IS734 – Data Analytics for Cybersecurity
Prediction of the characteristics of cyber attacks on IoT devices

Team Members:

Anthony McKinzie
Esha Singh
Lewis Davi
Taruna Sanjay Pakhare

November 8, 2023

Problem Statement:
Many IoT devices are vulnerable to cyber attacks due to their lack of consideration towards security. Methods to detect these cyber attacks currently need to be improved. We will attempt to leverage machine learning to identify the common characteristics of cyber attacks by analyzing network traffic on IoT devices. Then we will predict possible attacks by matching those characteristics to future network traffic.

Dataset (Anthony):
Before analyzing the data in Weka, we needed to make some formatting adjustments. The data came with quotes surrounding all nominal values which caused Weka to recognize the value as NULL. The data loads in as one of three types nominal, string, or numeric. To preprocess the data of certain algorithms, certain data types had to be converted. For example, J48 required the conversion of all data to a nominal format.

Methods:
J48 Decision Tree(Esha):
J48 is an algorithm for classification that is implemented in Weka. It is based on the C4.5 decision tree algorithm, which was developed by Quinlan (1993). J48 constructs a decision tree by recursively partitioning the training data into smaller subsets. At each partition, the algorithm selects the attribute that best separates the data into different classes. The algorithm then creates a node in the decision tree for that attribute and recursively partitions the data for each value of the attribute.

J48 can be used to classify data by training the algorithm on a subset of the data and then using the trained model to classify the remaining data. The algorithm can also be used to identify important features in the data by using the information gain measure. Information gain measures the amount of information that is gained by splitting the data on a particular attribute. Attributes with high information gain are more important for classification.

Here are some of the functions of the J48 algorithm:

- Classification: J48 can be used to classify data into different categories.
- Feature selection: J48 can be used to identify important features in the data.
- Error estimation: J48 can be used to estimate the error rate of a classification model.

J48 is a powerful and versatile algorithm that can be used for a variety of tasks in machine learning.

K-Means Clustering - Lewis

K Means clustering was initiated in Weka and using Python coding in Jupyter Notebook.
K-Means clustering does not allow for string values, creating considerable headache formatting the data as it is such a large dataset. Clustering was performed on 2 versions of the dataset;

- Version 1 was formatted using Python removing all string features and normalization preprocessing.
- Version 2 was formatted in Weka using StringtoNumeric preprocessing, and normalization preprocessing.

“ & ‘ symbols created null values within fields so this also needed to be formatted and removed during preprocessing. Feature selection preprocessing methods so far have been minimal due to attempting to gather as much data and visualizations as possible to identify any peculiarities or interesting trends/relationships that may not be obvious.

Apriori:
The Apriori algorithm is used in Weka for the "Edge-IIoTset Cyber Security Dataset of IoT & IIoT" project to identify frequent itemsets and association rules among the dataset's features. Frequent itemsets are sets of features that appear together frequently in the dataset, while association rules are rules that describe the relationships between frequent itemsets. These rules can be used to identify patterns in the data that may be useful for cybersecurity analysts, such as identifying common attack patterns or vulnerabilities.

The Apriori algorithm works by iteratively generating larger itemsets from smaller itemsets. At each iteration, it identifies the itemsets that appear together frequently enough to be considered frequent. These frequent itemsets are then used to generate larger itemsets in the next iteration. The process continues until no more frequent itemsets can be generated.

Once the frequent itemsets have been identified, the Apriori algorithm generates association rules from them. An association rule is a rule that describes the relationship between two itemsets. The rule has the form X --> Y, where X is the antecedent and Y is the consequent. The rule means that if X occurs, then Y is likely to occur as well. The strength of the rule is measured by its support and confidence. Support is the proportion of cases in the dataset that contain both X and Y. Confidence is the proportion of cases that contain Y, given that they contain X.

The Apriori algorithm can be used to identify frequent itemsets and association rules in a variety of datasets. In this project, the algorithm is used to identify patterns in the data that may be useful for cybersecurity analysts.

Here are some of the specific functions of the Apriori algorithm in this project:

- Identify frequent itemsets among the dataset's features.
- Generate association rules from the frequent itemsets.
- Measure the strength of the association rules using support and confidence.
- Identify patterns in the data that may be useful for cybersecurity analysts.

Preprocessing:

Preprocessing is an essential step in data preparation for machine learning tasks, and Weka provides a comprehensive set of tools for this purpose. In the context of the "Edge-IIoTset Cyber Security Dataset of IoT & IIoT," preprocessing plays a crucial role in ensuring the data quality and suitability for intrusion detection algorithms.

The primary functions of preprocessing in Weka for this project include:

1. Data Cleaning: Identifying and handling missing values, outliers, and inconsistencies in the dataset to ensure data integrity and reduce noise.

2. Data Transformation: Converting categorical data into numerical representation, such as one-hot encoding or label encoding, to make it compatible with machine learning algorithms.

3. Feature Scaling: Scaling numerical features to a common range, such as min-max normalization or standardization, to improve the performance of gradient-based optimization algorithms.

4. Dimensionality Reduction: Reducing the number of features by eliminating redundant or irrelevant features to improve computational efficiency and prevent overfitting.

5. Normalization: Converting numerical features to a common distribution, such as z-score normalization, to ensure equal contribution of each feature to the distance calculation in algorithms like k-nearest neighbors.

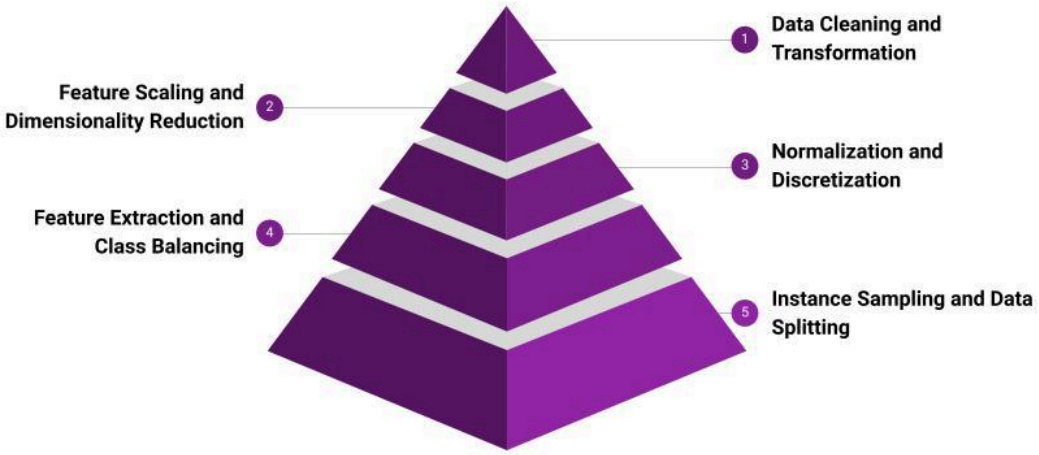
6. Discretization: Converting continuous numerical features into discrete intervals to make them compatible with algorithms that operate on nominal or ordinal data.

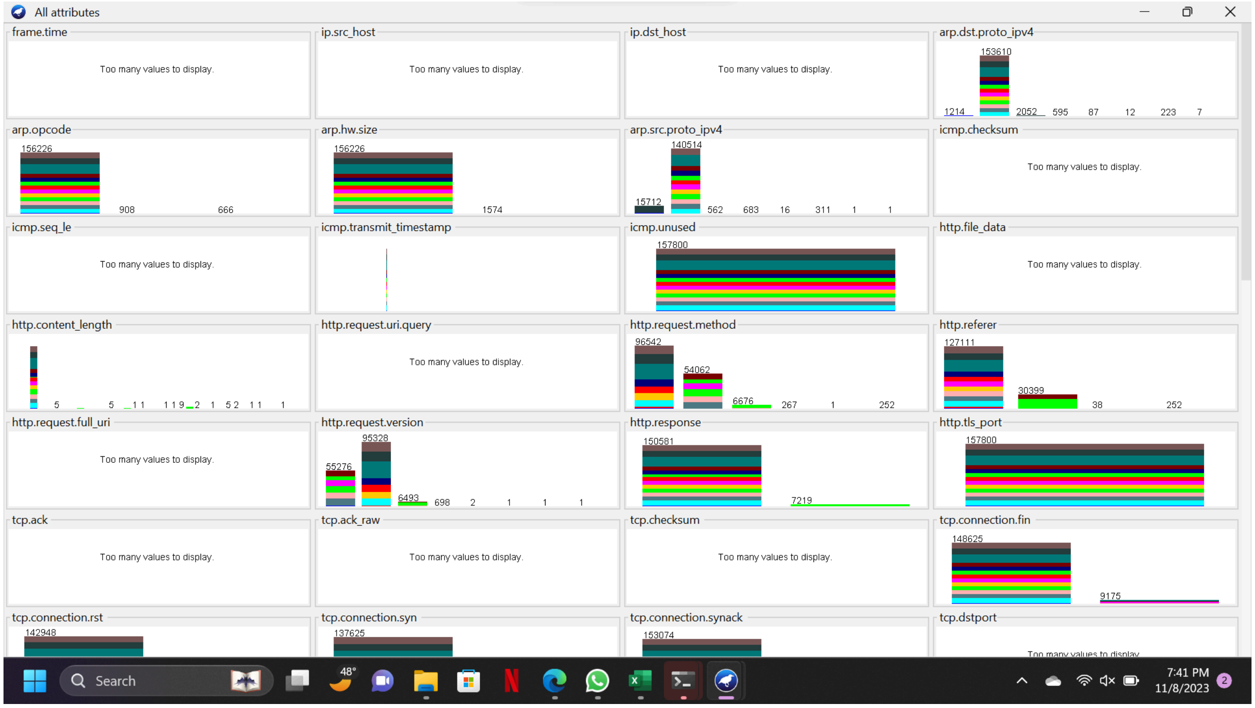
7. Feature Extraction: Creating new features from existing ones to capture more complex relationships or patterns in the data.

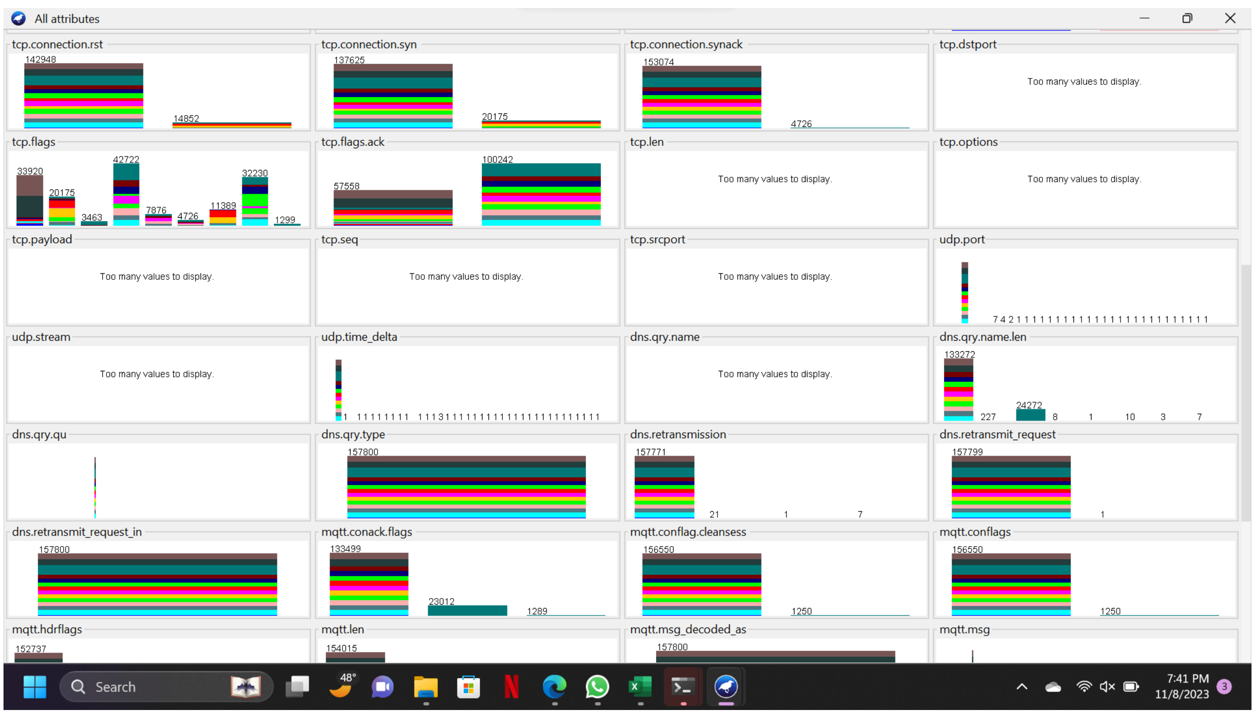
8. Class Balancing: Addressing the imbalance between normal and attack classes in the dataset to prevent biased classification models.

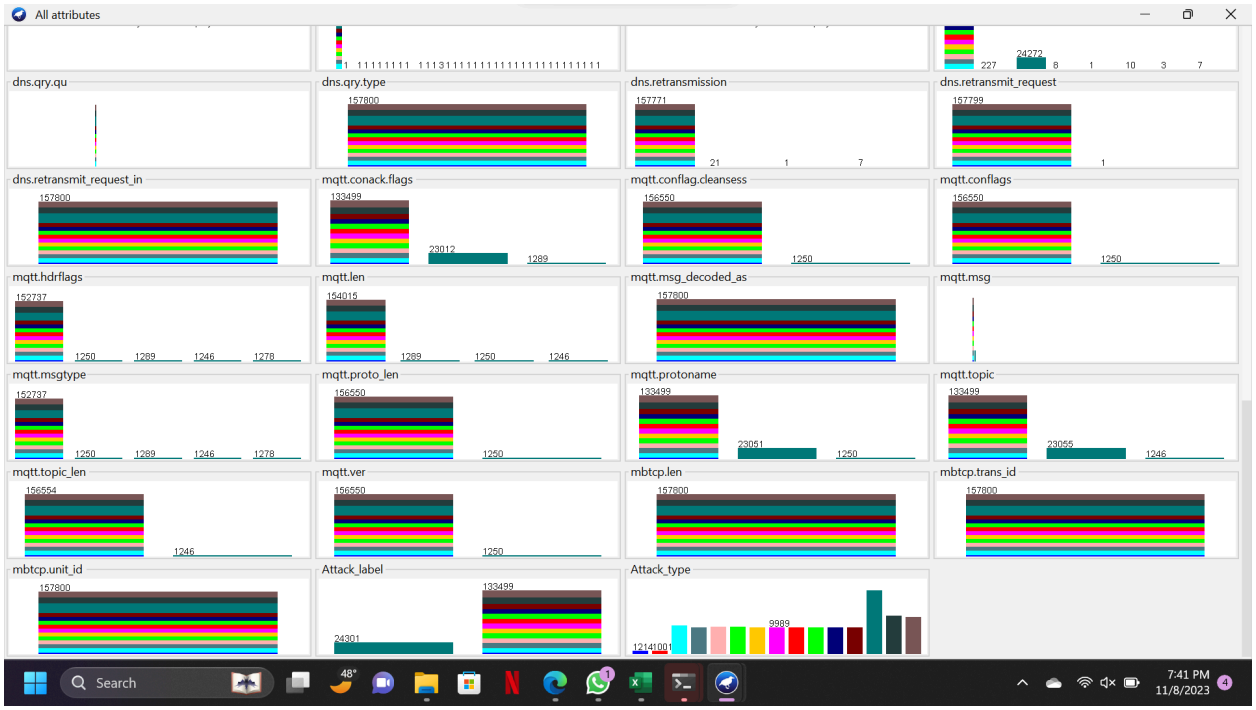
9. Instance Sampling: Up- or down-sampling instances to balance the class distribution and improve the detection of minority classes.

10. Data Splitting: Dividing the dataset into training, validation, and testing sets for model building, evaluation, and performance assessment.


1
Data Cleaning and Transformation
2
Feature Scaling and Dimensionality Reduction
3
Normalization and Discretization
4
Feature Extraction and Class Balancing
5
Instance Sampling and Data Splitting


All attributes
frame.time
Too many values to display.
ip.src_host
Too many values to display.
ip.dst_host
Too many values to display.
arp.dst.proto_ipv4
153610
1214 2052 595 87 12 223 7
arp.opcode
156226
908 666
arp.hw.size
156226
1574
arp.src.proto_ipv4
140514
15712 562 683 16 311 1 1
icmp.checksum
Too many values to display.
icmp.seq_le
Too many values to display.
icmp.transmit_timestamp
icmp.unused
157800
http.file_data
Too many values to display.
http.content_length
http.request.uri.query
Too many values to display.
http.request.method
96542
54062
6676 267 1 252
http.referer
127111
30399
38 252
http.request.full_uri
Too many values to display.
http.request.version
95328
55276
6493 698 2 1 1 1
http.response
150581
7219
http.tls_port
157800
tcp.ack
Too many values to display.
tcp.ack_raw
Too many values to display.
tcp.checksum
Too many values to display.
tcp.connection.fin
148625
9175
tcp.connection.rst
142948
tcp.connection.syn
137625
tcp.connection.synack
153074
tcp.dstport
Search
48°
7:41 PM
11/8/2023


All attributes
tcp.connection.rst
142948
14852
tcp.connection.syn
137625
20175
tcp.connection.synack
153074
4726
tcp.dstport
Too many values to display.
tcp.flags
33920
20175
3463
42722
7876
4726
11369
32230
1299
tcp.flags.ack
57558
100242
tcp.len
Too many values to display.
tcp.options
Too many values to display.
tcp.payload
Too many values to display.
tcp.seq
Too many values to display.
tcp.srcport
Too many values to display.
udp.port
udp.stream
Too many values to display.
udp.time_delta
dns.qry.name
Too many values to display.
dns.qry.name.len
133272
24272
227 8 1 10 3 7
dns.qry.qu
dns.qry.type
157800
dns.retransmission
157771
21 1 7
dns.retransmit_request
157799
1
dns.retransmit_request_in
157800
mqtt.conack.flags
133499
23012
1289
mqtt.conflag.cleansess
156550
1250
mqtt.conflags
156550
1250
mqtt.hdrflags
152737
mqtt.len
154015
mqtt.msg_decoded_as
157800
mqtt.msg
Search
48°
7:41 PM
11/8/2023



w

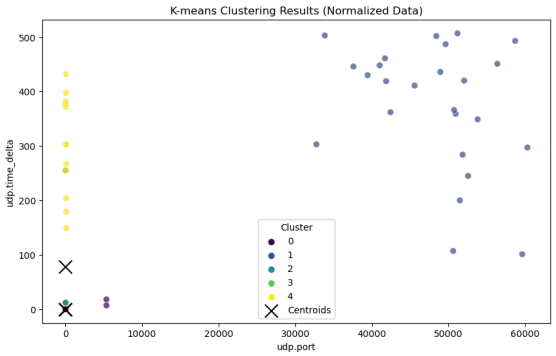
Results: w

J48 Decision Tree(Esha)

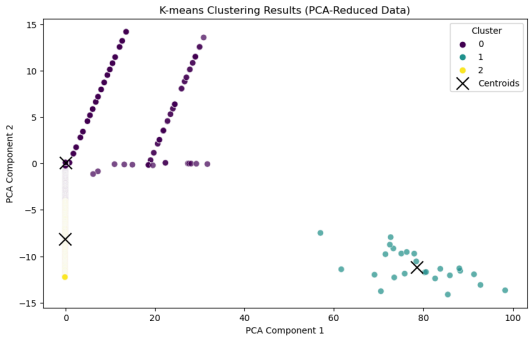
K-Means Clustering - Lewis

1. Dataset Version 1: Execution of python code to preprocess data (normalize, remove string columns, feature selection and run k-means clustering was initiated. I immediately saw clear issues with distribution of clusters and relationships due to the exclusion of the string features which do have relatable and distinguishable information between the data points that further help set differentiation and relationships/trends. Many of the features that remained as numeric/nominal find many duplicate values, but also some that vary in their own respect, so the strength of the remaining features used are not particularly the best. I improved on the plotting of the model to characterize clusters by appointing different features for the x and y axis. What I found worked best in creating a visual plot that distinguishes proper relationships between data points was using the Euclidean PCA distance component as the x and y axis to capture variance.

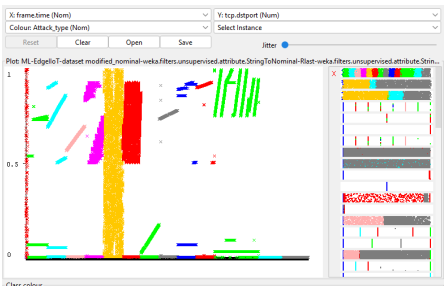
- K Means cluster with random features as x and y axis (not as good)

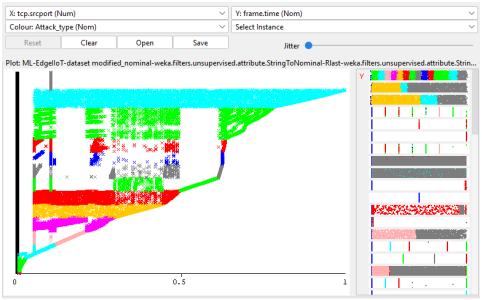


- K mean cluster plot with PCA component (better)



2. Dataset Version 2: Weka initiated preprocessing steps for evaluation using Stringto Numeric function and Normalization. After, K means filtered Clustering ran several plots and data findings were produced. I was able to locate varying features which help identify relationships and variance between the data that I will need to further explore. I need to continue to isolate feature selection to identify proper clustering of data points which help us identify irregular network activity. There were persisting issues with modifying the k means cluster to produce appropriate results as K means is not especially good for identifying outliers, so this may not be the best method to implement for final evaluations.





Apriori:(Taruna)

Challenges and Solutions (Esha):

Specific data formatting was needed to allow the large dataset to be analyzed by Weka.

The large number of data columns and rows makes it hard to analyze results when algorithms are run on the data set in its entirety. We must select only relevant attributes so that we can make meaningful conclusions about Weka's results.

Challenges:
The J48 decision tree algorithm, while useful for classification, can run into issues like overfitting, especially when dealing with imbalanced datasets. This means the tree can become too specific to the training data, potentially losing its generalization ability. On the other hand, decision trees, including J48, can sometimes be challenging to interpret due to their complexity. K-means clustering, another valuable technique, can encounter problems related to the initial placement of cluster centers. The results can vary depending on where the clusters start, making the algorithm sensitive to initialization.

K-means clustering is not well-suited for identifying anomalies or outliers within a dataset. Its primary purpose is to group similar data points, so it might not effectively highlight unusual or unexpected data instances. Apriori, a frequent itemset mining algorithm, can face computational challenges when applied to large datasets. It may demand significant processing power and time to generate frequent itemsets and association rules. Additionally, Apriori can generate an extensive number of rules, which, although informative, can be challenging to interpret and utilize effectively. Addressing these challenges often involves considering alternative algorithms or employing various techniques to mitigate these issues during the data analysis process.

Solution:
The J48 decision tree model can be harnessed to classify IoT sensor data, distinguishing between normal and anomalous readings. To ensure the model's accuracy and prevent overfitting, techniques like pruning and regularization can be employed. Furthermore, making the decision tree more understandable can be achieved through rule extraction and visualization methods.cess.

K-Means clustering serves as a valuable tool for identifying groups of IoT devices that exhibit similar behavior. These groups can be monitored for any unusual patterns or anomalies. To enhance the effectiveness of K-means clustering, techniques like k-means++ and fuzzy c-means clustering can be used to tackle the challenge of initializing cluster centroids. Additionally, for the detection of anomalies and outliers within the data, techniques such as isolation forests and local outlier factors can be brought into play.

The Apriori algorithm proves beneficial in generating association rules between various IoT sensor readings, enabling the detection of abnormal patterns in the data. To manage the computational complexity that can arise with this technique approaches like FP-growth and Eclat can be implemented. Furthermore, controlling the number of generated rules can be achieved by setting parameters like minimum support and minimum confidence. These methods

provide a toolkit for effectively harnessing machine learning for IoT data analysis while addressing specific challenges in the process.