XAI-Driven TinyML with Compact Models for Compromised IoT Device Detection

Under the Supervision of Dr. Mahendra Shukla Sir

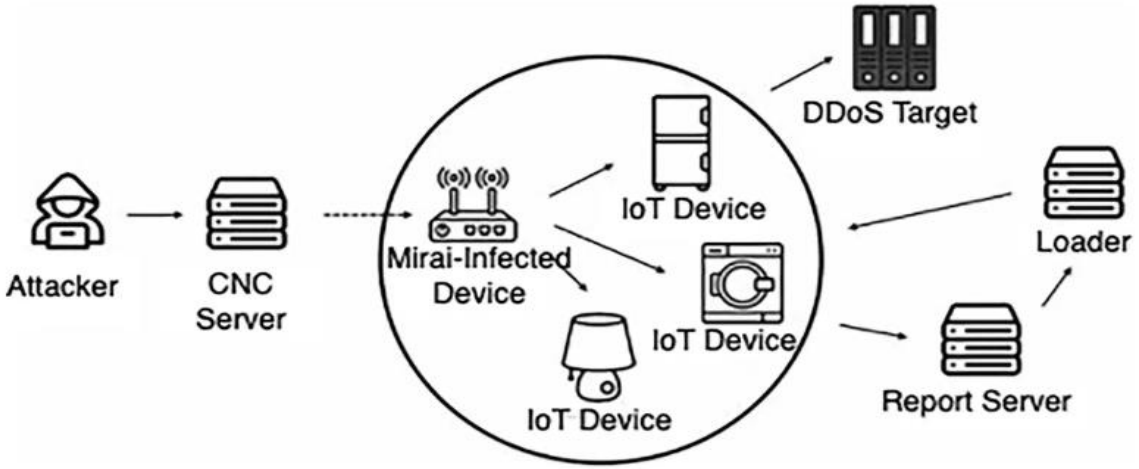
Group Members

NVNL Ishwarya (2021IMT-070)
Bodapatla Rakesh (2021IMT-022)
Gulla Akshaya (2021IMT-038)
V Ramana Reddy (2021IMT-110)

Introduction

- The Internet of Things (IoT) is a diverse ecosystem composed by a plethora of Internet-connected “things” from the real or digital world that can be remotely monitored or managed.
- Currently, many IoT devices are vulnerable because there is always a tradeoff between security and usability. Unfortunately, IoT manufacturers tend to maximize usability, fearing that average users will lose interest if a device cannot be used straight out of the box.

- Consider smart home devices like Wi-Fi-enabled security cameras. Many cameras come with default usernames and passwords (e.g., "admin/admin") to simplify setup. Users rarely change these credentials, making them an easy target for cybercriminals.
- In 2016, the Mirai botnet exploited such weak security, hijacking thousands of IoT devices worldwide to launch one of the most significant DDoS attacks,



Various Attacks in IOT

- **Battery draining attack**: It consists of reducing a device's battery life by continually sending data to it and reducing its "sleep time".
- **Sniffing attack**: The attacker monitors and captures packets that are sent or received by a given network or host to obtain credentials or other sensitive data.
- **Denial of Service (DoS) and Distributed DoS attacks**: An attacker floods a network or a host with huge amounts of data, causing entire systems to become unavailable.
- **IoT cloud service manipulation**: The attacker gains control or access to one or multiple cloud services. When this attack is successful, the intruder can obtain data directly from the database or even disrupt an IoT environment by generating false alerts.

Previous Works

1. Detecting Compromised IoT Devices Through XGBoost by Cruz et al. (IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS) proposed the use of XGBoost on IoT23 Dataset.
2. Energy consumption of on-device machine learning models for IoT intrusion detection (Internet of Things 21, Elsevier) by Tekin et al. performed analysis in terms of model inference time, training time, power consumption, and energy consumption.

Problem Statement

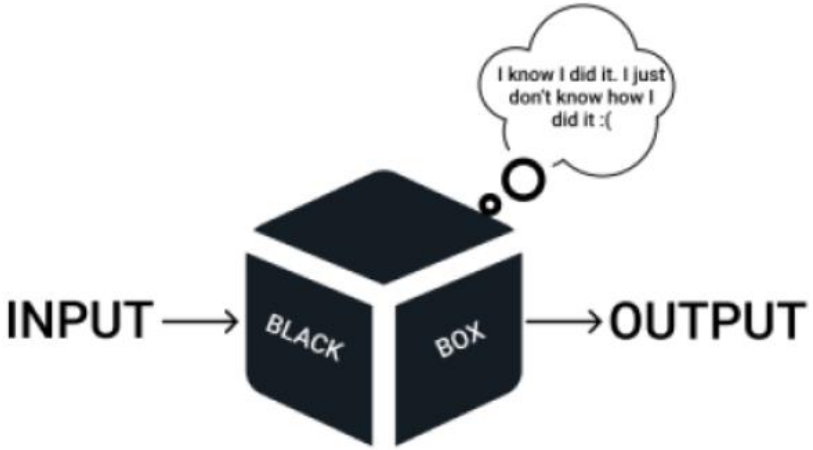
- Explainable Artificial Intelligence (XAI) for **feature selection**, identifying the most relevant and impactful features to enhance model performance while maintaining minimal computational overhead.
- Optimize **model efficiency** by reducing energy consumption, battery usage, and memory usage using Tiny Machine Learning (TinyML), ensuring seamless deployment on IoT edge devices without significantly compromising accuracy.

Dataset Description

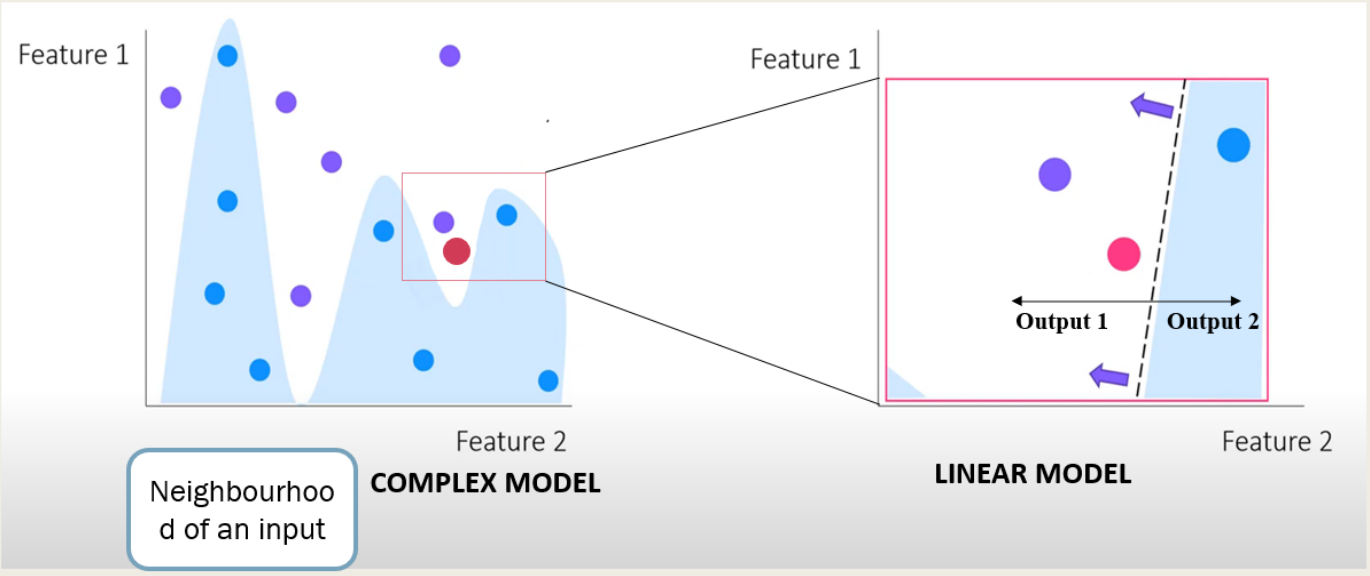
- The dataset used in this study is **IoT23 Dataset**. It consists of total 23 instances where 20 are malicious instance and 3 are benign instances.
- Dataset contains 26 features . Some of them are Duration, orig_ip_bytes, orig_ip_pkts etc.
- The dataset encompasses over 760 million packets and 325 million labeled flows, totaling more than 500 hours of network traffic. We have selected a subset (3.5M) from the dataset.
- The labels are ‘PartOfAHorizontalPortScan', 'Benign', 'Okiru', 'DDoS’, 'C&C-HeartBeat', 'C&C', 'Attack’. All instances other than 'Benign' are considered 'Attack'.

Explainable AI

- Explainable Artificial Intelligence (XAI) was first introduced in the 1980s however, it has not been the focus of research, as AI's growth has instead focused on predictive performance.
- In recent years, the need for XAI has increased as AI models are widely used in critical sectors where explanations for their reasoning and output are required. Example : Healthcare



LIME
(Local Interpretable Model Agnostic Explanations)



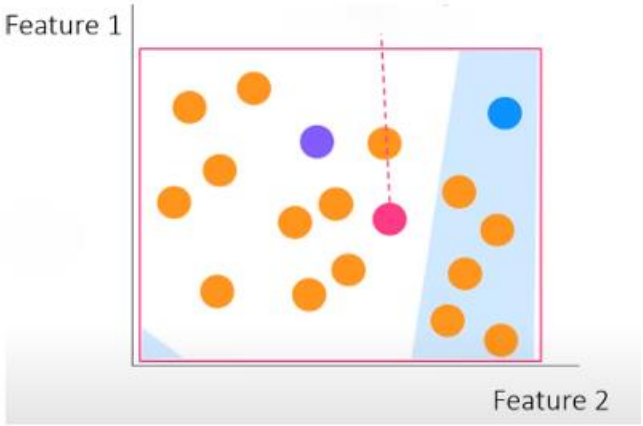


Figure 8: Perturbation of neighbourhood

Mathematical Formulation

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}} \; \mathcal{L}(f, g, \pi_x) + \Omega(g)$$

$x \in \mathbb{R}^d$ — number of features

G — Family of interpretable models

$f : \mathbb{R}^d \rightarrow \mathbb{R}$ — Complex model

g — Simple interpretable model

π_x — Neighbourhood of x

$$\mathcal{L}(f, g, \pi_x) = \sum_{z \in \mathcal{Z}} \pi_x(z) \left(f(z) - g(z)\right)^2$$

$f(z)$ — Complex model prediction

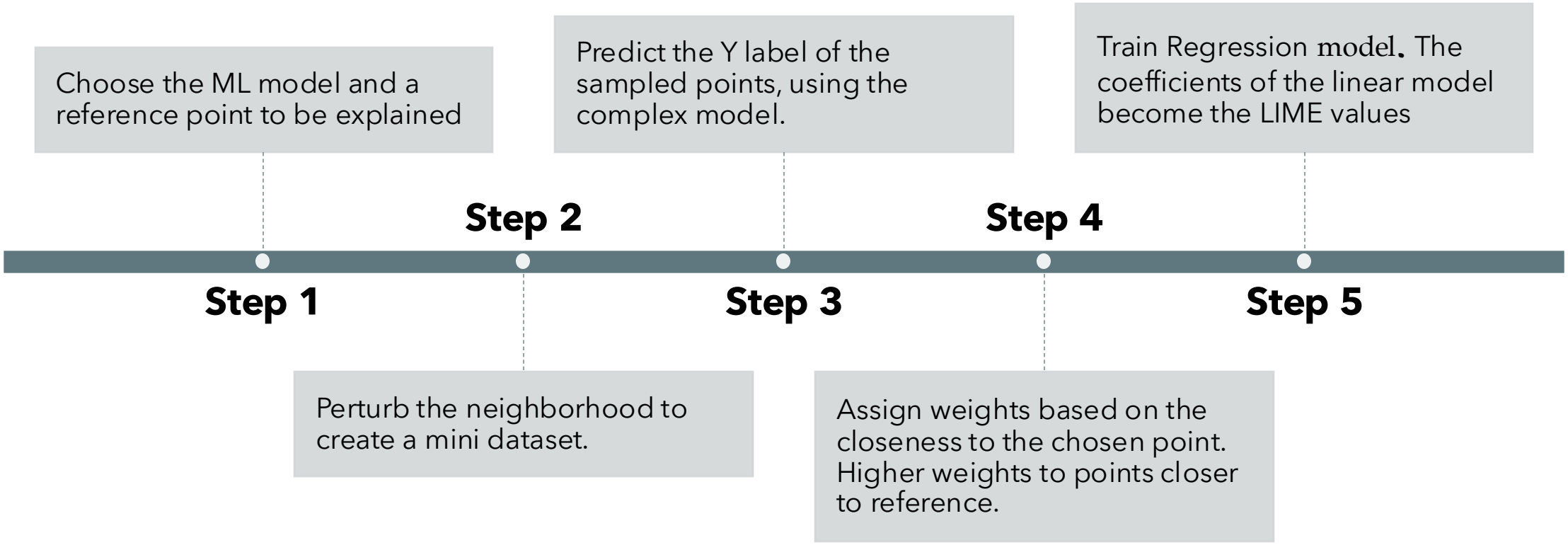
$g(z)$ — Simple model prediction

$$\pi_x(z) = exp(-D(x, z)^2 / \sigma^2)$$

D — Some distance function D

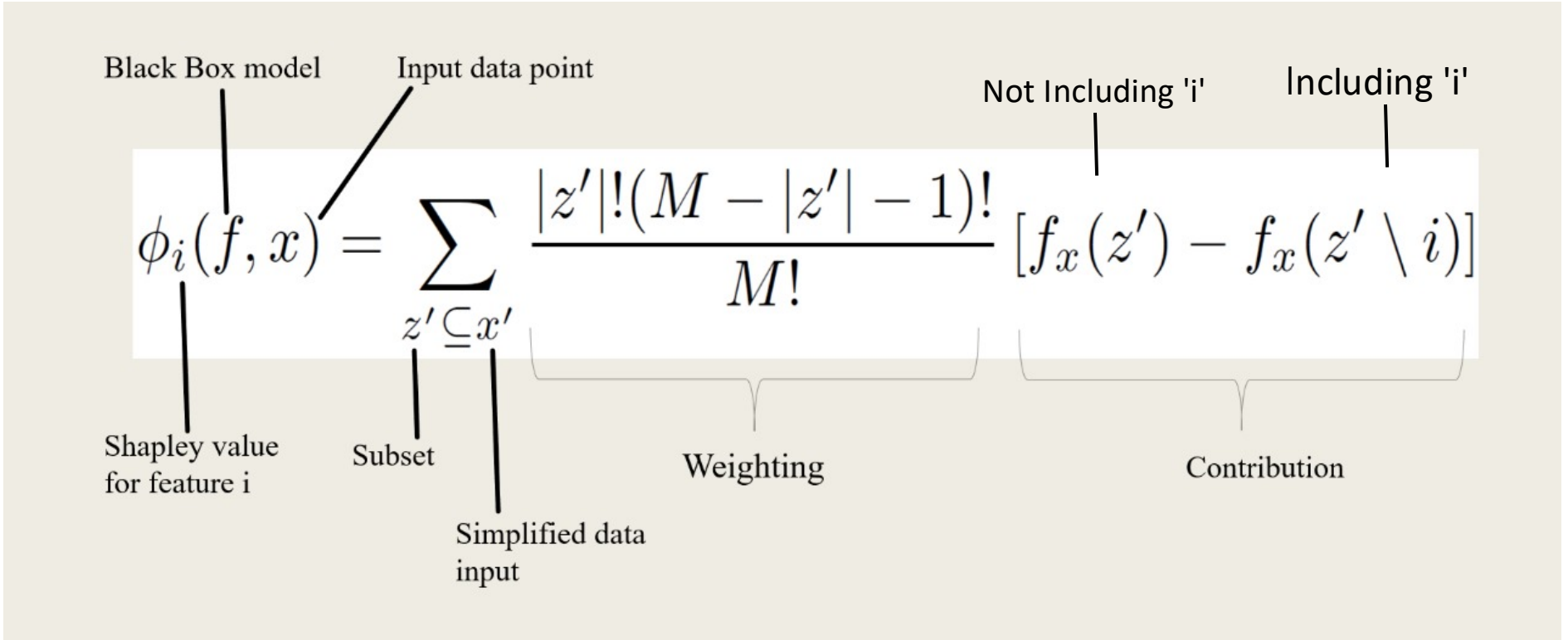
σ — Kernel width

LIME Algorithm



Shapley Additive Explanations (SHAP)

- Originally, the Shapley value was developed to assign a fair share of the total gains to each player based on their contribution to the coalition. This concept comes from Game Theory!
- The key idea is to consider all possible permutations of players and calculate their **marginal contribution** to the **coalition**.
- The idea is extended to Machine Learning. Instead of players in a coalition game, SHAP considers features in a machine learning model. It aims to explain the predictions of complex models by attributing each feature's contribution to the prediction outcome.


Black Box model
Input data point
Not Including 'i'
Including 'i'
$\phi_i(f,x) = \sum_{z' \subseteq x'} \frac{|z'|!(M-|z'|-1)!}{M!} [f_x(z') - f_x(z' \setminus i)]$
Shapley value for feature i
Subset
Simplified data input
Weighting
Contribution

Working of SHAP: An example

- Consider an example where 3 students participated in a Kaggle competition.
- Shapley value = Expected marginal contribution of each student.
- Marginal Contribution = Increase in coalition value due to a student joining a coalition

Position	Prize
1st	10000
2nd	7500
3rd	5000

$C_{abc} = 10000$

$C_{0} = 0$

$C_{ab} = 7500$

$C_{ac} = 7500$

$C_{bc} = 5000$

$C_{a} = 5000$

$C_{b} = 5000$

$C_{c} = 5000$

Working of SHAP: An example

- To calculate expected marginal contribution of student A, we have 4 coalitions. These are marginal contributions.
- By adding a probabilistic weight to the marginal contribution with the weights here, we obtain the expected marginal contribution.
- Shapley value for ‘a’ is
  5000*(1/3)+2500*(1/6)+2500*(1/6)+5000*(1/3) = 5000

$$C_{abc} - C_{bc} = 5000$$
$$C_{ab} - C_{b} = 2500$$
$$C_{ac} - C_{c} = 2500$$
$$C_{a} - C_{0} = 5000$$

$$C_{abc} - C_{bc} = 5000$$
$$P(C_{abc} - C_{bc}) = 1/3$$

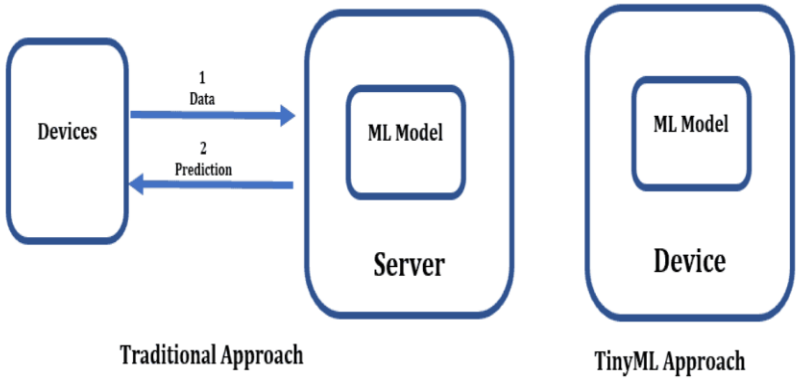
$$\left.\begin{array}{l} S_a S_b S_c \\ S_a S_c S_b \\ S_b S_a S_c \\ S_b S_c S_a * \\ S_c S_a S_b \\ S_c S_b S_a * \end{array}\right\} = \frac{2}{6} = \frac{1}{3}$$

Tiny Machine Learning (TinyML)

- Traditional Deep Learning models often encounter limitations such as large size, computational complexity, and high energy consumption. These factors make them less suitable for deployment on resource-constrained IoT end devices. A common approach to mitigate this issue is offloading complex computational tasks from sensor nodes to edge gateways.
- However, this method may not always be efficient, as the energy required to transmit data between devices can exceed the energy used to run an ML model on the device itself.
- In response to these challenges, TinyML has emerged as a specialized field focused on optimizing ML models for deployment on IoT end devices with limited resources.

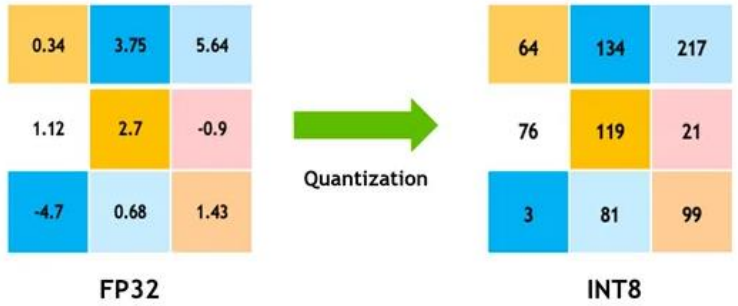
Advantages of TinyML

- **Bandwidth efficiency**: Enables local data processing and autonomous decision-making, eliminating the need to transmit collected data to the cloud.
- **Latency Reduction**: Minimizing dependence on external communication mitigates the problem of latency.
- **Energy efficiency**: Executing computations directly on devices uses less energy than sending data wirelessly.
- **Cost Savings**: TinyML contributes to cost savings in two distinct ways. First, by reducing data traffic and bandwidth requirements. Second, unlike more resource-intensive IoT devices that rely heavily on costly cloud resources, TinyML operates efficiently on low-cost, resource-limited hardware



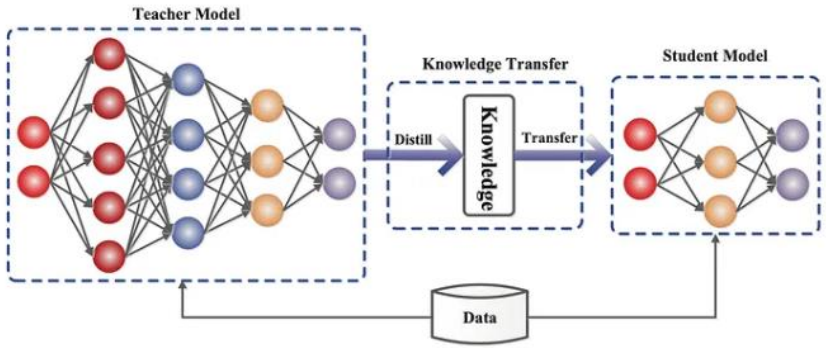
Quantization

- Quantization reduces model size and computational cost by converting high-precision floating-point numbers to lower-precision formats like 8-bit integers, enabling faster inference and deployment on resource-constrained devices.
- It’s like compressing a high-resolution image to a low-resolution one that still looks okay. It might lose some detail, but it's much smaller and quicker to load.
- Quantization error is present.



Knowledge Distillation

- Knowledge Distillation (KD) is a technique that facilitates the transfer of knowledge from a large, complex model — into a single, more compact model that is suitable for deployment in real-world scenarios.
- Teacher Model: Typically a deep, over-parameterized network with high predictive power but requiring substantial computation.
- Student Model: A smaller, more efficient model with fewer parameters and reduced complexity.



Knowledge Distillation

- Temperature Parameter : The soft labels are produced using the softmax function, which transforms the teacher model's logits (raw outputs before softmax) into probabilities. To smooth the probability distribution and allow the student to learn better, a temperature parameter T is introduced.
- For example, prob(Benign)=0.95, prob(Malign)=0.05. This is a very confident prediction by the Teacher. If T = 1, students copy the Teacher. After smoothing, prob(Benign)=0.75, prob(Malign)=0.25. Now student learns that it is most probably benign but there is a chance that is malign also. This extra information helps student learn intra class relationships better.

$$q_i = \frac{\exp(z_i^{(T)}/T)}{\sum_j \exp(z_j^{(T)}/T)}$$

where:

- $zi(T)$ is the logit for class i from the teacher model.
- T is the temperature hyperparameter.

$$L = \alpha \cdot L_{\text{CE}}(y, p) + (1 - \alpha) \cdot T^2 \cdot L_{\text{KL}}(q, p)$$

where:

- $L\text{CE}(y,p)$ is the cross-entropy loss between the true labels y and the student model's predictions p:

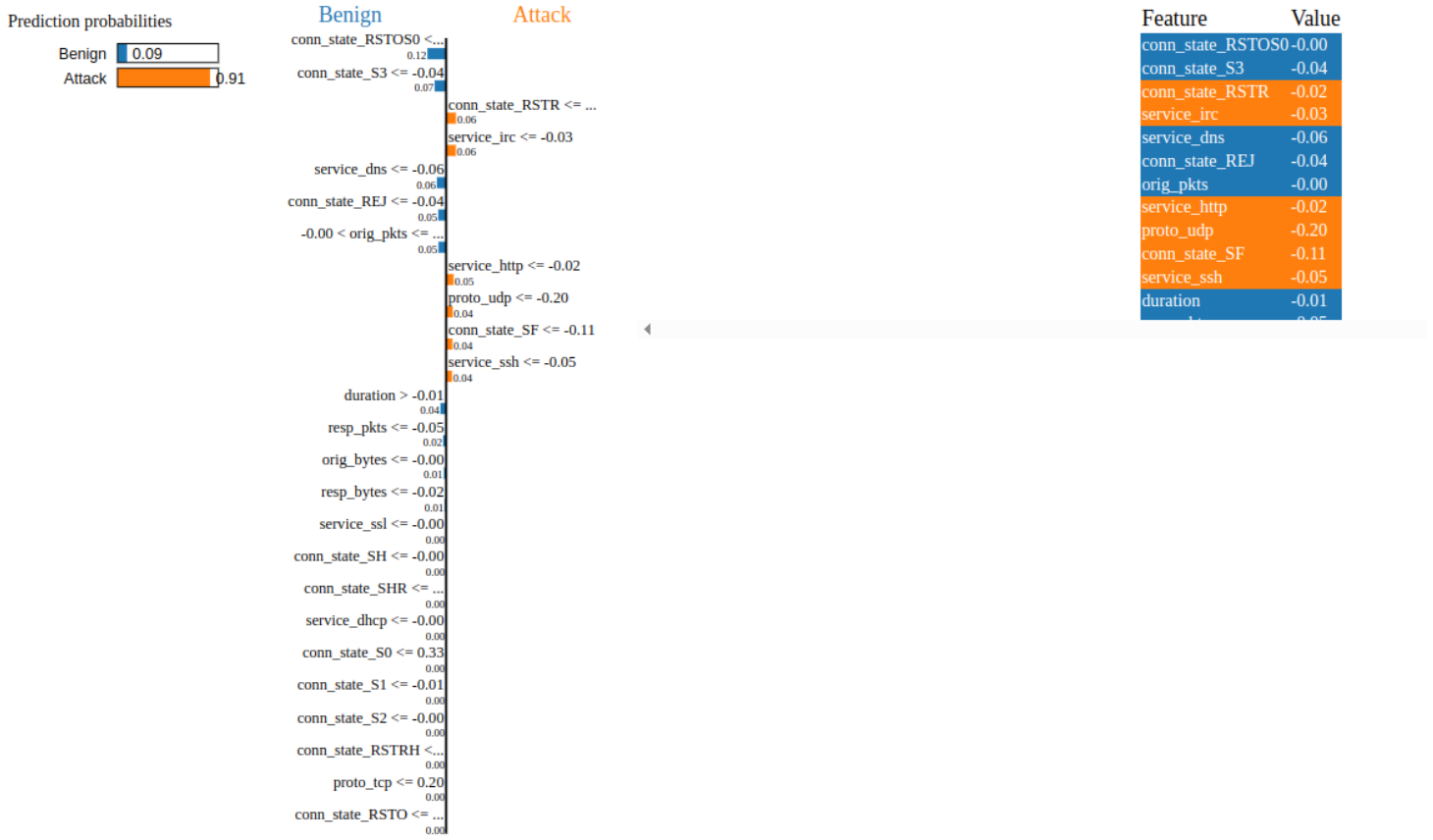
$$L_{\text{CE}}(y, p) = -\sum_i y_i \log(p_i)$$

- $L\text{KL}(q,p)$ is the Kullback-Leibler divergence between the teacher's soft predictions q and the student's soft predictions p:

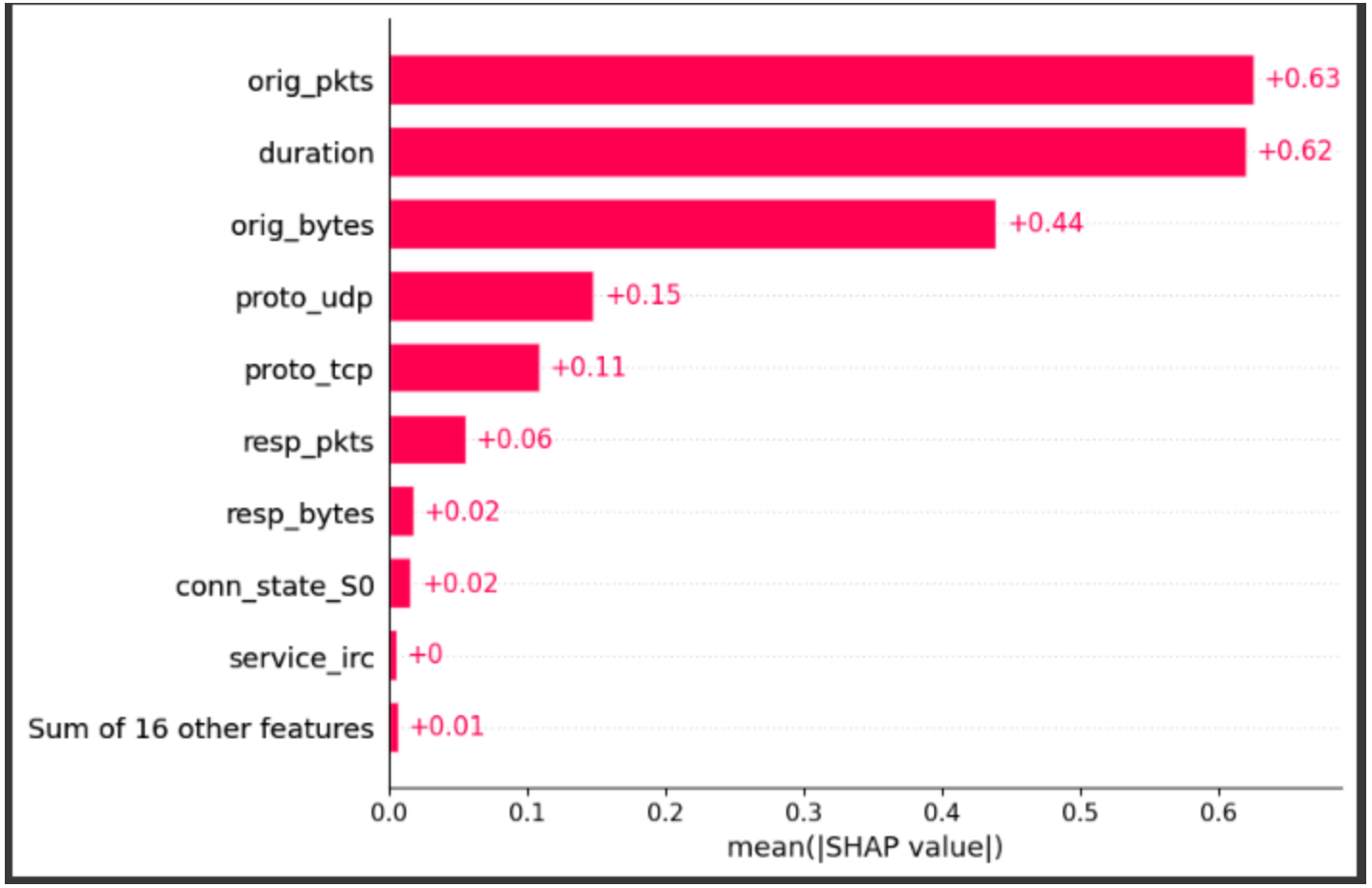
$$L_{\text{KL}}(q, p) = \sum_i q_i \log\left(\frac{q_i}{p_i}\right)$$

RESULTS

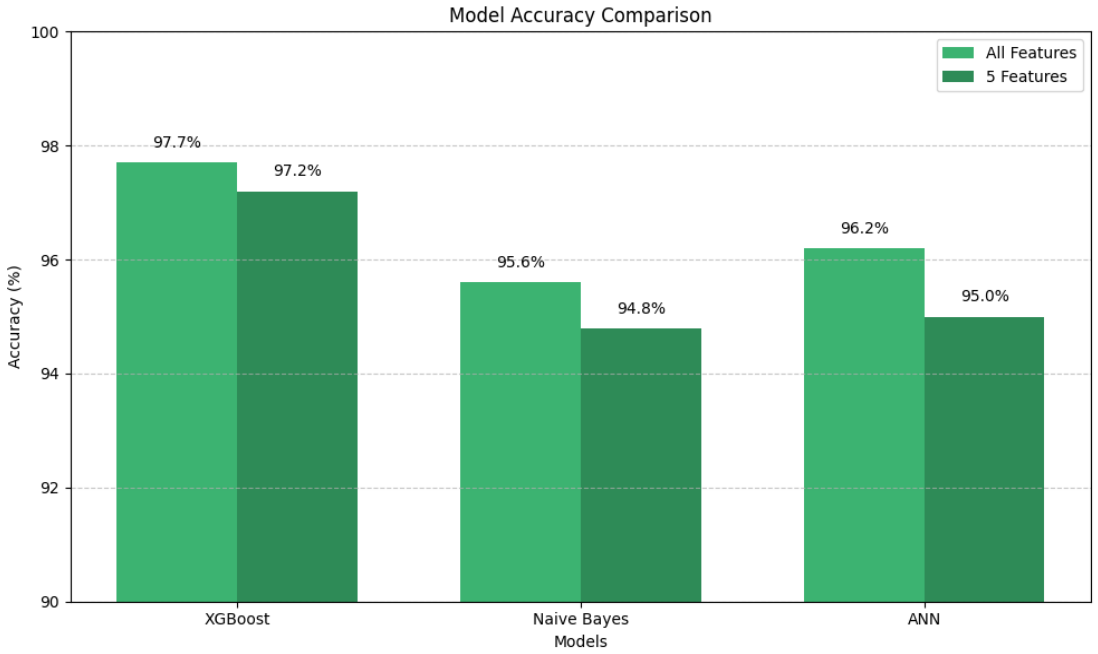
LIME Explanations

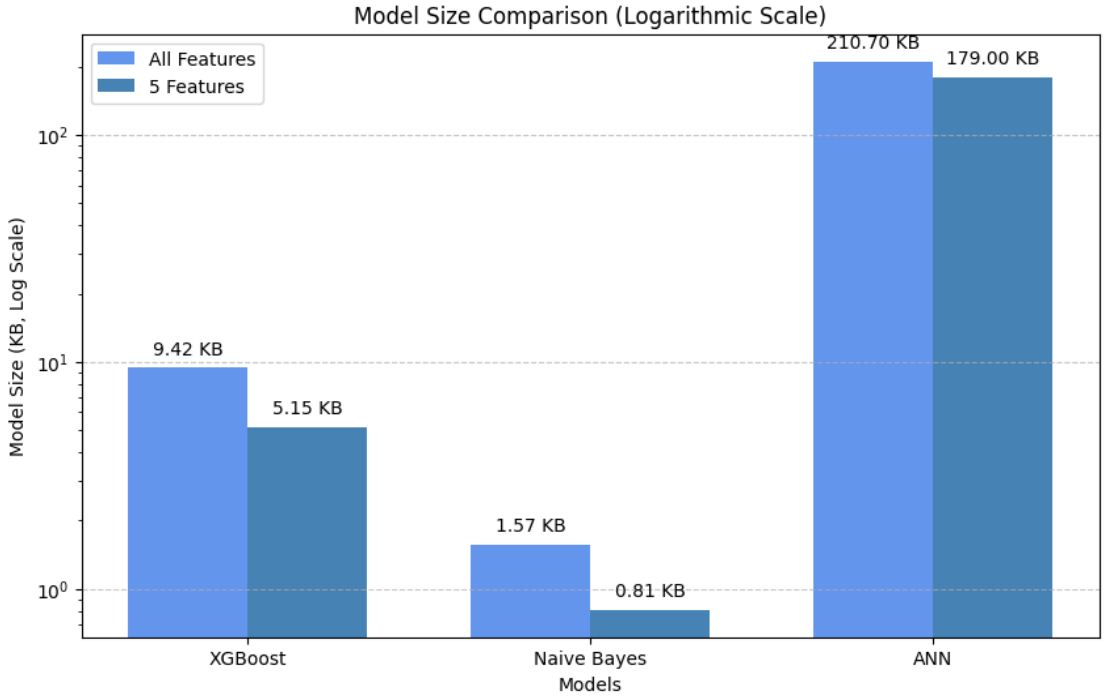


SHAP Explanations

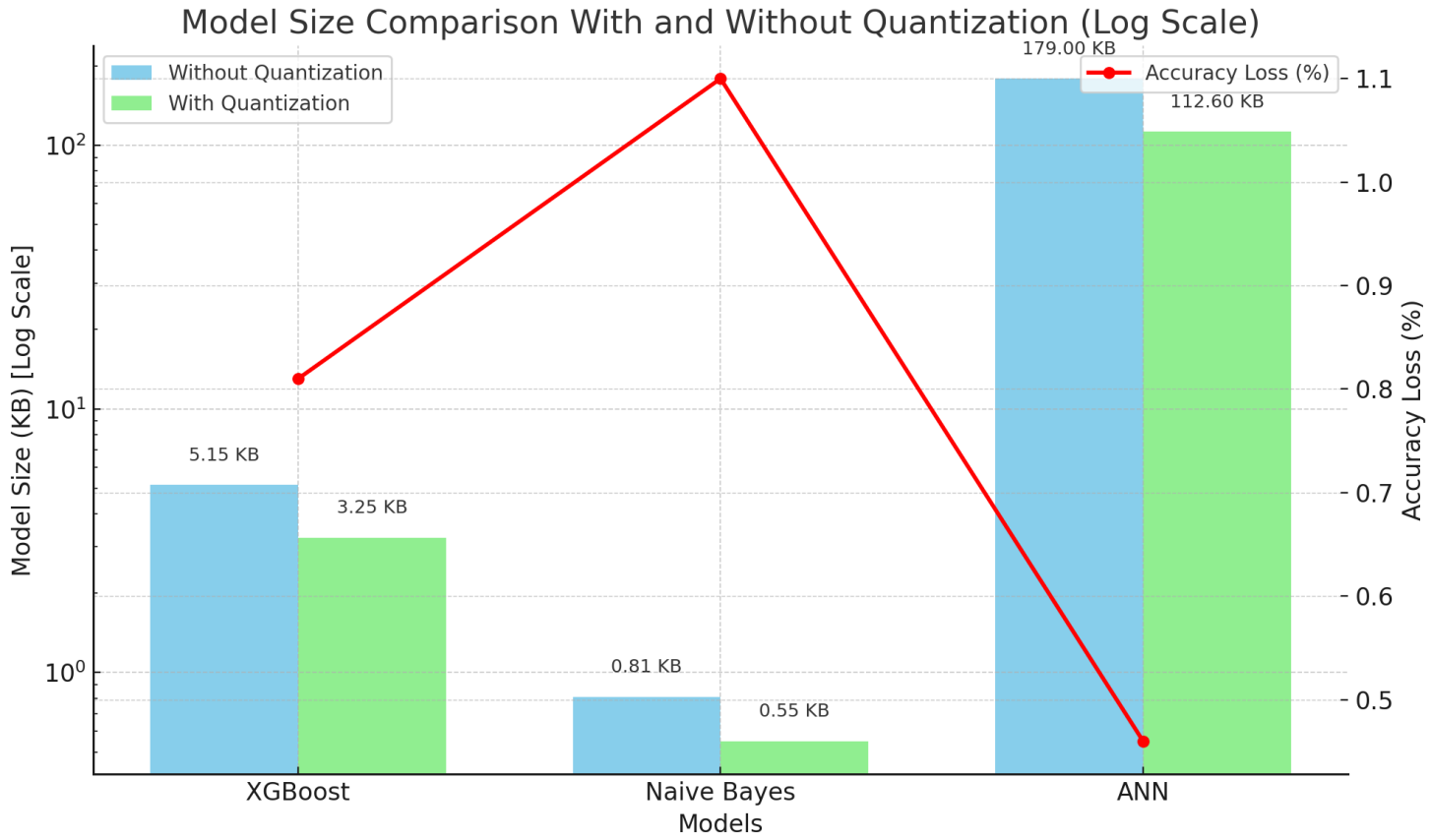


Effect of Feature Selection





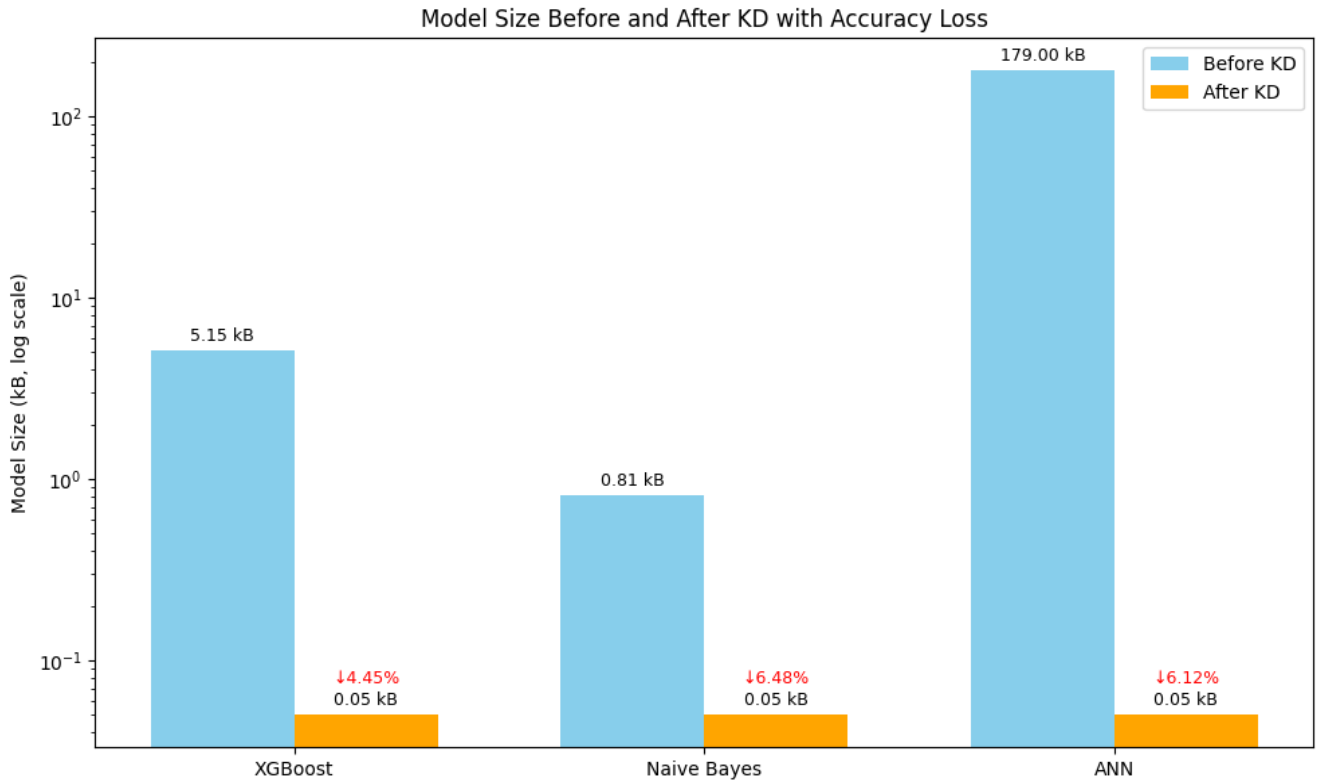
Effect of Quantization



- By using Quantization, we were able to achieve upto 1.4X of compactness for models. But this still is not deployment ready as we generally use Deep Learning Models of size of several MBs.

Effect of Knowledge Distillation

- We distilled the knowledge from XGB, NB and ANN to a more simpler Logistic Regression Model.
- Effectively number of parameters were brought down to 6 from 48800 (for ANN) with 6.12% reduction in accuracy.



Experimental Setup

- In the base paper, models were compared on Cloud, Edge Device, IoT device, IoT end device.
- We have simulated the working of models on Raspberry Pi 4 and Arduino Nano 33 using **Edge Impulse** platform.

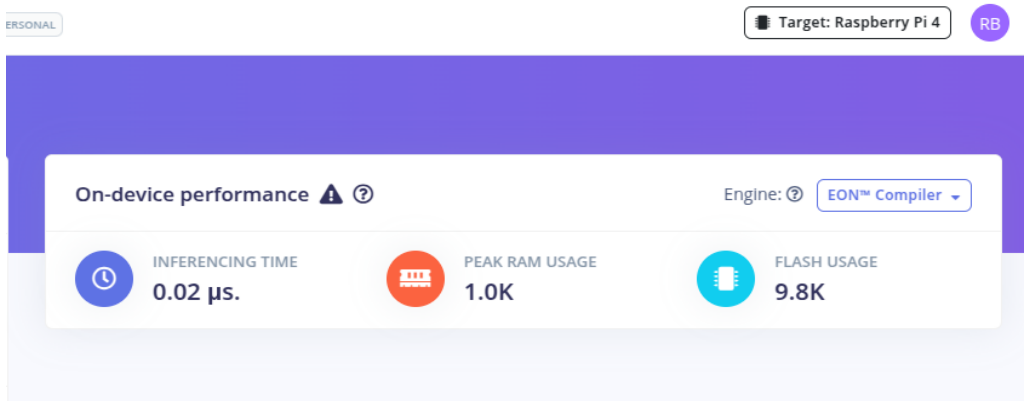
	Base Paper	Our Work
Cloud	Azure Cloud (32GB RAM)	Google Colab (12GB RAM)
Edge	Dell Laptop (i7-9750H, 16GB RAM)	HP Laptop (i5, 16GB RAM)
IoT Device	Raspberry Pi 4 (8GB RAM)	Raspberry Pi 4 (8GB RAM)
IoT End Device	STM32F412 (256KB RAM)	Arduino Nano 33 (256KB RAM)

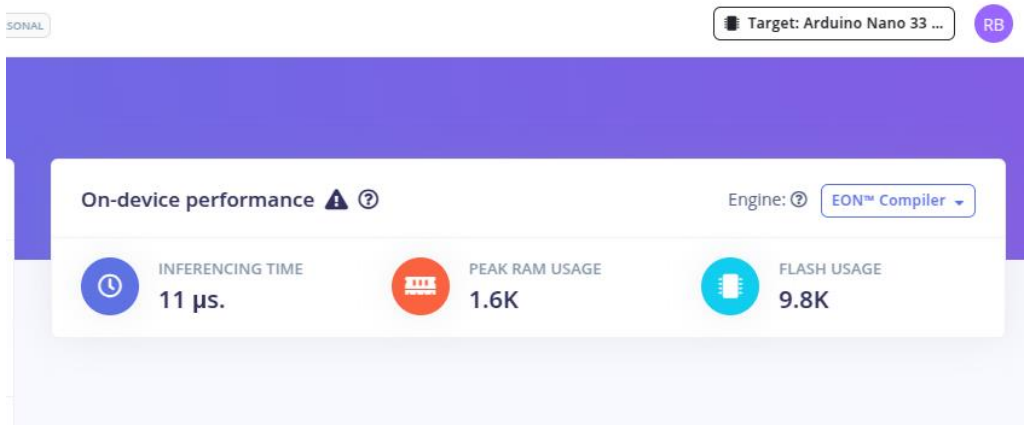
Training Time

- As our cloud have less RAM (12 GB) compared to Edge (Laptop) (16GB), results on the edge are slightly better on Edge than on Cloud

Model	Cloud (s)	Edge (s)
ANN	893.43	1847.94
ANN (Ours)	1324.28	1178.85
RF	31.75	166.83
LR	16.54	19.06
k-NN	8.16	17.61
DT	8.98	17.46
XGB (Ours)	1.2279	0.8395
NB	0.67	0.69
NB (Ours)	0.2071	0.1189

Inference Time



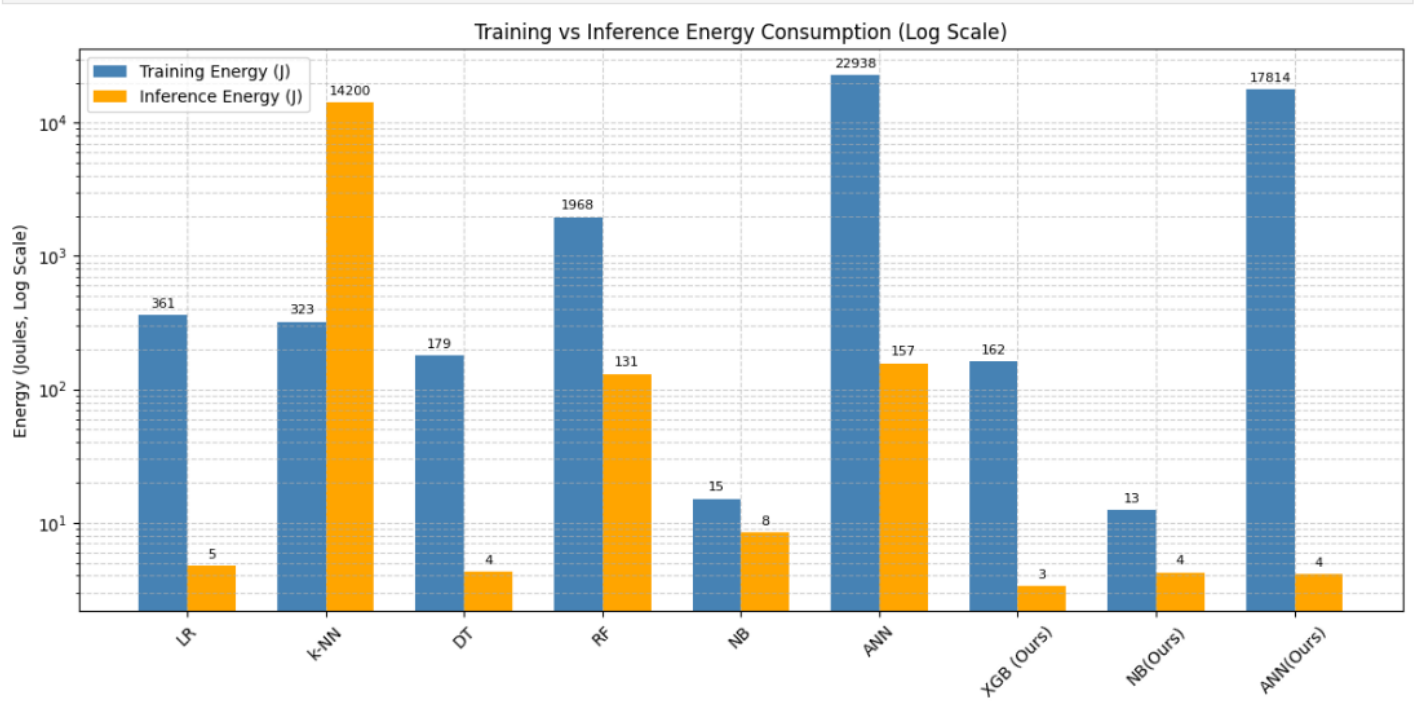


Model	Cloud (µs)	Edge (µs)	IoT Device (µs)	IoT End Device (µs)
LR	0.04	0.04	0.04	73.06
k-NN	513.31	530.14	2078.01	nan
DT	0.07	0.09	0.34	65.06
RF	0.62	1.88	10.1	2088
NB	0.19	0.34	2.13	602
ANN	1.28	2.97	5.58	nan
XGB (Ours)	0.019	0.017	0.024	11.6
NB (Ours)	0.014	0.018	0.02	11
ANN (Ours)	0.014	0.017	0.0245	11.21

Here XGB (Ours) means the student model trained from XGB teacher.

Energy Consumption

- We have used Intel's RAPL (Running Average Power Limit) to measure energy consumed in Joules.
- Checkpoints were made before and after training/inference and difference was measured. We ensured that no background apps are running except Jupyter for proper measurement.



Future Scope and Conclusion

- Explore techniques to ensure minimal reduction in accuracy while maintaining the compactness of the model.
- Implement incremental learning techniques to adapt to new types of threats dynamically without re-training on a server.

THANK YOU