Application of a Nine-Variate Prediction Ellipsoid for Normalized Data and Machine Learning Algorithms for Keystroke Dynamics Recognition

Sergiy Prykhodko[1,2,*,†] and Artem Trukhov[1,†]

[1] *Admiral Makarov National University of Shipbuilding, Heroes of Ukraine Ave., 9, Mykolaiv, 54007, Ukraine*
[2] *Odesa Polytechnic National University, Shevchenko Ave., 1, Odesa, 65044, Ukraine*

Abstract

Keystroke dynamics recognition is a crucial element in enhancing security, enabling personalized user authentication, and supporting various identity verification systems. This study offers a comparative analysis of a nine-variate prediction ellipsoid for normalized data and machine learning algorithms — specifically, autoencoder, isolation forest, and one-class support vector machine — for keystroke dynamics recognition. Traditional methods often assume a multivariate normal distribution. However, real-world keystroke data typically deviate from this assumption, negatively impacting model performance. To address this, the dataset was normalized using the multivariate Box-Cox transformation, allowing the construction of a decision rule based on a nine-variate prediction ellipsoid for normalized data. The study also includes outlier removal during preprocessing, further improving the model's performance.
The results revealed that the application of the Box-Cox transformation significantly enhanced both the accuracy and robustness of the prediction ellipsoid. Although all models demonstrated strong performance, the nine-variate prediction ellipsoid for normalized data consistently outperformed the machine learning algorithms. The study highlights the importance of careful feature selection and multivariate normalizing transformations in keystroke dynamics recognition. Future studies could benefit from broader datasets that include a wider range of user behaviors, such as variations in environmental factors and longer key sequences.

Keywords

keystroke dynamics, multivariate normal distribution, Box-Cox transformation, machine learning

1. Introduction

In recent years, keystroke dynamics recognition has become an effective method for biometric authentication. By analyzing the unique patterns and rhythms individuals demonstrate while typing, such as keystroke duration and the intervals between key presses [1], it becomes possible to create a distinctive typing profile for each user. Unlike traditional biometric methods like fingerprint or facial recognition, keystroke dynamics offers a non-intrusive and continuous form of authentication [2]. This makes it especially appealing for secure applications such as online banking, login systems, and access control.

The keystroke recognition process involves several essential stages to ensure accurate user authentication. It begins with the collection of a dataset, typically consisting of timestamps for keypress and key release events. From this raw data, key attributes such as hold times and inter-key intervals are extracted, which reveal the unique typing behavior of the user. A critical preprocessing step is the detection and removal of outliers - data points that significantly deviate from the expected behavior and could otherwise distort the results [3]. This step is vital for creating a cleaner dataset

Information Technology and Implementation (IT&I-2024), November 20-21, 2024, Kyiv, Ukraine
* Corresponding author.
† These authors contributed equally.
sergiy.prykhodko@nuos.edu.ua (S. Prykhodko); artem.trukhov@gmail.com (A. Trukhov)
0000-0002-2325-018X (S. Prykhodko); 0000-0002-7160-8609 (A. Trukhov)

and improving model accuracy. Once preprocessing is complete, classification models are applied to recognize new data inputs.

In traditional recognition tasks, classification typically involves assigning an object to one of several predefined categories. However, in the context of keystroke dynamics, one-class classification is more frequently employed. Unlike standard classification methods, which rely on a balanced dataset with both positive and negative examples, one-class classification focuses on building a model based solely on an individual's unique typing patterns, representing the target class, without the need for negative samples. This approach is particularly beneficial in authentication systems, where the goal is to continuously verify that the current user matches the known profile, rather than distinguishing between multiple users [4]. Closely related to outlier detection, one-class classification evaluates new data to determine if it aligns with the target profile, flagging any deviations as potential anomalies [5].

Prediction ellipsoids [6] and machine learning algorithms [7] are commonly utilized in the field of pattern recognition. The study aims to compare these models in keystroke dynamics recognition, assessing their performance, robustness, and applicability in real-world settings.

2. Literature review

Mathematical modeling techniques are pivotal in the field of keystroke dynamics recognition, aimed at improving accuracy and reliability. Recent advancements have integrated a range of approaches. Tree-based models, like random forests [8], classify data by constructing hierarchical structures, and learning feature splits that effectively differentiate between classes. Support vector-based methods [9] focus on maximizing the margin between classes to create optimal decision boundaries, while neural network models [10] capture complex patterns in keystroke data by processing information through multiple interconnected layers of nodes.

However, for user authentication systems, one-class classification is more commonly employed [11]. Among the leading techniques are prediction ellipsoids and machine learning algorithms such as one-class support vector machine (OCSVM) [12, 13], isolation forest (IF) [14], and autoencoder (AE) [15, 16]. OCSVM learns a decision boundary that separates target data points from outliers while maximizing the margin within the feature space. IF is an ensemble method that isolates anomalies by randomly selecting features and partitioning the data until anomalous points are isolated in smaller partitions, requiring fewer splits for target points. AE, as a neural network, learns an efficient representation of data by encoding inputs into a lower-dimensional space and reconstructing them. Anomalies are flagged by evaluating reconstruction errors, where higher discrepancies suggest potential outliers.

The use of prediction ellipsoids relies on the assumption that data conforms to a multivariate normal distribution [17]. In practice, however, this assumption often does not hold for real-world keystroke data [18]. To address this, normalization transformations are applied, adjusting the data to more closely align with a multivariate normal distribution and thereby improving the model's accuracy and robustness [19-20]. Techniques like univariate transformations (e.g., logarithmic or Box-Cox transformation) operate on individual features, while multivariate transformations, such as the multivariate Box-Cox transformation, consider relationships between features for a more holistic normalization approach.

This study focuses on comparing prediction ellipsoid for normalized data and machine learning algorithms such as OCSVM, IF, and AE, which are widely used and offer distinct approaches to one-class classification. In the context of keystroke dynamics recognition, accuracy, and efficiency are critical, making it essential to evaluate the effectiveness of different approaches. While prediction ellipsoid offers interpretability and computational efficiency, it can encounter limitations when dealing with non-Gaussian data distributions. On the other hand, machine learning algorithms such as OCSVM, IF, and AE provide alternative techniques, each with its own advantages and challenges when applied to keystroke dynamics.

3. Materials and methods

3.1. Keystroke dynamics dataset

In keystroke dynamics recognition, the quality and structure of the dataset play a crucial role in determining the performance and accuracy of the applied algorithms. A typical keystroke dynamics dataset records various temporal characteristics of an individual's typing behavior, including metrics such as the duration of key presses and the intervals between consecutive key events.

This study utilizes the CMU keystroke dynamics dataset, which captures detailed typing data from 51 subjects. Each subject was tasked with typing a static password string: ".tie5Roanl". The dataset records various keystroke timing features in seconds, including how long each key is pressed and the intervals between key presses. Data collection was conducted over eight distinct sessions per subject, with at least one day between sessions. Each session required subjects to type the password 50 times, resulting in 400 samples per individual and a total of 20,400 samples across all participants.

The dataset is organized by subject identifier, session number, repetition count, and 31 timing features. Columns are labeled to reflect specific keystroke metrics: H.key denotes the hold time for a particular key, measuring the duration from key press to release. DD.key1.key2 represents the keydown-keydown interval, i.e., the time between pressing two consecutive keys, while UD.key1.key2 indicates the keyup-keydown interval, measuring the time between releasing one key and pressing the next. Notably, UD times can be negative in some cases, and the sum of H times and UD times corresponds to the DD time for a given digraph.

To simplify the modeling process, this study focuses on 9 key properties, hold time for a particular key, forming the feature vector: X = { H.t, H.i, H.e, H.5, H.R, H.o, H.a, H.n, H.l }.

3.2. Outlier removal

After extracting feature vectors, the subsequent step involves detecting and removing outliers. This process is crucial because outliers can distort the analysis and undermine the performance of recognition models. By eliminating these anomalies, the dataset is refined, ensuring that the data better reflects typical user behavior, which in turn enhances model training.

One commonly used method for outlier detection is based on the squared Mahalanobis distance (SMD). However, SMD assumes that the data follows a multivariate normal distribution, which might not always be the case. To verify this assumption, it is necessary to assess the data's normality through statistical tests like the Mardia test, which is used in the study [21]. This test evaluates two aspects of multivariate normality: skewness β_1 and kurtosis β_2.

The Mardia test calculates skewness scaled by $N/6$, which follows a chi-square distribution with $p(p+1)(p+2)/6$ degrees of freedom, where p is the number of variables and N is the sample size. Kurtosis is compared to the normal distribution, with a mean of $p(p+2)$ and a variance of $8p(p+2)/N$. By comparing the calculated skewness and kurtosis values with those expected under a normal distribution, the test helps identify significant deviations from multivariate normality. If the data deviates significantly, normalization is required to transform a non-Gaussian vector $X = X_1, X_2, \dots, X_9{}^T$ into a Gaussian vector $Z = Z_1, Z_2, \dots, Z_9{}^T$.

Normalization transformations are essential in data analysis and machine learning, as they help stabilize variance, reduce skewness, and better align data with a multivariate Gaussian distribution. Univariate transformations, such as logarithmic transformations and univariate BCT, are typically applied to individual features. The logarithmic transformation is effective for stabilizing variance in positively skewed data, while the univariate BCT can handle both positive and negative skewness by optimizing a parameter (λ). However, univariate methods may fall short when features are interdependent, and the BCT can be sensitive to outliers due to the complexity of parameter estimation.

In contrast, multivariate transformations like the multivariate BCT consider the relationships between multiple features. The multivariate Box-Cox transformation builds upon the principles of the univariate Box-Cox transformation but applies it across multiple variables at once:

$$Z_j = x(\lambda_j) = \begin{cases} (X_j^{\lambda_j} - 1)/\lambda_j, & \lambda_j \neq 0; \\ \ln(X_j), & \lambda_j = 0. \end{cases} \tag{1}$$

While it is more computationally demanding, this transformation preserves correlations between variables, offering a more robust approach to normalizing complex datasets. The multivariate BCT improves the alignment of data with a multivariate normal distribution by optimizing parameters through methods such as maximizing the log-likelihood of transformed data, as discussed in the study [21]. Once the multivariate BCT is applied, the Mardia test should be repeated to verify the success of the normalization.

After normalization, outlier removal is performed iteratively using the SMD method, removing one data point per iteration based on the largest distance. This ensures that the most extreme values are eliminated first, leading to a cleaner and more representative dataset for subsequent analysis.

3.3. Prediction ellipsoid

A prediction ellipsoid is a multivariate tool used to assess whether a data point belongs to a specific target class. It operates by calculating the SMD for each point, which forms the left side of the comparison equation. This distance is then measured against a critical value derived from the chi-square distribution, which serves as the right side of the equation [22]:

$$(X - \bar{X})^T S_X^{-1} (X - \bar{X}) = \chi^2_{9,\ 0.005}. \tag{2}$$

The SMD follows a chi-square distribution with degrees of freedom corresponding to the number of features in the data, which in this case is 9. This allows for the calculation of a critical value based on the desired significance level, commonly set at 0.005 for one-class classification tasks. If a data point's SMD exceeds this critical value, it is classified as an anomaly, meaning it is likely part of a different class. If the SMD falls below the threshold, the point is considered an instance of the target class.

In cases where the data does not follow a normal distribution, a normalization process is implemented before constructing the nine-variate prediction ellipsoid, which is represented by the equation:

$$(Z - \bar{Z})^T S_Z^{-1} (Z - \bar{Z}) = \chi^2_{9,\ 0.005}. \tag{3}$$

For 9 degrees of freedom at a significance level of 0.005, the chi-square distribution provides a critical value of 23.59. Any data point with an SMD below this value is deemed to lie within the ellipsoid, signifying its membership in the target class.

3.4. Machine learning algorithms

3.4.1. One-class support vector machine

OCSVM constructs a decision boundary that separates target data from the rest of the feature space by finding a hyperplane with the maximum margin. This boundary is optimized by maximizing the distance between the hyperplane and the origin within a high-dimensional feature space. The OCSVM employs an implicit transformation function, denoted as $\varphi(\cdot)$, which is a non-linear projection evaluated through a kernel function. This kernel function maps the original feature space into a potentially higher-dimensional one: $k(x, y) = \varphi(x) \cdot \varphi(y)$ [23].

Several kernel functions are commonly used in OCSVM. The linear kernel computes dot products in the original feature space, making it ideal for linearly separable data. The polynomial kernel captures non-linear relationships by raising dot products to specific powers, allowing it to model more complex decision boundaries. The radial basis function kernel, using a Gaussian function, effectively captures intricate relationships, particularly in cases where data is not linearly separable. The sigmoid kernel, based on the hyperbolic tangent function, excels at capturing non-linear patterns, making it useful for handling complex relationships between features and classes.

The decision boundary that OCSVM learns is defined by the following equation:

$$g(x) = \omega^T \varphi(x) - \rho,$$

where ω represents the normal vector of the hyperplane, and ρ is the bias term.

OCSVM is formulated as a quadratic optimization problem, aiming to minimize the weight vector ω while maximizing the margin, subject to specific constraints. The optimization problem can be expressed as:

$$min_{\omega,\xi,\rho} \frac{||\omega||^2}{2} - \rho + \frac{1}{\nu N}\sum_{i=1}^{N} \xi_i ,$$

$$\text{subject to: } \omega^T \varphi(x_i) \geq \rho - \xi_i, \xi_i \geq 0,$$

where ξ_i are slack variables that account for separation errors, and $\nu \in (0,1]$ is the regularization parameter, which controls the balance between the number of outliers and the number of support vectors.

The optimization problem is typically solved in its dual form, producing a decision function that classifies new data points as either belonging to the target class or as anomalies. The final decision function is:

$$f(x) = sgn\big(g(x)\big).$$

The function returns a positive value for data points belonging to the target class and a negative value for anomalies.

3.4.2. Isolation forest

Unlike traditional methods that rely on modeling target points, IF takes a distinctive approach by focusing directly on isolating anomalies. This technique works by constructing isolation trees, where internal nodes represent features and their split values, and leaf nodes represent individual data points. The construction of isolation trees begins by randomly selecting a feature and a corresponding split value within its range. This random process continues until each data point is isolated in its own leaf node or until a specified maximum tree depth is reached [24]. The strength of this method lies in its ability to isolate anomalies without needing prior knowledge of the dataset's distribution. Anomalies, being easier to isolate since they reside in sparser regions of the feature space, require fewer splits from root to leaf nodes compared to normal data points. As a result, the average path length from the root to the leaf node for each data point is calculated across all trees in the forest.

The anomaly score for each data point is derived based on its average path length using the following formula:

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}},$$

where $E\big(h(x)\big)$ is the average path length of data point x across t isolation trees:

$$E\big(h(x)\big) = \frac{\sum_{i=1}^{t} h_i(x)}{t},$$

and $c(n)$ represents the average path length of an unsuccessful search in a binary tree:

$$c(n) = 2H(n-1) - (2(n-1)/n),$$

where $H(i) = \ln(i) + \gamma$, and γ being Euler's constant.

Data points with shorter path lengths, closer to the root of the tree, are more likely to be anomalies, while those with longer paths are considered targets. Based on the anomaly scores, a threshold is set to classify data points as either anomalies or normal. Points with scores above the threshold are flagged as anomalies, while those below are classified as normal data points.

3.4.3. Autoencoder

An autoencoder is a type of artificial neural network designed for learning efficient data representations, dimensionality reduction, and anomaly detection. As an unsupervised learning method, it consists of two key components: an encoder and a decoder.

The primary goal of an autoencoder is to learn a compressed and meaningful representation of the input data. The encoder's function is to map the input data into latent space, effectively compressing the data into a lower-dimensional form. This is typically achieved through a series of layers, where each layer applies non-linear transformations to the input data. The resulting latent space captures the most relevant features and patterns of the input, condensing its essential information. The decoder, on the other hand, is tasked with reconstructing the original input data from its latent space representation. Its architecture generally mirrors that of the encoder, but in reverse, and it applies a series of non-linear transformations to transform the latent representation back into the original data format [25].

During training, the autoencoder aims to minimize reconstruction error, which quantifies the difference between the original input and the reconstructed output. This is typically achieved by optimizing a loss function, such as mean squared error or binary cross-entropy, using gradient-based methods like backpropagation.

In recognition tasks, the autoencoder is trained using only instances of the target class, allowing it to learn the typical patterns and structure of normal data. When the autoencoder encounters new data, it will reconstruct the input with a low error if it belongs to the target class. However, if the input represents an anomaly, the reconstruction error will be higher, as the autoencoder is not well-equipped to accurately reconstruct unfamiliar instances. By establishing a threshold for the reconstruction error, anomalies can be detected and distinguished from normal instances.

3.5. Evaluation metrics

In one-class classification, where the objective is to differentiate between target instances and anomalies, evaluation metrics such as specificity, recall, precision, F1 score, and accuracy are crucial for assessing model performance [26].

These metrics are derived from the classification outcomes, which can be categorized into four groups: true positives (TP), representing correctly identified anomalies; false positives (FP), indicating instances mistakenly classified as anomalies; true negatives (TN), denoting correctly identified target instances; and false negatives (FN), reflecting actual anomalies that were misclassified as target instances.

Specificity evaluates the model's ability to correctly classify target instances, reflecting the proportion of accurately identified target instances out of all target instances:

$$Specificity = \frac{TN}{TN + FP}.$$

Recall gauges the model's ability to detect all actual anomalies, measuring the proportion of true anomalies correctly identified out of all existing anomalies:

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

Precision assesses the model's reliability when identifying anomalies, showing the proportion of true anomalies among all instances classified as anomalies:

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

F1 score provides a balanced evaluation by calculating the harmonic mean of precision and recall, offering a single metric that accounts for both aspects:

$$F1\ score = 2 * \frac{Precision * Recall}{Precision\ +\ Recall}.$$

Finally, the accuracy metric measures the overall correctness of the classification, taking both target instances and anomalies into account:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}.$$

After constructing the models, they will be evaluated using these metrics, enabling a comprehensive analysis of their performance.

4. Experiments

4.1. Data preparation and outlier removal

For the experiments, data with the identifier s015 was randomly selected for analysis, while data from s004 was used as a test set to evaluate the recognition of keystroke dynamics from a different individual. Outlier detection began by assessing whether the s015 dataset adhered to a multivariate normal distribution. The Mardia test revealed significant deviations, as the test statistic for multivariate skewness $N\beta_1/6$, at 391.54, exceeded the chi-square critical value of 215.53 for 165 degrees of freedom at a significance level of 0.005. Similarly, the multivariate kurtosis statistic β_2, with a value of 113.32, surpassing the critical value of 102.62 for a mean of 99, variance of 1.98, and significance level of 0.005, indicating non-normality and necessitating further normalization.

Normalization parameters were estimated using the maximum likelihood method, yielding the following estimates for the multivariate BCT: $\widehat{\lambda_1}$ = 0.9939, $\widehat{\lambda_2}$ = 1.3605, $\widehat{\lambda_3}$ = 1.2202, $\widehat{\lambda_4}$ = 1.7521, $\widehat{\lambda_5}$ = 2.2965, $\widehat{\lambda_6}$ = 1.0447, $\widehat{\lambda_7}$ = 1.6466, $\widehat{\lambda_8}$ = 1.3512, $\widehat{\lambda_9}$ = 2.0599.

After applying the nine-variate Box-Cox transformation with components (1), the Mardia test was performed again. The skewness statistic $N\beta_1/6$ was reduced to 212.07, which is below the chi-square threshold of 215.53, but the kurtosis statistic β_2 remained slightly elevated at 109.01, still above the critical value of 102.62. Despite some remaining non-normality, primarily due to outliers, the transformed dataset better approximated a multivariate normal distribution, improving the conditions for using SMD.

Subsequently, SMD was computed for each feature vector to identify potential outliers. These distances were compared to the chi-square critical value of 23.59 for 9 degrees of freedom at a 0.005 significance level. Any vectors with SMD exceeding this value were classified as outliers. The most extreme outlier, vector number 295 with an SMD of 37.44, was removed.

This process of outlier removal was iteratively repeated until all extreme points were excluded. After eliminating 6 outliers, the multivariate kurtosis statistic finally fell below the critical value, confirming that outliers had a substantial impact on the dataset's distribution.

Table 1 lists the SMD values and the corresponding indices for each outlier that was removed. This iterative process continued until no further significant outliers were detected, resulting in a refined dataset that was less affected by extreme values.

To mitigate any potential bias related to the order of the data, the final sample was randomly shuffled to ensure an even distribution across the training and test sets. The shuffled data was then split into two equal parts, with 195 vectors in each set.

The training set was utilized to build both the prediction ellipsoid and the machine learning models, allowing them to capture the underlying patterns and relationships within the data. Meanwhile, the test set was reserved to assess the performance of the models on data not previously encountered during training.

Table 1
Removed anomalies

№	SMD	Vector number	№	SMD	Vector number
1	37.44	295	6	26.963	323
2	36.962	160	7	26.868	45
3	30.742	306	8	25.776	263
4	28.833	388	9	24.515	294
5	28.662	214	10	23.972	204

Following this outlier removal process, the final set was obtained with the following vector of means: $\bar{X}$ = {0.07525; 0.07022; 0.07823; 0.063; 0.06911; 0.08829; 0.08605; 0.07505; 0.0751}, Table 2 presents the covariance matrix.

Table 2
The covariance matrix of the final set

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9
X_1	$0.0^{3}19$	$0.0^{4}26$	$0.0^{4}54$	$0.0^{4}13$	$0.0^{5}49$	$0.0^{4}12$	$0.0^{4}26$	$-0.0^{5}6$	$-0.0^{4}17$
X_2	$0.0^{4}26$	$0.0^{3}21$	$-0.0^{4}19$	$0.0^{5}51$	$0.0^{5}82$	$0.0^{4}14$	$0.0^{4}4$	$-0.0^{4}1$	$0.0^{5}14$
X_3	$0.0^{4}50$	$-0.0^{4}19$	$0.0^{3}25$	$0.0^{6}1$	$0.0^{5}35$	$0.0^{5}11$	$-0.0^{4}26$	$0.0^{4}27$	$0.0^{6}1$
X_4	$0.0^{4}13$	$0.0^{5}51$	$0.0^{6}1$	$0.0^{3}19$	$0.0^{4}16$	$-0.0^{5}52$	$0.0^{4}31$	$-0.0^{4}18$	$-0.0^{5}57$
X_5	$0.0^{5}49$	$0.0^{5}82$	$0.0^{5}35$	$0.0^{4}16$	$0.0^{3}13$	$0.0^{4}14$	$-0.0^{5}62$	$0.0^{5}56$	$0.0^{5}48$
X_6	$0.0^{4}12$	$0.0^{4}14$	$0.0^{5}11$	$-0.0^{5}52$	$0.0^{4}14$	$0.0^{3}12$	$-0.0^{5}36$	$0.0^{5}36$	$0.0^{5}86$
X_7	$0.0^{4}26$	$0.0^{4}4$	$-0.0^{4}26$	$0.0^{4}31$	$-0.0^{5}62$	$-0.0^{5}36$	$0.0^{3}35$	$-0.0^{4}38$	$0.0^{5}19$
X_8	$-0.0^{5}6$	$-0.0^{4}1$	$0.0^{4}27$	$-0.0^{4}18$	$0.0^{5}56$	$0.0^{5}36$	$-0.0^{4}38$	$0.0^{3}27$	$0.0^{5}7$
X_9	$-0.0^{4}17$	$0.0^{5}14$	$0.0^{6}1$	$-0.0^{5}57$	$0.0^{5}48$	$0.0^{5}86$	$0.0^{5}19$	$0.0^{5}7$	$0.0^{3}2$

Table 3
The covariance matrix of the training set

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9
X_1	$0.0^{3}2$	$0.0^{4}16$	$0.0^{4}62$	$0.0^{4}42$	$0.0^{5}53$	$0.0^{5}71$	$0.0^{4}38$	$0.0^{5}14$	$-0.0^{5}75$
X_2	$0.0^{4}16$	$0.0^{3}21$	$-0.0^{4}19$	$-0.0^{5}26$	$-0.0^{6}35$	$0.0^{4}14$	$0.0^{4}62$	$-0.0^{4}1$	$0.0^{5}15$
X_3	$0.0^{4}62$	$-0.0^{4}19$	$0.0^{3}26$	$-0.0^{6}84$	$0.0^{4}11$	$0.0^{5}71$	$-0.0^{4}51$	$0.0^{5}99$	$0.0^{4}17$
X_4	$0.0^{4}42$	$-0.0^{5}26$	$-0.0^{6}84$	$0.0^{3}21$	$0.0^{4}24$	$-0.0^{6}75$	$0.0^{4}37$	$-0.0^{4}28$	$-0.0^{5}87$
X_5	$0.0^{5}53$	$-0.0^{6}35$	$0.0^{4}11$	$0.0^{4}24$	$0.0^{3}13$	$0.0^{4}12$	$-0.0^{5}36$	$-0.0^{5}53$	$0.0^{5}72$
X_6	$0.0^{5}71$	$0.0^{4}14$	$0.0^{5}71$	$-0.0^{6}75$	$0.0^{4}12$	$0.0^{3}12$	$0.0^{4}1$	$0.0^{5}16$	$0.0^{4}18$
X_7	$0.0^{4}38$	$0.0^{4}62$	$-0.0^{4}51$	$0.0^{4}37$	$-0.0^{5}36$	$0.0^{4}1$	$0.0^{3}36$	$-0.0^{4}61$	$0.0^{6}69$
X_8	$0.0^{5}14$	$-0.0^{4}1$	$0.0^{5}99$	$-0.0^{4}28$	$-0.0^{5}53$	$0.0^{5}16$	$-0.0^{4}61$	$0.0^{3}27$	$0.0^{4}23$
X_9	$-0.0^{5}74$	$0.0^{5}15$	$0.0^{4}17$	$-0.0^{5}87$	$0.0^{5}72$	$0.0^{4}18$	$0.0^{6}69$	$0.0^{4}23$	$0.0^{3}22$

Table 3 presents the covariance matrix of the training set, which has the mean vector $\bar{X}$ = {0.07635; 0.07052; 0.07875; 0.06254; 0.06955; 0.08806; 0.08752; 0.07447; 0.07495}.

4.2. Prediction ellipsoid construction

The prediction ellipsoid should be constructed using data that follows a normal distribution, so verifying the data's normality is a necessary first step. Based on the Mardia test results, the multivariate distribution of this training sample deviates from normality. The test statistic for multivariate skewness $N\beta_1/6$ is 286.99, exceeding the critical value of 215.53 from the chi-square distribution for 165 degrees of freedom at a 0.005 significance level. Additionally, the test statistic for multivariate kurtosis β_2 is 105.43, also exceeding the critical value of 104.19, given a mean of 99, a variance of 4.062, and a 0.005 significance level.

To address this non-normality, the training set is normalized using a nine-variate BCT. The optimal parameters for this transformation were estimated using the maximum likelihood method: $\widehat{\lambda_1}$ = 1.3676, $\widehat{\lambda_2}$ = 1.4807, $\widehat{\lambda_3}$ = 1.078, $\widehat{\lambda_4}$ = 1.7393, $\widehat{\lambda_5}$ = 2.1004, $\widehat{\lambda_6}$ = 1.1498, $\widehat{\lambda_7}$ = 1.566, $\widehat{\lambda_8}$ = 1.1685, $\widehat{\lambda_9}$ = 2.1146.

After applying the BCT with components (1), the normalized training set has a mean vector $\bar{Z}$ = {0.70932; 0.66184; 0.86764; -0.57016; -0.47427; 0.81642; 0.62417; -0.81443; -0.47084}. The covariance matrix S_Z is presented in Table 4.

Table 4
The covariance matrix of the normalized training set

	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7	Z_8	Z_9
Z_1	$0.0^4 29$	$0.0^5 14$	$0.0^4 2$	$0.0^5 2$	$0.0^6 13$	$0.0^5 21$	$0.0^5 39$	$0.0^7 73$	$-0.0^7 68$
Z_2	$0.0^5 14$	$0.0^4 16$	$-0.0^5 48$	$-0.0^7 21$	$0.0^7 71$	$0.0^5 27$	$0.0^5 43$	$-0.0^5 16$	$0.0^7 31$
Z_3	$0.0^4 2$	$-0.0^5 48$	$0.0^3 18$	$-0.0^6 42$	$0.0^6 43$	$0.0^5 42$	$-0.0^4 11$	$0.0^5 51$	$0.0^6 65$
Z_4	$0.0^5 2$	$-0.0^7 21$	$-0.0^6 42$	$0.0^5 31$	$0.0^6 15$	$-0.0^6 15$	$0.0^5 14$	$-0.0^5 2$	$-0.0^7 32$
Z_5	$0.0^6 13$	$0.0^7 71$	$0.0^6 43$	$0.0^6 15$	$0.0^6 33$	$0.0^6 58$	$0.0^8 8$	$-0.0^6 17$	$0.0^6 13$
Z_6	$0.0^5 21$	$0.0^5 27$	$0.0^5 42$	$-0.0^6 15$	$0.0^6 58$	$0.0^4 55$	$0.0^5 15$	$0.0^6 92$	$0.0^6 72$
Z_7	$0.0^5 39$	$0.0^5 43$	$-0.0^4 11$	$0.0^5 14$	$0.0^8 8$	$0.0^5 15$	$0.0^4 22$	$-0.0^4 1$	$0.0^7 12$
Z_8	$0.0^7 73$	$-0.0^5 16$	$0.0^5 51$	$-0.0^5 2$	$-0.0^6 17$	$0.0^6 92$	$-0.0^4 1$	$0.0^3 11$	$0.0^6 89$
Z_9	$-0.0^7 68$	$0.0^7 31$	$0.0^6 66$	$-0.0^7 32$	$0.0^7 13$	$0.0^6 72$	$0.0^7 12$	$0.0^6 89$	$0.0^6 57$

The Mardia test performed on the normalized training set indicates conformity with multivariate normality. The test statistic for multivariate skewness $N\beta_1/6$ is 175.47, which is below the critical value of 215.53. Similarly, the test statistic for multivariate kurtosis β_2 is 99.76, which does not exceed the critical value of 104.19, confirming that the normalized set follows a multivariate normal distribution.

4.3. Implementation of machine learning algorithms

This section outlines the implementation of various machine learning algorithms used to recognize the keystroke dynamics data. Specifically, we explore the One-Class Support Vector Machine, Isolation Forest, and autoencoder models, each selected for their unique capabilities in anomaly detection and one-class classification.

The OCSVM is implemented in Python using the OneClassSVM object from the scikit-learn library. This implementation allows for the customization of several critical parameters, including the kernel function and the regularization parameter ν. For our analysis, we set ν to 0.03, which determines the acceptable proportion of training errors and establishes an upper limit for the fraction of outliers in the training dataset. The radial basis function kernel was chosen for its flexibility in modeling non-linear relationships among data points. The gamma parameter is set to "auto," allowing its value to be computed automatically based on the inverse of the number of features, influencing the range for each training example; lower values extend the influence while higher values localize it.

The IF algorithm, also implemented through sci-kit-learn [27], provides several tunable parameters for optimizing performance. A key parameter is the contamination level, which defines the threshold for categorizing new data points as either target or anomalous. After experimentation, a contamination value of 0.05 was determined to effectively balance the detection of true anomalies against false positives.

Additional significant parameters include n_estimators, which denotes the number of decision trees in the forest (set at 100), max_samples, indicating the maximum number of samples per tree (set to 256), and max_features, specifying the maximum number of features for splitting each node (set to 1.0 to utilize all features). To classify a sample as either target or anomalous, we compare the anomaly score against a defined threshold. The scores can range from negative to positive values; negative scores indicate a higher likelihood of being a target, while positive scores suggest a greater probability of being anomalous. The selection of the threshold value is application-dependent; in this analysis, a threshold of 0 yielded optimal results.

For the AE model, we utilized TensorFlow and Keras, leveraging their combined strengths in flexibility, scalability, and ease of use. Keras, as a high-level API for building neural networks atop TensorFlow, simplifies the process of constructing and training models. Meanwhile, TensorFlow provides the essential computational framework, ensuring efficient performance during training and inference.

Before passing the data into the neural network, min-max normalization is applied to each feature individually, scaling all features to a range of [0, 1]. This technique standardizes the features, promoting stable and efficient learning processes.

The AE architecture consists of an input layer configured to accept a nine-variate representation of the data. The model includes fully connected layers for encoding and decoding operations. During the encoding phase, the input data is compressed into a lower-dimensional representation, progressively reducing dimensionality from 9 to 8 and then to 6, creating a bottleneck in the network structure. This bottleneck layer compels the model to capture essential features of the input data while minimizing redundancy [28].

Each encoding layer employs rectified linear unit ReLU activation functions, introducing non-linearity that facilitates the extraction of complex features. The decoding phase reverses this process, expanding the dimensionality back to 8 and ultimately to the original 9 dimensions, using ReLU activation functions to retain the learned non-linear relationships. The final layer utilizes a sigmoid activation function to constrain output values within the range of [0, 1], a common choice for reconstruction and binary classification tasks that require smooth and interpretable outputs. The structure of the AE is illustrated in Figure 1.

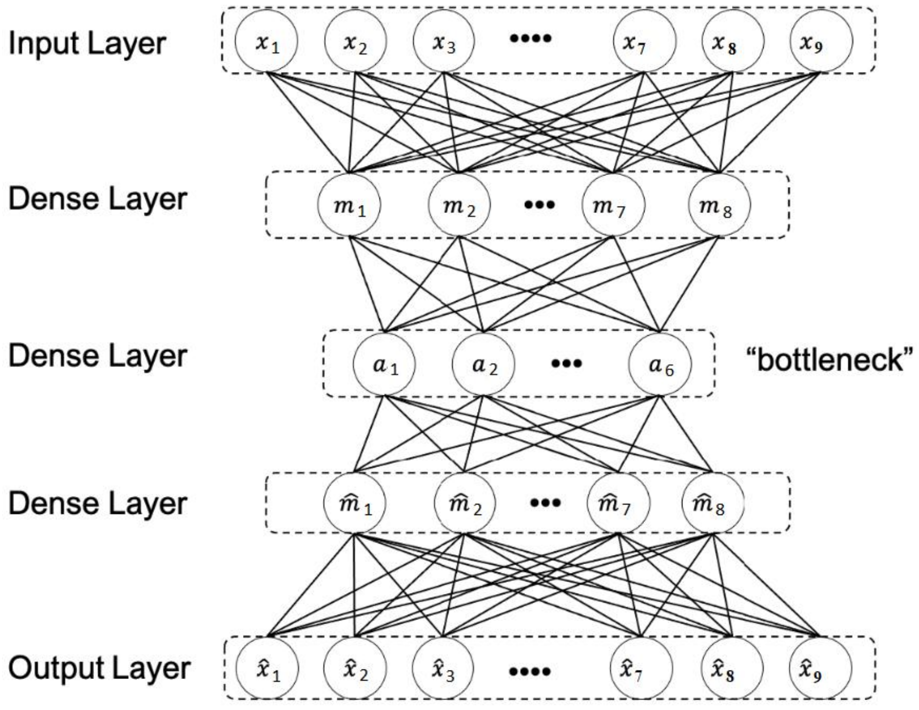
Input Layer
x_1 x_2 x_3 •••• x_7 x_8 x_9
Dense Layer
m_1 m_2 ••• m_7 m_8
Dense Layer
a_1 a_2 ••• a_6
"bottleneck"
Dense Layer
$\hat{m}_1$ $\hat{m}_2$ ••• $\hat{m}_7$ $\hat{m}_8$
Output Layer
$\hat{x}_1$ $\hat{x}_2$ $\hat{x}_3$ •••• $\hat{x}_7$ $\hat{x}_8$ $\hat{x}_9$

Figure 1: Autoencoder structure.

To train the model, we employed the Adam optimizer in conjunction with binary cross-entropy loss, a standard metric for reconstruction tasks aimed at minimizing the discrepancy between the original and reconstructed data. The Adam optimizer combines the strengths of AdaGrad and

RMSProp [29], dynamically adjusting the learning rate during training for faster convergence and improved performance. The binary cross-entropy loss effectively measures the difference between the input and reconstructed outputs, making it suitable for binary classification problems. The training process encompasses 25 epochs, with a batch size of 16 instances per batch. Shuffling the data at each epoch introduces variability, preventing the model from memorizing the training sequence, and thus enhancing generalization.

5. Results

Table 5 displays a comparison of the recognition performance among various methods, including the Prediction Ellipsoid for Non-Gaussian Data (PENGD) (1), the Prediction Ellipsoid for Normalized Data (PEND) (7), One-Class Support Vector Machine (OCSVM), Isolation Forest (IF), and Autoencoder (AE).

Table 5
Comparison of models

Model	Specificity	Recall	Precision	F1 score	Accuracy
PENGD	0.9795	0.9225	0.9893	0.9547	0.9412
PEND	0.9949	0.9700	0.9974	0.9835	0.9782
OCSVM	0.9744	0.9675	0.9872	0.9773	0.9697
IF	0.9333	0.9500	0.9669	0.9584	0.9445
AE	0.9641	0.9625	0.9821	0.9722	0.9630

All models evaluated in this study demonstrate commendable performance in keystroke dynamics recognition. However, the PENGD stands out with the lowest accuracy among the models assessed, indicating that while it can capture some patterns, it may struggle with more complex datasets, particularly due to the challenges posed by non-Gaussian data distributions. On the other hand, both the OCSVM and AE exhibit very good performance across multiple metrics, reflecting their capabilities in identifying true anomalies with high precision and recall. These models effectively leverage their respective architectures to capture intricate relationships within the data, contributing to their robust performance. In contrast, the IF did not perform as well as the other models.

Ultimately, the PEND emerged as the best-performing model, achieving the highest scores across key evaluation metrics. This reinforces the significance of normalization transformations in enhancing prediction ellipsoid models for recognition tasks, particularly in scenarios involving non-Gaussian data distributions.

6. Discussion

All models in this study exhibit strong performance in keystroke dynamics recognition, but PEND stands out as the best performer. The precision, recall, and F1 score of this model are the highest, demonstrating its ability to handle keystroke dynamics recognition tasks with remarkable accuracy. The performance of OCSVM and AE is also notable, offering very good results, while IF lags slightly behind the others.

The findings underscore that applying the nine-variate BCT played a critical role in boosting model performance, particularly by improving how the models handle non-Gaussian data. Multivariate transformations like BCT take into account the correlations between variables, allowing for a more accurate and comprehensive prediction ellipsoid. This, in turn, enhances the model's ability to identify intricate patterns in the data, improving both its accuracy and reliability.

However, there are certain disadvantages to using prediction ellipsoid for normalized data. A robust model typically requires a dataset of at least 100 instances, which can be a challenge for smaller datasets. Additionally, selecting the most appropriate normalization transformation can be complex, especially for datasets with intricate distributions or a large number of outliers. Another important factor is the choice of significance level, as this can influence the efficiency and reliability of the prediction ellipsoid.

Limitations also arise from the outlier removal process, as deleting 10 outliers during preprocessing may cause the model to miss some underlying patterns in the data. To mitigate this, more advanced normalization techniques, such as the Johnson transformation, could be considered to better align the model with the dataset's distribution, improving its ability to generalize across all relevant data points.

In this paper, the primary aim was to address the challenge posed by non-Gaussian data distributions in the context of biometric identification based on keystroke dynamics. Emphasizing the importance of normalization techniques, specifically the multivariate Box-Cox transformation, to enhance model accuracy with such data.

The dataset used in this study represents a 10-character password length, which may not be optimal for real-world applications. A password length of 20-22 characters, without the use of uppercase characters, is generally considered preferable, as it allows for more comprehensive feature extraction. Beyond keystroke length and character variety, several contextual factors, such as the user’s physical condition, time of day, mental state, and weather [30], were not considered in this research. However, these factors could play an important role in biometric identification based on keystroke dynamics.

In future research, a broader dataset that includes data reflecting the impact of environmental factors could be used, along with extended key sequences. The inclusion of these factors would provide a more realistic representation of user behavior. Additionally, the application of other normalization techniques, such as the Johnson transformation, could further enhance model accuracy by addressing the complexity of non-Gaussian distributions.

7. Conclusions

The focus of this paper was to address the challenges associated with non-Gaussian data distributions in the context of keystroke dynamics recognition. The study compared the performance of prediction ellipsoid models and machine learning algorithms, including OCSVM, IF, and AE. All models demonstrated a high probability of recognition. Notably, the prediction ellipsoid for non-Gaussian data had the lowest accuracy, highlighting the challenges posed by complex datasets. However, by applying the multivariate BCT, the prediction ellipsoid model for normalized data showed significant performance improvements, emphasizing the critical role of normalization when addressing non-Gaussian data distributions. The BCT not only improved the overall accuracy but also deepened the understanding of data patterns by considering correlations between variables, ultimately leading to a more precise prediction ellipsoid.

Despite these advancements, the study identified certain limitations and challenges. One significant drawback is the necessity for a large dataset, as constructing a reliable prediction ellipsoid model generally requires at least 100 instances. Furthermore, selecting the optimal normalization transformation remains a complex task, especially when dealing with datasets that contain outliers or exhibit highly intricate distributions. Another challenge lies in determining the appropriate significance level, which directly affects the reliability and efficiency of the prediction ellipsoid.

Looking ahead, future research could expand the dataset to include factors like environmental factors, as well as extended key sequences, to provide a more realistic representation of user behavior.

The incorporation of alternative normalization techniques, such as the Johnson transformation, could further enhance model accuracy by addressing the impact of non-Gaussian data. Further investigation into model complexity and feature selection for both prediction ellipsoid models and machine learning algorithms could offer valuable insights for improving keystroke dynamics recognition.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] L. de-Marcos, J. Martínez-Herráiz, J. Junquera-Sánchez, C. Cilleruelo, C. Pages-Arévalo, Comparing machine learning classifiers for continuous authentication on mobile devices by keystroke dynamics. Electronics, 2021. DOI: 10.3390/electronics10141622

[2] A. Alshehri, F. Coenen, D. Bollegala, Accurate Continuous and Non-intrusive User Authentication with Multivariate Keystroke Streaming, in: 9th International Conference on Knowledge Discovery and Information Retrieval, pp. 61-70, 2017. DOI: 10.5220/0006497200610070.

[3] G. Ismail M, Salem, Abd El Ghany, Aldakheel EA, S. Abbas, Outlier detection for keystroke biometric user authentication. PeerJ Computer Science, 2024. DOI: 10.7717/peerj-cs.2086

[4] M. Choi, S. Lee, M. Jo, JS. Shin, Keystroke dynamics-based authentication using unique keypad. Sensors, 2021; 21(6):2242. DOI: 10.3390/s21062242

[5] H. Marques, L. Swersky, J. Sander, R. Campello, A. Zimek, On the evaluation of outlier detection and one-class classification: a comparative study of algorithms, model selection, and ensembles. Data Min Knowl Disc 37, 1473–1517, 2023. DOI: 10.1007/s10618-023-00931-x

[6] S. Kim, D. Park, J. Jung, Evaluation of one-class classifiers for fault detection: Mahalanobis classifiers and the Mahalanobis–Taguchi system, Processes, 9(8), 1450, 2021. DOI: 10.3390/pr9081450

[7] H. Chang, J. Li, C. Wu, M. Stamp, Machine learning and deep learning for fixed-text keystroke dynamics. Artificial Intelligence for Cybersecurity, pp. 309-329, 2022. DOI: 10.48550/arXiv.2107.00507

[8] B. Saini, P. Singh, A. Nayyar, N. Kaur, K. Bhatia, S. El-Sappagh, J. Hu, A three-step authentication model for mobile phone user using keystroke dynamics. IEEE Access. 8. 125909-125922, 2020. DOI: 10.1109/ACCESS.2020.3008019

[9] Q. Li, H. Chen, CDAS: A continuous dynamic authentication system, in: Proceedings of the 2019 8th International Conference on Software and Computer Applications, pp. 447-452, 2019. DOI: 10.1145/3316615.3316691

[10] A. Sharma, M. Jureček, M. Stamp, Keystroke dynamics for user identification. arXiv preprint arXiv:2307.05529, 2023. DOI: 10.48550/arXiv.2307.05529

[11] N. Raul, R. Shankarmani, P. Joshi, A comprehensive review of keystroke dynamics-based authentication mechanism, in: International Conference on Innovative Computing and Communications. Advances in Intelligent Systems and Computing, Vol. 1059. Springer, Singapore, 2020. DOI: 10.1007/978-981-15-0324-5_13

[12] R. Toosi, M. Akhaee, Time-frequency analysis of keystroke dynamics for user authentication. Future Generation Computer Systems, 115, 438-447, 2021. DOI: 10.1016/j.future.2020.09.027

[13] ML Ali, K. Thakur, MA Obaidat, A hybrid method for keystroke biometric user identification. Electronics, 11(17):2782, 2022. DOI: 10.3390/electronics11172782

[14] I. Meenakshisundaram, I. Karunanithi, U. Sahana, Enhancing user authentication through keystroke dynamics analysis using isolation forest algorithm, in: 2024 Second International Conference on Emerging Trends in Information Technology and Engineering (ICETITE), pp. 1-5, 2024. DOI: 10.1109/ic-ETITE58242.2024.10493648.

[15] F. Trad, A. Hussein, A. Chehab, Free text keystroke dynamics-based authentication with continuous learning: a case study, in: 2022 IEEE 21st International Conference on Ubiquitous Computing and Communications (IUCC/CIT/DSCI/SmartCNS), Chongqing, China, 2022, pp. 125-131. DOI: 10.1109/IUCC-CIT-DSCI-SmartCNS57392.2022.00031

[16] Y. Patel, K. Ouazzane, V. Vassilev, I. Faruqi, G. Walker, Keystroke dynamics using auto encoders, in: 2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), pp. 1-8. Oxford, UK, 2019. DOI: 10.1109/CyberSecPODS.2019.8885203

[17] S. Prykhodko, L. Makarova, K. Prykhodko, A. Pukhalevych, Application of transformed prediction ellipsoids for outlier detection in multivariate non-gaussian data, in: 2020 IEEE 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), pp 359-362, 2020. DOI: 10.1109/TCSET49122.2020.235454.
[18] O. Oyebola, Examining the distribution of keystroke dynamics features on computer, tablet and mobile phone platforms, in: Mobile Computing and Sustainable Informatics: Proceedings of ICMCSI 2023, pp. 613-620. Singapore: Springer Nature Singapore. DOI: 10.1007/978-981-99-0835-6_43
[19] K. Lam, K. Meijer, F. Loonstra, E. Coerver, J. Twose, E. Redeman, J. Killestein, Real-world keystroke dynamics are a potentially valid biomarker for clinical disability in multiple sclerosis. Multiple Sclerosis Journal, 27(9), 1421-1431, 2021. DOI: 10.1177/1352458520968797
[20] S. Prykhodko, A. Prykhodko, I. Shutko, Estimating the size of web apps created using the CakePHP framework by nonlinear regression models with three predictors, in: IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT), pp. 333-336. LVIV, Ukraine, 2021. DOI: 10.1109/CSIT52700.2021.9648680
[21] Prykhodko S., Trukhov A. Application of a ten-variate prediction ellipsoid for normalized data and machine learning algorithms for face recognition. in: Selected Papers of the Seventh International Workshop on Computer Modeling and Intelligent Systems (CMIS-2024). Workshop Proceedings (CMIS-2024), Zaporizhzhia, Ukraine, May 3, 2024. CEUR Workshop Proceedings, Vol.3702, pp. 362-375, 2024. https://ceur-ws.org/Vol-3702/paper30.pdf.
[22] T. Etherington, Mahalanobis distances for ecological niche modelling and outlier detection: implications of sample size, error, and bias for selecting and parameterising a multivariate location and scatter method. PeerJ, 9. 2021. DOI: 10.7717/peerj.11436
[23] S. Todkar, V. Baltazart, A. Ihamouten, X. Dérobert, D. Guilbert, One-class SVM based outlier detection strategy to detect thin interlayer debondings within pavement structures using Ground Penetrating Radar data. Journal of Applied Geophysics, 192, 104392, 2021. DOI: 10.1016/j.jappgeo.2021.104392
[24] J. Lesouple, C. Baudoin, M. Spigai, JY. Tourneret, Generalized isolation forest for anomaly detection. Pattern Recognition Letters, Vol. 149, pp. 109-119, 2021. DOI: 10.1016/j.patrec.2021.05.022
[25] W. Xu, J. Jang-Jaccard, A. Singh, Y. Wei, F. Sabrina, Improving performance of autoencoder-based network anomaly detection on NSL-KDD dataset. IEEE Access, Vol. 9, pp. 140136-140146, 2021. DOI: 10.1109/ACCESS.2021.3116612
[26] W. Hilal, S. A. Gadsden, J. Yawney, Financial Fraud: A review of anomaly detection techniques and recent advances. Expert Systems with Applications, Vol. 193, 2022, 116429. DOI: 10.1016/j.eswa.2021.116429
[27] M. U. Togbe, M. Barry, A. Boly, Y. Chabchoub, R. Chiky, J. Montiel, T.V. Tran, Anomaly detection for data streams based on isolation forest using scikit-multiflow, in: Computational Science and Its Applications–ICCSA 2020: 20th International Conference, Cagliari, Italy, July 1–4, 2020, Proceedings, Part IV 20 (pp. 15-30). Springer International Publishing. DOI: 10.1007/978-3-030-58811-3_2
[28] M. Sewak, S. K. Sahay, H. Rathore, An overview of deep learning architecture of deep neural networks and autoencoders. Journal of Computational and Theoretical Nanoscience, Vol. 17, No. 4, pp. 182-188. 2020. DOI: 10.1166/jctn.2020.8648.
[29] I. H. Kartowisastro, J. Latupapua, A comparison of adaptive moment estimation and rmsprop optimisation techniques for wildlife animal classification using convolutional neural networks. Revue d'Intelligence Artificielle, Vol. 37, No. 4, pp. 1023-1030. 2023. DOI: 10.18280/ria.370424
[30] S. Bilan, M. Bilan, A. Bilan, interactive biometric identification system based on the keystroke dynamic, in: S. Bilan, M. Elhoseny, D. J. Hemanth (Eds.), Biometric Identification Technologies Based on Modern Data Mining Methods. Springer, Cham, pp. 39-58, 2021. DOI: 10.1007/978-3-030-48378-4_3