Morgan CS Special Publication 2021-10-001

Cyber Threat Analysis using Artificial Intelligence and Machine Learning

Department of Computer Science
School of Science
Morgan State University

October 2021

Morgan CS Special Publication 2021-10-001
Morgan State University Spec. Publ. 2021-10-001, 19 pages (October 2021)
CODEN: Morgan-CS

Executive Summary

Cyber threats can cause severe damage to computing infrastructure and systems as well as data breaches that make sensitive data vulnerable to attackers and adversaries. It is therefore imperative to discover those threats and stop them before bad actors penetrating into the information systems.

Cyber threats analysis involves analyzing billions of data either live or recorded from servers, firewalls, intrusion detection and prevention systems, and network devices. Signature based analyzing method is effective only if the attack vectors are pre-defined and already stored in the knowledge database. However, if threat actors change their behavior, it would be hard to capture. To identify the abnormal behaviors, threat analytics systems need to not only examine the irregular patterns at certain time but also observe the behavior changes comparing with the normal states.

The advancement of Artificial Intelligence (AI) has opened up new methods to analyze and understand data and user behaviors. Machine learning (ML) algorithms are able to not only analyze data efficiently but also accumulate the knowledge gained from previous learning. The ML models are getting improved from time to time with new feed-in data.

The accuracy of the AI/ML algorithms could be affected by many factors, from algorithm, data, to prejudicial, or even intentional. As a result, AI/ML applications need to be non-biased and trustworthy.

The Department of Computer Science faculty and students have conducted study to apply machine learning in cyber threat analysis. This white paper provides a survey of tools and process for countering those attacks. In addition, an open-source Python application using machine learning to identify cyber threat has been developed during the research process.

Key words

Artificial Intelligence; Machine Learning; Threat; Cybersecurity; Trust; Trustworthiness; Data Analytics; Ethical and Bias; User Navigation; Log Analysis; Cloud Computing.

1. Introduction

Cyber threat analysis involves billions of data records from servers, firewalls, intrusion detection and prevention systems, and network devices. Those include log data that have been stored on log servers and live data that are generated in real time. It has been a challenge to comb through those vast amount of data to find out threat vectors. Predefined signatures are effective but they would fail if new threat vectors emerge or adversaries' behavior changes.

The advancement of Artificial Intelligence has opened up new ways to analyze and understand data and user behaviors. Machine learning algorithms are able to not only analyze data efficiently but also build up knowledge gained from previous learning. Algorithms and models built for general purposes are available for immediate starting a project. The models become more accurate under supervised training when more data are fed in.

Machine learning can be classified as:

- **Unsupervised learning**. A clustering model attempts to find groups, similarities, and relationships within unlabelled data, Figure 1 illustrates a unsupervised learning model. [1]

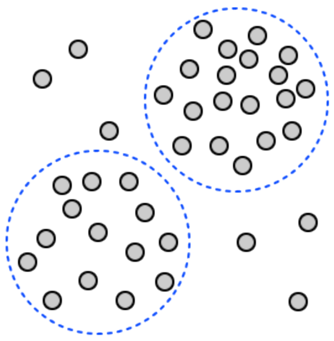

Fig. 1. Unsupervised Learning

- **Supervised learning**. A classification model to identify how input variables contribute to the categorization of data points. Figure 2 depicts a supervised learning model. [2]
- **Semi-supervised learning**. A classification model falls between supervised learning and unsupervised learning by combining a small amount of labeled data with a large amount of unlabeled data during training.
- **Reinforcement learning**. Reinforcement learning is characterized by a continuous loop where an agent interacts with an environment and measures the consequences

[1]image source: [1]
[2]image source: [1]

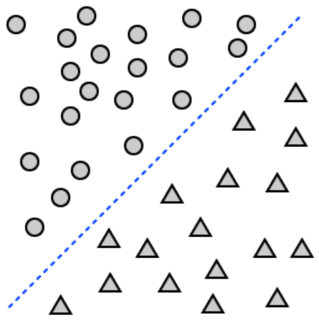

Fig. 2. Supervised Learning

of its actions. Figure 3 shows a reinforcement learning model. [3]

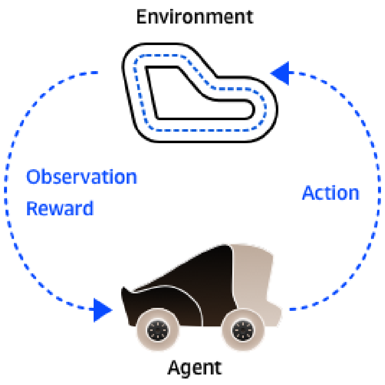

Fig. 3. Reinforcement Learning

- **Transfer learning**. Transfer learning stores knowledge gained while solving one problem and applies it to a different but related problem.

Public sector organizations operate external-facing applications with broad multi-task user interfaces. There are significant investments made in human-centered design, but new opportunities exist to enhance personalization of the interfaces through machine learning and AI to reduce burden on the public user in navigation and task performance. The problem involves leveraging information from user roles, prior behavior, schedules of required activities, and other characteristics to predict the intended reasons why users are entering a system at any given time, and where they plan to navigate and work. This information would be used to facilitate the users in those navigations. Tracking and log data from servers on AWS or Google Analytics can be used for exploration in the development of a methodology. More Machine Learning complex navigation analytics can be explored using behavior analysis method and visualized through business intelligence dashboards.

[3]image source: [1]

2. Current Status in AI/ML-based Models and Analytics

Artificial intelligence brings innovation in industry and everyday life. The Deep Blue chess computer can defeat the greatest human chess player in the world. The autonomous vehicles such as Tesla can drive on the road without human interactions. Machine learning can reveal a lot of things that human beings can hardly find out. By analyzing music using IBM Watson AI, people can learn the mode of songs and hence discovered that the majority of songs from 60s to now are in the mode of ”sadness”.

In cybersecurity, AI/ML is used to deep inspect the packets, analyze the network activities, and discover abnormal behaviors.

Sagar et al. conducted a survey of cybersecurity using artificial intelligence [2]. It discusses the need for applying neural networks and machine learning algorithms in cybersecurity.

Mittu et al. proposed a way to use machine learning to detect advanced persistence threats (APT) [3]. The approach can address APT that can cause damages to information systems and cloud computing platforms.

Mohana et al. proposed a methodology to use genetic algorithms and neural networks to better safe guard data [4]. A key produced by a neural network is said to be stronger for encryption and decryption.

With a grant from the National Science Foundation (NSF), Wang and Kelly developed a video data analytics tool that can penetrate into videos to ”understand” the context of the video and the language spoken [5].

Kumbar proposed a fuzzy system for pattern recognition and data mining [6]. It is effective in fighting phishing attacks by identifying malware.

Using Natural Language Processing (NLP), Wang developed an approach that can identify issues with cybersecurity policies in financial processing process [7] so financial banking companies can comply with PCI/DSS industry standard.

Harini used intelligent agent to reduce or prevent distributed denial of service (DDoS) attacks [8]. An expert system is used to identify malicious codes to prevent being installed in the target systems.

With a grant from National Security Agency (NSA), Wang and his team developed an intelligent system for cybersecurity curriculum development [9]. The system is able to develop training and curricula following the National Initiative of Cybersecurity Education (NICE) framework.

Dilrmaghani et al. provide an overview of the existing threats that violate security and privacy within AI/ML algorithms [10].

Gupta et al. studied quantum machine learning that uses quantum computation in artificial intelligence and deep neural networks. They proposed a quantum neuron layer aiming to speed up the classification process [11].

Mohanty et al. surveyed quantum machine learning algorithms and quantum AI applications [12].

Edwards and Rawat conducted a survey on quantum adversarial machine learning by adding a small noise that leads to classifier to predict to a different result [13]. By depolarization, noise reduction and adversarial training, the system can reduce the risk of adversarial attacks.

2.1 Bias in Existing AI/ML Algorithms

People hope the neutrality in AI/ML algorithms. Unfortunately, bias does exist. *Washington Post* published an article that ”credit scores are supposed to be race-neural. That’s impossible”. *Forbes* published a report that ”self-driving card are more likely to recognize white pedestrians than black pedestrians, resulting in decreased safety for darker-skinned individuals. A *Nature* paper reports that ”a major healthcare company used an algorithm that deemed black patients less worthy of critical healthcare than others with similar medical conditions. *Associate Press* published an article that ”financial technology companies have been shown to discriminate against black and latinx households via higher mortgage interest rates.

In general, bias can be classified into the following categories:

- **Algorithm Bias**. Bias as a result of inaccurate algorithms are used.
- **Data Bias**. Bias due to incorrectly sample the data for training that are not reflect the whole data set.
- **Prejudicial Bias**. Feeding model with prejudicial knowledge for example ”nurses are female”.
- **Measurement Bias**. Bias as a result of incorrect measurement.
- **Intentional Bias**. People embed unjust or discriminatory rules in the AI/ML models.

Trustworthy AI/ML is to discover those bias and build robust AI/ML algorithms that is trustworthy. For example, a Tesla with autopilot could crash onto a fire truck, which is hardly possible even for the worst human drivers. This shows that the nine ”eyes” (radar system) under artificial intelligence are still inferior than two eyes of human intelligence. To be trustworthy, Tesla cars need to be trained with reliable data that can ”filter” the ”noise” caused by the emergency light flashing, which changes the images of a fire truck.

Effort has been put in countering the AI bias. Obaidat et al. uses random sampling on images with a convolutional neural network (CNN) [14]. They tested using the Fashion-MNIST dataset that consists of 70,000 images, 60,000 for training and 10,000 for testing with an accuracy of 87.8%.

Bernagozzi et al. at IBM conducted a survey that reveals the presence of bias in chatbots and online language translators using two-tier method to rate bias [15] .

Lohia et al. at IBM proposed a framework that can detect bias to improve fairness [16]. The algorithms detects individual bias and then post-processing for bias mitigations.

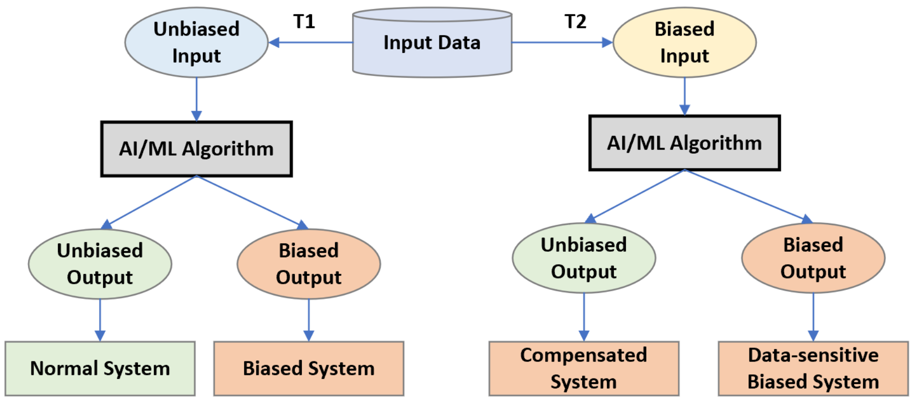

Fig. 4. Two-tier Analyzing Method for Rating Bias in AI/ML Algorithms

2.2 Using Adversarial ML Model to Discover Bias

Adversarial ML model is commonly used to attack ML algorithms in a way that the models would function abnormally. McAfee once attacked Tesla's system by adding a strip to a speed limit sign that fooled the car to drive 50 mph over the speed limit. A "stealth" ware technology with adversarial pattern on glasses and clothes can fool facial recognition systems. Adversatial attacks can be classified into three categories:

- **Evasion**. An attack uses steganography and other technologies to obfuscate the textual content.
- **Poisoning**. An attack to contaminate the training data.
- **Model stealing**. An attack to consider the target as a black box and try to extract data from the model.

2.3 Notation of Bias and Mitigation

Consider a supervised classification problem with features X $\in \mathscr{X}$, categorical protected attributes D $\in \mathscr{D}$, and categorical labels Y $\in \mathscr{Y}$. We are given a set of training samples $\{(x_1,d_1,y_1),...,(x_n,d_n,y_n)\}$ and would like to learn a classifier $\hat{y}: \mathscr{X} \times \mathscr{D} \rightarrow \mathscr{Y}$. For ease of exposition, we will only consider a scalar binary protected attribute, i.e. $\mathscr{D} = \{0,1\}$. The value $d = 1$ is set to correspond to the *privileged* group (e.g. whites in the United States in criminal justice application) and $d = 0$ to *unprivileged* group (e.g. blacks). The value $y = 1$ is set to correspond to a *favorable* outcome. Based on the context, we may also deal with probabilistic binary classifiers with continuous output scores $\hat{y}_S \in [0,1]$ that are thresholded to $\{0,1\}$.

One definition of individual bias is as follows. Sample i has individual bias if $\hat{y}(x_i, d = 0) \neq \hat{y}(x_i, d = 1)$.Let $b_i = I[\hat{y}(x_i, d = 0) \neq \hat{y}(x_i, d = 1)]$, where $I[\cdot]$ is an indicator function.

The individual bias score, $b_{S,i} = \hat{y}_S(x+i, d=1) - \hat{y}_S(x_i, d=0)$, is a soft version of b_i. To compute an individual bias summary statistic, we take the average of b_i across test samples.

One notion of group fairness known as *disparate impact* is defined as follows [16]. There is disparate impact if the division of the expected values

$$\frac{\mathbb{E}[\hat{y}(X,D)|D=0]}{\mathbb{E}[\hat{y}(X,D)|D=1]} \tag{1}$$

is less than $1-\varepsilon$ or greater than $(1-\varepsilon)^{-1}$, where a common value of ε is 0.2.

The test procedure usually divided into two steps: 1) determine whether there are any trials of individual bias, 2) discover the found bias against all samples.

Mitigation can be performed by changing the label outputs of the classifier $\hat{y}_i$ to other labels $\check{y} \in \mathscr{Y}$.

Zhang et al. use federated learning (FL) in privacy-aware distributed machine learning application [17]. Experiments show it can reduce the discrimination index of all demographic groups by 13.2% to 69.4% with the COMPAS dataset.

3. AI/ML-based Bias and Threat Analytics Tools

Artificial intelligence uses data to train models and uses an inference engine to draw a conclusion or predict the outcome. The overall architecture [4] can be shown as Fig. 5.

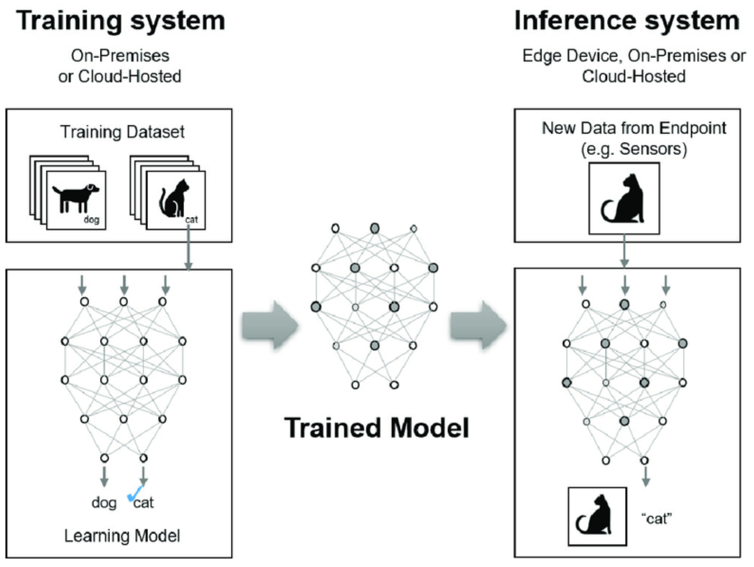
Training system
On-Premises or Cloud-Hosted
Training Dataset
dog
cat
dog
cat
Learning Model
Trained Model
Inference system
Edge Device, On-Premises or Cloud-Hosted
New Data from Endpoint (e.g. Sensors)
"cat"

Fig. 5. AI Architecture.

Su et al. utilize open source testing and analysis tools Hping3, Scapy to simulate DDoS flood and import the data collected from those simulated attacks into Splunk to identify possible attacks [19]. The Splunk machine learning toolkit extends the capabilities of Splunk.

[4]image source: [18]

Ngoc et al. proposed an early warning approach to counter APT. It analyzes the APT target using log analysis techniques [20].

Counterfit is an AI security risk assessment package developed at Microsoft. The open-source tool helps companies conduct AI risk assessment to ensure the AI/ML algorithms are non-bias, reliable, and trustworthy [21].

4. Development of Cyber Threat with Machine Learning Tool - CTML

Detecting web attacks using machine learning is an area that has drawn attention and requires continuous research and development. This project analyzes 822,226 log records from a company's web login page in a 5 hour time span. After cleaning and pre-processing the data, the CTML algorithm detected records that could potentially be attacks. It then calculated the likelihood (of attacks) based on the abnormal behaviors.

The main strategy is to use unsupervised learning for better understanding the distribution of the input data. Supervised learning is then applied for further classification and generating predictions. As a result, the CTML model could learn how to predict/classify on output from new inputs. Reinforcement learning (RL) learns from experiences over time. The algorithm can be improved with more data feed into the system.

The application first loads the input data into a Pandas dataframe, then removes features that are not of interests in detecting attacks. Next, the data are "compressed" from 800,000+ to around 40,000 by combining the records that have the same source and destination ip addresses in the same unit time period. The higher the compression rate, the more the duplications in the dataset. This improved the efficiency of machine learning process. Unsupervised machine learning is applied to the dataset using K-means clustering. The output three clusters are labeled as not-suspicious, suspicious, and transitional area.

The pre-processed data are then splitted into 0.66/0.33 for training/testing and further analyzing the likelihood of each response's abnormal behaviors. Using results (three clusters) from the unsupervised learning as a supervisor, the algorithm continues apply supervised machine learning to discover the threats. In addition to areas that are considered "confident" or "no confident", the transition (gray) area is further analyzed using k-mean clustering to separate into 2 clusters, labeled as "more suspicious" and "less suspicious". The "more suspicious" tags are then added into the suspicious activity dataset. By doing so it ensures the machine does not miss any responses that get filtered out during analyzing process but is still suspected having abnormal behaviors. The likelihood of the suspicion is calculated based on the percentage over the maximum response per second.

An attack for a general log-in page is defined as considerable number of visits, responses, callbacks in a short period of time. Thus, pre-processing the data by combining each duplicating responses per second helps determine the number of responses or visits that stands out.

The application can be improved by feeding into more and rich data so risks associated with human behaviors can be identified.

The 3-2 two-tier classification technique helps narrowing down the suspicious activi-

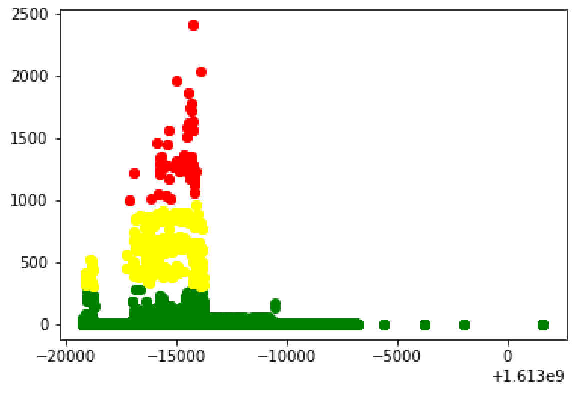

Fig. 6. Classification using Unsupervised Learning

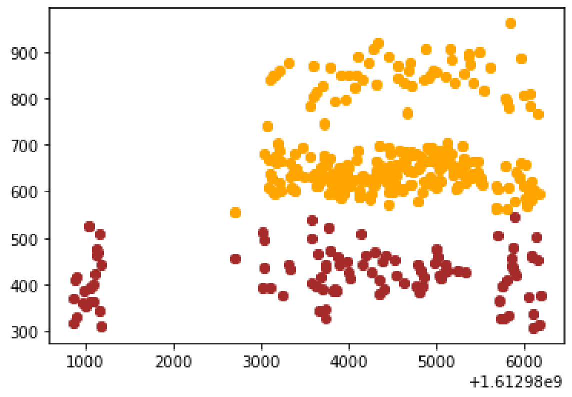

Fig. 7. Classification using Supervised Learning

ties. If the k-means clustering is applied only once with 2 clusters, the uncertain groups of dataset would possibly be wrong. Therefore, creating a transition (grey) area in the middle of two certainties helps detecting the potential attacks that could be missed.

The result is saved into result.csv and all detected attacks are saved in the suspicious_activity.csv.

The research was conducted at Morgan State University. A list of references can be found here [22–27].

Acknowledgments

This research is funded in part under a contract with *Softrams Inc.*.

References

[1] Aws. Available at https://aws.amazon.com.

[2] BS S, S N, Kashyap N, DN S (2019) Providing cyber security using artificial intelligence – a survey. *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, , pp 717–720. https://doi.org/10.1109/ICCMC.2019.8819719

[3] Mittu R, Lawless WF (2015) Human factors in cybersecurity and the role for ai. *2015 AAAI Spring Symposium*, , pp 39–43.

[4] andV K Venugopal KM, Sathwik H (2014) Data security using genetic algorithm and artificial neural network, . Vol. 5, pp 543–548.

[5] Wang S, Kelly W (2014) invideo – a novel big data analytics tool for video data analytics, , . pp 1–19. https://doi.org/0.1109/ITPRO.2014.7029303

[6] Kumbar SR (2014) An overview on use of artificial intelligence techniques in effective security management, . Vol. 2, pp 5893–5898.

[7] Wang S (2018) Risk analysis for financial banking and processing using artificial intelligence. Available at NationalCyberSummit,Huntsville,AL.

[8] Rajan HM (2017) rtificial intelligence in cyber security-an investigation, . Vol. 09, pp 28–30.

[9] Hodhod R, Khan S, Wang S (2019) Cybermaster: An expert system to guide the development of cybersecurity curricula, . Vol. 15, pp 70–78.

[10] Dilmaghani S, Brust MR, Danoy G, Cassagnes N, Pecero J, Bouvry P (2019) Privacy and security of big data in ai systems: A research and standards perspective. *2019 IEEE International Conference on Big Data (Big Data)*, , pp 5737–5743. https://doi.org/10.1109/BigData47090.2019.9006283

[11] Gupta S, Mohanta S, Chakraborty M, Ghosh S (2017) Quantum machine learning-using quantum computation in artificial intelligence and deep neural networks: Quantum computation and machine learning in artificial intelligence. *2017 8th Annual Industrial Automation and Electromechanical Engineering Conference (IEMECON)*, , pp 268–274. https://doi.org/10.1109/IEMECON.2017.8079602

[12] Mohanty JP, Swain A, Mahapatra K (2019) Headway in quantum domain for machine learning towards improved artificial intelligence. *2019 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS)*, , pp 145–149. https://doi.org/10.1109/iSES47678.2019.00040

[13] Edwards D, Rawat DB (2020) Quantum adversarial machine learning: Status, challenges and perspectives. *2020 Second IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, , pp 128–133. https://doi.org/10.1109/TPS-ISA50397.2020.00026

[14] Obaidat M, Singh N, Vergara G (2021) Artificial intelligence bias minimization via random sampling technique of adversary data. *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, , pp 1226–1230. https://doi.org/10.1109/CCWC51732.2021.9375929

[15] Bernagozzi M, Srivastava B, Rossi F, Usmani S (2021) Gender bias in online language translators: Visualization, human perception, and bias/accuracy trade-offs. *IEEE Internet Computing* :1–1https://doi.org/10.1109/MIC.2021.3097604

[16] Lohia PK, Natesan Ramamurthy K, Bhide M, Saha D, Varshney KR, Puri R (2019) Bias mitigation post-processing for individual and group fairness. *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, , pp 2847–2851. https://doi.org/10.1109/ICASSP.2019.8682620

[17] Zhang DY, Kou Z, Wang D (2020) Fairfl: A fair federated learning approach to reducing demographic bias in privacy-sensitive classification models. *2020 IEEE International Conference on Big Data (Big Data)*, , pp 1051–1060. https://doi.org/10.1109/BigData50022.2020.9378043

[18] Jeong D (2020) Artificial intelligence security threat, crime, and forensics: Taxonomy and open issues. *IEEE Access* 8:184560–184574. https://doi.org/10.1109/ACCESS.2020.3029280

[19] Su TJ, Wang SM, Chen YF, Liu CL (2016) Attack detection of distributed denial of service based on splunk. *2016 International Conference on Advanced Materials for Science and Engineering (ICAMSE)*, , pp 397–400. https://doi.org/10.1109/ICAMSE.2016.7840355

[20] Ngoc HL, Cong Hung T, Huy ND, Thi Thanh Hang N (2019) Early phase warning solution about system security based on log analysis. *2019 6th NAFOSTED Conference on Information and Computer Science (NICS)*, , pp 398–403. https://doi.org/10.1109/NICS48868.2019.9023899

[21] Azure counterfeit. Available at https://github.com/Azure/counterfit.

[22] Betarte G, Pardo A, Martínez R (2018) Web application attacks detection using machine learning techniques. *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, , pp 1065–1072. https://doi.org/10.1109/ICMLA.2018.00174

[23] Gong X, Zhou Y, Bi Y, He M, Sheng S, Qiu H, He R, Lu J (2019) Estimating web attack detection via model uncertainty from inaccurate annotation. *2019 6th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/ 2019 5th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, , pp 53–58. https://doi.org/10.1109/CSCloud/EdgeCom.2019.00019

[24] Tian Z, Luo C, Qiu J, Du X, Guizani M (2020) A distributed deep learning system for web attack detection on edge devices. *IEEE Transactions on Industrial Informatics* 16(3):1963–1971. https://doi.org/10.1109/TII.2019.2938778

[25] Geron A (2019) *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (O'Reilly), 2nd Ed.

[26] Torrano-Gimenez C, Nguyen HT, Alvarez G, Franke K (2015) Combining expert knowledge with automatic feature extraction for reliable web attack detection. *Security and Communication Networks* 8(16):2750–2767. https://doi.org/https://doi.org/10.1002/sec.603. https://onlinelibrary.wiley.com/doi/pdf/10.1002/sec.603 Available at https://onlinelibrary.wiley.com/doi/abs/10.1002/sec.603

[27] Abdulraheem M, Ibraheem N (2019) A detailed analysis of new intrusion detection dataset. *Semantic Scolar*, Vol. -, pp –.

List of Tables

List of Figures

Listings

Appendix A: Project on Github

https://github.com/p-neumann/ML_LogAnalysis

Appendix B: List of Code

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
from pandas import read_csv
from pandas.plotting import scatter_matrix
import csv
import numpy as np
import matplotlib.pyplot as plt
from scipy import linalg
from sklearn.decomposition import PCA, FactorAnalysis
from sklearn.covariance import ShrunkCovariance, LedoitWolf
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split

import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
from pandas import read_csv
from pandas.plotting import scatter_matrix
import csv
import numpy as np
import matplotlib.pyplot as plt
from scipy import linalg
from sklearn.decomposition import PCA, FactorAnalysis
from sklearn.covariance import ShrunkCovariance, LedoitWolf
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split

len_of_data = len(data)
len_of_data

date = []
time = []
account = []
sourceIP = []
destIP = []
interface = []
srcPort = []
dstPort = []
protocol = []
byte = []
packets = []
startTime = []
timeTook = []
status = []

```

```python
na = None
i = 1

while i < len(data):
    msg = data[i].get('message',na)
    # for messages with no data will get filtered out here
    if 'ACCEPT' in msg or 'REJECT' in msg:
        timeTokens = data[i].get('timestamp',na).split()
        date.append(timeTokens[0])
        time.append(timeTokens[1])
        tokens = msg.split()
        account.append(tokens[1])
        interface.append(tokens[2])
        sourceIP.append(tokens[3])
        destIP.append(tokens[4])
        #storing actions
        temp = 5
        tempStr = ''
        while(tokens[temp+1] != "OK"):
            tempStr = tempStr + tokens[temp] + ' '
            temp += 1
        actionTokens = tempStr.split()
        srcPort.append(actionTokens[0])
        dstPort.append(actionTokens[1])
        protocol.append(actionTokens[2])
        packets.append(actionTokens[3])
        byte.append(actionTokens[4])
        startTime.append(int(actionTokens[5]))
        timeTook.append(int(actionTokens[6]) - int(actionTokens[5]))
        if(tokens[temp] == 'ACCEPT'):
            status.append(1)
        else:
            status.append(0)
    i += 1
# print label as well
print('Date: ' + date[0])
print('Time: ' + time[0])
print('AWS account: ' + account[0])
print('Network interface: ' + interface[0])
print('Source IP: ' + sourceIP[0])
print('Destination IP: ' + destIP[0])
print('Status: ' + str(status[0]))
print('Source port: ' + srcPort[0])
print('Destination port: ' + dstPort[0])
print('Protocol: ' + protocol[0])
print('Packets of data: ' + packets[0])
print('Size of data: ' + byte[0] + ' bytes')
print('Start time: ' + str(startTime[0]))
print('Time used: ' + str(timeTook[0]) + ' seconds')
print('New size of dataset after filtering: ' + str(len(date)))

```

```
df = pd.DataFrame({
    "date":date,
    "time":time,
    "account":account,
    "interface":interface,
    "sourceIP":sourceIP,
    "destIP":destIP,
    "srcPort":srcPort,
    "dstPort":dstPort,
    "protocol":protocol,
    "packets":packets,
    "byte":byte,
    "startTime":startTime,
    "timeTook":timeTook,
    "status":status
})

df.columns

df.pop('date')
df.pop('time')
df.pop("account")
df.pop("interface")
df.pop('srcPort')
df.pop('dstPort')
df.pop('protocol')
df.pop('packets')
df.pop('byte')
df.pop('timeTook')

# Reformat numbers
df['startTime'] = pd.to_numeric(df['startTime'])
df['status'] = pd.to_numeric(df['status'])

df.head()

df = df.sort_values(['startTime','sourceIP', 'destIP',],ascending=[
   True, True, True])
df = df.reset_index(drop=True)
df.head()

#preprocess the data
#O(n)
processedDf = pd.DataFrame({
    "sourceIP":[],
    "destIP":[],
    "numOfRequest":[],
    "time":[],
    "acceptanceRate":[],
})
for index, line in df.iterrows():
    if index == 0:
        m = 100
        if line['status'] == 0:
            m = 0
        processedDf.loc[0] = [line['sourceIP'],line['destIP'],1,line['startTime'],m]
    else:
        n = len(processedDf)-1
        if processedDf.loc[n,'sourceIP']==line['sourceIP'] and processedDf.loc[n,'destIP']==line['destIP'] and processedDf.loc[n,'time']==line['startTime']:
            x = processedDf.loc[n,'numOfRequest']
            processedDf.loc[n,'numOfRequest'] = x + 1
            if line['status'] == 1:
                processedDf.loc[n,'acceptanceRate'] = (((x*processedDf.loc[n,'acceptanceRate']*0.01)+1)/(x+1)) * 100
            else:
                processedDf.loc[n,'acceptanceRate'] = (((x*processedDf.loc[n,'acceptanceRate']*0.01))/(x+1)) * 100
        else:
            m = 100
            if line['status'] == 0:
                m = 0
            processedDf.loc[n+1] = [line['sourceIP'],line['destIP'],1,line['startTime'],m]
processedDf.head()

processedDf.to_csv('processed_data.csv')

df = pd.read_csv('processed_data.csv')
df.pop('Unnamed: 0')
df.head()

from numpy import nan
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans

X = df.loc[:, ['numOfRequest', 'acceptanceRate']]
kmeans = KMeans(n_clusters = 3)
kmeans.fit(X)
df['is_attack'] = kmeans.labels_
df['is_attack'].value_counts()

filtered_label0 = df[df['is_attack']==0]
filtered_label1 = df[df['is_attack']==1]
filtered_label2 = df[df['is_attack']==2]

#Plotting the results
plt.scatter(filtered_label0['time'] ,filtered_label0['numOfRequest'], color = 'green')
plt.scatter(filtered_label1['time'] ,filtered_label1['numOfRequest']
     , color = 'yellow')
plt.scatter(filtered_label2['time'] ,filtered_label2['numOfRequest']
     , color = 'red')

plt.show()
plt.savefig('cluster_graph_1.png', dpi = 300)

from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
y = df['is_attack'].astype(int)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =0.33)
clf = LogisticRegression(random_state=0, verbose=1).fit(X_train,
    y_train)
y_pred = clf.predict(X_test)
print(classification_report(y_test,y_pred))

df['is_attack_supervised'] = clf.predict_proba(X)[:,1]
df.head()

suspicious_activity = df.sort_values(by='is_attack_supervised',
    ascending=False)
suspicious_activity = suspicious_activity[suspicious_activity['
    is_attack']==2]
suspicious_activity.head()

grayArea = df[df['is_attack']==1]

X = grayArea.loc[:, ['numOfRequest', 'acceptanceRate']]
kmeans = KMeans(n_clusters = 2) # is attack or not attack into 2
    clusters
kmeans.fit(X)
grayArea['possible_sus'] = kmeans.labels_
grayArea['possible_sus'].value_counts()

filtered_label0 = grayArea[grayArea['possible_sus']==0]
filtered_label1 = grayArea[grayArea['possible_sus']==1]

#Plotting the results
plt.scatter(filtered_label0['time'] ,filtered_label0['numOfRequest']
     , color = 'orange')
plt.scatter(filtered_label1['time'] ,filtered_label1['numOfRequest']
     , color = 'brown')

plt.show()
plt.savefig('cluster_graph_2.png', dpi = 300)

grayArea.head()

suspicious_activity.count()
```

```

grayAreaSus = grayArea[grayArea['possible_sus'] == 1]
grayAreaSus.pop('possible_sus')

suspicious_activity = pd.concat([suspicious_activity, grayAreaSus],
    ignore_index=True)
suspicious_activity['likelihood'] = suspicious_activity['
    numOfRequest']/suspicious_activity['numOfRequest'].max()
suspicious_activity = suspicious_activity.sort_values(by='likelihood
    ',ascending=False)
suspicious_activity.count()

suspicious_activity['likelihood'] = suspicious_activity['likelihood'
    ] * 100
suspicious_activity.sort_values(by='numOfRequest',ascending=True)

suspicious_activity['time'] = pd.to_datetime(suspicious_activity['
    time'],unit='s')
df['time'] = pd.to_datetime(df['time'],unit='s')

plt.scatter(suspicious_activity['time'], suspicious_activity['
    likelihood'], marker='.', edgecolors='white')
plt.xlabel("time")
plt.ylabel("likelihood of attack")
plt.show()
plt.savefig('suspicious_activity.png', dpi = 300)

number_of_suspicious = 0
for index, line in suspicious_activity.iterrows():
     number_of_suspicious = number_of_suspicious + line["numOfRequest
    "]
print("There are total number of " + str(number_of_suspicious) + "
    suspicious activities among the overall " + str(len_of_data) + "
    number of responses. ")
print("The number of unique suspicious activities are " + str(len(
    suspicious_activity)) + " with the highest duplicate of " + str(
    suspicious_activity["numOfRequest"].max()) + " among them.")

df.to_csv('result.csv')
suspicious_activity.to_csv('suspicious_activity.csv')

#end
```

Listing 1. CTML Code