Harnessing PySpark for Big Data Cybersecurity Analytics and Clustering

Goutham Babu Mosya(G01387090), Ganga Raju Parla(G01394334)

gmosya@gmu.edu,gparla@gmu.edu

1 Introduction

The objective of this project is to advance network intrusion detection capabilities using the NF-UQ-NIDS-v2 dataset, employing PySpark's powerful data processing strengths. Focused on the application and validation of clustering techniques, the project aims to develop and rigorously test an efficient intrusion detection model. Key phases include data analysis, preprocessing with label encoding, feature selection through Pearson correlation and RandomForestClassifier, and dimensionality reduction via PCA. The crux of the project lies in implementing KMeans and BisectingKMeans clustering methods. These methods makes use of PySpark SQL and ML modules which are pivotal for their ability to detect subtle and complex patterns in network traffic, essential for identifying potential intrusions. Clustering's effectiveness in this context is further scrutinized by comparing the resultant clusters against the attack classifications within the dataset. This comparison is a critical step, serving as a practical assessment of the model's accuracy in real-world scenarios. By evaluating how closely the clusters align with known attack types, the project not only contributes to the theoretical aspects of network security but also emphasizes its practical application in detecting and responding to network threats.

Through this approach, the project transcends traditional intrusion detection methodologies, offering insights into the effectiveness of clustering in discerning between normal and anomalous network behaviors. The integration of advanced data processing techniques and the strategic application of clustering algorithms underline the project's innovation. The outcome is expected to enhance the accuracy and efficiency of intrusion detection systems, making a significant contribution to the field of network security, especially in handling large-scale data in complex network environments.

2 Methods

2.1 Data Loading

The project commences with the ingestion of a sizable 12GB network traffic dataset, comprising a staggering 75,987,977 rows of data. The selection of this substantial dataset was

deliberate, offering a representative sample of network traffic crucial for our cybersecurity-focused exploration. The vast scale of the dataset, encompassing attributes such as IP addresses, port details, and protocol specifics, aligns with the complexities of real-world network traffic scenarios. By choosing a dataset of such magnitude, we not only challenge the computational capabilities of PySpark but also ensure the relevance and applicability of our analysis to scenarios where handling extensive data is paramount. Through this deliberate dataset selection, our project sets the stage for a comprehensive exploration of PySpark's prowess in the domain of big data analytics. As the resources are limited and working with this huge data is time consuming we did some of the tasks on a sample dataset, which is at least 1GB in size.

2.2 Analysis

In our data analysis, PySpark SQL played a pivotal role in unraveling the complexities of our extensive network traffic dataset. Leveraging its DataFrame capabilities, we efficiently displayed dataset summaries and schemas. The SQL functionalities proved crucial for calculating null value percentages, conducting mode analyses, and exploring attack distribution patterns. PySpark SQL's seamless integration enriched our analytical journey, providing a robust framework for insightful exploration and understanding of the dataset's nuances. These analyses encompassed tasks such as summarizing dataset statistics, evaluating schema details, calculating null value percentages, and delving into the distribution patterns of key features.

2.2.1 Summary Display

In the initial stages of data preprocessing, we harnessed the capabilities of PySpark SQL to provide a succinct yet insightful summary of our extensive network traffic dataset. By utilizing PySpark's DataFrame, we efficiently showcased key statistics, offering a quick and accessible overview of the dataset's underlying structure.

2.2.2 Null Value Analysis

A crucial step in data cleansing involved the meticulous examination of null values using PySpark SQL components. This analysis was essential for calculating the percentage of missing values in each column, providing a comprehensive assessment of data completeness. Identifying columns with missing values was pivotal for subsequent data refinement.

2.2.3 Value Percentage Calculation

PySpark SQL functionalities played a vital role in conducting a detailed mode analysis for each column. This involved unveiling the most frequently occurring values and calculating their respective percentages. By exploring these modes, we gained valuable insights into crucial patterns embedded within the dataset.

2.2.4 Attack Distribution Analysis

To understand the prevalence of various attack types, we leveraged PySpark SQL's robust querying and analytical capabilities. This analysis delved into attack distribution across different modes, offering insights into how different attacks manifested within the dataset. This comprehensive understanding contributed significantly to shaping subsequent stages of our data analysis pipeline.

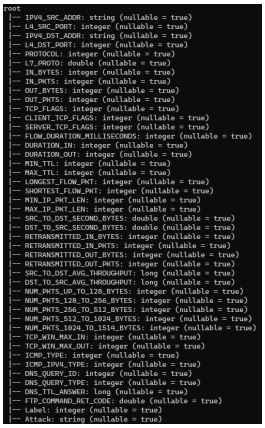

Figure 1: Data Schema

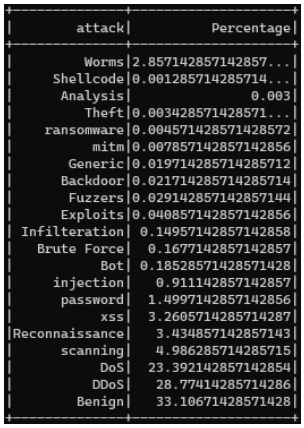

Figure 2: Attack values and Percentages

2.3 Data Preprocessing

As part of our comprehensive data preprocessing efforts, we ensured the dataset's integrity, handling any potential missing values with meticulous care and we strategically addressed string data in two columns using PySpark's 'StringIndexer' for enhanced machine learning compatibility.

2.3.1 Handling Missing Values

Our commitment to data integrity propelled us into a meticulous cleaning process aimed at ensuring the robustness of the dataset. An initial step involved a thorough examination for missing or null values. Remarkably, our scrutiny revealed a dataset free from such discrepancies, laying a strong foundation for subsequent analyses.

2.3.2 Label Encoding

Recognizing the importance of seamless integration with machine learning algorithms, we strategically employed PySpark's 'StringIndexer' to address the presence of string data. This powerful tool facilitated label encoding, transforming string values into integer representations. Beyond mitigating the challenges posed by non-numeric values, this transformation played a pivotal role in enhancing algorithmic interpretability, ensuring compatibility with subsequent analytical processes.

2.3.3 Standard Deviation and Data Refinement

In a further refinement of our dataset, we identified and addressed the presence of infinite standard deviation in specific columns, including ["SRC_TO_DST_SECOND_BYTES", "DST_TO_SRC_SECOND_BYTES"]. This strategic decision was crucial in maintaining the quality of our data and ensuring the robustness of subsequent analyses.

IPV4_SRC_ADDR	IPV4_SRC_ADDR_Index	IPV4_DST_ADDR	IPV4_DST_ADDR_Index
192.168.1.34	9.0	192.168.1.79	21.0
192.168.1.31	4.0	192.168.1.180	9.0
192.168.100.147	1.0	192.168.100.3	0.0
18.219.193.20	11.0	172.31.69.25	13.0
192.168.100.150	3.0	192.168.100.7	2.0
172.31.66.58	298.0	172.31.0.2	3.0
18.219.193.20	11.0	172.31.69.25	13.0
192.168.100.148	0.0	192.168.100.5	1.0
192.168.100.147	1.0	192.168.100.3	0.0
192.168.1.39	8.0	192.168.1.1	6.0
192.168.1.32	5.0	192.168.1.180	9.0
192.168.1.34	9.0	192.168.1.49	12.0
192.168.100.150	3.0	192.168.100.3	0.0
5.188.11.188	272.0	172.31.66.99	129.0
192.168.100.147	1.0	192.168.100.3	0.0
192.168.1.184	13.0	192.168.1.31	26.0
192.168.100.148	0.0	192.168.100.5	1.0
192.168.100.149	2.0	192.168.100.5	1.0
192.168.100.150	3.0	192.168.100.3	0.0
192.168.100.147	1.0	192.168.100.7	2.0

Figure 3: Label Encoded Columns

Column	Mean	StdDev
L4_SRC_PORT	40404.229849	18420.97754975953
L4_DST_PORT	3335.9690518	10657.88736302427
PROTOCOL	10.245085	5.630499150455311
L7_PROTO	53.25444790488263	78.880194798945
IN_BYTES	1087.1559726	115505.83650345192
IN_PKTS	9.7947504	635.839490360716
OUT_BYTES	3212.5473872	270672.3670183029
OUT_PKTS	4.7572078	193.4624976311842
TCP_FLAGS	24.1894738	58.46450595616647
CLIENT_TCP_FLAGS	22.1847986	58.54110248952804
SERVER_TCP_FLAGS	9.525084	18.056901074195668
FLOW_DURATION_MILLISECONDS	2313709.1749148	2140504.255146264
DURATION_IN	533.0120858	836.8419599325878
DURATION_OUT	14.8475514	118.21649352714104
MIN_TTL	53.5957194	39.57925666684549
MAX_TTL	53.654438	39.62156871803375
LONGEST_FLOW_PKT	264.9134832	430.3568867452309
SHORTEST_FLOW_PKT	62.7845426	44.899876840682886
MIN_IP_PKT_LEN	23.595787	26.983169815070344
MAX_IP_PKT_LEN	264.9134832	430.3568867452309
SRC_TO_DST_SECOND_BYTES	2.616038268028709E293	NaN
DST_TO_SRC_SECOND_BYTES	8.002402402416162E116	Infinity
RETRANSMITTED_IN_BYTES	78.980784	8327.960276304535
RETRANSMITTED_IN_PKTS	0.3942592	11.807909686670524
RETRANSMITTED_OUT_BYTES	564.5731278	13836.570147903634
RETRANSMITTED_OUT_PKTS	0.5691612	9.918048255677181
SRC_TO_DST_AVG_THROUGHPUT	2851523.5131264	1.5672516878393328E7
DST_TO_SRC_AVG_THROUGHPUT	7233031.9650176	6.6894130338585824E7
NUM_PKTS_UP_TO_128_BYTES	29.866216	1631.9137465450567
NUM_PKTS_128_TO_256_BYTES	0.9554386	10.904002176314837
NUM_PKTS_256_TO_512_BYTES	0.374737	18.15536693940119
NUM_PKTS_512_TO_1024_BYTES	0.4128878	58.5504536472228
NUM_PKTS_1024_TO_1514_BYTES	2.107623	187.53511131671794
TCP_WIN_MAX_IN	6012.5752248	13230.390024737351
TCP_WIN_MAX_OUT	8392.1410066	17718.876267106858
ICMP_TYPE	3750.1753254	11534.600874346466
ICMP_IPV4_TYPE	14.6491096	45.05700646019347
DNS_QUERY_ID	4532.2396796	13322.671817682392
DNS_QUERY_TYPE	1.400205	17.68630893994236
DNS_TTL_ANSWER	22607.3376108	9557266.529597562
FTP_COMMAND_RET_CODE	1.5084814	20.325090147099854
attackIndex	1.4226976	1.624888447557785
IPV4_SRC_ADDR_Index	412.5756898	3171.2203378699664
IPV4_DST_ADDR_Index	97.1569498	580.9265705303105

Figure 4: Mean and STD of each Column

2.4 Feature Selection

Feature selection is paramount in our project as it enables us to focus on the most relevant aspects of the dataset, enhancing model efficiency and interpretability. By carefully choosing features based on their correlation and importance, we streamline our machine learning models, optimizing their predictive accuracy while mitigating computational complexity in handling large-scale data.
In our feature selection strategy, we employed a comprehensive approach, leveraging both Pearson correlation analysis and a Random Forest classifier.

2.4.1 Correlation Analysis:

First, we conducted Pearson correlation analysis, systematically computing the correlation between each feature and the attack variable. This approach allowed us to unravel intricate relationships between features and the likelihood of the Label column, providing valuable insights into feature importance.

2.4.2 Random Forest Classifier:

Simultaneously, we harnessed the power of a Random Forest classifier to further refine our feature selection process. This involved evaluating the significance of each feature in predicting attacks, offering a different yet complementary perspective on feature relevance. Executed across all features, these procedures collectively equipped us with a comprehensive understanding of the distinct impact each feature has on the Label Column.

Following the dual-feature selection methods, we amalgamated the top-ranking features

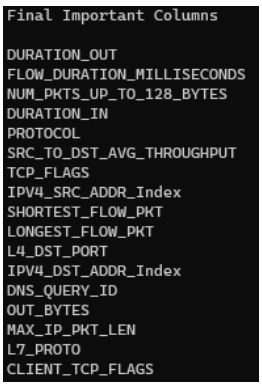

Figure 5: Final Columns Selected

from both correlation analysis and the Random Forest classifier. This fusion allowed us to create a refined set of features that collectively captured the most influential aspects for our subsequent machine learning models. This meticulous feature selection process ensures that our models are trained on the most discriminative and impactful features, enhancing their predictive capabilities and overall performance.

2.5 Dimensionality Reduction

Dimensionality reduction, particularly through methods like PCA, SVD is implemented in our project, is crucial for mitigating the "curse of dimensionality.". PCA effectively addresses this issue by transforming the original features into a reduced set of principal components, capturing the essential variance while discarding less informative dimensions. This not only aids in combating the curse of dimensionality but also contributes to improved model performance, resource utilization, and interpretability of results.

In the realm of dimensionality reduction, our exploration involved experimenting with different methods, specifically Singular Value Decomposition (SVD) and Principal Component Analysis (PCA). Despite testing both techniques, PCA emerged as the more effective approach, showcasing its versatility and robustness in capturing the essential patterns within the data. The pivotal point in our decision-making process was the scree plot analysis, a graphical representation of eigenvalues that aids in determining the optimal number of principal components. Notably, the scree plot exhibited significant dips at 3-4, 5-6, and 16-17 features, providing critical insights into the data's intrinsic structure.

Recognizing the significance of dimensionality reduction in enhancing computational efficiency and interpretability, we strategically chose to retain 5 principal components. This reduction not only streamlined the data but also maintained a meaningful representation of the underlying patterns. The judicious application of dimensionality reduction techniques contributes to a more manageable dataset, facilitating subsequent analysis and clustering procedures in our project.

2.6 Clustering

Clustering in our project serves as a fundamental step to identify inherent structures within the data, allowing for a more nuanced understanding of attack patterns and network behaviors. It aids in uncovering distinct groups, facilitating targeted analysis for different attack types. This process not only enhances the interpretability of our complex dataset but also contributes significantly to the overall efficacy of our security analysis by grouping similar instances together.
In our project, clustering was employed as a pivotal step for understanding the inherent patterns within the data. Using PySpark's implementation of both Bisecting K-Means and K-Means algorithms, we aimed to group instances.

2.6.1 Bisecting K-Means Algorithm

In our project, we harnessed the power of PySpark's Bisecting K-Means clustering algorithm as a foundational step in deciphering the inherent patterns within the extensive network traffic dataset. Through fine-tuning, we explored various configurations for the number of clusters, utilizing insights gleaned from Cross Validation with 10 Fold. The resulting clusters provided a lens through which we could discern natural groupings of data points, offering critical insights into the distribution of different attack types within these segmented clusters. This process not only refined anomaly detection but also deepened our understanding of unique characteristics associated with diverse attack patterns.

2.6.2 K-Means Algorithm

Parallelly, the implementation of PySpark's K-Means algorithm played a crucial role in our endeavor to understand data patterns and group instances based on their similarity in the reduced feature space derived from PCA. Like Bisecting K-Means, we engaged in a fine-tuning process for the number of clusters using Cross Validation with 10 Fold. The clusters obtained through K-Means added an extra layer of granularity, allowing us to delve into the nuanced prevalence of different attack types within each cluster. Leveraging the Silhouette Score for quantitative assessment, we ensured the quality and effectiveness of the clustering results, contributing to informed decision-making and the subsequent refinement of our analytical models.

2.7 Results

2.7.1 Correlation Analysis of Features with Label column

The graph visually portrays the correlation coefficients between individual features and the attack variable. This insightful representation highlights the varying degrees of influence each feature exerts on predicting cyber-attacks.

2.7.2 Correlation Analysis with Random Forest Classifier

The graph illustrates the feature importances computed by the Random Forest classifier, shedding light on the distinctive contribution of each feature to predicting cyber-attacks.

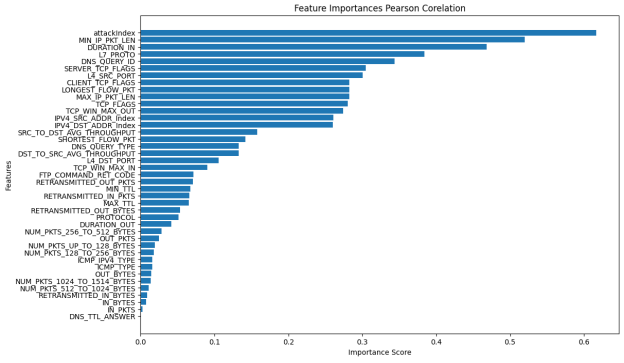

Figure 6: Columns vs Correlation score with Label Column

This analysis offers a complementary perspective to traditional correlation methods, enriching our understanding of feature relevance and aiding in the selection of impactful variables for subsequent modeling

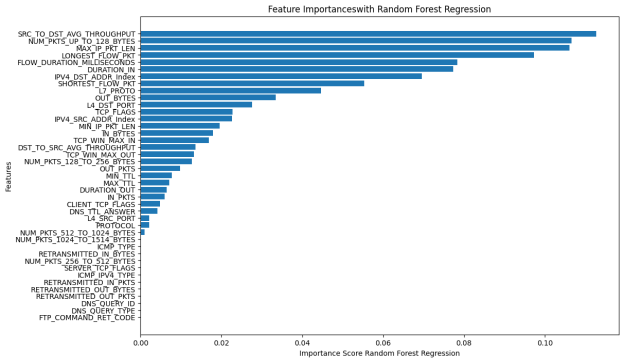

Figure 7: Columns vs Importance score with Label Column

2.7.3 PCA Scree Plot

The Scree Plot, plotted on a logarithmic scale, visually represents the explained variances obtained from Principal Component Analysis (PCA). This logarithmic transformation enhances the clarity of the plot, allowing us to discern crucial dips. These dips guided our decision to choose 5 features.

2.7.4 Silhouette Scores for BisectingKMeans and KMeans

The Silhouette Scores graph of cross validation (10 Fold) with k values from 2 to 25 illustrates the refinement process of Bisecting K-Means and K-Means clustering algorithms, revealing optimal configurations for well-defined clusters.Although the values peak at K=2 in both the model when the clusters percentages are compared, both the models performed very bad as all the values are assigned to one cluster hence the peak. As we go along the K values both models started performing well K=11 with regular amount of distribution to the clusters. When compared the highest points in both the methods, 21 seems to be the best K value

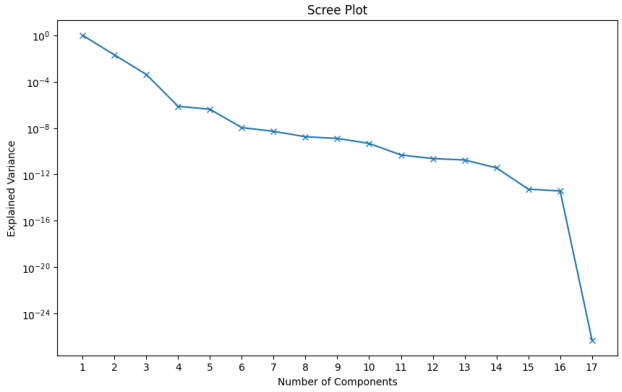

Figure 8: Scree Plot for PCA

since it is one of the highest point in K means and the starting point in the flat line for Bisecting K Means model, which is exactly the number of clusters which the Attack column has.

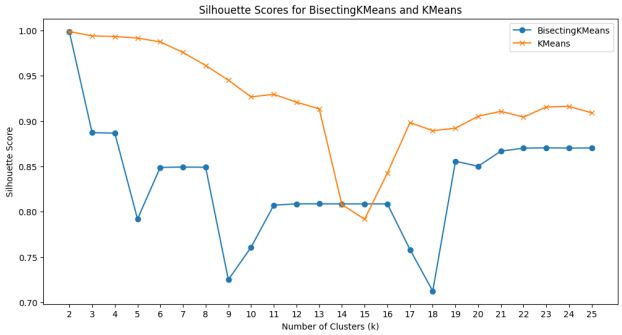

Figure 9: Average Silhouette from Cross Validation (10 Fold) vs K values

attackIndex	percentage
20.0	2.8571428571428574E-4
19.0	0.0012857142857142856
18.0	0.003
17.0	0.0034285714285714284
16.0	0.004571428571428572
15.0	0.007857142857142856
14.0	0.019714285714285712
13.0	0.021714285714285714
12.0	0.029142857142857144
11.0	0.040857142857142856
10.0	0.1495714285714858
9.0	0.1677142857142857
8.0	0.18528571428571428
7.0	0.911142857142857
6.0	1.4997142857142856
5.0	3.2605714285714287
4.0	3.434857142857143
3.0	4.986285714285715
2.0	23.392142857142854
1.0	28.774142857142857
0.0	33.10671428571428

Figure 10: Actual Clusters Percentages

prediction	percentage
10	1.4285714285714287E-4
20	1.4285714285714287E-4
3	2.8571428571428574E-4
8	5.714285714285715E-4
18	7.142857142857143E-4
11	0.0014285714285714286
6	0.0028571428571428572
17	0.0038571428571428572
15	0.024714285714285713
5	0.06142857142857142
2	0.1075714285714285
9	0.15042857142857144
16	0.20714285714285713
19	0.6597142857142857
14	1.052
0	2.197142857142857
12	4.991000000000005
7	6.051142857142857
13	20.991714285714284
1	31.083285714285715
4	32.41271428571429

Figure 11: Predicted Clusters Percentages

3 Conclusions

In Summary, we explored the use of advanced clustering techniques for network intrusion detection, employing the NF-UQ-NIDS-v2 dataset in a PySpark environment. The project's journey involved meticulous data preprocessing, feature selection using Pearson correlation and RandomForestClassifier, and dimensionality reduction through PCA, leading up to the application of KMeans and BisectingKMeans clustering algorithms. A key part of our analysis was comparing these clustering results with the dataset's attack column, aiming to validate the practical effectiveness of our model. However, an intriguing finding emerged: despite high Silhouette Scores, which typically indicate well-defined clusters, the use of lower k values in clustering led to a significant skew, with most data points being assigned to a single cluster. This highlighted a potential gap between theoretical clustering metrics and their real-world applicability, especially in complex datasets like NF-UQ-NIDS-v2.