University of Zurich^UZH

MASTER THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

Dataset Generation for ML Personal Tracker Detection

Ramize Abdili
Zürich, Switzerland
Student ID: 17-731-076

Supervisor: Katharina Müller, Dr. Bruno Rodrigues, and Prof. Dr. Burkhard Stiller
Date of Submission: June 15, 2024

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

ifi

Master Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
URL: http://www.csg.uzh.ch/

Declaration of Independence

I hereby declare that I have composed this work independently and without the use of any aids other than those declared (including generative AI such as ChatGPT). I am aware that I take full responsibility for the scientific character of the submitted text myself, even if AI aids were used and declared (after written confirmation by the supervising professor). All passages taken verbatim or in sense from published or unpublished writings are identified as such. The work has not yet been submitted in the same or similar form or in excerpts as part of another examination.

Zürich, 15.06.2024

[illegible]

Signature of student

Abstract

The majority of Internet of Things (IoT) devices are Bluetooth Low Energy (BLE) devices, which can be found every day in a variety of settings, including smart homes, the workplace, sports, entertainment, and medicine. The number of BLE devices is growing rapidly due to their low power consumption. However, BLE devices have the disadvantage that the transmitted data contains personal information and can be tracked. Therefore, the protection of personal information has become very important. This can be achieved by classifying the BLE devices into device types to identify unauthorized devices. This thesis focuses on creating a dataset by passively sniffing BLE packets to extract features that can be used to train Machine Learning (ML) models to identify and classify BLE devices. The ML models used in this thesis are the Random Forest (RF) classifier and the Multi-Layer Perceptron (MLP) model. The highest accuracy is achieved by the RF classifier at 99.98%.

Abstrakt

Die meisten Geräte des Internet of Things (IoT) sind Bluetooth Low Energy (BLE)-Geräte, die tagtäglich in einer Vielzahl von Umgebungen anzutreffen sind, zum Beispiel in Smart Home, am Arbeitsplatz, im Sport, in der Unterhaltung und in der Medizin. Die Zahl der BLE-Geräte nimmt aufgrund ihres geringen Stromverbrauchs rasch zu. Darüber hinaus haben BLE-Geräte den Nachteil, dass die übertragenen Daten persönliche Informationen enthalten und nachverfolgt werden können. Daher ist der Schutz persönlicher Informationen sehr wichtig geworden. Dies kann durch eine Klassifizierung der BLE-Geräte in Gerätetypen erreicht werden, um nicht autorisierte Geräte zu identifizieren. Diese Arbeit konzentriert sich auf die Erstellung eines Datensatzes durch passives Schnüffeln von BLE-Paketen, um Merkmale zu extrahieren, die zum Trainieren von Machine Learning (ML)-Modellen verwendet werden können, um BLE-Geräte zu identifizieren und zu klassifizieren. Die in dieser Arbeit verwendeten ML-Modelle sind der Random Forest (RF) classifier und das Multi-Layer Perceptron (MLP) Modell. Die höchste Genauigkeit erreicht der RF classifier mit 99,98%.

Acknowledgments

It is said far too seldom: I would like to express my deepest gratitude to Prof. Dr. Burkhard Stiller and Katharina O. E. Müller with whom it was a pleasure to work. Thank you for the great collaboration, interesting discussions, and insightful advice over the past six months. I really appreciate your competence and honesty very much, and I was always happy to have a look at my thesis with you.

Many thanks also go to the Communication Systems Group of the Department of Informatics of the University of Zurich for giving me the opportunity to work on such an interesting topic.

As well as my family, friends, and all who have supported and encouraged me during this time, I would like to express my great praise and thanks.

Contents

Declaration of Independence **i**

Abstract **iii**

Abstrakt **v**

Acknowledgments **vii**

1 Introduction **1**

1.1 Motivation 2

1.2 Description of Work 3

1.3 Thesis Outline 3

2 Related Work **5**

2.1 Research Gap 9

3 Background Knowledge **13**

3.1 Introduction of Bluetooth Low Energy Devices 13

3.2 BLE Protocol Stack 13

3.2.1 Host 14

3.2.2 Controller 15

3.3 BLE LL Packet Format 17

3.4 BLE Evolution 19

3.5 BLE Attacks 19

3.5.1 Sniffing Attack 20
3.5.2 Man in the Middle Attack 20
3.5.3 Replay Attack 20
3.5.4 Spoofing 21
3.5.5 Jamming 22
3.5.6 Session Hijacking 22
3.5.7 Packet Injection 22
3.5.8 Eavesdropping 23
3.6 Machine Learning 23
3.6.1 Decision Trees 24
3.6.2 Random Forest 24
3.6.3 Logistic Regression 25
3.6.4 Naive Bayes 25
3.6.5 K-Nearest Neighbour 26
3.6.6 Support Vector Machines 26
3.6.7 Neural Network 26

4 Methodology 29
4.1 Device Collection 30
4.2 Data Collection via Passive Sniffing with nRF Sniffer Tool 30
4.2.1 Initial Capturing of BLE Packets in Public Places 31
4.2.2 Capturing BLE Packets from Known Device Types 31
4.3 Dataset Labeling 32
4.4 Feature Extraction 33
4.4.1 Direct Extraction of Feature from Sniffed BLE Packets . . . 33
4.4.2 Derived Feature Extraction from Sniffed BLE Packets 33
4.5 Data Preprocessing 35
4.5.1 Data Cleaning 35

4.5.2 Removal of Unique Identifiers 35
4.5.3 Data Transformation 35
4.5.4 Data Balancing 36
4.6 Feature Selection 36
4.7 Data Splitting 38
4.7.1 Train-Test Splitting 38
4.7.2 Train-Validation Splitting 39
4.8 Device Classification via ML Approach 39
4.8.1 Training of ML Models 39
4.8.2 Evaluation Metrics 40
4.8.3 Random Forest Classifier 41
4.8.4 Multi-Layer Perceptron 42
4.9 Comparing ML Models 44

5 Results and Evaluation 47

5.1 Device Collection 47
5.2 Data Collection via Passive Sniffing with nRF Tool 48
5.2.1 Initial Capturing of BLE Packets in Public Places 48
5.2.2 Capturing of BLE Packets from Known Device Types 52
5.3 Dataset Labeling 54
5.4 Feature Extraction 57
5.4.1 Direct Extraction of Feature from Sniffed BLE Packets . . . 58
5.4.2 Derived Feature Extraction from Sniffed BLE Packets 64
5.5 Data Preprocessing 73
5.5.1 Data Cleaning and Removal of Unique Identifiers 73
5.5.2 Data Transformation 74
5.6 Feature Selection 75
5.7 Data Splitting 79

5.7.1 80/20 Data Splitting Method 79
5.7.2 Alternative Data Splitting Approach 79
5.8 Device Classification via ML Approach 79
5.8.1 Random Forest Classifier 80
5.8.2 Multi-Layer Perceptron Model 86
5.9 Comparing ML Models 93
5.10 Summary of Limitations 94

6 Conclusion and Future Work 97
6.1 Conclusion 97
6.2 Future Work 98

Bibliography 99

Abbreviations 107

Glossary 109

List of Figures 109

List of Tables 114

A Installation Guidelines 117
A.1 nRF Sniffer Tool 117
A.2 Running the Application 117

B Full List of Related Work 119

C Direct Feature Extraction 123

D Derived Feature Extraction 125

E Results of Direct Feature Extraction 127

F Results of Derived Feature Extraction 131

Chapter 1

Introduction

We are currently surrounded by a large number of IoT devices. Many of these devices are BLE devices and are controlled via Bluetooth technology [11]. These devices are often used in everyday life, such as in the kitchen, for sports, entertainment, gaming, in the office, in healthcare, in the security department, in smart homes, in the infrastructure, and in many other areas [9, 17]. BLE devices in e-healthcare measure the heart rate or blood flow [11]. In smart home environments, BLE devices include smart lights, smart thermostats, smart locks, or smoke detectors [24]. In sports, they are used for fitness tracking via smartwatches that measure the number of steps or heart rate [24]. Furthermore, the position of objects can be tracked using BLE devices such as AirTags. Attaching AirTags to keys or items of luggage, they can be quickly localized and found in the event of loss [24].

Billions of BLE devices are developed by the Bluetooth Special Interest Group (SIG) [9]. The Bluetooth specification includes Bluetooth Classic and BLE [22]. BLE devices are first introduced with version 4.0 of the Bluetooth Standard [12]. BLE differs from Bluetooth Classic by its link layer packet format and low power consumption capacity [12]. Thanks to their fast communication, they are widely used and are an essential part of our daily lives such as smartphones. BLE devices discover the connection via three different channels, making their connection much faster than that of Bluetooth Classic devices [10]. In addition, BLE devices are used for communication over short distances and for transmitting small amounts of data [24]. BLE is therefore a low power technology and is integrated into small devices with low-charge capacity and limited power supply, such as batteries [9, 11]. The low power consumption of BLE devices results from the fact that the screens are switched off as much as possible and the small amount of data transmission at low transmission speeds [24, 10]. Due to their low power consumption, short-range and fast communication, they are becoming increasingly popular and the number of BLE devices is growing rapidly [11]. Another important factor for the increasing number of BLE devices is the low-cost development compared to other technologies [24]. The biggest advantage of BLE is that it is integrated into smartphones [10]. With over 4.7 billion BLE devices developed in 2018, BLE devices are the most widely used wireless communication technology [12]. Despite their many advantages, BLE devices also have some disadvantages such as their limited data throughput [24]. Only very little data can be transmitted per defined period of time [24]. Furthermore, their range is limited as they are developed for applications with a short range [10]. With a range of only around 2.4 GHz in the ISM spectrum, metal objects or walls can block data transmission [24, 10].

To establish connectivity, BLE devices constantly transmit and exchange data [1, 2, 4]. They send data to another device that acts as a receiver [11]. The transmitted data is so-called Advertising Data (AdvData), which may include the user's location or personal information such as email addresses or usernames linked to an Apple ID on a smartphone [1, 11]. Unfortunately, these data exchanges also provide opportunities for external attackers to exploit this personal information for their purposes [1, 2, 4]. There are numerous security and privacy threats caused by having access to AdvData [23]. For example, DDoS attacks are one such threat, where the attackers obtain personal information or spy on the owner of the device through the camera [17]. By determining the location of a fitness tracker user, the attacker can locate the user [12, 21]. This private and personal information can be used for malicious activities and cause privacy threats by revealing a device's information [14, 23]. By hacking a medical device, an attacker could learn the user's health status and use this information to their own advantage [23].

Therefore, the protection of personal data has become very important in our increasingly interconnected world through BLE devices [2]. As the use of these devices increases, so does the number of attacks [20]. As most BLE devices are used in smart home environments, there is a high demand for the classification of BLE devices to distinguish between authorized and unauthorized devices [14]. Devices that should not send or receive BLE packets in an environment are so-called unauthorized or malicious devices and must be detected. Once an attacker device hacks into an environment and gains access to all connected devices, it can gain hidden access to all information [21]. Electronic locks, alarm systems, or medical devices that are controlled via BLE devices such as smartphones or laptops are also susceptible to security vulnerabilities and can be attacked [22].

With the rapid increase in devices, it is becoming increasingly difficult to track which devices are sending or receiving data to or from our own devices. Therefore, identifying and classifying such unauthorized devices is important [14]. To achieve better security for BLE devices and to overcome malicious attacks, identification of devices is needed [15]. If the types of devices connected in a smart home environment are known, these devices can be better managed and unknown devices can be more easily detected [16]. By identifying unknown devices, they can be blocked by the environment to prevent data exchange.

1.1 Motivation

Since most BLE devices are smart home devices that contain personal information, malicious attacks can have serious consequences. Due to security and privacy issues as well as the high need for personal data protection, the motivation of this thesis is to classify BLE devices so that unauthorized devices can be identified. This ensures a more secure environment for the users of these devices. As many attacks are passive and difficult to detect, the goal of this thesis is to develop a classification method. Within the scope of this thesis work, the transmitted packet data from BLE devices is collected and analyzed. The features of this data are extracted and ML models are developed that enable the identification and classification of BLE devices based on their features.

1.2 Description of Work

This thesis aims to generate a large dataset by collecting BLE packets through passive sniffing to determine if there are patterns in the data that could identify specific BLE devices, such as AirTags. ML approaches analyze the features extracted from the collected BLE packets and predict their device type.

In summary, the main objectives of this thesis are:

- Capturing of BLE packets through sniffing over an extended period of time
- Preprocessing and creation of a large dataset for analysis and pattern recognition
- Extracting features from the dataset that are characteristics of a device type
- Evaluating and developing ML models to classify and identify BLE devices based on their characteristics

1.3 Thesis Outline

Chapter 2 provides an overview of related work on BLE data collection, BLE device identification, and classification. Chapter 3 provides a brief background on BLE devices and their potential attacks. Chapter 4 explains the proposed methodology of data collection using a sniffing tool, extracting features, and designing two ML approaches to classify unknown devices based on the extracted features. Chapter 5 presents the results, evaluation, and limitations of the proposed methodology. Finally, Chapter 6 concludes this thesis with the conclusion and provides some potential future research approaches.

Chapter 2

Related Work

This chapter provides an overview of existing research related to this thesis. Most of the current research focuses on classifying IoT devices rather than BLE devices. There is limited research on classifying BLE devices. However, since BLE devices are a subset of IoT devices, the research on IoT device classification is closely related to this thesis. All related work is summarized in Table 2.1 by their number of devices, types of devices, number of features, if there are multiple models, and the model with the highest accuracy. The full list is in Appendix B. After introducing the related work, the research gap between existing literature and the objective of this thesis is explained.

Since the security of IoT devices plays an important role in ensuring that the user's data is protected, [3] developed an automatic IoT device classification. Their classifier can identify malicious devices. The traffic flows of each device packet are analyzed and important features such as the packet length, the timestamp, the protocol are extracted. The classifier is a Long Short-Term Memory Convolutional Neural Network (LSTM-CNN) model trained with the extracted features. The goal is to find patterns in the features to predict the type of an unseen device. Their dataset consisted of 21 devices categorized into 4 types of devices. Their model had a performance of 74.8%.

To detect unauthorized devices, [4] developed an intrusion detection method. Their dataset consists of six ZigBee devices. A CNN was used to extract unique features for each device from the physical layer of the packets. Afterward, the features were clustered. The training dataset consists of features from authorized devices. Their CNN model was an unsupervised classification model and did classification based on the clustering approach. Their results have shown that detecting unauthorized devices enhances the security of the devices.

[5] used Radio Frequency (RF) fingerprinting to identify Wi-Fi IoT devices by training an ML algorithm with their collected data. Similar to [4], they also extracted features from the physical layer of the devices. These features form the fingerprints and based on them, the model analyzes the traffic behavior of the devices to classify and identify unknown devices. The results have shown that

the fingerprints are similar to the same device but different from the fingerprints of other devices and that the Selective Kernel Network (SKNET) achieved the highest accuracy.

[13] also developed an ML algorithm to identify IoT devices. Their dataset is labeled and consists of 9 IoT devices. The ML algorithm is a supervised binary classifier that can group devices into IoT and non-IoT devices. Furthermore, it can group IoT devices into their device category. The network traffic of devices has been analyzed and network features were extracted from the Transmission Control Protocol (TCP) packets. These features include the IP address and the port number of a device, and they are unique. By training the classifier with these features, it can classify feature vectors into their device type. During training, the ML algorithm was optimized and achieved an accuracy of 99.28%.

[14] analyzed the communication traffic of the Wi-Fi devices to achieve a device classification. Their dataset consists of Wi-Fi devices in the smart home environment. They extracted features from the traffic of the devices. By using the dimensionality reduction and the Information Gain (IG) method to keep just the most important features, they have, in total, 13 extracted features. These features are the input for training the Logistic Regression (LR) model. The model has shown an accuracy of 99.79% in classifying devices into their type.

[15] developed an automated classification of IoT devices. Their developed classifier is a Decision Tree (DT) model, which classifies their 23 IoT device into device types by analyzing the selected features. These features are selected by a Genetic Algorithm (GA). Using just half of the features from the packet header is sufficient to classify the devices. Furthermore, the results showed that the accuracy was higher when using just half of the features rather than all features.

[16] developed a DT model too. The model aims to classify newly connected devices in smart home environments. Their dataset is labeled and they extracted features from the traffic flow and the packet payloads of the packets. Features such as the average packet length, inter-arrival times, flow size, and protocols were used to train the binary DT classifier. The model achieved an accuracy of 97% in automatically identifying newly connected devices. This has shown that home security was enhanced by correctly identifying smart home devices.

Due to vulnerabilities of IoT devices to attacks, [17] developed a stacked Autoencoder model. This model can learn the distribution of packet flows of devices to automatically extract features from the traffic network. These features are then clustered and are the input for training the model. By calculating the similarity between two distributions of flow packets and by comparing them, the Autoencoder model assigns the device to one cluster class and classifies it.

[18] has done similar work. They also extracted features from the traffic flow to identify devices and detect malicious ones. These features are the packets' direction, size, timestamp, and transport protocol and are the input for training the classifier. They build two classifiers: an RF and a DT classifier. The goal is to

detect malicious devices and to define the type of attack. The results have shown that RF achieves the best results.

Because of the increasing attacks on IoT devices, [?] developed a deep learning model, namely a multiclass classifier, to identify known and unknown devices. Unknown devices are those not on their so-called white list and should be unauthorized in the network traffic. Training a fully connected network model with 10 different IoT devices, the model achieves an accuracy of over 99% to detect and identify unauthorized devices correctly. Similar [21] had also a so-called white list. Devices not on that list should not be connected to the network and should be detected. By detecting them, the network flow is controlled and the sensitive data of devices from the white list can be protected. The extracted features from the network traffic are the input for training a supervised ML algorithm, namely an RF classifier. By learning the behavior of the devices from the white list, unauthorized IoT devices can be identified. Their white list consists of 17 IoT devices categorized into 9 types. The RF classifier should predict the device type by comparing the output with a given threshold. If the output value is above the threshold, the device is classified as unauthorized. The results have shown an accuracy of 96% to correctly detect unauthorized devices and an accuracy of 99% to correctly predict the type of devices from the white list.

To achieve autonomous IoT device type identification, [44] developed an unsupervised ML model focusing on periodic communication patterns. Their dataset includes 33 IoT devices. The model clusters communication fingerprints based on policy sets describing the network behavior without requiring a labeled dataset. Key features in these fingerprints include periodic flow count, protocol layer, flow type, port changes, period accuracy, duration, and stability. The model is a k-Nearest Neighbor (KNN) classifier. The goal of the classifier is to match the fingerprints with device types.

[45] collected BLE packets and extracted fingerprints from their Link Layer (LL). Fingerprints are unique patterns in the traffic flow of a device. They developed an MLP model that predicts the type of BLE devices based on differences in the LL. Their dataset consists of 7 different types with a total of 23 BLE devices. Their model is trained on fingerprint vectors focusing on characteristics such as packet headers, AdvData types, and specific fields like service UUIDs and manufacturer identifiers. After removing explicit identifiers and filtering relevant packet types the vectors are zero-padded and normalized. The MLP model is trained on these vectors to classify the BLE device types. They achieved an accuracy of 99.8% in correctly identifying BLE device types.

The goal of [47] is to classify Wi-Fi devices such as smartphones and laptops. They extracted features from the sniffed network communication. The RF classifier has shown the best performance among other ML models in correctly predicting the device type. The classifier is trained on features and their dataset includes 279 devices with known device types. One important feature is the Inter-Probe Period (IPP) indicating the frequency of transmitting data. Laptops have shown a higher frequency than smartphones. Another important feature is the standard deviation of the Received Signal Strength (RSS) which reflects the mobility of devices. Smartphones have shown a higher standard deviation of the RSS since they are more mobile than laptops.

Instead of classifying smartphones and laptops, [48, 49] focused on classifying devices as hubs, cameras, switches, healthcare devices, light bulbs, electronics, smartwatches, or sensors. They analyzed the packet header and extracted features such as the domain name, packet sent rate, packet length, and the number of protocol types. Their dataset has 12 devices categorized into 7 categories. They developed three different ML models: LR, RF, and Support Vector Machine (SVM). Results have shown that hubs have more temporary packet transmission compared to cameras.

To predict the device type, [50] developed also three ML models: RF, DT, and SVM. Their dataset includes 22 IoT devices categorized into 11 types like IP cameras, Amazon smart speakers, smart printers, and smartphones. By looking at the MAC address of the devices, their type could be inferred and thus they could label the dataset. The extracted features from the traffic include type and sub-type, size, direction, interarrival time, and rate of frames. These features are the input for training the classifiers.

Another important feature to identify devices is extracted by [51]. This feature is the Received Signal Strength Indication (RSSI) and is extracted from BLE packets. Since RSSI depends on the distance between two communicating devices, it can be used to calculate the distance between them. There are two types of vectors generated the so-called matched and non-matched vector. If two devices have the same device type, their features are marked as matched, otherwise, they are marked as non-matched. These two types of vectors train the classifier to predict whether two devices share the same device type. Furthermore, the trained classifier can also predict the device type of a given device.

[52] collected IoT devices in the smart home environment and labeled the dataset. They extracted features such as the TCP window size, entropy, and payload lengths. They defined a session as a series of packets with identical source addresses and computed metrics such as the total session packets and packets per session for each device. Packet header features consist of binary TCP features, while payload features encompass payload entropy, TCP payload length, and TCP window size. Small devices like light bulbs often have small window sizes, while more powerful devices such as video cameras tend to have variable and larger window sizes. They developed four different classifiers comparing the features of a given device with the features of devices from the dataset. Results have shown that the Gradient Boosting (GB) classifier has the best performance of detecting similar device types over KNNs, DT, and Majority voting.

To secure the communication for the BLE devices and to countermeasure spoofing attacks, [56] sniffed data traffic of BLE packets with the Ubertooth tool and extracted features from the LL and ATT/GATT service layer. Feature extraction involves analyzing advertising event intervals for transmission frequency, advertising channel sequences across three channels, and the probability distribution of advertising delays. The extracted features are the number of packets sent, the average throughput, and the burst rate. Fingerprints of devices are created based on these features and are stored in a database marked as a whitelist or blacklist device. They developed a RF model which is a supervised ML model. Results have shown that different device types have unique packet sequences indicating that the packet sequence feature is a characteristic of a device type.

[57] collected the traffic of BLE and Wi-Fi devices via the BLE sniffer and Wi-Fi adapter. Their dataset consists of 6 device types including door sensors, locks, temperature sensors, smart bulbs, cameras, and smart plugs. They extracted features such as the packet count, packet length, LL header length, Protocol Data Unit (PDU) type, and RF channel number. The device name and the Advertising Address (AdvA) are used to identify the type of device to label the dataset but are not used as features. After extracting features, they developed three classification models: KNN, Linear Discriminant Analysis (LDA), and RF. The models are trained on their extracted features and aim to identify the device type. The KNN classifier achieved the best performance. Results have shown that a device with a packet length greater than 80 bytes is a locking device, with a packet count over 40 packets/sec is a sensor device, with a packet count between 20-40 packets/sec is a locking device, and with the PDU type of scan request (SCAN REQ), advertising direct indications (ADV DIRECT IND), or connection requests (CONNECT REQ) is a door device. They calculated the feature's importance score and took the three most important features with the highest values. The three most important features are the packet length, LL header length, and packet count. The model is retrained using just these three features as input. Here the results have shown that the door and temperature sensor devices were misclassified because of having similar packet lengths, LL header lengths, and packet counts.

Unlike the others who have a labeled dataset, [55] have an unlabeled dataset and used deep reinforcement learning, namely an autoencoder model. The goal is to predict the device type based on the BLE RSSI. They clustered the feature vector in a cluster and assumed that if two feature vectors of two devices are close to each other and in the same cluster, then these devices should have the same device type. Labeling is done through user feedback and the reward function of reinforcement learning. The autoencoder model learns from the rewards and policies given to it and takes the optimal action to predict the device type.

2.1 Research Gap

As already mentioned, most of the existing research is based on IoT device classification, and very little research is done with a specific focus on BLE devices such as [45, 51, 55, 56, 57]. Therefore, there is little research gap in current research with a focus on BLE device classification. Since BLE devices are affected by attacks and most BLE devices appear in smart home environments, there is a high need to protect personal information. To address this issue, this thesis's goal is to classify BLE devices. By classifying BLE devices into their device types once, devices that

Paper	No. of Devices	Type of Devices	No. of Features	Mulitple Models	Model
[13]	9	PCs and Smartphones	4	no	Multi-stage Meta Classifier
[14]	41	4 types of Smart Home Environment	13	yes	LR, Logiboost
[15]	23	N/A	212	no	DT
[16]	6	Camera, Plug, Switch, Smart Home	6	yes	DT, Naive Bayes (NB), LR, SVM and RF
[17]	N/A	Cameras, Switches, Triggers, Hubs, Air Quality Sensors, Healthcare devices, Light Bulbs, Laptops, Mobile Phones, and Tablets	3	yes	NB, LSTM-Autoencoder
[18]	7	2 types	6	yes	RF, DT, RNN, ResNet, and ConvNet
[21]	17	9 types	300	no	RF
[58]	4	Security Camera, Motion Sensor, Smart Bulb, and Smart Plug	38	yes	RF, DT, SVM, KNN, Artificial Neural Network (ANN) and NB
[59]	N/A	N/A	37	yes	DT, SVM, NB, KNN, RF, GB
[45]	15	Smart Bands, Smart Watches, Smart Oximeters, Smart Humidifiers, Headphones, Smartphones, and PCs	10	no	MLP
[47]	N/A	Smartphones and Laptops	6	yes	RF, NB, SVM, DT
[50]	22	10 types	3	yes	RF, DT and SVM
[52]	14	7 types	6	yes	KNN, DT, GB, and Majority Voting
[48]	12	Hubs, Cameras, Switches, Triggers, Air Quality Sensors, Healthcare devices, Light Bulbs, Smartwatches, and Router	9	yes	LR, RF, SVM, MLP, LSTM, CNN
[57]	N/A	Door Sensors, Locks, Temperature Sensors, Smart Bulbs, Cameras, and Smart Plugs	6	yes	KNN, LDA, RF

Table 2.1: Summary of Related Work

are not part of a smart home environment could be detected. Since there are five similar existing research, this thesis differs from them by sniffing BLE packets with another tool, creating a data set with other devices, extracting other features, and developing other ML models.

Chapter 3

Background Knowledge

The background knowledge required for this thesis is provided in this chapter. First, it explains what BLE devices are and how they differ from other IoT devices. To analyze the BLE packets, the BLE protocol stack, and the BLE packet format are described. Moreover, malicious attacks that affect BLE devices and how they occur are explained. Lastly, seven different ML models are shown and discussed, as this thesis develops ML models to classify BLE devices.

3.1 Introduction of Bluetooth Low Energy Devices

There are over 10 billion IoT devices in daily use such as smart home devices or devices that support people with their medical issues [24]. Bluetooth was introduced in 2003 by the group Bluetooth SIG as a short-range technology [24]. There are two types: the Bluetooth Classic and the BLE [24]. BLE was released with version 4.0 in 2010 [24]. While Bluetooth Classic devices have a higher throughput and transmit more data, BLE devices have a very low throughput [24]. BLE devices have a low power consumption while Bluetooth Classic devices need more power energy [24]. This is the main reason why BLE is the most popular and most widely used wireless communication application among others such as ZigBee, Bluetooth Classic, and WiFi [24]. In recent years, the number of BLE devices is increased enormously because of the low power consumption [24].

3.2 BLE Protocol Stack

The BLE protocol, depicted in Figure 3.1 is a stack divided into three main parts: the Application, the Host, and the Controller [11]. On top of the BLE stack is the Application part which directly interacts with the user of the device via an interface [11, 26]. Highlighted in red is the Host Controller Interface (HCI) which is the communication transfer of the Controller and the Host [11, 22]. The Host and the Controller parts with their layers are described in the subchapters.

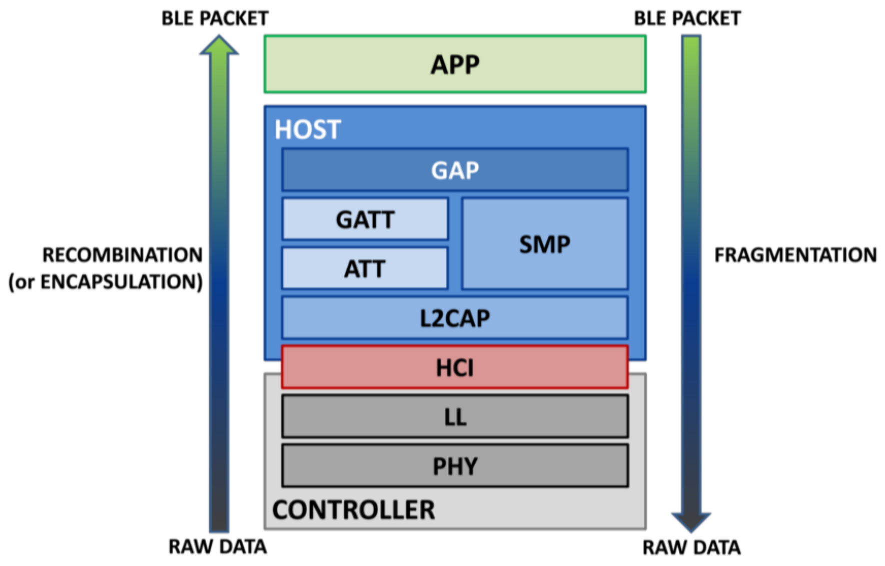


Figure 3.1: BLE Protocol Stack [11]

3.2.1 Host

The second part of the BLE stack is the Host. It has six sublayers: Generic Access Profile (GAP), Generic Attribute Profile (GATT), Attribute Protocol (ATT), Security Manager Protocol (SMP), Logical Link Control and Adaptation Protocol (L2CAP), and HCI (host side).

Generic Access Profile

The first layer of the Host is the GAP layer. It defines how BLE devices communicate and interact with each other [26, 24]. To discover other BLE devices and to ensure the connection between them, GAP needs to define the device roles, modes, and procedures [11, 9]. These device roles are: the Broadcaster, Observer, Peripheral, and Central [9]. Being a Broadcaster, the device only broadcasts data. Whereas, the device as an Observer only receives the broadcasted data [9]. A device has the Central role when it has multiple connections. When a device uses only one single connection with another device in the Central role, then it has the Peripheral role [9]. The Central role is also called the Master, and the Peripheral role is called the Slave. At any given time, a device can be either a Master or a Slave, but not both [9].

Generic Attribute Profile

The GATT layer defines rules on how the data should be packaged and transmitted from one device to another and who can access that data [26, 9]. To establish communication, the devices must be either a Client or a Server [26]. The Server device receives requests from a Client device and stores them for subsequent processing and response [26]. Then the Client device requests data from the Server devices and gets access to READ or WRITE that stored data [26, 9].

Attribute Protocol

The ATT layer provides the basis for the functioning of the GATT layer [26]. As previously mentioned, a device can be either a Client or a Server. This rule is defined in the ATT layer [26, 9]. Additionally, the ATT layer structures the exchanging data into attributes containing a 16-bit attribute handle, a Universal Unique Identifier (UUID), a value, and a set of permissions [26, 11]. It also defines the access to the stored data in the server. In other words, it defines which device is allowed to READ or WRITE that stored data in the server [26]. In summary, the ATT layer defines the rules of the exchanging data between the Client and Server in the GATT layer, and the GATT layer then follows these rules defined in the ATT layer so that communication between devices can be established [11].

Security Manager Protocol

The SMP layer does just one thing: it ensures communication security by encrypting and decrypting the transmitted data [26, 11].

Logical Link Control and Adaptation Protocol

The L2CAP is a multiplexer layer converting multiple protocols from the upper layers into standard BLE packets [26]. This step is necessary so that the BLE packets can be passed down to the lower layers, to the Controller part [24]. When the packet size exceeds the defined maximum BLE payload size of 23 bytes, the L2CAP breaks that packet into chunks [9, 10]. In contrast, for forwarding data into the upper layers, L2CAP combines multiple little packets into one single packet [10].

3.2.2 Controller

The third part of the BLE stack is the Controller. It has three sublayers: HCI (controller side), LL, and Physical Layer (PHY).

Host Controller Interface

The HCI is an interface for communication between the Controller and the Host parts [10]. There are commands defined so that the data can be converted into data packets and sent from the Controller to the Host [11]. On the other hand, it sends events back from the Controller to the Host [10].

Link Layer

The LL has three states: advertising, scanning, and connected states [10]. As explained in Chapter 3.2.1 there are the Broadcaster and the Observer roles. The Broadcaster only broadcasts data and the Observer receives the broadcasted data. By broadcasting data, the data is transmitted as advertising packets through a so-called advertising channel and the Observer receives the data through this channel [9]. The Broadcaster is also called an Advertiser and the Observers are also called Scanners [9]. The communication between two devices is established through the Broadcaster (Advertiser) mode and the Observer (Scanner) mode [11]. To get two devices connected, the Advertiser advertises through the channels a message that is ready to be connected [9]. It periodically sends messages to any Scanner device [11]. Broadcasting is the quickest method for sending data to multiple devices simultaneously [11]. On the other side of the channel, the Scanner device waits to get such messages and continuously scans the availability of them [9, 11]. As soon as the Scanner receives a message from its Advertiser, it sends a connection request message to the Advertiser. If the Advertiser approves the connection request, they enter the connected state and can communicate [10, 9]. The connected devices transmit permanently and periodically data [11]. In the connected state, the Advertiser device is also called Master and the Scanner device Slave [9]. A Master can have multiple connections with multiple Slaves, but each Slave has only one Master [9]. The reason BLE devices have low power consumption is that the Slaves are by default turned into sleep mode when they are not used [9].

The identification of devices during communication is done by the Bluetooth Device Address (BDADDR) [23]. It is a 48-bit identifier found within the AdvA field of the device packet [23]. There are two types of AdvA: Public Addresses and Random Addresses [10]. Public Addresses are the unique MAC addresses [23]. While the Public Addresses are fixed and do not change, Random Addresses are randomized [10]. Random Addresses have two types: Static or Private Addresses [10]. To prevent tracking, the Random Addresses are changed at intervals of up to 15 minutes [23].

Physical Layer

The last layer of the Controller is the PHY layer. Identical to Bluetooth Classic and Wi-FI, the frequency range of BLE is a 2.4 GHz Industrial Scientific Medical (ISM) band [22]. This band is divided into 40 RF channels [9, 10, 22]. There are two RF channels: the advertising and the data channels [9]. The goal of the devices in the advertising channel is to broadcast data and find another device to establish a connection with [9]. In the data channels, connected devices communicate [9]. Three of the 40 channels are advertising channels, the remaining 37 are data channels [9]. Since the frequency range is the same as for Bluetooth Classic and other wireless protocols, BLE devices use frequency hopping [22]. The three advertising channels 37, 38, and 39 are equally spread over the ISM band as shown in Figure 3.2. The advertising channels are the red ones and the data channels are the blue ones. This distribution of the advertising channels avoids interference with other protocols [22].

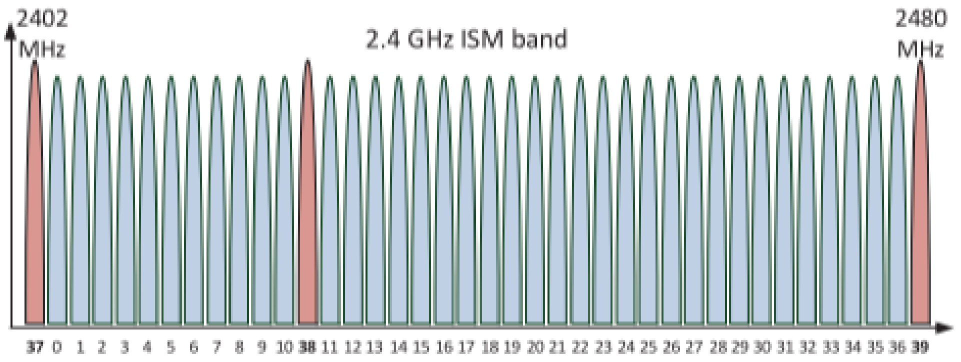


Figure 3.2: Distribution of Advertisement Channels over the ISM Band [33]

3.3 BLE LL Packet Format

In this chapter, the reference [76] can be assumed to be the source unless otherwise stated. There are two packet formats, the uncoded and the coded PHY, defined by the BLE v5.1 specification. Most BLE devices use the uncoded PHY shown in Figure 3.3. Each packet contains fields such as Preamble, Access Address, PD, and Cycling Redundancy Check (CRC). The constant tone field is optional. When data is transmitted on the LE 1M PHY the Permeable field is 1 octet, and on the LE 2M PHY it is 2 octets. The access address is the unique identifier of the packet. As data is transmitted on the advertising physical channel, the PDU is also known as an Advertising Physical Channel PDU.

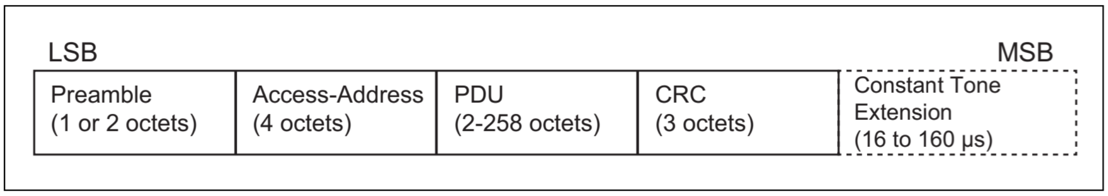

LSB
MSB
Preamble (1 or 2 octets)
Access-Address (4 octets)
PDU (2-258 octets)
CRC (3 octets)
Constant Tone Extension (16 to 160 µs)

Figure 3.3: Packet Format of the Uncoded PHY [76]

The PDU field contains the 16-bit PDU Header and the variable-size PDU Payload. The PDU Header is further subdivided into PDU Type, Reserved for Future Use (RFU), Channel Selection (ChSel), Transmitting Address (TxAdd), Target Address (RxAdd), and Length fields as shown in Figure 3.4. The ChSel field has a value of 1 if the ChSel Algorithm is supported, otherwise 0. The TxAdd field indicates whether the transmitter address is public or random. If it is public, the value of the TxAdd field is 1, otherwise 0. The same applies to the RxAdd field. The Length field is the payload length.

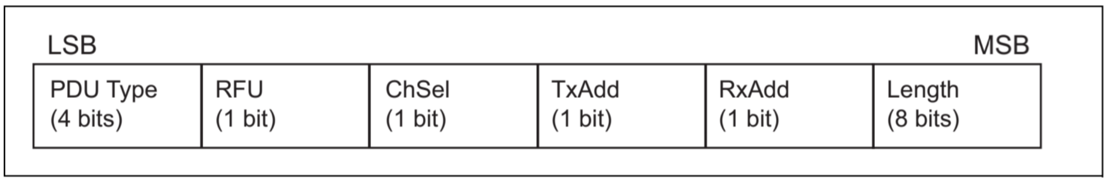


Figure 3.4: PDU Header of the Advertising Physical Channel

The PDU Type field indicates the type of PDU. There are three types of PDU. The advertising PDU has values as ADV_IND, ADV_DIRECT_IND, ADV_NONCONN_IND, or ADV_SCAN_IND. The scanning PDU has values of SCAN_REQ or SCAN_RSP. The initiating PDU has a value of CONNECT_IND. The descriptions of these PDU types are shown in Table 3.1.

PDU Type	**Description**
ADV_IND	Undirected advertising
ADV_DIRECT_IND	Direct advertising
ADV_NONCONN_IND	Undirected, nonconnectable, nonscannable advertising
ADV_SCAN_IND	Undirected, scannable advertising
SCAN_REQ	Scan request
SCAN_RSP	Scan response
CONNECT_IND	Connect request

Table 3.1: PDU Types and their Descriptions

If a device wants to transmit data, it has a PDU type of ADV_IND. On the other hand, a device of type ADV_NONCONN_IND will not accept a connection. The ADV_IND device sends connection requests to other devices to let them know that it wants to connect. The payload of type ADV_IND is shown in Figure 3.5. The payload has AdvA and AdvData fields. AdvA is the address of the advertiser which can be public or random. The AdvData contains the information about the device as defined by the manufacturer.

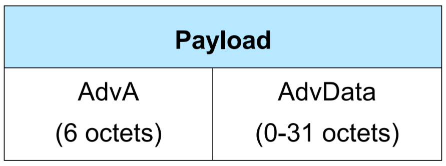


Figure 3.5: Payload of PDU Type of ADV_IND

The AdvData is further divided into several structures. Each structure has Length, AD Type, and AD Data fields, as shown in Figure 3.6. The AD Type field is the type of AD data and the Length field is the length of the AD type and data. The AD Data field contains the information about the device as defined by the manufacturer. The AD type can have values such as Device Name, Company ID, Service UUID, Flags, Manufacturer Specific Data, Tx Power Level, Service Data, Service Class, Local Name, Appearance, BIGInfo, BD_ADDR, Public Target Address, and Random Target Address [77].

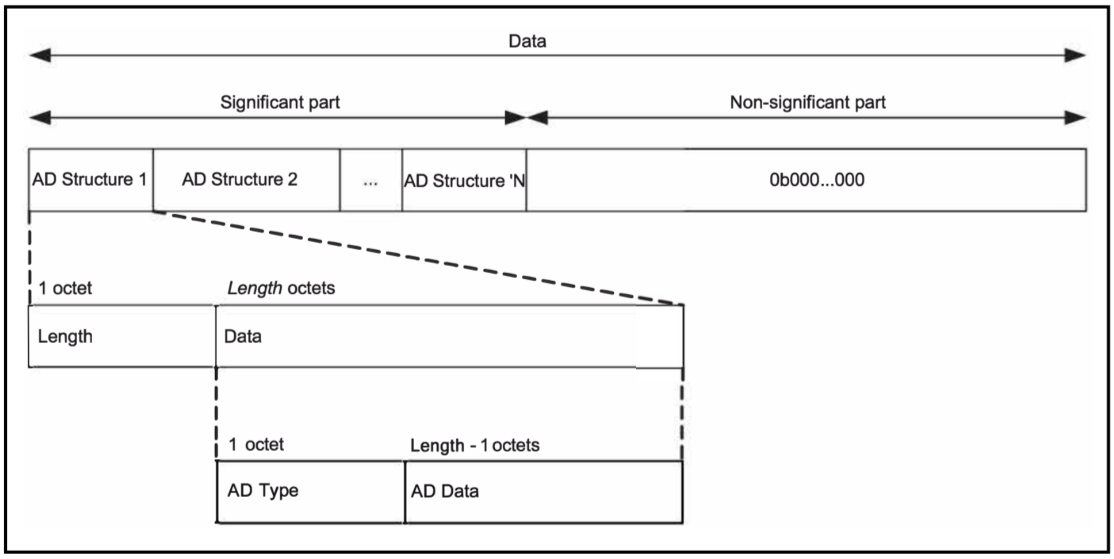


Figure 3.6: AdvData Structure

3.4 BLE Evolution

BLE was first introduced by version 4.0 in 2010. Since then, many other versions have been released. With each release of a new version, the Bluetooth specification changes such that the security of the devices is improved [22]. The security implementations of devices of version 4.0 can vary extremely. On one side, no security mechanisms are implemented, while on the other side, the highest level of security is that the exchanged data is encrypted and the data is protected against Man-in-the-Middle (MITM) attacks [22]. Because of security threats in version 4.0 and 4.1, version 4.2 introduced in 2014 has a major security update by implementing a pairing mechanism [22]. To ensure secure communication between devices, encryption is necessary. This means that MITM has to guess the right bits of a passkey to convince both connected devices to be the legitimate device [22]. By guessing a wrong bit, the MITM is detected [22]. However, MITM could still occur by forwarding messages between two connected devices [22]. The legitimate devices fail to realize that they are not directly communicating with each other [22]. Version 5.0 was introduced in 2016 with a main update that the security mode was strengthened [22]. In 2019 versions 5.1 and 5.2 are released [22].

3.5 BLE Attacks

Despite many advantages of BLE devices, such as low power consumption, they also have some weaknesses. As [22] said, there are some security gaps and privacy weaknesses of BLE devices such as sniffing, spoofing, injection of messages, Denial-Of-Service (DOS) attacks, localization tracking, and user behavior tracking. Through these attacks, attackers gain access to the privacy and personal information of the device user [22]. For example, having access to the personal information of Apple headsets, the attacker gains information about the full name of the user [23]. The location and user behavior can be tracked by having access to the fitness tracker device [22]. The attacker can gain information about the route while jogging and use this information for their benefit. Another issue is when the attacker can gain information about the medical condition of the user by tracking medical devices [23]. This kind of information should be kept private and out of the hands of others [23].

3.5.1 Sniffing Attack

Sniffing attacks involve passively sniffing BLE packets with a sniffer tool [22]. Passive sniffing does not disrupt the connection, the packets are just caught [25, 22]. Catching is done either in the advertising or connection mode [25]. In the advertising mode, the sniffer catches advertising packets [25]. Conversely, in connection mode, it catches packets transmitted during a connection of devices [25]. A connection between devices is created with an Access Address (AA) which is only transmitted at the beginning of that connection [22]. To get access to transmitted data, the attack has to catch this AA at the beginning state of the connection [22]. There exist two types of sniffers: those based on development kits and dedicated BLE sniffers [25]. The development kit-based BLE sniffers only catch one connection and miss other packets [25]. The price for these sniffers is under 100 US dollars [25]. In contrast, dedicated BLE sniffers are much more expensive because they can catch packets on all 40 channels and multiple connections simultaneously [25].

3.5.2 Man in the Middle Attack

Whereas sniffing attacks do not disrupt the connection, the MITM attacks can manipulate it [22]. As shown in Figure 3.7, the attacker places himself between two connected devices without being detected by them [22]. One device is the sender and the other one is the receiver. The data transmission can be disturbed or even injected with new messages by the attacker [27, 28]. Once the sender device sends data to the receiver device, the data first gets to the malicious device [22]. After data modification by the attacker, the malicious device passes the message further to the receiver device [22]. The two devices think they are communicating with each other, but in reality, the messages are being manipulated by the attacker [39]. Therefore, in the MITM attack the attacker can have unauthorized access to sensitive information such as passwords of accounts [28].

3.5.3 Replay Attack

The replay attack is shown in Figure 3.8 which is quite different from the MITM attack. The transmitted data packets from the sender device are intercepted and

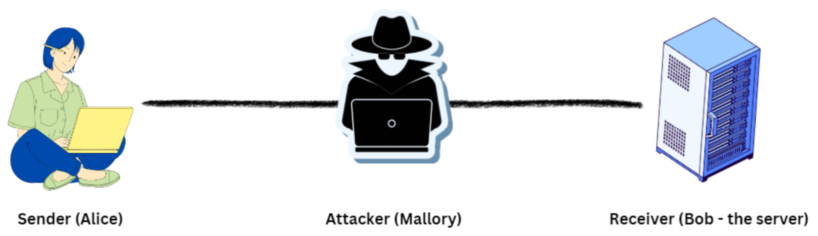


Figure 3.7: MITM Attack [28]

retransmitted at a later point in time by the attacker [27]. The receiver device thinks that the data retransmitted by the malicious device is coming directly from the sender device [27]. Also here, the attacker gains information about sensitive information such as login credentials [27]. The difference to MITM attacks is that replay attacks replay the data packets and do not modify them [28].

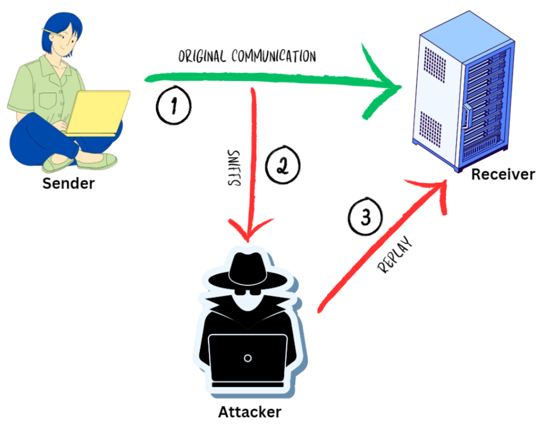


Figure 3.8: Replay Attack [28]

3.5.4 Spoofing

Devices that do not have secure pairing to communicate with other devices are affected by spoofing attacks [32]. For a spoofing attack to take place, two devices must already be connected [31]. For reasons like moving, the connection between devices may be lost due to the large communication distance [31]. As soon as they reach the distance to get connected again, the attacker imitates the BDADDR of a server device and pretends to be that device [31]. The attacker can insert spoofed data and send it to the previously connected client device [31]. That device will accept that data by thinking it is from a server device [31]. Figure 3.9 shows an example of a spoofing attack. The server device is the temperature sensor and the client device is the smartphone. The attacker pretends to be the temperature sensor and sends spoofed data to the smartphone by changing the real temperature value. The smartphone thinks the value comes from the temperature sensor and will adjust the temperature.

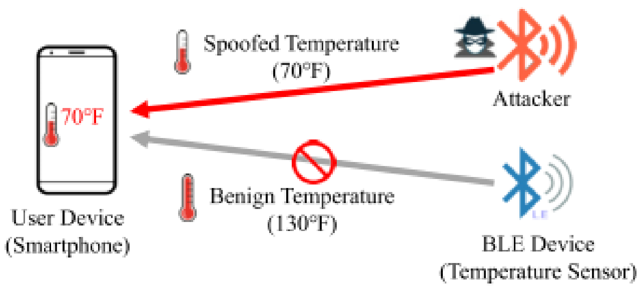


Figure 3.9: Spoofing [32]

3.5.5 Jamming

While spoofing attackers pretend to be legitimate devices, the jamming attackers want to disrupt communication between two devices by emitting the communication channel with noise [33]. This is done by doing interference during communication [40]. The jamming attacker therefore prevents the devices from getting connected. When disrupting the communication channels, the devices try to reconnect [38]. A jammer attacker can jam either a single channel or multiple channels at a time [33]. One limitation of jamming attacks is that the attacker must be within a range of one meter of the target devices [38].

3.5.6 Session Hijacking

In the session hijacking attack, the attacker forces the user to terminate its connection from an Access Point (AP) [34]. This forcing is due to the attacker imitating the AP's MAC address and de-authenticating the device, which leads to a termination of its connection [35]. By pretending to be that disconnected device, the attacker takes control of that existing communication session between two devices [34, 35]. Now the attacker can inject malicious data into that existing connection [36]. In Figure 3.10 a session hijacking attack is shown where an attacker pretends to be device A and sends malicious data to the server victim. However, session hijacking attacks do not change the connection state or do not cut the communication, it takes over the connection without breaking it with the other device [36]. That other device does not notice that a malicious device has taken over the communication session [36]. This attack occurs in the LL [36]. Hijacking the data of medical devices that measure the blood sugar level, oxygen level, or pulse leads to serious issues [36, 37].

3.5.7 Packet Injection

In a packet injection attack, the attacker injects manipulated packets into networks [42]. For example, a temperature sensor measures the temperature every hour and informs the user via an application whether the temperature is rising or falling. By

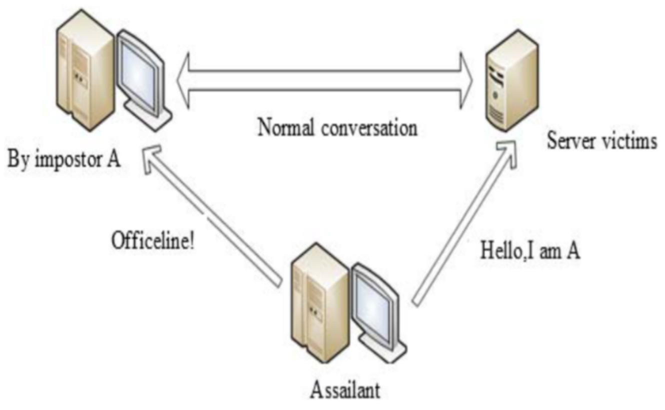


Figure 3.10: Session Hijacking [41]

injecting manipulated sensor data into the network, such that the sensor should measure the temperature once every four hours, the user will receive a delayed notification that a high temperature has been reached [30]. This type of attack also leads to serious problems. A limitation of the packet injection attack is that the attacker should be within ten meters of the target device that wants to inject malicious packets [38].

3.5.8 Eavesdropping

Eavesdropping is listening to private communications or packets being transmitted over a network [40]. The attacker tries to locate the data or the request to be transmitted [40]. Eavesdropping does not disrupt the conversation or manipulate the data, the attacker listens in silence to the data exchange. Since there is no disruption of the communication, eavesdropping is harder to detect compared to spoofing or jamming.

3.6 Machine Learning

ML covers three main categories: unsupervised, supervised, and reinforcement learning. In supervised ML the dataset should be labeled [61]. The task of supervised ML is to train a model by using input-output samples so that it can map an unseen input sample to its corresponding output sample [65]. These input-output samples are known as the training data. The training dataset is the input for training the model. By learning the pattern within the training data, the model should be able to predict the output variable for the test datasets [63]. Contrarily, unsupervised ML does not have a labeled dataset and must learn patterns within the data without knowing the outputs [66]. The goal is to cluster the data into clusters and categorize the data into subclasses to discover the classes of the unlabeled dataset [61]. Unsupervised ML is commonly used for description tasks and supervised ML for prediction tasks [66]. Within supervised ML, there are two subtypes: classification and regression. Classification is used to predict categorical outcomes, and regression is used to predict continuous outcomes [66]. In reinforcement learning, the agent should achieve the highest cumulative reward while interacting with the environment [65]. To interact with the environment, the agent takes actions and receives rewards [61]. Based on the received rewards, the agent chooses actions from a given set with the highest reward [63]. Since this thesis aims to classify BLE devices, these supervised ML models are introduced: DT, RF, LR, NB, KNN, SVM, and neural networks.

3.6.1 Decision Trees

DT are types of trees that arrange and classify the data based on feature values [61, 62]. Every branch of a DT is a decision rule, and every node is a feature of the data that needs to be classified [61, 62]. Data is classified and sorted based on the feature values, starting at the root node [61, 62]. The root node does not have incoming edges [63]. The DT depends on selecting the features that best partition the data at the root. To select the best features, the gini index could be used [61]. Each node in the tree is split and nonoverlapping sub-trees are created by dividing the data into subsets until the data is separated into subsets that belong to the same class [61, 62]. The nodes without outgoing edges are leaves and are at the bottom of the tree [63]. These leaves are the predicted outcomes, that belong to the most common class [63, 66]. It is a recursive partition algorithm starting at the root node and ending in the leave nodes [63, 66].

A disadvantage of DTs is that they can lead to overfitting [61, 62]. To avoid overfitting, the algorithm should be stopped before a data point fits the data perfectly, or it should be pruned and the DT should not grow to its full size [61, 62]. Furthermore, errors can spread across the trees [64]. Since DTs are graphs that show decisions and their outcomes, their main advantage is that they are easily understood [61, 65]. Moreover, DT manages interactions between features, is unaffected by outliers, is robust to noise, can handle linearly inseparable data, and can handle missing and redundant values [64]. According to [61], DTs perform well in classification when the data have discrete or categorial features.

3.6.2 Random Forest

Since DTs are prone to overfitting, combining multiple DTs and training them simultaneously together, prevents overfitting [66]. These stacked DTs are called RF classifiers and are shown in Figure 3.11. RF is an ensemble tree-based ML classification algorithm combining several DT models into an ensemble approach for classifying instances [66, 60]. Each tree has a random subsample of the training data [60]. Therefore unlike DT, the trees in the RF classifier do not depend just on one dataset [60]. Like DT also RF is a recursive partitioning classifier and iterates through each branch and node until reaching the leaves [66]. Each tree predicts an outcome result as shown in Figure 3.11. Overfitting is prevented since all trained trees are averaged and make a more accurate prediction [66]. Instead of averaging all trained trees, another approach is to take the class prediction through majority voting over all trained trees [64]. This aggregation outperforms every individual DT's output.

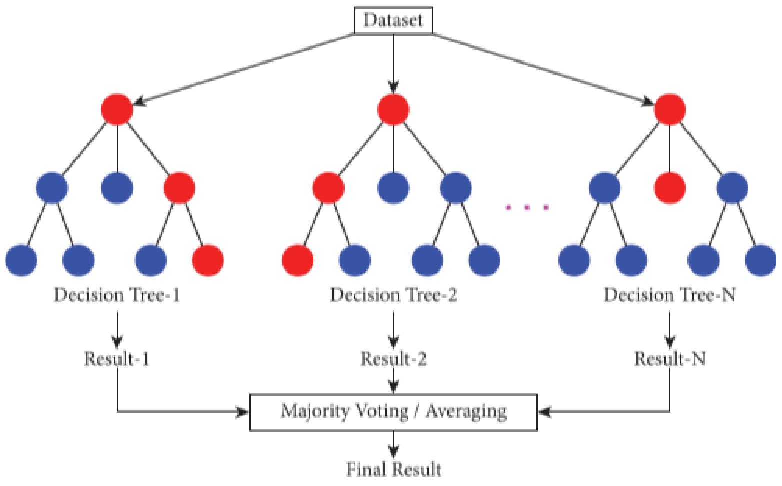


Figure 3.11: RF Classifier [60]

In addition to the advantage of preventing overfitting, RF is robust to noise and during training, the classifier feature importance is measured which shows which features are most relevant to predict the outcome [66]. Based on this the splits are determined by sorting the features and pure subsets are created [66]. Therefore the model's ability to discriminate between classes is improved [66]. It can handle missing data by maintaining accuracy and can also handle many input samples simultaneously. Having too many trees the RF classifier is very slow to be trained [64]. Another disadvantage compared to DT is that the trees in RF are less visually interpretative than one single tree in DT [66].

3.6.3 Logistic Regression

LR classifier measures the relationship between the input and output variables by returning conditional probabilities [46]. The input variables are the features in the dataset and the output variable is the target value [63]. Based on the relationship between them, the goal of the classifier is to predict the target value [63]. By finding the best curve that fits the dataset, LR takes the highest probability of a device belonging to a class. It can handle non-linearity but a disadvantage is that LR requires a large dataset to produce stable predictions [64].

3.6.4 Naive Bayes

Another ML model is the NB model. It is also a probability model that predicts the probability of an instance belonging to a class rather than classifying it [61].

The probability of belonging to a class should be as large as possible [61]. NB is robust to noise and needs less computational time for training the model cause of its simplicity [63, 64]. Because of its simplicity, the NB classifier assumes independence across all features in the dataset [61, 65] Therefore it is less accurate than other ML models [61].

3.6.5 K-Nearest Neighbour

KNN is an instance-based ML algorithm, which is a lazy-learning algorithm [61]. Compared to eager-learning algorithms such as DT, NB, and neural networks, KNN needs less computation time during training the model but more computation time during the classification of instances [61]. The classification of an input instance from the dataset into its class is done by looking at its k nearest neighbors [61]. The neighbor with the majority of instances that are near to that input instance is then the class label of it [61]. KNNs are affected by noise in the dataset and are sensitive to irrelevant features [64]. Furthermore, the output is dependent on the choice of the parameter k which affects the performance [64]. Another disadvantage is that KNNs need to store all computations and therefore are computationally expensive [61].

3.6.6 Support Vector Machines

The goal of SVM models is to find the best margin of a hyperplane to divide the data instances into classes by taking the largest distances from the hyperplane to the nearest data instance [61, 66]. The goal of the margins is to minimize the classification error by maximizing the distance between the margin and the classes [65]. If linearly separable data can not be separated by the hyperplane without having misclassifications, then a soft margin is used [61]. For nonlinearly separable data, the data are projected into a higher-dimensional feature space by selecting kernel functions [61, 65, 66]. The challenge here is the choice of the best kernel function [61]. Using kernel function, SVM can handle nonlinearly separable data. Furthermore, SVM is independent of the data size and number of features [64]. However, SVM depends on the choice of parameters and is very complex [64]. Additionally, SVM models are binary classifiers and a multi-class problem must be reduced to a collection of several binary classification problems [61]. This leads to a very low speed of training [61]. It is not specified how the predictions are integrated to get the best hyperplane because SVMs are black-box algorithms [66].

3.6.7 Neural Network

Neural networks are a collection of algorithms developed to mimic the human brain to find relationships in a given dataset [65]. They are capable of understanding the pattern in the dataset and generalizing the output [46]. Neural networks have three layers: the input, hidden, and output layers as shown in Figure 3.12. The input layer receives the input sample of the dataset [65]. The hidden layer processes these input samples and the output layer gives the computed output which is the prediction of a class [65]. The actual and computed output values of the neural networks are compared [65]. Based on the errors the parameters are adjusted and the the model is retrained [65]. Neural networks can also classify non-linear separable datasets [62]. One drawback is determining the number of hidden layers. Neural networks with too many hidden layers are prone to overfitting, whereas if there are too few hidden layers, the performance of the network will decline [62]. Therefore neural networks are sensitive to the number of hidden layers [64]. The most commonly used neural network is MLP which is robust to irrelevant input samples and to noise [64]. MLPs are very complex and hard to train, resulting in slower training of the model [64].

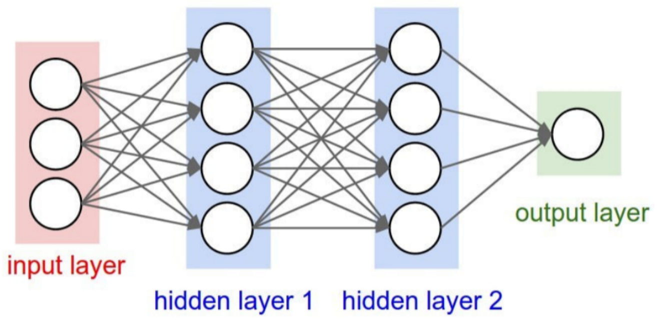


Figure 3.12: Neural Network [70]

Chapter 4

Methodology

In this chapter, the methodology of the thesis is explained in six parts as shown in Figure 4.1. At the bottom of the figure, BLE devices are collected. The different colors indicate their device types. By collecting the data of BLE devices using a sniffer tool, the dataset is generated. Features are extracted and selected from the dataset, which are specific characteristics of device types. The dataset undergoes four preprocessing steps described in Chapter 4.5 and is split into training and test datasets. In the end, ML models are implemented to do the classification task. The goal of the models is to predict the device type of given BLE devices.

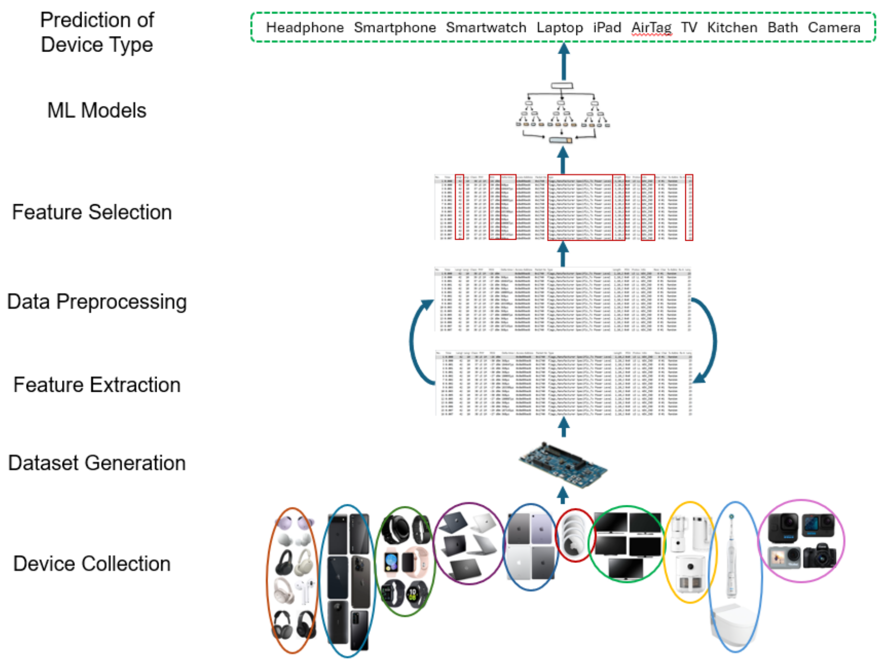

Prediction of Device Type
Headphone Smartphone Smartwatch Laptop iPad AirTag TV Kitchen Bath Camera
ML Models
Feature Selection
Data Preprocessing
Feature Extraction
Dataset Generation
Device Collection

Figure 4.1: Methodology Overview

4.1 Device Collection

To generate a dataset, BLE devices need to be collected. Since the sniffer tool sniffs all BLE devices at a certain distance, the BLE devices should be brought into an isolated room. This means that devices are collected that can be easily transported to an isolated room and are accessible, such as personal devices and devices from friends and acquaintances. The goal is to have as many different types of devices as possible. For the ML model to recognize unique patterns in the dataset of device types, each device type should include at least two devices. Based on the unique patterns of device types, the model can better identify and classify devices, improving its ability to make accurate predictions. All devices that are collected and sniffed are listed in Table 4.1. In total, there are 10 different device types such as headphones, smartphones, smartwatches, laptops, iPads, AirTags, TVs, kitchen devices, cameras, and bathroom devices. Each device type has at least two devices. The Headphone category is the largest group, including 8 devices. The dataset includes a total of 49 devices. The number of devices and the name of manufacturer of the devices per device type are listed in Table 4.1.

No.	Type	Count	Name of Manufacturer
1	Headphone	8	2 Bose, Samsung, Sony Linkbuds, Sony XM3, Sony XM4, Apple, Beats
2	Smartphone	6	2 Apple, Google, Huawei, Samsung, Nokia
3	Smartwatch	6	2 Apple, Ericsson, Huawei, Fitbit, Galaxy
4	Camera	6	GoPro4, GoPro5, GoPro9, GoPro10, Canon, Rollei
5	Laptop	5	2 HP, 2 Dell, Apple
6	TV	5	3 Samsung, 2 LG
7	iPad	4	All Apple
8	AirTag	4	All Apple
9	Kitchen	3	All Xiaomi
10	Bathroom	2	Oral-B Toothbrush, AquaClean Shower Toilet

Table 4.1: BLE Device Types and Their Name of Manufacturer

4.2 Data Collection via Passive Sniffing with nRF Sniffer Tool

The data packets of the collected BLE devices are sniffed via the nRF Sniffer tool and compiled into a dataset. Several BLE sniffers are available online[1], such as the nRF52840 Dongle, nRF52840 DK, nRF52833 DK, nRF52 DK, nRF51 DK, and nRF51 Dongle. In this thesis, the nRF52840 DK[2], known as the Nordic nRF Sniffer, is used. This tool sniffs nearby BLE packets within a specified range

[1] `https://www.nordicsemi.com/Products/Development-tools/nrf-sniffer-for-bluetooth-le`

[2] `https://www.nordicsemi.com/Products/Development-hardware/nRF52840-DK`

passively and in real-time [7]. The sniffed BLE packets are transmitted or received data by a device [7]. As already mentioned in Chapter 3.2.2, randomized addresses are changed at intervals of up to 15 minutes. Therefore the BLE packets are sniffed for 20 minutes to ensure consistency in data collection. The Appendix A.1 explains the installation guidelines for the nRF Sniffer and Wireshark tools. The Wireshark tool is used to sniff and visualize the captured data packets. These packets are saved as PCAP files and are converted into CSV files for further analysis.

4.2.1 Initial Capturing of BLE Packets in Public Places

Feasibility tests have been made by first sniffing data packets from devices in three public places. These three following places are selected:

- **Department of Informatics (IFI) of the University of Zurich (UZH)**: The reason this place is chosen is that the IFI has a large number of BLE devices and can therefore generate a large dataset.
- **Smart Home**: As most BLE devices are smart home devices and because of the weaknesses of BLE devices personal information can end up in the hands of attackers, also smart home devices are sniffed. This allows these devices to be classified and unauthorized ones to be identified.
- **Train Station**: BLE devices are sniffed at the train station to determine the high-traffic area. This shows the true volume of data transmission.

Based on the initial data capturing, it appears that specific data information in the packets is limited. Most AdvA are randomized and most packets do not contain information about the device name. The reason is that BLE devices' privacy is preserved while they are being sniffed. As there is not enough information about the packets due to privacy protection, it is difficult to identify the device type of each packet. Due to the limitation of having no clue about a device type, another approach is chosen which is described in the next subchapter.

4.2.2 Capturing BLE Packets from Known Device Types

Capturing BLE packets from known device types listed in Table 4.1 avoids the limitation of having no clue about a device type. The packets from these devices are sniffed in an isolated room to ensure that the sniffer tool only captures packets of devices with known device types and no packets from other devices. This isolated room is located in the basement of the IFI to minimize disturbance during sniffing. To be able to assign the packets to the corresponding device type later, devices of different types but with the same name of manufacturer, such as Apple smartphones, Apple headphones, or Apple laptops, must be sniffed separately. The reason for this is that the name of manufacturer on the device packets only reads *Apple* and it is therefore difficult to identify the device type. Separate sniffing of these devices ensures distinct identification.

4.3 Dataset Labeling

After evaluating the initial approach, a new approach is made where devices with known device types are sniffed. Each sniffed packet is annotated with its device type. The device type is also called a label. The information in the BLE LL is analyzed to identify the packets. This information contains the device name, the company ID, and the UUID of the sniffed packets, from which the device type can be derived. Conditions are implemented in Python so that each packet has a label in the dataset. One example of labeling two HP laptops, one Apple laptop, one Bose headphone, one Samsung headphone, and one Samsung TV is shown in Algorithm 1. The file name is checked and for each AdvA group, the company ID which is the name of manufacturer, and/or the UUID, and/or the device name is compared to a value to label that packet. Additionally, a second label *subcategory* is added to the CSV files which is the name of manufacturer. This *subcategory* label is needed for data splitting which is explained in Chapter 4.7.

Algorithm 1: Labeling Data Packets Example

```
Input: input_file_path, output_file_path Output: Labeled data packets
Load CSV file from input_file_path into dataframe df
Group by AdvA into unique_advertising_address
Initialize new columns label and subcategory in df
for advertising_address in unique_advertising_address do
    Extract device_name, company_ID, and UUID
    if '2 HP laptops, 1 Apple laptop, Bose & Samsung headphones, Samsung TV' in input_file_path then
        if company_ID equals 'Apple, Inc.' then
            Set label to 'Laptop' and subcategory to 'Apple laptop'
        if company_ID equals 'Microsoft' then
            Set 'label' to 'Laptop' and subcategory to 'Hp laptop'
        if company_ID equals 'Samsung Electronics Co. Ltd.' then
            Set label to 'Headphone' and subcategory to 'Samsung headphone'
        if advertising_address starts with 'Bose_33:04:64' or company_ID equals 'Lucimed' or UUID equals 'Bose Corporation,Google,Amazon.com Services, Inc.' then
            Set label to 'Headphone' and subcategory to 'Bose-1 headphone'
        if advertising_address starts with 'SamsungElect_25:41:bd' or device_name equals '[TV] Samsung Q60BA 65 TV Mind' then
            Set label to 'TV' and subcategory to 'Samsung TV'
Save dataframe df to output_file_path
```

4.4 Feature Extraction

Features that are characteristic of a device type are extracted from the sniffed BLE packets in the dataset. Certain features are directly extracted from them, while others are derived through subsequent calculations based on the packet data. The extracted features are the input for training the ML model. The goal of the model is to learn the unique pattern of a device type to predict the type of that device.

4.4.1 Direct Extraction of Feature from Sniffed BLE Packets

The column headers of CSV files are the extracted features. The complete list of these features is provided in Appendix C after the data preprocessing step. A summary of the most important features in the BLE LL is shown in Table 4.2. There are features related to the packet header and AdvData. *Packet Header Length* is a feature due to its variability across devices. The *PDU Type* feature is the type of broadcast packet, while the *ChSel Algorithm* feature is the ChSel strategy used. The *AdvData Type* feature lists all types such as Flags, Manufacturer Specific, Tx Power Level, Service Class UUIDs, Serice Data, Appearance, and Device Name. The *Length of the AdvData Type* feature is the corresponding length of the types. The *Power Level (dBm)* feature is the power used to transmit packets. As these features are extracted by [45], they are also extracted as features in this thesis. As mentioned in Chapter 2, other researchers have extracted features such as *RSSI*, *Length of Payload*, *Channel Index*, and *Delta Time*. The *RSSI* feature is the signal strength of a transmitted packet, the *Length of Payload* feature is the length of the transmitted messages, and the *Delta Time* feature is the measured time to send or receive a packet in microseconds. As their values vary between devices, they are also extracted as features.

Category	**Features**
Packet Header	Packet Header Length PDU Type ChSel Algorithm
AdvData	AdvData Type Length of the AdvData Type Power Level (dBm)

Table 4.2: Features in the BLE LL

4.4.2 Derived Feature Extraction from Sniffed BLE Packets

Additional features can be derived from the directly extracted features of the sniffed BLE packets. A flow is defined as packets from the same AdvA. A device has at least two flows, as the AdvA is changed at intervals of up to 15 minutes and the sniffing process takes 20 minutes. The complete list of derived features is in Appendix D. The *Duration Time* feature is calculated by taking the difference between the minimum and maximum time for each flow. The minimum and maximum time is the time captured by the sniffer while sniffing the packets. The *Number of Packets* feature is the sum of all packets of a flow. The frequency is calculated by how many packets are transmitted per second and how long it takes to transmit one packet by each flow. These are the *Packets per Second* feature and *Time per Packet* feature respectively. As [44] noted, the frequency is decreased if the screen of a device is switched off and on very infrequently and increased if the screen is switched on several times. That means that laptop devices will have a higher frequency than smartphone devices since the screen of laptop devices is not switched on and off frequently. The *Bytes per Second* feature is the total *Packet Header Length* feature divided by the *Duration* feature. Calculating the maximum, minimum, average, standard deviation, and variance of the *Packet Header Length* feature, the *RSSI* feature, the *Length of Payload* feature, and the *Delta Time* feature, more features are derived. The RSSI varies depending on the distance between two communicating devices. By calculating the standard deviation of the *RSSI* feature, devices with a higher likelihood of movement, such as smartphone devices, tend to have a higher standard deviation of RSSI values compared to less mobile devices like laptop devices [44]. Another feature is the direction of the packet, being either forward or backward. The direction is determined by analyzing the *Tx and Rx Address* features. Tx stands for transmitting and Rx for receiving and both either have a value of public or random. If a packet has a value in the Rx Address column, then this packet is marked as backward because it is receiving packets. If this field is empty and the Tx Address column has a value, then it is marked as forward because it is transmitting packets. Unlike all other derived features which are calculated per flow, the *Packet Direction* feature is done for each packet. Knowing the direction of each packet, the *Number of Forward Packets* and the *Number of Backward Packets* features are calculated for each flow by summing all packets of a flow which are marked as forward and backward respectively.

To summarize, these features are the derived features and are attached as new columns in the CSV files:

- Number of Packets
- Packets per Second
- Time per Packet
- Bytes per Second
- Packet Header Length: Maximum, Minimum, Average, Standard Deviation, Variance
- RSSI: Maximum, Minimum, Average, Standard Deviation, Variance
- Delta Time: Maximum, Minimum, Average, Standard Deviation, Variance

- Packet Direction
- Number of Forward Packets
- Number of Backward Packets

4.5 Data Preprocessing

The dataset undergoes four preprocessing steps: data cleaning, removal of unique identifiers, data transformation, and data balancing. To consolidate the preprocessing steps all CSV files are merged into a single CSV file.

4.5.1 Data Cleaning

By converting the PCAP files into CSV files, some values such as μ or $\surd$ are automatically converted into nonreadable values and must be replaced with "microseconds" and "true" accordingly. Columns with too many missing and constant values are removed from the CSV file. The reason is that they do not contribute to the classification task and are thus irrelevant. Additionally, columns such as *No.* and *Time* are also removed, as they are the chronological order in which the packets are sniffed and therefore do not contribute to the classification task.

4.5.2 Removal of Unique Identifiers

Unique identifiers, such as the AdvA, company ID (name of manufacturer), device name, and UUID, are highly specific to a single type of device. By training the model with distinct identifiers, the model can be easily trained to predict the device type. For example, if all packets with the device name "LE-WH-4" belong to the device type Headphone, the model will only predict the device type based on the device name and will not learn the pattern from the features. Therefore, removing unique identifiers directly linked to specific types is essential for a more robust and generalized model.

4.5.3 Data Transformation

Within the dataset, columns are numerical or categorial. Since ML models accept only numerical input, the categorial data needs to be transformed into numerical representation to solve classification problems [70]. Encoding techniques exist for this purpose, such as label encoding, and one-hot encoding [70]. There are two categories of categorial data: nominal with no specific order and ordinal with a specific order [70]. Nominal data do not have order or ranking and for ordinal data the order matters but the distinction between values is irrelevant [70].

Label encoding is used to encode ordinal data [70]. Every value in label encoding is transformed into an integer value while the order is maintained [70]. These integer values are arranged in an ordered sequence, with values closer to each other being very similar. For example, value 1 is closer to value 2 than to value 4. On the other hand, nominal data is transformed using one-hot encoding where there is no link between the categorial values [70]. In this technique, the categorial values are replaced by a set of binary values, with each unique integer value represented by a binary vector [70]. A value of 0 denotes the category's absence and 1 its presence [70]. For example, if there are three unique categorial values, the encoding might appear as [1, 0, 0], [0, 1, 0], or [0, 0, 1], indicating the presence or absence of each categorical value. As the extracted features in the dataset do not have an ordinal relationship between the categorial values, one-hot encoding is the preferred approach. These values are transformed using the `get_dummies()` function from the Pandas library[3].

4.5.4 Data Balancing

Some device types may transmit and have more packets than others, resulting in an imbalance. This can lead to misclassification of underrepresented classes [68]. To address this issue and prevent certain types from being overrepresented while others are underrepresented, data balancing techniques are used. Undersampling the majority classes and oversampling the minority classes are two common approaches used for handling imbalanced data [68, 69].

Undersampling involves reducing the number of samples in the majority classes by randomly removing samples from them until all classes have the same number of samples [68, 69]. This strategy creates a more balanced dataset and helps mitigate the bias introduced by the overrepresented classes [69]. In contrast, oversampling increases the number of samples in underrepresented classes by creating synthetic samples or duplicating existing ones [68]. This ensures that minority classes have a comparable influence during training as the majority classes [69].

For this thesis, considering the availability of a large dataset with 49 devices and sufficient sniffed packets for minority device classes, undersampling is used. This decision aims to address the data imbalance effectively while maintaining the integrity of the dataset. The undersampling method is performed after the models have been trained. If the accuracy is high enough, this step is not necessary. However, if the accuracy is low, undersampling is applied to the dataset and the models are retrained.

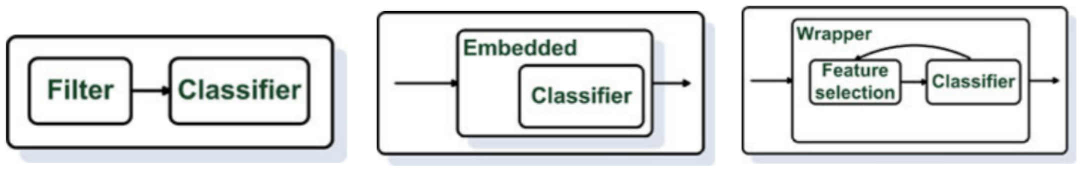
4.6 Feature Selection

Having a high number of features as input, it is difficult for ML models to handle it [72]. Feature selection is an important step as it helps refine the set of features

[3]`https://pandas.pydata.org/docs/reference/api/pandas.get_dummies.html`

by selecting the most relevant subset of features and discarding the irrelevant ones [72]. The reason is that not all extracted features are equally important for training a model to predict the outcome. Features are less important if they are used to train the model but do not improve the accuracy [71]. Therefore these features can be omitted. By excluding less important features, the risk of overfitting is reduced [72]. Additionally, feature selection reduces the training time and improves accuracy by focusing only on the most informative features [72]. By using the smallest subset of features, the trained model achieves an optimal performance.

In supervised ML, feature selection techniques are filter, embedded, and wrapper methods, as shown in Figure 4.2 [72]. Filter methods remove features based on their relation to the target value [72]. Features with a low correlation to the target value are excluded [72]. The classifier is trained on the remaining features [72]. Filter methods include the IG, Chi-square, Correlation, and Relief [72]. They do not interact with the classifier and are therefore independent of it [72]. They also have low computational cost and are fast [72]. Embedded methods train the classifier and check the accuracy of different subsets of features [72]. The subset of features with the highest accuracy is selected [72]. Unlike filter methods, embedded methods interact with the classifier and are able to capture feature dependencies [72]. The disadvantage is that they depend on the classifier [72]. Wrapper methods use a subset of features and train the classifier on that subset [72]. The subset is adjusted according to the output and the model is retrained [72]. As embedded methods, wrapper methods also interact with the classifier on which they depend. While the feature dependencies can be captured, wrapper methods are computationally expensive and overfitting may occur [72].



(a) Filter Method [72] (b) Embedded Method [72] (c) Wrapper Method [72]

Figure 4.2: Feature Selection Methods [72]

In this thesis, the filter and embedded methods are used to find the smallest subset of the most important features with the highest accuracy. Due to memory constraints and because wrapper methods are computationally expensive, they are not used. The filter method is implemented using the `SelectKBest()`[4] function from the scikit-learn Python library. The `SelectKBest()` function works for classification tasks and selects the k-best features based on their relation to the target variable. Since the dataset has numeric features, the `SelectKBest()` function is implemented to calculate the ANOVA F-value. This value is computed between each feature in the dataset and the device type. A high F-value indicates that the variance between the feature's value across different device types is very

[4]`https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html`

high compared to the variance within the device types. The embedded method is implemented using the `RandomForestClassifier()`[5] model and the `SelectFromModel()`[6] function, both from the scikit-learn Python library. This method takes into consideration the feature's importance to the RF model. The calculated importance scores of the features show the importance of each feature in making accurate predictions. The features with the highest score have a greater impact on the model to make predictions.

4.7 Data Splitting

The dataset is divided into training and test datasets. The training dataset is used to train the ML models. During model training, the training dataset is further divided into training and validation subsets. The validation dataset is used to evaluate and tune the models during training. Once the models are trained, their performance on unseen data is evaluated using the test dataset by measuring the difference between the actual and predicted values.

4.7.1 Train-Test Splitting

A common data splitting method for developing ML models, used by many other researchers such as [4] and [14], is to split the dataset approximately 80% into training and 20% into test datasets. Other common data splitting methods are 60/40, 70/30, or 90/10 if the dataset is large enough [78]. The split is done randomly. An alternative data splitting approach takes into account the name of manufacturer. One device per device type with a specific name of manufacturer is excluded from the training dataset, but all devices of that type are included in the test dataset. In order to retain 80% of the data in the training dataset and 20% in the test dataset, the device with the fewest data packets in its device type is excluded from the training dataset. For example, if there are smartwatch devices with names of manufacturer Apple, Samsung, and Huawei, and Huawei is the device with the fewest data packets, the packets from the Huawei smartwatch are not in the training dataset. The training dataset only contains packets from Apple and Samsung smartwatches. In contrast, the test dataset contains packets from all three devices. This approach aims to evaluate the ability of the model to predict unseen device packets from a certain name of manufacturer.

[5]`https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`

[6]`https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectFromModel.html`

4.7.2 Train-Validation Splitting

The training dataset is further divided into training and validation subsets. According to [62, 61, 73], the performance of trained ML models can be evaluated using three methods. One method is to split two-thirds of the training dataset for training and use the remaining third for performance estimation. An alternative approach, known as k-fold cross-validation, involves dividing the training dataset into k-equal subsets [73]. One subset is the validation subset and the other k-1 subsets are the training subsets. For each subset that is the validation subset, the classifier is trained on the union of the k-1 subsets and evaluated on the validation subset [73]. The performance of the classifier is the average of the error rates of each subset. A special method of the k-fold cross-validation approach is the leave-one-out cross-validation [73]. Here, each validation subset contains a single instance and therefore the number of instances equals the number of folds [73]. This method is more computationally intensive and is used when the number of instances is small [73]. On the other hand, when the amount of data is large, k-fold cross-validation is used [73]. As the dataset in this thesis contains 10 different device types and 49 devices, the k-fold cross-validation approach is used.

4.8 Device Classification via ML Approach

Two ML models are selected for the device classification task. The goal is to predict the device types by training the models with the provided training dataset and evaluating them with the provided test dataset. There are several ML models to consider, as listed and described in Chapter 3.6. In this thesis, the RF classifier and the lightweight MLP model are selected. The reason for the selection is discussed in the subsections 4.8.3 and 4.8.4.

4.8.1 Training of ML Models

Both ML models are trained on the training dataset including all extracted features, followed by measuring the performance of the model using the test data. As there are two approaches to data splitting, both models are trained twice, once using the 80/20 data splitting approach and once using the alternative data splitting approach. Additionally, all four models with the same structure as before are then trained by using the selected features by the methods described in Chapter 4.6. The RF classifier is trained using the features selected from the embedded method and the MLP model from the filter method. The number of features is chosen to be between 1 and 30. The models are trained on these $n = [1, 30]$ selected features, and the best number of features with the highest accuracy is selected. A total of eight ML models are trained, four RF classifiers, and four MLP models.

4.8.2 Evaluation Metrics

One evaluation metric is the accuracy. It is the proportion of the number of correct predictions to the total number of predictions made by the model [70]. The training accuracy measures how well the model performs on the training dataset and the test accuracy on the unseen test dataset.

The accuracy for each device type is calculated to determine which device type achieves the highest accuracy, as well as to compare the accuracies across device types and the influence of each device type on the overall accuracy. The formula is:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions Made}} \tag{1}$$

Evaluation metrics such as True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN), precision, recall, specificity, and F1-score are measured for each device type. TP and TN values show how many instances are correctly classified, and FP and FN values show how many instances are incorrectly classified by the model [74]. Precision is the proportion of correctly predicted positive instances to the total predicted positive instances [66, 70]. Recall also known as sensitivity is the proportion of correctly predicted positive instances to all instances that should have been identified as positive [66, 70]. Specificity is the proportion of correctly predicted negative instances to all instances that should have been identified as negative [66]. F1-score also known as F-measure or F-score is the weighted average of precision and recall [70]. It is calculated to show the balance between precision and recall [70]. The formulas are,

$$\text{Precision}_t = \frac{\text{TP}_t}{\text{TP}_t + \text{FP}_t} \tag{2}$$

$$\text{Recall}_t = \frac{\text{TP}_t}{\text{TP}_t + \text{FN}_t} \tag{3}$$

$$\text{Specificity}_t = \frac{\text{TN}_t}{\text{TN}_t + \text{FP}_t} \tag{4}$$

$$\text{F-Score}_t = \frac{2 \times \text{Precision}_t \times \text{Recall}_t}{\text{Precision}_t + \text{Recall}_t} \tag{5}$$

where t denotes the device type and where TP_t, TN_t, FP_t, and FN_t are the TP, TN, FP, and FN predictions for device type t [75].

Furthermore, the confusion matrix is plotted which is a $k \times k$ matrix, where k is the number of device types [74]. Its rows are the true labels and its columns are the labels predicted by the model [75]. It shows how many instances per device type are correctly and incorrectly predicted by the model. The elements on the diagonal of the confusion matrix show the number of correct predictions for each device type [75]. High values on the diagonal indicate that the model correctly classified the data. The off-diagonal values show the number of misclassifications [75]. It also shows how many instances of one device type are incorrectly assigned to which other device type.

4.8.3 Random Forest Classifier

The RF classifier is an ensemble ML model that combines multiple DT. The RF classifier is selected because it can handle a large number of features in the dataset. Since the dataset has many features and is complex, the RF classifier is suitable. As already mentioned in Chapter 3.6.2, the RF classifier computes feature importance, and by combining several DTs the RF classifier prevents overfitting. Furthermore, RF classifiers are robust to noise and can handle missing and nonlinear data. The RF classifier is implemented using `RandomForestClassifier()` function from the scikit-learn Python library[7] and require setting several parameters defined in the scikit-learn Python library[8] including:

- n_estimators, default=100: Number of trees
- criterion gini, entropy, log_loss, default=gini: Function used to measure the quality of a split
  - Gini: Gini impurity function
  - Log_loss and entropy: Shannon IG function
- max_depth, default=None: Maximum depth of the trees
  - If None: Nodes are extended until a minimum of min_samples_split samples is present in every leaf, or until every leaf is pure.
- min_samples_split default=2: Minimum number of samples needed to split an internal node
- min_samples_leaf, default=1: Minimum number of samples needed at a leaf node
- max_features, sqrt, log2, None, default=sqrt: Number of features to take into account while choosing the ideal split
  - If sqrt, then max_features=sqrt(n_features).
  - If log2, then max_features=log2(n_features).
  - If None, then max_features=n_features.
- bootstrap, default=True:
  - If True: each tree is built using bootstrap samples
  - If False: each tree is built using the entire dataset

The model is trained on the default parameters. To improve the resulting accuracy, the model is finetuned with hyperparameters defined in Table 4.3. The goal is to achieve a higher accuracy than with the default parameters.

[7]`https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`

[8]`https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`

Hyperparameter	Values
n_estimators	[50, 100, 200]
criterion	[”gini”, ”entropy”]
max_depth	[10, 20, None]
min_samples_split	[2, 4, 10]
min_samples_leaf	[1, 2, 5]
max_features	[”sqrt”, None]
bootstrap	[True, False]

Table 4.3: Hyperparameters for the RF Classifier

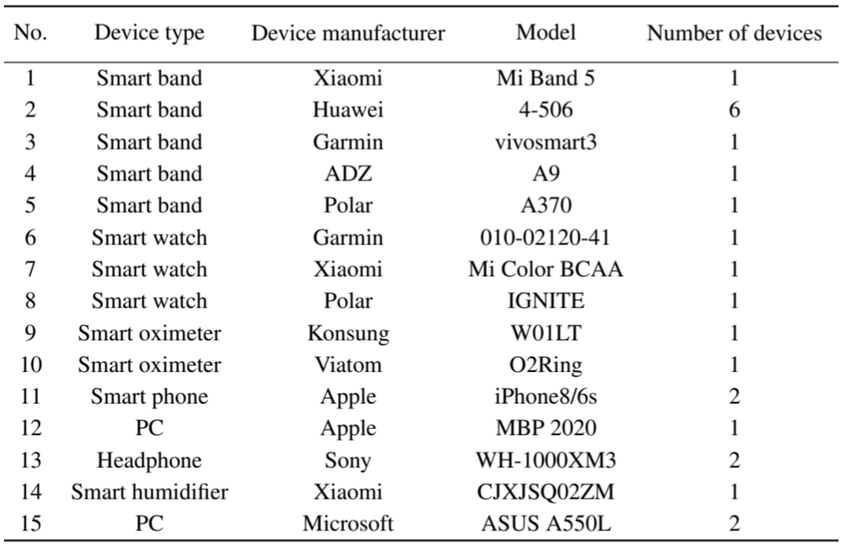
To check overfitting, the training and test accuracies are compared, and cross-validation with $k = 5$ is done. Overfitting occurs when the difference between the training and test accuracies is high, indicating that the training accuracy is close to 1 and the test accuracy is comparatively very low. Such values indicate that the model could memorize the training data, but does not perform well on unseen data. If the two accuracies are close to each other and there is not such a large difference between them, there is no overfitting.

Cross-validation is used to evaluate the performance of the model and to show how robust and stable the model is. The model is trained five times on $k = 5$ different subsets of the data. The cross-validation scores of $k = 5$ subsets and the average accuracy across all $k = 5$ cross-validation folds are calculated and compared to the test accuracy. If the cross-validation scores are close to the test accuracy, it means that the model performs well on unseen data. The learning curve is plotted which shows the accuracy of the model on the training dataset, the training score, and the cross-validation score with an increasing number of training samples. If the training score is high but the cross-validation score is low, this indicates overfitting and the model performs poorly on unseen data. If both scores converge and are high, it indicates that the model performs well on unseen data. If the training score does not decrease and the cross-validation score increases, it means that the model is performing better with more data.

4.8.4 Multi-Layer Perceptron

The reason for choosing the lightweight MLP model is that the research published in [45] uses a lightweight neural network based on MLP and is comparable to this thesis. They use a lightweight MLP model for three reasons. As the BLE packets differ amongst devices of the same type, this variation is difficult to measure. MLPs are capable of identifying different types of devices based on this variation. Furthermore, MLPs can automatically extract features from BLE packets. Since the data in the AdvData are characteristics specific to the device types, MLP models can identify devices by analyzing these characteristics. Because the data structure is simple and is not complex having images or text, a lightweight MLP is sufficient.

They use features such as the *PDU* and the *ChSel* from the packet header, the *Type* of AdvData, *Length* of the type of AdvData, *Flags*, *Service Data*, *Service UUID*, *Manufacturer Specific*, *Appearance*, and *Power Level*. Their dataset consists of 23 BLE devices categorized into 7 types as shown in Figure 4.3. Their BLE packets are collected with software radio-based BLE receivers. In total, they have collected 48'623 packets and formed training, validation, and test datasets in the ratio of 7:2:1 of the data. Using an MLP model, their goal is to recognize and categorize BLE devices. They achieve an accuracy of 99.8% to correctly classify the BLE devices with precision, recall, and F1-score being 0.9903, 0.9890, and 0.9888 respectively. By also using an MLP model in this thesis but with a different dataset and extracting different features, it is possible to compare the classification accuracies of the model from the research published in [45] and the model in this thesis.



No.	Device type	Device manufacturer	Model	Number of devices
1	Smart band	Xiaomi	Mi Band 5	1
2	Smart band	Huawei	4-506	6
3	Smart band	Garmin	vivosmart3	1
4	Smart band	ADZ	A9	1
5	Smart band	Polar	A370	1
6	Smart watch	Garmin	010-02120-41	1
7	Smart watch	Xiaomi	Mi Color BCAA	1
8	Smart watch	Polar	IGNITE	1
9	Smart oximeter	Konsung	W01LT	1
10	Smart oximeter	Viatom	O2Ring	1
11	Smart phone	Apple	iPhone8/6s	2
12	PC	Apple	MBP 2020	1
13	Headphone	Sony	WH-1000XM3	2
14	Smart humidifier	Xiaomi	CJXJSQ02ZM	1
15	PC	Microsoft	ASUS A550L	2

Figure 4.3: Dataset of BLE Devices from Research published in [45]

The structure of the lightweight MLP model is the same as described in [45]. It consists of two hidden layers of 128 neurons each. It uses the Rectified Linear Unit (ReLU) activation function and the Softmax activation on the output layer. Dropout layers are added between the two hidden layers, and between the hidden layer and the output layer, with probability P set to 0.3 to avoid overfitting. The structure of the MLP model is shown in Figure 4.4. After removing unique identifiers from the dataset, the data is preprocessed by zero padding and normalization. This is done by converting the hexadecimal value to a numerical representation and dividing it by 16. Since the dataset in this thesis undergoes one-hot-encoding to be transformed into numerical representation, zero-padding is not needed and the dataset only undergoes an additional preprocessing step of dividing the values by 16 to achieve the same data structure as in [45]. Data splitting remains as described in Chapter 5.7 and the 7:2:1 ratio is not applied.

The MLP model is implemented using the TensorFlow library[9] and is trained using 50 epochs, a batch size of 32, and a validation dataset of 20% of the data. After each

[9] `https://www.tensorflow.org/guide/core/mlp_core`

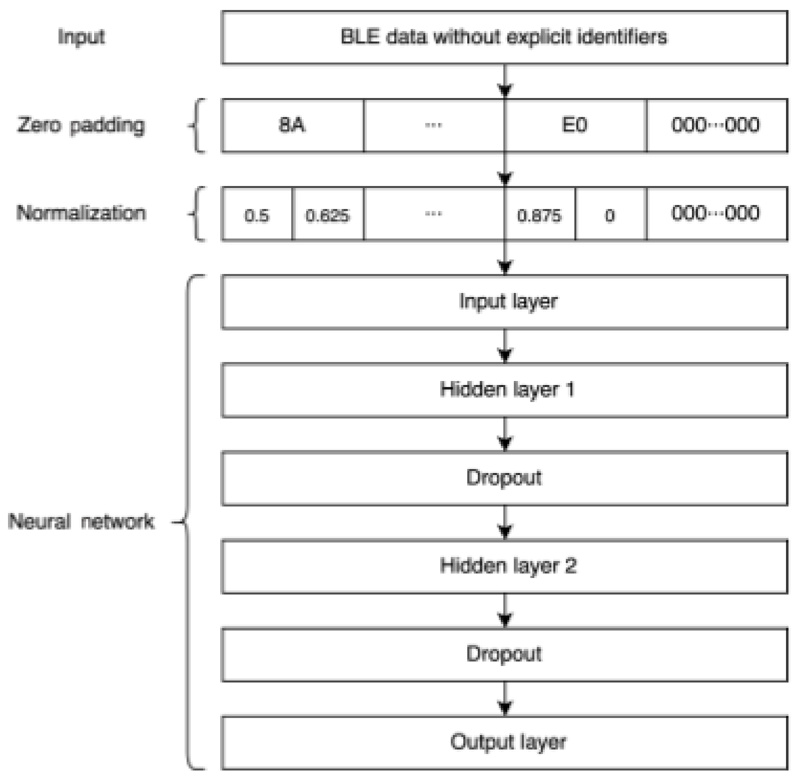


Figure 4.4: Structure of the MLP Model [45]

epoch, training accuracy, training loss, validation accuracy, and validation loss are computed. The loss measures how well the prediction made by the model matches the actual labels by calculating the difference between the predicted labels and the actual labels. A lower loss indicates a better performance of the model. The validation accuracy is the accuracy of the validation dataset and shows how well the model performs on unseen data. As already mentioned, overfitting occurs when the training accuracy increases at each epoch during training, but the validation accuracy decreases, and when the training loss decreases during training at each epoch and the validation loss increases. The error on the training data decreases but not on the validation data, indicating poor generalization.

4.9 Comparing ML Models

In summary, there are nine trained ML models:

1. RF classifier with 80/20 data splitting method
2. RF classifier with 80/20 data splitting method on selected features
3. RF classifier with alternative data splitting method
4. RF classifier with alternative data splitting method on selected features

5. MLP model with 80/20 data splitting method
6. MLP model with 80/20 data splitting method on selected features
7. MLP model with alternative data splitting method
8. MLP model with alternative data splitting method on selected features
9. MLP model from research published in [45]

Their performances are compared by looking at the difference in their accuracies. The model with the highest accuracies is declared to perform the best for the device classification task.

Chapter 5

Results and Evaluation

This chapter presents the results and evaluation of the proposed methodology. The sniffed BLE packets, the extracted features, and the ability of the models to correctly classify the packets are shown and discussed.

5.1 Device Collection

As mentioned in Chapter 4.1, the dataset consists of 49 devices, representing 10 different device types. There are 8 headphones, 6 smartphones, 6 smartwatches, 6 cameras, 5 laptops, 5 TVs, 4 iPads, 4 AirTags, 3 kitchen devices, and 2 bathroom devices as shown in Figure 5.1. Each device type consists of at least two devices. There are differences in the distribution of the devices within the device types. For example, the Headphone category is the most common, with a total of 8 devices, while the Bathroom category devices are the least common, with only 2 devices. The reason for this different distribution of devices is the accessibility and use of the devices in different circumstances. The fact that headphones are so widespread and commonplace explains their high number in the dataset. Being portable, they are easy to obtain. In addition, many people own more than one pair of headphones, which also increases the number of headphones. On the other hand, the reason for the lack of bathroom devices is that many people do not have BLE bathroom devices. This problem also applies to kitchen devices, where Bluetooth connectivity may not be as useful or desirable as for other devices.

There are limitations and potential biases in the device collection process. One limitation is the device selection criterion as only devices that are considered to be easily portable are collected. Less portable devices, such as security cameras or stationary devices, are excluded from this criterion. As a result, the dataset may not accurately represent the range of BLE devices. There is also an additional limitation when collecting devices through friends and acquaintances. Device categories that are less commonly owned or used by friends and acquaintances cannot be included due to the difficulty in accessing the devices.

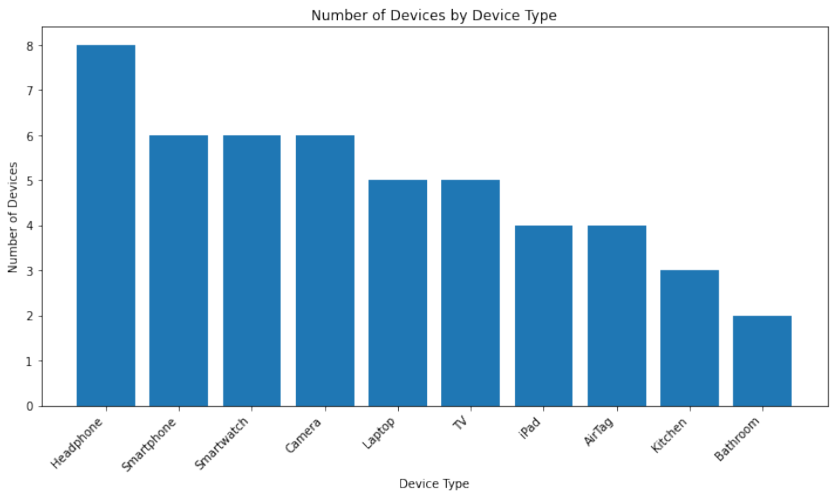


Figure 5.1: Number of Devices per Device Type

5.2 Data Collection via Passive Sniffing with nRF Tool

The packet data of BLE devices is passively sniffed for 20 minutes using the nRF Sniffer tool. Two approaches are taken: the first is to sniff all BLE devices in three public places, and the second is to sniff the 49 BLE devices.

5.2.1 Initial Capturing of BLE Packets in Public Places

BLE packet data is sniffed in three public places such as at the IFI, at home, and the train station. The packet details for packets sniffed at the IFI are shown in Figure 5.2. There is information about the nRF Sniffer tool and the BLE LL. The LL contains the most informative information such as the *Packet Header*, the *AdvA*, and the *AdvData* fields with information about flags, and manufacturer-specific data. Due to the strong privacy protection, the AdvA is mostly not public. As shown in Figure 5.2 the TxAdd is random which means that the AdvA is randomized. If the addresses are not randomized, the MAC address Lookup tool[1] could be used to determine the device type. Some BLE packets such as the one in Figure 5.2 contain information about the company ID. However, knowing that a packet is from an Apple device does not help to determine the device type. Furthermore, there are online databases with information about the packets and their device type, but they are not published.

One way to determine the device type is to filter the packets and keep only those packets where the *Device Name* field is not empty. By sniffing the packets at the IFI and home, and filtering for non-empty *Device Name* fields, these are the unique device names:

[1]https://dnschecker.org/mac-lookup.php

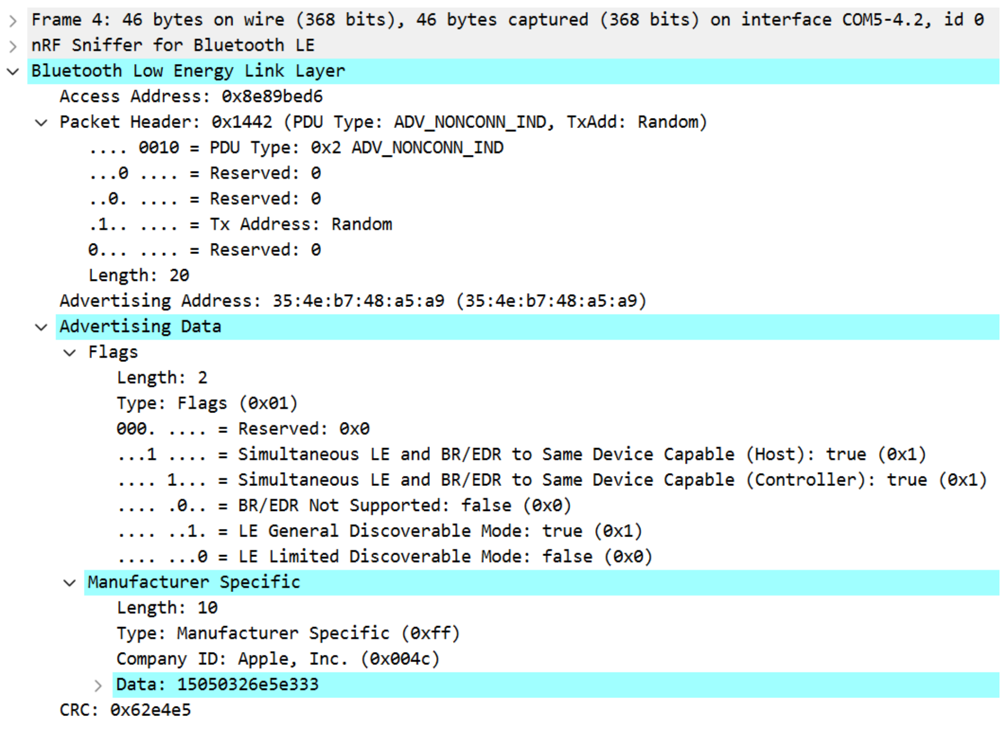

```
> Frame 4: 46 bytes on wire (368 bits), 46 bytes captured (368 bits) on interface COM5-4.2, id 0
> nRF Sniffer for Bluetooth LE
v Bluetooth Low Energy Link Layer
    Access Address: 0x8e89bed6
  v Packet Header: 0x1442 (PDU Type: ADV_NONCONN_IND, TxAdd: Random)
      .... 0010 = PDU Type: 0x2 ADV_NONCONN_IND
      ...0 .... = Reserved: 0
      ..0. .... = Reserved: 0
      .1.. .... = Tx Address: Random
      0... .... = Reserved: 0
      Length: 20
    Advertising Address: 35:4e:b7:48:a5:a9 (35:4e:b7:48:a5:a9)
  v Advertising Data
    v Flags
        Length: 2
        Type: Flags (0x01)
        000. .... = Reserved: 0x0
        ...1 .... = Simultaneous LE and BR/EDR to Same Device Capable (Host): true (0x1)
        .... 1... = Simultaneous LE and BR/EDR to Same Device Capable (Controller): true (0x1)
        .... .0.. = BR/EDR Not Supported: false (0x0)
        .... ..1. = LE General Discoverable Mode: true (0x1)
        .... ...0 = LE Limited Discoverable Mode: false (0x0)
    v Manufacturer Specific
        Length: 10
        Type: Manufacturer Specific (0xff)
        Company ID: Apple, Inc. (0x004c)
      > Data: 15050326e5e333
    CRC: 0x62e4e5
```

Figure 5.2: Packet Detail of Sniffed Packets

- Apple Pencil
- COTSUBU
- EDIFIER BLE
- JBL LIVE400BT-LE
- LE-Bose
- LE-LinkBuds
- LE_WF-C500
- LE_WH-1000XM3
- LE_WH-1000XM4
- MOMENTUM 4
- Q26 Pro(ID-F569)
- Venue-Tile
- Hue Lamp
- LE_WH-1000XM4
- Smart Tag
- VMAT

An online search of these device names can be used to determine the device type. The majority of known device names are packets of headphones. Other BLE packets with known device names belong to the Apple Pencil, Smartwatch, Lamp, or AirTag categories. However, some device names are not recognizable, such as *VMAT*, and their device type cannot be inferred. Calculating the percentage of packets with a known device name out of all sniffed packets, it is 3.16% at IFI and 4.88% at home. Only a tiny fraction of the sniffed data can be determined with the device type.

Another way to determine the device type is to take non-empty *Company ID* fields, which are the name of manufacturer, or non-empty *UUID* fields. Applying the filter for non-empty *Company ID* fields to the packets sniffed at the IFI and home will return these unique names of manufacturer:

- 3Com
- 3DiJoy Corporation
- 9Solutions Oy
- A & R Cambridge
- ACOS CO.,LTD.
- APT Ltd.
- Accel Semiconductor Ltd.
- Airtago
- Apple, Inc.
- Atheros Communications, Inc.
- Avago Technologies
- Barnacle Systems Inc.
- Beats Electronics
- Beautiful Enterprise Co., Ltd.
- Belkin International, Inc.
- Berner International LLC
- BlackBerry Limited
- Blue Clover Devices
- Broadcom Corporation
- C Technologies
- CONWISE Technology Corporation Ltd
- CORE Lighting Ltd
- Carol Cole Company
- Conexant Systems Inc.
- connectBlue AB
- Continental Automotive Systems
- Digianswer A/S
- Eclipse (HQ Espana) S.L.
- Eijkelkamp Soil & Water
- Elgato Systems GmbH
- Enflux Inc.
- Equinux AG
- Ericsson Technology Licensing
- For use in internal and interoperability tests
- Garmin International, Inc.
- General Motors
- Gimbal Inc.
- GimmiSys GmbH
- Group Sense Ltd.
- HM Electronics, Inc.
- Hewlett-Packard Company
- HUIZHOU DESAY SV AUTOMOTIVE CO., LTD.
- IBM Corp.
- Icom inc.
- Innovative Yachtter Solutions
- Intel Corp.
- Inventel
- John Deere
- KC Technology Inc.
- Life Laboratory Inc.
- LifeStyle Lock, LLC
- Lucent
- Lucimed
- Ludus Helsinki Ltd.
- M-Way Solutions GmbH
- Macronix International Co. Ltd.
- Marvell Technology Group Ltd.
- MediaTek, Inc.
- Mesh-Net Ltd
- Metanate Limited
- Microchip Technology Inc.
- Microsoft
- Microtronics Engineering GmbH
- Mitsubishi Electric Corporation
- Nanoleq AG
- Norwood Systems

- ORSO Inc.
- Otter Products, LLC
- Parthus Technologies Inc.
- Pepperl + Fuchs GmbH
- Plantronics, Inc.
- Qualcomm Technologies International, Ltd. (QTIL)
- Quintic Corp
- Renesas Electronics Corporation
- Revvo Technologies, Inc.
- Samsung Electronics Co. Ltd.
- Sam Labs Ltd.
- Seiko Epson Corporation
- Selfly BV
- Senix Corporation
- SiFli Technologies (shanghai) Inc.
- Socket Mobile
- Sony Corporation
- Staccato Communications, Inc.
- Summit Data Communications, Inc.
- TTPCom Limited
- TYRI Sweden AB
- Tangerine, Inc.
- Toshiba Corp.
- VEGA Grieshaber KG
- Vonkil Technologies Ltd
- Xiaomi Inc.
- Xtrava Inc.
- Zumtobel Group AG
- iopool s.a.

Applying the filter for non-empty *UUID* fields to the packets sniffed at the IFI and home will return these values:

- Amazon.com Services, Inc.
- Andreas Stihl AG & Co. KG
- Apple Inc.
- August Home Inc
- Ayla Networks
- Battery Service
- Blue Bite
- Bose Corporation
- CRESCO Wireless, Inc
- CSR
- Device Information
- Duracell U.S. Operations Inc.
- ERi, Inc.
- FTP
- Google
- Harman International
- Huawei Technologies Co., Ltd.
- Logitech International SA
- Nokia
- Philips Lighting B.V.
- Procter & Gamble
- Robert Bosch GmbH
- SENNHEISER electronic GmbH & Co. KG
- SMART INNOVATION Co., Ltd
- Samsung Electronics Co.
- Sonos, Inc.
- Sony Corporation
- TASER International, Inc.
- Telit Wireless Solutions GmbH
- Tile, Inc.

- Volkswagen AG
- alibaba
- ruwido austria gmbh

The majority of the names of manufacturer cannot be used to determine the device type as the device could cover several device types rather than one. For example, the device with the name of manufacturer "Apple" could be a smartphone, laptop, iPad, headphones, or an AirTag. Analogously, non-empty UUIDs cannot determine the device type as most UUIDs could cover multiple device types.

In summary, packets sniffed in public places cannot be uniquely categorized by device type. If they can be categorized by looking at the non-empty *Device Name* fields, only a tiny fraction of the sniffed packets can be categorized, and most will be headphones. For this reason, this approach to capturing BLE packets in public places is not useful for the classification task. Therefore, another approach is taken where BLE packets from known device types are sniffed.

5.2.2 Capturing of BLE Packets from Known Device Types

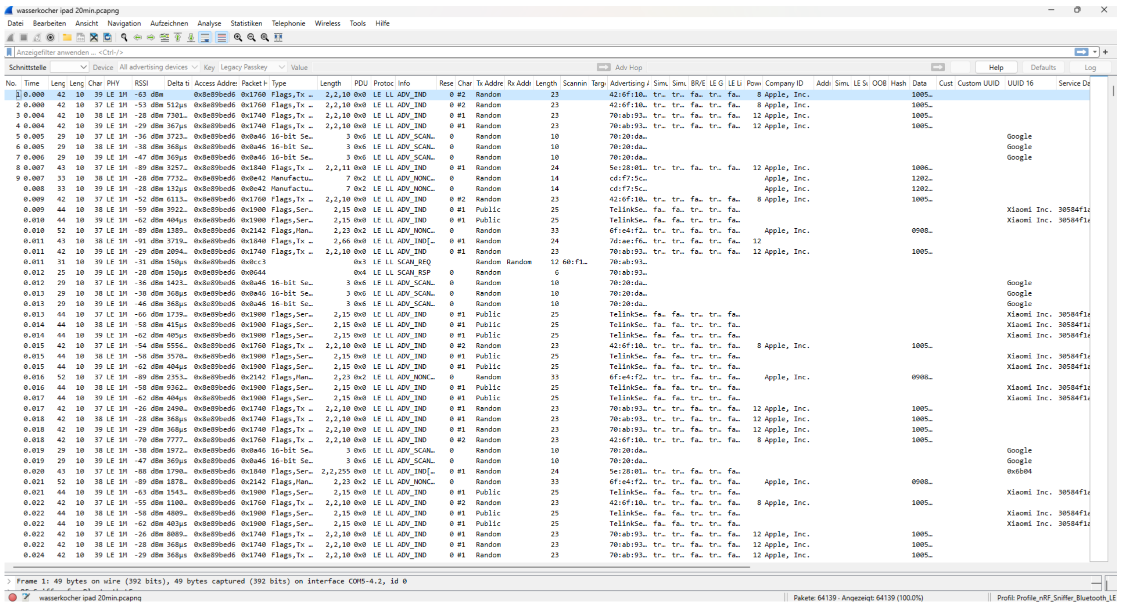
Since the packets sniffed in public places are not useful for the classification task, the approach is to capture BLE packets from known device types. The packets from the 49 devices already collected are sniffed in an isolated room to generate the dataset, except for the TVs and the AquaClean shower toilet device since they cannot be transported to that isolated room. Since they have unique device names, they are easily distinguishable from other sniffed BLE devices in the same location. Devices with the same name of manufacturer that belong to different device types are sniffed at different times and stored in different PCAP files. In addition, devices with no device name, company ID, or UUID information are sniffed separately and stored in separate PCAP files so that the type can still be determined. Devices whose packets contain device name, company ID, or UUID information can be sniffed together as long as they can be distinguished. As devices are brought by friends and acquaintances at different times and are immediately sniffed and returned, 22 PCAP files are stored, as shown in Table 5.1.

The PCAP file of sniffed packets from an Apple iPad and a Xiaomi kettle is visualized in Wireshark as shown in Figure 5.3. There are fields like *No., Time, Length of Payload, Length of Packet, Channel Index, PHY, RSSI, Delta Time, Access Address, Packet Header, Type, Length, PDU, Protocol, Info, Reserved, Channel Selection, Tx Address, RX Address, Length, Scanning Address, Target Address, AdvA, Simultaneous LE and BR/EDR to Same Device Capable (Host), Simultaneous LE and BR/EDR to Same Device Capable (Controller), BR/EDR Not Supported, LE General Discoverable Mode, LE Limited Discoverable Mode, Power Level, Company ID, Address Type, Simultaneous LE and BR/EDR to Same Device Capable (Host), LE Supported By Host, OOB Data Present, Hash C, Data, Custom UUID, UUID 16, Service Data, Device Name, BIG_Offset, BIG_Offset_Units, ISO_Interval, Num_BIS, NSE, BN, Sub_Interval, PTO, BIS_Spacing, IRC, Max PDU, Seed Access Address, BD_ADDR, SSP OOB Length, CRC*, and *Malformed*

No. of File	No. of Devices	Devices
1	1	Ericsson smartwatch
2	4	all Apple AirTags
3	3	all Apple iPads
4	5	2 HP laptops, Apple laptop, Bose-1 headphone, Samsung headphone
5	2	Apple smartwatch, Samsung TV
6	1	Google smartphone
7	1	Huawei smartwatch
8	3	Sony XM3 headphone, Sony XM4 headphone, Sony Linkbuds headphone
9	2	all Dell laptops
10	3	Nokia smartphone, Galaxy smartwatch, Samsung TV
11	2	Apple iPad, Xiaomi kettle
12	4	Samsung smartphone, Apple smartwatch, Oral-B tootbrush, Rollei camera
13	3	2 Apple smartphone, Huawei smartphone
14	2	Apple headphone, Fitbit smartwatch
15	4	Bose-2 headphone, Xiaomi mixer, Beats headphone, AquaClean shower toilet
16	2	Xiaomi airfryer, LG TV
17	2	LG TV, Samsung TV
18	1	GoPro9 camera
19	1	GoPro10 camera
20	1	GoPro5 camera
21	1	GoPro4 camera
22	1	Canon camera

Table 5.1: Stored PCAP Files of Sniffed BLE Packets

Packet. The *No.* field is the number of the packet sniffed and the *Time* field is the sniffing time when each packet is sniffed. Both are chronological. The *Length of Payload*, *Length of Packet*, *Channel Index*, *PHY*, *RSSI*, and *Delta Time* fields are from the nRF Sniffer data. The *Channel Index* field is either 37, 38, or 39 and indicates on which channel in the PHY layer the packets are sniffed. The *PHY* field is the PHY layer which has a value of LE 1M. The *RSSI* field indicates the signal strength. The *Delta Time* field is in microseconds and indicates the time taken to transmit the packet. The remaining fields are from the BLE LL. Detailed information is already described in Chapter 3.3.



```
No. Time  Leng Leng Char PHY  RSSI    Delta ti Access Addre Packet H Type        Length  PDU Protoc Info        Rese Char Tx Addre Rx Addr Length Scannin Targ Advertising A Simu Simu BR/E LE G LE Li Pow Company ID   Addi Simu LE Su OOB Hash Data  Cust Custom UUID UUID 16      Service Da
1   0.000  42  10   39 LE 1M -63 dBm          0x8e89bed6 0x1760 Flags,Tx …   2,2,10 0x0 LE LL ADV_IND      0 #2 Random              23           42:6f:10…   tr…  tr…  fa…  tr…  fa…   8 Apple, Inc.                       1005…
2   0.000  42  10   37 LE 1M -53 dBm 512µs    0x8e89bed6 0x1760 Flags,Tx …   2,2,10 0x0 LE LL ADV_IND      0 #2 Random              23           42:6f:10…   tr…  tr…  fa…  tr…  fa…   8 Apple, Inc.                       1005…
3   0.004  42  10   38 LE 1M -28 dBm 7301…    0x8e89bed6 0x1740 Flags,Tx …   2,2,10 0x0 LE LL ADV_IND      0 #1 Random              23           70:ab:93…   tr…  tr…  fa…  tr…  fa…  12 Apple, Inc.                       1005…
4   0.004  42  10   39 LE 1M -29 dBm 367µs    0x8e89bed6 0x1740 Flags,Tx …   2,2,10 0x0 LE LL ADV_IND      0 #1 Random              23           70:ab:93…   tr…  tr…  fa…  tr…  fa…  12 Apple, Inc.                       1005…
5   0.005  29  10   37 LE 1M -36 dBm 3723…    0x8e89bed6 0x0a46 16-bit Se…        3 0x6 LE LL ADV_SCAN…    0    Random              10           70:20:da…                                                                                  Google
6   0.005  29  10   38 LE 1M -38 dBm 368µs    0x8e89bed6 0x0a46 16-bit Se…        3 0x6 LE LL ADV_SCAN…    0    Random              10           70:20:da…                                                                                  Google
7   0.006  29  10   39 LE 1M -47 dBm 369µs    0x8e89bed6 0x0a46 16-bit Se…        3 0x6 LE LL ADV_SCAN…    0    Random              10           70:20:da…                                                                                  Google
8   0.007  43  10   37 LE 1M -89 dBm 3257…    0x8e89bed6 0x1840 Flags,Tx …   2,2,11 0x0 LE LL ADV_IND      0 #1 Random              24           5e:28:01…   tr…  tr…  fa…  tr…  fa…  12 Apple, Inc.                       1006…
9   0.007  33  10   38 LE 1M -28 dBm 7732…    0x8e89bed6 0x0e42 Manufactu…        7 0x2 LE LL ADV_NONC…    0    Random              14           cd:f7:5c…                                   Apple, Inc.                       1202…
    0.008  33  10   39 LE 1M -28 dBm 132µs    0x8e89bed6 0x0e42 Manufactu…        7 0x2 LE LL ADV_NONC…    0    Random              14           cd:f7:5c…                                   Apple, Inc.                       1202…
    0.009  42  10   37 LE 1M -52 dBm 6113…    0x8e89bed6 0x1760 Flags,Tx …   2,2,10 0x0 LE LL ADV_IND      0 #2 Random              23           42:6f:10…   tr…  tr…  fa…  tr…  fa…   8 Apple, Inc.                       1005…
    0.009  44  10   38 LE 1M -59 dBm 3922…    0x8e89bed6 0x1900 Flags,Ser…     2,15 0x0 LE LL ADV_IND      0 #1 Public              25           TelinkSe…   fa…  fa…  tr…  tr…  fa…                                                        Xiaomi Inc.  30584f1a
    0.010  44  10   39 LE 1M -62 dBm 404µs    0x8e89bed6 0x1900 Flags,Ser…     2,15 0x0 LE LL ADV_IND      0 #1 Public              25           TelinkSe…   fa…  fa…  tr…  tr…  fa…                                                        Xiaomi Inc.  30584f1a
    0.010  52  10   37 LE 1M -89 dBm 1389…    0x8e89bed6 0x2142 Flags,Man…     2,23 0x2 LE LL ADV_NONC…    0    Random              33           6f:e4:f2…   tr…  tr…  fa…  tr…  fa…     Apple, Inc.                       0908…
    0.011  43  10   38 LE 1M -91 dBm 3719…    0x8e89bed6 0x1840 Flags,Tx …     2,66 0x0 LE LL ADV_IND[…    0 #1 Random              24           7d:ae:f6…   tr…  tr…  fa…  tr…  fa…  12
    0.011  42  10   39 LE 1M -29 dBm 2094…    0x8e89bed6 0x1740 Flags,Tx …   2,2,10 0x0 LE LL ADV_IND      0 #1 Random              23           70:ab:93…   tr…  tr…  fa…  tr…  fa…  12 Apple, Inc.                       1005…
    0.011  31  10   39 LE 1M -31 dBm 150µs    0x8e89bed6 0x0cc3                        0x3 LE LL SCAN_REQ           Random   Random     12 60:f1…      70:ab:93…
    0.012  25  10   39 LE 1M -28 dBm 150µs    0x8e89bed6 0x0644                        0x4 LE LL SCAN_RSP      0    Random               6           70:ab:93…
    0.012  29  10   37 LE 1M -36 dBm 1423…    0x8e89bed6 0x0a46 16-bit Se…        3 0x6 LE LL ADV_SCAN…    0    Random              10           70:20:da…                                                                                  Google
    0.013  29  10   38 LE 1M -38 dBm 368µs    0x8e89bed6 0x0a46 16-bit Se…        3 0x6 LE LL ADV_SCAN…    0    Random              10           70:20:da…                                                                                  Google
    0.013  29  10   39 LE 1M -46 dBm 368µs    0x8e89bed6 0x0a46 16-bit Se…        3 0x6 LE LL ADV_SCAN…    0    Random              10           70:20:da…                                                                                  Google
    0.013  44  10   37 LE 1M -66 dBm 1739…    0x8e89bed6 0x1900 Flags,Ser…     2,15 0x0 LE LL ADV_IND      0 #1 Public              25           TelinkSe…   fa…  fa…  tr…  tr…  fa…                                                        Xiaomi Inc.  30584f1a
    0.014  44  10   38 LE 1M -58 dBm 415µs    0x8e89bed6 0x1900 Flags,Ser…     2,15 0x0 LE LL ADV_IND      0 #1 Public              25           TelinkSe…   fa…  fa…  tr…  tr…  fa…                                                        Xiaomi Inc.  30584f1a
    0.014  44  10   39 LE 1M -62 dBm 405µs    0x8e89bed6 0x1900 Flags,Ser…     2,15 0x0 LE LL ADV_IND      0 #1 Public              25           TelinkSe…   fa…  fa…  tr…  tr…  fa…                                                        Xiaomi Inc.  30584f1a
    0.015  42  10   37 LE 1M -54 dBm 5556…    0x8e89bed6 0x1760 Flags,Tx …   2,2,10 0x0 LE LL ADV_IND      0 #2 Random              23           42:6f:10…   tr…  tr…  fa…  tr…  fa…   8 Apple, Inc.                       1005…
    0.015  44  10   38 LE 1M -58 dBm 3570…    0x8e89bed6 0x1900 Flags,Ser…     2,15 0x0 LE LL ADV_IND      0 #1 Public              25           TelinkSe…   fa…  fa…  tr…  tr…  fa…                                                        Xiaomi Inc.  30584f1a
    0.015  44  10   39 LE 1M -62 dBm 404µs    0x8e89bed6 0x1900 Flags,Ser…     2,15 0x0 LE LL ADV_IND      0 #1 Public              25           TelinkSe…   fa…  fa…  tr…  tr…  fa…                                                        Xiaomi Inc.  30584f1a
    0.016  52  10   37 LE 1M -89 dBm 2353…    0x8e89bed6 0x2142 Flags,Man…     2,23 0x2 LE LL ADV_NONC…    0    Random              33           6f:e4:f2…   tr…  tr…  fa…  tr…  fa…     Apple, Inc.                       0908…
    0.016  44  10   38 LE 1M -58 dBm 9362…    0x8e89bed6 0x1900 Flags,Ser…     2,15 0x0 LE LL ADV_IND      0 #1 Public              25           TelinkSe…   fa…  fa…  tr…  tr…  fa…                                                        Xiaomi Inc.  30584f1a
    0.017  44  10   39 LE 1M -62 dBm 404µs    0x8e89bed6 0x1900 Flags,Ser…     2,15 0x0 LE LL ADV_IND      0 #1 Public              25           TelinkSe…   fa…  fa…  tr…  tr…  fa…                                                        Xiaomi Inc.  30584f1a
    0.017  42  10   37 LE 1M -26 dBm 2490…    0x8e89bed6 0x1740 Flags,Tx …   2,2,10 0x0 LE LL ADV_IND      0 #1 Random              23           70:ab:93…   tr…  tr…  fa…  tr…  fa…  12 Apple, Inc.                       1005…
    0.018  42  10   38 LE 1M -28 dBm 368µs    0x8e89bed6 0x1740 Flags,Tx …   2,2,10 0x0 LE LL ADV_IND      0 #1 Random              23           70:ab:93…   tr…  tr…  fa…  tr…  fa…  12 Apple, Inc.                       1005…
    0.018  42  10   39 LE 1M -29 dBm 368µs    0x8e89bed6 0x1740 Flags,Tx …   2,2,10 0x0 LE LL ADV_IND      0 #1 Random              23           70:ab:93…   tr…  tr…  fa…  tr…  fa…  12 Apple, Inc.                       1005…
    0.018  42  10   37 LE 1M -70 dBm 7777…    0x8e89bed6 0x1760 Flags,Tx …   2,2,10 0x0 LE LL ADV_IND      0 #2 Random              23           42:6f:10…   tr…  tr…  fa…  tr…  fa…   8 Apple, Inc.                       1005…
    0.019  29  10   38 LE 1M -38 dBm 1972…    0x8e89bed6 0x0a46 16-bit Se…        3 0x6 LE LL ADV_SCAN…    0    Random              10           70:20:da…                                                                                  Google
    0.019  29  10   39 LE 1M -47 dBm 369µs    0x8e89bed6 0x0a46 16-bit Se…        3 0x6 LE LL ADV_SCAN…    0    Random              10           70:20:da…                                                                                  Google
    0.020  43  10   37 LE 1M -88 dBm 1790…    0x8e89bed6 0x1840 Flags,Ser…  2,2,255 0x0 LE LL ADV_IND[…    0 #1 Random              24           5e:28:01…   tr…  tr…  fa…  tr…  fa…                                                        0x6b04
    0.021  52  10   38 LE 1M -89 dBm 1878…    0x8e89bed6 0x2142 Flags,Man…     2,23 0x2 LE LL ADV_NONC…    0    Random              33           6f:e4:f2…   tr…  tr…  fa…  tr…  fa…     Apple, Inc.                       0908…
    0.021  44  10   39 LE 1M -63 dBm 1543…    0x8e89bed6 0x1900 Flags,Ser…     2,15 0x0 LE LL ADV_IND      0 #1 Public              25           TelinkSe…   fa…  fa…  tr…  tr…  fa…                                                        Xiaomi Inc.  30584f1a
    0.022  42  10   37 LE 1M -55 dBm 1100…    0x8e89bed6 0x1760 Flags,Tx …   2,2,10 0x0 LE LL ADV_IND      0 #2 Random              23           42:6f:10…   tr…  tr…  fa…  tr…  fa…   8 Apple, Inc.                       1005…
    0.022  44  10   38 LE 1M -58 dBm 4809…    0x8e89bed6 0x1900 Flags,Ser…     2,15 0x0 LE LL ADV_IND      0 #1 Public              25           TelinkSe…   fa…  fa…  tr…  tr…  fa…                                                        Xiaomi Inc.  30584f1a
    0.022  44  10   39 LE 1M -62 dBm 403µs    0x8e89bed6 0x1900 Flags,Ser…     2,15 0x0 LE LL ADV_IND      0 #1 Public              25           TelinkSe…   fa…  fa…  tr…  tr…  fa…                                                        Xiaomi Inc.  30584f1a
    0.022  42  10   37 LE 1M -26 dBm 8089…    0x8e89bed6 0x1740 Flags,Tx …   2,2,10 0x0 LE LL ADV_IND      0 #1 Random              23           70:ab:93…   tr…  tr…  fa…  tr…  fa…  12 Apple, Inc.                       1005…
    0.022  42  10   38 LE 1M -28 dBm 368µs    0x8e89bed6 0x1740 Flags,Tx …   2,2,10 0x0 LE LL ADV_IND      0 #1 Random              23           70:ab:93…   tr…  tr…  fa…  tr…  fa…  12 Apple, Inc.                       1005…
    0.024  42  10   39 LE 1M -29 dBm 368µs    0x8e89bed6 0x1740 Flags,Tx …   2,2,10 0x0 LE LL ADV_IND      0 #1 Random              23           70:ab:93…   tr…  tr…  fa…  tr…  fa…  12 Apple, Inc.                       1005…

> Frame 1: 49 bytes on wire (392 bits), 49 bytes captured (392 bits) on interface COM5-4.2, id 0

wasserkocher ipad 20min.pcapng        Pakete: 64139 · Angezeigt: 64139 (100.0%)        Profil: Profile_nRF_Sniffer_Bluetooth_LE
```

Figure 5.3: Sniffed Packets from an Apple iPad and a Xiaomi Kettle visualized in Wireshark

5.3 Dataset Labeling

The PCAP files are converted to CSV files. The device type of the packets sniffed is determined by at least one of the fields such as *Device Name*, *Company ID,* and *UUID*. Therefore, these fields are analyzed in every single file. The values of these fields, which determine the device type, are listed in Table 5.2. For example, the Ericsson smartwatch is determined as a smartwatch by the value "Ericcson Technology Licensing" in the *Company ID* field or by the value "UM59 (ID-2663)" in the *Device Name* field. By implementing conditions that match the values as shown in Algorithm 1, each packet in the files is labeled. The label is the device type and is added to the CSV files as a new column called *label*. In addition, a second label is added as a new column called *subcategory*, which is the name of manufacturer.

Figure 5.4 shows the total and average number of packets for each device type. Because headphone devices are most commonly sniffed, they have the largest total

Device	**Field(s)**	**Value of the Field(s)**
Ericsson smartwatch	Company ID Device Name	Ericsson Technology Licensing UM59 (ID-2663)
Apple AirTags	Company ID	Apple, Inc.
Apple iPads	Company ID	Apple, Inc.
HP laptop	Company ID	Microsoft
Apple laptop	Company ID	Apple, Inc.
Bose-1 headphone	UUID	Bose Corporation
Samsung headphone	Company ID	Samsung Electronics Co. Ltd.
Apple smartwatch	Company ID	Apple, Inc.
Samsung TV	Device Name	[TV] Samsung Q60BA 65 TV Mind
Google smartphone	UUID	Google
Huawei smartwatch	Company ID	HUAWEI Technologies Co., Ltd.
Sony XM3 headphone	Device Name	LE_WH1000XM3
Sony XM4 headphone	Device Name	LE_WH1000XM4
Sony Linkbuds headphone	Device Name	LE_LinkBuds S
Dell laptop	Company ID	Microsoft
Nokia smartphone	Company ID	Nokia Mobile Phones
Galaxy smartwatch	Device Name	Galaxy Watch4 (5FHX)
Samsung TV	Device Name	[TV] Samsung Q80AA 50 TV
Xiaomi kettle	UUID	Xiaomi Inc.
Samsung smartphone	Company ID	Samsung Electronics Co. Ltd.
Oral-B toothbrush	Device Name	Oral-B
Rollei camera	Device Name	Rollei SmartBT
Apple smartphone	Company ID	Apple, Inc.
Huawei smartphone	UUID	Huawei Technologies Co., Ltd.
Apple headphone	Company ID	Apple, Inc.
Fitbit smartwatch	Device Name	Charge 2
Bose-2 headphone	Device Name	LE-Moonbeam
Xiaomi mixer	UUID	Xiaomi Inc.
Beats headphone	Company ID	Beats Electronics
AquaClean shower toilet	Device Name	Geberit AC PRO
Xiaomi airfryer	UUID	Xiaomi Inc.
LG TV	Device Name	[LG] webOS TV UP77009LB
LG TV	Device Name	[LG] webOS TV OLED55C17LB
Samsung TV	Device Name	[TV] Samsung 7 Series (55) Ba
GoPro4 camera	Company ID UUID	GoPro, Inc. GoPro, Inc.
GoPro5 camera	Company ID UUID	GoPro, Inc. GoPro, Inc.
GoPro9 camera	Company ID Device Name UUID	GoPro, Inc. GoPro 3267 GoPro, Inc.
GoPro10 camera	Company ID UUID	GoPro, Inc. GoPro, Inc.
Canon camera	Company ID Device Name	Canon Inc. EOSM50

Table 5.2: BLE devices and their Identifiers

number of packets. By considering the average number of packets per device type, bathroom devices have the highest average amount of packets compared to other devices, namely 65'402 packets. Laptops, cameras, smartwatches, smartphones, iPads, and kitchen devices have similar average packet counts, ranging between 12'000 and 28'000 packets. TVs and AirTags are the devices with the lowest average packet counts, with AirTags being the least. In total, there are 901'623 packets. The difference in the number of packets is related to the different uses of the devices. Even though headphones are widely used and actively transmit data, their average number of packets is not as high as bathroom devices. This indicates that bathroom devices must transmit data frequently for functions such as status updates or ongoing tracking. The low number of packets from the AirTags and TVs is because they transmit data infrequently during the sniffing process.

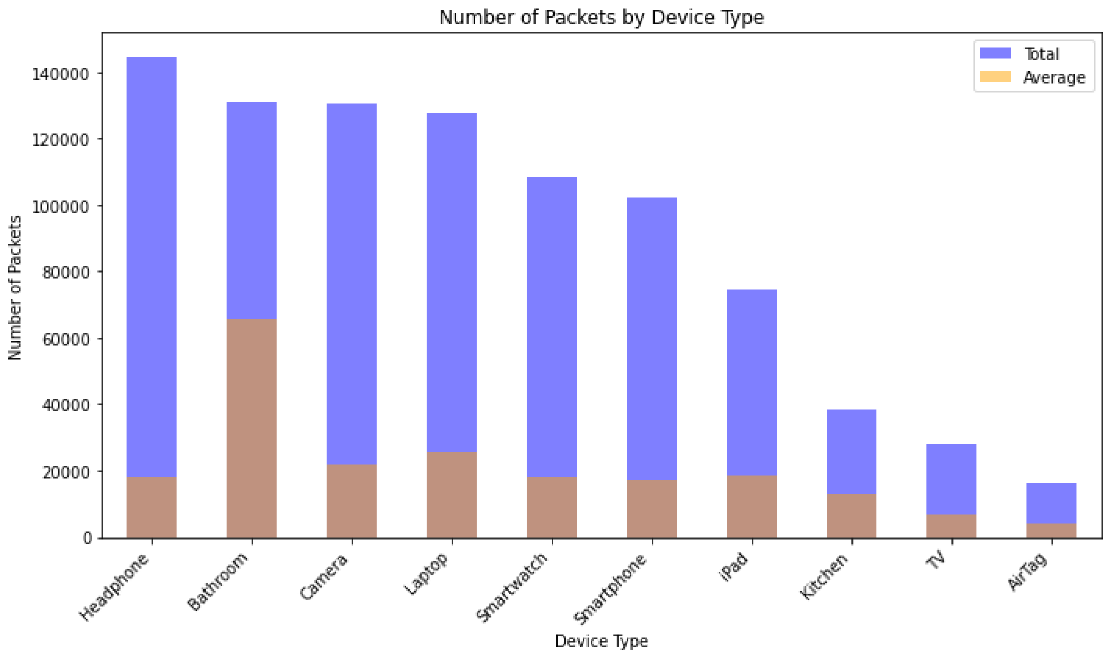


Figure 5.4: Number of Packets by Device Types

Figure 5.5 also shows the number of packets but by devices rather than by the device types. It can be seen that the toothbrush device contains the highest number of packets with 129'761 packets. As mentioned before, the toothbrush has to transmit data continuously while it is being used to update the status of the dental cleaning in real-time. The Rollei camera, Apple headphones, Sony Linkbuds headphones, Apple smartwatch, and Canon camera each transmit around 40'000 packets. This group of devices, along with the toothbrush device, has the highest number of packets. This indicates that these devices require frequent data transmission, possibly for ongoing data synchronization. The Nokia smartphone, Sony XM4 headphone, and Dell laptop each have around 30'000 packets. This amount of data transfer may be standard for devices such as laptops and smartphones in normal use, or for headphones when listening to music which need to transmit data regularly but not continuously. The kettle and the HP laptop each transmit around 20'000 packets and the Huawei smartphone, Apple iPad, Ericsson smartwatch, Google smartphone, Samsung smartphone, GoPro9 camera, Sony XM3 headphone, and GoPro10 camera each transmit around 10'000 packets. These devices show lower BLE activity. All other devices that are not mentioned transmit fewer than 10'000 packets each. The Beats headphones with only 16 packets and the Galaxy smartwatch with only 46 packets are the devices that transmit the least. This may be because these devices are used less frequently during sniffing or have a lower need for continuous transmission of packets.

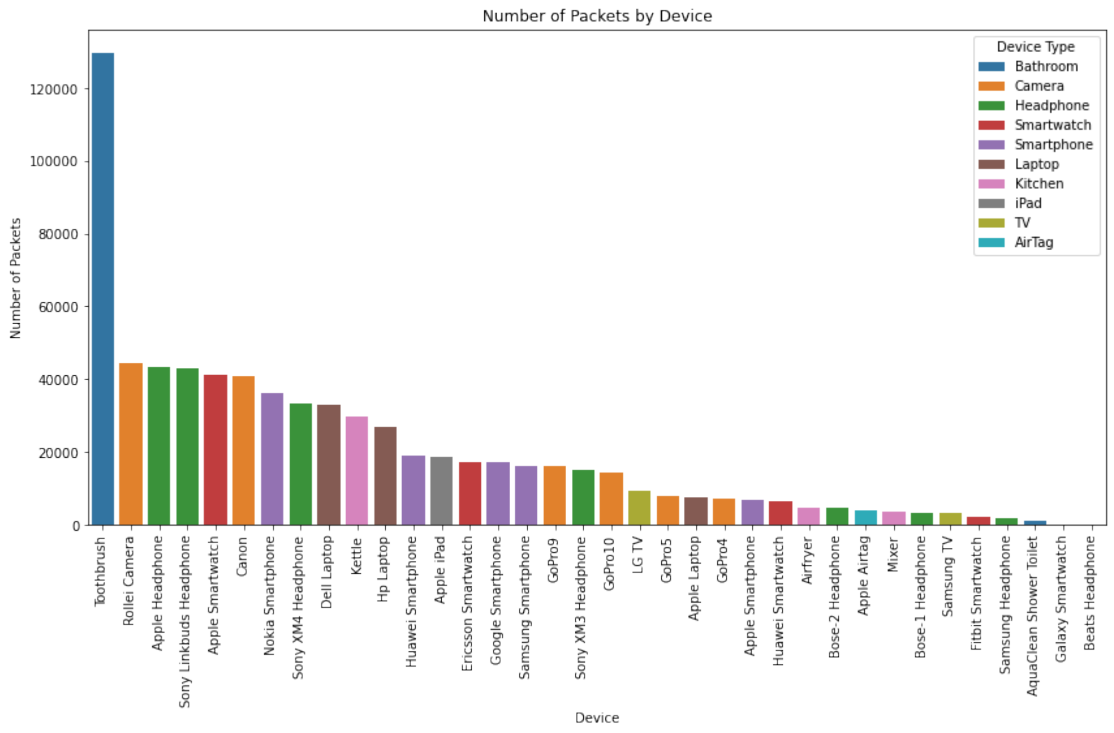


Figure 5.5: Number of Packets by Devices

By excluding the Beats headphones, Galaxy smartwatch, and Oral-B toothbrush from the data, Figure 5.6 provides a clearer view of the packet distribution among the remaining devices. It is noticeable that devices of the same type, such as headphones, have quite different values. For example, there are more than 40'000 packets for Apple headphones but less than 10'000 for Bose and Samsung headphones. Even devices of the same type can have different numbers of packets. Some of them may be actively used during the sniffing process and therefore transmit a large number of packets, while others are not actively used and therefore transmit a small number of packets. This shows a limitation of capturing BLE packets as devices can have different numbers of packets depending on how they are used in the sniffing process.

5.4 Feature Extraction

The main features are extracted from the sniffed BLE packets. First, features that are directly accessible from the sniffed data are extracted. Secondly, additional features are extracted as derived features by performing statistical calculations.

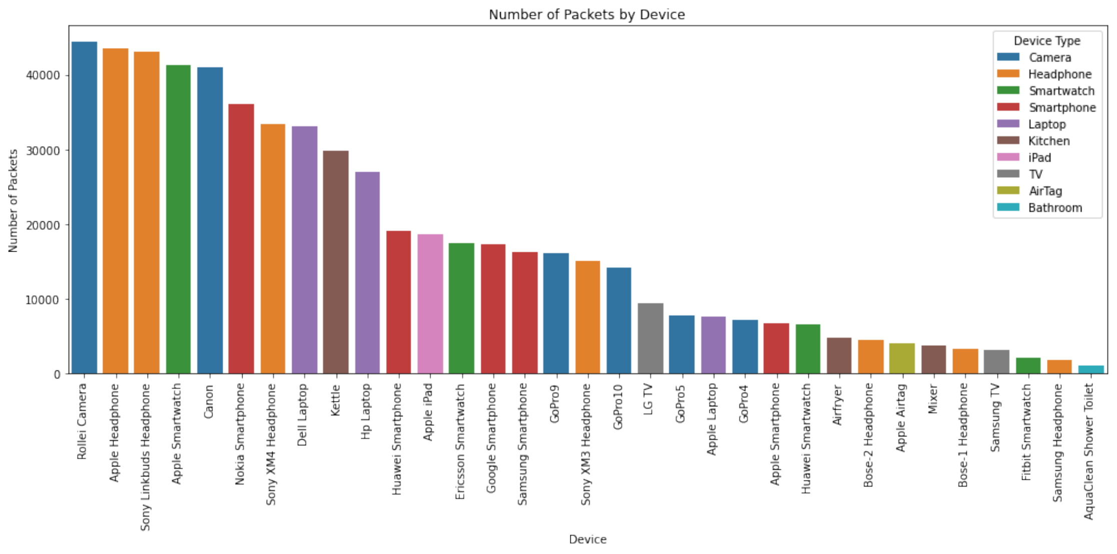


Figure 5.6: Number of Packets by Devices without Outliers

5.4.1 Direct Extraction of Feature from Sniffed BLE Packets

As already mentioned in Chapter 5.2.2 there are columns in the CSV files with column header names such as *Length of Payload, Length of Packet, Channel Index, PHY, RSSI, Delta Time, Access Address, Packet Header, Type, Length, PDU, Protocol, Info, Reserved, Channel Selection, Tx Address, RX Address, Length, Scanning Address, Target Address*, and *AdvA*. The data of these column header names are data from the nRF Sniffer and the BLE LL. Due to the data preprocessing step described in the next chapter, some of these column header names are removed. The remaining column header names are the *Length of Payload, Channel Index, RSSI, Delta Time, Packet Header, Type, Length, PDU, Info, Reserved, Channel Selection, Tx Address, RX Address, Length, Simultaneous LE and BR/EDR to Same Device Capable (Controller), BR/EDR Not Supported, LE General Discoverable Mode, LE Limited Discoverable Mode, Power Level, Simultaneous LE and BR/EDR to Same Device Capable (Host)*, and *LE Supported By Host.* These are the extracted features that are specific to device types and therefore important for the ML model to learn the pattern of device types. Some of them are shown in the subchapter and some of them are shown in Appendix E due to their lower importance for the classification task.

Length of Payload

The range of the *Length of Payload* feature is between 25 and 56 as shown in Figure 5.7. The y-axis is the number of packets. The device types are shown in different colors. AirTags have a length that varies from 25 to 56, showing that the AirTags data can vary in size or structure. On the other hand, bathroom devices transmit most of their packets with a length of 43, showing that the data of bathroom devices is almost a standard size. Cameras have a length of 31, 41, 42 48, 49, 52, and 56. Headphones, smartphones, and smartwatches have lengths ranging from 25 to 56, showing common characteristics. The majority of packets transmitted by kitchen devices have a length of 44. Laptops have lengths of 33, 39, 42, 43, 44, and 56, with 56 being the most common length. TVs have lengths of 31, 53, 55, and 56. The *Length of Payload* value with the most packets for TV is 55. Lastly, iPads have lengths of 25, 31, 33, 39, 42, 43, 44, 51, and 56. The variation in the *Length of Payload* feature indicates that devices of the same device type have variations in the size or structure of the data packets. By learning this pattern of variation within the device type, the ML model can predict the device type.

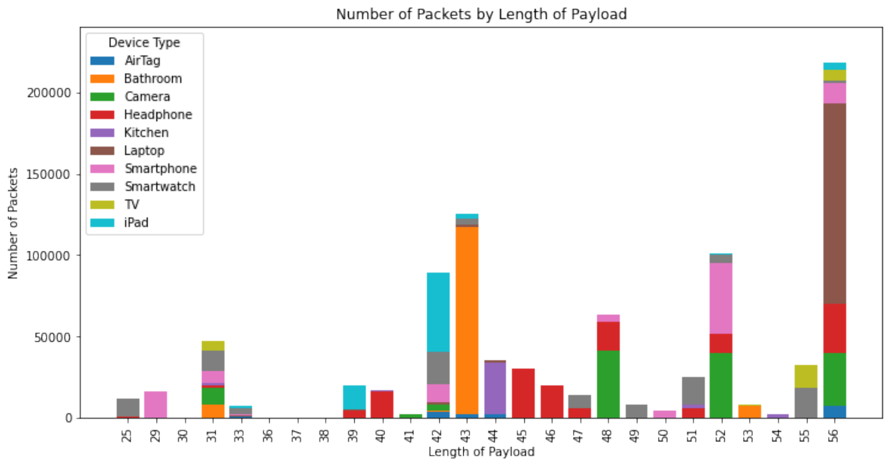


Figure 5.7: Number of Packets by *Length of Payload* Feature

Channel Index

The *Channel Index* feature has a value of either 37, 38, or 39 as shown in Figure 5.8. As can be seen, the values of the *Channel Index* feature for each type are evenly distributed across all 3 values. This shows that this feature does not help to characterize device types as all devices have a similar pattern.

RSSI

The strength of the signal received from different BLE devices is measured by the RSSI values, which vary from -93 dBm to -17 dBm as shown in Figure 5.9. Stronger signals are indicated by higher values, while lower values indicate weaker signals. IPads, laptops, and headphones have high RSSI values compared to others, while TVs have low RSSI values indicating that TVs have lower signal strength. The reason for the low signal strength could be that these devices have limited transmitting power. Kitchen and bathroom devices are in the middle range. Smartphones, cameras, and smartwatches have values in almost all x-values. This broad distribution means that the signals from these devices can be both strong and weak. By additionally learning the pattern of RSSI values of the device types, the model

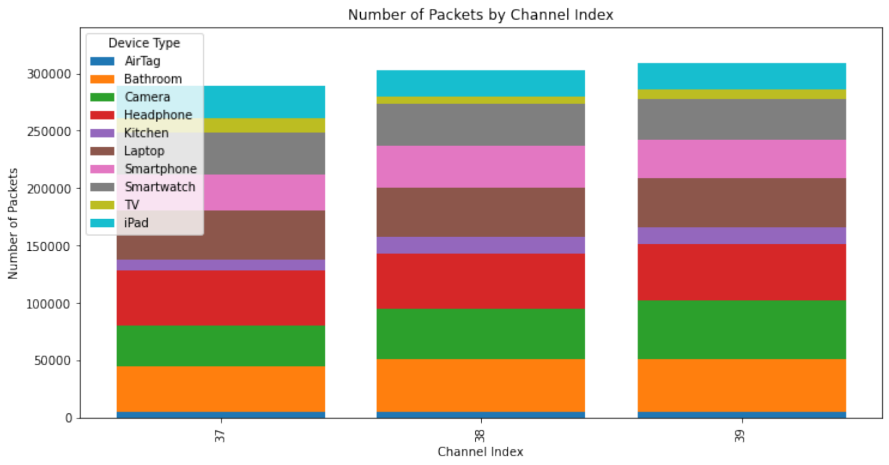


Figure 5.8: Number of Packets by *Channel Index* Feature

can make more accurate and reliable predictions about the device type because the pattern of RSSI values provides additional information about the device types.

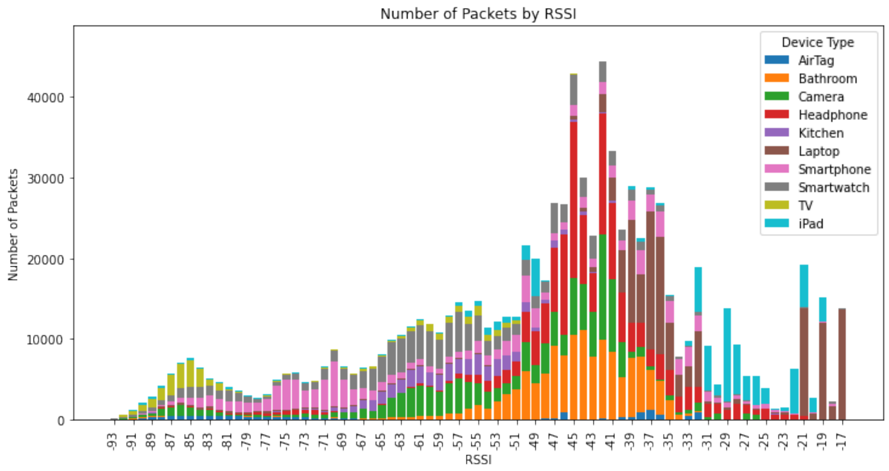


Figure 5.9: Number of Packets by *RSSI* Feature

Delta Time

The *Delta Time* feature is the time taken to transmit a packet in microseconds. As shown in Figure 5.10, the values range from 132 to 502 microseconds across all device types. A short *Delta Time* value indicates a fast transmission of the packets due to small packets or more power of the device. A value of 188 microseconds is unique to the Laptop category, a value of 384 microseconds to the Smartwatch category, a value of 455 to the Camera category, and a value of 502 microseconds to the Headphone category. With a value of 502 microseconds, the Headphone category has the highest value. This indicates that headphones need more time to transmit packets due to the transmission of audio data. This uniqueness is a characteristic of the device types and can be used for ML models to determine the device type. Also here by additionally adding the *Delta Time* as a feature, the model can make more accurate predictions.

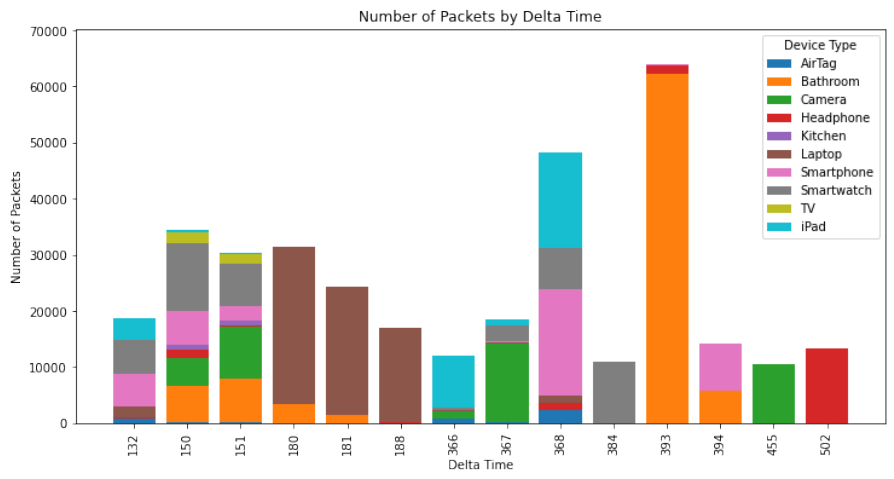


Figure 5.10: Number of Packets by *Delta Time* Feature

Info

The *Info* feature is the PDU type of AD. It includes values such as ADV_IND, ADV_NONCONN_IND, ADV_SCAN_IND, SCAN_REQ, SCAN_RSP as shown in Figure 5.11. The majority of packets have a value of ADV_IND indicating that this PDU type is used by the majority of devices. The majority of laptops have a type of ADV_NONCONN_IND meaning that they are not accepting a connection. Some headphones, smartphones, smartwatches, and iPads also have the same type. It is difficult to determine which device type corresponds to which specific type as all device types have at least three out of five possible values. This shows that this feature does not help to characterize device types. Furthermore, this variation shows that when the devices communicate with other devices, they send or receive packets, or send a request or a response.

Length

The *Length* feature is the length of the packet header and is shown in Figure 5.12. The x-axis ranges from 6 to 37. The data is spread across most device types, although lengths of 21, 26, and 27 are specific to the Headphone category, while a length of 10 is specific to the Smartphone category. Bathroom devices have lengths of 12 or 24, with most packets having a length of 24. The kitchen devices have a length of 25, while most laptops have a length of 37. TV devices have lengths of 12, 36, or 37. The lengths of the remaining devices are distributed along the x-axis. This distribution shows that some device types have a broad range of the *Length* feature, while others have specific lengths. This variation is due to the communication protocols that are specific to each device type. These specific length values are crucial for the ML model to learn the pattern in the data as they

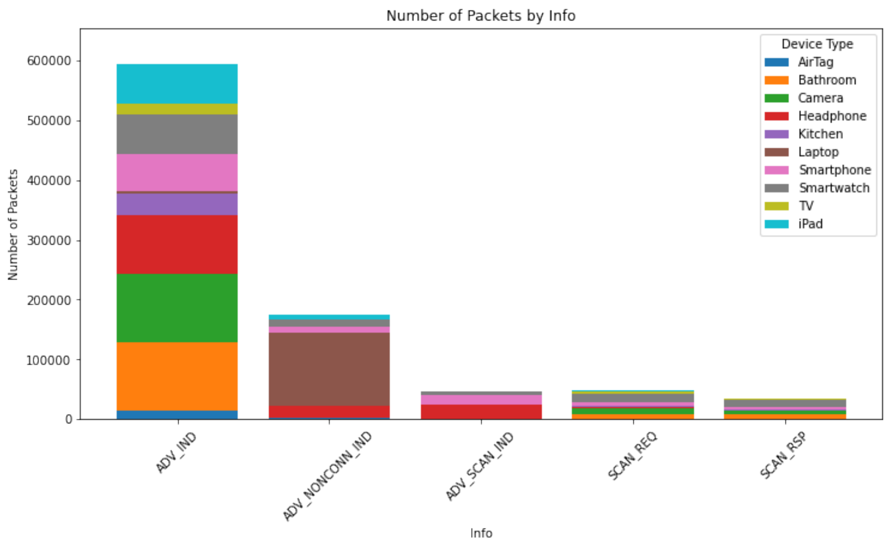


Figure 5.11: Number of Packets by *Info* Feature

are specific to a device type. By training the ML model with additional features, each feature provides unique insight into distinguishing between device types.

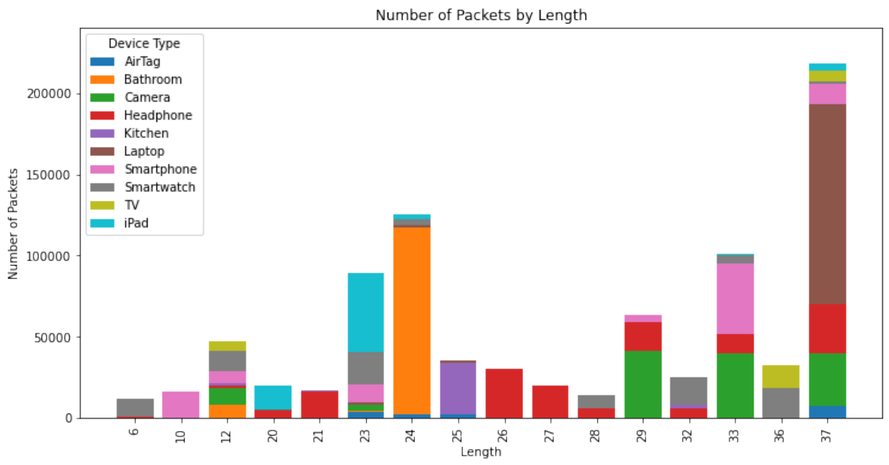


Figure 5.12: Number of Packets by *Length* Feature

Power Level

The *Power Level* feature measures the power intensity of a device, ranging from -44 to 17 dBm, as shown in Figure 5.13. Most of the *Power Level* feature values are specific to some device types. For example, a value of -44 dBm corresponds to TVs, values of -21 and -7 dBm to headphones, and a value of 0 dBm to bathroom devices. The other values are shared between device types. The value of 6 dBm is seen in AirTags and cameras and 7 dBm in AirTags and smartphones. In addition, smartphones only have a value of 7, bathroom devices 0, and TVs -44 dBm. AirTags have values of 6, 7, 9, 12, and 17 dBm. IPads, laptops, and smartwatches tend to have higher values, appearing on the right side of the x-axis, while headphones, bathroom devices, and TVs tend to be on the left side, indicating lower values. Lower values could mean that the device has less data to transmit or that the distance to its communicating device is shorter. This distribution shows that while some devices have unique *Power Level* characteristics, others share common values due to their different power consumption patterns. This indicates that different types of devices have different patterns of signal strength and power usage. Training ML models on these different patterns of the *Power Level* feature helps to classify devices since certain values indicate a specific device type. By adding the *Power Level* feature as an additional feature, the model is trained on a diverse set of features. This results in a model that is better able to generalize to unseen data than a model trained on a few features.

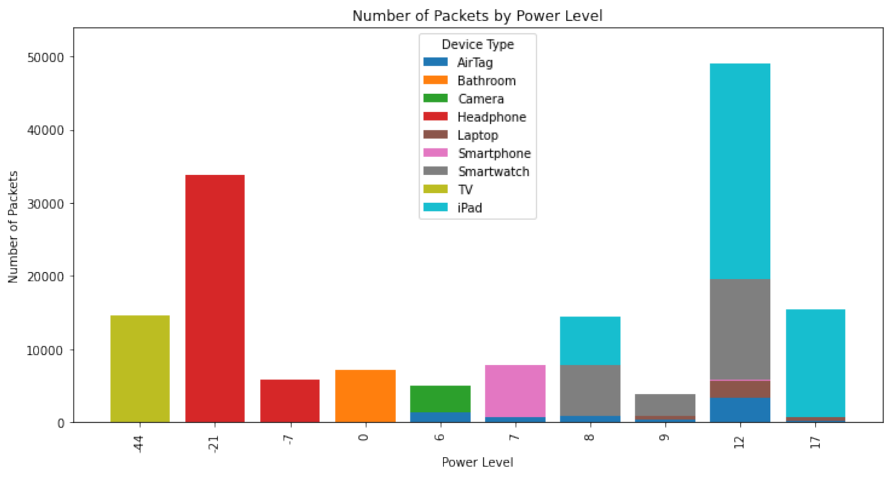


Figure 5.13: Number of Packets by *Power Level* Feature

Type

The *Type* feature is the AD type and is shown in Figure 5.14. Some device types have the same *Type* features and some have specific *Type* features. Headphones have AD types such as a combination of *Service Class UUIDs, Device Name, Flags, Manufacturer Specific, Tx Power Level, LE Bluetooth Device Address*, and *Service Data*. Cameras have AD types such as a combination of *Flags, Service Class, Manufacturer Specific, Appearance*, and *Device Name*. Smartphones have unique AD types such as a combination of *Service Class, Flags, Device Name*, and *Manufacturer Specific*. The AD Type of *Flags, Manufacturer Specific, Tx Power Level, Service Class, LE Bluetooth Device Address* is unique to TVs. The *Type* feature shows that different device types have specific combinations of AD types. This uniqueness of combinations is characteristic of devices such that the ML model can differentiate between the device types. Adding the *Type* feature to the set of features allows the model to make a more accurate prediction because each feature provides unique patterns to differentiate between device types.

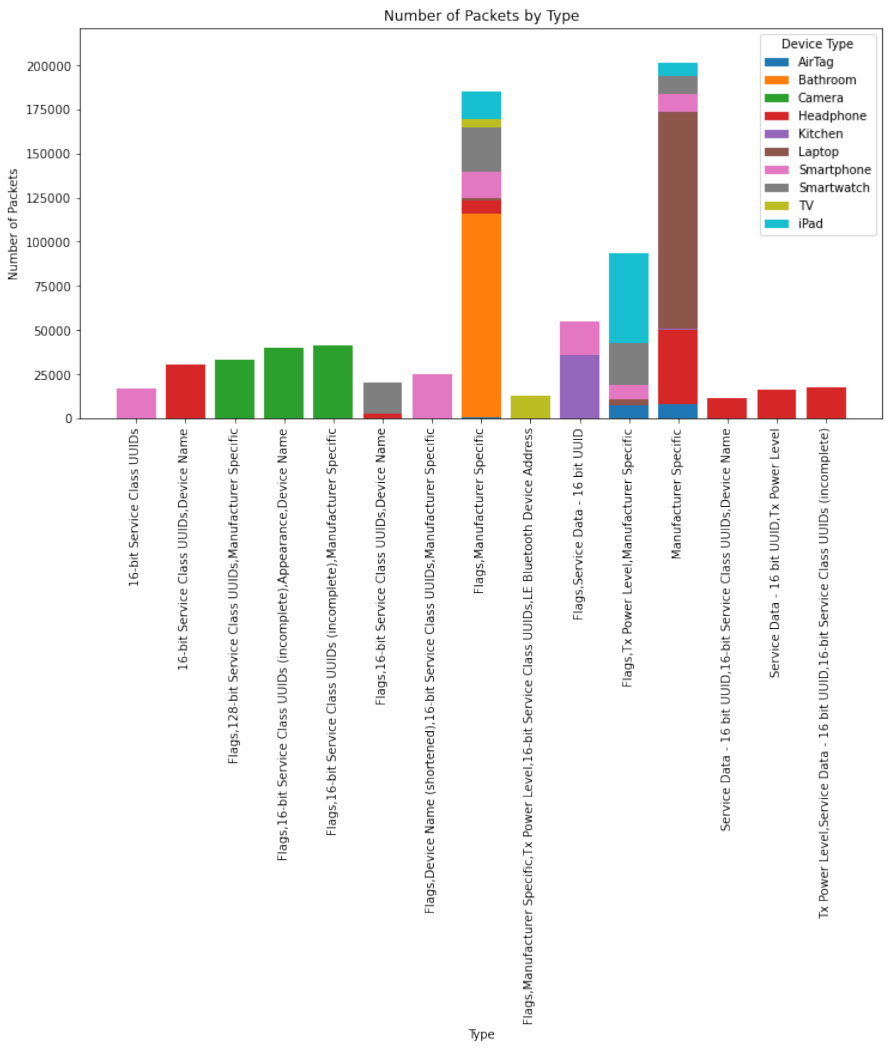


Figure 5.14: Number of Packets by *Type* Feature

5.4.2 Derived Feature Extraction from Sniffed BLE Packets

Based on the directly extracted features, further features are extracted. By taking the directly extracted features and performing statistical calculations based on them the derived features are extracted. The calculations are done by grouping the packets by their AdvA. Since each device is sniffed for 20 minutes and the AdvA is changed at intervals of up to 15 minutes, each device has at least two unique AdvAs. The full list of derived features is given in Appendix D. Some of these derived features are shown in the subchapter and some of them are shown in Appendix F due to the large number of them.

Packets per Second

The *Packets per Second* feature is shown in Figure 5.15 and is calculated by taking the number of packets per AdvA and dividing it by the time duration. The time duration is the difference between the minimum and maximum sniffed time of packets from that AdvA. This time duration is also an extracted feature and is shown in Appendix F. AirTags have a simple behavior, they have values of either 1 or 3. Bathroom devices have values of either 1 or 108. The value of 108 with 129'757 packets is excluded from the bar chart to avoid distortion. While one bathroom device has a low frequency of transmitting packets, the other bathroom device has a very high one. Cameras have six values: 1, 5, 6, 12, 34, and 37. However, headphones have a wide range of 14 different values, the most common being 27 and 46. Most kitchen devices have a value of 26. Laptops have values of 1 and a range between 28 and 31. This range of values shows that laptops have a more or less stable transmission rate. Smartphones have 9 different values with the most common being 30. Smartwatches have values of 8, 14, 31, and in a range from 1 to 5 showing a high variation due to their different functionalities. TVs have values of 1, 6, or 11. The most common values of iPads are 10, 11, or 12 showing a stable transmission rate. The values in Figure 5.15 are rounded for better visualization of the data. If the values are not rounded, each specific value uniquely identifies the device type. This indicates that the *Packets per Second* feature can distinguish between device types and is a feature that helps the ML model learn the pattern of device type to classify devices.

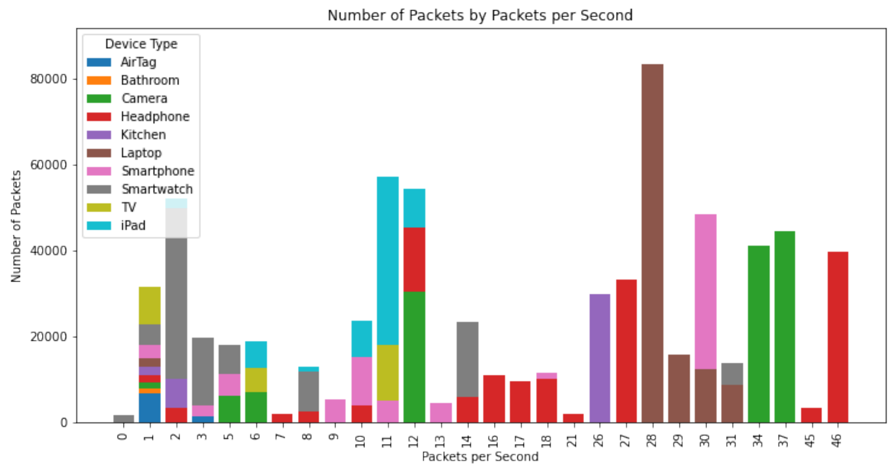


Figure 5.15: Number of Packets by *Packets per Second* Feature

Time per Packet

The *Time per Packet* feature is shown in Figure 5.16 and is calculated by dividing the time duration by the number of packets and represents the time it takes to transmit a packet in milliseconds. AirTags take 353, 704, or 736 milliseconds to transmit one packet. Bathroom devices have two values of either 9 or 1151. This large variation indicates that the devices are either in standby or active mode. Cameras have a range of packet transmission times of 27, 29, 83, 85, 175, 201, and 859. Most headphones have a value below 100 milliseconds showing that they require very little time to transmit a packet due to low-latency communication. Kitchen devices have values of 39, 422, 610, 623, and 689 with the majority of 39 milliseconds showing a short interval. This is due to regular updates or sensor readings which are common in kitchen devices. The majority of laptops have a time of either 33, 35, or 36 milliseconds showing a stable transmission frequency. Smartphones mainly transmit a packet for 34 milliseconds, with a few values around 100, 300, or 700 milliseconds. Most smartwatches have packet transmission times of 70, 315, or 423 milliseconds. TVs have packet transmission times of 93, 171, 691, and 820 milliseconds. IPads have values in the range of 87 to 171, and also values of 640 and 648 milliseconds. The *Time per Packet* feature shows that different devices have different patterns due to their communication needs. Devices such as cameras, headphones, laptops, and smartphones have low-latency communication, while others such as TVs and smartwatches take more time to transmit a packet. Using these unique patterns of each device type, the ML model can learn to distinguish between device types. Again, by adding this feature as an additional feature, the model is trained on a diverse set of features, resulting in better generalization to unseen data.

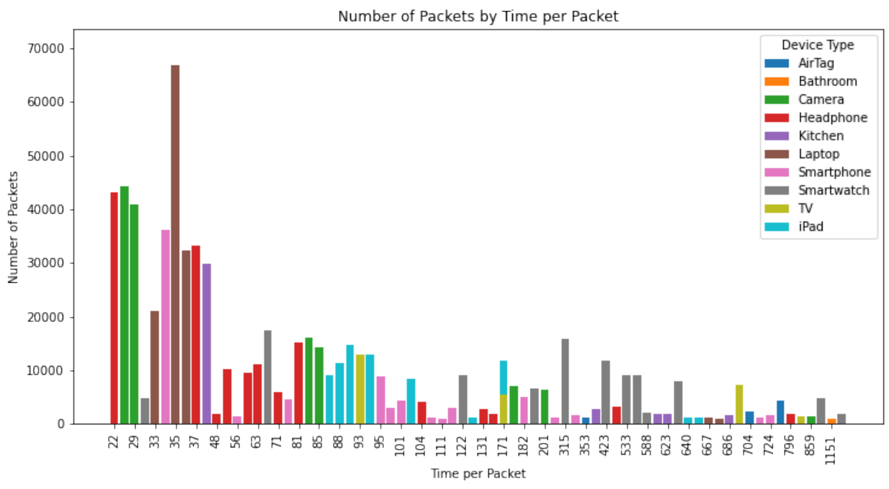


Figure 5.16: Number of Packets by *Time per Packet* Feature

Bytes per Second

The *Bytes per Second* feature is the total packet header length per AdvA divided by the time duration and is shown in Figure 5.17. Laptop devices have a speed of over 1000 bytes per second. Camera devices have a speed of around 34, 144, 166, 317, 323, 1125, or 1159 bytes per second with the majority over 1000 bytes per second. Headphones have a wide range of values with a majority speed of 1251 bytes per second. The majority of kitchen devices have a speed of 639 bytes per second with a small subset below 100 bytes per second. Smartphones also have a wide range of values but with a majority value of 852. The values for smartwatches also show a wide variation on the x-axis. The highest speed of TVs is 329 bytes per second, other values are 22, 47, and 210. The majority of iPad values are between 220 and 365 bytes per second. AirTags have values of 50, 52, and 71. Bathroom devices have values of either 20 or 2573. Again, the value of 2573 bytes per second is not shown in the bar chart to avoid distortion. Headphones, cameras, laptops, smartphones, and kitchen devices have the highest values, and smartwatches, AirTags, and TVs have the lowest. This shows a clear pattern in the distribution of the values that characterize the device types. By training the ML model on these characteristics of each device type, the model can learn specific patterns of device types to classify each device into its device type.

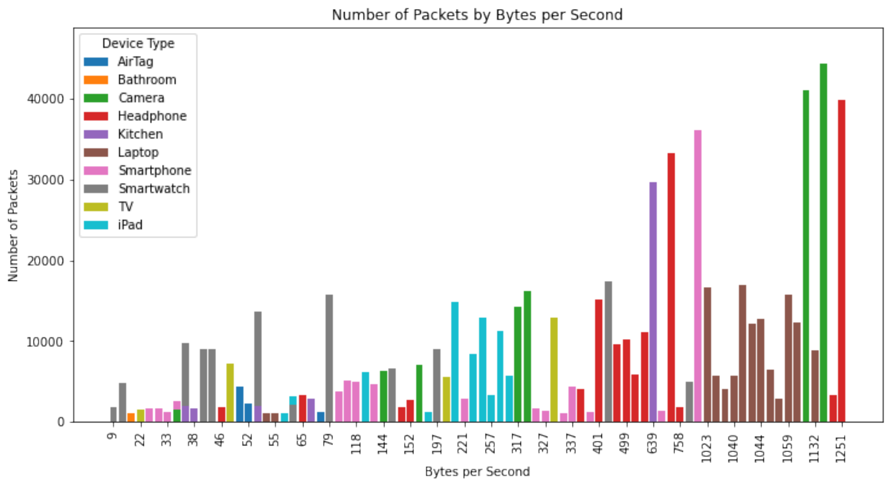


Figure 5.17: Number of Packets by *Bytes per Second* Feature

Average RSSI

The *Average RSSI* feature is shown in Figure 5.18. Laptop devices have the highest *Average RSSI* feature value of -19 dBm. Other values in the Laptop category range between -38 and -34 dBm. Smartphones have the second highest value of -23 dBm, but the majority have values of -74 dBm. IPads therefore have the second-highest values of -25 and 28 dBm. The majority of values for camera devices are between -59 and -42 dBm and for headphones between -49 and -31 dBm. Kitchen devices have values of -59, -57, -54, -49, and -46 dBm with the majority being -59 dBm. Smartwatches have a majority of -63, -61, and -57 dBm values. TVs have values of either -86, -79, -74, or -59 dBm, and AirTags have values of either -78, -41, -38, or -36 dBm. To avoid distortion, the value of -45 dBm from bathroom devices is removed. While some devices such as laptops have high values, others have a range of values across the x-axis. Headphones and cameras have a wide range of their values. On the other hand, most of the values are specific to device types which helps the ML model learn the pattern for classification tasks. By adding more features to train the ML model, the model can better understand the pattern of each device type and make predictions.

Standard Deviation RSSI

The *Standard Deviation RSSI* feature is shown in Figure 5.19 with a range on the x-axis between 1 and 19 dBm. The values of 14 or 19 dBm are assigned to camera

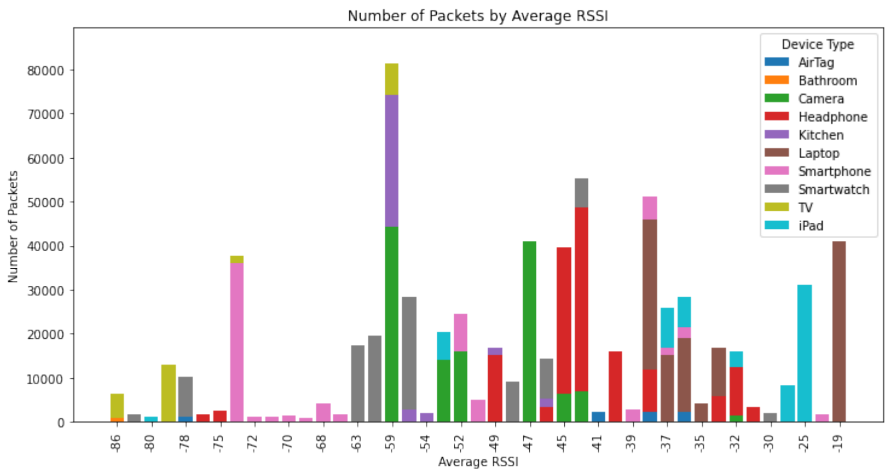


Figure 5.18: Number of Packets by *Average RSSI* Feature

devices, while other values are assigned to multiple device types. The majority of the laptops tend to have a very low standard deviation of RSSI namely 1 or 2 dBm, with a small number of packets having values of 4 or 6 dBm. AirTags have a standard deviation of either 3, 4, 6, or 8 dBm. Values in the Headphone category range from 2 to 7 dBm. Kitchen devices have values of either 5, 6, or 7 dBm. Smartphones range from 3 and 8 dBm and smartwatches from 5 and 10 dBm. TVs have a standard deviation of either 2, 5, 10, or 13 dBm. The values of bathroom devices are either 6 or 7 dBm. IPads have a wide range of values between 2 and 9 dBm. Overall, cameras and smartwatches have high *Standard Deviation RSSI* feature values, while laptops and headphones have low values. These high and low standard deviation values are strong indicators for the ML model to identify them. The broad ranges of other devices make them more difficult for the ML model to classify but they still have a pattern that can be recognized by the ML model.

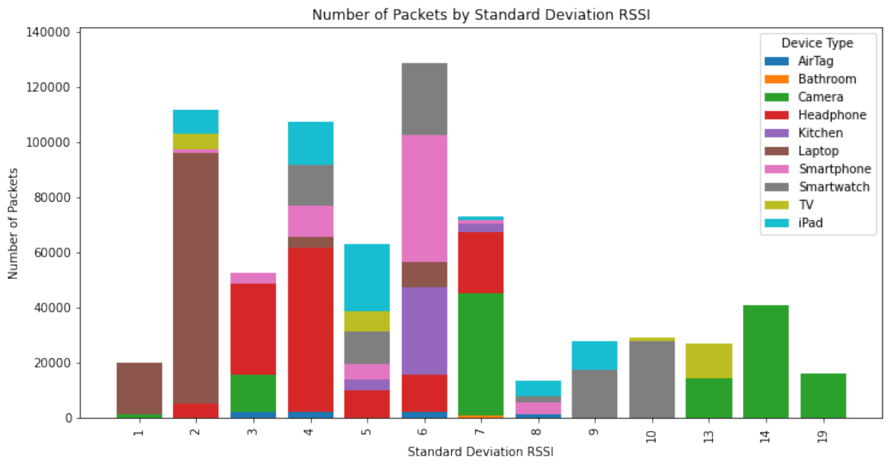


Figure 5.19: Number of Packets by *Standard Deviation RSSI* Feature

Variance RSSI

Instead of calculating the standard deviation of the RSSI, the variance of the RSSI is calculated which is more specific to device types as shown in Figure 5.20. Laptop devices have a very low variance of RSSI meaning that the RSSI values are around the average, and have stable positions. On the other hand, cameras, TVs, and smartwatches have a very high variance which means that their RSSI values vary a lot, are spread around the average value, and are in a moving state. Kitchen devices have a majority value of 38 dBm which is neither low nor high. Headphones, smartphones, and iPads have a wide variance, but most of their values are specific to them. TVs have an ambiguous behavior of their variance values, being either 2, 29, or over 100 dBm. AirTags have values of 10, 19, 40, or 61 dBm, and bathroom devices have values of either 34 or 46 dBm. Since each value on the x-axis is unique to the device types assigned, the *Variance RSSI* feature has a more specific pattern behavior of device types than the *Standard Deviation RSSI* feature. This is because most devices have either low or high overall values, while only a small number of device types have a wide range of values. The classification task can be performed by using this pattern to train the ML model.

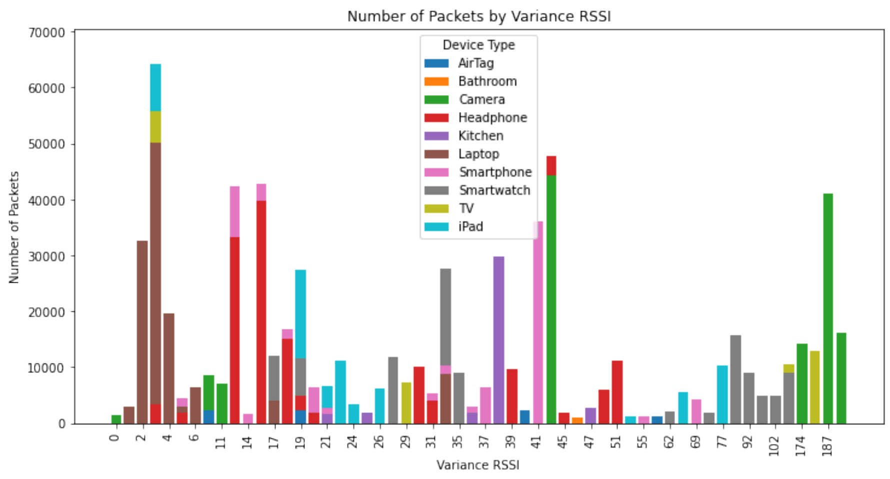


Figure 5.20: Number of Packets by *Variance RSSI* Feature

Average Packet Length

The *Average Packet Length* feature is shown in Figure F.11. Bathroom devices have an average of 24 and laptop devices 37. AirTags have values of 14, 23, 25, or 37. Camera devices have values ranging from 27 to 33. Headphones have values of 20, 26, 27, 28, 33, 36, and 37 showing a wide range. Smartphones, smartwatches, and iPads also show a wide range, ranging from 10 to 37 and 14 to 37, respectively. The majority of kitchen devices have a value of 25. TVs have values of 18, 30, 32, and 36. While bathroom devices and laptops have specific values, the values of other device types vary widely and therefore do not have an obvious pattern for this feature. The ML model has difficulty learning this unobvious pattern, while the pattern for bathroom devices and laptops is easy to learn by the model to classify them.

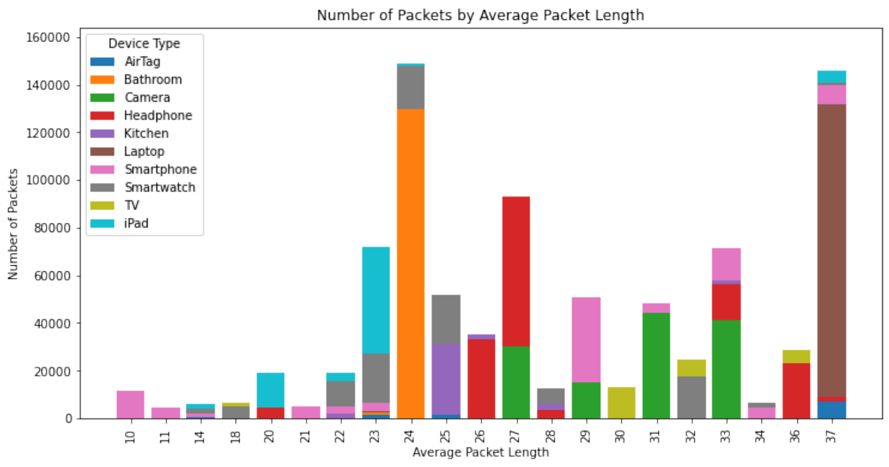


Figure 5.21: Number of Packets by *Average Packet Length* Feature

Standard Deviation Packet Length

Many device types such as AirTags, cameras, headphones, laptops, smartphones, smartwatches, and iPads have a *Standard Deviation Packet Length* feature values of 0 as shown in Figure 5.22. This means that most of these devices have the same packet length value within their type. Most smartwatches and TVs tend to have a higher standard deviation, above 7. While iPads tend to have a low standard deviation of below 4. Bathroom devices have values of 4 or 19, with the value 19 being the highest one. AirTags have values of 0, 2, and 3, cameras have values of 0, 5, 6, and 8, headphones have values of 0, 1, 2, 4, 5, and 6, and smartphones have values of 0, 2, 3, 4, 5, and 8. Kitchen devices have a narrower range of values of 1, 4, 7, and 8. Since smartwatches, TVs and bathroom devices have significant values, this shows that the *Standard Deviation Packet Length* feature is a characteristic feature for these devices. Based on this feature the ML model can differentiate these device types. Adding this feature to model training allows the model to capture more complex patterns and relationships in the data, resulting in more accurate predictions.

Variance Packet Length

The *Variance Packet Length* feature is shown in Figure 5.23. AirTags, cameras, headphones, laptops, smartphones, smartwatches, and iPads again have values of 0. Bathroom devices have specific values of 14 or 352, with the majority being 14 and the highest value being 352. As for the *Standard Deviation Packet Length* feature, smartwatches, and TVs tend to have high *Variance Packet Length* feature values, above 72. IPads and AirTags tend to have low values of less than 13 and 7 respectively. In addition to having a value of 0, the values of cameras, headphones,

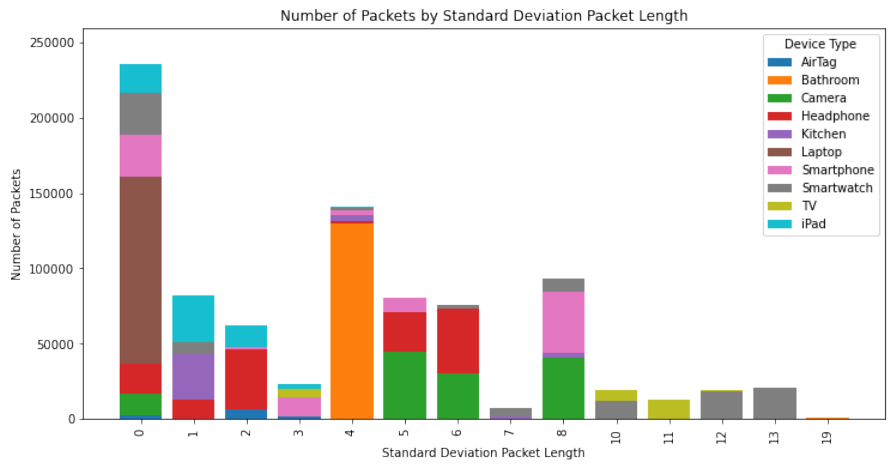


Figure 5.22: Number of Packets by *Standard Deviation Packet Length* Feature

and smartphones are in the middle of the bar chart which means that they do not have as high values as TVs and smartwatches. The majority of kitchen devices have a value of 2. Unlike the *Standard Deviation Packet Length* feature, the values of the *Variance Packet Length* feature are more specific to different device types. This specific pattern is useful for training the ML model to differentiate between device types. By including this feature, the model can better distinguish between different types of devices because it has more unique information.

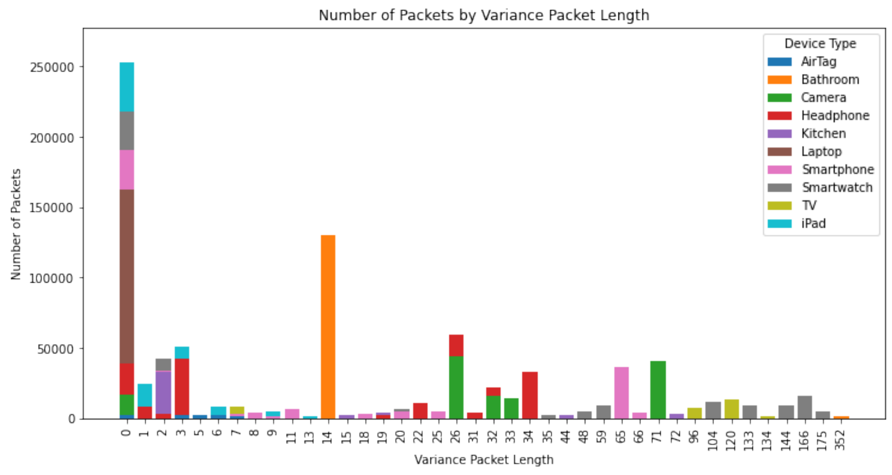


Figure 5.23: Number of Packets by *Variance Packet Length* Feature

Average Delta Time

The *Average Delta Time* feature is shown in microseconds in Figure 5.24. AirTags have values above 55'325 microseconds showing a very high average of delta time. Smartwatches, smartphones, headphones, and kitchen devices also have high average delta times but also low ones. Laptops and iPads have a medium to high average delta time compared to other device types. On the other hand, TVs and cameras have a medium to low average delta time. To avoid distortion in the bar chart, the values of bathroom devices are removed. They have values of either 5'353 or 5'465 which are in the medium to low average delta time category. Each value on the x-axis is uniquely assigned to a device type showing that this feature has a specific pattern for each device type. Because the pattern is specific, the ML model can learn the characteristics of each device type and easily predict the device type. By adding this feature, the set of features is more diverse which improves the model's ability to classify devices.

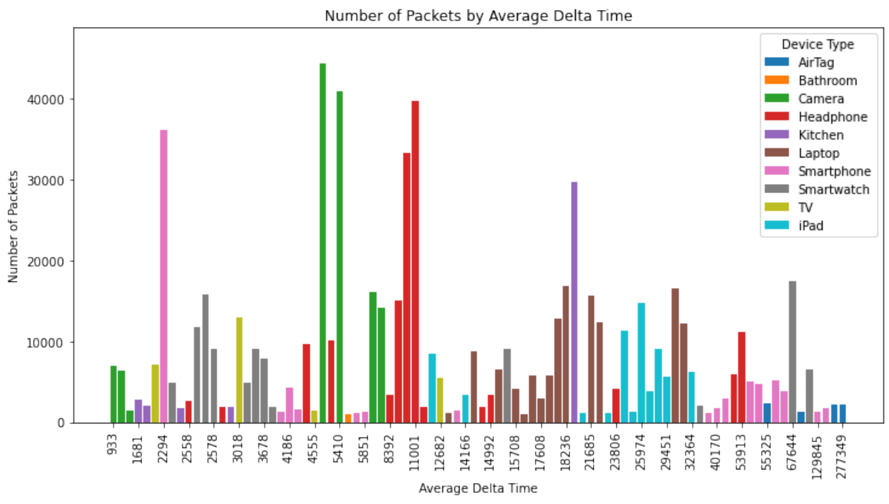


Figure 5.24: Number of Packets by *Average Delta Time* Feature

Standard Deviation and Variance Delta Time

The *Standard Deviation Delta Time* feature is also shown in microseconds in Figure 5.25. Its distribution is quite similar to the distribution of the *Average Delta Time* feature but with different values. AirTags, laptops, and iPads have a medium to high, TVs and cameras have a medium to low, and smartwatches, smartphones, headphones, and kitchen devices have a high but also a low average delta time. The values of bathroom devices are also removed from the bar chart and are of either 5'103 or 8'945 showing a medium to low average delta time. Once again, each value on the x-axis is uniquely assigned to a device type that shows a specific pattern to train the ML model to predict the device type. By including this feature, the model can capture more variability in delta times and is better able to distinguish between different device types.

Since the distribution of the *Variance Delta Time* features is also quite similar to the distribution of the *Average Delta Time* feature but with different values, it is shown in Figure F.17 in Appendix F. For *Variance Delta Time* feature, the same conclusion about the characteristics of the device types applies as for *Standard Deviation Delta Time* feature.

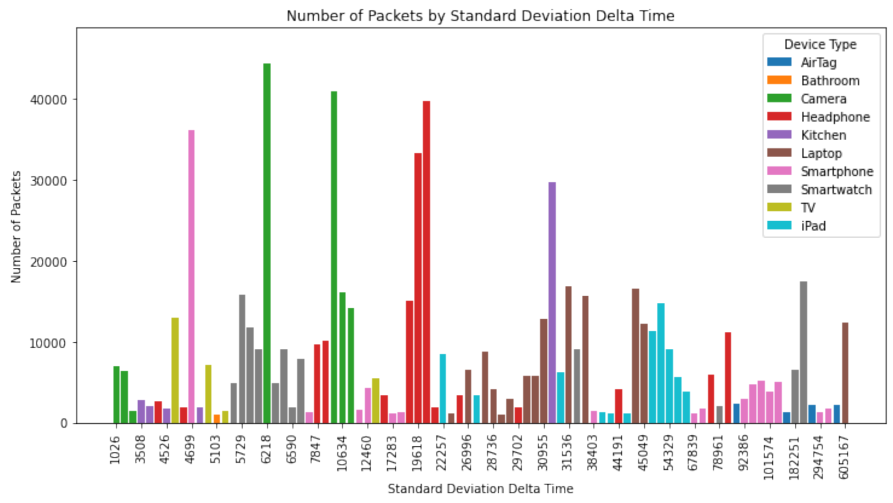


Figure 5.25: Number of Packets by *Standard Deviation Delta Time* Feature

5.5 Data Preprocessing

The data labeling and feature extraction are done individually on the 22 CSV files, adding new columns with header names such as ”label” and the name of the extracted features. These files are merged into a single CSV file with a total of 901’623 packets. This merged file is preprocessed by cleaning the data, removing unique identifiers, and transforming categorial data into a numerical representation.

5.5.1 Data Cleaning and Removal of Unique Identifiers

The extracted features are the columns in the CSV file. Columns with more than 99% of their data having missing values are removed. Additionally, columns with the same constant values do not contribute to the prediction task and are also removed. Furthermore, the columns *No.* and *Time* represent the chronological order of the sniffed packets and are therefore unnecessary and are removed from the dataset. Unique identifiers such as *AdvA*, *Company ID*, *UUID*, and *Device Name* are also removed. The *label* column which is the device type is not removed from the dataset since it is the target value that should be predicted by the ML models. After performing data splitting, the *subcategory* column is also removed. These columns are removed from the dataset:

- Target Address
- Address Type
- Simultaneous LE and BR/EDR to Same Device Capable (Host).1
- LE Supported By Host
- OOB Data Present
- Hash C
- Custom UUID
- BIG_Offset
- BIG_Offset_Units

- ISO_Interval
- Num_BIS
- NSE
- BN
- Sub_Interval
- PTO
- BIS_Spacing
- IRC
- Max_PDU
- Reserved.1
- Seed Access Address
- SSP OOB Length
- Malformed Packet
- Length of packet
- PHY
- Access Address
- Protocol
- No.
- Time
- AdvA
- Company ID
- UUID 16
- Device Name
- Subcategory

5.5.2 Data Transformation

Using one-hot encoding, categorial features are transformed into a numerical representation. These are the categorial features:

- Packet Header
- Type
- Length
- PDU Type
- Info
- Channel Selection Algorithm
- Tx Address
- Rx Address
- Scanning Address
- Simultaneous LE and BR/EDR to Same Device Capable (Host)
- Simultaneous LE and BR/EDR to Same Device Capable (Controller)
- BR/EDR Not Supported
- LE General Discoverable Mode
- LE Limited Discoverable Mode
- Power Level (dBm)
- Data
- Custom UUID.1
- Service Data
- BD_ADDR
- CRC
- Label
- Packet Direction

The *Packet Header* feature is not transformed into numerical by one-hot encoding since all values start with '0x' followed by integers. Instead, the substring '0x' is removed from the start of the string to transform it into a numerical value. Furthermore, the *Type* and *Length* feature are not transformed by one-hot encoding either. Each packet in the *Type* column is a combination of packet header types such as flags, manufacturer specific, service class, and device name. Using one-hot encoding would add a new column for each combination of the types. Because there are many combinations of the types in the dataset, one-hot encoding is not used. Instead, all unique type values are picked out, resulting in a total of 49. For each unique type value, a new column is added to the file. After that, each packet is iterated through and a value of 1 is appended to the type columns that occur in the combination, and a value of 0 is appended to the remaining type columns. The same is done with the *Length* feature, which also adds 49 new columns with values of their corresponding lengths.

The remaining categorial features are transformed using one-hot encoding. Since it takes the unique values of a feature and adds as many new columns as unique values, it is important to know how many new columns are added. Table 5.3 lists the features with their number of unique values. Features with more than 300 unique values are removed from the dataset to prevent an excessive increase in the file size and the training of the ML models. As a result, the *Scanning Address*, *Data*, *Service Data*, *BD_ADDR* and *CRC* are removed. The remaining categorial features, except the *label* column, are converted to numerical representation. Lastly, empty values are filled with 0.

5.6 Feature Selection

Since not all extracted features are relevant for predicting the device type, two feature selection methods are implemented. Applying the filter methods, the 30 most important features based on their relation to the target variable are listed in Table 5.4. The order of the importance of the features is descending. The feature *Number of Forward Packets* is at the top with the highest score value followed by features such as *Number of Packets*, *Sum Payload Length*, *Sum Packet Length*, and *Sum RSSI*. The scores are the ANOVA F-values. A high score means that the variance of the feature values across the device types is very high compared to the variance within the device types indicating that the feature is highly specific to a device type. The feature *Min Payload Length* has the same score as *Min Packet Length*, the feature *Average Packet Length* has the same score as *Average Payload Length*, and the feature *Standard Deviation Payload Length* has the same score as *Standard Deviation Packet Length.*

Applying the embedded methods, the 30 most important features in relation to the RF model are listed in Table 5.5. The order of the importance of the features is also descending. The feature *Sum Payload Length* has the highest importance score of 0.0602 followed by the features such as *Sum Packet Length*, *Average Payload Length*, *Number of Forward Packets*, and *Sum of Delta Time.* Higher scores

Feature	**No. Unique Values**
PDU Type	7
Info	24
Channel Selection Algorithm	2
Tx Address	2
Rx Address	2
Scanning Address	617
Simultaneous LE and BR/EDR to Same Device Capable (Host)	10
Simultaneous LE and BR/EDR to Same Device Capable (Controller)	10
BR/EDR Not Supported	10
LE General Discoverable Mode	9
LE Limited Discoverable Mode	10
Data	6732
Custom UUID	47
Service Data	1867
BD_ADDR	382
CRC	7430
label	10
Packet Direction	2

Table 5.3: Categorical Features and their Number of Unique Values

indicate a greater impact of the feature on the accuracy of the model. The feature *Sum Payload Length* has the most influence on the model in making predictions. Compared to the second important feature *Sum Packet Length* with a score of 0.0486, the first important feature *Sum Payload Length* has about 1.24 times more influence on the model than the second feature meaning that it characterizes device types more accurately.

Comparing the selected features of both methods, most of the features appear in both tables but in different order. Features such as *Info_ADV_NONCONN_IND*, *PDU Type_0x2*, *LE Limited Discoverable Mode_True*, *BR/EDR Not Supported_True*, *LE Bluetooth Device Address*, *Standard Deviation RSSI*, *Flags*, *16-bit Service Class UUIDs (incomplete)*, *Tx Address_Random*, *Manufacturer Specific_Length*, *Power Level (dBm)*, *Standard Deviation Payload Length*, and *Average Number of Forward Packet* are selected as important only by the filter method and features such as *Time per Packet*, *Average Backward Delta Time*, *Average Delta Time*, *Sum Backward Delta Time*, *Standard Deviation Delta Time*, *Max Packet Length*, *Std Forward Delta Time*, *Var Forward Delta Time*, *Average Forward Delta Time*, *Max Delta Time*, *Variance Delta Time*, *Max Payload Length*, *Average Number of Backward Packet*, and *Var Backward Delta Time* by the embedded method. While the filter method selects features extracted directly from the BLE LL, the embedded method selects only derived features to be the first 30 most important.

Feature	**Score**
Number of Forward Packets	1012683.7
Number of Packets	994208.4
Sum Payload Length	781632.5
Sum Packet Length	650533.5
Sum RSSI	571983.9
Number of Backward Packets	312498
Min Payload Length	173258.3
Min Packet Length	173258.3
Min RSSI	158535.4
Info_ADV_NONCONN_IND	150512.4
PDU Type_0x2	150101.4
LE Limited Discoverable Mode_True	128495.8
BR/EDR Not Supported_True	103749.1
Duration	92221.8
Average RSSI	78899.1
Average Packet Length	75267
Average Payload Length	75267
Sum of Delta Time	74557.6
Sum Forward Delta Time	74439.1
LE Bluetooth Device Address	69129.1
Standard Deviation RSSI	68867.3
Flags	68182.6
16-bit Service Class UUIDs (incomplete)	66783.7
Tx Address_Random	65373.2
Manufacturer Specific_Length	64314.2
Power Level (dBm)	61154
Standard Deviation Payload Length	57397.4
Standard Deviation Packet Length	57397.4
Average Number of Forward Packet	56421.5

Table 5.4: Filter Method: Selected Features and Their Scores

Feature	**Score**
Sum Payload Length	0.0602
Sum Packet Length	0.0486
Average Payload Length	0.0468
Number of Forward Packets	0.0447
Sum of Delta Time	0.0438
Number of Packets	0.0432
Sum RSSI	0.0398
Average Packet Length	0.0397
Time per Packet	0.0370
Min RSSI	0.0337
Number of Backward Packets	0.0332
Sum Forward Delta Time	0.0327
Min Packet Length	0.0285
Min Payload Length	0.0277
Average Backward Delta Time	0.0260
Average Delta Time	0.0202
Average RSSI	0.0194
Sum Backward Delta Time	0.0187
Standard Deviation Delta Time	0.0186
Max Packet Length	0.0185
Std Forward Delta Time	0.0183
Var Forward Delta Time	0.0182
Duration	0.0174
Average Forward Delta Time	0.0171
Max Delta Time	0.0161
Variance Delta Time	0.0151
Max Payload Length	0.0146
Standard Deviation Packet Length	0.0145
Average Number of Backward Packet	0.0142
Var Backward Delta Time	0.0127

Table 5.5: Embedded Method: Selected Features and Their Importance Scores

5.7 Data Splitting

5.7.1 80/20 Data Splitting Method

The dataset is divided into 80% training and 20% testing datasets. The training dataset contains 721'298 packets and the test dataset contains 180'325 packets. Figure 5.26 shows the number of packets of each device type in the training dataset in blue and in the test dataset in red.

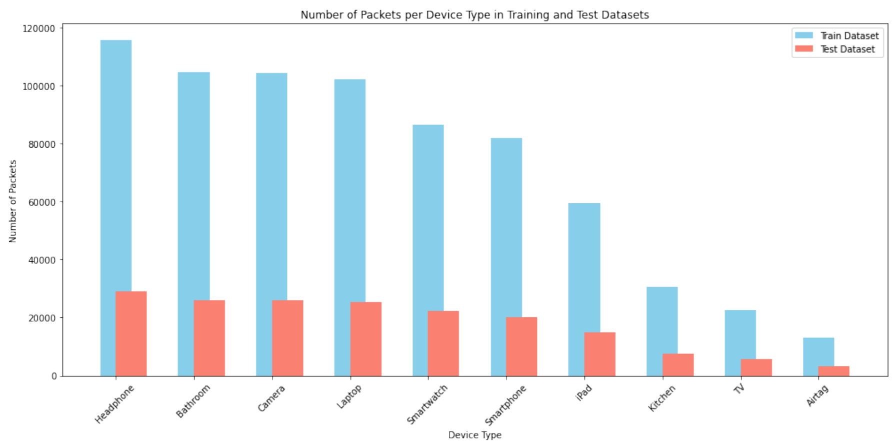


Figure 5.26: Total Packets of Training and Test Datasets

5.7.2 Alternative Data Splitting Approach

According to the alternative data splitting approach, the training dataset has 687'463 packets and the test dataset has 214'160 packets which is 76.25% and 23.75% of the total dataset. Table 5.6 shows which devices are included in the training dataset and which are additionally included in the test dataset and excluded from the training dataset. Since devices of the device types iPad and AirTags have the same name of manufacturer, these devices are included in both datasets.

5.8 Device Classification via ML Approach

The RF classifier and the MLP model are trained on the dataset divided once by the 80/20 data splitting method and once by the alternative data splitting method. This results in two RF classifiers and two MLP models. Then the two RF classifiers are trained with the selected features from the embedded feature selection method, and the two MLP models are trained with the selected features from the filter feature selection method. In total, eight models are trained and their accuracies are compared.

Device Type	**Devices in Training Dataset**	**Devices in Test Dataset**
Laptop	Dell laptop, Hp laptop	+ Apple laptop
Headphone	Apple headphone, Sony Linkbuds headphone, Bose-1 headphone, Bose-2 headphone Sony XM4 headphone, Sony XM3 headphone, Samsung headphone	+ Beats headphone
iPad	Apple iPad	Apple iPad
AirTag	Apple AirTag	Apple AirTag
TV	LG TV	+ Samsung TV
Kitchen	Xiaomi kettle, Xiaomi airfryer	+ Xiaomi mixer
Smartwatch	Apple smartwatch, Ericsson smartwatch, Fitbit smartwatch, Huawei smartwatch	+ Galaxy smartwatch
Camera	GoPro5, GoPro9, Rollei camera, Canon, GoPro10	+ GoPro4
Smartphone	Nokia smartphone, Samsung smartphone, Huawei smartphone, Google smartphone	+ Apple smartphone
Bathroom	Oral-B toothbrush	+ AquaClean shower toilet

Table 5.6: Devices in Training and Test Datasets

5.8.1 Random Forest Classifier

RF Classifier with 80/20 Data Splitting Method

Training the RF classifier with default parameters and on the training dataset generated by the 80/20 data splitting method, the model achieves a training accuracy of 99.9992% and a test accuracy of 99.9717%. Since both accuracies are close to each other and there is no large gap between them, there is no overfitting.

Training the model on k=5 different subsets of the data, the cross-validation scores are [0.85721336 0.98913628 0.86761126 0.99332868 0.87166434] and the average accuracy across all k=5 cross-validation folds is 91.58%. Since the cross-validation scores are close to the test accuracy, the model performs well on unseen data. Figure 5.27 shows the learning curves of the training and cross-validation scores. The training score is shown in red and the cross-validation score is shown in green. The shaded areas are the standard deviations of the scores. Both scores converge and have high values, which means that the model performs well on unseen data. Since the training score does not decrease and the cross-validation score increases, the model performs better with more data.

The accuracies for each device type are calculated and are shown in Table 5.7. All accuracies of the device types are above 99.597%. The highest accuracy is achieved through the TV devices and the lowest through the AirTags. The accuracy of AirTags may have to do with the fact that it has the least amount of data packets

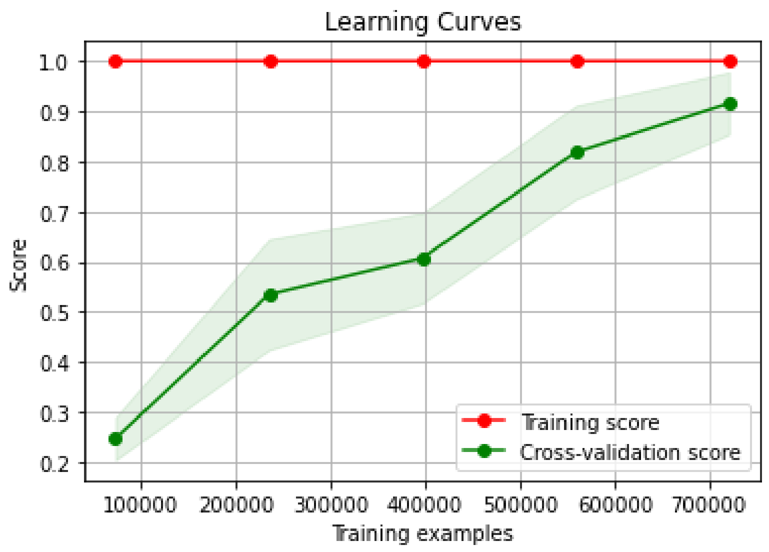


Figure 5.27: Learning Curves of the RF Classifer with 80/20 Data Splitting Method

compared to the other device types. Even when AirTags have the fewest data packets and therefore the lowest accuracy, the data balancing technique is not applied because their accuracy is still very high. These high accuracies of all device types show that the model performs very well in classifying each device type.

Device Type	**Test Accuracy**
Laptop	99.937
Headphone	99.993
iPad	99.96
AirTag	99.597
TV	100
Kitchen	99.987
Smartwatch	99.968
Camera	99.992
Smartphone	99.985
Bathroom	99.996

Table 5.7: Test Accuracies of each Device Type of the RF Classifer with 80/20 Data Splitting Method

The evaluation metrics such as precision, recall, specificity, F1-score, TP, TN, FP, and FN are computed for each device type and are listed in Table 5.8 except for TP and TN. Precision, recall, specificity, and F1-score are above 99%. High precision values mean that the model has a very low FP rate, which means that when the model predicts the type of a device, the prediction is almost always correct. High values of recall mean that the model has a very low rate of FN, which means that the model identifies the device types of almost all devices. High values of specificity mean that the model correctly identifies negative class types, for example when a device is predicted not to be a specific type, the prediction is almost always correct.

High F1-scores mean that the model performs well in both precision and recall. Overall, the model predicts the device types accurately. TP and TN have high values which indicates that the model correctly classified positive and negative instances, respectively. Additionally, the very low values of FP and FN indicate that the model makes very few incorrect predictions.

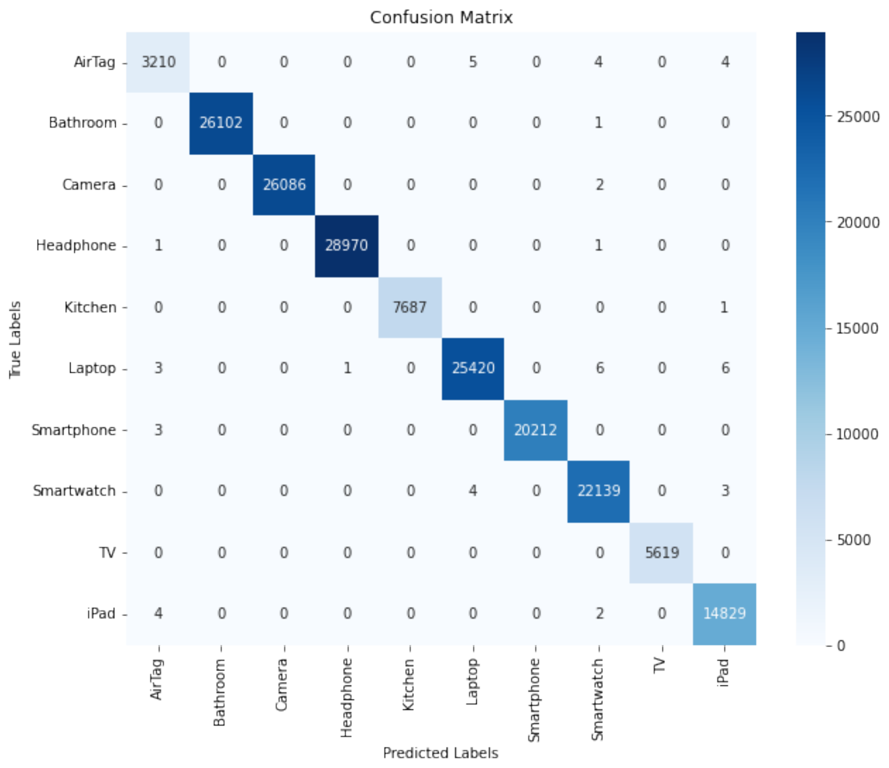
Label	**Precision**	**Recall**	**Specificity**	**F1-Score**	**FN**	**FP**
AirTag	0.996585	0.995966	0.999938	0.996276	13	11
Bathroom	1.000000	0.999962	1.000000	0.999981	1	0
Camera	1.000000	0.999923	1.000000	0.999962	2	0
Headphone	0.999965	0.999931	0.999993	0.999948	2	1
Kitchen	1.000000	0.999870	1.000000	0.999935	1	0
Laptop	0.999646	0.999371	0.999942	0.999509	16	9
Smartphone	1.000000	0.999852	1.000000	0.999926	3	0
Smartwatch	0.999278	0.999684	0.999899	0.999481	7	16
TV	1.000000	1.000000	1.000000	1.000000	0	0
iPad	0.999057	0.999596	0.999915	0.999326	6	14

Table 5.8: Performance Metrics of the RF Classifier with 80/20 Data Splitting Method

The confusion matrix is shown in Figure 5.28. The elements on the diagonal show the number of correct predictions for each device type. High diagonal elements indicate that the model accurately classifies data of its type. For example, the model correctly predicted 3'210 data as AirTags, 26'102 as bathroom devices, and 26'086 as camera devices. The off-diagonal elements show the misclassification. For example, the model incorrectly classified 4 AirTags and 6 laptop devices as smartwatch devices. Overall, the model has very high values along the diagonal and very few on the off-diagonal, which results in a good performance of the model.

As already mentioned in Chapter 5.6 using the embedded filter selection method Table 5.5 shows the first 30 most important features. The number of features with the highest accuracy for iterative training of the RF classifier is 16. This is achieved by starting with the top feature and iteratively adding one feature at a time until all 30 features are included. The model trained on the first 16 selected features achieves the highest training accuracy of 99.995% and test accuracy of 99.956%. Comparing the training and test accuracies with the accuracies of the model trained on all features shows that feature selection does not improve the model's performance, as they are lower than 99.9991% and 99.9717%, respectively. However, the accuracy of the model trained on the 16 most important features is still high and in addition, the training time is shorter and the computational cost is lower than the model trained with all features.

Due to memory limitations, hyperparameter tuning for the RF model could not be performed. However, even if the parameters are fine-tuned, the accuracy may not change much since it is high anyway.



True Labels \ Predicted Labels	AirTag	Bathroom	Camera	Headphone	Kitchen	Laptop	Smartphone	Smartwatch	TV	iPad
AirTag	3210	0	0	0	0	5	0	4	0	4
Bathroom	0	26102	0	0	0	0	0	1	0	0
Camera	0	0	26086	0	0	0	0	2	0	0
Headphone	1	0	0	28970	0	0	0	1	0	0
Kitchen	0	0	0	0	7687	0	0	0	0	1
Laptop	3	0	0	1	0	25420	0	6	0	6
Smartphone	3	0	0	0	0	0	20212	0	0	0
Smartwatch	0	0	0	0	0	4	0	22139	0	3
TV	0	0	0	0	0	0	0	0	5619	0
iPad	4	0	0	0	0	0	0	2	0	14829

Figure 5.28: Confusion Matrix of the RF Classifier with 80/20 Data Splitting Method

RF Classifier with Alternative Data Splitting Method

Using the training and test datasets created by the alternative data splitting method, the training accuracy is 99.9994% and the test accuracy is 83.5992%. Comparing these accuracies with the accuracies from the RF classifier using the 80/20 data splitting method, the training accuracy is higher but the test accuracy is much lower than that one. Performing cross-validation with k=5 the cross-validation scores are 85.72%, 98.91%, 86.76%, 99.33%, and 87.17%, and the mean cross-validation accuracy is 91.58%. The learning curves of the training and cross-validation scores are shown in Figure 5.29. While the training accuracy is very high showing that the model fits the training data perfectly, the test accuracy is lower showing that the model does not generalize well to unseen data. Since there is a gap between the training and test accuracies, overfitting is occurring. The cross-validation scores range from 85.72% to 99.33% which are variable showing that the performance of the model is not consistent across different folds of the data. The mean cross-validation accuracy is much lower than the training accuracy and closer to the test accuracy showing that the model may perform well on some subsets of the data but not on others. This leads to overfitting where the model fits the training data too well but does not generalize well to new data.

The test accuracies for each device type are shown in Table 5.9. Six device types have an accuracy greater than 96% except laptops have an accuracy of 76.08%, TVs of 28.27%, kitchen devices of 65.46%, and smartphones of 57.13%. The lowest

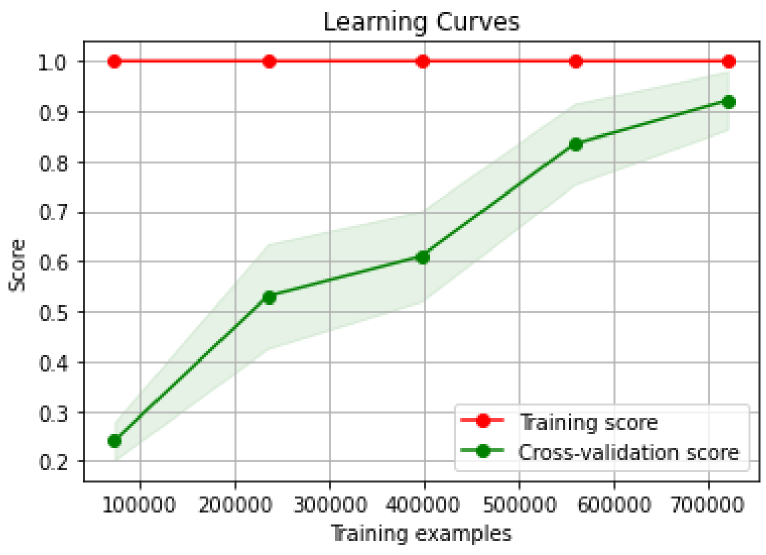


Figure 5.29: Learning Curves of the RF Classifier with Alternative Data Splitting Method

accuracy is 28.27% of the TV devices. The reason might be that the LG TVs and the Samsung TVs do not have a similar pattern in their data packets and the model cannot predict the device type of the Samsung TVs based on the learned pattern of the LG TVs. The same reason for low test accuracy applies to smartphones, bathroom devices, and laptops. While the model learns the pattern of Nokia, Samsung, Huawei, and Google smartphones, it could not correctly predict the type of Apple smartphones. The test accuracies of the other device types are very high showing that the model performs well on unseen data of these device types but not on TVs, smartphones, kitchen devices, and laptops. Since the reason for the low accuracy of laptops, TVs, kitchen, and bathroom devices is due to the data splitting approach, and these devices have enough data packets compared to the AirTags with a high accuracy, the data balancing technique is not applied.

Device Type	**Test Accuracy**
Laptop	76.08
Headphone	99.94
iPad	99.93
AirTag	99.57
TV	28.27
Kitchen	65.46
Smartwatch	99.95
Camera	99.99
Smartphone	57.13
Bathroom	96.13

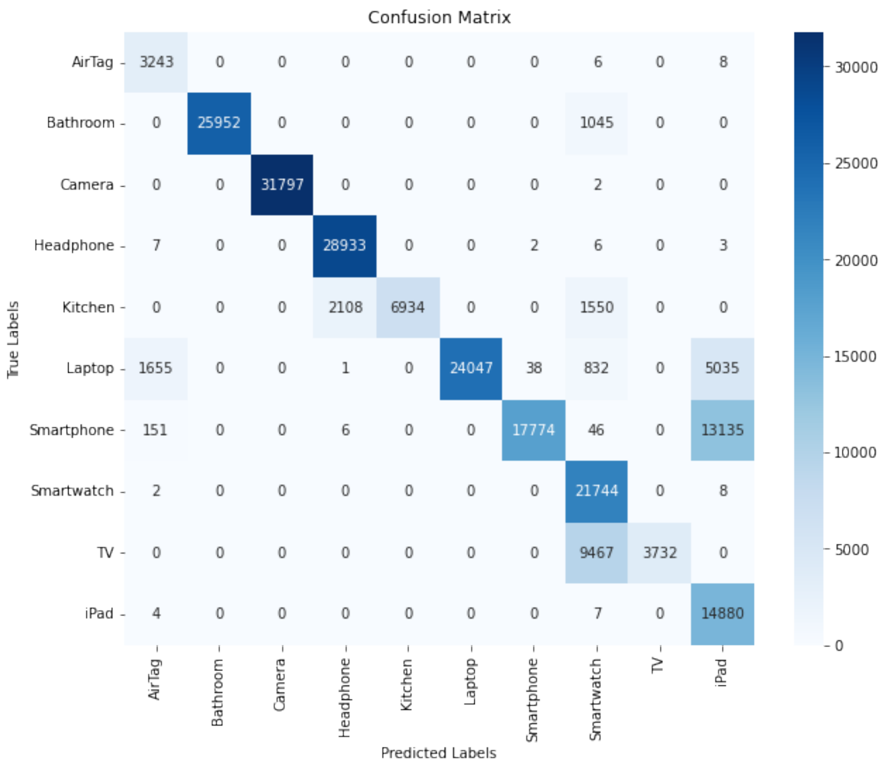
Table 5.9: Test Accuracies of each Device Type of the RF Classifier with Alternative Data Splitting Method

The precision, recall, specificity, F1-score, FP, and FN of each device type are listed in Table 5.10. AirTags have a high recall and specificity showing that the model performs well at identifying actual AirTag devices, but have a lower precision and F1-score showing that there are some FP. Bathroom devices and cameras have high precision, recall, specificity, and F1-score showing that the model performs well with no FP and very few FN. Headphones also have high precision, recall, specificity, and F1-score with very few FP or FN. Kitchen devices, laptops, and smartphones have high precision and specificity but low recall and F1-score showing that the model correctly predicts the device type but misses some devices to classify them as kitchen devices, laptops, and smartphones, respectively. On the other hand, smartwatches and iPads have high recall and specificity but low precision showing a very high number of FP. TVs have very high precision and specificity but very low recall and F1-score showing that most TV devices are missed by the model to be classified as a TV but the devices classified as TV are correctly predicted. These results show which devices are difficult for the model to classify. The devices with high FN and FP are not correctly classified into their category, and thus the model cannot predict each device to its correct type based on the training data.

Label	**Precision**	**Recall**	**Specificity**	**F1-Score**	**FN**	**FP**
AirTag	0.640656	0.995702	0.991375	0.779661	14	1819
Bathroom	1.000000	0.961292	1.000000	0.980264	1045	0
Camera	1.000000	0.999937	1.000000	0.999969	2	0
Headphone	0.931880	0.999378	0.988580	0.964449	18	2115
Kitchen	1.000000	0.654645	1.000000	0.791282	3658	0
Laptop	1.000000	0.760788	1.000000	0.864145	7561	0
Smartphone	0.997755	0.571291	0.999781	0.726567	13338	40
Smartwatch	0.626538	0.999540	0.932637	0.770258	10	12961
TV	1.000000	0.282749	1.000000	0.440848	9467	0
iPad	0.449968	0.999261	0.908721	0.620517	11	18189

Table 5.10: Performance Metrics of the RF Classifier with Alternative Data Splitting Method

The confusion matrix is shown in Figure 5.30 which shows the misclassification on the off-diagonal. 1’045 bathroom devices are classified as smartwatches, 2’108 kitchen devices as headphones, 1’550 kitchen devices as smartwatches, 5’035 laptops as iPads, 13’135 smartphones as iPads, and 9’467 TVs as smartwatches. This means that half of all kitchen devices, almost half of all smartphones, and 3/4 of all TVs are misclassified by the model indicating that the model has difficulty learning the pattern of these device types to make predictions. The reason is the same as mentioned above, that the LG TVs and the Samsung TVs do not have a similar pattern in their data packets and the model cannot predict the device type of the Samsung TVs. The same applies to smartphones and kitchen devices. By excluding the devices with specific names of manufacturer from the training dataset and applying the learning pattern from the training data, the device type of kitchen devices, TVs, and smartphones is difficult to predict because they have a different pattern in their data.



	AirTag	Bathroom	Camera	Headphone	Kitchen	Laptop	Smartphone	Smartwatch	TV	iPad
AirTag	3243	0	0	0	0	0	0	6	0	8
Bathroom	0	25952	0	0	0	0	0	1045	0	0
Camera	0	0	31797	0	0	0	0	2	0	0
Headphone	7	0	0	28933	0	0	2	6	0	3
Kitchen	0	0	0	2108	6934	0	0	1550	0	0
Laptop	1655	0	0	1	0	24047	38	832	0	5035
Smartphone	151	0	0	6	0	0	17774	46	0	13135
Smartwatch	2	0	0	0	0	0	0	21744	0	8
TV	0	0	0	0	0	0	0	9467	3732	0
iPad	4	0	0	0	0	0	0	7	0	14880

Figure 5.30: Confusion Matrix of the RF Classifier with Alternative Data Splitting Method

Using the first most important feature selected by the embedded filter selection method and iteratively adding one feature at a time until the top 30 features are included, the number of features with the highest training accuracy of 99.9968% and the highest test accuracy of 87.8385% is 22. Comparing these accuracies with the accuracies of the model trained on all features shows that the test accuracy is higher and the training accuracy slightly lower but still high enough. This shows that training the RF model on the most important features increases the test accuracy by 4.2% which means that the model performs better on unseen data than the model trained on all features.

5.8.2 Multi-Layer Perceptron Model

MLP Model with 80/20 Data Splitting Method

Training the MLP model as described in Chapter 4.8.4, Table 5.11 shows the training accuracies, training losses, validation accuracies, and validation losses of epochs 1, 10, 20, 30, 40, and 50. The training accuracy in the first epoch is 97.62% and increases to a maximum of 99.69% in the 10th epoch. It remains stable for the remaining epochs and decreases slightly to 99.65% in the last epoch. The validation accuracy is 99.70% in the first epoch and increases to 99.74% in the 50th epoch. The training loss decreases from 0.1139 to 0.0255 and then increases to 0.0556 by the 50th epoch. The validation loss decreases from 0.0124 to 0.0072 and then fluctuates slightly, reaching a value of 0.0106 by the 50th epoch. Since the training and validation accuracies are very high and stable the model performs well on both the training and validation datasets. The training loss increases after the 20th epoch, while the validation loss remains low. The overall training accuracy is 99.766% and the test accuracy is 99.7682% showing that the model performs well on unseen data. Overall, the model shows good performance with high and stable training and validation accuracies with a slight increase in training loss. Since the training and test accuracies are almost the same value, there is no overfitting and the model has learned the patterns in the data rather than memorizing them.

Epoch	**Training Acc.**	**Training Loss**	**Validation Acc.**	**Validation Loss**
1/50	0.9762	0.1139	0.9970	0.0124
10/50	0.9969	0.0255	0.9976	0.0112
20/50	0.9968	0.0322	0.9978	0.0072
30/50	0.9968	0.0313	0.9973	0.0103
40/50	0.9968	0.0453	0.9975	0.0121
50/50	0.9965	0.0556	0.9974	0.0106

Table 5.11: Training Metrics of the MLP Model with 80/20 Data Splitting Method

The test accuracies of each device type are computed and are shown in Table 5.12. All accuracies are above 99.4%, except for AirTags, which have an accuracy of 98.0137%, and TVs, which have an accuracy of 98.8940%. The accuracies of AirTags and TVs are slightly lower than the accuracies of the other device types, but still high. The lower accuracy of AirTags could be due to the dataset containing less data from AirTag devices compared to other device types. Even when AirTags have the fewest data packets and therefore a lower accuracy, the data balancing technique is not applied because their accuracy is still very high. The reason for the lower accuracy for TVs could be that TVs have similar patterns in their data to other device types, leading to misclassification of TVs.

Device Type	**Test Accuracy**
AirTag	98.01365611421478
Bathroom	99.8695318495779
Camera	99.90100517819068
Headphone	99.87534626038781
Kitchen	99.82046678635548
Laptop	99.77060591678532
Smartphone	99.84120682810639
Smartwatch	99.96373197932723
TV	98.89404209775241
iPad	99.44007465671244

Table 5.12: Test Accuracies of each Device Type of the MLP Model with 80/20 Data Splitting Method

Precision, recall, specificity, F1-score, FP, and FN are computed and listed in

Table 5.13. All device types have precision values close to 1 showing that the model makes very few FP errors. The recall values are also close to 1 for all device types showing that the model seldom misclassified devices. Specificity values are close to 1 across all device types showing that the model correctly classifies devices with very few errors. F1-scores are also close to 1 for all device types showing a good balance between precision and recall. Overall the model performs very well with very few FP and FN and can distinguish between different device types.

Label	Precision	Recall	Specificity	F1-Score	FN	FP
AirTag	0.991834	0.980137	0.999853	0.985951	32	13
Bathroom	0.999923	0.998695	0.999987	0.999309	17	1
Camera	0.999924	0.999010	0.999987	0.999467	13	1
Headphone	0.999931	0.998753	0.999987	0.999342	18	1
Kitchen	1.000000	0.998205	1.000000	0.999102	7	0
Laptop	0.999762	0.997706	0.999961	0.998733	29	3
Smartphone	1.000000	0.998412	1.000000	0.999205	16	0
Smartwatch	0.984199	0.999637	0.997763	0.991858	4	177
TV	0.998919	0.988940	0.999966	0.993905	31	3
iPad	0.998661	0.994401	0.999879	0.996526	42	10

Table 5.13: Performance Metrics of the MLP Model with 80/20 Data Splitting Method

The confusion matrix is shown in Figure 5.31. The elements on the diagonal show the number of correct predictions for each device type. High diagonal elements show that the model accurately classifies data of those types. For example, the model correctly predicted 1'579 AirTags, 13'013 bathroom devices, and 13'119 cameras. The off-diagonal elements show the misclassification. For example, the model incorrectly classified 27 AirTags as a smartwatch device and 24 laptops also as a smartwatch device. Overall, there are very few misclassifications and very high values on the diagonal showing that the model performs well.

Training the MLP model with the first feature selected from the filter feature selection method from Table 5.4 and iteratively adding one more feature till all 30 features are included, the MLP model achieves the highest training accuracy of 99.7861% and the highest test accuracy of 99.8048% trained on the first 27 features. Comparing these accuracies with the accuracies of the MLP model trained on all features, both accuracies of the MLP model trained on the 27 selected features are slightly higher than the accuracies of the model trained on all features. This shows that training the MLP model with the selected features improves accuracy while reducing training time and computational costs.

MLP Model with Alternative Data Splitting Method

Training the MLP model based on the dataset generated by the alternative data splitting method, the training accuracy, training loss, validation accuracy, and validation loss of epochs 1, 10, 20, 30, 40, and 50 are shown in Table 5.14. The

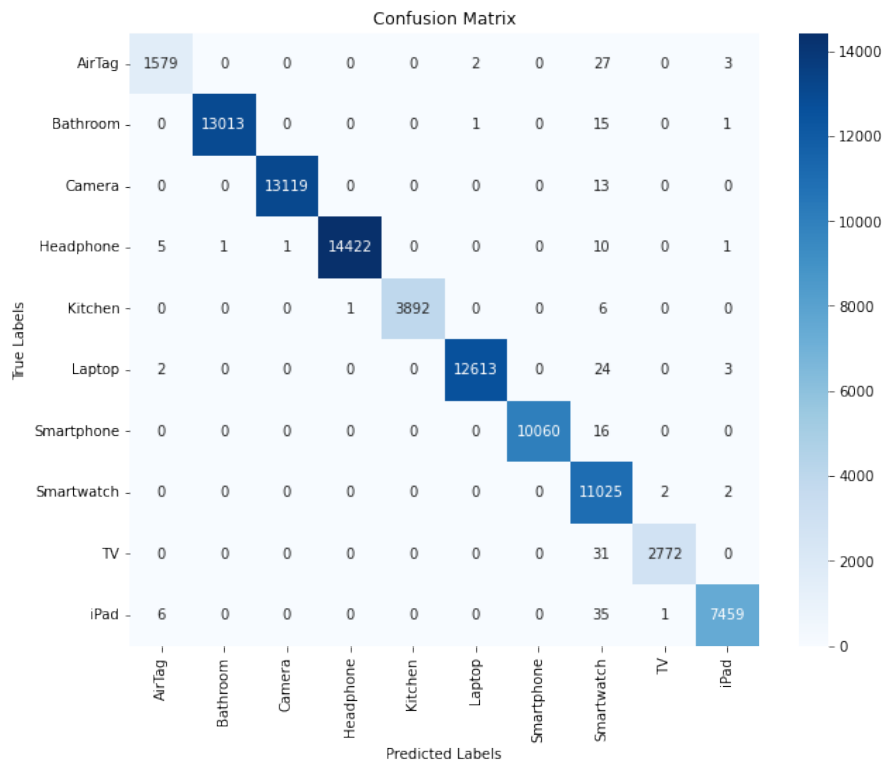


Figure 5.31: Confusion Matrix of the MLP Model with 80/20 Data Splitting Method

overall training accuracy is 84.7443% and the test accuracy is 77.6223%. The training accuracy ranges between 97.89% and 99.76% and is high across epochs. The training loss is low indicating that the model fits the training data well. On the other hand, the validation accuracy is very low and stays at the same value between 24.45% and 24.46%. The validation loss increases greatly from 60.3486 at epoch 1 to 1252.6884 at epoch 50. There is a large gap between the values of the training accuracy and validation accuracy, indicating overfitting. The model performs well on the training data but cannot generalize to the validation data. In addition, the increasing validation loss indicates that there is overfitting. The high increase of the validation loss over the epochs shows that the model may not learn the specific pattern of each device type from the training data. This leads to poor generalization as the model remembers the training data. Furthermore, the training accuracy is high but the test accuracy is very low, indicating that there is overfitting as the performance drops when the model should predict the device type of unseen data. As already mentioned, the reason could be that devices with specific names of manufacturer such as Samsung TVs are not included in the training dataset and the model is trained on data of other devices with different names of manufacturer such as LG TVs. Therefore, the model is not able to apply the learned pattern for example of LG TVs to Samsung TVs. To overcome this issue, the dataset should be more diverse, including more devices with different names of manufacturer in each device type category.

The test accuracies for each device type are shown in Table 5.15. While AirTags,

Epoch	Training Acc.	Training Loss	Validation Acc.	Validation Loss
1/50	0.9789	0.1076	0.2445	60.3486
10/50	0.9976	0.0230	0.2445	248.7953
20/50	0.9975	0.0242	0.2446	537.5908
30/50	0.9975	0.0334	0.2446	640.1320
40/50	0.9974	0.0790	0.2445	908.9257
50/50	0.9975	0.0340	0.2446	1252.6884

Table 5.14: Training Metrics of the MLP Model with Alternative Data Splitting Method

cameras, headphones, kitchen devices, smartwatches, and iPads have a very high accuracy of over 96%, laptops have an accuracy of 82.82%, smartphones of 80.2%, TVs of 33.3%, and bathroom devices of 0%. The high accuracy shows that the model performs well in classifying AirTags, cameras, headphones, kitchen devices, smartwatches, and iPads. The model misclassifies some laptops, smartphones, and TVs and cannot distinguish them well from other device types. An accuracy of 0% shows that the model cannot correctly identify and classify bathroom devices correctly and always misclassifies them. While laptops, TVs, smartphones, and bathroom devices have lower accuracy than other device types in the RF classifier with an alternative data splitting method, here laptops, TVs, smartphones, and bathroom devices have lower accuracy than other device types. The same reason as for the RF classifier with an alternative data splitting method applies. The 0% accuracy of bathroom devices could be because the AquaClean shower toilet device and the toothbrush do not have a similar pattern in their data packets, and the model cannot predict the device type of the AquaClean shower toilet device based on the learned pattern of the toothbrush. Since the reason for the low accuracy of laptops, TVs, smartphones, and bathroom devices is due to the data splitting approach, and these devices have enough data packets compared to the AirTags with high accuracy, the data balancing technique is not applied.

Device Type	Test Accuracy
AirTag	96.22351857537611
Bathroom	0.0
Camera	99.93396018742727
Headphone	99.83765673033747
Kitchen	98.60271903323263
Laptop	82.82396861554037
Smartphone	80.20377989200309
Smartwatch	99.56329870368668
TV	33.29797711947875
iPad	99.3217379625277

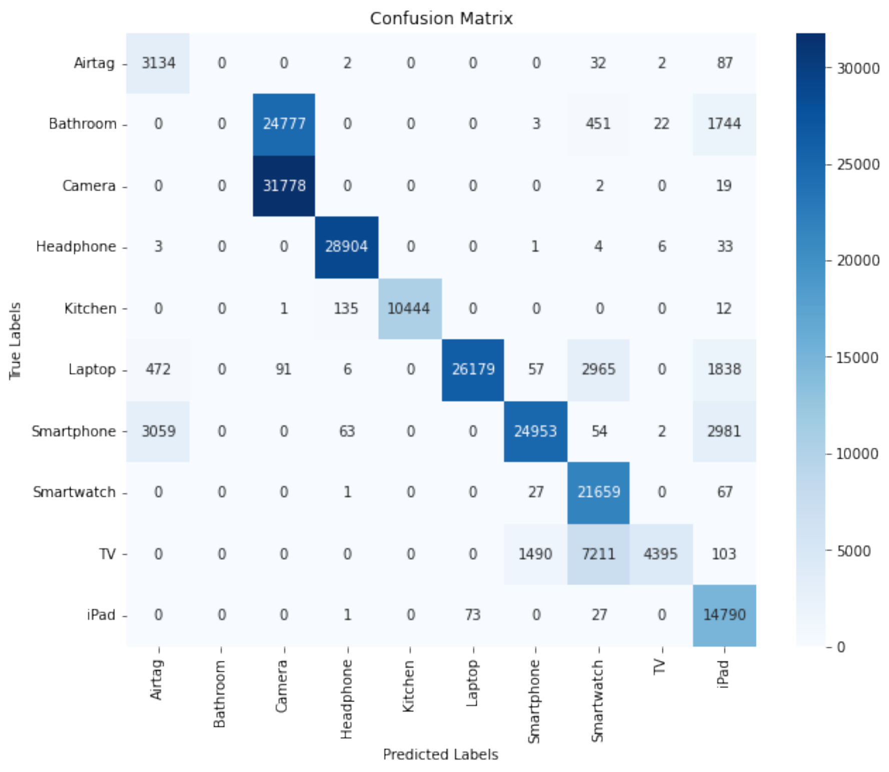
Table 5.15: Test Accuracies of each Device Type of the MLP Model with Alternative Data Splitting Method

The precision, recall, specificity, F1-score, FP, and FN of each device type are listed in Table 5.16. AirTags have high recall and specificity but low precision, indicating that the model predicts many FPs. Bathroom devices have a precision, recall, and F1-score of 0, and specificity of 1.00, indicating that the model fails to identify them. Cameras have high precision, recall, specificity, and F1-score, indicating that the model performs well in identifying them. Headphones have high recall but low precision, indicating a high FP value. Kitchen devices have high precision and specificity but low recall, showing that the model fails to correctly classify some kitchen devices. Laptops have high precision, specificity, F1-score, and good recall, showing that the model performs well in identifying laptops. Smartphones have high precision and specificity but low recall, showing that many smartphones are not identified by the model. Smartwatches have high recall and specificity but low precision, showing that the model predicted some FP. TVs have very high precision and specificity but very low recall, indicating that many TVs are not identified by the model. IPads have high recall and specificity but low precision, showing that the model also predicts many FP of them. High recall but low precision values of AirTags, headphones, smartwatches, and iPads show that the model predicts many FP on these types. Low recall but high precision values of kitchen devices, laptops, smartphones, and TVs show that many devices of these types are not identified by the model. The balanced performance values with high F1-scores such as cameras show a good balance between precision and recall and are well identified by the model.

Label	**Precision**	**Recall**	**Specificity**	**F1-Score**	**FN**	**FP**
AirTag	0.517115	0.997237	0.985619	0.681065	9	3033
Bathroom	0.000000	0.000000	1.000000	0.000000	26997	0
Camera	1.000000	0.999937	1.000000	0.999969	2	0
Headphone	0.496652	0.999033	0.841730	0.663470	28	29313
Kitchen	0.949043	0.734989	0.997947	0.828412	2807	418
Laptop	0.999958	0.760757	0.999995	0.864109	7562	1
Smartphone	0.997973	0.569748	0.999803	0.725375	13386	36
Smartwatch	0.610336	0.995633	0.928131	0.756765	95	13828
TV	0.971116	0.282749	0.999448	0.437977	9467	111
iPad	0.519576	0.994560	0.931279	0.682567	81	13694

Table 5.16: Performance Metrics of the MLP Model with Alternative Data Splitting Method

The confusion matrix of this MLP model is shown in Figure 5.32. On the diagonal is the number of correctly predicted data samples, and on the off-diagonal is the number of incorrectly predicted samples. The majority of data samples of AirTags, cameras, headphones, kitchen devices, laptops, smartwatches, and iPads are correctly classified into their device types, indicating that the model performs well for these device types. No data samples of bathroom devices are correctly predicted as bathroom devices, 24'777 data samples of them are misclassified as cameras, and 1'744 are misclassified as iPads. This shows that the model has difficulty distinguishing them based on their features. The misclassification of bathroom devices as cameras or iPads shows that the learned pattern of bathroom devices by the model is similar to the pattern of cameras and iPads, so the model cannot distinguish bathroom devices from them. Nearly 1/7 of the laptop data is misclassified as either smartwatches or iPads, and nearly 1/4 of the smartphone data is misclassified as either AirTag or iPad. It shows that the pattern of laptops is similar to the pattern of smartwatches and iPads, and the pattern of smartphones is similar to the pattern of AirTags or iPads, as the model has difficulty distinguishing between them. The majority of TV data are misclassified, mostly as smartwatches or smartphones, with only about 1/3 correctly predicted. The model cannot differentiate TVs and confuses them with smartwatches and smartphones.



	Airtag	Bathroom	Camera	Headphone	Kitchen	Laptop	Smartphone	Smartwatch	TV	iPad
Airtag	3134	0	0	2	0	0	0	32	2	87
Bathroom	0	0	24777	0	0	0	3	451	22	1744
Camera	0	0	31778	0	0	0	0	2	0	19
Headphone	3	0	0	28904	0	0	1	4	6	33
Kitchen	0	0	1	135	10444	0	0	0	0	12
Laptop	472	0	91	6	0	26179	57	2965	0	1838
Smartphone	3059	0	0	63	0	0	24953	54	2	2981
Smartwatch	0	0	0	1	0	0	27	21659	0	67
TV	0	0	0	0	0	0	1490	7211	4395	103
iPad	0	0	0	1	0	73	0	27	0	14790

Figure 5.32: Confusion Matrix of the MLP Model with Alternative Data Splitting Method

Training the MLP model using the first selected features by the filter feature selection method from Table 5.4 and iteratively adding one more feature till all 30 features are included, the highest training accuracy of 84.7625% and the highest test accuracy of 74.2188% is achieved by training the MLP model with 27 features. The training accuracy is slightly better but almost the same as the training accuracy of the MLP model trained on all features but the test accuracy drops by 3% compared to the test accuracy of the model trained on all features. The training accuracy is relatively high, which indicates that the model performs well in fitting the training data using the selected 27 features. The drop in test accuracy shows that the selected 27 features may not be good enough characteristics of the device types to train the model to generalize well to unseen data. Some other important information may be lost and the selected features may not fully represent the characteristics of the device types.

5.9 Comparing ML Models

Additionally, the DT[2], LR[3], and NB[4] are trained on the dataset generated by the 80/20 data splitting method with default parameters from the scikit-learn Python library. The KNN and SVM models could not be trained because training takes a very long time, at least 36 hours. All trained models are shown in Table 5.17. The RF classifier with an 80/20 data splitting approach has the highest test accuracy among all models which shows the best performance in classifying devices. The RF classifier with an 80/20 data splitting approach on selected features slightly reduces the accuracy compared to using all features showing that feature selection does not improve the performance of RF. The RF classifier with an alternative data splitting approach has a lower test accuracy than the RF classifier with an 80/20 data splitting method indicating that the model performance is dependent on the data splitting method. Since devices with specific names of manufacturer are only present in the test dataset in the alternative method, the model cannot apply the learned patterns from other devices of the same type to these devices, which means that the characteristics of the devices of the same type are not the same. If more devices per type and devices of different specific names of manufacturer are included in the training dataset, the model would perform better. The RF classifier with an alternative data splitting method on selected features shows a small increase in accuracy compared to using all features. This shows that feature selection increases the accuracy with the alternative data splitting method compared to the 80/20 data splitting method where feature selection decreases the accuracy.

The MLP model with an 80/20 data splitting method has a high accuracy, comparable to the RF classifier with the same data splitting method. This shows that the MLP model also classifies devices accurately. The MLP model with an 80/20 data splitting method on selected features achieves a higher accuracy compared to using all features showing that feature selection improves the MLP performance. On the other hand, the MLP model with an alternative data splitting method has a lower accuracy compared to the MLP model with an 80/20 data splitting method, which is consistent with the pattern shown in the RF classifier. The MLP model with an alternative data splitting method on selected features shows a decrease in accuracy compared to using all features indicating that feature selection may not be as effective with alternative data splitting.

Comparing the MLP model with an 80/20 data splitting method on all features with the MLP model from the research published in [45], the model from the research has an accuracy of 99.8% which is the same as the accuracy of this MLP model of 99.768% just rounding up. Since they extracted different features from

[2]`https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html`

[3]`https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`

[4]`https://scikit-learn.org/stable/modules/naive_bayes.html`

data packets of different devices as in this thesis, it is shown that the accuracies of the MLP models are the same.

DT achieves the second highest test accuracy, showing that the model performs well comparable to the RF classifier and the MLP models. LR has a relatively low accuracy compared to other models, indicating that the LR model does not perform well in classifying BLE devices and that the relationship between the features and the device type value may not be captured by a linear model. NB also has a low accuracy compared to other models indicating that NB has difficulty learning the pattern of the BLE device data and that the NB algorithm's assumptions of feature independence do not hold for this dataset. Overall, the RF classifier with an 80/20 data splitting method on all features, DT, and the MLP model with an 80/20 data splitting method on all features perform the best among the evaluated models.

ML Model	**Test Accuracy**
RF with 80/20 data splitting	99.972
RF with 80/20 data splitting on selected features	99.956
RF with alternative data splitting	83.599
Rf with alternative data splitting on selected features	87.839
MLP with 80/20 data splitting	99.768
MLP with 80/20 data splitting on selected features	99.805
MLP with alternative data splitting	77.622
MLP with alternative data splitting on selected features	74.219
MLP from research published in [45]	99.8
DT	99.961
LR	38.87
NB	39.86

Table 5.17: ML Models and their Accuracies

5.10 Summary of Limitations

As already mentioned, one of the limitations is that only devices that are portable and easy to obtain are included in the dataset. The reason for this is that the devices must be placed in an isolated room to sniff their data to enable the later process of identifying and labeling the data. To do this, no data from other unknown devices may be sniffed. This isolation of devices excludes many device types from the dataset, such as security cameras and other stationary devices that cannot be brought to an isolated room and are difficult to obtain. Furthermore, the dataset has more devices that are widely used and commonplace, such as headphones. The dataset could be skewed during the sniffing process and not represent the full range of different BLE devices.

Another limitation is that the data set does not contain enough different devices per device type. Splitting the dataset using the alternative data splitting method shows that the model fits the training data too well, but does not generalize well to new data. For example, if only Samsung and LG TVs are present and the Samsung TVs are excluded from the training data, the model cannot accurately predict the device type of Samsung TVs. By including more devices with different manufacturer names per device type, the dataset is more diverse and could overcome this limitation.

As already shown in Figure 5.6, devices of the same type, such as headphones, have quite different values. This shows that even devices of the same type can have different numbers of packets. The reason could be that some devices are actively used and transmit a lot of packets during the sniffing process, while others are inactive and only transmit a tiny amount of packets. This shows a limitation of capturing BLE packets because the number of packets of devices varies on how the devices are used during the sniffing process.

Chapter 6

Conclusion and Future Work

This chapter summarizes and concludes the main findings of this thesis. Furthermore, this chapter provides potential future work that can be done for further research.

6.1 Conclusion

Since the number of attacks on BLE devices is increasing and there are many security and privacy threats, there is a high need to identify and classify BLE devices. This thesis addresses the security and privacy threats of BLE devices by generating a dataset of different BLE devices and developing ML models to classify them. The motivation for this thesis is that most BLE devices are smart home devices that contain personal information, and since they are vulnerable to attacks, there is a high need to protect personal data.

The first objective is to capture BLE packets by sniffing over an extended period of time. This is achieved by first collecting BLE devices of different device types. In total, there are 10 types such as AirTags, cameras, headphones, smartphones, smartwatches, laptops, TVs, iPads, kitchen and bathroom devices, and 49 devices. The nRF Sniffer tool is used to collect BLE packets from each device. The second objective of preprocessing and creating a large dataset for analysis and pattern recognition is achieved by removing sniffed data that does not contribute to the classification task and transforming the data into a numerical representation. In addition, features that are characteristic of a device type are extracted from the dataset. Using feature selection methods, features such as *Number of Forward Packets*, *Number of Packets*, *Sum Payload Length*, *Sum Packet Length*, *Sum RSSI*, and *Sum Delta Time* are selected as the most important features to train the ML models. The final objective, to evaluate and develop ML models to classify and identify different BLE devices based on their characteristics, is achieved by developing RF classifiers and MLP models and training them on two different training datasets.

Training the RF classifier on the training dataset created by the 80/20 data splitting method achieves the highest accuracy of 99.972%. Training it with features selected by the embedded feature selection method does not improve the accuracy, but it is still high. Using the alternative data splitting method to create the training and test datasets, the accuracy of the RF classifier is reduced to 83.599%. The reason for the decrease in accuracy is that since devices with specific names of manufacturer, such as Samsung TVs, are not included in the training dataset, the model cannot apply the learned pattern of other TVs from the training dataset to unseen TVs.

The second highest accuracy of 99.961% is achieved by the DT classifier, followed by the MLP model trained on the training dataset created by the 80/20 data splitting method and with the first 27 features selected by the filter feature selection method with an accuracy of 99.805%. Again, using the alternative data splitting method to create the training and test datasets reduces the accuracy of the MLP model for the same reason as the RF classifier. Since the structure of the MLP model is the same as in the research published in [45], it is interesting to compare the accuracies. Since the accuracy from the research published in [45] is 99.8% and rounding up the accuracy of the MLP model in this thesis is also 99.8%, it shows that even with a different dataset and different extracted features, the MLP model in this thesis achieves the same accuracy.

6.2 Future Work

Several limitations of this thesis can be addressed for further research. Firstly, the generated dataset consists of only 10 device types. In the future, the dataset can be expanded to include more device types. Since the generated dataset only has easily portable devices to bring them into an isolated room, other devices such as security cameras and medical devices can also be included in the dataset to generate a more diverse dataset. This can be achieved by removing the condition that BLE devices should be brought into an isolated room to sniff their data packets. Since this condition arises because there is not enough information from the sniffed data to label the BLE packets, a solution should be found to allow other devices to be sniffed even if they are not brought into an isolated room.

In addition, each device type in the generated dataset has a minimum of 2 devices. This minimum can also be expanded to a larger number of devices per type so that there are enough devices with different names of manufacturer per device type. By increasing the number of devices per type, the accuracy of the trained ML models on the dataset generated by the alternative data splitting method could be increased. By training the ML models on different devices with different names of manufacturer and excluding the data packets of just one device with a specific name of manufacturer from the training dataset, the ML models can learn the pattern of device type based on the data packets of different devices with different names of manufacturer to accurately predict the device type of the excluded device.

Bibliography

[1] Celosia, Guillaume and Cunche, Mathieu. *Discontinued privacy: Personal data leaks in apple bluetooth-low-energy continuity protocols.* Proceedings on Privacy Enhancing Technologies, 2020, 26-46.

[2] Martin, Jeremy and Alpuche, Douglas and Bodeman, Kristina and Brown, Lamont and Fenske, Ellis and Foppe, Lucas and Mayberry, Travis and Rye, Erik C and Sipes, Brandon and Teplov, Sam. *Handoff all your privacy: A review of apple's bluetooth low energy continuity protocol.* arXiv preprint arXiv:1904.10600, 2019.

[3] Bai, Lei and Yao, Lina and Kanhere, Salil S and Wang, Xianzhi and Yang, Zheng. *Automatic device classification from network traffic streams of internet of things.* In: 2018 IEEE 43rd conference on local computer networks (LCN), 1-9, 2018.

[4] Bassey, Joshua and Adesina, Damilola and Li, Xiangfang and Qian, Lijun and Aved, Alexander and Kroecker, Timothy. *Intrusion detection for IoT devices based on RF fingerprinting using deep learning.* In: 2019 Fourth International Conference on Fog and mobile edge computing (FMEC), 98-104, 2019.

[5] Sobot, Srdjan and Ninkovic, Vukan and Vukobratovic, Dejan and Pavlovic, Milan and Radovanovic, Milos. *Machine Learning Methods for Device Identification Using Wireless Fingerprinting.* In: 2022 International Balkan Conference on Communications and Networking (BalkanCom), 183-188, 2022.

[6] Potts, Christian T. *Unsupervised Clustering of RF-Fingerprinting Features Derived from Deep Learning Based Recognition Models.* 2021.

[7] Nordic Semiconductor. *nRF Sniffer for Bluetooth LE/5/4.x/2.4GHz.* Trondheim, Norway. `https://infocenter.nordicsemi.com/pdf/nRF_Sniffer_BLE_UG_v4.1.x.pdf`. Accessed: June 15, 2024.

[8] PARTHAVI PARMAR. *Bluetooth Low Energy (BLE) Security and Privacy for IoT.* AUGUST 24, 2023. `https://www.einfochips.com/blog/bluetooth-low-energy-ble-security-and-privacy-for-iot/#:~:text=A%20BLE%20network%20can%20be,security%20gaps%20in%20the%20system`. Accessed: June 15, 2024.

[9] Gomez, Carles and Oller, Joaquim and Paradells, Josep. *Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology.* sensors, 12(9), 11734-11753, 2012.

[10] Afaneh, Mohammad. *Intro to Bluetooth low energy.* Novel Bits, 2018.

[11] Tosi, Jacopo and Taffoni, Fabrizio and Santacatterina, Marco and Sannino, Roberto and Formica, Domenico. *Performance evaluation of bluetooth low energy: A systematic review.* Sensors, 17(12), 2898, 2017.

[12] Sarkar, Sopan and Liu, Jianqing and Jovanov, Emil. *A robust algorithm for sniffing BLE long-lived connections in real-time.* In: 2019 IEEE Global Communications Conference (GLOBECOM), 1-6, 2019.

[13] Meidan, Yair and Bohadana, Michael and Shabtai, Asaf and Guarnizo, Juan David and Ochoa, Martín and Tippenhauer, Nils Ole and Elovici, Yuval. *ProfilIoT: A machine learning approach for IoT device identification based on network traffic analysis.* In: Proceedings of the symposium on applied computing, 506-509, 2017.

[14] Cvitić, Ivan and Peraković, Dragan and Periša, Marko and Gupta, Brij. *Ensemble machine learning approach for classification of IoT devices in smart home.* International Journal of Machine Learning and Cybernetics, 12(11), 3179-3202, 2021.

[15] Ahmet Aksoy and Mehmet Hadi Gunes, *Automated IoT device identification using network traffic*, In: *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, Pages: 1–7, Year: 2019, Organization: IEEE.

[16] Nesrine Ammar, Ludovic Noirie, and SÃ©bastien Tixeuil, *Autonomous identification of IoT device types based on a supervised classification*, In: *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, Pages: 1–6, Year: 2020, Organization: IEEE.

[17] Jorge Ortiz, Catherine Crawford, and Franck Le, *DeviceMien: network device behavior modeling for identifying unknown IoT devices*, In: *Proceedings of the International Conference on Internet of Things Design and Implementation*, Pages: 106–117, Year: 2019.

[18] Ola Salman, Imad H. Elhajj, Ali Chehab, and Ayman Kayssi, *A machine learning based framework for IoT device identification and abnormal traffic detection*, In: *Transactions on Emerging Telecommunications Technologies*, Volume: 33, Number: 3, Pages: e3743, Year: 2022, Publisher: Wiley Online Library.

[19] Yongxin Liu, Jian Wang, Jianqiang Li, Shuteng Niu, and Houbing Song, *Machine learning for the detection and identification of Internet of Things devices: A survey*, In: *IEEE Internet of Things Journal*, Volume: 9, Number: 1, Pages: 298–320, Year: 2021, Publisher: IEEE.

[20] Jaidip Kotak and Yuval Elovici, *IoT device identification using deep learning*, In: *13th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2020) 12*, Pages: 76–86, Year: 2021, Organization: Springer.

[21] Yair Meidan, Michael Bohadana, Asaf Shabtai, Martin Ochoa, Nils Ole Tippenhauer, Juan Davis Guarnizo, and Yuval Elovici, *Detection of unauthorized IoT devices using machine learning techniques*, In: *arXiv preprint arXiv:1709.04647*, Year: 2017.

[22] Matthias Cäsar, Tobias Pawelke, Jan Steffan, and Gabriel Terhorst, *A survey on Bluetooth Low Energy security and privacy*, In: *Computer Networks*, Volume: 205, Pages: 108712, Year: 2022, Publisher: Elsevier.

[23] Guillaume Celosia and Mathieu Cunche, *Saving private addresses: An analysis of privacy issues in the Bluetooth Low Energy advertising mechanism*, In: *Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, Pages: 444–453, Year: 2019.

[24] Novel Bits, *Bluetooth Low Energy (BLE) Complete Guide*, `https://novelbits.io/bluetooth-low-energy-ble-complete-guide/`, Accessed: December 5, 2023.

[25] Novel Bits, *Bluetooth Low Energy (BLE) Sniffer Tutorial*, `https://novelbits.io/bluetooth-low-energy-ble-sniffer-tutorial/`, Accessed: December 5, 2023.

[26] Spiceworks, *What is Bluetooth Low Energy?*, `https://www.spiceworks.com/tech/iot/articles/what-is-bluetooth-le/#:~:text=Bluetooth%20Low%20Energy%20or%20Bluetooth%20LE%20is%20defined%20as%20connectivity,devices%20across%20a%20short%2Drange`, Accessed: December 5, 2023.

[27] Kevin James, *Replay Attack vs Man-in-the-Middle Attack*, `https://cybersecurityforme.com/replay-attack-vs-man-in-the-middle-attack/#:~:text=In%20a%20replay%20attack%2C%20the,two%20parties%20without%20their%20knowledge`, Accessed: December 8, 2023.

[28] Lokajit Tikayat Ray, *Replay Attack*, `https://www.lokajittikayatray.com/post/replay-attack`, Accessed: December 5, 2023.

[29] Reda El Abbadi and Hicham Jamouli, *Takagi–Sugeno fuzzy control for a nonlinear networked system exposed to a replay attack*, *Mathematical Problems in Engineering*, **2021**, pp. 1–13, *Hindawi Limited*, (2021).

[30] Sode Pallavi and V Anantha Narayanan, *An overview of practical attacks on BLE Based IOT devices and their security*, in *2019 5th international conference on advanced computing & communication systems (ICACCS)*, IEEE, 2019, pp. 694–698.

[31] Jianliang Wu, Yuhong Nan, Vireshwar Kumar, Dave Jing Tian, Antonio Bianchi, Mathias Payer, and Dongyan Xu, {*BLESA*}*: Spoofing attacks against reconnections in Bluetooth Low Energy*, in *14th USENIX Workshop on Offensive Technologies (WOOT 20)*, 2020.

[32] Jianliang Wu, Yuhong Nan, Vireshwar Kumar, Mathias Payer, and Dongyan Xu, {*BlueShield*}*: Detecting spoofing attacks in Bluetooth Low Energy networks*, in *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, 2020, pp. 397–411.

[33] Sebastian Bräuer, Anatolij Zubow, Sven Zehl, Mehran Roshandel, and Soroush Mashhadi-Sohi, *On practical selective jamming of Bluetooth Low Energy advertising*, in *2016 IEEE Conference on Standards for Communications and Networking (CSCN)*, IEEE, 2016, pp. 1–6.

[34] Xiaobo Long and Biplab Sikdar, *A mechanism for detecting session hijacks in wireless networks*, *IEEE Transactions on Wireless Communications*, vol. 9, no. 4, pp. 1380–1389, 2010.

[35] Rupinder Gill, Jason Smith, and Andrew Clark, *Experiences in passively detecting session hijacking attacks in IEEE 802.11 networks*, in *ACSW Frontiers 2006: Proceedings of the 4th Australasian Symposium on Grid Computing and e-Research (AusGrid 2006) and the 4th Australasian Information Security Workshop (Network Security) (AISW-NetSec 2006) [CRPIT, Volume 54]*, pp. 221–230, 2006, published by Australian Computer Society.

[36] Romain Cayre, Florent Galtier, Guillaume Auriol, Vincent Nicomette, Mohamed Kaâniche, and Géraldine Marconato. *InjectaBLE: Injecting malicious traffic into established Bluetooth Low Energy connections.* In *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 388–399, 2021. `https://ieeexplore.ieee.org/document/9505586` (Datum: December 30, 2023)

[37] Talon Flynn, George Grispos, William Glisson, and William Mahoney. *Knock! knock! who is there? Investigating data leakage from a medical internet of things hijacking attack. Year:* 2020. (Datum: December 30, 2023)

[38] Aellison CT Santos, José L Soares Filho, Ávilla ÍS Silva, Vivek Nigam, and Iguatemi E Fonseca. *BLE injection-free attack: a novel attack on Bluetooth Low Energy devices. Journal of Ambient Intelligence and Humanized Computing*, pages 1–11, 2019. `https://link.springer.com/article/10.1007/s12652-019-01270-w` (Datum: December 30, 2023)

[39] Harry OâSullivan. *Security vulnerabilities of Bluetooth Low Energy technology (BLE). Journal:* Tufts University, Year: 2015. (Datum: December 30, 2023)

[40] Arash Tayebi, SM Berber, and Akshya Swain. *Wireless sensor network attacks: An overview and critical analysis with detailed investigation on jamming attack effects. Journal:* Sensing Technology: Current Status and Future Trends III, pages 201–221, 2015. `https://link.springer.com/chapter/10.1007/978-3-319-15693-6_13` (Datum: December 30, 2023)

[41] Yongle Wang and JunZhang Chen. *Hijacking spoofing attack and defense strategy based on Internet TCP sessions.* In *2013 2nd International Symposium on Instrumentation and Measurement, Sensor Network and Automation (IMSNA)*, pages 507–509, 2013. `https://ieeexplore.ieee.org/document/6781313` (Datum: December 30, 2023)

[42] Shuhua Deng, Xing Gao, Zebin Lu, and Xieping Gao. *Packet injection attack and its defense in software-defined networks. Journal:* IEEE Transactions on Information Forensics and Security, *Volume:* 13, *Number:* 3, pages 695–705, 2017. `https://ieeexplore.ieee.org/document/7821812` (Datum: December 30, 2023)

[43] Jishuai Li, Sujuan Qin, Tengfei Tu, Hua Zhang, and Yongsheng Li. *Packet injection exploiting attack and mitigation in software-defined networks. Journal:* Applied Sciences, *Volume:* 12, *Number:* 3, page 1103, 2022. `https://www.mdpi.com/2076-3417/12/3/1103` (Datum: December 30, 2023)

[44] Marchal, Samuel and Miettinen, Markus and Nguyen, Thien Duc and Sadeghi, Ahmad-Reza and Asokan, N. *Audi: Toward autonomous IoT device-type identification using periodic communication. IEEE Journal on Selected Areas in Communications*, 2019, 37(6), 1402-1412. (Datum: January 18, 2024).

[45] Zhang, Jinghui and Li, Xinyang and Li, Junhe and Dai, Qiangsheng and Ling, Zhen and Yang, Ming. *Bluetooth Low Energy Device Identification Based on Link Layer Broadcast Packet Fingerprinting. Tsinghua Science and Technology*, 2023, 28(5), 1-11. (Datum: January 18, 2024).

[46] Al Qathrady, Mimonah and Helmy, Ahmed. *Improving BLE distance estimation and classification using TX power and machine learning: A comparative analysis.* In *Proceedings of the 20th ACM International Conference on Modelling, Analysis and Simulation of Wireless and Mobile Systems*, 2017, 79-83. (Datum: January 18, 2024).

[47] Redondi, Alessandro Enrico Cesare and Sanvito, Davide and Cesana, Matteo. *Passive classification of Wi-Fi enabled devices.* In *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2016, 51-58. (Datum: January 18, 2024).

[48] Takasaki, Chikako and Korikawa, Tomohiro and Hattori, Kyota and Ohwada, Hidenari. *Traffic Behavior-based Device Type Classification.* In *2023 International Conference on Computing, Networking and Communications (ICNC)*, 2023, 353-357. (Datum: January 18, 2024).

[49] Takasaki, Chikako and Korikawa, Tomohiro and Hattori, Kyota and Ohwada, Hidenari. *Device Type Classification Based on Two-Stage Traffic Behavior Analysis. IEICE Transactions on Communications*, 2024, 107(1), 117-125. (Datum: January 18, 2024).

[50] Maiti, Rajib Ranjan and Siby, Sandra and Sridharan, Ragav and Tippenhauer, Nils Ole. *Link-layer device type classification on encrypted wireless*

traffic with COTS radios. In *European Symposium on Research in Computer Security*, 2017, 247-264. (Datum: January 18, 2024).

[51] Gagnon, Guillaume and Gambs, SÃ©bastien and Cunche, Mathieu. *RSSI-based Fingerprinting of Bluetooth Low Energy Devices.* In *International Conference on Security and Cryptography (SECRYPT 2023)*, 2023. (Datum: January 18, 2024).

[52] Bezawada, Bruhadeshwar and Bachani, Maalvika and Peterson, Jordan and Shirazi, Hossein and Ray, Indrakshi and Ray, Indrajit. *Behavioral fingerprinting of IoT devices.* In *Proceedings of the 2018 workshop on attacks and solutions in hardware security*, 2018, 41-50. (Datum: January 18, 2024).

[53] Sakr, Fouad and Berta, Riccardo and Doyle, Joseph and De Gloria, Alessandro and Bellotti, Francesco. *Self-Learning Pipeline for Low-Energy Resource-Constrained Devices. Energies*, 2021, 14(20), 6636. (Datum: January 18, 2024).

[54] Jiang, Xinlong and Chen, Yiqiang and Liu, Junfa and Gu, Yang and Hu, Lisha. *FSELM: Fusion Semi-Supervised Extreme Learning Machine for Indoor Localization with Wi-Fi and Bluetooth Fingerprints. Soft Computing*, 2018, 22, 3621-3635. (Datum: January 18, 2024).

[55] Mohammadi, Mehdi and Al-Fuqaha, Ala and Guizani, Mohsen and Oh, Jun-Seok. *Semisupervised Deep Reinforcement Learning in Support of IoT and Smart City Services. IEEE Internet of Things Journal*, 2017, 5(2), 624-635. (Datum: January 18, 2024).

[56] Gu, Tianbo and Mohapatra, Prasant. *Bf-IoT: Securing the IoT Networks via Fingerprinting-Based Device Authentication.* In *2018 IEEE 15th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, 2018, 254-262. (Datum: January 18, 2024).

[57] Aragon Jr, Angelito E. *Evaluating Machine Learning Techniques for Smart Home Device Classification. Year*, 2019. (Datum: January 18, 2024).

[58] Shahid, Mustafizur R., Blanc, Gregory, Zhang, Zonghua, and Debar, HervÃ©. "IoT devices recognition through network traffic analysis." In *2018 IEEE international conference on big data (big data)*, pp. 5187-5192. IEEE, 2018. `https://doi.org/10.1109/BigData.2018.8622604`.

[59] Pasyuk, A., Semenov, E., and Tyuhtyaev, D. "Feature selection in the classification of network traffic flows." In *2019 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon)*, pp. 1-5. IEEE, 2019. `https://doi.org/10.1109/FarEastCon.2019.8934964`.

[60] Muhammad Yaseen Khan, Abdul Qayoom, Muhammad Suffian Nizami, Muhammad Shoaib Siddiqui, Shaukat Wasi, and Syed Muhammad Khaliq-ur-Rahman Raazi. *Automated prediction of Good Dictionary EXamples (GDEX): a comprehensive experiment with distant supervision, machine learning, and*

word embedding-based deep learning techniques. Complexity, volume 2021, pages 1–18, 2021. Hindawi Limited.

[61] Sotiris B Kotsiantis, Ioannis Zaharakis, P. Pintelas, and others. *Supervised machine learning: A review of classification techniques. Emerging artificial intelligence applications in computer engineering*, 160(1):3–24, 2007. Amsterdam.

[62] Sotiris B Kotsiantis, Ioannis D Zaharakis, and Panayiotis E Pintelas. *Machine learning: a review of classification and combining techniques. Artificial Intelligence Review*, 26:159–190, 2006. Springer.

[63] Vladimir Nasteski. *An overview of the supervised machine learning methods. Horizons. b*, 4(51-62):56, 2017.

[64] Amanpreet Singh, Narina Thakur, and Aakanksha Sharma. *A review of supervised machine learning algorithms.* In *2016 3rd international conference on computing for sustainable global development (INDIACom)*, pages=1310–1315, year=2016. IEEE.

[65] Batta Mahesh. *Machine learning algorithms-a review. International Journal of Science and Research (IJSR).[Internet]*, volume=9, number=1, pages=381–386, year=2020.

[66] Tammy Jiang, Jaimie L Gradus, and Anthony J Rosellini. *Supervised machine learning: a brief primer. Behavior therapy*, volume=51, number=5, pages=675–687, year=2020. Elsevier.

[67] Sivanathan, Arunan and Gharakheili, Hassan Habibi and Loi, Franco and Radford, Adam and Wijenayake, Chamith and Vishwanath, Arun and Sivaraman, Vijay. *Classifying IoT devices in smart environments using network traffic characteristics.* IEEE Transactions on Mobile Computing, 2018, 18(8), 1745-1759.

[68] Chawla, Nitesh V and Bowyer, Kevin W and Hall, Lawrence O and Kegelmeyer, W Philip. *SMOTE: synthetic minority over-sampling technique.* Journal of Artificial Intelligence Research, 2002, 16, 321–357.

[69] Buda, Mateusz and Maki, Atsuto and Mazurowski, Maciej A. *A systematic study of the class imbalance problem in convolutional neural networks.* Neural Networks, 2018, 106, 249–259, Elsevier.

[70] Dahouda, Mwamba Kasongo and Joe, Inwhee. *A deep-learned embedding technique for categorical features encoding.* IEEE Access, 2021, 9, 114381–114391, IEEE.

[71] Jović, Alan and Brkić, Karla and Bogunović, Nikola. *A review of feature selection methods with applications.* In: 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2015, 1200–1205, IEEE.

[72] Bolón-Canedo, Verónica and Sánchez-Maroño, Noelia and Alonso-Betanzos, Amparo. *A review of feature selection methods on synthetic data.* Knowledge and Information Systems, 2013, 34, 483–519, Springer.

[73] Wong, Tzu-Tsung. *Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation.* Pattern Recognition, 2015, 48(9), 2839–2846, Elsevier.

[74] Rainio, Oona and Teuho, Jarmo and Klén, Riku. *Evaluation metrics and statistical tests for machine learning.* Scientific Reports, 2024, 14(1), 6086, Nature Publishing Group UK London.

[75] Heydarian, Mohammadreza and Doyle, Thomas E and Samavi, Reza. *MLCM: Multi-label confusion matrix.* IEEE Access, 2022, 10, 19083–19095, IEEE.

[76] Bluetooth Special Interest Group. *Core Specification 5.3.* Available at: `https://www.bluetooth.com/de/specifications/specs/core-specification-5-3/`. Accessed: May 29, 2024.

[77] Bluetooth Special Interest Group. *Core Specification Supplement 10.* Available at: `https://www.bluetooth.com/specifications/specs/core-specification-supplement-10/`. Accessed: May 29, 2024.

[78] Raschka, Sebastian. *Model evaluation, model selection, and algorithm selection in machine learning.* arXiv preprint arXiv:1811.12808, 2018.

Abbreviations

AA	Access Address
AD	Advertising
ADVB	Advertising Broadcast
AdvA	Advertising Address
AdvData	Advertising Data
ADV DIRECT IND	advertising direct indications
AI	Artificial Intelligence
ANN	Artificial Neural Network
AP	Access Point
ATTA	Attribute Protocol
BDADDR	Bluetooth Device Address
BLE	Bluetooth Low Energy
ChSel	Channel Selection
CNN	Convolutional Neural Network
CONNECT REQ	connection requests
CRC	Cycling Redundancy Check
DOS	Denial-of-Service
DT	Decision Tree
FN	False Negative
FP	False Positive
GA	Genetic Algorithm
GAP	Generic Access Profile
GATT	Generic Attribute Profile
GB	Gradient Boosting
HCI	Host Controller Interface
IFI	Department of Informatics
IG	Information Gain
IoT	Internet of Things
ISM	Industrial Scientific Medical
KNN	k-Nearest Neighbor
L2CAP	Logical Link Control and Adaptation Protocol
LDA	Linear Discriminant Analysis
LL	Link Layer
LR	Logistic Regression
LSTM	Long Short-Term Memory
ML	Machine Learning

MLP	Multi-Layer Perceptron
MITM	Man-in-the-Middle
NB	Naive Bayes
PDU	Protocol Data Unit
PHY	Physical Layer
ReLU	Rectified Linear Unit
RF	Radio Frequency
RF	Random Forest
RFU	Reserved for Future Use
RSS	Received Signal Strength
RSSI	Received Signal Strength Indication
SCAN REQ	scan request
SIG	Special Interest Group
SKNET	Selective Kernel Network
SMP	Security Manager Protocol
SVM	Support Vector Machine
TP	True Positive
TN	True Negative
TxAdd	Transmitting Address
TCP	Transmission Control Protocol
UZH	University of Zurich
UUID	Universally Unique Identifier

Glossary

BLE Device Any device that has BLE capabilities, allowing for energy-efficient short-range wireless communication.

List of Figures

3.1 BLE Protocol Stack [11] 14

3.2 Distribution of Advertisement Channels over the ISM Band [33] . . 17

3.3 Packet Format of the Uncoded PHY [76] 17

3.4 PDU Header of the Advertising Physical Channel 18

3.5 Payload of PDU Type of ADV_IND 18

3.6 AdvData Structure 19

3.7 MITM Attack [28] 21

3.8 Replay Attack [28] 21

3.9 Spoofing [32] 22

3.10 Session Hijacking [41] 23

3.11 RF Classifier [60] 25

3.12 Neural Network [70] 27

4.1 Methodology Overview 29

4.2 Feature Selection Methods [72] 37

4.3 Dataset of BLE Devices from Research published in [45] 43

4.4 Structure of the MLP Model [45] 44

5.1 Number of Devices per Device Type 48

5.2 Packet Detail of Sniffed Packets 49

5.3 Sniffed Packets from an Apple iPad and a Xiaomi Kettle visualized in Wireshark 54

5.4 Number of Packets by Device Types 56

5.5 Number of Packets by Devices 57
5.6 Number of Packets by Devices without Outliers 58
5.7 Number of Packets by *Length of Payload* Feature 59
5.8 Number of Packets by *Channel Index* Feature 60
5.9 Number of Packets by *RSSI* Feature 60
5.10 Number of Packets by *Delta Time* Feature 61
5.11 Number of Packets by *Info* Feature 62
5.12 Number of Packets by *Length* Feature 62
5.13 Number of Packets by *Power Level* Feature 63
5.14 Number of Packets by *Type* Feature 64
5.15 Number of Packets by *Packets per Second* Feature 65
5.16 Number of Packets by *Time per Packet* Feature 66
5.17 Number of Packets by *Bytes per Second* Feature 67
5.18 Number of Packets by *Average RSSI* Feature 68
5.19 Number of Packets by *Standard Deviation RSSI* Feature 68
5.20 Number of Packets by *Variance RSSI* Feature 69
5.21 Number of Packets by *Average Packet Length* Feature 70
5.22 Number of Packets by *Standard Deviation Packet Length* Feature . 71
5.23 Number of Packets by *Variance Packet Length* Feature 71
5.24 Number of Packets by *Average Delta Time* Feature 72
5.25 Number of Packets by *Standard Deviation Delta Time* Feature . . . 73
5.26 Total Packets of Training and Test Datasets 79
5.27 Learning Curves of the RF Classifer with 80/20 Data Splitting Method 81
5.28 Confusion Matrix of the RF Classifier with 80/20 Data Splitting Method 83
5.29 Learning Curves of the RF Classifier with Alternative Data Splitting Method 84
5.30 Confusion Matrix of the RF Classifier with Alternative Data Splitting Method 86

5.31 Confusion Matrix of the MLP Model with 80/20 Data Splitting Method . . . 89

5.32 Confusion Matrix of the MLP Model with Alternative Data Splitting Method . . . 92

E.1 Number of Packets by *Reserved* Feature . . . 127

E.2 Number of Packets by *Channel Selection Algorithm* Feature . . . 127

E.3 Number of Packets by *Tx Address* Feature . . . 128

E.4 Number of Packets by *Rx Address* Feature . . . 128

E.5 Number of Packets by *Simultaneous LE and BR EDR to Same Device Capable (Host)* Feature . . . 128

E.6 Number of Packets by *Number of Packets by Simultaneous LE and BR EDR to Same Device Capable (Controller)* Feature . . . 129

E.7 Number of Packets by *Number of Packets by BR EDR Not Supported* Feature . . . 129

E.8 Number of Packets by *LE General Discoverable Mode* Feature . . . 129

E.9 Number of Packets by *LE Limited Discoverable Mode* Feature . . . 130

F.1 Number of Packets by *Duration* Feature . . . 131

F.2 Number of Packets by *Minimum RSSI* Feature . . . 131

F.3 Number of Packets by *Maximum RSSI* Feature . . . 132

F.4 Number of Packets by *Sum RSSI* Feature . . . 132

F.5 Number of Packets by *Minimum Packet Length* Feature . . . 132

F.6 Number of Packets by *Maximum Packet Length* Feature . . . 133

F.7 Number of Packets by *Sum Packet Length* Feature . . . 133

F.8 Number of Packets by *Minimum Payload Length* Feature . . . 133

F.9 Number of Packets by *Maximum Payload Length* Feature . . . 134

F.10 Number of Packets by *Sum Payload Length* Feature . . . 134

F.11 Number of Packets by *Average Packet Length* Feature . . . 134

F.12 Number of Packets by *Standard Deviation Payload Length* Feature . 135

F.13 Number of Packets by *Variance Payload Length* Feature . . . 135

F.14 Number of Packets by *Minimum Delta Time* Feature 135

F.15 Number of Packets by *Maximum Delta Time* Feature 136

F.16 Number of Packets by *Sum Delta Time* Feature 136

F.17 Number of Packets by *Variance Delta Time* Feature 137

F.18 Number of Packets by *Packet Direction* Feature 137

F.19 Number of Packets by *Number of Forward Packets* Feature 137

F.20 Number of Packets by *Number of Backward Packets* Feature 138

F.21 Number of Packets by *Average Number of Forward Packet* Feature . 138

F.22 Number of Packets by *Average Number of Backward Packet* Feature 138

F.23 Number of Packets by *Minimum Delta Time* Feature of Forward Packets 139

F.24 Number of Packets by *Maximum Delta Time* Feature of Forward Packets 139

F.25 Number of Packets by *Sum Delta Time* Feature of Forward Packets 140

F.27 Number of Packets by *Standard Deviation Delta Time* Feature of Forward Packets 140

F.28 Number of Packets by *Variance Delta Time* Feature of Forward Packets 141

F.29 Number of Packets by *Minimum Delta Time* Feature of Backward Packtes 141

F.30 Number of Packets by *Maximum Delta Time* Feature of Backward Packtes 142

F.31 Number of Packets by *Sum Delta Time* Feature of Backward Packtes142

F.32 Number of Packets by *Average Delta Time* Feature of Backward Packtes 142

F.33 Number of Packets by *Standard Deviation Delta Time* Feature of Backward Packtes 143

F.34 Number of Packets by *Variance Delta Time* Feature of Backward Packtes 143

List of Tables

2.1 Summary of Related Work 10

3.1 PDU Types and their Descriptions 18

4.1 BLE Device Types and Their Name of Manufacturer 30

4.2 Features in the BLE LL 33

4.3 Hyperparameters for the RF Classifier 42

5.1 Stored PCAP Files of Sniffed BLE Packets 53

5.2 BLE devices and their Identifiers 55

5.3 Categorical Features and their Number of Unique Values 76

5.4 Filter Method: Selected Features and Their Scores 77

5.5 Embedded Method: Selected Features and Their Importance Scores 78

5.6 Devices in Training and Test Datasets 80

5.7 Test Accuracies of each Device Type of the RF Classifer with 80/20 Data Splitting Method 81

5.8 Performance Metrics of the RF Classifier with 80/20 Data Splitting Method 82

5.9 Test Accuracies of each Device Type of the RF Classifier with Alternative Data Splitting Method 84

5.10 Performance Metrics of the RF Classifier with Alternative Data Splitting Method 85

5.11 Training Metrics of the MLP Model with 80/20 Data Splitting Method 87

5.12 Test Accuracies of each Device Type of the MLP Model with 80/20 Data Splitting Method 87

5.13 Performance Metrics of the MLP Model with 80/20 Data Splitting Method 88

5.14 Training Metrics of the MLP Model with Alternative Data Splitting Method 90

5.15 Test Accuracies of each Device Type of the MLP Model with Alternative Data Splitting Method 90

5.16 Performance Metrics of the MLP Model with Alternative Data Splitting Method 91

5.17 ML Models and their Accuracies 94

B.1 Full List of Related Work 122

Appendix A

Installation Guidelines

A.1 nRF Sniffer Tool

In order to sniff the data packets of BLE devices with the nRF Sniffer tool, the *nRF Sniffer for Bluetooth LE* application should be installed from the Nordicsemi[1] website in the *Downloads* section. The section *Documentation* describes the minimum requirements, installation, running the nRF sniffer, using the nRF sniffer, and common sniffing actions. Furthermore, the Wireshark application should also be installed from the Wireshark[2] website.

A.2 Running the Application

Jupyter Notebook and Python v3.12 should be installed to run the application. All dependencies needed to run the application are included in the Python files in the GitHub repository. The generated dataset is a CSV file that should be downloaded from the link provided in the README file in the GitHub repository. The file path to the dataset should be adjusted in the Python files to read the dataset correctly.

[1]`https://www.nordicsemi.com/Products/Development-tools/nrf-sniffer-for-bluetooth-le`

[2]`https://www.wireshark.org/`

Appendix B

Full List of Related Work

Paper	No. of Devices	Type of Devices	Features	Model
[3]	21	4 types	Packet Number, Packet Length Average, Packet Length Peak, Control Packet Number, Control Packet Average, Control Packet Peak	N/A
[13]	9	PCs, Smartphones	TCP packets (4-tuples of Source and Destination IP Addresses and Port Numbers, from SYN to FIN)	Multi-stage Meta Classifier
[14]	41	4 types of Smart Home Environment	13 Network Traffic Features	LR, Logiboost
[15]	23	N/A	212 features	DT
[16]	6	Camera, Plug, Switch	Average Packet Length, Average of Inter-arrival Times of Packets in a Flow, Flowâs Size: Number of Packets in a Flow, Protocols used by the Flow, Manufacturer Name from MAC Address, Device Name	DT, NB, LR, SVM, and RDF
[17]	N/A	Cameras, Switches, Triggers, Hubs, Air Quality Sensors, Healthcare, Light Bulbs, Laptops, Smartphones Tablets	Domain Names, Port Numbers, and Cypher Suites	NB, LSTM-Autoencoder

Continued on next page

Table B.1 – continued from previous page

Paper	No. of Devices	Type of Devices	Features	Model
[18]	7	2 types	Packets' Direction, Size, Timestamp, Transport Protocol, Forward Direction, Backward Direction	RF, DT, RNN, ResNet, and ConvNet
[19]	N/A	N/A	N/A	KNN, SVM, RF
[?]	9	9 IoT (Smartphones, Computers), 1 non-Iot		Multiclass Classifier
[21]	17	9 types	300 features	RF
[58]	4	Security Camera, Motion Sensor, Smart Bulb, and Smart Plug	38 features: Size of the first N Packets Sent, Size of the first N Packets Received, N - 1 Packet Inter-arrival Times between the first N Packets Sent, N - 1 Packet Inter-arrival Times between the first N Packets Received	RF, DT, SVM, KNN, ANN, and NB
[59]	N/A	N/A	37 features: Protocol, Destination and Aource Ports, Amount of Packets, Bytes per Flow, Amount of Packets without Quaternary Data (TCP/UDP), Flow Starting Time, Flow Ending Time, Duration, Average Speed in Packets and Bytes, Max/Min/Avg/Std of Packet Sizes and Interpacket Gaps, Amount of TCP Packets, Sizes of the first 10 Packets	DT, SVM, NB, kNN, RF, GB

Continued on next page

Table B.1 – continued from previous page

Paper	**No. of Devices**	**Type of Devices**	**Features**	**Model**
[67]	28	Cameras, Lights, Plugs, Motion Sensors, Appliances, and Health-Monitors	Source/Destination Port Number, Payload Volume, TCP Window Size, Inter-arrival Time and Direction of Traffic, Average Time Between Successive Flows, Flow Duration, Inbound/Outbound Traffic Volume, Packet Size, Byte Distribution of Payload, Inter-arrival Times of Packets	N/A
[44]	N/A	23 types: IP Cameras, Smart Power Plugs, Light Bulbs, Sensors	v: Flow as a Sequence of Network Packets Sent from a given Source MAC Address, Periodic Flow, Period Accuracy, Period Duration, Period Stability	Unsupervised Clustering
[45]	23	7 types: Smart Bands, Smartwatches, Smart Oximeters, Smart Humidifiers, Headphones, Smartphones, and PCs	PDU and the Channel Selection from the Packet Header, the Type of AdvData, Length of the Type of AdvData, Flags, Service Data, Service UUID, Manufacturer Specific, Appearance, and Power Level	MLP
[47]	N/A	Smartphones and Laptops	Inter-Probe Period (IPP), Received Signal Strength (RSS), Coefficients of Variation, Number of Probe Requests with broadcast/known SSID, Device Manufacturer, Probe Request Frames	RF, NB, SVM, DT
[48]	12	7 types: Hubs, Cameras, Switches, Triggers, Air Quality Sensors, Healthcare, Light bulbs, Smartwatches and Router	Number of Packets, Traffic Send Rate, Number of Destination Addresses, Variation of Number of Send Packets, Sum of Packet Lengths, Average of Packet Lengths, Number of Protocol Types, and Number of Destination Addresses	LR, RF, SVM, MLP, LSTM, CNN
[50]	22	10 types	Header Information, Frame Size and the Timestamp of Frameâs Capture	RF, DT, and SVM.

Continued on next page

Table B.1 – continued from previous page

Paper	**No. of Devices**	**Type of Devices**	**Features**	**Model**
[52]	14	7 types	TCP Window Size, Entropy and Payload Lengths, Packet Header Feature, TCP Payload Length	KNN, DT, GB, and Majority voting
[57]	N/A	6 types: Door Sensors, Locks, Temperature Sensors, Smart Bulbs, Cameras, and Smart Plugs	7 features: Packet Length, BLE Link Layer Header Length, Packet Count, RF Channel Number, PDU type, Device Name, and Packet Time	KNN, LDA, RF
[55]	13	iBeacon	iBeaconsâ RSSI	Autoencoder
[56]	N/A	N/A	Advertising eEvent Interval, Advertising Channel Sequence: PDU, Advertising Delay Distribution, Throughput, Number Packets Sent, Burst Rate	RF

Table B.1: Full List of Related Work

Appendix C

Direct Feature Extraction

- Length of Payload
- Channel Index
- RSSI
- Delta time (μs end to start)
- Packet Header
  - PDU Type
  - Info
  - Reserved
  - Channel Selection Algorithm
  - Tx Address
  - Rx Address
  - Packet Header Length
- Scanning Address
- AdvData
  - AdvData Type
  - Length of the AdvData Type
  - Simultaneous LE and BR/EDR to Same Device Capable (Host)
  - Simultaneous LE and BR/EDR to Same Device Capable (Controller)
  - BR/EDR Not Supported
  - LE General Discoverable Mode
  - LE Limited Discoverable Mode
  - Power Level (dBm)

  - Data
  - Service Data
  - BD_ADDR
- CRC

Appendix D

Derived Feature Extraction

- Duration
- Number of Packets
- Packets per Second
- Time per Packet
- Bytes per Second
- Packet Header Length: Maximum, Minimum, Average, Standard Deviation, Variance
- RSSI: Maximum, Minimum, Average, Standard Deviation, Variance
- Length of Payload: Maximum, Minimum, Average, Standard Deviation, Variance
- Delta Time: Maximum, Minimum, Average, Standard Deviation, Variance
- Packet Direction
- Number of Forward Packets
- Number of Backward Packets
- Delta Time of Forward Packets: Maximum, Minimum, Average, Standard Deviation, Variance
- Delta Time of Forward Packets: Maximum, Minimum, Average, Standard Deviation, Variance

Appendix E

Results of Direct Feature Extraction

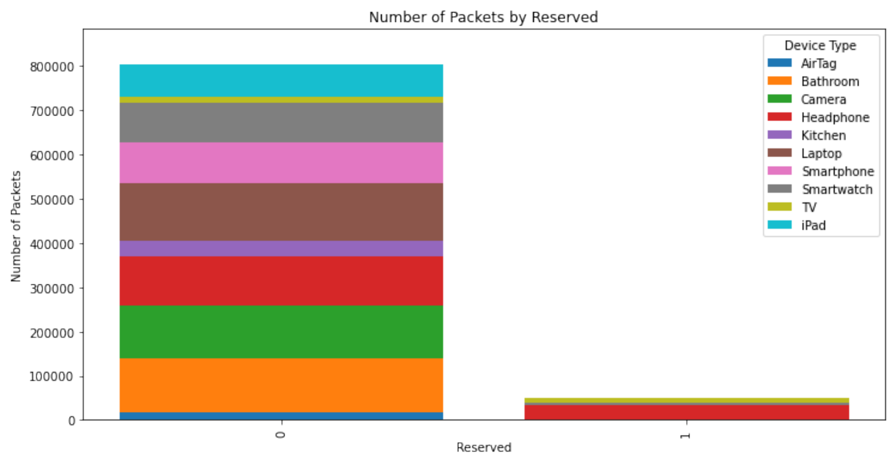


Figure E.1: Number of Packets by *Reserved* Feature

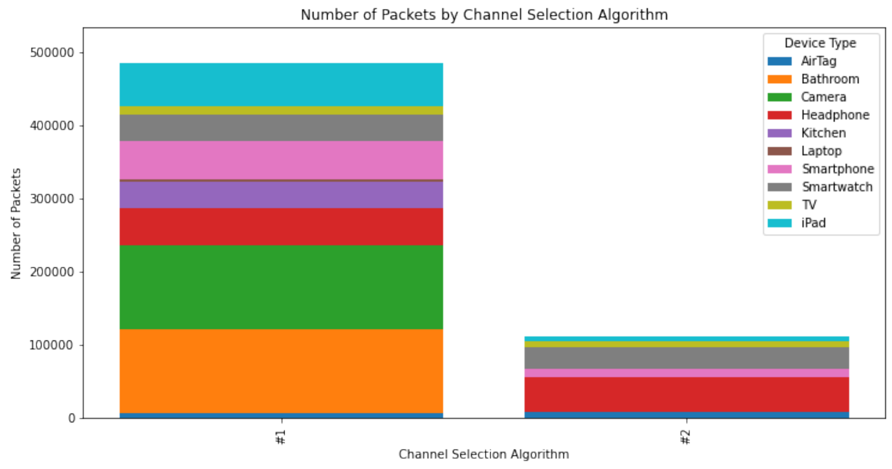


Figure E.2: Number of Packets by *Channel Selection Algorithm* Feature

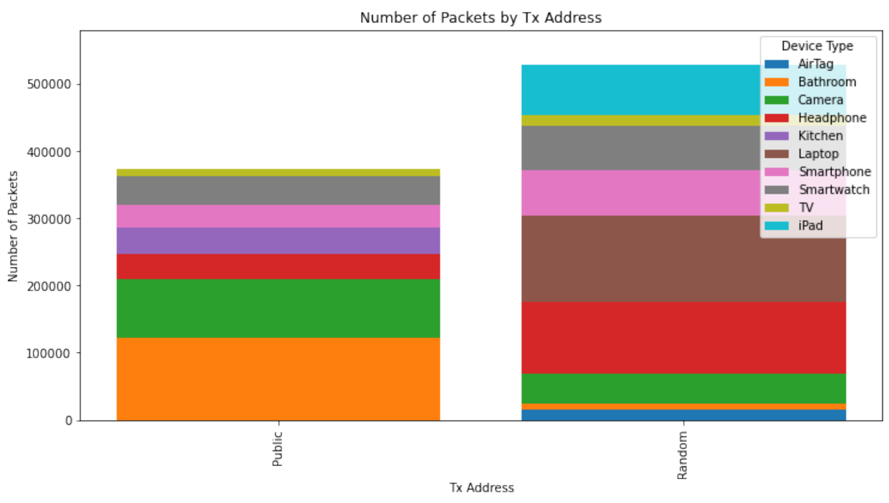


Figure E.3: Number of Packets by *Tx Address* Feature

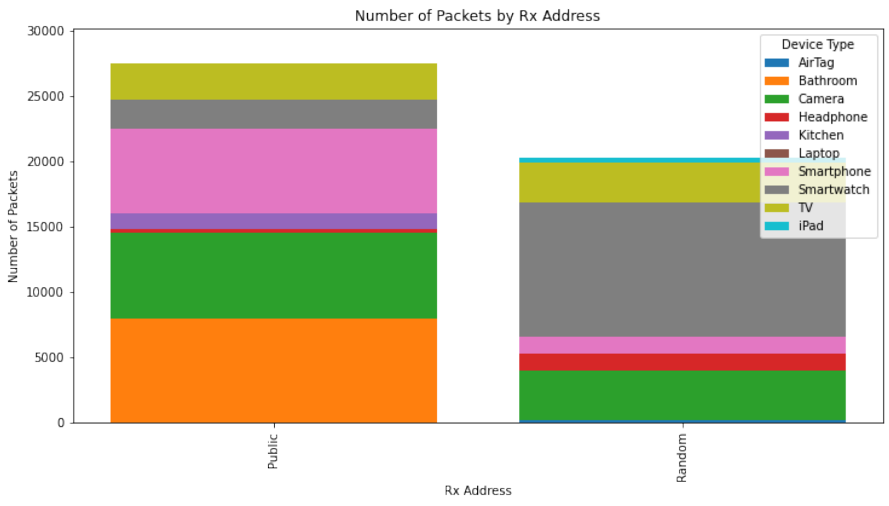


Figure E.4: Number of Packets by *Rx Address* Feature

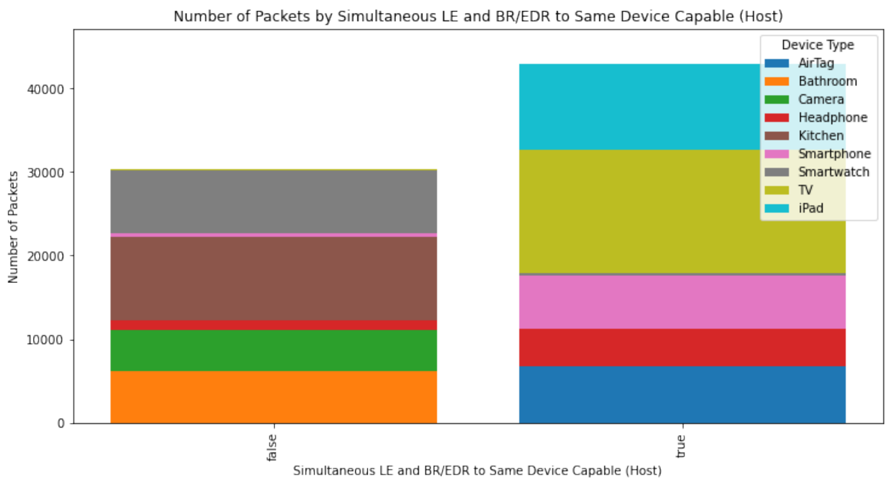


Figure E.5: Number of Packets by *Simultaneous LE and BR EDR to Same Device Capable (Host)* Feature

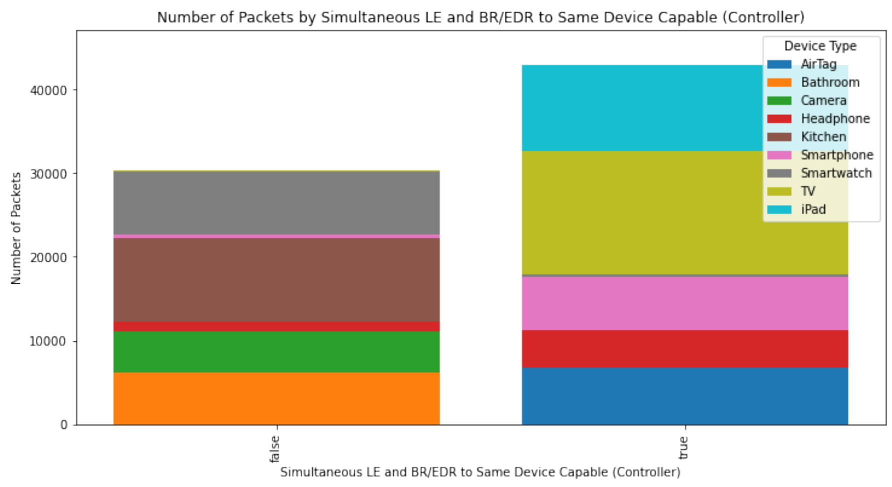


Figure E.6: Number of Packets by *Number of Packets by Simultaneous LE and BR EDR to Same Device Capable (Controller)* Feature

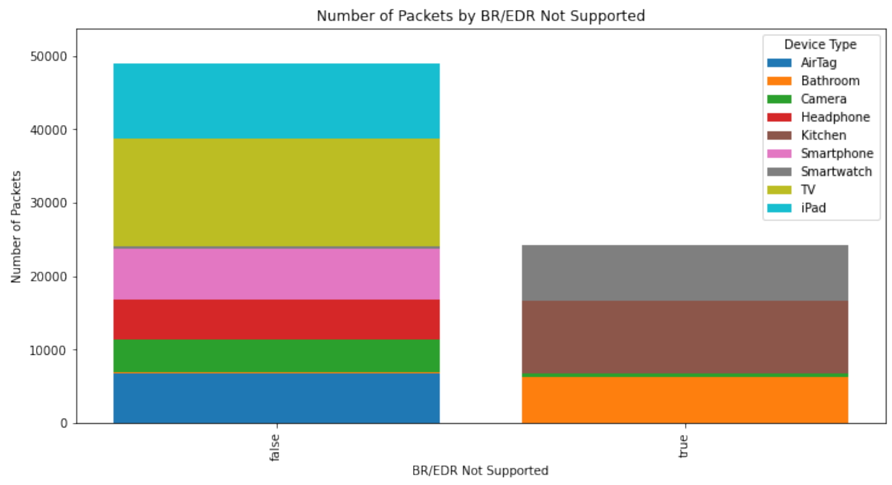


Figure E.7: Number of Packets by *Number of Packets by BR EDR Not Supported* Feature

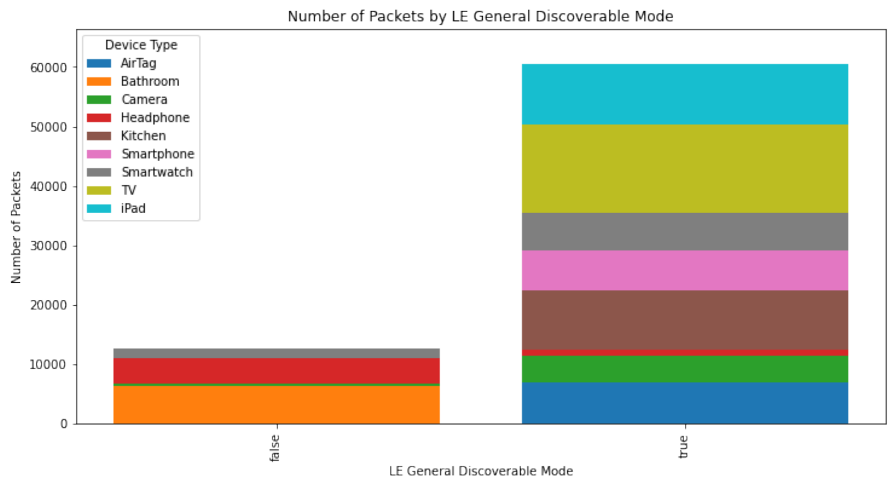


Figure E.8: Number of Packets by *LE General Discoverable Mode* Feature

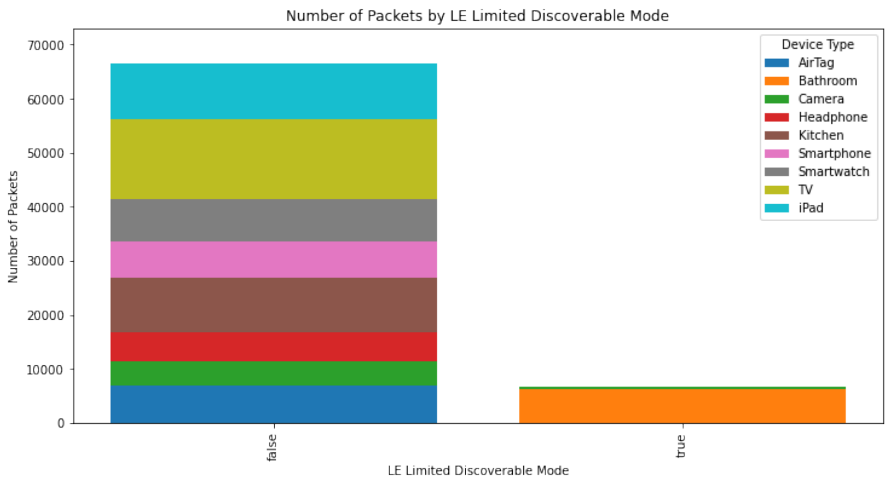


Figure E.9: Number of Packets by *LE Limited Discoverable Mode* Feature

Appendix F

Results of Derived Feature Extraction

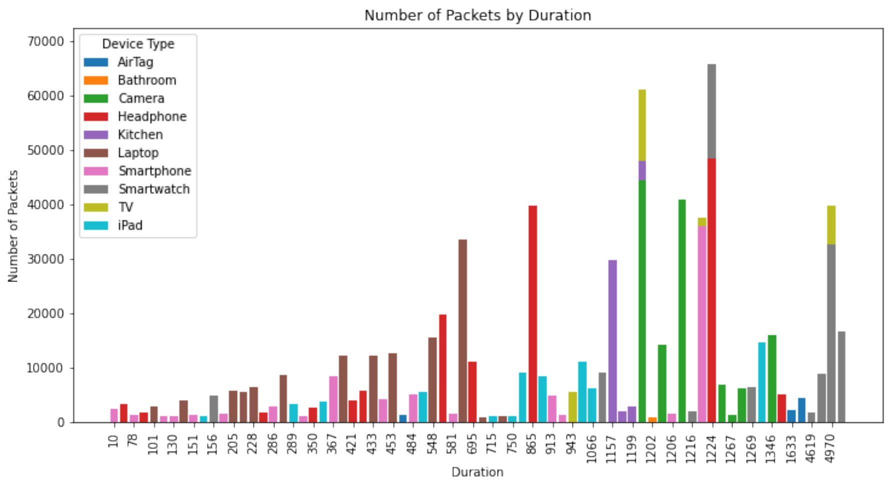


Figure F.1: Number of Packets by *Duration* Feature

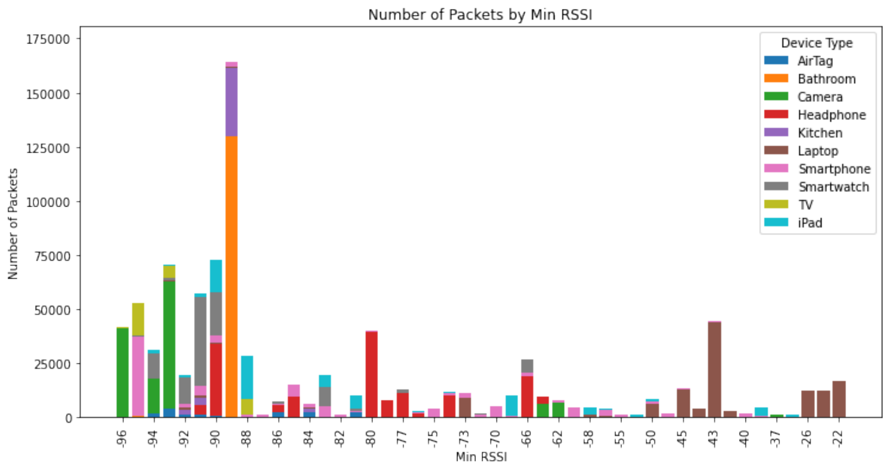


Figure F.2: Number of Packets by *Minimum RSSI* Feature

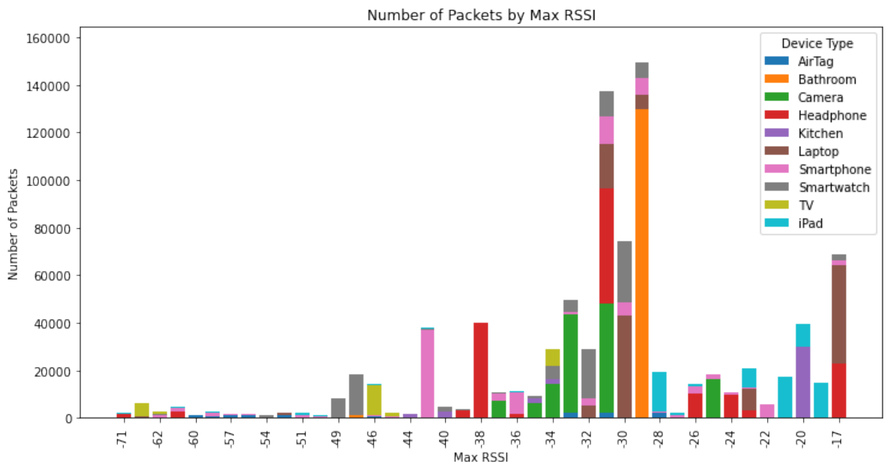


Figure F.3: Number of Packets by *Maximum RSSI* Feature

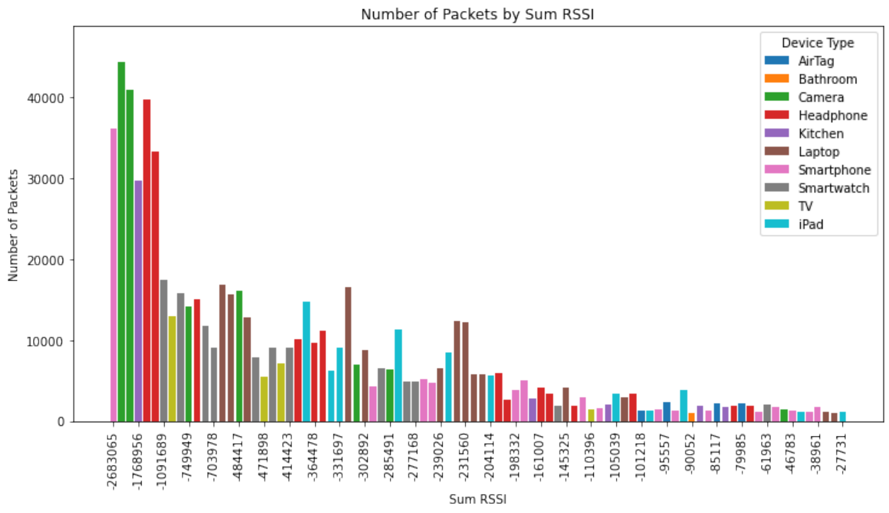


Figure F.4: Number of Packets by *Sum RSSI* Feature

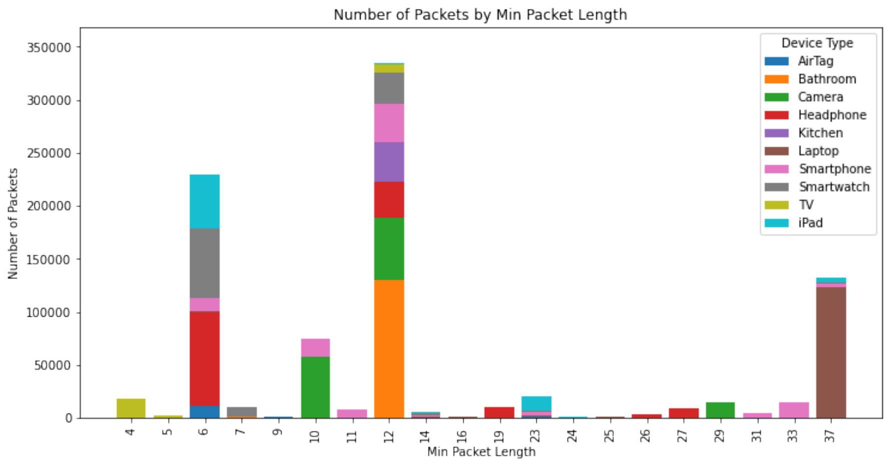


Figure F.5: Number of Packets by *Minimum Packet Length* Feature

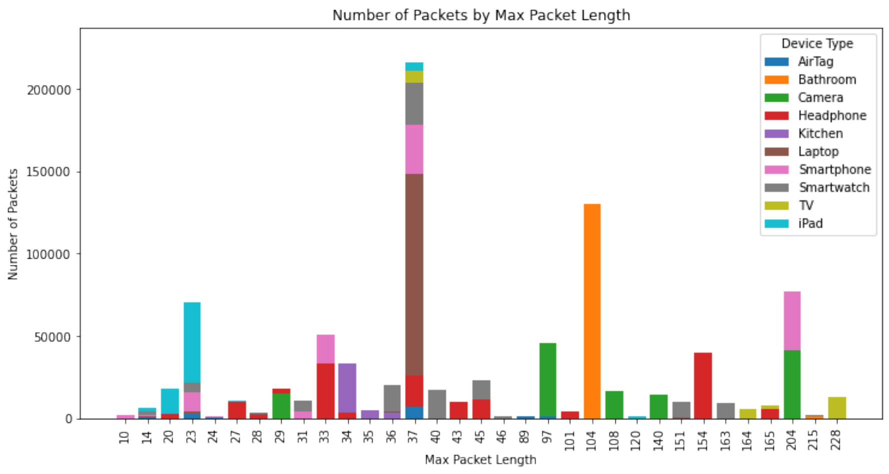


Figure F.6: Number of Packets by *Maximum Packet Length* Feature

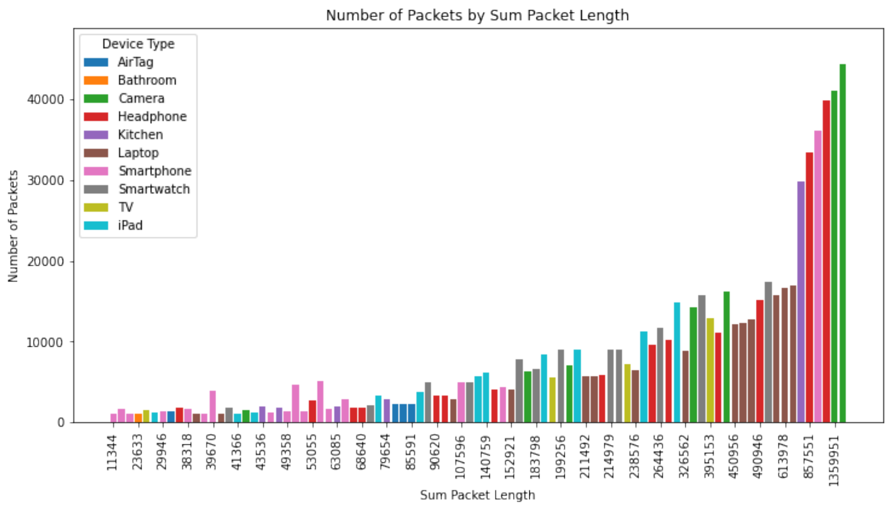


Figure F.7: Number of Packets by *Sum Packet Length* Feature

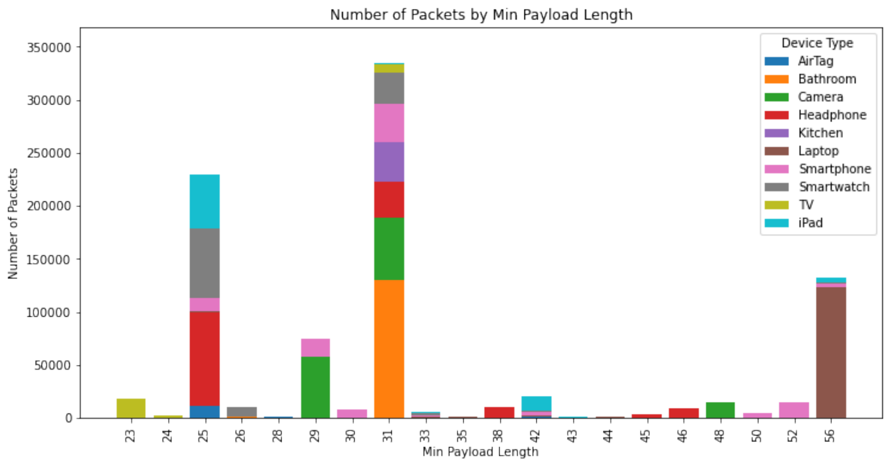


Figure F.8: Number of Packets by *Minimum Payload Length* Feature

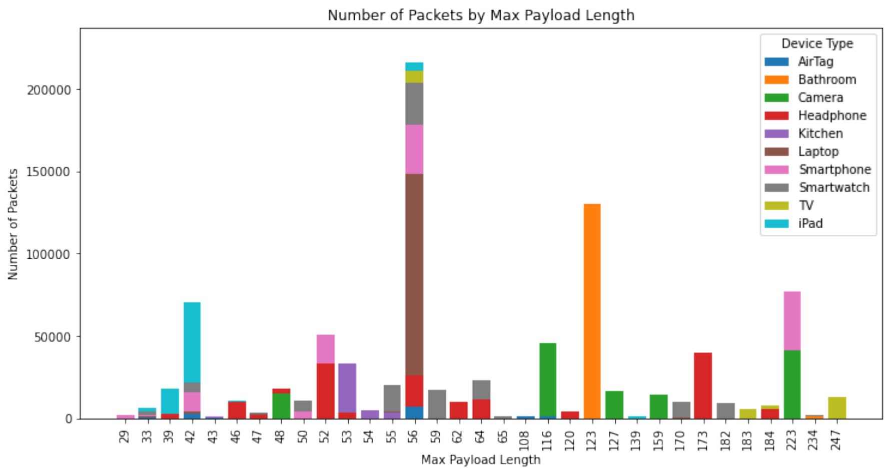


Figure F.9: Number of Packets by *Maximum Payload Length* Feature

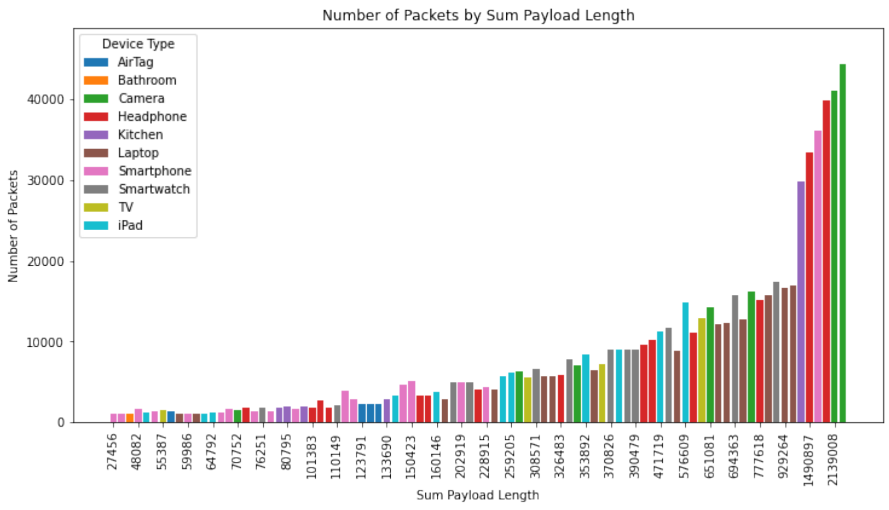


Figure F.10: Number of Packets by *Sum Payload Length* Feature

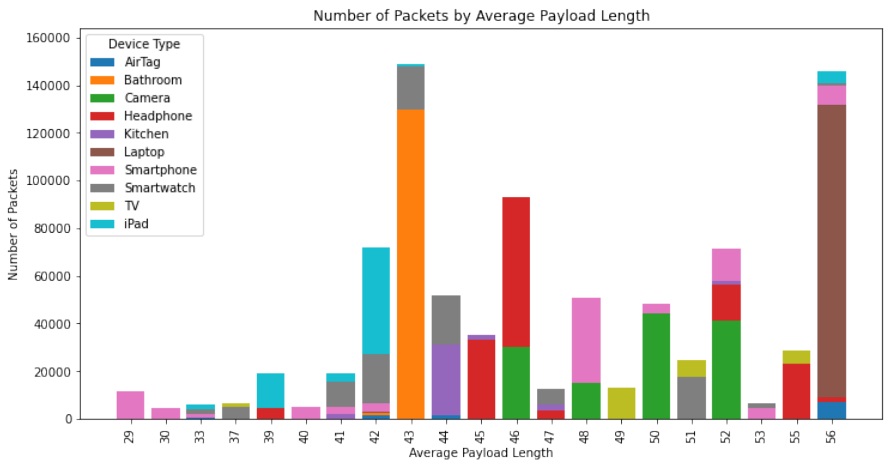


Figure F.11: Number of Packets by *Average Packet Length* Feature

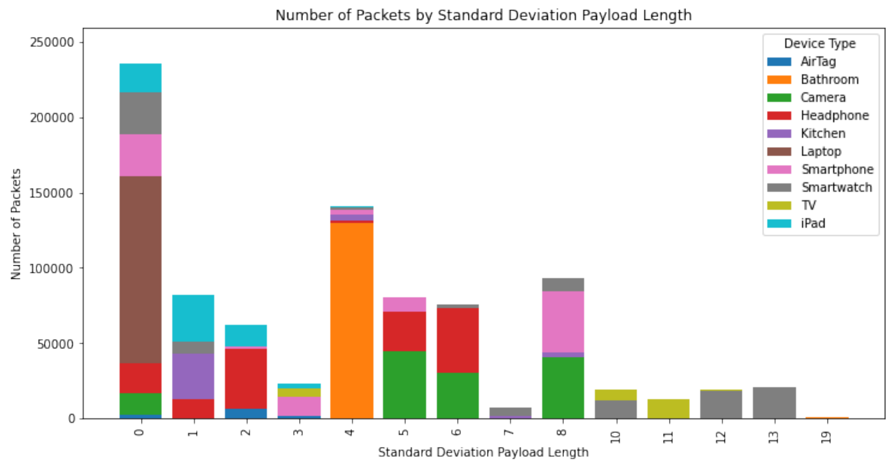


Figure F.12: Number of Packets by *Standard Deviation Payload Length* Feature

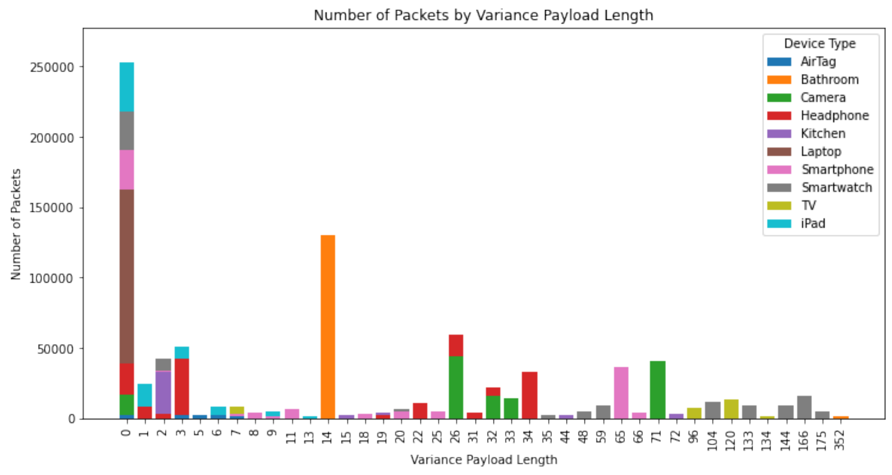


Figure F.13: Number of Packets by *Variance Payload Length* Feature

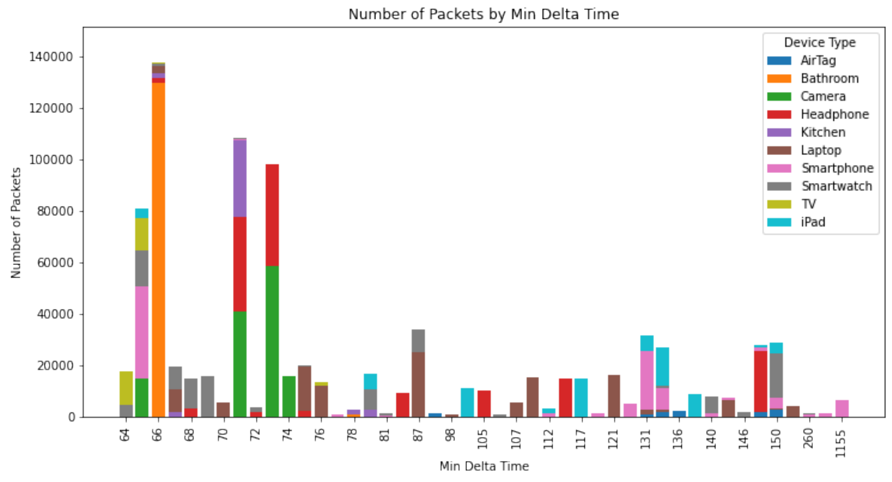


Figure F.14: Number of Packets by *Minimum Delta Time* Feature

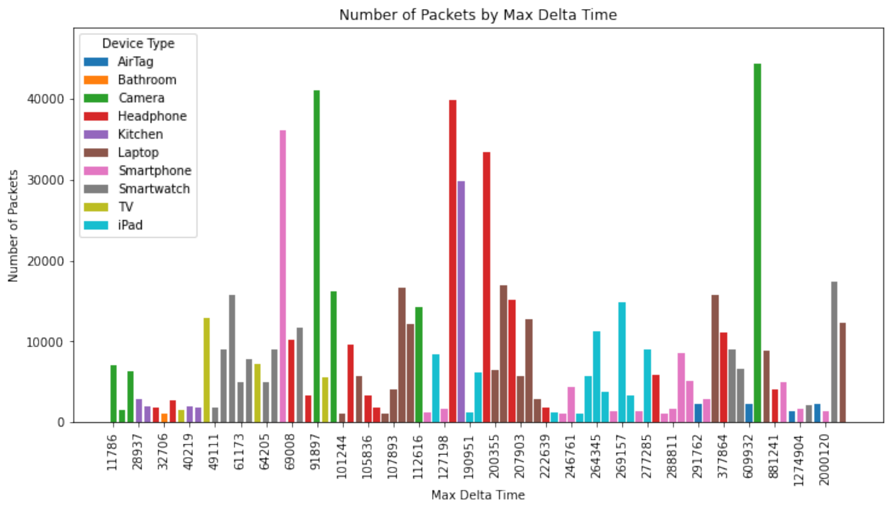


Figure F.15: Number of Packets by *Maximum Delta Time* Feature

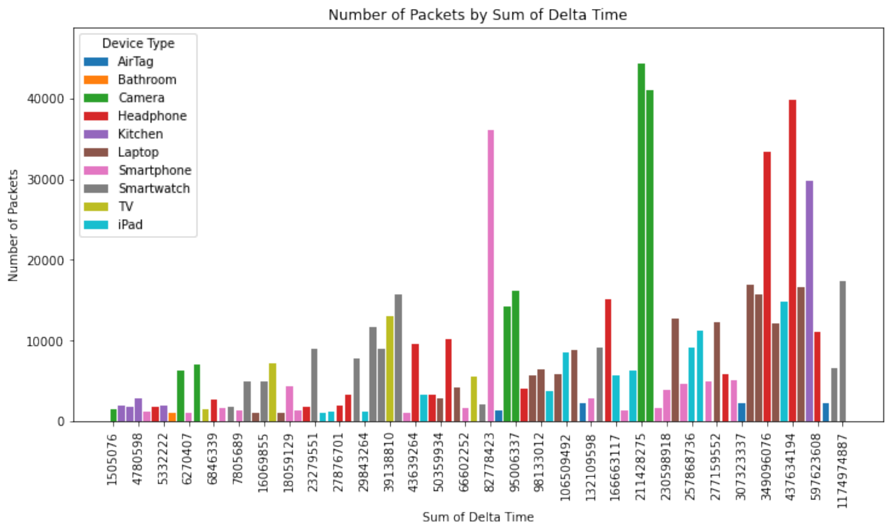


Figure F.16: Number of Packets by *Sum Delta Time* Feature

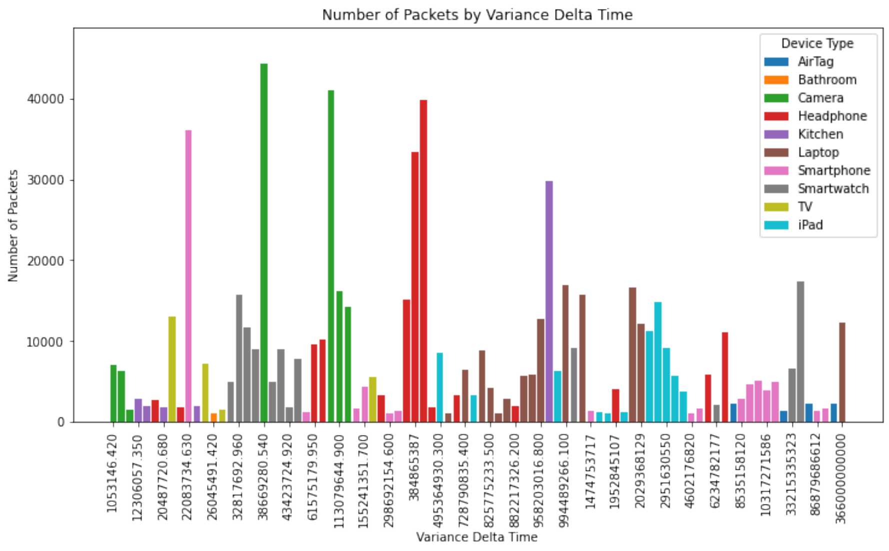


Figure F.17: Number of Packets by *Variance Delta Time* Feature

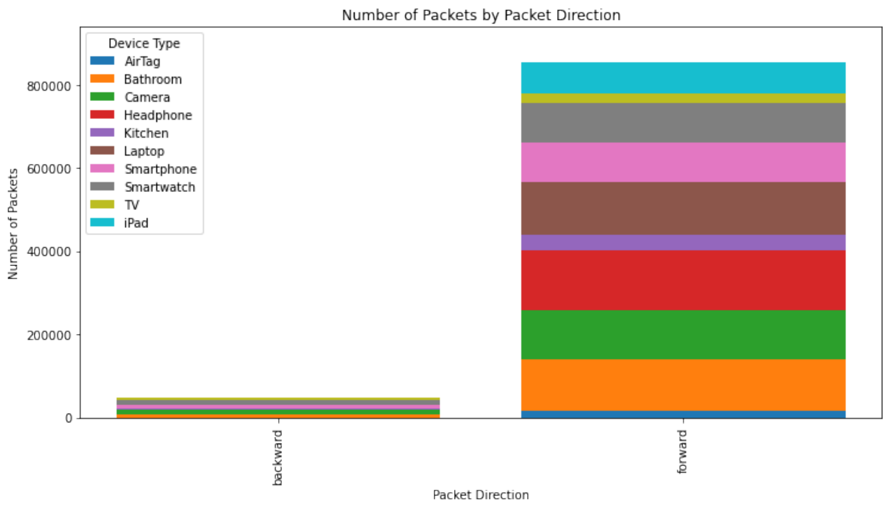


Figure F.18: Number of Packets by *Packet Direction* Feature

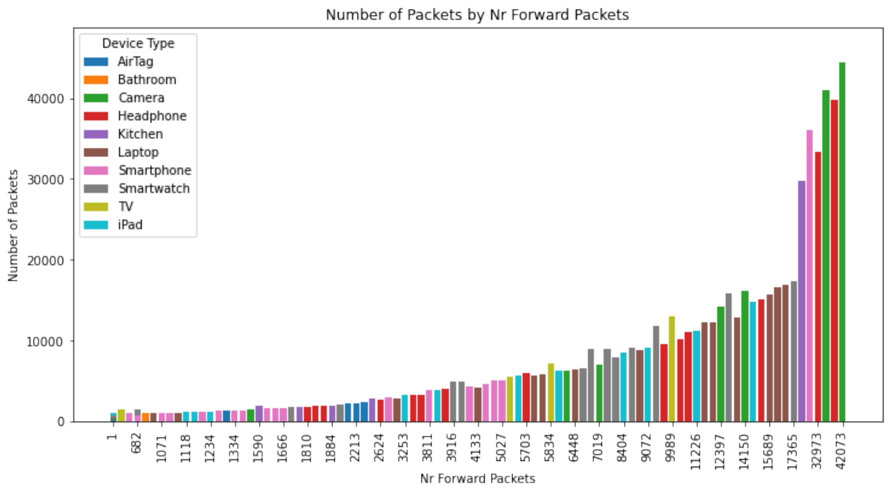


Figure F.19: Number of Packets by *Number of Forward Packets* Feature

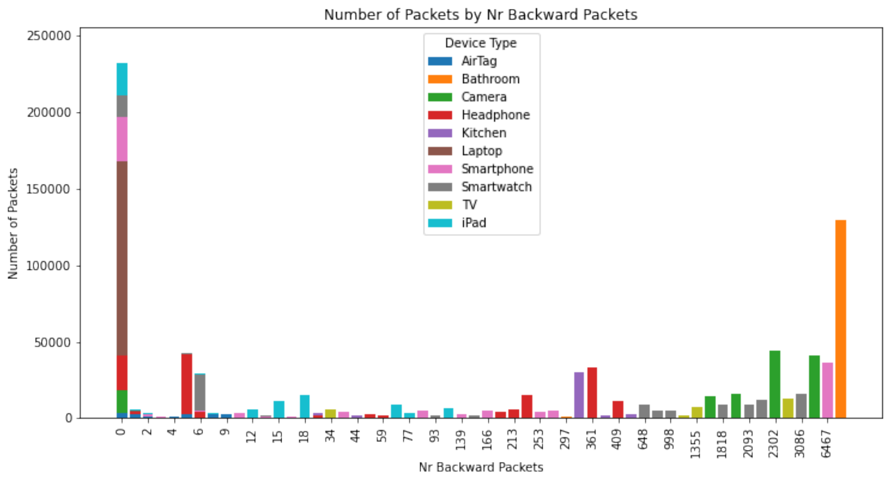


Figure F.20: Number of Packets by *Number of Backward Packets* Feature

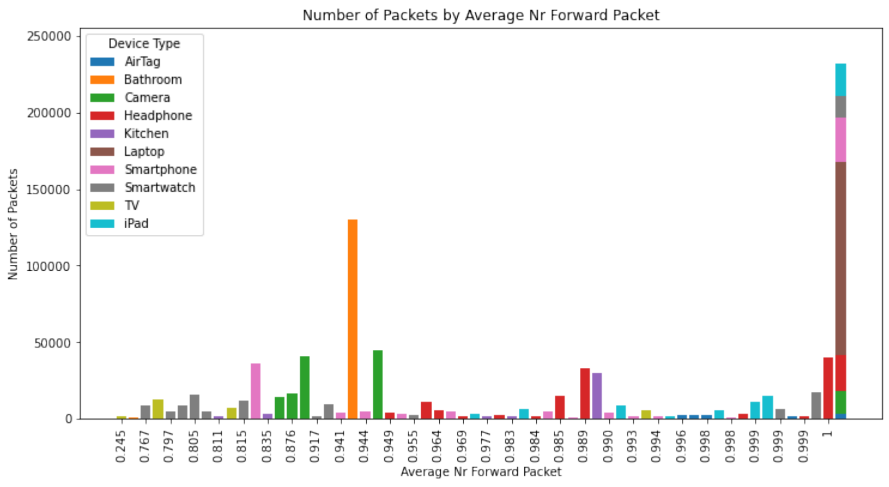


Figure F.21: Number of Packets by *Average Number of Forward Packet* Feature

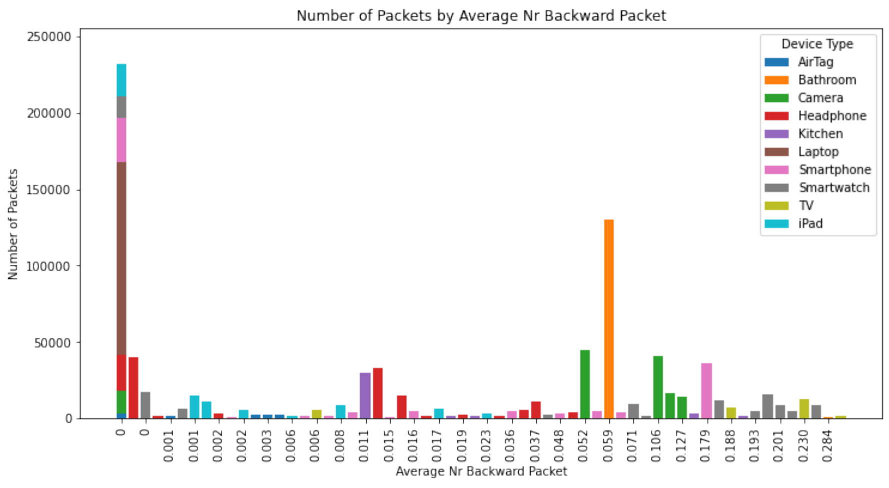


Figure F.22: Number of Packets by *Average Number of Backward Packet* Feature

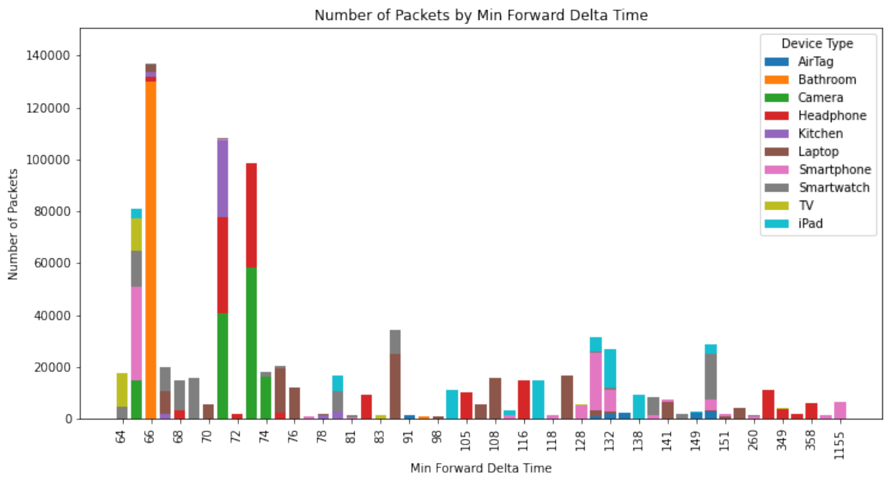


Figure F.23: Number of Packets by *Minimum Delta Time* Feature of Forward Packets

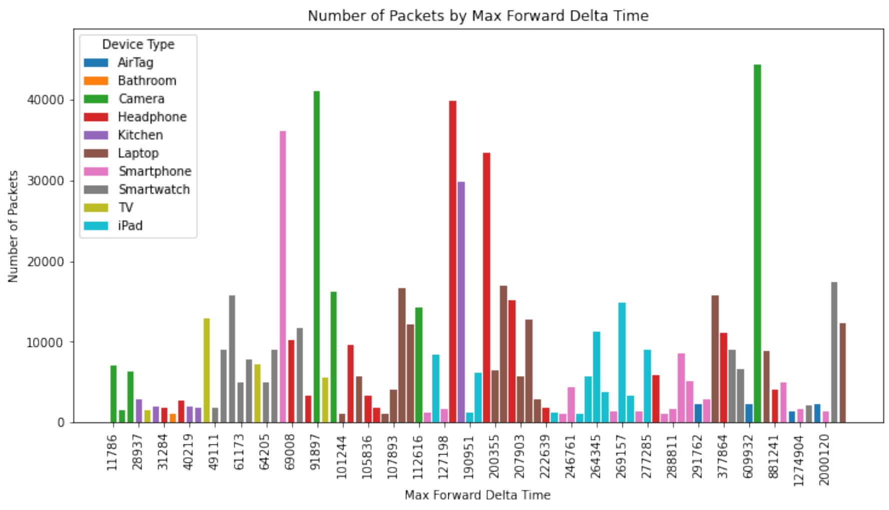


Figure F.24: Number of Packets by *Maximum Delta Time* Feature of Forward Packets

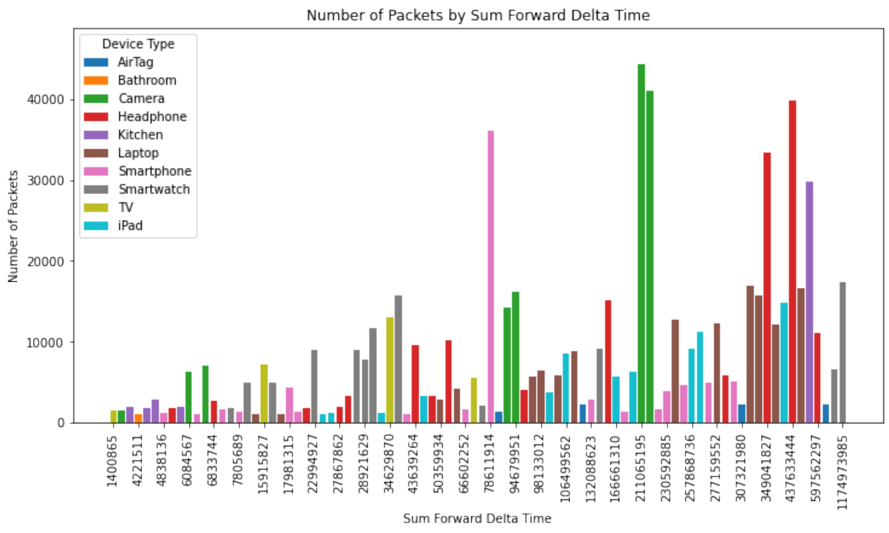


Figure F.25: Number of Packets by *Sum Delta Time* Feature of Forward Packets

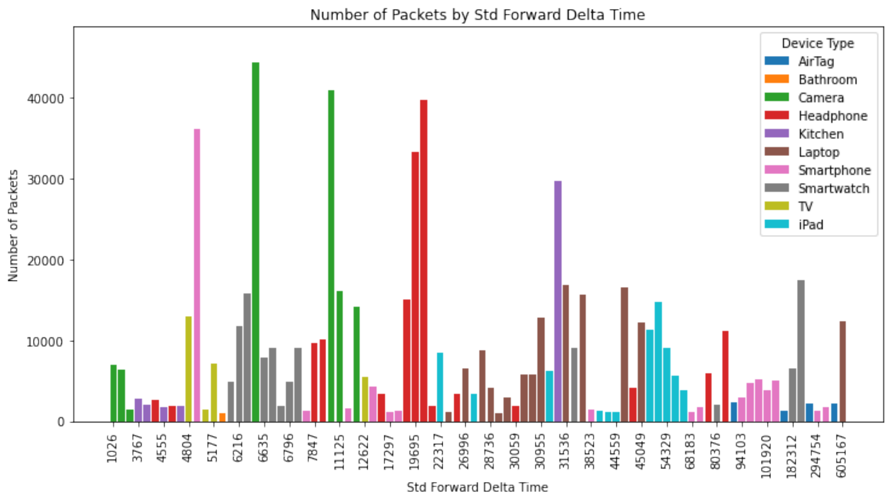


Figure F.27: Number of Packets by *Standard Deviation Delta Time* Feature of Forward Packets

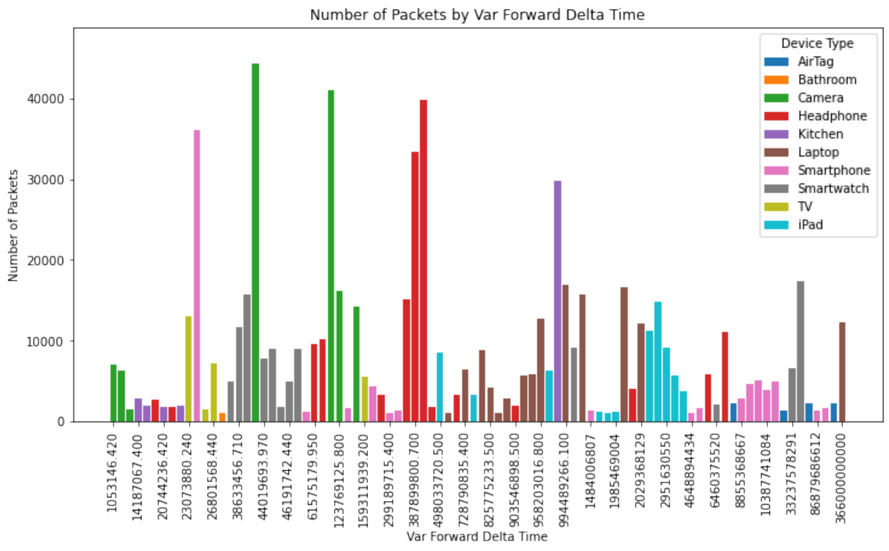


Figure F.28: Number of Packets by *Variance Delta Time* Feature of Forward Packets

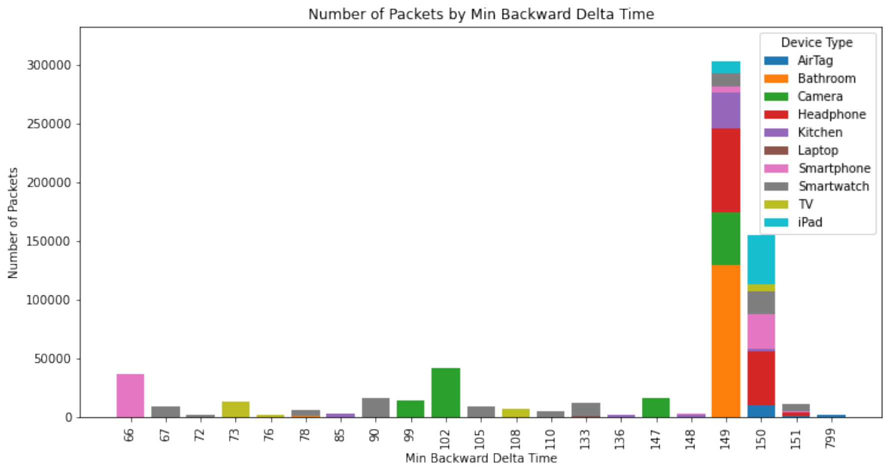


Figure F.29: Number of Packets by *Minimum Delta Time* Feature of Backward Packtes

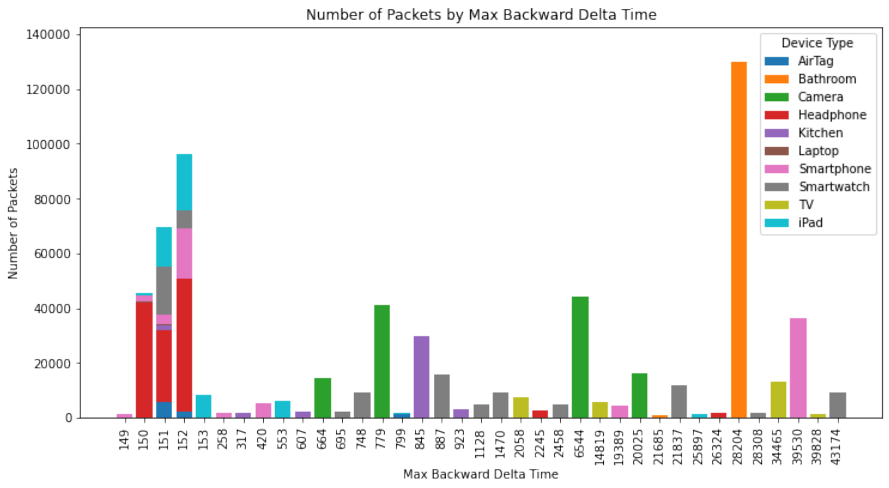


Figure F.30: Number of Packets by *Maximum Delta Time* Feature of Backward Packtes

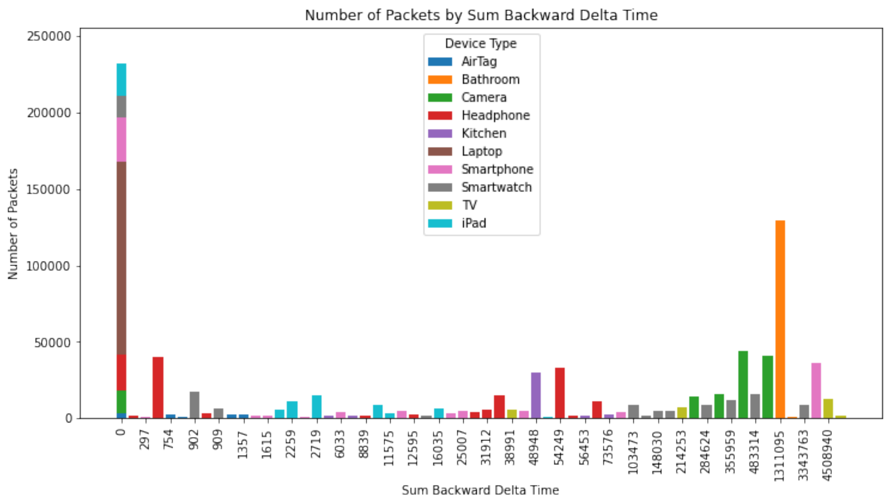


Figure F.31: Number of Packets by *Sum Delta Time* Feature of Backward Packtes

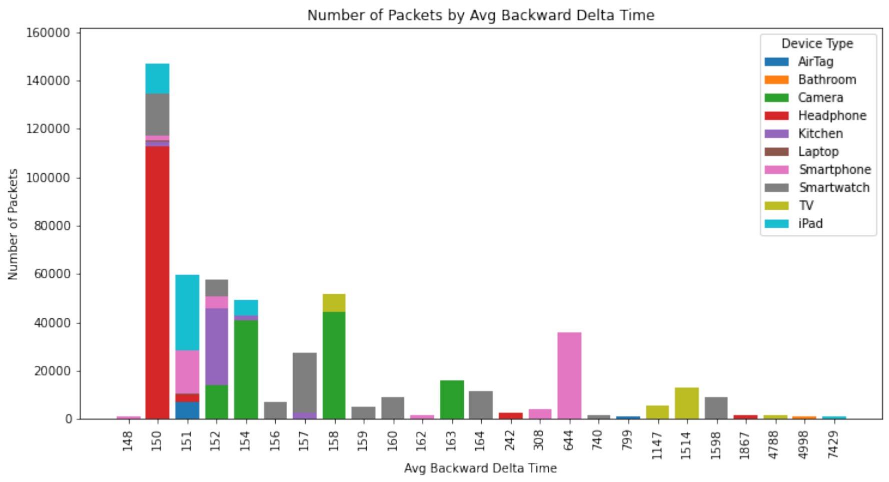


Figure F.32: Number of Packets by *Average Delta Time* Feature of Backward Packtes

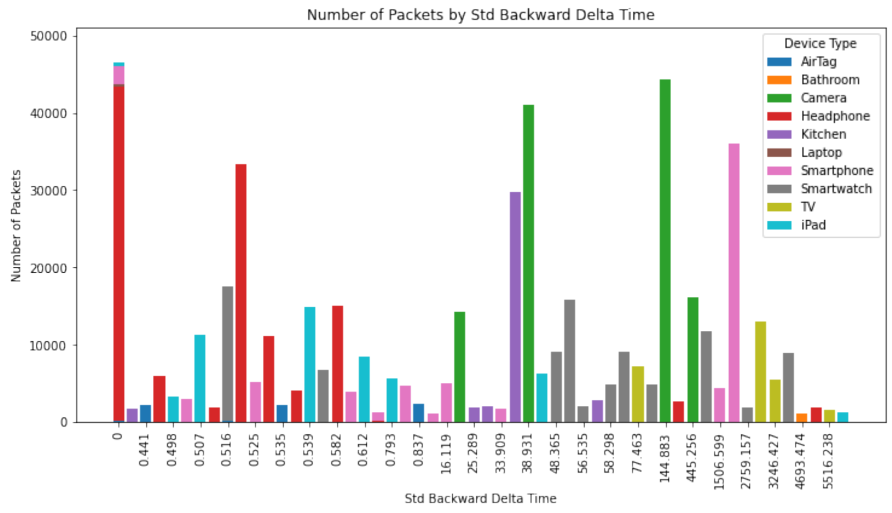


Figure F.33: Number of Packets by *Standard Deviation Delta Time* Feature of Backward Packtes

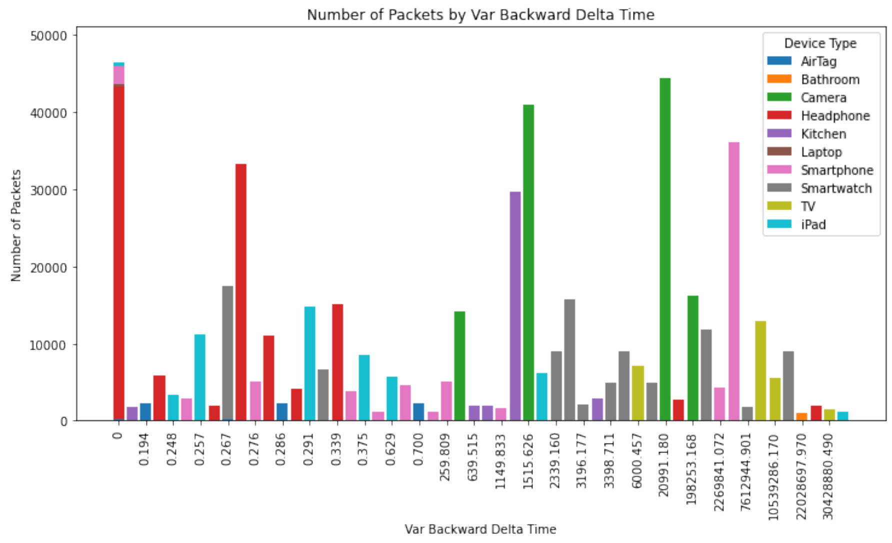


Figure F.34: Number of Packets by *Variance Delta Time* Feature of Backward Packtes