CSE3501:-ISAA

Review – 3

Group members: Shivam Chokhani (19BIT0362)

Priyanshi Singh(19BIT0111)

Sejal Bharti (19BIT0286)

Kunj Patel(19BIT0256)

Slot: F1+TF1,F2+TF2

Faculty: Professor Sumaiya Thaseen

Topic: Network Intrusion Detection System

Abstract

The increase of connected devices and the constantly evolving methods and techniques by attackers pose a challenge for network intrusion detection systems from conception to operation. As a result, we see a constant adoption of machine learning algorithms for network intrusion detection systems. However, the dataset used by these studies has become obsolete regarding both background and attack traffic. This work describes the AB-TRAP framework that enables the use of updated network traffic and considers operational concerns to enable the complete deployment of the solution. AB-TRAP is a five-step framework consisting of (i) the generation of the attack dataset, (ii) the bonafide dataset, (iii) training of machine learning models, (iv) realization (implementation) of the models, and (v) the performance evaluation of the realized model after deployment. We exercised the AB-TRAP for local (LAN) and global (internet) environments to detect TCP port scanning attacks. The LAN study case presented an f1-score of 0.96, and an area under the ROC curve of 0.99 using a decision tree with minimal CPU and RAM usage on kernel-space. For the internet case with eight machine learning algorithms with an average f1-score of 0.95, an average area under the ROC curve of 0.98, and an average overhead of 1.4% CPU and 3.6% RAM on user-space in a single-board computer. This framework has the following paramount characteristics: it is reproducible, uses the most up-to-date network traffic, attacks, and addresses the concerns to the model's realization and deployment.

Introduction

In our J component, we have worked on an Intrusion Detection System for the c2dc-AB/TRAP data set. Our solution has proven to be robust and can be scaled across based on traffic.Base Paper -An End-to-End Framework for Machine Learning-Based Network Intrusion Detection System The authors in this paper present AB-TRAP (Attack, Bonafide, Train, RealizAtion, and Performance), a framework comprising steps to build Attack and Bonafide datasets, train machine learning models, realize (implement) the solution in a target machine, and evaluate the protection module's performance. One of the main concerns in implementing effective Network Intrusion Detection Systems (NIDS) is the ability to adapt to new attacks and the evolution of the network traffic.

BASE PAPER REVIEW

Problem Addressed - The increase of connected devices and the constantly evolving methods and techniques by attackers pose a challenge for network intrusion detection systems from conception to operation. As a result, we see a constant adoption of machine learning algorithms for network intrusion detection systems. However, the dataset used by these studies has become obsolete regarding both background and attack traffic. This work describes the AB-TRAP framework that enables the use of updated network traffic and considers operational concerns to enable the complete deployment of the solution

Techniques Used - AB-TRAP is a five-step framework consisting of (i) the generation of the attack dataset, (ii) the bonafide dataset, (iii) training of machine learning models, (iv) realization (implementation) of the models, and (v) the performance evaluation of the realized model after deployment

Comparison Study - We test AB-TRAP in two environments: LAN and the Internet. In both cases, we achieve low- resource utilization protection modules, and Decision Tree provides the best performance for the training and realization phases. In the first case study, our results show an f1-score of 0.96, and the overhead is negligible,this represents the kernel-space implementation with Linux Kernel Modules. In the Internet case study, Decision Tree stills represent a good choice; however, other modules are also candidates for implementation, as is the Logistic Regression case. We see a more significant overhead compared to the first case study, and one of the reasons for this is the shifting of the implementation from kernel-space to user space.

LITERATURE SURVEY

1. **Deep Learning Approach for Intelligent Intrusion Detection System**

   Problem Statement - Malicious attacks are continually changing and are occurring in very large volumes requiring a scalable solution. There are different malware datasets available publicly for further research by cyber security community. However, no existing study has shown the detailed analysis of the performance

   Technique Used – A deep neural network (DNN), a type of deep learning model, is explored to develop a flexible and effective IDS to detect and classify unforeseen and unpredictable cyberattacks.

   Comparison Study – The proposed architecture able to perform better than previously implemented classical machine learning classifiers in both HIDS and NIDS ,this was the only framework which has the capability to collect network-level and host-level activities in a distributed manner using DNNs to detect attack more accurately.

2. **Survey of intrusion detection systems: techniques, datasets, and challenges**

   Problem Statement - Several machine learning techniques that have been proposed to detect zero-day attacks are reviewed. However, such approaches may have the problem of generating and updating the information about new attacks and yield high false alarms or poor accuracy. Here there can be found the summarized results of recent research and explored the contemporary models on the performance improvement of AIDS as a solution to overcome on IDS issues.

   Technique Used – Here there can be found the summarized results of recent research and explored the contemporary models on the performance improvement of AIDS as a solution to overcome on IDS issues.

In addition, the most popular public datasets used for IDS research have been explored and their data collection techniques, evaluation results

Comparison Study – The performance of the proposed RF classifier is rather high in terms of classification accuracy, F Measure and AUC. Furthermore, our results showed that RF is faster, robust and more accurate than the other classifiers. Random forest's runtime is quite fast, and it is able to detect phishing websites in comparison to the other classifiers.

3.A Supervised Intrusion Detection System for Smart Home IoT Devices

Problem Statement - Current insufficient security measures employed to defend smart devices make IoT the 'weakest' link to breaking into a secure infrastructure, and therefore an attractive target to attackers.

Technique Used – A three layer Intrusion Detection System (IDS) that uses a supervised approach to detect a range of popular network based cyber-attacks on IoT networks. The system consists of three main functions: 1) classify the type and profile the normal behaviour of each IoT device connected to the network, 2) identifies malicious packets on the network when an attack is occurring, and 3) classifies the type of the attack that has been deployed

Comparison Study The performance of the system's three core functions result in an F-measure of: 1) 96.2%, 2) 90.0%, and 3) 98.0%. This demonstrates that the proposed architecture can automatically distinguish between IoT devices on the network, whether network activity is malicious or benign, and detect which attack was deployed on which device connected to the network successfully

4. A New Intrusion Detection System Based on Fast Learning Network and Particle Swarm Optimization

Problem Statement - Supervised intrusion detection system is a system that has the capability of learning from examples about the previous attacks to detect new attacks. Using artificial neural network (ANN)-based intrusion detection is promising for reducing the number of false negative or false positives, because ANN has the capability of learning from actual examples

Technique Used – a developed learning model for fast learning network (FLN) based on particle swarm optimization (PSO) has been proposed and named as PSO-FLN. The model has been applied to the problem of intrusion detection and validated based on the famous dataset KDD99. Our developed model has been compared against a wide range of meta-heuristic algorithms for training extreme learning machines and FLN classifier.

Comparison Study – PSO-FLN has outperformed other learning approaches in the testing accuracy of the learning.

-

5. Boosting algorithms for network intrusion detection: A comparative evaluation of Real AdaBoost, Gentle AdaBoost and Modest AdaBoost

Problem Statement: Given the complex behavior of malicious organizations, it is important to adopt ML methods for internal access with good performance and low runtime. To overcome this challenge here, we use boosting algorithms and demonstrate capabilities of algorithms on IDS.

Technique Used: For the given model they have taken N training data samples such as:x1, y1,…,(xN,yN), with xi∈Rk and yi∈{−1,+1}, these data values are inserted in a random vector (X,Y). X is an N-dimensional vector that is used to check the network traffic features, and Y as an output class The aim of a standard ML classification problem is to predict the class label for Y, depends on the features vector X. In response to $Y \in \{-1,+1\}$, we focus on the classification of binary problems. For prediction of the outcomes of AdaBoost algorithms, they have taken five datasets according to the volume. KDDCUP99 is of large volume dataset, UNSW-NB15, TRAbID and NSL-KDD are of medium type datasets and CICIDS2017 dataset is considered as smallest in terms of volume.

Comparison Study: Advantages of ensemble learning overusing a single classifier is to obtain better performance. Aim is to learn a combination of weak and simple classifiers, with an error rate rigorously less than random guessing.

6. Performance evaluation of intrusion detection based on machine learning using Apache Spark

Problem Statement: This paper focuses on comparing intrusion detection ML algorithms such as SVMs, Naive Bytes, Decision tree and random forest. This paper shows that tree-based algorithms have proven to be the best of the lot. This paper uses apache spark to judge the algorithms on the basis of accuracy, sensitivity, training time, and prediction time.

Technique Used: The accuracy, sensitivity and specificity are calculated using four elements. Namely True positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). The paper also uses the NSL-KDD dataset (Tavallaee et al. 2009) which is an improvisation upon the popular KDD99 dataset.

Comparison:

Tableau 1. COMPARISON OF DIFFERENT INTRUSION DETECTION METHODS using UNSW-NB15 dataset

Methods	Accuracy	Sensitivity	Specificity	Training Time	Prediction Time
SVM	92.28	92.13	91.15	38.91	0.20
Naïve Bayes	74.19	92.16	67.82	2.25	0.18
Decision Tree	95.82	92.52	97.10	4.80	0.13
Random Forest	97.49	93.53	97.75	5.69	0.08

7.Ant Colony Induced Decision Tree for Intrusion Detection

Problem Statement: The field of intrusion detection, security and privacy has been stagnant, and ML has a great potential in this field. The given paper focuses on the classifier based on decision trees using an ant colony optimization instead of traditional CART techniques.

Technique Used: Without proper classification, detection is quite difficult. The divide and conquer approach are used to induce a decision tree, however this paper uses an ATM classifier which results in a greater accuracy and precision. The data set used is NLS-KDD (Tavallaee et al. 2009) which is an improvisation upon the popular KDD99 dataset. There are massive improvements in all measures.

Comparison:

Table 6: ATM results on NSL-KDD Test21 dataset

Evaluation	FULL	FULL FS	DOS	DOS FS	Probe	Probe FS	R2L	R2L FS	U2R	U2R FS	ATMa Per Attack
Error Rate (%)	39.72	39.06	24.99	24.89	21.99	22.63	52.10	51.56	7.99	8.31	35.15
Accuracy (%)	60.28	60.94	75.01	75.11	78.01	77.37	47.90	48.44	92.01	91.69	64.85
DR (%)	84.94	84.11	93.40	91.71	88.79	90.24	98.22	98.17	99.86	99.88	57.05
FAR (%)	45	44	34	33	32	34	91	90	92	96	0
Tree Quality (%)	98.91	99.00	99.72	99.69	99.39	99.42	99.55	99.56	99.91	99.91	99.64
Leaf Nodes	207.10	158.20	83.10	96.20	63.50	107.40	35.20	28.20	3.00	3.50	46.2
Runtime (s)	241.46	239.09	157.89	134.27	61.45	59.75	20.10	20.35	2.04	1.74	241.48
F-measure	0.44	0.44	0.71	0.71	0.79	0.79	0.62	0.63	0.96	0.96	0.73
Cost	852.67	875.73	1819.41	1419.46	303.34	347.52	114.51	110.65	17.69	17.29	1861.01

4. A Deep Learning Approach to Network Intrusion Detection

Problem Statement: A deep learning technique for intrusion detection, addressing concerns regarding the feasibility and sustainability of existing approaches when faced with the demands of modern networks is proposed by this paper.

Ant Colony Induced Decision Tree for Intrusion Detection
Problem Statement: The field of intrusion detection, security and privacy has been stagnant, and ML has a great potential in this field. The given paper focuses

on the classifier based on decision trees using an ant colony optimization instead of traditional CART techniques.

Technique Used: Without proper classification, detection is quite difficult. The divide and conquer approach are used to induce a decision tree, however this paper uses an ATM classifier which results in a greater accuracy and precision. The data set used is NLS-KDD (Tavallaee et al. 2009) which is an improvisation upon the popular KDD99 dataset. There are massive improvements in all measures.

Comparison:

A Deep Learning Approach to Network Intrusion Detection
Problem Statement: A deep learning technique for intrusion detection, addressing concerns regarding the feasibility and sustainability of existing approaches when faced with the demands of modern networks is proposed by this paper.
Technique Used: A nonsymmetric deep autoencoder (NDAE) for unsupervised feature learning and a deep learning classification model constructed using stacked NDAEs are proposed in this paper. The proposed classifier is implemented in graphics processing unit (GPU)-enabled TensorFlow and evaluated using the benchmark KDD Cup ’99 and NSL-KDD datasets.
Comparison Study: The stacked NDAE model is compared against DBN technique. According to the comparisons, this model offers up to a 5% improvement in accuracy and training time reduction of up to 98.81%. The proposed model reveals a consistent level of classification accuracy.
The proposed model cannot handle zero-day attacks. Also the existing evaluations don’t utilise real-world backbone network traffic.

Evaluation of machine learning techniques for network intrusion detection
Problem Statement: The early research work in the area of Network traffic anomaly and commercially available Intrusion Detection Systems (IDS) are mostly signature- based. The problem of signature-based method is that the database signature needs to be updated as new attack signatures become available and hence it is not suitable for real-time network anomaly detection. This work presented six commonly used machine learning techniques as well as an Ensemble method based on the six algorithms for network traffic anomaly

detection.
Technique Used: The seven techniques adopted are: KMeans, K-Nearest Neighbors (KNN), Fuzzy C-Means (FCM), Support Vector Machine (SVM), Naïve-Bayes (NB), Radial Basis Function (RBF) and Ensemble method comprising of the above mentioned six algorithms.
Comparison Study: The RBF classification technique worked the best with ROC value of 0.9741, whereas the Ensemble method wasn't far behind with ROC of 0.9631. This work also proposed to use information entropy as the traffic features followed by machine learning classification techniques for network anomaly detection.
Network Intrusion Detection Combined Hybrid Sampling With Deep Hierarchical Network:
Problem Statement: Due to the time-varying and complex network environment, the network intrusion samples are merged into huge amounts of normal samples, which leads to insufficient samples for model training and detection results with a high false detection rate. This paper proposes an intrusion detection system (IDS) based on the combination of hybrid sampling and deep hierarchical network, according to the problem of data imbalance.
Technique Used: Firstly, one-side selection (OSS) and Synthetic Minority Over-sampling Technique (SMOTE) are combined to construct a balanced dataset for model training. It can reduce the training time of the model and partially solves the problems of inadequate training from unbalanced samples. In addition, a network data preprocessing method is established for complex, multidimensional cyber threats, which is suitable for the proposed deep hierarchical network model. Then, input data is classified through the hierarchical network model constructed neural network (CNN) and Bi-directional long short-term memory (BiLSTM). The model extracts feature automatically through repeated multi-level learning. Two intrusion datasets (NSL-KDD and UNSW-NB15) have been employed to evaluate the performance of the proposed approach. The statistical significance tests prove that the proposed approach outperforms other classifiers.

A Novel Two-Stage Deep Learning Model for Efficient Network Intrusion Detection:
Problem Statement: Many NIDS techniques have recently been proposed, however, these techniques face significant challenges due to the continuous emergence of new threats that are not recognized by existing detection systems.

This paper proposes a novel two-stage deep learning (TSDL) model based on a deep stacked auto-encoder (DSAE) neural network, to deal with the problem of network intrusion detection.

The TSDL model comprises two stages; each stage contains two hidden layers with a soft-max classifier. The deep learning model is trained in a semi supervised manner. The first stage of the model, termed the initial decision stage, is used to classify the normal and abnormal states of network traffic. The deep learning model to operate can only be selected at this stage by the user. The second decision stage is employed to detect the normal state and other types of attacks. In the latter, this model works in a cascade manner enabling the final decision stage to efficiently classify various types of attacks. The model achieved good results, up to 99.996% for the KDD99 dataset and 89.134% for the UNSW-NB15 dataset, with a low FAR, in terms of multi-class detection accuracy. Additionally, in terms of efficiency, the execution time consumed by the proposed model is very low, which makes it appropriate for future deployment in real-time intrusion detection tasks. Comparison with state-of-the-art approaches demonstrated the robustness of the proposed model.

Performance Evaluation of Advanced machine learning algorithms for Network intrusion detection System

Problem Statement - Machine Learning (ML) techniques are used to help increase the accuracy of intrusion detection and significantly reduce the false negative and positive rate.

Technique Used – In the the following paper for the intrusion detection system they have used five Machine learning algorithms such as , Random Forest, Decision tree Gradient Boost method, AdaBoost model, Gaussian Naïve Bayes method. NIDS Dataset is used for the proposed system. The network system can be termed as related data. NIDS treats a group of data features as inputs. All data entries contain attributes in various forms such as float, total value, binary, and other suggested type. Labels with two values for each input can be 0 or 1, where 0 stands for normal, and 1 for not normal.

Comparison Study – The results show that Random Forest has the best accuracy followed by Decision tree and AdaBoost, Gradient Boost comes in fourth place with accuracy and we found that the Gaussian Naïve Bayes have the lowest accuracy among the five ML methods used. The accuracy of the acquisition of the learning separator for each machine is high on the database used. Gradient Boosting and AdaBoost have the best accuracy of 97.40% and 97.92%, respectively, but have the worst execution times of 12.36 (s) and 21.22 (s), respectively.**Technique Used:** A nonsymmetric deep autoencoder (NDAE) for unsupervised feature learning and a deep learning classification model constructed using stacked NDAEs are proposed in this paper. The proposed classifier is implemented in graphics processing unit (GPU)-enabled TensorFlow and evaluated using the benchmark KDD Cup ’99 and NSL-KDD datasets.

Comparison Study: The stacked NDAE model is compared against DBN technique. According to the comparisons, this model offers up to a 5% improvement in accuracy and training time reduction of up to 98.81%. The proposed model reveals a consistent level of classification accuracy.

The proposed model cannot handle zero-day attacks. Also the existing evaluations don’t utilise real-world backbone network traffic.

5. Evaluation of machine learning techniques for network intrusion detection

Problem Statement: The early research work in the area of Network traffic anomaly and commercially available Intrusion Detection Systems (IDS) are mostly signature-based. The problem of signature-based method is that the database signature needs to be updated as new attack signatures become available and hence it is not suitable for real-time network anomaly detection. This work presented six commonly used machine learning techniques as well as an Ensemble method based on the six algorithms for network traffic anomaly detection.

Technique Used: The seven techniques adopted are: KMeans, K-Nearest Neighbors (KNN), Fuzzy C-Means (FCM), Support Vector Machine (SVM), Naïve-Bayes (NB), Radial Basis Function (RBF) and Ensemble method comprising of the above mentioned six algorithms.

Comparison Study: The RBF classification technique worked the best with ROC value of 0.9741, whereas the Ensemble method wasn’t far behind with ROC of 0.9631. This work also proposed to use information entropy as the traffic features followed by machine learning classification techniques for network anomaly detection.

6. Network Intrusion Detection Combined Hybrid Sampling With Deep Hierarchical Network:

Problem Statement: Due to the time-varying and complex network environment, the network intrusion samples are merged into huge amounts of normal samples, which leads to insufficient samples for model training and detection results with a high false detection rate. This paper proposes an intrusion detection system (IDS) based on the combination of hybrid sampling and deep hierarchical network, according to the problem of data imbalance.

Technique Used: Firstly, one-side selection (OSS) and Synthetic Minority Over-sampling Technique (SMOTE) are combined to construct a balanced dataset for model training. It can reduce the training time of the model and partially solves the problems of inadequate training from unbalanced samples. In addition, a network data preprocessing method is established for complex, multidimensional cyber threats, which is suitable for the proposed deep hierarchical network model. Then, input data is classified through the hierarchical network model constructed neural network (CNN) and Bi-directional long short-term memory (BiLSTM). The model extracts feature automatically through repeated multi-level learning. Two intrusion datasets (NSL-KDD and UNSW-NB15) have been employed to evaluate the performance of the proposed approach. The statistical significance tests prove that the proposed approach outperforms other classifiers.

7. A Novel Two-Stage Deep Learning Model for Efficient Network Intrusion Detection:

Problem Statement: Many NIDS techniques have recently been proposed, however, these techniques face significant challenges due to the continuous emergence of new threats that are not recognized by existing detection systems. This paper proposes a novel two-stage deep learning (TSDL) model based on a deep stacked auto-encoder (DSAE) neural network, to deal with the problem of network intrusion detection.

Technique Used: The TSDL model comprises two stages; each stage contains two hidden layers with a soft-max classifier. The deep learning model is trained in a semi supervised manner. The first stage of the model, termed the initial decision stage, is used to classify the normal and abnormal states of network traffic. The deep learning model to operate can only be selected at this stage by the user. The second decision stage is employed to detect the normal state and other types of attacks. In the latter, this model works in a cascade manner enabling the final decision stage to efficiently classify various types of attacks. The model achieved good results, up to 99.996% for the KDD99 dataset and 89.134% for the UNSW-NB15 dataset, with a low FAR, in terms of multi-class detection accuracy. Additionally, in terms of efficiency, the execution time consumed by the proposed model is very low, which makes it appropriate for future deployment in real-time intrusion detection tasks. Comparison with state-of-the-art approaches demonstrated the robustness of the proposed model.

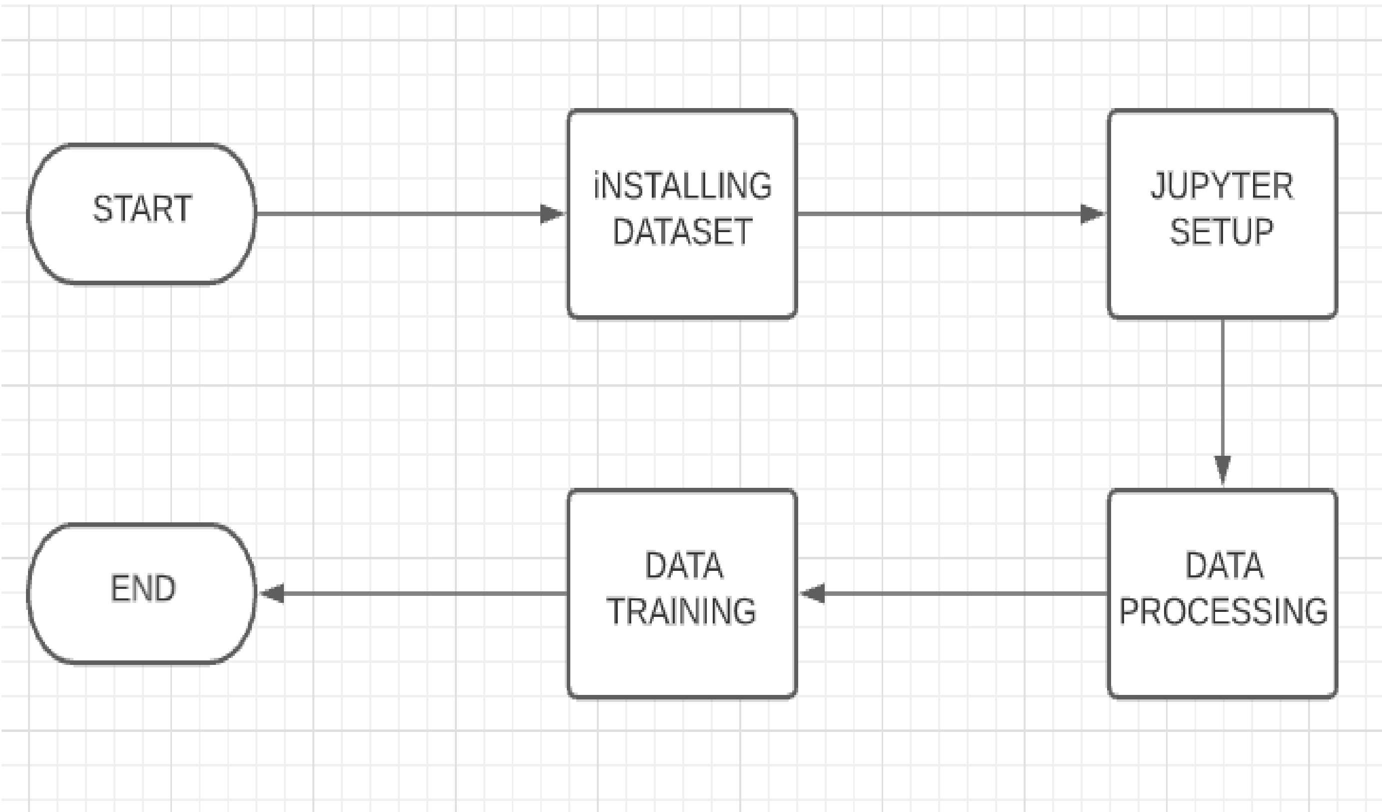
13 Performance Evaluation of Advanced machine learning algorithms for Network intrusion detection System

Problem Statement - Machine Learning (ML) techniques are used to help increase the accuracy of intrusion detection and significantly reduce the false negative and positive rate.

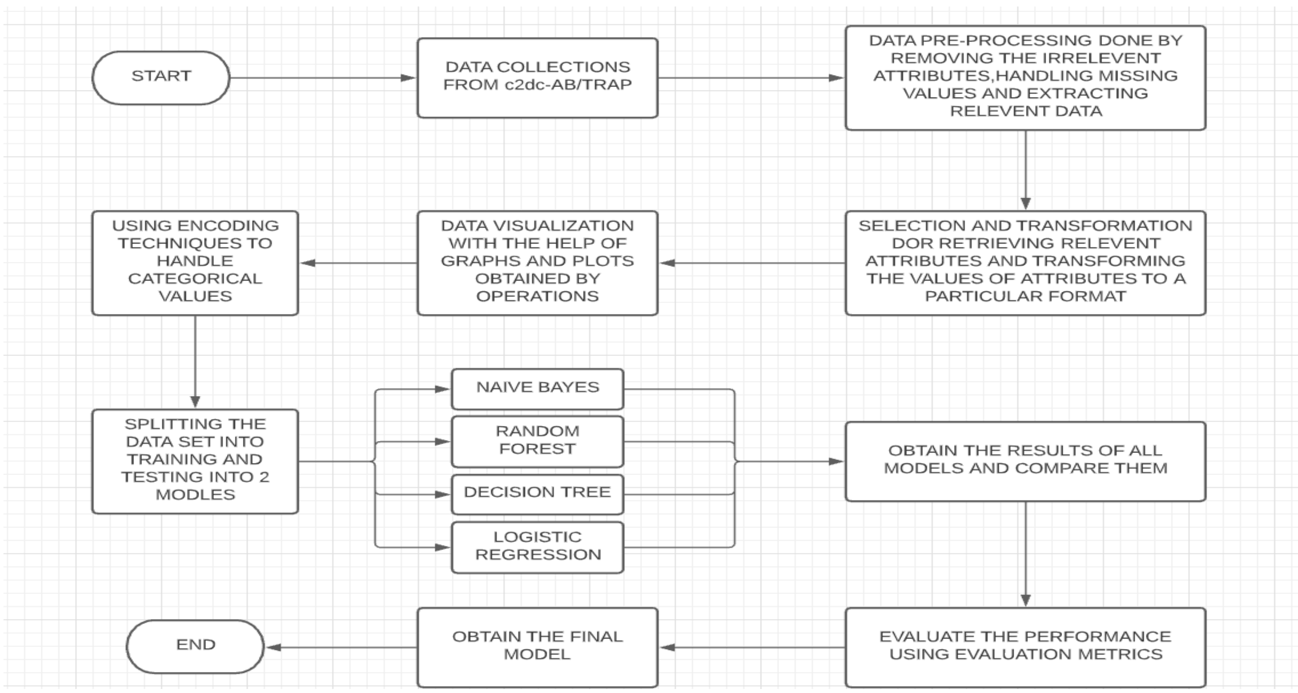
Technique Used – In the following paper for the intrusion detection system they have used five Machine learning algorithms such as , Random Forest, Decision tree Gradient Boost method, AdaBoost model, Gaussian Naïve Bayes method. NIDS Dataset is used for the proposed system. The network system can be termed as related data. NIDS treats a group of data features as inputs. All data entries contain attributes in various forms such as float, total value, binary, and other suggested type. Labels with two values for each input can be 0 or 1, where 0 stands for normal, and 1 for not normal.

Comparison Study – The results show that Random Forest has the best accuracy followed by Decision tree and AdaBoost, Gradient Boost comes in fourth place with accuracy and we found that the Gaussian Naïve Bayes have the lowest accuracy among the five ML methods used. The accuracy of the acquisition of the learning separator for each machine is high on the database used. Gradient Boosting and AdaBoost have the best accuracy of 97.40% and 97.92%, respectively, but have the worst execution times of 12.36 (s) and 21.22 (s), respectively.

High-Level Design

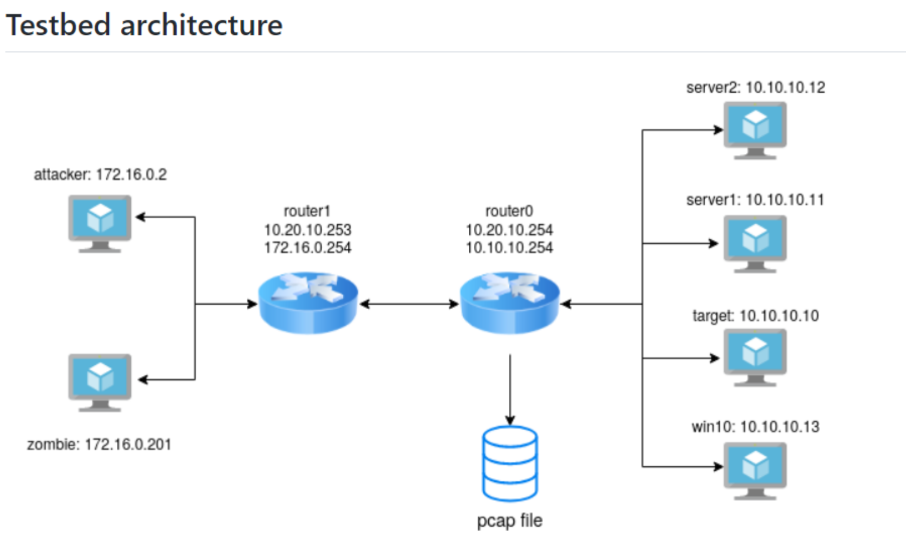


Low-Level Design



PREPROCESSING:

LAN



1. Convert features that were extracted from pcap as hexadecimal Filled the values with NaN with Zero.

```
In [30]: fields = ['eth.type', 'ip.id', 'ip.flags', 'ip.checksum', 'ip.dsfield', 'tcp.flags', 'tcp.checksum']

         for field in fields:
             df_labeled[field] = df_labeled[field].apply(lambda x: int(str(x), 16))
```

```
In [31]: bonafide = bonafide.fillna(0)
         for field in fields:
             bonafide[field] = bonafide[field].apply(lambda x: int(str(x), 16))
```

2. The packet structure for IPv4 and IPv6 are different. We have made the model for TCP value 6. Hence we are removing a protocol field different from TCP(value 6).

Check if there are packets with the protocol field different than TCP (value 6)

```
wrong_proto = full_data[full_data['ip.proto'] != 6]['label'].value_counts().values
full_data = full_data[full_data['ip.proto'] == 6]
print("It was found and removed", wrong_proto,"packets.")
```

```
It was found and removed [11708] packets.
```

3. **Features not applicable to this study according to Base Paper:**

- Remove features from link layer - layer 2:

- ➢ frame_info.time
- ➢ frame_info.encap_type
- ➢ frame_info.time_epoch
- ➢ frame_info.number
- ➢ frame_info.len
- ➢ frame_info.cap_len
- ➢ eth.type

- Remove features that are redundant or invariable

  - ➢ ip.version - we consider only IPv4
  - ➢ ip.proto - this study is applicable only to TCP
  - ➢ ip.src - this attribute is removed to allow the generalization of learning (not learn past attackers)
  - ➢ ip.dst - this attribute is removed to allow the generalization of learning (not learn past targets)
  - ➢ ip.flags - this is removed because we use bit-set of flags
  - ➢ tcp.flags - this is removed because we use bit-set of flags

In [34]:
```
full_data.drop(columns=['frame_info.time', 'frame_info.encap_type', 'frame_info.time_epoch', 'frame_info.number',
                        'frame_info.len', 'frame_info.cap_len', 'eth.type', 'ip.flags', 'ip.src', 'ip.dst',
                        'ip.version', 'ip.proto', 'tcp.flags'], axis=1, inplace=True)
```

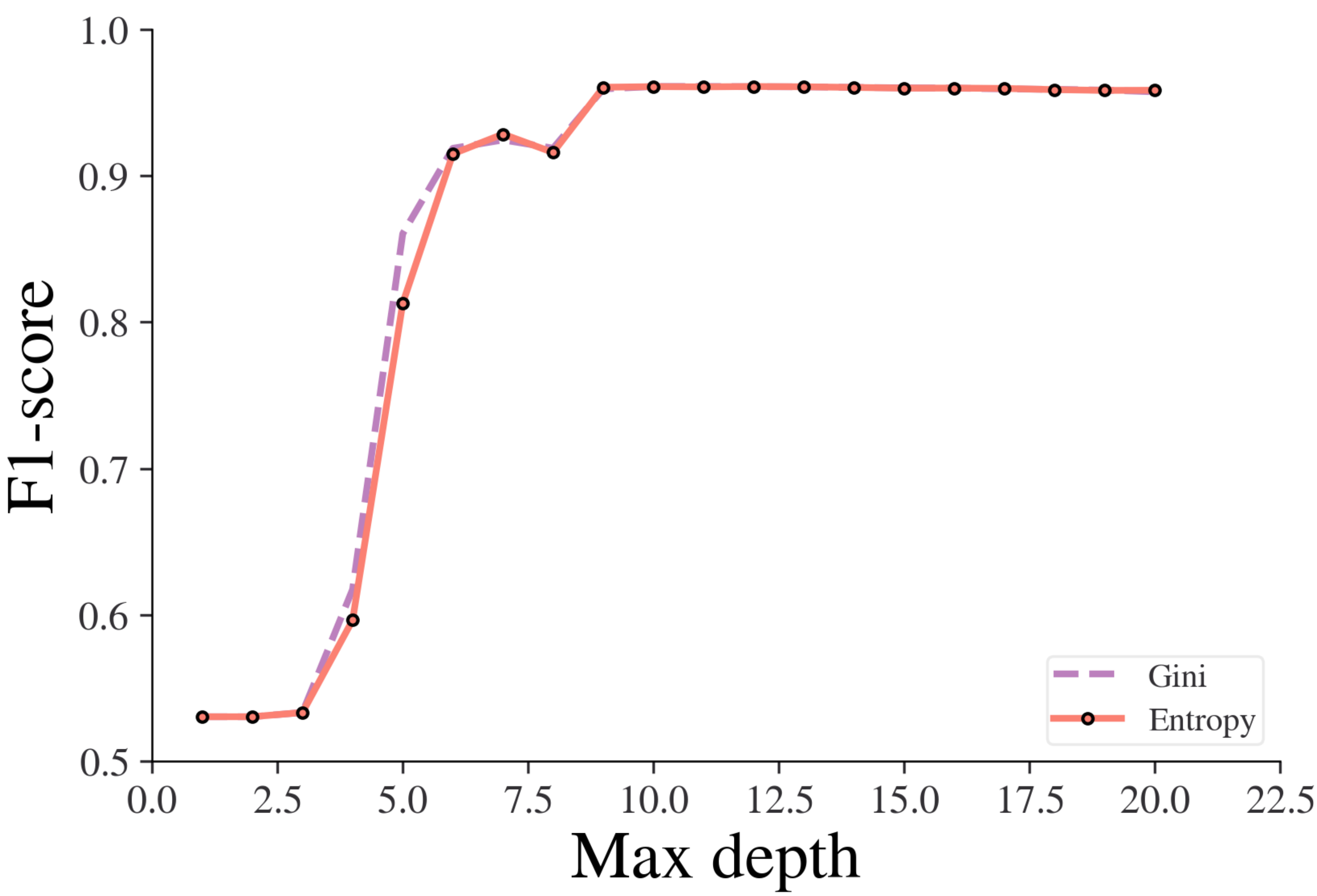
4. Features with zero variance (not useful for learning). Hence we are checking and dropping those columns where variance==0 is true.

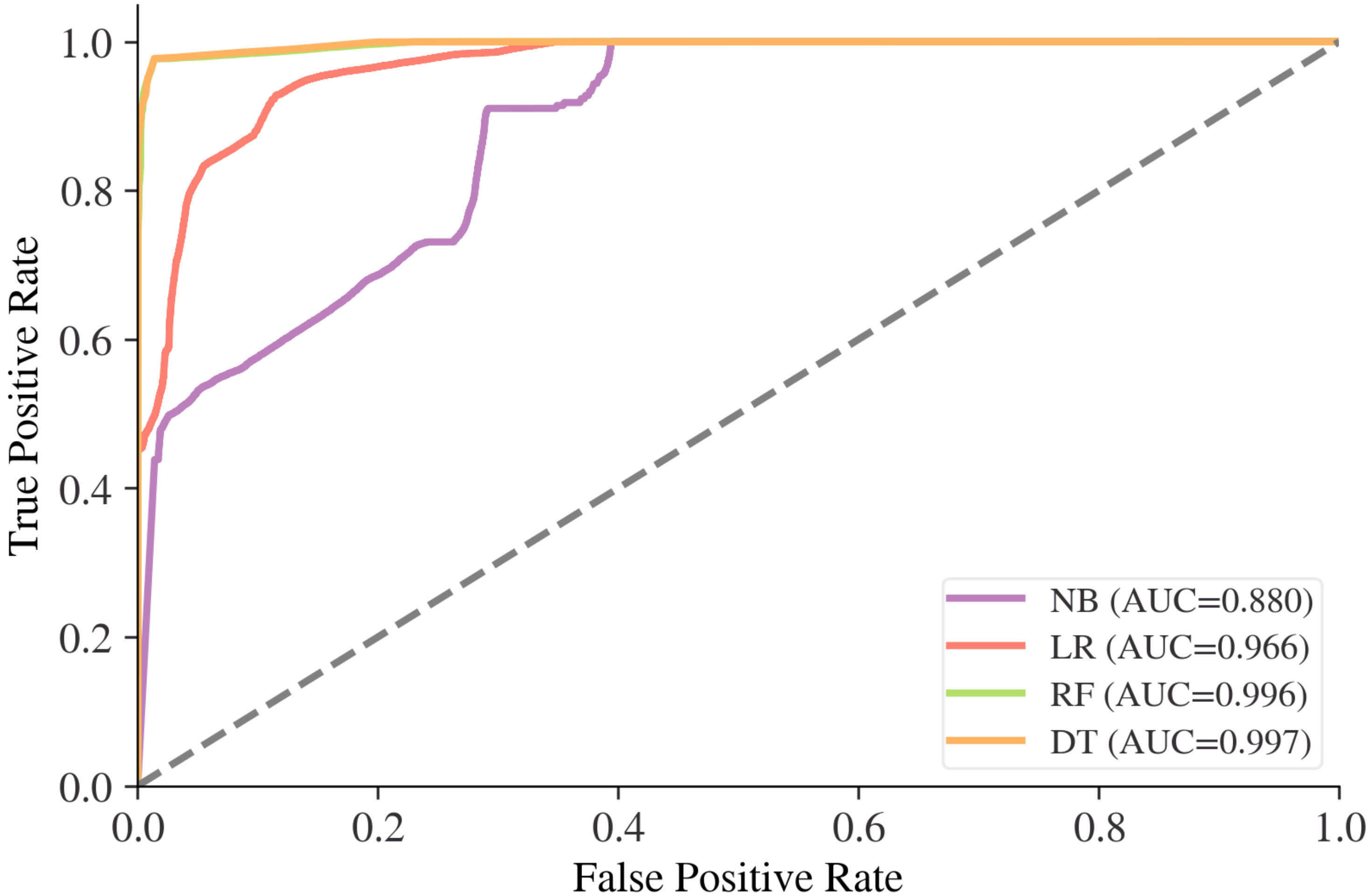
In [37]:
```
# check features with zero variance (not useful for learning)
(full_data.var() == 0)
```

Out[37]:
```
ip.hdr_len              True
ip.tos                  True
ip.id                  False
ip.flags.rb             True
ip.flags.df            False
ip.flags.mf             True
ip.frag_offset          True
ip.ttl                 False
ip.checksum            False
ip.len                 False
ip.dsfield             False
tcp.srcport            False
tcp.dstport            False
tcp.seq                False
tcp.ack                False
tcp.len                False
tcp.hdr_len            False
tcp.flags.fin          False
tcp.flags.syn          False
tcp.flags.reset        False
tcp.flags.push         False
tcp.flags.ack          False
tcp.flags.urg          False
tcp.flags.cwr          False
tcp.window_size        False
tcp.checksum           False
tcp.urgent_pointer     False
tcp.options.mss_val    False
dtype: bool
```

In [38]:
```
# remove columns with zero variance
full_data.drop(columns=['ip.hdr_len', 'ip.tos', 'ip.flags.rb',
                        'ip.flags.mf', 'ip.frag_offset'], axis=1, inplace=True)
```

5. Graphs:





Comparison of the ML Algorithms used for LAN

ML Algorithm	Storage used(KB)	Inference Time(ms)	CPU Usage(%)	RAM Usage(%)
Random Forest	223 KBytes	18.5(ms)	4.00-5.00%	11-11.5%
Naive Bayes	1.3KBytes	4(ms)	1.5-2.5%	10-12%
Logistic Regress ion	939KBytes	3.5(ms)	1.2-1.25%	10.50-11%
Decisio n Tree	13KBytes	4.5(ms)	2-3.5%	11.5-12.5%

Google Colab Link: https://colab.research.google.com/drive/1czqSfA1Ka-9B2Zv9hzpY90-hSFxXoDv4?usp=sharing

INTERNET

1. Convert features that were extracted from pcap as hexadecimal Filled the values with NaN with Zero.

```
In [7]: fields = ['eth.type', 'ip.id', 'ip.flags', 'ip.checksum', 'ip.dsfield', 'tcp.flags', 'tcp.checksum']

        for field in fields:
            df[field] = df[field].apply(lambda x: int(str(x), 16))
```

```
In [8]: bonafide = bonafide.fillna(0)
        for field in fields:
            bonafide[field] = bonafide[field].apply(lambda x: int(str(x), 16))
```

2. The packet structure for IPv4 and IPv6 are different. We have made the model for TCP value 6. Hence we are removing a protocol field different from TCP(value 6).

Check if there are packets with protocol field different than TCP (value 6)

```
In [10]: wrong_proto = full_data[full_data['ip.proto'] != 6]['label'].value_counts().values
         full_data = full_data[full_data['ip.proto'] == 6]
         print("Found and removed", wrong_proto,"packets from the original dataset.")

         Found and removed [52177] packets from the original dataset.
```

3. Features not applicable to this study according to Base Paper:

- Remove features from link layer - layer 2:
  - ➢ frame_info.time
  - ➢ frame_info.encap_type
  - ➢ frame_info.time_epoch
  - ➢ frame_info.number
  - ➢ frame_info.len
  - ➢ frame_info.cap_len
  - ➢ eth.type

- Remove features that are redundant or or non-variant (constant)

  - ➢ ip.version - we consider only IPv4
  - ➢ ip.proto - this study is applicable only to TCP
  - ➢ ip.src - this attribute is removed to allow the generalization of learning (not learn past attackers)
  - ➢ ip.dst - this attribute is removed to allow the generalization of learning (not learn past targets)
  - ➢ ip.flags - this is removed because we use bit-set of flags
  - ➢ tcp.flags - this is removed because we use bit-set of flags

In [12]:
```
full_data.drop(columns=['frame_info.time', 'frame_info.encap_type', 'frame_info.time_epoch', 'frame_info.number',
                        'frame_info.len', 'frame_info.cap_len', 'eth.type', 'ip.flags', 'ip.src', 'ip.dst',
                        'ip.version', 'ip.proto', 'tcp.flags'], axis=1, inplace=True)
```

4. Features with zero variance (not useful for learning). Hence we are checking and dropping those columns where variance==0 is true.

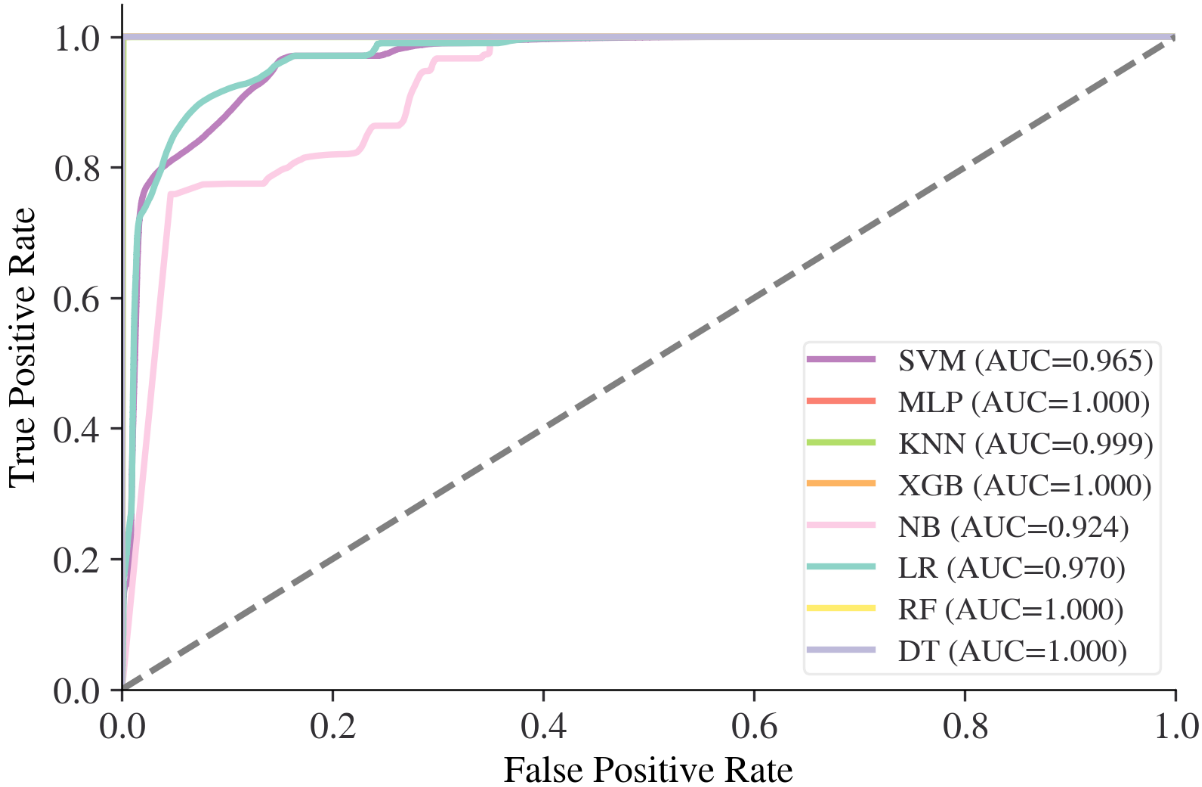
In [15]:
```
# check features with zero variance, they do not support the learning task
(full_data.var() == 0)
```

Out[15]:
```
ip.hdr_len               True
ip.tos                   True
ip.id                   False
ip.flags.rb              True
ip.flags.df             False
ip.flags.mf             False
ip.frag_offset           True
ip.ttl                  False
ip.checksum             False
ip.len                  False
ip.dsfield              False
tcp.srcport             False
tcp.dstport             False
tcp.seq                 False
tcp.ack                 False
tcp.len                 False
tcp.hdr_len             False
tcp.flags.fin           False
tcp.flags.syn           False
tcp.flags.reset         False
tcp.flags.push          False
tcp.flags.ack           False
tcp.flags.urg           False
tcp.flags.cwr           False
tcp.window_size         False
tcp.checksum            False
tcp.urgent_pointer      False
tcp.options.mss_val     False
dtype: bool
```

Remove columns with variance zero

In [16]:
```
full_data.drop(columns=['ip.hdr_len', 'ip.tos', 'ip.flags.rb',
                        'ip.flags.mf', 'ip.frag_offset'], axis=1, inplace=True)
```

5. Graphs:



Comparison of the ML Algorithms used for Internet ML

ML Algorithms	Storage Usage	Inference Time(ms)	CPU Usage(%)	RAM Usage
kNN	108MB	315ms	4-5%	20-27%
RF	223KB	20ms	4-5%	10-12%
NB	1.3KB	4.5ms	1-1.5%	10-14%
MLP	16KB	4ms	1.5-2.5%	10.5-12.5%
SVM	890B	4ms	1.9-3.9%	10-13%
LR	939B	4ms	1.72-3.6%	11-15%
DT	13KB	4ms	2-2.5%	12-14%
XGB	279KB	3.5ms	3-4.5%	25%

Google Colab Link: https://colab.research.google.com/drive/1zDx_kXkbCIDs3W9vntIYtdx_vzsrKT07?usp=sharing

Conclusion

In this project we are demonstrating ABTRAP (Attack, Bonafide, Train, RealizAtion, and Performance), a framework consisting of Steps to keep a record of attacks and authenticity and train your machine Learn the model and realize (implement) the target solution Machin the protection module to evaluate its performance. One of the main concerns when implementing an effective network Intrusion detection system (NIDS) is an adaptive feature For new attacks and the evolution of network traffic. ABTRAP systematizes the entire design chain and implements NIDS solutions and forming pipelines Create protection modules for evolving malicious things Activity. In addition, ABTRAP A process of repeated sources of network traffic (malicious) Or bonafide) is a module that can be input and updated (eg bonafide) Suitable for wireless updates or Linux kernel modules) The output. I am testing ABTRAP in two environments, LAN and LAN. the Internet. In both cases, resource utilization can be kept low. Protection modules and decision trees provide the best Training and implementation phase performance. In the first case study, the results show an f1 score of 0.96. Very little overhead this corresponds to Implementation of kernel space using Linux kernel modules. In an internet case study, a still image of a decision tree represents one Good choice; but other modules are also candidates Implemented as in the case of logistic regression. We see Great overhead compared to the first case Research, and one of the reasons for this is postponement Implementation from kernel space to user space. Hence we conclude that Naive Bayes is the most optimal classifier to be used in an Internet environment since it has very less inference time, and due to less use of storage space and CPU usage does not require high end systems to run the classifier.

In a LAN environment decision tree classifier has proved to be the best due to less inference time and CPU usage, even though it can be seen that RAM usage is more compared to Random Forest the overall requirements and throughput is optimal in Decision Tree.

REFERENCES –

1. R. VINAYAKUMAR, MAMOUN ALAZAB (Senior Member, IEEE), K. P. SOMAN , PRABAHARAN POORNACHANDRAN , AMEER AL-NEMRAT , AND SITALAKSHMI VENKATRAMAN.
   Link: - https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8681044
2. Reference: - Ansam Khraisat ,Iqbal Gondal, Pete Vamplew & Joarde Kamruzzaman Link: - https://cybersecurity.springeropen.com/articles/10.1186/s42400-019-0038-7
3. Reference: - Anthi, Eirini, Williams, Lowri, Malgorzata, Slowinska, Theodorakopoulos, Georgios and Burnap, Peter 2019. A supervised intrusion detection system for smart home IoT devices. IEEE Internet of Things.
4. Reference: - MOHAMMED HASAN ALI , BAHAA ABBAS DAWOOD AL MOHAMMED ALYANI, MOHAMAD FADLI ZOLKIPLI
   Link:- https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8326489
5. P. Verma, S. Anwar, S. Khan and S. B. Mane, "Network Intrusion Detection Using Clustering and Gradient Boosting," 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2018, pp. 1-7, doi: 10.1109/ICCCNT.2018.8494186
6. Amin Shahraki, Mahmoud Abbasi, Øystein Haugen. Boosting algorithms for network intrusion detection: A comparative evaluation of Real AdaBoost, Gentle AdaBoost and Modest AdaBoost. Engineering Applications of Artificial Intelligence, Volume 94, 2020, 103770, ISSN 0952-1976.
7. Mustapha Belouch, Salah El Hadaj, Mohamed Idhammad, Performance evaluation of intrusion detection based on machine learning using Apache Spark, Procedia Computer Science,Volume 127,2018,Pages 1- 6,ISSN 1877-0509
8. Hendrik, Frans & Leenen, Louise & de la Harpe, Retha. (2017). Ant Colony Induced Decision Trees for Intrusion Detection.
9. N. Shone, T. N. Ngoc, V. D. Phai and Q. Shi, "A Deep Learning Approach to Network Intrusion Detection," in IEEE Transactions on Emerging Topics in

Computational Intelligence, vol. 2, no. 1, pp. 41-50, Feb. 2018, doi: 10.1109/TETCI.2017.2772792.

10. M. Zaman and C. Lung, "Evaluation of machine learning techniques for network intrusion detection," NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium, 2018, pp. 1-5, doi: 10.1109/NOMS.2018.8406212.
11. K. Jiang, W. Wang, A. Wang and H. Wu, "Network Intrusion Detection Combined Hybrid Sampling With Deep Hierarchical Network," in IEEE Access, vol. 8, pp. 32464-32476, 2020, doi: 10.1109/ACCESS.2020.2973730.
12. F. A. Khan, A. Gumaei, A. Derhab and A. Hussain, "A Novel Two-Stage Deep Learning Model for Efficient Network Intrusion Detection," in IEEE Access, vol. 7, pp. 30373-30385, 2019, doi: 10.1109/ACCESS.2019.2899721.
13. Khan, Sharfuddin & Eswaran, Sivaraman & Honnavalli, Prasad. (2020). Performance Evaluation of Advanced Machine Learning Algorithms for Network Intrusion Detection System. 10.1007/978-981-15-3020-3_6
14. Acosta, M. R. C., Ahmed, S., Garcia, C. E., & Koo, I. (2020). Extremely randomized trees-based scheme for stealthy cyber-attack detection in smart grid networks. *IEEE access*, *8*, 19921-19933
15. Wang, Q., & Wei, X. (2020, January). The detection of network intrusion based on improved adaboost algorithm. In Proceedings of the 2020 4th International Conference on Cryptography, Security and Privacy (pp. 84-88).
16. I. Sumaiya Thaseen, B. Poorva and P. S. Ushasree, "Network Intrusion Detection using Machine Learning Techniques," 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic- ETITE), 2020, pp. 1-7, doi: 10.1109/ic-ETITE47903.2020.148.