St. Joseph's University

Bengaluru, Karnataka - 560027

Project in [BD2P5] Multivariate Statistics on

Network Intrusion Detection System

Submitted by:

Nixon Joji (222BDA01)

Yobah Bertrand Yonkou (222BDA15)

Supraja Yadav (222BDA46)

Sinchana R (222BDA48)

Alvina Joanna (222BDA63)

Saba Santhosh (222BDA65)

Abstract

This report outlines the development of a network intrusion detection system with the aim of detecting anomalies in network data, alerting clients about potential malicious attacks, and providing information about the type of attack detected. Initially, PCA was the tool of choice for dimensionality reduction and anomaly detection, but while it did an excellent job at retaining a lot of information in dimensionality reduction, it could not detect malicious data accurately. Subsequently, "Double PCA" was implemented by applying PCA twice; first to the entire dataset down to 7 components that could not have PCA implemented again due to the basic assumptions of PCA. Random forest was used for feature selection, and SMOTE was applied to balance unbalanced data. Isolation forest was used to detect outliers. The data was classified into benign and malicious using random forest, with further classification into specific types of malicious attacks. The system includes a user interface developed using Flask for backend and React JS for frontend.

Introduction

In a world that is moving towards faster, better, more efficient connectivity in order to enable technologies that are the stuff of science fiction fantasies, there lurks in the shadows an ever present and highly active cybercrime world. As the number of cyber crime increases, cyber security has become one of the major concerns in this world, especially for companies. Companies hold important and confidential data about the customers, clients, employees and much more, which upon being lost or corrupted, causes a huge loss for the company. Thus, detecting the intrusion and preventing the crime is much better and cost effective compared to retrieving data after the attack. This is where network intrusion detection comes into play. A network Intrusion detection system (IDS) is a software based application or hardware used to detect any malicious behaviour in a network.

In this project we developed a network intrusion detection system, named ISEEU, that detects any intrusion in the network, classifies the malicious attack into various types of attack and then notifies the concerned officials about the attack to take further actions to prevent it. We implemented various machine learning models, namely PCA, isolation forest and Random Forest, and we will be discussing the failures of a few models for our dataset and the final model developed. The user interface for the system was developed using Flask and ReactJS for easy monitoring of the system.

As proof of concept we referred to “Efficient Network Intrusion Detection Using PCA-Based Dimensionality Reduction of Features“ (Abdulhammed, Faezipour, Musafer, Abuzneid, 2018) where they used PCA for dimensionality reduction on CICIDS2017 intrusion detection and prevention dataset. The dataset represents network traffic flow which was collected over 5 days, Monday through Friday, and specifies the type of attack.

This study reflects on the failure of PCA and isolation forest in anomaly detection in this network flow dataset, and accustoms RandomForest for classification of the network into benign and malicious and then further classifies the malicious data into different types of attack which is discussed in detailed in this report.

Literature Survey

Monitoring a network to identify unusual patterns in the traffic flow involves looking at a variety of potential attributes that could convey the absence or presence of a network attack. Over time, monitoring has been automated with the help of machine learning algorithms which have proven very useful in multiple fields over the past decades. Some of these algorithms have been applied on large amounts of data to identify patterns that would otherwise be difficult for the human brain to process. Identifying patterns that deviate from normal traffic behaviour can be used to separate anomalous traffic from normal traffic. The complexity and efficiency of these algorithms is greatly influenced by the number of attributes or features taken into consideration during execution time; the lesser the number of features, the faster and probably more efficient the algorithms would be. Over time, multiple statistical techniques have been employed to handle this problem of high dimensional data, one of which is principal component analysis (PCA). Moreover, one of the most common problems with classification is that one class is predominant (usually the normal class) than the other. In this section, we will have a look at previous research that has made use of data balancing techniques and dimensionality reduction techniques, precisely PCA on the CIC2017 dataset to build an intrusion detection system (IDS).

In Abdulhammed et al, they set forth to build a network intrusion detection system (NIDS) using the CIC2017 dataset. The authors used PCA for dimensionality reduction and multiple classifiers with the aim of building an efficient NIDS. Through a comparative study on the performance of multiple classifiers fitted with 10 features obtained by reducing 81 features to 10 principle components (which explained 99.6% of the total variation in the data), the authors concluded that the Random Forest Classifier is robust and suitable for the the classification of both high and low dimensional correlated features. More so, because of the absence of parity between classes in the dataset, a uniformed based balancing technique was employed. It was observed that Random Forest had a slightly better performance on the imbalanced dataset than the balanced dataset, with an accuracy of 99.6% on the imbalance dataset and an accuracy of 98.8% on the balanced dataset.

Ripan et al., in a research titled "*An Isolation Forest Learning Based Outlier Detection Approach for Effectively Classifying Cyber Anomalies*" presented an Isolation Forest Learning-Based outlier detection technique to classify cyber anomalies. They show how Isolation Forest is an effective technique for identifying outliers. Furthermore, a comparative study of the classification accuracy with the dataset with and without outliers was carried out with the use of multiple classifiers, namely, K-Nearest Neighbour (KNN), Support Vector Machine (SVM), Naive Bayes (NB), Logistic Regression (LR) and AdaBooster Classifier (ABC). It was concluded that removing outliers from the dataset leads to improved accuracy, evident by the performance of KNN, SVM, NB, LR and ABC on the dataset with and without outliers. The accuracy scores for KNN, SVM, NB, LR and ABC were

99%, 99%, 90%, 96%, and 99%, respectively, and 100%, 99%, 95%, 98% and 100%, respectively, fitted with dataset with outliers and dataset without outliers, respectively.

Carrera et al. presented three new approaches which are a combination of unsupervised learning and deep learning techniques to build a near real-time anomaly detection system. The proposed algorithms are *Deep Autoencoding with GMM and Isolation Forest*, *Deep Autoencoder with Isolation Forest*, *and Memory Augmented Deep Autoencoder with Isolation Forest*. This experiment made use of KDD99, NSL-KDD, and CIC-IDS2017 datasets. Focussing on the CIC-IDS2017 dataset, the Shapley Adaptive Explanations (SHAP) was used to select 15 features from the CIC-IDS2017 dataset. At the end of the research, it was observed that the *Memory Augmented Deep Autoencoder with Isolation Forest* and with Extended Isolation Forest presented the best accuracy scores; 83.5% and 81.94% respectively.

Chawla et al., proposed a technique called Synthetic Minority Oversampling Technique (SMOTE) for dealing with imbalanced datasets. SMOTE samples data from the minority classes and generates synthetic samples with the help of the K-Nearest Neighbour algorithm. More so, SMOTE has proven to be a useful technique for handling imbalanced datasets. Stiawan et al. used SMOTE on the CIC-IDS2017 dataset for feature selection. Additionally, Algrhan et al. conducted an experiment to observe the effects of imbalance data on the performance of classification. It was observed that the performance of classifiers used (Decision Tree, Decision Jungle and SVM) increased after using SMOTE to introduce parity in the dataset.

Dataset

The data we used was the CIC-IDS dataset which was collected by the Canadian Institute for Cybersecurity in 2017.

Benefits of an Intrusion Detection System

With the internet permeating all aspects of our lives, the air it breathes is - the network that it runs on. Pull the plug and the music stops. With the ever evolving and developing science of telecommunication technologies, we're able to transmit more data - better and with low latency! However, those telecommunication channels that we use aren't 100% crook-proof. Just like the roads we walk on to get from point A to point B are vulnerable to crime, so are the network channels we depend on.

Thus the need for a sort of cyber watchman has come to be. Intrusion detection systems are one such tool. Just like a neighbourhood without a trusty watchman is more likely to be susceptible to break-ins, so is the network infrastructure of any organisation.

Similarly, robberies are expensive. Network intrusions are expensive. People break in - get the gold; in our case it's the data - and leave with a trail of chaos.

As we see the average cost of a data breach by industry, it's interesting to note that the industries we pay to keep our data confidential are the most affected and lose more money than other industries like public sectors (way down the list). Data protection and cost of business go hand in hand. Data protection in turn affects things like the price of products. Because you can lose customers, you might have to refund customers, you may need to deploy higher quality network infrastructure that lends directly to the opex included in product costs and so on.
Thus, software tools like ours that help with keeping an eye on the network will definitely benefit in the long run. Better safe than sorry.

Explain the implementation

This section gives an overview of the methodology and how we carried out the experiment. The process involves preprocessing, implementation of PCA, double PCA, isolation forest and random forest for feature selection and classification.

A. Preprocessing
   In this step, the dataset from all the 5 days are merged into one dataset. From this dataset unnecessary columns, namely, source IP, source port, destination IP, destination port, flow ID, and protocol, were dropped as they are not required for PCA. Infinity values are replaced as nan. Then all the missing values that were visualised using a heat map were dropped.
   From this original data we created two dataframes, one containing only BENIGN (normal data) records and the other data frame has MALICIOUS (attack or abnormal data) records. The reason for doing this is so that we can apply PCA only on the benign dataset in order to find the threshold for labelling observations as benign or malicious. Variables are created to hold the timestamp values of each type of data (benign, malicious and one for the entire dataset) which will be used when plotting the principal components and reconstruction loss. The timestamp and labels column are dropped.
   Using the entire dataset (both benign and malicious), a Standard Scaler object was created and later used to scale the values of all other data frames (all, benign, and malicious).

B. PCA
   The dataset is split into train and test, where the train data is used to create the PCA model, with n_components set to 1, while test data is used to check the performance of the model. The train data contains only the benign data while the test data contains both benign and malicious data. The PCA model was fitted on the scaled benign dataset and using this we transformed the rest of the dataset. Upon plotting this single component, we noticed a lot of spikes.

We later used the inverse_transform method to transform the single principle component of the benign dataset back to its original dimension.The reconstruction error was calculated by taking the difference between each data point in the original benign dataset and the corresponding data point (in terms of location) in the reversed PCA dataset and taking the sum of each row. This reconstruction error was plotted on a line graph. The aim here was to identify a threshold that could be used for labelling observations as benign or malicious. Also, PCA was applied on the malicious dataset, reversed and the reconstruction error was calculated and plotted.We tried to compare the two loss graphs to see whether we could identify a threshold. However, it was difficult to identify one, as both error graphs were very similar. Thus we moved on to the next technique 'Double PCA'.

C. Double PCA

This technique is a slightly modified version of PCA, where we apply PCA on the principal components obtained by applying PCA on the original dataset. Upon plotting the number of components vs cumulative sum of explained variance, we find that about 25 components collectively explain more than 95% of the data. Thus we apply PCA to the original benign dataset and bring it down to 25 principal components and later apply PCA on these 25 principal components to bring it down to 1 component.

We later used the inverse_transform method to transform this single principle component of the benign dataset back to its original dimension. The reconstruction error was calculated and upon plotting it we realised that it was similar to applying PCA once in the dataset. Thus this method could not help us with anomaly detection.

D. Isolation Forest

After failing to identify a threshold value that could be used to segregate our dataset into benign and malicious with PCA and Double PCA, we came across another method that can be used for anomaly detection: isolation forest. Isolation Forest is an ensemble unsupervised machine learning algorithm that can be used for anomaly detection. Isolation Forest segregates anomalies from normal data by identifying outliers. The two main assumptions of this algorithm is that outliers are the smallest group of data in a dataset and their values differ from that of normal data points. More so, since outliers are less and differ from normal data points, they would require a less number of splits as compared to normal data points. With Isolation Forest, we have to specify what percentage of our dataset is suspected to be outliers. This is done by assigning a value between 0.0 and 0.5, inclusive to the contamination parameter. Moreover, Isolation forest assigns an anomaly score to

each data point with lower values indicating that the datapoint is more likely to be abnormal, negative scores indicating outliers and positive scores indicating inliers.

Furthermore, we first applied Isolation Forest on our dataset with a contamination value of 0.1. Even though I could capture a greater portion of the malicious data into the outlier zone, a significant amount of outliers were still not recognised as non malicious. More so, we increased the contamination value from 0.1 to 0.3 which captured more malicious data into the outlier zone. Alongside the malicious data was a significant amount of normal (benign) datapoints thus making the model not an ideal model for anomaly detection.

E. Random Forest
Since the unsupervised learning methods were not suitable for anomaly detection in our dataset, we used the supervised learning technique i.e. Random Forest was used for anomaly detection and classification. Feature selection was done using Random Forest and using these features the dataset was first classified to detect benign or malicious data. If malicious data was detected, the data is further subjected to multiclass classification to detect the type of attack.

Future works

The model could use one of the wrapper methods of feature selection which are often considered to be robust.
This project could be made into a web extension or an API that minimises the user interaction to just a notification via mail.
Along with receiving a notification, the user could be given more details from the dashboard like a snapshot of what it looked like when the intrusion occurred.

Conclusion

PCA may have failed to give us the desired results but it still is one of the strongest tools for dimensionality reduction.
This project needed multiple attempts in finding the best algorithm for dimensionality reduction, anomaly detection and classification model. Our learnings from all the attempts have been mentioned before. We encourage the readers to reach out to us with their insights into how it could have been done differently and try out algorithms that handle abnormally spread out datasets like these better.

Appendix

List of features

Feature Name	Description
Flow duration	Duration of the flow in Microsecond
total Fwd Packet	Total packets in the forward direction
total Bwd packets	Total packets in the backward direction
total Length of Fwd Packet	Total size of packet in forward direction
total Length of Bwd Packet	Total size of packet in backward direction
Fwd Packet Length Min	Minimum size of packet in forward direction
Fwd Packet Length Max	Maximum size of packet in forward direction
Fwd Packet Length Mean	Mean size of packet in forward direction
Fwd Packet Length Std	Standard deviation size of packet in forward direction
Bwd Packet Length Min	Minimum size of packet in backward direction
Bwd Packet Length Max	Maximum size of packet in backward direction
Bwd Packet Length Mean	Mean size of packet in backward direction
Bwd Packet Length Std	Standard deviation size of packet in backward direction
Flow Byte/s	Number of flow packets per second
Flow Packets/s	Number of flow bytes per second
Flow IAT Mean	Mean time between two packets sent in the flow
Flow IAT Std	Standard deviation time between two packets sent in the flow
Flow IAT Max	Maximum time between two packets sent in the flow
Flow IAT Min	Minimum time between two packets sent in the flow
Fwd IAT Min	Minimum time between two packets sent in the forward direction
Fwd IAT Max	Maximum time between two packets sent in the forward direction
Fwd IAT Mean	Mean time between two packets sent in the forward direction
Fwd IAT Std	Standard deviation time between two packets sent in the forward direction
Fwd IAT Total	Total time between two packets sent in the forward direction
Bwd IAT Min	Minimum time between two packets sent in the backward direction
Bwd IAT Max	Maximum time between two packets sent in the backward direction
Bwd IAT Mean	Mean time between two packets sent in the backward direction
Bwd IAT Std	Standard deviation time between two packets sent in the backward direction
Bwd IAT Total	Total time between two packets sent in the backward direction

Fwd PSH flag	Number of times the PSH flag was set in packets travelling in the forward direction (0 for UDP)
Bwd PSH Flag	Number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP)
Fwd URG Flag	Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)
Bwd URG Flag	Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP)
Fwd Header Length	Total bytes used for headers in the forward direction
Bwd Header Length	Total bytes used for headers in the backward direction
FWD Packets/s	Number of forward packets per second
Bwd Packets/s	Number of backward packets per second
Min Packet Length	Minimum length of a packet
Max Packet Length	Maximum length of a packet
Packet Length Mean	Mean length of a packet
Packet Length Std	Standard deviation length of a packet
Packet Length Variance	Variance length of a packet
FIN Flag Count	Number of packets with FIN
SYN Flag Count	Number of packets with SYN
RST Flag Count	Number of packets with RST
PSH Flag Count	Number of packets with PUSH
ACK Flag Count	Number of packets with ACK
URG Flag Count	Number of packets with URG
CWR Flag Count	Number of packets with CWE
ECE Flag Count	Number of packets with ECE
down/Up Ratio	Download and upload ratio
Average Packet Size	Average size of packet
Avg Fwd Segment Size	Average size observed in the forward direction
AVG Bwd Segment Size	Average number of bytes bulk rate in the backward direction
Fwd Header Length	Length of the forward packet header
Fwd Avg Bytes/Bulk	Average number of bytes bulk rate in the forward direction
Fwd AVG Packet/Bulk	Average number of packets bulk rate in the forward direction
Fwd AVG Bulk Rate	Average number of bulk rate in the forward direction
Bwd Avg Bytes/Bulk	Average number of bytes bulk rate in the backward direction
Bwd AVG Packet/Bulk	Average number of packets bulk rate in the backward direction
Bwd AVG Bulk Rate	Average number of bulk rate in the backward direction

Subflow Fwd Packets	The average number of packets in a sub flow in the forward direction
Subflow Fwd Bytes	The average number of bytes in a sub flow in the forward direction
Subflow Bwd Packets	The average number of packets in a sub flow in the backward direction
Subflow Bwd Bytes	The average number of bytes in a sub flow in the backward direction
Init_Win_bytes_forward	The total number of bytes sent in initial window in the forward direction
Init_Win_bytes_backward	The total number of bytes sent in initial window in the backward direction
Act_data_pkt_forward	Count of packets with at least 1 byte of TCP data payload in the forward direction
min_seg_size_forward	Minimum segment size observed in the forward direction
Active Min	Minimum time a flow was active before becoming idle
Active Mean	Mean time a flow was active before becoming idle
Active Max	Maximum time a flow was active before becoming idle
Active Std	Standard deviation time a flow was active before becoming idle
Idle Min	Minimum time a flow was idle before becoming active
Idle Mean	Mean time a flow was idle before becoming active
Idle Max	Maximum time a flow was idle before becoming active
Idle Std	Standard deviation time a flow was idle before becoming active

Footnotes/Sources

1. Dataset
   Intrusion Detection Evaluation Dataset (CIC-IDS2017)
2. Feature description
   https://github.com/CanadianInstituteForCybersecurity/CICFlowMeter/blob/master/ReadMe.txt
3. Market research report
   Cost of a Data Breach Report 2022
4. Writing a white paper
   How to Write and Format a White Paper (With Examples)
5. Understanding PCA
   PCA (Principal Component Analysis) Machine Learning Tutorial
   Why are principal components in PCA (eigenvectors of the covariance matrix) mutually orthogonal? - Cross Validated
6. Efficient Network Intrusion Detection Using PCA-Based Dimensionality Reduction of Features
7. An Isolation Forest Learning Based Outlier Detection Approach for Effectively Classifying Cyber Anomalies
8. Combining Unsupervised Approaches for Near Real-Time Network Traffic Anomaly Detection
9. SMOTE: Synthetic Minority Over-sampling Technique
10. Important Features of CICIDS-2017 Dataset For Anomaly Detection in High Dimension and Imbalanced Class Dataset
11. SMOTE: Class Imbalance Problem In Intrusion Detection System