AnalysingTheEffectivenessofMachine LearningAlgorithmsinIntrusionDetection SystemsforIoTNetworks

MScResearchProject
MScinCloudComputing

SandeepKumarMylavarapu
StudentID:23204346

SchoolofComputing
NationalCollegeofIreland

Supervisor: RejwanulHaque

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Sandeep Kumar Mylavarapu
Student ID:	23204346
Programme:	MSc in Cloud Computing
Year:	2025
Module:	MSc Research Project
Supervisor:	Rejwanul Haque
Submission Due Date:	29/01/2025
Project Title:	Analysing The Effectiveness of Machine Learning Algorithms in Intrusion Detection Systems for IoT Networks
Word Count:	7144
Page Count:	20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Sandeep Kumar Mylavarapu
Date:	29th January 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Analysing The Effectiveness of Machine Learning Algorithms in Intrusion Detection Systems for IoT Networks

Sandeep Kumar Mylavarapu
23204346

Abstract– Network Intrusion Systems are very essential in managing networks and mainly ensure to keep their security information safe.Conventional approaches based on rule analysis have mostly failed to provide effective solutions for complicated attacks by learning from the outcomes of IDS implementing in complex environments like cloud computing and IoT.This work examines Random Forest, Support Vector Machine and K-Nearest Neighbour ML algorithms for IDS using the NSL- Knowledge Discovery in Databases.The analysis of these algorithms which considers the classification accuracy and estimations by using the measures of precision, recall, F1 scores and random forest that results the best outcome with an overall prediction accuracy followed by KNN with 97.45 Percent and SVM with 95.54 Percent.The findings of the study that discuss about the feature selection, data preprocessing and the model evaluation as critical areas that are important to the enhanced performance of IDS.Moreover problems like lack of balance of datasets, the problems that arise due to big data and high computational complexity in real-life networks are still open to be solved.To overcome these types of stated limitations the study which effectively suggests to use oversampling methods such as SMOTE, hyperparameter tuning and the integration of deep learning to improve the process of anomaly identification.For future studies it is recommended to work with efficient real-type datasets, together with ensemble solutions and employ ML IDS in real-world systems which comprise IoT and cloud networks to enhance performance flexibility and security.

1 INTRODUCTION

1.1 Problem Statement

Most of the traditional IDS techniques are rule-based and they seem to be very less effective in handling the dynamism of the attacks because the hackers and attackers are experienced and involved in the knowledge of systems and they are dynamic in their criminal activities.Such systems may be unable to identify these new types of attacks particularly as the networks grow in terms of the sophistication of their structure such as cloud computing, IoT, and distributed systems as shown in (1).The quantity of data in the network environments demands the capacity of real-time analysis from the rule-based IDS systems that face the problem.Regarding these challenges the ML algorithms offer a more effective solution because they learn from past examples and can look for patterns of regarding known and unknown threats.However a significant problem is determining suitable ML techniques and more importantly fine-tuning them to result in a high level of intrusion detection accuracy while maintaining computational efficiency.The algorithm performance of these will also depend on the effectivity of the input information and the selection of factors that to be used along with the characteristics of the particular algorithm that is to be used with all its advantages and disadvantages as shown in (2).This dissertation intends to solve these problems by comparing the performances of several machine learning methods in the circumstance of intrusion detection by using the NSLKDD dataset as the assest that offers a reliable and structured environment for these kinds of assessments.

1.2 Aim and Objectives of the Study

This main aim is to estimate the machine learning algorithms and to improve the function of IDS utilizing the NSL-KDD dataset.The main objective is to determine the optimum learning algorithm with high level of classification and anomaly response rates in realtime applications for intrusions in computer networks. The main objectives of the study are:
1. To evaluate the performance of selected Machine Learning algorithms incorporatingRandom Forest, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN) in identifying network intrusions accurately.
2. To analyse the influence of the feature selection methods on the performance andcomputational cost of these algorithms.
3. To analyse the effectiveness of proposed methods for different types of intrusion detection applications and to distinguish the advantages or disadvantages of each algorithm. 4. To deliver the insights and recommendations on how to apply the ML-based IDS models under real-life network situations.

1.3 Research Questions

To guide this study, the following research questions will be addressed:
1. Which of these machine learning algorithms yields higher accuracy, precision, andrecall for the goals of intrusion detection?
2. How does feature selection affect the performance and time consumption of thosemachine learning algorithms?
3. What are the strengths and limitations of each in identifying different forms of intrusions and how do these conclusions support the development of IDS systems with the use of Machine Learning?

1.4 Significance of the Study

There is a significant demand for smart and adaptable intrusion detection systems in the industrial market.By addressing the performance of the machine learning algorithms in IDS and this dissertation has provided valuable findings concerning the applicability and development of network security.The particular choice of the NSL-KDD dataset used in this study can be explained by its acceptance and enhancements over the previously utilized IDS datasets which allow using it as an appropriate reference standard for similar Machine Learning algorithms with which the results of this work may be compared as shown in (3).The contribution of this study is to present a comparative assessment of various Machine Learning techniques for IDSs to enhance the knowledge of network managers,security analysis and regarding the applicability and efficiency of various algorithms. Furthermore the information obtained from this study can be helpful to enhance the design and performance of IDS systems that will provide better results with respect to true positive ratios, few false positives/negatives and therefore a potential security status for organizations.

1.5 Summary

This is the introductory chapter that gives an overview of network security and specifically about IDS in the security of digital networks.Utilization of the traditional rule-based IDS system has its drawbacks so there is a possibility of using machine learning algorithms in the IDS system for better and effective IDS.The chapter also defines and determines about the research aims, objectives and highlights the importance of the comparative study of the chosen ML algorithms Random Forest, SVM, and KNN using NSL-KDD data. It also defines the demographic parameters of this study and its general assumptions of the model before delving into detailed procedures explained throughout the following chapters.

2 RELATED WORK

2.1 Background of the Study

Now a days the development of network security is the foremost essential attributes of Information Technology infrastructure.The modern technologies being adopted globally for the organizational change and for the purpose of the unknown attackers can make use of the new systems to obtain access to organizations' systems and can cause threats to the confidentiality, integrity, and availability of useful information assets. Intrusion Detection Systems (IDS) play a major role and the tools used to identify insecure and potentially damaging actions performed in the network environment as shown in (4).However,the results of traditional IDS, which work largely with predefined rules are usually not good enough due to the constant change features of cyber threats. Consequently there has been an increasing study of how ML algorithms can be integrated into IDS to increase their ability to detect and identify intrusions in real time.

IDS systems that incorporate machine learning as a basis for analysis provide a more adaptive approach to the problems since the approach identifies patterns and outliers within traffic data by itself. There are multiple number of approaches for training an Machine Learning algorithm to identify known attack patterns and the other approach can identify other attack patterns without the use of rules as shown in (5).Regarding IDS applications the employment of the Machine learning has confirmed the considerable potential since it makes it possible to consider the detailed information regarding the network activities and to identify various kinds of intrusions that includes new ones.The objective of this dissertation is to analyse the execution of various classifiers specifically in the domain of Intrusion Detection by using the NSL-KDD dataset is known as standard dataset benchmark in this area of study.

2.2 Machine Learning Algorithms in Intrusion Detection Systems

ML has brought a refreshing wave in IDS as compared to the traditional rule-based systems. Conventional IDS works under the set patterns of the system and cannot recognize unknown patterns and complex attack patterns which makes them unsuitable for the detection of zero-day threats as shown in (6). Futhermore the ML-based IDS can learn from the past records as regarding the patterns and irregularities that characterize an attack, and it is thus more effective than the rule-based IDS in detecting known and unknown attacks.Usually when Compared to the classical algorithms these modern technologies of Machine Learning adjust their flexibly for the changing patterns of the network configuration and are ideal for modern-day complex applications like Cloud Computing, the Internet of Things, and Distributed systems.

Some of the advantages that ML has been able to bring to IDS are the capability of the system to work well with a massive number of data sets that may involve real-time traffic flows and for the reduction of both the false positive and false negative rates. Random forests and SVM are good for small data sets and on otherside the unsupervised techniques are suitable for large data that are unlabelled and highlight outliers as shown in (7). There are some issues which are associated with the implementation of ML in IDS. Algorithms usually depend's on labeled datasets and are potentially high and the scalability is a critical issue in large networks.It has issues related to overfitting and require quick updates to address new threats are the following important challenges.The use of ML for IDS can be considered promising for developing efficient and adaptive methods to improve network security against constantly appearing threats.

2.3 Overview of Selected ML Algorithms for IDS

i.Random Forest

This is a form of a combined learning technique that builds several trees at the preparation phase and the final decision of the algorithm concerning classification problems is the voting results. The large size of the datasets enables the use of intricate models and in IDS applications and it is appreciated for its mechanism against overfitting and for its capability to manage large data sets in contrast to single decision trees as show in (8).The related IDS studies have illustrated that RF has a high accuracy capability to classify the various intrusion types. However, it can be problematic during fitting with high dimensional data and becomes computationally intensive for large samples. However due to its interpretability and good performance, it is widely used in IDS implementations.

ii.Support Vector Machine (SVM)

This is a method of a supervised learning algorithm that finds the maximum margin hyperplane between the classes of data. Because of its flexibility of linear and non-linear classification, it is suitable for IDS, especially for detecting both familiar and unfamiliar attacks. Based on other IDS studies, SVM has had high precision as well as recall as shown in (9). Though it is worst when it comes to computational cost and dimensionality and therefore cannot be used for large datasets with many features.Another drawback is that its kernels and parameters need to be selected properly, which is an extra difficulty when it comes practical use of IDSs.

iii.K-Nearest Neighbors (KNN)

This is a laid-down instance-based learning algorithm that can be used to classify a data set based on the nearest of a point to the rest of the points. It is straightforward to implement and efficient for small and balanced datasets making it suitable for IDS uses. KNN has reasonable performance in the detection of network intrusions especially for compact databases (10). However it suffers from shortcomings in handling either high dimensionality or imbalanced data and the low accuracy.It is having high time complexity in case of increasing database size because during classification for every instance distances are calculated. However the KNN remains a simple approach for IDS tasks.

2.4 NSL-KDD Dataset: A Standard for IDS Research

NSL-KDD is an improved form of the KDD'99 dataset which is usually known for testing Intrusion Detection System. This approach solves some problems such as the fact that the current records are sometimes full of redundant and duplicate data, which will make the great collection of IDS more balanced and appropriate for IDS research as shown in (11). The dataset includes normal and attack traffic which are classified into four types: It includes Denial of Service (DoS), Probe, Remote to Local (R2L) and User to Root (U2R) providing a rich environment for assessing IDS models of various classes based on different types of attacks. Evaluation criteria involving accuracy, precision, recall, and F1 score offer a systematic platform for analysing IDS models. However,the use of the given dataset is limited because the environment is rather simple and does not consider such threats as Advanced Persistent Threats. User-generated updates are required for the services to keep up with the dynamic nature of the network protocols as well as the IoT settings.

2.5 Real-World Application and Scalability of ML-Based IDS

Applying ML-based IDS to current network environments such as IoT and cloud computing also poses numerous problems due to the nature of these systems: distributed and constantly changing. This means IoT devices that are connected to a certain network are susceptible to many advances as shown in (12) he pointed out that for WSNs, there is a need to achieve a high detection rate and low false alert. Similarly, the cloud computing environment requires capabilities for real-time analysis of large data streams and proposed for big data challenge of next-generation networks a decentralized MultiAgent Reinforcement Learning (MARL) IDS as shown in (13).The scalability aspect also presents a major challenge; while popular algorithms like the SVM perform perfectly well within a relatively small data set, the problem of high algorithm complexity arises particularly when working with large data sets or data that are in high dimensions as shown in (14).

The practical application of IDS based on ML has left positive impressions where the networks are dynamic and distributed. Another study combined big data streaming with cloud computing to perform rapid preprocessing and real-time detection, and it registered 97.44 percent accuracy. Similarly, another used a Convolutional Neural

Network based model to enhance the security of Wi-Fi-based Wireless Sensor Network reaching an accuracy of 97 percentage.Such improvements stress the applicability of ML-based IDS in terms of scalability and the constant development in cyberspace threats that the concept faces and calls for further progress.

2.6 Literature Gap

This paper aims to fill two major research gaps in IDS research by assessing the effectiveness of three different ML algorithms; Random Forest, SVM, and KNN by using NSL-KDD dataset. To the best of authors' knowledge, no prior work has offered a systematic comparison of these algorithms in terms of classification accuracy, computational cost, and scalability when trained under practical limitations as shown in (15). Moreover, another key aspect for which little research has been conducted so far is how feature selection affects both the quality of the resulting models and their computational complexity. Such information is lacking in the literature, and this study shall endeavor to offer a comprehensive guide to effective IDS provisions.

3 RESEARCH METHODOLOGY

3.1 Research Philosophy

Employing the positivism research philosophy in this study focuses on objective reality, and empirical and measurable data which is suitable for assessing the machine learning (ML) algorithms for IDS. Positivism provides methodological support for a structured approach ensuring that the research can be restricted to measurable objects, such as Metrics of algorithms in the field of ML when analyzed on the NSL-KDD dataset as shown in (16).Referring to operationalization of research concerns and aiming at quantifiable results, this philosophy guarantees the impartiality and reproducibility of the work. It also helps to contrast the analyzed algorithms and make evidence-based conclusions about the efficiency of the intrusion detection based solely on the algorithm decision-making. The positivism approach retained the rigidity and assumed an objective posture guaranteeing that the results will assist in enhancing IDS practicable implementations.

3.2 Research Approach

In the present study by using the logical research approach that involved the development of theoretical propositions and hypotheses with regard to the performance of ML algorithms for IDS. Based on the NSL-KDD dataset, the study proposed these hypotheses whereby the performance of such algorithms as Random forest, SVM, and KNN has been assessed and compared against well-established benchmarks including accuracy rate, precision, and recall as shown in (17). It enabled a broad check of whether the existing theories regarding the appropriateness and inappropriateness of these algorithms correlate with the findings. This paper has incorporated deductive reasoning in structuring the research process as it gave a clear direction and a strong foundation in the validation of the hypothesis. This made the findings reasonable and closely connected to the research targets which provided practical suggestions to enhance IDS models.

3.3 Research Design

The choice of descriptive research design in this study made it possible to assess the performance of ML algorithms in IDS using the NSL-KDD dataset.This design aimed more at demonstrating and evaluating the characteristics of the algorithm's which are based on given criteria like accuracy, precision, and recall by using the algorithms which includes Random Forest, SVM, and KNN. The scope of the research was to provide an understanding of the character, applicability, strength, and weakness of these algorithms with the systematic approach to documenting characteristics collected during the study as shown in (18).With the help of a descriptive approach clear information was received on how feature selection, computational point of view, and algorithm relevancy influenced those different intrusions that developed a great theoretical basis for making practical suggestions and further research in the network security area.

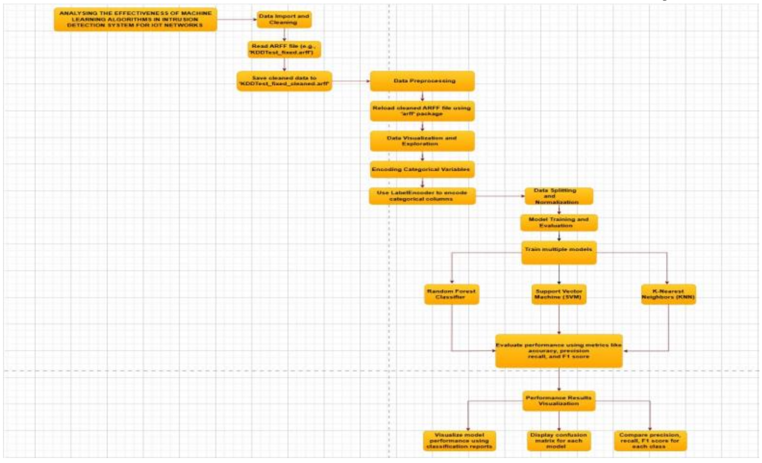

ANALYSING THE EFFECTIVENESS OF MACHINE LEARNING ALGORITHMS IN INTRUSION DETECTION SYSTEM FOR IOT NETWORKS
Data Import and Cleaning
Read ARFF file (e.g., 'KDDTest_fixed.arff')
Save cleaned data to 'KDDTest_fixed_cleaned.arff'
Data Preprocessing
Reload cleaned ARFF file using 'arff' package
Data Visualization and Exploration
Encoding Categorical Variables
Use LabelEncoder to encode categorical columns
Data Splitting and Normalization
Model Training and Evaluation
Train multiple models
Random Forest Classifier
Support Vector Machine (SVM)
K-Nearest Neighbors (KNN)
Evaluate performance using metrics like accuracy, precision recall, and F1 score
Performance Results Visualization
Visualize model performance using classification reports
Display confusion matrix for each model
Compare precision, recall, F1 score for each class

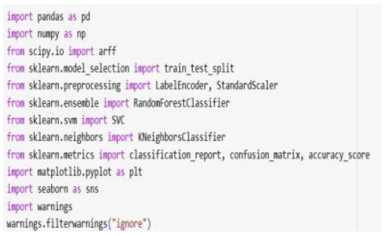
Figure 1: Design Architecture

3.4 Data Collection Method

In this research a secondary data collection technique was employed which is the NSLKDD dataset was obtained from the Kaggle website.Which is a renowned destination for all types of data sets and for solving the troubles related to Data Science . The dataset has been selected due to its LoF (Local Outlier Factor) and applicability for IDS research where it is often used as a benchmark for ML algorithms. Secondary data collection is advantageous since it will provide already structured and cleaned data which will reduce the time and costs to be spent gathering primary data as shown in (19). The present study used this dataset to assess the Random Forest, SVM, and KNN algorithms for intrusion detection ML models. The use of Kaggle prevented the act of collecting data from different sources taking a lot of time and hence the major problem of collecting data and processing it is sidestepped so that huge concentration is given to the evaluation of the algorithm and enhancement of IDS.

3.5 Data analysis techniques

Here Python is used as the main programming tool of this study due to the possibility of numerous Libraries that are personalized for machine learning and data analysis.



```
import pandas as pd
import numpy as np
from scipy.io import arff
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

Matplotlib and Seaborn were used for data visualization and Pandas and NumPy for data cleaning and feature selection as well as for preparing data for machine learning using Scikit-learn for Random Forest, SVM, and KNN models as shown in (20).Python libraries such as Matplotlib and Seaborn also helps in giving an understandable view which includes accuracy, precision and recall. When Python is used the performance was significantly enhanced which means it becomes easy to implement and able to test all the algorithms. I adjusted it with robust libraries that could deliver accurate and replicable results hence suitable for analysing the NSL-KDD dataset in IDS.

3.6 Ethical Consideration

This paper is concerned with ethical issues of using secondary data and I have complied with Kaggle policies regarding the use of the NSL-KDD data set.The data was analyzed only to meet the academic goals and objectives of the study while the confidentiality and anonymity of data in the dataset has been observed as shown in (21).It was equally important to achieve a high level of transparency, accuracy and objectivity of the research conducted. The discovery of this study wants to enhance the technological developments and the improvements in network security without compromising the integrity of the outcomes or exploiting the results inappropriately in handling research data.

3.7 Scope and Limitations of the Study

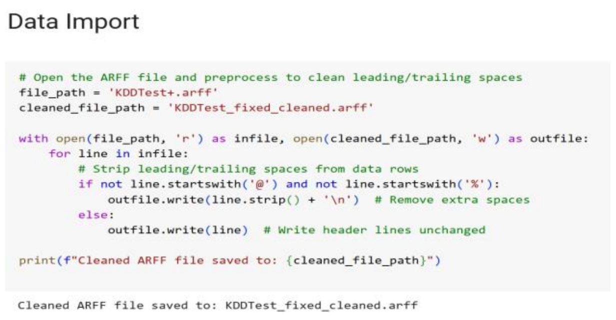
The research of this dissertation aim is to compare machine learning algorithms for network intrusion detection through the NSL-KD dataset. Furthermore it can be noted that this dataset is probably the best which suited effectively for the IDS research purpose.Although it should be marked out that any outcomes that obtained here may not be directly shared over to any other network where especially the traffic rules or specific types of threats or attacks are very different from the NSL-KDD set (Logeswari et al., 2023)x.There will be a primary focus on the estimation of the classification accuracy of the three algorithms Random Forest, SVM, and KNN which vary in their capabilities and limitations.

The other disadvantage of this is the machine learning models we have used for the detection of intrusions that need to be updated regularly to respond to new threats and for the tactics. Some of the algorithms which used in Machine learning like the SVM also pose some challenges in terms of computation and hence undermining their scalability in large networks as shown in (22). Nonetheless based on the previously mentioned limitations this study has the following objectives: To give a preliminary investigation on the performance of various ML algorithms in the IDS field and contribute to the development of the best method for future work in Network Security Systems.

4 RESULTS AND DISCUSSION

4.1 RESULTS

4.1.1 Data import



Data Import

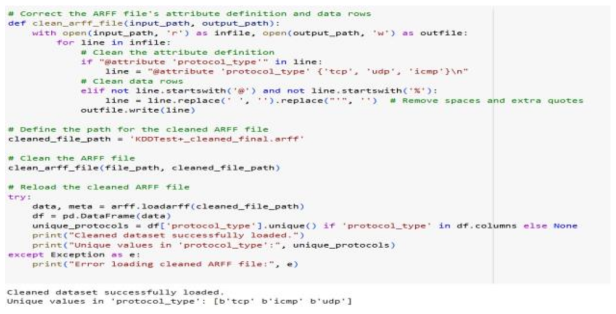
```
# Open the ARFF file and preprocess to clean leading/trailing spaces
file_path = 'KDDTest+.arff'
cleaned_file_path = 'KDDTest_fixed_cleaned.arff'

with open(file_path, 'r') as infile, open(cleaned_file_path, 'w') as outfile:
    for line in infile:
        # Strip leading/trailing spaces from data rows
        if not line.startswith('@') and not line.startswith('%'):
            outfile.write(line.strip() + '\n')  # Remove extra spaces
        else:
            outfile.write(line)  # Write header lines unchanged

print(f"Cleaned ARFF file saved to: {cleaned_file_path}")
```

```
Cleaned ARFF file saved to: KDDTest_fixed_cleaned.arff
```

Figure 2: Visualisation of the file import

The above code snip shows the cleaning process of the ARFF file by stripping out leading and trailing spaces from data lines but not from headers.Moves the record where the program reads the input file 'KDDTest-fixed.arff" and then erases all unnecessary value writing the record to the cleaned output file .Any line that starts with an '@' or percentage symbol is treated as a header line and copies to the file without modification and all other lines are cleaned of any trailing or leading white space and will be saved.The activity helps to check the data for any type of entry issues that might affect subsequent analysis and receives an acknowledgement message regarding the creation of the cleaned file.

4.1.2 Data Preparation



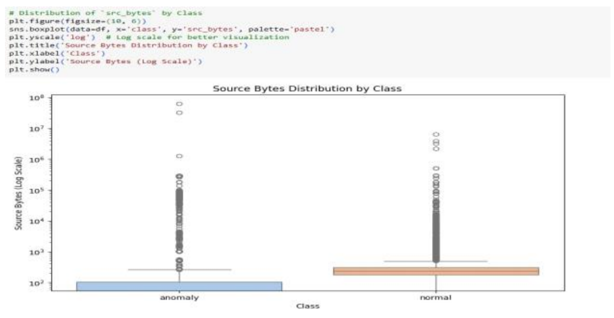
```
# Correct the ARFF file's attribute definition and data rows
def clean_arff_file(input_path, output_path):
    with open(input_path, 'r') as infile, open(output_path, 'w') as outfile:
        for line in infile:
            # Clean the attribute definition
            if "@attribute 'protocol_type'" in line:
                line = "@attribute 'protocol_type' {'tcp', 'udp', 'icmp'}\n"
            # Clean data rows
            elif not line.startswith('@') and not line.startswith('%'):
                line = line.replace(' ', '').replace("'", '')  # Remove spaces and extra quotes
            outfile.write(line)

# Define the path for the cleaned ARFF file
cleaned_file_path = 'KDDTest+_cleaned_final.arff'

# Clean the ARFF file
clean_arff_file(file_path, cleaned_file_path)

# Reload the cleaned ARFF file
try:
    data, meta = arff.loadarff(cleaned_file_path)
    df = pd.DataFrame(data)
    unique_protocols = df['protocol_type'].unique() if 'protocol_type' in df.columns else None
    print("Cleaned dataset successfully loaded.")
    print("Unique values in 'protocol_type':", unique_protocols)
except Exception as e:
    print("Error loading cleaned ARFF file:", e)
```

```
Cleaned dataset successfully loaded.
Unique values in 'protocol_type': [b'tcp' b'icmp' b'udp']
```

Figure 3: Cleaning of the imported data

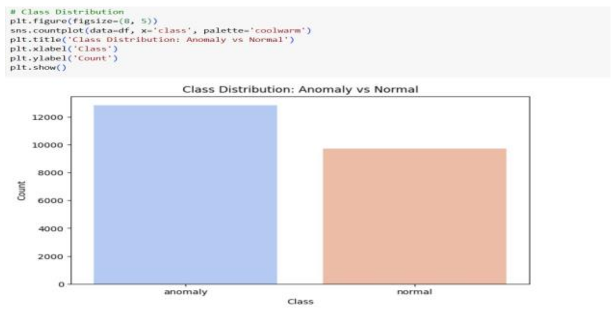
The above mentioned figure shows a process of data cleaning and data reloading in a data set in Attribute Relation File Format.Hence the ‘clean-arff-file‘ function is used to read an input ARFF file with its attribute definitions and data rows, modify the definition of the ’protocol type’ hence eliminating spaces or quotes.The cleaned file is saved in a new path.Following this the same ARFF file is read again after cleaning using the ‘arff‘ package then converting it to Pandas DataFrame and then the distinct values in the dataframe in the ‘protocol-type’ column are retrieved and printed.This guarantees its format is appropriate for other analyses or for being fed to a machine learning algorithm as shown in (23).

4.1.3 Class Distribution[Anomaly vs Normal]



```
# Distribution of 'src_bytes' by Class
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='class', y='src_bytes', palette='pastel')
plt.yscale('log')  # Log scale for better visualization
plt.title('Source Bytes Distribution by Class')
plt.xlabel('Class')
plt.ylabel('Source Bytes (Log Scale)')
plt.show()
```

Figure 4: Visulization of Source Bytes Distribution by Class



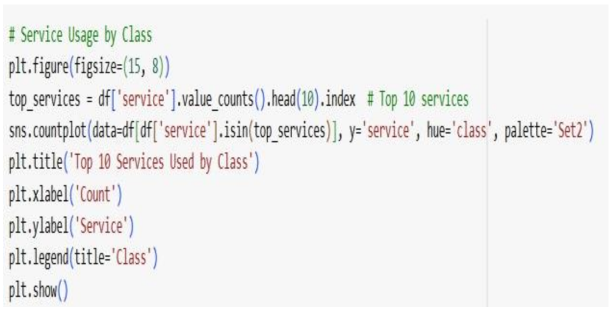
```
# Class Distribution
plt.figure(figsize=(8, 5))
sns.countplot(data=df, x='class', palette='coolwarm')
plt.title('Class Distribution: Anomaly vs Normal')
plt.xlabel('Class')
plt.ylabel('Count')
plt.show()
```

Figure 5: Visulization of the anomaly and Normal count in the data set

The above shown bar chart illustrates the class distribution of a dataset which is comparing the counts of two categories: ”anomaly” and ”normal”.The bar which is in blue colour is the ‘anomaly class’ with a count of more than 12,000, and the orange bar represents ‘normal class’ with count more than 10,000 only.This visualization raises a question which is related to the data distribution that suggests the number of ‘anomaly’ observations is greater than ‘normal’ observations.The label placed at the bottom of the chart are adequately worded as Class on the x-axis and Count on the y-axis.

4.1.4 Service usage by Class

The below mentioned figure is a horizontal bar chart that shows the results of the top 10 services that are sorted by substantive "Anomaly" and "Normal" classes. The entire bar defines the number of count of the certain service with 'Anomaly' in teal color and 'Normal' in orange color. A rather surprising fact is the distribution of the "http" service as this parameter is highest in the "normal" class and its count is above 6,000. Comparing to the other services such as "ftp data" and "telnet", wherein imbalance is observed more alternately, "private" and "domain-u" are significantly observed with "anomaly".This visualization also enables patterns in service usage to be shown for purposes of determining if the usage of particular services is related to the experience of anomalous behavior or if the services' normal functioning is anomalous.



```python
# Service Usage by Class
plt.figure(figsize=(15, 8))
top_services = df['service'].value_counts().head(10).index  # Top 10 services
sns.countplot(data=df[df['service'].isin(top_services)], y='service', hue='class', palette='Set2')
plt.title('Top 10 Services Used by Class')
plt.xlabel('Count')
plt.ylabel('Service')
plt.legend(title='Class')
plt.show()
```

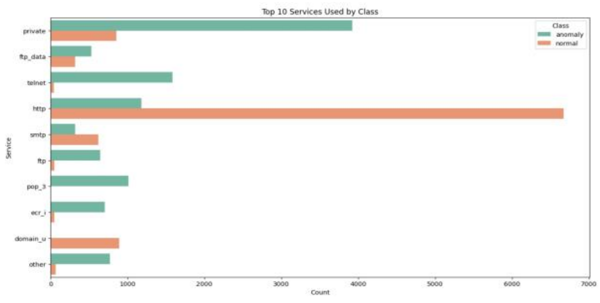
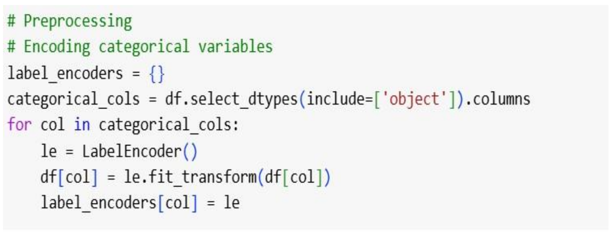


Figure 6: Visualization of the 10 services used by "class"

4.1.5 Data Preprocessing before Model Implementation

The below piece of code shows how the categorical variables in a given dataset can be preprocessed.On feature scaling it has a LabelEncoder which codes all variables with the object data type. All these columns are converted into Numeric Labels and the obtained label encoders are stored in a dictionary called label encoders for any re-mapping or further use.

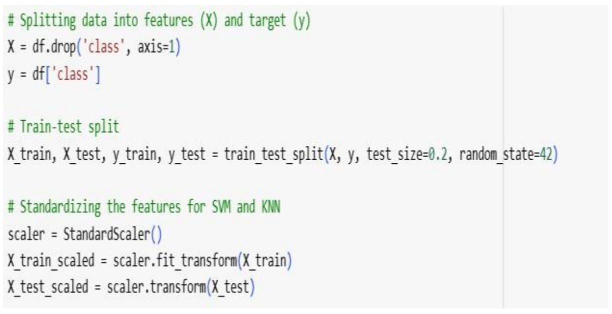


```
# Preprocessing
# Encoding categorical variables
label_encoders = {}
categorical_cols = df.select_dtypes(include=['object']).columns
for col in categorical_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le
```

Figure 7: Visualization of the encoding for categorical values

4.1.6 Data Split and Standardizing the features for SVM and KNN

The code shown above is an example of data pre-processing where the data is ready for machine learning. Firstly the feature variables are extracted by eliminating the ‘class’ column as the target variable and the feature dataset is named ‘X‘ and the target variable as ‘y‘.Then it separates the data using the function‘train- test-split’ into a training dataset and testing dataset with the testing size being 0.2 of the overall data size with added feature of reliability with ‘random-state’ of 42. Then the features are normalized using StandardScaler so as to standardize the training data as well as the testing data wherefeature originally transforming it to have zero mean and unit variance as given below as shown in (24).This basic step is pivotal for models such as the SVM and KNN models that run better when dealing with standardized data that greatly enhance model fitting and convergence.

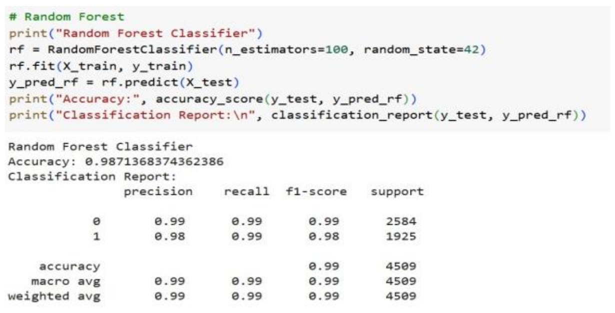


```
# Splitting data into features (X) and target (y)
X = df.drop('class', axis=1)
y = df['class']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardizing the features for SVM and KNN
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Figure 8: Visualization of Data Split and Standardizing the features for SVM and KNN

4.1.7 Random Forest Model Implementation with classification report

The snippet describes how it works to utilize and assess the Random Forest Classifier. The model is fitted from ‘X-train‘ and ‘y-train‘ datasets and the model predicts using ‘Xtest‘.It is also realized that the achieved accuracy is 98.71 percent according to the classification report.Precision, recall, F1-score for class 0 and class 1 is mentioned below:zero class is slightly higher in precision and appreciability of 0.99 than the 1 class at 0.98 remarkable. All the metrics at macro and weighted averages have values very close to one which are 0.99 for each of the metrics showing that the model is highly performing and the results are statistically balanced.

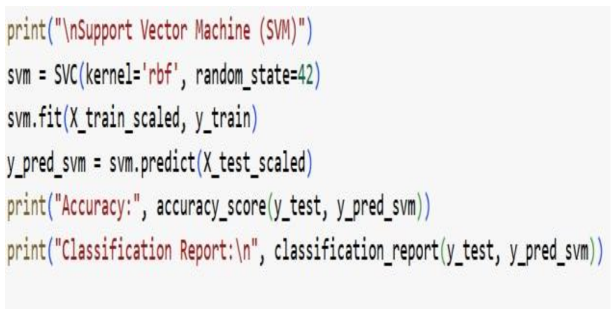


```
# Random Forest
print("Random Forest Classifier")
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred_rf))
print("Classification Report:\n", classification_report(y_test, y_pred_rf))
```

```
Random Forest Classifier
Accuracy: 0.9871368374362386
Classification Report:
               precision    recall  f1-score   support

           0       0.99      0.99      0.99      2584
           1       0.98      0.99      0.98      1925

    accuracy                           0.99      4509
   macro avg       0.99      0.99      0.99      4509
weighted avg       0.99      0.99      0.99      4509
```

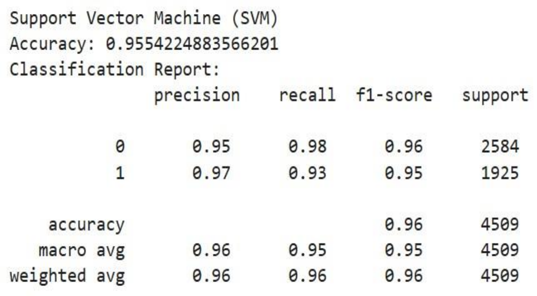
Figure 9: Visualization of the Random Forest Accuracy with Classification Report

4.1.8 SVM Model Implementation with classification report



```
print("\nSupport Vector Machine (SVM)")
svm = SVC(kernel='rbf', random_state=42)
svm.fit(X_train_scaled, y_train)
y_pred_svm = svm.predict(X_test_scaled)
print("Accuracy:", accuracy_score(y_test, y_pred_svm))
print("Classification Report:\n", classification_report(y_test, y_pred_svm))
```

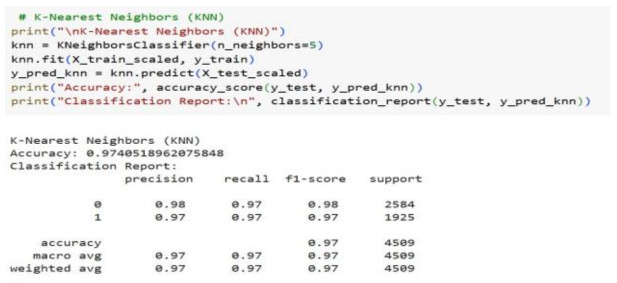
Figure 10: Visualization of the SVM Model



```
Support Vector Machine (SVM)
Accuracy: 0.9554224883566201
Classification Report:
               precision    recall  f1-score   support

           0       0.95      0.98      0.96      2584
           1       0.97      0.93      0.95      1925

    accuracy                           0.96      4509
   macro avg       0.96      0.95      0.95      4509
weighted avg       0.96      0.96      0.96      4509
```

Figure 11: Visualization of the SVM Model accuracy with classification report

It shows how to train a Support Vector Machine (SVM) classifier based on the RBF kernel and how we can evaluate it. The used training dataset is a scaled training dataset while testing is done on the other hand using the scaled test dataset.The classification report shows that it attains 95.54 percent for test accuracy. Precision, recall, and F1 score for Class 0 are 0.95 and for Class 1 are 0.95 and 0.97 respectively. It means that the macro and weighted averages are identical at 0.96 which confirms the better performance of the model.



```
 # K-Nearest Neighbors (KNN)
print("\nK-Nearest Neighbors (KNN)")
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train_scaled, y_train)
y_pred_knn = knn.predict(X_test_scaled)
print("Accuracy:", accuracy_score(y_test, y_pred_knn))
print("Classification Report:\n", classification_report(y_test, y_pred_knn))
```

```
K-Nearest Neighbors (KNN)
Accuracy: 0.9740518962075848
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.97      0.98      2584
           1       0.97      0.97      0.97      1925

    accuracy                           0.97      4509
   macro avg       0.97      0.97      0.97      4509
weighted avg       0.97      0.97      0.97      4509
```

Figure 12: Visualization of the KNN Model accuracy with classification report

4.1.9 KNN Model Implementation with classification report

They were able to describe and assess an selection of a K-Nearest Neighbors (KNN) classifier where the parameter for the number of neighbors was set to 5.The training is done by using the scaled training data while the test is performed using the scaled test data The classifier reaches the accuracy of 97.45 percentage with the help of the classification report.Data of Class 0 includes precision, recall, and F1-score of 0.98 while Class 1 has the corresponding values of 0.97.The macro mean and weighted mean for precision, recall, and F1 score are all at 0.97 and is suggestive of well balanced and high performance of the model for both classes.

4.1.10 CONFUSION MATRIX

i.Confusion Matrix of Random Forest The above image shows the Confusion Matrix

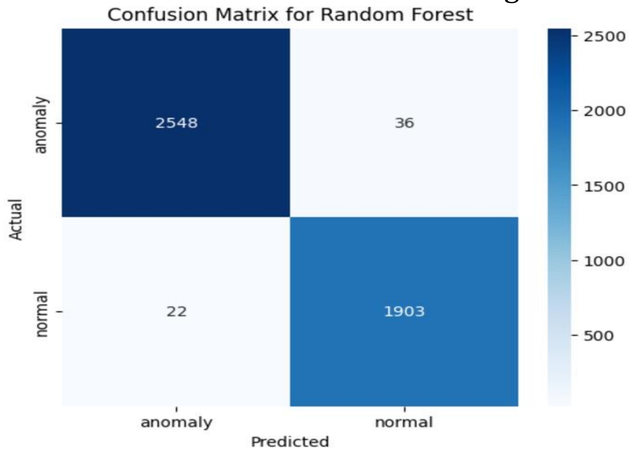


Figure 13: Visualization of Random Forest Confusion Matrix

of a Random Forest classification model.It visualizes the performance of the model that differentiates between two classes: ”Anomaly” and ”Normal.”
This confusion matrix that usually indicates strong performance as the majority of predictions are correct.However there are minor misclassifications with a small number of false positives and false negatives.
Confusion Matrix for Random Forest:

True Positives (anomaly predicted as anomaly): 2548
False Negatives (anomaly predicted as normal): 36
True Negatives (normal predicted as normal): 1903 False
Positives (normal predicted as anomaly): 22

ii.Confusion Matrix of SVM

The above figure shows the Confusion Matrix of a SVM classification model.It visualizes

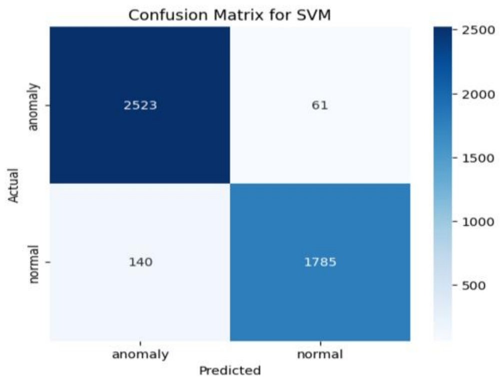


Figure 14: Visualization of SVM Confusion Matrix

the performance of the model that differentiates between two classes: ”Anomaly” and ”Normal.”
Confusion Matrix for SVM:
True Positives (anomaly predicted as anomaly): 2523
False Negatives (anomaly predicted as normal): 61
True Negatives (normal predicted as normal): 1785
False Positives (normal predicted as anomaly): 140

iii.Confusion Matrix of KNN

The below image shows a Confusion Matrix for a K-Nearest Neighbors (KNN) classific-

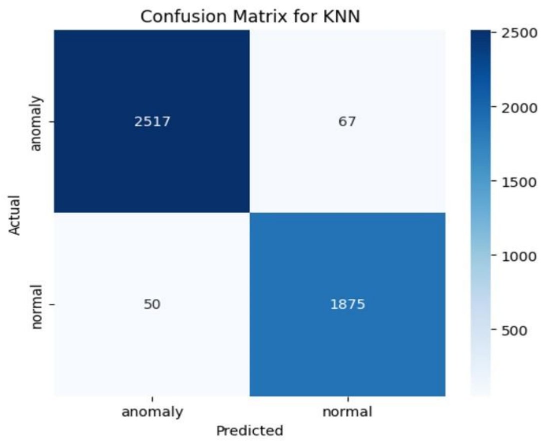


Figure 15: Visualization of KNN Confusion Matrix

ation model. It visualizes the performance of the model in distinguishing between two classes: ”anomaly” and ”normal.”
True Positives (anomaly predicted as anomaly): 2517 cases.
False Negatives (anomaly predicted as normal): 67 cases.
True Negatives (normal predicted as normal): 1875 cases.
False Positives (normal predicted as anomaly): 50 cases.
These matrices that highlight the performance differences while in detecting anomalies and normals between the models with Random Forest showing fewer misclassifications overall compared to SVM.

Best Performing Model: Random Forest Why it performed best because the Random Forest which accomplished the highest classification accuracy and performed exceptionally either well on both anomaly and normal instances.It had the lowest number of misclassified instances (false positives and false negatives) when related to SVM and KNN.The reason for this is because Random Forest combines multiple decision trees and offering reliability to overfitting and a strong capability to grip the inproper datasets and the compounded relationships in the data.
Worst Performing Model: SVM Why it is not performed well because the SVM has the lowest accuracy when compared to the three models.It misclassified the more number of instances and particularly in distinguishing normal instances with a total of 140 false negatives and 61 false positives.SVM struggles with high-dimensional data and large datasets like NSL-KDD as it requires more number of computational resources and fine-tuning of hyperparameters (e.g,kernel selection and regularization).Its performance also depends heavily on balanced datasets which might not have been fully achieved here.

4.2 CRITICAL ANALYSIS

The demonstrated processes and the analysis which valubly offers a strong foundation for preparing and evaluating Machine Learning algorithms on an intrusion detection dataset. But some of them should be viewed more critically for better understanding their opportunities and drawbacks. The cleaning steps performed on the ARFF files are crucial to deduplicate a sensible dataset, by removing unwanted features such as multiple spaces and quotes as shown in (25).We also get value, especially in the correct handling of the attribute, from including the unique value extraction from the ‘protocol type’ column. However, the preprocessing steps are based on assumptions of clean and formatted data and might not be prepared to deal with more diverse forms of anomalies in raw data that exists in real world applications.

The representation of the two classes and services shows the distribution of the given datasets and the usage of the classes and services from the data. These insights are quite helpful when it comes to interpreting data and designing ML models. But balancing the type of instances such as the occurrence of anomalies seen by the algorithm as abnormalities is critical to ensure equitable distribution and real-life performance as shown in (26). Categorical feature encoding and data normalization are successfully

performed, which is important for an ML model of SVM and KNN. Though they make models ready for feature analysis, standard scalers have the disadvantage of not considering individual feature scaling needs which could greatly affect performance enhancement impulses.
It could be observed that Random Forest, SVM and KNN are indeed impressive in their performances with an accuracy of 98.71 Percent, 95.54 Percent, and 97.45 Percent respectively.Still these results indicate high performance of the proposed models.Absence of the information about how the models can be tuned with respect to hyperparameters and he computational cost which are crucial in practical applications with high-dimensional networks as shown in (27). Furthermore assessing model performance on imbalance data sets aggregates imbalance problem, yet utilizing techniques such as SMOTE or class weighting could diminish the usability of the advantages stated in this paper. In conclusion, the existence of methodologically sound processes and high strength of results can be seen however extending certain concepts of the work such as addressing dataset imbalances, scalability, and computational complexity concerns will ultimately benefit the implementation of the devised concepts in actual intrusion detection systems.

5 CONCLUTION AND FUTURE WORK

According to the current study it is now possible to assert that they correspond effectively to the research objectives, which were to assess the performance of ML algorithms in intrusion detection with NSL-KDD dataset. Highest accuracy was found for the Random Forest classifier (98.71 Percent) compared to KNN (97.45Percent) and SVM (95.54 Percent).Confusion matrix was performed for all the models(KNN,SVM,Randomforest) and that is over both anomaly and Normal.The confusion matrix analysis specify that the Random Forest model effectively identified (2548) positive cases and (1903) negatives, along with a low number of false positives (36) and false negatives (22).The Support Vector Machine model result in a confusion matrix with high numbers of correctly identified true categories (2523) and false categories (1785) which suggests excellent classification potential. The Support Vector Machine model delivers incorrect results for two groups of cases: detection errors (noticing 140 false positives and missing 61 attacks) to a moreover extent than Random Forest-based models.Strong classification performance is shown in the K-Nearest Neighbors model through its confusion matrix which demonstrates 2517 true positives and 1875 true negatives.Analysis indicates that both false positive results (50) and false negative results (67) remain low which demonstrates the model's strong ability to detect attacks.These outcomes confirm using these models in detecting network intrusions, proving they are appropriate models for IDS applications. The measure of classification quotas such as precision, recall and F1 scores present fair and robust results in both anomaly and normal class.

The results are therefore optimistic and are as are with the propaganda of the benchmark NSL-KDD dataset. However, because of the specified dataset's shortcoming of not embodying modern network characteristics or more sophisticated threats like the advanced persistent threats, such findings cannot be implemented in real-world IoT or cloud network learning environment as shown in (28). Further, though the models showed good results, the scalability and the computational complexity especially in large heterogenous networks with active real time traffic poses a research question. The result analysis of the given study suggests that Random Forest algorithm being robust and understandable should be considered for application in IDS. However, better performance could be achieved by addressing dataset imbalances, as well as by investigating other feature engineering methods.

For future work, using more rea-world traces that capture the current network traffic will also be useful to have a better understanding and a wider range of validation. Further, it may be possible to obtain better results when deploying deep learning models or a combination of the methods proposed in the work in complex environments. Hyperparameter tuning and computational efficiency analysis may also ensure run-ability of these algorithms in real world IDS settings. Finally, it was recommended that techniques for handling imbalance datasets such as oversampling or SMOTE could enhance anomalous data detection in highly imbalanced datasets.

It is also equally important to note that adding more current data sets that consider more of today's network traffic would offer a wider validation possibility. It is also suggested that further elaboration of ensemble methods and deep learning approaches could enhance the identification of more advanced forms of cyber threats. Real-world application of these algorithms and their implementation instantaneously in applications such as the IoT and cloud would also be necessary. In addition, computational time of these models and hyperparameters tuning for real-time implementation would have made the results more applicable. Thus, the use of self-learning algorithms to address newer forms of cyber threats can guarantee effectiveness of the IDS. These directions, in principle, might greatly extend the possibilities and practical use of ML-based IDS solutions.

References

[1] Z. Azam, M. M. Islam, and M. N. Huda, "Comparative analysis of intrusion detection systems and machine learning based model analysis through decision tree," *IEEE Access*, 2023.

[2] R. A. Elsayed, R. A. Hamada, M. I. Abdalla, and S. A. Elsaid, "Securing iot and sdn systems using deep-learning based automatic intrusion detection," *Ain Shams Engineering Journal*, vol. 14, no. 10, p. 102211, 2023.

[3] G. Logeswari, S. Bose, and T. Anitha, "An intrusion detection system for sdn using machine learning," *Intelligent Automation & Soft Computing*, vol. 35, no. 1, pp. 867–880, 2023.

[4] D. Musleh, M. Alotaibi, F. Alhaidari, A. Rahman, and R. M. Mohammad, "Intrusion detection system using feature extraction with machine learning algorithms in iot," *Journal of Sensor and Actuator Networks*, vol. 12, no. 2, p. 29, 2023.

[5] U. Inayat, M. F. Zia, S. Mahmood, H. M. Khalid, and M. Benbouzid, "Learningbased methods for cyber attacks detection in iot systems: A survey on methods, analysis, and future prospects," *Electronics*, vol. 11, no. 9, p. 1502, 2022.

[6] M. S. Alsahli, M. M. Almasri, M. Al-Akhras, A. I. Al-Issa, and M. Alawairdhi, "Evaluation of machine learning algorithms for intrusion detection system in wsn," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 5, 2021.

[7] D. Nkashama, A. Soltani, J.-C. Verdier, M. Frappier, P.-M. Tardif, and F. Kabanza, "Robustness evaluation of deep unsupervised learning algorithms for intrusion detection systems," *arXiv preprint arXiv:2207.03576*, 2022.

[8] S. Waskle, L. Parashar, and U. Singh, "Intrusion detection system using pca with random forest approach," in *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*. IEEE, 2020, pp. 803–808.

[9] S. Amaran and R. M. Mohan, "Intrusion detection system using optimal support vector machine for wireless sensor networks," in *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)*. IEEE, 2021, pp. 1100–1104.

[10] I. Abrar, Z. Ayub, F. Masoodi, and A. M. Bamhdi, "A machine learning approach for intrusion detection system on nsl-kdd dataset," in *2020 international conference on smart electronics and communication (ICOSEC)*. IEEE, 2020, pp. 919–924.

[11] T. Su, H. Sun, J. Zhu, S. Wang, and Y. Li, "Bat: Deep learning methods on network intrusion detection using nsl-kdd dataset," *IEEE Access*, vol. 8, pp. 29575–29585, 2020.

[12] H. Sadia, S. Farhan, Y. U. Haq, R. Sana, T. Mahmood, S. A. O. Bahaj, and A. Rehman, "Intrusion detection system for wireless sensor networks: A machine learning based approach," *IEEE Access*, 2024.

[13] F. Louati, F. B. Ktata, and I. Amous, "Big-ids: a decentralized multi agent reinforcement learning approach for distributed intrusion detection in big data networks," *Cluster Computing*, pp. 1–19, 2024.

[14] S. P. Thirimanne, L. Jayawardana, L. Yasakethu, P. Liyanaarachchi, and C. Hewage, "Deep neural network based real-time intrusion detection system," *SN Computer Science*, vol. 3, no. 2, p. 145, 2022.

[15] M. A. Umer, K. N. Junejo, M. T. Jilani, and A. P. Mathur, "Machine learning for intrusion detection in industrial control systems: Applications, challenges, and recommendations," *International Journal of Critical Infrastructure Protection*, vol. 38, p. 100516, 2022.

[16] D. Kreps and F. Rowe, "Research perspectives: Using a process philosophy perspective in information systems research: Principles of creative evolution," *Journal of the Association for Information Systems*, vol. 25, no. 5, pp. 1410–1433, 2024.

[17] E. Saeedi Taleghani, R. I. Maldonado Valencia, A. L. Sandoval Orozco, and L. J. Garc´ıa Villalba, "Trust evaluation techniques for 6g networks: a comprehensive survey with fuzzy algorithm approach," *Electronics*, vol. 13, no. 15, p. 3013, 2024.

[18] Y. A. Al-Khassawneh, "An investigation of the intrusion detection system for the nsl-kdd dataset using machine-learning algorithms," in *2023 IEEE International Conference on Electro Information Technology (eIT)*. IEEE, 2023, pp. 518–523.

[19] A. Ali and M. M. Yousaf, "Novel three-tier intrusion detection and prevention system in software defined network," *IEEE Access*, vol. 8, pp. 109662–109676, 2020.

[20] M. Almehdhar, A. Albaseer, M. A. Khan, M. Abdallah, H. Menouar, S. Al-Kuwari, and A. Al-Fuqaha, "Deep learning in the fast lane: A survey on advanced intrusion detection systems for intelligent vehicle networks," *IEEE Open Journal of Vehicular Technology*, 2024.

[21] S. D. Roy, S. Debbarma, and A. Iqbal, "A decentralized intrusion detection system for security of generation control," *IEEE Internet of Things Journal*, vol. 9, no. 19, pp. 18924–18933, 2022.

[22] S. A. Bakhsh, M. A. Khan, F. Ahmed, M. S. Alshehri, H. Ali, and J. Ahmad, "Enhancing iot network security through deep learning-powered intrusion detection system," *Internet of Things*, vol. 24, p. 100936, 2023.

[23] Z. K. Maseer, R. Yusof, N. Bahaman, S. A. Mostafa, and C. F. M. Foozy, "Benchmarking of machine learning for anomaly based intrusion detection systems in the cicids2017 dataset," *IEEE access*, vol. 9, pp. 22351–22370, 2021.

[24] A. Fatani, M. Abd Elaziz, A. Dahou, M. A. Al-Qaness, and S. Lu, "Iot intrusion detection system using deep learning and enhanced transient search optimization," *IEEE Access*, vol. 9, pp. 123448–123464, 2021.

[25] M. Sarhan, S. Layeghy, N. Moustafa, M. Gallagher, and M. Portmann, "Feature extraction for machine learning-based intrusion detection in iot networks," *Digital Communications and Networks*, vol. 10, no. 1, pp. 205–216, 2024.

[26] L. Liu, P. Wang, J. Lin, and L. Liu, "Intrusion detection of imbalanced network traffic based on machine learning and deep learning," *IEEE access*, vol. 9, pp. 7550– 7563, 2020.

[27] G. D. C. Bertoli, L. A. P. Ju´nior, O. Saotome, A. L. Dos Santos, F. A. N. Verri, C. A. C. Marcondes, S. Barbieri, M. S. Rodrigues, and J. M. P. De Oliveira, "An endto-end framework for machine learning-based network intrusion detection system," *IEEE Access*, vol. 9, pp. 106790–106805, 2021.

[28] S. Kumar, "Survey of current network intrusion detection techniques," *Washington Univ. in St. Louis*, pp. 1–18, 2007.

Project-url: https://colab.research.google.com/drive/1krIxK66cG6cnKbpVuVNo91V7w51GKu scrollTo=kROqbHCuNcRg