. In contrast, the Django dataset is a labeled dataset where each user is required to input both their own answer and the previous user's answer, providing clear classification labels that better support model training. Additionally, the inputs from different users are not limited to the same content, extending beyond passwords, making the dataset more versatile and rich in information. These differences allow for a comprehensive analysis of our method's performance across both uniform and diverse typing contexts, highlighting the generalizability and robustness of the proposed approach.

4.1.2. Evaluation Metrics

To assess the performance of the classification models, we used the following evaluation metrics:

1. Accuracy: The proportion of correctly classified instances out of the total instances.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

where TP is true positives, TN is true negatives, FP is false positives, and FN is false negatives.

2. Recall: The proportion of actual positive instances correctly identified by the model.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{6}$$

3. G-Mean: The geometric mean of the sensitivity (recall) and specificity, providing a balanced measure of model performance across both classes.

$$\text{G-Mean} = \sqrt{\text{Recall} \times \text{Specificity}} \tag{7}$$

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{8}$$

4. F1-Score: The harmonic mean of precision and recall, providing a single measure of performance that balances both metrics.

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{9}$$

$$\text{Precision} = \frac{TP}{TP + FP}. \tag{10}$$

5. ROC (Receiver Operating Characteristic) analysis: various classifiers were evaluated on two keystroke dynamics datasets (CMU and Django) to assess their performance in distinguishing genuine users from impostors. For each classifier, the ROC curve was generated by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) across different threshold values. The Area Under the Curve (AUC) was calculated as a summary metric to compare classifier performance. A higher AUC indicates better discriminatory ability, with values closer to 1 signifying stronger performance in identifying minority classes accurately. This analysis allowed for a detailed comparison of each method's effectiveness, especially in handling data imbalance challenges within the keystroke dynamics datasets.

All methods were evaluated using five-fold cross-validation on every sub-dataset of each dataset to obtain the final results. In addition, we provided an average ranking of different methods across all datasets in the experimental section, making it easier to observe the specific position of each method. The formula for calculating the average ranking for each method is as follows:

$$\text{Average Ranking} = \frac{\sum_{i=1}^{n} R_i}{n}. \tag{11}$$

where R_i represents the rank of the method on dataset i, and n is the total number of datasets. This formula allows us to compute the overall performance of each method across multiple datasets.

4.1.3. Comparative Methods

We compared the performance of the following methods:

(1) Decision Trees (DT)
(2) Support Vector Machines (SVM)
(3) k-Nearest Neighbors (k-NN)
(4) Random Forest (RF)
(5) XGBoost
(6) Bagging
(7) SMOTE + Random Forest (SMOTE RF)
(8) SMOTE + XGBoost (SMOTE XGBoost)
(9) SMOTE + Bagging (SMOTE Bagging)

The inclusion of both standard machine learning models and ensemble learning methods (combined with SMOTE) allows for a thorough comparison. The goal was to determine whether applying SMOTE with ensemble models could yield superior results compared to traditional classifiers and how effectively these combinations could improve the detection of impostor inputs in keystroke dynamics authentication.

4.2. Experimental Results

4.2.1. Method Comparison Results

The experimental results for the Django and CMU datasets are summarized in Tables 1–4. These tables present the average scores and ranks of all methods across four evaluation metrics: accuracy, F-score, G-mean, and recall. The Django dataset, collected using our keystroke data collection platform, represents a balanced dataset. Table 2 shows the average scores of all methods, while Table 1 presents the average ranks. From Table 1, it is evident that ensemble learning methods combined with SMOTE (OverRF, OverXGBoost, and OverBagging) consistently outperform traditional machine learning methods (Decision Tree, SVM, k-NN) and standalone ensemble methods (Random Forest, XGBoost, Bagging) across all metrics. Specifically, OverRF achieved the highest accuracy (98.66), F-score (98.97), and G-mean (98.39), demonstrating the effectiveness of combining SMOTE with ensemble learning. The improvement in classification performance can be attributed to the balanced nature of the Django dataset, which allows the models to learn from a representative sample of both genuine and impostor inputs. The use of SMOTE helps address the class imbalance issue, ensuring that the minority class is well-represented during training.

Table 1. The average rank of all methods on 4 metrics in the Django dataset.

Method	Accuracy	F-Score	G-Mean	Recall
DT	3.85	4.00	3.94	2.87
SVM	5.66	5.79	5.84	3.85
KNN	7.16	7.37	6.90	2.15
RF	1.11	1.15	1.15	1.18
XGBoost	2.87	3.00	3.00	2.11
Bagging	2.37	2.44	2.42	1.58
SMOTE RF	1.06	1.10	1.06	1.11
SMOTE XGBoost	2.66	2.77	2.77	2.05
SMOTE Bagging	2.58	2.66	2.79	2.35

Table 2. The average score of all methods on 4 metrics in the Django dataset.

Method	Accuracy	F-Score	G-Mean	Recall
DT	90.59	92.58	89.92	89.52
SVM	81.72	80.85	77.68	85.89
KNN	77.55	78.40	79.36	93.95
RF	98.39	98.78	97.97	97.18
XGBoost	93.41	94.83	92.95	92.74
Bagging	94.89	95.88	94.72	95.16
SMOTE RF	98.66	98.97	98.39	97.98
SMOTE XGBoost	94.09	95.34	93.61	93.15
SMOTE Bagging	94.76	95.94	93.83	92.34

Table 3. The average rank of all methods on 4 metrics in the CMU dataset.

Method	Accuracy	F-Score	G-Mean	Recall
DT	7.86	7.92	6.76	6.43
SVM	7.96	7.98	8.57	8.57
KNN	6.61	6.73	7.16	7.06
RF	4.14	4.14	6.08	6.08
XGBoost	1.82	1.86	3.27	3.16
Bagging	5.25	5.31	5.94	5.76
SMOTE RF	2.41	2.55	3.08	3.04
SMOTE XGBoost	2.00	2.16	1.63	1.43
SMOTE Bagging	5.96	6.16	2.31	2.00

Table 4. The average score of all methods on 4 metrics in the CMU dataset.

Method	Accuracy	F-Score	G-Mean	Recall
DT	98.97	99.48	85.04	73.48
SVM	98.82	99.40	51.35	41.08
KNN	99.21	99.60	82.09	68.34
RF	99.43	99.71	83.74	71.59
XGBoost	99.70	99.85	93.15	87.05
Bagging	99.37	99.68	86.11	75.00
SMOTE RF	99.62	99.81	93.36	87.51
SMOTE XGBoost	99.70	99.85	96.84	94.00
SMOTE Bagging	99.23	99.61	95.96	92.71

The Django dataset was collected using our keystroke data collection platform and represents a balanced dataset. The average scores and average ranks of all methods are presented in Table 1 and Table 2, respectively.

From Table 1, it is evident that ensemble learning methods combined with SMOTE (OverRF, OverXGBoost, OverBagging) consistently outperform traditional machine learning methods (Decision Trees, SVM, k-NN) and standalone ensemble methods (Random Forest, XGBoost, Bagging) across all metrics:

SMOTE RF achieved the highest accuracy (98.66), F-score (98.97), and G-mean (98.39), demonstrating the effectiveness of combining SMOTE with ensemble learning. The improvement in classification performance can be attributed to the balanced nature of the Django dataset, which allows models to learn effectively from both genuine and impostor inputs. The use of SMOTE helps address the class imbalance issue, ensuring that the minority class is well-represented during training. This enhances the ability of models to correctly identify impostor inputs without sacrificing performance for genuine inputs.

The CMU dataset is inherently imbalanced, with a higher proportion of impostor inputs compared to genuine inputs. The average ranks and average scores and average scores of all methods are presented in Table 3 and Table 4, respectively.

From Table 3, the results indicate that while ensemble learning methods combined with SMOTE (OverRF, OverXGBoost, OverBagging) still perform well, the overall rankings for accuracy and F-score are generally lower compared to the Django dataset:

SMOTE XGBoost achieved the highest G-mean (96.84) and recall (94.00), highlighting its effectiveness in detecting minority class instances. The lower rankings for accuracy and

F-score on the CMU dataset can be attributed to its imbalanced nature. In an imbalanced dataset, models tend to predict the majority class (impostor users) more frequently, which can result in high accuracy but poor performance in identifying the minority class (genuine users). Metrics such as G-mean and recall are more indicative of the model's ability to identify minority class instances, which is why SMOTE-enhanced methods still perform well on these metrics despite the overall imbalance.

A detailed analysis reveals that the combination of SMOTE with Random Forest (SMOTE RF) yields particularly significant improvements, while the benefit for XGBoost and Bagging is somewhat smaller:

Random Forest benefits more from SMOTE due to its use of multiple Decision Trees as base classifiers. These trees, being lightweight and numerous, can effectively utilize the increased decision space provided by SMOTE, thereby better capturing the characteristics of minority class instances. In contrast, XGBoost and Bagging often use fewer base classifiers, which limits the benefits derived from SMOTE. This indicates that increasing the ensemble size can amplify the advantages of SMOTE, especially in complex classification scenarios.

The stacked bar charts in Figures 3 and 4 provide a visual representation of the rankings for different methods across the four key metrics (accuracy, F-score, G-mean, and recall) for both datasets:

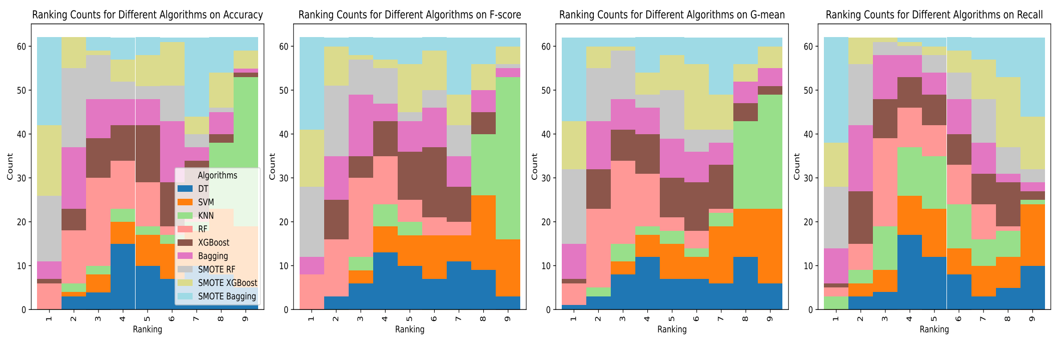

Figure 3. Ranking Counts for Different Algorithms on Django Dataset.

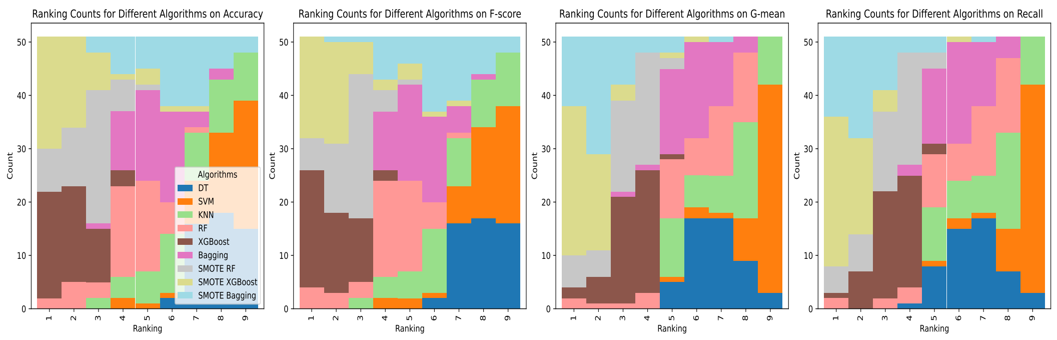

Figure 4. Ranking Counts for Different Algorithms on CMU Dataset.

In the Django dataset (Figure 3), SMOTE-enhanced ensemble methods consistently rank higher across all metrics, confirming their superior performance in a balanced dataset.

In the CMU dataset (Figure 4), the charts illustrate the challenges posed by class imbalance. While SMOTE-enhanced methods like OverXGBoost and OverRF rank highly for G-mean and recall, traditional methods such as Decision Trees and SVM tend to rank higher in accuracy and F-score. This discrepancy highlights the importance of using balanced evaluation metrics to assess true model effectiveness in imbalanced scenarios. These visualizations reinforce the numerical results and demonstrate that SMOTE-enhanced ensemble learning methods are particularly effective at addressing class imbalance, improving the robustness of keystroke dynamics authentication systems.

To further validate the effectiveness of the proposed method, we conducted pairwise comparisons of different methods using the Nemenyi statistical test. G-mean was selected as the representative metric, and statistical analyses were performed on both the CMU and Django datasets. The results are illustrated in Figures 5 and 6, where connecting lines indicate similar performance among different methods.

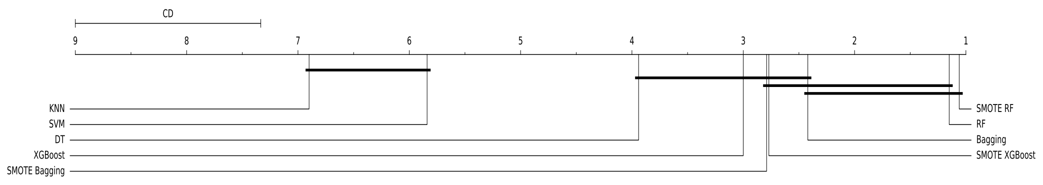

Figure 5. G-mean Nemenyi statistical significance test on Django dataset.

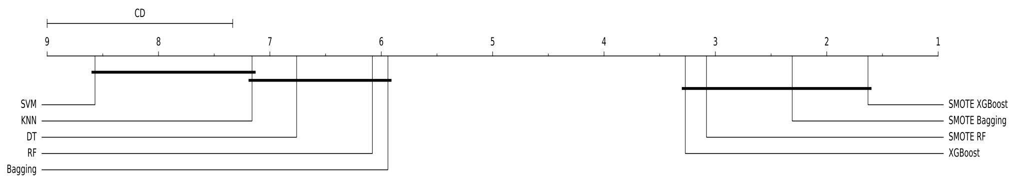

Figure 6. G-mean Nemenyi statistical significance test on CMU dataset.

The results show that the combination of SMOTE with ensemble learning methods yields significant improvements compared to other methods, particularly in metrics that are crucial for identifying minority classes. The Nemenyi test confirms that SMOTE-enhanced methods are statistically superior in terms of G-mean, emphasizing their robustness in handling imbalanced datasets.

4.2.2. Parameter Sensitivity Analysis

To further understand the impact of key parameters on the performance of our keystroke dynamics authentication system, we conducted a sensitivity analysis focusing on three critical parameters: the ensemble size in ensemble learning methods, the *k*-value in SMOTE, and the imbalance ratio in SMOTE. The results of these analyses are presented in Figures 7–10.

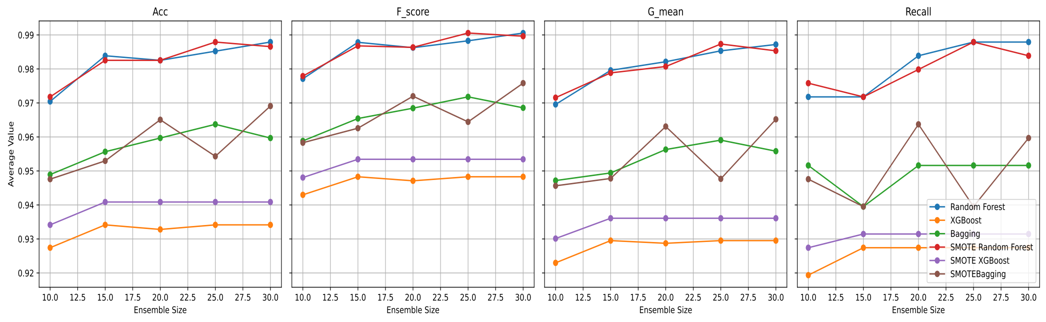

Figure 7. Impact of Ensemble Size on Performance Metrics (Django Dataset).

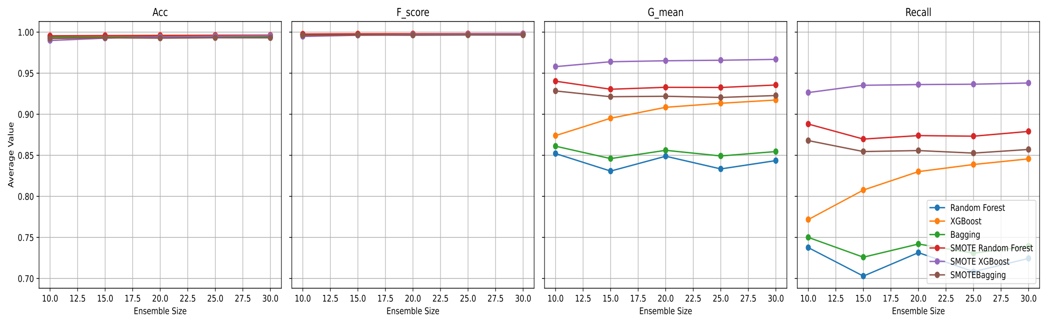

Figure 8. Impact of Ensemble Size on Performance Metrics (CMU Dataset).

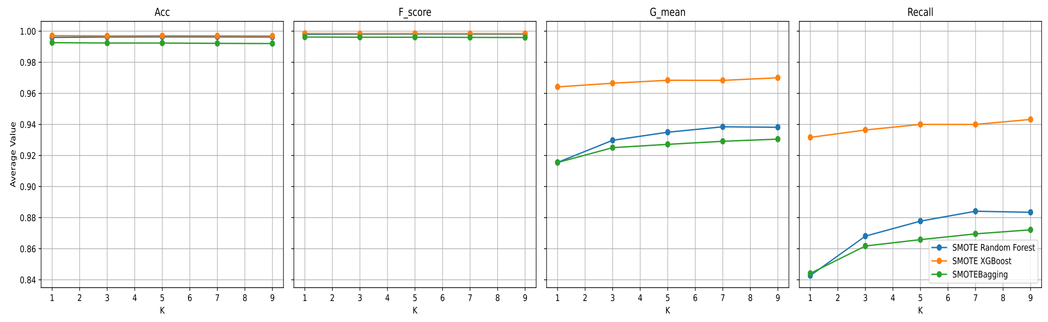

Figure 9. Effect of SMOTE k-Value on Performance Metrics (CMU Dataset).

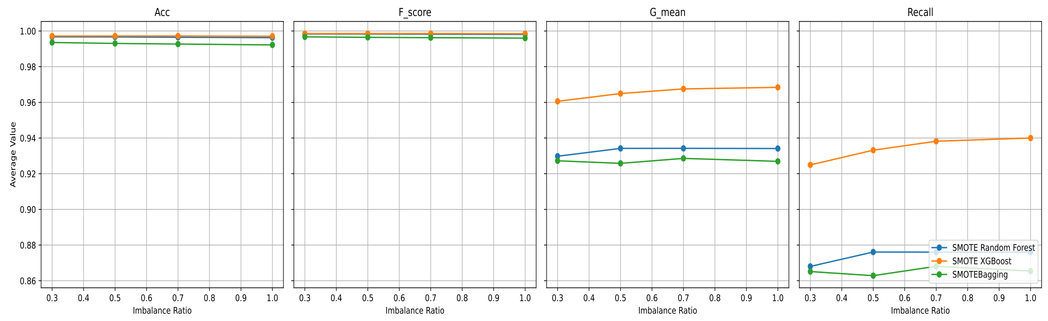

Figure 10. Effect of Imbalance Ratio on Performance Metrics (CMU Dataset).

Figures 7 and 8 illustrate the relationship between the ensemble size and the four evaluation metrics—accuracy, F-score, G-mean, and recall—for both the Django and CMU datasets.

The analysis in Figure 7 shows that increasing the ensemble size generally leads to improved performance across all metrics. This effect is particularly evident in the balanced Django dataset, where larger ensemble sizes consistently result in higher accuracy, F-score, G-mean, and recall. However, the improvement rate diminishes at larger ensemble sizes, suggesting a diminishing return effect. Specifically, beyond an ensemble size of approximately 25 to 30, the performance gain becomes marginal, and in some metrics (e.g., recall), a slight decrease is observed. This indicates that adding more models does not always guarantee significant improvements and may introduce redundancy.

In Figure 8, a similar trend is observed for the CMU dataset. Increasing the ensemble size initially improves accuracy, F-score, and recall, but the impact on G-mean is more variable due to the imbalanced nature of the dataset. The results indicate that while larger ensembles can mitigate some effects of class imbalance, over-increasing the ensemble size without adequate diversity among the base models may not significantly enhance model performance. This highlights the importance of balancing ensemble size with model diversity for optimal performance, particularly in imbalanced datasets.

The sensitivity analysis of the SMOTE *k*-value was conducted on the imbalanced CMU dataset, as shown in Figure 9. The *k*-value in SMOTE determines the number of nearest neighbors considered when generating synthetic samples.

The results reveal that *k*-values between 5 and 7 tend to produce the best results across all evaluation metrics. This range achieves a balance between generating a sufficient number of synthetic samples and maintaining diversity within the minority class. A low *k*-value (e.g., $k = 1$ or 2) results in synthetic samples that are too similar to the original minority class instances, potentially leading to overfitting. On the other hand, a high k-value (e.g., $k > 7$) may generate synthetic samples that overlap significantly with the majority class, thereby introducing noise and reducing the overall effectiveness of the synthetic samples. The results suggest that careful tuning of the *k*-value is critical to ensure the generation of high-quality synthetic samples that enhance model performance without compromising generalizability.

Figure 10 presents the relationship between the imbalance ratio in SMOTE and the four evaluation metrics for the CMU dataset. The imbalance ratio represents the degree of balance between the minority and majority classes after applying SMOTE, with a ratio of 1 indicating a perfectly balanced dataset.

The results demonstrate that performance stability improves as the imbalance ratio approaches 1, indicating that more balanced datasets tend to produce more consistent results. This is particularly true for metrics like G-mean and recall, which are critical for assessing the detection of minority class instances (impostors). The closer the imbalance ratio is to 1, the more effective the model is at generalizing across both classes. However, it is also important to avoid over-balancing, as an artificially equal class distribution might not reflect real-world conditions, potentially leading to reduced generalization in practice.

The parameter sensitivity analysis underscores the importance of carefully tuning parameters in ensemble learning and data resampling techniques to optimize the performance of keystroke dynamics authentication systems. Larger ensemble sizes generally improve classification performance, though there is a point of diminishing returns. The SMOTE *k*-value analysis reveals that selecting an appropriate *k* (typically between 5 and 7) is crucial for generating effective synthetic samples. Finally, the imbalance ratio analysis highlights the benefits of striving for a balanced dataset, as closer-to-equal class distributions lead to more stable and reliable model performance. By understanding and optimizing these parameters, we can further enhance the robustness and effectiveness of keystroke dynamics authentication, particularly in handling imbalanced datasets like the CMU keystroke dynamics benchmark.

These findings underscore the importance of parameter tuning in ensemble learning and data resampling techniques to optimize the performance of keystroke dynamics authentication systems. By carefully adjusting ensemble size, SMOTE k-value, and imbalance ratio, we can achieve a balance between model complexity and generalizability, enhancing the robustness and effectiveness of our approach, particularly for handling imbalanced datasets like the CMU keystroke dynamics benchmark.

Grid search (GridSearchCV) is a commonly used hyperparameter optimization method in machine learning that seeks the best parameter combination by traversing a given parameter grid. In our recent experiments, we applied grid search to optimize SMOTE-enhanced ensemble learning methods, including random forests, XGBoost, and Bagging, each combined with a 0.3 sampling strategy. The optimal parameter combinations for each method are as follows:

For SMOTE Random Forest, we found the best parameter set, which includes a maximum depth of 20 (`max_depth = 20`), a minimum sample split of 2 (`min_samples_split = 2`), and the number of estimators set to 50 (`n_estimators = 50`). These parameters help control the complexity of the trees while maintaining the diversity of the model, thereby enhancing the classification performance on imbalanced datasets.

For SMOTE XGBoost, we determined the optimal parameters to be a learning rate of 0.1 (`learning_rate = 0.1`), a maximum depth of 5 (`max_depth = 5`), and the number of estimators set to 150 (`n_estimators = 150`). These parameter settings help balance the learning speed and complexity of the model, leading to better performance on imbalanced datasets.

For SMOTE Bagging, we discovered the best parameter combination to be a maximum feature proportion of 0.5 (`max_features = 0.5`), a maximum sample proportion of 1.0 (`max_samples = 1.0`), and the number of estimators set to 20 (`n_estimators = 20`). These parameters help maintain model stability while increasing the model's generalization ability.

By using two-fold cross-validation (`cv = 2`), we evaluated the performance of these parameter combinations and selected the one with the highest average score. Although this method is computationally expensive, it ensures that we find the most optimized parameter settings, thereby improving the model's accuracy and robustness when dealing with imbalanced datasets. In this way, we can more precisely adjust the model to adapt to specific data characteristics and business needs.

4.2.3. Evaluation of Classifier Performance Using ROC Curves

Through a meticulous analysis of our model's performance, the ROC curve and AUC value provide compelling evidence that our method excels in classification tasks. The ROC curve clearly demonstrates the trade-off between the true positive rate (TPR) and the false positive rate (FPR) at different threshold settings. Furthermore, the AUC value, which is the area under the ROC curve, provides us with a single metric to assess the overall performance of the model; our model achieves a high TPR while maintaining a low FPR, indicating its excellent identification capability.

Based on the CMU dataset (Figure 11), Random Forest (AUC = 0.99) and XGBoost (AUC = 1.00) consistently outperform other classifiers, showcasing exceptional performance across various thresholds. In the SMOTE-enhanced strategy 1, both Random Forest and XGBoost achieve an AUC of 1.00, while Bagging attains an AUC of 0.97. These findings suggest that SMOTE, combined with ensemble learning techniques, significantly improves classification performance on imbalanced datasets. The results indicate that these approaches not only enhance minority class detection but also ensure reliable and accurate keystroke dynamics authentication across different datasets.

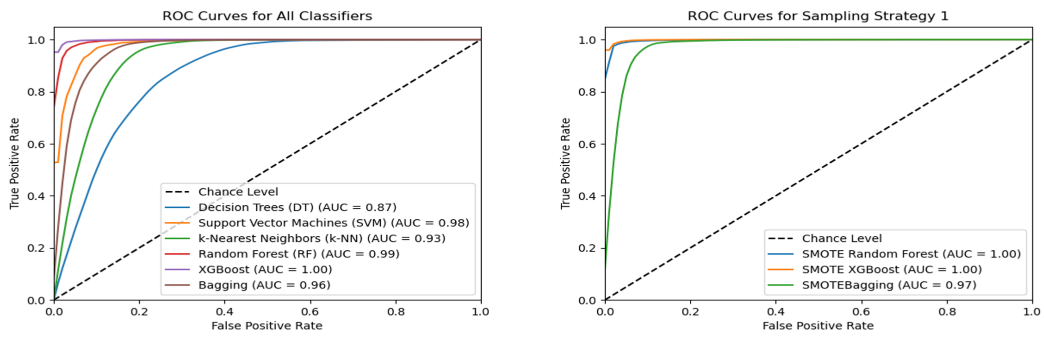

Figure 11. ROC Curves for different Classifiers on the CMU Dataset.

For the Django dataset (Figure 12), similar trends are observed, the ensemble learning methods, particularly Random Forest (AUC = 1.00), XGBoost (AUC = 0.95), and Bagging (AUC = 0.98), demonstrate superior classification performance compared to traditional classifiers such as Decision Trees (AUC = 0.89), SVM (AUC = 0.90), and k-NN (AUC = 0.92). The high AUC values for these ensemble methods indicate a strong ability to balance true positive and false positive rates, highlighting their robustness in distinguishing between genuine users and impostors. Additionally, SMOTE-enhanced Random Forest, XGBoost, and Bagging classifiers maintain high AUC values, further confirming the effectiveness of SMOTE in handling class imbalance in the Django dataset. These results not only validate the theoretical advantages of our method but also show great potential in practical applications.

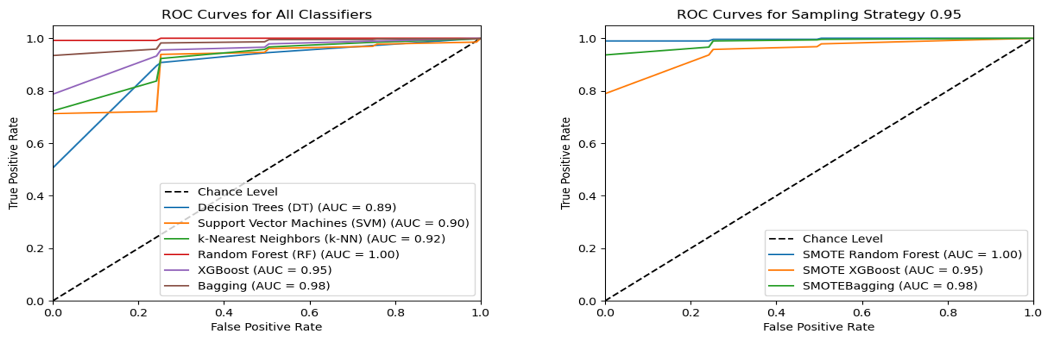

Figure 12. ROC Curves for different Classifiers on the Django Dataset.

4.2.4. Comparison of SMOTE-Enhanced Methods with Other Resampling Techniques

The proposed SMOTE-enhanced ensemble learning methods demonstrated superior performance compared to other commonly used resampling techniques, such as Random Oversampling, Random Undersampling, and ADASYN. Random Oversampling [29], while effective in balancing class distributions, often leads to overfitting by duplicating minority class samples, which limits the model's generalization capability. In contrast, SMOTE generates synthetic samples by interpolating between existing minority class instances, preserving data diversity and reducing the risk of overfitting.

Random Undersampling [30], on the other hand, achieves class balance by removing majority class samples, which can result in the loss of critical information and negatively impact model performance. The results showed that SMOTE outperformed Random Undersampling by maintaining a more comprehensive representation of the dataset, leading to higher classification accuracy and recall for minority classes.

While ADASYN focuses on generating synthetic samples for harder-to-classify minority instances [31], it can sometimes overemphasize these challenging areas, potentially introducing noise into the dataset. The SMOTE method used in this study provided a balanced approach, ensuring both class representation and data consistency. These advantages highlight the effectiveness of SMOTE in enhancing model performance for keystroke dynamics authentication, particularly in complex and imbalanced scenarios.

5. Conclusions

In this study, we conducted experiments combining SMOTE with ensemble learning methods to enhance the performance of keystroke dynamics authentication systems. By developing a data collection platform based on Django, we generated a balanced keystroke dynamics dataset and evaluated the effectiveness of our approach using the CMU benchmark dataset. The experiments included comparisons between traditional classifiers (such as Decision Trees, SVM, and k-NN), ensemble learning methods (such as Random Forest, XGBoost, and Bagging), and SMOTE-enhanced ensemble learning methods.

The demonstration shows that SMOTE-enhanced ensemble learning methods significantly outperformed traditional classifiers across multiple metrics, including accuracy, recall, G-mean, and F1-score. In particular, SMOTE-enhanced methods excelled in detecting minority classes, such as genuine users, addressing the class imbalance challenge effectively. On the imbalanced CMU dataset, these methods achieved high G-mean and recall values, ensuring robust performance in complex and imbalanced scenarios. ROC analysis further validated the strength of SMOTE-enhanced ensemble methods, confirming their ability to enhance the model's discriminative power.

Our findings indicate that combining SMOTE with ensemble learning can significantly improve the reliability and security of keystroke dynamics authentication systems in practical applications, particularly in defending against impersonation attacks on online platforms and enterprise environments. These methods are well-suited for use in applications that require real-time authentication and behavioral biometrics, such as online security systems.

In future research, we are committed to extracting information such as individual health status, environmental factors, and keyboard layout changes, and utilizing them as features for behavioral biometric recognition systems. Our goal is to develop an intelligent system that can respond in real time to changes in user health, adapt to different environmental conditions, and accommodate the physical characteristics of various keyboards. By integrating these features, our system will be able to more accurately capture user keystroke behaviors and maintain recognition accuracy and robustness amidst dynamic changes. We plan to use advanced data fusion techniques to combine these features with traditional keystroke dynamics features, thereby enhancing the overall performance of the recognition system. This multimodal feature processing approach will make our system more flexible and reliable across a variety of application scenarios, paving new avenues for the development of behavioral biometric technology.

Author Contributions: Conceptualization, X.W. and D.H.; methodology, X.W.; software, X.W.; validation, X.W. and D.H.; formal analysis, X.W.; investigation, X.W.; resources, X.W.; data curation, X.W.; writing—original draft preparation, X.W.; writing—review and editing, D.H.; visualization, X.W.; supervision, D.H.; project administration, D.H.; funding acquisition, D.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by NSF Award CNS-1650503. Wang and Hou were also supported by NSF Award TI-2122746.

Data Availability Statement: Data are unavailable due to privacy or ethical restrictions

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Ayeswarya, S.; Singh, K.J. A Comprehensive Review on Secure Biometric-Based Continuous Authentication and User Profiling. *IEEE Access* **2024**, *12*, 82996–83021. [CrossRef]
2. Giot, R.; El-Abed, M.; Rosenberger, C. Keystroke dynamics authentication. *Biometrics* **2011**, *1*, 157–182.
3. Intan, I. Combining of feature extraction for real-time facial authentication system. In Proceedings of the 2017 5th International Conference on Cyber and IT Service Management (CITSM), Denpasar, Indonesia, 8–10 August 2017; pp. 1–6. [CrossRef]
4. Alshanketi, F.; Traore, I.; Ahmed, A.A. Improving Performance and Usability in Mobile Keystroke Dynamic Biometric Authentication. In Proceedings of the 2016 IEEE Security and Privacy Workshops (SPW), San Jose, CA, USA, 22–26 May 2016; pp. 66–73. [CrossRef]
5. Wankhede, S.B.; Verma, S. Keystroke dynamics authentication system using neural network. *Int. J. Innov. Res. Dev.* **2014**, *3*, 157–164.
6. Andrean, A.; Jayabalan, M.; Thiruchelvam, V. Keystroke dynamics based user authentication using deep multilayer perceptron. *Int. J. Mach. Learn. Comput.* **2020**, *10*, 134–139. [CrossRef]
7. Cilia, D.; Inguanez, F. Multi-model authentication using keystroke dynamics for smartphones. In Proceedings of the 2018 IEEE 8th International Conference on Consumer Electronics-Berlin (ICCE-Berlin), Berlin, Germany, 2–5 September 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–6.
8. Kiyani, A.T.; Lasebae, A.; Ali, K.; Rehman, M.U.; Haq, B. Continuous user authentication featuring keystroke dynamics based on robust recurrent confidence model and ensemble learning approach. *IEEE Access* **2020**, *8*, 156177–156189. [CrossRef]
9. Quimatio, B.M.A.; Njike, O.F.Y.; Nkenlifack, M. User Authentification through Keystroke dynamics based on ensemble learning approach. In Proceedings of the CARI 2022–Colloque Africain sur la Recherche en Informatique et en Mathémathiques Appliquées, Sophia Antipolis, France, 4–7 October 2022.
10. Zhang, W.; Zhao, W.; Li, J.; Zhuang, P.; Sun, H.; Xu, Y.; Li, C. CVANet: Cascaded visual attention network for single image super-resolution. *Neural Netw.* **2024**, *170*, 622–634. [CrossRef] [PubMed]
11. Zhang, W.; Li, Z.; Li, G.; Zhuang, P.; Hou, G.; Zhang, Q.; Li, C. GACNet: Generate Adversarial-Driven Cross-Aware Network for Hyperspectral Wheat Variety Identification. *IEEE Trans. Geosci. Remote Sens.* **2024**, *62*, 1–14. [CrossRef]
12. Wongvorachan, T.; He, S.; Bulut, O. A Comparison of Undersampling, Oversampling, and SMOTE Methods for Dealing with Imbalanced Classification in Educational Data Mining. *Information* **2023**, *14*, 54. [CrossRef]
13. Filimonyuk, L. An Approach to Decision-Making Systems' Development in Case of Accidents' Causes Figuring out in Large-Scale Systems. In Proceedings of the 2022 15th International Conference Management of Large-Scale System Development (MLSD), Moscow, Russia, 26–28 September 2022; pp. 1–4. [CrossRef]
14. Galar, M.; Fernandez, A.; Barrenechea, E.; Bustince, H.; Herrera, F. A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches. *IEEE Trans. Syst. Man. Cybern. Part C Appl. Rev.* **2012**, *42*, 463–484. [CrossRef]
15. Zhu, T.; Lin, Y.; Liu, Y. Improving interpolation-based oversampling for imbalanced data learning. *Knowl.-Based Syst.* **2020**, *187*, 104826. [CrossRef]
16. Kaur, P.; Sharma, A.; Chahal, J.K.; Sharma, T.; Sharma, V.K. Analysis on Credit Card Fraud Detection and Prevention using Data Mining and Machine Learning Techniques. In Proceedings of the 2021 International Conference on Computational Intelligence and Computing Applications (ICCICA), Nagpur, India, 26–27 November 2021; pp. 1–4. [CrossRef]
17. Roy, A.; Cruz, R.M.; Sabourin, R.; Cavalcanti, G.D. A study on combining dynamic selection and data preprocessing for imbalance learning. *Neurocomputing* **2018**, *286*, 179–192. [CrossRef]
18. Sierra, B.; Lazkano, E.; Irigoien, I.; Jauregi, E.; Mendialdua, I. K Nearest Neighbor Equality: Giving equal chance to all existing classes. *Inf. Sci.* **2011**, *181*, 5158–5168. [CrossRef]
19. Liu, Z.; Cao, W.; Gao, Z.; Bian, J.; Liu, T.Y. Self-paced Ensemble for Highly Imbalanced Massive Data Classification. In Proceedings of the 36th IEEE International Conference on Data Engineering, Dallas, TX, USA, 20–24 April 2020.
20. Dodda, R.; Raghavendra, C.; Aashritha, M.; Macherla, H.V.; Kuntla, A.R. A Comparative Study of Machine Learning Algorithms for Predicting Customer Churn: Analyzing Sequential, Random Forest, and Decision Tree Classifier Models. In Proceedings of the 2024 5th International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 7–9 August 2024; pp. 1552–1559. [CrossRef]

21. Bernardo, A.; Gomes, H.M.; Montiel, J.; Pfahringer, B.; Valle, E.D. C-SMOTE: Continuous Synthetic Minority Oversampling for Evolving Data Streams. In Proceedings of the 2020 IEEE International Conference on Big Data (Big Data), Atlanta, GA, USA, 10–13 December 2020.
22. Ducray, B.; Cobourne, S.; Mayes, K.; Markantonakis, K. Comparison of dynamic biometrie security characteristics against other biometrics. In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017; pp. 1–7. [CrossRef]
23. Cieslak, D.A.; Chawla, N.V. *Learning Decision Trees for Unbalanced Data*; Springer: Berlin/Heidelberg, Germany, 2008.
24. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
25. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
26. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; ACM: New York, NY, USA, 2016; pp. 785–794.
27. Breiman, L. Bagging Predictors. *Mach. Learn.* **1996**, *24*, 123–140. [CrossRef]
28. Killourhy, K.S.; Maxion, R.A. Comparing anomaly-detection algorithms for keystroke dynamics. In Proceedings of the 2009 IEEE/IFIP International Conference on Dependable Systems & Networks, Lisbon, Portugal, 29 June–2 July 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 125–134.
29. Bours, P. Continuous keystroke dynamics: A different perspective towards biometric evaluation. *Inf. Secur. Tech. Rep.* **2012**, *17*, 36–43. [CrossRef]
30. Monaco, J.V.; Tappert, C.C. The partially observable hidden Markov model and its application to keystroke dynamics. *Pattern Recognit.* **2018**, *76*, 449–462. [CrossRef]
31. Zhong, Y.; Deng, Y.; Jain, A.K. Keystroke dynamics for user authentication: A large-scale investigation. *IEEE Trans. Biom. Behav. Identity Sci.* **2019**, *1*, 123–135.