COMP8325 Application of Artificial Intelligence in Cybersecurity

GROUP PROJECT

Adithya Sanjana Reddy Chilumula -46130713 and
Sai Divya Kandhadi-46724109

Macquarie University S1,2022

Table of Contents

Introduction

Machine learning is a type of artificial intelligence that allows computers to examine data, decide, and learn from their prior events. Machine learning techniques in cybersecurity assist security professionals save time by proactively spotting security issues and attacks, analysing them, and, in some situations, responding to them. Many modern security technologies include machine learning capabilities. Older reasoning approaches, such as manually established rules and analytical associations, are increasingly being phased out. The severity of cyber-attacks is increasing. Every year, thousands of millions of new malware variants are discovered. New malware programmes can run without needing binary files at all, avoiding detection by typical anti-virus software. Multilayered cyberattacks are becoming more common, combining network-based tactics, ransomware, and web server attacks. Insider threats are on the rise, and it can be difficult to tell the difference between regular user activity and insider attacks. Cell phones, switches, and routers in the workplace and in society, and IoT infrastructure are all being used by cybercriminals to carry out huge attacks. Most of these emerging dangers can be detected and mitigated using artificial intelligence (AI) technologies based on machine learning techniques. They can evaluate a far bigger quantity of data than people, automatically detect abnormal and strange activity, and explore risks by connecting many datasets.
Machine learning has four uses in cybersecurity.
1. Identifying threats to the network:
Machine learning can be used to examine massive volumes of externally and internally network traffic and spot trends that could indicate a security breach.
2. Application security that is automated:
Machine learning can be used by automated testing encryption technology to detect unusual traffic and stop it or respond to a threat immediately. They are capable of detecting malicious activity such as illegal access and the abuse of access privileges. They can also use dynamic or static code review to automatically find and avoid application vulnerabilities.
3.Monitoring emails:
Existing systems for detecting spam, viruses, and social control in email messages can be improved with machine learning. Pictures or even other files to emails can be classified using machine learning to assist detect dangers. Natural language processing (NLP) techniques look at the content in emails to see if they're parts of a phishing scheme, and they look at the links to see if they're safe.
4. Antivirus software of the future:
Antivirus software traditionally relies on database of known malware strains. This method is ineffective against zero-day malware, which are new infections that have yet to be discovered by researchers or antivirus vendors. It's also prone to malware "mutations," which occur when a virus is purposefully changed to avoid detection. Modern AI-based analysis looks at the source code or activities of malware to see if it's safe or if it's doing anything dangerous. Many businesses are establishing security operations centres (SOCs) with security information and event (SIEM) at the centre. A SIEM collects, analyses, and analyses safety data and information across the company in order to correctly assess and mitigate threats. Machine learning is used in current SIEM systems in several methods to better locate threats more correctly and react to it quicker, Malicious URL and probing plays a major role so in our project have choose these topics. (Orion, 2020) There are large numbers of URL links on a web application. Malicious actors may insert a brief iFrame into one of the websites in order to redirect readers to a malicious website. One malicious URL can infect a significant number of user machines in minutes if the affected domain is a prominent website with

millions of daily visitors. As a result, guarding against these malicious connections aids in the prevention of daily malware outbreaks. (Gupta, Gupta and Kukreja, 2021)

Phishing has become the most famous and effective attack method in recent years, allowing attackers to gain usernames and passwords, credentials, and financial data from users, as well as install malware or malicious software on the target's system, by impersonating a trustworthy source. (Shaik et al, 2021)

We have chosen Malicious URL detection and probing/port scan detection datasets for our group project. The order of our project is as follows: an introduction about the cyber security in machine learning and their needs and mentioned the uses of machine learning for cyber security. We have given a brief description about the literature review, Machine learning for cyber security applications and Adversarial machine learning in next step mentioned the dataset descriptions as we have chosen the malicious URL detection and probing/port scanning. We have used the anomaly models like Random tree classifier, decision tree, Isolation forest and explained the analysis and performance indicators of the machine learning models.

Literature review

Machine learning for cyber security applications

Machine learning, a subfield of artificial intelligence, creates presumptions about a machine's behaviour using formulae derived from past records and observable analyses. The machine would thereafter be able to change its actions and even do tasks for which it had not been explicitly adjusted. Machine learning is increasingly being used to detect threats and organically eliminate those before they could even cause havoc, thanks to its ability to recognize a large number of data and discriminate against potentially dangerous ones (Anish et al, 2021). The popularity of machine learning is attributed to two key factors: First, it could automate operations that formerly required human intervention, such as robotic mechanism control in manufacturing. Second, it can handle and analyse large volumes of data fast, as well as generate choices based on a variety of variables. In several domains, machine learning outperforms humans in terms of quality. Therefore, ML has a lot to offer in terms of cybersecurity (Avdoshin et al., 2019).

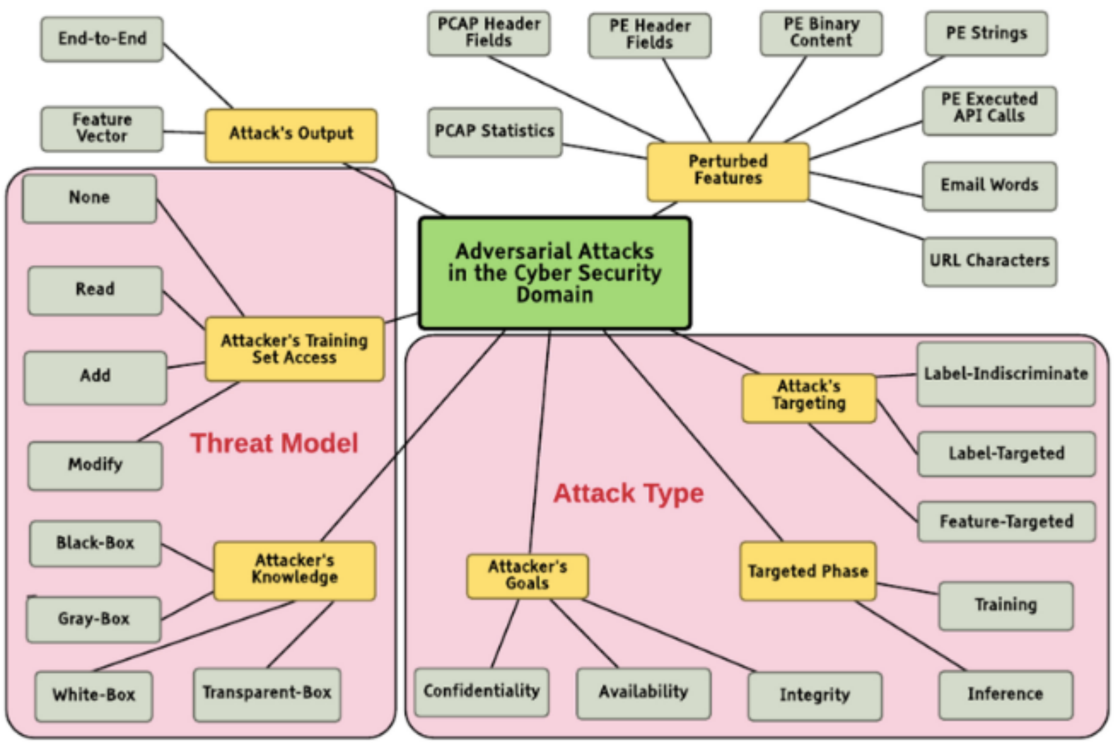
Looking at the multiple advantages, this project attempts to use Machine learning models to combat various types of cyber-attacks. For this project we have mainly focused on detecting malicious URL's and port scan detection. We have used various anomaly detection models such as Random Forest, Isolation Forest, SVM, K-neighbors Classifier, Decision Tree Classifier etc. Some of the most useful applications of Machine learning in cybersecurity are in detecting phishing emails and spam messages, identifying malware, detecting anomalies and vulnerabilities in software, ddos attack detection etc.

Machine learning models need to be trained using high-quality datasets in order to improve predictions and responses to assaults. Companies, on the other hand, do not disclose their data in general since it is sensitive, particularly when it comes to cybersecurity. However, we may solve this problem by anonymizing sensitive data in datasets and sharing them with the community (Amit et al, 2018). To prevent insider threats and respond to them in an automated manner, machine learning models monitor the behaviour of traffic flow and other variables such as changes in customer authentication and authorization. It is equally crucial to protect the organization from both external and internal dangers. The range of applications for machine learning in cybersecurity is vast, ranging from recognizing anomalies and unusual or suspicious behaviour to finding zero-day vulnerabilities and fixing known ones. (Martin, Rahouti, Ayyash and Alsmadi, 2021) To this report, the application of ML in preventing probing attacks will be discussed in depth as we have used the probing dataset in this project.

Cyber attackers frequently use port scanning and DDoS cyberattacks to check for vulnerabilities and attack the target's resources, respectively. A scanning tool discovers open ports in the target, informs about running services, and enumerates target information such as the operating system being used, memory utilized, and processing data while port scanning is in progress. The intention of port scanning would be to find susceptible sections of the target where exploitation might be feasible. The most common tool used to perform port scanning is Nmap. Port scanners send the request to a TCP or UDP port, either remote or local and watch for a response. They then mark the host's answer as 'open,' 'filtered,' or 'closed'(Aamir et al., 2021). Scanning for the 3306 port, for example, after installing and starting a MySQL server to determine if it's displaying as 'open' is a common use case for port scanning. In simple terms, port scanner sends network packets and waits for a response.

The security vulnerabilities detected by port scanning could be used by cybercriminals if the vulnerabilities are not patched before they are detected by cybercriminals. Attackers can use malicious code injections to raise their privileges in the network, which can lead to a data breach or other assaults. This could compromise the cybersecurity of the entire organization and lead to financial and reputational damage of the company. Machine learning can be used to detect anomalies and patterns in huge datasets to detect such vulnerabilities to avoid cyberattacks (Dasgupta and Akhtar, 2022). Anomaly detection involves things like API user behaviour patterns, security rules, and input validation, among other things, and classification is performed with algorithms like KNN. Machine learning techniques like classification evaluate vulnerable sections of code utilizing features like as HTTP requests and system calls that are initiated by syntax trees, among other things. Machine learning techniques such as Isolation Forest, Random Forest, Decision Tree Classifier and others are used to classify vulnerabilities.
Both Supervised and unsupervised machine learning models can be used to train and test the data which use labelled and unlabelled data respectively. However, To avoid the issues that exist in individual approaches, a hybrid model integrating unsupervised and supervised learning is preferable. This is accomplished by integrating the entropies of network features, resulting in high detection accuracy. (Kim and Kim, 2018) Overall, a review of relevant work and our project suggests that machine learning algorithms are very useful in detecting probing and malicious URLs.

Adversarial machine learning (security issues of machine learning models)



Machine Learning Adversarial Attacks and Countermeasures (Rosenberg, Shabtai, Elovici and Rokach, 2022).

Adversarial machine learning is the modelling of dynamic adversarial setups such as spam detection or threat detection, where a malicious user can meticulously deceive (or disrupt) the information, exploiting specific vulnerabilities in learning algorithms to compromise the security of the (targeted) machine learning system.

Machine learning algorithms, particularly deep learning techniques, have become increasingly popular in recent years in a variety of sectors, including cyber security. Machine learning systems, on the other hand, are prone to adversarial threats, which restricts their use, particularly in dynamic, adversarial contexts like cyber security, where true adversaries (e.g., virus authors) exist (Rosenberg, Shabtai, Elovici and Rokach, 2022). The computer vision domain has been the subject of the majority of adversarial attacks disclosed, including those presented at scholarly cyber security conferences. However, the cybersecurity realm (e.g., malware detection) appears to be a much more pertaining realm for adversarial attacks as there is no actual adversary in the computer vision realm (with a few exceptions, such as criminals who intend to intrude on autonomous vehicles' pedestrian detection methods to cause false propaganda or financial fraud. Ransomware creators, for example, rely on their ransomware's ability to avoid anti-malware programs that would prohibit both execution and decryption. (Rosenberg, Shabtai, Elovici and Rokach, 2022)

Adversaries, or malicious actors, may purposefully distort the models' training data in order to exploit certain flaws and execute out cyber-attacks. Security violation,
Influence and, specificity are the three elements that define Adversarial Learning.
We concentrate on influence since it is the most studied aspect of Adversarial Learning, and it is the most important for port scanning attacks. It defines two sorts of adversarial attacks: probing (exploratory) and causative. The adversary is described based on their behaviour as they try to uncover weaknesses by monitoring the result of learner's classifications of large datasets in exploratory attacks in order to construct suitable learning techniques.

Intrusion detection systems use machine learning algorithms to tackle DDOS and port scanning attacks. Port scanning comes under supervised learning category and is often detected by IDS since the log the IP address of the sender. IDS is focused on attack patterns that necessitate domain and network behaviour analysis skills. This increases dependency on huge datasets and domain knowledge for analysis purposes. Therefore, the adversary might target training and testing datasets to decrease its accuracy and efficiency.
One of the innovative ways that IDS systems can be used to combat port scanning is the use of a hybrid of Unsupervised and supervised Learning techniques called Deep Belief Networks, that identify signs of port scanning attacks by analysing network data. It's a deep neural model where the network layers provide data output with clear indicators from the same category of complicated structured data. It contains a two-phase training method, with the first phase being unsupervised learning using an unconstrained training mode based on the Restricted Boltzmann Machine (RBM), and the second phase being the Supervised Learning process (Viet, Van, Trang and Nathan, 2018).

Data sets description:

Malicious URL detection

Even though today's security features attempt to prevent suspicious sites and domain names, the attackers are able to evade detection by employing various ways. Researchers have investigated Malicious URL Detection to come up with useful solutions. The blacklist method, which employs records of known harmful URLs to filter incoming URLs, is one of the most common. Blacklists, on the other hand, have some restrictions, rendering this strategy worthless in the face of new dangerous websites being launched on a regular basis. During the recent decade, security components have begun to apply novel machine learning and machine intelligence prediction algorithms to deal with this problem. For Malicious URL Detection, they have begun to prioritize machine learning and ai predictions over signature-based detection as well as develop scanning engines(Ferhat et al,2021).

In the malicious URL detection, there are two data sets namely URL Data Set (Matlab), and URL Data Set (SVM-light) Before we go into our strategy, it's essential to understand the different sorts of malicious URLs. Not that all malicious URLs are created equal. In terms of their function, URLs used during malicious web pages, for example, are divided into three categories. To begin with, there are "landing URLs." This URL is quite close to the benign URL and is used to hide an attacker's identity. In terms of URL length and lexical format, they mimic benign URLs. Second, exploit toolkit "distribution URLs" are available. These URLs have a different lexical type than arriving URLs in terms of length and vocabulary format. Finally, the malware generates URLs as a result of drive-by downloads (Sungjin et al, 2022)

Probing/Port scanning

New and deadly cyber-attacks have heightened the necessity for comprehensive cybersecurity policies in recent years. Security Mechanisms like IDS, which can detect previously unknown intrusions, are needed to prevent innovative cyber threats.
Many academics have attempted to develop anomaly-based IDSs, but they have yet to be successful in detecting malicious network traffic frequently enough to be useful in actual networks. Clearly, producing IDSs that are adequate for use in the real world remains a problem for the security sector. Machine learning has proved to be quite effective in tackling a variety of complicated real-world issues, thus it's critical to examine how it may be used to identify port scanning attacks (Sungjin et al, 2022).

About probing/port scan detection dataset:
The probing/port scan detection dataset contains 3 csv files, namely malicious.csv, attack_labels.csv and normal.csv. The csv files contain pcap files generated by network devices. The dataset used in this project is owned by gubertoli, the malicious data set was generated by him using the network architecture as below directed towards the target machine having IP address 10.10.10.10.
The dataset utilised in this study by the machine learning models to understand the characteristics of malicious network traffic is the pcap file created on the internal router (router0).

Anomaly models

Finding anomalies or outliers, which are unexpected or irregular patterns in a dataset, is known as anomaly detection. These abnormal patterns may be detected using machine learning algorithms, which can then be classified into distinct breaches and cyberattacks. However not all anomalies constitute real intrusions, several techniques are employed to identify and distinguish them from malicious attempts.

Isolation forest

The isolation forest algorithm is a deep learning approach for detecting anomalies.
It's an unsupervised machine learning approach for detecting anomalies in data by isolating outliers.

The Decision Tree method is used in Isolation Forest. It separates outliers by selecting a feature at random from a feature set and then choosing a split number between the feature's max and min values at random. The abnormal data points will be distinguished from the entire data by this random division of features, which will result in shorter paths in trees.
The Isolation Forest technique is based on the notion that anomalies are made up of a small number of distinct observations, making them clearer. To isolate anomalies, Isolation Forest employs an ensemble of Isolation Trees for such provided data points.
Isolation Forest builds divisions on the dataset recursively by selecting random a feature and then a divide value for the value. Assuming that anomalies require fewer random partitions to be separated than "regular" locations in the dataset, anomalies will be the points in the tree with the shortest path length, path length being the set of edges crossed from the root node. (Ferhat et al, 2020)

Random Forest tree

Random Forest (RF) is an ensemble learning method that uses a clear majority to collect the results of numerous decision trees. The outputs of several classifiers are combined in deep learning, and a single decision has been made on behalf of the people. The bootstrap method is used to generate each decision tree in the forest by picking multiple samples from the dataset. The group then votes on the choices taken by many various individual trees and presents the class with the most votes as the committee's class estimate. CART (Classification and Regression Trees) algorithms and the boot bagging combination method are used to generate trees in the RF approach. This is divided into training and test data. Then in the training data set samples are collected and used in the bootstrap technique, the data will be in form of trees. In the RF model, it is started with a single node and the final node ends as a leaf and a class label is given in that. The main advantage of the RF is it can handle a large dataset also and this can also perform both Classification and Regression tasks.

Decision tree

In machine learning, a decision tree is a core component of a classification method that also effectively addresses to regression problems that use the classification rule. Its structure is like a flowchart, with each node represents representing a feature test, each leaf node representing the class label, and the branches representing conjunctions of characteristics that direct to the class labels. They are two different steps to build a Decision Tree.

1. Terminal node creation

2. Recursive splitting

They are advantages of using decision tree in Machine Learning:
Comprehensive- It considers all-possible results of a decision and traces every node to the optimal solution.
Versatile- Decision Trees can be built directly using mathematics and can also be used in conjunction with other computer-based systems.

Simplicity-Because it lacks complex equations or data structures, Decision Tree is among the simpler and more dependable algorithms. The calculation only necessitates basic statistics and math.

Specific- Each issue, decision, and result in a Decision Tree is assigned a specific value. It minimizes uncertainty and ambiguity while simultaneously increasing clarity.(Priya, 2020)

K-Nearest Neighbor (KNN):

The K-Nearest Neighbour algorithm depends on the Supervised Learning method and is one of the most basic Machine Learning techniques.

The K-NN method assumes that the new specific instance and existing cases are similar and places the new case in the group that would be most similar to existing categories. The K-NN method stores all available data and identifies a given data point depending on its similarities to the existing data. This means that new data can be quickly sorted into a well-defined category that use the K-NN method. The K-NN algorithm could be used for both regression and classification, but it is more commonly utilised for classification tasks.

Main advantages of K-Nearest Neighbor(KNN): It is simple to put into action. It can withstand noisy training data. If the training data is too large, it may be more effective.
(Java T point)

Source Code Description for Malicious URL Detection

In the first step of detecting the malicious URL with various Machine learning algorithms reading the data from the given dataset and checking the null values in the dataset from URL and Label, classified the data as good and bad in the dataset and plotted a data against good vs bad counts in the dataset and plotted in a bar graph, good is highest in the count.

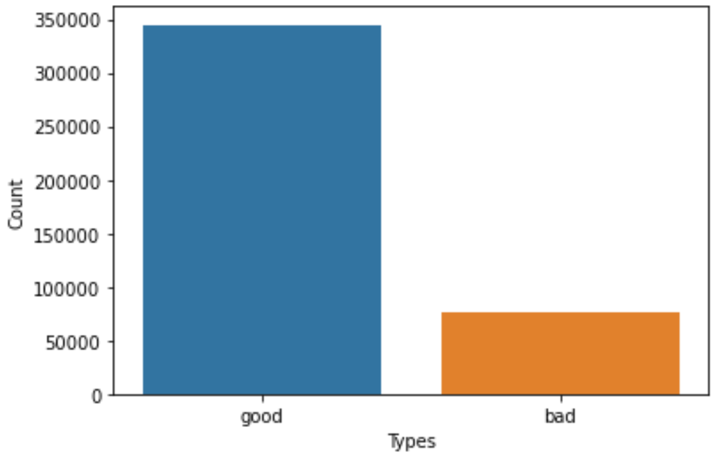


Figure: plotting data against the good vs bad counts in the given dataset.

In features extraction for the given dataset, replaced WWW with the empty string for further processing of data and then checked the URL length and verified the domain which belongs to it. Verified the URL which is containing the special feature '@','?','-','=','.','#','%','+','$','!','*',',','.
In the next step parsing the URL whether it contains any abnormal with urllib.parse library and plotting the bar graph for the abnormal URL. We need to parse URL to whether it contains any HTTP or http keywords with urllib.parsse library and plot the bar graph for http. In the next step we need to check for the digit and letter counts in the url in the given dataset, Shortening service, must be verified in the dataset, In URL, the shortener can reduce the length of the URL(Uniform Resource Locator) It minimizes the web page address into an easy way to remember like Bit.ly, shorte\.st …etc.

We must implement the shortening service and plot the bar checking for the IP address in the given dataset.

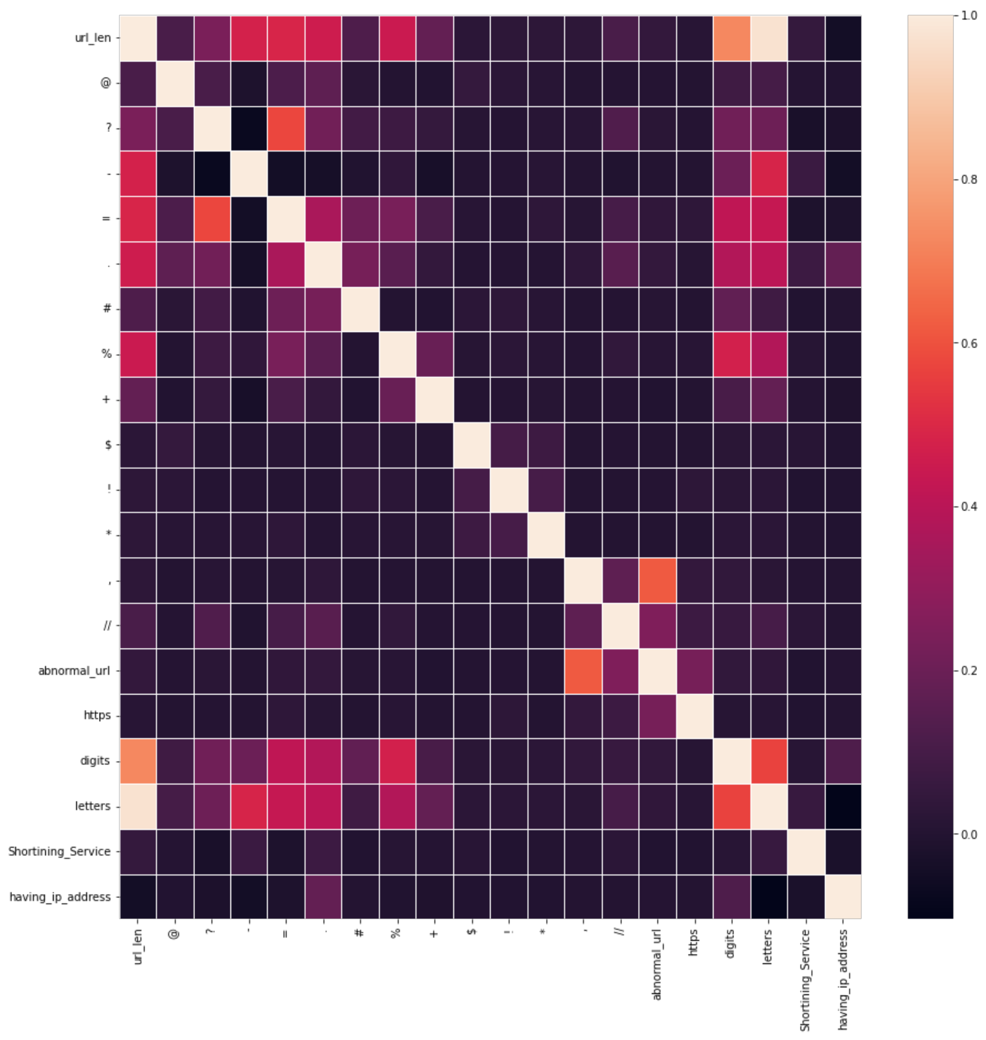


Figure: Heatmap of the features extracted from the given dataset.

In the following dataset is labelled good as 1 and bad as 0 under the category column. Data extraction with features for train and test purposes. In this Malicious URL detection used a Decision tree Classifier, Random Forest Classifier, Ada boost Classifier, SGD Classifier, Extra Tree classifier, Gaussian NB for classification with confusion matrix, and mentioned output of the algorithms with their respective accuracy for the given dataset. The Accuracy of Random tree Classifier is the highest with 0.89, the graph is attached below.

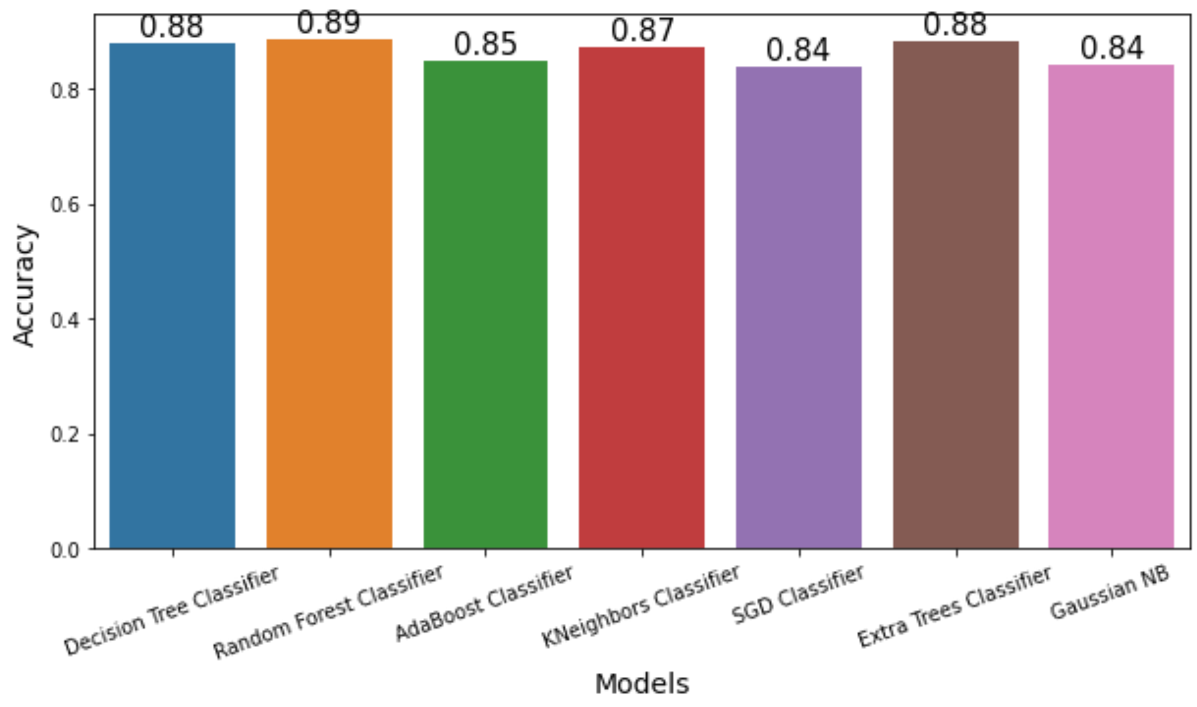


Figure: Accuracy of the Analogy models.

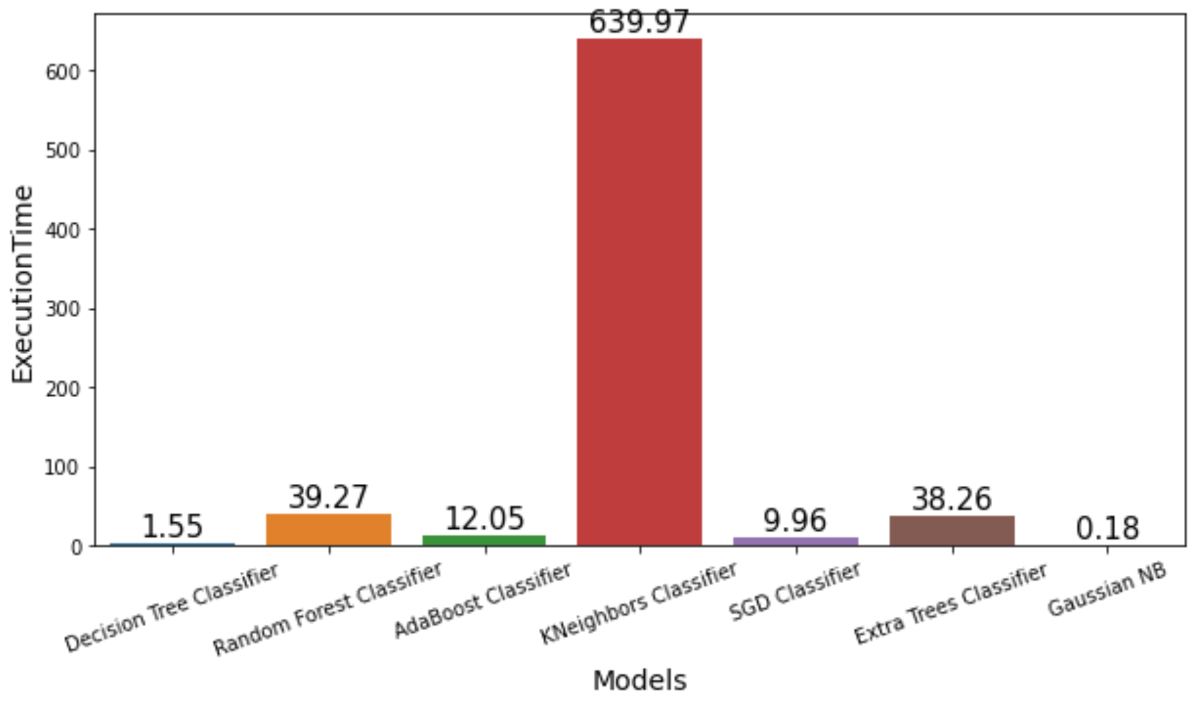


Figure: Execution time for the given dataset

From the above the graph as we have used Anomaly models like Decision Tree Classifier, Random Forest Classifier, AdaBoost Classifier, Kneighbors Classifier, SGD Classifier, Extra Trees Classifier, Extra Tree Classifier, Gaussian NB. Kneighbors is the highest among them with 639.97 Execution time.

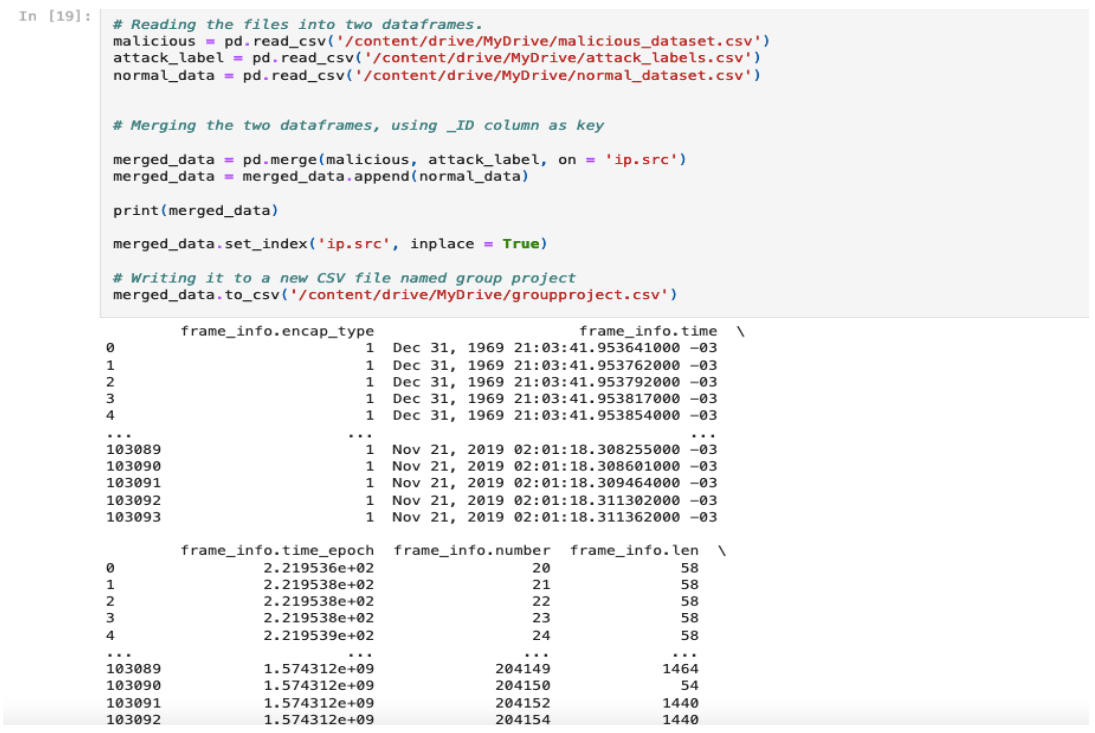
Source Code Description for Probing/Port scan Attack Detection

The following steps were implemented to obtain anomaly detection on the given probing/port scanning dataset:

Step 1: All the required python libraries and modules required to complete this machine learning task were imported.

Step 2: The datasets were stored in google drive and imported onto Google Collab for ease of execution.

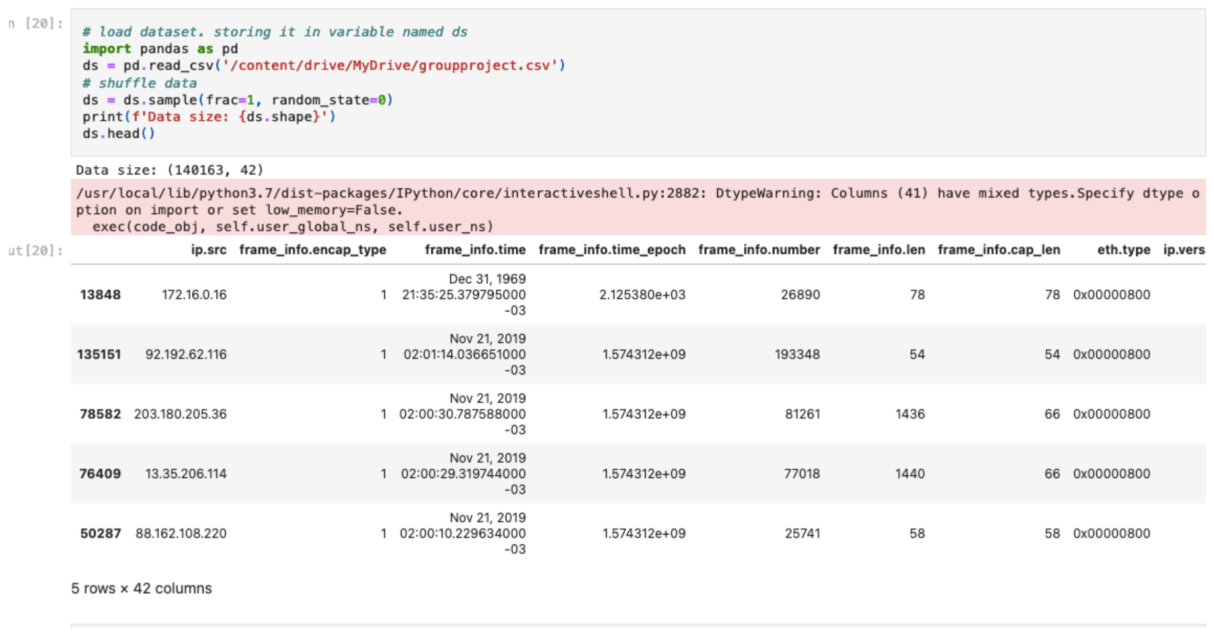
Step 3: The probing/port scanning dataset has 3 main csv files namely malicious.csv, normal.csv, attack_labels.csv containing the malicious dataset, normal dataset and attack labels respectively. These datasets are then merged to form a single large dataset. This merged dataset is written to a new csv file.



```
In [19]:
# Reading the files into two dataframes.
malicious = pd.read_csv('/content/drive/MyDrive/malicious_dataset.csv')
attack_label = pd.read_csv('/content/drive/MyDrive/attack_labels.csv')
normal_data = pd.read_csv('/content/drive/MyDrive/normal_dataset.csv')


# Merging the two dataframes, using _ID column as key

merged_data = pd.merge(malicious, attack_label, on = 'ip.src')
merged_data = merged_data.append(normal_data)

print(merged_data)

merged_data.set_index('ip.src', inplace = True)

# Writing it to a new CSV file named group project
merged_data.to_csv('/content/drive/MyDrive/groupproject.csv')
```

```
        frame_info.encap_type                        frame_info.time  \
0                           1  Dec 31, 1969 21:03:41.953641000 -03
1                           1  Dec 31, 1969 21:03:41.953762000 -03
2                           1  Dec 31, 1969 21:03:41.953792000 -03
3                           1  Dec 31, 1969 21:03:41.953817000 -03
4                           1  Dec 31, 1969 21:03:41.953854000 -03
...                       ...                                  ...
103089                      1  Nov 21, 2019 02:01:18.308255000 -03
103090                      1  Nov 21, 2019 02:01:18.308601000 -03
103091                      1  Nov 21, 2019 02:01:18.309464000 -03
103092                      1  Nov 21, 2019 02:01:18.311302000 -03
103093                      1  Nov 21, 2019 02:01:18.311362000 -03

        frame_info.time_epoch  frame_info.number  frame_info.len  \
0                2.219536e+02                 20              58
1                2.219538e+02                 21              58
2                2.219538e+02                 22              58
3                2.219538e+02                 23              58
4                2.219539e+02                 24              58
...                       ...                ...             ...
103089           1.574312e+09             204149            1464
103090           1.574312e+09             204150              54
103091           1.574312e+09             204152            1440
103092           1.574312e+09             204154            1440
```

Step 4: This merged csv file is loaded and stored in variable called ds. Then sampling and randomness is added to the data and then shuffled for uniformity.



```
n [20]:
# load dataset. storing it in variable named ds
import pandas as pd
ds = pd.read_csv('/content/drive/MyDrive/groupproject.csv')
# shuffle data
ds = ds.sample(frac=1, random_state=0)
print(f'Data size: {ds.shape}')
ds.head()
```

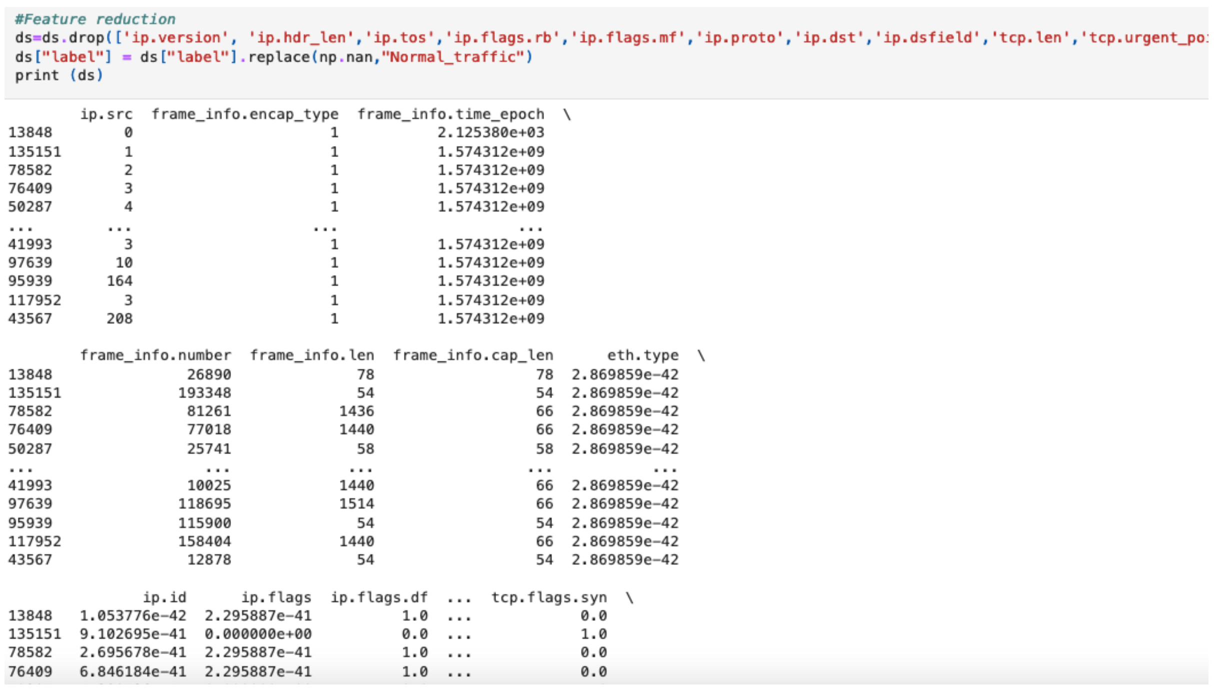
```
Data size: (140163, 42)
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2882: DtypeWarning: Columns (41) have mixed types.Specify dtype o
ption on import or set low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

ut[20]:

	ip.src	frame_info.encap_type	frame_info.time	frame_info.time_epoch	frame_info.number	frame_info.len	frame_info.cap_len	eth.type	ip.vers
13848	172.16.0.16	1	Dec 31, 1969 21:35:25.379795000 -03	2.125380e+03	26890	78	78	0x00000800	
135151	92.192.62.116	1	Nov 21, 2019 02:01:14.036651000 -03	1.574312e+09	193348	54	54	0x00000800	
78582	203.180.205.36	1	Nov 21, 2019 02:00:30.787588000 -03	1.574312e+09	81261	1436	66	0x00000800	
76409	13.35.206.114	1	Nov 21, 2019 02:00:29.319744000 -03	1.574312e+09	77018	1440	66	0x00000800	
50287	88.162.108.220	1	Nov 21, 2019 02:00:10.229634000 -03	1.574312e+09	25741	58	58	0x00000800	

5 rows × 42 columns

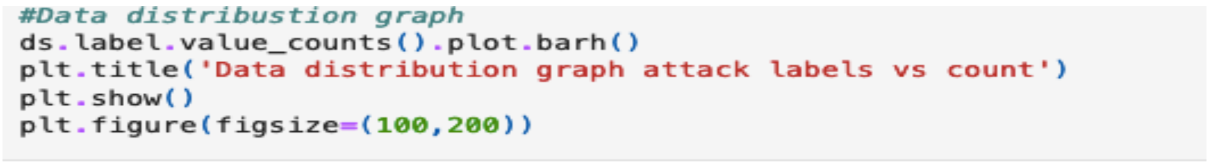
Step 5: Feature reduction is performed on the dataset and null values are deleted to improve the execution speed of the code.



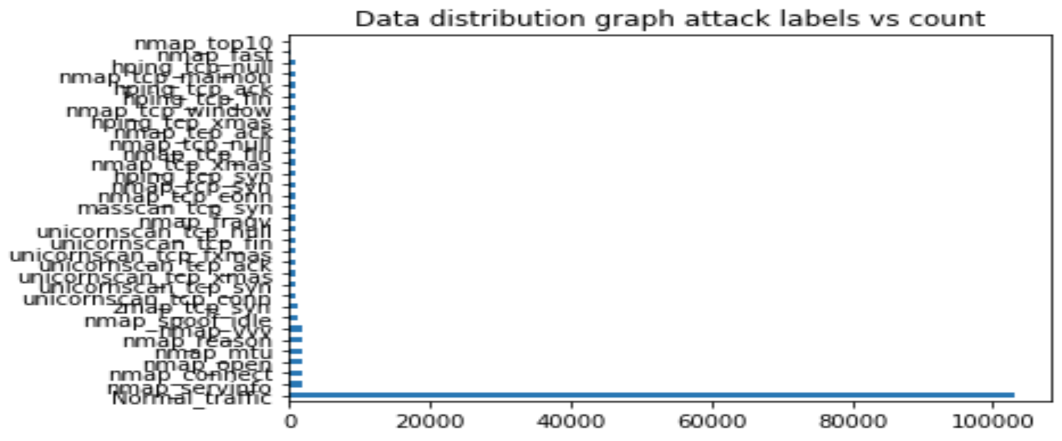
```
#Feature reduction
ds=ds.drop(['ip.version', 'ip.hdr_len','ip.tos','ip.flags.rb','ip.flags.mf','ip.proto','ip.dst','ip.dsfield','tcp.len','tcp.urgent_po
ds["label"] = ds["label"].replace(np.nan,"Normal_traffic")
print (ds)
```

```
        ip.src  frame_info.encap_type  frame_info.time_epoch  \
13848        0                      1           2.125380e+03
135151       1                      1           1.574312e+09
78582        2                      1           1.574312e+09
76409        3                      1           1.574312e+09
50287        4                      1           1.574312e+09
...        ...                    ...                    ...
41993        3                      1           1.574312e+09
97639       10                      1           1.574312e+09
95939      164                      1           1.574312e+09
117952       3                      1           1.574312e+09
43567      208                      1           1.574312e+09

        frame_info.number  frame_info.len  frame_info.cap_len      eth.type  \
13848               26890              78                  78  2.869859e-42
135151             193348              54                  54  2.869859e-42
78582               81261            1436                  66  2.869859e-42
76409               77018            1440                  66  2.869859e-42
50287               25741              58                  58  2.869859e-42
...                   ...             ...                 ...           ...
41993               10025            1440                  66  2.869859e-42
97639              118695            1514                  66  2.869859e-42
95939              115900              54                  54  2.869859e-42
117952             158404            1440                  66  2.869859e-42
43567               12878              54                  54  2.869859e-42

               ip.id      ip.flags  ip.flags.df  ...  tcp.flags.syn  \
13848   1.053776e-42  2.295887e-41          1.0  ...            0.0
135151  9.102695e-41  0.000000e+00          0.0  ...            1.0
78582   2.695678e-41  2.295887e-41          1.0  ...            0.0
76409   6.846184e-41  2.295887e-41          1.0  ...            0.0
```

Step 6: A bar graph is plotted after the feature reduction step. This bar graph shows the count of the attack labels in the data set. For example, data packets from nmap top10 is the least occurring in the dataset which occurs 26 times followed by nmap fast which occurs 206 . Normal_traffic has the highest count of 103094 in the dataset.

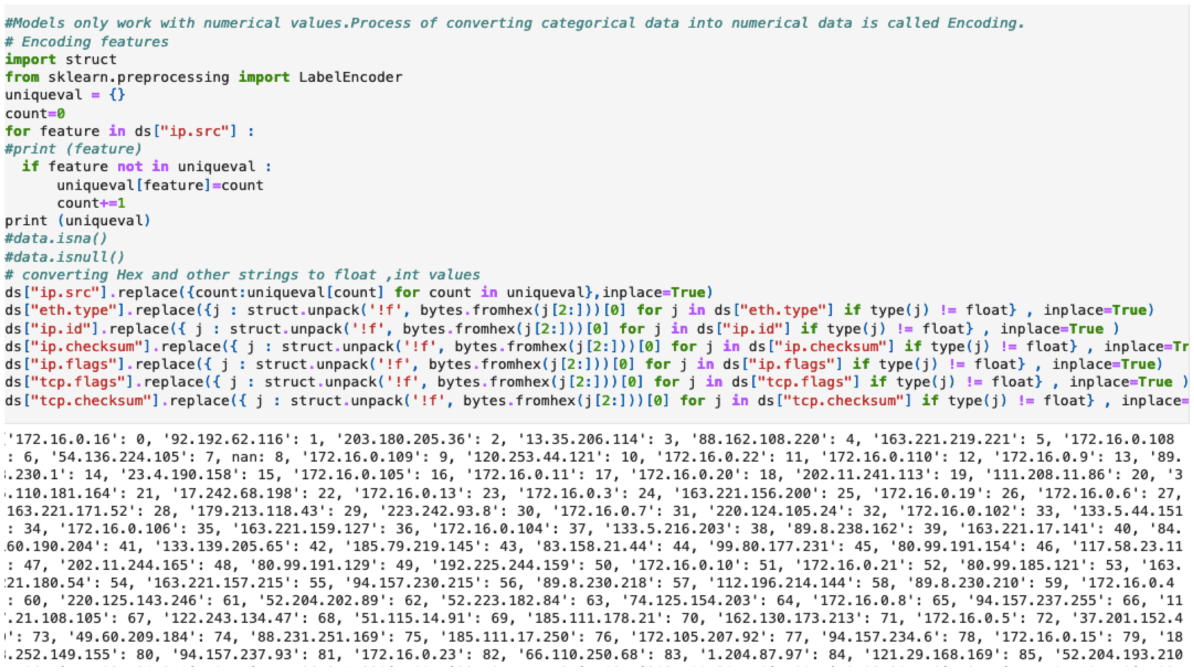


```
#Data distribustion graph
ds.label.value_counts().plot.barh()
plt.title('Data distribution graph attack labels vs count')
plt.show()
plt.figure(figsize=(100,200))
```



```
<Figure size 7200x14400 with 0 Axes>

<Figure size 7200x14400 with 0 Axes>
```

Step 7: The next step after feature reduction is to encode the data. The different datatypes are converted into numerical values. The null and nan values are replaced with zeroes as a part of data pre-processing.

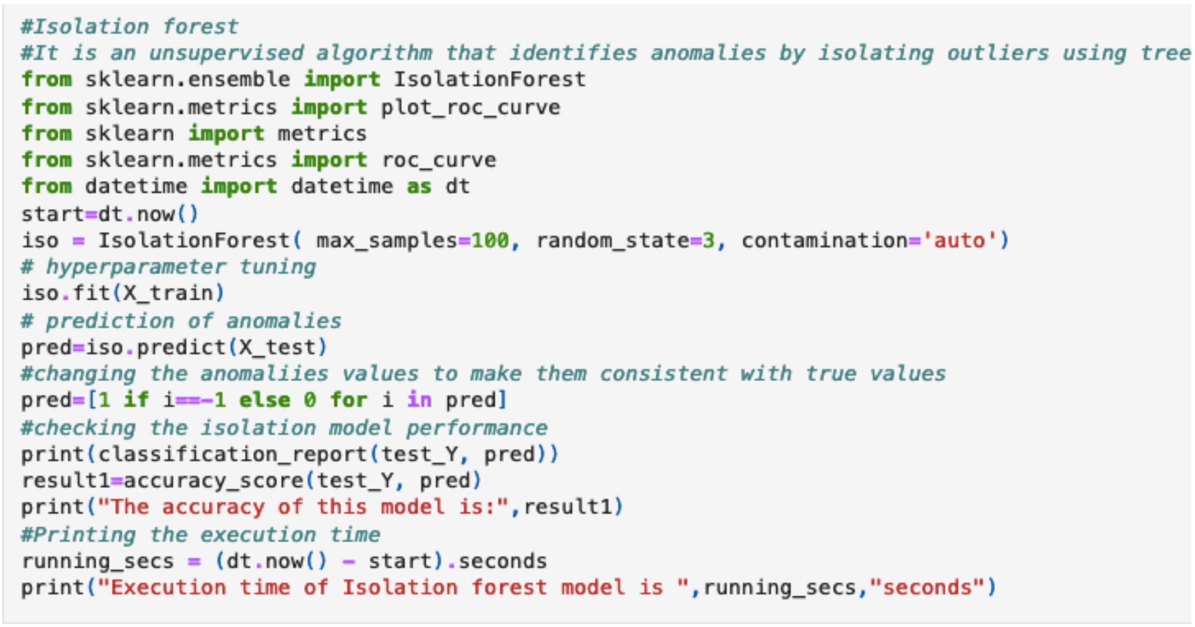


```
#Models only work with numerical values.Process of converting categorical data into numerical data is called Encoding.
# Encoding features
import struct
from sklearn.preprocessing import LabelEncoder
uniqueval = {}
count=0
for feature in ds["ip.src"] :
#print (feature)
  if feature not in uniqueval :
      uniqueval[feature]=count
      count+=1
print (uniqueval)
#data.isna()
#data.isnull()
# converting Hex and other strings to float ,int values
ds["ip.src"].replace({count:uniqueval[count] for count in uniqueval},inplace=True)
ds["eth.type"].replace({j : struct.unpack('!f', bytes.fromhex(j[2:]))[0] for j in ds["eth.type"] if type(j) != float} , inplace=True)
ds["ip.id"].replace({ j : struct.unpack('!f', bytes.fromhex(j[2:]))[0] for j in ds["ip.id"] if type(j) != float} , inplace=True )
ds["ip.checksum"].replace({ j : struct.unpack('!f', bytes.fromhex(j[2:]))[0] for j in ds["ip.checksum"] if type(j) != float} , inplace=Tr
ds["ip.flags"].replace({ j : struct.unpack('!f', bytes.fromhex(j[2:]))[0] for j in ds["ip.flags"] if type(j) != float} , inplace=True)
ds["tcp.flags"].replace({ j : struct.unpack('!f', bytes.fromhex(j[2:]))[0] for j in ds["tcp.flags"] if type(j) != float} , inplace=True )
ds["tcp.checksum"].replace({ j : struct.unpack('!f', bytes.fromhex(j[2:]))[0] for j in ds["tcp.checksum"] if type(j) != float} , inplace=
```

```
'172.16.0.16': 0, '92.192.62.116': 1, '203.180.205.36': 2, '13.35.206.114': 3, '88.162.108.220': 4, '163.221.219.221': 5, '172.16.0.108
: 6, '54.136.224.105': 7, nan: 8, '172.16.0.109': 9, '120.253.44.121': 10, '172.16.0.22': 11, '172.16.0.110': 12, '172.16.0.9': 13, '89.
.230.1': 14, '23.4.190.158': 15, '172.16.0.105': 16, '172.16.0.11': 17, '172.16.0.20': 18, '202.11.241.113': 19, '111.208.11.86': 20, '3
.110.181.164': 21, '17.242.68.198': 22, '172.16.0.13': 23, '172.16.0.3': 24, '163.221.156.200': 25, '172.16.0.19': 26, '172.16.0.6': 27,
163.221.171.52': 28, '179.213.118.43': 29, '223.242.93.8': 30, '172.16.0.7': 31, '220.124.105.24': 32, '172.16.0.102': 33, '133.5.44.151
: 34, '172.16.0.106': 35, '163.221.159.127': 36, '172.16.0.104': 37, '133.5.216.203': 38, '89.8.238.162': 39, '163.221.17.141': 40, '84.
.60.190.204': 41, '133.139.205.65': 42, '185.79.219.145': 43, '83.158.21.44': 44, '99.80.177.231': 45, '80.99.191.154': 46, '117.58.23.11
: 47, '202.11.244.165': 48, '80.99.191.129': 49, '192.225.244.159': 50, '172.16.0.10': 51, '172.16.0.21': 52, '80.99.185.121': 53, '163.
21.180.54': 54, '163.221.157.215': 55, '94.157.230.215': 56, '89.8.230.218': 57, '112.196.214.144': 58, '89.8.230.210': 59, '172.16.0.4
: 60, '220.125.143.246': 61, '52.204.202.89': 62, '52.223.182.84': 63, '74.125.154.203': 64, '172.16.0.8': 65, '94.157.237.255': 66, '11
.21.108.105': 67, '122.243.134.47': 68, '51.115.14.91': 69, '185.111.178.21': 70, '162.130.173.213': 71, '172.16.0.5': 72, '37.201.152.4
': 73, '49.60.209.184': 74, '88.231.251.169': 75, '185.111.17.250': 76, '172.105.207.92': 77, '94.157.234.6': 78, '172.16.0.15': 79, '18
.252.149.155': 80, '94.157.237.93': 81, '172.16.0.23': 82, '66.110.250.68': 83, '1.204.87.97': 84, '121.29.168.169': 85, '52.204.193.210
```

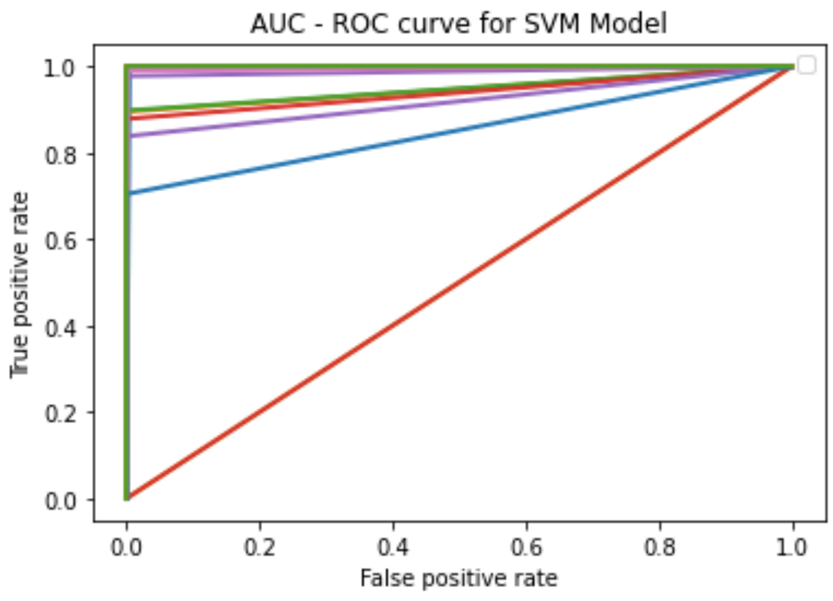
Step 8: The pre-processed data is then split into training and testing . A standard scalar function is used to modify the training data into a value distribution with a mean of 0 and a standard deviation of 1.

Step 9: In this step, the Isolation Forest Algorithm is implemented to detect the anomalies in the dataset. In hyper parameter tuning, we can check the approximate accuracy of the model by changing the random state value to 5, 3, 0 etc. Upon doing that , the accuracy for this model was around 60%. The execution time for this model was 7 seconds. The AUC ROC curve is attached below:

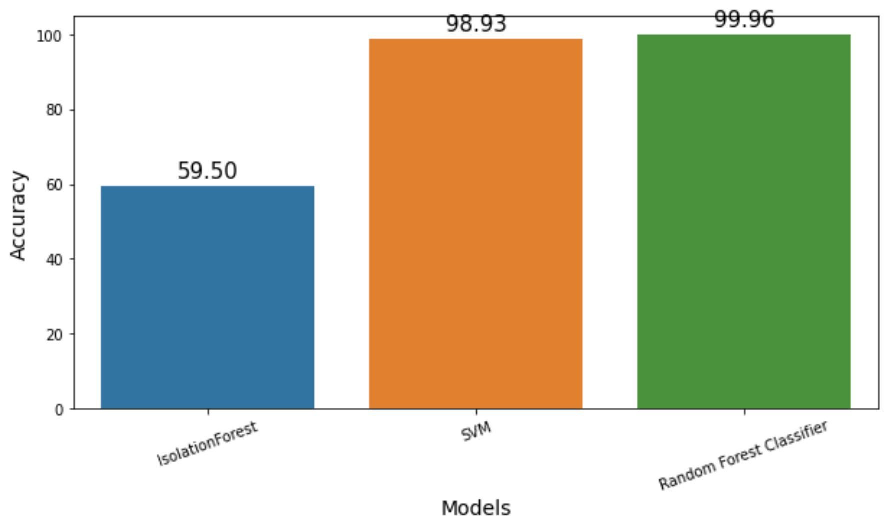


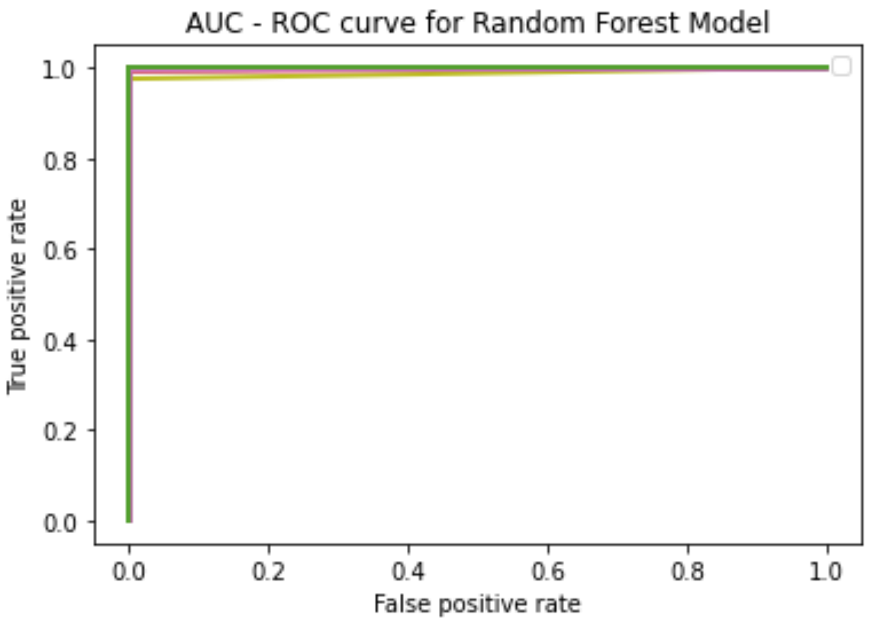
```
#Isolation forest
#It is an unsupervised algorithm that identifies anomalies by isolating outliers using tree
from sklearn.ensemble import IsolationForest
from sklearn.metrics import plot_roc_curve
from sklearn import metrics
from sklearn.metrics import roc_curve
from datetime import datetime as dt
start=dt.now()
iso = IsolationForest( max_samples=100, random_state=3, contamination='auto')
# hyperparameter tuning
iso.fit(X_train)
# prediction of anomalies
pred=iso.predict(X_test)
#changing the anomaliies values to make them consistent with true values
pred=[1 if i==-1 else 0 for i in pred]
#checking the isolation model performance
print(classification_report(test_Y, pred))
result1=accuracy_score(test_Y, pred)
print("The accuracy of this model is:",result1)
#Printing the execution time
running_secs = (dt.now() - start).seconds
print("Execution time of Isolation forest model is ",running_secs,"seconds")
```

Step 10: In this step, the SVM Algorithm is implemented to detect the anomalies in the dataset. In hyper parameter tuning, we can check the approximate accuracy of the model by changing the random state and n_estimators value. Upon doing that , the accuracy for this model was around 98%. The execution time for this model was 121 seconds.This model took the longest time to execute out of the three models. The AUC ROC curve for SVM model is attached below:



Step 11: In this step, the Random Forest Algorithm is implemented to detect the anomalies in the dataset. In hyper parameter tuning, we can check the approximate accuracy of the model by changing the random state .This model has the highest accuracy of 99.9%. The execution time for this model was 4 seconds. This model took the least amount of time to execute out of the three models and has the highest accuracy and highest AUC. This makes random forest the best ML model to detect port scanning attacks. The accuracy comparision graph and the AUC ROC curve for Random Forest model is attached below:





Conclusion

Machine learning has emerged as a critical technological advancement in the field of cybersecurity. It uses a variety of approaches to prevent digital threats and to strengthen the security foundation. Through this group project, we have learned to apply Machine Learning to solve cybersecurity threats such as port scanning attacks and phishing attacks for which we have built a port scan detection system and a malicious URL detection system respectively using various anomaly detection models. We have compared the performance of the various models using parameters like accuracy score, execution time, AUC-ROC curve, confusion matrix, and classification report.

On evaluating the accuracy score for prob scan detection models, we find that Random Forest Classifier has the highest accuracy score of 99.96% , for the given dataset followed by SVM model which has 98.93% accuracy. We have observed that unsupervised learning models such as Isolation Forest gave a low accuracy score which implies that it is not very efficient in detecting anomalies. In the case of Malicious URL detection, Random Forest had the highest accuracy score of 89% followed by Decision Tree Classifier and Extra Tree Classifier. The execution time was observed to be lowest for Random Tree Classifier in both the datasets. The Area under Curve (AUC) of ROC curves showed that the auc score tallied with the accuracy score of the respective machine learning algorithms.

To conclude, it can be inferred that using machine learning models is an efficient way to obtain the goal of detecting anomalies and preventing cyber-attacks. This project can be used in real life scenarios to detect malicious URL’s and port scanning attacks. Defending such attacks with the help of Machine Learning will thereby help in building cyber resilient organisations.

References

Aamir, M., Rizvi, S., Hashmani, M., Zubair, M. and Ahmad, J., 2021. Machine Learning Classification of Port Scanning and DDoS Attacks: A Comparative Analysis. Mehran University Research Journal of Engineering and Technology,.

Amit, I, Matherly, J, Hewlett, W, Xu, Z, Meshi, Y, & Weinberger, 2018,
Machine Learning in Cyber- Security - Problems, Challenges and Data Sets.

Anish Gupta, Ruchika Gupta and Gagan Kukreja, July 13 2021, Springer link, Cyber security using Machine Learning-Techniques and Business Applications.

Avdoshin, S., Lazarenko, A., Chichileva, N., Naumov, P. and Klyucharev, P., 2019. Machine Learning Use Cases In Cybersecurity. Trudy ISP RAN.

Dasgupta, D. and Akhtar, Z., 2022. Machine learning in cybersecurity: a comprehensive survey. Applications JDMS Journal of Defense Modeling and Simulation: Applications, Methodology, Technology,.

Ferhat Ozgur Catak, Kevser Sahinbas, Kevser Sahinbas, November 2020, Malicious URL Detection Using Machine Learning,

Gupta, A., Gupta, R. and Kukreja, G., 2021. Cyber Security Using Machine Learning: Techniques and Business Applications. Studies in Computational Intelligence,.

Orion Cassetto, March 19, 2020, Exabean Machine Learning for Cybersecurity: Next-Gen Cyber Defense (exabeam.com),

Kim, K. and Kim, H., 2018. Mobile and Wireless Technology. Link Springer,.

Shaik, Imtiyazuddin; Emmadi, Nitesh; Tupsamudre, Harshal; Narumanchi, Harika; Bhattachar, Rajan Mindigal Alasingara. Published on September 20, 2021, Privacy Preserving Machine Learning for Malicious URL Detection

Rosenberg, I., Shabtai, A., Elovici, Y. and Rokach, L., 2022. Adversarial Machine Learning Attacks and Defense Methods in the Cyber Security Domain. ACM Computing Surveys,.

Sungjin Kim, Jinkook Kim, Brent ByungHoon Kang, Volume 77, August 2018, Malicious URL protection based on attack habitual behavioral analysis.

Priya Pedamkar, 2020, EDUCBA, Decision Tree in Machine Learning | Split creation and Building a Tree (educba.com)

Java T point, K-Nearest Neighbor(KNN) Algorithm for Machine Learning - Javatpoint

Martin, K., Rahouti, M., Ayyash, M. and Alsmadi, I., 2021. Anomaly detection in blockchain using network representation and machine learning. Wiley,.

Viet, H., Van, Q., Trang, L. and Nathan, S., 2018. Using Deep Learning Model for Network Scanning Detection. ICPS Proceedings,.