Scientific Research Publishing

Journal of Computer and Communications, 2024, 12, 246-265
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Big Data & DDoS ATTACKS: A Discussion of Ensemble Algorithms to Detect Cyber Attacks

Anja Housden-Brooks

Anglia Ruskin University, Cambridge, UK
Email: anjahousdenbrooks@gmail.com

How to cite this paper: Housden-Brooks, A. (2024) Big Data & DDoS ATTACKS: A Discussion of Ensemble Algorithms to Detect Cyber Attacks. *Journal of Computer and Communications*, **12**, 246-265.
https://doi.org/10.4236/jcc.2024.1212014

Received: September 20, 2024
Accepted: December 27, 2024
Published: December 30, 2024

Abstract

The use of machine learning algorithms to identify characteristics in Distributed Denial of Service (DDoS) attacks has emerged as a powerful approach in cybersecurity. DDoS attacks, which aim to overwhelm a network or service with a flood of malicious traffic, pose significant threats to online systems. Traditional methods of detection and mitigation often struggle to keep pace with the evolving nature of these attacks. Machine learning, with its ability to analyze vast amounts of data and recognize patterns, offers a robust solution to this challenge. The aim of the paper is to demonstrate the application of ensemble ML algorithms, namely the K-Means and the KNN, for a dual clustering mechanism when used with PySpark to collect 99% accurate data. The algorithms, when used together, identify distinctive features of DDoS attacks that prove a very accurate reflection of reality, so they are a good combination for this aim. Impressively, having preprocessed the data, both algorithms with the PySpark foundation enabled the achievement of 99% accuracy when tuned on the features of a DDoS big dataset. The semi-supervised dataset tabulates traffic anomalies in terms of packet size distribution in correlation to Flow Duration. By training the K-Means Clustering and then applying the KNN to the dataset, the algorithms learn to evaluate the character of activity to a greater degree by displaying density with ease. The study evaluates the effectiveness of the K-Means Clustering with the KNN as ensemble algorithms that adapt very well in detecting complex patterns. Ultimately, cross-reaching environmental results indicate that ML-based approaches significantly improve detection rates compared to traditional methods. Furthermore, ensemble learning methods, which combine two plus multiple models to improve prediction accuracy, show greatness in handling the complexity and variability of big data sets especially when implemented by PySpark. The findings suggest that the enhancement of accuracy derives from newer software that's designed to reflect reality. However, challenges remain in the deployment of these systems, including the need for large, high-quality datasets and the potential for

adversarial attacks that attempt to deceive the ML models. Future research should continue to improve the robustness and efficiency of combining algorithms, as well as integrate them with existing security frameworks to provide comprehensive protection against DDoS attacks and other areas. The dataset was originally created by the University of New Brunswick to analyze DDoS data. The dataset itself was based on logs of the university's servers, which found various DoS attacks throughout the publicly available period to totally generate 80 attributes with a 6.40GB size. In this dataset, the label and binary column become a very important portion of the final classification. In the last column, this means the normal traffic would be differentiated by the attack traffic. Further analysis is then ripe for investigation. Finally, malicious traffic alert software, as an example, should be trained on packet influx to Flow Duration dependence, which creates a mathematical scope for averages to enact. In achieving such high accuracy, the project acts as an illustration (referenced in the form of excerpts from my Google Colab account) of many attempts to tune. Cybersecurity advocates for more work on the character of brute-force attack traffic and normal traffic features overall since most of our investments as humans are digitally based in work, recreational, and social environments.

Keywords

K-Means Clustering, The KNN Algorithm, PySpark, Ensemble Learning Methods, DDoS Attacks, Veracity, Malicious Traffic Alert Systems

1. Introduction

If one considers a popular online gaming platform like StarCraft hosting a highly anticipated tournament, when going live, players and viewers alike could find themselves unable to connect whilst experiencing disconnections. This may well be due to a DDoS attack targeting the game server by flooding it with bogus traffic. DDoS or Distributed Denial of Service attacks are malicious attempts to disrupt online services by overwhelming servers with a flood of traffic from multiple sources.

"These attacks render the targeted services unavailable to legitimate users, causing downtime, financial losses, and reputational damage to businesses. Unlike data breaches or malware infections, DDoS attacks aim to disrupt services" [1] rather than ruining data. This is achieved by swamping the target with mass traffic, making sites inaccessible to legitimate users. The motivations range, for example, could realise as a "competitor … launch[ing] a DDoS attack against a rival's website during a peak sales period to disrupt their business" [1] and gain an advantage over them.

Over time, the world has transformed from the technologically superior landscape at our feet today. People are now dependent on the internet. Gadgets are constantly interconnected to the digital ecosystem. The internet is a global information source. Despite this, cybercriminals' attacker knowledge has strengthened

and resulted in different methodologies being used to access company data stores, which is one example of a cyberattack. Moreover, DDoS attacks remain the most effective methods used by criminals to cause substantial damage to organizations globally in terms of operational, reputational, and financial problems. The victim's network is flooded with massive illegal traffic, hence denying genuine traffic from passing through for authorized users [2].

Recent data shows a significant evolution in the landscape of Distributed Denial of Service attacks, "with trends observed in 2024". Due to rising sophistication and volume in the first half of 2024 [3]. Cloudflare mitigated 8.5 million DDoS attacks, marking a 20% increase compared to the same period in 2023 [3]. The attacks are becoming increasingly sophisticated, leveraging advanced techniques that were once the domain of state-sponsored actors but are now accessible to ordinary cybercriminals, which means the data should be read as accurately as possible to gain control of the problem. The average size of these attacks has also grown, with many targeting critical infrastructure and services. This is why size to time ratio is important. Shifts in attack strategies can be seen through the facts and figures summed up here: "Although the total number of DDoS attacks has decreased by about 54.7%, the average attack size has increased by over 233% compared to the previous year" [3]. This suggests a strategic shift towards fewer but more impactful and, therefore, complex attacks.

K-Means Clustering, which is used with the KNN algorithm, incorporates the data accounting for the individual differences in attack profiles and creates grounds to recommend tailored intervention to precise degrees. This is due to the ability of the latter to cross-validate, a weakness of the K-Means, which works best on less complex datasets and can efficiently collate centroids but shows most effectiveness in preparing data for the application of the KNN in the instance of organizing big and complex datasets. Catching subtle and low-rate DDoS attacks that may elide threshold-based detection is also of interest. K-Means Clustering with the KNN enables real-time analysis of multiple traffic features simultaneously revealing the bursts of impactful DDoS attacks and implying idioms of character due to the excellent density range detection. Can K-Means clustering combined with the KNN and PySpark find new patterns that might be of interest to the wider body of work on DDOS Attacks? This study proves they do.

Cloud Computing says DDoS attacks are "A flood of traffic from users who share a single behavioral profile, such as device type, geolocation … unexplained surge in requests to a single page or endpoint … odd traffic patterns such as spikes at odd hours of the day or patterns that appear to be unnatural (e.g., a spike every 10 minutes)" [3]. Algorithms have already identified indicators, but more can be sought.

ML algorithms can analyze large volumes of network traffic data to identify patterns and anomalies connected with DDoS attacks. This allows for better accuracy differentiation between more severe attack spikes that could cause the company grave problems and less damaging ones to normal traffic. Furthermore, ML

models can process network data quickly, enabling timely identification of the ongoing issue of these attacks. The benefit of having as much information built into a generative system is a possibility for businesses and YouTubers alike to immediately mitigate responses through fast detection. By learning from historical data, machine learning approaches aim to minimize false alarms from less threatening traffic but create contingent computerized initializations when facing problematic attacks.

The objective then is to develop smart algorithms for detecting the character of attacks based on K-Means Clustering when used with the very versatile KNN as a double algorithm launched on the dataset on a PySpark foundation.

2. PySpark

Additionally, PySpark creates the environment for the data to organize before K-means and the KNN can successfully cluster which offers several key benefits for big data analysis through machine learning in terms of scalability and performance [4]. PySpark provides distributed computing capabilities, allowing K-means clustering and the KNN to be performed on very large datasets that would prove troublesome in terms of veracity and volume had PySpark not been used firstly.

For efficient data processing, PySpark's distributed data engine efficiently handles data preparation, feature engineering, and other preprocessing steps [5] before applying K-means and the KNN for the clustering process. This streamlines the entire workflow from raw data to supervised results. Moreover, iterative attempts to control the size of the data eventually leads to algorithm optimization in K-means which then translates to a more precise solution as opposed to taking giant leaps. Incremental Python code used in Colab pre-tunes centroid distributions through numeric relevance by using the greater or smaller than operand to highlight packet amount according to time duration thus highlighting differences in normal traffic and bruteforce attack traffic.

PySpark's in-built computing allows for fast iterations when refining [labels] [4]. This speeds up the overall clustering process when the algorithms are launched in large datasets. PySpark's MatplotLib library also provides a computerized canvas for the visualization of a K-means to KNN implementation that integrates seamlessly with other data processing steps [6]. PySpark allows K-means to be incorporated into more complex ML workflows. "K-Means is an unsupervised learning algorithm that clusters data into groups. You pass the algorithm some data and specify how many groups you would like it to find. It will then try to split the data into groups Furthermore, its streaming enables real-time clustering on streamed data" [7] which allows for dynamic multi-modal clustering of data points and attendant updating of cluster assignments on the quick. Moreover, parallel model training is also compatible with the K-means models with different parameters as in the centroids to average ratio and can be trained via parallel leveraging PySpark's distributed computing [8] which clearly enables more efficient parameter tuning.

3. Literature Review

Several studies have demonstrated the effectiveness of ensemble methods for DDoS detection and thus are a powerful approach for detecting attacks, offering improved accuracy and robustness compared to single machine learning models. This literature review re-examines recent research on ensemble methods for DDoS attack detection. When under attack, the system remains busy with false requests rather than offering usual services to legitimate agents. These attacks have been increasing with time and are now more complex. Consequently, it has become difficult to detect the character of attacks and thereby work to secure servers.

The dataset is characterized by contemporary types of attacks such as forward flows, packets in the forward direction, average number in bytes in the forward direction, average number of bytes in the backward direction with many other tabulated titles or various relevance. It was decided that the columns: flow duration and packets in the forward Direction were important what with the segmenting deployment capability of the binary label column in this study which binarized the data into malicious and non-malicious traffic according to preprocessed parameters.

"WEKA used to classify various types of attacks. It has been observed that J48 algorithm produced best results as compared to Random Forest and Naïve Bayes algorithms" [7]. J48 is an implementation of the C4.5 decision tree algorithm developed by Ross Quinlan. It is used to generate a decision tree for classification tasks. "J48 is an extension of the earlier ID3 algorithm, with additional features to address limitations of ID3" [8].

Random Forest, an ensemble of decision trees, has shown promising results in DDoS detection. A study by Das *et al.* [9] used an ensemble of unsupervised machine learning algorithms, including Random Forest, to implement an intrusion detection system with high accuracy for detecting attacks. The authors found that the ensemble approach outperformed existing methods and provided better generalization. The proposed approach "is validated using datasets with various modern type of attacks such as HTTP flood, SIDDoS and normal traffic" [10].

In a study where twelve clustering ensemble algorithms are compared to choose a basic, most efficient one, the k-means, with different initializations as generative mechanism and average-linkage as the consensus function was alone good although the dataset was smaller than the one used here. Generally, studies so far show that when ensemble size increases, the performance improves especially when data is of a similarity matrix [11] of a similarity matrix to partition data into clusters.

The K-means searches for its nearest neighbour but can be compromised by outliers unlike the KNN which more easily learns from patterns and is used by streaming giants Facebook and Spotify [12]. Generally, studies so far show "as ensemble size increases, the performance of clustering ensemble improves" [13]. Unilateral clustering ensemble algorithms are better than narrow clustering due to the influence of diversity on the clusters as instructive to "selecting members"

[13].

Using PySpark's in-built MatPlotLib for visualisation byway clustering ensemble methods are touched upon in one comprehensive paper that develops a multiple source dataset and observes its visualization utilizing PySpark to detect outliers and their possible meaning when applying the clustering algorithms. Instead of devoicing the outlier, the paper demonstrated that "the Kafka [used] as a message … [with] flat incremental clustering is done to develop a frame of the normal activities based on the dataset for training. The clusters are produced dynamically and are not fixed" [14]. In this case the classification is leveraged by the meaning of the outlier which PySpark allows.

Towards advances in use of sensor technology, it is now possible to acquire "hyperspectral data simultaneously in hundreds of bands" [15]. The new algorithms can both reduce the dimensionality of the initial data set by handling highly correlated bands and exploit the information in these data sets very effectively. The authors of the paper promote a set of feature extraction algorithms that are simple yet fast, especially effective for classification of high dimensional data which I believe posed a problem in this study for acquiring decent visual data that refers to Brute-Force traffic and not just Benign which was achieved yet was good for displaying the ability for complex and accurate visualization. The techniques deployed by these algorithms intelligently mix subsets of adjacent bands into a smaller number of features as is the usual case with the KNN which did provide excellent centroid partitioning as shown below using an example of normal traffic features and achieved 99% accuracy, providing very good results that can also be interchanged for the inspection of malicious traffic.

In the study top-down and bottom-up algorithms recursively partition bands into two—not particularly equal—sets of bands before replacing them by the mean values. "The bottom-up algorithm builds an agglomerative tree by merging highly correlated adjacent bands and projecting them… yielding high discrimination among classes" [15]. The latest algorithms find variable length "localized in wavelength, [and] favor grouping highly correlated adjacent bands that, either by taking their mean or … linear projection, [and thus] yield maximum discrimination" [15], The aforenoted study was conducted on the AVIRIS data set for a 12-class problem and reveal improvement in accuracy while using a smaller number of features.

Amin Karami's paper on the matter of preventing severe DDoS attacks in the current techno-climate looks at Information-Centric Networking (ICN), receiver-driven paradigm shifts in the broader tech horizon—and its relationship with defense—against new forms of potentially unknown attacks with the focus on privacy. Blocking malicious network traffic, thus limiting damage. DDoS attacks translate to many problems such as congestion control, and cache pollution. "In order to protect NDN infrastructure, we need flexible, adaptable and robust defense systems which can make intelligent—and real-time—decisions to enable network entities to behave in an adaptive and intelligent manner" [16]. Computational

intelligence materializes in "adaption, fault tolerance, high computational speed" [16] make them suitable.

To the matter of unsupervised data or semi-supervised machine learning can be used for obtaining categories of unlabeled or partially labeled data based on applicable metrics based on dissimilarity. Further on, the data is completely assigned the labels as per the algorithms' observed differentiation. "The features are taken for victim-end identification of attacks and the work is demonstrated with three features … using agglomerative and K-means … to label the data and obtain classes to distinguish attacks from normal traffic" [17]. In a study, K-Nearest Neighbors (the KNN), Support Vector Machine (SVM) and Random Forest (RF) [were] implemented to obtain classification. "The KNN, SVM and RF models in experimental results provide 95%, 92% and 96.66% accuracy scores respectively under optimized parameter tuning within given sets of values" [17]. Ultimately, the schema is also validated using a subset of benchmark dataset with new vectors of attack.

Numerous studies have focused on developing machine learning (ML) classifier and artificial intelligence (AI) assistance for the prediction so one can capture traffic data. Several prominent clustering algorithms have been employed in the context of DDoS attack detection, including the ones here.

K-means clustering is one of the most widely used and partitions data into K clusters by minimizing the sum of squared distances between data points and their respective cluster centers. Algorithms constructs a hierarchy of clusters, starting from individual data points and merging them into larger clusters based on similarity measures. Additionally, density-based methods like the KNN identify clusters based on regions of high density in the data. Traffic classification through clustering algorithms is used to classify network traffic into different categories of Normal traffic which is what was successfully achieved in this paper illustrating behavior if the centroids are intelligibly and logically organized in the preceding preprocessing stage. It is also worth noting that by clustering IP addresses or other network identifiers, clustering algorithms can help identify the sources of DDoS attacks and can be used to group similar DDoS attack patterns in proximation to the origin, enabling the selection of appropriate mitigation strategies.

Challenges and limitations exist, however. These belong to the area of parameter selection where preprocessing should be at once logical. The performance of clustering algorithms often depends on the selection of appropriate parameters. One has to know their dataset well to get it right as scalability may prove problematic. Network traffic volumes continue to grow, and clustering algorithms must be scalable to handle large datasets efficiently in the pursuit of accuracy as the presence of outliers in network data can degrade the accuracy if not dealt with in the tuning stages.

To address the challenges and limitations of traditional clustering algorithms, researchers have explored ensemble approaches that combine clustering with anomaly detection, machine learning, and deep learning. Additionally, future research

directions include areas of importance such as real-time detection, by developing clustering algorithms that can detect DDoS attacks on the quick to minimize damage. Adaptive clustering points to changing network conditions and evolving attack patterns. Further integration with network security systems is also a wider aim of clustering ensemble algorithms via existing network security systems to enhance their effectiveness. To this, leveraging cloud-based solutions is of interest. Cloud computing platforms scale clustering algorithms and improve their performance.

In the realm of DDoS Detection and severity, various machine learning models have been employed to achieve accurate results. For instance, adopting the feature selection technique with SVM classifier for early detection, and their results were compared with random forest (RF), naive Bayes (NB), decision tree (DT), and KNN classification models.

In early detection of diabetes disease Aminah *et al.*, used k-nearest neighbor (KNN) classifier model was used and the result was compared with that of the support vector machine (SVM) model achieving 85.6% accuracy. The results comparison was made with another machine learning classifier called SVM to authenticate the work.

By further combining PySpark's big data capabilities with advanced ensemble clustering, data scientists and analysts can perform precise analysis on massive datasets in a scalable and efficient manner that *speak with* accuracy. This unlocks insights that may not be possible with traditional single-machine implementations.

In another study, algorithms were implemented on PySpark to focus on the details and performance of algorithms on different distributed system setups for estimating future demand and sales. Insightful information for strategic decision-making, resource allocation, and inventory control which cordons anomalies for tabular view. In utilizing the selected machine learning algorithms for constructing a graphic illustration with the help of MatPlotlib enables a bespoke approach to visualizing systems that better fit the company/institution/streamer in the protection against malware. PySpark is used for its "scalability, parallel processing … and seamless integration with … MLlib [as] a powerful machine learning library … offering a comprehensive set of scalable and distributed algorithms that enable efficient processing and analysis of large datasets" [18]. and gain insight into solutions to fit the changing shifts in the technological climate.

4. Preprocessing and Collection: Methodology and Visualizations

Machine learning/AL algorithms used for model development.

Data Collection

The challenge is the volume and the veracity due to missing values and null values and other customizations. The dataset is about 600 mb and comprises of much schema. Here one installs the latest release of Pyspark in Google Colab and run it. It is important to consider spark as an interface between dataframe, caching

mechanism and structuring data.

As you can see below checking memory is very important when using big data before we connect to Google Drive and upload the DDoS Dataset

Code 1.

```
!df -h
from google.colab import drive
drive.mount('/content/drive')
!cat /proc/meminfo
!pip3 install pyspark
#To check the available memory
!free – h
#To check the available cores
! nproc
#Linking with Spark
#connect to google drive for    dataset
from google.colab import drive
drive.mount('/content/drive')
(dataframe)
df = spark.read.format('csv').load("/content/drive/MyDrive/Amin's dataset.csv", inferSchema = True, header = True)
```

Preprocessing

Moving data to disc and sending data over to memory is going to reduce performance but this problem occurs primarily with big datasets so we need have to be very careful about compression in order to reduce data. Initializing Spark shows the degree of memory in cores so one knows what we're dealing with in terms of processing. In format you have to explicate what data we are reading which is a csv file in this situation.

Once our data is uploaded.

Code 2.

```
Code continued:
df.rdd.getNumPartitions()
spark.conf.set('spark.sql.files.maxPartitionBytes', 128 * 1024 * 1024)
Data Source Input Split.
Default Partition size (128MB, text, csv, json, parquet)
Resource Allocation
Data Locality
```

Non-file based data (NonSQL, Amazon kinesis, Amazon DynamoDB, Google Bigtable, Amazon S3, Google Cload, Neoj4, Amazon Redshift): predicates or column-based partitioning.

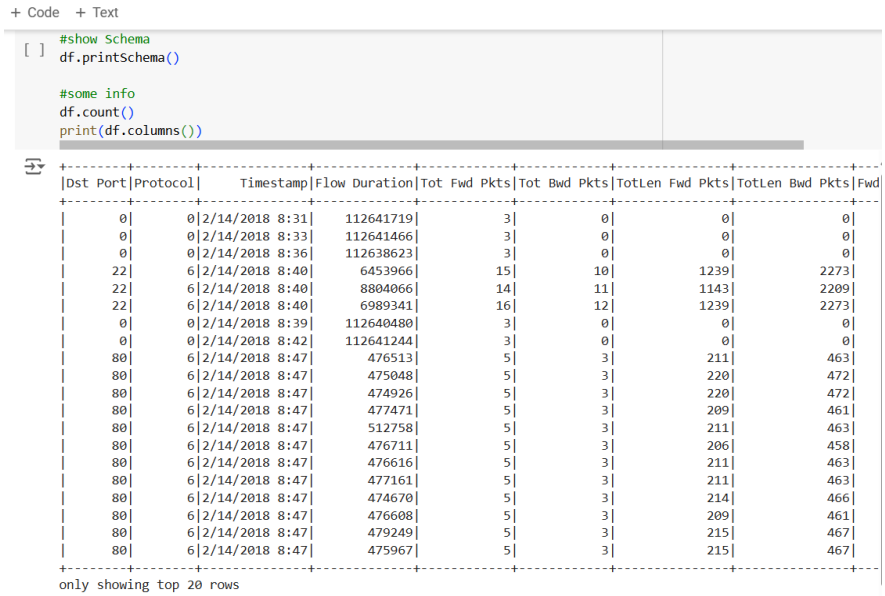
```
Custom Partitioning
#Increase the number of partitions
df2 = df.repartition(6)
df2.rdd.getNumPartitions()
```

6

df3 = df.repartition('Bwd Pkt Len Min')

df3.rdd.getNumPartitions()

Looking at the data we can see we have null values and data that has zero efficacy to the aim of finding patterns that allude to particularities. (see Figure 1) In order to process this data, we need to get the number of partitions. Ideally, each partition should be roughly 128 megabytes but might not evenly distribute data. PySpark has different sized partitioned files depending on format, but one can precondition partition size which was not necessary here because Spark took best route.



```
+ Code  + Text

[ ]  #show Schema
     df.printSchema()

     #some info
     df.count()
     print(df.columns())
```

Dst Port	Protocol	Timestamp	Flow Duration	Tot Fwd Pkts	Tot Bwd Pkts	TotLen Fwd Pkts	TotLen Bwd Pkts	Fwd
0	0	2/14/2018 8:31	112641719	3	0	0	0	
0	0	2/14/2018 8:33	112641466	3	0	0	0	
0	0	2/14/2018 8:36	112638623	3	0	0	0	
22	6	2/14/2018 8:40	6453966	15	10	1239	2273	
22	6	2/14/2018 8:40	8804066	14	11	1143	2209	
22	6	2/14/2018 8:40	6989341	16	12	1239	2273	
0	0	2/14/2018 8:39	112640480	3	0	0	0	
0	0	2/14/2018 8:42	112641244	3	0	0	0	
80	6	2/14/2018 8:47	476513	5	3	211	463	
80	6	2/14/2018 8:47	475048	5	3	220	472	
80	6	2/14/2018 8:47	474926	5	3	220	472	
80	6	2/14/2018 8:47	477471	5	3	209	461	
80	6	2/14/2018 8:47	512758	5	3	211	463	
80	6	2/14/2018 8:47	476711	5	3	206	458	
80	6	2/14/2018 8:47	476616	5	3	211	463	
80	6	2/14/2018 8:47	477161	5	3	211	463	
80	6	2/14/2018 8:47	474670	5	3	214	466	
80	6	2/14/2018 8:47	476608	5	3	209	461	
80	6	2/14/2018 8:47	479249	5	3	215	467	
80	6	2/14/2018 8:47	475967	5	3	215	467	

only showing top 20 rows

Figure 1. PySpark organised dataset.

Feature Extraction

Code 3.

Here we have partitions that need to decrease because there are so many. One re-creates data frame to 10 partitions. Compiling information to less partition's shuffles data based on columns which ultimately finalise with label which is binary, and after preprocessing will augment the data into benign and brute force traffic.

Repartitioning by Range will evenly distribute data across the column by the column Flow Duration as that is the dependant variable to measure packet influx shown in the example below which is important because attack traffic is proven to disseminate mass traffic by flow time

df6 = df.repartitionByRange(10,'Flow Duration')

df6.rdd.getNumPartitions()

#Get rid of all 'benign' data, but keep HTP Bruteforce due to its relevance to actual attacks

```
df6.write.option('header', T
rue) \
        .partitionBy("Label") \
        .mode("overwrite") \
        .csv("Label_df6")
df5.write.option('header', True) \
        .partitionBy("Label") \
        .mode("overwrite") \
        .csv("Label_df5")
df4.write.option('header', True) \.....
```

This element was returned to regularly and recalibrated to be correct. Dimensionality Reduction exists in multimodal types like flat mixed with integer and float and must be orchestrated properly to reduce the number of features in the DDoS dataset' while retaining as much of the important information as possible.

```
df3.write.option('header', True) \
        .partitionBy("Label") \
…….
df3.select("Flow Duration","Fwd Pkts/s").
+-------------+---------------+
|Flow Duration|         Fwd Pkts/s|
+-------------+---------------+
|                798|1253.1328320802|
```

Getting more familiar with data is inherent.

Flow Packets Rate is the number of packets transferred per second.

Creating the Conditions

The aim is to repartition organized data by the Label column to structure the entire data into two segments showing benign and brute force traffic character. Repartition over Label Column. It is important to have a healthy number partitions based on Label column so column size is correlative diversity**.** In other words, if its structured as non-binary – for example by using Flow Duration - then it will produce too many potentially unorganized values. Additionally, every null value is kept is a corresponding number co-exists within the row.

Continue to tune

```
df3.filter((df3["Flow Duration"] > 1000)). show(10)
df3.filter((df3["Flow Duration"] > 10000)).
```

MEAN appropriation

```
from pyspark.sql.functions import mean, sum, round, col #import col function
df4.groupBy('Label').agg(
        round(mean('Flow Duration'), 2).alias("mean_Flow Duration"),
        sum('Flow Duration').alias("sum_Flow Duration"),
        ).show(10)
#SORT
```

df3.sort(col('Pkt Len Max'), col('Flow Pkts/s'), ascending = False).show(10)

AGGREGATE AVERAGE OF FLOW DURATION

```
+--------------+------------------+-----------------+
|         Label|mean_Flow Duration|sum_Flow Duration|
+--------------+------------------+-----------------+
|SSH-Bruteforce|         183349.68|      34394383592|
|        Benign|        9773470.56|    6525023054919|
|FTP-BruteForce|               3.8|           735386|
+--------------+------------------+-----------------+
```

from pyspark.sql.functions import col,count, when

missing_values = df3.select([count(when(col(c).isNull(), c)).alias(c) for c in df3.columns])

missing_values.show(10)

……..

SETTING PARAMETERS (see Figure 2)

df_copy = df3.select("*")

df_filled = df_copy.fillna(subset=['Flow Duration'], value= 10000)#Fill NaNs in 'Flow Duration' with 10000

df_filled.show(40)

Min","Bwd URG Flags…….

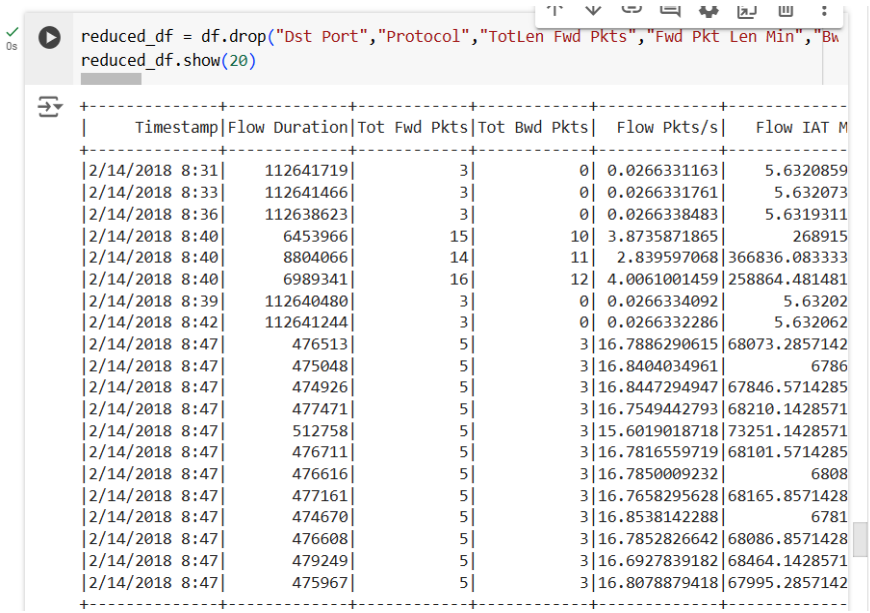


Figure 2. Flow duration.

Change values to Integer

from pyspark.sql.functions import collect_list

Select columns with float data type

float_cols = [f.name for f in df.schema if f.dataType.typeName() == 'float']

SUCCESS! FILE EXISTS - PROCEED WITH ANALYSIS.

```
import pandas as pd
from sklearn.cluster import KMeans
```

RUNNING K-MEANS CLUSTERING ALGORITHM TO NEW DATASET.

```
# Define the features you want to use for clustering
features = ["Flow Duration", "Flow Pkts/s", "Fwd Pkts/s", "Label"]
# Standardize features(important for K-Means)
scaler = StandardScaler()
Dimensionality Reduction:
# Convert selected features to numeric, handling non-numeric values
# Coerce errors to NaN and fill with 0
# Use PySpark's to_numeric function to convert columns to numeric type
for feature in features:
    df3 = df3.withColumn(feature, F.col(feature).cast("double"))
#Fill null values with o
df3 = df3.fillna(0, subset=features)
#Handle infinite values with Nans before scaling
……
scaled_features = scaler.fit_transform(df3_pandas[features])
# Apply KMeans Clustering
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(scaled_features)
# Add cluster label to original data
# Assign the result of kmeans.labels_to a new column in df3
df3_pandas['Cluster'] = kmeans.labels_ # Add cluster labels to the Pandas DataFrame
  super()._check_params_vs_input(X, default_n_init=10)
```

Cluster	Flow Duration	Flow Pkts/s	Fwd Pkts/s	Label
0	9.770899e+06	5.644587e+04	28917.946975	0.0
1	1.562041e+00	1.451033e+06	769551.414899	0.0
2	-7.100713e+11	-3.485627e-04	0.000000	0.0

VISUALIZATION

```
!pip install matplotlib
import matplotlib.pyplot as plt
# df_3pandas is your Pandas Dataframe with cluster labels
plt.figure(figsize=(8, 6))
plt.scatter(df3_pandas['Flow Duration'], df3_pandas['Label'], c=df3_pandas['Cluster'], …….
plt.show()
Analyse clusters
Results show variance problems.
```

CREATING SCATTERPLOT

```
!pip install seaborn
```

```
import seaborn as sns # Added this line import the seaborn library
# Install necessary libraries
!pip install scikit-learn
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
Implementations of the KNN
```

Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.5.0)

Accuracy: 0.9995994564051213:

Accuracy refers to how often the model correctly predicts the class label for a given input. Here the intermix of K-Means to reorganise and then KNN model discriminating a prediction of the right label for 99 out of 100 examples means accuracy is 99%.

Formula: Accuracy=Number of Correct PredictionsTotal Number of Predictions×100\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \times 100Accuracy=Total Number of PredictionsNumber of Correct Predictions ×100

```
# Load the dataframe
df = pd.read_csv('Label_df3.csv')
# Select features and target variable
features = ['Flow Duration', 'Flow IAT Max', 'Init Fwd Win Byts']
target = 'Label'
X = df[features]
y = df[target]
# split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
# Scale the features using StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
# Initialize the KNN classifier
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
# Make predictions on the testing set
y_pred = knn.predict(X_test)
# Create a scatter plot
# Select two features for visualization
feature1 = 'Flow Duration'
feature2 = 'Flow IAT Max'
```

Clearly the Data only shows characteristics of Benign Traffic, a testimony to accuracy of ensemble methods. The scatterplot allows for a clear spatial repre-

sentation of data points, which is crucial for understanding KNN's distance-based approach. (see Figure 3) Each point on the plot represents a data instance, with its position determined by feature values. The dense areas can indicate vulnerable times of the attack traffic due to large scope for opportunity, but it is the desired aim to visualize the brute force traffic if one is interested in Malware.

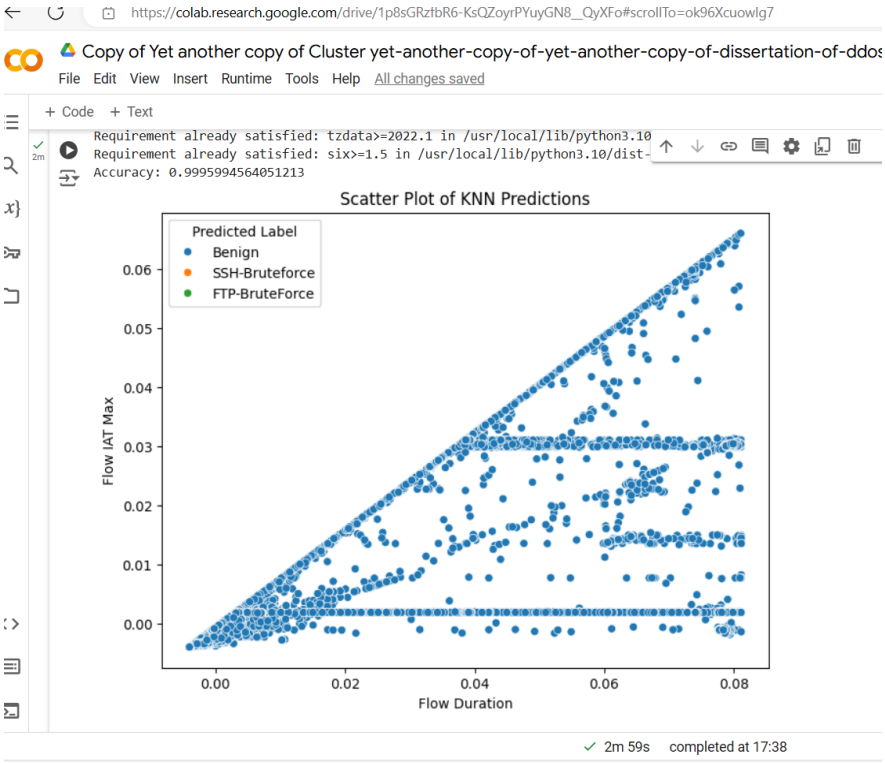


Figure 3. Scatter plot of KNN predictions.

5. ML/AI Model Development

A KNN Clustering Model will act through its ability to cross-validate data. The K-Means clustering alone as shown above can fall short of certain datapoints due to the computational intensitivity it might pose. This is because it searches for its nearest neighbour but can be compromised by outliers unlike the KNN which more easily learns from patterns and is used by streaming giants Facebook and Spotify [12]. By adding an extension to the algorithm. The results will be developed according to the best accuracy gleaned preprocessed data from multiple sources and provide a baseline guidance in creating software through Ml algorithms.

The key of the algorithm is to select the best K, that is the best neighbours to obtain the best prediction. For example, a low k produces variance that is higher as shown above, whereas a high k can produce more biased outcomes. Finding the best k is only possible through testing the effect of different values to enhance cross-validation.

To design a Brute Force Alert Software, it is important to understand the nature of attacks.

A brute force attack is a trial and error expedition with the aim of guessing passwords or encryption keys by strategically trying every possible combination. To design an effective alert system, one needs to understand common indicators of such attacks.

Key Indicators of Brute Force Attacks:

Abnormal login attempts or an anomolized spike in failed login attempts from a single IP address or multiple IP addresses in a short period; unusual access patterns; attempts to log in using common passwords and strange login times. Slow system performance and attendant significant decrease in the said system performance arise due to excessive login attempts. Network congestion from increased network traffic from specific IP addresses is another indicator.

Design Considerations for a Brute Force Alert Software

Real-time monitoring means that the software would continuously monitor system logs and network traffic for suspicious activity. Threshold-based alerts will ensure this and can be set within the categories of as an example - greater than 10,000 pkts/s per - highlighted in the coding subsection of this experiment. Set thresholds for failed login attempts, abnormal access patterns, or network congestion are also sub-features of importance in system design. When these thresholds are exceeded, the software should trigger an alert. IP address tracking should be kept track of. IP addresses that have attempted multiple failed logins are a cause for alert.

Alert System Algorithm:

1) Collect data through continuously monitor system logs and network traffic for login attempts.

2) Analyze data to identify patterns of abnormal login attempts, such as multiple failed attempts from the same IP address or unusual login times.

3) Compare against thresholds to compare the observed patterns against predefined thresholds.

4) Trigger alerts if the thresholds are exceeded, send alerts to designated recipients.

5) Update thresholds to adjust thresholds based on historical data and current security trends.

Evaluation of a proposed System

Traffic monitoring will implement real-time data collection from DDOS Attacks so naturally the aim would be to set up a system to continuously collect and analyze incoming traffic data. Calculating baseline traffic is of vital importance to establish context. Establishing baseline traffic patterns over a specific time period is gauged according to flow duration and flow.

An example of a Spike Detection Algorithm based on statistical analysis such as z-score analysis, to detect anomalies in traffic patterns.

Code:

python

```
def calculate_z_score(current_value, mean, std_dev):
```

return (current_value - mean) / std_dev

Set Deviation Threshold

Define a threshold for what constitutes a significant deviation (e.g., 3.5 standard deviations).

python

DEVIATION_THRESHOLD = 3.5

Minimum Request Threshold

Implement Request Count Check

Add a condition to ensure the spike involves at least 1000 requests.

MIN_REQUEST_THRESHOLD = 1000

…… return z_score > DEVIATION_THRESHOLD and (current_requests - mean_requests) >= MIN_REQUEST_THRESHOLD

Combine the spike detection and minimum threshold check to trigger alerts.

python

6. Experimental Results

Clustering algorithms are unsupervised machine learning techniques that group similar data points together based on predefined distance metrics. These algorithms are particularly useful for identifying patterns and anomalies within large datasets proving themselves by assisting the accrual of 99% accuracy in this study.

K-means clustering is one of the most widely used clustering algorithms which partitions data into K clusters by minimizing the sum of squared distances between data points and their respective cluster centers. Spectral clustering leverages the "eigenvalues and eigenvectors" [11] of a similarity matrix to partition data into clusters.

As shown in the scatterplot, Anomaly detection is demonstrated in the density of the clusters and detect deviations that likely indicate anomalous of benign and malicious traffic activity, such as problematic DDoS attacks. The traffic classification ability shows the collection of density which lead to conclusions of prominent traffic.

The data shows network traffic data with features: "Flow Duration", "Pkt Len Max", "Label", "Flow Pkts/s" and indicate the parameters of malicious traffic compared to null values across the Excel charts. The code performs various operations on this data, including data loading and exploration by loading data from a reduced DDoS CSV file, showing its schema, and calculating basic statistics. The algorithm cleans by handling missing values by dropping rows or filling them with constant values. Data reduction is enacted by removing irrelevant columns to simplify the dataset. Data transformation is imparted through the conversion of data types, aggregating data, and sorting it.

The K-Means Clustering algorithm groups similar data points. Classification and training as explained was further enacted by a K-Nearest Neighbors (KNN) model to predict the "Label" column outcomes as it pivoted the entire dataset and instructed features. Visualization allows us to see clusters and to thereby create

model predictions for system updates.

The model stands at 99% percent accuracy after the implementation of the second algorithm: the KNN after the K-means to hone the centroids by virtue of reaching a cross-validated accuracy of 0.99. As expected, the algorithm has reached an almost perfect classification when the code was augmented to equalize the scales and take away correlations between features.

7. Conclusions

By selecting features from big diversity data for low-dimensional data frames that focus on idiosyncrasies to answer a common research question like how do we achieve the most accuracy? The use of dual clustering algorithms at the time of writing is better. While clustering algorithms offer promising solutions for DDoS attack detection, they also face several challenges and limitations. These issues may fall within the component of parameter selection whereby the performance of clustering algorithms often depends on the selection of appropriate parameters that are meaningful to the character of the attack or received by the server. The number of clusters in K-means or the density threshold selection in the preprocessing phase is key to prearranging the K-Means for the extremely high-level of the KNN. This means removing null values, exploring outliers and removing columns that seek to serve little purpose. Basically, the prepping of the dataset, using K-Means before applying the KNN, was crucial to my best knowledge. PySpark pre-empts the detection of outliers, making them more easily achievable to accrue understanding from the get-go before applying the clustering algorithms.

Outliers unhinge the results of the K-Means, upsetting the balance of the reduced dataset by having too high variance and producing results that were skewed at the beginning of this study, which emphasizes the importance of mitigating centroid retardation through the omittance of columns that are not entirely useful thus reharmonizing the table in the tuning stage. Furthermore, the addition of the KNN at this point spoke more precisely with the pre-arranged data, providing valid results of top-level accuracy, reducing variance, and re-harmonizing the vectors in order to statistically find the averages of all relevant vectors.

Evolving attack patterns can be seen using this method to a very high rate of accuracy as long as features are selected with knowledge of a correlation as being dependent on one another. In this regard, the "Flow Duration" column to "Forward Packets" and others were centered before splitting the data by the label column to produce a clearer illustration of traffic differentiation, benign or malicious.

Ultimately, the intermix of K-Means to reorganise before the implementation of the KNN model discriminates to a precise prediction when using the right labels in preprocessing for 99 out of 100 examples means accuracy is 99%.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] (2024) ML Beginner's Guide to DDoS Attack Detection Model. Labeller. https://www.labellerr.com/blog/ddos-attack-detection/

[2] Kabanda, R., Byera, B., Emeka, H. and Mohiuddin, K.T. (2023) The History, Trend, Types, and Mitigation of Distributed Denial of Service Attacks. *Journal of Information Security*, **14**, 464-471. https://doi.org/10.4236/jis.2023.144026

[3] Cloudflare (2024) What Is a DDoS Attack? https://www.cloudflare.com/en-gb/learning/ddos/what-is-a-ddos-attack/

[4] GeeksforGeeks (2023) K-Means Clustering Using PySpark Python. https://www.geeksforgeeks.org/k-means-clustering-using-pyspark-python/

[5] Ranjan, A. (2024) Leveraging PySpark Pandas to Streamline Big Data Analysis. Medium. https://blog.devgenius.io/leveraging-pyspark-pandas-to-streamline-big-data-analysis-be4200bad3bf

[6] (2023) Unlocking Insights with K-Means Clustering: Applications & Benefits. ENCORD. https://encord.com/glossary/k-means-clustering/

[7] Gavin, L. (2016) Clustering Web Users with Streaming K-Means. https://www.lewisgavin.co.uk/Machine-Learning-Kmeans/

[8] Eva (2024) Pyspark: Applying Kmeans on Different Groups of a Dataframe. Stack Overflow. https://stackoverflow.com/questions/47224479/pyspark-applying-kmeans-on-different-groups-of-a-dataframe

[9] Das, S., Venugopal, D. and Shiva, S. (2024) A Holistic Approach for Detecting DDoS Attacks by Using Ensemble Unsupervised Machine Learning. https://drsaikatdas.com/papers/holistic.pdf

[10] Saini, P.S., Behal, S. and Bhatia, S. (2020) Detection of DDoS Attacks Using Machine Learning Algorithms. 2020 7*th International Conference on Computing for Sustainable Global Development* (*INDIACom*), New Delhi, 12-14 March 2020, 16-21. https://doi.org/10.23919/indiacom49435.2020.9083716

[11] Saravanan, N. and Gayathri, V. (2018) Performance and Classification Evaluation of J48 Algorithm and Kendall's Based J48 Algorithm (KNJ48). *International Journal of Computational Intelligence and Informatics*, **7**, 188-198. https://www.periyaruniversity.ac.in/ijcii/issue/marnew/2_mar_18.pdf

[12] Mueller, J. and Massaron, L. (2021) Machine learning. John Wiley & Sons.

[13] Wu, X., Ma, T., Cao, J., Tian, Y. and Alabdulkarim, A. (2018) A Comparative Study of Clustering Ensemble Algorithms. *Computers & Electrical Engineering*, **68**, 603-615. https://doi.org/10.1016/j.compeleceng.2018.05.005

[14] G., D.R. (2020) Real Time Anomaly Detection Techniques Using Pyspark Frame Work. *Journal of Artificial Intelligence and Capsule Networks*, **2**, 20-30. https://doi.org/10.36548/jaicn.2020.1.003

[15] Kumar, S., Ghosh, J. and Crawford, M.M. (2001) Best-Bases Feature Extraction Algorithms for Classification of Hyperspectral Data. *IEEE Transactions on Geoscience and Remote Sensing*, **39**, 1368-1379. https://doi.org/10.1109/36.934070

[16] Karami, A. (2015) The Use of Computational Intelligence for Security in Named Data Networking. Master's Thesis, Universitat Politècnica de Catalunya. http://hdl.handle.net/2117/95631

[17] Aamir, M. and Ali Zaidi, S.M. (2021) Clustering Based Semi-Supervised Machine

Learning for Ddos Attack Classification. *Journal of King Saud University—Computer and Information Sciences*, **33**, 436-446. https://doi.org/10.1016/j.jksuci.2019.02.003

[18] Harshitha, T., Tanya, S., Jaswanthi, T. and Venugopalan, M. (2024) A Scalable Machine Learning Model for Sales Forecasting Using PySpark. 2024 *International Conference on Knowledge Engineering and Communication Systems* (*ICKECS*), Chikkaballapur, 18-19 April 2024, 1-6. https://doi.org/10.1109/ickecs61492.2024.10616759