Sniff, Learn, detect: A Lightweight ML-Based Intrusion Detection Method Using Packet Features

Minor Project Research Report

Prepared by

Harshita Kumawat (2022BTECH040)

Kushagra Gupta (2022BTECH051)

(Section – A)

Under the guidance of

Dr. Devika Kataria

Submitted in
partial fulfillment of the requirements
for the course of

Minor Project (PR1103)

Institute of Engineering & Technology (IET)
JK Lakshmipat University

(April 2025)

ABSTRACT

Network sniffing is a technique used to monitor and analyze network traffic by identifying the types of packets, their source and destination IP addresses, ports, and the protocols in use. The effectiveness of packet sniffers can be significantly enhanced by incorporating machine learning algorithms to detect potential cyber-attacks targeting specific hosts.

In this study, live network traffic was captured using a sniffer developed with raw sockets, and packets from a DHCP starvation attack—generated through the Yersinia tool—were collected. The captured packets were stored in PCAP format and subsequently converted into structured CSV datasets for the purpose of attack analysis. Through feature extraction and preprocessing, key attributes such as IP addresses, port numbers, protocol types (e.g., TCP, UDP, DHCP DISCOVER, ICMP), and packet sizes were derived and transformed into suitable inputs for machine learning models.

A Naive Bayes classifier was employed to categorize the packets as either normal or malicious. The study emphasizes enhancing detection accuracy, reducing false positive rates, and improving the adaptability and efficiency of the model. Techniques such as Laplace smoothing, feature binning, and scalable training were used to address the zero-probability problem and ensure model robustness.

To go further that just malicious/benign classification, we will use Decision tree to further analyze components. After classification, Decision Tree is used further analyzing based on entropy or information gain for every feature to identify whether the attack had “Just Started,” is “Under Progress,” or is in a “Critical” stage. This makes it easier to understand the contribution of features towards the final decision.

This paper proposes a computationally efficient and effective approach for real-time network threat detection. The integration of machine learning into packet sniffing holds significant promise for strengthening network security, particularly in Intrusion Detection Systems (IDS) and broader cybersecurity infrastructures.

INTRODUCTION

In this wide digital era, computer networks are very prone to cyber-attacks as they are exposed to large amount of data. Among these, the DDoS or DHCP starvation or ICMP flooding attacks are the most commons ones.

Traditional Intrusion Detection Systems (IDS) are not that adaptable and scalable. They require manual attention and will not always recognize and detect the modern attack patterns which can be solved using Machine Learning algorithms that is based on learning data to distinguish between normal and anomalous data packets in real-time.

The project aims to enhance packet sniffing and attack detection by:

- Capturing live traffic
- Extracting the useful features from the PCAP file
- Training a machine learning model to classify malicious vs normal packets
- Further using entropy-based Decision Tree (ID3) to determine how severe the attack is.

The final objective is to make a simple framework to detect the novel attack patterns and improve IDS.

PROBLEM STATEMENT

In the evolving field of cyber-security, one of the most challenging issues faced by organization in detection of the Distributed Denial of Service (DDoS) attacks and flooding networks using different type of packets. In this attack, the network gets flooded with extensive traffic (packets) making it to slow down or even crashes the system degrading the performance.

The traditional Intrusion Detection Systems (IDS) used were all used signature based techniques to detect attacks which is unable to detect the novel attacks and recognize patterns. Because of these pre-defined rules, they fail to detect the attack if there is a change in method or pattern of the attack. They can also generate false positives, which means marking normal traffic as an attack or false negatives, which means missing the actual attacks.

Another big challenge is the network traffic is huge and keeps getting larger and complex with increasing digitalization. Analyzing raw packets in such case is a heavy task and time-consuming, which is not practical in real-life scenario. So there is a need of a framework that is automated, accurate and intelligent enough to analyze the packet traffic and detect is as early as possible.

This project focuses on solving these problems by using machine learning techniques:

- Naïve bayes Classification to predict weather the network packets are normal or malicious based on the features like protocol, SYN flag, and traffic count.
- A Decision Tree model to further classify the level of threat based on its severity, such as weather the attack is just started, in progress or at a critical stage.

By combining both the methods, the system detects the attack and provide accurate insight about the severity of the attack. This helps to improve decision making in real-life scenarios.

DATASET DESCRIPTION

The dataset used in this project consists of network packet data collected from live traffic when there is normal activity and a simulated attack scenario. The attack simulated was a DHCP starvation using Yersinia tool. The packets were captured using the Wireshark tool and traffic was stored in a PCAP format. And for analysis, PCAP file was converted in csv file to use pandas library to preprocess and visualize the data.

DATA COLLECTION PROCESS

- Network traffic was monitored and captured in real-time using Wireshark
- To create a labeled dataset, a DHCP Starvation Attack was executed. This type of attack floods the network with DHCP DISCOVER messages to exhaust IP addresses from the DHCP server. Also, ICMP flooding was done.
- The raw PCAP data was converted to CSV format using Scapy. Key fields like source IP, destination IP, protocol type, source and destination ports were extracted.

FEATURES USED

After cleaning and preprocessing, the dataset is enhanced with following features:

- SYN_Packet: Indicating weather the TCP packet has SYN flag set or not.
- src / dst: Source and destination IP addresses.
- proto: Protocol used (TCP, UDP, ICMP, DISCOVER, Other).
- count: Number of packets grouped by src, dst, proto, and SYN flag.
- attack: Binary label indicating whether the traffic is normal (0) or malicious (1).

PURPOSE

- Training and testing machine learning models like Naive Bayes and Decision Trees to classify packets as normal or malicious.
- Understanding patterns in traffic behavior based on the features extracted.
- Simulating the real-world scenarios to evaluate how well the model performs.
- Supporting feature-based classification and making decision based on traffic patterns rather than using pre-defined rules.

By building the dataset from scratch and using real-world traffic, we ensure that the data is customized and relevant for practical security applications.

DATA PREPROCESSING

Data Preprocessing is one of the most important steps while applying machine learning algorithms. In this project, the raw data captured is unstructured, noisy and not directly usable for model. So is processed to make it clean and organized, transforming it into structured format making it effective for classification and attack detection.

The following steps were carried out:

- Conversion to PCAP to CSV using python Scapy library.
- Features are extracted: source and destination IP addresses, protocol type and packet count.
- A binary column for SYN packet was created, where 1 if SYN flag is present, else 0.
- Dropping unnecessary columns.
- Grouping and counting packets based on src, dst, proto, and SYN_Packet, and a count column.
- Encoding categorical data as machine learning model works best with numerical data. proto, src, and dst columns were encoded using label encoding to convert them into numeric values.
- Features like count, proto_encoded, and SYN_Packet were normalized using min-max scaling to ensure all features were on the same scale.

Following were the outcomes:

- Clean, simplified and structured dataset.
- Reduced Noice and removal of irrelevant data. Clearly defined features.
- Highlight pattern in traffic behavior.
- Dataset prepared for machine learning model.

METHODOLOGY

Here we used 2 algorithms, Naïve Bayes and Decision Tree as:

- Both are simple to understand and implement giving clear reasons for prediction.
- Our dataset was small having basic features, hence complex models are like Random Forest is not needed.
- These models are quick and easy to train and test. Having light weight and suitable for real-time detection.
- The features are easy to handle and understand making the splitting for Decision Tree easier.
- Naïve Bayes and Decision Tree were the right fit for the balance between accuracy, speed and understanding.

NAÏVE BAYES ALGORITHM

- In Naïve Bayes, features like SYN packet, protocol type and count is used to classify the packets being normal or malicious.
- Laplace Smoothing was applied to avoid zero probabilities in conditional feature distributions.
- The classifier worked on the following principle of computing:
  $P(\text{Attack}|\text{Features}) \propto P(\text{Attack}) \cdot P(\text{SYN}|\text{Attack}) \cdot P(\text{proto}|\text{Attack}) \cdot P(\text{count}|\text{Attack})$
- Predictions were made for all protocol types: TCP, UDP, ICMP, DISCOVER, Others

DECISION TREE

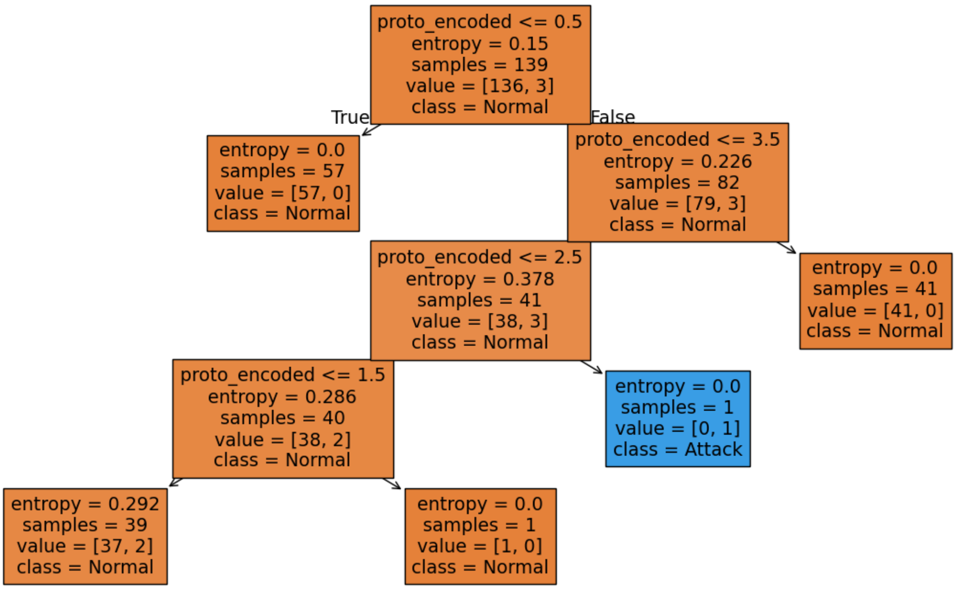
To gain more information about attack rather than just normal or malicious packets, a Decision Tree is implemented to determine the severity of the attack based on ID3 algorithm (Iterative Dichotomiser 3). ID3 uses entropy and information gain to select the best feature for splitting the data and building a decision/classification tree.

STEPS:

- The count has 3 levels: low, medium and high based on traffic intensity
  - low: count ≤ 0.0005
  - medium: $0.0005 <$ count ≤ 0.005
  - high: count > 0.005
- Features used
  - SYN packet indicating 1 if SYN flag present, else 0.
  - proto_encoded: Protocol type encoded as an integer (e.g., TCP = 0, UDP = 1, ICMP = 2, etc.)
  - count for intensity of network traffic.
- Training
  - A Decision Tree classifier is used using Skit-learn library of python.
  - The splitting was done based on entropy using ID3 logic.

- The maximum tree depth was set to 4 to avoid overfitting and maintain interpretability.
- The features used to train are: proto_encoded (encoded protocol type), count_binned (traffic volume), and SYN_Packet (SYN flag indicator).

- Visualization of tree
    - Root node has proto_encoded meaning protocol type.
    - If protocol type is TCP (<= 0.5), all packets are normal.
    - And for other protocol types, further checks like count and SYN flag were checked.
    - The attack predicted by model is shown in blue
    - Many of the branched predict normal traffic due to fewer attack samples.
    - Entropy is low which means the splits are clear and easy
    - The protocol type and count are the most important features



RESULTS AND EVALUATION

The goal is to detect weather the packet is normal or malicious based on the machine learning technique Naïve Bayes and computing its severity using Decision Tree. The system was evaluated based on how accurately it could classify the traffic and provide insight to severity of the attack.

ENCODING AND NORMALIZING DATA

	SYN_Packet	src	src_encoded	dst	dst_encoded	proto	proto_encoded	count	attack
0	0.0	0.0.0.0	0.000000	255.255.255.255	0.403846	DISCOVER	0.75	1.000000	1
1	0.0	103.180.115.15	0.019608	192.168.1.15	0.250000	TCP	0.00	0.001202	0
2	0.0	103.97.125.214	0.039216	192.168.1.15	0.250000	TCP	0.00	0.000418	0
3	0.0	103.97.125.214	0.039216	192.168.1.15	0.250000	UDP	0.25	0.000784	0
4	0.0	104.21.112.1	0.058824	192.168.1.15	0.250000	TCP	0.00	0.000000	0
...	...	...	...	...	...	...	...	...	...
134	1.0	2401:4900:889a:61a8:6841:2c42:e998:c95c	0.392157	2606:4700:8399:a0f8:3d39:4:d411:63fc	0.653846	Other	1.00	0.000052	0
135	1.0	2401:4900:889a:61a8:6841:2c42:e998:c95c	0.392157	2606:4700:839b:ec90:d739:1:6f6d:7354	0.673077	Other	1.00	0.000052	0
136	1.0	2401:4900:889a:61a8:6841:2c42:e998:c95c	0.392157	2606:4700:91b9:a0f8:3d39:1:d411:63fc	0.692308	Other	1.00	0.000052	0
137	1.0	2401:4900:889a:61a8:6841:2c42:e998:c95c	0.392157	2620:127:f00f:ff01::	0.730769	Other	1.00	0.000209	0
138	1.0	2401:4900:889a:61a8:6841:2c42:e998:c95c	0.392157	2a04:fa87:fffe::c000:4902	0.788462	Other	1.00	0.000052	0

NAÏVE BAYES RESULTS

Output:

```
🔍 **Protocol: TCP (0)**
🧮 Naive Bayes Scores:
P(Attack|F) = 0.0004316547
P(Normal|F) = 0.0005917426
✅ Prediction: **NORMAL**

🔍 **Protocol: UDP (1)**
🧮 Naive Bayes Scores:
P(Attack|F) = 0.0004316547
P(Normal|F) = 0.0004285033
🚨 Prediction: **ATTACK**

🔍 **Protocol: ICMP (2)**
🧮 Naive Bayes Scores:
P(Attack|F) = 0.0004316547
P(Normal|F) = 0.0000102025
🚨 Prediction: **ATTACK**

🔍 **Protocol: DISCOVER (3)**
🧮 Naive Bayes Scores:
P(Attack|F) = 0.0004316547
P(Normal|F) = 0.0000102025
🚨 Prediction: **ATTACK**

🔍 **Protocol: Other (4)**
🧮 Naive Bayes Scores:
P(Attack|F) = 0.0004316547
P(Normal|F) = 0.0000102025
🚨 Prediction: **ATTACK**
```

Note: Some misclassifications occurred due to overlap in feature values for attack and normal packets, which is a limitation of Naive Bayes being a simplistic model assuming feature independence.

DECISION TREE

Decision Tree classifier is using ID3 logic to train processed dataset using features like SYN_Packet, proto_encoded, and count. It learned to split on the basis of entropy using protocol types, traffic volume, and SYN flag presence.

- Protocol is the root node.
- The tree was able to correctly flag DISCOVER traffic as an attack.
- All other protocols like TCP, UDP, ICMP, and Other were predicted as normal due to limited attack samples.

```
🔍 Predictions for each protocol using ID3 Decision Tree

🔍 Protocol: TCP (0) → ✅ NORMAL
🔍 Protocol: UDP (1) → ✅ NORMAL
🔍 Protocol: ICMP (2) → ✅ NORMAL
🔍 Protocol: DISCOVER (3) → 🚨 ATTACK
🔍 Protocol: Other (4) → ✅ NORMAL
```

CONCLUSION

In this project, we designed and implemented a framework with packet sniffing, machine learning classification and decision tree analysis to detect network attack in real-time.

Initiating from capturing network packets, simulating DDoS and DHCP starvation attacks. The final PCAP file was created, then for analysis, important features like protocol type, SYN flags and packet count were extracted building a custom dataset reflecting real-world traffic. After preprocessing (encoding and normalization) the data is final as input for machine learning model.

A Naïve Bayes classifier was used to classify packets as normal or malicious based on the probability calculations using Laplace Smoothing. Laplace Smoothing helped to handle zero probability problem to detect attack probability based on the given features.

Then further we built Decision Tree (ID3 algorithm) to classify packets and analyze the severity of the attack. Splitting in Decision tree was based on entropy and information gain for making decision by understanding how different features can contribute.

Naïve Bayes provided simple and quick results for binary classification. And Decision Tree further break it down in stages like normal, attack just started, attack in progress or critical attack based on feature threshold.

So, finally this project shows, how combining statistical model and machine learning algorithm can help create more intelligent and detailed Intrusion Detection System (IDS). This method will adapt new attack pattern and improve real-time network defense making it more efficient and contributing to modern cyber security solutions.

LEARNING OUTCOME

Following is the learning we achieved:

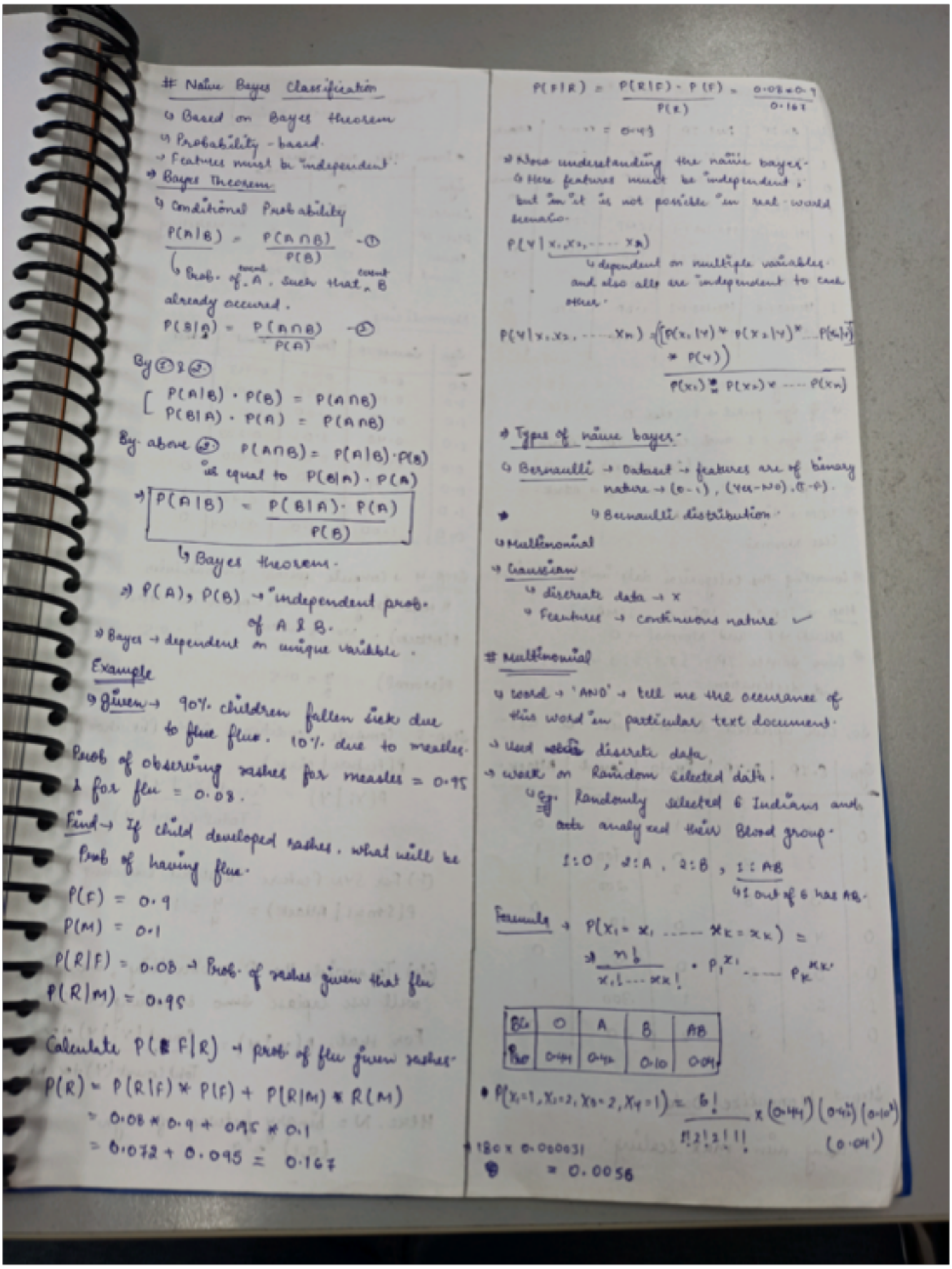
- Understood the network traffic and how different type of traffic behave (TCP, UDP, ICMP, DHCP DISCOVER).
- Data processing skills were improved and how cleaning, transforming, encoding, and normalizing real-world data is important for analysis and suitable for machine learning models.
- Applied Naïve bayes and Decision Tree algorithms and solved the problems like zero probability problem using Laplace Smoothing and splitting in Decision Tree.
- Identified the most relevant features that will be useful to analyze the traffic behavior so that better model performance can be there.
- Developed an automated framework as a real-world solution adapting the traditional Intrusion Detection System (IDS).

REFERENCES

1. Gregorczyk, M., Żórawski, P., Nowakowski, P., Cabaj, K. and Mazurczyk, W., 2020. Sniffing detection based on network traffic probing and machine learning. IEEE Access, 8, pp.149255-149269.
2. Rahman, O., Quraishi, M.A.G. and Lung, C.H., 2019, July. DDoS attacks detection and mitigation in SDN using machine learning. In 2019 IEEE world congress on services (SERVICES) (Vol. 2642, pp. 184-189). IEEE.
3. SEED Labs – Network Security Labs: Sniffing and Spoofing Lab. SEED Project, Syracuse University. Available at: https://seedsecuritylabs.org/Labs_20.04/Networking/Sniffing_Spoofing/
4. Wenliang (Kevin) Du, Computer & Internet Security: A Hands-on Approach, 3rd Edition, Syracuse University, 2023

APPENDICES

NAÏVE BAYES HAND CALCULATION



Naive Bayes Classification

- Based on Bayes theorem
- Probability - based.
- Features must be independent.

Bayes Theorem

- Conditional Probability

$P(A|B) = \frac{P(A \cap B)}{P(B)}$ - ①

Prob. of event A, such that event B already occured.

$P(B|A) = \frac{P(A \cap B)}{P(A)}$ - ②

By ① & ②

$P(A|B) \cdot P(B) = P(A \cap B)$
$P(B|A) \cdot P(A) = P(A \cap B)$

By above ② $P(A \cap B) = P(A|B) \cdot P(B)$ is equal to $P(B|A) \cdot P(A)$

$\boxed{P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}}$

Bayes theorem.

$P(A), P(B)$ → independent prob. of A & B.

Bayes → dependent on unique variable.

Example

Given → 90% children fallen sick due to flue flu. 10% due to measles. Prob of observing rashes for measles = 0.95 & for flu = 0.08.

Find → If child developed rashes, what will be Prob of having flu.

$P(F) = 0.9$
$P(M) = 0.1$
$P(R|F) = 0.08$ → Prob. of rashes given that flu
$P(R|M) = 0.95$

Calculate $P(F|R)$ → prob. of flu given rashes.

$P(R) = P(R|F) * P(F) + P(R|M) * R(M)$
$= 0.08 * 0.9 + 0.95 * 0.1$
$= 0.072 + 0.095 = 0.167$

$P(F|R) = \frac{P(R|F) \cdot P(F)}{P(R)} = \frac{0.08 * 0.9}{0.167}$
$= 0.43$

Now understanding the naive bayes.
Here features must be independent, but it is not possible in real-world scenario.

$P(Y|x_1, x_2, \ldots x_n)$
dependent on multiple variables and also all are independent to each other.

$P(Y|x_1, x_2, \ldots x_n) = \frac{([P(x_1|Y) * P(x_2|Y) * \ldots P(x_n|Y)] * P(Y))}{P(x_1) * P(x_2) * \ldots P(x_n)}$

Types of naive bayes.

- Bernaulli → Dataset → features are of binary nature → (0-1), (Yes-No), (T-F).
  - Bernaulli distribution.
- Multinomial
- Gaussian
  - discrete data → x
  - Features → continuous nature

Multinomial

- word → 'AND' → tell me the occurance of this word in particular text document.
- Used discrete data
- work on Random selected data.
  - Eg. Randomly selected 6 Indians and analysed their Blood group.

1:O, 2:A, 2:B, 1:AB
1 out of 6 has AB.

Formula → $P(X_1 = x_1 \ldots X_K = x_K) = \frac{n!}{x_1! \ldots x_K!} \cdot p_1^{x_1} \ldots p_K^{x_K}$

BG	O	A	B	AB
Pro	0.44	0.42	0.10	0.04

$P(X_1=1, X_2=2, X_3=2, X_4=1) = \frac{6!}{1!2!2!1!} \times (0.44^1)(0.42^2)(0.10^2)(0.04^1)$

180×0.000031
$= 0.0056$

Implementing on dataset

Syn	S. IP	Dest. IP	Proto	Count	Attack
1	192.168.1.2	192.168.1.1	TCP	500	Attack
0	192.168.1.3	192.168.1.1	UDP	5	Normal
1	192.168.1.4	192.168.1.1	TCP	700	A
1	192.168.1.5	192.168.1.1	ICMP	200	A
0	192.168.1.6	192.168.1.1	TCP	10	N
0	192.168.1.7	192.168.1.1	TCP	3	N
1	192.168.1.8	192.168.1.1	UDP	300	A
0	192.168.1.9	192.168.1.1	ICMP	20	N

⇒ labelling the entities.
→ If Syn packet → 1 else 0.
→ If Syn = 1 and count > 100 → Attack (DoS attack)
→ If proto = ICMP & count > 150 → Attack
→ UDM → high → attack.
Else Normal.

⇒ Converting the categorical data into numerical.

Map → TCP → 0, UDP = 1, ICMP → 2
Attack → 1 and Normal → 0.

- Give Source IP = {0, 1, 2, 3, 4, 5, 6, 7} and destination = 8

So, our updated dataset will be as:-

Syn	S. IP	D. IP	Proto	Count	Attack.
1	0	8	0	500	1
0	1	8	1	5	0
1	2	8	0	700	1
1	3	8	2	200	1
0	4	8	0	10	0
0	5	8	0	3	0
1	6	8	1	300	1
0	7	8	2	20	0

Step-2 Normalize Data

↳ Using min-max scaling.

$$\boxed{X_{norm.} = \frac{X - X_{min}}{X_{max} - X_{min}}}$$

- From table, our values.

	Min	Max
Syn	0	1
Source IP	0	7
Dest. IP	8	8 ← neglecting this.
Proto.	0	2
Count	3	700

Normalizing

Syn	Source IP	Prot.	Count	Label
1.0	0.0	0.00	0.713	1
0.0	0.14	0.50	0.002	0
1.0	0.28	0.00	1.0	1
1.0	0.42	1.0	0.283	1
0.0	0.57	0.0	0.010	0
0.0	0.71	0.0	0.0	0
1.0	0.85	0.5	0.426	1
0.0	1.00	1.0	0.024	0

Step-4 → Compute prior probabilities
↳ How likely each class appears in dataset

$P(\text{Attack}) = \frac{\text{No. of attacks}}{\text{Total s}} = \frac{4}{8} = 0.5$

$P(\text{Normal}) = \frac{4}{8} = 0.5$

Step-5 Compute conditional prob. (likelihoods)
P(Feature | class).

$$P(x_i \mid y) = \frac{\text{Count}(x_i \mid y)}{\text{Total Count}(y)}$$

(1.) For SYN Feature (without smoothing)

$P(\text{Syn}=1 \mid \text{Attack}) = \frac{4}{4} = 1.$

(2.) To avoid the 0 probability, we will use laplase ~~smo~~ smoothing ($\alpha = 1$)

For that, $P(x_i \mid y) = \frac{\text{Count}(x_i \mid y) + \alpha}{\text{Total Count}(y) + \alpha \cdot N}$

Here, N = binary features of Syn (0,1) = 2

Now again Calculate

$P(SYN=1 \mid attack) = \frac{4+1}{4+1\times2} = \frac{5}{6} \simeq 0.83$

$P(SYN=0 \mid attack) = \frac{0+1}{4+1\times2} = \frac{1}{6} \simeq 0.167$

$P(Pro.=0 \mid att.) = \frac{2+1}{4+1\times3} = \frac{3}{7} = 0.4286$

$P(P=0 \mid N) = \frac{2+1}{7} = 0.42$

$P(P=1 \mid A) = \frac{1+1}{7} \simeq 0.28$

$P(P=1 \mid N) = 0.28$

$P(P=2 \mid A) = \frac{1+1}{7} = 0.28$

$P(P=2 \mid N) = 0.28$

$P(SYN=1 \mid N) = \frac{1+1}{4+2} = 0.33$

$P(SYN=0 \mid N) = \frac{3+1}{4+2} = 0.66$

→ Now for count, we have to chose the threshold.

→ Observing the dataset.

Attack → {500, 700, 200, 300} → >20

Normal → {5, 10, 3, 20} → ≤20

→ 20 as threshold value

Now for count →

→ low (≤20)

↳ attack → $\frac{0+1}{4+2} = 0.166$

↳ Normal → $\frac{4+1}{4+2} = 0.833$

→ High (>20)

↳ attack → $\frac{4+1}{4+2} = 0.833$

↳ Normal → $\frac{0+1}{4+2} = 0.166$

→ How we will predict whether it is attack packet or not.

→ Calculate

$P(A \mid Features) = P(F \mid A) \times P(A)$

$P(N \mid F) = P(F \mid N) \times P(N)$

If $P(A \mid F) > P(N \mid F)$ → attack

else if $P(N \mid F) > P(A \mid F)$ → normal

Eg. → SYN=1, Proto → 2, Count → 0·8

Calculate → $P(A \mid F)$ & $P(N \mid F)$

$P(A) = 0.5$

$P(N) = 0.5$

$$\left.\begin{aligned} P(SYN=1 \mid A) &= 0.83 \\ P(P=2 \mid A) &= 0.28 \\ P(C>20 \mid A) &= 0.833 \end{aligned}\right] \text{Attack.}$$

$$\left.\begin{aligned} P(SYN=1 \mid N) &= 0.33 \\ P(P=2 \mid N) &= 0.28 \\ P(C>20 \mid N) &= 0.166 \end{aligned}\right]$$

$P(A \mid F) = 0.83 \times 0.28 \times 0.833 \times 0.5$
$= 0.096$

$P(N \mid F) = 0.33 \times 0.28 \times 0.166 \times 0.5$
$= 0.0076$

here $P(A \mid F) > P(N \mid F)$

$0.096 > 0.0076$

→ Attack

DECISION TREE HAND CALCULATION

Decision Tree Algorithm

Dataset for analysis:-

Syn	Proto	Count	label
1	TCP	50	A start
1	TCP	150	A in progr.
1	TCP	300	A critical
0	TCP	5	N
0	TCP	10	N
0	UDP	2	N
0	UDP	3	N
0	ICMP	1	N
0	ICMP	2	N
1	TCP	40	A start
1	TCP	200	A in pro.
1	TCP	350	A critical

Count → Low < 100
Medium → 100 - 250
High > 250

Convert categorical to numerical.

TCP = 0   A Start → 0
UDP = 1   A In-prog. → 1
ICMP = 2   A Critical → 2
Normal → 3

⇒ Calculate Entropy of the Root Node

$H(s) = -\Sigma (pi * \log_2 (pi))$

Total S → 12

Attact start → 2
Attack In prog. → 2
Attack Critical → 2
Normal → 6

$H(S) = -[(2/12) \log_2 (2/12) + (2/12) \log_2 (2/12) + (2/12) \log_2 (2/12) + (6/12) \log_2 (6/12)]$

$\Rightarrow -[0.1667(-2.585) + 0.1667(-2.585) + 0.1667(-2.585) + (0.5)(-1)]$

$\Rightarrow -[-0.4308 - 0.4308 - 0.4308 - 0.5]$

$\Rightarrow$ ~~1.7928~~ 1.7924 --(1)

Calculate Information gain for each potential split.

(1.) Protocol

TCP → 7 (total)
↳ 2 A start, 2 critical
↳ 2 Progress, 2 Nomal

UDP & ICMP → are of all same classes, so, Entropy → 0

$H(TCP) = -[(2/7) \log_2 (2/7) + (2/7) \log_2 (2/7) + (2/7) \log_2 (2/7) + (2/7) \log_2 (2/7)]$

$= 1.8424$

weighted average

↳ Total TCP → 7, UDP = 2, ICMP → 2
↳ Overall Total Samples = 12

$\Rightarrow (7/12)(1.8424) + (2/12)(0) + (2/12)(0)$

$\Rightarrow 1.0747$

Information gain

IG (Protocol) = H(S) - 1.0747   from (1)

$= 1.7924 - 1.0747$

$= 0.7177$

(2.) Syn

Syn = 1 → 6 (2 start , 2 progress, 2 criti.)
Syn = 0 → 6 (all Normal)

$H(Syn = 1) = -3[(^2/_6) \log_2 (^2/_6)]$
$= 1.585$

$H(Syn = 0) \rightarrow 0$

weighted average =
$\Rightarrow (6/12)(1.585) + (6/12)(0)$
$\Rightarrow 0.7925$

$IG(SYN) = 1.7924 - 0.7925$
$= 0.9999$

(3) Count

Count ≤ 100 → 6 → 2 start , 4 Normal
Count > 100 → 6 → 2 progress, 2 critical
2 Normal.

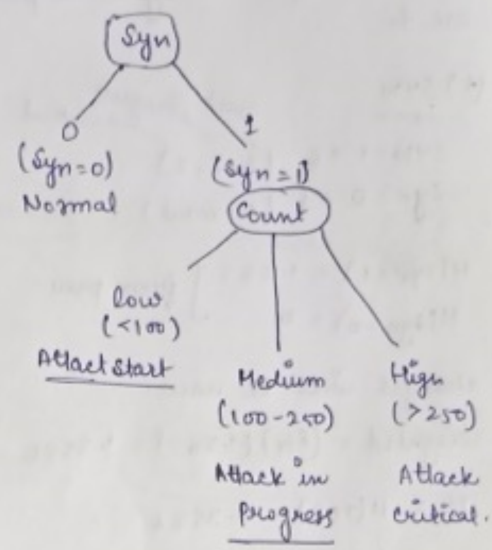
$H(count \le 100) = -[(^2/_6)\log_2(^2/_6) + (^4/_6)\log_2(^4/_6)]$
$= 0.9183$

$H(count > 100) = -[(^2/_6)\log_2(^2/_6) + (^2/_6)\log_2(^2/_6) + (^2/_6)\log_2(^2/_6)]$
$= 1.585$

weighted average
$\Rightarrow (6/12)(0.9183) + (6/12)(1.585)$
$\Rightarrow$ 1.2516

$IG(count) = 1.7924 - 1.2516$
$= 0.5408$ bits.

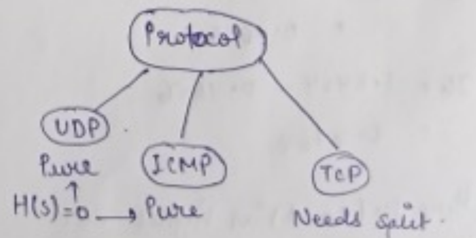
Here, highest IG is of Syn = 0.99
So, If we choose Syn as root Node then our decision tree will be as-



and for Protocol, there will be no split. because for TCP there is Syn = 1 (all in one class, so it is pure).
and Remaining UDP & ICMP the entropy is 0.

~~So, we will stab~~
So, if we start from the Protocol.

TCP → 7 , UDP → 2 , ~~ICPM~~ ICMP → 3



$H(TCP) = 1.8424$

Now again calculate entropy, where TCP is there.

So, total samples having TCP as proto. are 7.

(1.) SYN

SYN=1 → 6 (2, 2, 2) start, Progress, critical

Syn=0 → 1 (Normal)

$H(syn=1) = 1.585$
$H(syn=0) = 0$] from prev.

changes will be now.

weighted = $(6/7)(1.585) = 1.3586$

$IG = H(TCP) - 1.3586$
$= 1.8424 - 1.3586$
$= 0.4838$

(2.) Count.

Count ≤ 100 → 3 (2 start, 1 Normal)

Count > 100 → 4 (2 Progress, 2 critical)

$H(\le 100) = -[(2/3)\log_2(2/3) + (1/3)\log_2(1/3)]$

$H(>100) = -[(2/4)\log_2(2/4) + (2/4)\log_2(2/4)]$

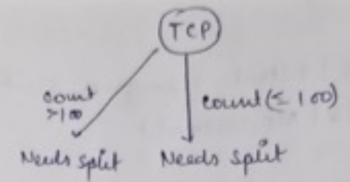
$H(\le 100) = 0.9183$

$H(>100) = 1.0$

weighted → $(3/7)(0.9183) + (4/7)(1.0)$
$= 0.9656$

$IG = 1.8424 - 0.9656$
$= 0.8768$

Here IG (count) is higher so. TCP branch split on count



Now for (count ≤ 100)

(1.) Syn Split

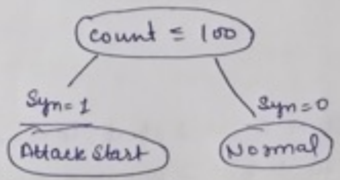
Syn = 1 → 2 (both start)

Syn = 0 → 1 (Normal.

$H(syn=1) = 0$

$H(syn=0) = 0$

IG = perfect split.



Now for count (>100) → definately there will be attack.

So, for split we will take new threshold as 275

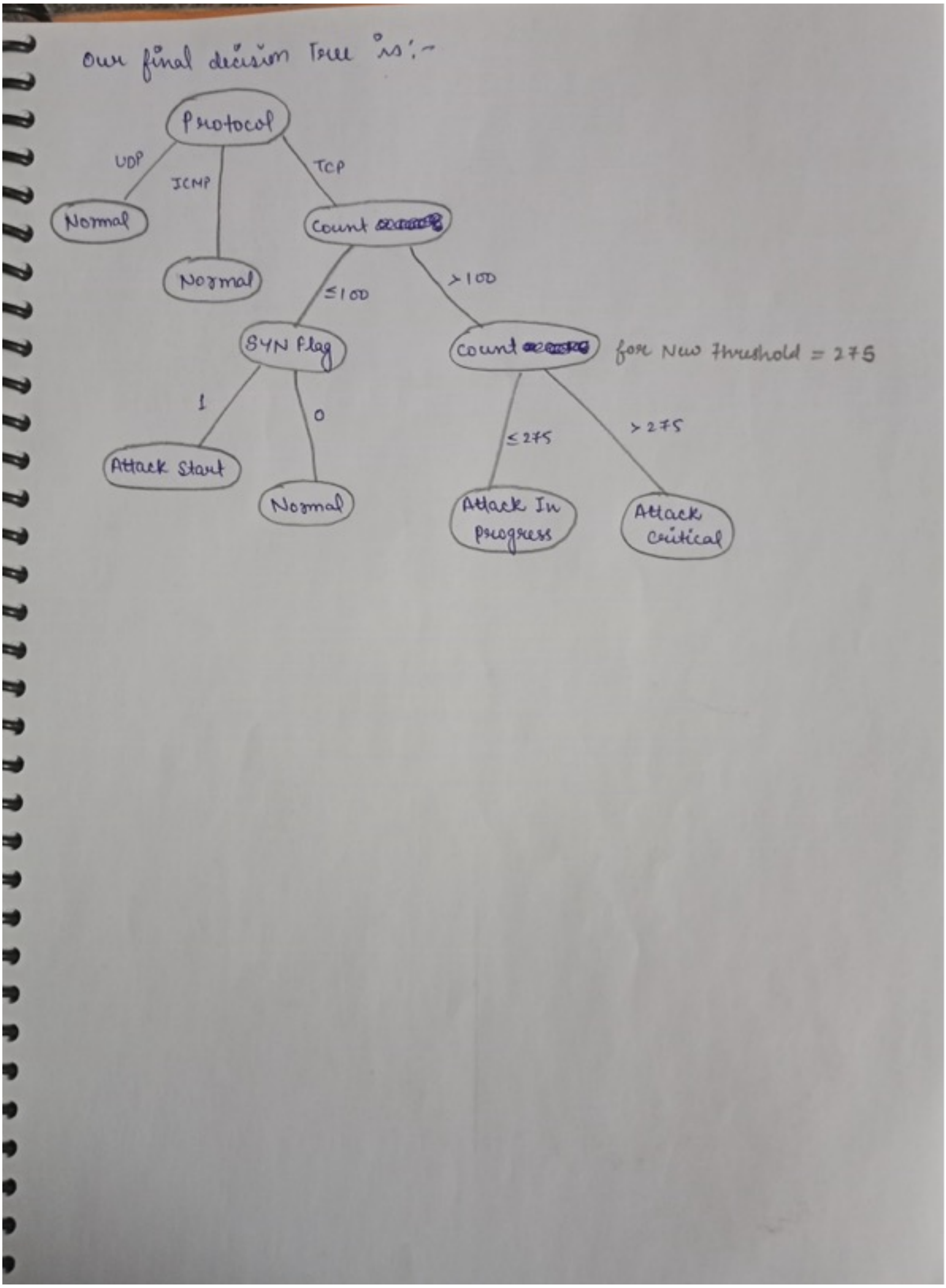
From here
$H(>100) = 1.0$

Possible Split.

count (≤ 275) = 2 → both A progress

count (> 275) = 2 → both A critical

So both are same, then IG = perfect split.


Our final decision Tree is:-
Protocol
UDP
ICMP
TCP
Normal
Normal
Count
≤100
>100
SYN Flag
Count
for New Threshold = 275
1
0
Attack Start
Normal
≤275
>275
Attack In progress
Attack critical