Global Economic Horizons

ASPUR PUBLISHING

Vol. 01, (2025) 113-122, doi: 10.61552/GEH.2025.01.015 - http://geh.aspur.rs

OPTIMIZING NETWORK INTRUSION DETECTION FOR ENHANCING DIGITAL FINANCIAL SYSTEM SECURITY AND INVESTMENT ENVIRONMENT STABILITY

Phan-Anh-Huy Nguyen [1]
Anh Huynh Van
Duyen Ngo Ky
Nhi Nguyen Thi Yen
Vy Nguyen Thi Thuy

Received 11.08.2025.
Revised 06.09.2025.
Accepted 19.10.2025.

Keywords:

K Nearest Neighbors (KNN) classification; PCA feature, Netflow Datasets, Binary classification, Multiclass classification, Intrusion Detection System.

Original research

A B S T R A C T

As internal networks become increasingly essential for modern IT systems, they face heightened risks from cyber threats, including network intrusions that can disrupt operations and compromise sensitive data. This study focuses on enhancing Intrusion Detection Systems (IDS) by utilizing NetFlow datasets to detect anomalies and potential intrusions efficiently. By integrating Principal Component Analysis (PCA) for feature reduction and applying Grid Search for hyperparameter tuning, the proposed model achieves faster execution times and improved detection accuracy. Initial evaluations demonstrate the model's effectiveness in binary classification scenarios, suggesting its scalability for multi-class classification tasks. These findings contribute to developing a practical and adaptable solution for securing internal networks against evolving threats. This study achieved over a 97% reduction in training time through the Binary Classification experiment and improved training accuracy by over 12% and test accuracy by 21% through the Multiclass Classification experiment.

1. INTRODUCTION

Internal networks serve as the backbone of modern IT infrastructures, ensuring smooth communication and operational efficiency. However, the increasing sophistication of cyberattacks, particularly network intrusions, poses significant challenges to maintaining network security. These intrusions can lead to data breaches, operational disruptions, and financial losses, emphasizing the need for advanced detection systems.

Traditional Intrusion Detection Systems (IDS) often face significant hurdles when tasked with processing the ever-growing volume and complexity of network data, frequently struggling to maintain both high accuracy and efficiency simultaneously (Bulajoul et al., 2013). This inherent limitation underscores the critical need for innovative methodologies to substantially improve IDS performance (Kotha, 2017) parameters (Mavikumbure et al., 2024).

This paper explores the methodologies for building an improved IDS model using NetFlow data, including data preparation, feature extraction, and model optimization. The results demonstrate the potential of combining PCA and advanced optimization techniques to create a scalable and robust solution for detecting network intrusions in real-world environments.

[1] Corresponding author: Phan-Anh-Huy Nguyen
Email: huynpa@hcmute.edu.vn

The remainder of this paper is arranged in the following manner: Section 2 presents the literature review, Section 3 covers the research method, Section 4 discusses the experimental results, and the study concludes in Section 5.

2. LITERATURE REVIEW

2.1 Intrusion Detection System

Intrusion Detection Systems (IDSs) are essential tools for monitoring and analyzing events in computer networks to identify security breaches and unauthorized access attempts (Bace & Mell, 2001; Prasad et al., 2015). These systems can be implemented as software or hardware solutions and are designed to detect various types of attacks, including those from external sources and insider threats (Mell, 2001). IDSs have become increasingly important due to the rising number and severity of network attacks in recent years (Bace & Mell, 2001). They employ various techniques such as machine learning, DNA sequencing, pattern matching, and data mining to learn from past attacks and prevent similar future intrusions (Shrivastava, 2017). While IDSs are crucial for maintaining network security, they should not be used in isolation but rather as part of a comprehensive IT security framework (Mell, 2001). The effectiveness of IDSs depends on proper selection, configuration, and management within specific system and network environments (Bace & Mell, 2001).

2.2 KNN Classification Model Analysis

The K-Nearest Neighbors (KNN) algorithm is a widely used classification method in machine learning and data mining. While effective, traditional KNN faces challenges such as low efficiency and dependency on optimal k-value selection (Guo et al., 2003). To address these issues, researchers have proposed various improvements and analyses. Guo et al. (2003) introduced a KNN model-based approach that automatically determines the optimal k-value and improves classification speed. Normalization techniques, such as Z-Score and Min-Max, have been explored to enhance KNN's performance on datasets like IRIS (Pandey & Jain, 2017). Boyko & Muzyka (2021) investigated multimodal data analysis methods to increase overall accuracy and minimize risks in KNN classification. Additionally, different distance measures, including Euclidean, Chebychev, and Manhattan, have been compared for KNN implementation, with Manhattan distance showing high performance on the KDD dataset (Mulak & Talhar, 2015). These studies collectively contribute to improving KNN's accuracy, efficiency, and applicability in various classification tasks.

2.3 Optimizing KNN Algorithms

Recent research has focused on optimizing K-Nearest Neighbors (KNN) algorithms for improved performance and efficiency. Cao et al. (2023) proposed a strategy combining balanced KD trees and multi-threaded parallel computing to enhance search speed while maintaining accuracy for large-scale datasets. Japa et al. (2019) developed an optimization algorithm using vector space models to restrict the search area and reduce comparisons. Dadhania and Dhobi (2012) improved the traditional KNN algorithm by optimizing cross-validation, reducing processing time and space requirements. Peterson et al. (2005) explored genetic algorithm-facilitated KNN optimization using various similarity measures, including Euclidean distance, cosine similarity, and Pearson correlation. They found that weight-optimized classifiers using cosine similarity or Pearson correlation often outperformed their Euclidean counterparts. These studies demonstrate the potential for significant improvements in KNN algorithm performance through various optimization techniques, addressing challenges such as high computational complexity and low search efficiency in large-scale and high-dimensional datasets.

2.4 Principle Component Analysis (PCA)

Principal Component Analysis (PCA) is a widely used technique for dimensionality reduction and feature extraction in various applications. Several studies have explored PCA-based methods for feature selection. Song et al. (2010) proposed a method that evaluates feature significance using PCA eigenvectors, effectively reducing dimensionality while maintaining recognition accuracy. Murali (2015) applied PCA to extract feature vectors from face images, highlighting its effectiveness in identifying similarities and differences in large datasets. Luo et al. (2008) introduced the concept of PC dominant features (PCDF) and developed an algorithm to select original features based on PCs. Cui and Dy (2008) presented an orthogonal principal feature selection method that selects original features most correlated with principal components while minimizing correlation between selected features. This approach preserves the PCA property of expressing retained variance as the sum of orthogonal feature variances, demonstrating improved performance compared to non-orthogonalized methods.

2.5 Optimizing KNN via Grid Search

Grid search optimization of K-nearest Neighbor (KNN) algorithm has shown significant improvements in various classification tasks. In lung cancer prediction, grid search cross-validation (CV) identified an optimal K value of 3, achieving 96% accuracy (Kusuma & Sasongko, 2023). Similarly, for breast cancer detection, grid search optimization increased accuracy from 90.10% to 94.35% (Assegie, 2021). In preeclampsia classification with imbalanced data, KNN optimized by grid search outperformed decision trees (Sukamto et al., 2023). For text categorization using KNN with BM25 similarity, an efficient grid search technique was proposed to tune three hyperparameters, resulting in at least a tenfold speedup compared to conventional methods (Ghawi & Pfeffer, 2019). These studies demonstrate that grid search optimization can significantly enhance KNN performance across various domains, including medical diagnostics and text classification, by finding optimal

hyperparameter combinations and addressing challenges such as imbalanced datasets.

2.6 Addressing Imbalanced Datasets

Imbalanced datasets, where one class is significantly underrepresented, pose challenges in classification tasks across various domains (Ramyachitra & Manikandan, 2014). Standard classifiers often struggle with imbalanced data, as they tend to favor the majority class and overlook the minority class (Chawla, 2005). To address this issue, researchers have developed sampling techniques to balance datasets, including under-sampling and over-sampling methods (Hoens & Chawla, 2013). Additionally, novel classification algorithms that are less sensitive to class imbalance have been proposed (Hoens & Chawla, 2013). Evaluating classifier performance on imbalanced datasets requires careful consideration, as traditional accuracy metrics may not be appropriate (Chawla, 2010). Alternative performance measures have been suggested to better assess classifiers in imbalanced scenarios (Chawla, 2010). The imbalanced dataset problem is prevalent in various applications, such as fraud detection, biomedical research, and remote sensing, highlighting the importance of developing effective solutions to handle this challenge (Ramyachitra & Manikandan, 2014).

2.7 Binary Classification Techniques

Binary classification is a fundamental task in machine learning that involves categorizing elements into two distinct classes (Kumari & S. Srivastava, 2017). It has wide-ranging applications across various fields, including information technology, business, and medical diagnosis (Phetlasy et al., 2015). Researchers have proposed diverse approaches to improve classification accuracy, such as combining multiple classifiers sequentially to minimize false positives and false negatives (Phetlasy et al., 2015). Some methods formulate binary classification as a generalized eigenvalue problem, offering comparable accuracy with lower computational complexity (Guarracino et al., 2007). More recent innovations include hypergraph-based algorithms that can handle unstructured data, multiple data sources, and missing values, reducing the need for extensive preprocessing and feature engineering (Quemy, 2019). These advancements in binary classification techniques continue to enhance performance and applicability across various domains, addressing challenges in areas such as text categorization and sockpuppet detection (Kumari & Srivastava, 2017).

2.8 Multiclass Classification Techniques

Multiclass classification is a crucial machine learning problem with applications in various domains. Several approaches have been proposed to improve classification accuracy and efficiency. Ridge regularization and shared hyperparameters have been shown to enhance the performance of linear combinations of multiclass classifiers (Reid, 2010). A novel method combining binary pairwise classifiers has demonstrated superior results compared to existing schemes (Reid, 2010). Another approach, inspired by information transmission theory, models misclassification as bit inversion errors and incorporates classifier dependence using a Boltzmann machine decoder (Takenouchi & Ishii, 2009). Random Forest classifiers have also been successfully applied to multiclass problems, achieving high prediction accuracy (Agarwal et al., 2022). When comparing different multiclass classification methods, "one-against-one" and directed acyclic graph SVM (DAGSVM) approaches have been found to be more suitable for practical use, particularly in remote sensing applications (Sonar & Deshmukh, 2014)

3. METHOD

In this study, we use the K-Nearest Neighbors (KNN) model for intrusion detection in the internal network based on NetFlow datasets, aiming to improve performance through two experiments: Binary Classification and Multiclass Classification. First, we apply Principal Component Analysis (PCA) to reduce the dimensionality of the data and use Grid Search to find the optimal hyperparameters in the Binary Classification experiment. The KNN model is then optimized and its performance tested. To address class imbalance in the Multiclass Classification experiment, we combine data balancing techniques, followed by model optimization, which helps improve the model's accuracy. These experiments allow us to evaluate and optimize the KNN model for network intrusion detection.

3.1 Dataset Description

In this study, the authors' team uses the TON_IoT dataset referenced in the paper "NetFlow Datasets for Machine Learning-based Network Intrusion Detection Systems," with data sourced from Kaggle. The paper introduces an experiment on a new IoT network architecture to evaluate AI applications, with data from IoT services, operating systems, and storage networks. Instead of using a regular dataset, the NetFlow dataset is chosen because it provides detailed and real-time information about network traffic, including source and destination ports, protocols, and packet counts. This level of granularity is crucial for accurately detecting intrusions within an internal network. NetFlow data enables continuous monitoring and analysis of network behavior, making it an ideal choice for identifying abnormal patterns and potential security threats, which traditional datasets might not capture as effectively.

Dataset: NF_ToN_IoT Dataset

Link:

https://www.kaggle.com/datasets/dhoogla/nftoniot/data

	L4_SRC_PORT	L4_DST_PORT	PROTOCOL	L7_PROTO	IN_BYTES	OUT_BYTES	IN_PKTS	OUT_PKTS	TCP_FLAGS	FLOW_DURATION_MILLISECONDS	Label	Attack
0	63318	443	6	91.00	181	165	2	1	24	327	0	0
1	57442	15600	17	0.00	63	0	1	0	0	0	0	0
2	57452	15600	17	0.00	63	0	1	0	0	0	0	0
3	138	138	17	10.16	472	0	2	0	0	0	0	0
4	51989	15600	17	0.00	63	0	1	0	0	0	0	0

Figure 1. The data used

The dataset consists of 12 columns, which are the attributes of network flows, and multiple rows, each corresponding to a single network flow (Figure 1):

1. **L4_SRC_PORT:** Source port of the Layer 4 protocol. This is the port from which data is sent from the originating device.
2. **L4_DST_PORT:** Destination port of the Layer 4 protocol. This is the port to which data is sent on the receiving device.
3. **PROTOCOL:** The protocol being used.
4. **L7_PROTO:** Layer 7 protocol, corresponding to specific applications or services.
5. **IN_BYTES:** Number of incoming bytes. This represents the amount of data transmitted to the device.
6. **OUT_BYTES:** Number of outgoing bytes. This represents the amount of data sent from the device.
7. **IN_PKTS:** Number of incoming packets.
8. **OUT_PKTS:** Number of outgoing packets.
9. **TCP_FLAGS**: Flags of the TCP protocol, indicating the connection's state.
10. **FLOW_DURATION_MILLISECONDS:** The duration of the network flow, measured in milliseconds.
11. **Label:** Data label, typically used for classification purposes.
12. **Attack:** Indicates whether the flow represents an attack (0: not an attack, 1: an attack).

3.2 Method details

3.2.1 K-Nearest Neighbors (KNN) classification model

K-nearest neighbors (KNN) is a simple machine learning algorithm used to classify data points by calculating the distance between them. In KNN, the predicted class for each test sample is the class that the majority of the nearest neighbors (k) from the training set belong to.

Let the training set be $T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where x_i is the feature vector of a sample and $y_i \in \{c_1, c_2, \ldots, c^m\}$ is the corresponding class label, with $i = (1, 2, \ldots, n)$. For a test sample x, its class y can be determined by the following formula:

$$y = \arg max_{c_j} \sum_{x_i \epsilon N_k(x)} I(y_i = c_j),$$
$$i = 1,2, \ldots . n; j = 1,2, \ldots, m,$$

Where I(x) is an indicator function, I = 1 when $y_i = c_j$, otherwise I = 0; $N_k(x)$ is the set of the k-nearest neighbors of x.

The KNN algorithm assumes that similar data points are located close to each other in the feature space. The main task is to determine the k nearest points to the test data. To calculate the distance between two points, there are various formulas, and the choice of the appropriate formula depends on the specific case. Below are three basic ways to calculate the distance between two data points x and y with k features:

Euclidean:

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Manhanttan:

$$\sum_{i=1}^{k}|x_i - y_i|$$

Minkowski:

$$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q)\right)^{1/q}$$

Advantages:

- **Easy to use:** Simple, accurate, and suitable for beginners.
- **Quick adaptation:** Automatically updates when new data is added.
- **Few parameters:** Only requires selecting k and distance metric.

Disadvantages:

- **Scalability issues:** Requires a lot of memory and computational time.
- **Limitations with high-dimensional data:** Performs poorly when there are too many features in the data.
- **Prone to overfitting/underfitting:** Needs careful selection of k to balance between these two issues.

3.2.2 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a transformation method that reduces a large number of correlated variables into a smaller set of uncorrelated variables, where the new variables are linear combinations of the original variables.

PCA Model

Consider a dataset with k variables, which are represented by j principal components such that (j < k). The first principal component is expressed as:

$$PC_1 = a_1X_1 + a_2X_2 + a_3X_3 + \cdots a_kX_k$$

Advantages:

- Effectively reduces data dimensionality.
- Eliminates noise and reduces multicollinearity.
- Supports data visualization.

Disadvantages:

- Loses the interpretability of original features.
- Not suitable for nonlinear data.
- Sensitive to the scale of the data

3.2.3 Optimizing KNN via Grid Search

Grid Search is a traditional method used for hyperparameter tuning in machine learning. It exhaustively tries every combination of the provided hyper-parameter values to find the best model. Hyperparameter tuning can further reduce the error on the testing dataset. Grid Search allows us to find out the perfect set of hyperparameters that in turn increase the accuracy and decrease the error on our Machine Learning models.

In the k nearest neighbors model, we know the hyperparameter k = (number of nearest neighbors). KNN with values **k = 1** and **k = 5** are likely to give different outputs even though they are given the same input. This study uses the Grid Search technique for the process of finding optimal hyperparameter values in the model. Grid Search Cross Validation is a term used to refer to Grid Search and Cross-Validation techniques, namely methods for selecting combinations of models and hyperparameters by testing each combination one by one and validating each combination.

3.2.4 Addressing Imbalanced Datasets

Data imbalance is one of the common phenomena in binary classification problems, such as spam email detection, fraud detection, default prediction, and medical diagnosis. In cases where the data ratio between the two classes is 50:50, it is considered balanced. When there is a difference in distribution between the two classes, such as 60:40, the data is considered imbalanced. Balanced Accuracy is another well-known metric both in binary and in multi-class classification; it is computed starting from the confusion matrix.

$$Balanced\ Accuracy = \frac{\frac{True\ Positive}{Total_{row_1}} + \frac{True\ Negative}{Total_{row_2}}}{2}$$

Where:
True Positive is the number of instances where the model correctly predicted the positive class.
True Negative is the number of instances where the model correctly predicted the negative class.

3.2.5 Binary classification

For binary classification, let X = Rd represent a d-dimensional feature space, and Y = {+1, −1} represent the label set. Suppose p(x, y) is the unknown joint probability distribution over the random variables (x, y) ∈ X x Y. The objective of binary classification is to develop a binary classifier g: X → R that minimizes the classification risk:

$$R(g) = \mathbb{E}_{\mathrm{p(x,y)}}[\ell(g(x), y)] \quad (1)$$

In this context, $\ell(\cdot,\cdot)$ represents a non-negative binary-class loss function, such as the 0-1 loss or logistic loss. Let $\pi_+ = p(y = +1)$ and $\pi_- = p(y = -1)$ denote the prior probabilities of the positive and negative classes, respectively. Also, let $p_+(x) = p(x|y = +1)$ and $p_-(x) = p(xy = -1)$ represent the class-conditional probability densities for positive and negative classes. The classification risk from Eq. (1) can be rewritten as:

$$R(g) = \pi + \mathbb{E}_{\mathrm{p+(x)}}[\ell(g(x), +1) + \pi_-\mathbb{E}_{p-(x)}[\ell(g(x), -1)]$$

3.2.6 Multiclass classification

Multiclass Classification plays a crucial role in Intrusion Detection Systems (IDS) by identifying and categorizing different types of cyberattacks. Instead of merely distinguishing between normal and malicious traffic, multiclass classification enables IDS to specifically recognize attack types, such as Denial of Service (DoS) attacks, Remote-to-Local (R2L) attacks, Probe attacks, and many others.

Intrusion Detection Systems (IDS) operate through a systematic process: first, they collect network traffic and system behavior data, extracting key features like packet counts, IP addresses, and protocol details. This data is then preprocessed—cleaned, normalized, and transformed—to suit machine learning models. Using algorithms such as Decision Trees, SVM, Neural Networks, or KNN, models are trained on labeled datasets where each label represents either normal behavior or a specific attack type. When new traffic is detected, the model classifies it as normal or malicious, identifying the exact attack type if applicable. Finally, IDS responds by alerting administrators or executing automatic preventive measures.

The evaluation metrics used are listed below:

1. Accuracy

Accuracy is one of the most popular metrics in multi-class classification and it is directly computed from the confusion matrix.

Accuracy provides a comprehensive measure of a model's correctness across the entire dataset. It considers each individual instance equally, with all units contributing the same weight to the overall accuracy score. When shifting focus from individual instances to classes, discrepancies in class sizes become apparent - some classes may have a large number of instances, while others may have very few. In such cases, classes with more instances will disproportionately influence the accuracy score compared to smaller classes.

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

2. Recall (Detection Rate or TPR)

Recall is the ratio of correctly predicted positive instances (True Positives) to the total number of actual positive instances in the dataset. It is calculated by dividing the number of True Positives by the total count of actual positives (the row sum of the positive class). Instances that are incorrectly labeled as negative despite being positive are referred to as False Negatives. Recall reflects the model's ability to accurately identify all positive instances in the dataset, providing a measure of its effectiveness in detecting the positive class.

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

3. Precision

Precision represents the proportion of correctly predicted positive instances out of all instances predicted as positive. It is calculated by dividing the number of True Positive - cases where the model accurately identifies positive instances - by the total number of positive predictions, which includes both True Positives and False Positives. False Positives refer to instances incorrectly classified as positive when they are actually negative.

Precision, therefore, reflects how trustworthy the model's predictions are when it identifies a case as positive, providing a measure of reliability for positive classifications.

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

4. F1-Score

The F1-Score evaluates the performance of a classification model based on the confusion matrix, combining Precision and Recall into a single metric using their harmonic mean.

$$F1\text{-}Score = 2 * \frac{Recall * Precision}{Recall + Precision}$$

5. AUC (Area Under the Curve)

AUC measures the performance of a classification model by calculating the area under the ROC curve, which plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various thresholds. AUC values range from 0 to 1, with higher values indicating better model performance in distinguishing between classes. An AUC of 1 means perfect classification, while 0.5 indicates a random model.

$$AUC = \int_0^1 TPR(FPR) d_{FPR}$$

6. Score time (µs): Score time refers to the time a model takes to make predictions on a given dataset. It is an important metric for evaluating the efficiency of a model, particularly in real-time applications or when dealing with large datasets. Lower score times indicate faster, more efficient models.

Where:

- TP: True Positive
- TN: True Negative
- FP: False Positive
- FN: False Negative

3.2.7 Model Evaluation Metrics

In this study, the model's performance is evaluated based on two main metrics: Accuracy and Training and Prediction Time. Accuracy is calculated for both the training and test sets, providing insight into the model's ability to learn from the training data and its generalization capability on unseen data. The training and prediction time is measured as a combined metric, reflecting the time required by the model to train and make predictions on new data. These metrics are used to assess the impact of improvements, including the application of PCA, hyperparameter optimization (via Grid Search), and data balancing techniques.

4. EXPERIMENTAL RESULTS

4.1 Binary Classification Experiment

4.1.1 Before Applying PCA Feature Extraction

The table below shows the performance summary of the models before applying PCA Feature extraction:

Model	Train Score	Test Score	Time
KNN	0.995941	0.994419	1266.98
Decision Tree	0.999998	0.999266	10.5316
XGBM	0.99963	0.999355	9.62155
light GBM	0.999275	0.999142	14.2116
CatBoost	0.998859	0.998808	161.362
Naive Baye Model	0.945459	0.945478	0.643348

Figure 2. The model parameters before applying the PCA technique in Binary Classification

Note:

- The first column represents the names of the models.
- The Train Score column shows the accuracy of the training set.
- The Test Score column shows the accuracy of the test set.
- The Time column shows the running time of the models.

The KNN model (Figure 2) has the longest running time, so the experiment will focus on improving the execution time of the KNN model by applying PCA feature extraction.

4.1.2 After applying PCA Feature extraction

After applying the PCA feature extraction to the KNN model, the experimental results. It is evident that the model's training time has significantly decreased from 1266.98 seconds to 99.6244 seconds while the accuracy remains the same.

4.1.3 Optimizing the KNN model by using grid search to find the hyperparameters

Subsequently, by using grid search to find the hyperparameters for optimizing the KNN model,

The model's training time has been significantly reduced from 99.6244 seconds to 30.038 seconds and the accuracy has also increased slightly.

Figure 3. A visualization chart comparing the running time of the KNN model across each experiment in Binary Classification.

Based on the performance comparison chart of three KNN methods (regular KNN, KNN with PCA, and optimized KNN with PCA) evaluated by execution time, training score, and test score, the results show that through the improvement process, the authors have enhanced performance, significantly reducing the execution time of the KNN model across three stages of optimization (Figure 3).

4.2 Multiclass Classification Experiment

4.2.1 Training model with normal features

nts, the accuracy of the train and test sets is quite low, only above 71% for the train set and above 58% for the test set. This could be due to imbalanced data.

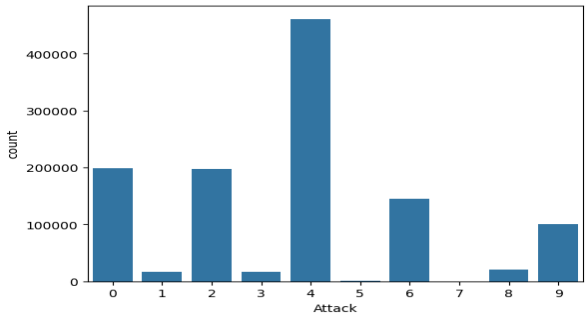

Figure 4. Data before balancing

The classes [1, 3, 5, 7, 8, 9] have fewer samples, so we need to resample these classes to address the class imbalance issue (Figure 4).

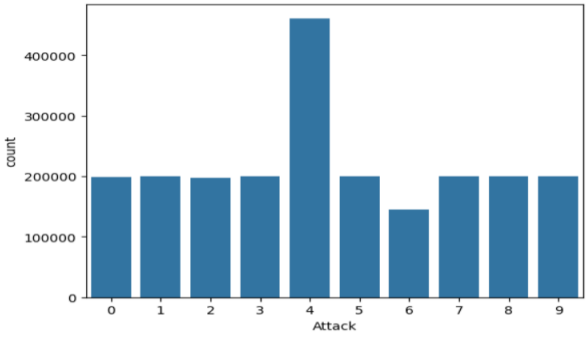

Figure 5. Data after balancing

Next, after training the KNN model with balanced data (Figure 5). It can be seen that the accuracy of the test set has increased significantly from 58% to 74%, and the train set accuracy has increased from 71% to 81%.

4.2.2 Apply PCA Feature Extraction

Next, PCA was applied to the KNN model with balanced data. The results showed a significant reduction in running time, from 296.1355 seconds to 173.3013 seconds.

4.2.3 Optimizing the KNN Model by Using Grid Search to Find The Hyperparameters

To further improve the model's performance, the authors' team attempted to optimize the model using the optimal hyperparameters found in the Binary experiment, and the results that it is observed that the accuracy of the training set increases while the accuracy of the test set decreases, indicating overfitting. This suggests that the hyperparameters identified in the Binary Classification experiment are not suitable for the Multiclass Classification experiment. Therefore, new hyperparameters need to be determined and the KNN model optimized. The results are as follows (Figure 6 and Figure 7):

```
[ ] print("Optimized hyper-parameters:", grid.best_params_)
    grid_train, grid_test = grid.score(x_train, y_train), grid.score(x_test, y_test)

    grid_elapsed_time = time.time() - start_time
    print("Elapsed time:", grid_elapsed_time, "seconds")
    print(f"Train Score: {grid_train}")
    print(f"Test Score: {grid_test}")

Optimized hyper-parameters: {'metric': 'manhattan', 'n_neighbors': 5, 'weights': 'uniform'}
Elapsed time: 1737.7637481689453 seconds
Train Score: 0.8167164434766642
Test Score: 0.7442579418368227
```

Figure 6. Find new hyperparameters for the Multiclass Classification experiment

```
start_time = time.time()

KNN_model = KNeighborsClassifier(metric='manhattan', n_neighbors=5, weights='uniform')
KNN_model.fit(x_train, y_train)

grid_train, grid_test = KNN_model.score(x_train, y_train), KNN_model.score(x_test, y_test)

grid_elapsed_time = time.time() - start_time
print("Elapsed time:", grid_elapsed_time, "seconds")
print(f"Train Score: {grid_train}")
print(f"Test Score: {grid_test}")

Elapsed time: 183.36654686927795 seconds
Train Score: 0.8167164434766642
Test Score: 0.7442579418368227
```

Figure 7. Performance of the Multiclass Classification experiment after optimization

4.2.4 Summary

Below is a summary of the results in tables and charts to visualize and easily track the performance of the KNN model at each stage of the Multiclass Classification experiment (Figure 8):

Model	Train Score	Test Score	Time
KNN Imbalanced dataset	0.71267	0.584331	145.725
KNN	0.816084	0.744646	296.136
KNN with PCA	0.816039	0.744264	173.301
Optimized KNN with PCA (Binary)	0.857453	0.707852	53.4531
Optimized KNN with PCA	0.816716	0.744258	183.367

Figure 8. KNN model performance across experiments

It can be seen that the accuracy of the KNN model before balancing the data was quite low, with just over 58% on the test set and 71% on the training set. After applying various data balancing methods, PCA, and optimization, the accuracy of the models improved significantly, with over 74% and 81% on the test and training sets, respectively. Moreover, the time taken also decreased Substantially with the balanced dataset, specifically 296 seconds before applying PCA and 183 seconds after optimization.

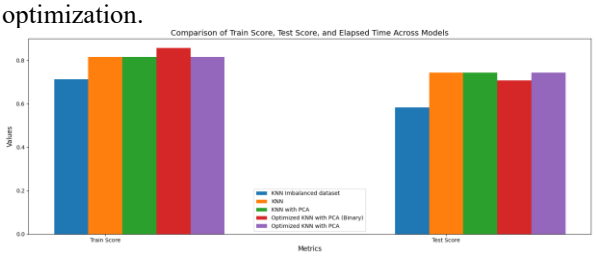

Figure 9. Visualization chart comparing the accuracy of the KNN model across experiments

Figure 9 visualizing the accuracy of the training and test sets across different stages. The red accuracy column will not be used for comparison with the other columns, as it represents the optimized performance with hyperparameters found in the Binary Classification Experiment, which experienced overfitting.

Figure 10. Visualization chart comparing the runtime of the KNN model across experiments in Multiclass Classification

Figure 9 and 10 illustrate a comparison of the runtime of KNN models in multiclass classification experiments across three cases (standard KNN, KNN with PCA, and optimized KNN with PCA). The results show that standard KNN has the highest runtime. After applying improvements with KNN and PCA, a significant reduction in runtime is evident in the chart. Finally, the optimized KNN with PCA demonstrates the lowest runtime, reflecting the efficiency gained from optimization and dimensionality reduction.

5. CONCLUSION

In summary, after researching and implementing the project "Designing a Model for Detecting Intrusions in Internal Networks Based on Network Traffic," the authors successfully improved the runtime performance of the KNN (K-Nearest Neighbors) model based on previous studies. The improvement process started with the initial KNN model, followed by the application of PCA (Principal Component Analysis) to reduce the data dimensions, thereby minimizing complexity and accelerating the training process. Ultimately, optimizing KNN with PCA significantly reduced the model's training time without significantly affecting its accuracy. Experimental results demonstrated that through each improvement phase (initial KNN, KNN with PCA, and Optimized KNN with PCA), the model's runtime was noticeably reduced. This enhancement improves the efficiency of intrusion detection in internal networks while maintaining the model's accuracy and reliability. These advancements contribute substantially to practical applications in intrusion detection systems by reducing system load and enhancing network security and monitoring capabilities.

In the context of the rapidly expanding digital economy, intrusion detection mechanisms represent not only a technical security solution but also an important element of modern financial governance. The efficiency of network traffic monitoring and anomaly detection contributes to strengthening the resilience of digital financial infrastructures that are increasingly exposed to cross-border cyber threats. Improved detection capability reduces operational uncertainty within financial institutions and supports the stability of electronic payment ecosystems, which are essential for the development of digital commerce and investment activities.

From a broader policy perspective, the development of efficient intrusion detection models indirectly supports economic competitiveness by reducing cybersecurity risk exposure in digital financial operations. As investment decisions in technology-intensive sectors increasingly incorporate cybersecurity governance indicators, reliable network protection mechanisms help improve the perceived safety of the business environment, particularly for multinational enterprises operating data-dependent services.

References:

Agarwal, N., Srivastava, R., Srivastava, P., Sandhu, J., & Singh, P. P. (2022, May). Multiclass classification of different glass types using random forest classifier. In *2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS)* (pp. 1682-1689). IEEE.

Assegie, T. A. (2021). An optimized K-Nearest Neighbor based breast cancer detection. *Journal of Robotics and Control (JRC)*, *2*(3), 115-118.

Bace, R.G., & Mell, P. (2001). Intrusion detection systems. *NIST Special Publication on Intrusion Detection Systems*.

Boyko, N., & Muzyka, M. (2021, October). Analysis of Multimodal Data for Classification Problems by Using Methods of Machine Learning. In *2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T)* (pp. 525-534). IEEE.

Bulajoul, W., James, A., & Pannu, M. (2013, September). Network intrusion detection systems in high-speed traffic in computer networks. In *2013 IEEE 10th International Conference on e-Business Engineering* (pp. 168-175). IEEE.

Cao, K., Wu, J., Huang, Q., & Gan, Y. (2023, August). Optimization Study of KNN Classification Algorithm on Large-Scale Datasets: Real-Time Optimization Strategy Based on Balanced KD Tree and Multi-threaded Parallel Computing. In *2023 4th International Conference on Intelligent Computing and Human-Computer Interaction (ICHCI)* (pp. 423-427). IEEE.

Chawla, N. V. (2010). Data mining for imbalanced datasets: An overview. *Data mining and knowledge discovery handbook*, 875-886.

Cui, Y., & Dy, J. G. (2008, July). Orthogonal principal feature selection. In *the Sparse Optimization and Variable Selection Workshop at the International Conference on Machine Learning (ICML)*

Dadhania, S. S., & Dhobi, J. S. (2012). Improved kNN Algorithm by Optimizing Cross-validation. *International Journal of Engineering Research and Technology*, *1*(3), 1-6.

Ghawi, R., & Pfeffer, J. (2019). Efficient hyperparameter tuning with grid search for text categorization using kNN approach with BM25 similarity. *Open Computer Science*, *9*(1), 160-180.

Guarracino, M. R., Cifarelli, C., Seref, O., & Pardalos, P. M. (2007). A classification method based on generalized eigenvalue problems. *Optimisation Methods and Software*, *22*(1), 73-81.

Guo, G., Wang, H., Bell, D., Bi, Y., & Greer, K. (2003, November). KNN model-based approach in classification. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"* (pp. 986-996). Berlin, Heidelberg: Springer Berlin Heidelberg

Hoens, T. R., & Chawla, N. V. (2013). Imbalanced datasets: from sampling to classifiers. *Imbalanced learning: Foundations, algorithms, and applications*, 43-59.

Japa, A., Brown, D., & Shi, Y. (2019, February). Towards optimizing data analysis for multi-dimensional data sets. In *Future of Information and Communication Conference* (pp. 614-625). Cham: Springer International Publishing.

Kotha, N. R. (2017). Intrusion Detection Systems (IDS): Advancements, Challenges, and Future Directions. *International Scientific Journal of Contemporary Research in Engineering Science and Management*, *2*(1), 21-40.

Kumari, R., & Srivastava, S. K. (2017). Machine learning: A review on binary classification. *International Journal of Computer Applications*, *160*(7).

Kusuma, S. T., & Sasongko, T. B. (2023). Optimasi K-Nearest Neighbor dengan Grid Search CV pada Prediksi Kanker Paru-Paru. *The Indonesian Journal of Computer Science*, *12*(4).

Luo, Y., Xiong, S., & Wang, S. (2008, September). A PCA based unsupervised feature selection algorithm. In *2008 Second International Conference on Genetic and Evolutionary Computing* (pp. 299-302). IEEE.

Mavikumbure, H. S., Cobilean, V., Wickramasinghe, C. S., Varghese, B. J., Carlson, R. B., Rieger, C., ... & Manic, M. (2024). Cy-Phy ADS: cyber-physical anomaly detection framework for EV charging systems. *IEEE Transactions on Transportation Electrification*, *10*(4), 9904-9917.

Mell, R. B. A. P. (2001). Intrusion detection systems. *National Institute of Standards and Technology (NIST), Special Publication*, *51*.

Mulak, P., & Talhar, N. (2015). Analysis of distance measures using k-nearest neighbor algorithm on kdd dataset. *Int. J. Sci. Res*, *4*(7), 2319-7064.

Murali, M. (2015). Principal component analysis based feature vector extraction. *Indian Journal of Science and Technology*, *8*(35), 1.

Pandey, A., & Jain, A. (2017). Comparative analysis of KNN algorithm using various normalization techniques. *International Journal of Computer Network and Information Security*, *10*(11), 36.

Peterson, M. R., Doom, T. E., & Raymer, M. L. (2005, June). GA-facilitated classifier optimization with varying similarity measures. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation* (pp. 1549-1550).

Phetlasy, S., Ohzahata, S., Wu, C., & Kato, T. (2015, December). Sequential Combination of Two Classifier Algorithms for Binary Classification to Improve the Accuracy. In *2015 Third International Symposium on Computing and Networking (CANDAR)* (pp. 576-580). IEEE.

Prasad, S. N. S. E., Srinath, M. V., & Basha, M. S. (2015). Intrusion detection systems, tools and techniques–an overview. *Indian journal of science and technology*, *8*(35), 1-7.

Quemy, A. (2019). Binary classification in unstructured space with hypergraph case-based reasoning. *Information Systems*, *85*, 92-113.

Ramyachitra, D., & Manikandan, P. (2014). Imbalanced dataset classification and solutions: a review. *International Journal of Computing and Business Research (IJCBR)*, *5*(4), 1-29.

Reid, S. R. (2010). *Model combination in multiclass classification*. University of Colorado at Boulder.

Shrivastava, M. U. (2017). A Review of Intrusion Detection System. *International Journal on Recent and Innovation Trends in Computing and Communication*, *5*(5), 556-558.

Sonar, R., & Deshmukh, P. (2014). Multiclass classification: A review. *Int. J. Comput. Sci. Mob. Comput*, *3*(4), 65-69.

Song, F., Guo, Z., & Mei, D. (2010, November). Feature selection using principal component analysis. In *2010 international conference on system science, engineering design and manufacturing informatization* (Vol. 1, pp. 27-30). IEEE.

Sukamto, S., Hadiyanto, H., & Kurnianingsih, K. (2023). KNN optimization using grid search algorithm for preeclampsia imbalance class. In *E3S Web of Conferences* (Vol. 448, p. 02057). EDP Sciences.

Takenouchi, T., & Ishii, S. (2009). A multiclass classification method based on decoding of binary classifiers. *Neural computation*, *21*(7), 2049-2081.

Anh Huynh Van
HCMC University of Technology and Education,
Vietnam

Duyen Ngo Ky
HCMC University of Technology and Education,
Vietnam

Nhi Nguyen Thi Yen
HCMC University of Technology and Education,
Vietnam

Vy Nguyen Thi Thuy
HCMC University of Technology and Education,
Vietnam

Phan-Anh-Huy Nguyen
HCMC University of Technology and Education,
Vietnam
huynpa@hcmute.edu.vn
ORCID : 0000-0002-8955-1436