Spring 6-21-2023

User Profiling through Zero-Permission Sensors and Machine Learning

Ahmed ElHussiny
aelhussiny@aucegypt.edu

Graduate Studies

User Profiling through Zero-Permission Sensors and Machine Learning

A THESIS SUBMITTED BY

Ahmed Khalid ElHussiny

TO THE

Department of Computer Science and Engineering

SUPERVISED BY

Professor Sherif Aly and Professor Tamer ElBatt

16/5/2023

in partial fulfillment of the requirements for the degree of
Master of Computer Science

Declaration of Authorship

I, Ahmed Khalid ElHussiny, declare that this thesis titled, "User Profiling through Zero-Permission Sensors and Machine Learning" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

_____

Date:
15/05/2023

_____

STUDENT TO INSERT HERE THE PAGE WITH THE SIGANTURES OF THE THESIS DEFENSE COMMITTEE

Abstract

With the rise of mobile and pervasive computing, users are often ingesting content on the go. Services are constantly competing for attention in a very crowded field. It is only logical that users would allot their attention to the services that are most likely to adapt to their needs and interests. This matter becomes trivial when users create accounts and explicitly inform the services of their demographics and interests. Unfortunately, due to privacy and security concerns, and due to the fast nature of computing today, users see the registration process as an unnecessary hurdle to bypass, effectively refusing to provide services with personalization information. In other cases, they may provide inaccurate profile information, either due to lack of accuracy, or for malicious purposes. In this thesis, we use machine learning with zero-permission sensors to test the degree to which it can be used to effectively profile a user without necessitating any explicit input. We do so through first iterating through building an application that collects data from the following zero-permission sensors: the gyroscope, accelerometer, and ambient light sensor. Following that, we pass the data through a multi-step transformation process for feature selection, filtration, and homogenization. We then pass this processed training data through machine learning algorithms, enabling accurate user profiling without the need for explicit information gathering. We additionally test the minimum timespan needed to accurately profile a user, and test three machine learning models. We find that it is indeed possible to accurately predict the biological gender of a user, given 1-day intervals, and using a support vector machine.

Acknowledgements

I wish to express my appreciation to the Department of Computer Science and Engineering of The American University in Cairo for making this study possible.

Sincere gratitude is extended to Professor Sherif Aly and Professor Tamer ElBatt for their utmost support, tutelage, and guidance. The writing of this thesis was interrupted by the COVID-19 pandemic, and I wish to express my endless appreciation for the patience the professors have shown me in completing my work.

Special thanks are due my wife, Nour, for the patience she has shown, her constant, consistent, and unwavering faith in me, and the encouragement, motivation, and inspiration given to me during my graduate study. Her many sacrifices are here gratefully and ceaselessly acknowledged [79].

My sister, Jeena, and my friends', Aly and Ahmed, belief in me were always the biggest motivations and the strongest supports, and for them I am eternally grateful.

Contents

List of Figures

List of Tables

List of Abbreviations

HTML	**Hyper-Text Markup Language**
PWC	**PriceWaterhouseCooper**
MAU	**Monthly Active Users**
FBI	**Federal Bureau of Investigations**
CIA	**Central Intelligence Agency**
US	**United States**
GDPR	**General Data Protection Regulation**
SI	**Standard International**
ERT	**Extremely Randomized Trees**
OBD-II	**On-Board Diagnostics – II**
SVM	**Support Vector Machine**
KNN	**K-Nearest Neighbor**
HAR	**Human Activity Recognition**
PDR	**Pedestrian Dead Reckoning**
PNS	**Pedestrian Navigation System**
SMS	**Short Message Service**
SDK	**Software Development Kit**
UI	**User Interface**
ID	**Identifier**
API	**Application Programming Interface**
VPN	**Virtual Private Network**
IRB	**Institutional Review Board**
NB	**Naïve Bayes**
LG	**Logistic Regression**
SV	**Support Vector Machine**
JSON	**JavaScript Object Notation**

List of Symbols

$p(x)$	Probability of object x
$p(x \mid d)$	Probability of object x, given D
Δ	Delta, or “change in”

Chapter 1

Overview

When the World Wide Web was first conceived, it was thought of as an information system, capable of delivering information to users. The pages delivered to individuals around the world, therefore, could be static: like pages of a book. An author with knowledge in a specific field would compose specific information, orient it as they please in HTML, and could even hyperlink to their sources, or other places where information could be found. With the dawn of the participative web, pages delivered to users could be generated by programs that relied on user input. These were no longer the static pages of the past; a user could see their own name after "Hello," and that was very exciting. In a sense, this was the very first step towards what we experience today.

Today's web is mostly user-generated. One need only look at the most popular websites in the world to understand that user content, instead of the authoritative content of the past, has come to dominate the Internet. Social media, personal blogging, volunteer-written encyclopedias, and video-sharing websites have become the driving force of modernity. And with the prevalence of mobile and pervasive computing, user-generated content is only on the rise. With such huge volumes of this content - an estimated average of 1.7 Megabytes are generated per individual per second [1] - users are finding difficulty locating and subscribing to content they find interesting. This is exactly why personalization plays a key role in today's web. If users can find content they are interested in, or receive advertisements that are relevant to them, they are more likely to engage with that content and spend more time on a given website. The question then becomes not one of generating new content, but rather of how to properly target specific content to a specific user.

There is a very easy way to personalize user content: simply use the information that they have provided about themselves during the registration process. Simple demographic information like a user's gender and age can easily allow for the generic filtration of content to properly target said user. For a while, this worked well. Users understood the expectation to have to register and provide information about themselves to be able to use a service. Unfortunately, with repeating data breaches, privacy concerns, and data misuse scandals by the largest companies in the world, users' trust has quickly eroded. A lot of users see the registration process not only as a hinderance, an extra hurdle for having to use a service, but also a potential security concern. When users do register, they are often providing false information. The most obvious indicator for this is the rise of temporary email services, with users often preferring to use a temporary email rather than provide a service with their actual

email address.

This creates a very interesting gap. Users want content to be personalized to their profiles but refuse to provide the information necessary for this personalization for privacy fears. What this thesis aims to do is provide a method that relies on machine learning through which the most basic factor of this personalization can occur: biological gender. The data necessary for this profiling would not come explicitly from users, circumventing tampering, nor would it come from identifiable sources, successfully addressing users' privacy concerns. Instead, it would come from the zero-permission sensors embedded in users' mobile devices.

This, however, presents an interesting set of questions that come associated with it. Which sensors should we be using? How long should we be recording data in order to get an accurate prediction of the biological gender? Is it possible at all? And if it is indeed possible, which machine learning paradigms and algorithms should we use to accurately perform this prediction?

These are exactly the questions this thesis delves into and attempts to provide an answer for, as explained in detail in the following sections.

Chapter 2

Motivation

For users to be able to find the content that they are seeking in this data-congested world, personalization is very important. Let us consider YouTube, the world's largest video sharing service with over 2 billion monthly active users. It is so popular that the Q3 2019 Sandvine Global Internet Report shows it accounting for 8.7% of all global Internet traffic [9]. By Q1 2022, YouTube's domination had grown to account for 14.61% of all global Internet traffic [61]. More specifically, the 2019 Sandvine Mobile Phenomena Report also reveals that YouTube accounts for 35% of all mobile web traffic [11], while the 2021 report places YouTube as consuming 24.6% of all mobile web traffic [62]. While the number has decreased over time, one must remember that this is a percentage, and that overall internet usage has risen significantly in the 2 years between the reports. It is almost important to note that this "decline" is largely contributed to the rise of competitor TikTok [62], which is yet another highly personalized service. But continuing to use YouTube as an illustrative example, it is clear that it alone represents a significant portion of the Internet. According to the most recent statistics released by YouTube, 70% of what users watch is determined by the recommendation algorithm [13], not explicitly sought out by them. Effectively, this means that the recommendations algorithm is responsible for approximately 6.1% of all Internet traffic and 17.2% of mobile web traffic. These staggering figures emphasize the absolute importance of this recommendation algorithm for the business of YouTube, and personalization as a whole to other services, and highlight the impact it has on our world. An important question then becomes, how does it work?

A paper published by Google, the parent company of YouTube, aims to explain the inner workings of the recommendation algorithm. Put simply, the algorithm relies on deep neural networks to determine the best potential recommendations for a given video for a given user. In the paper, Google states that the most important input to the neural network is the user's search and watch histories [4]. It is a logical approach: if a user has searched for or watched a specific video, perhaps they want to see a similar video. As users consume more content, YouTube begins to form a picture of their tastes, behaviors, and interests. But what about users with sparse search and watch histories? This is commonly known as the cold-start problem with recommender systems. Lika et al explain that there are three types of cold-start problems: users with sparse histories, items with sparse interactions (videos with fewer views, in this example), and items with sparse interactions for users with sparse histories [10].

In order to get around the first type of cold-start problem, sparsity of user history, YouTube

also provides its recommender neural network with other inputs. In the paper, they state that demographics are the most important input to their neural network for users with sparse histories. These demographics are a user's geographic region, device, logged-in state, gender, and age [4]. For users with sparse histories, these factors become the primary inputs to the neural network, thereby actively determining what gets seen.

With this example of one service, we can already see the importance of collecting and understanding demographic data. However, demographics do a lot more than simply drive user engagement and help people find the content they are looking for. In today's web, demographics have a monetary value that is quite significant.

For the fiscal year of 2019, Alphabet, Google's parent company, reported over $15.1 billion from YouTube's advertising alone [2]. For the fiscal year of 2022, YouTube's advertising revenue had grown to $29.24 billion, very close to doubling over just three years. Meanwhile Facebook, owning properties accounting for 20% of worldwide mobile traffic [11], made $16.6 billion in just the second quarter of the 2019 fiscal year [6] and $32.17 billion in the fourth quarter of 2022 from advertising alone [71], again almost doubling in just three years. Interestingly, both Facebook [5] and YouTube's [3] respective support pages list demographics, such as "age and gender," as the very first method through which advertisers could target specific audiences. PWC reports that "Internet advertising revenues in the United States totaled $107.5 billion for the full year of 2018." [7] The same report but only covering the first half of 2019, the first 6 months of the year generated $57.9 billion, a 16.9% increase over 2018's first half's $49.5 billion [8]. Finally, the same report covering fiscal year 2021, reports that internet advertising revenues in the United States had risen to $189.3 billion [72]. Importantly, these reports only cover revenue in the United States, and none of the many other countries in the world.

Revenue is not everything, however. Targeted advertising not only helps the advertisers by ensuring that the target market is the one being reached, but research also shows that 75% of users prefer fewer advertisements that are more aligned with their needs and interests, rather than more advertisements that are randomized [12]. Users listed the reduction of irrelevant advertising, the potential to discover unfamiliar items that could align with their interests, and the convenience of quickly finding items of interest as their primary reasons for this preference. According to a survey conducted with 1,500 participants in 2019 by InfoGroup, 90% of consumers say that messages that have not been personalized for them are regarded as "annoying" to them [73]. Furthermore, 67% of Millenials/Gen-Z'ers fully expect any advertising they see to be personalized [74].

The motivation is simple. User demographics, perhaps most importantly age and gender, play a key role not only in assisting users to find the content they did not even know they wanted and protecting them from being bombarded with advertisements irrelevant to them, but also in generating massive amounts of revenue for the most visited services in the world.

Having highlighted the importance of personalization to modern-day systems, it is clear that

having authentic user data is of the utmost importance. According to Facebook's Q4 2019 Earnings Report [14], on Facebook alone, and excluding the other social media entities owned and operated by the platform, there were 2.5 billion monthly active users (MAU). By Q4 2022, that number had risen to 2.96 billion MAU [63]. By Facebook's own estimates [15], 5% of the MAU were fake accounts. The problem is so prevalent that Facebook has had to remove over 3 billion accounts between October of 2018 and March of 2019. And in Q4 of 2022 alone, Facebook "actioned" 1.3 billion accounts for being fake ones [64]. Even with these estimations indicating a significant problem, others seem to think it is much larger. PlainSite, a government transparency project and legal research tool, published a report [16] stating clearly, "Facebook has been lying to the public about the scale of its problem with fake accounts, which likely exceed 50% of its network." With no way to really determine the truth, it is safe to assume that the real number is somewhere in the middle.

The problem is far from being only Facebook's. Between January of 2018 and April of 2020, Twitter has removed over 250,000 users [17][18] and have stated that they will remove up to 6% of their total accounts [19] for misrepresenting who they are. While this number remains much smaller than Facebook's, Twitter admits [17] that it has a more difficult time determining fake users than Facebook. The Australian newspaper published an article interviewing Dan Woods, a former FBI and CIA security specialist. His estimate, based on research he conducted, was that around 80% of Twitter users are fake [65]. Perhaps this is why Twitter revealed it actually actions a million accounts each day for being "spam accounts" in a call with its executives in 2022 [75]. This problem of fake users is much larger than just simply having accounts for whom the recommendation algorithm would not be as effective. It is important to remember that Facebook and Twitter use the profile information of a user to properly target advertising, meaning that businesses are potentially losing a lot of money advertising to some people thinking they are of a completely different profile.

Not only is financial loss at stake here. Several American congressmen suggested that former US President Donald Trump would not have been elected President of the United States if it were not for the spread and reach of news stories propagating false information, or fake news. Hunt and Gentzkow, in their research, did confirm the existence and spread of political misinformation during the 2016 election, and that it was strongly tilted in the favor of Donald Trump [20]. The spread of this misinformation is only possible through "spamming," or the act of repeatedly sharing the same article, by the same or different accounts, with the purpose of making it gain popularity and forcing it to appear on others' feeds. This is only possible if one is in control of several accounts, with different profiles, hence fake accounts.

But in addition to the woes of financial losses, and potential political manipulation, there is also potential humanitarian effects. Seriously lacking in formal research, with it only starting to trickle in publications, we are all familiar with the concept of catfishing, or adopting a different "persona" online, for the purposes of fooling others into believing the existence of that persona. This concept is so prevalent that a simple Google search will display results such

as “top 10 states with the highest catfishing rates.” This of course can lead people to become invested in relationships with people other than who they think they are [76]. This form of online impersonation can also be used for cyberbullying, for identity theft, or even for performing elaborate social-engineering or phishing attacks. Overall, it decreases trust in users to use specific services, and makes them disinterested in engaging in the web as we know it [76]. This is why we are seeing social trends emerge encouraging people to delete social media and dating applications.

In addition to all the problems caused by the abundance of fake profiles, there is also the problem of no profiles at all. According to research conducted by International Translation Resources [21] on 4000 subjects, 15.6% of users uninstall mobile applications due to a complex registration process. Users view registration as an unnecessary hurdle between them and the content they want to access. That is exactly why the Apple Human Interface Guidelines recommend delaying the authentication of a user as much as possible, first convincing a user of the value or use of the application before asking them to register or login [22]. With data breaches, from even the most popular and “reliable” services, most notably the Cambridge Analytica scandal for Facebook, users are also afraid about their information being leaked. Another clear indication that users are unlikely to register is the advent of “social logins,” single sign-on systems created by popular social networks. These systems not only show that users do not want to register for the different services and remember the different credentials, but also highlights the problem of fake accounts on social media platforms. This fake account will not only be used on the social network, but it will spread this misinformation on other websites that use social logins.

Chapter 3

Problem Definition and Resolution Approach

In this section, we formally define the problem we are facing, as well as our suggested resolution approach. We find that our work will have five main contributions, as discussed below.

Through our research, we have found that users simply do not want to register for services, and when they do register, the information that they provide cannot be validated and cannot be trusted. This presents a very important contradiction between users not providing the information necessary for personalization, and yet wanting and expecting applications to personalize the content to their profiles. In addition to being a problem in and of itself, the challenge of lacking and untrustworthy-when-available user data is the root to many others. This data challenge drives the issues for online impersonation, loses potentially billions of dollars on marketing and advertising revenue, disabling healthy economic competition and introducing the potential for economic attacks through simple impersonation. And even at a fundamental level, political manipulation is possible due to the same data challenge, affecting policy and direction. In our research, we hope to address this problem of profile fulfillment or validation, thereby making our research question straightforward: How accurately can we determine the biological gender of a user given inputs from zero-permission sensors such as those recording the physical manipulation and lighting environment of the user's device?

This question is very multi-faceted. First off, there is of course the logic behind the selection of biological gender as the demographic value of choice. This is driven by our research stating that it is the most important factor contributing to personalization and is easily determined due to its binary nature. We are cognizant that determining selective gender is a much more complex question to attempt to answer, but we focus our research on the biological gender of birth. It is also important to note that, in most cases of online impersonation (or catfishing), users were found to use a biological gender different than their own [76].

Following that, why have we selected zero-permission sensors as the input through which we plan on making this determination? This was done for two reasons: relying on physical sensors in a user's device ensures that they are tamper resistant. The sensors are constantly collecting information about the user's environment and handling of the device, and therefore would only be tampered with through the most careful and deliberate of measures. "Faking"

the biological gender of an account would require the malicious user to either go through the process of spoofing the sensor data or actually replicating the sensor pattern of the other gender. This places the technical challenge of faking the gender of a profile at a much higher bar, and therefore being a lot less accessible and a lot more difficult.
Additionally, due to being zero-permission by design, the sensors do not require any user input, flagging permissions, or any additional steps on the part of the user. This makes this process a seamless, non-invasive one that presents the user with no extra hurdles, removing that constraint as a problem for user registration.
After the selection of zero-permission sensors as the category of choice we would target, there comes a decision on which specific sensors we would use. We explore this question through experimenting with all possible combinations of the three most prevalent zero-permission sensors: the accelerometer, gyroscope, and ambient light sensor. This would help us determine which one or combination of these sensors is the best driver for accurate predictions regarding the biological gender of a given user.
Another important part of the question we are attempting to answer is how long does the process need to run to accurately determine the gender of the user? We explore this question through experimenting with different timespans to determine the minimum amount of time necessary in order for our applied machine learning algorithms to make an accurate prediction about the gender of a given user.
And speaking of machine learning, the final decision of our exploration targets that of the machine learning algorithm selection. Which machine learning algorithm would perform best for accurately predicting the gender of a user given a specific timespan of specific zero-permission sensors? We explore this question through repeating all of our experiments with three different machine learning algorithms instead of just one arbitrarily selected algorithm. This will allow us to more accurately define the best algorithm given the task of predicting a user's biological gender from machine learning on zero-permission sensor data alone.

3.1 Thesis Contribution

In this subsection, we explore the main contributions our suggested resolution approach will have. We have determined five main contributions that this thesis will result in. We believe these contributions to be of significant valuable impact, helping tackle the problems of lack of data, and lack of truthful data, as well as helping distill the value of the suggested resolution approach.

3.1.1 Dataset Collection

In our research for related work, we were unable to find a dataset of mobile users' zero-permission sensors. While there are indeed open datasets for accelerometer and gyroscope data such as UCI-HAR, MobiAct, and MotionSense [99], these datasets tend to be extremely

focused on recognizing human activity and do not include data from other zero-permission sensors such as the ambient light sensor, for example. By amassing and availing this dataset of users' accelerometer, gyroscope, and ambient light sensor data, we are not only collecting data for the purposes of this research, but also potentially assisting future research that may need a dataset of this information. We intend to avail this dataset publicly and for free for the research community on a well-established dataset repository, such as Kaggle or GitHub, to support future research. The dataset, as well as its collection methodology and method, are described in detail in Section 5.1.

3.1.2 Exploring Feasibility of User Profiling through Zero-Permission Sensors

Our first step is to determine whether zero-permission sensors could be used, along with a classification method, to properly determine users' respective biological genders at all. If it can be used, then we will have provided applications and services a way to identify and verify user-entered information, potentially more accurately than explicit user input, which may or may not be accurate. This will assist recommendation, personalization, and advertisement targeting algorithms in achieving their targets. Thereby, we will take a big step towards the eradication of fake user profiles and all that it entails. Additionally, we will also provide services the capability to "fill in" pieces of data about users, simplifying or eliminating the need for a registration process.

3.1.3 Exploring and Analyzing the Trade-Offs Associated with Reducing the Timespan of the Data on the Profiling Accuracy

Our second set of experiments does not focus on whether profiling is possible using data collected from zero-permission sensors and classification, but rather on testing the minimum timespan needed for the profiling to be accurate. While our first target is to determine that possibility and will therefore use the full dataset collected, the second set attempts to perform the same procedure, but with increasingly smaller timespans. By doing so, we will determine how fast services can accurately profile a user and to what degree of confidence. A description of the experiments participating in this exploration can be found in Section 5 below.

3.1.4 Exploring and Analyzing the Trade-Offs Associated with Reducing the Timespan of the Data on the Profiling Accuracy

While our second set of experiments focuses on reducing the timespan of the dataset being fed to a classifier, the third set of experiments focuses on reducing the actual sensor data being fed to said classifier. Through that, we will determine which sensors have the largest impact on accurate user profiling, reducing the need for services to collect data from sensors that not only may end up being useless to the profiling use-case, but may potentially be introducing extra "noise," unnecessarily confusing the classifier. By determining the minimal set of sensors needed for profiling a user, we also reduce the overhead that the services must do to

properly classify and profile a user. A description of the experiments participating in this exploration can be found in Section 5 below.

3.1.5 Exploring and Analyzing the Effectiveness of Multiple Machine Learning Algorithms on Zero-Permission Sensor Data

After surveying potential machine learning algorithms, our research will select the three most relevant algorithms for our specific use-case. Instead of arbitrarily selecting one or the other, we will pass the data through all three algorithms, and compare their acquired results. Effectively, this allows the data to be tested across different potential algorithms and determine if the augmentation of the other options helped or hindered the overall classification and profiling process. This exploration is approached through an experimental behavior across all experiments performed described in Section 5, concurrent to the factor being tested, whether it be feasibility, timespan, or sensor combination. Therefore, each of the experiments will be repeated for all three of the algorithms being assessed.

Chapter 4

Related Work

In this chapter, we discuss background information and related work in the field. The three most important points of this research are user profiling, zero-permission sensors, and machine learning on zero-permission sensors. We provide an extended discussion of each in a respective subsection, examining previous work in the field, how it compares to our own planned work, and seeking guidance from previous research. We also assess the merits and fallbacks of the topic. This is followed by a section where we focus on the combination of these concentrations: the use of zero-permission sensors in the activity of user profiling, identification, or authentication. The figures below describe the taxonomy of this section.

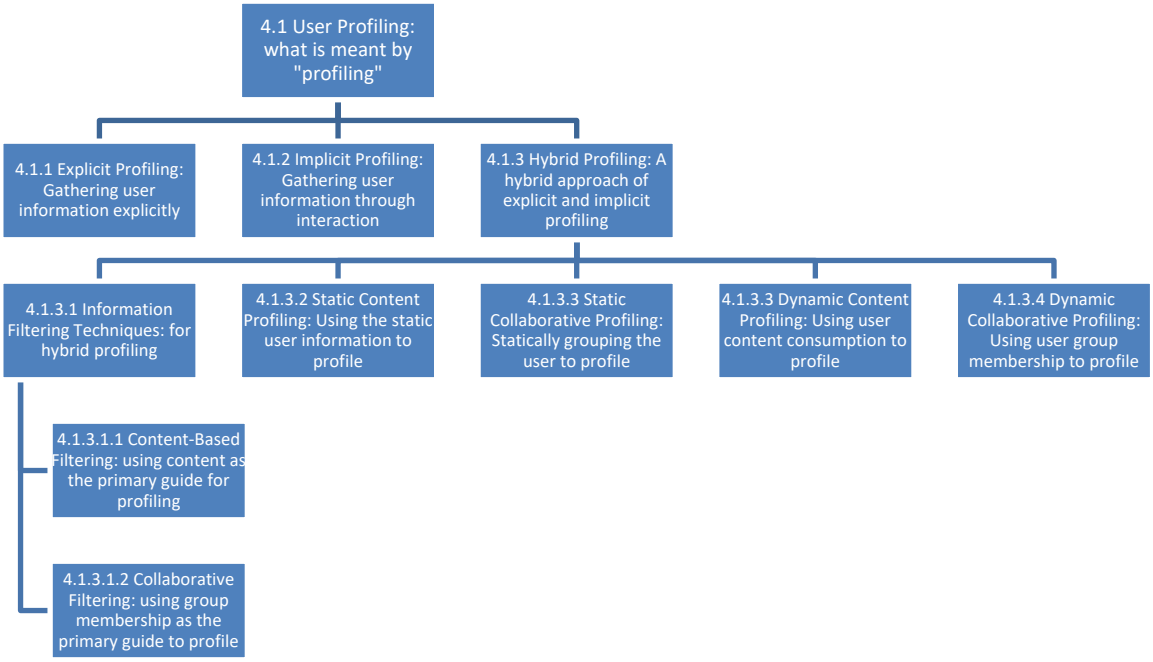

Figure 1: Taxonomy of Subsection 4.1 User Profiling

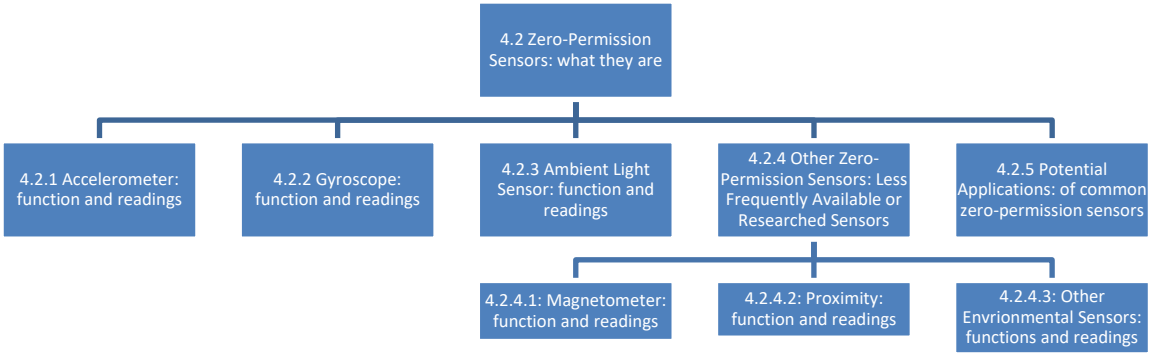

Figure 2: Taxonomy of Subsection 4.2: Zero-Permission Sensors

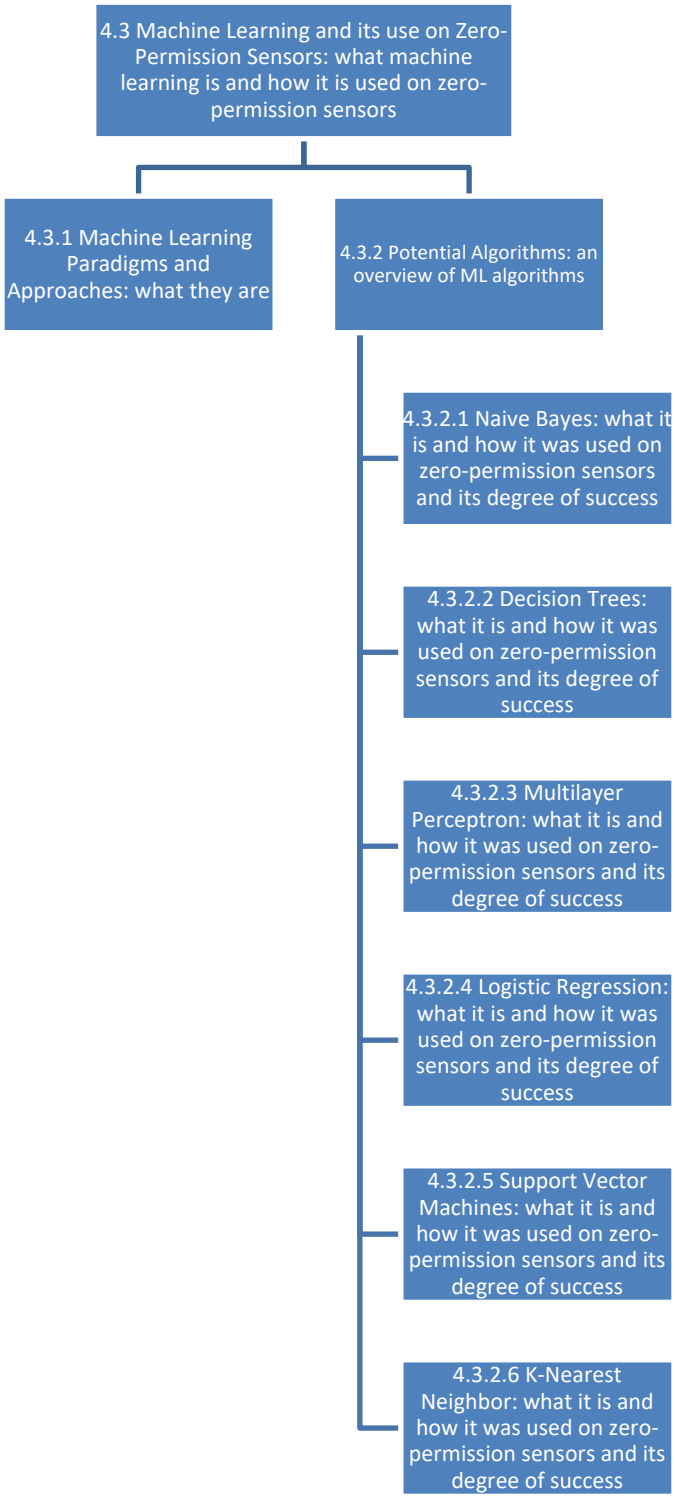

Figure 3: Taxaonomy of Subsection 4.3: Machine Learning on Zero-Permission Sensors

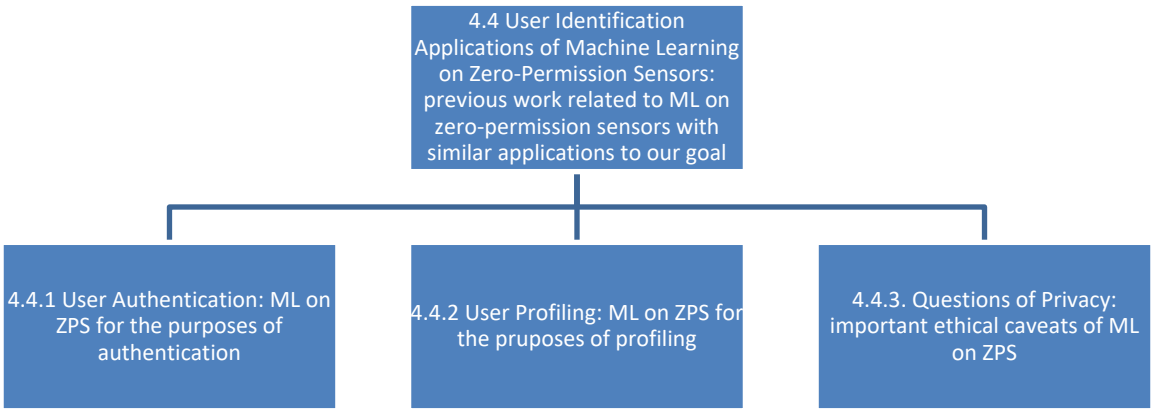

Figure 4: Taxonomy of Subsection 4.4: Work with Similar Applications

4.1 User Profiling

Kanoje et. al [33] succinctly define user profiling as "the process of extracting, integrating and identifying the keyword-based information to generate a structured profile and then visualizing the knowledge out of these findings." We certainly agree with this. While some sources, even Kanoje in different research [23], limit the profile to the "interest domain" of a user, we strongly believe that the profile itself – who the user is – informs the interest domain of the user – what the user likes –, not that it is part of it. Hence, our view of a profile is simply key data and characteristics about a user that can be used to infer other attributes. In Kanoje et al's analysis of user profiling, they break it down into three main subcategories: explicit user profiling, implicit profiling, and a combination of both called hybrid profiling [23]. We will dedicate a subsection to understanding each.

4.1.1 Explicit Profiling

Explicit or static user profiling is defined as analyzing a user's characteristics from users themselves, usually through surveys or electronic registration forms [24]. The reason this type of profiling is also referred to as static highlights one of the main problems associated with it: the profile itself does not change or evolve as the user's interests change and evolve [24]. If a user creates a profile while having no interest in sports and then develops this interest, he or she would have to go and manually change their profile to reflect this newly acquired taste. Additionally, this type of profiling is biased to a user's subjectivity and may not reflect correct values [24]. Often users may be unaware of their own potential interests, focusing only on what they know, rather than what they could potentially end up liking. This is, of course, not even considering malicious users that deliberately provide the registration system with false information for one reason or another, as discussed in the overview and motivation sections above.

4.1.2 Implicit Profiling

Implicit, or dynamic, profiling is the opposite of explicit profiling. Instead of relying on user-supplied data, the system itself is what performs the analysis of a user's activities and actions to determine the interests of a user [24]. This can also sometimes be referred to as behavioral, adaptive, or ontological profiling [23]. The main advantage with this method of profiling is that it grows and adapts to a user's interests as they themselves adapt, given that the process of profiling is a repeating effort, not performed only once. Most commonly, this method of profiling is done through continuously understanding a user's web browsing patterns and forming an understanding of the user and their interests based on said patterns [23]. In some cases, it is even possible to refine these user profiles to the degree of using them for identification and authentication, a step further than profiling [34]. Another approach is to analyze the interactions with the service for which the profile is being built. It is not difficult to imagine that YouTube will profile a user as a child if their entire watch history is children's cartoons, but perhaps more interesting is the research conducted by De Andrés et al. In their paper, they build user profiles relying on the accuracy and time between different computer mouse interaction operators. These operators were "pointing," "dragging," "key pressing," and "mental" which is used whenever the user must perform a simple decision [36]. They were indeed capable of determining the demographic profile (age and gender) of a user based solely on these interactions, showing that users of differing demographic profiles performed these interactions differently than one another [36], a very promising finding for our research, different as it is.

Two other common approaches include scrapping a user's social media [35] for information, perhaps explaining the boom in social-media login services, tracking cookies, and scrapping external web pages for information about a profile to generate another profile [35][33]. Unfortunately, all of these methods could only get as good as the source information they are scrapping from. This means that, for example, if a user has created a false social media profile and uses it to log into another service, the service's "implicitly-created" profile would also be false. Additionally, laws such as the The General Data Protection Regulation (GDPR) in the European Union and anti-tracking mechanisms in modern browsers and operating systems have made this process less possible for systems to implement, highlighting the trade-off between the privacy users covet and the personalization these same users demand of the services they use.

4.1.3 Hybrid Profiling

Hybrid profiling is the combination of either static or dynamic profiling with different information filtering techniques in recommendation systems [23] to establish a user's profile-generating attributes. To understand how this happens practically, and the different types of

hybrid profiling methods, we discuss below the various information filtering techniques that contribute to the effort of profiling a user.

4.1.3.1 Information Filtering Techniques

The method through which content is filtered in recommender systems is its heart and core. The question is a simple one: for a given user, how should content recommendations, regardless of what they may be in terms of videos, songs, restaurants, or even the social media posts that comprise the user's feed, be made? While there are many approaches, the two most interesting, widely used, and widely researched approaches are content-based filtering and collaborative filtering.

4.1.3.1.1 Content-Based Filtering

Content-based filtering focuses on the content that a user consumes and compares that to other available content. If the new content is found to have a large degree of similarity to those that the user consumes, then it becomes a viable option to be recommended to a user [24]. On a video-sharing website, for example, if a user already watches a lot of sports videos, content-based filtering would suggest recommending more sports videos to the same user. This process can sometimes be further enhanced using explicit inputs from the user. For example, Avery and Zeckhauser [25] were first to find that incorporating a user's ratings of the recommended content and using said ratings as filters for recommending additional content was a successful method. This concept later evolved by Fuchs and Zanker to allow users to not only rate the content, but to perform a multivariate rating of the content [26] to understand a user's opinion more granularly. In their case, it was focused on hotel reservations, and therefore, instead of having users rate their overall stay experience, they had users rate individual aspects of the stay such as the comfort and cleanliness to allow for recommendations to target users with similar specific criteria. We could see how a similar approach could be followed for other specific recommendations. While this concept of multivariate rating was indeed successful [26], Ramscar et al quickly proved this effort too tedious for users [27] having to perform these multivariate ratings for each piece of content consumed. This is perhaps what gave rise to research in the area of implicit rating and filtering, and the associated well-cited patents [29]. Some ways through which this can be accomplished is to identify specific actions that a user would naturally take to indicate a positive rating, such as saving, completing, repeated viewing, etc., and their opposite, such as deleting, non-completed viewings, etc. to indicate a negative rating [28].

4.1.3.1.2 Collaborative Filtering

In collaborative filtering, unlike content-based filtering focusing on content similarity as the

criterion through which other content is recommended to a user, the users themselves are placed in similarity groups. This is usually done by clustering the profiles. The algorithm then expects users in the same group to have similar interests, thereby if enough users in a group find a piece of content interesting, it is recommended to other users in a group [24]. For example, if user A is in the same similarity group to user B due to some shared attributes (or perhaps event content consumed), user A would be recommended content that user B has found interesting, and vice-versa. Interest, itself, is expressed through similar means as in content-based filtering: either explicit or implicit rating. While there are many different approaches to improving the accuracy of collaborative filtering [30][31][32], the key challenge facing this filtering approach is the method through which the users are clustered, providing for the most similarity between users of the same group, and enough dissimilarity between users of different groups [24].

4.1.3.2 Static Content Profiling

In static content profiling the focus is on collecting and maintaining user information [24]. A simple snapshot is taken of a user's content consumption in time and used to profile the user once. The disadvantage of that is, as in simple explicit profiling, there is no real growth or development to the user profile over time. If a user happened to join a video-sharing platform, for example, to view children's cartoons, and that is when the profiling was done, the user would be profiled as a child, despite them not actually being one. The advantage is that this type of profiling is easy to perform and not at all, resource intensive.

4.1.3.3 Static Collaborative Profiling

In static collaborative profiling, more information goes into profiling the user. A user is explicitly stated as being part of the group, whether by choice (through social media group memberships, for example) or assignment (through being placed in a specific administrative group or age group, for example). The interests of the group indicate the potential interests for the user and help inform his or her profile [24]. The advantage with this approach is that it does help take the user out of their own personal "bubble." If a user is interested in one type of content, he or she will also be shown additional types of content that are consumed by the specific group. The disadvantage here is that, again, it is static. Users do not "change" their collaborative group, despite their actual views or opinions changing. As an illustrative example, let us consider a hotel booking system. A user may be determined to be interested in luxury hotels due to their affluence or previous reservations. With static collaborative profiling, if the user were to lose their job, they would continue to receive recommendations of hotels that people of the same group were reserving – likely targeting a luxurious standard over economical options – despite no longer being applicable. Another disadvantage is that, while this approach takes the user out of their own personal bubble, it puts them in the "group

bubble," failing to recommend other pieces of content that may provide a viewpoint different from their own, further inciting divisionism.

4.1.3.4 Dynamic Content Profiling

Dynamic content profiling is different. It combines the virtues of dynamic profiling with the prowess of content-based filtering. Basically, a user's interests are determined based on the content that they had previously consumed. As the content changes or shifts from one category to the other, so does the content [24]. One main disadvantage of this profiling method is that it does not expose the user to content that may be interesting but does not strictly fit their profile.

4.1.3.5 Dynamic Collaborative Profiling

Dynamic collaborative profiling acts similarly to dynamic content profiling but for a group, instead of a specific individual. The interests of the group are the sum of the interests of the individuals within that group [24]. This removes this "bubble" that a user is placed in through content profiling, exposing the user to other content that may be interesting based on the interests of the group. Additionally, this type of profiling addresses the fallbacks of dynamic collaborative profiling through making the group determination itself dynamic. The user can "flow" between groups and different interest sets, as their interests and affiliations change and develop over time.

4.2 Zero-Permission Sensors

Zero-permission sensors are sensors that do not explicitly require the user's permission to be used by an application. Applications could be running in the background completely undetected and without requiring any input from the user, monitoring these sensors, and collecting and/or sharing their readings with the cloud [38].

4.2.1 Accelerometer

The accelerometer is a device used for measuring linear acceleration in meters per second squared along a moving body [39][40]. In this particular case, we are focusing on tri-axial accelerometers, measuring linear acceleration along the three axes: x, y, z. In Figure 5 below, the accelerometer measures acceleration along the axes themselves.

4.2.2 Gyroscope

The gyroscope is a device used for measuring angular acceleration in radians per second along

a moving body [39][40]. Again, we are directing our interest to triaxial gyroscopes measuring rotation along the three axes: x, y, and z. The difference between the accelerometer and gyroscope is perhaps best illustrated by Figure 5 [39]. The gyroscope measures rotation along the three axes in the figure.

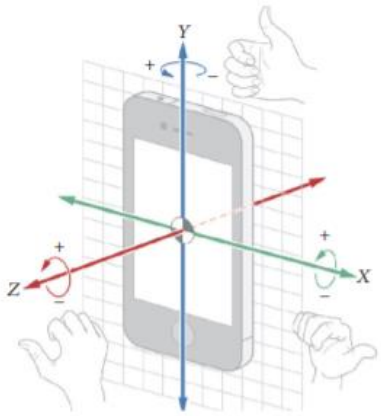

Figure 5: Accelerometer and gyroscope measurements on a mobile device. The three axes intersecting with the center of the device represent the readings by an accelerometer along the physical axes. The three circles along the tips of the axes represent the gyroscope readings along the three axes.

4.2.3 Ambient Light Sensor

The ambient light sensor measures the illuminance in a device's environment [42] in lux, which is the standard unit of illuminance in the SI system.

4.2.4 Other Zero-Permission Sensors

There are other zero-permission sensors in some devices, but they are less common and have significantly fewer applications. In our research, we have elected not to use them, however explain their usages here for deeper understanding of their potential roles in related works or future works. We also include these sensors and their definitions to highlight the multitude of potential data that could be gathered or otherwise used, without the user even being made aware of the fact. Other sensors, such as step detection for example, require the user to allow the device [41].

4.2.4.1 Magnetometer

A magnetometer measures the strength of the ambient geomagnetic field, again along the three physical axes [42]. Combined with software algorithms, these values can indicate which way the phone is facing.

4.2.4.2 Proximity

A proximity sensor measures the distance from the sensor in centimeters [41]. This sensor is typically placed close to the earpiece to determine whether the device is being held up to the person's ear.

4.2.4.3 Other Environmental Sensors

There are also other zero-permission sensors that may be placed in a user's device that potentially gather data about a user's environment [42]. Those include the device and ambient temperatures in degrees Celsius, the ambient air pressure in hectopascals, and the relative humidity in percentage [42].

4.2.5 Potential Applications

The potential applications of the most common Zero-Permission sensors include using the accelerometer and gyroscope as game controllers, orientation identifiers, and the recognition of physical human activity [39][40]. An ambient light sensor can be used to detect the surroundings of the device to determine whether it is in a dark environment (such as a dark room or a pocket) or a well-lit environment [42].

4.3 Machine Learning and its use on Zero-Permission Sensors

In this section, we provide a discussion of machine learning in general, before surveying a number of specific and prominent examples of applying different machine learning algorithms to zero-permission sensor data.

4.3.1 Machine Learning Paradigms and Approaches

In his book, Introduction to Machine Learning, Alpaydin [37] defines machine learning as "programming computers to optimize a performance criterion using example data or past experience." He continues to explain that there are three major paradigms of machine learning: reinforcement, unsupervised, and supervised. Reinforcement learning allows the machine to be able to generate actions based on the "utility" of such actions. It provides a method to the machine to be able to assess the benefit of an action taken under particular circumstances, and the machine learns to maximize this benefit. Artificially intelligent non-playable characters (NPCs) in a game, for example, could be taught to maximize interference with a player's objective to increase the difficulty of a game. In unsupervised learning, the only data available is input, without any associated output. Therefore, the learning process becomes one of grouping similar data to each other – or clustering – without being able to "label" this data. If the data is of a constrained number of classes, this labeling can happen manually following the automated clustering performed by the algorithm. In supervised learning, the aim is to map an input to an output, all as a supervisor

provides the correct mapping for a subset of the inputs. In that sense, the criterion being optimized is the error between the correct input-to-output mapping, and the mapping provided by the algorithm. Perhaps the most discriminating aspect for the type of learning to be used, and then the method of learning, is the input data. Several aspects of the data contribute to this decision: its type – textual, graphical, audible, unstructured –, the volume of the data – do we have large amounts of data or just a few examples –, its labels – whether the data is or can be labelled or not –, and the density of the data – the volume per class –. Therefore, it becomes important to our research to understand doing machine learning in the specific context of the type of data that we will be dealing with. Already, before diving too deep into the potential algorithms and their particularities, we can see that mapping inputs (zero-permission sensor data) to outputs (a user's biological gender) is the aim of our research, and therefore we chose to select supervised learning as our selected approach.

4.3.2 Potential Algorithms

Seeing how popular it has become of a field in recent times due to the advent and increased popularity of smartwatches and fitness trackers, and the relative affordability of the sensors and the computing power to perform the classification on those sensor data, most researchers focus their zero-permission sensor machine learning efforts in the area of activity recognition and classification. While the applications of their classification efforts are different from our own, even ranging to working with animals instead of humans, they are still interacting with very similar data from the same set of sensors. For this reason, we survey a number of telling experiments to determine the potential machine learning algorithms we may use and the approaches taken to that effort. We remain, however, mindful of the difference of application. We explore experiments with similar data that focus on applications like our own – user profiling, identification, and authentication – in section 4.4.

4.3.2.1 Naïve Bayes

The naïve Bayes classifier performs its classification through relying on the Bayesian theorem which states

$$p(c_j|d) = \frac{p(d|c_j)p(c_j)}{p(d)}$$

given that p(c_j | d) is the probability that d belongs to the class c_j, p(d | c_j) is the given probability of d being in class c_j, p(c_j) is the probability of c_j occurring in the dataset, and p(d) is the probability of d occurring in the dataset [85]. A more straightforward way to think about this is that the naïve Bayes algorithm is able to predict the probability that a given feature belongs to a specific class by assuming the effect of a particular attribute on the feature, ignoring the effect of

other attributes and accounting for the probabilities of the class and the feature occurring [43].

In their experiment, Joshua et al used the naïve Bayes method to attempt to classify construction activities on data collected from accelerometers placed at mason workers' waists. After filtering their features for the most distinctive and testing multiple machine learning algorithms, they were able to reach a maximum accuracy of 70.33% using naïve Bayes [43] as their classifier. As seen in the following subsections, they were able to obtain better results using other classifiers.

Aziz et al tried to detect falls by an accelerometer also worn around the waist. Naïve Bayes was one of the classification algorithms they tested, and it provided the highest sensitivity (recognizing a fall), and a 91% specificity (correctly recognizing a non-fall) [50], the lowest score by comparison to the other tested algorithms. This is a possible indication of the sensitivity of the naïve Bayes algorithm to abrupt motion, which could potentially skew results. Overall, it shows that naïve Bayes may be very prone to false positives.

Albert et al also tried to detect falls, however this time with smartphone accelerometers, instead of a dedicated one worn on the subject. This level of device freedom is more relevant to our experiments, despite the difference in application. Their results seem to confirm Aziz et al's findings that naïve Bayes was poor at fall detection, as it was only accurate 66.3% of the time [52], a large difference from its next competing algorithm.

In their research, Wu et al attempted to classify thirteen different physical activities – mainly jogging, walking different speeds, sitting, going upstairs and downstairs at different speeds – using gyroscope and accelerometer readings from an iPod Touch. The device was placed, strapped in an armband for the jogging activities, and in the subject's front shorts pocket for the other activities. For the weighted average of all the activities being classified, naïve Bayes had a 63.2% accuracy, the lowest score of all evaluated algorithms [51].

Similarly, Yin et al attempted to classify the activities of walking, running, and sitting using a loosely placed smartphone in the subject's pocket. They assessed multiple machine learning algorithms to perform their classification, one of which was the naïve Bayes. They found that it had the least accuracy among the tested algorithms. That least accuracy, however, was at a whopping 99.1645% [85], displaying that while it was the least accurate of those tested, it still performed an excellent job at the task of differentiating between the different activities that were being classified. It is, however, important to note that Yin et al's feature selection algorithm was questionable, choosing to select every 20th feature without justifying this approach. Additionally, no mention of the number of participating subjects is made in the paper. The difference in results between Wu et al and Yin et al's experiment may be attributed to a few factors, such as the experimental setup or the difference in time between the two papers allowing for better-developed technologies perhaps with more sensitive accelerometers.

Edeib et al attempted to detect falls – either forward, to the right, or to the left – using data collected from the triaxial accelerometer, combined with an additional data point that aggregates the data from the three axes for the total magnitude of motion. In their experiment, they found that naïve

Bayes performed the worst at the task, reaching an accuracy of 91% [91], relatively low by comparison to the other algorithms they tested.

Palmerini et al also attempted fall detection experiments. In them, they relied exclusively on accelerometer data but also fed to multiple machine learning algorithms, one of which was naïve Bayes. They found that it provided an accuracy of 99.6% [94], sharing the highest accuracy with the logistic regression model, contrary to Edeib et al.

Continuing with the application of fall-detection, in Al Nahian et al's experiments, they used the accelerometer readings from three open-source datasets: UR Fall, MobiFall, and UP Fall. They also used six different machine learning algorithms to detect and classify a falling action, one of which was naïve Bayes. For the UR Fall dataset, Naïve Bayes shared the spot of highest accuracy at detecting the falls with random forests at 99%. With the UP Fall dataset, it had a middling 97% accuracy. Finally, with the MobiFall dataset, seemingly the most optimized for fall recognition, it scored equally to all of other algorithms at 99% accuracy [100]. This gave it an average, across the datasets, of 98.33%, sharing second place with the logistic regression algorithm.

4.3.2.2 Decision Trees and Random Forests

Decision trees are simple trees on which each node represents a test of a specific attribute, each branch an outcome of said test, and each leaf a classification. These classifiers are simple, fast, and reasonably accurate, explaining their popularity [43]. This form of classification is illustrated in a simple example classifying whether or not a person should play soccer based on weather attributes in Figure 6 below.

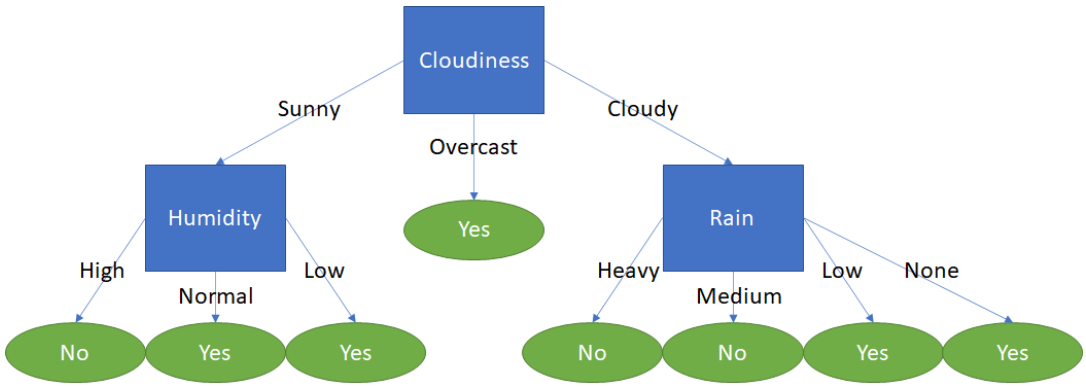

Figure 6: Sample decision tree illustrating the classification for whether or not someone should play soccer, to illustrate how a decision tree works

In the aforementioned construction activity classification experiment conducted by Joshua et al, they also tested the performance of decision trees. They were able to obtain a maximum accuracy of 76.67%, a significant improvement over naïve Bayes' 70.33% [43].

In Aziz et al's fall-detection experiment, the sensitivity and specificity of decision trees were also measured. It provided a 94% sensitivity, a mediocre score relative to the other algorithms, and 96%

specificity, equal to the best score [50]. Again, these results are confirmed by Albert et al, showing decision trees to have an accuracy of 95.9% at detecting falls [52]. While that makes it the second worst, the difference between it and the worst evaluated (naïve Bayes at 66.3%) is significant, and places it in the range of very accurate.

Althobaiti et al also used decision trees as one of the assessed machine learning algorithms in their fall detection experiment. Their experiment, however, was unique in the sense that it not only attempted to classify falls, but also 6 other human activities, including jumping, lying down, bending, sitting, standing and walking. This means that they were able to provide two accuracies from their experiments for each of the assessed algorithms, testing whether or not the activity was classified correctly, and whether or not the activity was a fall or not. Decision trees provided a middling performance at detecting whether or not the activity was a fall at 97.14%. The middling performance continued at detecting the actual activity itself with an overall accuracy of 91.24% [96].

In Wu et al's aforementioned human activity classification research, decision trees scored a weighted average of 83.0% across the activities, a relatively mediocre score [51].

Ali et al also attempted to recognize human activity – particularly sitting, walking, jogging, and falling – with the J48 algorithm generating decision trees. The trees were trained on the data coming from the embedded accelerometer in a phone placed in the subject's upper right pocket. They were able to achieve accuracies of 82.76%, 69.56%, 70.56%, 60.15% for each of the aforementioned activities respectively [97]. Again, this is a very mediocre score, with an overall average accuracy of 70.76%. It is unfortunate that they did not attempt other machine learning algorithms to get a better sense of whether the issue was their data, the model, or the way they trained it.

Random forests are a specific variation of decision trees. In it, instead of calculating every potential decision in a single tree as in classical decision trees, multiple decision trees are generated where the decisions to be considered are selected randomly but the progression is only on the best decision. This makes the classification significantly more efficient, and the accuracy loss is not huge due to statistical laws [46]. Further still, extremely randomized trees are another variation very similar to random forests. In extremely randomized trees, however, the progress from one node to another is not based on the best result, but rather also randomly selected [47].

Zdravevski et al attempted an experiment to automatically identify intended jogging periods using accelerometers. In their experiment, they used both random forests and extremely randomized trees. They were able to find that random forests were not as successful as the other classification methods they were evaluating, however extremely randomized trees were the most accurate when provided data from the respective favored hip of each subject, or data from the respective favored hip and favored ankle of each subject [44]. This is a strong indication to the viability of ERT as a classification algorithm for accelerometer data.

Fattahi et al attempted to use random forest classifiers to assess driver behavior for aggressive maneuvers including careless driving, aggressive lane changing, and tailgating. Although their application is quite different from ours, the methodology of applying a machine learning algorithm

on accelerometer data is the same. They found that they were able to reach an overall accuracy of 89% [80]. It is important to note here, however, that they did not only employ random forest classifiers to conduct their prediction, but rather also included a second step of using a multilayer perceptron – discussed in the following subsection – without discriminating the results of the random forest classifier alone. Additionally, they also used data coming from the OBD-II of the car, which they found to be more relevant to the classification [80] than those coming from the smartphone sensors, putting into question both the data fed to the model, and the model itself.

B.R. dos Reis et al. also experimented with an application different from ours, but with a similar methodology. They attempted to classify cattle activities – grazing, lying, resting, and walking – using a triaxial accelerometer, gyroscope, and magnetometer. They also evaluated different machine learning algorithms and found that the random forest classifier was best, among those evaluated, at the classification performance, identifying grazing at a 93% accuracy, lying at 92% accuracy, walking at 94% accuracy, and resting at 92% accuracy [81].

Riaboff et al also attempted to classify cattle activities, same as dos Reis et al, but with some additional activities for ruminating. They tested multiple machine learning algorithms and were able to reach an accuracy of 97% with random forests [93]. Among those tested, this was a middling performance, neither providing the best nor the worst classification accuracy.

Abdull Sukor et al focused on comparing different machine learning algorithms for classifying six human activities: standing, sitting, lying, going upstairs, going downstairs, and walking. One of the assessed machine learning algorithms was decision trees. They found that they were able to reach a 95.36% accuracy with the original accelerometer features, and an even better 96.85% with a dimensionality reduction process they performed [87].

Shakya et al conducted an in-depth analysis of comparing the capability to recognize human activities between various machine learning and deep learning algorithms. The aim was to compare the performance of decision trees, random forests, K-nearest neighbor, convolutional neural networks, and recurrent neural networks at recognizing activities such as jogging, lying down, sitting, going upstairs, standing, walking, biking, and going downstairs [88]. They also conducted these experiments with two different datasets. In their experiments, they found that decision trees were able to reach an accuracy of 90.57% for the first dataset and 87.96% for the second dataset [88], to an average overall accuracy of 89.27% across both datasets. With their experiments with random forests, they were able to reach accuracies of 95.77% and 89.72% for the first and second datasets respectively [88]. The overall average accuracy between both datasets was 92.75%. This shows a significant improvement while using random forests over decision trees, perhaps due to their increased complexity.

Gomes et al chose to focus on a different aspect of activity recognition. Instead of directing their attention to the type of activity that was being conducted, instead they focused on classifying the intensity of the activity being conducted, varying from light to moderate to vigorous. In their experiments, they assessed the activity intensity using three different machine learning algorithms on accelerometer data collected from a smartphone placed at the subject's waist. They also

attempted two different alternatives: the first was one classifier being used to detect both the activity itself and the intensity, the second was using one classifier for each of the classification tasks: the activity and its intensity. One of the algorithms tested was random forests. They found that random forests were correctly able to identify the activity along with the intensity of the activity 77% of the time [95], placing it right in the middle between the k-nearest neighbor accuracy and the support vector machine accuracy. When classified separately, they found that random forests performed the best at recognizing the activity being conducted (at 97%) and the worst at recognizing the intensity of the activity (at 79%) [95]. Calling it the best and the worst, however, is misleading, given that it only had one percentage point difference from K-Nearest neighbor which was the worst at recognizing the activity conducted and the best at recognizing the intensity of said activity [95].

In Edeib et al's fall detection experiment, another one of the machine learning algorithms they tested was decision trees. They found that it performed best at the task with an overall accuracy of 97%, beating naïve Bayes' 91% and SVM's 95% [91]. In their fall-detection experiments, Palmerini et al found that random forests provided a middling performance, with an accuracy of only 98.9% [94], relatively low in comparison to the other models they tested.

Al Nahian et al used both decision trees and random forests to detect falls from accelerometer readings in three datasets. While decision trees shared the position of least accurate with many of the other machine learning algorithms at 97% for the UR Fall dataset, random forests on the other hand had the best classification accuracy for the same dataset at 99%. This pattern continued with the UP Fall dataset, with decision trees providing the least accuracy of all assessed algorithms at 96%, while random forests shared the position of second most accurate with other algorithms at 98%. With the MobiFall dataset, both random forests and decision trees had an accuracy of 99% [100]. Across the three datasets, decision trees had an overall accuracy of 97.33%, the least accurate at detecting a fall from the three datasets, while random forests had an overall accuracy of 98.67%, the most accurate at detecting a fall across all three datasets.

4.3.2.3 Multilayer Perceptron

A multilayer perceptron is a neural network in which different "neurons" (input-output units) are connected in a network. Each connection has an associated weight which is adjusted during the learning phase, and a specific calculation is performed inside each neuron to determine its result. Data features act as inputs and the potential classification labels are the outputs [43]. It is trained through backpropagation of the errors and adjusting the weights between the connections [85].

In the masonry classification experiment, the researchers also tested the multilayer perceptron as a potential classification method. Their result of 79.83% accuracy proved to be the most accurate of the three classification methods tested [43].

Wu et al also evaluated the multilayer perceptron in their activity classification research. It scored a weighted average of 83.4% across the different activities, very similar to decision trees, and an

overall mediocre score.

As aforementioned, Fattahi et al also used a multilayer perceptron as a second "layer" to their classification of erratic driver behavior after random forest classifiers. They were able to reach a rather impressive 89% accuracy, but again they were primarily using the data coming from the OBD-II port in the car, reducing the impact of the smartphone sensors [80].

Yin et al found the most success at their attempt of classifying walking, running, and sitting activities with the multilayer perceptron, reaching an accuracy of 99.8956% [85].

Among the different machine learning algorithms Abdull Sukor et al tested to recognize six common human activities was a neural network, a specific kind of a multilayer perceptron. When using the original features, prior to a dimensionality reduction process they applied, the neural network was able to reach an accuracy of 97.54%, the highest of the assessed algorithms [87]. Interestingly, after the application of their dimensionality reduction process, the neural network was able to classify the activities with 100% accuracy [87]. This is an extremely interesting figure, encouraging the use of neural networks for the recognition of human activity.

Shakya et al in their comparative analysis of traditional machine learning and deep learning algorithms did not suffice with just testing one deep learning algorithm. Instead, they tested both convolutional neural networks and recurrent neural networks for their classification of human activities. They found that convolutional neural networks were able to reach an accuracy of 92.22% for the first dataset and a very impressive 99.12% for the second dataset [88]. This provides an average accuracy of 95.67% across both datasets. Meanwhile, recurrent neural networks performed worse than convolutional neural networks. They provided accuracies of 81.74% and 95.65% for the two datasets respectively, with an overall accuracy of 88.7%. This shows that, for the purposes of recognizing human activities from triaxial accelerometer data, convolutional neural networks will perform significantly better than recurrent neural networks.

Ferrari et al also performed a similar comparative analysis of traditional machine learning and a deep learning algorithm. Their deep learning approach of choice was to use transfer learning on the ResNet Convolutional Neural Network. Their analysis was also very comprehensive, having tested three different datasets of accelerometer and gyroscope data, both in conjunction and separately. They found that ResNet performed the best on the accelerometer-only alternative providing 90.73%, 92.98%, and 99.47% for the UCI-HAR, MobiAct, and MotionSense datasets respectively. The Convolutional Neural Network continued its superior performance with the gyroscope-only alternative, providing accuracies of 89.36%, 96.09%, and 98.07% for each of the three datasets respectively. Interestingly, when the combination of accelerometer and gyroscope readings were used, ResNet performed best with only two of the datasets with accuracies of 96.46% and 99.08% for the UCI-HAR and MotionSense datasets respectively. k-nearest neighbor performed better with the MobiAct dataset at 94.20% accuracy compared to ResNet's 92.94% accuracy [99]. Overall, ResNet had an average accuracy across the three datasets of 94.39% for the accelerometer-only alternative, 94.51% for the gyroscope-only alternative, and 96.16% for the combination of accelerometer and gyroscope readings. This provides an overall accuracy, across

the datasets and the combinations, of 95.02%, an extremely impressive result.

4.3.2.4 Logistic Regression

Logistic regression is a probabilistic model, similar to others like naïve Bayes and support vector machines, used to determine the probability of a specific event. It is binary in nature, however multiple combinations of it can allow for classification across several classes [45]. Its main advantages are its simplicity of understanding, and speed [44].

In Zdravevski et al's jogging classification experiment, they employed logistic regression as one of the evaluated classification models. It succeeded in being the best classification model when all of the data, regardless of subject favorability, was fed to the classifier, providing a whopping accuracy of 99.90% [44].

In the fall detection experiment by Aziz et al, logistic regression scored 95% on sensitivity and 92% on specificity [50], comparatively mediocre scores in both categories. This is again confirmed by Albert et al in their own research, showing logistic regression as having a 98.0% accuracy at fall detection [52]. While that only puts it at an 0.2% disadvantage from being equal to the top accuracy, it is also just a 0.1% difference from third place, confirming its relative mediocrity.

In Al Nahian et al's fall detection experiments, logistic regression was also used. It provided – along with other assessed algorithms – the least accuracy of 97% for the UR Fall dataset. On the other hand, for the UP Fall dataset, it actually was the most accurate at 99%. With the MobiFall dataset, it – and the other machine learning algorithms – had 99% accuracy [100]. This gives it an overall average accuracy across the different datasets of 98.33%, a respectable second place shared with naïve Bayes.

The classification mediocrity continues in Wu's research, with logistic regression scoring a weighted average of 77.2%, the second worst score of the evaluated algorithms, and with a significant 13% gap between it and the most accurate evaluated algorithm [51].

But contrary to both aforementioned researches, Palmerini et al found that logistic regression shared the position of best overall accuracy at fall detection with naïve Bayes, at 99.6%.

B.R. dos Reis et al's experiment for classifying cattle activities also used logistic regression and although they did not report the specific accuracies obtained by using logistic regression, they did report that logistic regression performed the worst among the three evaluated machine learning algorithms [81].

Fang et al attempted to classify human activity but were particularly focused on the subjects getting on and off a bus. In order to do so, they provided users with a smartphone that they could place anywhere on their body as they undertook the action, and labeled seven particular activities – subjects being stationary, walking, running, going upstairs, going downstairs, going up a bus, and going down a bus [84]. They attempted their classification through multiple machine learning models, one of which was logistic regression. They found that among their selected algorithms, logistic regression performed the worst, at 83.9% accuracy, relatively low to the other algorithms. They do, however, note the freedom of movement of the device introducing potential noise that

may have confused the machine learning algorithms. While they did perform extensive feature selection, they do mention in the limitation of the study that they were unable to find a feature that specifically targeted the task they were attempting. Perhaps a more refined feature extraction method would have provided better results.

4.3.2.5 Support Vector Machines

A support vector machine (SVM) is a supervised machine learning algorithm that relies on calculating a "decision boundary" between different classes to be able to classify a sample. It attempts to cluster the data into groups, drawing "lines" between the different groups, to attempt to understand how to classify a newly incoming datapoint [48].

SVMs were also evaluated by Zdravevski et al in their jogging research. They found that SVM provided the highest overall accuracy, a 99.94%, but only when fed the data from the favored ankle of the subject [44]. This may indicate that while SVM could be capable of providing the best overall accuracy, it may also be the most sensitive to data noise, requiring the features of the input data to be well-filtered and well-selected.

In the fall detection experiment by Aziz et al, SVM scored the highest score on specificity, 96%. While this score is equal to other classification algorithms evaluated in the same experiment (decision trees and k-nearest neighbor), it had the highest sensitivity of its equal competitors, also 96% [50]. Albert et al also confirm SVM as the best for detecting falls, finding it at a 98.2% accuracy at the task [52].

In Palmerini et al's own fall detection experiments, support vector machines continued to shine, providing a 99.3% accuracy [94] at correctly classifying a fall from accelerometer data, only a 0.3% difference from the highest accuracies achieved by both logistic regression and naïve Bayes.

Al Nahian et al's fall detection experiments also used support vector machines as one of the tested algorithms. Using the accelerometer readings from the UR Fall dataset, support vector machines were capable of classifying a fall accurately 97% of the time, sharing the position of least accurate. With the UP Fall dataset, SVMs shared the position of second most accurate at 98%. When the MobiFall dataset was used, SVMs were 99% accurate at detecting a fall, equal to the other assessed algorithms [100]. This gave SVMs an overall accuracy across the datasets of 98%, putting it squarely near the bottom at third most accurate (out of four) along with k-nearest neighbor.

In Althobaiti et al's activity recognition experiments, of which a fall was one, support vector machines provided the best accuracy at detecting whether or not the activity was a fall at 98.48% [96]. It also continued to shine at detecting the actual activity being conducted, reaching an accuracy of 93.33%, only 0.41% away from the best accuracy achieved by the linear discriminant analysis algorithm [96].

Kilany et al performed a human activity classification experiment, not unlike Wu et al's. While Wu and his team did not test SVM as one of their classification algorithms, Kilany only used SVM as his classification algorithm. Between different trials, he was able to show that SVM had an accuracy between 85% and 93% at classifying human activity [53]. While those numbers, if compared

directly to Wu's research, seem to indicate that SVM could potentially have the highest accuracy at detecting human activity, it is important to remember that Kilany's research came a few years after Wu's and used different devices. The increase in accuracy is not necessarily attributable to the advantage of using SVM over the other learning algorithms, but could be a symptom of technological progress. Only Ren et al actually compared SVM and KNN (the most accurate algorithm in Wu's research [51]). Their findings provide Kilany et al's merit, showing SVM scoring consistently higher than KNN.

B.R. dos Reis et al also evaluated SVMs and found that it had a middling performance in the accuracy of classification [81], relative to the superior random forests and the inferior logistic regression. Riaboff et al confirmed the same with their own cattle activity recognition experiment, finding SBMs to provide an accuracy of 97% [93], close to being the best of the models tested, but being beaten by 1% by the superior XGB algorithm.

Similar to dos Reis' experiment, Yang et al attempted to classify broiler – chickens raised particularly for meat production – activities such as resting, walking, feeding, and drinking through machine learning on triaxial accelerometer data. One of the machine learning algorithms they tested was support vector machines. They experimented with different sampling rates, but the average accuracies across the different rates for the different activities were 96%, 99%, 90%, and 92% for resting, walking, feeding, and drinking respectively [89] to an overall mean accuracy across the sampling rates and activities of 94.25%. This was significantly better than the other machine learning algorithm tested.

Using support vector machines, Ahmed et al attempted to use triaxial accelerometer and triaxial gyroscope data to predict a host of human activities, both static and dynamic. They were table to reach a 97.93% accuracy for walking on a level surface, 97% accuracy for walking downstairs, 95.45% accuracy for walking upstairs, 98.34% accuracy for standing, 98.31% accuracy for sitting, 97.3% accuracy for lying, 95.74% for the motion from standing to sitting, 95.70% for the motion from sitting to standing, 96.77% for the motion sitting to lying, 94.74% for the motion from lying to sitting, 96.67% for the motion from standing to lying, and 97.75% for the motion from lying to standing. Overall, this provides a combined average accuracy across the twelve classified activities of 96.81%, a very impressive and promising figure [82].

It is important, however, to note three factors that bias these results, even if by a small amount. First of all, Ahmed et al conducted these experiments in a controlled laboratory environment, while the subjects were being observed, which alters human behavior into a more controlled rather than erratic one. Additionally, the sensors – placed in a Samsung Galaxy S II phone – were mounted specifically at the subjects' waists, not allowed any degree of freedom as in our proposed experiments. Finally, Ahmed et al performed very specific feature selection, without which they still had impressive accuracies for the twelve activities (an average of 90.84% [82]), but less so than with the combination of the feature selection.

Fang et al's bus ascension and descension experiment also used support vector machines and found that they provided a superior performance with an accuracy of 88.9% to logistic regression's

83.9% accuracy [84].

Yin et al's experiment classifying walking, running, and sitting used Support Vector Machines to a success rate of 99.4526%. While that is an extremely high accuracy, SVMs only accounted for the position of second best, behind the multilayer perceptron, and ahead of the naïve Bayes.

The final algorithm Abdull Sukor et al tested for recognizing the most common human activity was support vector machines. With the original accelerometer features, the SVM was able to reach an accuracy of 90.19% [87]. This number again faced an increase to 92.87% after their dimensionality reduction process. This process caused an increase in accuracy across all three of the tested machine learning models [87], giving merit to this process as a potential data pre-processing step that would help our selected models improve in accuracy.

The most interesting aspect of Yin et al and Abdull Sukor et al's respective researches, however is that they confirm each other's one particular finding. In both sets of experiments, support vector machines are only outperformed by the multilayer perceptron. However, seeing as the multilayer perceptron could be considered a more modern deep-learning approach, this places support vector machines as the best performing classical machine learning algorithm assessed in both sets of experiments.

Continuing with the pattern of Human Activity Recognition (HAR), Husain et al's experiment relied on data coming from accelerometer and gyroscope to classify activities using support vector machines. They were able to reach an overall accuracy of 95% across the activities being classified [92].

In Ferrari et al's comparative research between traditional machine learning algorithms and deep learning, one of the selected machine learning algorithms was support vector machines. They found that, while it performed poorer than transfer learning on ResNet, it provided a performance quite similar to the other assessed machine learning algorithm: k-nearest neighbor, albeit never beating ResNet like k-nearest neighbor did. Support vector machines provided accuracies of 79.51%, 77.93%, and 90.04% for the accelerometer-only readings from the UCI-HAR, MobiAct, and MotionSense datasets respectively [99]. This provides an average overall accuracy of 82.49%. The gyroscope-only readings from the three datasets provided accuracies of 72.93%, 64.19%, and 86.92% respectively [99] with an overall average of 74.68%. Combining both the accelerometer and gyroscope datapoints yielded accuracies of 86.83%, 79.13%, and 85.87% respectively [99] with a combined average of 83.94%. Across the three datasets and combinations of sensors, support vector machines had an overall accuracy of 80.37%. Another interesting pattern noticed is that when data from only one sensor was used, support vector machines consistently performed best with the MotionSense dataset, and worst with the MobiAct dataset, with the UCI-HAR dataset resting in the middle. This pattern changed only when data from both sensors were used, with the UCI-HAR and MotionSense switching positions (but only by a margin of 0.96%), with MobiAct continuing to be the least performing.

Gomes et al's experiment with detecting the intensity of the activity, rather than the actual activity being conducted, also used support vector machines as one of the machine learning algorithms

being tested. They found that SVMs provided the least accuracy at detecting the activity being conducted along with the intensity of said activity from accelerometer data coming from a subject's waist, at only 72% [95]. This made them abandon it as an assessed machine learning algorithm in the alternatives where the activities and their intensities were classified separately. This was surprising to us, given the other works that successfully used support vector machines, and were often even able to achieve the most superior results with the algorithm.

Park et al attempted to do pedestrian dead reckoning (PDR) – which is a type of pedestrian navigation system (PNS) that uses inertial sensors, such as the accelerometer and gyroscope, to estimate a pedestrian's speed and direction – using support vector machines. In their experiments, they attempted different placements for the phone and tested for the relative accuracy of the other placement. When the SVM was trained with accelerometer and altitude data from a handheld position, they were able to achieve 98.13% accuracy for prediction from more test points from a handheld position. When they attempted to perform the prediction on data from the back trouser pocket or front trouser pocket, they received 0% accuracy for both. When the SVM was trained with data from the back trouser pocket, it provided 94.83% accuracy for the test points from the back trouser pocket, but only 0.99% accuracy for the front trouser pocket and 0.32% for the handheld position. Finally, when data from the front trouser pocket was used to train the SVM, it was accurate to 99.01% with test points from the front trouser pocket, but only accurate to 5.17% and 1.56% for the back trouser pocket and the handheld position, respectively [90].

This provides a few interesting points of insight. First, SVMs seem to be excellent at detecting steps from accelerometer and altitude data, reaching an accuracy up to 99.01%. However, it also shows that SVMs will be very sensitive to the training dataset provided, showing a significant degradation in accuracy when the test points came from a different position than the training points. Finally, it also shows that if one were to want to train an SVM for step detection, the best placement would be the front trouser pocket as, not only did it provide for the best accuracy across the three positions, but it also provided for the best – albeit still quite low – accuracies when the test points came from different positions.

In the fall detection experiment conducted by Edeib et al, they were able to reach an overall accuracy of 95% with support vector machines. This was a middle performance, relative to naïve Bayes' 91% and decision trees' 97% [91]. They do however state that the results of the SVM were very close to the best accuracy achieved, and it would have sufficed as the algorithm of choice.

4.3.2.6 K-Nearest Neighbor

The k-nearest neighbor (KNN) algorithm works by calculating the distance between a sample point and the different points in the graph. A constant K determines the number of neighbors to be considered who are closest to the sample point. Based on these nearest neighbors, a classification can be made [49].

In Aziz et al's fall detection experiment, k-nearest neighbor scored the highest score on specificity, 96%. While that is the case, it also had the lowest sensitivity of all evaluated algorithms [50]. This

indicates a high potential for false negatives from the k-nearest neighbor algorithm. Palmerini et al's fall detection experiments confirmed these results, with k-nearest neighbor providing the least accuracy of the assessed machine learning models at only 95.8% [94]. Albert et al found KNN to be 97.9% accurate at detecting falls, putting it squarely in the middle, relative to the other evaluated models.

Al Nahian et al's fall detection experiments also utilized k-nearest neighbor as one of the assessed machine learning algorithms. It shared the position of least accurate at only 97% when trained with accelerometer data from the UR Fall dataset. With the UP Fall dataset, it provided a middling performance, sharing the position of second most accurate at 98%. Like the other assessed algorithms, it was 99% accurate with the MobiFall dataset [100]. This meant that it had an average accuracy of 98% across the datasets, making it share the position of third most accurate (out of four) with support vector machines.

In Althobaiti et al's human activity recognition experiments, a unique approach was taken. The same set of experiments were repeated for multiple K constants. These provided some very interesting results. First, for the classification of whether or not the activity itself was a fall, trying with 1 nearest neighbor proved just as accurate as trying with 3 nearest neighbors at 98.1% accuracy [96]. Changing K to 5 nearest neighbors proved to improve the accuracy, reaching 98.29% at detecting whether or not the activity was a fall, equating it with the Linear Discriminant Analysis and shy only 0.19% of the best accuracy achieved by support vector machines. Increasing the neighbors, however, doesn't always guarantee an improvement of classification performance, as increasing the neighbors to 7 dropped the accuracy of detecting a fall to 97.71% [96], putting it very close to the bottom.

These results continued to be consistent at detecting the activity in the same research, not just whether it was a fall or not. Using 1 nearest neighbor provided the lowest accuracy of 90.10% [96]. Interestingly, the same lowest accuracy was achieved by using 7 nearest neighbors. Using 3 nearest neighbors provided the highest accuracy of the KNN substitutes at 91.05%, but still far from the highest overall accuracy achieved by the Linear Discriminant Analysis at 93.71%. Using 5 nearest neighbors only dropped the accuracy to 90.67% [96].

KNN continued to prove its accuracy in Wu's research, being 90.2% accurate for the weighted average of the classified activities, the highest of all evaluated learning algorithms [51]. This lends merit to KNN as a potential classifying algorithm for accelerometer data.

Fang et al's experiments also vouch for the robustness of k-nearest neighbor as a machine learning algorithm for predicting human activity from accelerometer data, showing it as the most accurate of the algorithms tested, with an average accuracy of 95.3% across the different activities, relative to SVM's 88.9% and logistic regression's 83.9% [84].

Braganca et al, in their proposed Human Activity Recognition system, called HAR-SR [86], use the k-nearest neighbors algorithm as one of the steps through which they identify human activity. In order to test their system, they conducted multiple different experiments with different combinations of sensors, different position placements, and different datasets. The most relevant

one to this work is the position-independent all-sensor result, as our users will not be placing their phones in a fixed position on their bodies, and also as we will be using the all-sensor dataset as the baseline experiment for our analysis. While we will certainly be experimenting with different combinations of sensors, the most related result of their work here is the aforementioned one.

Overall, their proposed algorithm, of which KNN is a part, was able to reach an overall accuracy of 77.82%. This was slightly skewed by their proposed system's inability to distinguish among stationary activities [86]. The system performed significantly better as classifying between stationary and dynamic activities and among dynamic activities themselves.

Shakya et al's final tested machine learning algorithm was the k-nearest neighbor. This provided the best accuracy across the machine learning algorithms for both datasets with 97.6% and 90.16% respectively [88]. With an overall accuracy of 93.88%, the k-nearest neighbor algorithm far outperforms decision trees and random forests for these experiments, as far as the assessed machine learning algorithms go. One of the more interesting aspects of this set of experiments, however, is that while working with the two datasets, dataset number one seemed to be better classified using traditional machine learning algorithms, while dataset number two was better classified using deep learning algorithms. This gives the rather important insight of needing to match the dataset to the machine learning mechanism to use, providing a refutation to the notion that deep learning mechanisms would necessarily perform better at classification than traditional machine learning approaches, regardless of the dataset provided.

Ferrari et al's similar experiment also compared the k-nearest neighbor algorithm. They found that when using the accelerometer-only readings from the UCI-HAR, MobiAct, and MotionSense datasets, k-nearest neighbor was capable of achieving accuracies of 73.71%, 87.69%, and 79.19% respectively [99] averaging 80.19% accuracy. When the gyroscope-only readings were used, the accuracies were 70.74%, 78.54%, and 85.16% for each of the datasets respectively [99] with an average of 78.14%, the lowest of the averages. When the readings from both accelerometer and the gyroscope were considered, the resulting accuracies for each of the datasets were 82.36%, 86.25%, and 74.08% [99] with an overall average of 80.89%, the highest of the averages. Across the different combinations of sensors and the different datasets, the k-nearest neighbor algorithm provided an accuracy of 79.74%, lower than both support vector machines and the ResNet tested, even while performing higher than both individually in some cases. Unlike the support vector machines, however, there was no consistent pattern with k-nearest neighbor performing best or worst with one specific dataset.

In Gomes et al's experiments classifying activity intensity from accelerometer data streamed from a subject's waist, they also found that k-nearest neighbor provided the best accuracy for detecting both the activity and its intensity simultaneously at 79% [95]. When classified separately, KNN continued to be the best at recognizing the intensity of the activity at 80%, while performing worst at recognizing the actual activity being conducted at 96% [95]. Again, however, it is important to note that these positions of best and worst respectively are narrowly achieved, with only a single percentage point's difference for each of the positions from the random forests algorithm.

Yang et al's broiler behavior recognition experiment also used k-nearest neighbor as one of the assessed algorithms. Again, they repeated their experiments with different sampling rates, but the average accuracies were 96%, 97%, 83%, and 86% [89] for resting, walking, feeding, and drinking respectively. This provides an overall accuracy across the timespans and activities of 90.5%, significantly lower than the 94.25% overall average accuracy obtained by support vector machines.

4.4 User Identification Applications of Machine Learning on Zero-Permission Sensors

In this section, we provide an investigation of the works that performed machine learning on zero-permission sensor data for applications that are like ours in nature. Namely, we focus on the applications of user profiling, and identification / authentication. Seeing as the works here are fewer than those addressing machine learning on zero-permission sensors at a general level, we will be splitting this section into subsections focused on the application of the work, rather than the method of machine learning used as previously done, with an additional section on the potential consequences of this kind of work.

4.4.1 User Authentication

Seeing the amounts of research conducted in user authentication really shows how important of a field it is, and how users still perceive it as an obstacle or a hindrance. Simplifications in user authentication have gone from auto-filling passwords in the browser to integrating fingerprint and facial recognition authentication in everyday apps on smartphones, to even password-less logins in recent times, yet still more ease is being sought after. That is why automated user identification and authentication research is an active field with multiple contributions. We will focus on a few of these works pertaining to machine learning on zero-permission sensor data.

Zaharis et al used the accelerometer, albeit in a Wii Remote instead of a smartphone, in a unique way to perform user authentication. They had the user register a 3-dimensional signature and then attempt to recreate it in space. The accelerometer collected data about the position of the device, as well as the rate of motion in creating this signature. They would then match an input pattern to be matched to the registered signature. If a match is verified, then the user is authenticated properly. They did not use machine learning, but rather other factors such as the elapsed time for signature completion, maximum and minimum accelerations, etc. While the security of this approach is questionable, they were able find that users were successfully authenticated 98.2% of the time, with zero false positives in their experiment [55]. These numbers are very impressive; however, it is important to note that the sample size was only four users. Additionally, signing your name in the air with your phone every time you want to access it is arguably even more of a hindrance than currently established authentication methods,

in addition to being socially awkward.

Shi et al's proposed SenGuard authentication and security system takes a different approach altogether. It highlights that authentication today is based on a single-shot approach, rather than a continuous one. Entering your password and leaving your device unlocked as you get a cup of coffee, for example, would leave you vulnerable to attacks. Instead, they suggest that authentication should be done in a continuous form and implement such a system in their proposed SenGuard authentication mechanism. Although it does not exclusively rely on zero-permission sensors for authentication, they are a large and important factor of their inputs. They find that users implicitly have a walking signature, unique to themselves. Using a naïve Bayes classification model, they calculated the confusion matrix in Figure 7 for four users [56]. Those numbers are indeed very impressive and represent a high degree of confidence for user authentication through the zero-permission sensors.

actual \classified	User A	Others
User A	0.971	0.029
Others	0.036	0.964

actual \classified	User B	Others
User B	0.955	0.045
Others	0.018	0.982

actual \classified	User C	Others
User C	0.968	0.032
Others	0.036	0.964

actual \classified	User D	Others
User D	0.958	0.042
Others	0.036	0.964

Figure 7: Confusion Matrix for four users' walking patterns, relying on the accelerometer [56]. It shows that users can be classified appropriately with a high degree of confidence.

Nickel et al also focused on users' walking signatures as a form of authentication. They used the accelerometer in a phone attached to the right side of the hip of each subject. Data was collected from each subject and then fed into Hidden Markov Models. They found that, combining quorum voting which merges multiple classification results to one, they were able to reach a minimum equal error rate of 5.81% [57], an impressive number. It is important to note, however, that this experiment was done in a highly controlled setting, with the walking route being predetermined. It would be interesting to see how these numbers compare to a real-world setting where the monitoring is being done over an unspecified route.

Strada et al attempted a similar experiment, recognizing users by their gait. Unlike Nickel et al, they did not use a smartphone, but rather placed and glued Inertial Measurement Units (IMUs) containing a triaxial accelerometer in the right sole of the shoe that participants used. Their final aim was to develop a product called the Wahu shoe, which would be capable of adapting to the external environmental factors like terrain, temperature, and humidity and to the user's own state and provide services such as foot pressure analysis and fall prevention.

Their experiments, however, focused entirely on identifying the user based solely on their steps. They had 5 male subjects, each 23 years old walk for around 12 to 15 minutes each, collecting around 3700 steps for the participants. They divided these steps into 1,932 and 1,802 for the training and testing datasets respectively. They also used two machine learning algorithms, k-nearest neighbor and linear discriminatory analysis. They were able to correctly correlate the step to the user with an accuracy of 97.8% using the k-nearest neighbor and a whopping 99% using the linear discriminatory analysis [98]. While these results are very impressive on the gait identification front, our own assessment is that their experiments veered from their original objective, not clearly correlating exactly how the identification of the user would assist their proposed shoes in adapting to the user's environment or own state or prevent falls, for example. It can, however, be very compelling research in the area of authenticating a user simply by the way they walk, with the accuracies being very promising. Seeing the success of these experiments at not only identifying users' traits, but going further as to authenticate them, inspires a lot of confidence in our own experiment, attempting to profile users. While there are major differences to account for, such as the technological change between the time these experiments were conducted and now, in addition to conducting the full experiment in a real-world setting, we see these results as positive indications to the success of our own.

4.4.2 User Profiling

We were only able to find one work that had a similar method and application to our own. Gao et al's experiment attempted to predict five personality traits about their subjects: extraversion, agreeableness, conscientiousness, neuroticism, and openness. They did this combining accelerometer data to recognize the physical activity of the users, comparing it to a baseline which relied exclusively on phone usage (phone calls, active applications, SMS messages, etc.). Their data was fed to a support vector regression model, combining the concepts of support vector machines and linear regression, to perform the prediction. They found that combining the accelerometer data to the phone activity data improved the accuracy of the prediction model over the baseline for two of the five personality traits: agreeableness and conscientiousness. They also found that this inclusion performed better for females than it did for males [58]. While these numbers question the validity of using accelerometer data for prediction, it is important to highlight that the aim here was to profile personality, and not demographic factors as our experiment attempts.

4.4.3 Questions of Privacy

Relying on zero-permission sensors to detect data about users without them explicitly allowing it does bring ethical questions of privacy. Narain et al conducted an experiment where they collected data from the accelerometer, gyroscope, and magnetometer. They did

not use machine learning, but rather treated the experiment as a maximum-likelihood problem. Using only this data and street information from Open Street Map, they were able to show that the actual route taken by a user shows up on a calculated shortlist of ten potential routes over 50% of the time in eleven cities where they conducted simulations. In two cities where they conducted real driving experiments, they found that the correct route is in the top 5 predicted routes 50% and 30% of the time, respectively [38]. This shows a significant capability of user tracking through sensors that requires zero permissions whatsoever. We believe this is what may have prompted them to comment on the "perils" of zero-permission sensor data collection later [59]. A very similar experiment conducted by Liang et al attempted to not only infer the route a user was taking, but also the actual position of the user. Again, relying only on zero-permission sensors, they found that the inferred position of the user was accurate over 65% of the time. Over 86% of the inferences were correct to just 500 meters. For the full dataset, all of their estimations were correct to 2.0 kilometers [34]. This shows that not only could user routes be tracked using sensors that do not require any permissions, but also that the actual position of the user could be tracked using the same sensors, with a very good degree of accuracy. Narain and Liang et al make it a point in their respective papers to warn against the potential location privacy leakages that could occur without users ever knowing about it.

Chapter 5

Experimental Setup

In this section, we discuss the experimental design decisions made. After data collection and preparation, three sets of experiments will be conducted. The focus of the first set is to establish a baseline for the capability of user profiling. The second set attempts to optimize the process by determining the minimum timespan needed for accurate profiling, while the third set attempts to refine the process by determining the most accurate combination of sensors. Concurrent to those sets of experiments is the repetition of each of them on the different machine learning algorithms to compare their respective results.

5.1 Data Collection and Preparation

One of the major challenges of conducting this research is the lack of an available dataset. This meant that we had to collect our own data before starting any experimentation.

5.1.1 Application Design

During the first attempt for data collection, an Android application was developed and provided to a select group of volunteers. The application source code is available in a public GitHub repository linked in the appendices of this thesis. The application was developed with Java as the programming language of choice (with Kotlin as an alternative we did not select), with a target SDK version of 29 (Android 10). It used the Jetpack suite of libraries to reduce boilerplate code, support the design of the user interface, and support backwards compatibility to a minimum SDK version of 23 (Android Marshmallow). It also used Google Material UI Library to provide ready-to-use user interface elements consistent with the design language of Android applications at the time of the application development.
The application largely consisted of seven major components described below.

5.1.1.1 DatabaseHelper

The DatabaseHelper class extends Android's SQLiteOpenHelper. It sets a few constants such as the filename for the SQLite database that will be used to collect the data from the sensors as well as the user data, the version of the database to allow for upgrading or changing the contained information during testing periods, the structure of the database with the associated tables and columns, and the logic for the initial creation and upgrading of the database. We

chose that the initial creation should simply create the tables, while the upgrade would also drop the existing tables prior to the creation, implying that every version change would result in a complete rewrite of the database.
The structure of the database itself is quite simple. It is composed of two tables: UserData – which acts as a very simple key-value store for keys we will discuss in later components – and SensorData, composed of four columns: an ID to reference the recording, an ID to reference the sensor from which the recording was being made (with 1 being the accelerometer, 2 being the gyroscope, and 3 being the ambient light sensor), the value of the sensor reading (concatenated as comma-separated values in the instances where triaxial readings are provided), and the final column recording the epoch time at which this sensor recording was made.

5.1.1.2 MainActivity

The Main Activity acts as the hub of logic for the application itself, performing a few functionalities. The first thing it does is interact with the DatabaseHelper class to obtain an existing instance of the database (which would create one if one did not exist). Following that, it would attempt to query the UserData table for the existence of the following keys: name, consentDate, signature, gender, and ageGroup. If any of these keys were not found in the userData table, MainActivity would close and return the user to RegisterActivity. If all of the keys were indeed found, the application logic would continue.

Followed by checking for the existence of the UserData, MainActivity would check if permissions were granted to the application. In order to ensure successful operation of the application, it required two permissions primarily: the ability to detect Android's boot-successful event to be able to run as soon as the phone was started, and the ability to constantly run as a foreground service to collect data at all times without being stopped for lack of user interaction with the application. If the permissions were not granted, the user would be prompted to grant those permissions to the application. If they were, the application logic would continue.

Given that these two permissions were indeed granted, the application would show an indeterminate progress dialog to the user, and begin executing in a separate thread the following logic. It would query the application's SharedPreferences (an application-level key-value store provided by the Android API) for the recording time of the last reading of each of the sensors and store them in a hashmap. It would then check if the application was being optimized by Android's battery optimization features. This would affect the quality of our data and the sampling rate at which we could constantly operate. Therefore, if Android's battery optimization features were active for the application, the user would be taken to the appropriate phone settings screen where they can turn off the battery optimization features

for this data collection application. Following that, the application would check whether the SensorService, described below, was already started. If not, it would start the service. It would then query the service for the sensors that it was actively recording and display in the application a green bar for them. A red bar would be displayed by default, if the sensor was not one of the ones included in the service.

While that thread is executing in the background, the foreground thread is also preparing the interactivity of the user interface elements. In this case, it is an extremely simple one: just one button that enables the user to share the database. When clicked, the application first checks that it has the ability to read and write to the device's external storage. If not, the user would be prompted to grant those permissions. Following that, the application creates a separate thread that gets access to the SQLite database file, copies it to somewhere accessible in the user's device, and adds a timestamp to the filename. Of course, important error handling is also done in this portion, accounting for storage limitations or file duplication, etc. After the file was copied correctly, Android's sharing API is called upon to allow for the file to not only be accessible in the storage, but also shared immediately to any supporting application or interface such as WiFi-Direct or Bluetooth. This was primarily used to share the database directly to Google Drive where it could be stored in the cloud for access by the researchers.

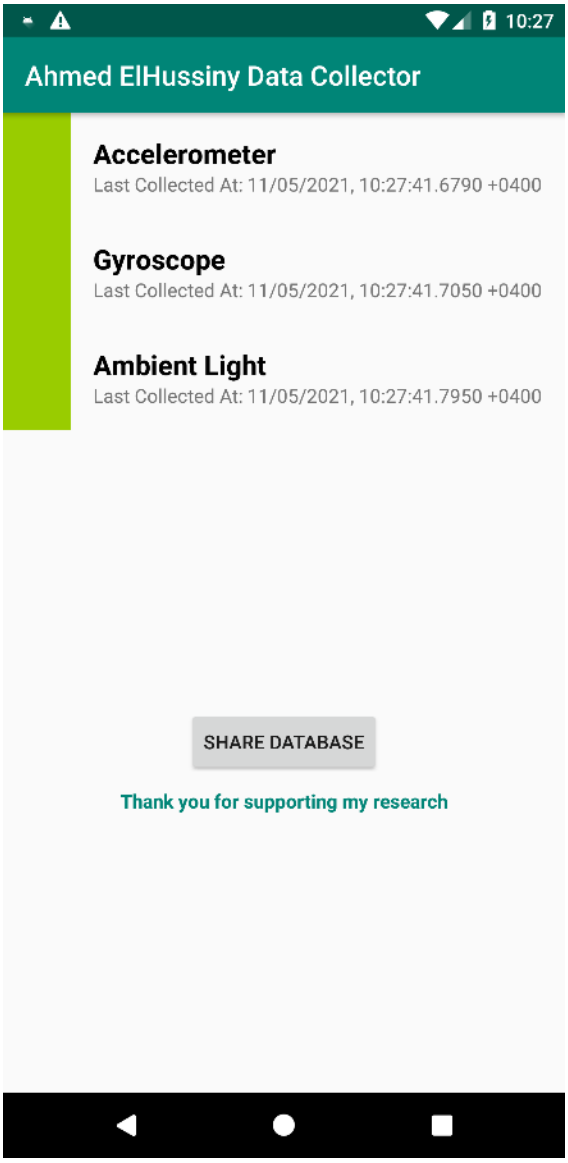

Figure 8: MainActivity: Default application screen showing option to share the data collected and displaying the latest collection time for each sensor, as well as the green indicator bars for the sensors indicating that their readings are being actively recorded

5.1.1.3 SensorService

Easily the heart and soul of the application, the SensorService is an Android service (application logic without a user interface) responsible for collecting the sensor data. First it obtains access to the application database, then works with Android's sensor manager to register a new listener for the accelerometer, the gyroscope, and the ambient light sensors. It also initiates an Ongoing (or "sticky") notification serving two important functions: the first is alerting the user to the fact that their data is actively being collected, allowing the user to monitor if the application were to fail for any reason and prompting the user to restart it. Second of all, the ongoing notification allows the service to continue running in the

background, without interrupting its workflow due to the user's lack of interaction with the application itself. This is a very common use case, employed most typically by applications that require the ability to continue running in the background, such as playing music or VPN services.

The listener to the accelerometer, gyroscope, and ambient light sensors is registered at SENSOR_DELAY_FATEST, further discussed in the Data Density section below. The function of it is to listen to changes in any of these sensors and obtain their values. If the sensor values are multiple ones (such as those coming for each of the axes in the cases of the triaxial accelerometer and the triaxial gyroscope used), the values are concatenated as a comma-separated string. The values, along with the sensor responsible for them, and the current time are stored in the database's SensorData table. This collection time is also used to update the SharedPreferences, recording the latest time for which a sensor reading was done for each of the sensors. This was done so that it can be used later by MainActivity for display purposes without having to query the extremely large database for the latest recording time for each of the sensors, seriously degrading application performance.

As a safeguard mechanism, if the SensorService were to be destroyed by the Android operating system for device resource optimization, despite it being placed in a thread set for MAX_PRIORITY, then it would initiate a restartService broadcast through the operating system. This broadcast would be listened to by the Restarter components described below.

Figure 9: Ongoing notification informating users of the data collection process, and allowing the service to continue running in the background

5.1.1.4 Restarter

The Restarter component is a very simple extension of the Android BroadcastReceiver. It listens for a broadcast coming from the application containing the message restartService. It then checks if the service is indeed already started and, if it is, ignores the received broadcast. If, however, the service had been terminated for Android's resource optimization, the Restarter starts the SensorService component, guaranteeing the continuation of the data collection process.

5.1.1.5 BootReceiver

Registered in the application's manifest, the BootReceiver component is another simple Android BroadcastReceiver. Its function is to listen to the Android boot-completed broadcast, indicating that the device has been turned on, and start the SensorService component. This guarantees that the data collection process would continue after the device had been turned off, either deliberately by the user, or due to power failure or device malfunction.

5.1.1.6 RegisterActivity

If the user data were not found in the MainActivity, the user would be taken back to the RegisterActivity, the logic for which is quite simple. It would start by checking the application's SharedPreferences file for the user's consent to the Institutional Review Board (IRB) agreement. If it were not found, users would be taken back to the consent activity. Otherwise, they would be shown a user interface allowing them to select their biological gender and age group. The only way for users to proceed is through clicking the floating action button in the bottom right corner. What that would do is first validate that users had indeed made an appropriate selection for both questions asked on the form and show them an error message if they had not. Otherwise, it would get the consent data from the SharedPreferences – which it had queried for at the beginning of its lifecycle –, combine it with the user's selected demographic data, and add this data as key-value information to the UserData table within the database before proceeding the user to MainActivity.

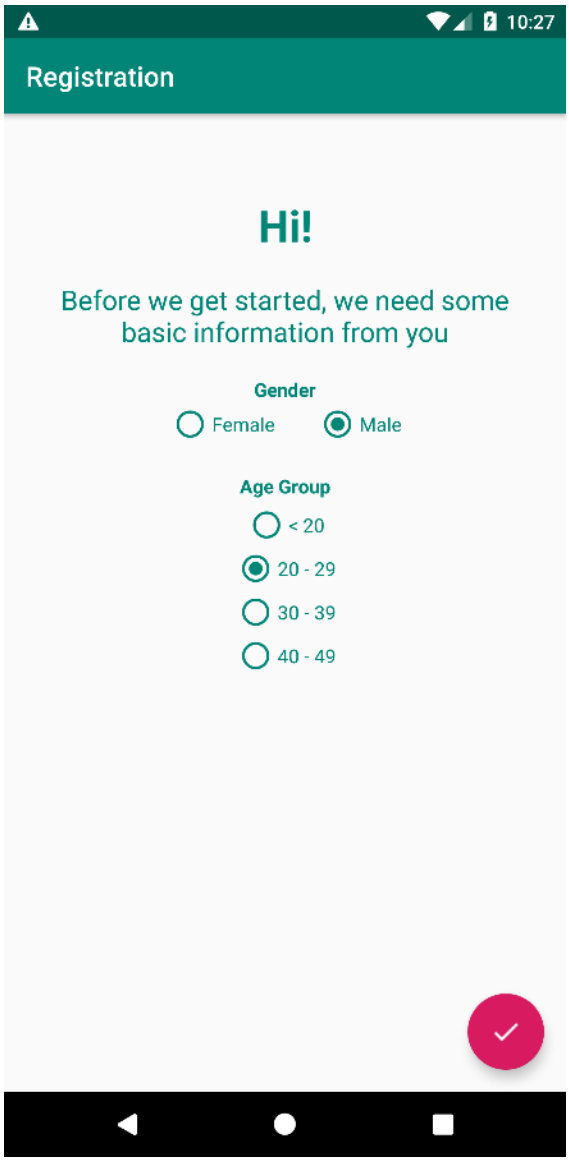

Figure 10: RegisterActivity: Step 2 of the data collection application - User Registration allowing users to input their biological gender and age group

5.1.1.7 ConsentActivity

If the Institutional Review Board (IRB) agreement consent is not found in the SharedPreferences at launch of the RegisterActivity, the user is taken to ConsentActivity. This has a very simple user interface, showing the user the terms of the IRB agreement, and some basic information about the research being conducted. The only way for a user to proceed is through selecting the "Accept" button at the bottom the screen. At which point, the user would be presented with an acceptance dialog, allowing them to enter their full name as well as provide a signature, consenting to the data collection by the application. In the acceptance dialog, users' only path forward is to click the "accept" button within it. Doing that prompts

the application to validate that the user has indeed written their full name (confirmed by at least being 5 characters long and containing of a space), and that the user had provided their signature. If not, users would be prompted to do so. After completing the acceptance form correctly, the accept button stores a number of key-value pairs within the application's SharedPreferences. First it stores the fact that the user has indeed pressed the accept button, indicating their acceptance to the data collection, it also stores the user's full name, the current date of the device, and a base64-encoded bitmap of the user's provided signature. It would then take the user to the RegisterActivity.

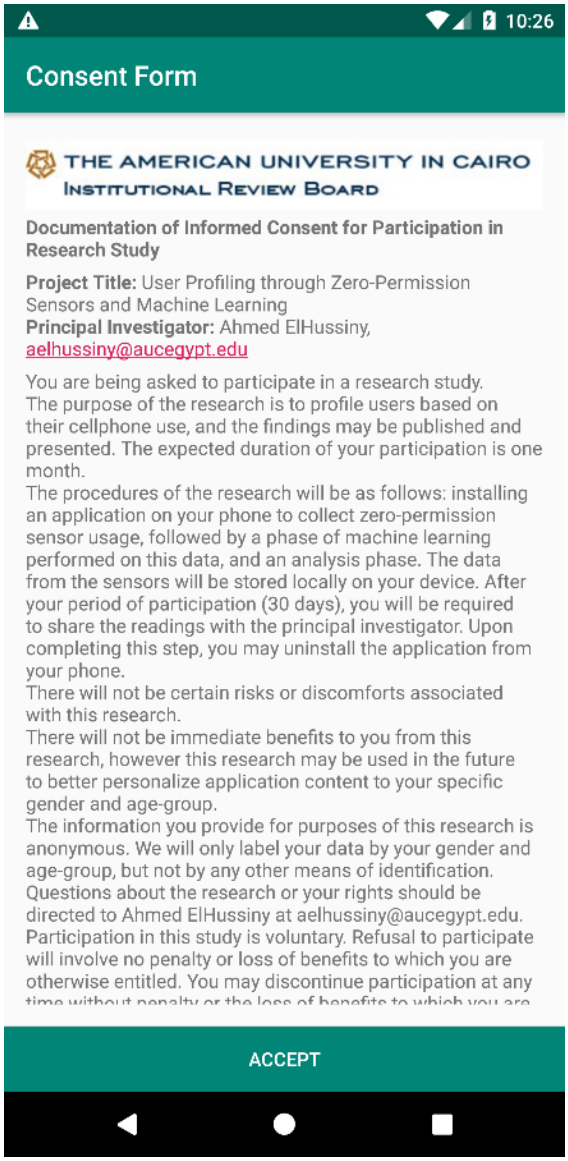

Figure 11: Step 1 of the data collection application - Institutional Review Board Agreement

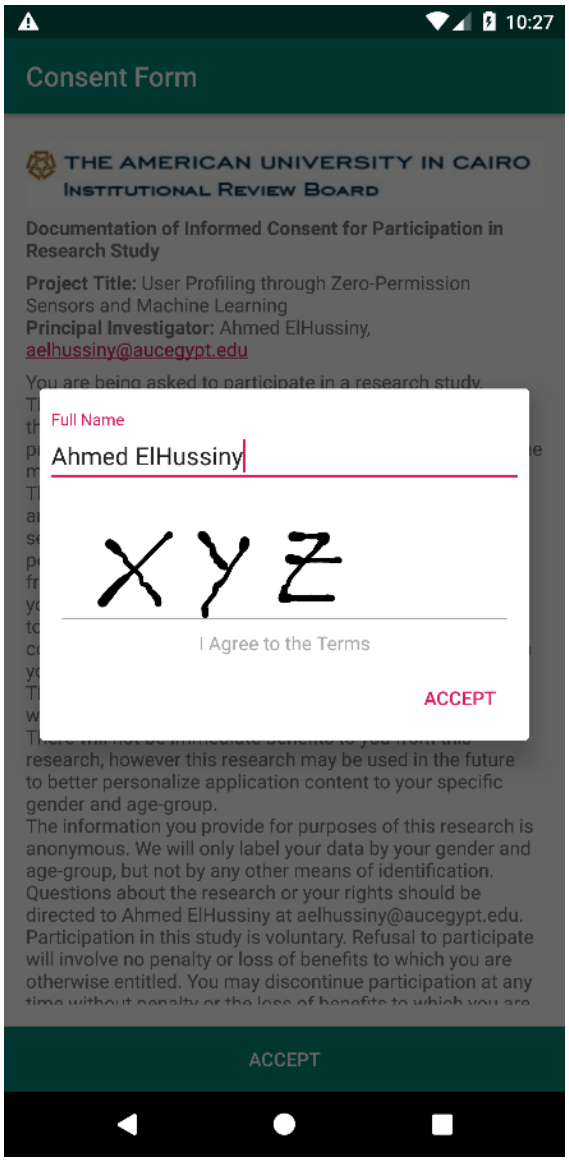

Figure 12: Step 1 of the data collection application - Institutional Review Board Consent

5.1.2 First Attempt

The volunteers were to install the application to their own personal phones. The application itself consisted of two simple steps: agreeing to the Institutional Review Board (IRB) agreement, followed by setting their demographic data – to act as the data label later used for machine learning. After these two steps were completed, the application would start listening to data incoming from three sensors – the accelerometer, gyroscope, and ambient light sensors. Users were made aware of this through an Ongoing notification. A screen letting the users know the latest time of collection for each sensor and providing the option to share the data at the end of the research was also provided.

5.1.3 Data Density

When registering a listener to a sensor in the Android operating system, developers have to select a sampling rate that sensor events are delivered at. Some of the available constants are: SENSOR_DELAY_NORMAL, SENSOR_DELAY_UI, SENSOR_DELAY_GAME, and SENSOR_DELAY_FASTEST. The sampling rate each of these delay constants determines is deliberately not shared. The reason is that the sampling rate chosen by the developer is only a "suggestion" to the Android operating system. While Android will try its best to follow the suggestion, if the system is overloaded at any point, other actions may be prioritized over collecting the sensor event [60]. However, these values have been practically measured to be between 215 and 230 milliseconds for SENSOR_DELAY_NORMAL, 75 to 90 ms for SENSOR_DELAY_UI, 35 to 40 ms for SENSOR_DELAY_GAME, and 15 to 20 ms for SENSOR_DELAY_FASTEST. Our application uses SENSOR_DELAY_FASTEST, meaning that we can expect a sensor event from each of our sensors every 15 to 20 milliseconds. This places our suggested sampling rate between 50 and 66Hz, with an expected average of around 58Hz. In addition, and in order to make sure that other less important applications are not prioritized over the data collection, we have set the thread listening to the sensors at MAX_PRIORITY indicating to the Android operating system that this thread should only be ignored in extreme cases.
In order to prevent the data from getting too large (theoretically around 13 million data points a day), we implemented a policy of not storing repeating values. This would ensure that a phone left stationary without any changes to its position, orientation, or lighting conditions (such as when on a nightstand when sleeping) would not be constantly recording data. It's important to note here that, given the sensitivity of these sensors, they are likely to still collect some data points due to some minute movements or lighting changes during the night, but it should theoretically be significantly less than in periods of activity.

5.1.4 The Challenge

Even with the aforementioned optimization of not recording repeating values, the application was generating between 0.5 and 1.0 gigabytes of data a day. With users often dealing with limited storage capacity to begin with, this presented a challenge. Additionally, the application prioritized itself over all other applications, meaning that users found their phones behaving much slower. Also, due to the constantly running nature of the application, users' phones were much warmer (which was less comfortable) and more likely to deplete their batteries significantly faster. Not only was that an inconvenience to the users, but also it made for inconsistent and constantly interrupted data. Unfortunately, the data from the first attempt was scrapped as we prepared for our new approach.

5.1.5 Second Attempt

Given the challenges encountered by the first iteration, we determined that most of these problems arose from users relying on lower-performance phones. Users with high-performance or "flagship" devices reported significantly fewer issues and had significantly better data consistency. This is why, for the second attempt, our approach was different. Instead of providing users with the application to install on their personal phones, we would provide them with the phones directly. We obtained two devices: a Samsung Galaxy Note 10+ and a Samsung Galaxy S20+. The Note 10+ was designated as the device to collect the data for males, while the S20+ was designated for females. Users would receive said device and agree to use it as their primary device until they pass it on to the next volunteer.

5.1.6 Selection Criteria

In order to remove age as a potential factor in how phones are interacted with, we limited our users to the age group of 20 to 30 years old. All volunteers would also be right handed, from similar social backgrounds, between the heights of 4'11" and 5'10", and physically fit and healthy. Each phone would spend 3 days with each volunteer, and then be passed on to the next volunteer. Six male volunteers and five female volunteers were identified to participate in the research. This would provide a grand total of 18 days of male datapoints and 15 days of female datapoints.

5.1.7 Caveats

It is important to note that even with this approach, some caveats still exist. First of all, even with the high-performance phones, the "always-on" and high frequency recording approach of the application means that it will get warmer and lose battery faster than a phone in normal operation. This means that it might be slightly less comfortable to users than their regular phones, leading to potentially differing usage, and will require more charging time, again potentially differing from their "typical" usage. In order to reduce the impact of charge-time, volunteers were also provided with a 20,000 milliampere hour power bank with each phone, enabling them to charge while "in-use." However, the added bulk of the power banks may also contribute to differing usage. An important consolation here is that the aim of the research is not to determine the smartphone usage of males and females under typical circumstances, but rather to determine if it is possible to predict the gender of the users, given the same circumstances. Since both sets of volunteers were dealing with the same warmth and charge-time issues, they could be considered constants in our experiments, allowing us to proceed with the research.

5.1.8 Data Cleanup and Augmentation

Prior to starting experimentation, we planned on cleaning up the data and preparing it to be consumed by our machine learning models. For that, we are writing Python scripts to massage the data to better fit the models. The aim of the Python scripts is to use the multiple databases collected from the users and create multiple datasets out of it that match the data that would be needed for the experiment. These scripts are described in section 6.1 below.
For the first set of experiments, there isn't much done to the data. The script just checks for regularity in the data and ensures there is no corruption. The data is then combined across the different databases, labeled with its respective label, and divided into 2 portions: a portion representing 70% of the data to be used as the training set, and a portion representing 30% of the data to be used as the validation set.
For the second set of experiments, the script will actually rely on the dataset created for the first experiment. Seeing as this set of experiments is focusing on the effect of the timespan in gender determination accuracy, the script's job will be more involved. Using the dataset created by combining the different database, and split into training and validation datasets, the script will parse each dataset and create out of it different "slices" representing 1 day, 1 hour, and 5 minutes.
For the third set of experiments, the script will also rely on the dataset created for the first experiment. With the focus this time being on the effect of the specific sensors on gender determination accuracy, the script will slice the datasets based on sensor type, for all combinations mentioned further.

5.2 Summary of Experiments

In this section, we focus on providing a summary of the experiments and the dependent and independent variables and how they will change in the below table. Each set of experiments is then described in detail in a following subsection.

Table 1: Table describing all the experiments that will be conducted as part of this thesis

	Gender-Determination Capability	Effect of Timespan on Gender-Determination	Effect of Sensor Selection on Gender-Determination
Described In	Section 5.3	Section 5.4	Section 5.5
Machine Learning Algorithms	Naïve Bayes, Support Vector Machine, Logistic Regression	Naïve Bayes, Support Vector Machine, Logistic Regression	Naïve Bayes, Support Vector Machine, Logistic Regression
Timespan	Full Timespan (3 days per user)	Three trials will be conducted: • 1 day per user • 1 hour per user • 5 minutes per user	Full Timespan (3 days per user)
Sensors	Accelerometer, Gyroscope, Ambient Light Sensor	Accelerometer, Gyroscope, Ambient Light Sensor	Six trials will be conducted: • Accelerometer only • Gyroscope only • Ambient Light Sensor only • Accelerometer and Gyroscope • Accelerometer and Ambient Light Sensor • Gyroscope and Ambient Light Sensor

5.3 The Capability of Predicting Gender using Machine-Learning on data from Zero-Permission Sensors

In this experiment, the aim is to measure the effect of zero-permission sensor data on user profiling capability. We hypothesize that zero-permission sensor data can, with a reasonable degree of accuracy, predict the biological gender of a user. In this experiment, the full dataset (three days of collected data points per participant from the three sensors) will be provided to three different machine learning algorithms: Naïve Bayes, Support Vector Machines, and Logistic Regression. These algorithms were selected as they displayed a higher degree of accuracy than other options in our research. We will be conducting a comparative analysis of the results from the different machine learning algorithm. From that, we will be able to determine the most accurate individual classifier for the use case of user profiling through zero-permission sensors, whether combining individual classifiers could provide a more accurate classification, and the degree to which we can accurately profile users based on data collected from zero-permission sensors. We will use the results of this experiment as the baseline to compare the other attempts to. Future references to "baseline results" are to the results of this experiment.

5.4 The Effect of Timespan on Predicting Gender through Machine-Learning on Data from Zero-Permission Sensors

While the first set of experiments determines the accuracy to which different machine learning algorithms can profile users, it does so while utilizing the full volume of the data, spanning three days from each participant. This is far from optimal as it requires a large amount of collected data over a long period of time. Our experiments indicate that every day for every participant is between 500 megabytes and 1.5 gigabytes of raw data. Even after data preparation and feature extraction, the amount of data that must go through the machine learning algorithms is huge, and classification will be very computationally expensive and time intensive. Therefore, it becomes an important endeavor to experiment with smaller timespans to obtain a degree of classification accuracy like that established by the baseline in the first experiment.

This experiment tests the effect of data timespan on classification accuracy. We hypothesize that classification accuracy will be negatively impacted by reducing the timespan of the data. In this experiment, the independent variable of timespan will change from the control condition of 3 days per participant to three other conditions: a 1-day slice, a 1-hour slice, and a 5-minute slice. The same procedure as the baseline experiment will be used: zero-permission sensor data from all three sensors (accelerometer, gyroscope, and ambient light) for the given time slice will be provided to the machine learning algorithms and left to train. Following the

training phase, zero-permission sensor data of the same timespan size will be provided to the trained models and asked to predict the gender. The results of each of the models will be calculated and compared to the baseline results.

5.5 The Effect of Sensor Selection on Predicting Gender through Machine-Learning on Data from Zero-Permission Sensors

The first and second sets of experiments both utilize data from all 3 sensors for which this research is being conducted: the accelerometer, gyroscope, and ambient light sensors. This does not help us determine which sensor has the greatest contribution to the profiling result. It is also possible that one sensor is introducing noise or similar data across the different profiles, resulting in an improved classification accuracy upon its exclusion. Therefore, we will repeat the first set of experiments again, but instead of using the data from all three sensors, we will conduct different iterations of the experiment with all the possible combinations of sensors: accelerometer only, gyroscope only, light only, accelerometer and gyroscope, accelerometer and light, gyroscope and light. The accuracy obtained from each combination will be compared to the one obtained in the first set of experiments. From that, we will be able to determine if the data collected could be further refined to focus on a specific combination of sensors to achieve the best results. This experiment tests the contribution of each individual combination of sensors on the classification accuracy. We hypothesize that the combination of accelerometer and gyroscope (excluding the ambient light sensor) will result in the best classification accuracy. The independent variable is the set of sensors used, and its conditions are the aforementioned combinations, compared to the control of the baseline experiment. The same machine learning and accuracy calculation procedures as the baseline experiment will be used.

Chapter 6

Experiments and Results

In this section, we discuss the implementation portion of the experiments, challenges faced, and the actual preparation process that the data went through, in order to be ready to be processed by the models that we had set up. We pose the specific questions and exactly how we will answer those questions. Followed by that, we present the obtained results and analyze them, providing the answers to the questions we posed and assessing the accuracy of our experiments and hypothesis.

6.1 Methodology

In this subsection, we present the specific questions we are trying to answer, the experiments we will conduct to answer these questions, and the process the data had to be put through in order to allow for these questions to be answered.

6.1.1 Data Processing Pipeline

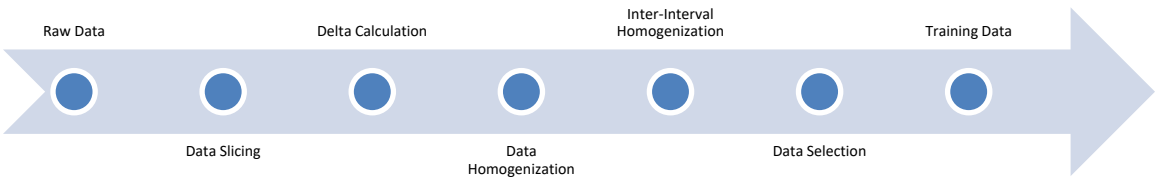

Figure 13: Data Processing Pipeline Steps

Prior to the experimentation phase, we needed to put the data through some very extensive processing to convert the raw data into training data ready for consumption by the various machine learning models we intended on running. This is due to the data consisting of 22.9 Gigabytes – nearly half a trillion records –, divided as 8.97 Gigabytes (186.24 billion records) for the female class and 13.9 Gigabytes (286.1 billion records) for the male class.
Chen et al. found in their experiments that using more sensors provided richer data [83], and

therefore we were interested in testing this theory by maintaining the data from all the sensors in our data processing pipeline and then using the experiments themselves as the testbed through which the filtration of the different sensors would take place. Furthermore, they highlight the importance of resampling the data to guarantee a balanced dataset across the different activities [83]. Since our research does not focus on activity classification, the rebalancing needed to happen across two dimensions: the gender, and the actual sensor data collected. However, specific logic needed to be applied for this resampling, which is further explained below.

This pipeline was built incrementally and was comprised of five major steps. The first step was the data slicing step. The target of this step is to break the full, raw datasets, for both the male and female classes, into smaller datasets sliced by the amount of time that the experiment will be run for, or what we refer to as the timespan, time slice or interval. This was a first step into chunking the data, and also making it significantly more manageable to deal with from a volume perspective.

The second step was delta calculation. In this step, we loop over the individual time slices created by the data slicing step, and instead of maintaining the raw values, we calculate the delta between the current reading and the previous reading from the same sensor. This highlights the changes in sensor readings, amplifying the importance of significant changes over the minimal ones, instead of maintaining the raw values which do not mean much relatively on their own.

The third step was the data homogenization. In this step, it was important to make the readings look similar to one another. Given that some sensors – namely accelerometer and gyroscope – gave three readings (and therefore three delta values in the previous step) and the ambient light sensor gave just one reading, the aim of this step was to make sure that each one of the values processed by the models looked the same. Therefore, we adopted the padding approach with the light sensor, inserting zeroes in the other two vector sections, guaranteeing that each reading would always be a three-digit vector of the changes from the previous reading.

The fourth step was the inter-interval homogenization. This step was run at the beginning of each experiment, instead of one complete separate step of the data processing pipeline. What this step does is guarantee that each of the slices ran through the model as an individual training sample is the same size as the others. To understand the significance of this step, let us first establish that data entered to a model must be of the same size [66][67]. Now let us consider the 1-hour time slice, for example. Some 1-hour intervals will be full of change activity, where the user is very active, moving around, and changing lighting environments. Other 1-hour intervals, where the user is sleeping for example, will have very little change activity. To guarantee that these intervals were the same size as far as the model is concerned, we had to decide between the two most common approaches used in machine learning: cropping and padding [68]. In typical machine learning, the padding approach either resizes

the image through typical image rescaling mechanisms which would "pad" the image with data, or through zero-padding. Zero-padding is a technique that pads the borders of the image with the zero-value (black) to just focus on the image [68]. Since we are working with numerical data, our version of data padding would insert values to the less active intervals, padding them to be the same size as the most active intervals. The challenge in going this route is that the data quality itself would be affected: there would be a lot of decisions concerning where the augmentation data should be inserted within the intervals itself, and exactly what values to add [69]. The cropping mechanism, in typical machine learning, crops the image to be focused on the subject the model is attempting to classify [68]. Since we are working with numerical data, our version of cropping would be more about the filtration of data than it is about clipping parts of an image. With the cropping approach, we would filter down the most significant intervals to be of the same size as the least significant intervals. With that came the challenge of selecting which readings to keep in the significant intervals. In machine learning, the cropping is done to focus on the subject, so we needed to determine how we would define our "subject." For that reason, this step adds a new metric to the dataset, "significance." Significance combines each reading into a single value, summing up the absolute values of the change readings already in the data. We would then only keep the most significant values, that match the size of the smallest interval. Another approach suggested by other researchers for this type of feature extraction is performing a separate machine learning effort on the dataset in order to first extract the features before performing the primary machine learning effort tasked with classifying the biological gender of the user. This approach would have been, in itself, experimental, and was outside the scope of our thesis.

The fifth and final step was the splitting of the data into the training dataset – determined at 70% of the provided training data – and the prediction dataset – the remaining 30% of the training data. This was again conducted as a step within the running of the model itself. The reason we have chosen to include this step as an integrated one, rather than an individual step in the pipeline, is because we wanted to simulate real-world conditions where the data would not be statically split and selected. This introduced more entropy in the selection of the dataset and provided a better reflection of real-world scenarios.

6.1.2 Experimentation

In this subsection, we discuss the experiments conducted and the purpose behind each of them.

6.1.2.1 The Capability of Predicting Biological Gender

The first question our thesis attempts to answer is whether we can use zero-permission sensors to be able to detect the gender of a user, with accuracy over 50%, 50% serving as a

base value of making a "guess" between the two classes of biological genders. To answer this question, we used the full dataset, divided into three-day intervals, relying on the readings from all three of our sensors – the accelerometer, gyroscope, and ambient light sensor. In this set of experiments, we will simply pass the data through the machine learning models and see how they compare to our selected metrics, defined in section 6.1.3.

The first model, referenced in the code as NB-3Day-AllSensors.py focuses on experimenting with Naïve Bayes. It starts with loading all the data for the 3-Day interval and excluding a random set of 4 3-Day groups to act as the validation dataset, leaving 8 3-Day intervals to form the training dataset. After conducting the inter-interval homogenization step of the data processing pipeline as described above, a GaussianNB model is created using the Scikit-learn library. The model is then trained incrementally, using the model's partial_fit function on each of the files in the 8 sample files. Progress is reported to the screen in the form of a progress bar. After completing the training, the model is then tested for its accuracy by having it predict the gender for the four 3-Day groups preselected for validation. The results of those predictions are used to generate a classification report.

The second model, referenced in the code as LG-3Day-AllSensors.py tests the Logistic Regression algorithm. Since the Logistic Regression algorithm may be trained multiple times, reducing for error, our approach was different. We again load all the data for the 3-Day intervals and exclude 4 3-Day intervals entirely to act as our validation dataset. We also select, but not exclude, two 3-Day intervals from the training dataset to act as a testing dataset, showing us the progress of the learning effort. We then perform the inter-interval homogenization step, followed by using the Scikit-Learn library to build a Stochastic Gradient Descent Classifier with the loss defined as being determined through logistic regression. We do set the maximum iterations to 3,000 without the potential for early stopping. The reason we chose to do that is because we were not going to be serving the dataset to the model's fitting function all at once, but rather partially through the model's partial_fit function. The justification is that each of these datasets is very large and therefore could not be processed at once had they been served to the model. We perform a loop over a maximum of 10 epochs. Within each epoch, we first train the model on the 8 available 3-Day intervals. Following that, we use the 2 selected test intervals to make the model predict the associated biological gender and generate a classification report from them. We check the loss of the model and if it had gotten consistently worse for 3 epochs, we stop the training early and take the best trained model. Finally, we perform the predictions on the validation dataset selected at the start and generate a classification report for them.

The third model, referenced in the code as SV-3Day-AllSensors.py tests the Support Vector Machines algorithm. Here, our approach is almost exactly the same as the Logistic Regression approach, with the same set of steps followed: loading the data, excluding the validation set, selecting the test set, and performing the inter-interval homogenization. We also again use Scikit-Learn's Stochastic Gradient Descent classifier, however this time with the loss being

defined as Hinge, creating a Support Vector Machine. We follow the same training, testing, early-stopping, and validation logic used for the Logistic Regression resulting in a classification report.

6.1.2.2 The Effect of Timespan on Predicting Biological Gender

The second question our thesis attempts to answer is if the size of the interval affects the accuracy of predicting the biological gender of a user. To answer this question, we are again using the full dataset, and again using the readings from all three of the sensors, but divided into 1-day, 1-hour, and 5-minute intervals. The results of this set of experiments are then compared relative both to each other and to the 3-day intervals conducted in the baseline experiment.

First, testing with the Naïve Bayes algorithm, the code – referenced as NB-1Day-AllSensors.py – first loads all of the 1-Day intervals, excluding 6 for validation. The same processing, training, and prediction logic as the ones used for the 3-Day intervals are again used here. The same is also done with the 1-Hour intervals in NB-1Hor-AllSensors.py with 160 validation exclusions and with the 5-Minute intervals in NB-5Min-AllSensors.py with 3,400 validation exclusions.

Following that, we tested the Logistic Regression algorithm with LG-1Day-AllSensors.py. Again, the maximum number of epochs was 10, with a patience of 3 epochs getting consistently worse. Given that we had more samples, however, we were able to set both the test and validation dataset sizes to 6 samples each. With the 1-Hour intervals in LG-1Hor-AllSensors.py, each was composed of 160 samples. And for the 5-Minute intervals in LG-5Min-AllSensors.py, each was 1700. The same logic as the one used for the 3-Day intervals was applied for all of them.

Support Vector Machines were similarly tested with SV-1Day-AllSensors.py, SV-1Hor-AllSensors.py, and SV-5Min-AllSensors.py, with the same number of samples used as those for the Logistic Regression tests.

6.1.2.3 The Effect of Sensor Selection on Predicting Biological Gender

The third question asks whether any of the sensors is introducing unnecessary noise that is confusing the machine learning algorithms, particularly if the set of sensors used affect the accuracy of predicting the biological gender of a user. To answer this question, we used the full dataset, divided into three-day intervals, but divided into every possible combination of sensors – accelerometer only, gyroscope only, ambient light only, accelerometer and gyroscope, accelerometer and ambient light, gyroscope and ambient light. We compare the results of those sensor combination both to each other and to the combination of all sensors conducted in the baseline experiment.

To perform our experiments, we used the same models we did for the 3-Day interval tests.

However, we added a step prior to the data homogenization, that filters the data for the sensors applicable to the specific experiment we were conducting. For example, in NB-3Day-Acceler.py, only readings from sensor ID 1 are included. In LG-3Day-AccelLight.py, only readings from sensors IDs 1 and 3 are included.

6.1.2.4 The Effect of Model Selection on Determining Biological Gender

The final question attempts to determine which of the machine learning models we had selected performs best given the task. How does the machine learning model used affect the accuracy of predicting the biological gender of a user? To definitively answer this question, we conducted each of the aforementioned experiments three times, once for each of the machine learning algorithms we are testing in this thesis: Naïve Bayes, Support Vector Machines, and Logistic Regression. We compare these results relative to one another.
We used classical machine learning models, over the more modern deep learning models. Previous research backs up the argument of using these three classical algorithms, relative to other machine learning ones. We are, however, fully cognizant of the fact that this research was conducted in a time where deep learning models weren't as popular as they are now, and that newer technologies can support better research. Even with this recognition, it is important to note that matching the dataset to the appropriate machine learning algorithm is an effort of its own. As aforementioned, Shakya et al found that of the two datasets they tested, one was significantly better classified using the classical machine learning algorithms rather than the more modern deep learning approaches [88]. The scope of this research was always limited to machine learning, not deep learning, attempting to reach strong accuracies with relatively inexpensive and quick solutions using limited resources. This is to simulate the possibility that these computations can be done on a user's mobile device. Additionally, given the datasets we were working with, the size of the data itself was massive being comprised of 22.9 Gigabytes (472.34 billion records), divided into 8.97 Gigabytes (186.24 billion records) for the female class and 13.9 Gigabytes (286.1 billion records) for the male class. Our solution to that was using the partial_fit functionality of machine learning models to incrementally teach the model on new data, without losing the knowledge of the previous training samples. This type of incremental learning was only available for machine learning models, not deep learning ones. Online, or incremental, deep learning is a much larger challenge, with many research articles devoted to furthering it. We decided that it was outside the scope of this thesis to attempt deep learning on this large of a dataset, and contrary to the purpose of simulating limited resources.

6.1.3 Result Recording and Comparison

Each one of the Python scripts developed for the individual experiments conducted concludes its functionality by using Scikit-Learn's Metrics classification_report function, generating

important metrics about the prediction results of the algorithm discussed in detail below. This classification report, along with other information about the model such as the training, test when applicable, and the validation datasets used, are stored in a JSON file that also includes the timestamp at which this model was run at a folder location indicative of the experiment conducted. The path for such a JSON file would be \Results\{Algorithm_Used}\{Interval_Size}\{Sensors_Used}\results_{timestamp}.json.
Later, a script was written that goes through these JSON files and plots a chart of the selected metrics for each of the experiments, for relative comparison. This chart was used internally for our own analysis purposes, discussed in detail in the Results and Analysis subsection below, and would be superfluous to include in this thesis.

6.1.4 Metric Selection

We selected four metrics to consider our results and include in our analysis provided in section 8 below.

1. Accuracy: describing the accuracy of the model. It measures the total number of correct predictions, divided by the total number of predictions [78]. The issue with this metric, is that it focuses on the "correctness" of a model, rather than giving more weight to exactly how the model is making mistakes, be that with a focus on making more false positives or false negatives. It is a good generalization for a model, but definitely leaves room for insight by other metrics.

$$\text{accuracy} = \frac{\#\ \text{correct predictions}}{\#\ \text{total data points}}$$

Figure 14: Equation for Accuracy [78]

2. Macro-Average Precision: describing the average precision across the classes. Precision focuses on the positives, rather than the negatives. It is the division of true positives, over the sum of all positive predictions, true and false [78]. In our case, it measures how many times a female was predicted to be female over all the times a female prediction made, repeats the process for the male class, and calculates the average between them.

$$\text{precision} = \frac{\#\ \text{happy correct answers}}{\#\ \text{total items returned by ranker}}$$

Figure 15: Equation for Precision [78]

3. Macro-Average Recall: describing the average recall across the classes. Recall measures the ratio of true positive predictions to the total actual positives [78]. In our case, it measures how many times a female was predicted to be female over all the times a female was present in the dataset. Again, it repeats the same for the male class,

and then calculates the average.

$$\text{recall} = \frac{\#\ \text{happy correct answers}}{\#\ \text{total relevant items}}$$

Figure 16: Equation for Recall [78]

4. Macro-Average F1-Score: describing the F1-score across the classes. The F1-Score was created as a ratio between precision and recall [78]. It is simply two times the division of multiplying the precision and the recall over summing the precision and the recall. It is used as a balancing metric between precision and recall and providing more insight into why the model is reaching the accuracy that it is reaching.

$$F_1 = 2\frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

Figure 17: Equation for F1-Score [78]

6.2 Results and Analysis

In this section, we furnish the results of our conducted experiments, and provide some insight into these results. We do so first by summarizing the experiment results in a tabular format in Table 2 below, followed by an exploration of each of the main questions our thesis tries to answer and how the results reflect on these questions. Table 2 shows our selected metrics – the accuracy, macro average precision, macro-average recall, and macro-average F1-Score – for each of the 10 experiments conducted. However, since each of those experiments were conducted three times – once with each of the machine learning models we chose to assess – they are also reported three times, once for each. The cells are color-coded to indicate their proximity from 100%, with green indicating 100% and red indicating 0%.

Table 2: Table summarizing all experiment results, with values color-coded based on their respective distance from 0 and 100, with the maximum values formatted

	Interval	3-Day	3-Day	3-Day	3-Day	3-Day
	Sensors	Accelerometer, Gyroscope, & Ambient Light	Accelerometer	Gyroscope	Ambient Light	Accelerometer & Gyroscope
Naïve Bayes	*Accuracy*	50	50	50	50	50
	Precision	75	75	75	75	75
	Recall	50	50	50	50	50
	F1-Score	33.33	33.33	33.33	33.33	33.33
Logistic Regression	*Accuracy*	25	50	50	50	50
	Precision	16.67	75	75	75	75
	Recall	25	50	50	50	50
	F1-Score	20	33.33	33.33	33.33	33.33
Support Vector Machines	*Accuracy*	50	75	75	50	50
	Precision	75	83.33	83.33	75	75
	Recall	50	75	75	50	50
	F1-Score	33.33	73.33	73.33	33.33	33.33
	Interval	**3-Day**	**3-Day**	**1-Day**	**1-Hour**	**5-Minutes**
	Sensors	**Accelerometer & Ambient Light**	**Gyroscope & Ambient Light**	**Accelerometer, Gyroscope, & Ambient Light**	**Accelerometer, Gyroscope, & Ambient Light**	**Accelerometer, Gyroscope, & Ambient Light**
Naïve Bayes	*Accuracy*	25	25	83.33	48.75	52.47
	Precision	62.5	62.5	91.67	62.4	60.35
	Recall	50	50	50	52.65	52.75
	F1-Score	20	20	45.45	38.99	41.98
Logistic Regression	*Accuracy*	25	50	50	75.62	50.41
	Precision	16.67	75	75	75.16	53.41
	Recall	25	50	50	75.69	51.64
	F1-Score	20	33.33	33.33	75.28	43.8
Support Vector Machines	*Accuracy*	50	50	83.33	72.5	54.06
	Precision	75	50	87.5	72.93	58.78
	Recall	50	50	83.33	71.52	55.39
	F1-Score	33.33	50	82.86	71.63	49.84

6.2.1 The Capability of Predicting Biological Gender

In this case, we are trying to answer the question of whether it is possible to accurately predict the biological gender of a user, given the full dataset divided into three-day intervals, with readings coming from the accelerometer, gyroscope, and ambient light sensors. Since we ran this experiment three times, one for each of the machine learning models we are assessing, we have three different result sets, as can be seen in Figure 18 below.

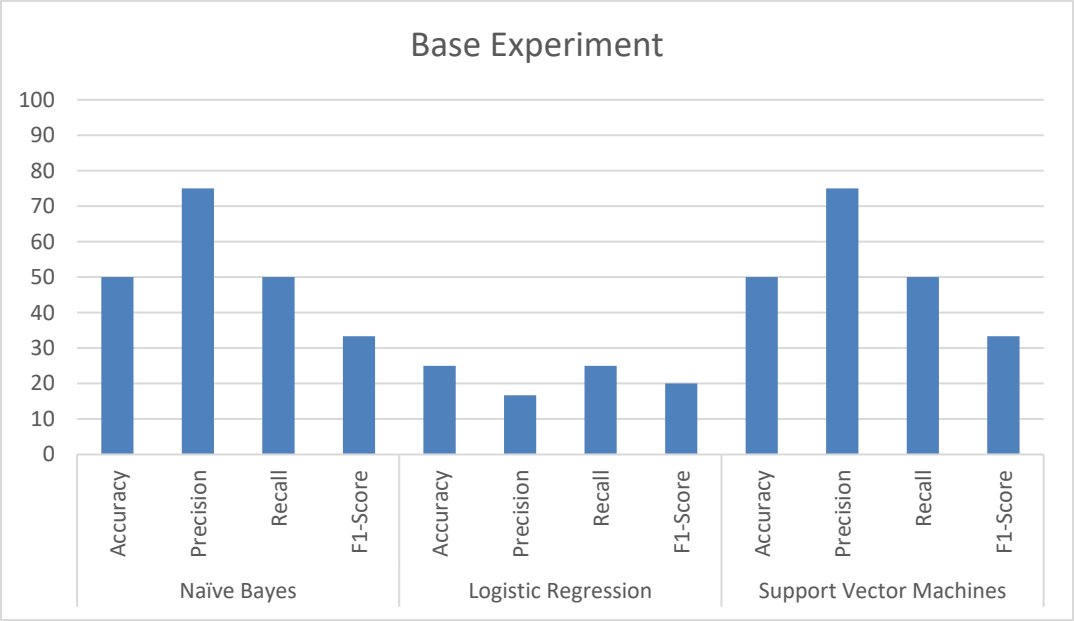

Figure 18: Results of Base Experiment

To perform the analysis, we will select the results of the best model, being either the Naïve Bayes or the Support Vector Machines, given that they performed equally across all four of our metrics. The significant reduction in the Logistic Regression Model's capability to accurately predict the gender is, however, noted.

In terms of accuracy, we were aiming for a value that is greater than 50%, which is the equivalent weight of the biological gender classes for performing a simple guess. None of these models have achieved that result, given the full dataset. This misleadingly suggests that biological gender cannot be accurately predicted through machine learning on zero-permission sensor data. Therefore, it was very important to conduct the additional set of experiments to determine whether the selection of smaller timespans or different sensor combinations would affect the result of the capability at all. Already, however, we are seeing logistic regression performing as the worst model for this dataset.

The macro average precision sits at 75%. With an accuracy of only 50%, this indicates that the model is correctly predicting the biological gender of the user, but with an extreme bias. To explain this further, let us consider what actually happened with the Naïve Bayes model.

After training, the validation dataset consisted of four data points or "intervals." These data points were evenly split across the male and female classes, two for each of them. Despite that, the model predicted all of the validation datapoints to be female. Performing the calculations, we see that this would result in the following highlighting the inflated precision, despite the low accuracy.

$$accuracy = \frac{2\ (correct\ predictions)}{4\ (total\ predictions\ made)} = 50\%$$

$$precision_{female} = \frac{2\ (correct\ female\ predictions)}{4\ (total\ female\ predictions)} = 50\%$$

$$precision_{male} = \frac{0\ (correct\ male\ predictions)}{0\ (total\ male\ predictions)} = 100\%$$

$$precision_{average} = \frac{100 + 50}{2} = 75\%$$

Recall additionally sits at 50%, since the model does make the correct prediction 100% of the time for the female class, but 0% of the time for the male class. This again indicates the model's lack of capability to correctly predict the gender of a user through machine learning on zero-permission sensors alone.

Since the F1-Score is a balance of the precision and the recall, and we have already analyzed the results of each of them to indicate the overall failure of the model, diving into its calculation is irrelevant.

6.2.2 The Effect of Timespan on Predicting Biological Gender

In this set of experiments, we are trying to answer two questions: what the smallest interval we can use to accurately predict biological gender through machine learning on zero-permission sensor readings is, and if using a smaller interval than the 3-day intervals used for the base experiment would improve the results.

For that, we consider the results of the 12 experiments conducted, as shown in Table 3. The cells are color-coded to indicate their proximity from 100%, with green indicating 100% and red indicating 0%. Additionally included are the 3-Day interval dataset results for relative comparison with the new intervals tested.

Table 3: Table showing the accuracy, precision, recall, and F1-Score results of running three machine learning models on four different interval data

	Interval	3-Day	1-Day	1-Hour	5-Minutes
Naïve Bayes	*Accuracy*	50	83.33	48.75	52.47
	Precision	75	91.67	62.4	60.35
	Recall	50	50	52.65	52.75
	F1-Score	33.33	45.45	38.99	41.98
Logistic Regression	*Accuracy*	25	50	75.62	50.41
	Precision	16.67	75	75.16	53.41
	Recall	25	50	75.69	51.64
	F1-Score	20	33.33	75.28	43.8
Support Vector Machines	*Accuracy*	50	83.33	72.5	54.06
	Precision	75	87.5	72.93	58.78
	Recall	50	83.33	71.52	55.39
	F1-Score	33.33	82.86	71.63	49.84

First, let's analyze the 5-minute intervals, hoping that these would be the most accurate since they would require the least data, and would therefore make for faster predictions. Unfortunately, this was not the case. While there is indeed an improvement for all 3 machine learning models used in terms of accuracy and F1-Score (balancing precision and recall), over using the 3-Day intervals, there it is not a very significant one. The highest accuracy achieved with the 5-minute intervals is only 54.06% with Support Vector Machines. The F1-Score is always below 50%, showing that the models are always biased, either to one class or the other, and very much so judging by these numbers.

The 1- hour intervals performed better than the 5 minutes overall, though not with the Naïve Bayes model. Using these intervals with the logistic regression algorithm provided the best results, reaching an accuracy of 75.62% or correctly predicting the gender of the user for a 1-hour interval for 121 of the 160 hours the model had never seen before. In terms of precision and recall, the following calculations are done to reach averages of 75.16% and 75.27%, respectively.

$$precision_{female} = \frac{51\ (correct\ female\ predictions)}{74\ (total\ female\ predictions)} = 68.92\%$$

$$precision_{male} = \frac{70\ (correct\ male\ predictions)}{86\ (total\ male\ predictions)} = 81.40\%$$

$$recall_{female} = \frac{51\ (correct\ female\ predictions)}{67\ (total\ females\ in\ validation\ set)} = 76.11\%$$

$$recall_{male} = \frac{70\ (correct\ male\ predictions)}{93\ (total\ males\ in\ validation\ set)} = 75.27\%$$

These are very positive indications, showing only a small bias towards the male class.

Correctly classifying the gender based on data collected for only 1 hour over 75% of the time is a leap forward relative to simply making a guess on the gender, the equivalent of using time intervals of 3 days.

If it were not for the negative results of the 3-Day intervals, one could easily assume that increasing the interval size would result in better results. This is evident by the increment from 5 minutes to 1 hour, and again by the increment from 1 hour to 1 day. With the 1-Day intervals, our models were able to reach accuracies of 83.33% both in the cases of Naïve Bayes and Support Vector Machines, correctly classifying the gender of the user given 1-Day of Zero-Permission Sensor data for 5 of the 6 days the model had never seen before. This is an extremely impressive result that may have been improved by the existence of a larger validation set. While both models had the same accuracies, however, the Naïve Bayes model had a significantly worse F1-Score, constantly predicting the day to be female, despite the existence of a male day in the validation set. The Support Vector Machine was significantly more balanced, with an F1-Score of 82.86%, indicating a very small bias. The calculations leading to these results are furnished below.

$$precision_{female} = \frac{3\ (correct\ female\ predictions)}{4\ (total\ female\ predictions)} = 75\%$$

$$precision_{male} = \frac{2\ (correct\ male\ predictions)}{2\ (total\ male\ predictions)} = 100\%$$

$$precision_{average} = \frac{100 + 75}{2} = 87.5\%$$

$$recall_{female} = \frac{3\ (correct\ female\ predictions)}{3\ (total\ females\ in\ validation\ set)} = 100\%$$

$$recall_{male} = \frac{2\ (correct\ male\ predictions)}{3\ (total\ males\ in\ validation\ set)} = 66.67\%$$

$$recall_{average} = \frac{100 + 66.67}{2} = 83.33\%$$

Looking at the results of this set of experiments alone would have led to the false indication that increasing the size of the interval immediately leads to an increment in the prediction accuracy. However being able to compare them to the "base" set of experiments, utilizing a dataset comprised of 3-Day intervals better informs the analysis, showing that there exists an interval that is too large to perform an accurate prediction. Put simply, there is a point where behavior becomes too generalized, simply human, and not separated by biological gender at all. Additionally, we believe that having a larger dataset may have contributed to better classification results with the 1-Day models, but unfortunately could not test this hypothesis due to the lack of data.

6.2.3. The Effect of Sensor Selection on Predicting Biological Gender

In this case, we are trying to answer two questions: which set of sensors will allow for the best prediction of biological gender through machine learning on zero-permission sensor readings is, and if using a different set of sensors than all three – as used for the base experiment – would improve the results.

For that, we consider the 21 experiment results shown in Table 4. The cells are color-coded to indicate their proximity from 100%, with green indicating 100% and red indicating 0%. Additionally included are the All-Sensor dataset results for relative comparison with the new sensor combinations tested.

First with Naïve Bayes, we notice that the accuracy does not improve with filtering sensors, relative to using all of them. In fact, when using a combination of two sensors, only using the accelerometer and gyroscope resulted in the same accuracy, with using any other combination of two sensors resulting in a noticeable degradation in accuracy. The pattern previously noticed with the all-sensor dataset continues, with models strongly favoring one class or the other, resulting in a rather poor F1-Score.

With Logistic Regression, an interesting pattern emerges. Although the accuracy continues to be very poor, reaching at most 50%, this "best accuracy" is only achieved through using any combination of sensors that does not include all 3 and is not just the accelerometer and ambient light. This indicates the Logistic Regression algorithm is more sensitive to data coming from multiple sensors, with results consistently (with one exception) being constantly better when the data was coming from either 1 or 2 sensors. While that improvement does indeed highlight the sensitivity of Logistic Regression, it still does not help us answer our question as no improvement in accuracy over the minimum required 50% was achieved.

The superiority of Support Vector Machines continues, once more, in this set of experiments. While faring poorly (though equal to the best) with the combination of all sensors, there is a significant improvement in SVM's ability to predict the biological gender when the data is coming from just one sensor – as long as that sensor is not the ambient light. Ambient light alone was unable to inform our Support Vector Machine enough for it to be able to accurately predict the biological gender of a user from a 3-Day interval. When data was coming from either accelerometer or the gyroscope, however, there was a significant improvement in its prediction ability, with accuracy rising from 50% in the cases of any other combination to 75%. Precision as well was higher at an average of 83.33% in both instances, and so was recall at 75%. This led to a much more-balanced average F1-Score of 73.33%.

So, which combination of sensor should be used? Generally, most combinations of sensors, at the 3-Day interval proven to be large enough for behavior to be genericized, will not be capable of accurately predicting the gender of a user. However, two sensors: the accelerometer and gyroscope, will significantly outperform the others when combined with learning on Support Vector Machines to enable more accurate prediction of the biological

gender.

Table 4: Table showing the accuracy, precision, recall, and F1-Score results of running three machine learning models on seven different combinations of sensor readings

	Sensor(s)	Accuracy	Precision	Recall	F1-Score
Naïve Bayes	*All Sensors*	50	75	50	33.33
	Accelerometer	50	75	50	33.33
	Gyroscope	50	75	50	33.33
	Ambient Light	50	75	50	33.33
	Accelerometer & Gyroscope	50	75	50	33.33
	Accelerometer & Ambient Light	25	62.5	50	20
	Gyroscope & Ambient Light	25	62.5	50	20
Logistic Regression	*All Sensors*	25	16.67	25	20
	Accelerometer	50	75	50	33.33
	Gyroscope	50	75	50	33.33
	Ambient Light	50	75	50	33.33
	Accelerometer & Gyroscope	50	75	50	33.33
	Accelerometer & Ambient Light	25	16.67	25	20
	Gyroscope & Ambient Light	50	75	50	33.33
Support Vector Machines	*All Sensors*	50	75	50	33.33
	Accelerometer	75	83.33	75	73.33
	Gyroscope	75	83.33	75	73.33
	Ambient Light	50	75	50	33.33
	Accelerometer & Gyroscope	50	75	50	33.33
	Accelerometer & Ambient Light	50	75	50	33.33
	Gyroscope & Ambient Light	50	50	50	50

6.2.4. The Effect of Model Selection on Predicting Biological Gender

In this case, we are trying determine which type of model produces the best results of predicting the biological gender for a user, given zero-permission sensor data. In order to answer this question, we repeated all of our experiments three times, one for each machine learning model: Naïve Bayes, Logistic Regression, and Support Vector Machines. This means that each model was run 10 times, to a total of 30 experiments, producing the results shown in Table 2.

These results can be hard to digest, so let us consider the model with the best results achieved, for each of the algorithms we assessed, recorded in Table 6. Again, the cells are color-coded to indicate their proximity from 100%, with green indicating 100% and red indicating 0%.

Table 5: Table showing the contributing parameters (interval and sensor combination) and the metrics with their values for each of the three machine learning algorithms tested

Algorithm	Interval	Sensors	Metric	Value
Naïve Bayes	1-Day	Accelerometer, Gyroscope, & Ambient Light	Accuracy	83.33
			Precision	91.67
			Recall	50
			F1-Score	45.45
Logistic Regression	1-Hour	Accelerometer, Gyroscope, & Ambient Light	Accuracy	75.62
			Precision	75.16
			Recall	75.69
			F1-Score	75.28
Support Vector Machines	1-Day	Accelerometer, Gyroscope, & Ambient Light	Accuracy	83.33
			Precision	87.5
			Recall	83.33
			F1-Score	82.86

While Naïve Bayes was capable of reaching an accuracy equal to the highest, at 83.33%, and the overall highest precision throughout the 30 experiments, when paired with the 1-Day interval data coming from the accelerometer, gyroscope, and ambient light, it had extremely poor recall at only 50%, greatly hurting its F1-Score. Overall, we would not call this a successful model, at its best, with the task of predicting a user's biological gender from zero-permission sensor data.

Logistic Regression consistently performed the poorest of the three assessed algorithms, with its highest accuracy being 75.62%. It was interesting to see, however, that this accuracy was balanced with an admirable F1-Score, balancing both the precision and the recall. It was also very interesting to see that while the other algorithms performed their best with the 1-Day intervals, Logistic Regression was more fine-grained, performing best with the data from the 1-Hour intervals. Consistent with the results of the other algorithms, however, it did so with the full combination of sensors.

Support Vector Machines are definitely our selection, among the assessed algorithms, for the task of predicting a user's biological gender from zero-permission sensor data. Not only was it able to achieve an accuracy of 83.33% for 1-Day interval data from all three sensors, it did so by only missing 1 out of the given 6 datapoints. This led to both a high precision and high recall, therefore a high F1-Score. While that was the case in its best run, Support Vector Machines consistently performed better than the other algorithms, regardless of interval size or sensor combination. As aforementioned, the only exception was the 1-Hour All-Sensors variant of our experiments where the Logistic Regression was superior, but even then, not only did SVM come in second place, but it was a close second as well, being only 3.12% less accurate. This means that it incorrectly classified only 5 more datapoints, of the 160 provided,

than Logistic Regression did. It also did so with balance, having a high overall F1-Score, which may be attributed to the 5 incorrect classifications. Overall, Support Vector Machines were the superior choice, among the three algorithms, and led to impressive results.

These findings are fairly consistent with the works surveyed, with Naïve Bayes often performing well, but in very specific circumstances, Logistic Regression being fairly mediocre at the task of learning from zero-permission sensor data, and Support Vector Machines consistently shining as one of the best, if not the best, tested traditional machine learning algorithm.

Chapter 7

Conclusions & Future Work

Determining the biological gender of a user is extremely important. It is the top factor used for targeting content and advertising in a cold-start problem – where the companies do not know any information about the user – or battling online impersonation – with platforms bearing the brunt of actioning accounts that misrepresent who they are online. This targeting of content and advertising is the lifeline for many of these companies, and it is a significant force of the way the internet works right now. This is clearly illustrated by the trends in advertising revenue showing a growth in the United States alone from $107.5 billion in 2018 [7] to $189.3 billion [72] in 2022, accounting for almost 1% of the entire United States Gross Domestic Product [77]. Even with the financials aside, we have shown that YouTube's recommendation algorithm alone is responsible for over 10% of all global Internet traffic [61], showing that recommendation and personalization play a huge factor in how everybody interacts with the internet today.

Therefore, it is important to ask if it is possible to accurately predict the biological gender of a user based solely on machine learning on zero-permission sensor data. For this question, our answer would have to be a resounding yes. While most models were not able to perform better than a binary guess, some models had accuracies that were significantly higher. We recommend that future researchers attempt to work with a larger dataset that may be more insightful in the potential accuracies that can be reached. While we were limited to a small pool of volunteers (six males and five females) and a tight window of data collection due to COVID-19, we recommend increasing the scope both in terms of the number of participants, as well as the number of days data is collected for each individual. Having a larger dataset will help corroborate the results of the experiment to make it more accurate for predicting gender on a larger scale. This would propel the results of the experiment forward, allowing for a more inclusive prediction across a larger range of people and duration.

It may also be of merit to assess how accurate the biological gender determination would be with a different variety of volunteers selected. While we controlled discriminatory factors such as age, activity level, and height, we recommend diversifying not only these factors, but others that might be considered relevant to particular biological genders. Similarly, this would lead to a more comprehensive dataset that accounts for such nuances, allowing for the prediction to be more conclusive at a larger and more accurate scale.

Additionally, we suggest that researchers test whether these results are affected if a user were to have a different selected gender over their biological gender. Finally, we recommend that

future work attempt to also include the age group of a user as part of the user profiling attributes that could be predicted.

It is also important to ask about the minimum interval that can be used to accurately predict the biological gender of a user based solely on machine learning on zero-permission sensor data. For this question, not only have we shown that the minimum interval that can be used without the data being too sparse is 1 day, but we have also shown that using the 1-day data is even more accurate than using the 3-day data. This is because all human behavior will blend at some point and become genericized. We recommend that future researchers dig deeper into this question. While we do determine that the 1-day interval size is the best in our assessed interval sizes, the next larger interval was 3 times as large, and the next smaller interval was one twenty-fourth as large. We encourage future researchers to experiment with intervals that are longer than 1 day, but shorter than 3 days, and shorter than 1 day but longer than 1 hour to test if the true minimum interval is somewhere in between, and if higher accuracies can be reached with the new proposed interval sizes.

But which sensors should be used? For this question, we have shown that working with the data coming from all three of the accelerometer, gyroscope, and ambient light sensors produced the best results. We recommend that future researchers try experimenting with other zero-permission sensors, such as the magnetometer, to see if they will influence the accuracy at all. Additionally, we recommend future researchers try different combinations of intervals and sensors. Our experiment kept the interval size constant when we were experimenting with different sensors and kept the sensors constant when we were experimenting with different interval sizes. It is possible that reaching higher accuracies may be a combination of manipulating both factors simultaneously.

And how can we determine which data to use, given the large volume and density of sensor readings? For this effort, we have chosen to come up with a data processing pipeline that relies on traditional data filtering mechanisms to focus on the sensor readings of the most significance. Perhaps future works should attempt a separate effort of performing machine learning on the raw data to extract the features that would have the most participatory effect to the subsequent machine learning effort of classifying a user's demographics.

Which machine learning models are best to accurately predict biological gender based solely on machine learning on zero-permission sensor data? For this question, we have shown that Support Vector Machines perform best for the task, relative to Naïve Bayes and Logistic Regression models, reaching accuracies of up to 83.33% in a balanced approach unbiased towards any one class or the other. We recommend that future researchers experiment with other machine learning algorithms and potentially with deep learning models such as convolutional neural networks in environments that can ingest the full datasets.

References

[1] “2018 Domo Report,” Domo, inc., 2018.

[2] “2019Q4 Alphabet Earnings,” Alphabet, 2019.

[3] “About targeting for video campaigns,” Google Ads Help. [Online]. Available: https://support.google.com/google-ads/answer/2454017?hl=en.

[4] P. Convington, J. Adams, and E. Sargin, “Deep Neural Networks for YouTube Recommendations,” in proceedings of Recsys, 2016, pp. 191–198.

[5] “Facebook Advertising Targeting Options,” Facebook for Business. [Online]. Available: https://www.facebook.com/business/ads/ad-targeting.

[6] “Facebook Reports Second Quarter 2019 Results,” Facebook.

[7] “Full Year 2018 IAB Internet Advertising Revenue Report,” PricewaterhouseCoopers.

[8] “Half Year 2019 IAB Internet Advertising Revenue Report,” PricewaterhouseCoopers.

[9] “Internet Phenomena Report,” Sandvine Incorporated, 2019.

[10] B. Lika, K. Kolomvatsos, and S. Hadjiefthymiades, “Facing the cold start problem in recommender systems,” Expert Systems with Applications, vol. 41, no. 4, pp. 2065–2073, 2014.

[11] “Mobile Phenomena Report,” Sandvine, 2019.

[12] H. Pauzer, “71% of Consumers Prefer Personalized Ads,” Adlucent. [Online]. Available: https://www.adlucent.com/blog/2016/71-of-consumers-prefer-personalized-ads.

[13] J. E. Solsman, “YouTube's AI is the puppet master over most of what you watch,” CNet.Com, 10-Jan-2018. [Online]. Available: www.cnet.com.

[14] “Facebook Q4 2019 Results,” Facebook.

[15] “Community Standards Enforcement Report,” Facebook.

[16] Greenspan, A., 2019. Facebook, Inc.. Reality Check. PlainSite.

[17] C. Timberg and E. Dwoskin, "Twitter is sweeping out fake accounts like never before, putting user growth at risk", Washington Post, 2020. [Online]. Available: https://www.washingtonpost.com/technology/2018/07/06/twitter-is-sweeping-out-fake-accounts-like-never-before-putting-user-growth-risk/.

[18] I. Hannah Lucinda Smith, "Twitter shuts fake accounts that spout tyrants' propaganda", Thetimes.co.uk, 2020. [Online]. Available:

https://www.thetimes.co.uk/article/twitter-bans-20-000-fake-accounts-with-state-links-rz3n9wg8q.

[19] "Twitter to Delete 6% of All Accounts in Huge Cull", The Independent, 2020. [Online]. Available: https://www.independent.co.uk/life-style/gadgets-and-tech/news/twitter-fake-followers-lost-delete-accounts-cull-a8444236.html.

[20] H. Allcott and M. Gentzkow, "Social Media and Fake News in the 2016 Election", Journal of Economic Perspectives, vol. 31, no. 2, pp. 211-236, 2017. Available: 10.1257/jep.31.2.211.

[21] International Translation Resources, Why Do People Uninstall Apps?. 2019.

[22] "Authentication - User Interaction - macOS - Human Interface Guidelines - Apple Developer", Apple Developer, 2020. [Online]. Available: https://developer.apple.com/design/human-interface-guidelines/macos/user-interaction/authentication/.

[23] S. Kanoje, S. Girase, and D. Mukhopadhyay, 'User Profiling Trends, Techniques and Applications', arXiv:1503.07474 [cs], Mar. 2015.

[24] D. Poo, B. Chng and Jie-Mein Goh, "A hybrid approach for user profiling", 36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the, 2003. Available: 10.1109/hicss.2003.1174242.

[25] C. Avery and R. Zeckhauser, "Recommender systems for evaluating computer messages", Communications of the ACM, vol. 40, no. 3, pp. 88-89, 1997. Available: 10.1145/245108.245127.

[26] M. Fuchs and M. Zanker, "Multi-criteria Ratings for Recommender Systems: An Empirical Analysis in the Tourism Domain", Lecture Notes in Business Information Processing, pp. 100-111, 2012. Available: 10.1007/978-3-642-32273-0_9.

[27] M. Ramscar, H. Pain and J. Lee, "Do We Know What the User Knows, and Does It Matter? The Epistemics of User Modelling", User Modeling, pp. 429-431, 1997. Available: 10.1007/978-3-7091-2670-7_42.

[28] Nichols, D. Implicit rating and filtering. In Proceedings of the Fifth DELOS Workshop on Filtering and Collaborative Filtering (Budapest, Hungary, 1998), ERCIM, pp. 31–36

[29] D. Warner and M. Myer, "Implicit rating of retrieved information in an information search system", 6665655B1, 2000.

[30] M. Najafabadi, M. Mahrin, S. Chuprat and H. Sarkan, "Improving the accuracy of collaborative filtering recommendations using clustering and association rules mining on implicit data", Computers in Human Behavior, vol. 67, pp. 113-128, 2017. Available: 10.1016/j.chb.2016.11.010.

[31] M. Ranjbar, P. Moradi, M. Azami and M. Jalili, "An imputation-based matrix factorization method for improving accuracy of collaborative filtering systems",

Engineering Applications of Artificial Intelligence, vol. 46, pp. 58-66, 2015. Available: 10.1016/j.engappai.2015.08.010.

[32] M. Nasiri and B. Minaei, "Increasing prediction accuracy in collaborative filtering with initialized factor matrices", The Journal of Supercomputing, vol. 72, no. 6, pp. 2157-2169, 2016. Available: 10.1007/s11227-016-1717-8.

[33] S. Kanoje, D. Mukhopadhyay and S. Girase, "User Profiling for University Recommender System Using Automatic Information Retrieval", Procedia Computer Science, vol. 78, pp. 5-12, 2016. Available: 10.1016/j.procs.2016.02.002.

[34] Y. Yang, "Web user behavioral profiling for user identification", Decision Support Systems, vol. 49, no. 3, pp. 261-271, 2010. Available: 10.1016/j.dss.2010.03.001.

[35] J. Tang, L. Yao, D. Zhang and J. Zhang, "A Combination Approach to Web User Profiling", ACM Transactions on Knowledge Discovery from Data, vol. 5, no. 1, pp. 1-44, 2010. Available: 10.1145/1870096.1870098.

[36] J. De Andrés, B. Pariente, M. Gonzalez-Rodriguez and D. Fernandez Lanvin, "Towards an automatic user profiling system for online information sites", Online Information Review, vol. 39, no. 1, pp. 61-80, 2015. Available: 10.1108/oir-06-2014-0134.

[37] Alpaydin, E., 2020. Introduction to Machine Learning. 4th ed. MIT Press, p.3-11.

[38] S. Narain, T. Vo-Huu, K. Block and G. Noubir, "Inferring User Routes and Locations Using Zero-Permission Mobile Sensors", 2016 IEEE Symposium on Security and Privacy (SP), 2016. Available: 10.1109/sp.2016.31.

[39] Liang, Y., Cai, Z., Han, Q. and Li, Y., 2017. Location Privacy Leakage through Sensory Data. Security and Communication Networks, 2017, pp.1-12.

[40] Android Developers Documentation. 2023. Motion Sensors. [online] Available: https://developer.android.com/guide/topics/sensors/sensors_motion.

[41] Android Developers Documentation. 2023. Position Sensors. [online] Available: https://developer.android.com/guide/topics/sensors/sensors_position.

[42] Android Developers Documentation. 2023. Environment Sensors. [online] Available: https://developer.android.com/guide/topics/sensors/sensors_environment.

[43] L. Joshua and K. Varghese, "Accelerometer-Based Activity Recognition in Construction", Journal of Computing in Civil Engineering, vol. 25, no. 5, pp. 370-379, 2011. Available: 10.1061/(asce)cp.1943-5487.0000097.

[44] E. Zdravevski, B. Risteska Stojkoska, M. Standl and H. Schulz, "Automatic machine-learning based identification of jogging periods from accelerometer measurements of adolescents under field conditions", PLOS ONE, vol. 12, no. 9, p. e0184216, 2017. Available: 10.1371/journal.pone.0184216.

[45] "Introduction to the Logistic Regression Model", Applied Logistic Regression, pp. 1-33, 2013. Available: 10.1002/9781118548387.ch1.

[46] L. Breiman, "Random Forests", Machine Learning, vol. 45, no. 1, pp. 5-32, 2001. Available: 10.1023/a:1010933404324.

[47] P. Geurts, D. Ernst and L. Wehenkel, "Extremely randomized trees", Machine Learning, vol. 63, no. 1, pp. 3-42, 2006. Available: 10.1007/s10994-006-6226-1.

[48] C. Cortes and V. Vapnik, "Support-vector networks", Machine Learning, vol. 20, no. 3, pp. 273-297, 1995. Available: 10.1007/bf00994018.

[49] T. Cover and P. Hart, "Nearest neighbor pattern classification", IEEE Transactions on Information Theory, vol. 13, no. 1, pp. 21-27, 1967. Available: 10.1109/tit.1967.1053964.

[50] O. Aziz, M. Musngi, E. Park, G. Mori and S. Robinovitch, "A comparison of accuracy of fall detection algorithms (threshold-based vs. machine learning) using waist-mounted tri-axial accelerometer signals from a comprehensive set of falls and non-fall trials", Medical & Biological Engineering & Computing, vol. 55, no. 1, pp. 45-55, 2016. Available: 10.1007/s11517-016-1504-y.

[51] W. Wu, S. Dasgupta, E. Ramirez, C. Peterson and G. Norman, "Classification Accuracies of Physical Activities Using Smartphone Motion Sensors", Journal of Medical Internet Research, vol. 14, no. 5, p. e130, 2012. Available: 10.2196/jmir.2208.

[52] M. Albert, K. Kording, M. Herrmann and A. Jayaraman, "Fall Classification by Machine Learning Using Mobile Phones", PLoS ONE, vol. 7, no. 5, p. e36556, 2012. Available: 10.1371/journal.pone.0036556.

[53] M. Kilany, A. Hassanien and A. Badr, "Accelerometer-based human activity classification using Water Wave Optimization approach", 2015 11th International Computer Engineering Conference (ICENCO), 2015. Available: 10.1109/icenco.2015.7416344.

[54] X. Ren, W. Ding, S. Crouter, Y. Mu and R. Xie, "Activity recognition and intensity estimation in youth from accelerometer data aided by machine learning", Applied Intelligence, vol. 45, no. 2, pp. 512-529, 2016. Available: 10.1007/s10489-016-0773-3.

[55] A. Zaharis, A. Martini, P. Kikiras and G. Stamoulis, "“User Authentication Method and Implementation Using a Three-Axis Accelerometer”", Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 192-202, 2010. Available: 10.1007/978-3-642-16644-0_18.

[56] W. Shi, J. Yang, Yifei Jiang, Feng Yang and Yingen Xiong, "SenGuard: Passive user identification on smartphones using multiple sensors", 2011 IEEE 7th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2011. Available: 10.1109/wimob.2011.6085412.

[57] C. Nickel and C. Busch, "Classifying accelerometer data via hidden Markov models to authenticate people by the way they walk", IEEE Aerospace and Electronic Systems Magazine, vol. 28, no. 10, pp. 29-35, 2013. Available: 10.1109/maes.2013.6642829.

[58] N. Gao, W. Shao and F. Salim, "Predicting Personality Traits From Physical Activity Intensity", Computer, vol. 52, no. 7, pp. 47-56, 2019. Available: 10.1109/mc.2019.2913751.

[59] S. Narain, T. Vo-Huu, K. Block and G. Noubir, "The Perils of User Tracking Using Zero-Permission Mobile Apps", IEEE Security & Privacy, vol. 15, no. 2, pp. 32-41, 2017. Available: 10.1109/msp.2017.25.

[60] Android Developers Documentation. 2021. Sensor Manager. [online] Available: https://developer.android.com/reference/android/hardware/SensorManager

[61] "Internet Phenomena Report," Sandvine Incorporated, 2022.

[62] "Mobile Phenomena Report," Sandvine, 2021.

[63] "Facebook Q4 2022 Results," Facebook.

[64] "Community Standards Enforcement Report Q4 2022," Facebook.

[65] D. Swan, "Twitter: Former FBI, CIA security expert says 80% of accounts 'bots'," The Australian, 31-Aug-2022. [Online].

[66] "Sklearn.naive_bayes.Gaussiannb," Scikit-learn.

[67] "Sklearn.linear_model.SGDClassifier," Scikit-learn.

[68] M. Hashemi, "Enlarging smaller images before inputting into convolutional neural network: zero-padding vs. interpolation", J Big Data 6, 98, 2019. Available: 10.1186/s40537-019-0263-7.

[69] H. Tang, A. Ortis, S. Battiato. "The Impact of Padding on Image Classification by Using Pre-trained Convolutional Neural Networks." Image Analysis and Processing – ICIAP 2019. ICIAP 2019, vol 11752. Springer, Cham. Available: 10.1007/978-3-030-30645-8_31.

[70] "2022Q4 Alphabet Earnings," Alphabet, 2022.

[71] "Facebook Reports Fourth Quarter 2022 Results," Facebook.

[72] "Full Year 2021 IAB Internet Advertising Revenue Report," PricewaterhouseCoopers.

[73] D. Doty, "A reality check on advertising relevancy and Personalization," Forbes, 13-Aug-2019. [Online].

[74] "Trends in Customer Trust," Salesforce Research.

[75] B. Ortutay, "Twitter says it removes 1 million spam accounts a day," AP NEWS, 07-Jul-2022. [Online].

[76] M. Simmons, J. Suk Lee. "Catfishing: A Look into Online Dating and Impersonation." Social Computing and Social Media, Springer, Cham. Available: 10.1007/978-3-030-49570-1_24.

[77] "Gross Domestic Product, Fourth Quarter and Year 2021." The United States Bureau of Economic Analysis.

[78] A. Zheng, Evaluating machine learning models. O'Reilly Media, Inc., 2015.

[79] F. ElHussiny. "An Econometric Analysis of Factors Affecting Land Values in Western Oklahoma."

[80] A. Fattahi, A. Golroo, and M. Ghatee. "Driver Behavior Assessment Using Multi-Layer Perceptron and Random Forest Via Smartphone Sensors and Obd II," Social Science Research Network, Elsevier, 2023. Available: 10.2139/ssrn.4416107.

[81] B.R. dos Reis, S. Sujani, D.R. Fuka, Z.M. Easton, and R.R. Whit. "Comparison among grazing animal behavior classification algorithms for use with open-source wearable sensors," Social Science Research Network, Elsevier, 2023. Available: 10.2139/ssrn.4414970.

[82] N. Ahmed, J. Rafiq, and R. Islam, "Enhanced Human Activity Recognition Based on Smartphone Sensor Data Using Hybrid Feature Selection Model," Sensors 2020, 20, 317, MDPI. Available: 10.3390/s20010317.

[83] Y. Chen and C. Shen. "Performance Analysis of Smartphone-Sensor Behavior for Human Activity Recognition," IEEE Access, no. 5, pp. 3095 – 3110, 2017. Available: 10.1109/ACCESS.2017.2676168.

[84] L. Fang, S. Yishui, and C. Wei. "Up and Down Buses Activity Recognition using Smartphone Accelerometer." Proceedings of IEEE Information Technology, Networking, Electronic and Automation Control Conference, pp. 761-765, 2016. Available: 10.1109/ITNEC.2016.7560464.

[85] X. Yin, W. Shen, J. Samarabandu, X. Wang. "Human Activity Detection Based on Multiple Smart Phone Sensors and Machine Learning Algorithms," Proceedings of the IEEE 19th International Conference on Computer Supported Cooperative Work in Design (CSCWD), pp. 582 – 587, 2015. Available: 10.1109/CSCWD.2015.7231023.

[86] H. Braganca, J. Colonna, W. Lima, and E. Souto. "A Smartphone Lightweight Method for Human Activity Recognition Based on Information Theory." Sensors 2020, 20, 7, MDPI. Available: 10.3390/s20071856.

[87] A.S. Abdull Sukor, A. Zakaria, and N. Abdul Rahim. "Activity Recognition using Accelerometer Sensor and Machine Learning Classifiers," Proceedings of the IEEE 14th International Colloquium on Signal Processing & its Applications (CSPA 2018), IEEE, 2018. Available: 10.1109/CSPA.2018.8368718.

[88] S. Shakya, C. Zhang, and Z. Zhou. "Comparative Study of Machine Learning and Deep Learning Architecture for Human Activity Recognition Using Accelerometer Data," International Journal of Machine Learning and Computing, 8, 6, Springer, 2018. Available: 10.18178/ijmlc.2018.8.6.748.

[89] X. Yang, Y. Zhao, G.M. Street, Y. Huang, S.D. Filip To, and J.L. Purswell. "Classification of Broiler Behaviours using Triaxial Accelerometer and Machine Learning," Animal: The International Journal of Animal Biosciences, 15, Elsevier, 2021. Available: 10.1016/j.animal.2021.100269.

[90] S. Park, S. Heo, and C. Park. “Accelerometer-based Smartphone Step Detection using Machine Learning Technique,” Proceedings of the IEEE International Electrical Engineering Congress, 5, IEEE, 2017. Available: 10.1109/IEECON.2017.8075875.

[91] S. Edeib, R. Dziyauddin, and N. Amir. “Fall Detection and Monitoring using Machine Learning: A Comparative Study,” International Journal of Advanced Computer Science and Applications, 14, 2, 2023. Available: 10.14569/IJACSA.2023.0140284.

[92] R. Husain, R. Khan, and R. Tagi. “Machine Learning Modelling Based on Smartphone Sensor Data of Human Activity Recognition,” i-Manager’s Journal on Computer Science, 10, 4, pp. 1 – 8, 2023. Available: 10.26634/jcom.10.4.19341.

[93] L. Riaboff, S. Poggi, A. Madouasse, S. Couvreur, S. Aubin, N. Bedere, E. Goumand, A. Chauvin, and G. Plantier. “Development of a methodological framework for a robust prediction of the main behaviors of dairy cows using a combination of machine learning algorithms on accelerometer data,” Computers and Electronics in Agriculture, 169, 2020, Elsevier. Available: 10.1016/j.compag.2019.105179.

[94] L. Palmerini, J. Klenk, C. Becker, and L. Chiari, “Accelerometer-Based Fall Detection Using Machine Learning: Training and Testing on Real-World Falls,” Sensors, vol. 20, no. 22, p. 6479, Nov. 2020. Available: 10.3390/s20226479.

[95] E. Gomes, L. Bertini, W. R. Campos, A. P. Sobral, I. Mocaiber, and A. Copetti, “Machine Learning Algorithms for Activity-Intensity Recognition Using Accelerometer Data,” Sensors, vol. 21, no. 4, p. 1214, Feb. 2021. Available: 10.3390/s21041214.

[96] T. Althobaiti, S. Katsigiannis, and N. Ramzan, “Triaxial Accelerometer-Based Falls and Activities of Daily Life Detection Using Machine Learning,” Sensors, vol. 20, no. 13, p. 3777, Jul. 2020. Available: 10.3390/s20133777.

[97] S. E. Ali, A. N. Khan, S. Zia and M. Mukhtar, "Human Activity Recognition System using Smart Phone based Accelerometer and Machine Learning," 2020 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT), 2020, pp. 69-74. Available: 10.1109/IAICT50021.2020.9172037.

[98] S. Strada, J. Paris, F. Piccoli, D. P. Tucci, P. Casali and S. Savaresi, "Machine Learning Recognition of Gait Identity via Shoe Embedded Accelerometer," 2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics), 2020, pp. 852-857. Available: 10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics50389.2020.00146.

[99] A. Ferrari, D. Micucci, M. Mobilio, and P. Napoletano. “Human Activities Recognition Using Accelerometer and Gyroscope.” Ambient Intelligence. AmI 2019. Lecture Notes in Computer Science, 11912, Springer. Available: 10.1007/978-3-030-34255-5_28.

[100] J. Al Nahian, T. Ghosh, H. Al Banna, M. Aseeri, M. Nassir Uddin, M. Ahmed, M. Mahmud, and M. Kaiser. "Towards an Accelerometer-Based Elderly Fall Detection System Using Cross-Disciplinary Time Series Features," IEEE Access, 9, pp. 39413-39431, 2021. Available: 10.1109/ACCESS.2021.3056441.

Appendix 1: Institutional Review Board Approval

CASE #2019-2020-051

THE AMERICAN UNIVERSITY IN CAIRO
INSTITUTIONAL REVIEW BOARD

To: Ahmed ElHussiny

Cc: Dr. Sherif Aly

From: Atta Gebril, Chair of the IRB

Date: Jan 5, 2020

Re: Approval of study

This is to inform you that I reviewed your revised research proposal entitled "User Profiling through Zero-Permission Sensors and Machine Learning" and determined that it required consultation with the IRB under the "expedited" category. As you are aware, the members of the IRB suggested certain revisions to the original proposal, but your new version addresses these concerns successfully. The revised proposal used appropriate procedures to minimize risks to human subjects and that adequate provision was made for confidentiality and data anonymity of participants in any published record. I believe you will also make adequate provision for obtaining informed consent of the participants.

This approval letter was issued under the assumption that you have not started data collection for your research project. Any data collected before receiving this letter could not be used since this is a violation of the IRB policy.

Please note that IRB approval does not automatically ensure approval by CAPMAS, an Egyptian government agency responsible for approving some types of off-campus research. CAPMAS issues are handled at AUC by the office of the University Counsellor, Dr. Ashraf Hatem. The IRB is not in a position to offer any opinion on CAPMAS issues, and takes no responsibility for obtaining CAPMAS approval.

This approval is valid for only one year. In case you have not finished data collection within a year, you need to apply for an extension.

Thank you and good luck.

Atta Gebril

Dr. Atta Gebril
IRB chair, The American University in Cairo
2046 HUSS Building
T: 02-26151919
Email: agebril@aucegypt.edu

Institutional Review Board
The American University in Cairo
AUC Avenue, P.O. Box 74
New Cairo 11835, Egypt.
tel 20.2.2615.1000
fax 20.2.27957565
Email: aucirb@aucegypt.edu

Appendix 2: Institutional Review Board Extension

CASE #2020-2021-098

THE AMERICAN UNIVERSITY IN CAIRO
INSTITUTIONAL REVIEW BOARD

To: Ahmed ElHussiny
Cc: Dr. Sherif Ali
From: Atta Gebril, Chair of the IRB
Date: Jan. 5, 2021
Re: Extension of CASE #2019-2020-051

This is to inform you that I reviewed your revised research proposal entitled " **User Profiling through Zero-Permission Sensors and Machine Learning**" and determined that it required consultation with the IRB under the "expedited" category. As you are aware, the members of the IRB suggested certain revisions to the original proposal, but your new version addresses these concerns successfully. The revised proposal used appropriate procedures to minimize risks to human subjects and that adequate provision was made for confidentiality and data anonymity of participants in any published record. I believe you will also make adequate provision for obtaining informed consent of the participants.

This approval letter was issued under the assumption that you have not started data collection for your research project. Any data collected before receiving this letter could not be used since this is a violation of the IRB policy.

Please note that IRB approval does not automatically ensure approval by CAPMAS, an Egyptian government agency responsible for approving some types of off-campus research. CAPMAS issues are handled at AUC by the office of the University Counsellor, Dr. Ashraf Hatem. The IRB is not in a position to offer any opinion on CAPMAS issues, and takes no responsibility for obtaining CAPMAS approval.

This approval is valid for only one year. In case you have not finished data collection within a year, you need to apply for an extension.

Thank you and good luck.

Dr. Atta Gebril
IRB chair, The American University in Cairo
2046 HUSS Building
T: 02-26151919
Email: agebril@aucegypt.edu

Institutional Review Board
The American University in Cairo
AUC Avenue, P.O. Box 74
New Cairo 11835, Egypt.
tel 20.2.2615.1000
fax 20.2.27957565
Email: aucirb@aucegypt.edu

Appendix 3: Data Processing Pipeline – Data Slicing Step Source Code

```
import sqlite3
import pandas as pd

con = sqlite3.connect("../Databases/database_copy_1621497091932_day16_f_ex_p.db")

cur = con.cursor()

cur.execute("SELECT * FROM sensordata limit 1")
firstrow = cur.fetchall()[0]
startTime = firstrow[3]
sliceCount = 1
sliceSize = 24 * 60 * 60 * 1000
sliceType = "day"
offset = 0
total = 0
batchSize = 10000
saveSize = 1000000
lastFrameHadData = True

df = pd.read_sql_query("SELECT * from sensordata where 1 != 1", con)

while lastFrameHadData:
    processedTime = 0
    if (len(df.index) > 0):
        processedTime = (df.iloc[-1]["sensortime"] - startTime) / (sliceSize)

        processedId = df.iloc[-1]["_id"]

        print("Processing. Reached {} records totalling {:10.4f}
{}s".format(processedId, processedTime, sliceType))

    try:
        newFrame = pd.read_sql_query("SELECT * from sensordata LIMIT {} OFFSET
{}".format(batchSize, offset), con)

        newFrameSize = len(newFrame.index)

        total += newFrameSize

        if(newFrameSize > 0):
            offset += batchSize
```

```
            outsideTheSlice = newFrame[newFrame["sensortime"] >= startTime +
sliceCount * sliceSize]

            if(len(outsideTheSlice.index) > 0):
                df = df.append(newFrame[newFrame["sensortime"] < startTime +
sliceCount * sliceSize])

                df.to_csv("./output/female/{}/{}_{}.csv".format(sliceType,
sliceCount, total))

                sliceCount += 1
                df = outsideTheSlice
            else:
                df = df.append(newFrame)

                if (len(df.index) >= saveSize):
                    df.to_csv("./output/female/{}/{}_{}.csv".format(sliceType,
sliceCount, total))

                    cur.execute("DELETE from sensordata where _id in (SELECT _id
from sensordata LIMIT {})".format(offset))
                    offset = 0

                    df = pd.read_sql_query("SELECT * from sensordata where 1 !=
1", con)
        else:
            lastFrameHadData = False

            df.to_csv("./output/female/{}/{}_{}.csv".format(sliceType,
sliceCount, total))
    except:
        lastFrameHadData = False
        df.to_csv("./output/female/{}/{}_{}.csv".format(sliceType, sliceCount,
total))
con.close()
```

Appendix 4: Data Processing Pipeline – Delta Calculation Step Source Code

```
import pandas as pd
import os
import numpy as np
sensors = [1, 2, 3]
gender = "male"
curslice = "snz"
filenames = []
for (dpath, dnames, fnames) in
os.walk(os.path.join(".","output",gender,curslice)):
    filenames.extend(fnames)
filenames = sorted(filenames, key = lambda filename :
int(filename[0:filename.index("_")]))
day = 1
totalRows = 0
newTotalRows = 0
for idx in range(0, len(filenames)):
    filename = filenames[idx]
    print("PROGRESS: {}%. FILE NUMBER: {}/{}. NAME:
{}".format(idx/len(filenames)*100, idx+1, len(filenames), filename))
    curday = filename[0:filename.index("_")]
    if (curday != day):
        day = curday
    path = os.path.join(".", "output", gender, curslice, filename)
    fileFrame = pd.read_csv(path)
    for sensor in sensors:
        print("--> SENSOR: {}".format(sensor))
        df = fileFrame[fileFrame["sensor"]==sensor].iloc[:, 2:]
        totalRows += len(df.index)
        if len(df.index) > 0:
            df["sensortime"] = df["sensortime"].diff()
            df = df[df["sensortime"] <= 1000]
            if sensor == 3:
                valDiffFrame = pd.DataFrame(df['value']).apply(pd.to_numeric,
errors='raise').diff()
                df["diff"] = pd.DataFrame(valDiffFrame["value"].astype(str))
            else:
                valDiffFrame =
pd.DataFrame(df['value'].str.split(",").tolist()).apply(pd.to_numeric,
errors='coerce').diff()
                if len(valDiffFrame.index) > 0:
                    diffColFrame = pd.DataFrame({"diff":
valDiffFrame[0].astype(str) + "," + valDiffFrame[1].astype(str) + "," +
```

```
valDiffFrame[2].astype(str)})
                    df["diff"] = diffColFrame["diff"].values
            if 'diff' in df.columns:
                df = df[~df["diff"].isin([np.nan, "nan,nan,nan","nan", 0])]
                df = df[~df["sensortime"].isin([np.nan, "nan,nan,nan","nan", 0])]
                newTotalRows += len(df.index)
                df.to_csv("./output/{}/{}/delta/{}_{}.csv".format(gender,
curslice, filename[0:filename.index(".")], sensor))
print("Reduction of {}%".format(newTotalRows/totalRows*100))
```

Appendix 5: Data Processing Pipeline – Data Homogenization Step Source Code

```
import pandas as pd
import os
gender = "female"
slicing = "snz"
filenames = []
for (dpath, dnames, fnames) in
os.walk(os.path.join(".","output",gender,slicing,"delta")):
    filenames.extend(fnames)
filenames = sorted(filenames, key = lambda filename :
int(filename[0:filename.index("_")]))
lastDay = 1
dayFrame = pd.DataFrame()
for filename in filenames:
    path = os.path.join(".", "output", gender, slicing, "delta", filename)
    fileFrame = pd.read_csv(path)
    sensor = filename[-5:-4]
    print(filename, sensor)
    if not fileFrame.empty:
        if int(sensor) == 1:
            fileFrame[['diffX','diffY','diffZ']] =
fileFrame['diff'].str.split(",", expand=True)
            fileFrame[['diffX','diffY','diffZ']] =
fileFrame[['diffX','diffY','diffZ']].apply(pd.to_numeric, errors="coerce")
            fileFrame = fileFrame.iloc[:, [1,3,5,6,7]]
        elif int(sensor) == 2:
            fileFrame[['diffX','diffY','diffZ']] =
fileFrame['diff'].str.split(",", expand=True)
            fileFrame = fileFrame._convert(numeric=True)
            fileFrame['diffZ'] = fileFrame['diffZ'].fillna(0)
            fileFrame[['diffX','diffY','diffZ']] =
fileFrame[['diffX','diffY','diffZ']].apply(pd.to_numeric, errors="coerce")
            fileFrame = fileFrame.iloc[:, [1,3,5,6,7]]
        elif int(sensor) == 3:
            fileFrame['diffX'] = fileFrame['diff']
            fileFrame.insert(len(fileFrame.columns), "diffY", 0)
            fileFrame.insert(len(fileFrame.columns), "diffZ", 0)
            fileFrame[['diffX','diffY','diffZ']] =
fileFrame[['diffX','diffY','diffZ']].apply(pd.to_numeric, errors="coerce")
            fileFrame = fileFrame.iloc[:, [1,3,5,6,7]]
        print(fileFrame.head(3))
        if filename[:filename.index("_")] == str(lastDay):
            print("Appending")
```

```
            dayFrame = pd.concat([dayFrame, fileFrame])
        else:
            print("Saving")
            if not dayFrame.empty:
                print("DAY " + str(lastDay))
                print(dayFrame.head(3))
                dayFrame.to_csv("./output/{}/{}/delta/nbprep/{}/{}.csv".format(ge
nder, slicing, gender, str(lastDay)), index=False)
                dayFrame = fileFrame
                lastDay = filename[:filename.index("_")]
```

Appendix 6: GitHub Repositories

The code for the Android application responsible for collecting data can be found at https://github.com/aelhussiny/ThesisDataCollection.

The code for the data processing pipeline, machine learning models, and result comparison script can be found at https://github.com/aelhussiny/ThesisPython.