International Journal of Allied Sciences (IJAS)
Volume.16, Number 9; September, 2025;
ISSN: 2836-3760| Impact Factor: 8.13
https://zapjournals.com/Journals/index.php/Allied-Sciences
Published By: Zendo Academic Publishing

NETWORK INTRUSION DETECTION USING BIG DATA ANALYTICS: A PYSPARK AND HIVE APPROACH FOR UNSW-NB15

Daniel U. Okon

Article Info

Keywords: Cybersecurity; Network Intrusion Detection; Big Data Analytics; Apache Spark; Apache Hive; UNSW-NB15; Random Forest; Machine Learning

DOI
10.5281/zenodo.17135943

Abstract

Network intrusion detection remains a critical challenge as cyber threats continue to evolve in complexity and scale. This study investigates the application of big data analytics for intrusion detection using the UNSW-NB15 dataset. Apache Hive was used for large-scale querying and feature analysis, while PySpark was used for advanced analytics, including descriptive statistics, correlation, hypothesis testing, and dimensionality reduction. A RF classifier was developed and evaluated for both binary and multi-class intrusion detection tasks. The experimental results demonstrate a 99.99% accuracy in binary classification and 98.62% in multi-class classification, highlighting the effectiveness of combining Hive and PySpark for scalable intrusion detection. These findings underscore the importance of big data frameworks in strengthening cybersecurity defence systems.

1. Introduction

The proliferation of digital technology and internet connectivity has resulted in a huge increase in cyber risks, with network intrusions being the most serious and damaging threat faced by modern enterprises (Chen et al., 2019). Figure 1 depicts the amount of damage caused by cybercrime reported to the IC3 between 2001 and 2020 (Federal Bureau of Investigation, 2021). The annual loss of complaints referred to the IC3 in the most recent period was US$4.2 billion. These alarming statistics highlight the crucial need to install effective intrusion detection systems (IDS) to protect important data and resources from various cyber-attacks. As described by Aminanto et al. (2017), by identifying malicious activity, providing early warnings, and permitting fast countermeasures, intrusion detection systems (IDS) serve a critical role in detecting and minimising the impacts of cyber-attacks, Due to the obvious reason that cyber threats are becoming more sophisticated, ongoing developments in IDS are required to keep up with the dynamic threat landscape, assuring the security of enterprises and individuals against the ever-increasing risk of cybercrime (Wang et al., 2018). Figure 2 displays intrusion detection, in which the firewall operates as an intrusion detector located among several networks,

Faculty of Computing, Department of Cybersecurity, University of Port Harcourt, Port Harcourt Rivers, Nigeria
Email: daniel.okon@uniport.edu.ng

filtering data that may constitute a threat (Alrawashdeh & Alhamid, 2020). Although it is feasible to improve network security, by employing the use of IDS, the ability of traditional intrusion detection systems (IDS) to handle and analyse large amounts of network traffic data in real time is limited. As a result, there is a growing demand for more resilient and adaptive network security solutions, which can be accomplished by using big data analytics methods (Mell, 2015).

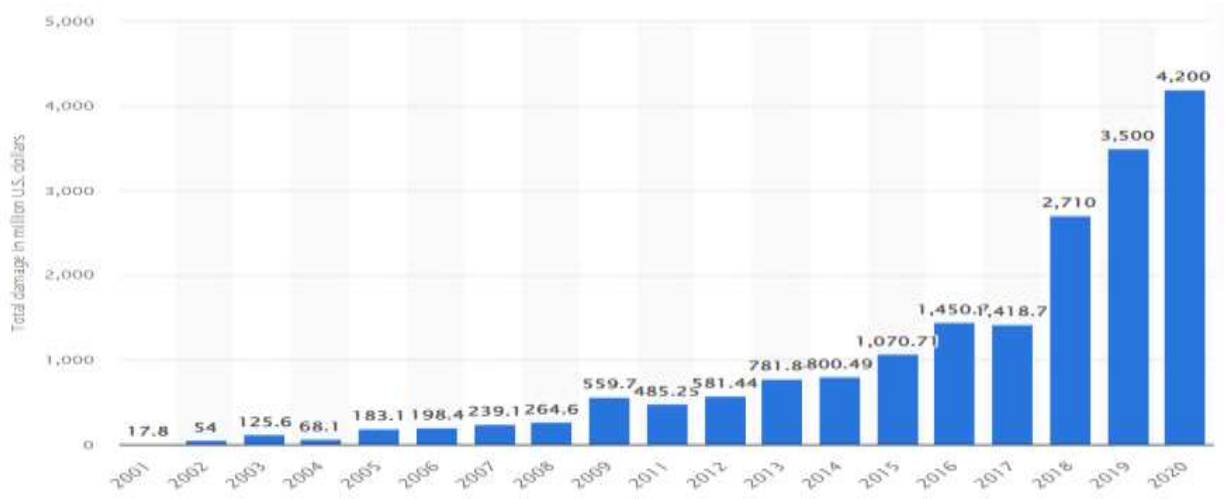

Figure 1 the Internet Crime Complaint Centre (IC3) from 2001 to 2020 Statistics

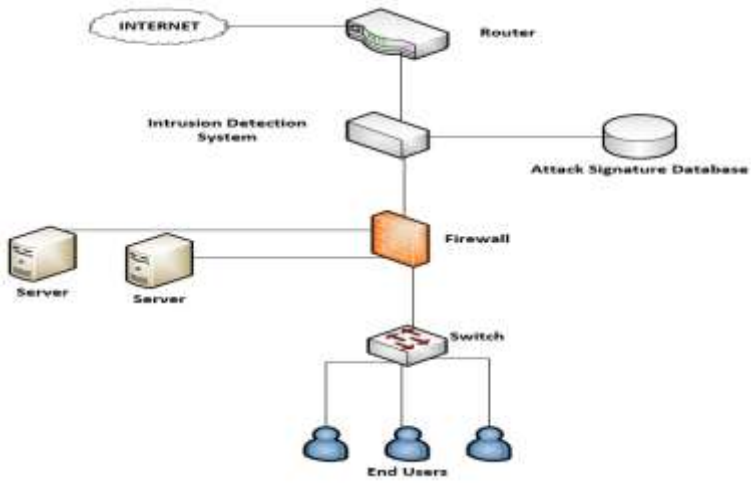

Figure 2 Network security system

Big data analytics have emerged as an indispensable element in diverse domains, encompassing network security. The introduction of distributed computing platforms, such as Apache Hadoop and Apache Spark, has altered the approach of researchers and professionals to the administration of large datasets, allowing for unequalled speed and efficacy in data processing and analysis. Furthermore, Apache Hive provides a solid foundation for running SQL-like queries on large datasets, easing the process of deriving important insights from big data. The objective of this study is to perform a thorough examination of network intrusion detection using big data analytics. The study will concentrate on the execution of Apache Spark and Apache Hive on the UNSW-NB15 dataset. This study aims to utilise various analytical techniques to extract meaningful patterns and associations from the dataset.

The obtained findings will be applied in constructing a classifier model for binary and multi-classification of the identified attack types.

1.1 Objectives

This study aims to explore the potential of big data analytics to improve the identification and categorisation of network intrusions. Our objective is to achieve the following goals:

- We employed Apache Hive for big data queries and analysis on the UNSW-NB15 dataset to generate insights for end users. This requires a thorough comprehension of the dataset, the development of Hive queries, the use of relevant visualisation tools, and the presentation of findings both numerically and graphically, as well as brief interpretations.
- Analyse and interpret UNSW data using at least four analytical methods, such as descriptive statistics, correlation, hypothesis testing, and density estimation, to acquire insights into the dataset's underlying patterns and relationships.
- Design and develop a binary and multi-classifier RF model for attack type classification, leveraging the insights gained from the analytical methods.
- Evaluation of both binary and multi-class classifier models using different evaluation metrics, such as accuracy, precision, recall, and F1-score.

1.2 Apache Spark

Apache Spark is a powerful distributed computing solution that analyses massive amounts of data efficiently by using in-memory caching and optimised query execution. Spark is an adaptable and scalable solution for big data processing, with a rich library suite that includes MLlib for machine learning, GraphX for graph processing, and Spark Streaming for real-time data processing. It is accessible to developers and academics worldwide because it is open-source, enabling continual progress and innovation in the field of distributed computing. It is an excellent platform for big data analytics in network intrusion detection because of its inherent fault tolerance, scalability, and extensibility (Chowdhury et al., 2011). Apache Spark was used to execute the analytical techniques and construct the classifier model. The MLlib library offered by Spark presents an extensive range of ML algorithms that enable diverse analytical methods for the UNSW-NB15 dataset and the creation of the RF classifier model.

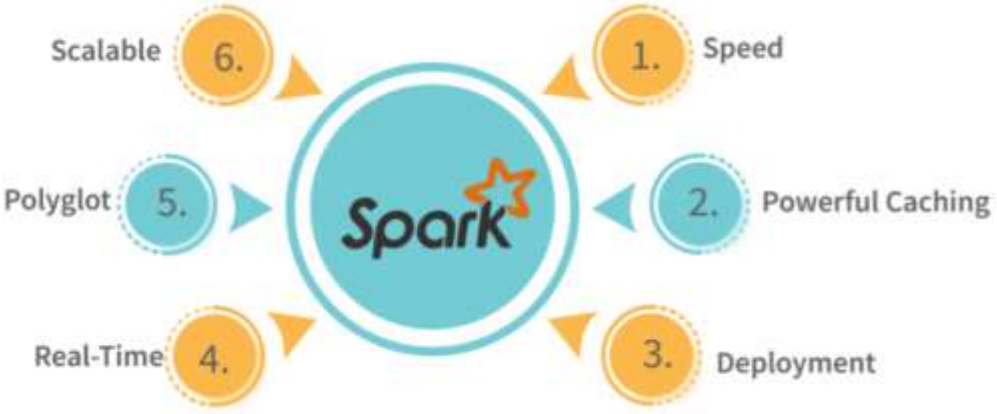

Figure 3 Advantages of Spark

1.3 Apache Hive

Apache Hive is a data warehousing solution that has been developed on the Hadoop Distributed File System (HDFS). Facilitates the storage and administration of voluminous datasets in a distributed setting. Hive provides a query language, referred to as HiveQL, that bears resemblance to SQL (Apache Hive TM, 2011). This language enables users to perform intricate queries on vast datasets without the need for advanced programming proficiency. The combination of Apache Hive and Apache Spark improves the efficacy and expedites the execution of big data analytical assignments, making it an indispensable instrument in our scholarly inquiry. This study employs Apache Hive as a tool for large-scale data querying and analysis on the UNSW-NB15 dataset. Multiple Hive queries are executed to extract pertinent features, preprocess the data, and conduct exploratory data analysis. The insights derived from these queries serve as the foundation for the subsequent implementation of the analytical methods and the development of the classifier model.

Figure 4 Apache Hadoop Ecosystem

2. UNSW Dataset

Researchers from the University of New South Wales and the Australian Centre for Cyber Security created the UNSW-NB15 dataset to provide a thorough network traffic dataset that is tailored to assess the efficacy of network intrusion detection systems (Moustafa, n.d.). The dataset comprises more than 2.5 million records, each of which corresponds to a network connection. The records have 49 features, including the source and destination IP addresses, protocols, and payload characteristics, that encompass different aspects of the connection. The UNSW-NB15 dataset includes various attack types, including denial of service (DoS), remote code execution (RCE), and brute force attacks. The dataset includes a label that indicates the type of attack and a binary label that distinguishes between ordinary and malicious connections. The dataset's intricate and varied character makes it a strong option for evaluating big data analytics approaches in the network intrusion detection domain. Figure 4 shows the normal and attack distributions in the dataset. From the chart, we can see that the most common attack is the generic attack, followed by the exploit, fuzzer, and DDO attacks.

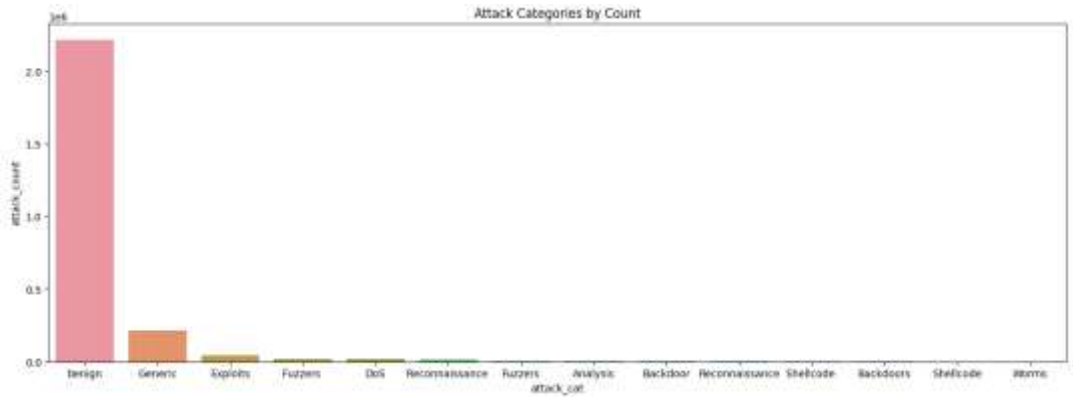

Figure 5 Distribution of normal and attack labels

2.1 Attack Types

The UNSW-NB15 dataset contains a diverse range of attack types classified into nine main classes. The aforementioned categories represent many expressions of cyber dangers, providing a thorough portrayal of the challenges experienced by computer networks' intrusion detection systems. The following sections provide a concise summary of each assault type found in the dataset:

- Fuzzers: Fuzzers are attacks that produce and transmit random, flawed, or invalid inputs to a target system to cause unexpected behaviour, crashes, or security vulnerabilities.
- Analysis: Analysis attacks focus on identifying and exploiting weaknesses in target systems. In these assaults, scanning the network for open ports, services, or applications, as well as testing systems for flaws, is common. Examples of analysis attacks include port scanning, OS fingerprinting, and vulnerability scanning.
- DoS: Denial of Service attacks seek to impair the availability of a network, service, or application by overwhelming it with excessive traffic or resource usage. DoS attacks can be carried out via various techniques, including SYN flood, ICMP flood, and HTTP flood. The primary purpose of these assaults is to make the target system inaccessible or ineffective, causing inconvenience to its users.
- Exploit: Exploit attacks employ known weaknesses in software, hardware, or protocols to gain unauthorised access, control, or privileges within a target system. These attacks frequently rely on the availability of public or private exploit code and can result in arbitrary code execution, data exfiltration, or system penetration.
- Generic: Generic attacks cover various approaches that do not fit into any particular category. These attacks may include network traffic manipulation, unauthorised data access, or the exploitation of legitimate system capabilities. Generic attacks can provide particular challenges to network intrusion detection systems due to their diversity.
- Reconnaissance: Reconnaissance attacks gather information about a target network, system, or organisation in order to discover possible targets for future assaults. Passive techniques, such as eavesdropping on network traffic, and active approaches, such as DNS enumeration or social engineering, can be used in these assaults.

Comprehending the various attack types and their unique characteristics is crucial in the creation of effective NIDS. This enables the development of models that can accurately differentiate between benign and malicious network traffic patterns and classify attacks according to their respective categories. This study explores the diverse attack categories present in the UNSW-NB15 dataset to establish a RF framework for both binary and

multi-classification objectives. This study aims to enhance the ability to detect and classify network intrusions for rapid identification.

3. Big Data Query and Analysis using Apache Hive

Big data analytics are critical in identifying and mitigating cyber threats, particularly in the domain of network intrusion detection. This study presents the results of our data analysis on the UNSW-NB15 dataset using Apache Hive queries. The goal of this study is to convert unprocessed data into useful information for end recipients and to draw meaningful conclusions from the data. We use Hive queries to perform various analyses and provide the results in both numerical and graphical representations.

3.1 Query 1: Top 5 attack categories by count

The primary objective of our preliminary investigation is to determine the five most common attack categories based on their frequency. To accomplish this task, we execute the following query on Apache Hive:

Figure 6 Query: Top 5 Code Attacks

This query selects the attack category and enumerates the occurrences of each attack type. It also identifies the attack category and records the number of instances of each attack type. We deliberately ignore “benign” traffic in our query because it does not represent harmful activities. The query then groups the results by attack category and displays them in descending order according to the attack count. The output only includes the top five results.

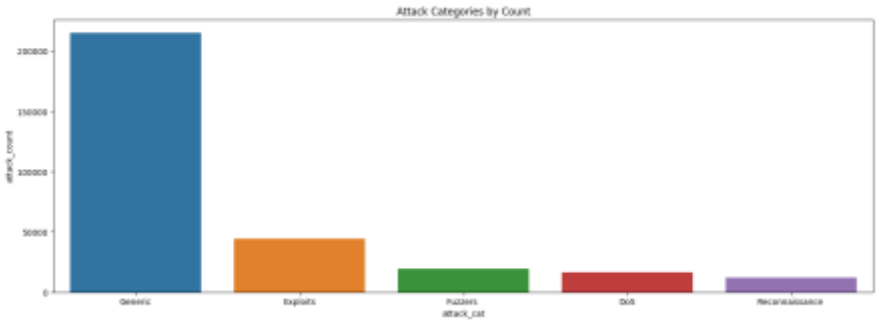

Figure 7 Top 5 Attack Labels

According to the results of this query, the “generic” attack category is the most common, accounting for 215,481 attacks. Following this are “Exploits” (44,525), “Fuzzers” (19,195), “DoS” (16,353), and “Reconnaissance” (12,228). Figure 8 shows a bar chart created with Python’s Seaborn package to visually depict the findings. In the UNSW-NB15 dataset, the “Generic” attack category is by far the most common type of attack.

3.2 Query 2: Top 10 source group with IP involved in the attacks

Our analysis aims to identify and pinpoint the top 10 source group with IP implicated in attacks by analysing the frequency of the observed source IP addresses. The following query was implemented to accomplish this task:

Query 2: Top 10 source IPs involved in attacks

```
[ ] query2 = """
    SELECT srcip, COUNT(*) as attack_count
    FROM unsw_hive
    WHERE Label = 1
    GROUP BY srcip
    ORDER BY attack_count DESC
    LIMIT 10
    """

    result2 = hive_context.sql(query2).toPandas()
```

Figure 8 Query: Top 10 source IPs involved in attacks

This query retrieves the source IP addresses and counts each occurrence. By filtering out non-attack traffic with the label field, we ensure that only malicious activities are considered. The results are then grouped by source IP, sorted by attack count in descending order, and limited to the top IP entries. The results of this query show that the source IP '175.45.176.1' is the most common, with 117,908 assaults. '175.45.176.3' (106,418), '175.45.176.0' (74,279), and '175.45.176.2' (22,678) follow.

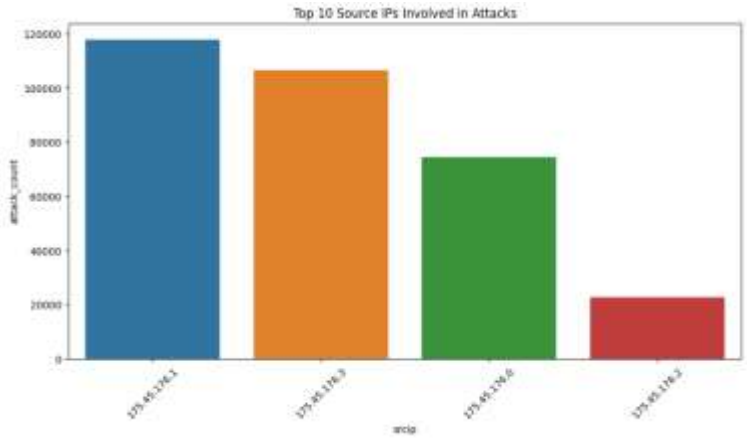

Figure 9 Top 10 source IPs involved in attacks

To visually present these findings, a bar chart created with the Seaborn library in Python is employed, as displayed in Figure 9. The graph highlights that the top three source group with IP contribute to a significantly higher number of attacks than the remaining source group with IP in the dataset. This information is crucial in understanding the behaviour of the most active attackers and devising strategies to mitigate their occurrences.

3.3 Query 3: Attack distribution according to protocol

This query examines the attack distribution by protocol. This query focuses on the protocol column and counts the number of attacks for each protocol by excluding malicious traffic with a Label of 1 (i.e., malicious traffic with a Label of 1 The results are then categorised by protocol and based on the number of attacks, ordered in descending order. Finally, the output is restricted to the top 25 protocols with the most attacks.

Query 3: Attacks distribution by protocol

```
query3 = """
SELECT proto, COUNT(*) as attack_count
FROM unsw_hive
WHERE Label = 1
GROUP BY proto
ORDER BY attack_count DESC Limit 25
"""

result3 = hive_context.sql(query3).toPandas()
```

Figure 10 Query: Attack distribution by protocol

UDP is the most targeted protocol, with 223,750 reported attacks, followed by TCP (58,184) and UNAS (16,202). These findings are visually represented in Figure 11, which displays a bar plot of the attack distribution by protocol. The results of this analysis provide valuable insights into the behaviour of attackers targeting specific network protocols. These data can be used to inform the design and setup of intrusion detection systems, which can be tailored to detect and mitigate assaults on various protocols. Furthermore, this analysis lays the groundwork for future research into the vulnerabilities and hazards associated with various network protocols, allowing for the creation of more effective security measures.

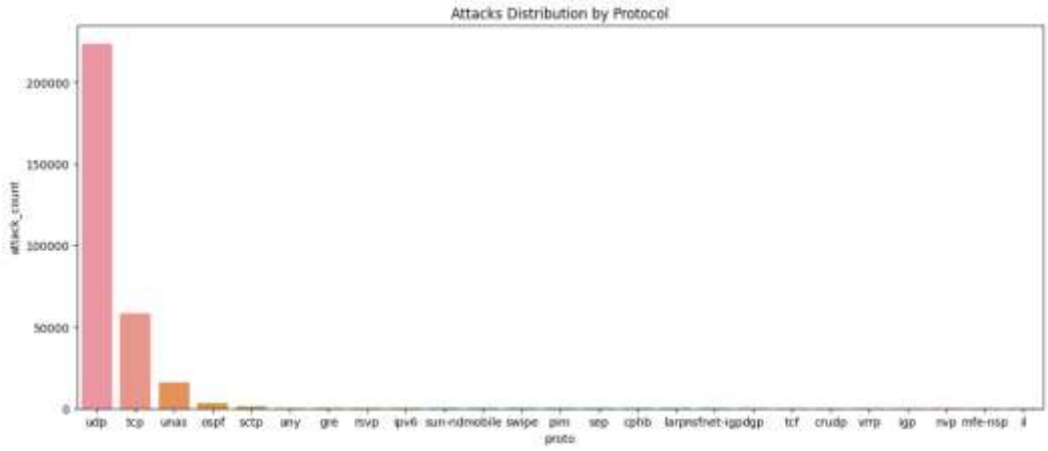

Figure 11 Attack distribution by protocol

3.4 Query 4: Average duration and total byte count per category of attack

This analysis aims to ascertain the average duration and total byte count of attacks in the UNSW-NB15 dataset for each attack category, as shown in Figure 12. This query selects the attack type and calculates the average duration and total byte count for each category while eliminating benign traffic using a WHERE clause. The findings are then categorised by attack category, sorted in descending order based on total byte count, and displayed in tabular form.

Query 4: Average duration and total byte count per attack category

```
[ ] query4 = """
    SELECT attack_cat, AVG(dur) as avg_duration, SUM(sbytes + dbytes) as total_bytes
    FROM unsw_hive
    WHERE attack_cat != 'benign'
    GROUP BY attack_cat
    ORDER BY total_bytes DESC
    """

    result4 = hive_context.sql(query4).toPandas()
```

Figure 12 Query: Average duration and total byte count per category of attack

The bar and line plots are used together to visualise the results, as shown in Figure 13. The graph shows the overall byte count for each assault category, with the average duration represented by a line plot. The entire byte count is shown on the left y-axis, and the average time is shown on the right y-axes, respectively. According to the findings, the attack categories with the highest total byte counts are "Exploits," "DoS," and "Generic." “Fuzzers,” “Exploits,” and “DoS” have the longest average durations. These findings have significant implications for designing and implementing security measures in each category to detect and mitigate attacks.

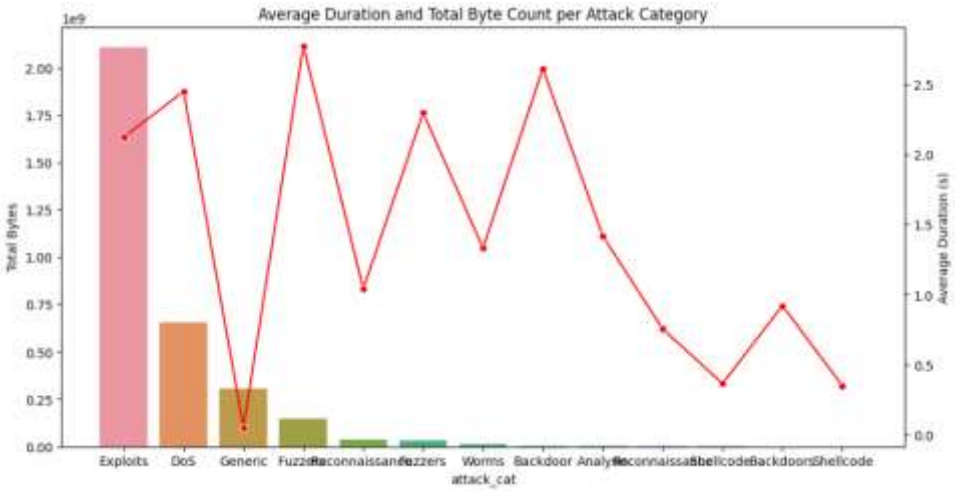

Figure 13 Average duration and total byte count per attack category

3.5 Query 5: Proportion of attacks with a non-zero response body length, grouped by the attack category

The following query was created to analyse the proportion of attacks in the UNSW-NB15 dataset with non-zero response body length for each attack category. The query selects the attack category and computes the total number of assaults, number of attacks with a non-zero response body length, and percentage of attacks with a non-zero response body length for each category, omitting the ‘benign’ traffic category. The findings are then tabulated and organised by attack category and ranked in descending order according to the percentage of attacks with a non-zero response body length.

Query 5: Proportion of attacks with non-zero response body length, grouped by attack category

```
[20] query5 = """
     SELECT attack_cat,
            COUNT(*) as total_attacks,
            SUM(CASE WHEN res_bdy_len > 0 THEN 1 ELSE 0 END) as attacks_with_body,
            (SUM(CASE WHEN res_bdy_len > 0 THEN 1 ELSE 0 END) / COUNT(*)) * 100 as percentage_with_body
     FROM unsw_hive
     WHERE attack_cat != 'benign'
     GROUP BY attack_cat
     ORDER BY percentage_with_body DESC
     """

     result5 = hive_context.sql(query5).toPandas()
```

Figure 14 Query: Proportion of attacks with a non-zero response body length, grouped by the attack category

The analysis attack type had the highest percentage of attacks with a non-zero response body length, followed by "Worms" and "Exploits." This implies that the presence of a non-zero response body length is more connected with certain types of attacks than others. On the other hand, the "Fuzzers," "Reconnaissance," "Shellcode," and "Backdoors" attack categories have a very low percentage of attacks with a non-zero response body length. These findings can be used to create more effective intrusion detection and prevention systems that consider each attack category's unique characteristics, allowing for improved cyber threat defence.

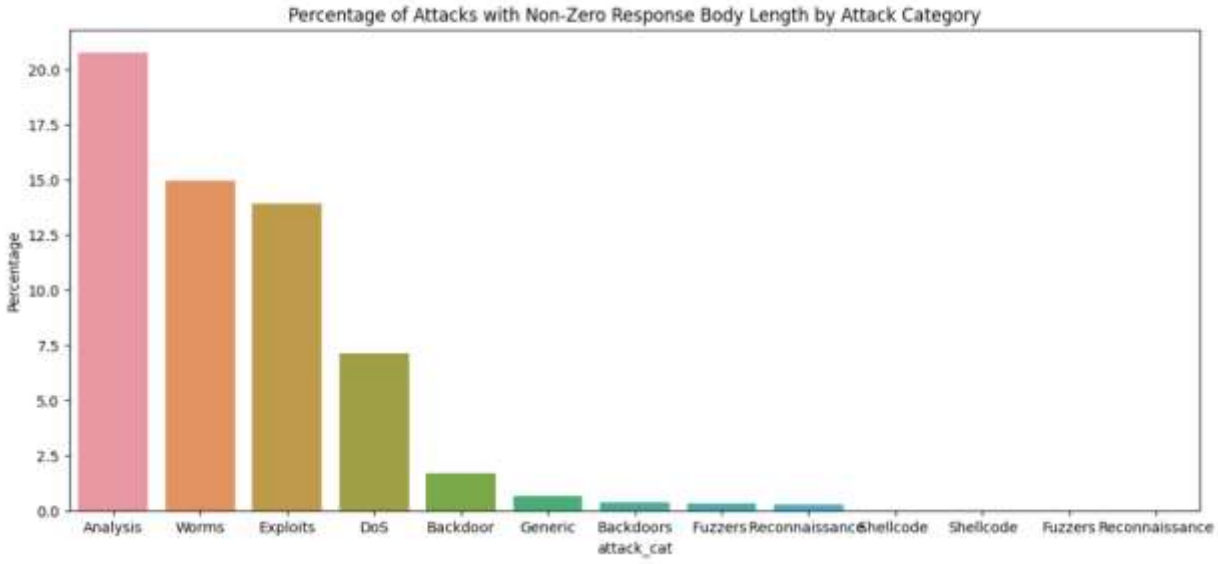

Figure 15 Proportion of attacks with non-zero response body length, grouped by attack category

4. Advanced Analytics Using PySpark

Big data analysis and interpretation are critical components of data science and are used to derive important insights from massive amounts of data. To gain a better understanding of the data, various analytical techniques, such as descriptive statistics, correlation, hypothesis testing, and density estimation, must be employed. PySpark was used to analyse and interpret the UNSW-NB15 dataset using at least four analytical methodologies.

4.1 Descriptive Summary

A descriptive analysis is a statistical analysis in which the main features of a dataset are summarised and described. The purpose of descriptive analysis is to provide a high-level overview of the dataset, highlight potential flaws or patterns, and inform subsequent studies. The initial step in descriptive analysis is to identify which of the dataset's numeric columns may be studied. In this task, 39 columns were chosen, comprising information such as the source and destination ports, connection length, and amount of bytes exchanged. Summary statistics, such as the count, mean, standard deviation, minimum, and maximum values, have been computed using these columns. The summary statistics were produced in PySpark using the describe() method, which generates a DataFrame containing the summary statistics for each numeric column. The toPandas() method was used to convert the Spark DataFrame to a Pandas DataFrame for easier display and analysis. The results showed that the 'dur' variable has a mean of 0.6588 s and a maximum of 8786.6377 s, indicating that the dataset has some unusually long durations. The "sbytes" variable has a mean of 4340.07 bytes and a maximum of 14355774 bytes, indicating various packet sizes. The "ct_srv_src" and "ct_srv_dst" variables have a maximum value of 67, which suggests that a few sources and destinations are responsible for a large number of connections. Overall, by analysing the summary statistics, we can obtain an idea of the range of values for each numeric column and the distribution of values across the dataset.

	summary	sport	dsport	dur	sbytes	dbytes	sttl	dttl	sloss	dloss	...	is_ftp_login
0	count	2539739	2539739	2539739	2539739	2539738	2539739	2539739	2539739	2539739	...	2539739
1	mean	30536.931006583205	11235.096789685767	0.6588634003800393	4348.072263330996	36432.01132478574	62.78149802937214	30.770449978384	5.164547223159545	16.331423819534212	...	0.017353579639878
2	stddev	20441.216792342224	18438.200883577165	13.925767633289283	56489.398122286232	161105.38488824756	74.02669956848524	42.85192218951872	22.518368387797222	66.59788363285116	...	0.13346518417617913
3	min	0	0	0.0	0	0	0	0	0	0	...	0
4	max	65535	65535	8786.6377	14355774	14657531	255	254	5319	5507	...	4

5 rows × 40 columns

Figure 16 Descriptive analysis results

4.2 Hypothesis testing

Hypothesis testing is a statistical tool for determining whether observed differences in sample data are real or random variation. In this task, we investigate the possibility of a statistically significant difference in the mean value of the "dur" characteristic between normal and attack traffic. Following the null and alternative hypotheses for this task:

- The null hypothesis (H0): The mean "dur" value is the same for both normal and attack traffic.
- Alternative hypothesis (HA): The mean "dur" value is different for normal traffic and attack traffic.

The "dur" values for both regular and intrusive network traffic are collected from the dataset to test the hypothesis. The t-statistic and p-value were then computed using a two-sample t-test. The t-test assumes that the data follows a normal distribution and that both groups' variances are comparable.

Hypothesis Testing

- Appling Hypothesis Testing To check if the mean dur value is different between normal traffic and attack traffic
- Null hypothesis (H0): The mean 'dur' value is the same for both normal traffic and attack traffic.
- Alternative hypothesis (HA): The mean 'dur' value is different for normal traffic and attack traffic.

```
[ ] normal_traffic = data.filter(data.Label == 0).select('dur').rdd.flatMap(lambda x: x).collect()
    attack_traffic = data.filter(data.Label == 1).select('dur').rdd.flatMap(lambda x: x).collect()

    t_statistic, p_value = ttest_ind(normal_traffic, attack_traffic)
```

```
[ ] print("P Value is:",p_value)

    P Value is: 0.0018154559048992132
```

Figure 17 Hypothesis testing code

The null hypothesis is rejected when the p-value is less than the preset significance level, commonly set at 0.05. Boxplot of 'dur' attribute for both normal and attack traffic was created to visualise the results of the hypothesis. Figure 18 illustrates the median, interquartile range, and outliers for each group, as well as the p-value. In this case, the p-value was 0.0018, which is less than the significance level. As a result, the null hypothesis was rejected, indicating that normal and attack traffic have different means, as corroborated by the median 'dur' value of the plot, which is greater than for attack traffic.

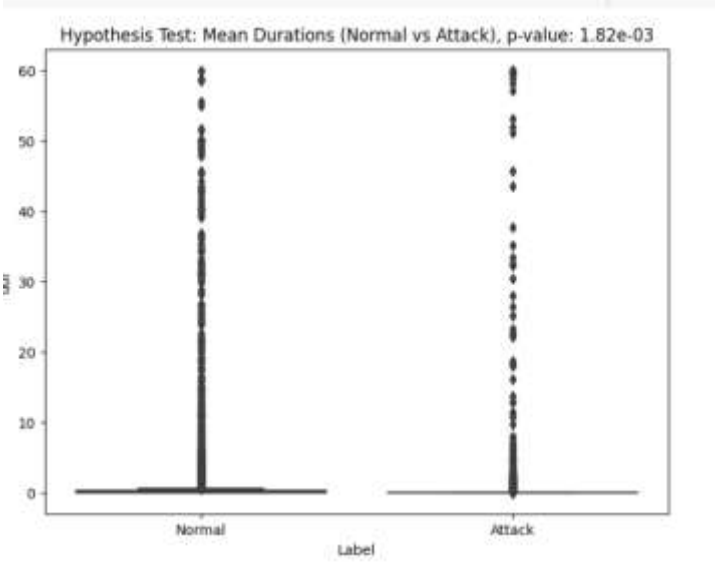

Figure 18 Hypothesis test boxplot

4.3 Correlation

The Pearson correlation was implemented to understand the relationship between numerical features in the UNSW-NB15 dataset. Pearson's correlation coefficient is a statistical measure that quantifies the magnitude and direction of a linear association between two variables (Turney , 2022). The values for Pearson correlation coefficient ranges from -1 to 1, where a value of -1 signifies a complete negative correlation, indicating that as one variable increases, the other decreases. A value of 1 indicates a perfect positive correlation, indicating that both variables increase or decrease together, and a value of 0 indicates no correlation. The categorical variables were eliminated from the dataset, and the correlation between the numerical features was assessed to identify

highly correlated features. In the code, we used the drop() method to eliminate categorical variables from the dataset and the VectorAssembler() function to create a correlation assembler. The Correlation.corr() function was then used to calculate the Pearson correlation coefficient and the results were recorded in a data frame. Finally, we printed out highly correlated pairs of characteristics (correlation coefficient > 0.8 or -0.8) as shown in Figure 22.

```
columns_to_drop = ['dstip', 'proto','state','service','attack_cat','srcip']
corrleation_matrix = data.drop(*columns_to_drop)

## Creating Correlation Assembler
vector_col = "corr_features"
assembler = VectorAssembler(inputCols=corrleation_matrix.columns, outputCol=vector_col)
vect_cor = assembler.transform(corrleation_matrix).select(vector_col)

## Correlation Matrix
matrix = Correlation.corr(vect_cor, vector_col)

## Applying Pearson Correlation
correlation = Correlation.corr(vect_cor,vector_col,"pearson").collect()[0][0]

rows = correlation.toArray().tolist()
cor_df = spark.createDataFrame(rows,corrleation_matrix.columns)

## Visulize
cor_df_pd = cor_df.toPandas()
cor_df.show()
```

Figure 19 Correlation coefficient

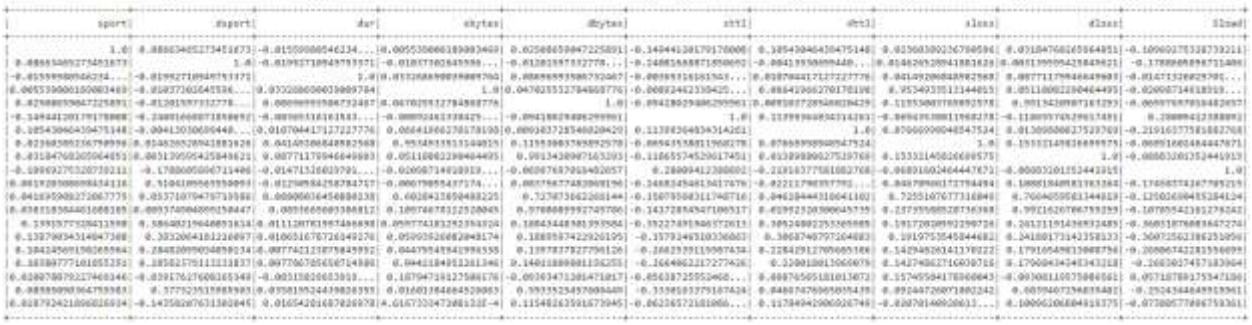

Figure 20 Correlation Graph of the

```
Highly correlated pairs (correlation coefficient > 0.8 or < -0.8):
sloss - sbytes: 0.9534933513144015
dloss - dbytes: 0.9913420907163293
Dpkts - dbytes: 0.9700089992745766
Dpkts - dloss: 0.9021626706759259
Dpkts - Spkts: 0.8225776151624578
dwin - swin: 0.9972069954000567
Ltime - Stime: 0.9999999997797463
synack - tcprtt: 0.9313053022033396
ackdat - tcprtt: 0.9108907922403802
is_sm_ips_ports - Sintpkt: 0.8066290486949693
ct_state_ttl - sttl: 0.9060746751965445
ct_ftp_cmd - is_ftp_login: 0.8496322369550772
ct_srv_dst - ct_srv_src: 0.9567312281500591
ct_dst_ltm - ct_srv_src: 0.8366557815433092
ct_dst_ltm - ct_srv_dst: 0.8598876413790725
ct_src_ ltm - ct_srv_src: 0.8406144171558931
ct_src_ ltm - ct_srv_dst: 0.8246727417093495
ct_src_ ltm - ct_dst_ltm: 0.9385165230613475
ct_src_dport_ltm - ct_srv_src: 0.8697401995300258
ct_src_dport_ltm - ct_srv_dst: 0.875945140205581
ct_src_dport_ltm - ct_dst_ltm: 0.9601552558311022
ct_src_dport_ltm - ct_src_ ltm: 0.9455356377357514
ct_dst_sport_ltm - ct_srv_src: 0.8405524119730095
ct_dst_sport_ltm - ct_srv_dst: 0.848828346393225
ct_dst_sport_ltm - ct_dst_ltm: 0.8801999225813902
ct_dst_sport_ltm - ct_src_ ltm: 0.8602689290121267
ct_dst_sport_ltm - ct_src_dport_ltm: 0.9214880967474458
ct_dst_src_ltm - ct_srv_src: 0.9421756410442572
ct_dst_src_ltm - ct_srv_dst: 0.9510264424997046
ct_dst_src_ltm - ct_dst_ltm: 0.876487530129229
ct_dst_src_ltm - ct_src_ ltm: 0.8567978501003428
ct_dst_src_ltm - ct_src_dport_ltm: 0.9108925294471405
ct_dst_src_ltm - ct_dst_sport_ltm: 0.8903024737020199
Label - sttl: 0.9044106464509674
Label - ct_state_ttl: 0.8736948424214096
```

Figure 21 Highly Correlated Features

The results show that numerous numerical features are highly associated with one another. For example, the correlation coefficient between “sloss” and “sbytes” is 0.95, indicating a strong positive correlation between these two features. Similarly, the correlation coefficient between “dloss” and “dbytes” is 0.99, whereas it is 0.96 for “ct_srv_dst” and “ct_srv_src.” Furthermore, some features are significantly connected with two or more other features, such as “Dpkts,” which is highly correlated with both “dloss” and “Spkts,” with correlation values of 0.99 and 0.82, respectively.

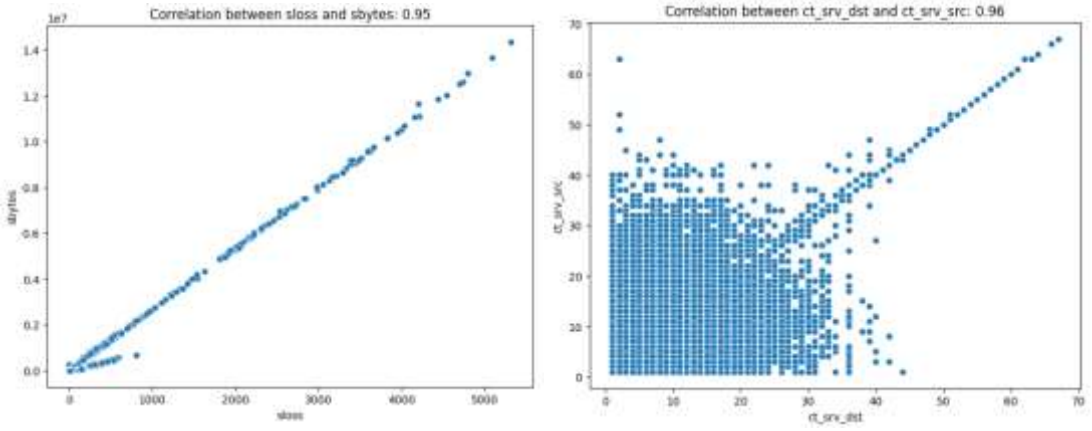

Figure 22 Correlation Plot

4.4 Principal component analysis (PCA)

PCA is a technique used to reduce the dimensionality of data. This technique involves identifying patterns in data and then translating the data into a space with fewer dimensions (Volpi, 2020). The goal is to maintain as much diversity in the original data as possible. Simply put, these assists in identifying the major factors that account for the dataset's variability. For this task, the PCA was used on the UNSW dataset. The Vector Assembler was created to assemble all data columns into a single feature column. The data are then reduced to two dimensions using PCA with k = 2. The resulting two dimensions are saved in the pca_features column, as shown in Figure 23.

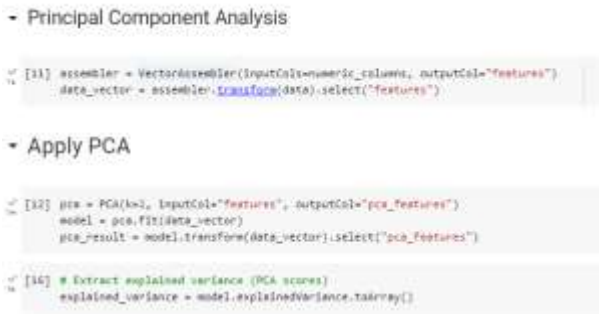

Figure 23 PCA code

The explained variance, which is a measure of how much variation in the original dataset is retained by each primary component, is calculated. The first principal component (PCA 1) covers 99.88% of the variation in the UNSW dataset, whereas the second principal component (PCA 2) captures only 0.12%. This shows that the first principal component can explain the majority of the variation in the dataset and that the second principal component is less important in explaining the variation in the data, as shown in Figure 24 by the scatterplot which used to visualise the data in the new two-dimensional space created by PCA.

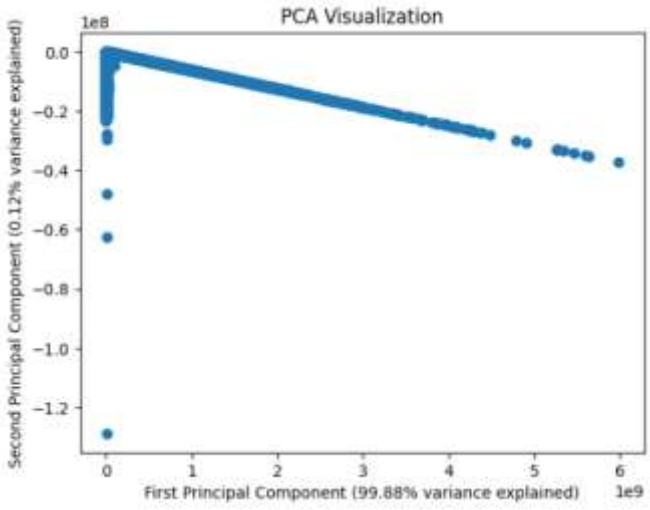

Figure 24 PCA Component Scatter Graph

5. Classification Model

Classification is a supervised learning strategy that uses a categorical variable as the goal variable. It involves using a labelled dataset to train a model and predicting the class label for future cases. The RF algorithm is an ensemble learning strategy that builds numerous decision trees and aggregates their output to enhance overall prediction accuracy and avoid overfitting. This method has several advantages, including noise resistance, overfitting resistance, and the capacity to handle large datasets with many features. As a result, the RF technique is commonly utilised for classification tasks such as network intrusion detection. The utilisation of classification algorithms is critical in enhancing the efficacy of IDS by successfully differentiating between distinct categories of activity. For this task, our primary goal is to create classifiers for binary and multi-class classification using the RF method, a powerful ensemble learning approach that has demonstrated amazing precision and durability in a variety of ML scenarios.

5.1 Binary classification

The classification issue in network intrusion detection can be treated as a binary or multi-class problem. The algorithm differentiates between two classes: normal traffic and attack traffic. The initial step is to generate an index for the categorical target variable, which is then used to create a feature vector with numeric features. Then, the dataset is partitioned into an 80% training set and a 20% testing set using a randomised seed. The RandomForestClassifier is created using features and indexed target columns. A parameter grid is formulated to assess various permutations of the number of trees (numTrees) and the maximum tree depth (maxDepth) (RandomForest—PySpark 3.4.0 Documentation, n.d.) to enhance the model's performance. Cross-validation is used to train the model and assess its efficacy on data that has not been previously observed. The results were evaluated using different evaluation metrics, such as accuracy, precision, recall, f1-score, and confusion matrix. Table 1 summarises the results of the binary classification using the Random Forest classifier, with values expressed as percentages.

Table 1. Binary Classification Results

Class	Precision	Re-call	F1Score	Support
Normal (0.0)	**1.00**	**1.00**	**1.00**	**443,683**
Attack (1.0)	**1.00**	**1.00**	**1.00**	**64,623**
Accuracy			**1.00**	**508,306**
Macro Avg	**1.00**	**1.00**	**1.00**	**508,306**
Weighted Avg	**1.00**	**1.00**	**1.00**	**508,306**

By comparing the random forest classifier outputs to the evaluation criteria, we achieved a binary classification accuracy of 99.99%, demonstrating the usefulness of the algorithm for intrusion detection. The RF classifier had a precision score of 99.99%, recall score, and F1-score of 99.99%. These findings demonstrate the classifier's excellent performance in discriminating between normal and attack traffic in the UNSW-NB15 dataset. Figure 25 depicts the confusion matrix for binary classification, which provides more insight into the performance of the random forest classifier. The diagonal elements of the matrix indicate that all the data points are correctly

categorised and there was not a single misclassified data. Overall, the confusion matrix indicates the capacity of the RF classifier to correctly categorise the normal and attack labels on the UNSW-NB15 dataset.

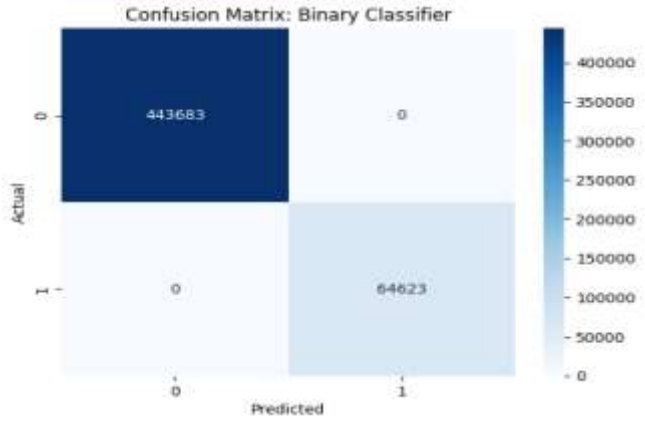

Figure 25. Binary classification confusion matrix.

Figure 26. Binary classification code.

5.2 Multiclassification

Similar to binary classification, the RF classifier was used for multi-classification problems, except that the StringIndexer used to index the category column was changed from "label" to "attack_cat" and VectorAssembler was used to assemble all numerical features into a single "features" column. Using an 80-20 split ratio, the data was then divided into training and test sets. The performance of the model was evaluated using the same evaluation metrics. Table 2 shows that the model achieved an overall accuracy of 0.98, suggesting that it correctly classified most cases in the dataset. Furthermore, the table shows that the model had differing degrees of success in recognising traffic from various classes. While the model performed exceptionally well in identifying classes such as benign, generic, fuzzy, and DoS, it struggled to correctly classify instances of several other classes, such as backdoors and worms, which can also be seen through the confusion matrix, as shown in Figure 27. Overall, the model may not be adequate for identifying specific types of network attacks, and additional study and fine-tuning may be required to improve its performance. It is also worth noting that the class imbalance of the dataset may have contributed to the poorer precision and recall scores for some classes, as some of these classes had on average only 30–500 data points. Overall, the results indicate that the RF can be a useful tool for detecting specific

types of network attacks, but its performance may be limited by the nature of the data and the specific attack types being targeted.

Table 2. Multiclassification Results

Class	Precision	Re-call	F1Score	Support
Benign	1	1	1	443683
Generic	1	0.98	0.99	43458
Exploits	0.57	0.97	0.72	8887
Fuzzers	0.78	0.87	0.83	3839
DoS	0.81	0.02	0.04	3268
Reconnaissance	0.84	0.68	0.75	2445
Fuzzers	0.94	0.18	0.3	1043
Analysis	0.81	0.13	0.22	567
Backdoor	0.7	0.02	0.04	339
Reconnaissance	0	0	0	341
Shellcode	0.76	0.56	0.65	261
Backdoors	0	0	0	95
Shellcode	0	0	0	37
Worms	1	0.02	0.05	43
Macro Average	0.66	0.39	0.4	508306
Weighted Average	0.99	0.98	0.98	508306

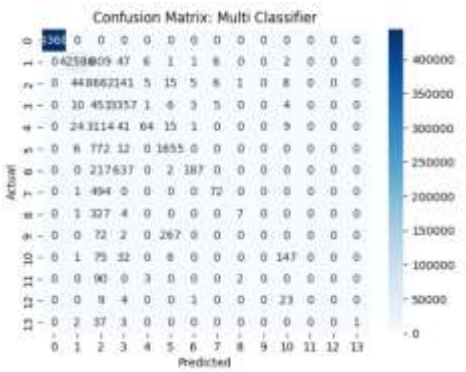

Figure 27 Multiclassification confusion matrix

Discussion

This study reinforces the role of big data analytics in addressing the scalability and accuracy challenges of IDSs. Traditional IDS solutions fail to efficiently manage large-scale traffic; however, the integration of Hive and PySpark provides both computational speed and analytical depth. The RF model delivered near-perfect binary classification accuracy, confirming its robustness for distinguishing normal from malicious traffic. Multi-class results, while strong overall, revealed limitations in detecting minority attack classes such as backdoors and worms. This suggests the need for further work on class imbalance, ensemble methods, or hybrid DL models. The analysis of traffic by source IPs, protocols, and attack categories provides practical insights for security practitioners in prioritising defence strategies.

6. Conclusion

This study demonstrates that combining Apache Hive and PySpark significantly enhances the effectiveness of NID systems. The proposed framework achieved state-of-the-art accuracy in both binary and multi-class scenarios by leveraging the UNSW-NB15 dataset. The findings confirm that big data tools can scale intrusion detection to meet modern network demands while providing interpretable analytical insights. Future research should focus on addressing class imbalance, incorporating real-time detection, and exploring deep learning–based extensions to further strengthen IDS capabilities.

References

Alrawashdeh T, Alhamid M. 2020. Intrusion detection system using machine learning International Journal of Advanced Computer Science and Applications, 11(6), 7-14. https://doi.org/10.14569/IJACSA.2020.0110602

Aminanto, E., Wibisono, H., & Adi, K. (2017). Intrusion Detection System Using Cloud Computing Data Mining Techniques Journal of Telecommunication, Electronic and Computer Engineering, vol. 9, no. 3–8, pp. 43–47.

Chen, X., Li, D., Chen, M., and Zou, D. (2019). Cybersecurity and privacy protection: Survey, taxonomy, and open issues. IEEE Communications Surveys & Tutorials, 21(3), 2333-2370. https://doi.org/10.1109/COMST.2019.2914962

Federal Bureau of Investigation (FBI). (2021). Internet Crime Complaint Centre (IC3) Report 2020. Retrieved from https://www.ic3.gov/Media/PDF/AnnualReport/2020_IC3Report.pdf

Wang, Y., Wang, J., Huang, L., & Yao, X. (2018). Intrusion detection system based on improved GMM algorithm Journal of Physics: Conference Series, 1096, 032023. https://doi.org/10.1088/1742-6596/1096/3/032023

Chowdhury, M., Zaharia, M., Ma, J., Jordan, M. I., and Stoica, I. (2011). Managing data transfers in computer clusters with orchestra. ACM SIGCOMM Computer Communication Review, 41(4), 98-109. https://doi.org/10.1145/2043164.2018448

Moustafa, N. (2015). The UNSW-NB15 dataset. Research Data Australia. Available from: https://researchdata.edu.au/the-unsw-nb15-dataset/1957529

Turney, S. (2022). Pearson correlation coefficient (r) | Guide and examples. Scribbr. https://www.scribbr.com/statistics/pearson-correlation-coefficient/

Volpi, G. F. (2020). The most gentle introduction to PCA. Towards Data Science. https://towardsdatascience.com/the-most-gentle-introduction-to-principal-component-analysis-9ffae371e93b

Li, J., Wu, Y., & Zhang, H. (2021). Deep learning methods for network intrusion detection: A survey. Computers & Security, 102, 102153. https://doi.org/10.1016/j.cose.2020.102153

Shone, N., Ngoc, T. N., Phai, V. D., & Shi, Q. (2022). Hybrid deep learning approach for network intrusion detection Journal of Information Security and Applications, 67, 103182. https://doi.org/10.1016/j.jisa.2022.103182

Zhang, Y., Sun, Y., & Lin, X. (2023). Scalable ML for big data intrusion detection in cloud environments Future Generation Computer Systems, 144, 85-97. https://doi.org/10.1016/j.future.2023.01.005

Alqahtani, A., & Wang, H. (2024). A survey on big data analytics for cybersecurity: Challenges and opportunities. IEEE Access, 12, 5573-5590. https://doi.org/10.1109/ACCESS.2024.3349557