A MACHINE LEARNING APPROACH TO STATION IDENTIFICATION IN A Wi-Fi NETWORK

by

Burak Önal

B.S. Electrical & Electronics Engineering, Boğaziçi University, 2013

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical & Electronics Engineering
Boğaziçi University

2025

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Ali Emre Pusane, for his unwavering support, patience, and confidence in me. His invaluable guidance, encouragement, and mentorship have been instrumental throughout this journey.

I am profoundly grateful to my family, especially my parents, whose unwavering belief in me and continuous motivation have been a driving force in completing this degree. Their encouragement and insistence have been invaluable.

I would also like to extend my heartfelt thanks to my dear friends. Seçkin, for his countless motivational speeches that kept me focused and determined; İlhan, for the invaluable discussions that enriched my understanding and perspective, and my wife, Hazal, whose limitless support, patience, and belief in me made this journey possible. Without their presence and encouragement, this work would not have been completed.

ABSTRACT

A MACHINE LEARNING APPROACH TO STATION IDENTIFICATION IN A Wi-Fi NETWORK

Wi-Fi technology has become a vital component of modern life, driving seamless communication and effortless access to information. With its continuous evolution, the number of connected devices has grown exponentially. This rapid expansion has introduced challenges, one of which is station fingerprinting in Wi-Fi networks. In this work, we tackeled this problem, station identification in a Wi-Fi network, using a multi-class supervised classification approach. The dataset is obtained from a real-world scenario from a private ISP, encompasses observations from physical and network layers, traffic patterns and protocol features. Our study integrates these feature groups to develop robust classifiers for identifying ten distinct station types.

In this work, we explored various classifiers across 2.4GHz and 5GHz interfaces. A critical focus is on evaluating the performance of these models independently and in combination through stacking methods. Band stacking merges predictions from 2.4GHz and 5GHz of the same classifier type, while an advanced stacking approach integrates outputs of different classifier types.

Experimental results compares individual classifier performances against stacking methods. The findings underscore the potential of combining heterogeneous features and leveraging stacked classifiers for station identification, offering a scalable and efficient solution for fingerprinting in a real-world Wi-Fi network set ups.

ÖZET

BİR Wİ-Fİ AĞINDAKİ İSTASYONLARIN TESPİT EDİLMESİNDE MAKİNE ÖĞRENMESİ YAKLAŞIMI

Wi-Fi teknolojisi, kesintisiz iletişim ve bilgiye kolay erişim sağlayarak modern hayatın vazgeçilmez bir parçası haline gelmiştir. Sürekli gelişimiyle birlikte, bağlı cihazların sayısı da katlanarak artmıştır. Bu hızlı genişleme, Wi-Fi ağlarında istasyon parmak izi çıkarma gibi çeşitli zorlukları beraberinde getirmiştir. Bu çalışmada, Wi-Fi ağlarında istasyon tanımlama sorununu, çok sınıflı güdümlü öğrenme yaklaşımı kullanarak ele aldık. Veri seti, özel bir internet servis sağlayıcısından elde edilmiş gerçek dünya verisidir ve fiziksel ve ağ katmanlarından, trafik desenlerinden ve protokol özelliklerini içeren gözlemlerden oluşmaktadır.

Çalışmamızda 2.4GHz ve 5GHz arayüzlerinde çeşitli sınıflandırıcıları inceledik. Temel odak noktası, bu modellerin performanslarını bağımsız ve yığma yöntemleriyle birlikte değerlendirerek incelemektir. Band yığma yöntemi, aynı sınıflandırıcı türünün 2.4GHz ve 5GHz tahminlerini birleştirirken, ileri bir yığma yaklaşımı farklı sınıflandırıcı türlerinin çıktısını birleştirmektedir.

Deneysel sonuçlar, bireysel sınıflandırıcı performanslarını yığma yöntemleriyle karşılaştırmaktadır. Bulgular, heterojen özelliklerin birleştirilmesi ve yığma sınıflandırıcıların istasyon tanımlamada kullanılmasının, gerçek Wi-Fi ağı kurulumlarında parmak izi çıkarma için ölçeklenebilir ve verimli bir çözüm sunduğunu vurgulamaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$C_{i,j}$	Count of instances where i, j are true and predicted classes
I(X; Y)	Mutual information between random variables X and Y
r_{xy}	Correlation coefficient r_{xy} between two variables X and Y
$\sigma(\cdot)$	non-linear activation function in MLP nodes
Θ_k	input data of a single tree in RF

LIST OF ACRONYMS/ABBREVIATIONS

AP	Access Point
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
FI	Feature Importance
FN	False Negative
FP	False Positive
FPR	False Positive Rate
GNB	Gaussian Naive Bayes Classifier
HGBT	Histogram-based Gradient Boosting Trees
HTTP	Hypertext Transfer Protocol
IQR	interquartile range
MLP	Multi-layer Perceptron
NB	Naive Bayes
PI	Permutation Importance
PR	Precision-Recall
QoS	Quality of Service
RF	Random Forest Classifier
ROC	Receiver Operator Characteristic
RSSI	Received Signal Strength Indicator
SNR	Signal to Noise Ratio
SSDP	Simple Service Discovery Protocol
TCP	Transmission Control Protocol
TP	True Positive
TPR	True Positive Rate
UDP	User Datagram Protocol
UPnP	Universal Plug and Play
Wi-Fi	Wireless Fidelity

1. INTRODUCTION

With the rapid growth of Wi-Fi technologies and their integration with mobile and IoT devices, the number and diversity of connected devices have increased significantly [1]. From state-of-the-art cloud computing platforms to IoT devices, through web-based or embedded systems, the modern connected society relies heavily on a secure and high QoS Wi-Fi networks [2]. According to most recent Cisco's Annual Internet Report, the number of devices connected to IP networks is projected to reach 29.3 billion by 2023, up from 18.4 billion in 2018 [3]. As the number and capability of connected devices increase, a variety of applications in healthcare, transportation and energy sectors are getting more common. [4–6].

However, this trend brings also the concerns for security and privacy [7]. The heterogenity of devices and the volume of data they generate create a big challenge for operating a reliable, secure Wi-Fi network. A rogue IoT device in a wireless network can be very dangerous and sensitive information can leake to unauthorized parties [8]. To help addressing these challenges, station fingerprinting has emerged as an important research area. Station fingerprinting focuses on identifying devices connected to Wi-Fi networks. Regardless of the application, granularity of the detection or underlying data layer(s), the approaches are varying; but in the end, they are all relying on utilizing device specific features, the fingerprints. The motivation behind these efforts are sourcing from two primary goals [9]:

- Device Misbehavior Detection: To detect cyberattacks, malfunctioning or unauthorized devices, and other anomalies within the network.
- Device Identification: To classify devices with varying levels of granularity and fully exploit their capabilities.

Initially, station identification efforts concentrated on security applications, such as detecting unauthorized accesses or anomalies in device behavior or network metrics

[10–14]. In time, while network security is still a top priority, the scope of fingerprinting has expanded to include optimizing device performance and enhancing user experience, leading to efforts focusing on station classification [15, 16]. This shift is driven by the increasing demand for understanding device capabilities and use cases to ensure QoS and efficient resource allocation.

The majority of researches conduced in this area treated station fingerprinting as a supervised problem. Before visiting them however, it should be noted that there are unsupervised approaches as well [17–20]. The key idea in these researches are to map devices into latent representative spaces, where different devices are linked with clusters or probability distributions. The reseraches used k-means [21] and DBSCAN [22] algorithms. Prior to the advent of AI, allowing machine learning and deep learning algorithms for a wide range of applications, statistical approaches were popular among researches [23–25]. These works revolut around building a distance metric, like Euclidean distance or IQR, to determine anomalies in the network.

Approaches relying on supervised learning can be categorized according input data layers [26–28]. Broadly speaking we can categorize these approaches into following schema:

- Physical layer features
  - Channel/location based features obtained from RSSI, SNR, CSI as well as frequency responses [29–31]
  - Hardware imperfections based features exploiting device specific signatures visible in received signal form, amplitude or phase as well I/Q imbalance. [11, 32, 33]
- MAC layer features [34, 35]
- Network activity and traffic patterns based features, [36–40]
- Upper layer features including TCP, UDP, DNS, DHCP and HTTP [16, 41, 42]

Among these feature sets, models built upon physical layer features perform better for both abnormal activity and rogue device detection as well classifying tasks. However, physical layer data is hard to collect. Most of the commercial radios interface is limited and incapable of capturing and sharing this data. Therefore, researches utilizing physical layer data require either a special harware equipment to capture those features, or they conducted their research over a simulated data set.

On the other hand, MAC and upper layer features are easier to capture. However, there are limitations in those approaches. Features derived from different protocols could be shared within station classes. Moreover, due to privacy concerns, they can be cloaked and become obsolte. One such example would be MAC randomization [43].

One advantage of utilizing network activity and traffic patterns is that it leads to model behavioural variations in devices [9]. While they are good at device type detection, they fall short in identifying individual stations as device types do not necessarily correlate with identities.

In this work, as we will see in Chapter 2, we are limited to a data set obtained from a real-world scenario. The data set includes observations from physical-layer (although on a very high level), network and traffic patterns, and protocol information. We will be tackling a supervised multi-class classification problem. Except from MAC layer features, we will be utilizing all the feature groups mentioned above and measure their effects on classification performance. Also, we will introduce a method for stacking classifiers [44]. The flow of this work can be summarized as:

- Merge physical-layer, network and traffic patterns and protocol features to construct a comprehensive dataset.
- Derive new features and aggregations.
- Train classifiers on subsets of features, compare their performances.
- Stacking meta classifiers into a final classifier to achieve optimal performance.

To the best of our knowledge, there is no study that evaluates physical-layer and upper layer features and device behaviour via traffic activity on the same model.

The structure of this work is organized as follows: Chapter 2 introduces the dataset and provides background information. Chapter 3 introduces our pipeline for station identification. Section 3.1 explains feature engineering and selection, highlighting the methods used to enhance the dataset. Section 3.2 presents single-stage classifiers, applied with various datasets and algorithms, and evaluates their performance using confusion matrices. Section 3.3 revisits these classifiers and combines them into a stacked model. The performance of these stacked classifiers is similarly analyzed through confusion matrices. Chapter 4 provides a comprehensive evaluation of both single-stage and stacked classifiers using f1-score, along with visual tools such as ROC and PR curves. Definitions of these metrics, details on feature engineering methods, and descriptions of other performance indicators are included in the respective sections. Finally, in Chapter 5, we conclude our work and discuss potential areas for improvement and directions for future research in Section 5.1.

2. DATA SET

The dataset analyzed in this study originates from a private ISP and encompasses anonymized data from approximately 1,500 households monitored over a 60-day period. Each household, referred to as a "line," represents a single Wi-Fi network. The dataset includes a diverse range of connected devices, with the following three key characteristics shared by all lines:

- Each line represents a single-AP home, meaning there is only one AP per network, with no extenders or secondary APs deployed at the site.
- All lines feature the same AP hardware and software configurations, ensuring uniformity across the dataset.
- Each AP operates on dual-band radios, providing connectivity over both 2.4 GHz and 5 GHz frequencies.

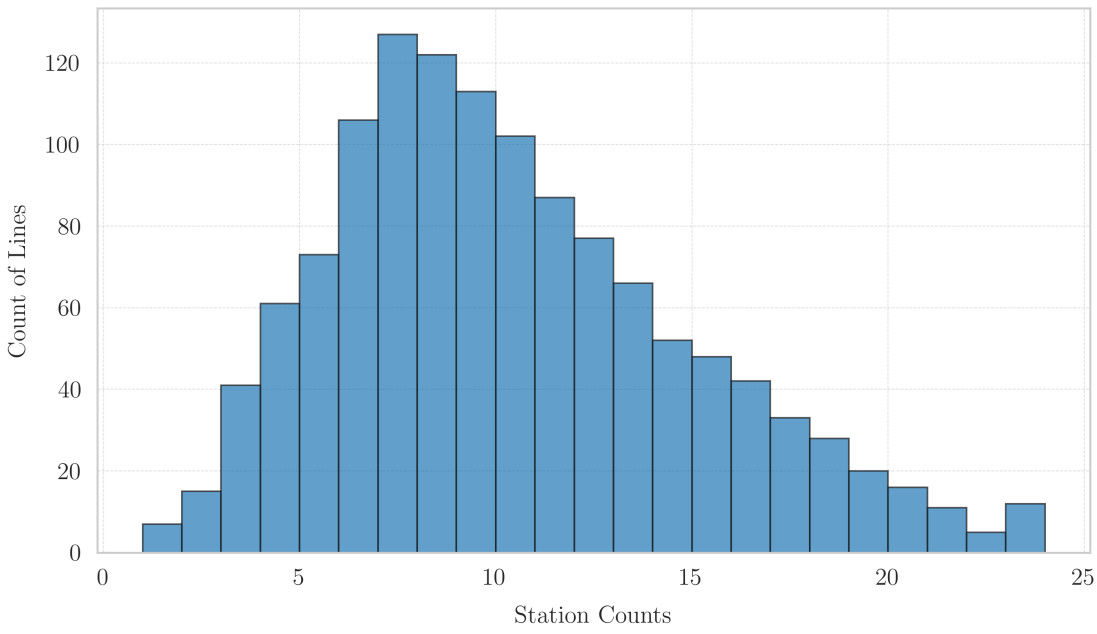

Figure 2.1: Distribution of stations per line.

The histogram in Figure 2.1 illustrates the distribution of station counts per line. The station types in the data set are:

- computer
- gaming console
- printer
- samsung phone
- smart home iot
- smart phone
- streaming device
- tablet
- ipad
- iphone

Certain categories, such as iphone, ipad and samsung phone, represent well-defined sets of devices with consistent hardware specifications and potentially uniform usage patterns. In contrast, broader categories like smart home iot, smart phone and streaming device encompass a diverse range of functionalities. For example, the smart home iot category includes devices such as smart assistants, security cameras, lighting systems, thermostats and other iot-enabled equipment. Similarly, streaming device category covers televisions and platforms like apple tv or chromecast. The smart phone class comprises all mobile phones capable of connecting to an AP, except those manufactured by dominant players like Apple and Samsung, which are classified separately as iphone and samsung phone.

Figure 2.2 illustrates the distribution of these station types across the dataset. 4 station types—iphone, samsung phone, computer and streaming device collectively—account for more than 70% of the population. This distribution poses a significant challenge for classification models due to the data imbalance, particularly for the remaining six station types.

The dataset is structured into three primary sources of information, each capturing distinct aspects of station behavior and network characteristics.

- `protocol data`: Captures network identifiers and configurations.
- `Wi-Fi operational data`: Records connectivity and signal metrics, such as RSSI, SNR and transmitted/received traffic.
- `throughput data`: Covers measurements for throughput estimation along with other metrics.

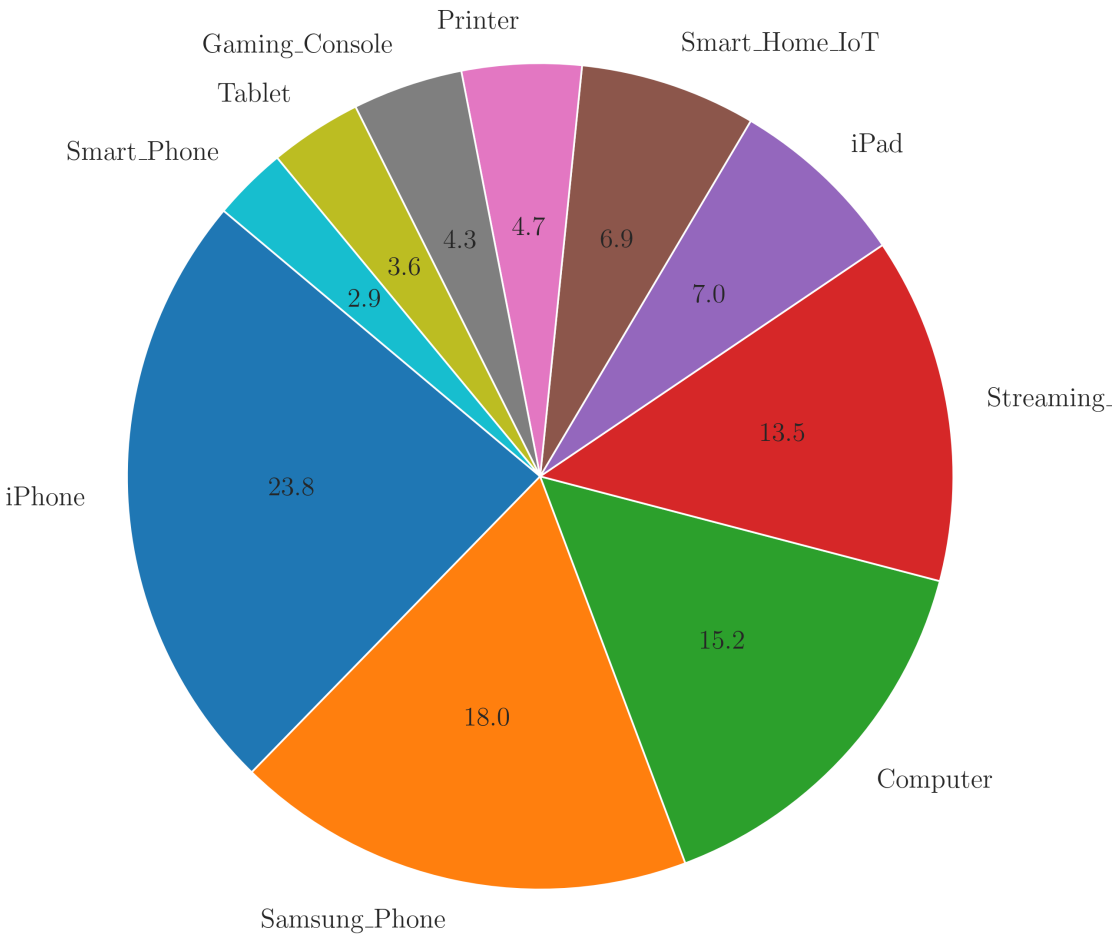

Figure 2.2: Station type distribution.

2.1. Protocol Data

Protocol data is collected daily and includes two key identifiers; `DHCP options` and `SSDP user-agent`. An example of collected data is given Table in 2.1.

The `SSDP user-agent` is typically used in UPnP environments, enabling devices to discover and communicate with each other seamlessly on a local network. The `SSDP user-agent` provides additional context about the device's software or hardware characteristics.

`DHCP options` are configuration parameters exchanged between a DHCP client and server during the network configuration process. These options specify various settings such as IP address allocation, subnet mask, default gateway, DNS servers and other network-related information. They are particularly useful in dynamically configuring devices within a network.

Only 25% of the dataset contains valid `SSDP user-agent` information, limiting its usability. As a result, `DHCP options` becomes the primary identifier available for analysis. Figure 2.3 shows top 50 `DHCP options` patterns across station labels. Notably, the figure displays consistent patterns for certain device classes. For instance, `ios` devices like iphone or ipad share similar option strings, while other options are unique to specific classes, such as printer, smart home iot or streaming device types.

Table 2.1: Protocol data.

day	**mac**	**ssdpUserAgent**	**dhcpOptions**	**label**
YYYY-MM-DD	XX:XX:XX:XX	Microsoft Edge	1,3,6,15,31,33,43,44,46,47,119,121,249,252	Computer
YYYY-MM-DD	XX:XX:XX:XX	<NA>	1,3,6,15,26,28,51,58,59,43,114,108	Samsung_Phone
YYYY-MM-DD	XX:XX:XX:XX	Google Chrome, Microsoft Edge	1,3,6,15,31,33,43,44,46,47,119,121,249,252	Computer
YYYY-MM-DD	XX:XX:XX:XX	<NA>	1,3,6,15,26,28,51,58,59,43	Streaming_Device
YYYY-MM-DD	XX:XX:XX:XX	<NA>	1,121,3,6,15,108,114,119,252	iPhone

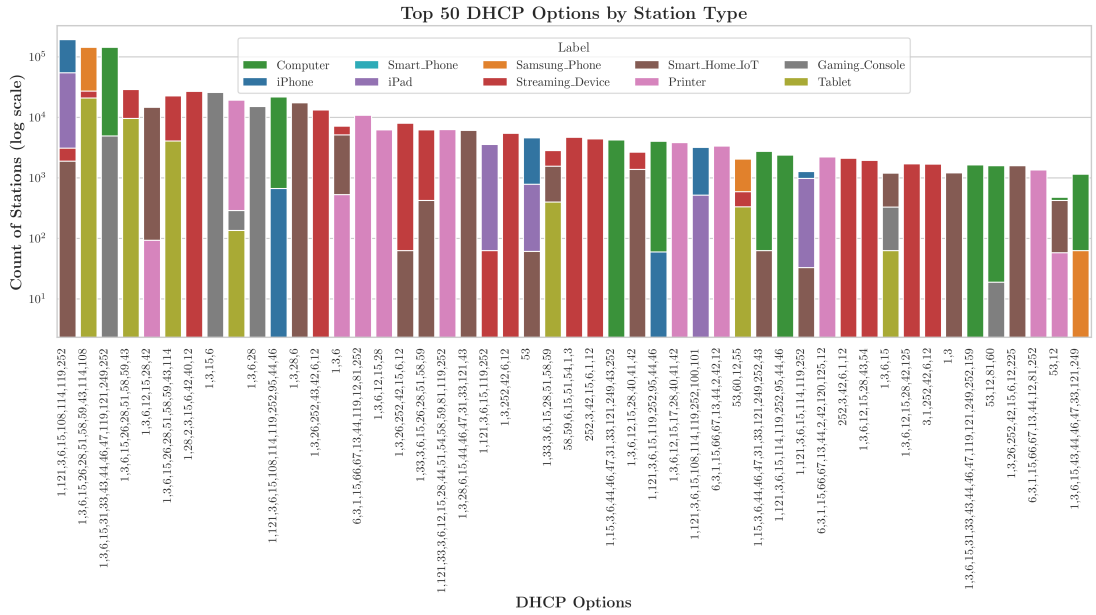

Figure 2.3: Top 50 dhcpOptions by station type.

2.2. Wi-Fi Operational Data

`Wi-Fi operational data` is collected over a 15 minute measure interval. An example of collected data is given Table in 2.2.

Table 2.2: Wi-Fi operational data.

timeStamp	mac	band	noise	inNetwork	txPackets	txFailed	txRateKbps	rssi	txRetried	txMcastPackets	txKBytes	txMcastKBytes
YYYY-MM-DDTHH:MM:SS	X:X:X:X	2G	-90	898	3144	1	70095	-62	1	14905	1396.663	1658.889
YYYY-MM-DDTHH:MM:SS	X:X:X:X	2G	-96	900	271	3	189768	-49	0	5942	69.875	879.698
YYYY-MM-DDTHH:MM:SS	X:X:X:X	2G	-97	900	70	0	53950	-65	0	11929	8.897	1253.565
YYYY-MM-DDTHH:MM:SS	X:X:X:X	5G	-87	884	4970	1	2288453	-56	0	10586	5902.086	2130.481
YYYY-MM-DDTHH:MM:SS	X:X:X:X	5G	-91	899	467	0	840185	-59	2	4803	95.381	595.043

The fields from the data set are:

- `timeStamp`: Masked timestamp of the record.
- `mac`: Masked MAC address of the station.
- `band`: The Wi-Fi band on which the station is operating.
- `noise`: Average noise level measured over the interval [dBm].
- `inNetwork`: Duration that the station remained connected during the measurement interval [sec].

- `txPackets`: Count of unicast packets successfully transmitted by the station.
- `txFailed`: Count of unicast packets that failed to transmit successfully.
- `txRateKbps`: Data transmission rate of the station [Kbps].
- `rssi`: Measured RSSI [dBm].
- `txRetried`: Count of unicast packets that required retransmission attempts.
- `txMcastPackets`: Number of multicast packets successfully transmitted by the station.
- `txKBytes`: Volume of unicast data transmitted by the station [KBytes].
- `txMcastKBytes`: Volume of multicast data transmitted by the station [KBytes].

Based on RSSI and noise, we can calculate SNR. SNR is defined as the ratio of the signal power to the noise power.

$$\texttt{SNR} = \frac{\texttt{Signal Power}}{\texttt{Noise Power}} \tag{2.1}$$

As both RSSI and noise are measured in dBm, one can write SNR as

$$\texttt{SNR} = \texttt{RSSI} - \texttt{Noise}. \tag{2.2}$$

Wi-Fi data offer insights into usage behavior of the station over time, as it is collected at regular 15-minute intervals, allowing for temporal analysis. This time-component aspect is reflected in the hourly distribution shown in Figure 2.4. Additionally, SNR is a valuable indicator of the communication medium conditions experienced by the station. Figures 2.5 - 2.7 show the distribution of RSSI, noise and the derived metric, SNR. These metrics will be explored further in Section 3.1.1.

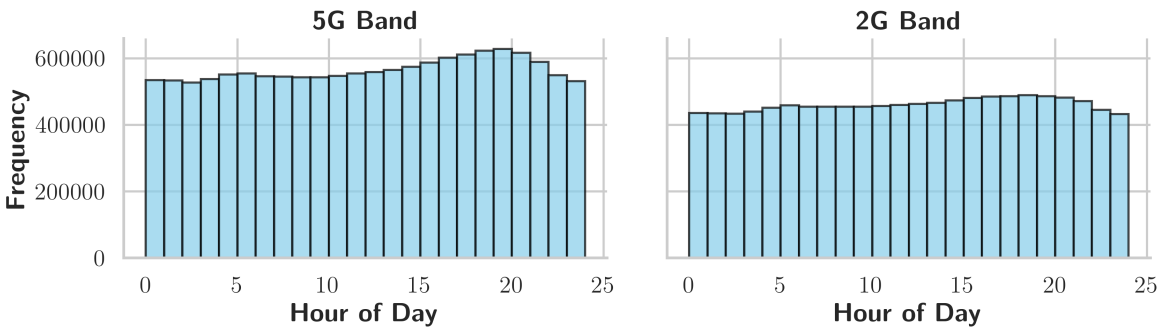

Figure 2.4: Hourly distribution for 5GHz and 2.4GHz bands.

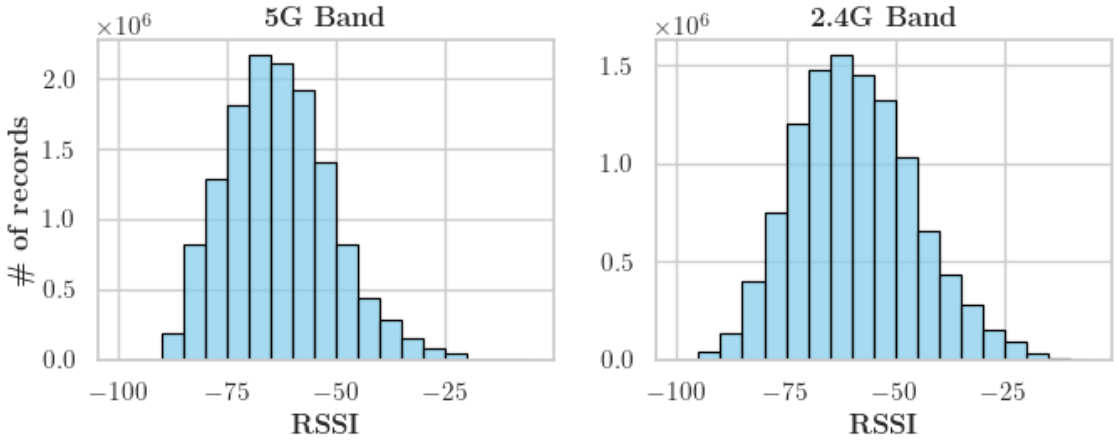

Figure 2.5: RSSI distribution for 5GHz and 2.4GHz bands.

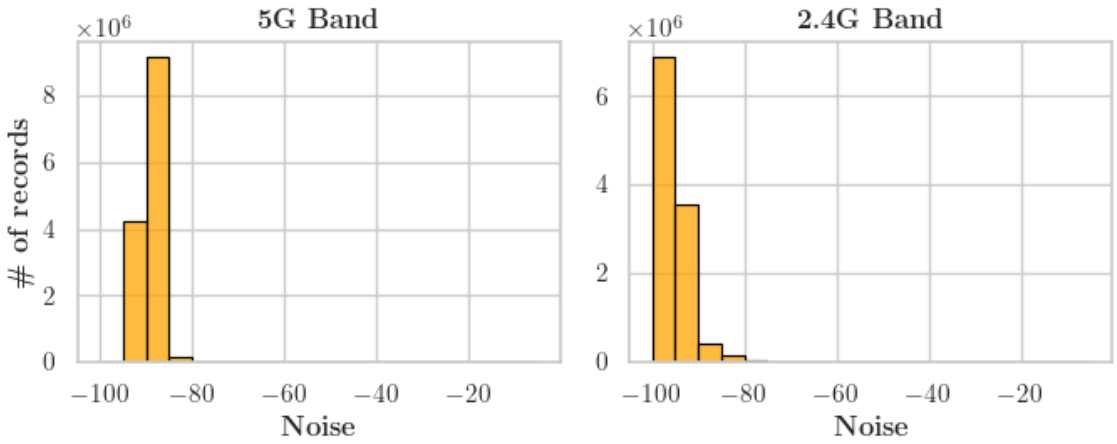

Figure 2.6: Noise distribution for 5GHz and 2.4GHz bands.

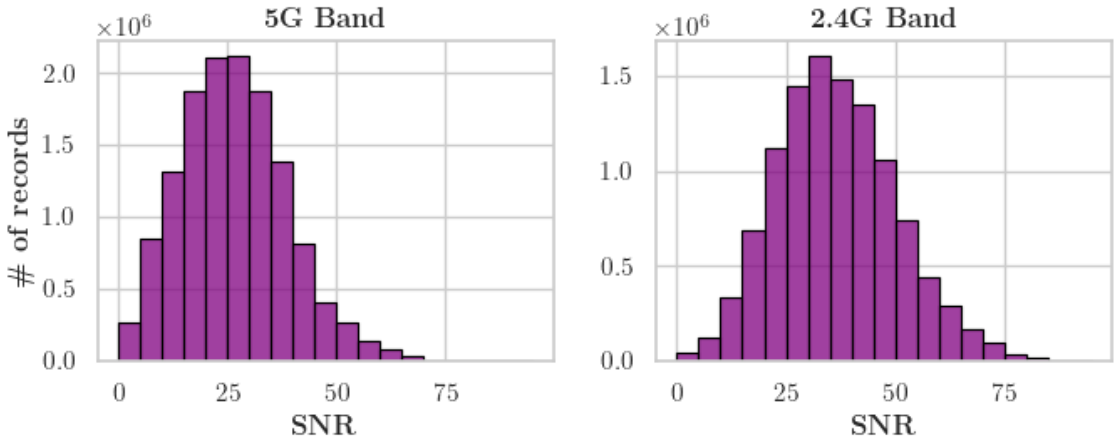

Figure 2.7: SNR distribution for 5GHz and 2.4GHz bands.

2.3. Throughput Measurement Data

Throughput measurement data have measurements conducted on stations to estimate its throughput. An example of collected data is given in Table 2.3.

Table 2.3: Throughput measurement data

timeStamp	mac	band	staTputKbps	staTxPackets	staTxFailed	wakeUpMs	rssi	noise	staTxPacketsRemaining	sentKBytes
YYYY-MM-DDTHH:MM:SS	X:X:X:X	5G	375267	1580	0	0	-65	-88	1	11733
YYYY-MM-DDTHH:MM:SS	X:X:X:X	5G	310601	6248	0	0	-60	-88	1	9460
YYYY-MM-DDTHH:MM:SS	X:X:X:X	5G	192392	989	0	178	-61	-90	1	5906
YYYY-MM-DDTHH:MM:SS	X:X:X:X	2G	73350	753	0	108	-56	-97	0	2346
YYYY-MM-DDTHH:MM:SS	X:X:X:X	2G	3144	60	0	55	-74	-94	0	169

The fields from the data set are:

- `timeStamp`: Masked timestamp of the record.
- `mac`: Masked MAC address of the station.
- `band`: The Wi-Fi band over which the station is operating.
- `staTputKbps`: Station throughput [kbps].
- `staTxPackets`: Count of packets successfully transmitted by the station.
- `statxFailed`: Count of packets failed to transmit successfully.
- `wakeUpMs`: Response time of the station for probing measurement [ms].
- `rssi`: Measured RSSI [dBm].
- `noise`: Average noise measured over the interval [dBm].
- `staTxPacketsRemaining`: Count of packets required to transmit.
- `sentKBytes`: Volume of data transmitted by the station [KBytes].

As these measurements are creating a load on stations traffic for a while, they may decrease QoS. Therefore, measurements are done when there is no active usage. Consequently, unlike the routinely gathered Wi-Fi data, throughput data lack temporal characteristics. Nonetheless, as illustrated through Figures 2.8 - 2.10, the distribution of RSSI, noise and SNR, along with features derived in Section 3.1.2 provides important insights.

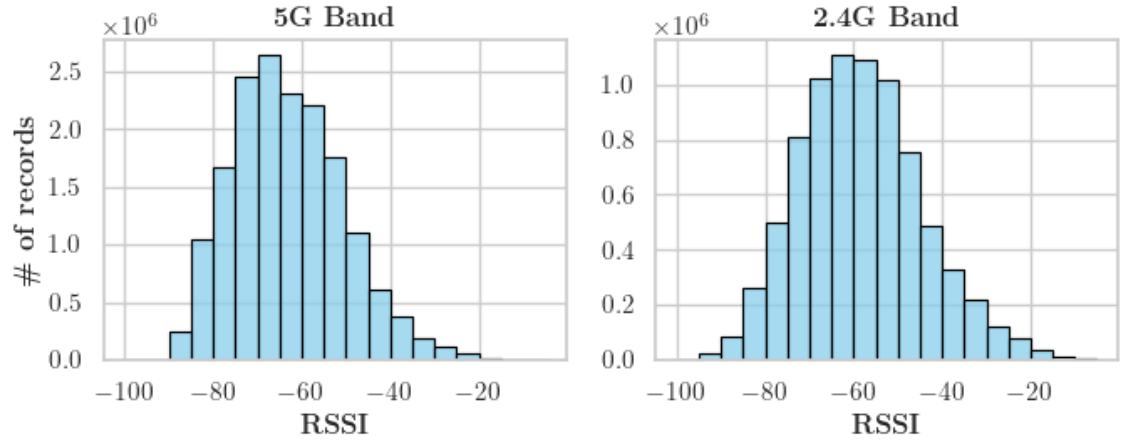

Figure 2.8: RSSI distribution for 5GHz and 2.4GHz bands.

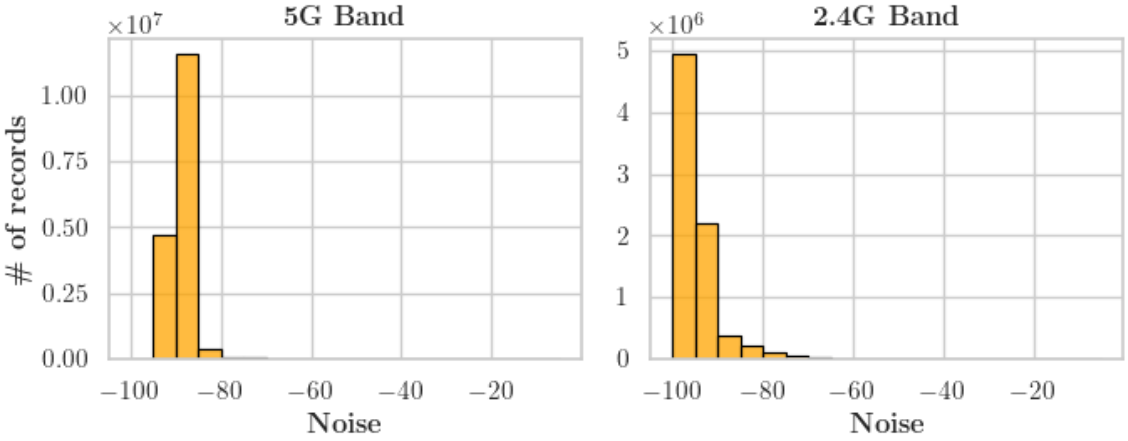

Figure 2.9: Noise distribution for 5GHz and 2.4GHz bands.

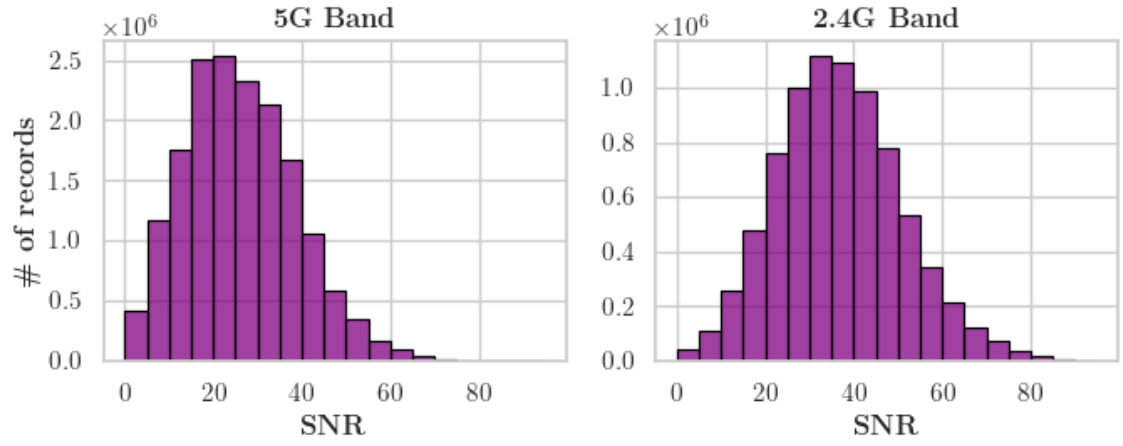

Figure 2.10: SNR distribution for 5GHz and 2.4GHz bands.

3. STATION FINGERPRINTING PIPELINE

In this chapter, we present a pipeline for station identification. We will explore the dataset, derive new features and evaluate their predictive power through techniques like correlation analysis and mutual information scoring.

The processed data is then used to train and evaluate multiple classifiers, ranging from probabilistic models like Naive Bayes to ensemble and neural network methods. We further enhance the pipeline by introducing stacking techniques, combining predictions across bands and from different classifiers to improve overall performance.

Finally, we compare the performance of these approaches using key metrics and discuss the results.

3.1. Feature Engineering

We will process Wi-Fi data, throughput data and protocol data. We derive new features, examine correlation charts and measure mutual information score of all features.

Starting with protocol data, utilizing Wi-Fi data and throughput data, the `band` dimension can be added to Figure 2.3, represented in Figure 3.1 and Figure 3.2. `dhcpOptions` from protocol data is a categorical feature and will be mapped to numberical values via a process called label encoding. Given a categorical variable X with distinct categories $x_1, x_2, \ldots, x_n$, label encoding assigns each unique category an integer value $y \in \{0, 1, \ldots, n-1\}$ such that

$$f(x_i) = y_i. \tag{3.1}$$

Label encoding transforms each unique label into an integer, maintaining category distinctions while allowing numerical computations for working with classifiers in following sections.

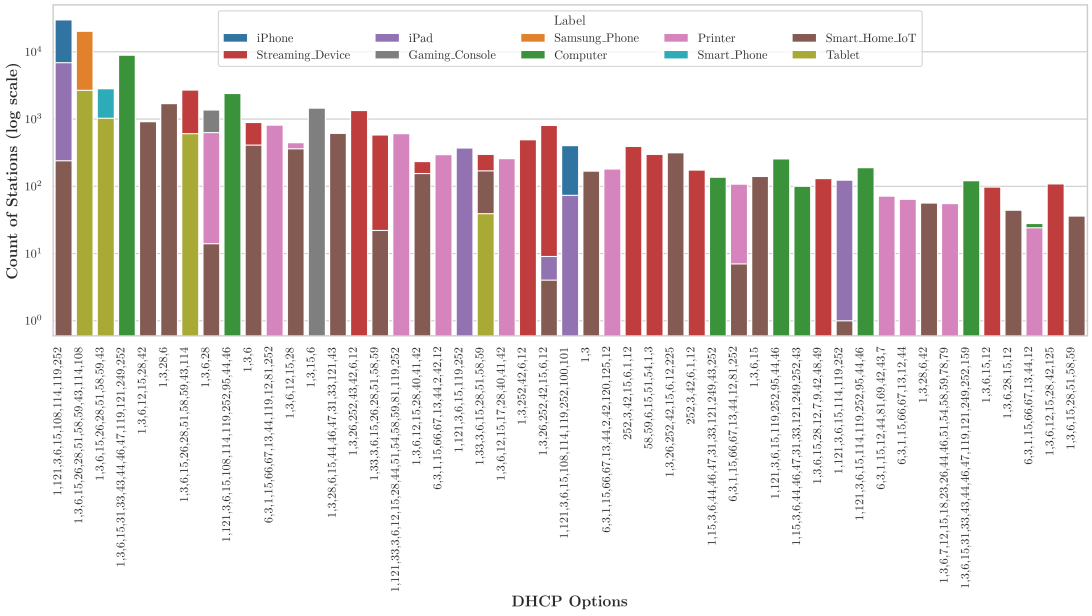

Figure 3.1: Top 50 dhcpOptions by Station Type - 2.4GHz.

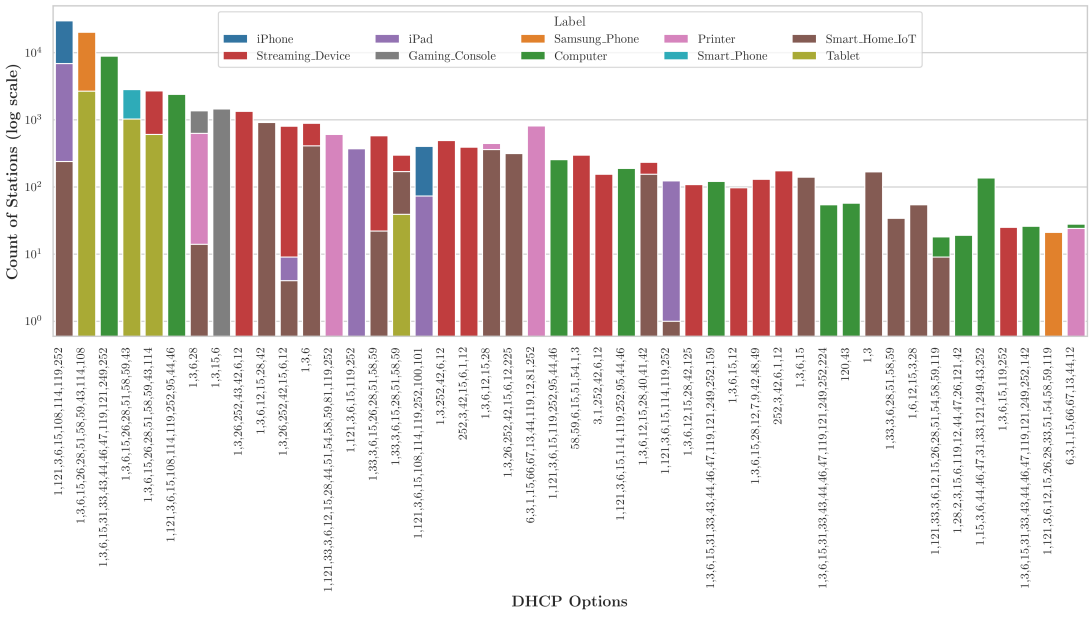

Figure 3.2: Top 50 dhcpOptions by Station Type - 5GHz.

3.1.1. Wi-Fi Operational Data

We start analyzing Wi-Fi operational data by looking at it's correlation matrix. A correlation matrix is a table displaying the correlation coefficients between multiple variables. Each cell in the matrix shows the correlation between two variables, ranging from -1 to +1, where a value of +1 indicates a perfect positive linear relationship, -1 represents a perfect negative linear relationship and 0 implies no linear relationship. The correlation coefficient r_{xy} between two variables x and y is calculated as

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{3.2}$$

where x_i and y_i are individual observations, $\bar{x}$ and $\bar{y}$ are the means of x and y and n is the number of observations. Figures 3.3 and 3.4 show that `txPackets` and `txBytes`, as well as `txMcastPackets` and `txMcastBytes`, exhibit a high degree of correlation.

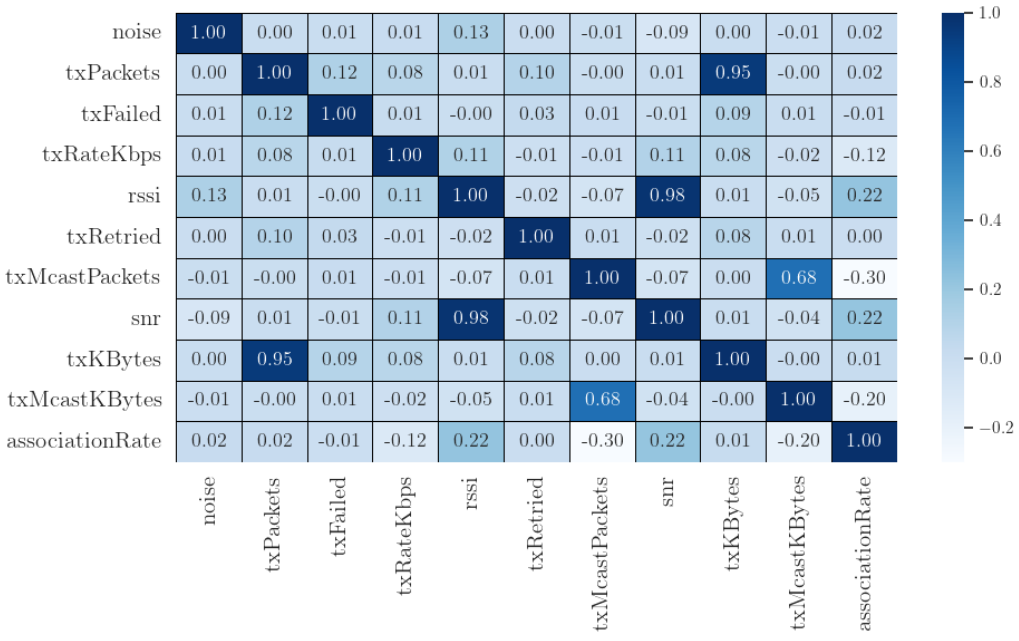

Figure 3.3: Wi-Fi operational data correlation - 2.4GHz.

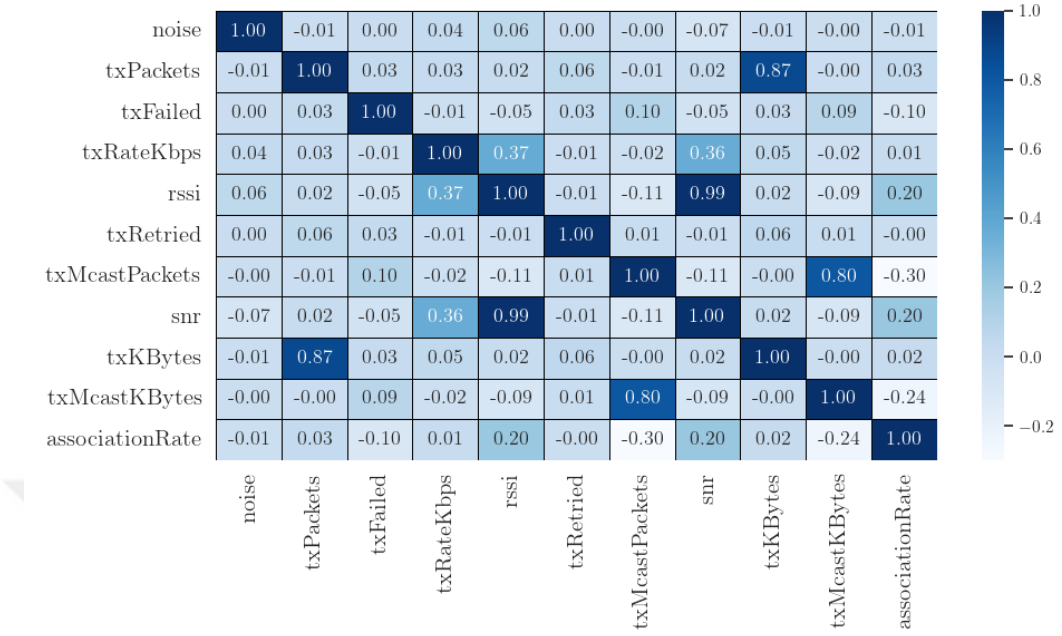

Figure 3.4: Wi-Fi operational data correlation - 5GHz.

Following new fileds are derived:

$$\texttt{txPacketSize} = \frac{\texttt{txBytes}}{\texttt{txPackets}} \tag{3.3}$$

$$\texttt{txMcastPacketSize} = \frac{\texttt{txMcastBytes}}{\texttt{txMcastPackets}} \tag{3.4}$$

$$\texttt{txPacketRate} = \frac{\texttt{txPackets}}{\texttt{txPackets} + \texttt{txFailed} + \texttt{txRetried}} \tag{3.5}$$

With `txPacketSize` defined, we can remove `txKBytes`. Similarly, `txMcastPacketSize` and `txPacketRate` replace `txMcastKBytes` and `txFailed` & `txRetried`, respectively. Additionaly, a strong correlation is observed between RSSI and SNR, which is expected since SNR is derived from both RSSI and noise. We can drop RSSI and noise. However, to be able to measure stations mobility, we can keep standart deviation of RSSI.

As mentioned in Section 2.2, Wi-Fi data is a collection of 15-min interval measurements. This means that there is a time component of this data, i.e. station specific usage patterns through out a time interval can be analyzed. This behaviour can be useful in station detection as different stations will have different usage characteristics over time. To this end, first we define a new field, `associationRate`, which is calcu-

lated as the ratio of time spent in the network, `inNetwork`, to the measure interval of 15 minutes, `elapsedTime`.

$$\texttt{associationRate} = \frac{\texttt{inNetwork}}{\texttt{elapsedTime}} \tag{3.6}$$

It is hard to capture temporal behaviours of stations from 15 minute resolution data without any preprocess. Most of the stations do not have a full session. In other words, the data, in terms of 15 min sessions, are sparse. Figure 3.5 depictes the distribution of session numbers per day for each station. Out of 96 possible 15-min sessions, the median of sessions is around 15 for 2.4GHz and 35 for 5GHz. We can retrieve temporal units like hour, day, week of the each record from `timeStamp`; set a new aggragation level and make the data more dense -hopefully- without comprimsing the information carried. To analyze the data across time intervals, we discretize 24 hours into 4 bins, where each bin represents a 6-hour interval. The bins are defined as

$$\text{bins} = \{[0, 6), [6, 12), [12, 18), [18, 24)\}. \tag{3.7}$$

For each `mac`, `band` and `hour bin` (from `hour_[0,6)` to `hour_[18,24)`), the following fields are aggregated weekly to capture trends and behaviors over time.

- `txPackets`
- `txMcastPackets`
- `txPacketSize`
- `txMcastPacketSize`
- `txPacketRate`
- `associationRate`

For `associationRate`, the aggregation is done over percentiles defined as

$$Q = \{0.25, 0.75, 0.99\}, \tag{3.8}$$

yielding $\texttt{associationRate}_Q$. The dispersion over RSSI is calculated as the standart deviation for each `mac` and `band` over a week period. The new distribution of sessions, calculated over 6 hours of interval, are depicted in Figure 3.6. The median is 2 for

2.4GHz and 3 for 5GHz and quite a lot of stations over both bands reaches the full sesssion, 4.

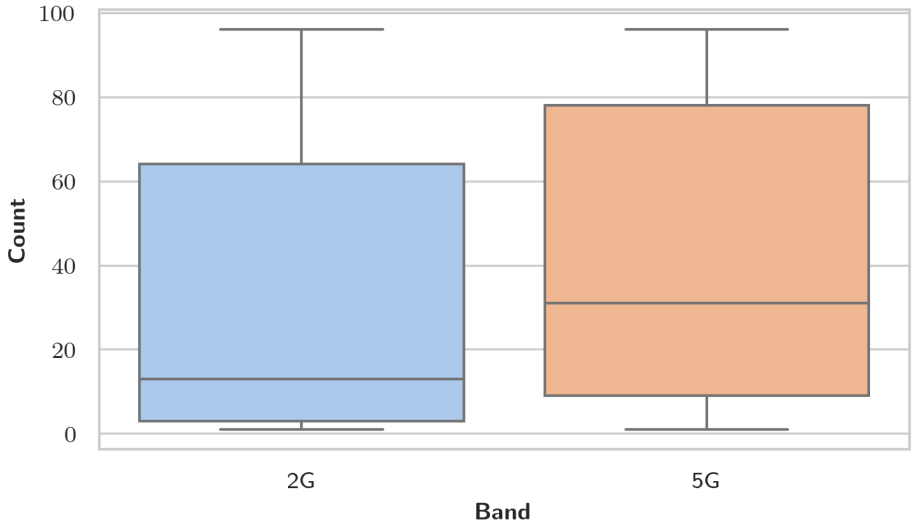

Figure 3.5: Wi-Fi operational data 15-min session distribution.

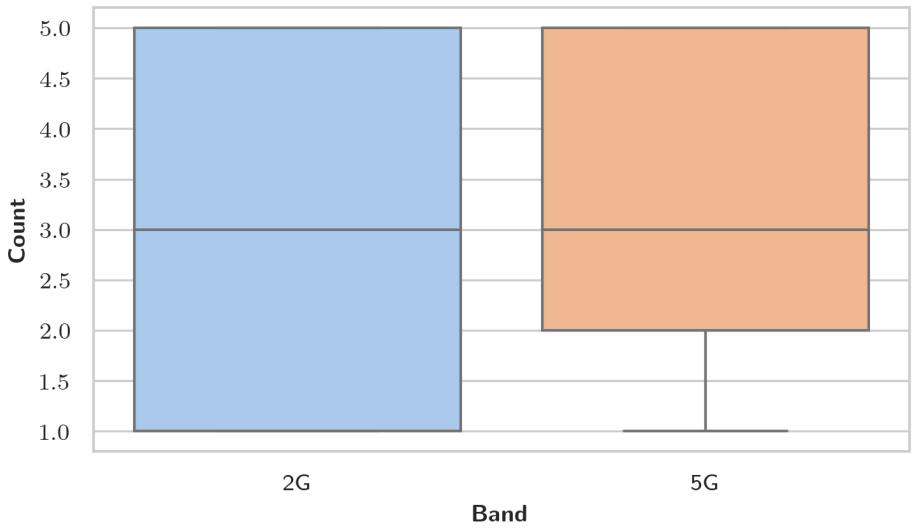

Figure 3.6: Wi-Fi operational data 6-hour interval distribution.

As referred in Section 2.2, SNR is an important indicator of the physical channel conditions that the station operates. These conditions, along with the station characteristics, can impact the transmission rates. To capture the effect of SNR on

txRateKbps better, SNR is mapped to bins defined as

$$\text{bins} = \{[0, 20), [20, 40), [40, \infty)\}. \tag{3.9}$$

Using the same percentiles as $\texttt{associationRate}_Q$ defined in Equation 3.8, for each mac, band and snr_bin, $\texttt{txRateKbps}_{\texttt{snr\_bin},Q}$ is calculated weekly.

3.1.2. Throughput Measurement Data

Figure 3.7 and 3.8 shows the correlation matrix of features in throughput data. Again, RSSI and SNR show a very strong correlation. we will discard RSSI and noise while keeping standart deviation of RSSI to capture station mobility. sentKBytes and staTputKbps have a strong correlation as well. We will keep them as they are pointing related but different things, as first one shows the speed while the second one shows the volume of transmission.

Unlike Wi-Fi data, throughput data records lack temporal significance, as measurements are conducted when stations are idle and not transmitting traffic. Thus, we assume that these measurements are performed randomly, with their time component carrying no meaningful information. However, similar to SNR and txRateKbps relationship, for the same snr bins and quantiles defined in Equations 3.8 and 3.9, $\texttt{staTputKbps}_{\texttt{snr\_bin},Q}$ and $\texttt{sentKbytes}_{\texttt{snr\_bin},Q}$ are calculated. With the same dimensions as $\texttt{staTputKbps}_{\texttt{snr\_bin},Q}$ and $\texttt{sentKbytes}_{\texttt{snr\_bin},Q}$ but marginalized over snr_bin dimension, $\texttt{wakeUpMs}_Q$ are calculated. Finally, again similar to txPacketRate, staTxPacketRate is derived. As with Wi-Fi data, all calculations are refreshed weekly.

$$\texttt{staTxPacketRate} = \frac{\texttt{staTxPackets}}{\texttt{staTxPackets} + \texttt{staTxFailed} + \texttt{staTxPacketsRemaining}} \tag{3.10}$$

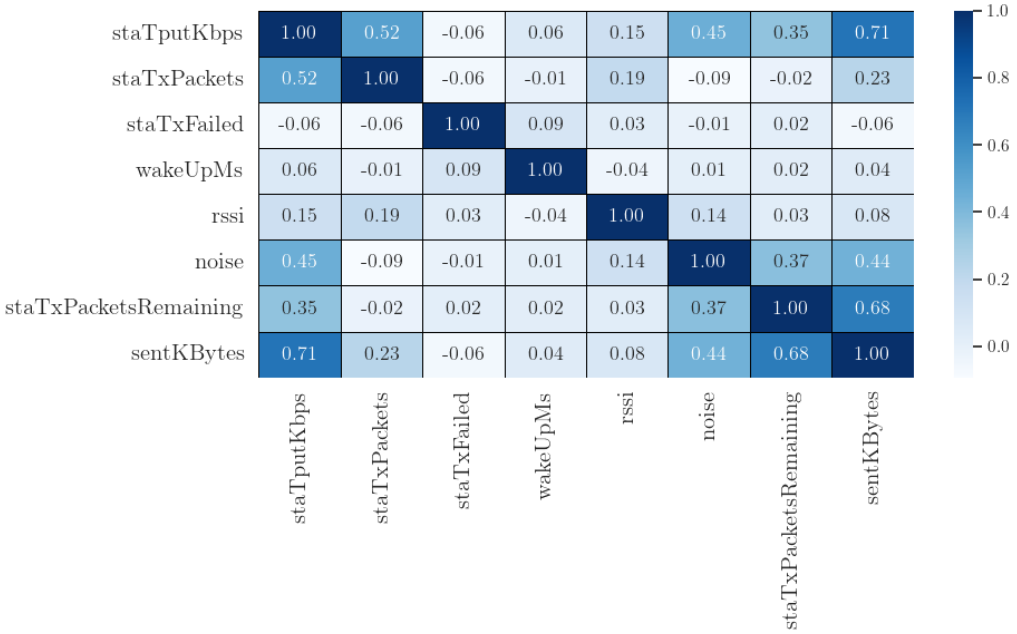

Figure 3.7: Throughput measurement data correlation - 2.4GHz.

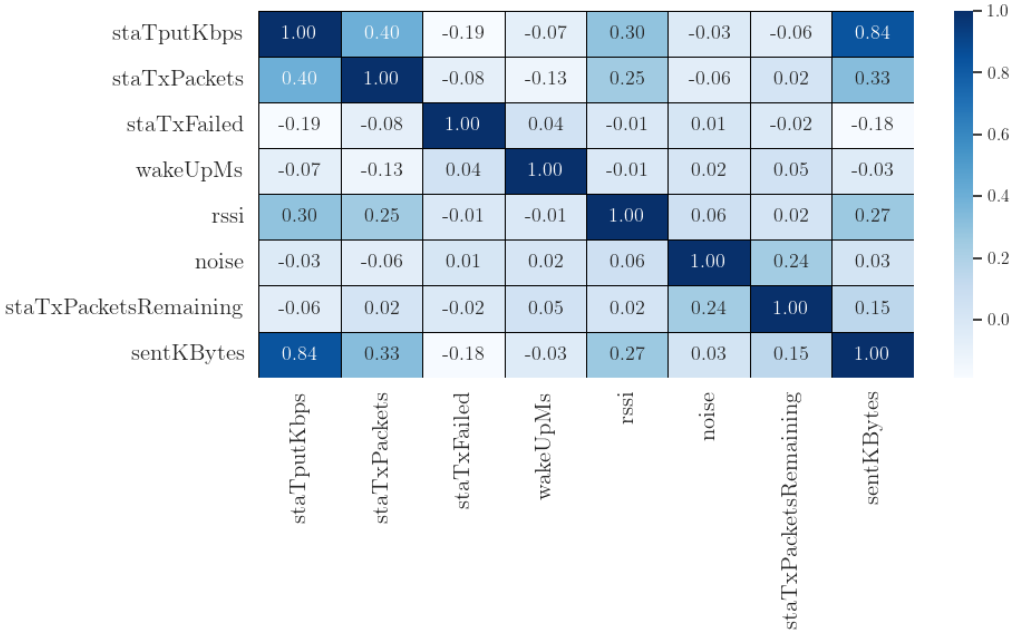

Figure 3.8: Throughput measurment data correlation - 5GHz.

3.1.3. Mutual Information Scores of Features

To see how the features are good in terms of explaining the station type, we can calculate mutual information scores. MI scores quantify the dependency between each feature and the target variable, station types, per band. MI measures the reduction in uncertainty about the target variable given knowledge of the feature. A higher MI score implies a stronger association between the feature and the target, meaning that the feature provides more information about the target variable. For a feature X and a target variable Y, the mutual information $I(X;Y)$ is defined as

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right) \tag{3.11}$$

where $p(x,y)$ is the joint probability distribution of X and Y, while $p(x)$ and $p(y)$ are the marginal probability distributions of X and Y. In practice, the mutual information scores are calculated independently for each feature and the target variable.

We use MI estimator from the scikit-learn library [45] to evaluate the dependency between features and the target variable. The mutual information scores are depicted in through Figures 3.9 - 3.11. These scores are unitless, as they are computed to compare feature relevance rather than correspond to a physical quantity.

From the charts, it is evident that `dhcpOptions` stands out as the most significant feature across both bands. This is followed by the SNR features (notably `snr_40+`), hour features (notably `hour_[0, 6)`) and RSSI dispersion features as `rssi_std_x` derived from Wi-Fi data and `rssi_std_y` derived from throughput data.

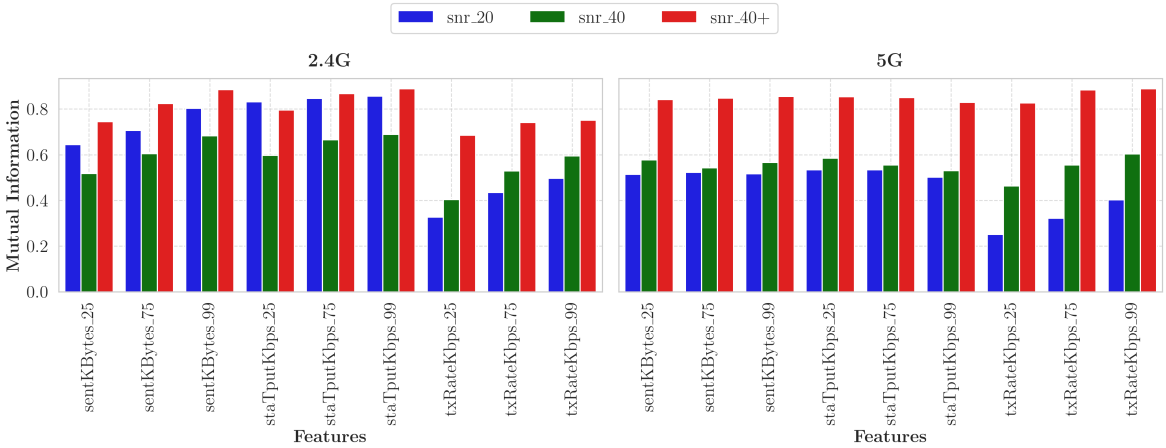

Figure 3.9: Mutual information of SNR features.

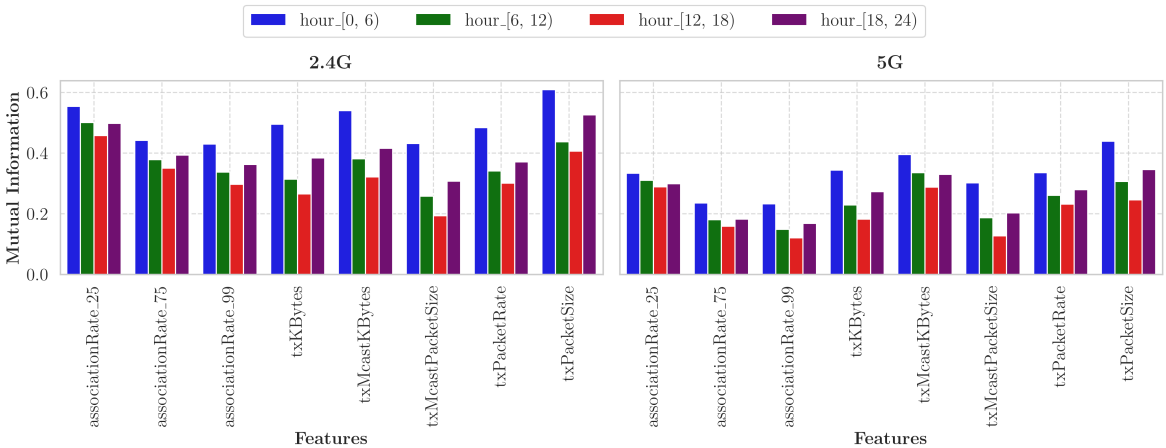

Figure 3.10: Mutual information of hour features.

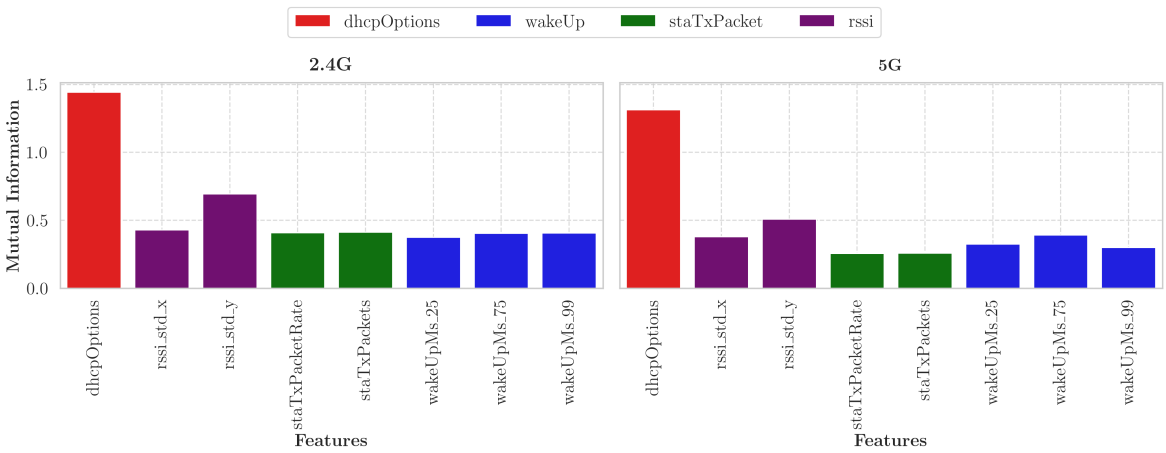

Figure 3.11: Mutual information of remaining features.

3.2. Single Stage Classifiers

In this section, we will be feeding the features obtained from Section 3.1 into various classifiers. We will start with a simple model and gradually increase its complexity, monitoring its performance along the way. The performance metrics we will be using are the confusion matrix. A confusion matrix is an $N \times N$ matrix, where N is the number of classes that the classifier is attempting to predict. Each row represents the actual class and each column represents the predicted class. The matrix definition is given as

$$\text{Confusion matrix} = \begin{bmatrix} \mathrm{C}_{1,1} & \mathrm{C}_{1,2} & \cdots & \mathrm{C}_{1,N} \\ \mathrm{C}_{2,1} & \mathrm{C}_{2,2} & \cdots & \mathrm{C}_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{C}_{N,1} & \mathrm{C}_{N,2} & \cdots & \mathrm{C}_{N,N} \end{bmatrix} \tag{3.12}$$

where $\mathrm{C}_{i,j}$ represents the count of instances with i representing the true class and j the predicted class.

To make a reasonable comparison between classess, the train and test set are obtained with 70-30% ratio with the same seed. The training set is used for model training while test set is reserved for performance evaluation. For 2.4GHz and 5GHz bands, separate classifiers are trained.

3.2.1. Naive Bayes Classifier

NB is a probabilistic classifier based on Bayes' theorem, which calculates the probability of a label given certain feature values by assuming that all features are conditionally independent, given the class label. This assumption, known as the Naive Bayes assumption, simplifies the calculation of the joint probability distribution and allows the model to be trained efficiently. Bayes' theorem is expressed as

$$P(C|X) = \frac{P(X|C)P(C)}{P(X)} \tag{3.13}$$

where

- $P(C)$ is the prior probability of the class C,
- $P(X|C)$ is the likelihood of observing X given class C,
- $P(X)$ is the marginal likelihood of the feature vector X,
- $P(C|X)$ is the posterior probability.

Our first classifier, NB, is built using `dhcpOptions` only. Since `dhcpOptions` is a categorical feature, as discussed in Section 3.1, it was first transformed into `dhcpOptions_encoded` by applying label encoding. As highlighted in Chapter 3.1.3, `dhcpOptions` demonstrates the highest MI score, making it a good candidate for being the base model.

The likelihood $P(X|C)$ is calculated by counting how frequently `dhcpOptions` patterns are observed for each station type in the training data. The model then selects the class with the highest posterior probability as the predicted label.

NB classifier trained over `dhcpOptions` is implemented via `CategoricalNB` from scikit-learn library [45]. The confusion matrix of the NB classifier based on `dhcpOptions` is given in Figures 3.12 - 3.13.

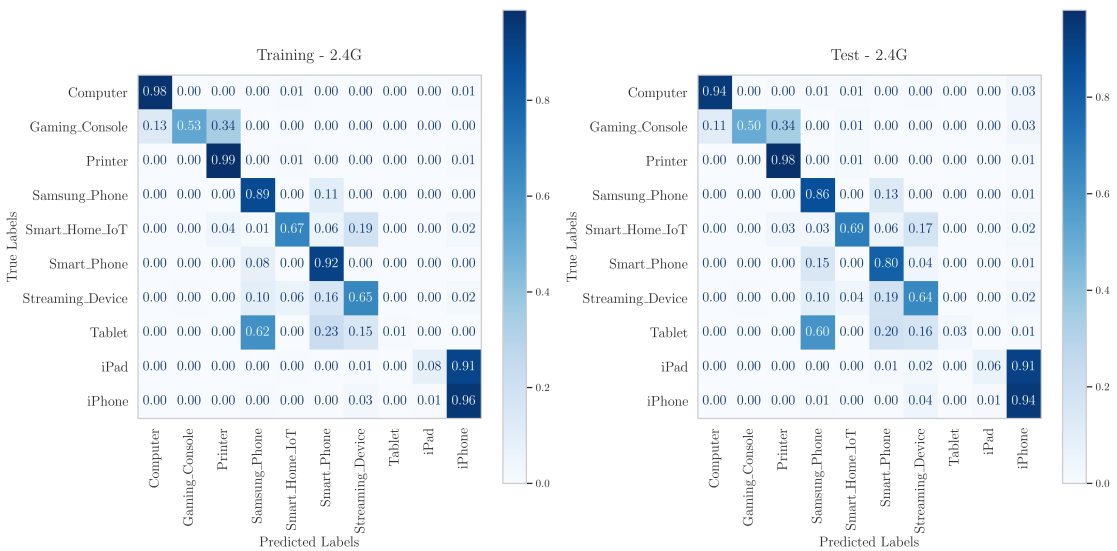

Figure 3.12: Confusion matrix of NB - 2.4GHz.

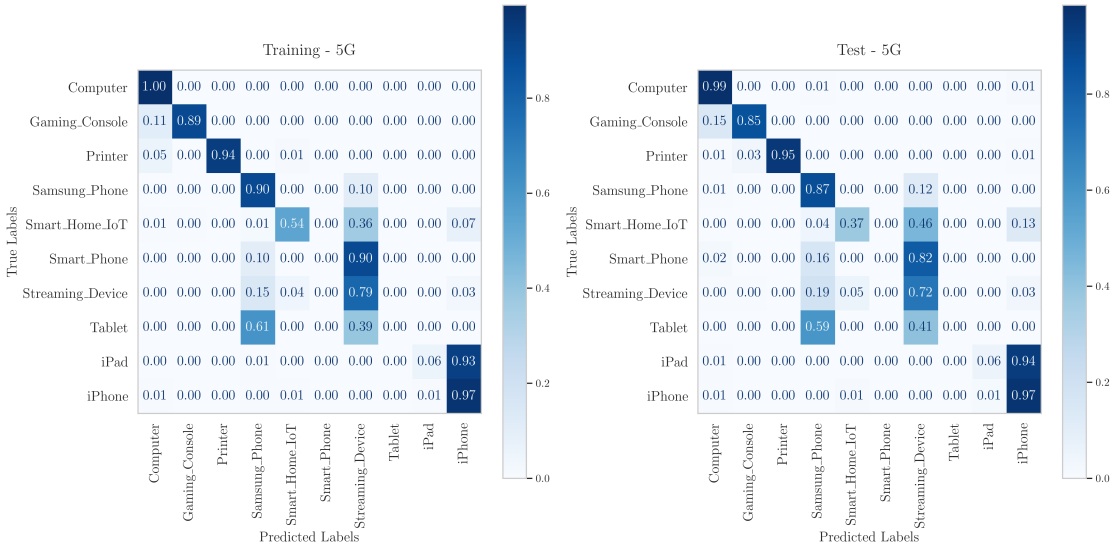

Figure 3.13: Confusion matrix of NB - 5GHz.

For continuous features, we use a Gaussian variant of NB, which assumes that each feature follows a normal distribution within each class. The likelihood is calculated as

$$P(X_i|C) = \frac{1}{\sqrt{2\pi\sigma_C^2}} \exp\left(-\frac{(X_i - \mu_C)^2}{2\sigma_C^2}\right) \tag{3.14}$$

where μ_C and σ_C^2 are the mean and variance of the feature X_i for class C.

GNB classifier is implemented via `GaussianNB` from scikit-learn library [45]. To meet the assumptions of GNB, we restrict the feature set to the following features and normalize them to have zero mean and unit variance before training:

- `snr_40+` features,
- `hour_[0, 6)` features,
- `staTxPacket` features,
- `rssi` features.

The confusion matrices of GNB based on these features are shown through Figures 3.14 - 3.15.

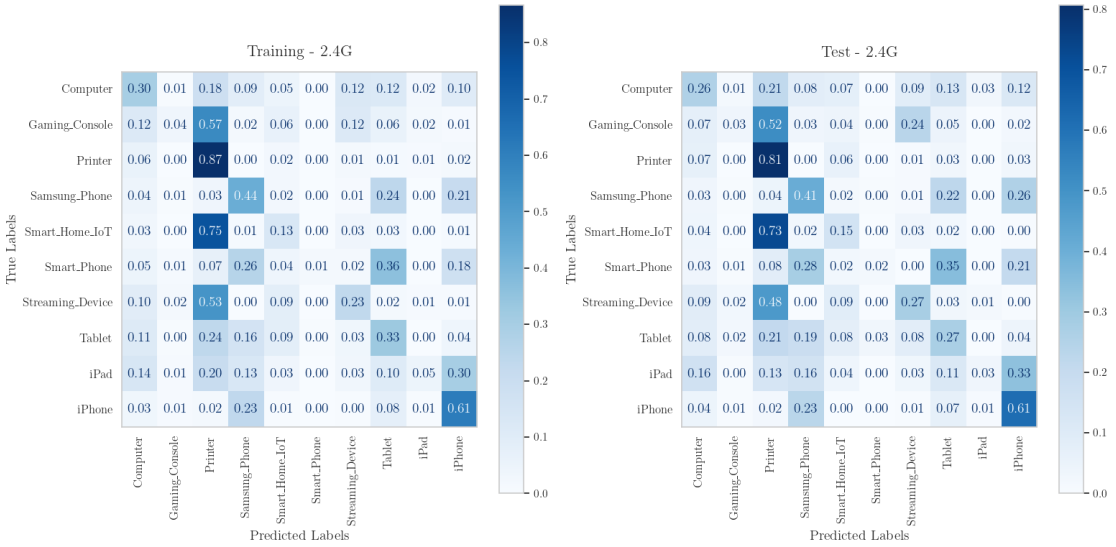

Figure 3.14: Confusion matrix of GNB - 2.4GHz.

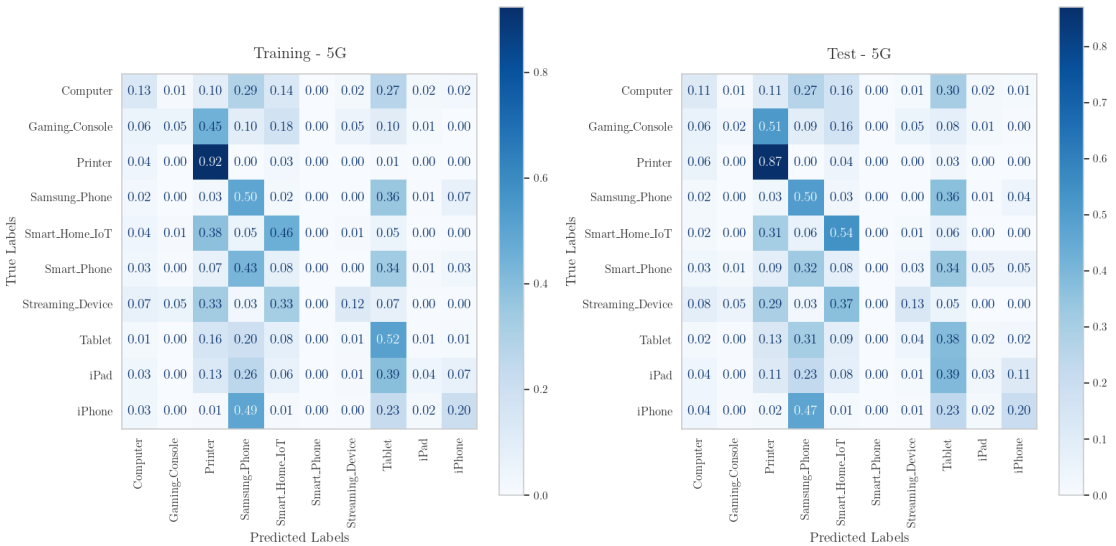

Figure 3.15: Confusion matrix of GNB - 5GHz.

It can be observed from the confusion matrices that NB classifier performs well for certain device types like `computer`, `printer`, `samsung phone`, `iphone` and `streaming device`. This is consistent with Figures 3.1 - 3.2, showing that some `dhcpOptions` are station type-specific. However, as expected, NB classifier struggles to distinguish between stations that share similar `dhcpOptions`. For instance, on both bands, `ipad` predictions are dominated by `iphone` and `tablet` predictions are frequently confused with `samsung phone`, `smart phone` and `streaming device`. On 5GHz, `streaming`

`device` predictions are scattered over `smart home iot`, `smart phone` and `tablet`.

Despite these limitations, NB classifier performs better than GNB. Even with hand-picked features and normalization, GNB struggles to achieve accurate predictions. The assumption of feature independence is too naive for this dataset, which contains complex relationships between features. Additionally, limiting the feature set to `snr_40+` and `hour_[0, 6)` features reduces the model's ability to leverage the full information available. To address these shortcomings, more advanced classifiers will be explored in subsequent sections.

3.2.2. Random Forest Classifier

Random Forest is an ensemble classifier that consists of a collection of decision trees $\{h(x, \Theta_k)\}_{k=1}^{K}$, where each tree $h(x, \Theta_k)$ is trained on a random subset of data samples and features. For classification, the final prediction for a sample x is determined by a majority vote across all trees

$$\hat{y} = \text{majority vote}\{h(x, \Theta_k)\}_{k=1}^{K} \tag{3.15}$$

where

- $h(x, \Theta_k)$ denotes the k-th decision tree in the forest,
- Θ_k represents the random variables governing the selection of samples and features for the k-th tree,
- K is the total number of trees in the forest.

Each decision tree is trained by recursively splitting data based on features that minimize impurity measured by the Gini impurity. The Gini impurity for a node containing classes $C_1, C_2, \ldots, C_m$ is defined as

$$\text{Gini} = 1 - \sum_{i=1}^{m} p_i^2 \tag{3.16}$$

where p_i is the probability of a randomly chosen sample belonging to class C_i. A lower Gini impurity indicates a purer node, as the samples in the node belong more

predominantly to a single class. During training, each tree aims to split the data in a way that minimizes the Gini impurity, thereby increasing classification accuracy.

Using multiple trees reduces variance, enhancing the generalization ability of the model. Additionally, the random selection of features for each tree reduces overfitting by preventing the model from focusing on specific patterns.

We can calculate feature importances of a random forest classifier by analyzing the reduction in the Gini impurity across all trees. For feature x_i, $FI(x_i)$ is calculated based on its contribution to reducing impurity at each split in each tree

$$FI(x_i) = \frac{1}{|T|} \sum_{k=1}^{|T|} \sum_{j \in \text{nodes}(k)} \Delta I_{i,j} \cdot \frac{n_j}{N} \tag{3.17}$$

where

- $\Delta I_{i,j}$ is the reduction in impurity achieved by splitting on feature x_i at node j,
- n_j is the number of samples that reach node j,
- N is the total number of samples in the dataset,
- $|T|$ is the total number of trees in the forest.

We used the `RandomForestClassifier` from scikit-learn library [45] with a set of hyperparameters determined through grid search. A hyperparameter is a parameter whose value is set before the learning process begins, as opposed to model parameters, which are learned during training. Grid search is an exhaustive search method used to find the optimal hyperparameter values by evaluating the model's performance across all possible combinations of specified parameter values. The hyperparameters of `RandomForestClassifier` are presented in Table 3.1. These values were selected based on the results of the grid search, ensuring a balance between model complexity and generalization performance. The full set of features introduced in Section 3.1.3 was used to train the model.

Figure 3.18 presents the feature importances computed by the RF, highlighting the relative contribution of each feature to the model's predictions. The performance of the classifier is summarized in Figures 3.16 and 3.17, which display the confusion matrices for the 2.4GHz and 5GHz bands, respectively.

Table 3.1: Hyperparameters for RF.

Hyperparameter	**Value**
`n_estimators`	300
`max_depth`	10
`min_samples_split`	32
`min_samples_leaf`	16
`max_features`	0.20

Figure 3.18 demonstrates that `dhcpOptions` dominate the rest of the features on both bands. This observation aligns with Figure 3.11 and the performance of NB classifier in the previous section. However, such dominance of a single feature is not ideal, as it may overshadow the contribution of other potentially informative features. This could limit the model's ability to leverage the full complexity and interplay of the feature set, resulting in suboptimal generalization. We will explore this idea further in the next chapter.

Overall, RF classifier performs better than NB due to its relatively improved performance over ipad, tablet, smart phone and smart home iot types. That being said, predictions for ipad with iphone, tablet and smart phone with samsung phone remain confused on both bands. Additionally, the performance for the printer and gaming console classes has deteriorated compared to NB.

3.2.3. Histogram-based Gradient Boosting Trees

HGBT, a gradient boosting algorithm, iteratively improves its predictive performance by minimizing a specified loss function over multiple weak learners. The weak learners in HGBT are decision trees, optimized with a histogram-based approach that efficiently binarizes continuous features, for computational speed and performance [46].

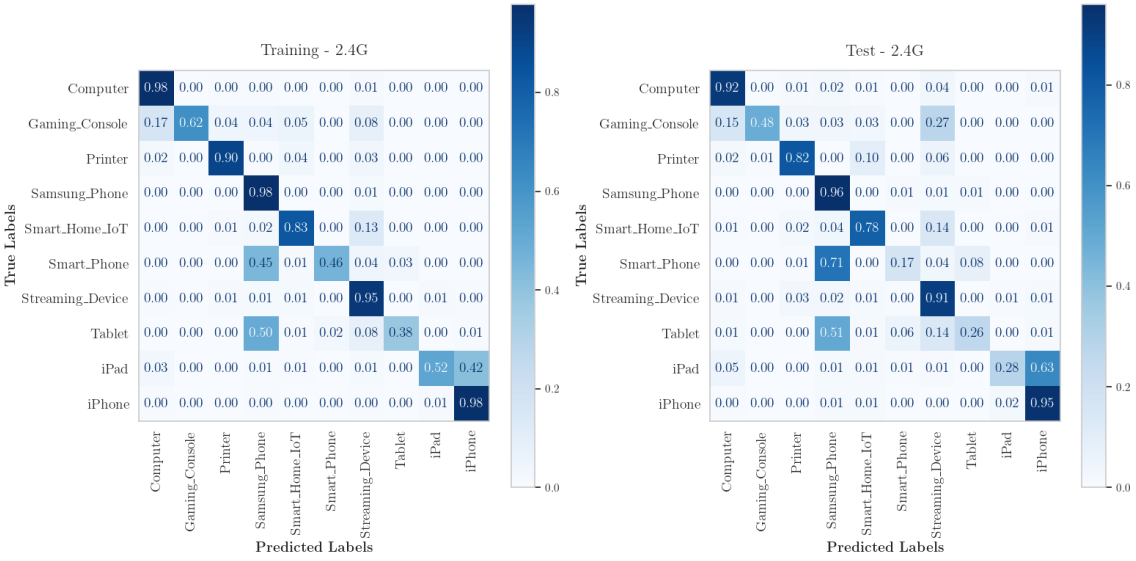

Figure 3.16: Confusion matrix of RF - 2.4GHz.

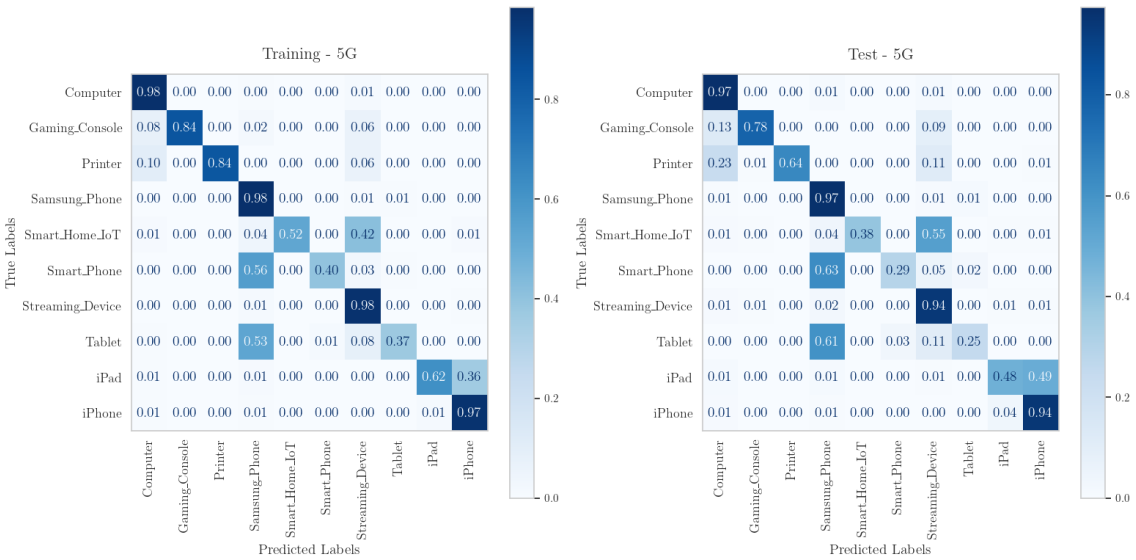

Figure 3.17: Confusion matrix of RF - 5GHz.

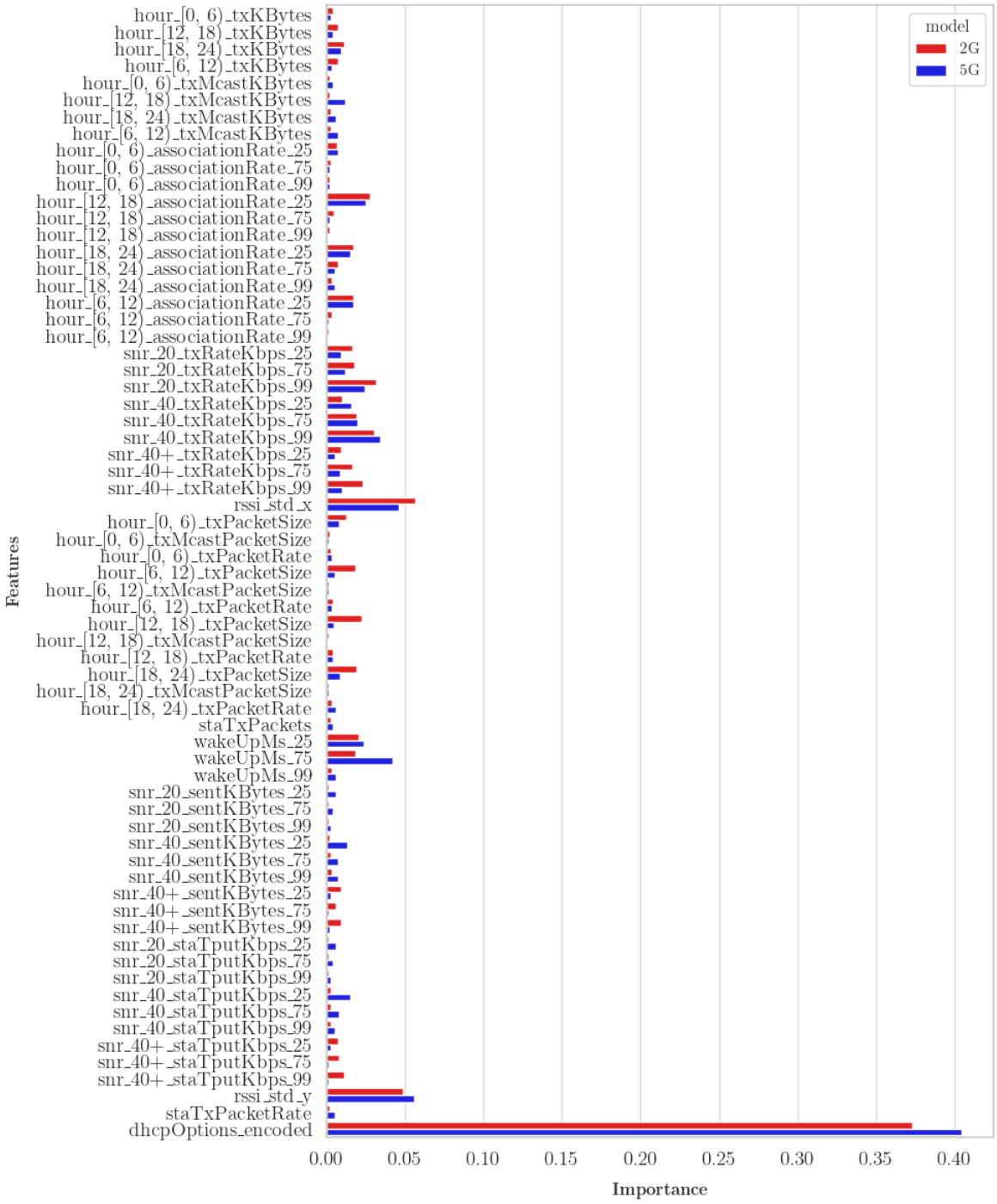

Figure 3.18: Feature importances of RF.

HGBT trains a sequence of estimators $\{h_m(x)\}_{m=1}^{M}$ where each estimator corrects the errors of its predecessors through gradient descent on the loss function. For a classification problem, the prediction for a sample x is expressed as

$$\hat{y} = \arg\max_{y} \sum_{m=1}^{M} h_m(x) \tag{3.18}$$

where

- M is the number of boosting iterations,
- $h_m(x)$ is the decision tree predictor in the m^{th} boosting round,
- the objective is to minimize the cumulative loss across all rounds.

We used `HistGradientBoostingClassifier` from scikit-learn library [45]. Unlike the impurity-based splitting in RF, the gradient-boosting mechanism in HGBT does not naturally produce a clear metric for feature contribution. To measure the effects of features, however, we can use permutation importance. PI is determined by shuffling each feature and observing how the model's performance changes as a result. The steps can be summarized as follows:

(i) For feature x_i, the values of x_i in the test set are randomly shuffled, breaking any association between x_i and the target variable,
(ii) the model's performance is recalculated on the modified dataset,
(iii) the importance of x_i is defined as the decrease in model performance.

$$PI(x_i) = \mathbb{E}\left[\text{Performance}_{\text{original}} - \text{Performance}_{\text{shuffled } x_i}\right] \tag{3.19}$$

where

- $\text{Performance}_{\text{original}}$ is the model's performance on the unaltered dataset,
- $\text{Performance}_{\text{shuffled } x_i}$ is the model's performance after shuffling x_i,
- $\mathbb{E}$ is calculated over multiple shuffling repetitions to obtain a stable estimate.

To measure PI, we used `permutation_importance` from scikit-learn library [45]. The hyper parameters for HGBT is selected via grid search and shown in Table 3.2. PI of features and confusion matrix are shown through Figures 3.21 - 3.20.

Table 3.2: Hyperparameters for HGBT.

Hyperparameter	**Value**
`max_iter`	200
`min_samples_leaf`	16
`learning_rate`	0.01
`max_leaf_nodes`	15
`max_bins`	128
`l2_regularization`	0.1
`early_stopping`	True

Figure 3.21 reveals that, similar to RF, `dhcpOptions` overwhelmingly influence the model's predictions, dominating the other features on both bands. This observation reinforces the concern discussed in Section 3.2.2 regarding the potential overshadowing of other features by a single dominant feature. While `dhcpOptions` provides strong predictive power, its dominance raises questions about the model's ability to fully leverage the diversity of the feature set.

Despite this limitation, Figures 3.19 and 3.20 indicate that HGBT achieves slightly better overall performance than RF. The classifier demonstrates excellent performance for six classes, including gaming console, printer and streaming device. However, similar challenges remain in accurately distinguishing between certain overlapping categories such as smart phone, smart home iot, tablet and ipad.

3.2.4. Multi-Layer Perceptron Classifier

The last classifier we experimented with is MLP. An MLP classifier is a feedforward artificial neural network that consists of multiple layers of nodes. Each node in the network applies a non-linear activation function to its input and passes the result to the next layer.

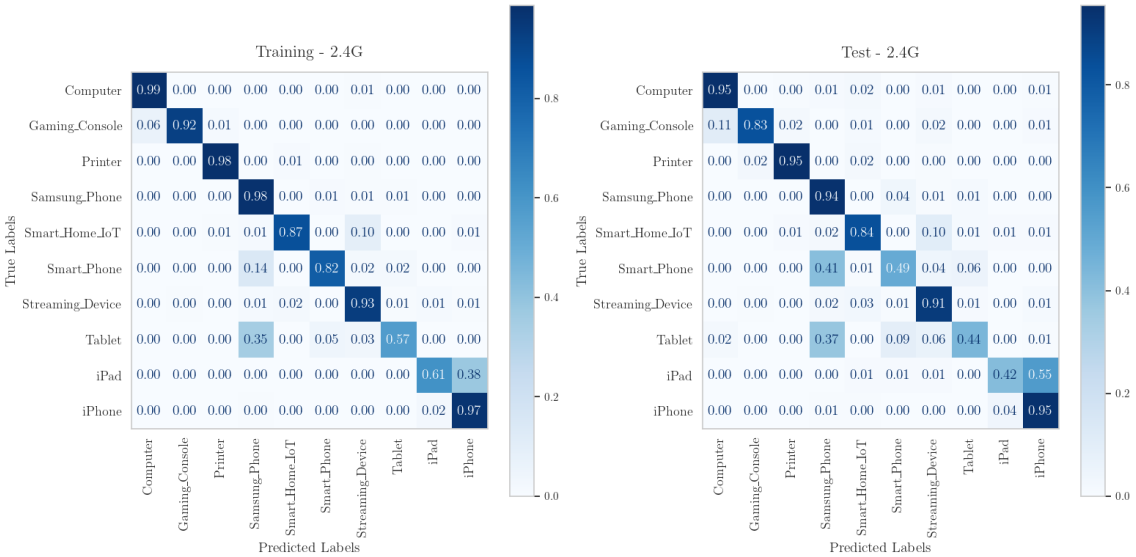

Figure 3.19: Confusion matrix of HGBT - 2.4GHz.

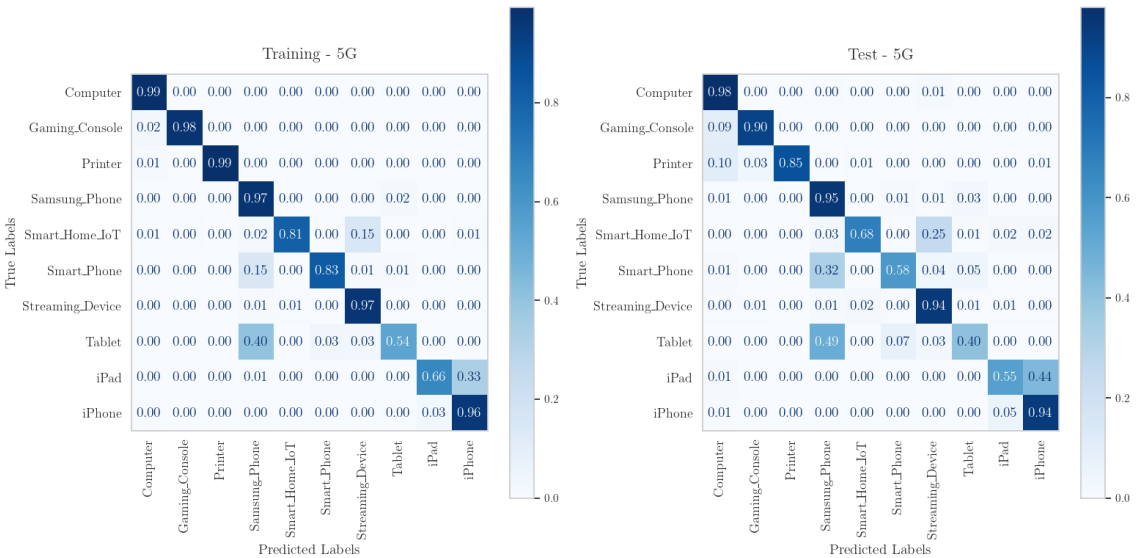

Figure 3.20: Confusion matrix of HGBT - 5GHz.

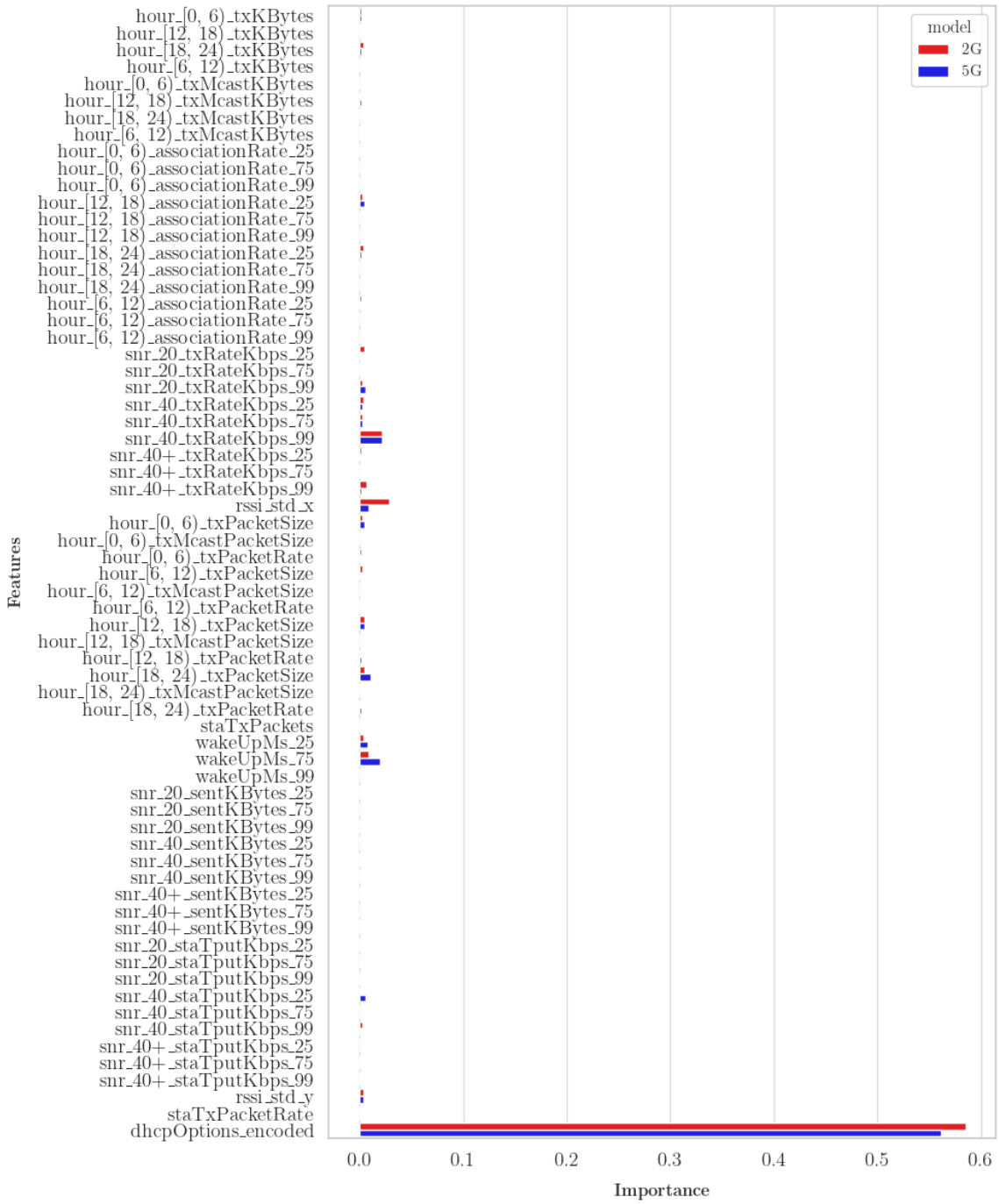

Figure 3.21: Feature importances of HGBT.

The MLP architecture consists of L layers, with the first $L-1$ layers serving as hidden layers for feature extraction and the last layer designed to output class probabilities. For a given input x, the output of the MLP is computed as

$$\hat{y} = \text{softmax}\left(W_L \cdot \sigma(W_{L-1} \cdot \ldots \sigma(W_1 \cdot x + b_1) \cdots + b_{L-1}) + b_L\right) \tag{3.20}$$

where

- W_l and b_l are the weights and biases of the l-th layer,
- $\sigma(\cdot)$ is the non-linear activation function,
- softmax is the activation function applied to the final layer to produce probabilities.

Unlike tree-based classifiers such as RF and HGBT, MLP classifier cannot inherently interpret categorical features encoded as integers via label encoding. This limitation arises because MLP relies on continuous numerical inputs and treating label-encoded integers as numerical values introduces unintended ordinal relationships between categories, which can distort the learning process.

To address this, the `dhcpOptions` feature was transformed using one-hot encoding. One-hot encoding represents each unique category as a separate binary feature, where a value of 1 indicates the presence of that category and 0 indicates its absence. This approach ensures that all categories are treated equitably, without implying any numerical order or hierarchy among them, enabling MLP classifier to effectively process categorical data. In our case, one-hot encoding expanded the feature set by adding 116 new columns, increasing the dimensionality significantly.

MLP also requires all numerical features to be scaled to facilitate efficient training. This is because neural networks use gradient-based optimization methods, which can be negatively impacted by features with vastly different scales. To address this, we used `StandardScaler` from scikit-learn library [45]. This scaler standardizes each feature by subtracting its mean and dividing by its standard deviation, ensuring that all features

have zero mean and unit variance,

$$x_{\text{scaled}} = \frac{x - \mu}{\sigma} \tag{3.21}$$

where μ and σ are the mean and standard deviation of the feature, respectively.

We used `MLPClassifier` from scikit-learn library [45]. To optimize the MLP architecture and hyperparameters, grid search was employed. Table 3.3 summarizes the hyperparameters selected.

Table 3.3: Hyperparameters for MLP Classifier.

Hyperparameter	**2.4GHz Model**	**5GHz Model**
`hidden_layer_sizes`	$(300, 150, 25)$	$(400, 250, 25)$
`alpha`	0.8	0.5
`learning_rate_init`	0.001	0.001
`max_iter`	1000	1000
`activation`	`relu`	`relu`
`solver`	`adam`	`adam`
`early_stopping`	`True`	`True`

Figures 3.22 and 3.23 show the confusion matrices for the 2.4GHz and 5GHz MLP models. Compared to RF and HGBT, the MLP model demonstrates noticeably weaker performance. While RF and HGBT leverage label-encoded `dhcpOptions`, the MLP model uses a one-hot-encoded representation, which appears to limit its ability to effectively utilize this feature. This limitation is most evident in the significantly degraded performance of the MLP model on computer, printer, streaming device and gaming console classes.

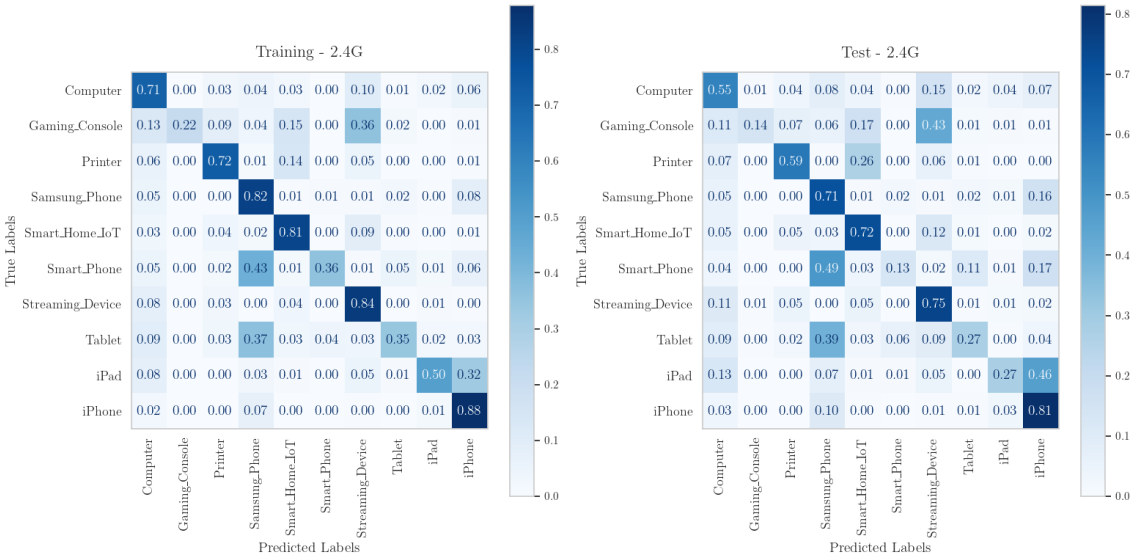

Figure 3.22: Confusion Matrix of MLP - 2.4GHz.

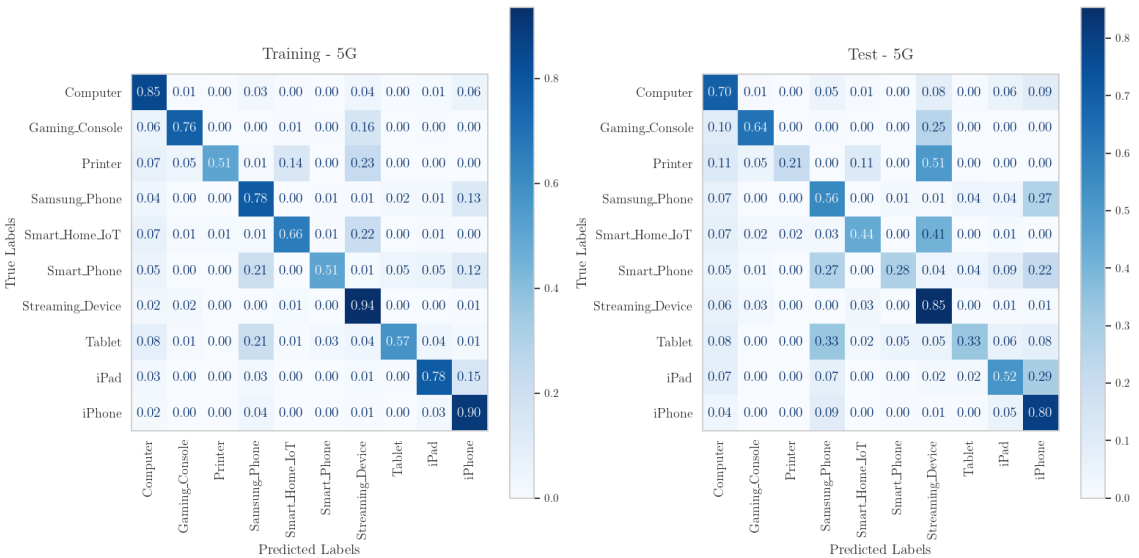

Figure 3.23: Confusion Matrix of MLP - 5GHz.

3.3. Stacked Classifiers

The classifiers explored in the previous chapter generate station class predictions for both bands, 2.4GHz and 5GHz. Their performances were evaluated using a confusion matrix, which highlights the recall metric across different classes. The confusion matrix uses recall to ensure that the sum of each row, representing the true label, equals to 1. Another important metric for classification problems is precision, along with the harmonic mean of precision and recall, known as f1-score. For a given class i, these metrics are defined as

$$\text{precision}_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i} \tag{3.22}$$

$$\text{recall}_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i} \tag{3.23}$$

$$\text{f1-score}_i = 2 \cdot \frac{\text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i} \tag{3.24}$$

where TP_i, FP_i and FN_i denote the true positives, false positives and false negatives for class i, respectively.

Most stations are capable of operating in both bands. 2.4GHz band supports lower Tx/Rx rates compared to the 5GHz, but it offers better RSSI as the station moves farther from AP. This difference in characteristics enables dual-band stations to dynamically switch between bands based on the prevailing network conditions. For instance, a station may transition from 5GHz to 2.4GHz when signal strength diminishes, prioritizing connectivity over speed.

On the other hand, certain stations are restricted to operating on a single band. This limitation may arise due to the station's Wi-Fi chipset, which supports only 2.4GHz (e.g. printers, smart home iot devices), or due to AP settings configured by the user. In any case, there is a significant overlap between the stations classified by models trained on 2.4GHz and 5GHz.

Broadly speaking, the classifiers exhibit comparable performance across both bands. However, for specific station classes, the performance of the models trained on 2.4GHz and 5GHz can vary considerably. Figure 3.24 presents the f1-scores for all classifiers and station types, highlighting these difference.

To improve station identification regardless of the band, predictions from both bands can be merged into a final prediction. Instead of using the raw predictions, class probabilities generated by the models trained on 2.4GHz and 5GHz can be combined and fed into a classifier. The following sections will explore this approach in detail.

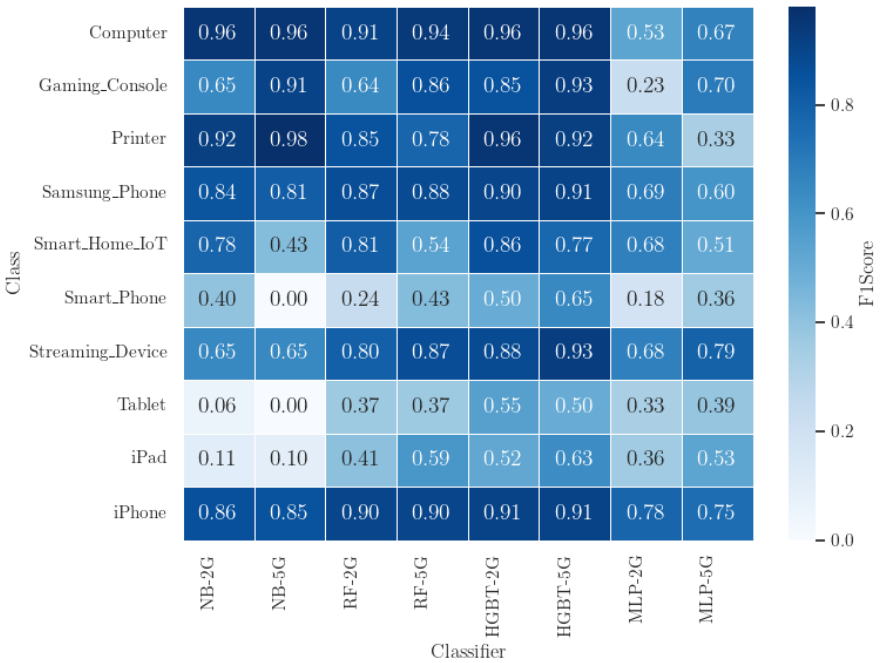

Figure 3.24: F1-scores of class and classifiers.

3.3.1. Stacking Bands

We apply band stacking idea to NB, RF, HGBT and MLP classifiers separately. As depicted in Figure 3.25, models are trained on bands independently, producing class probabilities. These probabilities are stacked together to form a new feature set for

the final classifier, as $\mathbf{X}_{\text{final}}$.

$$\mathbf{X}_{\text{final}} = \begin{bmatrix} P_{\text{Computer\_2g}} & P_{\text{Computer\_5g}} \\ P_{\text{Gaming\_Console\_2g}} & P_{\text{Gaming\_Console\_5g}} \\ P_{\text{Printer\_2g}} & P_{\text{Printer\_5g}} \\ P_{\text{iPad\_2g}} & P_{\text{iPad\_5g}} \\ \vdots & \vdots \end{bmatrix} \tag{3.25}$$

For the final classifier, logistic regression is selected due to its simplicity and interpretability, making it a natural choice for stacked generalization [44]. It models the probability of a class y as

$$P(y = 1|\mathbf{X}_{\text{final}}) = \frac{1}{1 + e^{-(\mathbf{w}\cdot\mathbf{X}_{\text{final}}+b)}} \tag{3.26}$$

where $\mathbf{X}_{\text{final}}$ is defined in Equation 3.25, $\mathbf{w}$ is the weight vector and b is the bias term. One advantage of using logistic regression is that it learns separate parameters for each class. This means that, while we trained NB, RF, HGBT and MLP classifiers for different bands, the logistic regression provides a unique model for each class by optimizing distinct weights for each output class. We used `LogisticRegressionCV` from scikit-learn library [45]. The hyper parameters of the logistic regression classifier is given in Table 3.4.

Table 3.4: Hyperparameters for logistic regression.

Hyperparameter	**Value**
`fit_intercept`	True
`max_iter`	400
`penalty`	`l2`
`solver`	`sag`
`Cs`	0.1
`multi_class`	`multinomial`

Figure 3.26 presents the coefficients calculated by logistic regression for each class, with RF serving as the meta-classifier. The coefficients show similar patterns when HGBT, MLP, or NB is used as meta-classifiers. Notably, certain class probabilities have multiple positive coefficients contributing to different classes. For instance, `computer_proba2g` and `computer_proba5g` exhibit high coefficients for the computer class but also show positive contributions for the gaming console class. Similarly, iphone

and ipad, or samsung phone with smart phone and tablet, display overlapping positive coefficients. This behavior is expected, as these are the classes the classifiers find challenging to distinguish, and the final classifier coefficients confirm these difficulties.

Figure 3.27 shows f1-score for all classifiers and station types, stacked version of the classifiers added. It's visible from this chart that, stacking bands increase performance for most of the classes; and overall except for NB - 5G, the stacked classifier outperforms 2.4GHz and 5GHz classifiers.

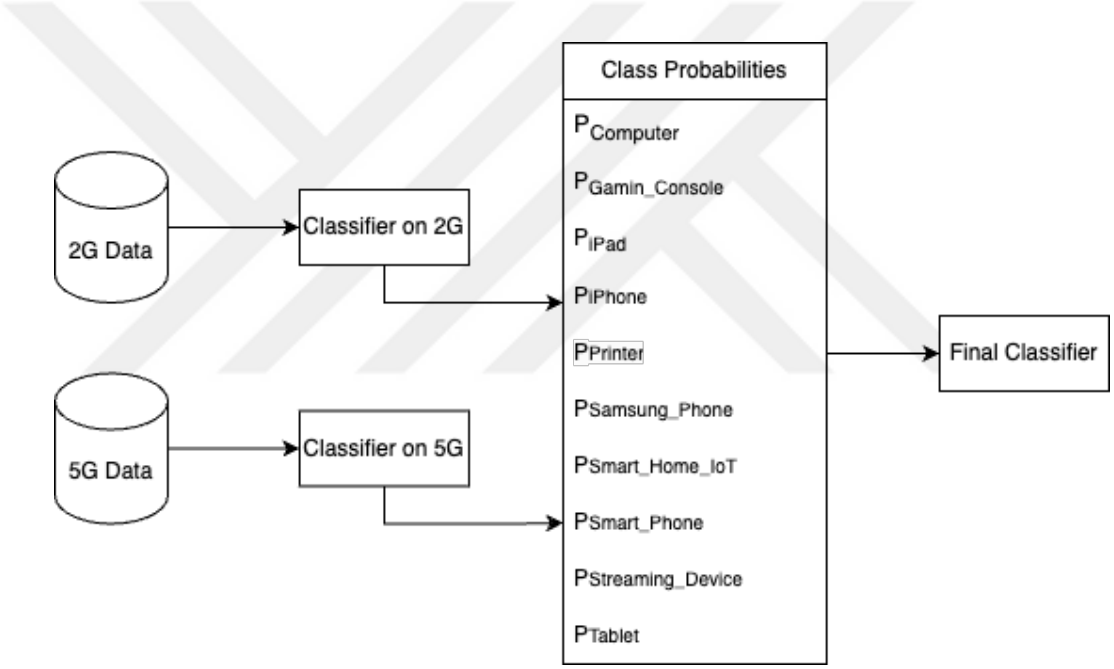

Figure 3.25: Band stacking schema.

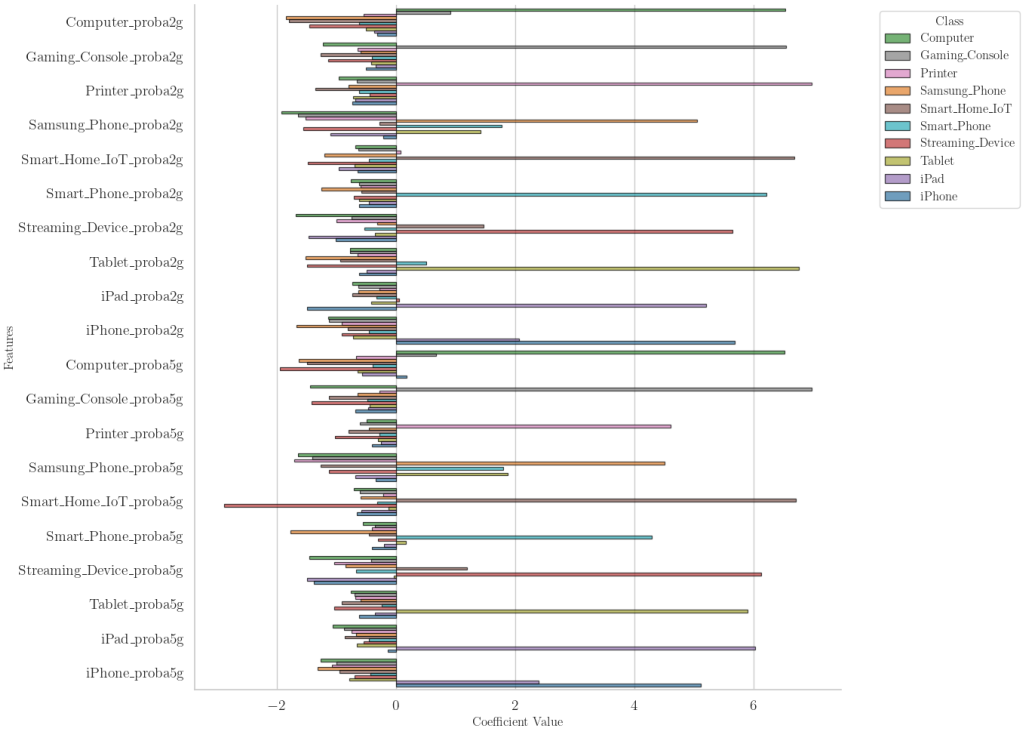

Figure 3.26: Logistic regression coefficients - RF as meta classifier.

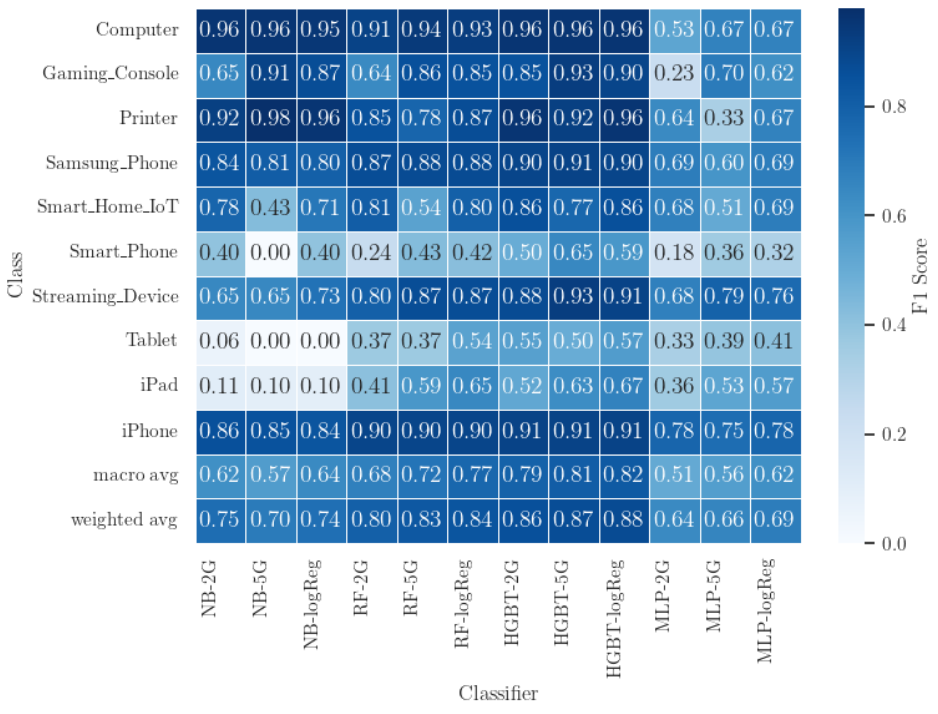

Figure 3.27: F1-scores of class and classifiers.

3.3.2. Stacking NB With Other Classifiers

Here, we take a step further by stacking NB classifier with RF, HGBT and MLP classifiers separately. As illustrated in Figures 3.18 and 3.21, `dhcpOptions` emerges as the most influential feature, dominating others in terms of predictive power. However, Figure 3.27 reveals that NB classifiers trained with `dhcpOptions` alone are insufficient for effectively distinguishing between the tablet, smart phone and ipad classes. The same figure also demonstrates that both RF and HGBT classifiers outperform NB, particularly for these challenging classes.

Furthermore, as discussed in Section 3.2.4, using MLP introduces an additional challenge: the requirement for one-hot encoding of `dhcpOptions`. This preprocessing step significantly increases the dimensionality of the input data, expanding the number of columns by approximately threefold. While Figure 3.27 confirms that MLP classifier generally performs worse than RF and HGBT and often worse than NB, it shows a notable improvement over NB for classes where `dhcpOptions` fail to provide sufficient discriminatory power.

To mitigate the dominance of `dhcpOptions` over other features and to significantly reduce the input dimensions for MLP, we discard `dhcpOptions` from the input data for RF, HGBT and MLP. These models are trained using the remaining features. The `dhcpOptions` feature, however, is exclusively utilized in NB. All models are trained for both bands, their class probabilities are stacked, represented as $\mathbf{X}_{\text{final}}$, fed into a final classifier, as depicted in Figure 3.28. The hyper parameters for RF, HGBT and MLP are obtained via grid search.

$$\mathbf{X}_{\text{final}} = \begin{bmatrix} P_{\text{Computer\_2g\_nb}} & P_{\text{Computer\_2g\_rf/hgbt/mlp}} & P_{\text{Computer\_5g\_nb}} & P_{\text{Computer\_5g\_rf/hgbt/mlp}} \\ P_{\text{Gaming\_Console\_2g\_nb}} & P_{\text{Gaming\_Console\_2g\_rf/hgbt/mlp}} & P_{\text{Gaming\_Console\_5g\_nb}} & P_{\text{Gaming\_Console\_5g\_rf/hgbt/mlp}} \\ P_{\text{Printer\_2g\_nb}} & P_{\text{Printer\_2g\_rf/hgbt/mlp}} & P_{\text{Printer\_5g\_nb}} & P_{\text{Printer\_5g\_rf/hgbt/mlp}} \\ P_{\text{iPad\_2g\_nb}} & P_{\text{iPad\_2g\_rf/hgbt/mlp}} & P_{\text{iPad\_5g\_nb}} & P_{\text{iPad\_5g\_rf/hgbt/mlp}} \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \tag{3.27}$$

Table 3.5: Hyperparameters for RF as meta classifier.

Hyperparameter	Value
n_estimators	300
max_depth	10
min_samples_split	32
min_samples_leaf	16
max_features	0.20

Table 3.6: Hyperparameters for HGBT as meta classifier.

Hyperparameter	Value
max_iter	200
min_samples_leaf	32
learning_rate	0.01
max_leaf_nodes	15
max_bins	128
l2_regularization	0.1
early_stopping	True
n_iter_no_change	20

Table 3.7: Hyperparameters for MLP as meta classifiers.

Hyperparameter	Value (2.4GHz)	Value (5GHz)
activation	relu	relu
alpha	0.8	0.5
early_stopping	True	True
hidden_layer_sizes	(400, 200, 50)	(300, 150, 25)
learning_rate_init	0.001	0.001
max_iter	1000	1000
solver	adam	adam

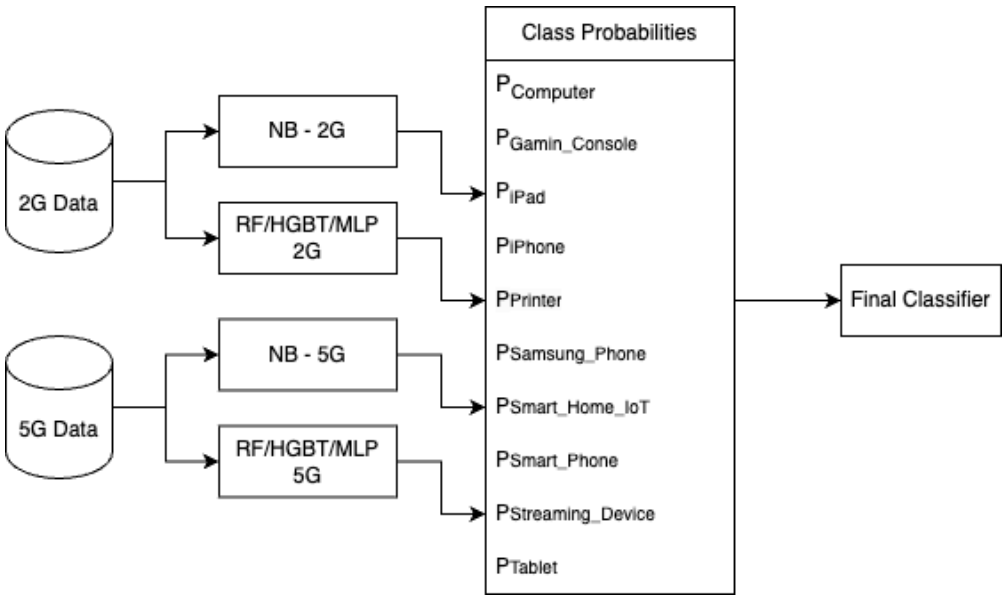

Figure 3.28: Stacking NB with RF/HGBT/MLP.

The feature importances of the meta-classifiers, RF and HGBT are presented in Figure 3.29. Compared to Figure 3.18 and 3.21, here we see a more balanced chart.

We can pick logistic regression as the final classifier, like in the previous section. We use the same parameters shown in Table 3.4. Figure 3.30 shows the coefficients of the logistic regression with RF/HGBT stacked with NB being the meta classifiers. As $\mathbf{X}_{\text{final}}$ doubles in size with this approach, we can experiment with more complex classifiers than logistic regression as the final classifier. We evaluate RF and MLP classifiers as the final classifier. Again, we tune hyper parameters via grid search and Table 3.8 and 3.9 show them. The performance of stacked classifiers over all stations are shown in Figure 3.31

Stacking NB with RF and MLP significantly improves performance over band-stacked RF and MLP. NB-RF variants achieve up to a 4-point macro-averaged f1-score gain over RF-stacked models. NB-MLP variants exhibit a substantial performance boost over the MLP-stacked models, making them competitive with the RF and HGBT stacked variants. However, for HGBT, there is a slight decline in the macro-averaged f1-score when comparing NB-HGBT against HGBT-stacked models. We will discuss these results in the next section.

Table 3.8: Hyperparameters for RF as the final classifier.

Hyperparameter	**RF**	**HGBT**	**MLP**
`max_depth`	12	12	12
`min_samples_leaf`	16	16	16
`min_samples_split`	16	16	16
`n_estimators`	100	150	100

Table 3.9: Hyperparameters for MLP as the final classifier.

Hyperparameter	**RF**	**HGBT**	**MLP**
`alpha`	0.1	0.1	0.1
`hidden_layer_sizes`	(20, 10)	(20, 10)	(20,)
`learning_rate_init`	0.001	0.001	0.0005
`max_iter`	200	200	200

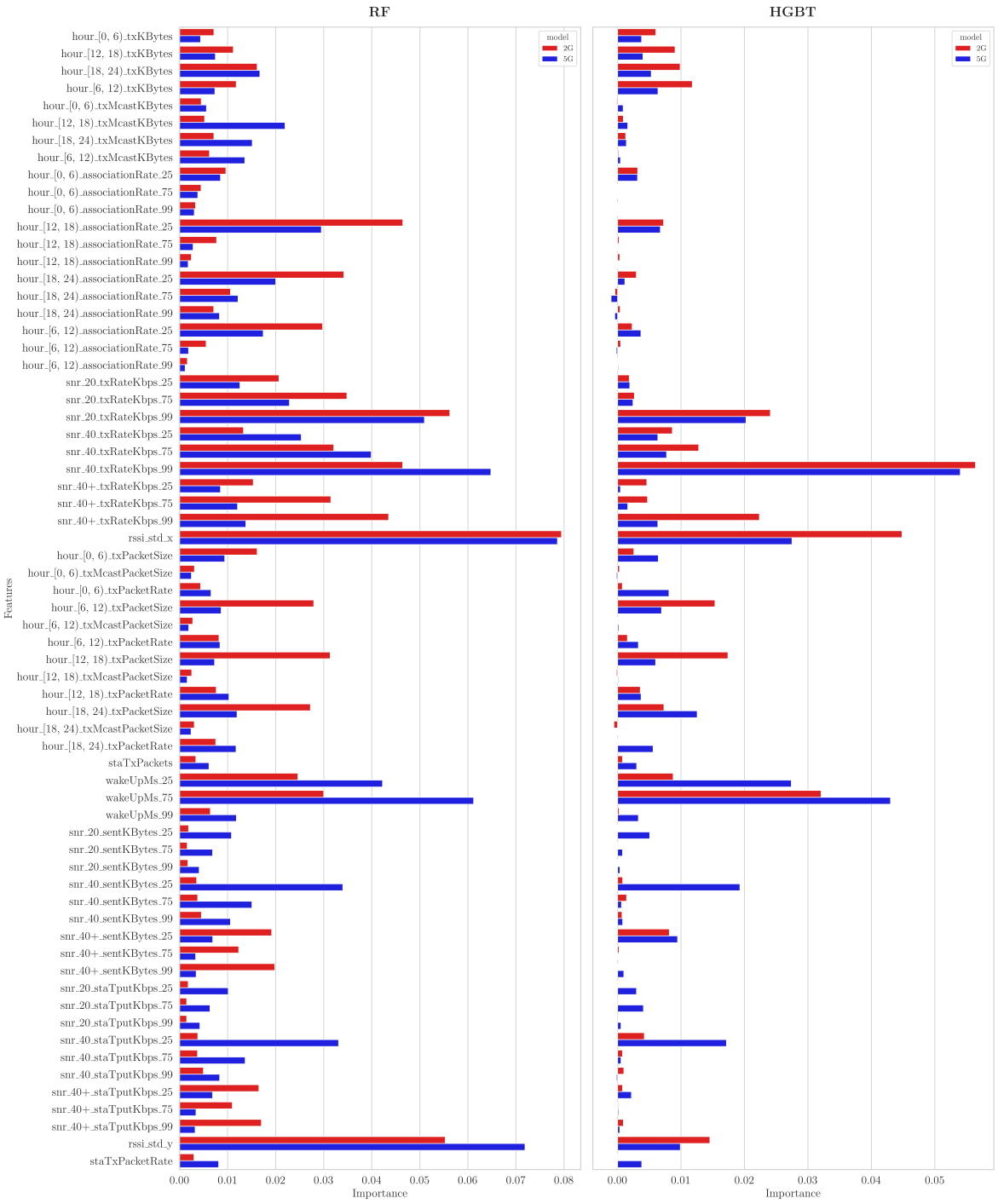

Figure 3.29: Stacked RF and HGBT feature importances.

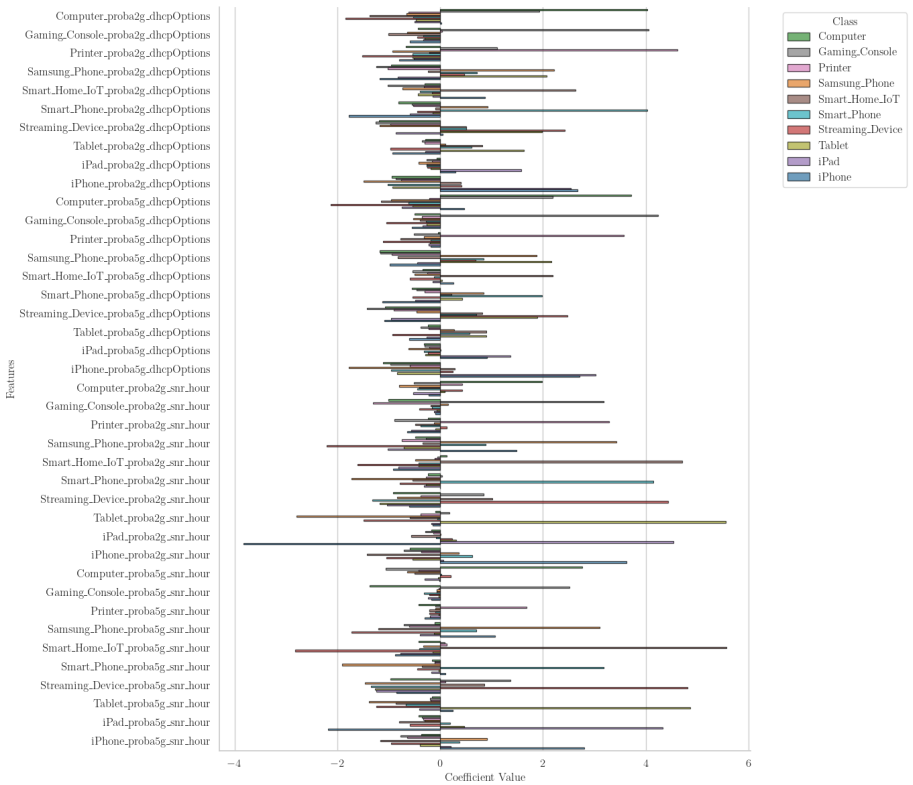

Figure 3.30: Logistic regression coefficients - NB-RF stacked.

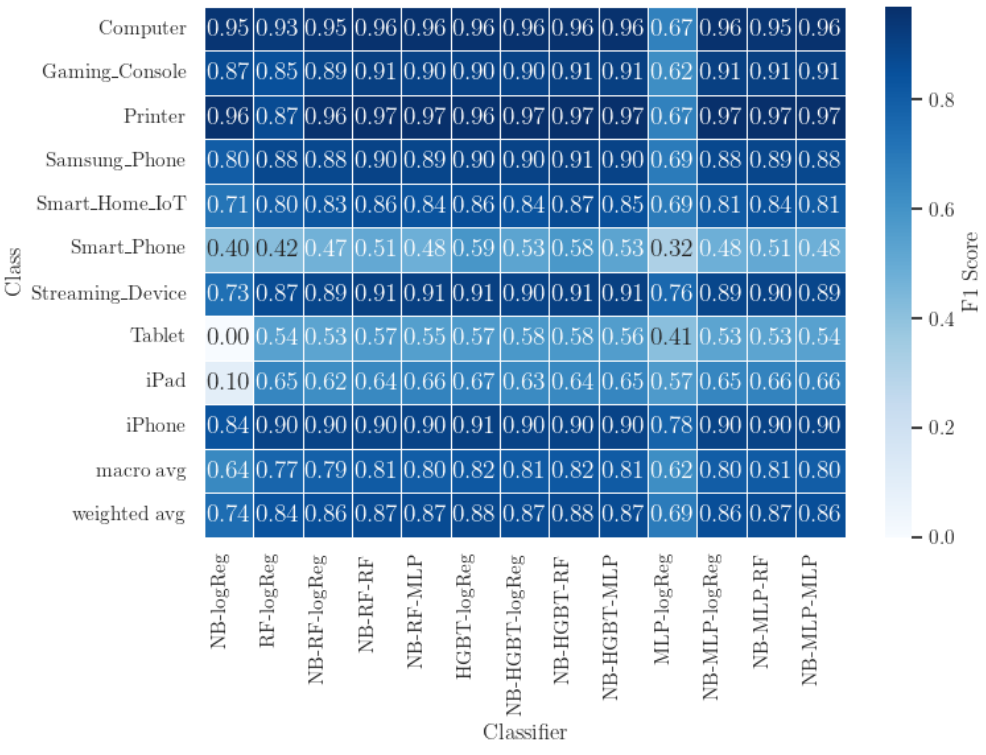

Figure 3.31: F1-scores of class and classifiers.

4. RESULTS

The results presented in Figure 3.27 demonstrate that band stacking generally enhances classifier performance, with notable improvements across most models. However, NB-stacked marginally underperforms NB-2G by one point in f1-score. Both RF-stacked and HGBT-stacked outperform NB classifiers family. In contrast, the MLP classifier struggles with one-hot-encoded `dhcpOptions`, performing worse than NB, underscoring its challenges in effectively handling this feature representation.

Figure 3.31 reveals further improvements through stacking NB with other classifiers. Stacking NB with RF and MLP significantly boosts classifier performance compared to band stacked models. NB-RF-RF architecture shows 3 points improvment in f1-score compared to RF-stacked. HGBT-stacked and NB-HGBT-RF have similar performances. The most remarkable gains are seen with MLP, where NB-MLP-RF outperforms MLP-stacked by nearly 20 points, achieving comparable performance to the best classifiers among HGBT and RF variants. Across all configurations, RF emerges as the most effective final-stage classifier, delivering consistent and superior performance.

Figure 3.31 shows that the highest classification performance is achieved by both HGBT-stacked and NB-HGBT-RF architectures. For each of these classifiers, the macro-averaged f1-score, which treats all classes equally regardless of their prevalence in the dataset, is 0.82, while the micro-averaged f1-score, which weights each class by its prevalence, is 0.88.

Stacking HGBT classifiers across bands forms the HGBT-stacked architecture, requiring a final-stage logistic regression classifier. Even with grid search for optimal regularization, runtime increases by 3%. The NB-HGBT-RF architecture adds an NB classifier alongside HGBT, but since NB trains significantly faster, it contributes only an 8% runtime increase, assuming no parallelization. Finally, training an RF classifier

with fixed hyperparameters adds 11%, bringing the total runtime overhead to 22%.

Among the individual station types, tablet and smart phone exhibit the lowest f1-score of 0.58, followed by iPad with 0.64. Aside from smart home iot, which attains an f1-score of 0.87, all other station types exceed 0.90 f1-score.

To show the individual performances of stacked classifiers over classes, both ROC curves and PR curves are helpful. The ROC curve is a graphical representation that illustrates the trade-off between TPR and FPR at various threshold settings.

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.1}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{4.2}$$

An ideal model has a ROC curve that approaches the top-left corner, indicating a high TPR and a low FPR. The PR curve, on the other hand, focuses on the trade-off between precision and recall, making it particularly useful for imbalanced datasets where one class significantly outweighs the others.

While ROC curves provide an overall evaluation of the classifier's performance across all decision thresholds, PR curves are more informative when assessing performance on rare or minority classes. In such cases, PR curves emphasize the classifier's ability to correctly predict the positive class, which is critical when the cost of false positives and false negatives is not symmetric. The ROC and precision recall curves of each class and classifiers are shown through Figures 4.1 - 4.10.

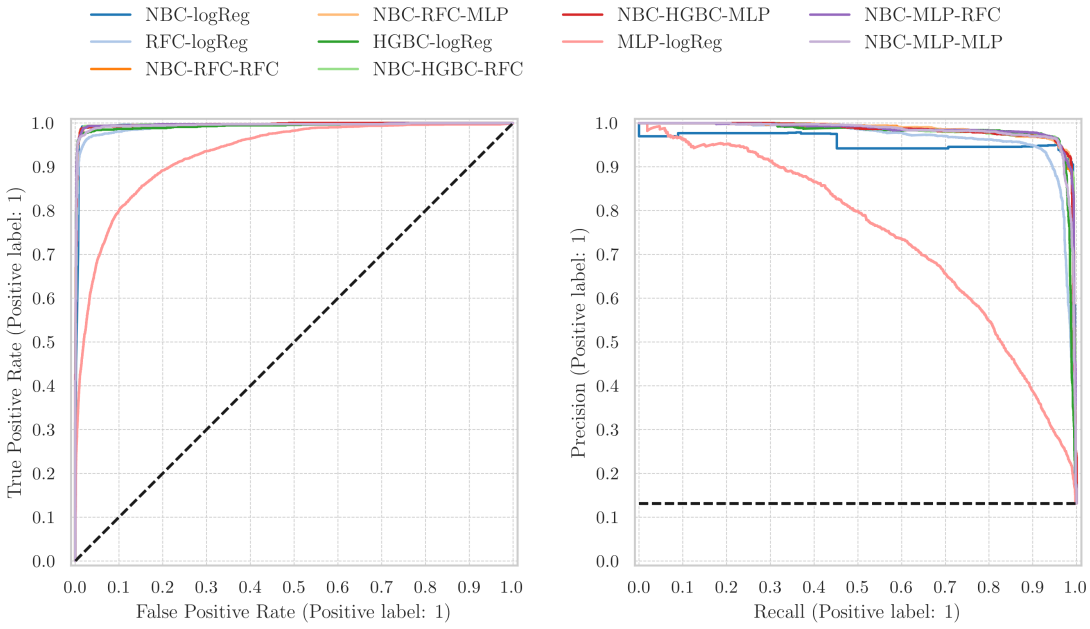

Figure 4.1: ROC and PR curves - computer.

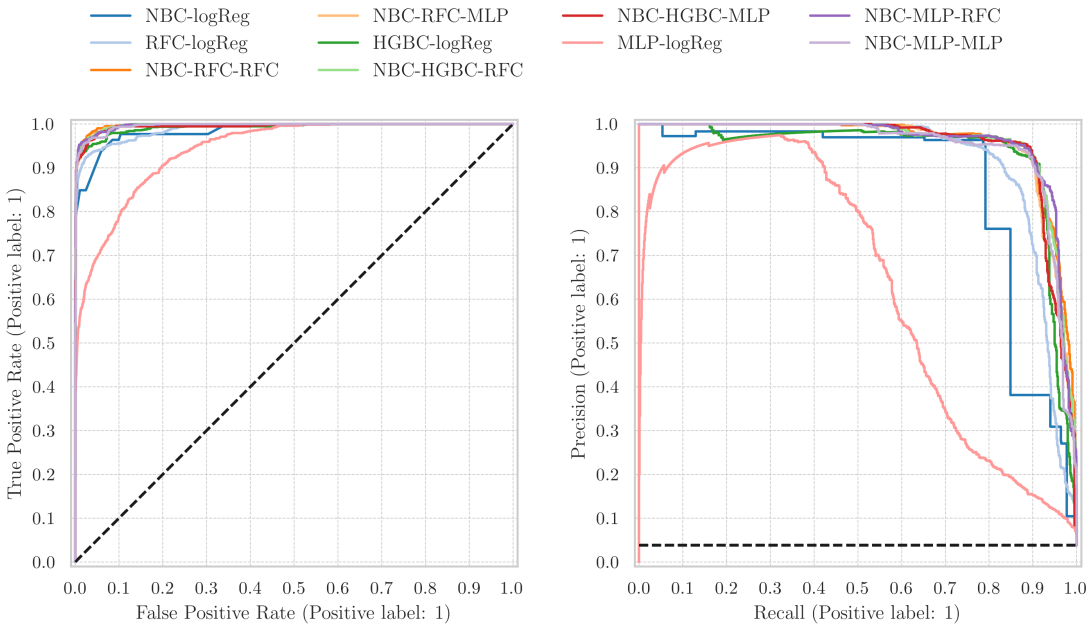

Figure 4.2: ROC and PR curves - gaming console.

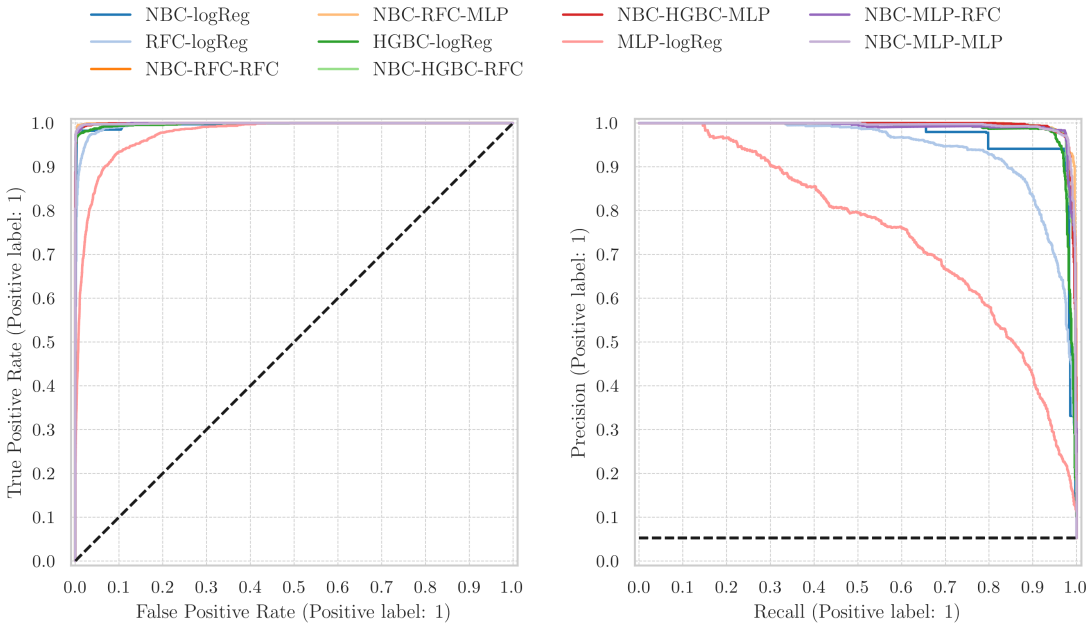

Figure 4.3: ROC and PR curves - printer.

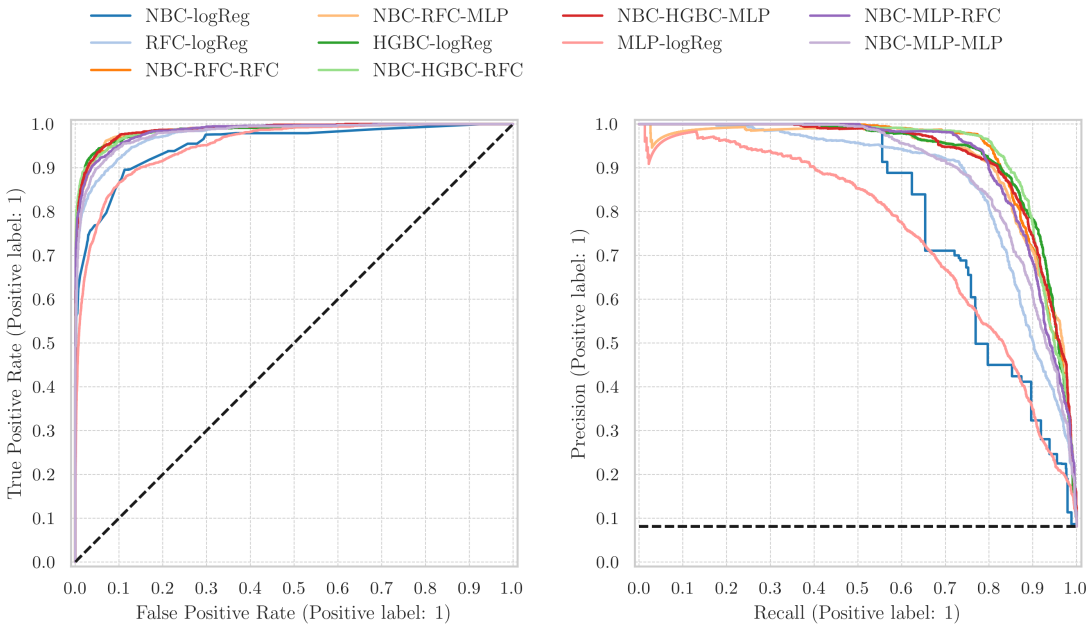

Figure 4.4: ROC and PR curves - smart home iot.

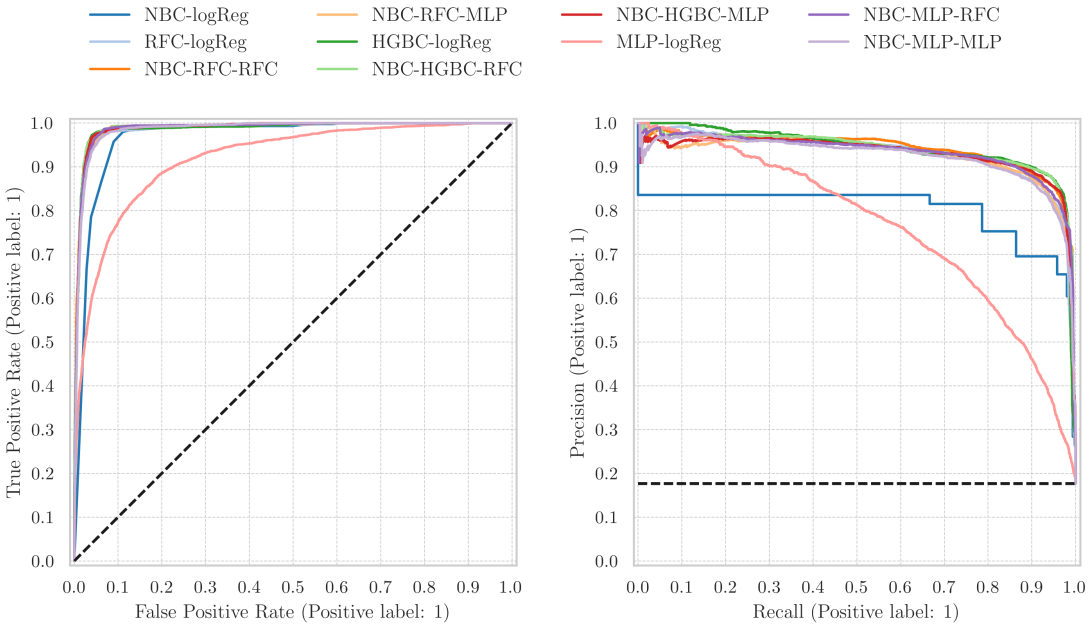

Figure 4.5: ROC and PR curves - samsung phone.

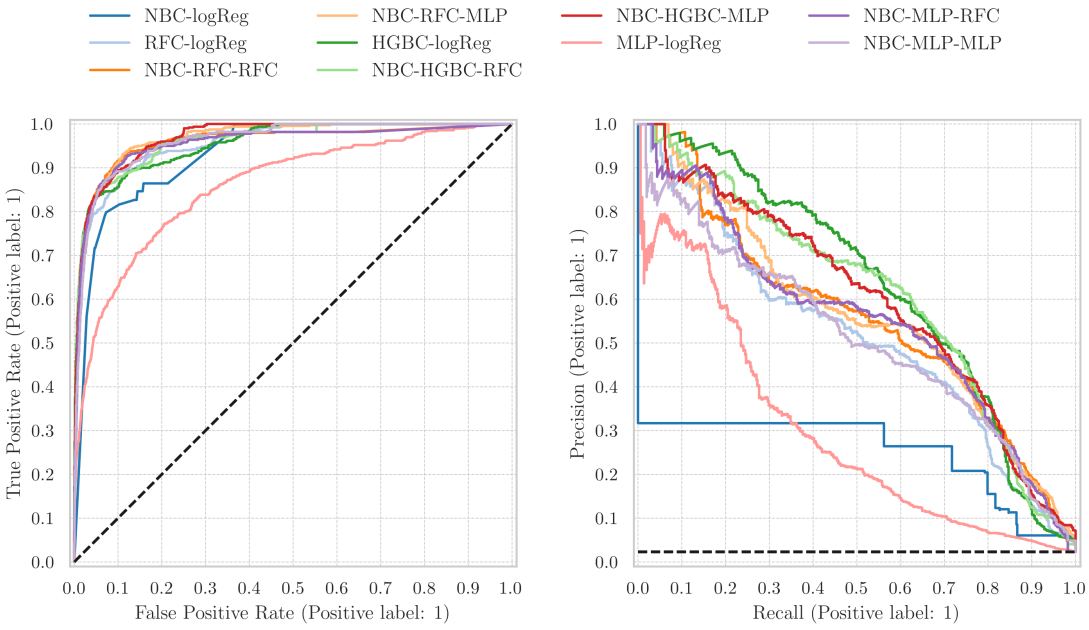

Figure 4.6: ROC and PR curves - smart phone.

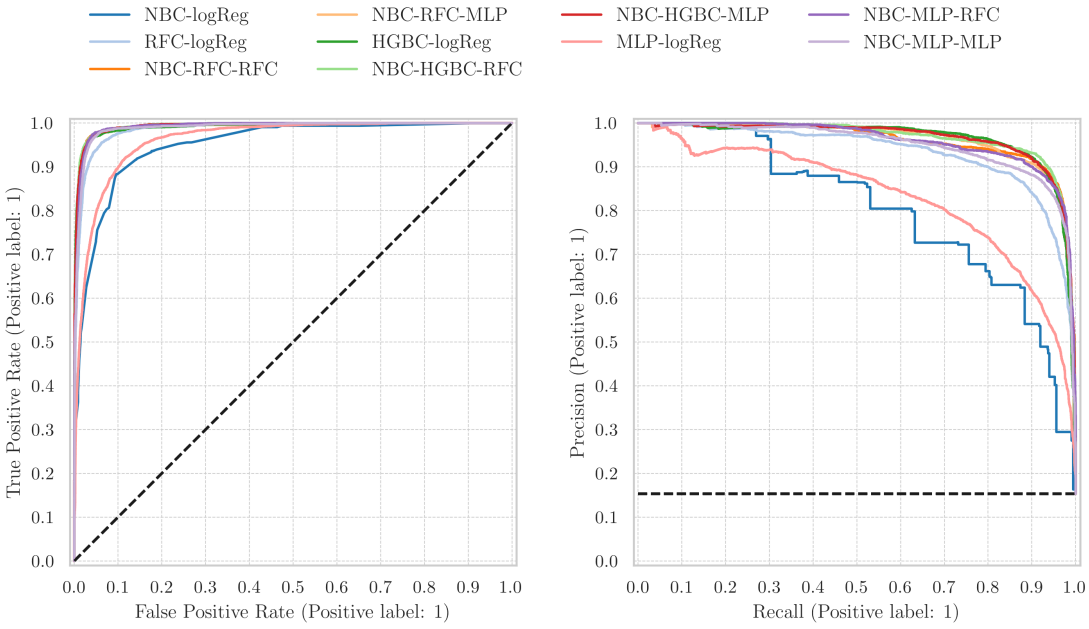

Figure 4.7: ROC and PR curves - streaming device.

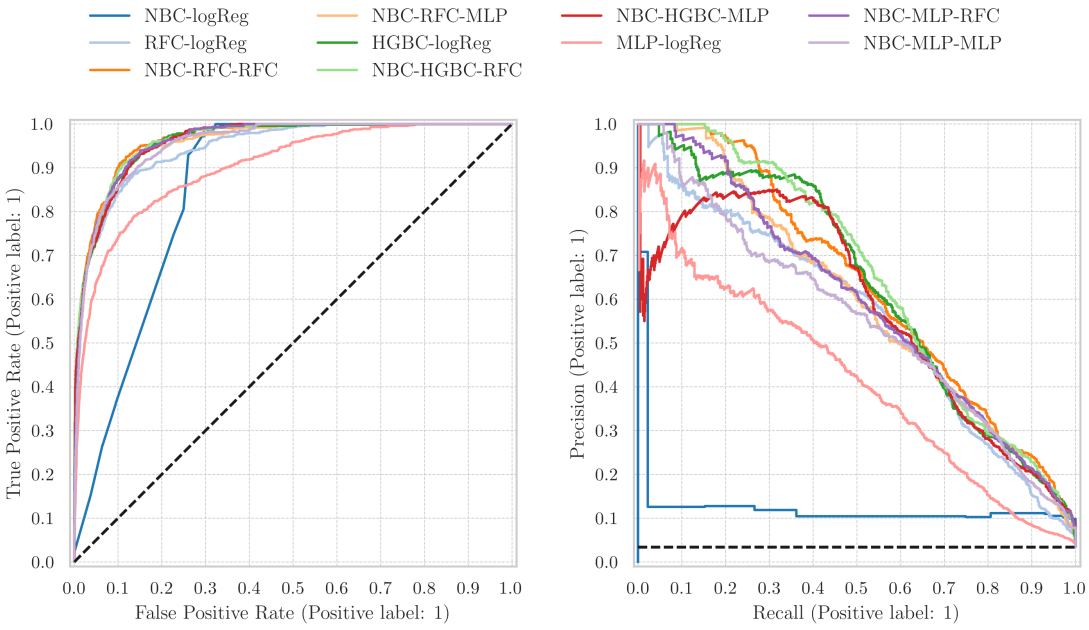

Figure 4.8: ROC and PR curves - tablet.

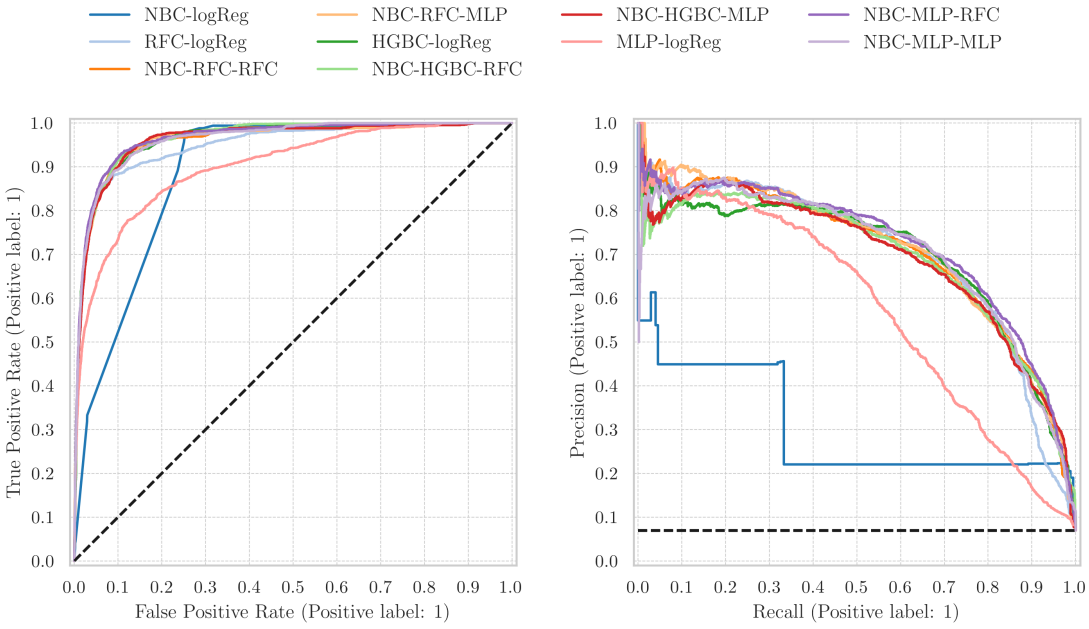

Figure 4.9: ROC and PR curves - ipad.

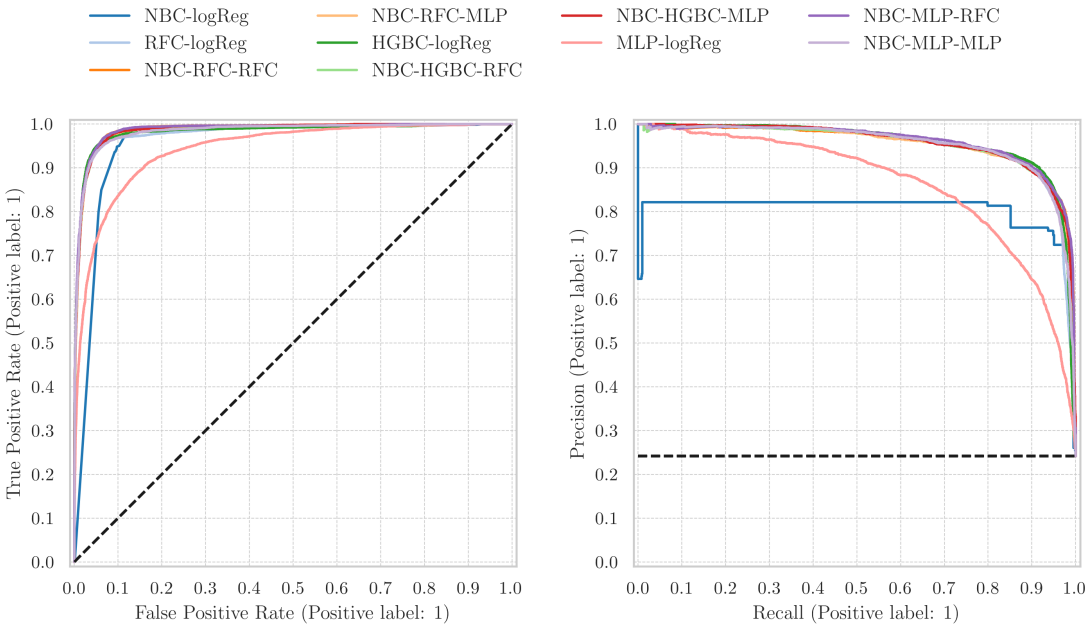

Figure 4.10: ROC and PR curves - iphone.

5. CONCLUSION

In this study, we explored various pipelines for addressing a 10-class station identification problem. The classifiers evaluated include NB, RF, HGBT and MLP, applied to both 2.4GHz and 5GHz bands. NB classifiers rely exclusively on label-encoded dhcpOptions features, while RF, HGBT and MLP utilize the broader feature set described in Section 3.1. Section 3.3.1 outlines the band stacking methodology, where classifiers trained independently on 2.4GHz and 5GHz bands are merged, with logistic regression chosen as the final-stage classifier. In Section 3.3.2, we extended this idea by stacking NB with RF, HGBT and MLP, leveraging different final-stage classifiers, including logistic regression, RF and MLP. This approach separated `dhcpOptions` from the feature set of RF, HGBT and MLP models, allowing their outputs to be combined with NB.

Our top-performing classifier architectures, HGBT-stacked and NB-HGBT-RF, achieved a macro-averaged f1-score of 0.82 and a micro-averaged f1-score of 0.88. The surveys [9] and [28] mention various studies reporting a wide range of accuracy and related metrics, spanning from 80% to 99%, though it is often unclear whether these values correspond to micro- or macro-averaged scores. Additionally, the station types used in these datasets differ, further complicating comparisons. Researches leveraging datasets based on physical layer characteristics, wireless signal features, or hardware imperfections generally report higher success rates. However, most datasets in these studies are private, making direct comparisons difficult. The nature of the dataset plays a crucial role in shaping the research approach, and variations in data sources significantly impact reported performance. Despite these challenges, our approach—built upon high-level physical layer features, network and traffic patterns, and protocol characteristics—achieves competitive performance in a 10-class station identification task without requiring any specialized data collection setup.

The results of this study underscore that leveraging device behavior, usage patterns and Wi-Fi features, in conjunction with protocol-level information, enables the development of effective classifiers for station identification. As security concerns grow and privacy measures increasingly limit access to traditional station identifiers such as `MAC` addresses, `dhcpOptions`, or SSDP information, the ablity of processing usage patterns, traffic metrics and signal-related attributes becomes paramount. We hope that this study provides a robust framework for station identification that aligns with future challenges in network management and security.

5.1. Future Work

While we obtain significant insights into station identification and classification, several areas for improvement and further exploration remain. One of the primary limitations is the imbalance in station class representation within the dataset. Classes such as ipad, tablet and smart phone have significantly fewer samples compared to others and acquiring more data for these underrepresented classes could enhance model performance and reduce bias. Additionally, the dataset could benefit from increased temporal coverage and a greater number of lines and stations, enabling a more comprehensive and robust analysis.

Another area for improvement involves addressing anomalies and outliers in the data. The presence of special days, known ISP issues and other irregularities may distort model training and evaluation. Developing preprocessing techniques to identify and remove these anomalies would help produce cleaner and more reliable data.

An alternative approach worth exploring is training class-specific classifiers. Instead of using a single classifier per band, a dedicated classifier for each station class per band could be developed. This approach has the potential to capture class-specific patterns more effectively and improve performance for classes with high inter-class overlap.

Finally, there is potential for further exploration in feature engineering. While this study utilized several derived features, the development of novel features or advanced engineering techniques could provide additional insights and improve classification accuracy.

REFERENCES

1. K. Shafique, B. A. Khawaja, F. Sabir, S. Qazi, and M. Mustaqim, "Internet of Things (IoT) for Next-Generation Smart Systems: A Review of Current Challenges, Future Trends, and Prospects for Emerging 5G-IoT Scenarios," *IEEE Access*, vol. 8, pp. 23 022–23 040, 2020.

2. Y. Sun and Others, "Internet of Things and Big Data Analytics for Smart and Connected Communities," *IEEE Access*, vol. 4, pp. 766–773, 2016.

3. Cisco, "Cisco annual internet report (2018–2023)," https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html, 2020, accessed on November 27, 2024.

4. H. Song, R. Srinivasan, T. Sookoor, and S. Jeschke, *Smart Cities: Foundations, Principles, and Applications*. New Jersey, USA: John Wiley & Sons, 2017.

5. Y. Zhang, L. Sun, H. Song, and X. Cao, "Ubiquitous WSN for Healthcare: Recent Advances and Future Prospects," *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 311–318, 2014.

6. Y. Sun and H. Song, *Secure and Trustworthy Transportation Cyber-physical Systems*. New York, USA: Springer, 2017.

7. J. Wurm, K. Hoang, O. Arias, A.-R. Sadeghi, and Y. Jin, "Security Analysis on Consumer and Industrial IoT Devices," in *Proceedings of the 2016 21st Asia and South Pacific Design Automation Conference*, pp. 519–524, Macao, Macao, 2016.

8. O. Shwartz, Y. Mathov, M. Bohadana, Y. Elovici, and Y. Oren, "Reverse Engineering IoT Devices: Effective Techniques and Methods," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4965–4976, 2018.

9. P. M. S. Sanchez, J. M. J. Valero, A. H. Celdran, G. Bovet, M. G. Perez, and G. M. Perez, "A Survey on Device Behavior Fingerprinting: Data Sources, Techniques, Application Scenarios, and Datasets," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1048–1077, 2021.

10. A. K. Jain, L. Hong, S. Pankanti, and R. Bolle, "An Identity-Authentication System Using Fingerprints," *Proceedings of the IEEE*, vol. 85, no. 9, pp. 1365–1388, 1997.

11. V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless Device Identification With Radiometric Signatures," in *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking*, pp. 116–127, San Francisco, California, USA, 2008.

12. J. Liu, F. R. Yu, C.-H. Lung, and H. Tang, "Optimal Combined Intrusion Detection and Biometric-Based Continuous Authentication in High Security Mobile Ad Hoc Networks," *IEEE Transactions on Wireless Communications*, vol. 8, no. 2, pp. 806–815, 2009.

13. V. Selis and A. Marshall, "A Classification-Based Algorithm to Detect Forged Embedded Machines in IoT Environments," *IEEE Systems Journal*, vol. 13, no. 1, pp. 389–399, 2018.

14. S. Barbhuiya, Z. Papazachos, P. Kilpatrick, and D. S. Nikolopoulos, "RADS: Real-Time Anomaly Detection System for Cloud Data Centres," *arXiv Preprint arXiv:1811.04481*, 2018.

15. S. Marchal, M. Miettinen, T. D. Nguyen, A.-R. Sadeghi, and N. Asokan, "AUDI: Toward Autonomous IoT Device-Type Identification Using Periodic Communication," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1402–1412, 2019.

16. M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma, "IoT Sentinel: Automated Device-Type Identification for Security Enforcement in IoT," in *Proceedings of the 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pp. 2177–2184, Atlanta, GA, USA, 2017.

17. K. Gao, C. Corbett, and R. Beyah, "A Passive Approach to Wireless Device Fingerprinting," in *Proceedings of the 2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)*, pp. 383–392, Atlanta, GA, USA, 2010.

18. N. T. Nguyen, G. Zheng, Z. Han, and R. Zheng, "Device Fingerprinting to Enhance Wireless Security Using Nonparametric Bayesian Method," in *Proceedings of the 2011 IEEE INFOCOM*, pp. 1404–1412, Shanghai, China, 2011.

19. N. Shone, Q. Shi, M. Merabti, and K. Kifayat, "Misbehaviour Monitoring on System-of-Systems Components," in *Proceedings of the 2013 International Conference on Risks and Security of Internet and Systems (CRiSIS)*, pp. 1–6, La Rochelle, France, 2013.

20. I. Hafeez, M. Antikainen, A. Y. Ding, and S. Tarkoma, "IoT-KEEPER: Detecting Malicious IoT Network Activity Using Online Traffic Analysis at the Edge," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 45–59, 2020.

21. K. Krishna and M. N. Murty, "Genetic K-Means Algorithm," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 29, no. 3, pp. 433–439, 1999.

22. T. N. Tran, K. Drab, and M. Daszykowski, "Revised DBSCAN Algorithm to Cluster Data With Dense Adjacent Clusters," *Chemometrics and Intelligent Laboratory Systems*, vol. 120, pp. 92–96, 2013.

23. G. Spanos, K. M. Giannoutakis, K. Votis, and D. Tzovaras, "Combining Statistical and Machine Learning Techniques in IoT Anomaly Detection for Smart Homes,"

in *Proceedings of the 2019 IEEE 24th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pp. 1–6, Limassol, Cyprus, 2019.

24. J. Pacheco and S. Hariri, "Anomaly Behavior Analysis for IoT Sensors," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 4, p. e3188, 2018.

25. R. Ferrando and P. Stacey, "Classification of Device Behaviour in Internet of Things Infrastructures: Towards Distinguishing the Abnormal from Security Threats," in *Proceedings of the 1st International Conference on Internet of Things and Machine Learning*, pp. 1–7, Liverpool, United Kingdom, 2017.

26. Q. Xu, R. Zheng, W. Saad, and Z. Han, "Device Fingerprinting in Wireless Networks: Challenges and Opportunities," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 94–104, 2015.

27. S. Sobot, V. Ninkovic, D. Vukobratovic, M. Pavlovic, and M. Radovanovic, "Machine Learning Methods for Device Identification Using Wireless Fingerprinting," in *Proceedings of the 2022 International Balkan Conference on Communications and Networking (BalkanCom)*, pp. 183–188, Sarajevo, Bosnia and Herzegovina, 2022.

28. Y. Liu, J. Wang, J. Li, S. Niu, and H. Song, "Machine Learning for the Detection and Identification of Internet of Things Devices: A Survey," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 298–320, 2021.

29. X. Wang, L. Gao, S. Mao, and S. Pandey, "CSI-Based Fingerprinting for Indoor Localization: A Deep Learning Approach," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 1, pp. 763–776, 2016.

30. Y. Zou, Y. Wang, S. Ye, K. Wu, and L. M. Ni, "TagFree: Passive Object Dif-

ferentiation via Physical Layer Radiometric Signatures," in *Proceedings of the 2017 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 237–246, Big Island, Hawaii, USA, 2017.

31. J. Hua, H. Sun, Z. Shen, Z. Qian, and S. Zhong, "Accurate and Efficient Wireless Device Fingerprinting Using Channel State Information," in *Proceedings of the IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pp. 1700–1708, Honolulu, HI, USA, 2018.

32. A. C. Polak, S. Dolatshahi, and D. L. Goeckel, "Identifying Wireless Users via Transmitter Imperfections," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 7, pp. 1469–1479, 2011.

33. S. Dolatshahi, A. Polak, and D. L. Goeckel, "Identification of Wireless Users via Power Amplifier Imperfections," in *Proceedings of the 2010 Conference Record of the Forty Fourth Asilomar Conference on Signals, Systems and Computers, Pacific Grove*, pp. 1553–1557, California, USA, 2010.

34. L. C. C. Desmond, C. C. Yuan, T. C. Pheng, and R. S. Lee, "Identifying Unique Devices Through Wireless Fingerprinting," in *Proceedings of the First ACM Conference on Wireless Network Security*, pp. 46–55, Alexandria, VA, USA, 2008.

35. C. Neumann, O. Heen, and S. Onno, "An Empirical Study of Passive 802.11 Device Fingerprinting," in *Proceedings of the 2012 32nd International Conference on Distributed Computing Systems Workshops*, pp. 593–602, Macau, China, 2012.

36. T. OConnor, R. Mohamed, M. Miettinen, W. Enck, B. Reaves, and A.-R. Sadeghi, "HomeSnitch: Behavior Transparency and Control for Smart Home IoT Devices," in *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, pp. 128–138, Miami, FL, USA, 2019.

37. M. R. Shahid, G. Blanc, Z. Zhang, and H. Debar, "IoT Devices Recognition

Through Network Traffic Analysis," in *Proceedings of the 2018 IEEE International Conference on Big Data (Big Data)*, pp. 5187–5192, Seattle, WA, USA, 2018.

38. M. E. Ahmed, J. B. Song, Z. Han, and D. Y. Suh, "Sensing-Transmission Edifice Using Bayesian Nonparametric Traffic Clustering in Cognitive Radio Networks," *IEEE Transactions on Mobile Computing*, vol. 13, no. 9, pp. 2141–2155, 2013.

39. W.-j. Hsu, D. Dutta, and A. Helmy, "Mining Behavioral Groups in Large Wireless LANs," in *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking*, pp. 338–341, Montréal, Québec, Canada, 2007.

40. M. Papadopouli, H. Shen, and M. Spanakis, "Characterizing the Duration and Association Patterns of Wireless Access in a Campus," in *Proceedings of the 11th European Wireless Conference 2005 - Next Generation Wireless and Mobile Communications and Services*, pp. 1–7, Berlin, Germany, 2005.

41. R. Perdisci, T. Papastergiou, O. Alrawi, and M. Antonakakis, "IoTFinder: Efficient Large-Scale Identification of IoT Devices via Passive DNS Traffic Analysis," in *Proceedings of the 2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 474–489, Genoa, Italy, 2020.

42. D. H. Hagos, A. Yazidi, Ø. Kure, and P. E. Engelstad, "A Machine-Learning-Based Tool for Passive OS Fingerprinting With TCP Variant as a Novel Feature," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3534–3553, 2020.

43. I. Vasilevski, D. Blazhevski, V. Pachovski, and I. Stojmenovska, "Five Years Later: How Effective is the MAC Randomization in Practice? The No-at-All Attack," in *Big Data Processing and Mining: 11th International Conference, ICT Innovations 2019*, pp. 52–64, Ohrid, North Macedonia, 2019.

44. D. H. Wolpert, "Stacked Generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.

45. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-Learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

46. GuolinKe, Qi Meng and Finley, Thomas and Wang, Taifeng and Chen, Wei and Ma, Weidong and Ye, Qiwei and Liu, Tie-Yan, "Lightgbm: A Highly Efficient Gradient Boosting Decision Tree," *Advances in Neural Information Processing Systems*, vol. 30, pp. 3146–3154, 2017.