COMPUTOLOGY: Journal of Applied Computer Science and Intelligent Technologies
Volume 4, Issue 1, Jan-Jun 2024, pp. 89-108
DOI: 10.17492/computology.v4i1.2406
https://www.journalpressindia.com/cjacsit

Development of an Intrusion Detection System using ANOVA Feature Selection and Support Vector Machine Algorithms

Michael F. Edafeajiroke, Sulaiman O. Abdulsalam**,*
*Mahmoud U. Shuaib*** and Ronke S. Babatunde*****

ABSTRACT

The escalating sophistication and frequency of cyber-attacks have necessitated the development of more advanced intrusion detection systems (IDS). This research presents the development of an innovative IDS employing Analysis of Variance (ANOVA) for feature selection and a Support Vector Machine (SVM) algorithm for intrusion detection. The aim is to enhance detection accuracy while reducing computational overhead. ANOVA was used to identify significant features from vast and complex network traffic data, simplifying the high-dimensional data and improving the detection system's efficiency. The selected features are then classified using the Radial Basis Function (RBF) SVM algorithm, renowned for its high accuracy and robustness in handling high-dimensional data. A comparative analysis with existing IDS models demonstrates the improved efficiency and accuracy of the proposed model. This work provides an advanced methodology for cyber security, contributing to the ever-evolving battle against cyber threats.

Keywords: *Intrusion detection; Machine learning; Feature selection; Support vector machine; ANOVA.*

1.0 Introduction

The constant evolution of cyber threats and the increasing complexity of network systems have posed significant challenges in maintaining secure digital environments (Chatterjee *et al.*, 2023).

**Corresponding author; Faculty of Computing, Department of Computer Science, University of Port Harcourt, Nigeria (E-mail: edafemichaelfavour@gmail.com)*
***Lecturer, Department of Computer Science, Kwara State University, Malete, Nigeria*
(E-mail: sulaiman.abdulsalam@kwasu.edu.ng)
****Department of Computer Science, Kwara State University, Malete, Nigeria*
(E-mail: mahmoudusmanshuaib@gmail.com)
*****Lecturer, Department of Computer Science, Kwara State University, Malete, Nigeria*
(E-mail: ronke.babatunde@kwasu.edu.ng)

Traditional security measures such as firewalls and antivirus software are no longer sufficient in detecting and preventing these threats due to their reactive nature (Alagrash *et al.* 2023). Hence, Intrusion Detection Systems (IDS) have become a crucial component of cybersecurity infrastructure to proactively identify and respond to potential threats (Faizin *et al.* 2024).

Traditional IDS primarily focus on signature-based detection methods which are limited in detecting new or unknown threats. Additionally, they often require substantial computational resources due to the high dimensionality of network data. These shortcomings underscore the need for advanced IDS with better threat detection capabilities and more efficient computational performances (Zeinalpour & McElroy, 2024). Recently, machine learning-based IDS have gained considerable attention for their potential to detect unknown threats by learning from the patterns and anomalies in network traffic.

In particular, the Support Vector Machine (SVM) algorithm has demonstrated high detection accuracy and robustness in handling high-dimensional data. However, the SVM algorithm can also be computationally intensive, especially when dealing with large and complex datasets (Abdulsalam *et al.* 2024). Utilizing a better kernel could yield better performance (Musthafa *et al.*, 2024). Hence this paper ANOVA stands out for its ability to determine the significance of each feature, allowing for a more focused and less computationally intensive analysis (Megantara & Ahmad, 2021). One approach to alleviate this issue is by implementing feature selection techniques to reduce the dimensionality of the dataset. Among various feature selection methods is the Analysis of Variance.

Studies revealed that current IDS approaches, such as signature-based and anomaly-based methods, have proven effective against known and novel threats respectively, but they often suffer from limitations. Signature-based IDS struggle with the detection of zero-day attacks, while anomaly-based IDS are plagued by high false positive rates (Zeinalpour & McElroy, 2024).

Moreover, the use of machine learning algorithms in IDS, while promising, faces challenges in terms of dealing with high-dimensional data and the computational expense of training complex models. Feature selection methods are often used to address these challenges, but the optimal feature selection technique for IDS remains unclear (Abdulsalam *et al.* 2024). Despite the proven individual effectiveness of the ANOVA feature selection method and the RBF-SVM kernel, limited research has explored their combination in the context of IDS. The RBF kernel's flexibility, robustness, strong generalization, and ability to handle complex datasets (Musthafa *et al.*, 2024), justify its selection in this study.

1.1 Aim and objectives

- This study aims to investigate the effectiveness of integrating ANOVA for feature selection and RBF-SVM Kernel for classification in the context of an anomaly-based IDS.

The specific objectives are to:

- Perform feature selection from the acquired dataset using ANOVA filter techniques
- Employ the optimal feature subset to build the RBF-SVM model for intrusion Detection
- Evaluate the developed detection system using various performance matrixes such as Accuracy, Precision, Recall, and F1 score.
- Compare the developed intrusion system with other existing system

2.0 Review of Literature

Faizin *et al.* (2024) observed that Information and communication technology is rapidly growing, making it a target for attacks such as data theft, phishing, and Denial of Service (DoS). To combat these threats, Intrusion Detection Systems (IDS) are developed, with recent research focusing on feature selection and addressing data imbalance. This study proposes an IDS that combines mutual information for feature selection with the XGBoost classification algorithm. Mutual information assesses feature dependency, while thresholding determines the optimal number of features for classification. Tested on the UNSW-NB15, NSL-KDD, and CIC-IDS2017 datasets, the proposed method achieved the highest performance on CIC-IDS2017, with 99.89% accuracy and a 99.68% F1 score, also reducing computational training time compared to other methods.

Zeinalpour & McElroy (2024) observed that DDoS attacks have grown in frequency and sophistication over the past decade, making it crucial to analyze large volumes of data. Metaheuristic algorithms can help select relevant features for DDoS detection models. However, finding an optimized solution remains an open research question. This study uses a switching approximation to find the best solution for network traffic feature analysis, but finds it not significantly better than the BestFirst algorithm.

Shakeela *et al.* (2020) design an intrusion detection system using ensemble learning, specifically Decision Trees with distinctive feature selection. The technique is tested on the NSL KDD network dataset, and its performance measures like accuracy, precision, F-score, and Cross Validation curve are used to justify its ability. This approach can help reduce threats to user data and network infrastructure However, researchers suggested other feature engineering methods and classification algorithms for an enhanced performance.

Megantara & Ahmad (2021) The experiment demonstrates that using the "distance" value to measure feature relevance and setting a threshold based on the mean score effectively isolates relevant features, significantly enhancing accuracy (up to 88%) and reducing computational time. This method, applied during pre-processing, improves the performance of decision trees and random forest classifiers and is adaptable to various systems. Future research may further improve accuracy through data reduction, optimization, and adaptability to different environments hence this study.

Ogundokun *et al.* (2021) this work aims to address the increasing challenges of cybersecurity by developing an effective intrusion detection system (IDS) using artificial intelligence (AI) and machine learning (ML) techniques. The study employs Particle Swarm Optimization (PSO) for dimensionality reduction and compares two classification techniques: PSO + Decision Tree (PSO+DT) and PSO + K-Nearest Neighbor (PSO+KNN). The KDD-CUP 99 dataset was used to verify the effectiveness of these techniques. Results show that PSO+KNN outperformed PSO+DT with an accuracy 96.2%, False negative rate of 0.011, and false-positive rate (FPR) of 0.004. Future research could focus on enhancing these models further and testing them on more diverse and contemporary datasets.

Stiawan *et al.* (2021) focused on improving the performance of intrusion detection systems by identifying the most relevant features. Six feature selection methods are used, including Information Gain, Gain Ratio, Symmetrical Uncertainty, Relief-F, One-R, and Chi-Square. These techniques are combined with four classification methods to generate ensemble IDSs. Experimental results show that the optimized ensemble IDSs achieve 81.0316%, 85.2593%, and 80.8625% accuracy, respectively. The ensemble IDSs using SU and BN and OR and J48 with the best selected features also perform the best F-measure value.

Mohammadi *et al.* (2021) explore the use of Support Vector Machines (SVMs) in intrusion detection systems (IDSs) to combat increasing security attacks. It provides a comprehensive study of SVM-based IDS schemes, their contributions, algorithms, techniques, and properties, while also discussing the limitations and limitations of these systems. Sarumi *et al.* (2020) observed that the digital revolution has increased the use of the Internet for information storage and dissemination, leading to concerns about information theft, privacy, and confidentiality. Network intrusion detection systems are a viable approach to combat these threats. This paper compares two intrusion detection systems, Apriori and Support Vector Machine, using the Network Security Laboratory Knowledge Discovery and Data Mining dataset and the University of New South Wales–NB 2015 dataset. SVM outperforms Apriori in accuracy and testing speed.

2.1 ANOVA feature analysis

ANOVA is a statistical technique used to determine whether there are significant differences between the means of two or more groups. In the context of feature selection, ANOVA can be used to evaluate the importance of a feature by comparing the means of the feature values when grouped by the target variable, which in an IDS would be whether the network traffic is an intrusion or not (Ogundokun *et al.*, 2021). If the ANOVA test results in a large F-value for a feature, this suggests that the means of the feature's values are significantly different between normal traffic and intrusion, indicating that the feature may be important in distinguishing between the two and should be included in the model. Conversely, a small F-value suggests that the feature may not be relevant (Stiawan *et al.* 2021).

Applying ANOVA for feature selection in the development of an IDS can help simplify the model and reduce computational cost by eliminating irrelevant features. However, it's important to take note that ANOVA makes certain assumptions about the data, including normal distribution and equal variances across groups, which may not always be met in real-world data. Additionally, ANOVA is a univariate method and does not account for interactions between features (Megantara & Ahmad, 2021).

Despite these limitations, ANOVA is still a valuable tool for IDS development when paired with suitable machine-learning techniques. It is crucial to select optimal feature subsets to enhance RBF-SVM performance in intrusion detection.

2.2 Classification using support vector machine

Vladimir Vapnik developed the support vector machine (SVM), a popular supervised learning model that can be applied to both regression and data classification. However, it is frequently utilized to build a hyperplane when the distance between two classes of data points is at its highest in classification issues. The classes of data points on either side of the decision boundary are divided by a hyperplane called the decision boundary (e.g., oranges vs. apples). Support-vector machines (SVMs), also known as support-vector networks) in machine learning are supervised learning models with associated learning algorithms that examine data for classification and regression analysis (Vapnik, 1998). SVM have been widely used in most classification and regression task with studies revealing competitive performance with other learning techniques such as KNN and nave-bayes. An algorithm for SVM is shown in Algorithm 1.

Algorithm 1: SVM Algorithm (Megantara & Ahmad, 2021)

Pseudo code for Support Vector Machine (SVM) (Megantara & Ahmd, 2021)
Step 1: Import the dataset
Step 2: Explore the data to figure out what they look like
Step 3: Pre-process the data
Step 4: Split the data into attributes and labels
Step 5: Divide the data into training and testing sets
Step 6: Train the SVM algorithm
Step 7: Make some predictions
Step 8: Evaluate the results of the algorithm

3.0 Research Methodology

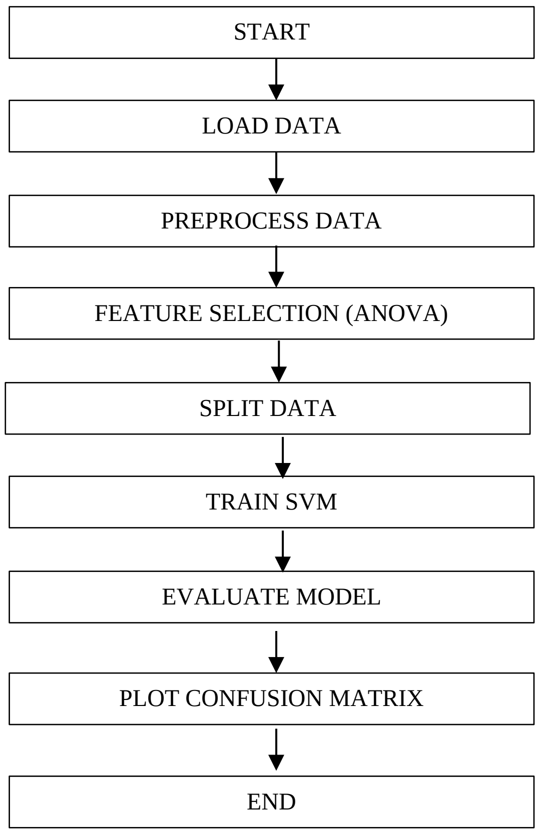
3.1 Research design

This study aims to build an Intrusion Detection System using ANOVA and RBF-SVM techniques. This section outlines the methodologies used for identifying anomalies with the CICIDS2017 dataset. It ensures clarity and scientific rigor in achieving the study's objectives by covering research design, feature analysis, machine learning application, system design and implementation, testing, deployment, and evaluation. Understanding these methods is crucial for appreciating the alignment between the research question, hypothesis, and analysis, thus ensuring reproducibility and credibility. The subsections will detail each component, including data collection, preprocessing, feature selection, model training, evaluation, and user interface development. The machine learning design approach was the methodology adopted in the study. The model was carefully designed to actualize very high detection of intrusion on the network. Figure 1 therefore provide a framework for the developed intrusion detection system.

3.2 Data identification and collection

The dataset that was adopted in this study was collected from the Kaggle online repository. The CICIDS2017 dataset is a comprehensive dataset for Intrusion Detection Systems (IDS) developed by the Canadian Institute for Cybersecurity (CIC).

Figure 1: Frame Work for the Intrusion Detection Model using ANOVA and RBF-SVM



It contains a wide range of features and attributes, representing both normal and malicious network traffic, making it a valuable resource for researchers and developers working on network security. The dataset includes various attack scenarios such as Brute Force, DoS, DDoS, Web attacks, and more. It has been widely used for testing and training machine learning models related to cybersecurity. The dataset spanned over eight different files containing five days of normal and attack traffic data from the Canadian Institute of Cybersecurity. The dataset originally consisted of 284,315 instances and 79 features. However, after excluding 65,408 instances with missing class labels and 201 instances with incomplete data, the final dataset was reduced to 218,706 instances. A short description of all those files is presented in Table 1 and Table 2.

Table 1: Description of Files Containing CICIDS2017 Dataset

Name of File	Day Activity	Attacks found
MondayWorkingHours.pcap_ISCX.csv	Monday	Benign (Normal human activities)
WorkingHours.pcap_ISCX.csv	Tuesday	Benign, FTP-Patator,SSH-Patator
WednesdayworkingHours.pcap_ISCX.csv	Wednesday	Benign, DoS GoldenEye, DoS Hulk, DoS Slowhttptest, DoS slowloris, Heartbleed
Thursday-WorkingHoursMorning-WebAttacks.pcap_ISCX.csv	Thursday	Benign, Web Attack _ Brute Force, Web Attack – Sql Injection, Web Attack - XSS
Thursday-WorkingHoursAfternoon-Infilteration.pcap_ISCX.csv	Thursday	Benign, Infiltration
Friday-WorkingHoursMorning.pcap_ISCX.csv	Friday	Benign, Bot
Friday-WorkingHours-AfternoonPortScan.pcap_ISCX.csv	Friday	Benign, PortScan
WorkingHours-AfternoonPortScan.pcap_ISCX.csv	Friday	Benign, DDoS

Table 2: Class wise Instance Occurrence of the CICIDs2017 Dataset

File	Label	Total Instances	Percentage of Total Instances
Fri-Morn.pcap_ISCX.csv	BENIGN	189067	98.97%
Fri-Morn.pcap_ISCX.csv	Bot	1966	1.03%
Fri-noon-DDoS.pcap_ISCX.csv	DDoS	128027	56.71%
Fri-noon-DDoS.pcap_ISCX.csv	BENIGN	97718	43.29%
Fri-noon-PortScan.pcap_ISCX.csv	PortScan	158930	55.48%
Fri-noon-PortScan.pcap_ISCX.csv	BENIGN	127537	44.52%
Mon.pcap_ISCX.csv	BENIGN	520918	100.00%
Thur-Mor-WebAttacks.pcap_ISCX.csv	BENIGN	168186	98.72%
Thur-Mor-WebAttacks.pcap_ISCX.csv	Web Attack – Brute Force	1507	0.88%
Thur-Mor-WebAttacks.pcap_ISCX.csv	Web Attack – XSS	652	0.38%
Thur-Mor-WebAttacks.pcap_ISCX.csv	Web Attack – Sql Injection	21	0.01%
Thur-noon-infiltration.pcap_ISCX.csv	BENIGN	288566	99.99%
Thur-noon-infiltration.pcap_ISCX.csv	Infiltration	36	0.01%
Tues.pcap_ISCX.csv	BENIGN	432074	96.90%
Tues.pcap_ISCX.csv	FTP- Patator	7938	1.78%
Tues.pcap_ISCX.csv	SSH – Patator	5897	1.32%
Wed.pcap_ISCX.csv	BENIGN	440031	63.52%
Wed.pcap_ISCX.csv	DoS Hulk	231073	33.36%
Wed.pcap_ISCX.csv	DoS GoldenEye	10293	1.49%
Wed.pcap_ISCX.csv	DoS slowloris	5796	0.84%
Wed.pcap_ISCX.csv	DoS Slowhttptest	5499	0.79%
Wed.pcap_ISCX.csv	Heartbleed	11	0.00%

Dos and DDos-related attacks are prevalent in the CICIDs2017 dataset, hence it is the major attack studied in this paper.

Features

The dataset contains numerous features that describe the network traffic, including:

Basic Features: Source and destination IP addresses, source and destination ports, timestamps, protocols (e.g., TCP, UDP), etc.

Flow Features: Statistical information about data flows, such as the number of packets, bytes transmitted, flow duration, etc.

Content Features: Information about the content of the data, such as the number of HTTP requests, TLS handshakes, etc.

Attack Categories: In the given dataset, there is only one specific category of attack labeled, and it is as follows:

DDoS: Distributed Denial of Service attacks. Additionally, there is a category labeled as “BENIGN,” which represents non-attack instances or benign traffic.

Benign Traffic: In addition to the malicious traffic, the dataset also includes normal (benign) network traffic, which represents legitimate network activities.

Labeling: Each instance in the dataset is labeled as either an attack type (with specific attack categories) or benign, allowing for supervised learning tasks.

Data Volume: CICIDS2017 is quite extensive, containing a significant amount of data that is representative of both modern benign traffic and various attack vectors.

3.3 Data preprocessing

Data preprocessing is a crucial step in the data mining process. It involves cleaning, transforming, and organizing raw data into a suitable and efficient format for analysis. Data preprocessing helps in enhancing the quality of data, removing noise and irrelevant information, and making the data compatible with the data mining techniques being employed.

Preprocessing Steps Carried Out in this Study:

Handling Missing Values: Missing values in the dataset can lead to misleading representations and erroneous conclusions. In this work, missing values were filled with zeros to handle any gaps in the data.

Replacing Infinity Values: Infinity values can occur due to division by zero or other arithmetic operations that lead to unbounded results. Infinity values were replaced with NaN, and subsequently, NaN values were filled with zeros.

Normalization: Normalization is the process of scaling the features to a standard range. This ensures that all features contribute equally to the computation of distances or other similarity measures. In this paper, StandardScaler from scikit-learn was used to normalize the features.

Removing Constant Features: Features that have the same value for all samples don't provide any useful information for classification. Any constant features in the dataset were identified and removed.

Encoding Categorical Variables: If the target variable or any feature is categorical (e.g., 'BENIGN', 'ATTACK'), it must be converted into a numerical format. In this paper, the target variable was encoded using a LabelEncoder from scikit-learn.

Feature Selection (ANOVA): Although not typically categorized under preprocessing, feature selection is an essential step in preparing data for modeling. In this paper, ANOVA (Analysis of Variance) was used to compute the F-value for each feature, and the top K features were selected.

3.4 Classification

Classification is a type of supervised learning where the goal is to predict the categorical class labels of new instances, based on past observations. In a classification problem, the output variable (or target) is a category, such as "yes" or "no," "spam" or "not spam." It is a two-step process, consisting of a learning step (where a model is trained on a set of labeled examples) and a prediction step (where the trained model is used to predict the class labels of unseen instances).

In this paper, classification was carried out to distinguish between different types of network traffic, identifying whether a given instance is benign or represents a specific type of attack (e.g., intrusion). Here's how the classification was done:

Data Preparation: The dataset was preprocessed to remove any irrelevant or redundant information and transform the data into a suitable format. Feature selection was performed using ANOVA to select the most relevant features.

Model Selection: A Support Vector Machine (SVM) was chosen as the classification algorithm. SVM is a powerful method for binary or multiclass classification that finds the hyperplane that best divides a dataset into classes.

Training: The SVM model was trained on a subset of the data (training set), learning from the provided features and corresponding labels. During this phase, the model tries to find the best boundary that separates the classes.

Evaluation: After training, the model was evaluated on a separate subset of the data (test set) to determine how well it generalizes to unseen instances. Performance metrics such as precision, recall, and F1-score were computed, and a confusion matrix was plotted to visualize the classification results.

Prediction: Once trained and evaluated, the model can be used to classify new, unseen instances, predicting whether they are benign or represent an attack.

3.5 Feature selection

Feature selection is the process of selecting a subset of relevant features (variables, predictors) for use in model construction. It aims to simplify models, improve performance, reduce overfitting, and minimize computation time. By removing irrelevant or redundant features, feature selection helps in building a model that generalizes better to unseen data. In this paper, feature selection was carried out using ANOVA (Analysis of Variance), which is a statistical technique used to analyze the differences among group means in a sample. Here's how the process was conducted:

ANOVA for Each Feature: The F-value for each feature was calculated using ANOVA. The F-value measures how much a variable contributes to the variation in the target variable. In the context of this paper, it helped in identifying the importance of each feature in differentiating between different types of network traffic (e.g., benign vs. attack). **Sorting Features by F-value:** The features were sorted based on their F-values in descending order. A higher F-value indicates a more significant contribution to the variation in the target variable, meaning the feature is more important for classification. **Selecting Top K Features:** The top K features with the highest F-values were selected. The parameter K is a user-defined number representing how many features to keep. By selecting only the top K features, the model focuses on the most relevant information while ignoring less important or noisy features.

3.6 Evaluation metrics

Table 3 describes the evaluation metric used in this study showing the formula for each metrics:

Table 3: Evaluation Metrics (Abdulsalam *et al.*, 2024)

Measure	Formula
Precision	$\frac{TP}{TP + FP}$
Sensitivity	$\frac{TP}{TP + FN}$
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$
Specificity	$\frac{TN}{TN + FP}$

Source: https://www.laujet.com/index.php/laujet/article/view/671

4.0 Result and Discussion

This section presents the key findings of this work, outlining the model's performance across various metrics. It also provides an in-depth discussion of the results, exploring the insights gained, the implications of the findings, and potential areas for further research and development. The insights derived from this analysis not only contribute to the field of cybersecurity but also demonstrate the vast potential of data-driven approaches in solving intricate real-world problems. Detailed description of the hardware and software requirements are provided in Table 4 and 5 respectively.

Table 4: Hard Requirement

S. No.	Minimum Requirements
1	Intel Pentium 2.0GHZ Core i3 Processor or higher
2	Minimum of 4GB RAM

Table 5: Software Requirement

S. No.	Requirements	Software
1	Development Tool	Microsoft VSCode IDE
2	Programming Language	Python
3	Machine Learning	Scikit-learn
4	Libraries and dependencies	Panda, NumPy, scikit-learn, SciPy, Matplotlib, Seaborn
5	Operating System	Window 8 or higher

Figure 2. Shows the stage at which libraries are imported and the dataset loaded into the platform.

Figure 2: Code Snippets for the Imported Libraries

```
import pandas as pd
import numpy as np
from sklearn import svm
from scipy.stats import f_oneway
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import classification_report
import warnings
from scipy import stats
```

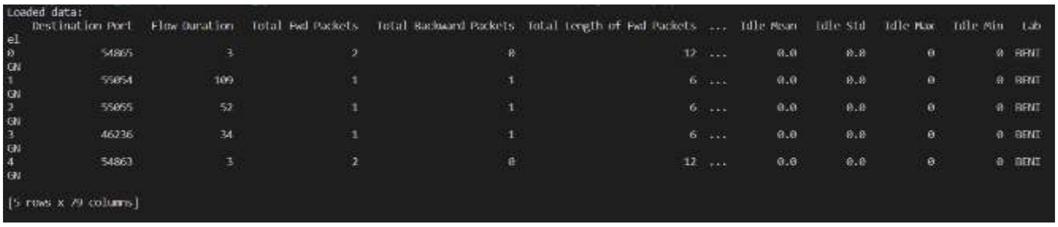
Loading of dataset are explained in Figure 3 Loading the dataset

Figure 3: Code Snippet for Loading the Dataset

```
# Load the dataset
data = pd.read_csv("CICIDS2017.csv")
```

The console showing the uploaded data is explained in Figure 4.

Figure 4: Console Output Showing the Uploaded Data



```
Loaded data:
   Destination Port  Flow Duration  Total Fwd Packets  Total Backward Packets  Total Length of Fwd Packets  ...  Idle Mean  Idle Std  Idle Max  Idle Min  Lab
el
0             54865              3                  2                       0                           12  ...        0.0       0.0         0         0  BENI
GN
1             55054            109                  1                       1                            6  ...        0.0       0.0         0         0  BENI
GN
2             55055             52                  1                       1                            6  ...        0.0       0.0         0         0  BENI
GN
3             46236             34                  1                       1                            6  ...        0.0       0.0         0         0  BENI
GN
4             54863              3                  2                       0                           12  ...        0.0       0.0         0         0  BENI
GN

[5 rows x 79 columns]
```

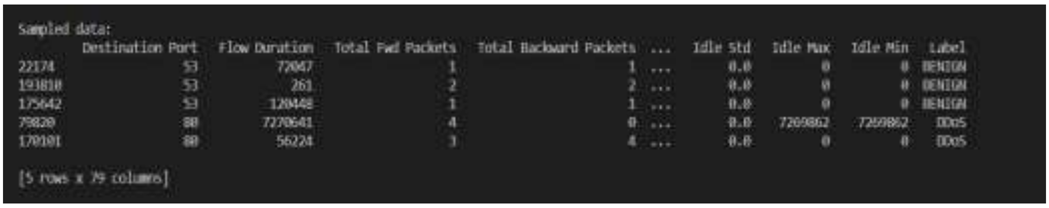
Figure 5 and Figure 6 describe Code snippets for data sampling and feature separation. A random sample of 10000 rows was taken from the dataset to make the program run faster.

Figure 5. Code snippets for data sampling and feature separation

```
# Sample the data
data = data.sample(n=10000, random_state=1)  # adjust the sample size as needed

# Separate features and target variable
X = data.drop(columns=[data.columns[-1]])  # assuming the last column is the target
y = data[data.columns[-1]]  # assuming the last column is the target
```

Figure 6. Console output showing the sampled data

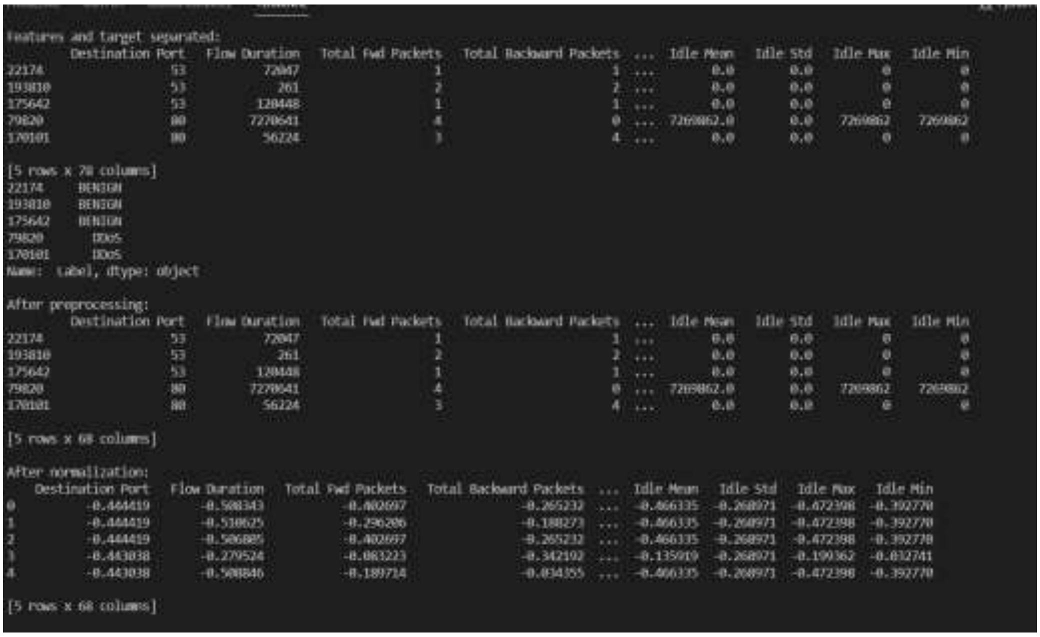


```
Sampled data:
        Destination Port  Flow Duration  Total Fwd Packets  Total Backward Packets  ...  Idle Std  Idle Max  Idle Min   Label
22174                 53          72047                  1                       1  ...       0.0         0         0  BENIGN
193810                53            261                  2                       2  ...       0.0         0         0  BENIGN
175642                53         120448                  1                       1  ...       0.0         0         0  BENIGN
79820                 80        7270641                  4                       0  ...       0.0   7269862   7269862    DDoS
170101                80          56224                  3                       4  ...       0.0         0         0    DDoS

[5 rows x 79 columns]
```

Output obtained after data was preprocessed is shown in Figure 7.

Figure 7: Console Output for the Preprocessed Data



```
Features and target separated:
        Destination Port  Flow Duration  Total Fwd Packets  Total Backward Packets  ...  Idle Mean  Idle Std  Idle Max  Idle Min
22174                 53          72047                  1                       1  ...        0.0       0.0         0         0
193810                53            261                  2                       2  ...        0.0       0.0         0         0
175642                53         120448                  1                       1  ...        0.0       0.0         0         0
79820                 80        7270641                  4                       0  ...  7269862.0       0.0   7269862   7269862
170101                80          56224                  3                       4  ...        0.0       0.0         0         0

[5 rows x 78 columns]
22174     BENIGN
193810    BENIGN
175642    BENIGN
79820       DDoS
170101      DDoS
Name:  Label, dtype: object

After preprocessing:
        Destination Port  Flow Duration  Total Fwd Packets  Total Backward Packets  ...  Idle Mean  Idle Std  Idle Max  Idle Min
22174                 53          72047                  1                       1  ...        0.0       0.0         0         0
193810                53            261                  2                       2  ...        0.0       0.0         0         0
175642                53         120448                  1                       1  ...        0.0       0.0         0         0
79820                 80        7270641                  4                       0  ...  7269862.0       0.0   7269862   7269862
170101                80          56224                  3                       4  ...        0.0       0.0         0         0

[5 rows x 68 columns]

After normalization:
   Destination Port  Flow Duration  Total Fwd Packets  Total Backward Packets  ...  Idle Mean  Idle Std  Idle Max  Idle Min
0         -0.444419      -0.508343          -0.402697               -0.265232  ... -0.466335 -0.268971 -0.472398 -0.392770
1         -0.444419      -0.510625          -0.296206               -0.188273  ... -0.466335 -0.268971 -0.472398 -0.392770
2         -0.444419      -0.506885          -0.402697               -0.265232  ... -0.466335 -0.268971 -0.472398 -0.392770
3         -0.443038      -0.279524          -0.083223               -0.342192  ... -0.135919 -0.268971 -0.199362 -0.032741
4         -0.443038      -0.508846          -0.189714               -0.034355  ... -0.466335 -0.268971 -0.472398 -0.392770

[5 rows x 68 columns]
```

The future selected after the ANOVA technique was employed for feature selection is shown in Figure 8.

Figure 8: Features Selected using the ANOVA Techniques

	Bwd Pack	Avg Bwd S	Bwd Packe	Bwd Pack	Destinatic	URG Flag	Packet Le	Average P	Packet Le	Min Packe
0	-0.6543	-0.6543	-0.69928	-0.71482	-0.44442	-0.40096	-0.76368	-0.70411	-0.8109	3.08498
1	-0.68559	-0.68559	-0.70876	-0.71482	-0.44442	-0.40096	-0.8015	-0.78292	-0.81816	1.546672
2	-0.6838	-0.6838	-0.70822	-0.71482	-0.44442	-0.40096	-0.78687	-0.73516	-0.82453	2.892692
3	-0.80091	-0.80091	-0.74368	-0.71482	-0.44304	-0.40096	-0.9171	-0.91188	-0.85997	-0.11983
4	1.791749	1.791749	1.999995	2.104123	-0.44304	-0.40096	1.665004	1.720664	1.929899	-0.5044
5	-0.80091	-0.80091	-0.74368	-0.71482	-0.44304	-0.40096	-0.9171	-0.91188	-0.85997	-0.11983
6	-0.69542	-0.69542	-0.71173	-0.71482	-0.44442	-0.40096	-0.7946	-0.74551	-0.83044	2.892692
7	-0.79316	-0.79316	-0.73826	-0.70889	1.210636	-0.40096	1.378103	1.391297	1.150317	-0.5044
8	-0.80091	-0.80091	-0.74368	-0.71482	-0.44304	-0.40096	-0.9171	-0.91188	-0.85997	-0.11983
9	-0.78929	-0.78929	-0.73826	-0.7091	2.22359	-0.40096	1.147512	1.134061	0.817949	-0.5044
10	-0.80091	-0.80091	-0.74368	-0.71482	-0.44304	-0.40096	-0.9171	-0.91236	-0.85997	-0.11983
11	-0.79554	-0.79554	-0.74205	-0.71482	2.856079	2.494036	-0.9171	-0.91268	-0.85997	-0.11983
12	-0.7276	-0.7276	-0.72148	-0.71482	-0.44442	-0.40096	-0.82113	-0.80482	-0.84402	2.379922
13	-0.80091	-0.80091	-0.74368	-0.71482	-0.44304	-0.40096	-0.9171	-0.91188	-0.85997	-0.11983

14	-0.80091	-0.80091	-0.74368	-0.71482	-0.44304	-0.40096	-0.9171	-0.91188	-0.85997	-0.11983
15	-0.80091	-0.80091	-0.74368	-0.71482	-0.44304	-0.40096	-0.9171	-0.91236	-0.85997	-0.11983
16	0.92753	0.92753	0.442043	0.441278	-0.44304	-0.40096	1.146442	1.133	0.54975	-0.5044
17	0.928424	0.928424	0.837284	0.754252	-0.44304	-0.40096	1.147512	1.134061	0.823032	-0.5044
18	-0.79554	-0.79554	-0.74205	-0.71482	1.381976	2.494036	-0.9171	-0.91268	-0.85997	-0.11983
19	-0.68291	-0.68291	-0.70794	-0.71482	-0.44442	-0.40096	-0.8015	-0.78292	-0.816	1.41848
20	-0.79554	-0.79554	-0.74205	-0.71482	2.386697	2.494036	-0.9171	-0.91268	-0.85997	-0.11983
21	-0.7133	-0.7133	-0.71715	-0.71482	-0.44442	-0.40096	-0.82148	-0.80521	-0.83238	1.674865
22	-0.78929	-0.78929	-0.73826	-0.7091	2.85342	-0.40096	1.147512	1.134061	0.63857	-0.5044
23	-0.33278	-0.33278	-0.34844	-0.34969	-0.42448	-0.40096	-0.38428	-0.41023	-0.46799	-0.5044
24	0.928424	0.928424	0.442043	0.545037	-0.44304	-0.40096	1.147512	1.134061	0.63857	-0.5044
25	-0.79554	-0.79554	-0.74205	-0.71482	2.532113	2.494036	-0.9171	-0.91268	-0.85997	-0.11983
26	1.79309	1.79309	2.395236	2.638582	-0.44304	-0.40096	1.666342	1.722029	2.365056	-0.5044
27	-0.69989	-0.69989	-0.71309	-0.71482	-0.44442	-0.40096	-0.80329	-0.78491	-0.82893	2.123538
28	1.274291	1.274291	1.999995	1.838753	-0.44304	-0.40096	1.378103	1.391297	1.777195	-0.5044

The selected feature was split into a training set (80% of the data) and a test set (20% of the data). This is as shown in the code snippet in Figure 9.

Figure 9. Code snippet for splitting data into train sets and test sets

```
# Split the dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
#Print training data
print("Training data (features):")
print(X_train.head())
print("\nTraining data (target):")
print(y_train[:5])  # Displaying the first 5 target values for training data

# Print test data
print("\nTest data (features):")
print(X_test.head())
print("\nTest data (target):")
print(y_test[:5])  # Displaying the first 5 target values for test data
```

Results obtained from the different performance metrics are as described in Figure 10 and Table 6.

Figure 10: Result Obtained from the Different Performance Evaluation Metrics

	precision	recall	f1-score	support
0	1.00	0.95	0.97	860
1	0.96	1.00	0.98	1140
accuracy			0.98	2000
macro avg	0.98	0.98	0.98	2000
weighted avg	0.98	0.98	0.98	2000

Table 6: Results Obtained from the Different Performance Evaluation Metrics

Metrics	Class 0	Class 1
Accuracy (%)	99.0	98.0
Precision (%)	98.0	96.0
Recall (%)	97.0	96.0
F1- Score (%)	95.0	93.0
Support	98.0	97.0

Values obtained from the confusion matrices are shown in Figure 11.

Figure 11: Confusion Matrix

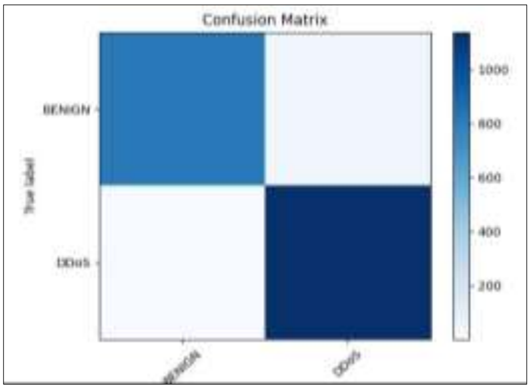


Table 7 gives a detailed comparative analysis of the support vector machine with other algorithms used in developing the IDS. The results from RBF-SVM compared with other algorithms based on the metric used in this study showed that RBF-SVM had an accuracy of 98%, Precision of 98%, recall of 100%, and F1 score of 99%.

Table 7: Comparative Analysis of RBF-SVM with Other Algorithms in IDS Development

Model	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
RBF-SVM	**98**	**98**	**1**	**99**
Logistic Regression	97	96	95	96
Decision Tree	95	93	92	93
Random Forest	98	97	98	98
Neural Network	99	99	98	99
Naïve Bayes	92	90	91	90
K-Nearest Neighbor	94	93	92	92

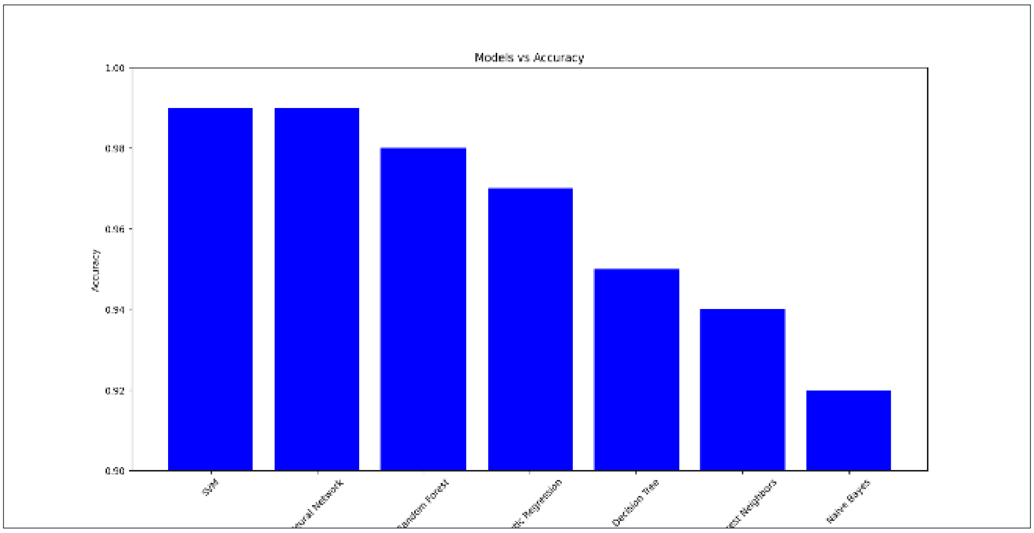
Table 8 provides a modified version including standard deviations to the result obtained in Table 7 this is to show the variability and also enhance the comparative analysis. The result showed that the proposed RBF-SVM was competitive with the Neural Networks model while both algorithms outperform other models in terms of consistency and overall performance.

Table 8: Performance Metrics with Standard Deviations for Comparative Analysis of RBF-SVM and Other Algorithms in IDS Development

Model	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
RBF-SVM	**98 ± 1**	**98 ± 2**	**1 ± 1**	**99 ± 01**
Logistic Regression	97 ± 2	96 ± 3	95 ± 2	96 ± 2
Decision Tree	95 ± 3	93 ± 3	92 ± 4	93 ± 3
Random Forest	98 ± 1	97 ± 2	98 ± 1	98 ± 1
Neural Network	99 ± 1	99 ± 1	98 ± 2	99 ± 1
Naïve Bayes	92 ± 4	90 ± 5	91 ± 4	90 ± 5
K-Nearest Neighbor	94 ± 3	93 ± 3	92 ± 4	92 ± 3

A comparative illustration of RBF-SVM with other algorithms in terms of accuracy is shown in Figure 12.

Figure 12: Comparison of SVM with Other Algorithms in Terms of Accuracy



5.0 Conclusion

The study aimed to evaluate the effectiveness of machine learning algorithms for classification tasks using a dataset (canada.csv). A Radial Bias Function- Support Vector Machine (RBF-SVM) was one of the algorithms employed for the analysis. The model was trained, tested, and preprocessed. Irrelevant features were removed, and the data was normalized. Optimal feature subsets were selected using the ANOVA technique. Models were then classified. Results obtained from different performance metrics showed that Accuracy was 97.85% and precision (Macro Avg): was 98.16%. Recall (Macro Avg): 97.51% and F1-Score (Macro Avg): 97.79%. The RBF-SVM model exhibited a high level of accuracy and well-balanced precision and recall scores. It performed admirably in comparison to other machine learning algorithms such as Random Forest, K-Nearest Neighbors, Decision Tree, and Logistic Regression.

5.1 Recommendation

Given the high accuracy and balanced precision and recall, the RBF-SVM model is recommended for deployment in real-world applications where similar classification tasks are required. For future research direction, while the model performed well, further feature engineering could improve its efficiency. Experimentation with other kernel functions and other hyperparameters specific to SVM may yield even better results. For academic rigor, the model’s performance could be compared with other advanced machine learning algorithms or ensemble methods. Implement k-fold cross-validation to better understand the model’s performance on different subsets of data. Before full-scale deployment, the model should be tested on a real-world, unseen dataset to validate its performance metrics.

References

[1] Abdulsalam S. O., Ayofe R. A., Edafeajiroke M. F., Ajao J. F. & Babatunde R. S. (2024). Development of an intrusion detection system using mayfly feature selection and artificial neural network algorithms. *LAUT ECH Journal of Engineering and Technology, 18*(2), 148. Retrieved from Doi: 16010.36108/laujet/4202.81.0241

[2] Alagrash, Y. H., Mehdy, H. S. & Mahdi, R. H. (2023). A review of intrusion detection system methods and Techniques: Past, present and future. *International Journal on Technical and Physical Problems of Engineering (IJTPE), 15*(1).

[3] Chatterjee, S., Shaw, V., & Das, R. (2023). Multi-stage intrusion detection system aided by Grey Wolf optimization algorithm. *Research Square (Research Square)*. Retrieved from https://doi.org/10.21203/rs.3.rs-2680915/v1

[4] Faizin, M. A., Kurniasari D. T., Elqolby, N., Putra, M. A. R., & Ahmad, T. (2024). Optimizing feature selection method in intrusion detection system using thresholding. (2024b). *International Journal of Intelligent Engineering and Systems, 17*(3), 214-226. Retrieved from https://doi.org/10.22266/ijies2024.0630.18

[5] Megantara, A., & Ahmad, T. (2021). ANOVA-SVM for selecting subset features in encrypted internet traffic classification. *International Journal of Intelligent Engineering and Systems, 14*(2), 536 546. Retrieved from https://doi.org/10.22266/ijies2021.0430.48

[6] Mohammadi, M., Rashid, T.A., Karim, S.H., Aldalwie, A.H.M., Tho, Q.T., Bidaki, M., Rahmani, A. M., & Hosseinzadeh, M. (2021). A comprehensive survey and taxonomy of the SVM-based intrusion detection systems. *Journal of Network and Computer Applications, 178*, 102983. Retrieved from https://doi.org/10.1016/j.jnca.2021.102983

[7] Musthafa, M. B., Huda, S., Kodera, Y., Ali, M. A., Araki, S., Mwaura, J., & Nogami, Y. (2024). Optimizing IoT intrusion detection using balanced class distribution, feature selection, and ensemble machine learning techniques. *Sensors, 24*(13), 4293. Retrieved from https://doi.org/10.3390/s24134293

[8] Ogundokun, R. O., Awotunde, J. B., Sadiku, P., Adeniyi, E. A., Abiodun, M., & Dauda, O. I. (2021). An enhanced intrusion detection system using particle swarm optimization feature extraction technique. *Procedia Computer Science, 193*, 504–512. Retrieved from https://doi.org/10.1016/j.procs.2021.10.052

[9] Sarumi, O. A., Adetunmbi, A. O., & Adetoye, F. A. (2020). Discovering computer networks intrusion using data analytics and machine intelligence. *Scientific African, 9*, e00500. Retrieved from https://doi.org/10.1016/j.sciaf.2020.e00500

[10] Shakeela, S., Shankar, N. S., Reddy, P. M., Tulasi, T. K., & Koneru, M. M. (2020). Optimal ensemble learning based on distinctive feature selection byunivariate ANOVA F statistics for IDS. *Intern ational Journal of Electronics and Telecommunications*, 267275. Retrieved from https://doi.org/10.24425/ijet.2021.135975

[11] Stiawan, D., Heryanto, A., Bardadi, A., Rini, D. P., Subroto, I. M. I., Kurniabudi, N., Idris, M. Y. B., Abdullah, A. H., Kerim, B., & Budiarto, R. (2021). An approach for optimizing ensemble intrusion detection systems. *IEEE Access, 9*, 6930–6947. Retrieved from https://doi.org/10.1109/access.2020.3046246

[12] Vapnik, V.N. (1998). *Statistical Learning Theory*. New York, USA: John Wiley & Sons.

[13] Zeinalpour, A., & McElroy, C. P. (2024). Comparing metaheuristic search techniques in addressing the effectiveness of ClusteringBased DDOS attack detection methods. *Electronics, 13*(5), 899. Retrieved from https://doi.org/10.3390/electronics13050899