NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION

MSc THESIS

Occupancy Detection in Indoor Environments Based on Wi-Fi Measurements and Machine Learning Methods

Fatih M. Koç

Supervisor (or supervisors): **Dimitris Syvridis**, Professor

ATHENS

SEPTEMBER 2022

MSc THESIS

Occupancy Detection in Indoor Environments Based on Wi-Fi Measurements and Machine Learning Methods

Fatih M. Koç
S.N.: 7115192100005

SUPERVISOR: **Dimitris Syvridis**, Professor

ABSTRACT

Mobile devices are currently significantly becoming part of our daily lives due to the wireless communication capabilities have enabled a series of high level services. These Wi-Fi equipment are continuously sending packets stated as probe requests that can be captured using wireless sniffers. In this thesis, we tried to solve the problem of exploiting such a methodology to complete occupancy estimation by considering how many people exist in a specific space. At first, we discussed collecting Wi-Fi probe request packets using the Raspberry Pi device and analysing them with packet analyzer tools. We operated data collection in different environments, ranges in different level densities and used the mobile camera as the ground truth value. Afterwards, we represented how we can use MAC addresses and power level information for indoor prediction in the proposed linear ridge regression model using different approaches. We introduced a cheap and precise occupancy estimation model based on the capture of Wi-Fi frames user's devices. The model is applied on low-cost hardware and utilized a supervised learning model to fit different environments. The experiments of such indoor estimations have been implemented in different scenarios to demonstrate the validity of the proposed solution and evaluate its results. The outcomes specify that mobile devices have good potential for predictin of the number of people in the space.

SUBJECT AREA: Occupancy Estimation

KEYWORDS: IEEE 802.11, Probe Request, MAC Address, Crowd Counting

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Co-funded by the
Erasmus+ Programme
of the European Union

AKNOWLEDGMENTS

This Master Thesis has been accomplished in the framework of the European Funded Project: **SMART Telecom and Sensing Networks (SMARTNET)** - Erasmus+ Programme Key Action 1: Erasmus Mundus Joint Master Degrees – Ref. Number 2017 – 2734/001 – 001, Project number - 586686-EPP-1-2017-1-UK-EPPKA1-JMD-MOB, coordinated by **Aston University,** and with the participation of **Télécom SudParis, member of IP Paris** and **National and Kapodistrian University of Athens.** The internship has been accomplished at the **Aalborg University, The Technical Faculty of IT and Design Department of Electronic Systems, Connectivity Section**.

This master thesis taught me that even if something appears simple, it contains a lot of thought and work. I believe that the experience gained during the previous semester was invaluable to me. I learned that nothing goes as planned and that there are always challenges along the way to success. Working alone requires complete responsibility for the consequences of my actions. My supervisors were always available to assist me when I felt lost. Working with them was enjoyable, and it was one of the factors that motivated me to do my best. I am pleased with the thesis's outcome, and the sense of accomplishment will stay with me for a long time.

September 20, 2022

1.INTRODUCTION

A crowd is described as a large number of people gathering for some reason, such as religious occasions, sports events, or political gatherings. Crowd counting's purpose is to count the number of objects, such as people or cars, that are part of a selected space. The purpose of this project is to design and develop a count of people in the spaces at a low cost. The crowd counting methods may be used for many other purposes as well, for instance, recurring events such as festivals or parades. The same approach could be helpful for estimating the boarding time delay in airports or understanding people's behavior while shopping at malls. In summary, estimating the distribution of people in a given place has become an effective tool for social and business intelligence studies. Nowadays, it is really important to count crowds for safety reasons and to create routes for crowds to exit different areas in case of an emergency. Crowd counting methods have many challenges, such as high mess-up and varying object density. These problems can increase prediction errors and reduce estimation. This work presents occupancy detection in indoor spaces that relies on Wi-Fi measurements using the machine learning (ML) method. It is important to find the number of people to improve the suitability when using services. For instance, there are many challenges to improving their services by counting people in lines, airports, stadiums, and shopping malls. It leads to preparing useful guidance in emergency situations. The current works have achieved such estimations with image recognition or Wi-Fi signal techniques [1]. Subsequently, Wi-Fi signal-based methods are less affected by light intensity or high density than image recognition. In this study, we focused on Wi-Fi signal-based techniques.

Many works about counting people based on sensors and cameras have been presented [2]. The Wi-Fi based methods have many advantages over sensors and cameras such as very cheap tools, low instalment costs, cleaning or maintenance is not required every day, with linking tools areas can be set up and controlled individually, compliant to EU data privacy regulations as all data is anonymous and independent from outside circumstances like humidity and temperature. Conversely, there are a lot of things to consider such that other signals that might be received from the outside venue, short visiting time to receive a ping from a certain phone, inappropriate timeouts and people will keep crowding the location.
Estimation of the number of people with Wi-Fi signal techniques based on the probe request (PR), with their unique MAC (Media Access Control) addresses [3]. Recently, the number of Wi-Fi embedded devices such as smartphones has increased. The Wi-Fi signal methods can easily estimate the number of people by counting the unique MAC addresses in PRs. Nonetheless, a single smartphone might have more than one MAC address when MAC address randomization is applied to the smartphone, which means an original MAC address is replaced by numerous bogus MAC addresses to avoid attackers from tracking people. This conclusion means a greater number of unique MAC addresses than the real number of people.

Hence, the final result must have the signal level of the ping. We need to pick up as many pings as possible by surfing through channels. We need a well-worked algorithm that must be applied to get the most accurate counting rate and the fastest reaction time without any mistakes. Crowd estimation is an important research topic in artificial intelligence applications as it provides an effective way for crowd control and management. It is difficult to satisfy the accuracy and speed requirements of engineering applications with the current methods. In this paper, we suggest counting crowds by an optimized ML technique.

2.STATE OF THE ART

2.1. Use Cases of the Wi-Fi Method

In this section, we will see the Wi-Fi method use cases. Prasertsung and Horanont (2017) used the Wi-Fi PR frames as a monitoring method to categorize the number of customers visiting a coffee shop [4]. They claim that the number of customers increases by an average of 30% on a promotion day this can be used to search how a promotion can push customers into stores. Shen (2018) et al. proposed a group detection system using Wi-Fi in the mall [5]. Experimental results show that this method could be capable of detecting over 90% of the groups with an accuracy of 91%. The particular case of people estimation on public transport such as buses is a difficult application because, unlike the estimation in a static place such as a shop, it is necessary to consider that the vehicle is in motion.

Although there are many examples of solutions that analyze Wi-Fi traffic to count people and track devices in the literature, most of them do not consider the MAC address randomization effect. Currently, since the diffusion of MAC address randomization adopted by manufacturers has increased, it is essential to present methods for the fact of randomization.

2.2. Crowd Counting Technique: Wi-Fi Probe Request Frames

There are many techniques to estimate the number of people indoors and outdoors, such as Wi-Fi, Bluetooth, other radio frequency (RF) technology, video, and audio recording. There were several methods that have been studied for this project in the literature. However, the approach chosen to solve this problem is to use the Wi-Fi probe request frames for estimating crowd density in different places of application. There are management frames to cope with the connection between (Access Point) APs and devices in the 802.11 standards. One of them is the PR frame. With PR frames, it is possible to estimate the number of smartphones in an environment without having installed any software on the device itself. Since PRs are not encrypted and can be captured and decoded with the help of wireless sniffers passively, without connecting to a network, it is preferred. It is possible to extract the MAC address of the device, the RSSI (Received Signal Strength Indicator), the SSID (Service Set IDentifier), the sequence counter, and the time with this type of frame. Occupancy of people is practical for businesses in providing better services while saving money. Our goal is to address the problem through a solution that is resistant to MAC addressing randomization strategies.

In this section, it is described how the system is designed for collecting Wi-Fi probes, as well as how we deal with anonymized the collected data, how we collect the data, and how we process it to obtain people's occupancy. The system consists of three elements: people, a wireless monitor (Wi-Fi monitor), and a data process [6]. For example, people have their own smartphones in an indoor environment, and it continually sends PRs that record randomized MAC addresses. The Wi-Fi monitor continuously collects the PRs and sends them to a storage. The collected probes are stored and we made them available through an interface to the processing service. The algorithm estimates the number of people within a crowded venue. It should be noted that some devices that are in neighbouring rooms or approach the monitored room for a few seconds may be detected during monitoring. It is necessary to design a mechanism that, filters out unwanted detections and counts only people who actually enter and remain in the monitored environment. The processing service makes the collected probes via the interface and

processes them. The goal of the processing is to filter probes from devices that are in an indoor environments.

2.3. Wi-Fi Probe Request Frames Implementations

It is possible to estimate the number of people in a certain place with lower costs with Wi-Fi PR frames. This method does not store errors over time, does not require a predefined path where users must be and can cover a large area. Moreover, it is possible to anonymize the MAC address to guarantee user privacy.
Handte et al. (2014) introduced one of the first methods to estimate crowd density by monitoring Wi-Fi PR frames. They modified the firmware of some existing APs and created a web service that allows the upload of the latest crowd density measurements. The system was able to always detect around 20% of the people on average since in 2014 there were fewer mobile devices than the present [6]. Yet, they didn't implement the method in current years.
Schmidt (2014) worked on social density estimation by exploiting captured Wi-Fi PR packets [7]. The method works by capturing Wi-Fi packets from smartphones using a dedicated scanning device: a Raspberry Pi with a battery and D-Link WiFi module. In order to avoid a false result that comes from neighbouring devices, the database of MAC addresses is created to filter out PRs. Four directions photos are taken every 10 minutes for validation. They claim that the method has good results however it is not suitable for high accuracy applications.
Yaik et al. introduced (2016) the correlation of crowd density with Wi-Fi PRs. The experiment was performed in the university during an open day event for 8 hours. As a ground truth checking, the authors used manual people counting using a counter at the entrance of the event. A Wi-Fi monitor is placed close to the entrance of the venue. The study shows a strong correlation coefficient of 0.893 [8].
Heitor et al. introduced (2018) a method without MAC randomization that trained classifiers to segment mobile from motionless appliances through its Wi-Fi performance pattern. Thus, they used data collected from various devices and in another environment and evaluated by using some ML algorithms. The best outcomes were with logistic regression achieving 0.99 (ROC area) according to their work [1].
Mikkelsen and Madsen (2016) presented a system to anonymize the MAC address of the sniffed PR frames. They put two thresholds to send them to a server and to analyze them: minimum value of the RSSI and minimum detection time. The ratio between the estimated number of devices is achieved at around 50% by setting the two thresholds. They claimed that it can give good results by using ML techniques [9].
Wang (2019) et al. used the random forest approach to find occupants using the Wi-Fi connection. The method was tested in an office with an occupancy of 22-27 people and a peak occupancy of 48-74 people and the RMSE (Root Mean Square Error) is four people on the test set. For more than 70% of estimations, the errors are within two people counts, and for more than 90% of estimations, the errors are within six people counts. Usage of ML techniques has a good impact on estimating the number of people but this approach does not have a communication system for the transmission of data in real time [10].

2.4. Related Works

MAC addresses can be used for beneficial activities such as crowdedness estimation, marketing, and risk maps. The MAC addresses randomization systems presented around

2014. It makes all standard MAC address based crowd monitoring systems count the same device more than once [11]. Thus, it is essential to create a new crowd monitoring system tolerant to MAC address randomization to estimate the number of devices accurately. There are many works regarding counting crowds Wi-Fi based however none of them explore the impact of MAC randomization. In fact, most applications focus on solving the device user mapping directly by using ground truth data and raw PR observations instead of trying to solve the fake devices and detection latency problems first. Nowadays, analyzing and defeating MAC randomization was studied by many recent papers. The operating systems (OS) for mobile devices have now implemented MAC randomization to protect user privacy before associating with the wireless APs.

Vanhoef et al. studied (2016) different methods for linking a single or multiple observed randomized MAC to its source device enabling location tracking and counting [12]. They use various packet fields which provide useful information about device identity. For example, the Wi-Fi protected setup (WPS) field of a PR can be applied to connect randomized PRs to their source device. To avoid such derandomization works, device manufacturers removed the WPS content field and increased sequence numbers.

Marco et al. (2020) presented an approach to find the limitations introduced by the randomization procedures that allow for extracting useful data for smart cities development [16]. They obtained the most relevant information elements within PRs and utilized clustering algorithms such as DBSCAN and OPTICS to find the exact number of devices. As a result, the accuracy of 65.2% and 91.3% using the DBSCAN and the OPTICS algorithms has been achieved in the experiment, respectively.

Matte et al. (2016) represented a method to fingerprint the PRs sent by one device. The device's fingerprint is computed using the (information equipment) IEs contained within the PRs [13]. Inter burst times were also included in this information which improved accuracy. The identification of the single fingerprint was calculated using the k-nearby neighbors (KNN) algorithm. Each fingerprint corresponds to one device. The results showed that counting the detected devices with an accuracy of 75%.

Franklin et al. (2006) introduced a system that analyzes the interframe time of packets allowing the making of a device driver fingerprint. Usage of timing as a feature could add errors in clustering and classification [14].

As seen in the studies usage of timing as the primary feature is problematic, we can discard inter burst and inter frame times but take into account the IEs content and how it varies according to the circumstance.

Another study has used fingerprinting also in the ISO/OSI (Open Systems Interconnection) physical layer (PHY). Brik et al. (2008) suggest a method that is able to identify the origin interface of an 802.11 frame by presenting a passive analysis of radio frequencies [15]. Particularly, ML tools are used to have an accuracy of 99% for device counting, but this approach is good only in a laboratory environment and it is useless if applied to a real world environment since the high radio interference and the necessity of a complicated setup to collect data.

Freudiger et.al. (2015) studied probing behaviour in various experimental settings [16]. He showed that probing frequency is depend on different factors like manufacturer, battery level, user interaction and number of stored SSIDs of AP. During the time of the experiments, MAC randomization was not widely established across manufacturers but they noted randomized PRs by an Apple IOS 8 device. Additionally, they studied the first possibilities for relating a randomized probe to its original device by using precise packet fields like SN or WPS information.

Ribeiro et al. worked on centring around eliminating MAC randomization used by most phones to anonymize the user [17]. They used an RSSI based approach in combination with trilateration to locate smartphones. Based on the location estimated, they managed to find, whether a new MAC address seen matches a new device or is just a new MAC

address of a device doing MAC randomization for privacy purposes. They claimed that this RSSI based localization approach is accurate to decide whether a new faced address in the same area corresponds to a new device or the already seen one.
Yuyi et. al. (2021) represented Vision and TrueSight, two new crowd monitoring algorithms that estimate the number of devices based on MAC address crowd monitoring [18]. As well as PRs, Vision uses data packets and beacon packets to moderate the effect of randomization. Besides, TrueSight uses SNs and clustering to estimate the number of devices. As a result of this study, Vision can pick 440 randomly generated MAC addresses into one group and count only once without installation of software and TrueSight can estimate the number of devices with an 75% accuracy.

3.FUNDAMENTALS

3.1. Background of Wi-Fi Theory: Wi-Fi, MAC Structure and Probe Request

3.1.1. IEEE 802.11 (Wi-Fi)

The Institute of Electrical and Electronics Engineers (IEEE) has developed 802 sets of standards, which specify the requirements and recommendations for networking (IEEE). Relevant standards contain Ethernet, LAN (Local Area Networks) and WLAN (Wireless Local Area Networks). It describes a set of the PHY and MAC protocols being used to perform computer communication in WLAN. The IEEE 802.11 standard series defines the protocols that account for WLAN and is a subset of the IEEE 802 collection of LAN technical standards. The standard is constantly developing in order to improve performance and reliability [19]. The Wi-Fi alliance uses the IEEE 802.11 standard set to approve equipment as Wi-Fi. As a consequence, all Wi-Fi registered devices follow the 802.11 specifications.

When data or information is transferred between several locations without the use of an optical or electrical conductor that serves as a medium for the transmission, this is referred to as wireless communication. The utilization of electromagnetic radiation, sometimes known as radio waves (RW), is the technique for wireless communication that is most frequently utilized. Different frequency bands have been created for these RWs. The frequency bands utilized for Wi-Fi are the super high frequency (SHF) band, which varies from 3 to 30 GHz, and the ultrahigh frequency (UHF) band, which ranges 300 MHz to 3 GHz. IEEE 802.11 operates at a variety of frequencies, including the 2.4 GHz, 5 GHz, 6 GHz, and 60 GHz bands.
In order to reduce interference frequency bands are splited into overlapping channels. In the 2.4 GHz band, there are 14 channels with a spacing of 5 MHz except channel 14, which has a space of 12 MHz as seen in the figure 1. In Europe, channels 1 to 13 are frequently used. The 2.4 GHz band has 20 MHz wide channels. The full spectrum is only 100 MHz wide and the channel centers are 5 MHz apart. As an outcome, the 11 channels must eventually overlapping in to fit inside the 100 MHz available. There are numerous frequency methods used by each standard [20].

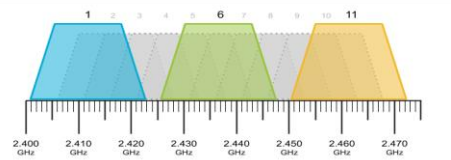


Figure 1. Non-Overlapping Channels

On the 2.4 GHz band, nonetheless, channels 1, 6, and 11 are separated enough from one another so that they have enough space between their channel centers and do not overlap. Channels 1, 6, or 11 will need you to share the channels with other networks (co-channel interference) but this is still a much better choice than having to cope with adjacent channel interference which affects all other channels.
When a device looks for a nearby network, PRs are delivered in bursts across several channels consecutively. The number of probes sent out varies depending on a number of variables, including the OS, screen condition, and the manufacturer of the device. When the battery is running low, a device makes every effort to conserve power and lowers the frequency of PRs [21]. Considering all of these limitations, the Wi-Fi sniffer

system needs to be capable of efficiently cover multiple channels in order to simultaneously collect more frames. It is certain that increasing sniffer density surely provide more data.
Before joining any network, clients, stations or mobile stations must first detect it. Simply putting the cable into the wired connection will locate the network. Before the joining process can start using a wireless connection, the compatible network must be identified. This network discovering is referred as scanning. Finding an appropriate AP for the client to connect to now or in the future is the goal of client scanning.
Massive amount of information are carried in each Wi-Fi packet that is sent between a mobile device and a wireless AP. That offers new opportunities to learn location information and mobility behaviour related to mobile users using existed Wi-Fi infrastructure. Each radio in a wireless AP is constantly searching for new RF transmitters. While 802.11b/g/n radios operate between 2.4 and 2.4835 GHz, 802.11a radios operate between 5.15 and 5.85 GHz.
Passive scanning and active scanning are the two types of scanning. The regulatory domain chosen during the initial deployment of the AP is the country of operation and radios by default carry out both sorts of scans on all channels permitted by that authority. While both types of scanning are enabled by default, active scanning is only carried out on channels where it is permitted to transmit by local laws. Active scanning is not permitted on channels that need radar detection with dynamic frequency selection (DFS) or that are not permitted for unlicensed usage.

APs serve as a means of communication between mobile stations and other networked devices. A mobile station must be in the appropriate connection state before it can broadcast traffic through an AP.
The three 802.11 connection states are:

- Not authenticated or associated.
- Authenticated but not yet associated.
- Authenticated and associated.

Before bridging to take place, a mobile station needs to be in an authenticated and associated condition. To achieve an authenticated and associated state, the mobile station and AP will transfer many 802.11 management frames. The 802.11 association process starts with PR from mobile station (MS) and then the AP respond with probe response, the MS will try to authenticate to the AP depending on the authentication used and the AP can ask for a second round of authentication from the MS. When the authentication is done MS will send an association request to the AP, the AP will then respond with an association response packet. Once this is done data transmission finally take place figure 2.

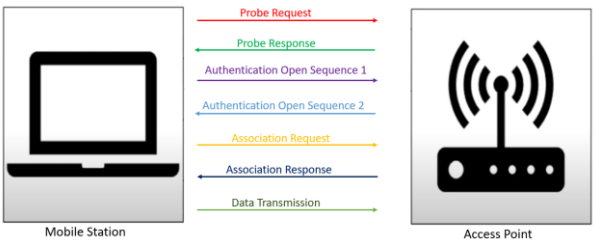


Figure 2. 802.11 Association Process

The two options for channel sniffing that are most frequently discussed in the literature are fixed single channel monitoring and channel hopping, which involves rapid switching between channels at regular intervals. Fixed channel monitoring captures more packets

than channel hopping according to a test campaign that specifically used three non-overlapping channels [21]. The wireless adapter's limited ability to capture on more than one channel at once explains the circumstance. Channels 1, 6, and 11 may be the channel on which it is preferable to sniff. Although channel 1 is typically preferred for sniffing other research have declared that no in-depth information of statistics on which channel is the most used is available [22]. While it is expected that the test results won't be much impacted by the channel selection.

3.1.2. 802.11 MAC

The 802.11 specifications were developed to establish wireless connections by utilizing the wired networking standards that had already been established. It is improving to satisfy the requirements of wireless data transport. As a result, the MAC for WLAN has been adjusted to consider the wireless transmission.
Each Wi-Fi device has a MAC address that only identifies itself in the local network which is created of six octets of bits. The first three octets are named by the IEEE to the device manufacturer and establish the Organization Unique Identifier (OUI). The other three octets are called Network Interface Controller (NIC) and are named by the manufacturer [24] as depicted in the figure 3.

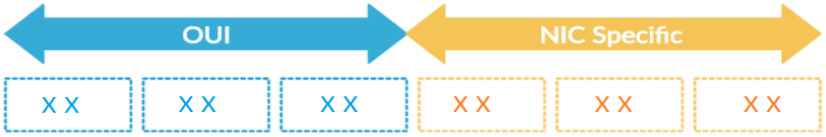


Figure 3. MAC Address Format

In Wi-Fi applications, if the second least bit of the first octet is adjusted to 0, then the MAC address should be globally unique. Else, when this bit is adjusted to 1, the MAC address should be locally administered as a result the MAC address is randomly created.

X2:XX:XX:XX:XX:XX

X6:XX:XX:XX:XX:XX

XA:XX:XX:XX:XX:XX

XE:XX:XX:XX:XX:XX

Figure 4. Locally Administered MAC Addresses

In other word, any device that produces MAC addresses at random must set the locally managed bit. As shown in the figure 4 there are four different locally administered address that can be used, where ‘X’ can be any hex value.
The figure 5 shows how the sniffer examines the address fields in a management frame utilizing wireshark.

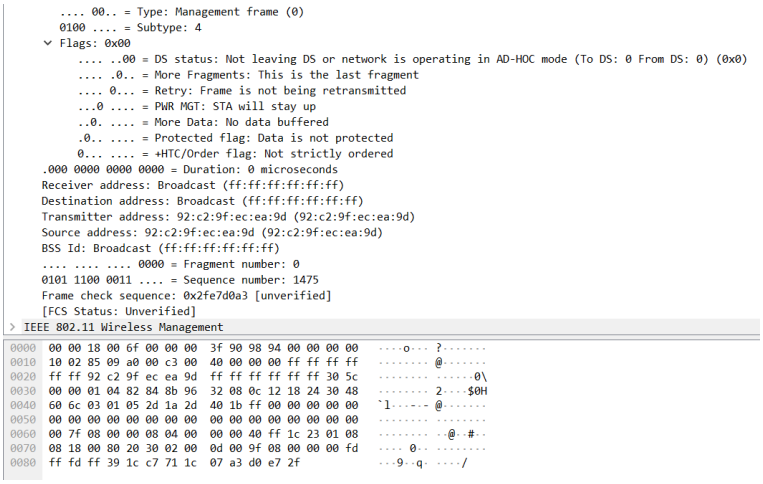


Figure 5. Address Fields in Wireshark

3.1.3. Probe Requests

PRs are signals that are constantly broadcast by Wi-Fi enabled devices like smartphones, laptops, and tablets. When a Wi-Fi client tries to connect to a Wi-Fi network, the first technique is to scan for beacon frames, which are frames broadcast by Wi-Fi routers to inform Wi-Fi clients of their presence. The second method is to send PRs, which include the device's unique MAC address as well as its type, manufacturer, and model. Because a Wi-Fi client can connect to a Wi-Fi router without waiting for a beacon frame from the router, using probe requests is preferable.
WLAN clients or stations use PR frames to scan the area for the presence of a WLAN network. PRs are made up of two components: the SSID to be probed and the supported rates of the device transferring the PR. Stations receiving these PR frames must decide whether or not the PR transmitters can join the network. To have successful PRs, stations transmitting them should have rates supported by the network which they wish to join. As a result, the network's SSID should be included in the PR frame.
Freudiger [21] performed a thorough experiment of how various factors, such as monitor channel configurations, the number of SSIDs stored in the PNL, and device configurations, influence Wi-Fi PRs. It has been illustrated that three antennas, each set to a fixed nonoverlapping channel, collect the most probes.

3.2. Monitor Mode

WNICs (Wireless Network Interface Controllers) can operate in different modes: managed mode, promiscuous mode and monitor mode. In this work we are focusing the monitor mode that is a listening mode that only exists for wireless adapters.

Monitor mode, also known as RFMON (Radio Frequency Monitor) mode, is a defined mode for wireless cards in the IEEE 802.11 standards [23]. Whenever a wireless card is setted to monitor mode, it can’t connect to other networks because it is monitoring all networks around it, so we can't utilize the interface for usual networking. We can use a wireless card to monitor networks around us or for standard networking tasks like web browsing, but not both at the same time. Using a supplemental network card, one for standard networking and one for monitoring is one way around this limitation. To sniff traffic in a wireless network, we must first set the wireless adapter to monitor mode. In theory, we also could set it to promiscuous mode to achieve the same result, but this is not always the case. Monitor mode is a wireless interface operation mode in which the interface passively collects information sent wirelessly.

3.3. Machine Learning

An ANN first experiences a training phase during which it learns to identify patterns in data. The network compares its actual output to what it was supposed to create during this supervised phase in order to produce the desired output. The backpropagation method is used to adjust the difference between the two results. This means that the network adjusts the weight of its connections between the units backward, from the output unit to the input unit, until the difference between the actual and expected outcome creates the least amount of error [25].
They are different in terms of how the models have been trained and the quality of the necessary training data. Each technique has particular advantages, during the learning stage of the ML lifecycle, supervised ML requires labelled input and output data. Once the model has figured out how the input and output data are related, it may be used to categorize previously unexplored datasets and estimate results. The requirement for labelled training data differentiates supervised learning from unsupervised learning. Unsupervised machine learning processes unlabeled or raw data, whereas supervised machine learning utilizes labelled input and output training data.
In our circumstance, we want to use ML to estimate or predict the number of people in an area based on the Wi-Fi packets we collect over time. There are two types of supervised machine learning algorithms such as regression and classification. While the classification predicts discrete outputs, regression predicts continuous value outputs. Classification identifies input data as part of a learned group such as determining if a tumour is benign or malignant. Regression predicts outcomes from continuously changing data such as estimating the price of a house in euros. Linear regression finds a straight line that represents the relationship. Thus, in this section, the linear regression algorithm is demonstrated and selected for occupancy estimation.

3.3.1. Linear Regression Definition & Working Principle

The goal of the linear regression is to obtain the best fit line that can accurately predict the output for the continuous dependent variable. Simple linear regression is used when only one independent variable is utilized to make a prediction and multiple linear regression is used when there are more than two independent variables. The algorithm establishes the relationship between the dependent variable and the independent variable by locating the best fit line. Additionally, the relationship must be linear.

A linear regression algorithm is used to study relationships between two continuous variables first is the independent variable (x) also referred to as the predictor and the second is the dependent variable (y) also referred to as an outcome. The straight line equation can be used to fit a line. The equation gives the output variable based on the input variable and slope of the line. The equation 1 below can be used to identify the line.

$$y= b+\theta_1 X+e \qquad (1)$$

In the above equation, y is the dependent output variable, X is the independent input variable, b is point at which the line meets the y-axis also called as an (intercept) bias and θ_1 is the weights (slope of the line).

The best fit line as the line that minimizes the sum of squared errors (SSE) or mean squared error (MSE) between our target variable (y) and our predicted output is used to determine the relationship between the dependent variable and one or more independent

variables. In Linear Ridge Regression with Ordinary Least Squares (OLS), the main purpose of the best fit line is that our predicted values should be closer to our actual value. In other words, we minimize the error (least square cost function) which is the difference between the predicted and the observed values. These errors are also called as residuals. The residuals are specified by the vertical lines displaying the difference between the actual and estimated value. We first optimize this cost function to determine the weight parameters. Either the gradient descent (GD) algorithm or the (OLS) approach can be used for this process. These methods will allow us to determine the parameter weights for our model [26].

3.3.2. Different Approaches to Ridge Regression

We can implement a ridge regression model by using the following methods:

1. Solving model parameters (closed form equations) [27]
2. Usage of optimization algorithm (gradient descent, stochastic gradient, etc.) [28]

If computing matrix inverse is not a concern, the closed form solution may be preferred for smaller datasets. The GD or SGD approaches are preferred for very large datasets or datasets where the inverse of XTX might not even exist for example the matrix is non-invertible as in the case of perfect multicollinearity.

When using OLS, an analytical solution or a closed form solution with an exact answer can be obtained. The reason for using GD rather than OLS is in simple statistical models with small data sizes and a small number of independent variables. Furthermore, when the dependent and independent variables have a nonlinear relationship, a simple statistical model is inadequate to solve the issue.

If there are more features and a larger data set, OLS is especially difficult to implement. Since we need matrix inversion in that case, which requires more computational power. The magnitude of the available features will influence computation speed. The solution will be challenging to determine. The GD is considerably faster than the OLS. GD is typically used in conjunction with regularization-based models (L2-L1). Because there are no closed-form solutions for nonlinear complicated issues, OLS does not perform well in this situation. Another requirement for OLS is that the number of data points exceed the number of features. As GD does not have this restriction, it is decided to build model with GD and compared with other models. Instead of solving the normal equation, we can differentiate the ridge cost function to achieve its partial derivatives and then use GD to iteratively approach the optimal solution however we also implemented close form solutions.

Normal equations are a method of solving the quadratic equation for the weights that decrease the MSE directly. However, solving normal equations require inverting a N x N features matrix, which is an O($\mathrm{N\ features}^3$) operation. Therefore, when N features exceeds 1000, it begins to slow down, whereas GD is O(N). For that reason we wanted to implement directly GD algorithm in order to be not have problem of close for equation method. On the other hand in the experiment we had compare the result of the close form solutions as well.

GD is an optimization technique for minimizing a function's value. Mostly in ML, we typically define some cost function J(w) that informs us how well the model fits our data

and $w= [w_1 \dots w_n]$ are the model parameters that we want to change, such as the coefficients in a linear regression problem ($y=w_0 + w_1 x$).
After the weight updates and assign of the learning rate, which indicates how quickly we update our model parameters or how large steps we take when changing the values of the model's parameters, as shown in the paraboral curve in figure 6. The algorithm iterates until convergence is reached or when the gradient is so small that weights do not change.

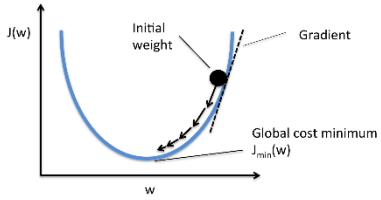


Figure 6. Weights Update by Learning Rate in Paraboral Curve

In the above scenario, we only have one parameter 'w' and our goal is to reduce the cost function J(w). The intuition is that the gradient's sign indicates the direction we must move to minimize J.
To solve a specific derivative of loss with respect to a specific weight, GD considers all of the data points in the data set. That is, at each epoch, we take all records and estimate the y, and backpropagation updates the weight. The function that will be applied is known as GD. When we use the GD optimizer, we take into account all of the data points in our dataset. When considering all data points, the weights converge quickly since it considers all data points. GD is classified into three types. We will discuss three of them in this section since we applied all of them.

1.Gradient Descent (GD) is also referred to as batch gradient descent. It requires computing the gradient by using entire dataset or training set to determine the best solution. The objective is to achieve the optimal solution which could be the local or global optimal solution. Utilizing derivative of the model to update a parameter in a specific iteration requires moving all of the samples in our training set each time to generate a single update. However, when we required to run through large numbers of samples, this could be a significant challenge. Since a GD requires moving through the whole data set during each iteration that expends a significant amount of time and computational power [25].

2.Stochastic Gradient Descent (SGD) is an optimization algorithm variant that saves time while searching for the best optimal solution. In SGD, the dataset is shuffled properly to avoid pre-existing orders before splitted into m examples. As a result, the SGD algorithm can randomly select one example from the dataset per iteration instead of going through the entire dataset at once. This random estimation of the data set eliminates the computational cost linked with GD while achieving faster iteration and a lower convergence rate. The process simply takes one random SGD sample and iterates it. SGD makes use of this concept to accelerate the GD process. In most circumstances, the close estimation that SGD provides for parameter values is sufficient since they reach the ideal values and continue to oscillate there. However, since it takes and iterates one example at a time, it produces more noise than we would prefer. Although using the entire dataset is extremely useful for locating global minimum in a less noisy or random way, the problem occurs when datasets become extremely large. In SGD, since only one sample from the dataset is chosen at random for each iteration, the path taken by the algorithm to reach the minimum is usually noisier than your typical GD algorithm. However is not a issue that much since the path taken by the algorithm does not matter as long as we get the minimum and with a importantly shorter training time.

3. Averaged Stochastic Gradient Decent (ASGD) is to do regular stochastic gradient descent $\theta_i^{n+1} = \theta_i^n - \alpha \frac{\partial J\theta_i^n}{\partial \theta_i}$ however then choose the mean as the final solution. In the case that GD is working well $\bar{\theta}_i = \frac{1}{N} \sum_{t=1}^{N} \theta_i$ then this approach will converge to the optimal solution $\lim_{N\to\infty} \bar{\theta}_i = \theta_{opt}$. Averaging is applied to decrease the impact of noise. In practice, GD may approach the optimal but not actually converge to it, instead oscillating it around. Averaging the results of SGD will offer a solution that is more likely to be close to the optimal in this case.

3.4. Model evaluation of Ridge Regression

To determine how good a model is, we must first define evaluation. This is almost always the root mean square error in linear regression (RMSE). This is simply the root of the sum of square errors between our prediction and the true observation. Using our model parameter for the dataset, we will predict the value for the target variable. The predicted value is then compared to the actual value in the dataset. Using a formula in equation 7, we compute the cost function. Using a cost function, we can assess the accuracy of our estimated function. The built model aims to minimize this loss function or error in order to bring the prediction value closer to the actual value.
Ridge regression decrease estimation variance. The ridge regression has the potential to improve predictive accuracy. The estimations become more stable and accurate. Ridge regression is also better than linear regression at handling nonlinear relationships between predictor and outcome variables.
Ridge regression is superior to linear regression in several ways. First, it is less sensitive to collinearity than linear regression. Second, It can be used even when the data contains outliers. Third, it does not need perfectly normalized data. Finally, even when the number of variables exceeds the number of observations, ridge regression can be used. However, ridge regression has some drawbacks. First, if the data set is large, it can be computationally costly. Second, since the Ridge term (L2 norm) modifies the coefficients, the results of ridge regression can be difficult to interpret. This is because the cost function contains a quadratic term (non-linear), which makes optimization more difficult. This can make defining the model's results difficult. Finally, the L2 norm is sensitive to outliers and can produce unstable results in the presence of outliers in the data.
The first step in learning a neural network is to define a cost function also known as a loss function that measures how well the network predicts outputs on the test set. The next step is to find a set of weights and biases that minimizes the cost. The mean squared error (MSE) is a commonly used function that measures the difference between the actual and estimated values of prediction. The coefficient of determination, or R2, is a metric that indicates the goodness of fit of a model. It is a statistical measure of how closely the regression line approximates the actual data in the context of regression. It is thus critical when a statistical model is used to predict future results or to test hypotheses.

Numerous regression models use distance metrics to evaluate convergence to the best possible result. Even the definition of the best result must be measured by some metric. Typically, the (MSE) or Root Mean Squared Error (RMSE) are used. The square root of the MSE error is used to return it to the original unit, while keeping the property of penalizing higher errors. Therefore, the square root of the mean of these residuals is denoted by equation 2:

$$RMSE= \sqrt{\frac{\sum_{i=1}^{N}(Predicted-Actual)^2}{N}} \quad (2)$$

where predicted is the value predicted by us $h(x) = \Theta_1 X + \Theta_0$, actual is the actual values and N is the number of rows in the training set (observations). To improve our prediction, we must c the cost function.

3.5. Types of Model Fit

The capability of a model must be balanced to reduce bias and variation (prediction errors). These errors would assist us not only for accurate models but also to avoid the fault of underfitting and overfitting. The bias is the difference between our model's average prediction and the actual number we are aiming to predict. High bias models oversimplify the model and pay insignificant attention to the training data. It always clues to extreme errors in testing and training data. The variance is the variability of a model's prediction for a certain data point or value which indicates how widely distributed our data are. A model with a large variance pays close attention to the training data and does not apply to new data. As a result, these models have significant error rates on test data yet perform quite well on training data. Assume we have a very accurate model, thus the error of the model will be low, implying a low bias and variance. Similarly, when the variance increases, the spread of data points increases which results in less accurate prediction. And when the bias increases the error between our predicted value and the observed values rises.

Underfitting occurs when the data has a high bias thus the model does not perform correctly in the training data. Overfitting occurs when the data has a high variance meaning that the model performs well on training data but not accurately on the evaluation set as depicted in figure 7.

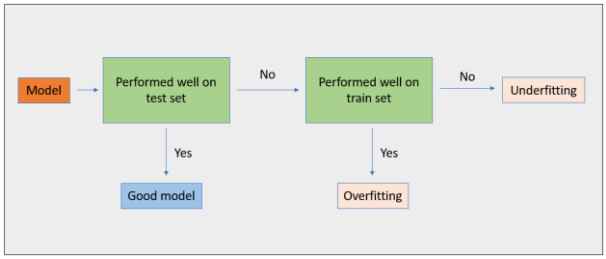


Figure 7. Overfitting and Underfitting Diagram

Overfitting occurs in the following ways when the training data is not cleaned and includes unwanted values when the model has a significant variance, when the training data size is insufficient and when in the case of high variance and low bias occurs. There exist two ways to overcome overfitting either reducing the model complexity or regularisation method. In regularization, we typically keep the same number of features while decreasing the magnitude of the coefficients. Underfitting occurs in the following ways when the model makes accurate but incorrect predictions at first when in the case of high bias and low variance occurs when the size of the training dataset used is not enough. To solve underfitting or high bias, We can eliminate underfitting by increasing the amount of training data used. Another approach is to expand the number of features which increases model complexity and thus decreases high bias.

While using linear regression, particularly with small datasets, there is a high risk of overfitting the model. This is due to the low bias, a smaller training set increases the variability in how the model performs on testing sets.

3.6. Model Improvements

3.6.1. Selecting the Right Features for Model

When dealing with a high-dimensional dataset, it would be inefficient to use all of the variables because some of them may be presenting unnecessary information. We would need to choose the right set of variables that provide an accurate model. It should be considered the variables that are chosen should not be correlated with one another. Instead of selecting the variables by hand, we can set any statistical measure like R-square, t-stat etc. to a selecting criterion.

3.6.2. Regularization in Machine Learning

The issue we usually experience is that we always optimize the cost function on the training data. When we build an ML model, we divided the samples into two groups such as training and test data. We do not demonstrate the testing data portion of the model, which causes a problem known as overfitting. Although the model will fit well on training data, it may not fit well on test data. This means that any new data we use for model output will either be inefficient or ineffective. As a result, avoiding overfitting is a critical aspect of training the ML model. If the model is overfitting, its accuracy will be low. This occurs because the model is trying to capture too much noise in the training dataset.

Ridge regression (L2) and lasso regression (L1) are two well-known shrinking methods that are frequently used in linear regression [25]. Both regularizations are of the same type, but the formulation of the loss function differs between the two. In the case of L2, we take the square of the errors, while in the case of L1, we take the modulus of the errors. We can have multiple answers with a lasso but only one solution with ridge type. They basically work by introducing some bias into the model to reduce the variability of how the model performs on test sets. This is particularly useful when the linear regression model's coefficients are spread out. Ridge regression comes into play here.

Large weights are not expected in the model. If the weights are very large, even minor changes in the weights can possibly make a big difference in the target variable. The difficulty is to determine the optimum weights because we want to make the target variable more resistant to changes in the parameters. The goal of using regularization is to select the most appropriate set of features for the model to ensure that the prediction accuracy is high. In regularization, we typically keep the same number of features while decreasing the magnitude of the coefficients.

3.7. Proposed Ridge Regression Model

The regularization methods add a penalty term to the model's loss function that maximizes accuracy means minimizing the loss function. Ridge regression is a type of linear regression in which coefficients are penalized. This method can be used to decrease the effects of multicollinearity in L2, which can be caused by high correlations between predictors or between predictors and independent variables.

L2 is used in ML to decrease linear model overfitting. L2 is used when there are several highly correlated variables. It helps to avoid overfitting by penalizing variable coefficients. L2 decreases overfitting by including a penalty term in the error function that causes the coefficients to shrink in size. L2 is similar to ordinary least squares regression (OLS), except that the penalty term prevents the coefficients from becoming too large. This is useful in situations where there is a lot of noise in the data because it keeps the model from being sensitive to individual data points. L2 is frequently used associated with other ML methods such as cross-validation to reduce overfitting even further. In addition, L2 is less sensitive to outliers than linear regression. L2 has the disadvantage of being computationally costly and requiring more data to produce accurate results.

The variance of the coefficients is significantly reduced when they are shrunk. When we shrink the coefficient estimates, we principally bring them closer to zero. The need for a shrinkage method arises from issues with data underfitting or overfitting.

$$J(\Theta) = \frac{1}{T}\left[\sum_{t=1}^{T} (\Theta^T X(t) - y(t))^2 + \lambda \sum_{i=1}^{n} \theta_i^2\right] \quad (3)$$

Equation 3 is divided into two parts: the least square function and the term regularization to the optimization function. This is a new and improved cost function. The advantage of L2 over Linear Regression is that it takes advantage of the bias variance. As λ increases, the coefficients get closer to zero.

The fundamental flaw of Ridge Regression is that it produces a model that contains all (t) predictors, regardless of the size of their coefficients, which can be challenging for models with a lot of features. This problem is solved via variable selection in L2 by utilizing the L2 penalty. In contrast to L1 penalty, which considers the coefficient's absolute value rather than squaring it.

3.8. Fundamentals of the Ridge Regression

The scaling of features is an important step in modelling algorithms with datasets. The obtained data includes features of various dimensions and scales. Different scales of data features have a negative impact on dataset modelling. It results in a biased prediction outcome in terms of misclassification error and accuracy rates. Therefore, prior to modelling, the data must be scaled.

Normalization and standardization are the two most commonly used techniques for scaling numerical data before modelling. Standardization is a scaling methodology that converts the statistical distribution of the data into the following format to make it scale-free in equation 4.

$$z = \frac{x - \mu}{\sigma} \quad \text{where } \mu = \frac{\sum_{i=0}^{n} x}{\text{count } x} \quad \text{and } \sigma = \frac{\sum_{i=0}^{n} (x - \mu)^2}{\text{count } x} \quad (4)$$

As a result, the whole data set scales with a zero mean (μ) and unit variance. To shift the distribution to have a mean of zero and a standard deviation (σ) of one, standardization scales each input variable separately by subtracting the mean and dividing by the standard deviation.
Linearity considers that the independent variables and dependent variables have a linear relationship. It is significant to validate its performance after building the model. In this project, we will focus on residuals and R2 scores.

Multicollinearity is based on the assumption that there is no correlation between the predictors used in the regression. Using the corr() function from pandas dataframe, we can compute the pearson correlation coefficient between each column in our data to see if there is any correlation between our predictors. After that, we can use seaborn's heatmap() function to display it as a heatmap.
Because we wouldn't be able to differentiate between the individual effects of the independent variables on the dependent variable, multicollinearity can be problematic in a regression model. Multicollinearity might not have as much of an impact on the model's accuracy. However, we risk losing accuracy in identifying the contributions of specific features in our model, which can be trouble for interpretability.
The following issues could lead to multicollinearity: the issues with the dataset at the time it was created may have caused multicollinearity. A lack of ability to manipulate the data or poorly designed experiments may be the cause of these issues. If new variables are created that depend on other variables, multicollinearity may also happen. Making the dataset's identical variables available, a multicollinearity issue can also result from the incorrect use of dummy variables and multicollinearity issues can occasionally result from a lack of data.
To assess how well one independent variable is described by the other independent variables, the R2 value is calculated. When R2 is high, that means a variable's relationship to other variables is highly correlated. As it becomes closer to 1, the correlation between the target and estimated variables is considered to be higher.
To evaluate prediction performance objectively, it is necessary to divide the dataset. Most of the time, it is enough to randomly divide the dataset into two subsets: First, we used the training set to train or fit our model. For instance, we could use the training set to identify the ideal weights for linear regression. Second, a true statement of the final model requires the test set. It is not recommended for fitting. Although there are numerous packages for data science and ML, for the purposes of this tutorial, we'll concentrate on the model selection package's function train test split () from sklearn.
We can reach the intended behaviour by using optional keyword arguments. The quantity of train size specifies the size of the training set. If we provide a float, it must be between 0.0 and 1.0 and specifies how much of the dataset will be used for testing. If we set it to one, the total number of training samples will be defined by an integer. The default value is none. Test size is the number that specifies the size of the test set. It is comparable to train size. Either the train size or the test size must be specified. If neither is provided, 0.25, or 25%, of the dataset will be used for testing by default. We should use data that hasn't been used for model fitting to estimate ML models' predictive performance objectively. Because of this, we must divide our dataset into training, test, and occasionally validation subsets using sklearn's train test split() function.

4. METHODOLOGY PROPOSED SYSTEM

4.1. Wi-Fi sniffing

The device sends a specific message which is called Probe Request (PR) which is used to recognize the accessible Wi-Fi networks around the device and their information. Mobile Wi-Fi devices broadcast straightforward packets known as PRs to discover nearby 802.11 APs. The MAC address supported data rate, and supported connection to an AP are all provided within those unencrypted communications. Packet filtering is the process of collecting data on a Wi-Fi channel; it requires a Wi-Fi antenna which supports monitor mode and particular software to record packets. It receives answers to connect to the network when it sends a burst of these messages with associated a time period. Every APs receive PRs within time and replies to the device by sending a probe response frame to start the connection. Capturing PR frames is simple like receiving any other Wi-Fi frame. It can be done via a compliant receiver set in monitor mode.
The PRs broadcast management packets by Wi-Fi enabled devices. These are broadcast by all Wi-Fi enabled devices regardless of the manufacturer or model of the devices. While some devices even utilize the PRs as a less accurate form of localisation, they continuously send PRs when Wi-Fi has been switched off. Therefore, these signals can be used to identify the occurrence of Wi-Fi enabled mobile devices. The sniffer will capture the wireless traffic in the network with the help of a NIC placed into monitor mode. Although it is very easy to implement to capture data, such a system must deal with several problems [29].
Wi-Fi PR records can be passively collected by sensing signals sent by mobile devices over the air, that do not need any actions from event participants, in contrast to GPS data, which are captured actively [30]. It can reduce manpower costs for data collection while covering a much larger population of participants. Furthermore, while GPS can only be utilized outside, Bluetooth and Wi-Fi systems can be utilized to monitor crowds both inside and outside. Indoor museums examples are studied in [31] and hospitals are studied in [32]. Outdoor examples include such that pedestrian commercial districts and streets [33]. Bluetooth sensing and Wi-Fi PR records are better suited for monitoring and comprehending crowds at a large social gathering. Bluetooth detection records are more accurate than Wi-Fi PR records because Bluetooth has a smaller range [34]. However, since a mobile device is more likely to have Wi-Fi turned on than Bluetooth, Bluetooth detection records have a much lower detection rate, which can be as low as 3% of Wi-Fi PR records, based on the research [35]. As a result, in order to ensure the detection rate and scalability of this work, Wi-Fi PR records are collected and analysed in order to understand crowd behaviours at a large social event. Also since Wi-Fi signal-based methods are less affected by light intensity than image recognition-based approaches, therefore we applied the Wi-Fi signal-based method.
Prior studies that used Wi-Fi PRs at social events or in specific environments can be divided into two groups based on according to whether the researched areas are outdoor or indoor. We are primarily concerned with indoor space in this work. Although previous studies have shown the efficiency of mining Wi-Fi PRs to comprehend crowd behaviours in both outdoor and indoor settings, the collected data has not been critically analysed.
Particularly, in previous studies methods used to evaluate the collected data mostly include domain knowledge-based processing, visualization and statistics [36]. With the fast evolution of ML algorithms, we believe that more supervised learning algorithms for mining Wi-Fi PRs should be discovered. As a result, the purpose of this paper is to fill that gap.

The typical Wi-Fi communication range is approximately 35 meters indoors and more than 100 meters outdoors. One can be specified that the location information of devices, and consequently of people, by combining the time at which PRs were received, the MAC addresses they contained, and the locations of the sensors. For privacy reasons, several recent mobile devices have begun to use randomly generated local MAC addresses while they are not connected to Wi-Fi, but not all of them have this schema. Less than 50% of the sample's devices have randomized MAC addresses, and there is no way to map PRs to specific people using just MAC addresses, according to a 2017 study by Martin et al. [37]. Additionally, it has been demonstrated that it is still possible to derive important conclusions about crowd behaviours from datasets that contain a significant portion of randomly generated MAC addresses [38].

Each Wi-Fi-enabled device is assigned a physical address by the manufacturer or a random address by the software as a security function. This address is also the source address property of the PR frame, which can be learned from the packet. The randomization of MAC addresses is the development of producing virtual MAC addresses by end devices while scanning for AP in the Wi-Fi framework. This is presented to devices whose real MAC address remains unknown which stops tracking. The device and AP adjust the connection when they find themselves. Afterwards, the device uses its real MAC address since only starting from that moment the whole communication is encrypted. Some manufacturers send PRs with a randomized MAC address to avoid device tracking.

Subsequent steps have been taken by mobile device developers to make it more difficult for someone to be tracked. The timeline in figure 8 shows how Apple and Android have implemented security measures in recent years. By 2017, both businesses were scanning and searching for potential network connections while randomly generating the MAC addresses of mobiles. In 2018, Android introduced randomization when connecting to a network, and by 2019, its operating system (OS) had adopted this practice generally. Apple recently made progress by randomizing all MAC addresses upon connection and rotating randomized MAC addresses every day during 2020. By making any serious efforts to crack MAC randomization temporary, the representation of this concept assists in further discouraging attackers who can do so.

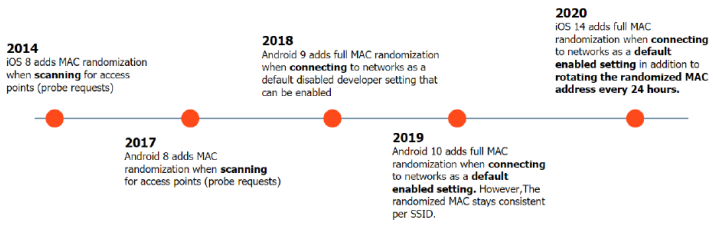


Figure 8. Timeline of Recent MAC Randomization Improvements Made by Apple and Android

It has become required to hide MACs since, even though these addresses don't include any personal details, data cross-checking could still associate it with private details. By using disposable interface identifiers users' privacy has improved. This means that PRs do not use the real MAC address of the device. For instance, a new MAC address can be used for each scan repetition, where one scan repetition is based on sending PRs in all usable channels. It is important to differentiate valid MAC addresses from randomized ones to estimate occupancy precisely and avoid overestimation. We can use the OUI part of the MAC address to test if the address is authentic or not by considering the OUI in the IEEE table of registered vendors. If the OUI is not in the table, the MAC address is considered randomized else it is non-random. [37].

Moreover, as the local bit, when this is set (1) it means that the MAC address is locally assigned and does not need to be unique(locally administrative). More than 99% of locally

assigned MAC addresses are actually randomized, according to a recent study by Martin et al. [37]. As a result, the local bit's presence as in OUI is a significant indicator of randomization.

The MAC address randomization process is managed by the device OS and there aren't standard algorithms for this procedure and each OS applies its own randomization operations. The counting methods that use MAC addresses as device identifiers are put to the test by MAC address randomization. Such kinds of methodologies have evolved as a result of this functionality.

Linux OS presented the MAC address randomization starting from version 3.18 of its kernel. Most Wi-Fi drivers are arranged to change the MAC address every 60 seconds. Though, the Linux OS has three methods for assigning a MAC address: use the real MAC address, use a totally virtual MAC address or use a partly virtual MAC address keeping the first three octets equivalent to the real OUI of the network card manufacturer.

In Windows OS, the random MAC addresses are used in the PRs and also during the authorisation and association periods before the network connection.

Google announced the randomization of the MAC addresses starting with version 6 of Android OS. This is enabled by default on every device in version 8. Google's implementation uses a set of bogus MAC addresses for network discovery.

Apple presented the MAC address randomization in iOS version 8. based on laboratory works with different models. The iOS devices randomize the MAC address for each burst of PRs. In this case, the level of privacy guaranteed is very high. Overall, each manufacturer implements special algorithms; the result is a high variability of the MAC addresses randomization process.

A method to avoid Wi-Fi enabled devices from being tracked is MAC randomization. The device won't display its actual MAC address while in the air when this option is set. The random MAC will be used for all MAC layer operations. The unique MAC address may be used by some locations, like malls, shops, or other open spaces, to monitor visitor movement. The position tracking and information metrics will suffer from a device's MAC address being changed regularly. Few devices will produce various random MAC each time when they scan based on the randomly generated MAC lifetime, and they will create a different MAC each time they connect to the AP seen in figure 9. Furthermore, only just a few devices will produce the same random MAC address each time they connect to the same SSID.

No.	Time	MACAddress	Signal strength (dbm)
377	60.347497	XiaomiCo_e3:89:19	-33 dBm
378	61.848815	b6:3e:2c:6e:26:12	-66 dBm
379	61.868884	b6:3e:2c:6e:26:12	-65 dBm
380	62.118793	3a:68:c4:eb:5b:2d	-58 dBm

Figure 9. Random MAC and Non-random MAC Display in Wireshark

We expect to capture PRs only from the devices that are around. Though, a sniffer is able to capture PRs from all devices in its communication range. That range may be 150 meters for Wi-Fi [2]. Hence, It is important to decrease the communication range of the sniffer to match the threshold of the place. This can be done by setting an RSS threshold and considering only probe requests with an RSS higher than such a threshold.

Nowadays it is common that always carry one Wi-Fi enabled device but this may not be always the case. Some people may not have a device or carry a device off or even have more than one Wi-Fi device. The number of unique MAC addresses observed from the captured PRs can't be mapped directly to the number of occupants in a space. To solve these problems, we can use ML algorithms as a supervised learning model that is able to adapt to the specific area under consideration and to its specific conditions. Furthermore, randomized Wi-Fi MAC addresses have a many to one relationship with the source

devices. To address this issue, we employ a supervised learning model that takes as input the two sets of previously described features as well as ground truth occupancy data provided manually.

4.2. Capturing Probe Request Data with Wi-Fi Sniffer

In this section, we will now discuss the Wi-Fi sniffing process. PRs are plain text packets broadcasted by Wi-Fi mobile devices to discover 802.11 APs in their nearby area [1]. This unencrypted message includes source data. Sniffing is the method of collecting data on a Wi-Fi channel and needs a Wi-Fi antenna that supports monitor mode and specific software to collect packets [39].
Many devices can be used to activate monitor mode either out of the box or with modified firmware. In addition to enabling packet capture, the monitor mode also allows the capture of other physical data means that each record contains fields including MAC address, time of detection and RSSI. No other information about the devices is held to protect people's privacy, making it impossible to trace back to specific people [30].
Wireless adapters that are installed by default in a computer can not be set in monitor mode thus we may need to purchase an external wireless adapter compatible with monitor mode that can be connected through a computer port [39]. The packets can be captured on a Wi-Fi interface either in managed mode or if the hardware supports it, monitor mode as well. The usage of monitor mode is not needed if we are not interested in IEEE 802.11 management/control frames or radiotap headers and we only care about traffic to/from your capture device. What we'll get instead are packets that have fake IEEE 802.3 framing. However, in the case of we do care about radiotap information or capturing all traffic on a particular channel, we will either require to fit our interface card to monitor mode or utility an external device capable of collecting IEEE 802.11 traffic. Note that not all Wi-Fi cards support monitor mode and support may vary depending on OS.

The tcpdump sniffer is used to record Wi-Fi packets. All other packets are filtered out so that only Wi-Fi PRs are received which are the only ones in which we are interested. These frames are typically transmitted in bursts on each of the Wi-Fi frequency channels that are currently in use, with a temporal periodicity that depends on the device's manufacturer, OS, and operational mode like active vs. stand-by. In this project, we take into account the use of Wi-Fi PRs, which are constantly transmitted from Wi-Fi enabled smart devices to accomplish occupancy estimation [19]. Wi-Fi sniffer equipment is utilized for passively capturing indoor PRs from Wi-Fi devices such as smartphones and tablets that do not require connectivity to a Wi-Fi network. This data is then used to determine the occupancy count for each zone [40].
Much prior studies in the research have successfully used Wi-Fi sniffers for passive Wi-Fi packet collection. Nevertheless, there isn't a standard for how to utilize the channel or how to listen for packets containing implementation details. When the device searches for a nearby network, PRs are sent in bursts across multiple channels. The frequency of probes differs depending on a number of factors, including device manufacturer, OS, and screen status. When a device's battery is low, it tries to conserve as much power as possible by setting a low frequency of PRs [16].

Given these limitations, the Wi-Fi sniffer should be able to cover multiple channels and receive more frames at the same time. It is unquestionable that rising sniffer density would result in more data. A three channel sniffer is typically recommended in an 802.11b/g/n (2.4GHz) environment. This requires the sniffing device to have three wireless adapters (antennas), with the antennas set to channels 1, 6, and 11; but, this increases the cost

and design complexity of these devices. We are especially interested in creating and understanding the principles of a channel monitoring scheme. We first only use one Wi-Fi module for our prediction of the number of people in space, rather than changing the number of sniffers or the coverage area [2].

Wi-Fi adapters are devices that connect to the internet. Most laptops, tablets, and smartphones include a Wi-Fi card. Data is sent from the device to the internet by way of packets in a wireless environment by sending a packet request to the router.

The router receives the requested packet from the internet, and once it has obtained the webpage, it sends the information back to the device in the form of packets, controlling all traffic to connected devices. Although the promiscuous mode is used for packet sniffing, monitor mode (IEEE 802.11) allows you to access all data packets, even if they were not sent through this mode and controls traffic received on wireless only networks. Monitor mode can collect all of these packets, which are not only directed to their device but also to other network connected devices.
The data captured by the Wi-Fi sniffer contains time stamps that provide the time of captured data, the MAC address of the Wi-Fi enabled device and the signal strengths of the Wi-Fi device. The Kali Linux OS has been installed to capture data information from the equipped Wi-Fi sniffer equipment. The information is saved on the Wi-Fi sniffer device's SD card. Although most Wi-Fi devices are used to link to an AP, they can also be configured to monitor Wi-Fi traffic, including PRs. Such capacity can be easily obtained by using a Linux laptop, empowering the monitor mode on a wireless network interface card (NIC), and installing linux based software Wireshark for packet capturing which enables to collect all of traffic between adjacent APs and client devices without requiring a network connection or data transmission. This means that passively collecting PRs is possible using off the shelf hardware. There are numerous portable and small Wi-Fi sniffers available that use off the shelf hardware. However, not all data collected during PRs is relevant and useful for further analysis. Because not every method works for every device, we set the wireless interface to monitor mode by allowing iw configuration using airmon-ng, tcpdump to collect and obtain the most relevant elements of PRs efficiently. We established the wireless card into monitor mode to be capable of detecting nearby devices passively. This study proposes a simple model based on the following steps to capture and extract probe requests are illustrated in figure 10.

Figure 10. Steps to Collect Wi-Fi Probe Requests

While collecting PRs, the wireless adapter adds extra information such as radiotap headers [11]. Radiotap is a de facto standard that includes data like RSSI, timestamp, and channel frequency. However, storing or sending the raw captured data to an SD card directly is inefficient because only the following elements are relevant for passive Wi-Fi tracking: The Source MAC Address is used as a unique identifier when not randomized, while the RSSI can be utilized to predict the distance (that will be useful to set threshold value). Due to privacy reasons, the MAC address is randomized on the collecting device itself and never sent as clear text.

4.2.1. Configure Interface

The sniffing is operating in the Raspberry and the interface has been configured to monitor mode and assigned to a particular channel afterwards the data collection has begun. This setup configures the sniffing interfaces, begins the sniffing in the indicated interface, and saves the captured data in a file for the specified channel.

4.2.3. Data Filtering

Several related works observe that Wi-Fi PRs frames are transmitted with a periodicity of 2–5 min by most personal devices [41]. The value of the minimum Time to Live (TTL) in this work is set to 2 at first. This has the impact of removing inaccurate MAC addresses from the estimate for example people passing by the environments. Thus, each PR in a burst differs only slightly from the next. This implies that an aggregation duration is preferred for more precise measurements.
Depending on the type of measurement, data should be captured in shorter time periods to create a heatmap of nearby devices. While only receiving basic usage statistics, such as the number of devices present, the data can be aggregated in a short amount of time. It should also be noted that smartphones frequently send PRs in bursts. Such bursts can contain up to 50 PRs per second [2] and are utilized to probe for various SSID.

4.3. System Specification

The new approaches use ML models to solve the MAC address randomization issues. The objective of counting people is the safety and privacy of information collection and generation. Therefore, recognized access and use of the platform should be ensured accordingly to the operator's role and allowed permissions [12]. Furthermore, the anonymization methods and therefore the remainder of the protection features should provide the chance for compliance with information and data privacy or data protection regulations. The platform must be affordable in terms of cost and straightforward to use.
From the purpose of view of the precision requirement of the system, when estimating the number of individuals it's difficult to determine an actual value. The requirement to hide different styles of events and environments suggests that the platform should be scalable and versatile, making it possible to update the hardware and software when necessary. The design and implementation of the platform should be supported by standardizations and open solutions.

4.4. System Requirements

The initial purpose of this research is to collect a series of probe requests and process them for counting people. We can do this by using a Wi-Fi receiver to capture and analysis of PR frames, data extraction, transmission and storage, analysis of PR frames patterns and provide an estimation of the number of people. Finally, we can compare the processed people counts to the ground truth.

4.5. System Design

The methodology and system design choices are presented in this chapter. We explained the system's aspects and their functionalities, and then explained how the developed system works. Because the objective is to reduce costs while increasing the accuracy of the solution. Thus we should rely on a radio-based method. Our design requires an indoor sniffing device (which is Raspberry Pi in our case). This device would

then measure the PRs frames on a timeframe, and compute the estimated number of people in each environment using a trained model.
Despite the fact that each obtained PR allows for the processing of a variety of data from the device, two of the most significant for this work are the device's MAC address and the RSS indicator. The RSSI measures the power of the received packet in decibel milliwatts. These two types of data may be sufficient for trying to perform occupancy estimation: a sniffer device receives PR frames for a set period of time, counts the number of individual MAC addresses discovered with RSS greater than a certain threshold, and returns its prediction.
We can simply identify features of our samples by integrating these two characteristics. As an example, each sampling would collect several PR packets and we would then set as a feature the number of unique MAC addresses found with power less than a certain threshold. These could be used as features to train the model.
It takes a few steps to collect and aggregate Wi-Fi PRs. It is essentially divided into three major parts, which are discussed in the sections that follow.
1. The first step includes passively capturing Wi-Fi data with IoT devices.
2. The second part is the preparation process, which includes cleaning unnecessary information and aggregating information over a variable time period. The data can then be saved in CSV (Comma separated value) files.
3. The final step is to process the captured data, which is accomplished by utilizing ML algorithms.

The system is described in figure 11 as follows and the detail of the component in the block has been discussed in this section.

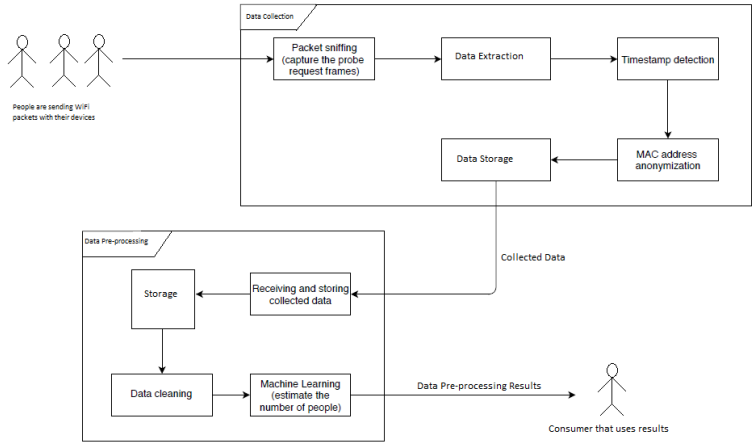


Figure 11. System Architecture

The first system component is a data collector which is situated in the area of interest. With its implementations, it accomplishes data preprocessing by sniffing Wi-Fi packets, capturing PR frames, extracting useful information from these, detecting the current timestamp, using anonymized MAC addresses which leads to no privacy issues, and saving the data in storage.
At first, the pre-processing stage is concerned with receiving and storing collected data in storage. Then it focuses on data cleaning: RSSI thresholding, creating two different features from randomized and non-randomized MAC addresses and calculating the number of devices existing in the area of interest. Ultimately, it addresses data analysis using ML to determine the number of devices detected and the number of people existing. At the completion of this process, the consumer receives the results of the preprocessing such as the number of people in the place and may use this data to enhance its business.

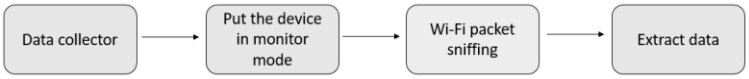


Figure 12. Flowchart Representing the Data Collector Logic

Figure 12 depicts the flowchart of the data collector logic. When the data collector is turned on and in monitor mode, packet sniffing can begin. Just after that, the Wi-Fi packet analyzer will enable us to obtain the data that we captured. It records PR frames. The RSSI as well as MAC address information is extracted when a PR frame is captured. When a packet is detected, the timestamp and other details are saved. The OUI is used to determine the device's manufacturer, and the local bit is used to determine whether the MAC address is random.

To eliminate other wireless devices and receive only smartphone devices, the list of well-known smartphone manufacturers is used and checked against. This is achieved by comparing the OUI portion of the MAC address to the manufacturer's known OUI. The list is created using the manufacturer registered MAC addresses from the IEEE's public listing. The list that was used is included in appendix b at the conclusion of this thesis.

We chose to clean this set as well because we reviewed the OUI table from the IEEE standard[1] the dataset contains different shapes of MAC addresses as shown in table 1. It is required that use some algorithm, that cleans the dataset for the OUI table and we checked the 0.000706019 MAC address case which the Python script ignores by checking different conditions such as

Check 1: ':' is not present in MAC
Check 2: '.' is present in MAC
Check 3: there is no alphabet in MAC

Apart from that, we checked the first 3 octets of the MAC address in both columns in the dataset as seen in table 1. There was a case that combined one column with a comma. We also need to ignore empty cells in the dataset since before that we received many times errors about the dataset containing “nan” which means that it has an empty cell. We ignored the empty current cell in the OUI table by checking if a current cell in the OUI table is "empty". After some research, we found a clearer OUI table which consists of separately all the MAC addresses and vendor names, the python script work for that as well.

Table 1. List of OUI Mac Address and Vendors

	A	B
1	Mac Prefix	Vendor Name
2	00:00:0C,"Cisco Systems, Inc",false,MA-L,2015/11/17;;	
3	00:00:0D	FIBRONICS LTD.
4	0.000706019	GATEWAY COMMUNICATIONS

The captured data is received and stored in the pre-processing section. Data cleaning and data analysis are the two primary pre-processing functions. Figure 13 depicts the diagram representing the pre-processing and cleaning logic. At first, data is received in pre-processing when it is posted by the data collector. The information is then saved in storage. To remove data from devices that are too far away from the data collector, an RSSI-based threshold is used. The position of the data collector has to be taken into account, as well as the space where they are located for cleaning and analysis of the data.

The unique devices are extracted with their existence time. The following parameters must be suited to the study case: minimum time and power threshold which helps remove the randomized MAC addresses. Ultimately, data analysis is used to estimate the number

[1] OUI Table from IEEE standard

of people who appear in the area of interest. To obtain the final outcomes for example the number of people in the space, we implemented ML into this project.
We searched for randomized unique MAC addresses that have an RSSI greater than certain power thresholds. To evaluate randomization, the MAC address's OUI is looked it up within the most recent OUI table accessible on the IEEE.

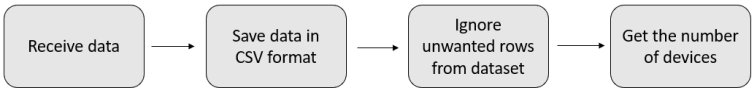


Figure 13. Flowchart Representing Cleaning Logic

If it is not found or if the local bit is set (1), the address is labelled as random otherwise it is not random. For instance in column C, there is Apple which will be checked from the OUI table and set as a non-random MAC address while ae:2a:79:d1:93:26 will be a random MAC address which can be confirmed through local bit is set to 1. Based on the power threshold settings, this process will not consider anything beyond that threshold as well as a feature.

Table 2. Filtered Data Samples

No.	Time	MACAddress	Signal strength (dbm)
52	0.824	Apple_81:ff:a7 (38:f9:d3:81:ff:a7)	-58 dBm
56	0.886	Cisco_f3:53:84	-67 dBm
57	0.914	ae:2a:79:d1:93:26	-70 dBm
105	2.017	Google_aa:58:c2	-41 dBm
153	3.07	Chongqin_2e:f1:f9 (a4:97:b1:2e:f1:f9)	-71 dBm

Within an aggregation period, multiple RSSI values for a single MAC address are possible. We created a python script for filtering data and we set the power threshold -70dBm for filtering which leads us to ignore the far distance MAC address since they will have less than -70dBm.
Moreover, by extracting values such as the OUI or the local bit from the MAC address, data can already be filtered on the capturing device. This enables us to exclude frames with OUI from manufacturers known for producing more connectivity hardware such as Apple or Cisco as seen in table 2.

4.6. System Algorithm in Python

There exist two type of MAC addresses in the captured CSV files which are Apple_62:8d:05 and 1e:0d:63:d0:0f:59. We indicate if vendorname_xx:xx:xx as a first type and xx:xx:xx:xx:xx. And if we see in the dataset “vendorname_” as the first type of MAC address and if we see “:” colon as the second type of MAC address.
We checked these type of MAC address that we collected during our experiments existed in the OUI table or not in the both columns. We search for the first type of MAC address till “_” underline and the first three octect of the entire MAC address for the second type of the MAC address. We converted the second type of MAC addresses to the binary format from the hexadecimal since we need to check the first octets' last second bit for the local bit assigned.
In order to avoid collecting distance mobile receivers' MAC addresses, initially, we set the alpha power threshold to -70dBm. As we decide to have two features, one feature is assigned for the randomized unique MACs greater than alpha power threshold and as an another feature non randomized unique MACs greater than alpha threshold power. We ignore the current MAC and move to the next MAC address in the case of power less than a specific threshold that we set before. In the case of type 1 MAC address, we checked whether it is in the OUI table or not and based on that we decided that it is random when it is not in the list of OUI table or it is non-random when it is in the list of

OUI table. In the second type of MAC addresses case, we check whether the local bit is on (1) or off (0) and then based on that we decided whether it is random or non-random respectively.
Pandas package is the most widespread data manipulation in python. The basic process of loading data from a CSV file into a Pandas DataFrame is achieved using the “read_csv” function in Pandas. We read both columns of the OUI table to check MAC addresses with the help of pandas. We cleaned the dataset of the OUI table since it included not clear MAC addresses shapes such as 0.001087963 and empty cell that we ignored both cases by checking it consists of “:”, “.” any alphabet or empty cell respectively.
When we collected the data in CSV files, we observed the different looks of MAC addresses as well and need to clean them. There were “192.168.. “ and “fe80:: “ those come from networks that we want to ignore. We checked if the MAC addresses not includes “:” or include “::” and set them to the empty cell since the Python script will already ignore that lines.
We brought all CSV files we collected in one folder for each experiment and read all CSV files in the folder. Since each CSV file indicated like groundtruth_people_vx and in this case we were able to take ground truth value y(t) from file names automatically. And for the input samples which are our two features we either append values to the end of an array or stack arrays in sequence vertically based on the number of elements in the array (size).
As we remove duplicates in datasets afterwards we decide to implement a Python code for filtering out duplicates as well. The filtering process works as follows we check the power level column and MAC addresses to remove duplicated MAC addresses based on the power levels. When we have power less than or equal to the power level threshold, we considered that the last of MAC address else sort the column includes power level and keep the last after sorting in ascending in this case we were able to keep the closest one to the threshold.
In the ridge regression section, we have alpha, the number of iterations and L2 penalty parameters. Initially, we set the weights and bias to zero after that applied gradient descent learning. We calculate the gradients with the help of another function for updating weights in gradient descent.

4.7. Proposed model for occupancy with Ridge Regression

Since a smartphone connected to an AP constantly transmits data packets, the number of people can be estimated using these packets. When a smartphone is not connected to an AP, it typically periodically sends PRs to look for other APs. Every Wi-Fi probe packet sent by a customer is captured along with its time, RSS and corresponding transmitter address. Every sensor's collected data would be stored on the sniffer itself. A CSV file is used to store the data that was obtained. For a simpler method, the data is manually copied to a computer and fed to the counting algorithm. The complete system is shown in figure 14.

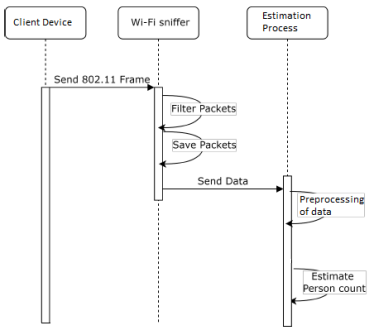


Figure 14. Proposed Model Principle

We implemented occupancy estimation employing a single piece of hardware capable of sniffing Wi-Fi frames. The market already offers several affordable options for this task for instance the €60 Raspberry PI3 model B + is equipped with Broadcom's BCM2837B0 chipset capable of 802.11ac/b/g/n. The intended deployment scenario is illustrated in figure 15.

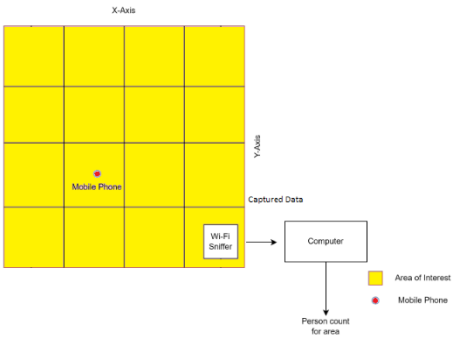


Figure 15. Implementation of Occupancy Estimation

As mentioned in section 3, we obtained from the Wi-Fi PRs recorded during time interval t mainly two types of features to address the MAC Randomization and RSSI threshold issues.

• $\mathrm{random}_x(t)$, $x_{min} < x < x_{max}$ a group of features, each of which counts the number of randomized unique MAC addresses with RSSIs greater than x dBm in their collected PRs. To evaluate randomization, the OUI of the MAC address is searched in the most recent IEEE OUI table. The address is labeled as random if it is not found or if the local bit is set.

• $\mathrm{nonrandom}_x(t)$, $x_{min} < x < x_{max}$ each feature in this set, like the others, counts the number of non-randomized unique mac addresses collected in the interval whose RSSI is greater than x dBm. It is obvious that selecting the proper threshold is critical for accurate occupancy estimation.

We decrease the RSS with a penalty term in the ridge regression. Vectorization is one of the most effective techniques for improving the efficiency of the ML code. ML algorithms need a lot of calculations and that can be time-consuming. Assume that we have a dataset with 100,000 data points and want to perform some type of operation on each of them. If we utilize a regular loop for this, situations will quickly become very inefficient. Our code will run much faster if we vectorize our operation and only perform a few large vector or matrix operations using something like NumPy. There are 2 different classes of ways to solve the parameters to fit the best possible model. In this section, we determined two approaches for the ridge regression parameters calculation such as in closed form (normal equation) and using an iterative gradient descent algorithm.

4.7.1. The closed form (Normal Equation)

The linear function (ridge regression model) is defined as: y=f(X)+ ε
f(X) is a statistical formula written in vectorized form, where it takes the form of the sum of p covariance and coefficient products.

$$f(X)= \theta_0 + \theta_1 X_1 + \theta_2 X_2 \ldots + \theta_p x_p \text{ where } X_0 = 1$$

That is identical to the case of linear regression, except with more covariance. We need $n \times (p-1)^2$ for n data samples with p covariance each to compute the response variable for a given sample iteratively.

$$\hat{y}=\hat{\theta}_0 + \hat{\theta}_1 X_1 + \hat{\theta}_2 X_2 + \cdots \hat{\theta}_p X_p$$

Consider the following regression problem with N observations (rows) and p predictors (columns);

$$y=\begin{bmatrix} y_1 \\ y_2 \\ . \\ y_n \end{bmatrix}, \quad X=\begin{bmatrix} 1 & X_{1,1} \cdots & X_{1,p} \\ \vdots & \ddots & \vdots \\ 1 & X_{n,1} \cdots & X_{n,p} \end{bmatrix}, \quad \theta=\begin{bmatrix} \theta_0 \\ \theta_1 \\ . \\ \theta_p \end{bmatrix}, \quad \varepsilon=\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ . \\ \varepsilon_n \end{bmatrix} \quad \text{and } \hat{y} = \hat{\theta}\, X + \varepsilon$$

where y (vector of outcomes for each of N observations) is n × 1 vector or n dimensional column vector, x is n × $(p+1)$ design matrix each of the N observations is represented in a row. We also add in an additional 1 to each observation to account for the intercept or 'bias' term. θ is (p+1) × 1 dimentional vector of weigths and ε is an n dimensional column vector. Note that θ_0 represents the y-axis intercept of the model and therefore x_0=1.

To get our predictions, we multiply our weights by our observations X, a process known as matrix multiplication. Each entry of the vector in matrix multiplication takes multiple regression equation form.

$$\hat{y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ . \\ \hat{y}_n \end{bmatrix} = \begin{bmatrix} \hat{\theta}_0 + \hat{\theta}_1 & X_{1,1} + \cdots \hat{\theta}_p & X_{1,p} + \varepsilon_1 \\ \vdots & \ddots & \vdots \\ \hat{\theta}_0 + \hat{\theta}_1 & X_{n,1} + \cdots \hat{\theta}_p & X_{n,p} + \varepsilon_n \end{bmatrix}$$

We compute the coefficients using this matrix form by minimizing the sum of squares of residuals with the L2 norm. Considering the matrix for the residual n dimensional vector for all n samples, which is equivalent to the difference of vectors y and $\hat{y}$ (our real value minus our estimate). It is worth noting that the prediction is obtained by multiplying the weights by the number of observations. The 'e' symbol represents the residual vector.

$$e=\begin{bmatrix} e_1 \\ e_2 \\ . \\ e_n \end{bmatrix} = \begin{bmatrix} y_1 - \hat{y}_1 \\ y_2 - \hat{y}_2 \\ . \\ y_n - \hat{y}_n \end{bmatrix} = y - \hat{y}$$

In general, pre-multiplying a vector by its transpose yields a sum of squares. RSS close form solution can be rewritten as a product of $e^T e$

$$RSS = e^T e$$

$$RSS = (y - \hat{y})^T (y - \hat{y})$$

$$RSS = (y - X\hat{\theta})^T (y - X\hat{\theta})$$

$$RSS = (y^T - X^T \hat{\theta}^T)(y - X\hat{\theta})$$

$$RSS = y^T y - y^T X \hat{\theta} - \hat{\theta}^T X^T y + \hat{\theta}^T X^T X \hat{\theta}$$

To minimize the RSS, a stationary point referring to a minima is formed when the slop with respect to a variable at that point is zero. We need to determine the estimation of $\hat{\theta}$ which the partial derivative of the RSS with respect to $\hat{\theta}$ is 0. Since we are in the ridge regression case we have following regularized cost function,

$$J(\theta) = RSS + \lambda\, \hat{\theta}^T \hat{\theta} \text{ (vector form)}$$

Where RSS is residual sum of square comes from least square are in the linear regression case.

$$J(\theta) = (y - X\hat{\theta})^T (y - X\hat{\theta}) + \lambda\, \hat{\theta}^T \hat{\theta}$$

$$J(\theta) = (y^T - \hat{\theta}^T X^T)(y - X\hat{\theta}) + \lambda\, \hat{\theta}^T \hat{\theta}$$

$$J(\theta) = (y^T y - \hat{\theta}^T X^T)(y - X\hat{\theta}) + \lambda \hat{\theta}^T \hat{\theta}$$

$$J(\theta) = y^T y - 2\, \hat{\theta}^T X^T y + \hat{\theta}^T X^T X \hat{\theta} + \lambda\, \hat{\theta}^T \hat{\theta}$$

We can obtain the above equation by setting the derivative of the linear regression cost function (the MSE) to zero and solve for our model parameters. Taking derivative to try to estimate of coefficients as following

$$\frac{\partial J(\theta)}{\partial \theta} = -2\,X^T y + 2\,\hat{\theta}\,X^T X + 2\,\lambda\,\hat{\theta}$$
$$\hat{\theta} = 2\,X^T y + 2\,\hat{\theta}\,X^T X + 2\,\lambda\,\hat{\theta} = 0$$
$$-X^T y + \hat{\theta}\,X^T X + \lambda\,\hat{\theta} = 0$$
$$X^T y = \hat{\theta} X^T X + \lambda\,\hat{\theta}$$
$$X^T y = \hat{\theta}(X^T X + \lambda\,I)$$

The inverse of X^{-1} of a square matrix X is the unique matrix such that $X^{-1}X = I = XX^{-1}$. Using the closed form solution we compute the weights of the model by differentiating this function, set it equal to zero and solved for $\hat{\theta}$ and we get following expression in equation 5.

$$\hat{\theta} = \begin{bmatrix} \hat{\theta}_0 \\ \hat{\theta}_1 \\ . \\ \hat{\theta}_{p-1} \end{bmatrix} = (X^T X + \lambda\,I)^{-1}\ X^T y \quad (5)$$

This approach eliminates the need to select a learning rate (α), as GD does not. Since identity matrix is scaler with dimensions (n+1) x (n+1) is added with a zero in the top left corner to account for the bias (or intercept) term. The solution to this equation, like the normal equation for ridge regression, gives us the ideal parameters for our $\hat{\theta}$ immediately, in one step that solving the normal equation is our best choice in most cases. Coefficients are calculated by solving the unknown equation of y and predicting the value $\hat{y}$. However, if there is a large amount of features, an iterative algorithm such as GD may be preferable.

Instead of solving the normal equation, we can differentiate the ridge cost function to achieve its partial derivatives (its gradient) and then use GD to iteratively approach the optimal solution.

4.7.2. Ridge Regression with Gradient Descent

We want to implement a low cost system that can predict the number of people. We used the ridge regression approach for this. The cost function is also represented by J. Ridge regression, like linear regression aims to minimize the RSS however with a slight change to fit the training examples as perfectly as possible [28]. As we know, linear regression estimates the coefficients using the values that minimize the following equation 6.

$$J(\Theta) = \frac{1}{T}\left[\sum_{t=1}^{T}(\Theta^T X(t) - y(t))^2\right] \quad (6)$$

Linear regression adjusts all features and determines unbiased weights to minimize the cost function that could lead to the issue of overfitting. Linear regression is also incapable of dealing with collinear data. In a summary, linear regression is a high variance model. Ridge regression is used to solve this problem. Ridge regression adds an L2 penalty (square of the magnitude of weights) to the linear regression cost function. This is done to prevent the model from overfitting the data. We use the multiple linear ridge regression method because first, we had two independent variables (features) later we had more features to overcome underfitting.

We have some data as we observed the independent variables x_{i1}, $x_{i2}..x_{in}$ and the dependent variable (or response variable) y along with it. In our datasets, we had a different number of observations for each testbed. Let X(t) be an "n-dimensional" column vector achieved by grouping all extracted features from Wi-Fi sniffing during time frame t.

$$X(t) = [\text{random}_{x\,min}(t), \ldots., \text{random}_{x\,max}(t), \text{nonrandom}_{x\,min}(t), \ldots., \text{nonrandom}_{x\,max}(t)]^T$$

We have n independent variable j=1,2,3.....n and x_{ij} is i^{th} training example of j^{th}feature.

$$x_{ij}=(x_{i1}\ x_{i2}\ \ldots x_{in})$$

Now we combined all available individual vectors into a single input matrix of size (t, n) and refer to it as the X input matrix, which contains all training examples.

$$X(t)=\begin{pmatrix} x_{11} & x_{12}\cdots & x_{1n} \\ \vdots & \vdots & \vdots \\ x_{t1} & x_{t2} & x_{tn} \end{pmatrix}$$

We are trying to determine the best θ and b values. It is used the deep learning (DL) conventions θ and b, which stand for weights and biases respectively. We propose estimating the occupancy ŷ(t) using the weighted sum in vectorized form:

$$\hat{y}(t) = \Theta^T X(t) = \theta_0 x_{i0} + \theta_1 x_{i1} + \theta_2 x_{i2} + \cdots \theta_j x_{ij} \ldots. + \theta_n x_{tn}$$

The relationship between independent and dependent variables is associated by this equation because we had two features at first and it is easy to show calculation with that in our model that are generalized as follows.

$$\hat{y}(t) = b + \theta_1 x_{i1} + \theta_2 x_{i2}$$

Its significance is that it provides flexibility. Thus, using such an equation, the model attempts to estimate a value y, which could be a value we required, such as the number of people in the indoor environment. We tried to compare to the ordinary least squares (OLS) cost function, ridge regression works with an improved cost function due to the effect of the sum of the squares of the coefficients in the L2 term. The optimization technique will become easier to solve and shrink the coefficients. This penalty term motivates the model to achieve a balance between well fitting the training data and being simple. As a result, ridge regression can assist in enhancing an ML model's generalizability.

This function is now lowered to estimate the coefficients. Here, λ is the tuning parameter that determines how much we want to penalize our model's flexibility. A model's flexibility is represented by an increment in its coefficients, and if we want to reduce the cost function, these coefficients must be small. This is how the ridge regression technique prevents coefficients from becoming excessively large.

Notice that except for the intercept β_0 we shrink the estimated association of each variable with the response. This intercept is a measure of the mean value of the response when $x_{i1} = x_{i2} = \ldots = x_{ip} = 0$. If λ = 0, the penalty term has no effect, and the estimates produced by ridge regression will be equal to least squares. However, as λ→∞, the impact of the shrinkage penalty becomes more significant and the ridge regression coefficient predictions will approach zero.

We essentially used a loss function to compare the predicted ŷ and observed y. The loss function can be defined in a variety of forms, however in this work, we specified it as the squared difference between ŷ and y. The ground truth occupancy values y(t), t = 1 . . . T is provided to the system when available. In general, the lower the $J(\Theta)$ is the better. The reconfigured cost function for ridge regression, which includes a penalty term (lambda) is given below, where $\Theta = [\theta 1 \ldots \theta n]^T$ are the weights achieved across regularized linear regression as follows; Ridge regression's loss function is:

$$J(\Theta) = \frac{1}{T}[\sum_{t=1}^{T} (\Theta^T X(t) - y(t))^2 + \lambda \sum_{i=1}^{n} \theta_i^2] \qquad (7)$$

This cost is the normalized sum of the individual loss functions. The weights are penalized by a positive parameter lambda in this cost function. The cost function is composed of two functions. The first is the cost function which is the same as the one used in the linear regression model. This term assures that the training data fits correctly. The L2 penalty or regularization term is the second term. The goal of this term is to keep the parameters as small as possible.

The calculation in equation 7 depicts ridge regression, in which the RSS is adjusted by adding the shrinkage quantity. This function is now minimized to estimate the coefficients. The GD method is essential for minimizing the loss function and achieving our goal of predicting close to the original value.

The objective is to make an effort to reduce the difference between ŷ and y because we want it to be small. This is accomplished using the (GD) optimization algorithm, the weights are updated incrementally after each epoch. To reduce our cost function, we can take the partial derivative of J(θ) with respect to θ and equalize it to 0. The derivative of a function is nothing more than the change in output of a function caused by a small change in input.

$$\frac{\partial J(\theta_i)}{\partial \theta_i} \quad \text{where i=0,1,2, .. , n}$$

Because this model has n parameters, we have a gradient with n different partial derivative components. Each of these will appear as follows:

$$\frac{\partial J(\theta_i)}{\partial \theta_i} = -\frac{2}{T}\sum_{t=1}^{T} X(t)\left(y(t) - \Theta^T X(t)\right) + 2\lambda\theta_i$$

GD is an effective optimization algorithm that aims for a local or global minimum of the function. An update rule is each iteration of a GD determined parameter weight. This rule is determined by the prior weight, the learning rate (alpha) and the gradient of the function with respect to n. To solve the regression coefficient, the normal equation improves the loss function. When the model becomes complex, the solution speed becomes insufficient to find the regression coefficient. The optimization algorithm was changed to the GD method, and its recursive equation is:
It is basically renewing the values of θ_1 and θ_2 using the value of gradient, as in this equation 8. This algorithm is trying to find the appropriate weights by constantly updating them, keeping in mind that we are looking for values that minimize the loss function.

$$\theta_i^{n+1} = \theta_i^{n} - \alpha \frac{\partial J\theta_i^{n}}{\partial \theta_i} \qquad (8)$$

The equation depicts the steps for training our parameters (weights). This implies subtracting the derivative of the cost function with respect to the weight multiplied by the learning rate and updating each weight. We compute the derivative of each parameter with respect to the cost function for each epoch or iteration and take a step in that direction. This makes sure that we reach a minimum.
In the real world, it is not simple, because the learning rate can be too high or too low, causing the system to become stuck in local optima. We set an initial value of θ_0 to begin iterating in the direction of the negative gradient, moving from $\theta_n$to the next point θ_{n+1}, until the function's extreme point is reached (where the gradient value is 0). If we use the GD method to find the minimum value of the loss function of ridge regression, the expression of each round of iteration is simplified.

$$\theta_i^{n+1} = (1-2\lambda\alpha)\,\theta_i^{n} + 2\left(-\frac{\partial J(\theta_i)}{\partial\,\theta_i}\right)$$

The right side of this equation is simply the OLS update rule. The only difference between ridge and OLS is in the first term $[(1-2\lambda\alpha)\theta_i^{n}]$. Therefore ridge regression is equivalent to minimizing the weight by a factor of these multiple terms and then using the same update rule as OLS. The regularization causes coefficients to shrink. The GD as shown in the equation is dependent on the learning rate, which controls the step size.
The methodology for training our model is straightforward, with forwarding and backpropagation. Training simply includes updating our weight values on a regular basis. The graph for the linear regression model with gradient descent is shown in figure 16.

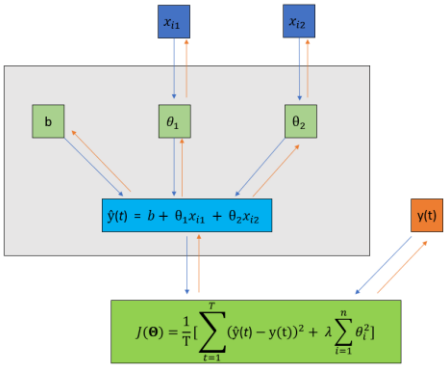


Figure 16. Computation Graph for Linear Regression Model with Gradient Descent

Here θ_1 and θ_2 are the weights of their corresponding features like x_{i1}, x_{i2} and b is a constant called the bias as illustrated in figure x. We can review this diagram from top to bottom for forwarding propagation and bottom to top for backpropagation.
Because GD is all about updating the weights, they should begin with some values which is known as initialising weights. The GD model sets these values to zero with np.zeros. Weights can be set up in a wide range of ways (zeros, ones, uniform distribution, normal distribution, truncated normal distribution, etc.) In this project, we initialised the weights by using zeros and the bias with zero.

We can distribute our dataset into equal sized smaller groups. Each group is known as a batch and it contains a number of examples known as the batch size. We should obtain the number of observations in our data when we multiply these two numbers. For example, the test1 dataset contains 26 examples, and because we set the batch size to 1 in this training, we get 26 batches in total.
Since we have the values of x_{i1}, x_{i2}, θ_1, θ_2 and b ready, we can compute $\hat{y}(t)$.

$$\hat{y}(t) = b + \theta_1 x_{i1} + \theta_2 x_{i2}$$

We can compare how far $\hat{y}(t)$ and y(t) data from each other by calculating the loss function $J(\Theta)$ as defined earlier.

Let's first compute all the partial differentials before adjusting the weights and biases (θ_1, θ_2 and b). These will be required later when we update the weight.

$$\frac{\partial J(\Theta)}{\partial \hat{y}(t)} = 2\{\hat{y}(t) - y(t)\}$$ Partial derivative w.r.t $\hat{y}(t)$

$$\frac{\partial \hat{y}(t)}{\partial b} = 1$$ Partial differential of $\hat{y}$ w.r.t. b

$$\frac{\partial \hat{y}(t)}{\partial \theta_1} = x_{i1}$$ Partial differential of $\hat{y}$ w.r.t. θ_1

$$\frac{\partial \hat{y}(t)}{\partial \theta_2} = x_{i2}$$ Partial differential of $\hat{y}$ w.r.t. θ_2

It should be noted that the partial differentials follow the values from the current batch.

Pay close attention to the path from the dark green node to the light green node from the figure 16. They collaborate their way up from the bottom. This is GD updating the weights with backpropagation and the gradient values.

$$b' = b - \alpha \frac{\partial J(\Theta)}{\partial b} \qquad \text{Gradient descent update for b}$$

Where, b is current value, b' is value after update, α is learning rate and $\frac{\partial J(\Theta)}{\partial b}$ is gradient (partial differential of L w.r.t. b). To obtain the gradient, multiplying the paths from L to b using the chain rule as following in the equation 9.

$$\frac{\partial J(\Theta)}{\partial b} = \frac{\partial J(\Theta)}{\partial \hat{y}(t} \; X \; \frac{\partial \hat{y}(t)}{\partial b} \qquad (9)$$

If we want to write both coefficients and intercepts from our model in one general equation, we can compute as follows;

$$\theta_i^{n+1} = (1-2\lambda\alpha)\, \theta_i^{n} + 2 \left(-\frac{\partial J(\theta_i)}{\partial\, \theta_i}\right)$$

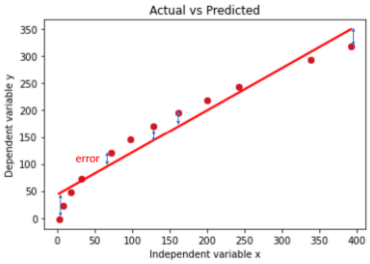
After dealing with the first batch, we must repeat the preceding steps for the remaining batches, namely examples 2 to 26 for the test1 experiment. When the model has iterated through all of the batches once, we complete one epoch. In practice, we extend the epoch beyond 1. When our configuration has seen all of the observations in our dataset once, we call this an epoch. However, one epoch is almost never sufficient for the loss to converge. This number is manually tuned in practice.

Nevertheless, there is one drawback when our data exceeds 1 million (noisy data), which will require more time to upload data and more computational power. We want to overcome it, we aim to use GD rather than a close form equation. We also used from sklearn library SGD, which takes one random data point and quickly updates the weights using that gradient, which has side effects such as a longer convergence time. It may not be practical in order to use the normal equation formula. Although the close form solution works in most cases, it becomes computationally infeasible to compute $(X^T X + \lambda I)^{-1}$ in some cases, such as when the number of features (value of n) is large. Because the inverse operation has an $O(n^3)$ runtime complexity, it becomes computationally infeasible for large values of n. In some cases, due to a mathematical property known as a singularity, $(X^T X)$ becomes non-invertible. As a result, the equation's value can not be calculated mathematically. It could happen in scenarios where many more features exist than the number of observations ($T \leq n$).

Since there is the advantage of using an iterative algorithm like GD over a closed form solution in general we implement the algorithm that we described before section 4.7.2 in python. Because our data was limited, we used our own GD approach in python. Following that, we continue to use python sklearn.linear model.Ridge and SGDregressor library, which employs linear least squares with L2 regularization.

4.8. Evaluation of Gradient Descent Algorithm

Since it was really challenging to directly implement the model to the estimation task therefore before trying our model in our estimation, first we decided to check whether our multivariate regression approach is working or not to do that we created two input features and a true output value. When we compare the actual and predicted values, it is observed that they are close enough to each other which means their predicted values are good. When we tried directly to our ridge regression model in estimation before, there were very significant error differences between the actual and predicted values. Therefore we decided to implement a simple example of our model by creating random values and observing at that time we need to clean our dataset more. Because the model indicated to us when we had a very significant difference between input samples (features) and output labels (actual value) because we developed these datasets specifically to test it.




```
X train shape: (3, 2)
X test shape: (12, 2)
MSE test:                    1503.3152396886708
RMSE test:                     38.77260940004774
R2 score test:               0.9033921187784416
    Actual Value  Predicted Value  Difference
0              2        -2.138933    4.138933
1             72       120.867067  -48.867067
2            128       170.069466  -42.069466
3            162       194.670666  -32.670666
4            392       317.676665   74.323335
5             32        71.664667  -39.664667
6              8        22.462267  -14.462267
7            338       293.075465   44.924535
8            200       219.271866  -19.271866
9             98       145.468266  -47.468266
10           242       243.873066   -1.873066
11            18        47.063467  -29.063467
```

Figure 17. Gradient Descent Algorithm Testing Evaluation

We observed from figure 17 the actual value versus our predicted value and the differences among them. As RMSE is 38.77 and R2 score is 90% which is in the case of 12 test and 3 train splitting. Since the R2 scores is close to 1 which presents our model is good. 90% of the variation in the y values is accounted for by the x values.

4.9. Ethical Considerations

When capturing data about people, we should keep two major concerns in mind: privacy and security. Data collection is a controversial topic, particularly since the implementation of the general data protection regulation (GDPR) in Europe. In this project, we are only interested in obtaining the total number of customers rather than tracking any individual customer or person's movement through a store. The protection of privacy has been newly followed in compulsory laws like the data protection law of the EU by declaring MAC addresses as personal data. Many device manufacturers introduced MAC randomization as a preventive privacy method.
Since our collection method is passive, we do not try to decrypt any data or take any active measures to stimulate or alter normal network behaviour while outside of our lab environment. We do not observe personally identifiable information since we limit our analysis to 802.11 management frames and unencrypted packets. Even though we analyse IP addresses, we do not use these layer 3 addresses in our experiment. Even with an IP address, there is no way to match the address to a specific person. There is no way to map MAC addresses to specific individuals. Finally, in terms of beneficence and respect for people, our work stands with no risk of destroy, while the opportunity for network measurement and security gives a general advantage. To sum up, our experiments were ruled ineligible for person subject research.

5. IMPLEMENTATION

5.1. Occupancy monitoring in indoors using Wi-Fi Probe Requests

This chapter describes how the system's components were carried out. Occupancy monitoring is essential because it can be used for things like energy management, surveillance, and security. We passively capture PR messages using Wi-Fi Raspberry Pi equipment and the information is used to compute the occupancy count at various time intervals.
The implementation is divided into two parts: (a) the data collection device and (b) the data processing device. After this chapter , it'll be clear how the system was applied for testing and validation, which will be discussed in the following section.

5.1.1. Hardware

Formerly, we aimed to utilize a Raspberry Pi Model 3 B+[2] as seen in the figure 18, which we already had, and then purchase supplementary Wi-Fi dongles that could be used in monitor mode. We then used Raspberry Pi (RPi) 3 Model B +, which has a 1.4GHz Quad-Core Broadcom BCM2837B0 processor and a 2.4GHz IEEE 802.11b/g/n/ac wireless LAN. The Raspberry Pi 3 Model B+ outperforms the Model B in many aspects, including a faster CPU speed (1.4 GHz vs. 1.2 GHz), risen Ethernet throughput, and dual-band WiFi.
The Raspberry Pi is a small, inexpensive computer the size of a credit card that connects to a computer monitor or TV and operates with a regular keyboard and mouse. With the help of this capable little handset, people of all ages can learn about computing how to program in Scratch or Python. 32 GB microSD card was also used as mass storage. After further research, however we installed the off-the-shelf Kali Linux OS on the RPi. The in-built Wi-Fi module supports monitor mode, so an external USB WiFi adapter is not needed.
As we were already planning to purchase additional adapter, for that reason we did not purchase an additional adapter. With this advancement, we were able to lower the cost of the units we needed to purchase. It was decided to use the Raspberry Pi Model 3 B+ because it meets many of the requirements: it is relatively inexpensive, has outstanding online support, has Wi-Fi capabilities, and we can set it up as a monitor without purchasing any additional equipment. Only PRs were logged to persistent memory, while the received data frames were collected using tcpdump.
The resulting dumps were sent to an external computer and transformed to .pcap files (packet capture) which Wireshark can open. Each PR message contains the following information: the source MAC address, the OUI that identifies the radio chip vendor, the RSS of the collected probe packet, and the probe frame timestamp.

Figure 18. Raspberry Pi (RPi) 3 Model B +

[2] Raspberry Pi Model

5.2. Ground Truth Management

We created python scripts to manage the ground truth and collect it in the interest area. The quantity of people in the room is written on each dataset csv file in a folder, such as 5 people v1, which indicates the data collected at a specific time as previously dedicated with an actual 5 people in the room and v1 is the first sample in 2 minutes. It then saves this data as a ground truth value for model estimation.

5.3. Choice of Operating System

Kali Linux[3] is a Linux distribution based on Debian that is maintained and funded by offensive security. Kali Linux includes hundreds of tools for various information security tasks like penetration testing, security research, computer forensics, and reverse engineering. Kali Linux is a multi-platform solution that information security professionals and nonprofessionals can use. We chose the Linux distribution Kali because it already includes a kernel (Re4son-Pi-Kernel)[4] that enables us to use the Raspberry Pi in monitor mode.
Therefore a kernel with the firmware patch was used. This produced an image that can be captured using the additional new wireless interface wlan0mon. Kali Linux comes with a number of useful tools that are ready to use. A complete list of Kali's tools[5] can be found at the website. We're especially interested in a few wireless-specific tools, as shown in the table. Airmon-ng is used to allow wireless interface monitor mode, and tcpdump is used to capture Wi-Fi packets [44].
List of wireless specific tools is introduced in the table 3. The Airdump-ng can record raw 802.11 frames to a capture file. Wireshark[6] can then be used to analyze this file. The airdump-ng is used to list all networks in our nearby area and display valuable information about them. Because it is a packet sniffer, it is designed to capture all of the packets that surround us while we are in monitor mode. We can operate it against all of the networks in our proximity to obtain useful information such as the mac address, channel name, encryption type, and number of clients connected to the network before aiming the target network. We can also perform it against a specific AP to only collect packets from a specific Wi-Fi network.

Table 3. List of Wireless Specific Tools

Tools	Description
Airmon-ng	Airmon-ng[7] is used to allow and deactivate monitor mode on wireless interfaces.
Tcpdump	Tcpdump[8] is a robust command line packet analyzer.
Wireshark	Wireshark is a network traffic monitoring software that operates on a network interface

[3] KaliLinux
[4] Re4son Pi Kernel
[5] KaliTools
[6] Wireshark
[7] Airomon-ng
[8] Tcpdump

5.4. Testbed

The first of a four-story office building in Aalborg University's connectivity section was chosen as a testbed. We concentrated on a single lab guest office with a floor space of about 20 m2. This project's data collection period is on June 2022. Every two minutes, data on occupant counts was captured. The Xiaomi mi lite 8 mobile phone was used to collect ground truth data at the lab's entrance. The camera's occupant counts would represent as the ground truth data. We manually count the net number of people passing through each entrance to validate the camera's measurement accuracy. Further, we did another test in the canteen with more crowded space.

5.5. Data Capturing

Tcpdump is a command-line packet analyzer for capturing network traffic. While it appears standard with many UNIX systems, we also include configuration command in our installation script to ensure compatibility with lightweight Linux distributions. Tcpdump can not only publish network traffic to the console, however it can also save it to PCAP files. The first method was to capture Wi-Fi packets using airodump-ng. Airodump-ng can record raw 802.11 frames to a capture file. Wireshark can then be used to analyze this file. Tcpdump and airmon-ng were used in combination, and the configurations are included in the appendix. Aside from that, the figure 19 shows IEEE informations details.

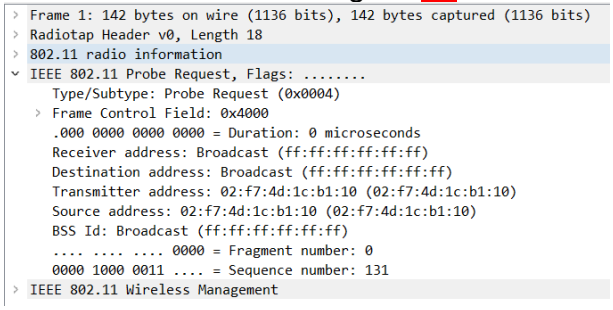


Figure 19. IEEE Informations Illustration via Wireshark

We discovered raw 802.11 packets frames in Kali with some configuration[9]. Interface configuration in the appendix tcpdump functions do the following;

1. check if the interfaces support monitor mode or not
2. set to monitor mode the interfaces that supports it;
3. in our case we created wlan0mon wireless interface in wlan0 with the help of airmon-ng tool which enables packet capturing and exporting data to text file as well without this tool configuration we were not able to see IEEE informations. Basically the airmon-ng software to start monitoring on the wlan0mon connection
4. specifying the channel 6 to the monitor interface that helps us to capture raw 802.11 frames from the chosen channel 6

We collect in monitor mode because we are interested in radio layer information about packets. We plan to set up Wi-Fi interfaces and capture pcaps of Wi-Fi management messages on fixed channel 6, which is in the center of the Wi-Fi spectrum and one of the non-overlapping channels. The reason for selecting a fixed channel is that it receives more packets [21].
The system is designed to produce a dataset containing radio interference captures of devices in space. It can collect Wi-Fi packets through interfaces that support monitor

[9] Appendix A

mode. In monitor mode, the card can listen to all packets that pass by. The mode of wireless devices is set to managed by default, which means that the wireless device will only capture packets with our device's MAC address as the destination MAC. It will only collect packets that are intended for my Kali machine. It's also worth noting that the interface's name has changed from wlan0 to wlan0mon as seen in the figure 20.

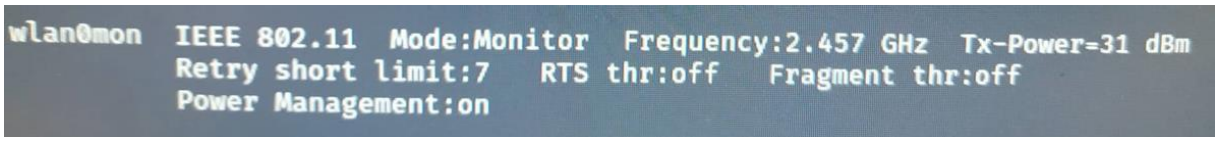


Figure 20. Interface in Monitor Mode

Regarding this configuration, we capture PRs every 2 minutes and save them in a pcap file in wireshark, which is then converted to a csv file and saved in a single folder for uploading in a python script. While collecting data, everyone turned on the Wi-Fi network, but we had no idea if they were associated or not. The data file includes Wi-Fi channel messages in PCAP format as well as CSV tables. Various recording were made for different cases (for example, a file named 5 people v1 which indicates 5 number of people as an actual data in this case) as seen in the figure 21. The python code will accept the first integer as a ground truth. Files were collected in the Connectivity lab at Aalborg University in an area with no specific shielding.

Figure 21.Captured Data with Tcpdump

The Wireshark[10] OUI lookup tool as seen in the figure 22 makes it simple to find OUIs and other MAC address prefixes. It makes use of the Wireshark manufacturer dataset, which is a collection of OUIs and MAC addresses compiled from various sources. One example is seen 28:a0:2b MAC of first 3 octect addresses to the Apple company.

Figure 22. Wireshark OUI lookup

The wireshark screen is divided into three sections: the packet list pane, the packet details pane, and the packet bytes pane. The 802.11 contains only the 802.11 information, whereas the 802.11 radio tap header contains the 802.11 information as well as some additional radio information provided by the WLAN interface driver. In Kali Linux, we receive 802.11 headers in monitor mode, as shown in the figure 23.

[10] Wireshark OUI lookup

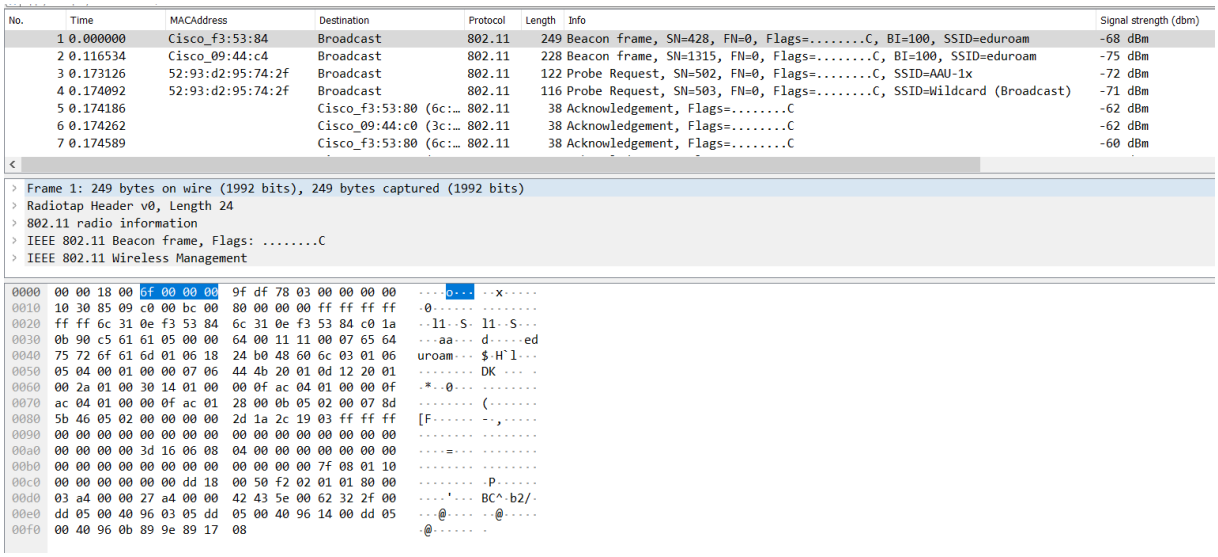


Figure 23. Wireshark GUI

The captured data from the Wi-Fi sniffer is captured and loaded to the computer aimed for evaluation in the form of a CSV file. The data sets are then labelled with the ground truth value at the start of the CSV file that contains the data. The data sets must then be combined into a single large folder. We observed 74 different Wi-Fi mac addresses for the case of 7 people in the room, with 495 of Cisco f3:53:80 address among each different mac address during test1 case. This means that we had the same Cisco f3:53:80 MAC address 495 times. Which meant that we tested the model's performance with a large number of observations python script has been implemented for that[11].

5.6. Packet Analyzing

Wireshark is a popular network sniffing tool with a GUI for decoding many protocols and filters. Wireshark is a network traffic monitoring software that operates on a network interface. It is now the most widely used network management software [21].
Tcpdump is also a popular network analysis tool because it combines simplicity and effectiveness in a single interface. It is a free and open-source network functionality licensed under the BSD license. Tcpdump makes use of a commandline interface (CLI) to provide packet content concepts in a variety of formats depending on the command. In Linux, tcpdump is a network packet sniffer. Essentially, it listens on an interface and dumps any packets that pass through it.
PCAP is an useful tool for file analysis and network monitoring. Wireshark and other packet collection software assist you in collecting network traffic and converting it to a human-readable format. This pcap file can be established on any device by capturing files on that system, sharing them with another device, and analyzing the captured packets from this pcap file. Both tcpdump and Wireshark can read pcap files.
Although wireshark appears to be more efficient than tcpdump in regards to efficiency, tcpdump is preferred for its speed in capturing packets. Tcpdump's performance accuracy is ideal for rapid scans and packet capture. However wireshark is always the first choice for complex scans.
The reason for using tcpdump instead of wireshark to capture is that capturing requires administrative privileges. For example, if you run wireshark as sudo, all of its other features will be in administrative mode or at a higher privilege level. As a result, there may be some security benefits to running tcpdump as pseudo and then loading the

[11] Python script for virtualization MAC addresses

captures into wireshark once the captures are complete. Eventually, Wireshark was used to analyze the data, but tcpdump was used to collect it.
To read and manage the data in the analyzer, we used the Pandas library. Pandas DataFrame is used to clean the data and calculate the number of devices. The Sklearn library is used to analyze the cleaned data and train the regressor with the help of the ground truth. The ML component is implemented using this library.
Ultimately, we created a script to compute metrics, such as error information, and graphs, such as histograms, error distribution, and comparison of detected devices and people estimated with the actual number of people present.

6.EVALUATION

6.1. Comparison of our model and Python sklearn models

In this section we will demonstrate the evalutaion with different model has been used to do this we will use the test1 testbed initially. The first experiment is conducted in an indoor university laboratory. The environment has an area of about 20 sqm and is characterized by 9 fixed seats and a daily number of occupants between 1 and 7. The measurement campaign lasted for several days, and a total of 26 ground truth observation were collected. Since the laboratory focus is on wireless communication technologies, with daily activities on wireless device testing and experimentation, a great amount of background noise in terms of Wi-Fi packets is present in the dataset. The residents were also leaving time to time from the laboratory. In addition some of them were carrying more than one smartphone.

Also in this section we will be performing linear ridge regression both our model and sklearn libraries in Python. We builded the algorithm that has been introduced in the section 4.7.2 close form equation and GD algoritm. The used scikit learn libraries are Ridge with cholesky, Ridge with sag, SGDRegressor and lastly SGDRegressor with avarege SGD and our own model for the first experiment. Here in ridge library, the ‘cholesky’ uses the standard scipy.linalg.solve function to achieve a closed form solution and the ‘sag’ uses a stochastic average gradient descent. We used our model and sklearn.linear model for evaluation. The Python models were created to illustrate multilinear ridge regression analysis.

For the random state working principle, we used Scikit-train test split() and LinearRegression() functions. We used the train test split() function to divide the dataset into train and test sets . By default, the function shuffles the data before splitting (with shuffle=True). The shuffling process is controlled by the random state hyperparameter in the train test split() function. When we obtained different train and test sets across different executions with random state=None and the shuffling process was out of control. We kept random_state = 0 for each experiment.

We got the same train and test sets across multiple executions when random state=1. We got the same train and test test sets across different executions with random state=2, but the train and test sets are different from the previous case with random state=1.

The train and test sets have a direct impact on the model's performance score. Since the train test split() function returns different train and test sets with different integer values for random state, the random state hyperparameter indirectly influenced the model's performance score.

We got three different RMSE values for the model depending on the integer value used in the random state hyperparameter. That brings us an important discussion that which value we will accept as the correct RMSE value. We will not accept any single value. Instead, we got the average of these RMSE values. It is better to re-run the code as many times as possible for example 10 times and get the average RMSE. We did automaticaly this with Scikit-learn cross_val_score() function instead of manually.

Depending on the problem and the dataset, we can use a different algorithm to determine the learning rate. The most basic algorithm is to choose a constant, such as 10e-3.

We begun by using it and gradually increase or decrease it based on how quickly or slowly GD finds the minimum.

The StandardScaler() function in the python sklearn library allows us to standardize data values into a standard format. We begin by constructing an object of the StandardScaler() function. Furthermore, we use fit transform() in conjunction with the assigned object to transform and standardize the data.

To avoid overfitting we splited the data into the training set and the testing set. We train our model on the training set first, and then use data from the testing set to assess the accuracy of the resulting model. Studies have shown that using 20-30% of the data for testing and the remaining 70-80% for training yields the best results. Therefore we chose the train size and test size with 80% and 20% respectively.

The same factor lambda is applied to all weights. The hyperparameter lambda can be used to control the strength of regularization. There are different cases for tuning lambda values such as ridge regression equals linear regression in the case of the lambda is set to be 0 also all weights are reduced to zero if lambda is set to infinity. As a result, we should choose lambda value in between 0 and infinity.

The learning rate for GD is an important hyperparameter to set when training a model. This parameter, as previously stated, scales the magnitude of our weight updates in order to minimize the network's loss function. If learning rate is too low, training will move very slowly since of making very small changes to the weights in the model. However, setting learning rate too high can result in undesirable divergent behavior in the loss function. These examples are depicted below. A small learning rate requirements many updates before getting the minimum point. The optimal learning rate swiftly reaches the minimum point. A very high learning rate causes extrem updates that points the divergent performances.
We moved through a complete ML pipeline. We begun by loading and displaying the information from which we will be learning, while also performing exploratory data analysis (EDA). The data then pre-processed, and models built to fit it. This method is then evaluated and if it is satisfactory we used it to estimate new values based on new input. The dataset contain 18 features input variable and 1 target variable. Since the data inputs contains very large number without cleaning and recevied ValueError: Input contains NaN, infinity or a value too large for dtype('float64') in our model implementation therefore for 18 features we scaled the input samples.
In the experiments we subdivided power levels in order to increase the number of feature as following which previously was considering only -70 dBm and higher power levels as seen in the table 4. Apart from that at each power level range we have 2 feature (random and non random) same as previously. We did not consider above -75 dbm since above this range signals are not probably coming from outside because the size of the room is around 20 meter square.

Table 4. Features Selection with Power Thresholds

Power level	Random Feature	Non Random Feature
$\alpha \leq -75$	X	X1
$-75 < \alpha \leq -70$	X2	X3
$-70 < \alpha \leq -65$	X4	X5
$-65 < \alpha \leq -60$	X6	X7
$-60 < \alpha \leq -55$	X8	X9
$-55 < \alpha \leq -50$	X10	X11
$-50 < \alpha \leq -45$	X12	X13
$-45 < \alpha \leq -40$	X14	X15
$-40 < \alpha \leq 0$	X16	X17

Python[12] file is for our new experiment has been created. The dataset details can be reached out by following file[13] that we had the grouth truth value between 1 and 7.
In order to decrease the underfiting we increased the number of features to the 18, tried to remove some noisy data as described in the section 4.6. Since when number of iteration increase the underfitting reduces we pick the value for that 1000 for all the time as generally used value.
The main focus in our work was avoiding the overfitting as we describe in the section for that ridge regularization method was used. It depends on the training data size is high or not, we increased the training data size to %80 in order to reduce the overfitting. This is the one of our reason as well to chose high training set.
In this section we will represent the data analysis with test 1 experiment other experiments result includes same plots simply updating the csv files for each different test the folder named[14] will show the data analysis. The Python script contains several plots, including how the records are distributed for each column and how the actual output is obtained by plotting with the features. We begin by analyzing exploratory data. We should first familiarize with your data, which contains loading it in, displaying features, examining their relationships and formulating hypotheses based on the observations. The dataset is a CSV file that includes the features as well as the real output obtained as a ground truth value.
We loaded the data into a data frame using pandas and peek at the first 5 values using the head() method as seen in the figure 24. The shape of our dataset is (26,19) checked by df.shape property. We have 26 rows and 19 columns that's 26 entries containing a 18 features and 1 real value.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	feature1	feature2	feature3	feature4	feature5	feature6	feature7	feature8	feature9	feature10	feature11	feature12	feature13	feature14	feature15	feature16	feature17	feature18	real
2	38	226	368	123	707	491	657	2582	97	2272	3	12	0	0	0	3	0	1	1
3	23	221	256	81	445	462	200	1253	87	15	6	5	1	1	0	0	5	0	1
4	229	683	152	682	917	1284	1556	3160	596	338	0	12	0	0	0	0	17	15	1
5	26	275	188	551	353	209	1956	2273	534	996	9	34	0	2	0	1	0	77	1
6	5	223	23	114	110	245	59	22	21	32	2	1	0	0	0	0	0	19	1

Figure 24. First 5 values of Data Frame of Test1

Correlations among numeric values in a DataFrame are calculated and displayed using the corr() method. This implies that the predictors in the regression are unrelated to one another. To see if there is any correlation between our predictors, we can use the corr() function from pandas dataframe to compute the pearson correlation coefficient between each column in the dataset. In this figure 25, the first feature has a 1.0 (100%) correlation, while the second feature has a 100% correlation to the second feature, and so on. Any

[12] Test1 evaluation python script
[13] Dataset for test1
[14] Plot test 1

variable will have a one-to-one mapping to itself. The correlation between the first and third features, on the other hand, is 0.29. Being closer to -1 or 1 is considered a strong positive correlation. A high linear correlation indicates that we will be able to predict the value of one feature based on the value of another. We also used seaborn's heatmap() to identify the strongest and weakest correlations based on (warmer) red and (cooler) black tones

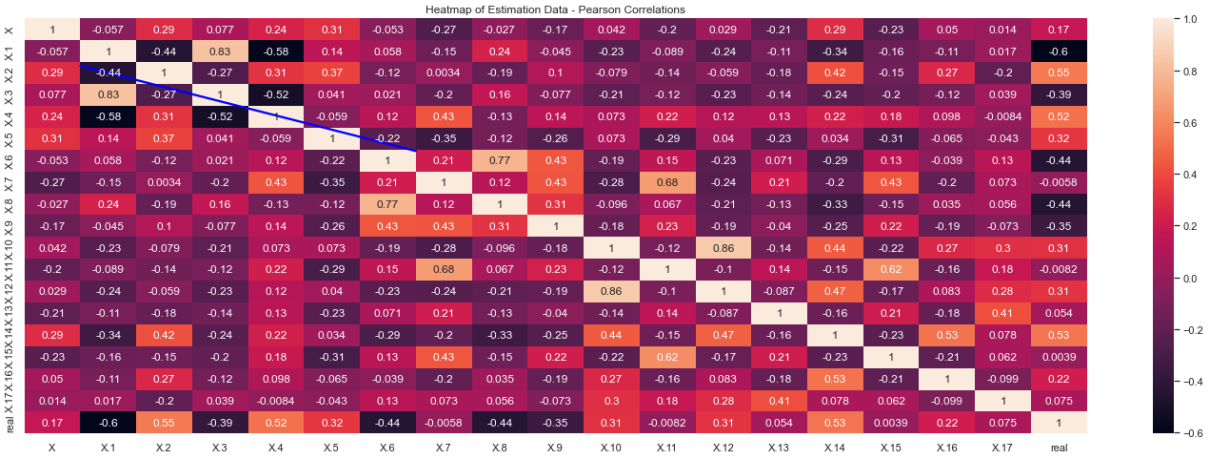


Figure 25. Pearson Correlation Coefficients

We can observe the pairwise correlation between all variables after plotting the correlation matrix and color scaling the background. We also included the dependent variable real here. This is because selecting the independent variables to include in the model is a secret trick for us. In the case of uncertainty about which variables to include in the model, simply created a correlation matrix and chose the independent variables that have a high correlation with the dependent variable.

Back to the multicollinearity problem, it is clear from the correlation matrix that is quite a few variables are correlated to each other with higher value. There is one pair of independent variables with 0.86 correlation which are X10 and X12. "X10" and "X12" columns are highly correlated which might cause multicollinearity therefore we may remove that one of them however it can cause the remove of the actual MAC address and may give us more error at the end for that reason we leave number of feature with 18.

In order to improve the accuracy more feature selection can ben improved more in depth. Although we already try the model with 2 number of features we did not choose them in our model. The reason behind that is with 2 features we were not considering the power range, at the beginning we were only setting the power threshold with some value. Afterward we decide to increase number of features to 18 as describe in the table 4 because of the underfitting issue.

When dealing with a high dimensional dataset, it would be inefficient to use all of the variables because some of them may be presenting unnecessary information. We would need to choose the right set of variables that provide an accurate model. It should be considered the variables that chosen should not be correlated with one another.
Using the describe() function to examine the min and max columns of the describe table, we can observe that the minimum value in our data is 0 and the maximum value is 1264. This means that our data range is very large that implies our data variability is also high. Furthermore, it is observed that the means are significantly different from the standard deviations by comparing the values of the mean and standard deviation columns, such

as 101,42 and 136.25, 256.15 and 333.21, and so on, That implies our data is far from the mean which adds to the variability.
We already have two indications that our data is spread out, which is not in our favor, since it makes it more difficult to have a line that can fit from 0 to 9226 in statistical terms, to explain that variability. We improve this by scaling. Because linearity states a linear relationship between the dependent and independent variables. In our case since we have multiple independent variables, we can do this by using a scatter plot to see our predicted values versus the actual values as seen in the figure 26.

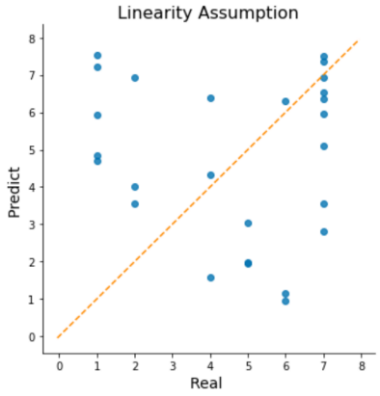


Figure 26. Linear Relationship Between the Actual and Predicted Value

As seen in the figure 27(a) and 27(b) from dataset with 1 actual people counted case show us the observations with respect to power level for the non random and random MAC addresses. We had 5 sample for 1 actual people for this experiment and plotted for each of them seperately. The left and right figure is for random observations and non random observations are presented respectivaly with respect to power levels. The most observations are observed between -70dbm and -50dbm in this case with the help of python script[15].

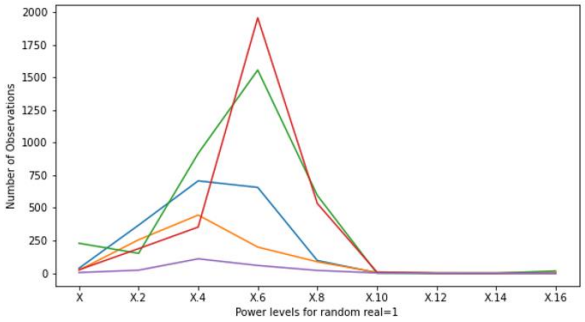


Figure 27 Figure 27 (a). Random Number of Observations

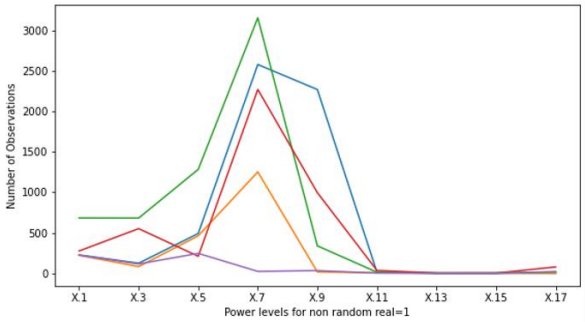


Figure 27 (b). Non-random Number of Observations

We loaded some python libraries we will be using, such as pandas, numpy, matplotlib, sklearn, etc. We will also load our dataset into a dataframe by using the pandas library. Our dataset consist of 19 columns with 18 features and actual value and 27 rows. We will try to predict the number of people depending on its features.

After setting our X and y sets, we can divide our data into train and test sets. We will be using the same random state 0 and 80% of our data for training. After splitting the data, we can train our multiple regression model. To train our model we can execute the same code as before, and use the fit() method of the ridge.
After fitting the model and finding our optimal solution, we can also look at the 18 coefficients of the features in the test1 experiment. In the multiple linear regression model,

[15] Python script for plotting random and non random number of observations

we have 18 variables and 18 coefficients. Those coefficients meaning that following the same interpretation of the coefficients of the linear regression, this means that for a unit increase in the first feature, there is a decrease of -0.019 in real value since weight 1 of the test1 is depicted -0.019 from our GD algorithm result for more informations we will see the comparison of weights in figures later.
The final step of preprocessing is standardizing. It is essential to bring the values of each predictor to a similar scale. Otherwise, some columns will be dominating over others. For this context, StandardScaler() is used where the columns are scaled in respect to their variance. It is an essential step before applying ridge regression.

6.2. Evaluating the Multivariate Model

In this project, we used a few libraries: pandas to read the dataset file, sklearn.model selection to seperate the training and testing datasets, and matplotlib to draw the regression line. Following that, we loaded our CSV file to discover the dataset by using pd as a pandas reference variable and calling the read csv() function with the file name. Before splitting the dataset into training and testing datasets, we must first identify the dependent and independent variables. When these variables are ready, we can begin splitting up the dataset.
Then we fited our x and y into a ML model to predict the output, however before that, we have to import the Ridge from the sklearn.linear_model and create an object of the Ridge class.
We will compare the result in different scenario with the behavor of the lamda, learning rates. After exploring, training and looking at our model predictions our final step is to evaluate the performance of our multiple linear ridge regression. We want to understand if our predicted values are too far from our actual values or not. We did this by calculating the RMSE metrics. Our RMSE value (using GD method) is 1.25 that means that our model might get its prediction wrong by adding or subtracting 1.25 from actual value. Below figure shows that the result after tuning parameters and achieved best RMSE by taking into acount of the L2 term and learning rate. In this work we did not tune the iterations, we fixed it always with 1000. The model parameters are setted as test size=0.2, random state=0. In order to get not different result each time that is because dataset splitting is random by default therefore random state is setted to the zero. After running the function, the output varies. The dataset's samples are shuffled at random and then divided into training and test sets based on the size specified.
The comparison of the models illustrated in the table 5. The model we build by using GD achieved lowest RMSE value as 1.25.

A line of best fit is a graph that depicts the general direction that a set of points appears to follow. The main goal of line of best fit is to get our predicted value closer to the actual value. The better the fit, the higher the R^2 value. The testing data yields a higher coefficient in this case as seen in the figure 28 higher the R2 with higher the weights.

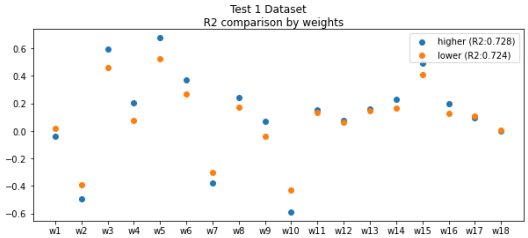


Figure 28. R2 Score Comparison with Weights

We observed that performance of our model with in given excel[16] file is good on test set compare to the train set, it is not creating overfitting or underfitting problem (train RMSE: 1.17) that much but the error might be decrease by tuning more in the future. We observed that since during our experiment it was noisy environment, there were some people with more than 1 smartphone and some people were moving out time to time therefore the error 1.25 is quite expected.

The illustrations in the figure 29 represented for our model (figure a) and sklearn.linear_model libraries such as sgdregressor (figure b), average sgdregressor (figure c), ridge cholesky (figure d) and ridge sag (figure e) respectively in best of line. For example, the graphs below demonstrate two sets of simulated data: The observations are shown as dots. The model's predictions (the line of best fit) are shown as a different color line. The distance between the observations and their predicted values (the residuals) are presented.

It is known that the larger R^2 and the smaller RMSE show the better performance of models. We observed that when the R^2 is high, the observations are close to the model's predictions. In other words, most points are close to the line of best fit. The model we build with GD has quite high R^2 value that means the our model prediction is better than the other models that been used.

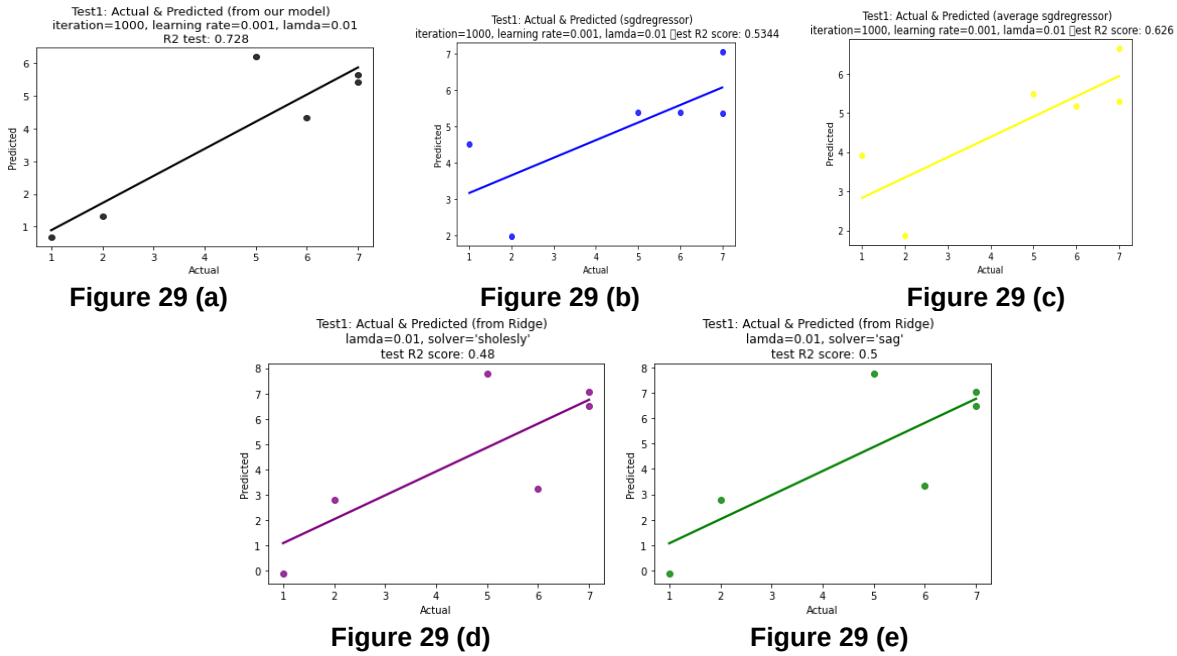


Figure 29 (a) **Figure 29 (b)** **Figure 29 (c)**

Figure 29 (d) **Figure 29 (e)**

Figure 29. Illustration of 5 Different Models in Best of Line

Because the scatter plots demonstrate residual points evenly distributed around the diagonal line, we can assume that our independent and dependent variables have a linear relationship. Since our model RMSE error reached minimum value compare to the other relevant model, it is obvious that from the figure 29 of residual points are more closer to the diagonal line as expectedly.

To use the sag solver from Ridge sklearn library, we do not set the step size (learning rate) since the solver computes the step size based on your data and alpha. The step size is set to 1 / (alpha_scaled + L + fit_intercept) where L is the maximum sum of squares for over all samples. While in the random state hyperparameter is shuffled we obtained

[16] Excel file for test1

significantly different RMSE values for the model depending on the integer value used. We can not accept any single correct RMSE value. Instead, we used the average of these RMSE values. It is better to re-execute the code as many times as possible (e.g. 10 times) and get the average RMSE. Doing this manually is time comsuming. Instead, we can automate this with sklearn cross_val_score() function by setting cross to the 10 and then take average of 10 RMSE values and achieved following result in the case of use of ridge with 'sag' as seen in the figure 30. As a result among 10 different RMSE values average RMSE 1.64 is obtained.

```
RMSE values:  [2.45 1.59 3.57 2.07 0.75 1.1  0.54 1.35 2.4  0.61]
RMSE average:  1.64
```

Figure 30. Average of 10 RMSE Value

6.3. Model Parameters

After building our model, we reached few significant values from our model. Those values are the coefficients and intercept values of the models. After spliting the dataset we used the training set to fit the model. Ridge/sgdregressor creates the object that represents the model that fits the model and returns it. With ridge regression, fitting the model means determining the best intercept (model.intercept_) and slope (model.coef_) values of the regression line.

The intercept value is the estimated average value of our dependent variable when all of our independent variables values is 0. In our case this means that in the case all features are 0 we will have about 3.935 real value as seen in the figure 31.

```
Trained Weights         [-0.036 -0.492  0.592  0.201  0.676  0.369 -0.377  0.241  0.07  -0.589
  0.156  0.076  0.159  0.231  0.493  0.197  0.095  0.   ]
Trained bias         3.935
```

Figure 31. Model Parameters

The value of θ_0 is approximately 3.935. This illustrates that model predicts the response 3.935 when x=x1=x2,..,=x17 = 0. An increase of x by 1 yields a rise of the predicted response by 0.036. Similarly, when x1 grows by 1, the response decreases by -0.036 and so on so far.

We have 18 coefficients (weights) that represent the relationship of our independent variable to the dependent variable, where a change of exactly one in the independent variable changes the value of our dependent variable by the same amount as the coefficient.

When using ridge regularization, we must choose the optimal penalty coefficient, represented by lambda (alpha in the sklearn library). The penalty for coefficients becomes stronger as lambda increases as shown in figure 32. This represents the coefficients getting smaller (shrinking) when lambda decreases. The smaller the value of lambda, the higher would be the magnitude of the coefficients[17].

[17] Python script for weights comparison with lamda

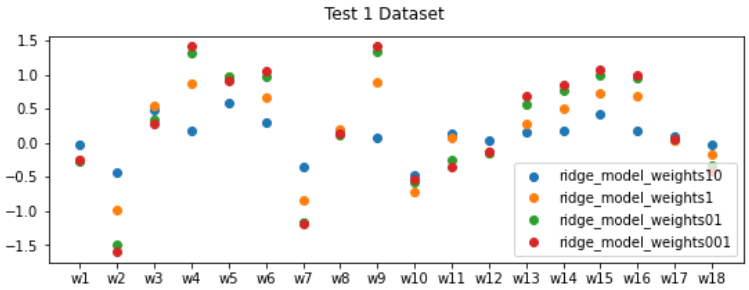


Figure 32. Weights in Change with Lamda Value

For the stopping criteria, the tol (tolerance) is used. This parameter uses the stopping criterion for iterations. This instructs scikit to stop looking for a minimum (or maximum) once a certain level of tolerance is reached when it close enough. The value of tol will vary depending on the objective function being minimized and the algorithm used to find the minimum.

In the case of SGDregressor we examined the tolerance value for minimum RMSE for that we tune the tol parameter, in the case of different lamda and learning rate in different tolerance value as seen in the excel file[18] we found the minimum 1.772 RMSE value with lamda=1 and learning rate= 0.01 and plot in the other tol value as well and observed that when tol value is decreases our RMSE value also decreases in this experiment. We stoped when the norm of the gradient is below some threshold[19] (tol) is 0.0001 to avoid overfitting as seen in the figure 33.

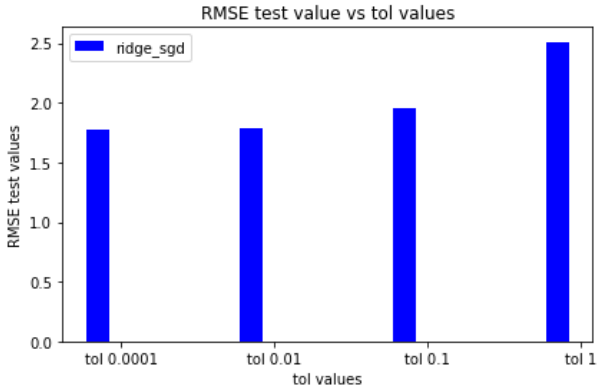


Figure 33. RMSE Value Changes with Tolerance Values

6.4. Model Validation

It is critical that we validate the model's performance after we have built it. We can assess a model by examining its coefficient of determination (R^2) as we discussed before. The coefficient of determination is the percentage of total variation in the dependent variable explained by variation in the independent variable. R^2 values are calculated as follows in the equation 10.

$$R^2 = 1 - \frac{\sum(\text{output}_{\text{actual}} - \text{output}_{\text{predicted}})^2}{\sum(\text{output}_{\text{actual}} - \text{output}_{\text{mean}})^2} \qquad (10)$$

[18] https://github.com/mfatihkoc/master_thesis/blob/main/tol_rmse_sgd.xlsx
[19] Python script for RMSE value changes with tolerance values

where $\text{output}_{\text{mean}}$ is the mean of the actual value of output variable for all data points in the training or testing datasets. The indications of other parameters ($\text{output}_{\text{actual}}$ and $\text{output}_{\text{predicted}}$) are the same as the corresponding ones in equation 3.
Although using input and output trains to test goodness of fit is possible, it is not recommended. Test data is used to generate an unbiased estimate of the predictive performance of the model.
In this case, we achieved the R^2 value is 72.8% for testing and 75.9% for training as seen in the figure 34. R^2 ranges between 0 and 1, where R^2 =0 means there are no linear relationship between the variables and R^2 =1 shows a perfect linear relationship. In our case, we obtained R^2 score about 0.728 which means 72.8% of our dependent variable can be explained using our independent variables.

```
R^2 train: 0.759, test: 0.728
```

Figure 34. Obtained R2 Score Values in Testing and Training Sets

Noting that value of lamda which is hyperparameter of ridge model means that they are not automatically learned by the model instead they have to be set manually.

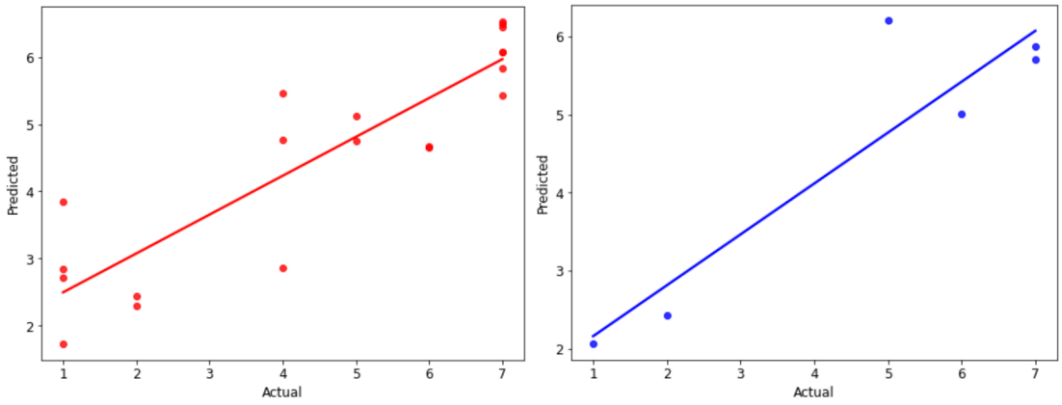


Figure 35. Predicted Performance in Training (a) & Predicted Performance in Testing (b)

The red dots represent the features and real output pairs used for training as seen in the above figure 35. The red line, known as the estimated regression line, is defined by the intercept and slope of the model fitting results as seen in the figure 35(a). That means that it is reflecting only the positions of the red dots. The blue dots show the test set that we splitted. It is used to predict the performance of the model with data not used for training as seen in the figure 35(b).

Here is the regression line are fitted in to the actual and predict number of people plot. In this experiment we tried to minimize the errors with different models. We measured the residuals by implementing the equation 7 with root of the sum of squared of residuals (RMSE) with L2 term (ridge regression cost function).
If the regularization coefficient is too small, the regularization term has no effect on training, and thus overfitting may occur. If the regularization coefficient is too large, the minimization decreases the values of the parameters regardless of the modeling error, and thus underfitting may occur. We observered that in the figure 36 when lamda= 0.01, RMSE test is 1.7 and RMSE train is 0.39 and when lamda=10 , RMSE test is 1.015 and RMSE train is 1.0759 in ridge cholesky model from sklearn library. As it is seen that increase of the lamda reduces the overfitting which means train and test set get closer and low bias.

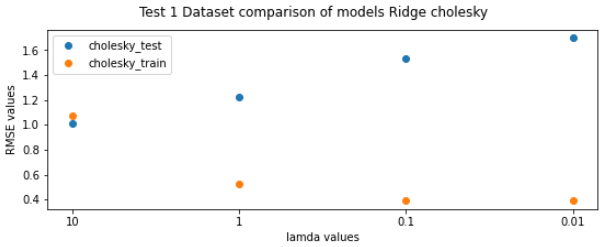


Figure 36. Comparison of Testing and Training Data with Lamda

Similarly, SGDRegressor (loss='squared_error', penalty='l2') and Ridge solves the same optimization problem, via different means. For regression with a squared loss and a l2 penalty, another variant of SGD with an averaging strategy is available with Stochastic Average Gradient (SAG) algorithm, available as a solver in Ridge. The concrete loss function can be set via the loss parameter. SGDRegressor supports loss="squared_error" through ordinary least squares with depends on the lamda value.
Here, we compared the our model with ridge sklearn library that using the SGD and SAGD respectively. Initially, our model had higher error with higher lamda value and the reached the less RMSE value with lower lamda value at the end. The other model were tuned with tol value and then the figure is depicted. The other two model errors were close to each other since they did not use the learning rate and 1000 number of iterations as GD as discussed in the section 4.7.2. In contrast to our model they used maximum 1000 number of iterations. SGDregressor with average computes the averaged SGD weights across all updates and stores the result in the coef_ attribute. If set to an int greater than 1, averaging will begin once the total number of samples seen reaches average. Therefore average=1000 will begin averaging after seeing 1000 samples. We observered from the figure 37 that with SGD higher lamda lower the RMSE in compare to the SAGD, with SAGD lower the lamda lower the RMSE.

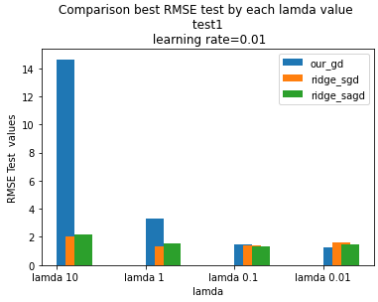


Figure 37. Comparison of Best RMSE with Each Lamda Value

We compared the result with in different test space that we collected the data and observed the details. Ridge model used sag solver that does not include learning rate tuning parameter has been done based on the lamda (alpha) value that is 10 and tol=0.0001 is reached almost same weigths with our model when we use the learning rate=0.1, lamda=10). As expected, it seems it is surronded to the zero in the figure 38.

Figure 38. Weights Comparison with Our GD Model and Ridge Sklearn Library

In order to confirm the final performance comparisons are sincere, we compared the models based on errors calculated with the ground truth occupancy data from the testing set. Table 5 shows the performance of occupancy estimation models for both training and testing sets.
Based on the research [42] models have a lower variance than OLS models. We observed as seen in the table 5, the variance is very low. However, due to the penalty term (L2), they usually have a slightly higher bias than OLS models which the model had high RMSE value first and then by tuning with lamda value we reduced the bias as well. Our model reached the following RMSE values as depicted in the table 5 with the same 0.001 learning rate value. It shows that increase of the lamda reduces the RMSE with same learning rate.

Table 5. RMSE Value in Training and Testing Sets

Model (Our GD)	Performance (Training Set) RMSE	Performance (Testing Set) RMSE
Lamda=10	14.633	14.979
Lamda=1	3.277	3.224
Lamda=0.1	1.445	1.373
Lamda=0.01	1.25	1.177

Since after tuning the parameters we reached the lowest RMSE[20] value 1.25 in our model with lamda 0.01. In the table 6 we observed that how it effect the RMSE value within different learning rate between 0.00001 and 0.01[21]. The table shows that with the 0.001 learning rate it reached the minimum RMSE on test set and it fits the linear line more better that the other case, that conclues that with this learning rate it genarly fit the best line of fit[22].

Table 6. RMSE Value in Training and Testing Sets

Model (Our GD)	Performance (Training Set) RMSE	Performance (Testing Set) RMSE
Learning rate=0.00001	5.094	4.998
Learning rate=0.0001	4.125	4.092
Learning rate=0.001	1.25	1.177
Learning rate=0.01	1.452	0.543
Learning rate=0.1	1.783	0.531

Oscillating performance can be caused by weights that diverge. Since a learning rate is too small it seem in the table 6 it is not converge or may get stuck on a suboptimal solution because the error is high. While we increased the learning rate step by step, it seems that GD inadvertently increase rather than decrease the training error even after some point we got infinite value.

Here we plotted the comparison of RMSE value with learning rate and lamda values respectively by implementing our GD model to the experiment 1. Figures 39(a), 39(b) show the values of RMSE with the increments of learning rate, regularization coefficient respectively. It is obvious that RMSE decreases firstly and then increases with the increments of learning rate. On the other hand, RMSE stay around same band firstly and then increases with the increments of regularization cofficient (lamda). For the moment, when regularization coefficient is smaller the regularization term has no impact on training that causes overfitting on the contrary, when regularization coefficient gets larger, the

[20] Excel file for learning rate vs RMSE comparison table
[21] Excel file for lamda vs RMSE comparison table
[22] Python script for comparison learning rate with RMSE

minimization reduces the values of the parameters regardless of the modelling error that causess underfitting. Therefore the model is reached minimum error with 0.003 learning rate and 0.1 lamda.

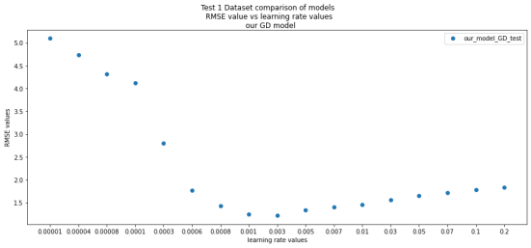


Figure 39(a).RMSE with Learning Rate

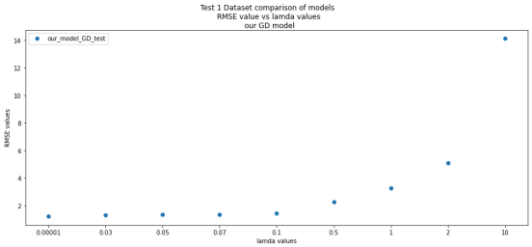


Figure 39(b). RMSE with Lamda

6.5. Conclusion of Model for the Experiment 1

Our model passed the model validation steps that allows us to conclude that our model can predict future population numbers using the 18 independent variables [f1,..,f18]. However, our model has an R^2 score of 72.8%, implying that there are approximately 27.2% unknown factors influencing our real value. In addition compare to the other model as seen in the table 5. our proposed model achieved 1.25 RMSE value which means that we had 1.25 error with actual number of people in the space.

6.6. Performance of Occupancy Prediction Models

In this section we will evaluate the models that we have been used for our experiments. In addition to the experiment 1 as before we tested it, we had other experiment scenarios as following

Experiment 2: The second experiment is conducted in the same laboratory as experiment. The daily number of occupants were between 1 and 5. The measurement campaign lasted for several days, and a total of 49 ground truth observation were collected[23].

Experiment 3: This experiment is also conducted in an indoor university laboratory. We call some other people from their office to our lab and the maximum number of actual people in the laboratory was reached to 10 this time as seen in the blured picture for privacy concern in figure Figure 40 (b). The total of 85 ground truth observation were collected in one day. It was challeging the find do capturing experiments since people mostly was in vocation.

Experiment 4: This experiment is conducted in the canteen of university. The environment has an area of about 40 sqm and is characterized by around 60 fixed seats and a daily number of occupants between 1 and 19. As seen in the figure 40(a) due to fact of privacy concern the testbed area picture captured before people coming to the canteen. The space was included with kitchen without door and there were an another office just in the canteen with other people that we did not count for them as a ground truth. They joined the canteen from time to time that caused great amount of background noise in terms of Wi-Fi packets is present in the dataset. The measurement campaign lasted for around 24 minutes and a total of 32 ground truth observation were collected[24]. For example while recording actual number of people in the room since lastly we were 2 people but at the end the dataset I wrote the actual number of people was 4. That is because of that the people were moving to the canteen time to time which we were not able to distunguish

[23] CSV file for collected data test2
[24] CSV file for collected data in experiment 4 (canteen scenario)

that. This could be improve in the future by using camera or other model to achive the ground truth value properly.

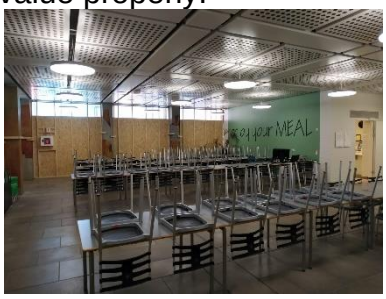


Figure 40(a). Testbed Experiment 4 (Canteen)

Figure 40(b) Testbed Experiment 3

6.6.1. Final performance comparisons among all models

In this section[25] we compared our model with the SGDRegressor library which used SGD and SAGD respectively as seen in the figure 41. First we observed that both SGDRegressor values quite close to each other as seen in the figure. However SGD itself reaches less error compare to the averaged SGD till 0.001 learning rate in each lamda afterward they matched among each other perfectly and errors becomes same in both model. In the figure 41(b) it is obvious that RMSE reduces initialy and after that increases with the increments of learning rate similary with figure 41(c) and figure 41(d). In the figure it is observed that in some case the difference is quite far it may occur in the case of the tuning parameters with different model may varries the oscillation. Our model reaches the optimal (less) error with 0.01 lamda 0.001 learning rate as seen in the figure 41(d).

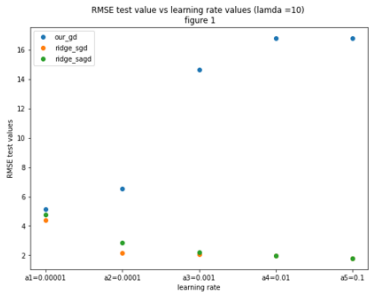


Figure 41 (a)

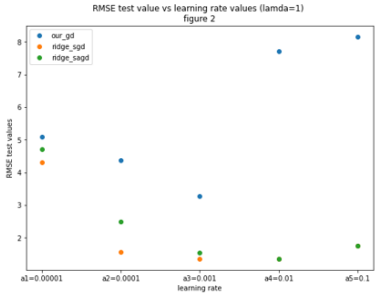


Figure 41 (b)

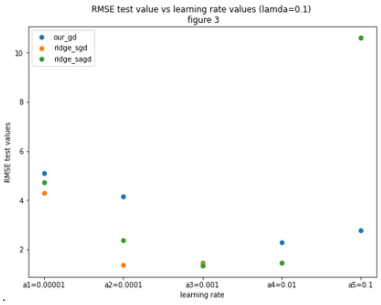


Figure 41 (c)

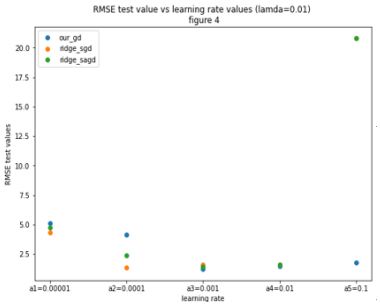


Figure 41 (d)

Figure 41. Comparison of The RMSE Values vs Learning Rate with Different Lamda Value

According to [43] if the learning rate is too slow, the loss function decreases very slowly, whereas if the learning rate is too fast, the cost increases or oscillates as also seen in the figure 41. We observed from figure 41 that after we set the learning rate to the 0.3 it oscillated which does not seen in the plot. It is clear from the figure that at first our model has higher error compare to the other models (SGD and SAGD) although it reaches

[25] Python script for plotting learning rate vs RMSE, comparison of model

lowest error with lamda 0.01 and learning rate 0.001. At last, therefore these tuned parameters have been chosen for the optimal value of the RMSE.
We compared the RMSE with learning rate in 4 different test experiment with our GD model in the figure 42. The figure show the values of RMSE with the increments of learning rate and regularization coefficient in combine with 4 different test scenarios. We plotted the comparison of RMSE value with learning rate and lamda values respectively by implementing our GD model to the 4 different test bed scenarios. Figure 42(a), figure 42(b), figure 42(c) and figure 42(d) show the values of RMSE with the increments of learning rate, regularization coefficient respectively. It is obvious that RMSE decreases firstly and then increases with the increments of learning rate with lamda range 1 and 0.01. And the RMSE increases with the increments of regularization cofficient (lamda). The model is reached minimum error with 0.001 learning rate and 0.01 lamda.
The RMSE[26] values of the 4 different test experiments experiment 1, experiment 2, experiment 3 and experiment 4 are 1.25, 1.18, 2.0329 and 3.4781, respectively. We had higher error in last experiment with maximum number of people 19 in the dataset. That was because of the number of people was changing very quickly and while taking manually notes we did not matched with the actual number of people at the end. For example, while doing experiment we noted 4 people at the end and put that value to the algorithm as a real value however it was 2 people. In the experiment 3 the maximum actual number of people was 10, the model had higher error compare to the experiment 1 and 2 where maximum number of people was 7 and 5 respectively. But the second experiment includes higher number of samples compare to the first experiment. We observed that when the samples and maximum number of the actual people increases cause the more error in our experiment but this may not be the case as always.

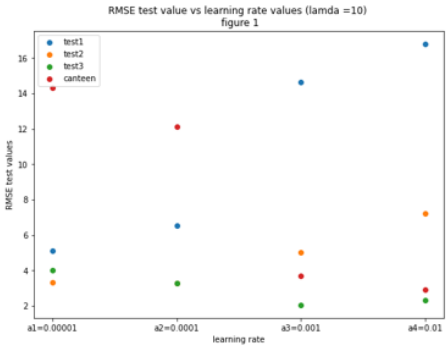


Figure 42 (a)

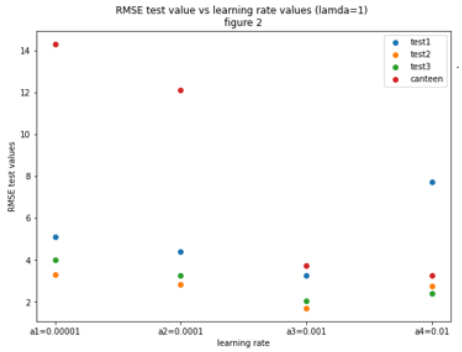


Figure 42 (b)

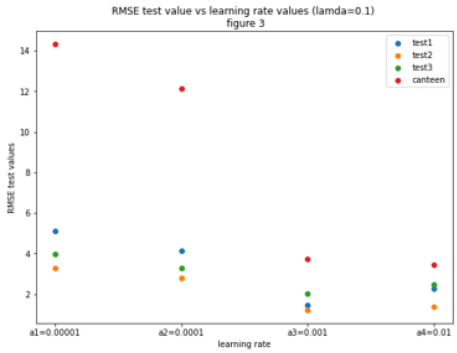


Figure 42 (c)

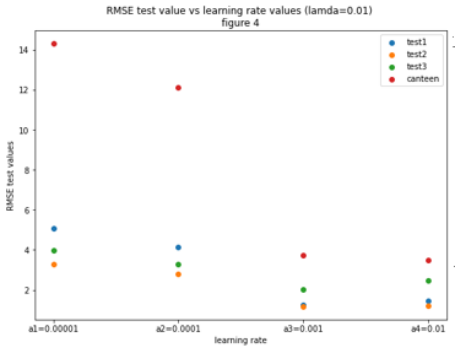


Figure 42 (d)

Figure 42. Comparison of the RMSE with Learning Rate in 4 Different Test Experiment

[26] Python script for 4 different experiments

7. CONCLUSION

According to the literature review presented in chapter 2, smartphone sensors can be used to predict the social density level of the enviroment. We described the some fundemantals in the chapter 3. We described our model and data collection processing in the chapter 4. The occupancy monitoring in indoors using Wi-Fi Probe Requests has been discussed and also carried out experiments, which are described in chapter 5, to evaluate the hypothesis. In the chapter 6 we evaluated the model and compared of our model and Python sklearn models with in different kind of testbeds. In chapter 7, we discussed our results, limitations and future works.

As smart devices equipped with WLAN communication technologies are becoming more and more widespread, the research focus is directed on the opportunities for utilizing signaling messages of such communication technology for estimation of number of people. The way how a mobile device searches for nearby APs with Wi-Fi probes device discovery procedure are transmitted can be exploited to estimate the number of active devices in a certain area. This approach allow discovery and counting people carrying a Wi-Fi capable device. Since the number of smart devices is constantly growing, that makes utilizing Wi-Fi communication technologies realistic and promising for people density estimation.

In this thesis, occupancy prediction is studied by passively monitoring Wi-Fi probe requests captured from smartphones. We used a linear ridge least squares technique to estimate the number of people for smartphones based on PR data collected at different testbeds. Our results demonstrate that PRs can be a practical key for occupancy estimation in future smart buildings, which can have application such as surveillance and energy management.

Finally, we conducted occupancy experiments in an indoor environment and conducted a detailed study on mission planing which assisted us in capturing various information by passively monitoring Wi-Fi PRs captured from smartphones. Our study found that PRs could be a practical solution for occupancy monitoring.

At the first we considered the problem we wanted to solve. And we reseached that there have been different approaches using different environment characteristics to find a solution. But when accuracy and performance and speed is one of the most significant demands for the final solution, there are many limitations with the traditional methods as we discussed before.

For that reason in the experiment section, we tried to tune our model to have low error and using an ML lets us modify our configuration dynamically. This distinguishes our approach from traditional methods.

7.1. Different Factors Influence the Wi-Fi Probe Requests

We need to collect data and estimate algorithm accuracy as we already discussed. Analysis of indoor surroundings has indicated that there are two main challenges of the estimation using Wi-Fi probes. The algorithm may give the wrong estimation due to the inclusion of Wi-Fi devices that are outside and due to exclusion of people without an active Wi-Fi enabled device. We need to show how probes received from people outside can be filtered out thus reducing the underestimation problem.

Many studies have proved that more PRs are captured when channel hopping is not used since wireless adapters can only capture on a single channel at any given time.

As descried in the previous section, the choice of channel is expected not to have an important influence on the tests. Freudiger et.al. (2015) experimentally studied how different factors have an impact on the WiFi PRs, including monitor channel

configurations, number of SSIDs stored in the PNL and device configurations [16]. They captured the largest number of probes with three antennas with each set to a fixed non-overlapping channel. The performance of probes depends on the device manufacturers since the number of probes is dependent on the number of known SSIDs. The open device screen shows that a fake Wi-Fi beacon in the vicinity will drive more PRs.
The other possible factors that have impact on the number of received probe packets are a number factors including signal strength of the AP, channel utilization frequency and the number of devices in the space.

7.2. Constraint of Wi-Fi sniffer

There are factors such as channel capacity and link quality that affect the overall sniffing performance. Wi-Fi sniffer limitations are as follows; some people may not carry devices with a wireless interface, some people may not have their Wi-Fi enabled device, some people may hold more than one device, some devices might have numerous Wi-Fi adapters and as the mobile device goes through different venues rapidly some transmissions may not be noticed.

7.3. System improvements

We could improve in the future the model while collecting the data in different mode such as active-screen modes and inactive-screen modes, power-saving modes, with the device keeping the Wi-Fi interface switched off,etc.
We may be do experiment in following conditions; a group of people was periodically checking their smartphones, a group was moving around the place while using their devices, a group remained seated, a group was moving around the place without using their devices, decision of time duration (etc. 30 mins - 1hour), some group carry not only one smartphone, decision of places indoor which can have PR from outside even or check outdoor as well seperately, crowdness of area (10-100 people), analyze the behavior of system for less and more crowd places respectively, conduct tests in different wireless environments to investigate the possible factors that affect the received number of packets, compare the sniffing performance with different sniffers in terms of numbers of packets, number of devices captured and RSS levels recorded, situations the counting task is affected by the number of static devices in the same area (etc. tablet, smart TV) collect in some certain period of time of data and manually annotated ground truth to validate and evaluate the functioning [21].

7.4. Future Work

Increasing the Wi-Fi scanning time of 2 to 5 intervals results in more consistent results because the sensor device can pick up more PRs. However we may get a higher delay from real time. One method for determining an optimal scanning period is to configure four identical RP is with the only difference being the time between scans. Furthermore, we could also do classification analysis to analyze whether we can improve the performance of the prediction. We could expand the range of the area and number of occupants in indoor and do experiment. This project can be implemented for public transportations, stadiums or hospitals where supporting social distance since the nowadays the covid problems rises.
Also this work can be evaluated to the mobile version by developing a mobile version of sensors since the mobile sensors ara part of the same working principle as the Wi-Fi sniffer based static sensors.

APPENDIX A

Wifi sniffing configuration in Kali Linux with TCPDUMP

$ifconfig # showing interfaces that are supported
$sudo ifconfig wlan0
$airmon-ng start wlan0 # set the wireless interface into monitor mode
#$iwconfig # To list the wireless interfaces on the system, Wlan0: wireless line interface, eth0, lo will apper, command shows that the mode is set to Monitor
$sudo ifconfig wlan0mon down #turn interface down, command is used for disabling the #Managed mode
$sudo iwconfig wlan0mon mode monitor # command is used to enable interface wlan0mon #monitor mode
$sudo ifconfig wlan0mon up #interface device up, command is used to enable the interface
$sudo iwconfig wlan0mon chan 6 # set the channel 6
$ sudo tcpdump -i wlan0mon -w #captureing packets that specificly specified wlan0mon #interface
$sudo tcpdump -i wlan0mon -w /home #to the file directory
$sudo timeout 120 tcpdump -i wlan0mon -w # capturing data till 2 minutes

APPENDIX B

List of known smartphone manufacturers
A list of known smartphone manufacturers [] used to filter out other wireless peripherals. The list is compiled from the public listing of the manufacturer-registered MAC addresses of IEEE (found at http://standards-oui.ieee.org/oui/oui.txt).

MacPrefix	Vendor Name
00:00:0C	Cisco Systems, Inc
00:00:0D	FIBRONICS LTD.
00:00:0E	FUJITSU LIMITED
00:00:1B	Novell, Inc.
00:00:23	ABB INDUSTRIAL SYSTEMS AB
00:00:31	QPSX COMMUNICATIONS, LTD.
00:00:37	OXFORD METRICS LIMITED
00:00:3C	AUSPEX SYSTEMS INC.
00:00:3E	SIMPACT
00:00:3F	SYNTREX, INC.
.	.
.	.
.	.

ABBREVIATIONS - ACRONYMS

IEEE	Institute of Electrical and Electronics Engineers
Wi-Fi	Wireless Fidelity
TCPDUMP	TCP/IP Data-Network Packet Analyzer
WLAN	Wireless Local Area Network
MAC	Media Access Control Address
RFMON	Radio Frequency Monitor
ML	Machine Learning

REFERENCES

[1] Jean Conti, Tiago Silveira, Heitor Silvério Lopes “Wi-Fi Device Identification in Crowd Counting Using Machine Learning Methods” In: ResearchGate, Learning and Nonlinear Models (2018)
[2] Edoardo Longo, AlessandroE.C. Redond, Matteo Cesana, “Accurate occupancy estimation with WiFi and bluetooth/BLE packet capture”, in: Computer Networks 163, (2019).
[3] Wei Xi, Jizhong Zhao, Xiang-Yang Li, Kun Zhao, Shaojie Tang, Xue Liu, Zhiping Jiang, “Electronic Frog Eye: Counting Crowd Using WiFi” in IEEE INFOCOM 2014-IEEE Conference on Computer Communications, (2014).
[4] Pichaya Prasertsung and Teerayut Horanont. “How does coffee shop get crowded? Using WiFi footprints to deliver insights into the success of promotion". In: Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers. 2017, pp. 421-426.
[5] Jiaxing Shen, Jiannong Cao, Xuefeng Liu, and Shaojie Tang. “SNOW: Detecting shopping groups using WiFi". In: IEEE Internet of Things Journal 5.5 (2018), pp. 3908-3917.
[6] Marcus Handte, Muhammad Umer Iqbal, Stephan Wagner, Wolfgang Apolinarski, Pedro Jose Marron, Eva Maria Munoz Navarro, Santiago Martinez, Sara Izquierdo Barthelemy, and Mario Gonzalez Fernandez. “Crowd density estimation for public transport vehicles". In: EDBT/ICDT Workshops. 2014, pp. 315-322.
[7] André Schmidt. “Low-cost Crowd Counting in Public Spaces.” In: (2014).
[8] Ooi Boon Yaik, Kong ZanWai, Ian K.T.Tan, and Ooi Boon Sheng. “Measuring the Accuracy of Crowd Counting using Wi-Fi Probe-Request-Frame Counting Technique.” In: Journal of Telecommunication, Electronic and Computer Engineering (JTEC) 8.2 (2016), pp. 79–81. issn: 2289 8131.
[9] Lars Mikkelsen, Radoslav Buchakchiev, Tatiana Madsen, and Hans Peter Schwefel. “Public transport occupancy estimation using WLAN probing". In: 2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM). IEEE. 2016, pp. 302-308.
[10] Zhe Wang, Tianzhen Hong, Mary Ann Piette, and Marco Pritoni. “Inferring occupant counts from Wi-Fi data in buildings through machine learning". In: Building and Environment 158 (2019), pp. 281-294.
[11] Yuyi Cai, Manabu Tsukada, Hideya Ochiai, and Hiroshi Esaki. 2021. “MAC address randomization tolerant crowd monitoring system using Wi-Fi packets.” In Asian Internet Engineering Conference (AINTEC ’21), December 14–16, 2021, Virtual Event, Japan. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3497777.3498547.
[12] M. Vanhoef, C. Matte, M. Cunche, L. S. Cardoso, and F. Piessens, ”Why mac address randomization is not enough: An analysis of wifi network discovery mechanisms," in Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, pp. 413-424, 2016.
[13] C. Matte et al. “Defeating MAC Address Randomization Through Timing Attacks”. In: 2016, pp. 15–20.
[14] J. Franklin et al. “Passive Data Link Layer 802.11 Wireless Device Driver Fingerprinting”. In: (2006).
[15] V. Brik et al. “Wireless device identification with radiometric signatures”. In: 2008, pp. 116–127.
[16] J. Freudiger, “How talkative is your mobile device? an experimental study of wi-fi probe requests,” in Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks, 2015, pp. 1–6.
[17] R. H. Ribeiro, B. B. Rodrigues, C. Killer, L. Baumann, M. F. Franco, E. J. Scheid, and B. Stiller, “Asimov: a fully passive wifi device tracking,” in 2021 IFIP Networking Conference (IFIP Networking). IEEE, 2021, pp. 1–3.
[18] Yuyi Cai, Manabu Tsukada, Hideya Ochiai, and Hiroshi Esaki. 2021. “MAC address randomization tolerant crowd monitoring system using Wi-Fi packets.” In Asian Internet Engineering Conference (AINTEC ’21), December 14–16, 2021, Virtual Event, Japan. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3497777.3498547.
[19] LUIZ OLIVEIRA, DANIEL SCHNEIDER, JANO DE SOUZA, WEIMING SHEN 2019. “Mobile Device Detection Through WiFi Probe Request Analysis”. SPECIAL SECTION ON DATA MINING FOR INTERNET OF THINGS, August 7, IEEE, 2019.
[20] Ling Pei, Jingbin Liu, Yuwei Chen, Ruizhi Chen, Liang Chen. “Evaluation of fingerprinting-based WiFi indoor localization coexistedwith Bluetooth” In: Pei et al. The Journal of Global Positioning Systems (2017) 15:3 DOI 10.1186/s41445-017-0008-x.
[21] Yan LI, Johan Barthelemy, Shuai Sun, Pascal Perez, and Bill Moran “A Case Study of WiFi Sniffing Performance Evaluation” In: IEEE, VOLUME 4, 2016.
[22] Arunesh Mishra, Minho Shin, William Arbaugh, “An Empirical Analysis of the IEEE 802.11 MAC Layer Handoff Process.” In: ACM SIGCOMM Computer Communication Review Volume 33, Issue 2 April 2003, pp 93–102https://doi.org/10.1145/956981.956990.
[23] G. P. Zanetti and C. E. Palazzi, "Non-invasive node detection in IEEE 802.11 wireless networks," 2010 IFIP Wireless Days, 2010, pp. 1-5, doi: 10.1109/WD.2010.5657729.

[24] Alok Pandey,Jatinderkumar R. Saini,"Counter Measures to Combat Misuses of MAC Address Spoofing Techniques", Int. J. Advanced Networking and Applications 1358 Volume: 03, Issue: 05, Pages: 1358-1361 (2012).
[25] Nikhil Ketkar, (2017), “Deep Learning with Python A Hands-on Introduction”,chapter 1,8,1314.
[26] Gary C. McDonald, “Ridge regression”, John Wi ley & Sons, Inc. Volume 1, July/August 2009 WIREs Computational Statistics.
[27] Ronald de Vlaming and Patrick J. F. Groenen, “The Current and Future Use of Ridge Regression for Prediction in Quantitative Genetics”, Hindawi Publishing Corporation BioMed Research International Volume 2015, Article ID 143712, 18 pages http://dx.doi.org/10.1155/2015/143712, 24 December 2014.
[28] Fanming Huang, Dan Li, Jiachen Xu, Yutao Wu, Yidan Xing1, Zan Yang, “Ridge Regression Based on Gradient Descent Method with Memory Dependent Derivative”, 2020 IEEE.
[29] Maria-Dolores Cano, Antonio Guillen-Perez “A WiFi-based method to count and locate pedestrians in urban traffic scenarios”, October 2018, DOI:10.1109/WiMOB.2018.8589170, Conference: 2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob).
[30] Y. Zhou, B. P. L. Lau, C. Yuen, B. Tuncer, and E. Wilhelm, “Understanding urban human mobility through crowdsensed data,” IEEE Communications Magazine, vol. 56, no. 11, pp. 52–59, 2018.
[31] Y. Yoshimura, S. Sobolevsky, C. Ratti, F. Girardin, J. P. Carrascal, J. Blat, and R. Sinatra, “An analysis of visitors’ behavior in the louvre museum: A study using bluetooth data,” Environment and Planning B: Planning and Design, vol. 41, no. 6, pp. 1113–1131, 2014.
[32] A. J. Ruiz-Ruiz, H. Blunck, T. S. Prentow, A. Stisen, and M. B. Kjærgaard, “Analysis methods for extracting knowledge from large-scale
wi-fi monitoring to inform building facility planning,” in 2014 IEEE International Conference on Pervasive Computing and Communications (PerCom). IEEE, 2014, pp. 130–138.
[33] F.-J. Wu and G. Solmaz, “Crowdestimator: Approximating crowd sizes with multi-modal data for internet-of-things services,” in Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services. ACM, 2018, pp. 337–349.
[34] C. Chilipirea, C. Dobre, M. Baratchi, and M. van Steen, “Identifying movements in noisy crowd analytics data,” in 2018 19th IEEE International Conference on Mobile Data Management (MDM). IEEE, 2018, pp. 161–166.
[35] L. Schauer, M. Werner, and P. Marcus, “Estimating crowd densities and pedestrian flows using wi-fi and bluetooth,” in Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2014, pp. 171–177.
[36] T. S. Prentow, A. J. Ruiz-Ruiz, H. Blunck, A. Stisen, and M. B. Kjærgaard, “Spatio-temporal facility utilization analysis from exhaustive wi-fi monitoring,” Pervasive and Mobile Computing, vol. 16, pp. 305–316, 2015.
[37] Jeremy Martin , Travis Mayberry, Collin Donahue, Lucas Foppe, Lamont Brown, Chadwick Riggins, Erik C. Ry, Dane Brown, “A Study of MAC Address Randomization in Mobile Devices and When it Fails”, US Naval Academy, 31 March 2017.
[38] Paolo Galluzzi, Edoardo Longo, Alessandro E. C. Redondi, Matteo Cesana, “Occupancy Estimation Using Low-Cost Wi-Fi Sniffers”, Computer Networks 163 (2019) 106876.
[39] Lucia Pintor, Luigi Atzori, “A dataset of labelled device Wi-Fi probe requests for MAC address de-randomization Computer Networks”, 205 (2022), 108783, ScienceDirect.
[40] Yuki FURUYA, Hiromu ASAHINA, Masashi YOSHIDA, Iwao SASASE, "FellowIndoor Crowd Estimation Scheme Using the Number of Wi-Fi Probe Requests under MAC Address Randomization", IEICE TRANS. INF. & SYST., VOL.E104–D, NO.9 SEPTEMBER 2021.
[41] A.E. Redondi, M. Cesana, “Building up knowledge through passive wifi probes”, Comput. Commun. 117 (2018) 1–12.
[42] Ashouri, Araz; Newsham, Guy R.; Shi, Zixiao; Gunay, H. Burak, “Day ahead prediction of building occupancy using WiFi signals” 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE), pp. 1237-1242, 2019-08-22.
[43] G. Dreyfus, “Neural Networks, Methodology and Applications”, Springer-Verlag, Heidelberg, (2005), 115.
[44] Satish Kumar, “Rajinder Singh A Comparative Study of Various Wireless Network Monitoring Tools”, 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC).

6-Month Timeline

2022 Mar Literature Review
Study of previous research and theories, identify areas of controversy, contested claims, highlight gaps that may exist in previously, decision of the model selection

2022 Apr Writing Literature Review
model implementation discussion, writing the literature review, decision of the data collection software, model suggestion

2022 May Start Coding
Python implementation, linear regression model analyses, and digging deeper into the benefits and drawbacks of linear regression

2022 Jun Data Collection
Start collecting data in the university laboratory, downloading and using Kali Linux OS, Getting use of the tcpdump- wireshark, continue for python implementation

2022 Jul Continue For Coding Collecting
data in the university canteen, analyzes of the captured data, model evaluation, getting familiar with the python sklearn library, python model improvements

2022 Aug Final Approaches
Writing master thesis,preperation of the presentation