Università degli Studi dell'Insubria

Dipartimento di Scienze Teoriche e Applicate

Enhancing Data Privacy and Security Related Process Through Machine Learning

A thesis presented by

Md. Zulfikar Alom

Submitted to the
Department of Theoretical and Applied Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

Advisor: Professor Elena Ferrari

Advisor: Professor Barbara Carminati

Jury :

Reviewers :	Dr. Marco Mesiti	- Università degli Studi di Milano
	Dr. George Pallis	- University of Cyprus
Examiners :	Professor Maria Luisa Damiani	- Università degli Studi di Milano
	Professor Sergio Mascetti	- Università degli Studi di Milano
	Dr. Pierluigi Gallo	- Università degli Studi di Palermo

defended on October 09, 2019

Abstract

Machine learning (ML) is one of the most exciting technologies being developed in the world today. Machine learning gives computers the ability to learn just like humans do. The appeal and pervasiveness of ML technology is growing. ML methods are being improved, and their ability to understand and provide answers to real problems is highly appreciated. These achievements have led to the adoption of ML in several areas, such as big data, computer vision, and medical analysis. The main advantage of ML is its prediction capability. This benefit motivated us to exploit ML in the domains of data security and data privacy.

Nowadays, individuals are becoming increasingly dependent on numerous online services to make their lives more comfortable and convenient. To offer such services, service providers collect, store, and process a massive amount of personal information about individual users. However, although individuals voluntarily provide such personal information to service providers, they often have no idea how their information is subsequently used. This may also cause serious privacy threats as users lose control over their data. To tackle this problem, and to gain more control over their data, individuals can alter their privacy settings, stating how their data can and cannot be used and managed by the service providers. Unfortunately, the average user might find it difficult to properly set up privacy settings due to lack of knowledge and subsequent lack of decision-making abilities regarding the privacy of their data. Since ML has a strong prediction ability, in this thesis, first of all, we use ML technology to try to predict the best privacy settings for users.

Another benefit of ML is that it can eliminate (or reduce) the need for human participation in some tasks. More precisely, ML can help to reduce the load of human-based annotation. For instance, the scenario of intrusion detection or a fake news detection system, where individual involvement is needed to define the rules, and human expertise is required to annotate the sample data based on those rules. Defining rules and annotating samples is an exhausting and time-consuming task. Instead, an ML model can quickly learn to label samples, which unload some tasks from humans. Since the ML approach has the potential to considerably cut down on manual efforts by humans, our second task in this thesis is to exploit ML technology to redesign security mechanisms of social media environments that rely on human participation for providing such services. In particular, we use ML to train spam filters for identifying and removing violent, insulting, aggressive, and harassing content creators (a.k.a. spammers) from a social media platform. It helps to solve violent and aggressive issues that have been growing on the social media environments. The experimental results show that our proposals are efficient and effective.

Acknowledgement

First of all, I would like to thank the Almighty Allah for blessing me to accomplish this thesis. Without His blessing, I could not able to continue this work.

I cannot obtain this Ph.D. without my excellent advisors, **Prof. Elena Ferrari** and **Prof. Barbara Carminati**. I would like to express the most sincere gratitude to my advisors. They played the key roles in my Ph.D. In addition to their academic guidance, they have shown a great deal of patience towards me and supported my research in a way that goes beyond a professional relationship. Their caring attitude has always given me the strength to continue my research endeavor.

I gratefully acknowledge the members of my Ph.D. committee for their time and valuable feedback on my thesis.

I am thankful to my Ph.D. colleagues, especially, Bikash, Stefania, Gokhan, Shah, Alberto, Christian, Tu, Deniz, Federico, Riccardo, and all my Italian friends in here, for showing me Italian hospitality and enriching my life greatly. Without them, my Ph.D. would never be such a great experience.

I would like to thank Mauro Santabarbara for his time in helping me with the computer that I used during my research. I would like to thank Roberta Viola for aiding me in processing the official paperwork during my Ph.D.

I would like to thank all the people who take participation in my experiments. I really appreciate their efforts and time for helping me to evaluate my research work.

Last but not least, I am deeply thankful to my family: my beloved parents, my sisters and their husbands, and my brother, for their unconditional love and spiritual support in all my life.

Dedication

I would like to dedicate this doctoral dissertation ...

To my mother **Anowara Begum**
To my father **Forhad Hossain**
To my sister **Farhana Parvin**
To my sister **Farzana Yasmin**
To my brother **Toriqul Islam**
To my little niece **Maisha**
To my little nephew **Fahad**

Contents

List of Figures

List of Tables

1 Introduction

Machine learning (ML) is a subset of artificial intelligence whereby computers learn data autonomously. Machine learning refers not only to the use of computers for calculations and data retrieval but also to combining those two capabilities of a computer system so that it appears to be learning and making rational decisions in accordance with previously observed data. In 1952, Arthur Samuel wrote the first ML program which learned to play checkers [1]. In the mid-1980s, several people independently discovered the back-propagation algorithm, which boosted ML technology, enabling more powerful neural networks with hidden layers to be trained [2]. In the 21st century, businesses have realized the power of ML technology and several companies have launched large ML-based projects (e.g., GoogleBrain[1], DeepFace[2], DeepMind[3]) to stay ahead of their competitors.

Today, ML technology is becoming more pervasive and appealing because it has demonstrated the ability to solve real problems. The major advantages of ML technology are its prediction capability and its ability to reduce the need for human activities to perform tasks [3]. These benefits have led to its adoption in the data security and privacy domain. Today, to make their lives more comfortable and convenient, individuals are becoming dependent on a variety of online services. The providers of these services collect, store, and process a vast amount of personal information about individuals. Clients of these providers provide personal information for a specific purpose. However, they often have no idea how their personal information will be used subsequently. This may cause serious privacy threats as it means that users lose control over their data. To address this challenge and to enhance privacy control, individuals can explicitly express their privacy preferences, stating conditions on how their data should be used and managed by the service providers. However, average users may find it difficult to effectively manage their privacy settings due to a lack of knowledge or decision-making abilities regarding data privacy [4, 5]. As ML has good prediction capabilities, in this thesis, we attempt to predict the most appropriate privacy settings for users by exploiting ML technology. Specifically, we propose a privacy preference prediction framework for a specific domain, the Internet of Things (IoT) based smart environment.

[1]https://ai.google
[2]https://deepface.ir
[3]https://deepmind.com

We note that ML technology has the potential to considerably cut down on manual efforts by humans and exploit its benefits in the cybersecurity domain to protect individuals from cyber attacks. For instance, most previous intrusion detection systems used signature-based approaches to detect attacks in a network. Today, ML can offer new network security solutions called network traffic analytics to perform an in-depth analysis of all traffic at each layer and detect attacks and anomalies. More specifically, an ML regression model can be used to predict the network packet parameters and compare them with the normal ones, and classification can be used to identify different classes of network attacks such as scanning and spoofing. Likewise, in cybersecurity, a regression model can be used for fraud detection. The features (e.g., the total amount of suspicious transactions and location of fraudulent transactions) can be used to determine the probability of fraudulent actions. Moreover, ML classification methods can be used to detect spam and fake news. For spam detection, a spam filter separates spam messages and users from normal ones. Previously, spam detection in social media environments was required human expertise to define the rules and annotate the data (e.g., mark users/messages as spam or non-spam) based on those rules. Assigning annotators to annotate the sample data is an exhausting, time-consuming, and extremely expensive task. Since ML can be used to train spam filters automatically, it brings the advantage of eliminating the need for human participation. In this thesis, we attempt to exploit ML technology to train spam filters for identifying and removing violent, insulting, aggressive, and harassing content creators (i.e., spammers) from the Twitter platform. The goal is to improve Twitter's existing spam-detection mechanisms (e.g., [6, 7, 8]).

1.1 Machine Learning in an IoT-based Smart Environment

Internet of Things (IoT) based smart environments are physical spaces, enriched with connected sensors and smart devices that offer various services (e.g., home automation, entertainment, and health monitoring) to support individuals in their daily activities. In order to provide personalized services based on individual habits, smart environments usually collect, store, and process vast amounts of personal information on individuals. For instance, IoT-based home automation systems monitor users' behavior using motion sensors, Wi-Fi signals, or facial recognition technology to identify their presence in rooms and automatically adjust the temperature or lighting. In general, service providers collect these data in accordance with their privacy policies. A privacy policy defines how a service provider collects and manages customers' personal information, the purpose for which that information may be used and the length of time it can be retained. To have more control over how their personal data are used, individuals can specify their privacy preferences, stating how their data should be used and managed by the service provider. In a smart environment, privacy checking is handled by hard matching users' privacy preferences against service providers' privacy policies and denying all services whose privacy policies do not fully match an individual's privacy preferences. This means that a user can only access a service if the provider's policies fully match all his or her privacy preferences. Therefore, if individuals do not choose suitable privacy settings, they may not be able to access many services. Due to a lack of knowledge and decision-making abilities with regard to privacy management, many users experience difficulty when selecting privacy preferences. For instance, a user may choose a privacy preference whereby he or she agrees to share his or her health-related data (e.g., blood-pressure and heart-beat data) acquired through a smart device with a healthcare service provider if the data are stored for only 100 days. A healthcare service provider whose privacy policy states that it will store data for 110 days clearly does not satisfy the user's privacy preference, even though it is close to satisfying them. In a real-life scenario, if a user has a health condition, she or he would likely not care about data retention for a further 10 days and would likely make an exception to her or his privacy preference in order to access the service. However, a hard privacy matching mechanism would deny the user access to the service without considering the possible benefits for the user. Therefore, we examine how privacy checking in IoT-based environments can be made more flexible. First, we propose a soft privacy matching mechanism, which can relax, in a controlled way, some of the conditions of users' privacy preferences to match service providers' privacy policies. To achieve this aim, we exploit ML algorithms to build a classifier that can make decisions on future service requests by learning which privacy preference components a user is willing to relax, as well as the relaxation range (i.e., by how much the preferences may be relaxed). We then extend our approach to take into account individuals' contextual information. Contextual information refers to any piece of data on an individual that can be used to define his or her current situation. We have witnessed that individual's privacy preferences may vary based on his or her contextual information [9, 10, 11]. For instance, a user may feel comfortable accessing entertainment services while at home but not during office hours when he or she is at work. To achieve more fine-grained control, a user can set different privacy preferences for different contexts. However, since users' contexts frequently change, this might be an extremely complex and time-consuming task. To address this issue, this thesis also proposes a context-based privacy management service that uses ML technology to help users manage their privacy preferences for different contexts.

1.2 Machine Learning in a Social Media Environment

In the last few years, online social networks (OSNs), such as Twitter, Facebook, and LinkedIn, have become extremely popular communication tools [12, 13]. Users spend a substantial amount of time on OSNs developing friendships with people they know or who have similar interests and sharing messages about real-life issues including news, events, social problems, and political crises. Moreover, social networks give businesses an unprecedented opportunity to connect with customers. However, due to the significant popularity of OSNs, boosted by the proliferation of social networks, OSNs also attract the interest of cybercriminals (i.e., spammers) [14, 15]. Spammers exploit the implicit trust between users to achieve their malicious aims. For example, on Twitter, spammers create malicious links within tweets, spread violent, aggressive and fake news, and send insulting and harassing messages to legitimate users. To deal with this problem, rule-based approaches are used to detect spammers. Generally, a rule-based approach means that humans define rules, and, based on those rules, annotate the sample data [16, 17]. However, defining rules and annotating samples is an exhausting and time-consuming task. Since ML has the ability to reduce human participation in such tasks, it can be used to train spam filters to annotate samples. In this thesis, we exploit ML technology to detect spammers on Twitter using several newly proposed features (e.g., graph-based features) that are more effective and robust than existing features (e.g., number of followings/followers).

1.3 Main Contributions

The main contributions of this thesis lie in its exploitation of ML techniques (1) to improve data privacy in IoT-based smart environments and (2) to enhancing data security by redesigning the security mechanism of a social media environment.

In summary, this thesis offers the following main research contributions:

- We propose a *soft privacy matching mechanism* that enables fine-grained learning of the extent to which individuals are willing to relax their privacy preferences. More specifically, we demonstrate that the proposed privacy-checking mechanism is able to relax, in a controlled way, some conditions of users' privacy preferences in order to match with service providers' privacy policies.

- As an extension of the above-mentioned mechanism, we propose an *ML-based privacy preference management service* that helps individuals to manage their privacy preferences in different contexts. More specifically, we design a framework that infers individuals' privacy preferences based on their contextual information.

- This thesis also focuses on redesigning security mechanisms in social media environments that rely on human participation to provide services. To this end, we exploit ML to investigate the behavior of spammers on Twitter with the goal of improving existing spam-detection mechanisms. To detect spam accounts on Twitter, we design a set of novel graph-based and content-based features that are more effective and robust than existing features. The results show that the proposed set of features offers better performance than existing state of the art approaches.

1.4 Thesis Organization

The dissertation is organized into six chapters, which are briefly described below.

Chapter 1: Introduction

In this chapter, we mainly discuss the motivation for this dissertation and its main contributions.

Chapter 2: Literature Review

We review the literature on data security and privacy issues in different domains. More particularly, we discuss previous work on privacy preference settings in IoT-based smart environments. We then summarize the existing work on exploiting users' contextual information to build privacy preference models to protect personal data in different contexts. The chapter closes with a discussion of related work that applied ML technology to redesign the security mechanism of social media environments.

Chapter 3: ML-based Privacy Preferences Adaptation

In this chapter, we describe the proposed *soft privacy matching mechanism*, which can relax, in a controlled way, some of the conditions of users' privacy preferences to match with service providers' privacy policies. We exploit ML algorithms to build a classifier that can make decisions on future service requests by learning which privacy preference components a user is willing to relax and the extent to which he or she is willing to relax them. We then conduct experiments to determine which learning approach provides better accuracy in IoT-based smart environment.

Chapter 4: ML-based Privacy Preferences Suggestions

In this chapter, we present the proposed *ML-based privacy preferences management service*, which helps users to manage their privacy preference settings in different contexts. More precisely, we focus on users' contextual information and define a learning approach exploiting contextual features to learn users' privacy preferences.

Chapter 5: ML-based Spam Accounts Detection on Twitter

In this chapter, we exploit ML to redesign the security mechanism of the social media environment and make it a more secure place for individuals. To this end, we investigate the behavior of spammers on Twitter with the goal of improving existing spam-detection mechanisms. We present a method of classifying Twitter users based on several new features and show that the proposed technique can be more effective and robust than existing spam-detection methods.

Chapter 6: Conclusion and Future work

This chapter summarizes the thesis, discussing the results that we have obtained, and outlining plans for further developments along with new research objectives.

1.5 Related Publications

The research activities described in this thesis have resulted in the following publications:

- Md Zulfikar Alom, Barbara Carminati, Elena Ferrari, "Adapting Users' Privacy Preferences in Smart Environments" *2019 IEEE International Congress on Internet of Things (ICIOT, 2019)*, pp. 165-172, Milan, Italy.

- Md Zulfikar Alom, Barbara Carminati, Elena Ferrari, "Helping Users Managing Context-based Privacy Preferences" *2019 IEEE International Conference on Services Computing (SCC, 2019)*, pp. 100-107, Milan, Italy.

- Md Zulfikar Alom, Barbara Carminati, Elena Ferrari, "Detecting Spam Accounts on Twitter" *2018 IEEE/ACM International Conference on Advances in Social*

Networks Analysis and Mining (ASONAM, 2018), pp. 1191-1198, Barcelona, Spain.

Literature Review 2

In this chapter, we review the proposals dealing with data privacy and security issues in different domains. To this end, in Section 2.1, we discuss the related works that exploited ML strategies to learn users' privacy settings in the different environments. Afterwards, in Section 2.2, we review previous works that applied ML technology to redesign the security mechanism of social media environments.

2.1 ML-based privacy settings mechanisms

In the last few years, many studies have been devoted to improve user privacy settings by recommending privacy choices to the users. For instance, Lee et al. [18] proposed the concept of intelligent software that helps users to make better privacy decisions in the IoT-based environments. To this end, the authors adopted ML tools and performed clustering analysis on the collected preferences to understand users' privacy concerns towards IoT applications and services. Recently, Nakamura et al. [19] proposed a ML approach to provide users personalized default privacy settings for online services. The proposed approach combines prediction and clustering techniques for modeling the privacy profile associated with users' privacy preferences. The authors asked a set of 80 questions to each individual user at the time of registration. Similarly, Singh et al. [20] proposed a suite of semi-supervised approach in order to learn the privacy aptitudes of Personal Data Storage owners. The learned models are then used to answer third party access requests. The authors showed that their approach provides a better accuracy than [19] with the same training set.

Substantial research efforts have been also made to suggest privacy preferences to the users of social networks. Although almost all social network platforms have a privacy setting page to allow users to set up their privacy preferences, most of the users are facing many problems in the privacy setting specification task due to its complexity and lack of enough privacy knowledge [21]. Thus, researchers have investigated solutions that automatically configure user privacy settings with minimal effort. For instance, Sadeh et al. [22] proposed an automated mechanism for mobile social networking applications for making privacy decisions on behalf of the users. The authors exploited supervised learning (i.e., Random Forest) and made a comparison between the user-defined sharing policies and ML-based ones.

The authors claimed that the machine-generated policies have better accuracy than the user-defined ones. Likewise, Fang et al. [23] proposed a model that infers access control policies for personal information sharing in online social networking services. They also used a supervised ML approach to learn users' privacy preferences by iteratively asking them questions regarding their sharing activities with friends. The authors used personal profiles information (e.g., age, gender) with feedback of the users as a training dataset and they trained the personalized ML models. Bilogrevic et al. [24] proposed a ML privacy-preserving information sharing model for mobile social networks, called SPISM, that semi-automatically decides whether or not to share personal information and at what level of granularity. The authors used a supervised ML-based logistic classifier to predict users' privacy decisions.

Moreover, a number of studies have been carried out to understand user's privacy preferences by taking into account user's contextual data. For example, Wijesekera et al. [25] proposed a novel privacy management system that relies on user's contextual information, to improve user privacy decision making capability in mobile platforms. The author implemented model uses four different contextual features for making a decision on a resource request, namely, the name of the app requesting the permission, the app in the foreground at the time of the request (if different than the app making the request), the requested permission type (e.g., Location, Camera, Contacts), and the visibility of the app making the request. Similarly, Liu et al. [26] proposed a personalized privacy assistant that pro-actively produces permission settings for Android applications on behalf of the users. The authors developed ML classifiers to predict users' decisions for each permission request by using their privacy profiles. Smith et al. [27] presented different solutions that enable people to share contextual information in mobile networks, whereas, in [28], authors observed that people's willingness for sharing information are impacted of various factors. More particularly, in [28], authors have done the study of 42 participants, who self-report aspects of their relationships with 70 of their friends, including frequency of collocation and communication, closeness, and social group. The study shows that (a) self-reported closeness is the strongest indicator of willingness to share; (b) individuals are more likely to share in scenarios with common information (e.g., we are within one mile of each other) than other kinds of scenarios (e.g., my location wherever I am); and (c) frequency of communication predicts both closeness and willingness to share better than frequency of collocation. In [29] authors presented a framework for automatic estimation of privacy risk of data based on the sharing context. More precisely, the authors used *Item Response Theory (IRT)* on top of the crowdsourced data to determine the sensitivity of items and diverse attitudes of users towards privacy. To this end, firstly, they determine the feasibility of IRT for the cloud scenario by asking workers feedback on *Amazon mTurk* on various sharing scenarios. It obtains a good fit of the responses with the theory and thus shows that IRT, a well-known psychometric model for educational purposes, can be applied to the cloud scenario. After that, the authors present a lightweight mechanism such that users can crowdsource their sharing contexts with the server and determine the risk of sharing a particular data item(s) privately. Finally, they use the *Enron* dataset to simulate their conceptual framework and also provide experimental results using synthetic data. The result shows that the proposed scheme converges quickly and provides accurate privacy risk scores under varying conditions.

Liang et al. [30] developed a learning approach that recommends context-aware app by utilizing a tensor-based framework so as to effectively integrate user's preferences, app category information and multi-view features. More particularly, the authors used a multi-dimensional structure to capture the hidden relationships between multiple app categories with multi-view features. They developed an efficient factorization method which applies Tucker decomposition to learn the full-order interactions within multiple categories and features. Furthermore, authors employ a group *1-norm regularization* to learn the group-wise feature importance of each view with respect to each app category. In the experiments, they have demonstrated the effectiveness of the proposed method by using two real-world mobile app datasets. [31] proposed user's location sharing privacy preferences model by considering contextual information. More particularly, authors have investigated the users' location sharing privacy preferences with three groups of people (i.e., Family, Friend, and Colleague) in different contexts, including check-in time, companion, and emotion. The study reveals that location sharing behaviors are highly dynamic, context-aware, audience-aware, and personal. In particular, emotion and companion are good contextual predictors of privacy preferences. It also shows that there are strong similarities or correlations among contexts and groups. Furthermore, the work shows that despite the dynamic and context-dependent nature of location sharing, it is still possible to predict a user's in-situ sharing preference in various contexts. More specifically, it explores whether it is possible to give users a personalized recommendation of the sharing setting they are most likely to prefer based on context similarity, group correlation, and collective check-in preference. The results demonstrated that personalized recommendations that could be helpful to reduce both user's burden and privacy risk. Yuan et al. [32] proposed a privacy-aware model for photo sharing based on ML by exploiting contextual information. The proposed model utilizes image semantics and requester contextual information to decide whether or not to share a particular picture with a specific requester in a certain context. Likewise, [33] proposed a privacy preference framework that semi-automatically predicts sharing decision, based on personal and contextual features. More precisely, the author presents a novel information-sharing system that decides (semi) automatically, based on personal and contextual features, whether to share information with others and at what granularity, whenever it is requested. The author uses of (active) machine learning techniques, including cost-sensitive multi-class classifiers based on support vector machines (SVM). Based on a personalized survey about information sharing, which involves 70 participants, results provide insight into the most influential features behind a sharing decision, the reasons users share different types of information and their confidence in such decisions. In [34], authors presented a privacy preference model for helping users to manage their privacy in context-aware systems in term of sharing location on the basis of the general user population using crowd-sourcing architecture. Bigwood et al. [35] have evaluated different ML algorithms so as to build information sharing models. More particularly, the author uses a dataset collected from a location-sharing user study (i.e., 80 participants) to investigate whether users' willingness to share their locations can be predicted. They find that while default settings match actual users' preferences only 68% of the time, machine learning classifiers can predict up to 85% of users' preferences. Moreover, the results demonstrate that using these predictions instead of default settings would reduce the over-exposed location information by 40%. Likewise, Schlegel et al. [36] proposed a mechanism that summaries the number of requests made by the requester for sharing his/her location. The authors propose an intuitive mechanism for summarizing and controlling a user's exposure on smartphone-based platforms. The approach uses the visual metaphor of eyes appearing and growing in size on the home screen, the rate at which these eyes grow depends on the number of accesses granted for a user's location, and the type of person (e.g., family vs friend) making these accesses. The authors also claim that their approach gives users an accurate and ambient sense of their exposure and helps them take actions to limit their exposure, all without explicitly identifying the social contacts making requests. Through two user studies (i.e., 43 and 41 participants), the result shows that the proposed interface is indeed effective at summarizing complex exposure information and provides comparable information to a more cumbersome interface presenting more detailed information.

However, our approach differs from all the above mentioned proposals in that none of the above works addressed the issue of learning how much a privacy preference can be relaxed in order to access a service. To the best of our knowledge, we are the first showing that ML tools can be effectively used to learn users privacy relaxing aptitudes toward satisfying service providers' policies.

2.2 ML-based security mechanisms

ML technology has been significantly improved in recent years. It has shown the ability to reduce human-participation in some tasks. For this benefits, many researchers have been used ML technologies to redesign the security mechanisms for online social networks (OSNs), where individual was required to define rules, and based on those rules, human expertise annotated the sample data.

However, the overwhelming popularity of OSNs, it attracts the interest of cyber-criminals, therefore, a significant number of studies have been focused on security mechanisms of OSNs to protect individuals from cyber attacks. For instance, Gao et al. [37] collected two different data sets for detecting spammers on OSNs. The authors collected 187 million Facebook wall posts generated by roughly 3.5 million users and Twitter dataset that contains over 17 million public tweets. Authors generated social graph and exposed spammers according to the users' social degree, interaction history, and cluster size. Ala'M et al. [7] used four ML classifiers and some of the most common account-based (e.g., default image, following to followers ratio) and text-based (e.g., number of tweets per day, suspicious words, repeated words, text to links ratio, comments ratio, tweet time pattern, different description from tweets, different following interest from tweets) features for detecting spammers on Twitter. Similarly, Ameen et al. [6] used four ML classification algorithms and 13 text-based features including the age of an account (days), number of re-tweets, number of followers, number of following/friends, number of favorites, number of user mentions, number of lists, number of tweets, number of hash tags, number of URLs, number of characters, number of digits, time of sending the most recent tweet. Benevento et al. [14] compared two approaches for detecting spam profiles and spam tweets. Initially, they built their model to identify spam profiles based on account-based features. Then, they used both account-based and text-based features to classify the tweets into spam and non-spam categories. Lee et al. [38] used 10 ML classifiers and two different data sets for detecting spammers on Twitter. Chen et al. [39] collected a large dataset of over more than 600 million public tweets. Then, they extracted 12 light-weight features[1] and used them into six ML algorithms to classify the tweets into spam and non-spam categories.

Some hybrid approaches, for example, Sing et al. [8] used three feature sets, namely trust score, content-based and user-based features and four ML classification algorithms for classifying the users into spammers and non-spammers. Similarly, Wang et al. [40] focused on spam tweets detection rather than detecting spam accounts. They used two hand-labeled data sets (i.e., Social honeypot and 1KS-10KN) and four feature sets, i.e., user-based, content-based, n-gram, and sentiment features. Fazil et al. [13] also proposed an hybrid framework, exploiting users' meta-data, interaction-based, community-based and tweet text-based features to detect spammers on Twitter. They used 19 features and three popular ML algorithms for classifying users into spammers and non-spammers. Similarly, Hai Wang et al. [41] used a social graph model, by leveraging on the followings and followers relationships. They extracted from Twitter around $25K$ users, and 20 recent tweets for each user, along with $49M$ friends/followers relationships. To assess their detection method, they used four different ML classifiers to classify users into spammers and

[1]A feature that is relatively easy to extract from the user's profile on Twitter is called a light-weight feature.

legitimate users. However, more information about the different machine learning classifiers/algorithms that have been widely used to detect spammers in online social networks is given in Section 2.2.1.

One key issue is that spammers normally evade user-based detection features by obtaining more followers. Since, spammers repeat posting with malicious URLs for attracting legitimate users to visit particular sites, so their tweets show strong homogeneous characteristics. For this reason, many researchers used tweet content-based features (e.g., tweet similarity, duplicate tweet count, etc.) to detect spam accounts. Although, using simple techniques, such as posting heterogeneous tweets and act as normal users, spammers can avoid these detection techniques. On the other hand, we propose graph-based features, which are based on social relationships in the real-world scenario, for instance, legitimate users usually follow accounts whose owners are their close friends, relatives or family members, as such these accounts are likely to have a relationship together and build a triangle in their networks. Likewise, they follow each other and increase number of bi-directional links between them. However, spammers can change their tweets by changing words but they cannot change the URLs since they want users to visit a particular website. Moreover, spammers intend to post more tweets with URLs than legitimate users, so we have considered URLs to tweet ratio and average tweet per day features. We also proposed average likes per tweet, spammers usually posted more tweets but they could not get response (e.g., likes) from users, so this feature may be useful for the detection process. More information about the proposed features are given in the section 5.1.

As summary, Table 2.1 lists the used features, classifiers, data sets and the results of the approaches described so far.

2.2.1 Spam detection ML techniques

There are many ML classification algorithms that have been widely used to detect spam users in online social networks. Here, we have discussed some of them, namely, Naive Bayesian, k-NN, Decision Tree, Random Forest, Logistic Regression, SVM, and XGBoost, briefly described in what follows [42].

Naive Bayesian (NB). Naive Bayesian classifier is based on the probability theory (i.e., Bayes theorem) [42]. This model is widely used because it gives good performance and requires less computational time for training the model. The main assumption of this algorithm is that the features of a dataset are independent, it means that the probability of one attribute does not affect the probability of the other. However, let us consider C represents the class (i.e., spammer or non-spammer) and D defines a Twitter user's profile, which may belong to class spammer or non-spammer. The Naive Bayesian classifier, which is based on Bayes theorem, can be described as follows [43]: $P(C|D) = \frac{P(D|C)*P(C)}{P(D)}$, where, $P(C)$ and $P(D)$ are the probability of C and D, respectively. These are called the prior probability and their values can be computed from the training data. $P(D|C)$ is called the conditional probability, which means the probability of D given that C happens. Likewise, $P(C|D)$ means the probability of C given that D happens, it is called the posterior probability [42, 43].

k-Nearest Neighbor (k-NN). k-Nearest Neighbor classifier is a very simple supervised learning algorithm, which stores all available instances and classifies new instances based on the similarity measure (e.g., distance functions). The instances are classified by majority vote of their neighbors, the instances being assigned to the class which is the most common amongst its k nearest neighbors. If k = 1, then the instance is simply assigned to the class of its nearest neighbor [44].

Decision Tree (DT). Decision Tree is an extension of ID3 algorithm which uses Entropy and Information Gain to construct a decision tree [45]. It is a very powerful and widely used classifier because it is very simple and gives good results by using less memory space than other algorithms. The major disadvantage of this algorithm is that it takes long time for training the classification model. Generally, Decision tree classification method is divided into two phases: tree building and tree pruning [43]. In the tree building phase, it recursively partitions a dataset using depth-first greedy approach or breadth-first approach until all the data items belong to the same class label. In the tree pruning phase, it works for improving the classification accuracy by minimizing over-fitting problem. The Decision tree consists of root, internal and leaf nodes, where the root node is the topmost node of decision tree, internal node corresponds to a test condition on an attribute, branch corresponds to results of the test conditions, and a leaf node corresponds to a class label.

Random Forest (RF). Random Forest is a very flexible and easy to use machine learning classifier that consists of a collection of tree structured classifiers [46]. It randomly selects the features to construct a collection of decision trees. It is one of the most widely used algorithms, because of its simplicity and it can be used for both classification and regression tasks.

Logistic Regression (LR). Logistic Regression is an extension of simple regression method. In logistic regression, the output of linear regression is passed through the activation function. Softmax function can be used as an activation function, it is a very popular function to calculate the probabilities of the events. It can be defined as follows: $\sigma(z)_j = \frac{\exp^{z_j}}{\sum_{k=1}^{K} \exp^{z_k}} \; for \; j = 1, 2, .., K$, where z is a vector of the inputs to the output layer (since we have 2 output class i.e., spam and non-spam, so there are 2 elements in z) and j is the index of output units. The output values of this function is always in the range $[0,1]$, and the sum of the output values is equal to 1. Generally, LR is intended for binary classification, so that we classify input to class 1 when the output of this function is closed to 1 (i.e., output > 0.5) and classify to class 2 when the output is closed to 0 (i.e., output ≤ 0.5) [47].

Support Vector Machine (SVM). Support Vector Machine is probably one of the most popular machine learning algorithm. It performs classification by finding the hyperplane that maximizes the margin between the two classes. The vectors which define the hyperplane are called the support vectors. An ideal SVM produces a hyperplane which completely separates the vectors into two non-overlapping classes. However, perfect separation may not be possible, so in this case, SVM finds the hyperplane which maximizes the margin and minimizes the mis-classifications [48].

eXtreme Gradient Boosting (XGBoost). XGBoost has become a widely used and popular machine learning algorithm. It is an implementation of gradient boosted decision trees. More particularly, it is an ensemble method which sequentially adds predictors and corrects the previous models. However, instead of assigning weights to the classifiers after every iteration, this method fits the new model to new residuals of the previous prediction and then minimizes the loss [49].

2.3 Thesis contributions with respect to the literature review

In this section, we discuss how we address the limits in existing privacy preference and security management frameworks. For this purpose, we do not simply study the existing techniques rather we propose different approaches for enhancing users data privacy and security. The majority of existing works require users to manually set up their privacy preferences by selecting yes or no options for a set of questions. However, several studies have shown that with this approach, the average user might find it difficult to properly set up privacy settings [21, 50]. To resolve this issue, we propose the privacy preference frameworks that exploit machine learning tools for learning user privacy preferences according to users' feedback so as to set up privacy preferences automatically. Until now, some recent research work also exploits machine learning tools to implement privacy preference frameworks [19, 22, 33]. The main intention of these studies is to design a framework that can automatically set up user privacy preferences with minimal user efforts. Although they did not investigate how to relax some conditions of users privacy preferences

Ref. ID	Features	Classifiers	Dataset	Results
[14]	User-based Tweet content-based	SVM	1065 users' profiles 355 spammers 710 legitimate users	Accuracy with only user attributes : 84.5% both user and content attributes: 87.6%
[38]	User-based Tweet content-based	Decorate, LogitBoost HyperPipes, Bagging RandomSubSpace, BFTree, FT SimpleLogistic, LibSVM Classification Via Regression	500 users' profiles 168 spammers 332 legitimate users	Decorate classifier gives the best accuracy: 88.98%
[41]	Graph-based Tweet content-based	DT, Neural Network SVM, NB	500 users' profiles 3% spam users' account	NB gives the best performance Precision: 91.7% Recall: 91.7% F1-score: 91.7%
[40]	User-based Tweet content-based n-gram Sentiment	NB, k-NN, SVM DT, RF	2 hand labeled data sets Social Honeypot Dataset 1KS-10KN Dataset	RF gives the best performance with F1-score: 94%
[7]	Tweet content-based User-based	DT Multilayer Perception k-NN, NB	82 users' profiles	NB gives the highest accuracy: 95.7%
[6]	User-based Tweet content-based	NB, J48 RF, IBK	1183 users' profiles 355 spammers 828 legitimate users	RF gives the best performance with accuracy: 92.95%
[8]	Trust-based Tweet content-based User-based	Bayes Net, Logistic, J48 RF, AdaBoostM1	19581 users' profiles 11059 spammers 8522 legitimate users	RF gives the best performance with accuracy: 92.1%
[13]	Metadata-based Tweet content-based Interaction-based Community-based	Random Forest Decision Tree Bayesian Network	2000 users' profiles 1000 spammers 1000 legitimate users	RF gives the best performance with F1-score: 97.9%
[37]	Tweet content-based Graph-based	C4.5 Decision Tree	2 labeled data sets Facebook Dataset consists of 187 million wall posts Twitter Dataset consists of 17 million tweets	Facebook Dataset True Positive: 80.9% Twitter Dataset True Positive: 69.8%
[39]	Tweet content-based Account-based	Random Forest, C4.5 Decision Tree, Bayes Network, kNN Naive Bayes, SVM	6.5 million spam tweets 6 million non-spam tweets	RF gives the best performance with F1-score: 93.6%

Table 2.1: Summary of the reviewed Twitter spam detection papers

to access services. We recall that a user can only access a service if the provider's policies fully match all his or her privacy preferences. Considering such issue the proposed approach tries to learn how much a user's privacy preference can be relaxed to access a service, by exploiting different machine learning tools. Moreover, to improve user security in social networks, we also exploit ML tools to train the spam filters automatically for identifying and removing spam accounts from social media environments. In the previous work, for spamming detection individuals involvement was needed to define some rules and based on those rules human expertise annotated the sample data (i.e., spam or non-spam). However, defining rules and annotating samples is an exhausting and time-consuming task. The ML technology bears the potential to considerably cut down on manual efforts and unloads tasks from humans. This advantage led us to exploit ML technology to solve the problem of human-based spam detection on social networks. So far, some recent research work also exploits ML technology to detect spammers on social networks [6, 7, 14]. The key issue of these methods is that they have used user-based and text-based features to detect spammers on social networks. However, spammers normally evade such kind of feature-based (i.e., user-based and text-based) detection methods. To solve this limitation, we redesign an ML-based security mechanism to detect spammers on social networks by using several newly proposed features (e.g., graph-based features) that are more effective and robust than existing user-based and text-based features.

2.4 Chapter summary

In this chapter, we have focused on the solutions that can provide privacy preference frameworks to the users for setting privacy preferences according to their privacy requirements. For privacy settings, different frameworks have been discussed. Most of the proposals have been discussed based on ML technology. Moreover, we have also discussed the usage of ML approach to social media environment so as to eliminate human-based spam detection techniques. In this thesis, we propose a new framework based on ML approach so as to help the users for configuring their privacy settings. We also propose an ML-based approach to redesign the security mechanisms of social network platforms.

3 ML-based Privacy Preferences Adaptation

In this chapter, we propose an automatic flexible privacy matching model that helps users in protecting their personal data as well as accessing more number of services, by exploiting ML technologies. More precisely, we propose a soft privacy matching mechanism, able to relax, in a controlled way, some of the conditions of users' privacy preferences such to match with service providers' privacy policies. The key idea is to adopt ML to infer, for each user, which component of a privacy preference (e.g., purpose, data, retention, and recipient) she/he is willing to relax and how much (i.e., relaxation range). At this purpose, we exploit supervised ML algorithms over a labeled training dataset, consisting of provider service requests and related policies, and user decisions on joining/denying these services. The learning algorithm takes as features the decision and the distance values between each component of user's privacy preferences and provider' policy, and learn a classifier able to state when privacy preferences can be relaxed.

To show the effectiveness of the proposed approach, we have carried out several experiments. In particular, we have experimentally tested different supervised ML algorithms [42], namely, Support Vector Machine (SVM), Naive Bayesian (NB), and Random Forest (RF) to see which one gives better performance in the considered scenario. At this purpose, we developed a web application through which evaluators were able to: (1) label a training dataset of service joining requests, and (2) give their feedback on service joining decisions (i.e., accept/reject) suggested by the system for new service joining requests. We have tested the proposed approach by using two different groups of evaluators: university-based evaluators (i.e., 25 CS students from two universities in two different geographical areas); crowd-based evaluators (i.e., 160 workers from a crowd-sourcing platform). The obtained results show that about 96.5% of university-based evaluators and 83.4% of crowd-sourcing based evaluators are satisfied with the decisions taken by the system.

The rest of this chapter is organized as follows. Section 3.1 provides the idea of smart environment modeling, whereas Section 3.2 explains the building blocks of the proposed approach. Section 3.3 illustrates the experimental results.

3.1 Smart environment modeling

We model a smart environment as a set of N services, tailored for smart devices, and provided by different service providers, where each service has its own privacy policy. A service provider's privacy policy is formalized as follows.

Definition 1 **(Privacy Policy)**. *A privacy policy of a service provider S, denoted as S_{pp}, is a tuple (sp, p, d, ret, rec), where sp is the service provider name, p is the access purpose according to which sp wants to collect users' data, d is the data to which the policy applies, ret indicates how long sp will store the data, whereas rec specifies whether sp wants to share d with third parties.*

Example 1 *Let us consider a service provider, named, $My\_Diagnostic$, and suppose that, according to its privacy policy, it collects users' $blood\_pressure$, $weight$, $height$, $disease\_name$ data for $diagnosis$ purpose, stores this data for 120 $days$ and share it with $third\_party$. According to Definition 1, the service provider's privacy policy can be specified as follows:*

$$S_{pp} = \begin{cases} sp = My\_Diagnostic \\ p = diagnosis \\ d = blood\_pressure, weight, height, disease\_name \\ ret = 120\ days \\ rec = yes \end{cases}$$

Likewise, to provide control of personal data, a user can set up his/her privacy preferences, which state the conditions according to which his/her data has to be used and managed [51, 52]. We formally define a user's privacy preference as follows.

Definition 2 **(Privacy Preference)**. *A privacy preference for a user U, denoted as U_{pp}, is a tuple (sp, p, d, ret, rec), where, sp is the service provider name to which the preference applies, p denotes the purpose for which sp is allowed to collect the data denoted by d, ret specifies how long the service provider can store the data, whereas rec indicates whether additional third party entities can use the data.*

Example 2 *Let us consider a user* U *that wishes to release his/her $blood\_pressure$, $weight$, $heart\_beat$ data to $My\_Diagnostic$ only for $treatment$ purpose. Moreover, (s)he wants that the data will not be retained more than 100 $days$, allowing*

$My\_Diagnostic$ to share it with $third\_party$. These privacy requirements can be encoded through the following privacy preference:

$$U_{pp} = \begin{cases} sp = My\_Diagnostic \\ p = treatment \\ d = blood\_pressure, weight, heart\_beat \\ ret = 100 \ days \\ rec = yes \end{cases}$$

When a user U enters into a smart environment, service providers send joining requests to U's devices (e.g., smart watch, fitness tracker, etc). These requests consist of information about provided service as well as the related privacy policy.[1] According to a traditional privacy matching, U's software agent performs an hard matching of U's privacy preference against the providers' privacy policy, by denying those services that do not fully match. This conventional binary decision *(i.e., yes or no)* method is very restrictive because users cannot access services even if their preferences are almost satisfied by service providers' policies. To overcome this limitation, we propose a more *flexible privacy matching mechanism*, able to dynamically relax some conditions of users' privacy preferences in order to match providers' privacy policies. The design of such a mechanism requires to address some research challenges. The first is that, as a matter of fact, different individuals have different privacy aptitudes, so, the flexible privacy matching has to take into account user's perspective wrt the release of personal information. This has to be considered in deciding which components of a privacy preference (e.g., purpose, data, retention) should be relaxed. Depending on the considered user, different components might have a difference relevance for the decision. Additionally, user's perspective is also relevant to understand how much the matching mechanism can relax the selected component.

To cope with these subjective aspects, we design a mechanism to learn: (1) which components will be relaxed, and (2) how much they could be relaxed. To this purpose, we first need a metric able to measure the distance between user's preference and provider's policy components (see Section 3.2.1). Then, we exploit ML algorithms, and we take into account user feedback to create a training dataset on which learning algorithms build the classifiers. The learned classifiers are then used to answer future service requests. The following section explains the overall architecture of the proposed approach and how it works.

[1]For the sake of simplicity, in what follows, we assume that a single privacy policy is in the joining request. However, multiple privacy policies can be easily supported as well.

3.2 Flexible privacy matching

As introduced in the previous section, our learning strategy has two main goals: (i) understanding which conditions in a user privacy preference can be relaxed and how much, and (ii) building a classifier to predict future decisions on new service joining requests. These steps will be described in the following by starting from the metrics to quantify the distance between a user privacy preference and a service privacy policy.

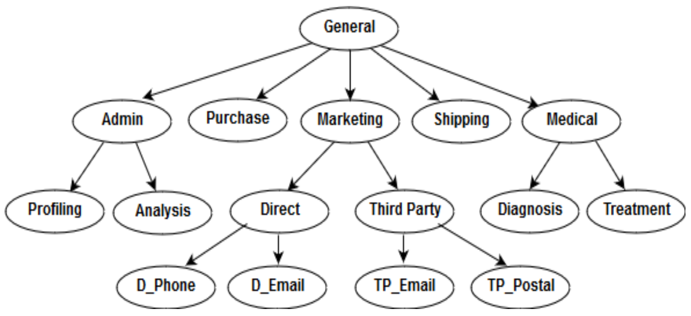


Figure 3.1: An example of purpose tree

3.2.1 Metrics

As mentioned earlier, when a user U enters into a smart environment, service providers send joining requests to his/her devices. When a user's privacy preference fully satisfies the service providers' privacy policy, the user joins the service. In contrast, if the user's privacy preference does not fully satisfy the service privacy policy, we measure their distance to determine how far the user's preference is to satisfy the provider's privacy policy. To do so, we measure the distance of each preference component, as explained in what follows.

Purpose distance: a purpose is the reason for which a service provider wants to collect users' personal data. Service providers always inform the data owner about their intended purposes via their privacy policies [53]. For instance, the following privacy policy: "*The service collects blood_pressure data for treatment purpose*", means that the service provider collects blood pressure data only for treatment purpose, and not for other purposes (e.g., research, and so on).

As usual, we consider purposes organized into a *purpose tree* [53], called T_p (see an Example in Figure 3.1). Let p be a purpose in the purpose tree T_p, and let $\downarrow p$ represents all descendants of p including p itself. Let p_i be a purpose in the purpose tree, we say that p_i matches p, if $p_i \in \downarrow p$. For instance, according to Figure 3.1, if a user allows data sharing for medical purpose, this means that the user does not have any problem to share data with service providers if their intended purpose is treatment or diagnosis. In contrast, if the service provider's purposes in S_{pp} does not match with users' intended purpose in U_{pp}, then we measure the distance between them, by leveraging on the the *Wu and Palmer similarity* [54] metric.[2]

Definition 3 *(***Purpose distance***). Given two purposes $\overline{p}$ and p. Let ccn be the closest common ancestor of the purpose $\overline{p}$ and p in the purpose tree, $depth(ccn)$ be the number of edges from the root to ccn, $dis(\overline{p})$ and $dis(p)$ be the distance of the purpose $\overline{p}$ and p from ccn, respectively. Purpose distance is defined as follows:*

$$\Delta p(\overline{p}, p) = 1 - \frac{2 * depth(ccn)}{dis(\overline{p}) + dis(p) + 2 * depth(ccn)}$$

Example 3 *Let us consider the user privacy preference and service provider's privacy policy presented in Examples 1 and 2, where the purpose in U_{pp} is treatment, whereas, provider's purpose is diagnosis. According to the purpose tree in Figure 3.1, the ccn between* treatment *and* diagnosis *is* medical*. Moreover, the distances between* medical *and* treatment*, and* medical *and* diagnosis *are both 1. Therefore:*

$$\begin{aligned}\Delta p(treatment, diagnosis) &= 1 - \frac{(2 * 1)}{(1 + 1 + 2 * 1)} = 1 - \frac{2}{4} \\ &= 1 - 0.50 \\ &= 0.50\end{aligned}$$

Data distance: the data component refers to the data objects for which users set up their privacy preferences. It may contain different types of data, such as personal data (e.g., name, address, gender, phone number, email, date of birth, and so on.), credit card data (e.g., card number, card types, expiry date, and so on.), healthcare data (e.g., blood pressure, heart beat, weight, etc.). However, since both user's privacy preference data field and service provider's privacy policy data field are defined as a set of data items, to measure the distance between them we exploit the *Jaccard coefficient* [55].

Definition 4 *(***Data distance***). Let U_{pp} be a user's privacy preference, and S_{pp} be a service provider's privacy policy. The data distance between their data components is defined as follows:*

[2]Note that here and for the other components, alternative similarity metrics can be easily used as well.

$$\Delta d(U_{pp}.d, S_{pp}.d) = 1 - \frac{|U_{pp}.d \cap S_{pp}.d|}{|U_{pp}.d \cup S_{pp}.d|}$$

Example 4 *Let us consider the user privacy preference and service provider's privacy policy presented in Examples 1 and 2, where the data to which the preference applies are* blood_pressure, weight, heart_beat*, whereas provider's requested data are:* blood_pressure, weight, height, disease_name*. Hence, the distance value of can be calculated as follows:*

$$\begin{aligned}\Delta d(U_{pp}.d, S_{pp}.d) &= 1 - \frac{2}{5} = 1 - 0.40\\ &= 0.60\end{aligned}$$

Retention distance: the retention component of a user privacy preference represents how long a service provider can store, use, and process his/her data, whereas, the analogous component in a provider privacy policy represents how long it wants to store users' data. Since retention can be expressed as a numerical value, we use *Euclidean distance* [56] to measure the retention distance.

Definition 5 *(***Retention distance***). Let U_{pp} be a user's privacy preference, and S_{pp} be a service provider's privacy policy. Let $max(U_{pp}.ret, S_{pp}.ret)$ be the maximum value between the retention components. Therefore, the retention distance is defined as follows:*

$$\Delta ret(U_{pp}.ret, S_{pp}.ret) = \frac{|U_{pp}.ret - S_{pp}.ret|}{max(U_{pp}.ret, S_{pp}.ret)}$$

Example 5 *Let us consider again the user privacy preference (U_{pp}) and service provider's privacy policy (S_{pp}) presented in Examples 1 and 2, where the retentions are* 100 *and* 120*, respectively. Consequently, the retention distance is calculated as follows:*

$$\begin{aligned}\Delta ret(U_{pp}.ret, S_{pp}.ret) &= \frac{|100 - 120|}{max(100, 120)} = \frac{20}{120}\\ &= 0.16\end{aligned}$$

Recipient distance: the data owner can specify whether any additional third-party can use his/her data, besides the service provider itself, by using the recipient attribute of user preferences. For simplicity, we assume that the recipient component of a policy/preference is modeled as a binary value. We therefore use the *Hamming distance* [56].

Definition 6 *(**Recipient distance**). Let U_{pp} be a user's privacy preference, and S_{pp} be a service provider's privacy policy. Therefore, the recipient distance is defined as follows:*

$$\Delta rec(U_{pp}.rec, S_{pp}.rec) = U_{pp}.rec \oplus S_{pp}.rec$$

Hence, the distance is zero if the user's and service provider's retention values are equal, it is one, otherwise.

3.2.2 Learning strategy

The key idea is to exploit learning algorithms to build a classifier able to decide whether a privacy policy, not matching user's privacy preference conditions, has to be anyway accepted due to user's tendency in relaxing his/her privacy preferences. That is, we need a classifier able to decide whether: (i) the not matching attributes of the privacy policy are components that the user is prone to relax (e.g., retention), and (ii) the not matching attributes' values are within a range tolerated by the user (e.g., 10 days). At this purpose, the proposed solution has a first training phase (see Figure 3.2), where the user is required to judge a set of service provider privacy policies (enclosed into a set of Service Joining Requests (SJRs)), labeling them as to be accepted or denied. On this labelled dataset, we then build a classifier able to predict labels for future service join requests based on both: (i) user's preferences on components to be relaxed and (ii) the corresponding relaxation range.

To select the privacy preference components to be relaxed, we use the labeled SRJs as features set on which building the classifier. However, it does not address the prediction on user tolerated values. At this purpose, for each labeled service privacy policy, we compute the distance between it and user's privacy preference, by leveraging on the metrics introduced in Section 3.2.1.

These distances together with the user assigned label represent the features set on which algorithms run. Once the learning phase ends, the learned classifiers are used to label future service joining requests (SJRs) as to be accepted or rejected. More precisely, when a new service joining request SJR arrives, the proposed mechanism firstly computes the policy distance between it and user's privacy preference; secondly, based on these distances, it identifies to which class SJR belongs to (i.e., accept or reject).

Among the available learning strategies, we decided to use supervised machine learning. Indeed, since, we need user feedback to take into account the subjective

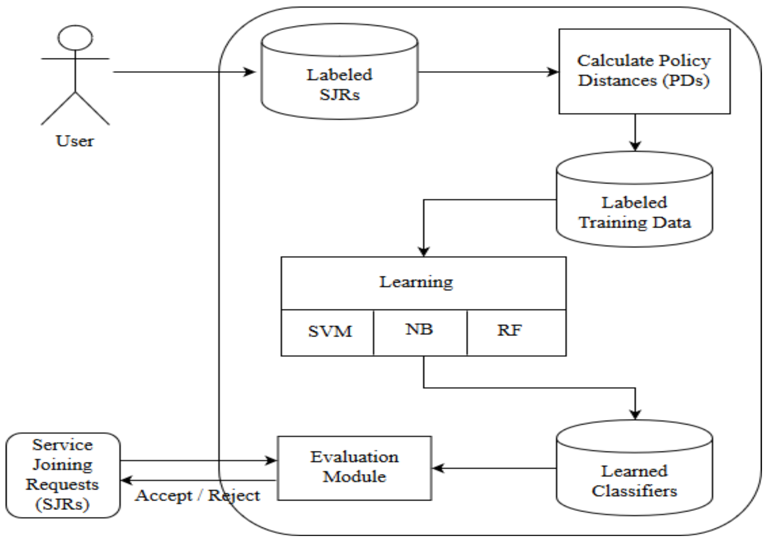


Figure 3.2: Learning architecture

aptidude of the user towards relaxing his/her privacy preferences, unsupervised learning approaches do not fit. Likewise, we do not consider semi-supervised learning approaches, because we are able to perform learning by asking a reasonable number of questions (50) to the users (see Section 3.3), so we think that, by burdening not too much the users, we are able to have the dataset needed for supervised learning. Among possible supervised machine learning algorithms, we choose Support Vector Machine (SVM), Naive Bayesian (NB), and Random Forest (RF) [42].

3.3 Experiments and results

In this section, we present the results of the experimental evaluation carried out to show the effectiveness of the proposed learning approach in capturing users' privacy preference tolerance (i.e., relaxing aptitude). We recall that, to learn user's tolerance in relaxing his/her privacy preferences, we posed some questions to the users. The answer to these questions are then used as a labelled training dataset on which algorithms build the classifiers. To evaluate whether the classifier correctly works on new service joining requests, we need also to ask users to give their feedback on the decisions generated by the system. To do so, we have developed a web application through which evaluators first label a training dataset of service joining requests as depicted in Figure 3.3 (i.e., learning phase), and then give their feedback on labels associated by the system to new service joining requests as depicted in Figure 3.4 (i.e., testing phase). More particularly, the whole process has been divided into two phases. In the first phase (learning phase), each evaluator is presented a privacy preference U_{PP} and asked to associate a decision (*yes*, *no*, or *maybe*) with 50 service joining requests, by assuming U_{PP} as his/her privacy preference. We recall that a service joining request consists of a providers' privacy policy. Based on that, evaluators make decisions on whether they will accept the service joining requests or not. Afterwards, for each evaluator, the collected training dataset is used by three supervised ML algorithms (i.e., SVM, NB, and RF) to learn his/her privacy behaviors (see Figure 3.2). In the second phase (the evaluation phase), the system asks the evaluators whether they are satisfied with the system-generated decision. The evaluators can give *yes*, *no*, and *maybe* as answer where *Maybe* means that the evaluator does not have any opinion on the system-generated decision.

3.3.1 Settings

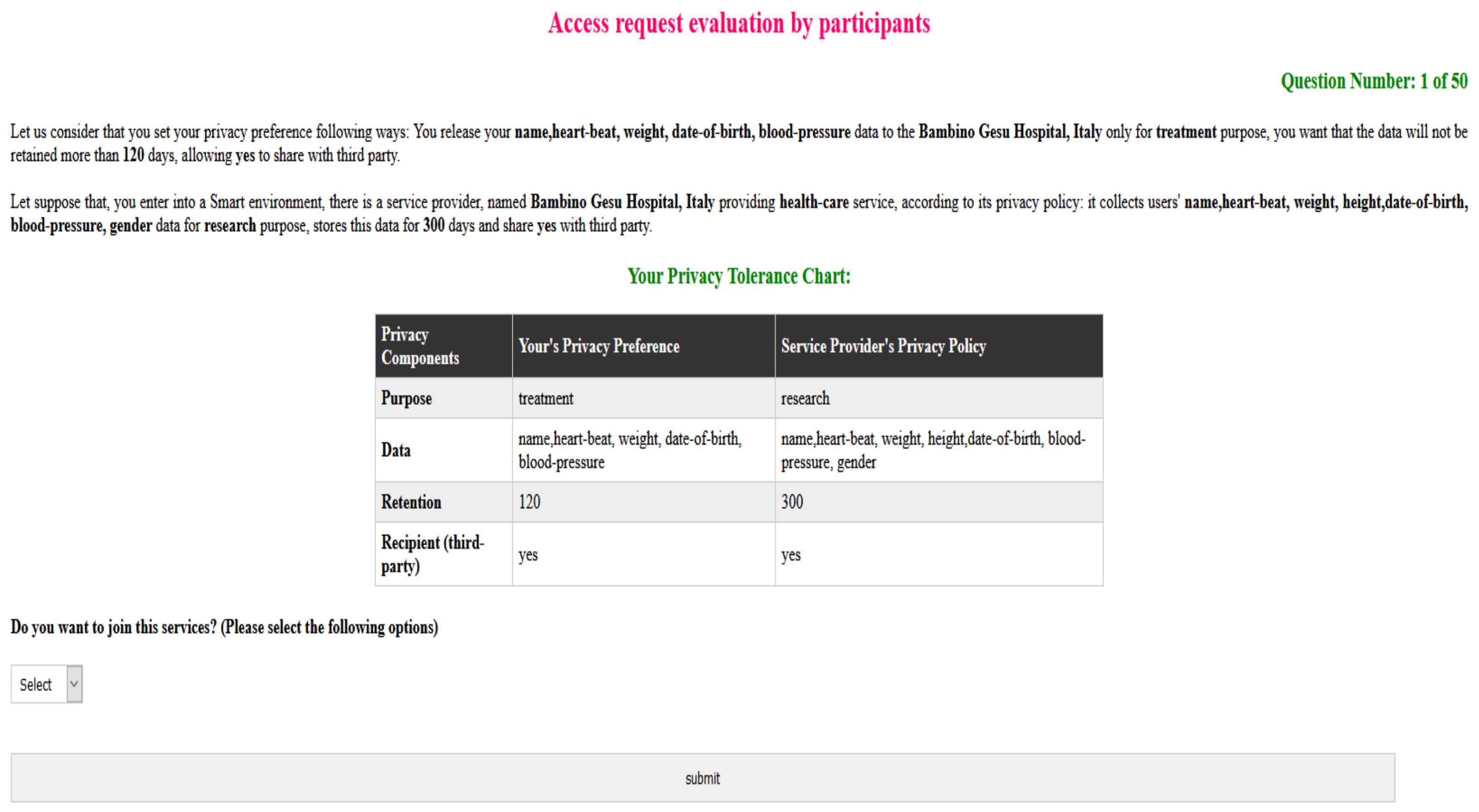
Users' privacy preferences and service providers' privacy policies generation. We have randomly generated users' privacy preferences and service providers' privacy policies. Since both privacy preferences and privacy policies consist of four components (i.e., purpose, data, retention, and recipient), we have first generated possible values for these fields. To build a realistic and meaningful dataset, we have defined 12 purposes (e.g., *payment, treatment, research, and so on.*) for which a service provider is allowed to collect users' data. Moreover, we define 30 different data types (e.g., *name, email_id, phone_number, date_of_birth, credit_card, blood_pressure, and so on.*) to which privacy policies/preferences can refer to. In addition, we consider the retention time between 30 to 365 days, whereas third-party data access status is either *Yes* or *No*.

Evaluator groups. In order to evaluate the performance of the proposed approach with different datasets, we have collected datasets using two types of evaluators, namely, university-based and crowd-sourcing based evaluators. To check the quality of evaluators contribution, we have measured the time that each evaluator has spent on giving feedback, thus to exclude from experimental results those that have devoted too little time.

- **University-based evaluators:** we collected data from 25 CS students from two different universities, placed in two different geographical areas.[3] 6 students are from the University I, and 19 students are from University II.

- **Crowd-sourcing based evaluators:** we have used the Microworkers crowd-sourcing platform[4] with the aim of having a bigger group of evaluators with different nationalities and ages. We have collected data from 160 evaluators (aka workers), by only selecting those having a good work record (i.e., a minimum rating of 4 out of 5). Once the worker accepted the job offer, (s)he

[3]To adhere to the blind review process, we do not report here details on the involved universities.
[4]https://www.microworkers.com



Access request evaluation by participants

Question Number: 1 of 50

Let us consider that you set your privacy preference following ways: You release your **name,heart-beat, weight, date-of-birth, blood-pressure** data to the **Bambino Gesu Hospital, Italy** only for **treatment** purpose, you want that the data will not be retained more than **120** days, allowing **yes** to share with third party.

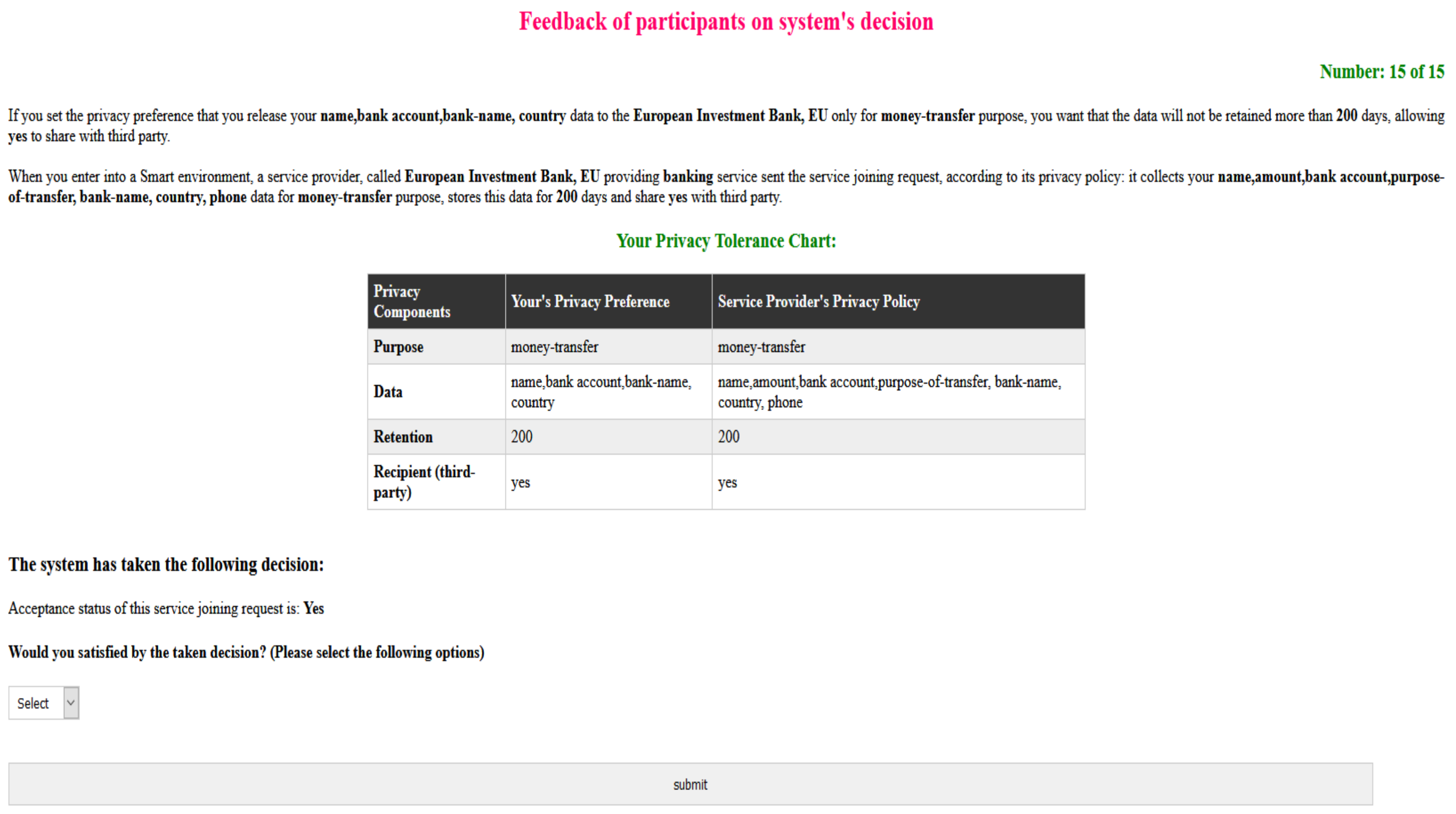
Let suppose that, you enter into a Smart environment, there is a service provider, named **Bambino Gesu Hospital, Italy** providing **health-care** service, according to its privacy policy: it collects users' **name,heart-beat, weight, height,date-of-birth, blood-pressure, gender** data for **research** purpose, stores this data for **300** days and share **yes** with third party.

Your Privacy Tolerance Chart:

Privacy Components	Your's Privacy Preference	Service Provider's Privacy Policy
Purpose	treatment	research
Data	name,heart-beat, weight, date-of-birth, blood-pressure	name,heart-beat, weight, height,date-of-birth, blood-pressure, gender
Retention	120	300
Recipient (third-party)	yes	yes

Do you want to join this services? (Please select the following options)

Figure 3.3: Learning phase



Feedback of participants on system's decision

Number: 15 of 15

If you set the privacy preference that you release your **name,bank account,bank-name, country** data to the **European Investment Bank, EU** only for **money-transfer** purpose, you want that the data will not be retained more than **200** days, allowing **yes** to share with third party.

When you enter into a Smart environment, a service provider, called **European Investment Bank, EU** providing **banking** service sent the service joining request, according to its privacy policy: it collects your **name,amount,bank account,purpose-of-transfer, bank-name, country, phone** data for **money-transfer** purpose, stores this data for **200** days and share **yes** with third party.

Your Privacy Tolerance Chart:

Privacy Components	Your's Privacy Preference	Service Provider's Privacy Policy
Purpose	money-transfer	money-transfer
Data	name,bank account,bank-name, country	name,amount,bank account,purpose-of-transfer, bank-name, country, phone
Retention	200	200
Recipient (third-party)	yes	yes

The system has taken the following decision:

Acceptance status of this service joining request is: **Yes**

Would you satisfied by the taken decision? (Please select the following options)

Figure 3.4: Testing phase

has been redirected to our web application to conduct both learning and evaluation phase.

Experimental setup. After collecting the datasets from the two evaluator groups, we have trained the learning algorithms by exploiting the R platform [57]. We run the experiments on 3.00 GHz Intel Core i5 processor 8 GB RAM using the Windows 10 OS.

Performance metrics. We exploit the 3 X 3 confusion matrix in Table 3.1. More precisely, each column of the matrix represents the predicted class, whereas, each row represents the true class. The diagonal elements of the matrix represent the number of items that have been correctly classified, whereas, other elements of the matrix specify the error. According to the confusion matrix, we define the evaluation metrics (i.e., accuracy, precision, recall, and F1-score) given in Table 3.2.

		Predicted class		
		Yes	No	Maybe
True class	Yes	TP_{yes}	$E_{yes,no}$	$E_{yes,maybe}$
	No	$E_{no,yes}$	TP_{no}	$E_{no,maybe}$
	Maybe	$E_{maybe,yes}$	$E_{maybe,no}$	TP_{maybe}

Table 3.1: Confusion matrix

Accuracy = $(TP_{yes} + TP_{no} + TP_{maybe})$ / total number of samples
Precision Yes = $TP_{yes} / (TP_{yes} + E_{no,yes} + E_{maybe,yes})$
Precision No = $TP_{no} / (TP_{no} + E_{yes,no} + E_{maybe,no})$
Precision Maybe = $TP_{maybe} / (TP_{maybe} + E_{yes,maybe} + E_{no,maybe})$
Recall Yes = $TP_{yes} / (TP_{yes} + E_{yes,no} + E_{yes,maybe})$
Recall No = $TP_{no} / (TP_{no} + E_{no,yes} + E_{no,maybe})$
Recall Maybe = $TP_{maybe} / (TP_{maybe} + E_{maybe,yes} + E_{maybe,no})$
$F1_C = (2 * Precision_C * Recall_C)/(Precision_C + Recall_C)$, where $C \in \{Yes,\ No,\ Maybe\}$

Table 3.2: Metrics

3.3.2 Performance evaluation

In this subsection, we report the experiments we perform to evaluate the accuracy, satisfaction level, and F1-score of the proposed approach by comparing SVM, NB, and RF.

Accuracy. In this experiment, we compare the accuracy obtained by using different classifiers on the two datasets (i.e., university-based and crowd-sourcing based). At this purpose, we use the training datasets and re-label them with the built classifiers.

As shown in Figure 3.5, about 96.4% and 94.5% of the university-based and crowd-sourcing based datasets have correctly labeled by RF, respectively. Likewise, around 81.8% and 77.2% of the university-based and crowd-sourcing based training datasets have been correctly labeled by SVM, respectively, whereas, only 67% and 65.5% of the two training datasets have been correctly labeled by NB, respectively. Therefore, we can see that RF gives better performance than SVM and NB.

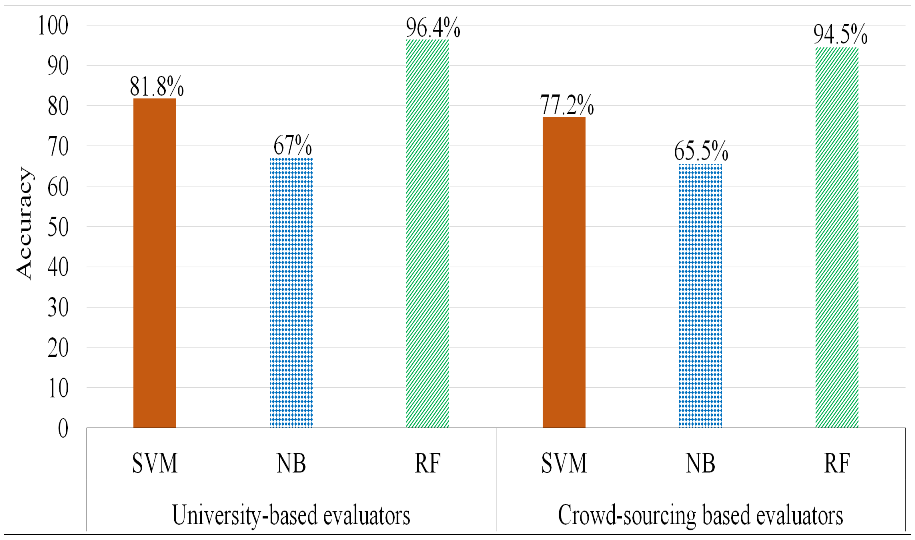


Figure 3.5: Accuracy of different classifiers

Satisfaction level. In this experiment, we show how many evaluators are satisfied with the decisions taken by the system. We consider the feedback received by the evaluators during the testing phase. In this phase, the web application shows new service joining requests (i.e., privacy policies of service providers) with the corresponding system-generated decisions (i.e., labels) and it asks the evaluators whether they are satisfied with the system-generated decision. We consider a total of 15 new service joining requests. Among them, 12 request decisions have been generated by the three classifiers, where each classifier generated the label for 4 service joining requests. The remaining 3 service joining requests are taken from the set of joining requests that the evaluators have already labeled during the learning phase. This allows us to measure the evaluators' quality (as later explained). As shown in Figure 3.6, around 96.5% and 83.4% of the university-based and crowd-sourcing based evaluators are satisfied with the decisions taken by the system using RF, respectively. Similarly, around 78.2% and 70.5% of the university-based and crowd-sourcing based evaluators are satisfied with the decisions taken by SVM, respectively, whereas, about 78.4% and 72.6% of the university-based and crowd-sourcing based evaluators are satisfied with the decisions taken by NB, respectively. Therefore, by this experiment, we can see that all algorithms achieve a good satisfaction level (above 70%), whereas, RF outperforms the others.

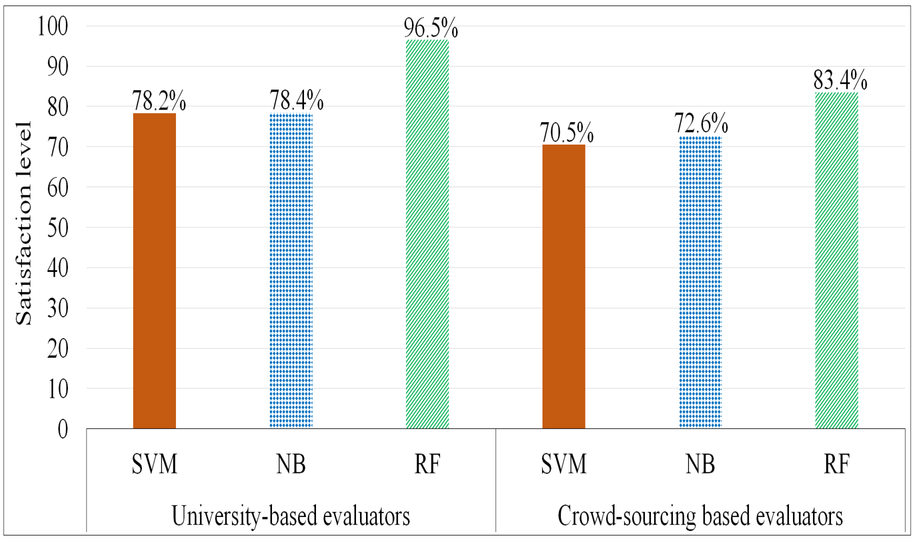


Figure 3.6: Evaluators satisfaction level

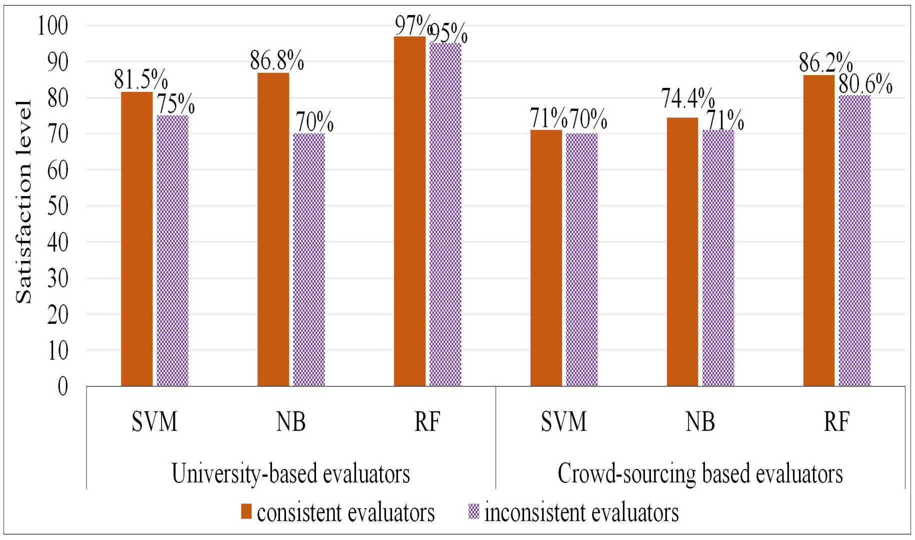


Figure 3.7: Satisfaction level of consistent and inconsistent evaluators

Evaluators quality. As mentioned earlier, to measure the evaluators' quality and how a badly labelled training dataset impacts on the satisfaction level, we have used three service joining requests which have been labeled in the first phase and presented them again in the evaluation phase. More precisely, in the second phase, the web application shows these three service joining requests along with the decision given by the users in the training phase. We then collect the label (i.e., satisfaction level) evaluators assign to these decisions. Based on this, we measure whether the evaluator is consistent or not with his/her previous decision. We assume that an evaluator is consistent if two out of three decisions taken during the training and the evaluation phase match. The percentage of consistent evaluators is 76% for the university-based, whereas it is 80% for the crowd-sourcing based. Figure 3.7 shows the satisfaction level of consistent and inconsistent evaluators for the two groups. As expected, the satisfaction level of consistent evaluators is greater than the satisfaction level of inconsistent evaluators.

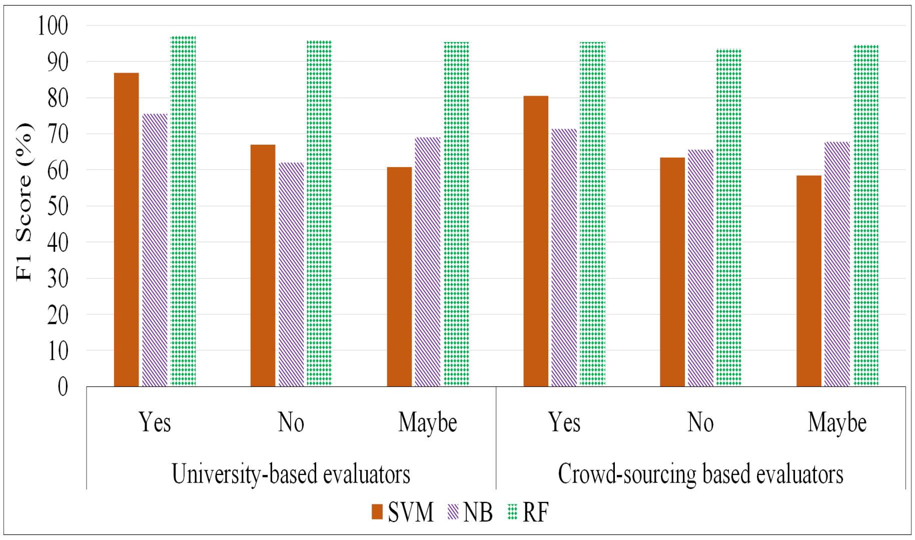


Figure 3.8: Comparison of F1 score of different classifiers for training datasets

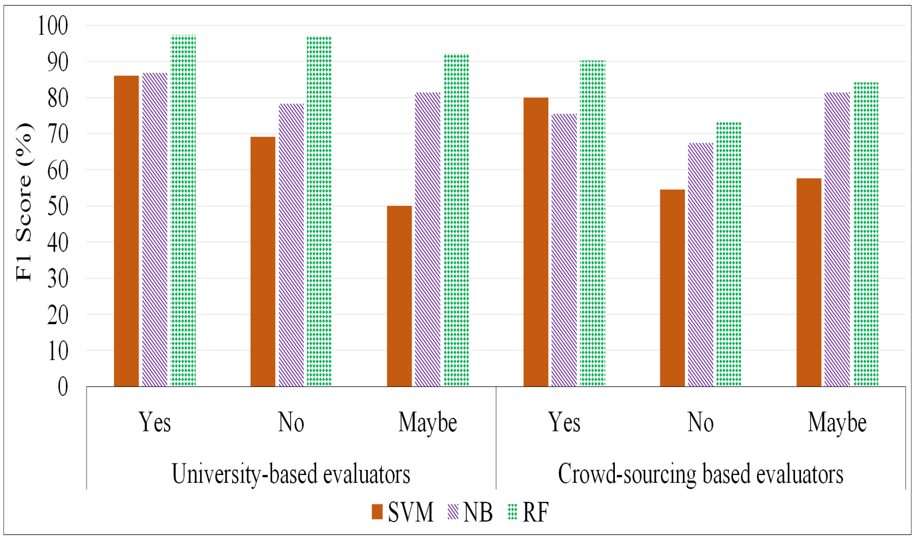


Figure 3.9: Comparison of F1 score of different classifiers for testing datasets

F1-score. We measured the F1-score for each class (i.e., Yes, No, and Maybe) in the training dataset (see Table 3.3) and in the testing dataset (see Table 3.4). Figures 3.8 and 3.9 shows the comparison for the different classifiers. From our analysis, it can be observed that, for both datasets, RF gives greater F1-score for all three classes than other learning algorithms. In particular, service acceptance (aka Yes) gives the highest F1-score for both datasets (97.4% and 90.3%, respectively).

However, from the above results analysis, we can see that RF classifier provides better performance in terms of accuracy, F1-score, and satisfaction level. This is motivated by the fact that the labels collected from evaluators are not uniformly

		SVM			NB			RF		
		Yes	No	Maybe	Yes	No	Maybe	Yes	No	Maybe
University-based evaluators	Precision	82.2%	81.9%	67.77%	86.18%	61.06%	58.2%	96.94%	95.8%	97.59%
	Recall	92.21%	56.66%	55.17%	67.26%	63.06%	84.58%	97.61%	95.87%	93.29%
	F1-score	86.9%	66.98%	60.82%	75.55%	62.04%	68.95%	97.27%	95.83%	95.4%
Crowd-sourcing based evaluators	Precision	78.26%	72%	64.45%	77.94%	61.36%	62.05%	95.84%	94.27%	94.69%
	Recall	82.88%	56.61%	53.31%	65.65%	70.38%	74.41%	94.85%	93.08%	94.42%
	F1-score	80.50%	63.38%	58.35%	71.26%	65.56%	67.67%	95.34%	93.67%	94.55%

Table 3.3: Performance comparison of different learning algorithms for the training datasets

		SVM			NB			RF		
		Yes	No	Maybe	Yes	No	Maybe	Yes	No	Maybe
University-based evaluators	Precision	83.09%	73.07%	67%	86%	82.8%	73.3%	96.6%	97%	100%
	Recall	89.39%	65.5%	40%	87.7%	74.3%	91.6%	98.3%	97%	85.7%
	F1-score	86.12%	69.5%	50.09%	86.84%	78.32%	81.43%	97.4%	97%	92.3%
Crowd-sourcing based evaluators	Precision	78.8%	50%	83.3%	81.9%	60.8%	84.2%	94.6%	64%	90.3%
	Recall	81.2%	60%	44%	70%	75.8%	78.87%	86.4%	85.6%	78.9%
	F1-score	80%	54.5%	57.58%	75.48%	67.47%	81.4%	90.3%	73.24%	84.2%

Table 3.4: Performance comparison of different learning algorithms for the testing datasets

distributed, and RF has a good ability to deal with datasets with imbalanced class frequencies.

3.4 Chapter summary

This chapter presented a privacy preference adaptation framework that is able to relax, in a controlled way, individuals' privacy preferences in order to join more services. This approach exploits ML algorithms (i.e., SVM, NB, and RF) that discloses the personal data whose release has been explicitly learned from data owners previous feedbacks. We have extensively tested the proposed approach by using evaluators recruited from university students as well as a crowd-sourcing platform. The obtained results show that with the Random Forest (RF) classifier, around 96.5% of university-based evaluators and 83.4% of crowd-sourcing based evaluators are satisfied with the decisions taken by the system.

4 ML-based Privacy Preferences Suggestions

Today, most of the provided services are context dependent, that is, the deliver of the service is based on users' contextual information (e.g., time, location). Examples are location-based services, IoT-based services and so on. Typically, contexts can impact the user privacy preferences, for instance, a user may feel comfortable to access entertainment services when (s)he stays at home, but (s)he will not be comfortable to access the same type of services during office hours, when he/she is in his/her office. Many studies show how contextual information is important in privacy preferences specification [22, 27, 58, 59]. For example, Nissenbaum et al. [59] shows that most of the privacy preference models fail to protect against violations of user privacy preferences because they do not keep into account contextual information. As a matter of fact, many of the existing privacy preferences frameworks (e.g., [20, 60, 61]) do not consider individuals' contextual information to make privacy aware decisions.

To cope with this limitation, users might specify context-based privacy preferences, that is, privacy preferences stating conditions on how personal data has to be used based on current situation (e.g., no access to entertainment services when the location is office). This brings the nice benefit of increasing the user control over his/her data. However, since a user may interact with several contexts, it also increases the number of preferences that (s)he has to specify and manage, resulting in a very complex and time-consuming task. For this reason, we propose a ML-based privacy preferences management service that helps users to manage their privacy preferences when they move to a new context. More particularly, we design a framework that infers individuals' privacy preferences based on their contextual information.

To show the feasibility of the proposed approach, we have conducted some preliminary experiments. In particular, we compared the proposed approach with a naive approach, that is, a classifier suggesting context-based privacy preferences without leveraging on previously specified context-based privacy preferences. At this purpose, we have extended the approach in [61], where a learning approach has been proposed to suggest *non context-based* user privacy preferences. The approach in [61] first creates a training dataset of user labels on a set of service requests (e.g., a label is the accept/deny decision on a given service request), then generate a classifier on it, able to automatically decide if a new service request has to be accepted or denied. In order to compare the proposed approach with [61], we extended the latter such as to collect users' labels on service requests complemented with context information. Moreover, we test different supervised ML algorithms, namely, Logistic Regression (LR), Random Forest (RF), and Naive Bayesian (NB) [42]. The obtained results show that the proposed approach gives better performance than the naive approach.

The rest of this chapter is organized as follows. In Section 4.1, we describe our proposal, whereas Section 4.2 shows the results of our preliminary experiments.

4.1 Proposed methodology

As mentioned earlier, individuals' privacy expectations are highly context dependent, and since users change their context very often, setting up privacy preferences for every new context is a very complex and time-consuming task. To address this issue, we design a service helping users to manage privacy preferences setting when they move from one context to another. The main idea is to infer the best privacy preferences for the new context leveraging on privacy preferences previously specified by the user for different contexts. Thus, as a first step, we need to select among the contexts for which the user has already specified a preference, those that are similar to the new one. In doing this, we do not simply rely on a similarity measure between contexts, but we also want to keep into account users' perspective (see Section 4.1.3). Indeed, given a similarity measure, two existing contexts could have the same distance to the new one but differ on a few fields (e.g., time, location) that are very relevant for that user. As such, we would like also to take into account, for a target user, his/her preferences on which context field is more informative and thus should have more relevance in the similarity measure.

Once the most similar and most relevant context has been selected, the system has to retrieve the corresponding privacy preference. Here, the basic assumption is that this preference might represent a good match for the new context but some slightly modifications might be needed as well. In order to understand whether and how the identified privacy preference has to be adapted for the new context, we want to take into account once again user's perspective. More precisely, to learn which fields of the identified privacy preference need to be modified (e.g., purpose, data, retention, and recipient), we exploit ML to infer, from each user, which component and how is willing to adapt it.

All the above described steps will be described in the following, starting from the modeling of contexts and context-based privacy preferences.

4.1.1 Contexts and context-based privacy preferences modeling

A context is defined by the information used to characterize the present situation of individuals (e.g., activity, time, etc.). For the sake of simplicity, in this paper, we consider contexts containing information on four different dimensions, namely, time, location, activity, and social. The latter two represent the action the user is currently doing (e.g., work, run, relaxing, etc) and companion with whom (s)he is (e.g., alone, friends, colleagues, etc). However, the approach can be extended to the consideration of additional contextual information.

More formally, user contexts can be defined as follows.

Definition 7 **(User context)**. *A context for a user* U*, denoted as* CTX_U*, is a set of pairs* $\{(tm, v_{tm}), (lc, v_{lc}), (ac, v_{ac}), (sl, v_{sl})\}$*, where the first component is a contextual property and the second its corresponding value. More precisely,* tm *denotes a time,* lc *a user location,* ac *denotes a user activity, whereas* sl *specifies the social dimension (i.e., user companion).*

Example 6 *Let us suppose that a user* U*'s* $location$ *is* $library$*, time is* $Monday\ morning$*, and (s)he is* $studying$ *with his/her* $brother$*. Therefore, the current context of* U *can be modelled in the following way:*

$$CTX_U = \begin{cases} tm = Monday\ morning \\ lc = library \\ ac = studying \\ sl = brother \end{cases}$$

To provide a more fine-grained control of personal data release, a user can set up different privacy preferences for different contexts, stating the conditions according to which his/her data has to be used and managed in that particular context. We formally define a user's context-based privacy preference as follows.

Definition 8 **(Context-based privacy preference)**. *A context-based privacy preference for a user* U*, denoted as* CTX_{pp_U}*, is a tuple (*CTX*,* PP*), where,* CTX *is a* *context, and PP is a tuple (p, d, ret, rec), where, p denotes the purpose for which a service provider is allowed to collect the data denoted by d, ret specifies how long the service provider can store the data, whereas rec indicates whether additional third party entities can use the data.*

Example 7 *Let us consider a user* U *that wishes to release his/her $name$, $date\_of\_birth$, $certificates$ data only for $admission$ purpose. Moreover, (s)he wants that the data will not be retained more than 260 $days$, allowing service providers to share it with $third\_party$. Let us assume that the user wants this preference to be enforced in the context presented in Example 6. Therefore, such context-based privacy requirements can be encoded through the following context-based privacy preference:*

$$CTX_{pp\_U} = \begin{cases} tm = Monday\ morning \\ lc = library \\ ac = studying \\ sl = brother \\ p = admission \\ d = name, date\_of\_birth, certificates \\ ret = 260\ days \\ rec = yes \end{cases}$$

4.1.2 Context distance metrics

As mentioned earlier, when a user moves to a new context, we calculate a similarity score between the new context and all the contexts for which the user has already defined a privacy preference. To find the most similar context and corresponding privacy preference, we measure a distance to determine how far the new context is from existing contexts. To do so, we measure the distance of each context component, as explained in what follows.

Time distance: time can be expressed as a numerical value[1], hence, we can use the *Euclidean distance* [56] to measure the time distance between different contexts.

Definition 9 (**Time distance**). *Let CTX_{U_n} be a user new context, and CTX_{U_p} be a user's prior context. Let $max(CTX_{U_n}.tm, CTX_{U_p}.tm)$ be the maximum value between the time components. Therefore, the time distance is defined as follows:*

[1]For the sake of simplicity, in this thesis, the time is expressed by only considering 4 time slots for each weekday (e.g., morning ($6.00 - 11.59$), afternoon ($12.00 - 17.59$), evening ($18.00 - 23.59$), and night ($0.00 - 5.59$)).

$$D_{tm}(CTX_{U_n}.tm, CTX_{U_p}.tm) = \frac{|CTX_{U_n}.tm - CTX_{U_p}.tm|}{max(CTX_{U_n}.tm, CTX_{U_p}.tm)}$$

Location distance: In this work, rather than the exact GPS location, we are interested in modelling locations that can be sensitive for personal data release (e.g., home, office). To this end, we rely on the Aura Location Identifier (ALI) model [62]. The main idea of this model is to decompose physical spaces into different levels of spaces. For instance, the campus of the *University of Insubria* can be decomposed into several spaces: *Rossi Building, Morselli Building, Antonini Building, etc*. Each of these buildings is in turn divided into smaller composing sub-spaces, until reaching enough precision. This hierarchical representation is called a *space tree*, where each node corresponds to a given space in the physical environment. Figure 4.1 shows part of a space tree (i.e., location hierarchy) for the *University of Insubria*. By exploiting this hierarchy, we measure the distance between spaces, by leveraging on the *Wu and Palmer similarity* [54] metric.[2]

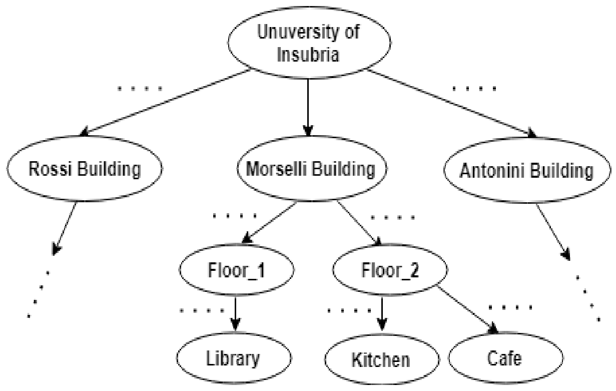


Figure 4.1: Location hierarchy

Definition 10 **(Location distance)**. *Given two location l_1 and l_2, let ccn be the closest common ancestor between l_1 and l_2 in the space tree, $depth(ccn)$ be the number of edges from the root to ccn, $dis(l_1)$ and $dis(l_2)$ be the distance of location l_1 and l_2 from ccn, respectively. Therefore, the location distance is defined as follows:*

$$D_{lc}(l_1, l_2) = 1 - \frac{2 * depth(ccn)}{dis(l_1) + dis(l_2) + 2 * depth(ccn)}$$

Activity distance: we exploit the daily living activities ontology [63] to build an individuals' *activities hierarchy* (see Figure 4.2). If a_1 and a_2 are two activities, we

[2]Note that alternative similarity/distance metrics can be easily used as well. Here, we have selected the Euclidean distance, and Wu and Palmer similarity/distance metrics because of their simplicity and ability to provide high performance.

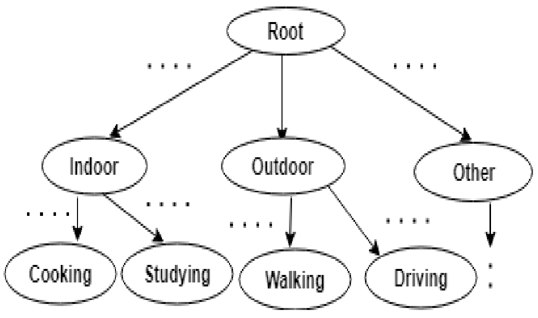


Figure 4.2: Activities hierarchy

measure their distance, denoted as $D_{ac}(a_1, a_2)$, as in Definition 10, by exploiting the activity hierarchy instead of the location one.

Social distance: to calculate the distance between the social attributes, we exploit an ontology similar to the one used by social networks [64]. If s_1 and s_2 are two social attributes, we measure their distance, denoted as $D_{sl}(s_1, s_2)$, as Definition 10, by exploiting the social hierarchy (cfr. Figure 4.3). However, it is noteworthy to mention that the children of the root node are disjoint. Therefore, the nodes that belong to the different subtrees, their distance would be maximum. For understanding more clearly, let's take an example, look at the social hierarchy as shown in Figure 4.3, here, the distance between $Brother$ and $Close\_friends$ is greater than the distance between $Brother$ and $Sister$, because $Brother$ and $Close\_friends$ nodes belong to the different subtrees, whereas the nodes $Brother$ and $Sister$ belong to the same subtree.

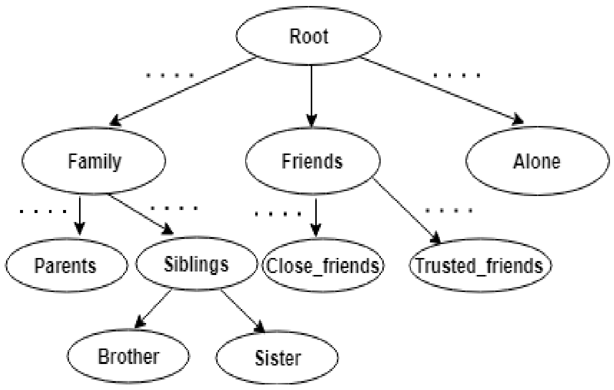


Figure 4.3: Social hierarchy

Example 8 *Let us consider a user U's context-based privacy preference (CTX_{pp_U}) presented in Example 7. Let us assume that U moves to a new context, according to which location is $kitchen$, time is $Monday\ morning$, activity is $cooking$ with $parents$. According to the location, activity, social, time distance definitions and the hierarchies* *shown in Figure 4.1, 4.2 and 4.3, we can measure the distance between each context components as follows:*

$$\begin{aligned} D_{lc}(library, kitchen) &= 1 - \frac{(2*1)}{(2+2+2*1)} = 1 - \frac{2}{6} \\ &= 1 - 0.34 \\ &= 0.66 \end{aligned}$$

Likewise,

$$\begin{aligned} D_{ac}(studying, cooking) &= 1 - 0.5 = 0.5 \\ D_{sl}(brother, parents) &= 1 - 0.4 = 0.6 \\ D_{tm}(Monday\ morning, Monday\ morning) &= 0 \end{aligned}$$

4.1.3 Context similarity

When computing the similarity between two contexts, we take into account which are the contextual information that a user considers more relevant in determining the similarity. Therefore, we add weights to the similarity measure to keep into account the user perspective. Such weights will be learnt by using machine learning algorithms (as explained in Section 4.1.4).

Definition 11 **(Context similarity score)**. *Let CTX_{U1} and CTX_{U2} be two contexts for user U. Let $w_1, \ldots w_4$ be the weights associated with each of the four context attributes. Therefore, the similarity score is defined as follows:*

$$\mathrm{Sim}_w(CTX_{U1}, CTX_{U2}) = 1 - \frac{\begin{pmatrix} w_1 * D_{tm}(CTX_{U1}.tm, CTX_{U2}.tm) + w_2 * D_{lc}(CTX_{U1}.lc, CTX_{U2}.lc) \\ + w_3 * D_{ac}(CTX_{U1}.ac, CTX_{U2}.ac) + w_4 * D_{sl}(CTX_{U1}.sl, CTX_{U2}.sl) \end{pmatrix}}{4}$$

Example 9 *Let us consider user U's context-based privacy preference presented in Example 7, and the new contexts and related distance measures illustrated in Example 8. Suppose that the learned weights are $w_1 = 0.1$, $w_2 = 0.2$, $w_3 = 0.3$, and $w_4 = 0.4$. Therefore, according to Definition 11, the similarity score is calculated as follows:*

$$\begin{aligned} Sim_w(CTX_1, CTX_2) &= 1 - \frac{0.1*0 + 0.2*0.66 + 0.3*0.5 + 0.4*0.6}{4} \\ &= 1 - \frac{0 + 0.132 + 0.15 + 0.24}{4} \\ &= 1 - 0.13 \\ &= 0.87 \end{aligned}$$

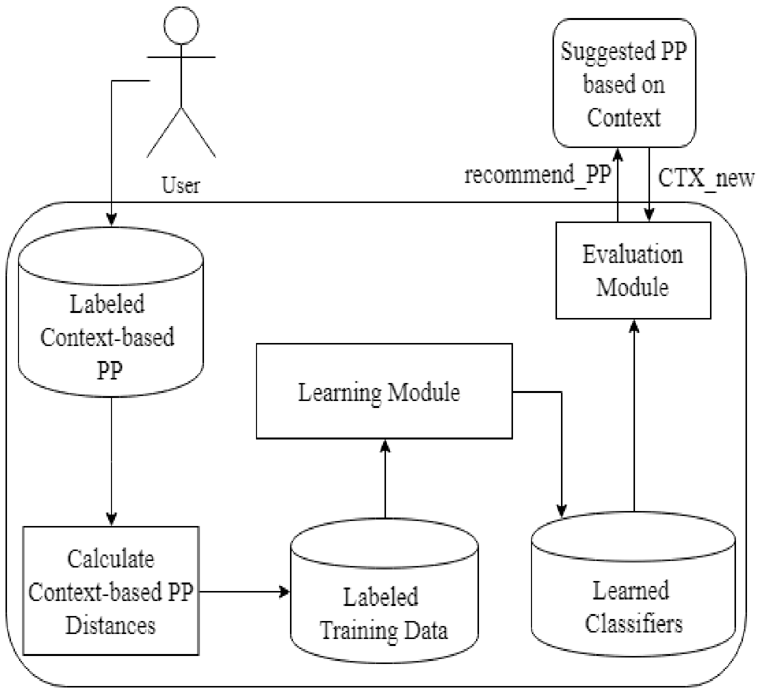


Figure 4.4: Learning architecture

4.1.4 Learning mechanism

In this subsection, we explain the learning strategy we have designed. Since we need users feedback, we exploit supervised ML algorithms, namely, Logistic Regression (LR), Random Forest (RF), and Naive Bayesian (NB) [42]. Moreover, the preliminary experiments we have carried out show that, with a reasonable user burden (i.e., asking only 30 questions), we are able to get a dataset that is adequate for supervised learning, hence, we do not consider semi-supervised learning approaches. However, it should be mentioned that our collected dataset is small. Since the neural network does not work well with small data, therefore, we do not consider deep learning approaches (e.g., neural networks) in the analysis.

We use ML algorithms to build a classifier able to decide which privacy preference will be set for the user new context. More particularly, we build a classifier able to decide: (i) which elements of the context are more relevant (e.g., location, time) for the target user; and (ii) how much privacy preference components associated with preferences defined for similar contexts could be modified.

For this purpose, the proposed solution has a first training phase (see Figure 4.4), where the user is required to judge whether a given privacy preference works for a certain context or (s)he wishes to modify it. More precisely, let CTX_{new} be a new context and CTX_{sim} the most similar context to CTX_{new}.[3] We take the privacy preference associated with CTX_{sim}, denoted as PP_{sim_ctx}, and we ask the user if (s)he would adopt PP_{sim_ctx} in CTX_{new} as is or would like to modify it. In the latter case, we let the user modify it obtaining a new preference, denoted as PP_{mod}. Then, we measure the distance between each component of PP_{sim_ctx} and PP_{mod}, denoted as Dis_{pp}. For measuring the distance between privacy preferences components, we rely on the metrics introduced in the previous chapter (see Section 3.2.1). Finally, the features set on which we build the proposed classifier consists of Dis_{ctx}, Dis_{pp} and the assigned label (adopt/modify), where Dis_{ctx} is the set of distance between each component of CTX_{new} and CTX_{sim}.

Once the learning phase is concluded, we exploit the learned classifiers to infer privacy preferences for new contexts. More particularly, when the user moves to a new context CTX_{new}, the proposed approach (i.e., evaluation module) computes similarity scores between the new context and all contexts for which a privacy preference has been previously specified. We recall that according to Definition 11, this score exploits $w_1, \ldots, w_4$ weights to take into account which context field (e.g., time or location) is more relevant to the user. These weights are initialized with values of weights of the classifier model built in the training phase. As an example, w_1 weight, associated to the time context field in Definition 11, is initialized with the value of weight computed by classifier for the feature containing the distance between time in Dis_{ctx}.

Then, it selects the privacy preference of the most similar context, and adapts the selected privacy preference according to the learnt user's adapting attitudes. More precisely, to determine how much each privacy preference component has to be modified, we exploit the correlation between privacy preference components and the context fields. For this purpose, we used a linear regression model [65], defined as $Y = r_1x_1 + r_2x_2 + \ldots + r_nx_n + \epsilon$ where, $r_1, \ldots, r_n$ are the regression coefficients, $x_1, \ldots, x_n$ are the independent variables, ϵ is the constant, and n is the number of attributes (our case $n = 4$). To train this model, for each user, we have used the feature set Dis_{ctx} as independent variables and as a target variable/label (i.e., Y) the value of Dis_{pp}. Once trained, this model can be used to update each privacy preference component. For example, let consider, as the privacy preference component that needs to be modified, the retention attribute ret_{mod} having value 260 days. Let assume that the learned coefficients $r_1, \ldots, r_4$ are 2, 8, 15, and 11, respectively, and the constant value ϵ is -10. Let suppose the distance values between the new context and the similar context is the one presented in Example 8. Therefore, the update retention value would be: $ret_{updt} = ret_{mod} + Y = 260 + (2 * 0 + 8 * 0.64 + 15 * 0.5 + 11 * 0.6 - 10) = 270$ days.

[3]We assume that user has inserted a preliminary small set of context-based privacy preferences. CTX_{sim} is selected among contexts associated to these preferences.

4.2 Experiments

In this section, we illustrate a series of preliminary experiments we have performed to show the effectiveness of the proposed approach. More particularly, to demonstrate the feasibility of the proposed approach, we compare it with the naive strategy of inferring new context-based privacy preferences from scratch. At this aim, we have extended the approach proposed in [61], where a learning approach has been proposed to suggest traditional (i.e., non context-based) privacy preferences. More precisely, [61] creates a training dataset of user labels on a set of service requests (e.g., accept/deny decision on a given service request), and builds a classifier on this training dataset, able to automatically decide if a new service request should be accepted or denied. To make a fair comparison with the proposed approach, we modify the learning strategy proposed in [61] so that it can infer privacy preferences from service requests containing also contextual data. We first measure the accuracy and F1 score obtained by using LR, RF, and NB. Next, we compute the *satisfaction level* of users regarding privacy preference suggestions generated by both approaches using various learning strategies. In addition, we also evaluate the user quality in terms of feedback on the training dataset to examine how a badly labeled training dataset impacts user satisfaction.

4.2.1 Experimental settings

To generate a meaningful dataset, we consider the following contextual information: 23 different locations (e.g., *home, office, university*, and so on), 7 days (e.g., *Sunday, Monday*, etc.) with 4 time slots (e.g., *morning, afternoon, evening, and night*), 12 different user activities (e.g., *studying, meeting, sleeping*, and so on), and 10 different social attribute values (e.g., *alone, family, friends*, and so on). Moreover, for privacy preferences components, we have considered purposes, data types, retention time and recipient value as did in Section 3.3.1. To collect labels for the training dataset, we developed a web application through which users can give feedback on system-generated privacy preferences. We have recruited two types of evaluators, namely, university-based and crowd-sourcing based evaluators. More particularly, we first collected data of 10 CS students from the Islamic University, Bangladesh. Then, to have a bigger group of evaluators with different nationalities and ages, we used the Microworkers crowd-sourcing platform.[4] From this platform, we have collected data from 50 evaluators (aka workers). We recall that to learn user's aptitude in adapting his/her context-based privacy preferences, we posed some questions to the users. The answers to these questions are then used as a labeled training dataset on which we build the classifiers. It should be pointed out that we have collected two types

[4]https://www.microworkers.com

		Predicted class	
		Yes	No
True class	Yes	TP_{yes}	$E_{yes,no}$
	No	$E_{no,yes}$	TP_{no}

Table 4.1: Confusion matrix

Metrics
Accuracy = $(TP_{yes} + TP_{no})$ / total number of samples
Precision Yes = TP_{yes} / $(TP_{yes} + E_{no,yes})$
Precision No = TP_{no} / $(TP_{no} + E_{yes,no})$
Recall Yes = TP_{yes} / $(TP_{yes} + E_{yes,no})$
Recall No = TP_{no} / $(TP_{no} + E_{no,yes})$
$F1_C = (2 * Precision_C * Recall_C)/(Precision_C + Recall_C)$, where $C \in \{Yes, No\}$

Table 4.2: Metrics definition

of dataset. First, we collect a labeled training dataset for the naive approach from 30 users (i.e., 10 CS students plus 20 workers), denoted in what follows as naive approach dataset, then we exploit other 30 users to collect a labeled training dataset for evaluating our approach (we name this as proposed approach dataset).

After collecting the datasets, we have trained the machine learning classifiers by exploiting the R platform [57]. To evaluate whether the classifier correctly works on the new context, we also ask users to give their feedback on the privacy preference suggestions generated by the system (i.e., testing phase). In order to measure the effectiveness of the proposed approach, we consider the confusion matrix illustrated in Table 4.1. According to this, we exploit the standard evaluation metrics, namely, accuracy, precision, recall, and F1-score, illustrated in Table 4.2.

4.2.2 Effectiveness

In this experiment, we carried on a comparative analysis of accuracy and F1-score obtained by the proposed approach and the naive one using different classifiers.

Accuracy. As a first experiment, we use the training datasets and re-label them with the built classifiers. As shown in Figure 4.5, about 99% and 98% of the proposed approach and naive approach datasets have been correctly labeled by RF, respectively. Likewise, around 98% and 93.2% of the proposed approach and naive approach datasets have been correctly labeled by LR, respectively, whereas, only 88.3% and 76.7% of the two training datasets have been correctly labeled by NB, respectively.

Therefore, we can see that RF gives better performance on proposed approach than the naive one.

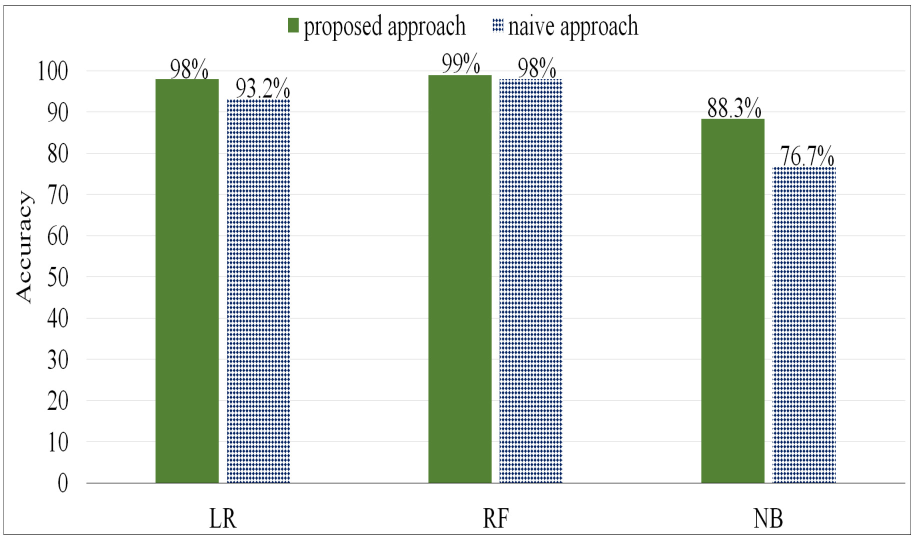


Figure 4.5: Comparison of accuracy of different approaches

F1-score. We have calculated the F1-score for each class (yes, no) for comparing the performance among the learning approaches over the training dataset and testing dataset (see Table 4.3). Figures 4.6 and 4.7 represent the F1-score comparison for the two approaches using different classifiers. It can be observed that, for both approaches, RF gives greater F1-score. More particularly, by using RF, the proposed approach achieves 97% and 94% F1-score for the class label *yes* on the training and testing dataset, respectively.

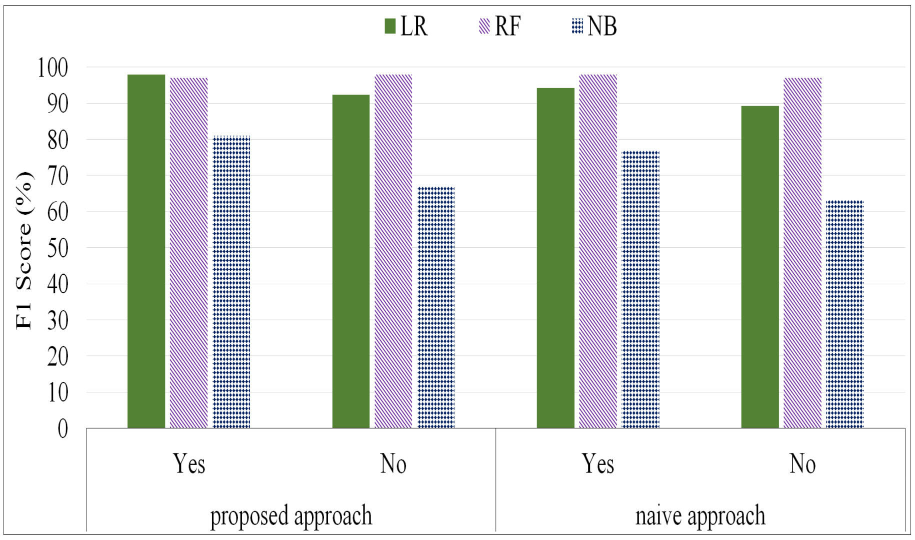


Figure 4.6: Comparison of F1-score of different classifiers for training dataset

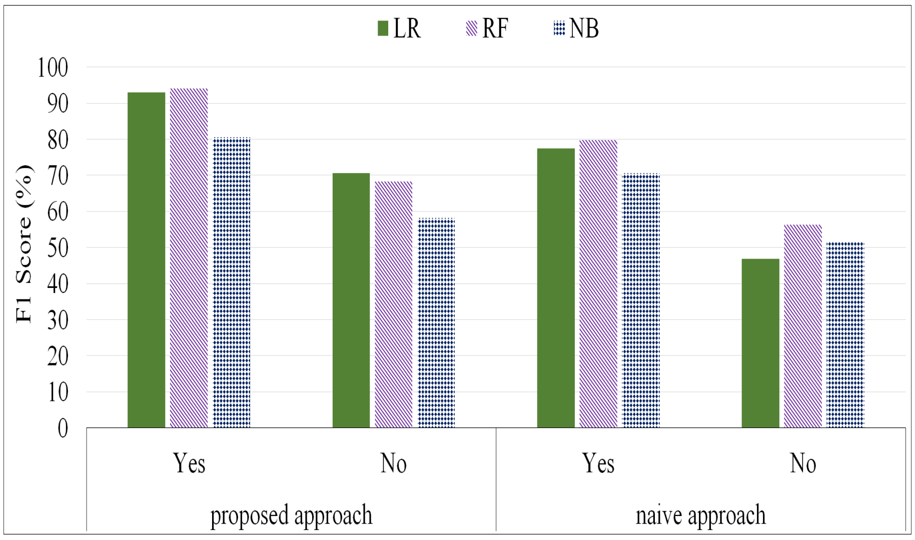


Figure 4.7: Comparison of F1-score of different classifiers for testing dataset

4.2.3 Participant evaluation

In this experiment, in order to evaluate user satisfaction, we collect feedback from the users regarding the privacy preference suggestions taken by both approaches using various learning strategies.

Satisfaction level. We exploit the developed web application to show evaluators the system-generated privacy preferences for the new context, and we ask the evaluators to give their feedback regarding the system-generated suggestions. More precisely, we have shown to each evaluator 15 context-based privacy preferences, where, 12 of them have been generated by the classifiers, whereas the remaining 3 are taken from the set of context-based privacy preferences that the evaluators have set up during the learning phase. These are used for checking the consistency of evaluators feedback and measuring the evaluators' quality (as later explained). As shown in Figure 4.8, about 65% of the evaluators are satisfied with the suggestions given by the proposed approach using RF, whereas, around 57.5% of the evaluators are satisfied with the suggestions given by the naive approach. Similarly, by using LR, around 62% and 53.3% of the evaluators are satisfied with the suggestions given by the proposed approach and the naive approach, respectively. Likewise, by using NB, about 49% and 41.6% of the evaluators are satisfied with the suggestions given by the proposed approach and the naive one, respectively. Therefore, it is clear that the proposed approach achieves higher satisfaction level than the naive one.

Evaluators quality. Through this experiment, we are interested in investigating how a badly labeled training dataset impacts the satisfaction level. With this aim, we used some techniques to identify consistent and inconsistent evaluators. To do so, we have taken 3 context-based privacy preferences from the set of context-based privacy

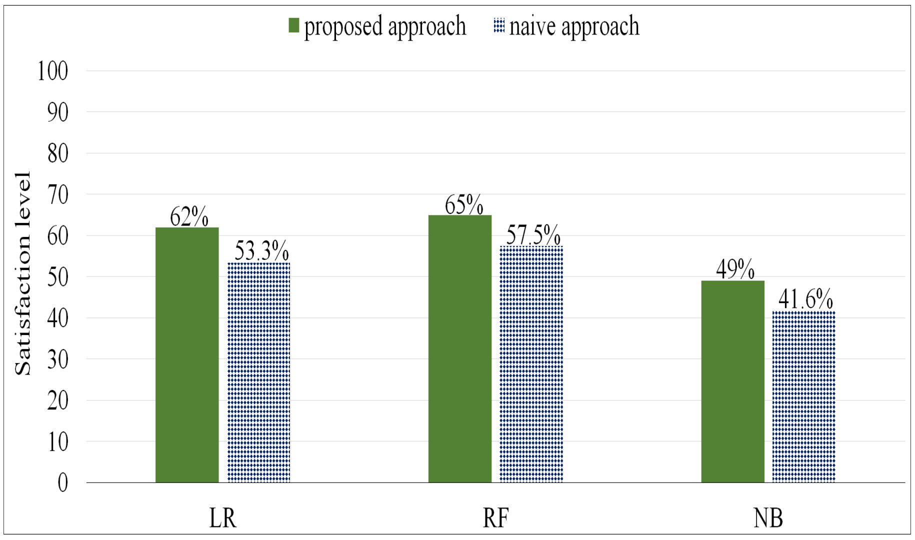


Figure 4.8: Comparison of evaluators satisfaction level for different approaches

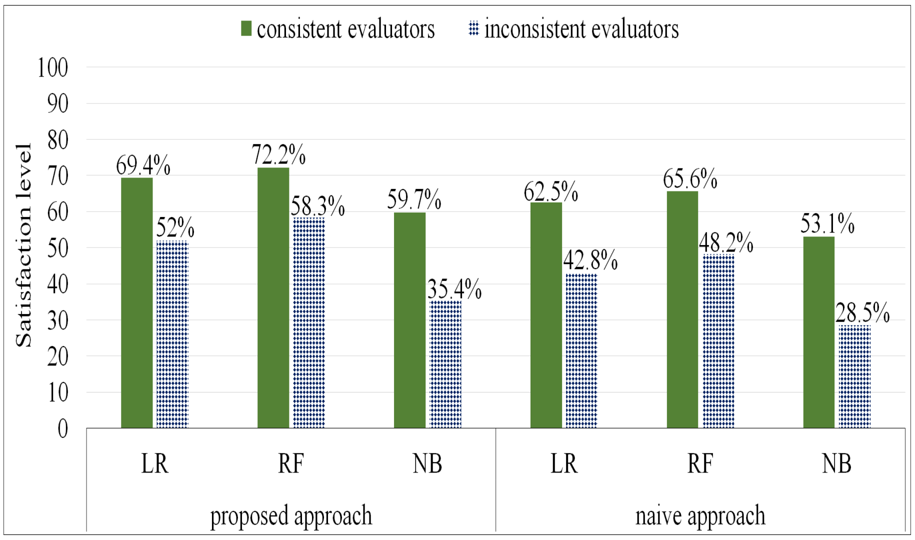


Figure 4.9: Satisfaction level of consistent and inconsistent evaluators

preferences that evaluators have labeled during the training phase. Then, in the testing phase, the web application shows these privacy preferences again to them, and collect the label (i.e., satisfaction level) they assign. Based on this, we can judge whether the evaluator is consistent or not in his/her decisions. We consider that an evaluator is consistent if any two out of three decisions taken during the training and the testing phase match. Figure 4.9 presents the comparative analysis of the satisfaction level of consistent and inconsistent evaluators for both the approaches. It can be seen that, the satisfaction level of consistent evaluators is greater than the satisfaction level of inconsistent evaluators. However, even in the worst case, about 35.4% of inconsistent evaluators are satisfied with the decisions by the proposed approach, whereas, it is only 28.5% for the naive approach.

			LR		RF		NB	
			Yes	No	Yes	No	Yes	No
Training dataset	Proposed approach	Precision	98.1%	96.6%	96.1%	99.5%	83.9%	75.5%
		Recall	99%	90.7%	96.4%	97%	83.8%	64.1%
		F1-score	98%	92.4%	97%	97.8%	81%	66.9%
	Naive approach	Precision	93.4%	93.2%	99%	98%	79.2%	60.6%
		Recall	95.1%	87.2%	98%	97%	75.1%	68.3%
		F1-score	94.2%	89.3%	98%	97%	76.9%	63.2%
Testing dataset	Proposed approach	Precision	91.7%	73.3%	92.3%	68.9%	79.93%	58.8%
		Recall	96.6%	68.9%	95.3%	68%	83.33%	61.4%
		F1-score	93%	70.6%	94%	68.2%	80.56%	58.13%
	Naive approach	Precision	76.4%	52.8%	79.2%	57.6%	73.2%	55.1%
		Recall	80.7%	46.4%	81.8%	57.8%	71%	54.3%
		F1-score	77.5%	46.9%	79.8%	56.3%	70.6%	51.6%

Table 4.3: Performance comparison of different learning algorithms for the training and testing datasets

However, from the above experimental results, it is clear that the proposed approach provides better performance in terms of accuracy and satisfaction level. Besides, these results let us think that this is a good direction to study and there is room for improvement. By doing more experiments, we need to understand why users are not satisfied with the system-generated suggestions and how we can increase their satisfaction level.

4.3 Chapter summary

In this chapter, we have extended the approach presented in Chapter 3 by considering users' contextual information. More particularly, we have proposed an ML-based privacy preferences management service for helping users to manage their context-based privacy preferences. To show the feasibility of the proposed approach, we have compared the proposed approach with a naive approach, by using three different ML algorithms. The experimental results show that the proposed approach provides better performance than the naive approach.

5 ML-based Spam Accounts Detection on Twitter

Twitter, a microblogging service launched in 2006, is one of the most popular online social network platform, where users post messages of around 280 characters, known as *"tweet"*. It has been reported that Twitter has over 330 million monthly active users as of 2019, that posted over 500 million tweets every single day [66]. However, due to the huge popularity of Twitter, it also attracts the interest of cybercriminals [14, 15]. Attackers exploit the implicit trust relationships between users in order to achieve their malicious aims. In previous, for spamming detection individuals involvement was needed to define some rules and based on those rules human expertise annotated the sample data. However, defining rules and annotating samples is an exhausting and time-consuming task. The ML technology bears the potential to considerably cut down on manual efforts and unloads tasks from the humans. This advantage led us to exploit ML technology to solve the problem of human-based spam detection on social networks. Therefore, in this chapter, we tried to exploit ML approach to train the spam filters automatically for identifying and removing spam accounts from the Twitter platform.

To detect spammers on Twitter, we have considered both graph-based features (i.e., triangle count of user's network, the ratio of triangle count to the number of followers of a user, and the ratio of bi-directional links) and content-based features (i.e., unique URL ratio, URL to tweet ratio, average tweets per day of a user, and average likes per tweet of a user). To assess our detection method, we selected, from the popular social honeypot dataset [38], 325 Twitter accounts, where 168 are considered legitimate users and 157 spam users. We have used several ML classification algorithms for distinguishing between spammers and non-spammers accounts. Through the experiments, we show that the proposed set of feature gives better performance than existing state of the art approaches. Moreover, our results show that, among all considered classifiers, Random Forest (RF) classifier gives the better performance. By using this classifier, our suggested features can achieve 92% precision and 91% F1-score.

In summary, this chapter provides the following main contributions:

- we design a set of novel graph-based and content-based features that have been proved to be powerful for spam account detection on Twitter.

- we use seven ML algorithms, namely: k-Nearest Neighbor (k-NN), Decision Tree (DT), Naive Bayesian (NB), Random Forest (RF), Logistic Regression (LR), Support Vector Machine (SVM), and eXtreme Gradient Boosting (XGBoost) to classify spam and legitimate users.

- we compare the proposed set of features with [6], [7], [8], and [67] showing that our features provide better accuracy.

- by using the feature ranking method (i.e., information gain), we ranked the top 10 most influencing detection features among all the features used by state of the art approaches.

5.1 Proposed features

In this section, we present the proposed set of features, consisting of three graph-based and four content-based features.

5.1.1 Graph-based features

Twitter allows users to build their own social graph. A social graph represents the following and follower relationships among users. From the social graph of a target user u, we extract three graph-based features: (1) triangle count of user u's network, (2) the ratio of triangle count to number of followers of u, and (3) the ratio of bi-directional links from the users' social graph.

Triangle count of user u's network: We compute the total number of triangles of u's network, *(Triangle_Count(u))*. In the social network, a triangle exists if a user/node has two adjacent nodes which, in turn, are also adjacent to each other. To find out spammer and non-spammer users, we make use of this feature because legitimate users usually follow accounts whose owners are their close friends, colleagues, or family [12, 68], as such these accounts are likely to have a relationship together. Therefore, an high number of triangles implies that the user is legitimate. On the other hand, spammers usually blindly follow other accounts, these accounts do not know each other and have a lower relationship among them. Thus, compared to legitimate users, spam users will have a smaller number of triangles.

The ratio of u's triangles to number of u's followers: To evade the triangle count feature, spammers could create many fake accounts so as to form triangles, by building links among these fake accounts. Moreover, spammers can purchase followers, thus sometimes the number of followers of spammers is greater than the one of legitimate users. Thus, this feature ($RateTNF(u) = \frac{Triangle\_Count(u)}{N_{fer}(u)}$), where, $N_{fer}(u)$ refers to the number of followers of user u, can help us to detect spam users, even if spam users generate fake triangles in their social networks.

The ratio of bi-directional links of u: When any two users' accounts follow each other, we refer to this as a bi-directional link. The number of bi-directional links of an account reflects the reciprocity between an account holder and its followings. In general, spammers follow huge amount of legitimate users, but they cannot force them to follow back, thus their number of bi-directional links will probably be low. On the other hand, legitimate users usually follow their family members, friends, and co-workers who will follow them back. It means that the number of bi-directional links of legitimate users will be high. So, we can use this feature for distinguishing between spammers and legitimate users. We define ratio of bi-directional links as follows: $Rate_{bilink}(u) = \frac{N_{bilink}(u)}{N_{fer}(u)+N_{fing}(u)}$, where $N_{bilink}(u)$ refers to the number of followings of a user u which follow him back, whereas $N_{fer}(u)$ is the number of followers of user u, and $N_{fing}(u)$ is the number of followings of user u.

5.1.2 Content-based features

These features are properties related to text of tweets. In previous work, many researchers used content-based features i.e., duplicate tweets, suspicious words, repeated words, tweet time patterns, for detecting spammers. Thus, we consider four new content-based features that can isolate spammers from non-spammers, namely: 1) Unique URL ratio, 2) URL to tweet ratio, 3) Average tweets per day of user u, and 4) Average likes per tweet of user u.

Unique URL ratio: A way to gain money from spammer activities is to force legitimate users to visit a particular site. As such, spammers post the same URL several times. $URate_{url}(u)$ is the ratio of the number of unique URLs posted by user u to the number of total URLs posted by him/her. A higher $URate_{url}(u)$ means that user u is a legitimate user. Similarly, the lower $URate_{url}(u)$ is, the higher is the chance of being a spammer account. We define unique URL ratio as follows : $URate_{url}(u) = \frac{N_{unique\_URLs}(u)}{N_{all\_URLs}(u)}$.

URL to tweets ratio: Spammers post a huge number of URLs compared to legitimate users. This feature $Rate_{url\_tweet}(u)$ defines the ratio of number of URLs posted by a user u to the number of tweets posted by him/her. A high value of this feature means that user u is a spam user. Likewise, a lower value means a higher chance of being a legitimate user. URL to tweets ratio is defined as follows : $Rate_{url\_tweet}(u) = \frac{N_{all\_URLs}(u)}{N_{tweets}(u)}$.

Average tweets per day of u: This feature refers to the ratio of the number of tweets posted by a user u to the age of an account (days). More precisely, $Avg_{tweet}(u) = \frac{N_{tweets}(u)}{Age(u)}$. For making money or spreading fake news, spammers tend to post more tweets than legitimate users. Thus, a higher value of $Avg_{tweet}(u)$ means that user u is likely to be a spam user, whereas a lower value of $Avg_{tweet}(u)$ means an higher chance of being a legitimate user.

Average likes per tweet of u: This feature defines the ratio of the number of likes of user u's tweets over the number of tweets posted by u. It is expressed by $Avg_{likes}(u) = \frac{N_{likes}(u)}{N_{tweets}(u)}$. Since, spam users do not get more likes for their tweets, so a high value of $Avg_{likes}(u)$ means user u is a legitimate user, whereas a lower value means user u is a spam user.

5.2 Experiments and results

In this section, we present the results of experimental carried out to show the effectiveness of the proposed set of features.

5.2.1 Data collection

In order to build our model, we need a dataset of Twitter users classified as spammers and legitimate users. For this reason, we used the Twitter Social Honeypot dataset [38] in which users have been already classified as spammers and legitimate users based on tweet content, user behavioral and topological features. The authors created and manipulated 60 social honeypot accounts on Twitter to attract spammers. Thereafter, they used Expectation-Maximization (EM) clustering algorithm and then manually grouped their harvested users into spammers and legitimate users. The dataset consists of $41,499$ user accounts, with pre-classified accounts of $22,223$ spammers and $19,276$ legitimate users that were captured during an eight month period in 2010. In this dataset, most of the users do not have their list of followings/followers; hence values of their interaction and novel graph-based features (i.e., triangle count, bi-directional links) will be zero, which forces classifiers to be biased for spamming detection. Therefore, we consider only those users who have a complete list of followers and followings. Moreover, we also excluded those users who posted tweets in non-English language. We randomly selected 325 seed Twitter accounts including 168 legitimate users and 157 spammer users' profile from this dataset. Since the dataset is quite old, we have manually checked all the collected dataset (i.e., 325 users account) and found that all data are still labeled correctly. To extract the followers and followings relationship among these seed users, a web crawler was developed based on Twitter API [69]. In addition, we collected 20 most recent tweets with the URLs. The basic characteristics of the dataset are shown in Table 5.1.

Property	**Value**
Number of Twitter accounts	325
Number of legitimate accounts	168
Number of spammer accounts	157
Number of followings	21676
Number of followers	5039
Average number of followings	66
Average number of followers	15
Number of tweets	6500
Number of extracted URLs	2506
Number of unique URLs extracted	1346
Number of triangles	5037

Table 5.1: Characteristics of the dataset

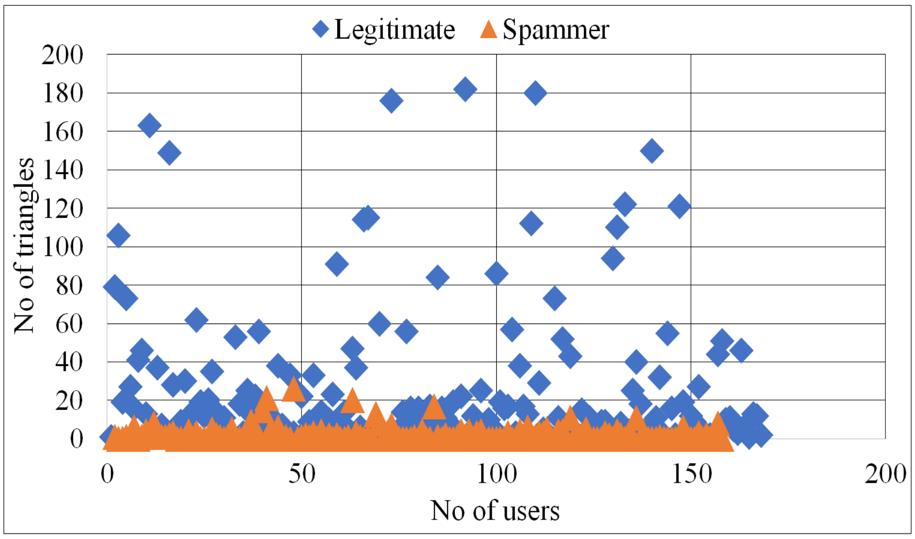


Figure 5.1: Number of triangles

5.2.2 Evaluation metrics

In the evaluation, we consider the confusion matrix illustrated in Table 5.2, where a means the number of spammers that have been correctly classified, b represents the number of spammers which are misclassified as non-spammers, c expresses the number of non spammers which are misclassified as spammers, and d refers to the number of non-spammers that have been correctly classified. We used four widely adopted machine learning metrics, that is: accuracy, precision, recall, and F1-score.

Accuracy (A) is ratio of the total number of correctly classified instances of both classes over the total number of all instances in the dataset and is expressed by: $A = \frac{(a+d)}{(a+b+c+d)}$. Precision (P) refers to the ratio of the number of correctly classified instances to the total number of instances and is expressed by: $P = \frac{a}{(a+c)}$. Recall (R) defines the ratio of the number of instances correctly classified to the total number of predicted instances and is expressed by: $R = \frac{a}{(a+b)}$. Finally, F1-score (F1) is measured as a weighted average of the precision and recall, and is defined as: $F1 = \frac{2P*R}{(P+R)}$.

		Predicted class	
		Spammer	Non-spammer
True class	Spammer	a	b
	Non-spammer	c	d

Table 5.2: Confusion matrix

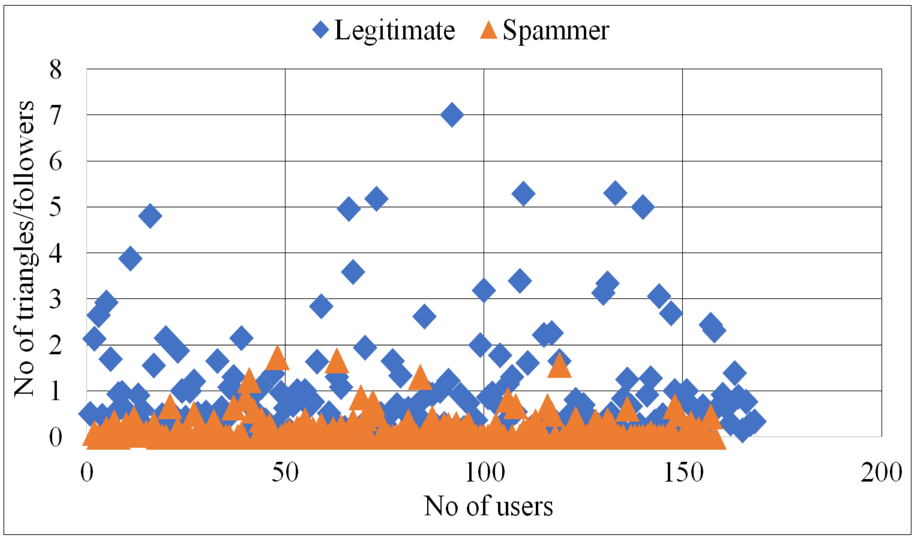


Figure 5.2: Number of triangles/followers

5.2.3 Data analysis

In this subsection, we analyze the collected dataset. As we can see from Fig. 5.1, showing the characteristics of graph-based features, the number of triangles for legitimate users is higher than those for spammers. Likewise, from Fig. 5.2, we see that the ratio of the number of triangles to number of followers for legitimate users is higher than for spammer users. Fig. 5.3 shows that the number of bi-directional links of each account which reflects reciprocity between user accounts is higher for legitimate users than for spammer users.

Fig. 5.4 - 5.5 show the differences between the considered content-based features of spammers and legitimate users. From Fig. 5.4, we see that legitimate users tend to

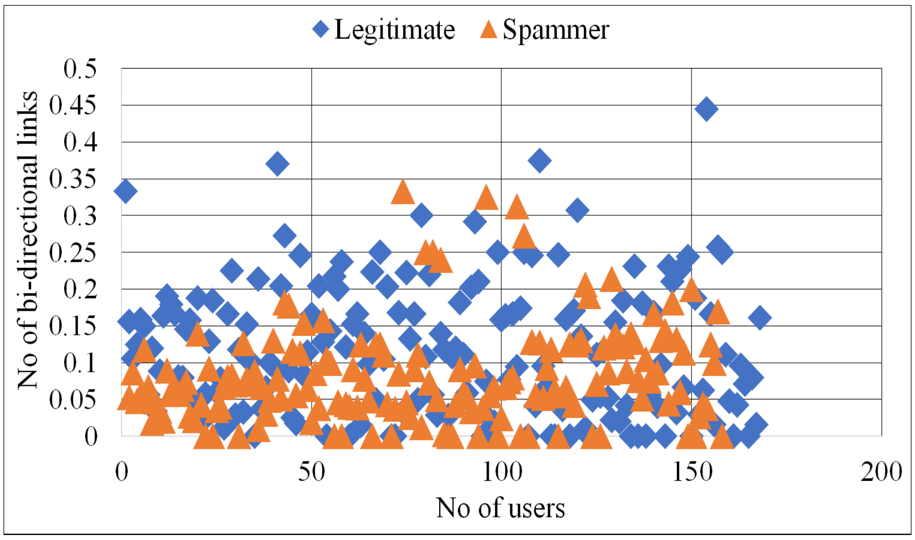


Figure 5.3: Number of bi-directional links

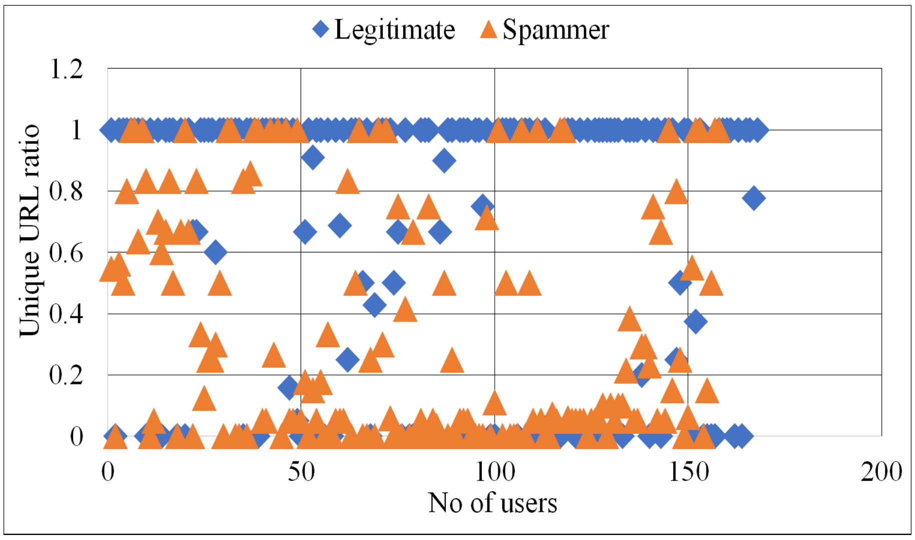


Figure 5.4: Unique URL ratio

post a unique link in their tweets. As we expected, spammer users tend to post more links in their tweets than legitimate users, but from Fig. 5.5, we see that the number of unique links does is almost the same for non-spammers and spammer users.

5.2.4 Evaluation

We evaluate our proposed approach through performance comparison and feature ranking, by using different machine learning tools.

Performance comparison: In this experiment, we compare the performance of our approach (E) with four existing state of the art approaches, namely: (A) [6], which

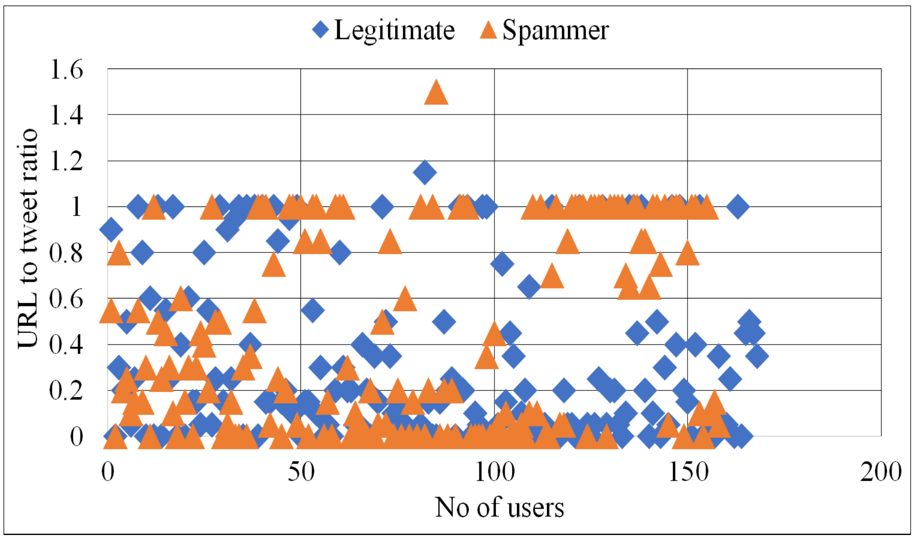


Figure 5.5: URL to tweets ratio

used 12 features; (B) [7], which used 10 features; (C) [8], which used 10 features; and (D) [67], which used 17 features.

We selected these four approaches for comparison because these are the latest published state of the art approaches for spam detection on Twitter, and it was possible to extract all of the features they considered from our dataset. We conducted our evaluation by using seven different ML classifiers, namely: k-NN, DT, NB, RF, LR, SVM, and XGBoost (see Section 2.2.1). For each ML classifier, we compute four performance metrics: accuracy, precision, recall, and F1- score.

As shown in Fig. 5.6 - 5.9, our proposed approach outperforms every considered approaches. More particularly, from Fig. 5.6, we can see that the accuracy of our approach (i.e., RF of E) is greater than the others. It reaches highest accuracy of 91% for RF classifier and lowest accuracy of 74% for NB. Likewise, from Fig. 5.7, we can see that the precision of our approach is greater than the other approaches. It achieves highest precision value of 92% for both RF and XGBoost classifiers. Especially, under the NB the precision value of our approach is 0.04% lower than approach D and 0.01% lower than approach A. Similarly, for SVM the precision value of our approach is 0.08% lower than approach C. On the other hand, we can see that the precision of the other five ML classifiers are the highest. In the same way, from Fig. 5.8, we can see that the recall value of E is lower than approach C and D under SVM and Naive Bayes respectively, whereas the recall value of the other five ML classifiers are the highest.

From Fig. 5.9, we can see that the F1-score of our work under all ML classifiers is also the highest. More particularly, the highest F1-score of our approach is 91% (RF in E), and the lowest F1-score of our approach is 74% (NB in E). From the above discussion, we see that our new feature set is more effective to detect Twitter spammers than other existing approaches.

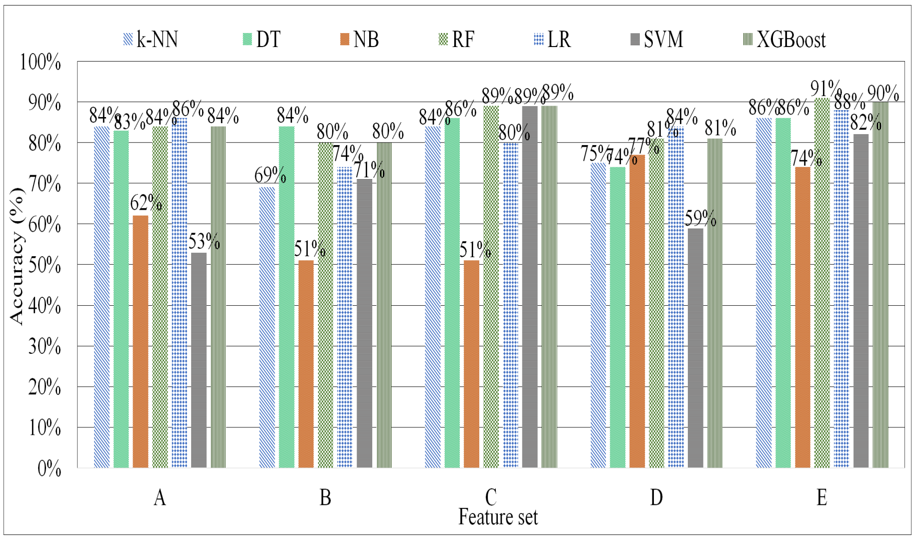


Figure 5.6: Accuracy

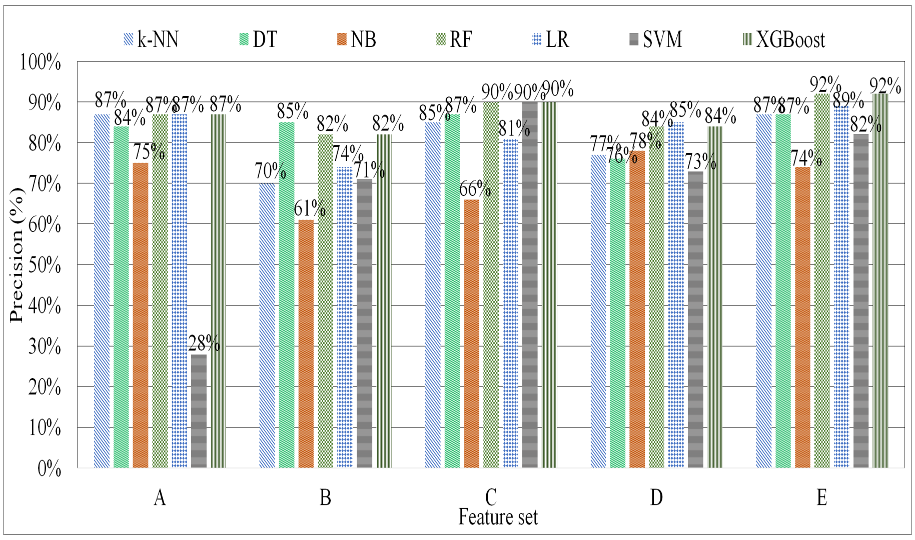


Figure 5.7: Precision

Feature ranking: In order to verify the importance of the considered features, we used feature selection method. It is also known as variable selection or attribute selection, which is the process of selecting relevant features in terms of the target learning problem. The purpose of feature selection is to remove redundant and irrelevant features because these features can reduce the learning accuracy and the quality of the model. However, we used information gain feature selection method, that are available on Weka [70]. Weka supports feature selection via information gain using the $Info\_Gain\_Attribute\_Eval$ attribute evaluator. It calculates the information gain (i.e., entropy) for each attribute. This value varies from 0 (i.e., no information) to 1 (i.e., maximum information). The attributes that contribute more information will have a higher information gain value and can be selected, whereas

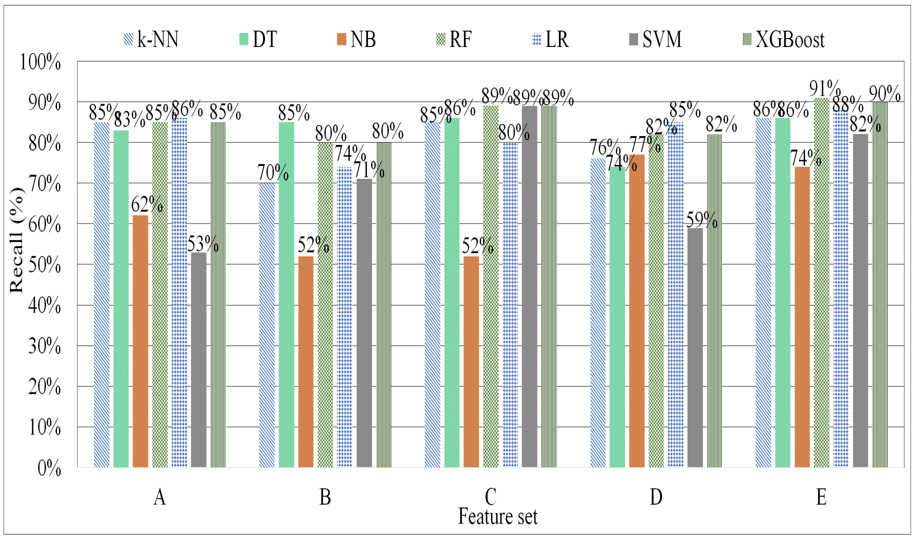


Figure 5.8: Recall

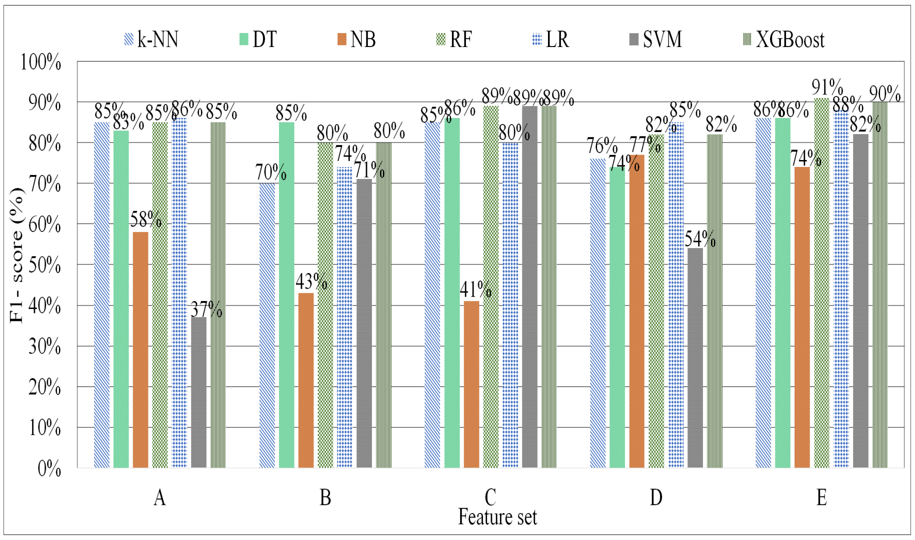


Figure 5.9: F1-score

those that do not add much information will have a lower score and can be removed. Information gain can be calculated as follows: $IG(C, P_i) = H(C) - H(C|P_i)$, where C is the output class, P_i and H is the entropy.

The result listed in Table 5.3 indicates the top most 10 important attributes among 55 features. Interestingly, we see that 5 of our features are included among the top 10 important features. The first and third most important attributes in the list are the number of triangles and the number of triangles to number of followers.

Furthermore, we verify the importance of the top 10 features, by measuring the F1-score. We calculated F1-score considering: (1) all top 10 attributes, which we labeled as X, (2) 5 of our features that are included among the top 10 ones, which we labeled as Y, and (3) all of our 7 features, which labeled as Z. From Fig. 5.10, we

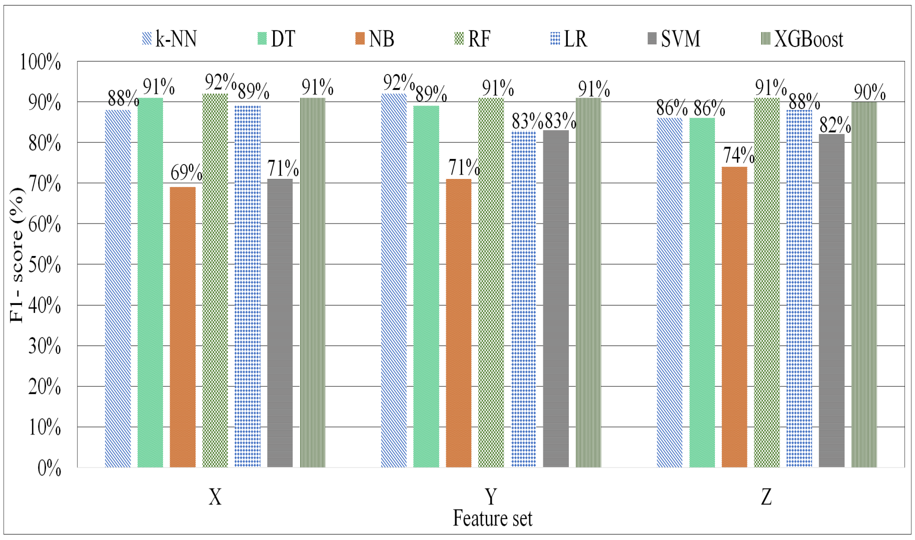


Figure 5.10: F1-score

Rank	Information gain
1	Number of triangles
2	Age of an account (days)
3	Number of triangles to number of followers
4	Number of followers
5	Number of tweets
6	Average tweets per day
7	Unique URL ratio
8	Average likes per tweet
9	Reputation
10	Fifo ratio

Table 5.3: Top 10 features

can see that the highest F1-score is 92% for RF, when we consider all top 10 features (RF in X), whereas we get the highest F1-score of 92% for k-NN classifiers on Y, and 91% for both RF and XGBoost. For Z, we get the highest F1-score of 91% for RF, and the lowest F1-score is around 74% (NB in Z).

According to the experimental results, it can be observed that the top features identified by the feature selection method (i.e., information gain) gives slightly improved performance. We conclude that the features identified by information gain, which are number of triangles, age of an account (days), and number of triangles to number of followers, are very important and higher influencing features in the process of identifying spam users on Twitter.

5.3 Chapter summary

In this chapter, we have designed a new and more robust set of features to detect spammers on Twitter. We have considered both graph-based and tweet content-based features, and applied them into seven different ML algorithms. In the experiment, Random Forest (RF) gives the better results compared to other algorithms, with an accuracy of 91%, precision 92%, and F1-score 91%. Through the performance comparison analysis, we showed that our proposed solution is feasible and is capable to give better results than other existing state of the art approaches.

6 Conclusion and Future work

This thesis focused on enhancing individuals' data privacy and security using machine learning technologies. Machine learning has very good prediction capabilities, and previous research has shown that average users are not skilled in defining their privacy preferences effectively. Therefore, in this thesis, we attempt to predict the best privacy settings for users with ML technology. In Chapter 3, we have described the proposed *soft privacy matching mechanism*, which is able to relax, in a controlled way, some conditions of users' privacy preferences to match service providers' privacy policies. We have considered different learning approaches to test which one performs better in the IoT-based smart environment. We have tested our approaches extensively using evaluators enrolled from the university students and through a crowd-sourcing platform and achieved promising results. In Chapter 4, we have extended the approach presented in Chapter 3 by considering users' contextual data. In this chapter, we have presented a privacy preferences management service that helps users to manage their privacy preferences settings in different contexts. More precisely, we have focused on users' contextual information and defined a learning approach exploiting contextual features to learn users' privacy preferences. The results have shown that contextual data are essential for setting users' privacy preferences. We have also exploited the benefits of ML (e.g., the elimination or reduction of the necessity for human activity) to redesign the security mechanisms in the social media environment (e.g., spam detection) and make it more secure for individuals. In Chapter 5, we have investigated the behavior of spam users on Twitter with the goal of improving existing spam-detection mechanisms. We have presented a method to classify Twitter users based on several new features and shown that the proposed technique can be extremely effective and more robust than existing spam-detection methods. More specifically, we have used ML to train the spam filters to address issues of violence and aggression that have been increasing in social media environments by identifying and removing violent, insulting, aggressive, and harassing content creators from the social media environments.

In the future, we plan to extend our work in several directions. The first direction will be to implement a prototype of our proposed approach and test it in various real-world scenarios. Moreover, we plan to extend the approach discussed in Chapter 3 by analyzing how to address the uncertainty that the "maybe" label introduces to the classifier and examining whether a binary classifier (i.e., using "yes" and "no" labels) could improve the satisfaction level of the proposed approach. Next, to develop the approach discussed in Chapter 4, we plan to investigate other ontology-based distance measures [71] to assess how they might influence the context selection process. Besides, we will investigate other states of the art approaches to compare with the proposed approach. In addition, we will conduct more experiments and user studies to examine why users are dissatisfied with the system-generated suggestions and how we can increase their satisfaction levels. Finally, our future plan is to extend the approach presented in Chapter 5 by proposing a more effective model to easily classify various types of spammers within various social networks such as Facebook and LinkedIn.

Bibliography

[1] Samuel, Arthur L. “Some Studies in Machine Learning Using the Game of Checkers. II—Recent Progress”. In: *Computer Games I*. Springer, 1988, pp. 366–400 (cit. on p. 15).

[2] Cilimkovic, Mirza. “ Neural networks and back propagation algorithm ”. In: *Institute of Technology Blanchardstown, Blanchardstown Road North Dublin* 15 (2015) (cit. on p. 15).

[3] *Advantages of Machine Learning Technology*. https://data-flair.training/blogs/advantages-and-disadvantages-of-machine-learning/ (cit. on p. 15).

[4] Acquisti, Alessandro and Brandimarte, Laura and Loewenstein, George. “Privacy and human behavior in the age of information”. In: *Science* 347.6221 (2015), pp. 509–514 (cit. on p. 15).

[5] Solove, Daniel J. “Introduction: Privacy Self-Management and the Consent Dilemma”. In: *Harvard Law Review* 126 (2013), p. 1880 (cit. on p. 15).

[6] Ameen, Aso Khaleel and Kaya, Buket. “Detecting Spammers in Twitter Network”. In: *International Journal of Applied Mathematics, Electronics and Computers* 5.4 (2017), pp. 71–75 (cit. on pp. 16, 27, 31, 32, 66, 71).

[7] Ala’M, Al-Zoubi and Faris, Hossam and others. “Spam profile detection in social networks based on public features”. In: *Information and Communication Systems (ICICS), 2017 8th International Conference on*. IEEE. 2017, pp. 130–135 (cit. on pp. 16, 27, 31, 32, 66, 72).

[8] Singh, Monika and Bansal, Divya and Sofat, Sanjeev. “Who is Who on Twitter–Spammer, Fake or Compromised Account? A Tool to Reveal True Identity in Real-Time”. In: *Cybernetics and Systems* (2018), pp. 1–25 (cit. on pp. 16, 27, 31, 66, 72).

[9] Kim, Seung-Hyun and Ko, Han-Gyu and Ko, In-Young and Choi, Daeseon. “ Effects of Contextual Properties on Users’ Privacy Preferences in Mobile Computing Environments ”. In: *2015 IEEE Trustcom/BigDataSE/ISPA*. Vol. 1. IEEE. 2015, pp. 507–514 (cit. on p. 17).

[10] Saleh, Rafiy and Jutla, Dawn and Bodorik, Peter. “Management of Users’ Privacy Preferences in Context”. In: *2007 IEEE International Conference on Information Reuse and Integration*. IEEE. 2007, pp. 91–97 (cit. on p. 17).

[11] Kapitsaki, Georgia M. “ Reflecting user privacy preferences in context-aware web services”. In: *2013 IEEE 20th International Conference on Web Services*. IEEE. 2013, pp. 123–130 (cit. on p. 17).

[12] Yang, Chao and Harkreader, Robert and Gu, Guofei. “Empirical evaluation and new design for fighting evolving twitter spammers”. In: *IEEE Transactions on Information Forensics and Security* 8.8 (2013), pp. 1280–1293 (cit. on pp. 18, 66).

[13] Fazil, Mohd and Abulaish, Muhammad. “A Hybrid Approach for Detecting Automated Spammers in Twitter”. In: *IEEE Transactions on Information Forensics and Security* 13.11 (2018), pp. 2707–2719 (cit. on pp. 18, 27, 31).

[14] Benevenuto, Fabricio and Magno, Gabriel and Rodrigues, Tiago and Almeida, Virgilio. “Detecting spammers on twitter”. In: *Collaboration, electronic messaging, anti-abuse and spam conference (CEAS)*. Vol. 6. 2010. 2010, p. 12 (cit. on pp. 18, 27, 31, 32, 65).

[15] Grier, Chris and Thomas, Kurt and Paxson, Vern and Zhang, Michael. “@ spam: the underground on 140 characters or less”. In: *Proceedings of the 17th ACM conference on Computer and communications security*. ACM. 2010, pp. 27–37 (cit. on pp. 18, 65).

[16] Wu, Chih-Hung. “ Behavior-based spam detection using a hybrid method of rule-based techniques and neural networks ”. In: *Expert Systems with Applications* 36.3 (2009), pp. 4321–4330 (cit. on p. 18).

[17] Basnet, Ram B and Sung, Andrew H and Liu, Quingzhong. “ Rule-based phishing attack detection ”. In: *Proceedings of the International Conference on Security and Management (SAM)*. The Steering Committee of The World Congress in Computer Science, Computer . . . 2011, p. 1 (cit. on p. 18).

[18] Lee, Hosub and Kobsa, Alfred. “Privacy preference modeling and prediction in a simulated campuswide IoT environment”. In: *Pervasive Computing and Communications (PerCom), 2017 IEEE International Conference on*. IEEE. 2017, pp. 276–285 (cit. on p. 23).

[19] Nakamura, Toru and Kiyomoto, Shinsaku and Tesfay, Welderufael B and Serna, Jetzabel. “Easing the Burden of Setting Privacy Preferences: A Machine Learning Approach”. In: *International Conference on Information Systems Security and Privacy*. Springer. 2016, pp. 44–63 (cit. on pp. 23, 30).

[20] Singh, Bikash Chandra and Carminati, Barbara and Ferrari, Elena. “Learning privacy habits of PDS owners”. In: *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE. 2017, pp. 151–161 (cit. on pp. 23, 49).

[21] Lipford, Heather Richter and Besmer, Andrew and Watson, Jason. “Understanding Privacy Settings in Facebook with an Audience View.” In: *UPSEC* 8 (2008), pp. 1–8 (cit. on pp. 23, 30).

[22] Sadeh, Norman and Hong, Jason and Cranor, Lorrie and Fette, Ian and Kelley, Patrick and Prabaker, Madhu and Rao, Jinghai. “Understanding and capturing people’s privacy policies in a mobile social networking application”. In: *Personal and Ubiquitous Computing* 13.6 (2009), pp. 401–412 (cit. on pp. 23, 30, 49).

[23] Fang, Lujun and LeFevre, Kristen. “Privacy wizards for social networking sites”. In: *Proceedings of the 19th international conference on World wide web*. ACM. 2010, pp. 351–360 (cit. on p. 24).

[24] Bilogrevic, Igor and Huguenin, Kévin and Agir, Berker and Jadliwala, Murtuza and Hubaux, Jean-Pierre. “Adaptive information-sharing for privacy-aware mobile social networks”. In: *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*. ACM. 2013, pp. 657–666 (cit. on p. 24).

[25] Wijesekera, Primal and Reardon, Joel and Reyes, Irwin and Tsai, Lynn and Chen, Jung-Wei and Good, Nathan and Wagner, David and Beznosov, Konstantin and Egelman, Serge. “Contextualizing privacy decisions for better prediction (and protection)”. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM. 2018, p. 268 (cit. on p. 24).

[26] Liu, Bin and Andersen, Mads Schaarup and Schaub, Florian and Almuhimedi, Hazim and Zhang, S Aerin and Sadeh, Norman and Agarwal, Y and Acquisti, A. “Follow my recommendations: A personalized privacy assistant for mobile app permissions”. In: *Symposium on Usable Privacy and Security*. 2016 (cit. on p. 24).

[27] Smith, Ian and Consolvo, Sunny and Lamarca, Anthony and Hightower, Jeffrey and Scott, James and Sohn, Timothy and Hughes, Jeff and Iachello, Giovanni and Abowd, Gregory D. “Social disclosure of place: From location technology to communication practices”. In: *International Conference on Pervasive Computing*. Springer. 2005, pp. 134–151 (cit. on pp. 24, 49).

[28] Wiese, Jason and Kelley, Patrick Gage and Cranor, Lorrie Faith and Dabbish, Laura and Hong, Jason I and Zimmerman, John. “Are you close with me? are you nearby?: investigating social groups, closeness, and willingness to share”. In: *Proceedings of the 13th international conference on Ubiquitous computing*. ACM. 2011, pp. 197–206 (cit. on p. 24).

[29] Harkous, Hamza and Rahman, Rameez and Aberer, Karl. “C3p: Context-aware crowd-sourced cloud privacy”. In: *International Symposium on Privacy Enhancing Technologies Symposium*. Springer. 2014, pp. 102–122 (cit. on p. 24).

[30] Liang, Tingting and He, Lifang and Lu, Chun-Ta and Chen, Liang and Philip, S Yu and Wu, Jian. “A broad learning approach for context-aware mobile application recommendation”. In: *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2017, pp. 955–960 (cit. on p. 25).

[31] Xie, Jierui and Knijnenburg, Bart Piet and Jin, Hongxia. “Location sharing privacy preference: analysis and personalized recommendation”. In: *Proceedings of the 19th international conference on Intelligent User Interfaces*. ACM. 2014, pp. 189–198 (cit. on p. 25).

[32] Yuan, Lin and Theytaz, Joël and Ebrahimi, Touradj. “Context-dependent privacy-aware photo sharing based on machine learning”. In: *IFIP International Conference on ICT Systems Security and Privacy Protection*. Springer. 2017, pp. 93–107 (cit. on p. 25).

[33] Bilogrevic, Igor and Huguenin, Kévin and Agir, Berker and Jadliwala, Murtuza and Gazaki, Maria and Hubaux, Jean-Pierre. “A machine-learning based approach to privacy-aware information-sharing in mobile social networks”. In: *Pervasive and Mobile Computing* 25 (2016), pp. 125–142 (cit. on pp. 25, 30).

[34] Toch, Eran. “Crowdsourcing privacy preferences in context-aware applications”. In: *Personal and ubiquitous computing* 18.1 (2014), pp. 129–141 (cit. on p. 26).

[35] Bigwood, Greg and Abdesslem, F Ben and Henderson, Tristan. “Predicting location-sharing privacy preferences in social network applications”. In: *Proc. of AwareCast* 12 (2012), pp. 1–12 (cit. on p. 26).

[36] Schlegel, Roman and Kapadia, Apu and Lee, Adam J. “Eyeing your exposure: quantifying and controlling information sharing for improved privacy”. In: *Proceedings of the Seventh Symposium on Usable Privacy and Security*. ACM. 2011, p. 14 (cit. on p. 26).

[37] Gao, Hongyu and Chen, Yan and Lee, Kathy and Palsetia, Diana and Choudhary, Alok N. "Towards Online Spam Filtering in Social Networks." In: *NDSS*. Vol. 12. 2012, pp. 1–16 (cit. on pp. 27, 31).

[38] Lee, Kyumin and Caverlee, James and Webb, Steve. "Uncovering social spammers: social honeypots+ machine learning". In: *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2010, pp. 435–442 (cit. on pp. 27, 31, 65, 68).

[39] Chen, Chao and Zhang, Jun and Chen, Xiao and Xiang, Yang and Zhou, Wanlei. "6 million spam tweets: A large ground truth for timely Twitter spam detection". In: *Communications (ICC), 2015 IEEE International Conference on*. IEEE. 2015, pp. 7065–7070 (cit. on pp. 27, 31).

[40] Wang, Bo and Zubiaga, Arkaitz and Liakata, Maria and Procter, Rob. "Making the most of tweet-inherent features for social spam detection on twitter". In: *arXiv preprint arXiv:1503.07405* (2015) (cit. on pp. 27, 31).

[41] Wang, Alex Hai. "Don't follow me: Spam detection in twitter". In: *Security and cryptography (SECRYPT), proceedings of the 2010 international conference on*. IEEE. 2010, pp. 1–10 (cit. on pp. 27, 31).

[42] Kotsiantis, Sotiris B. "Supervised machine learning: A review of classification techniques". In: (2007) (cit. on pp. 28, 29, 33, 40, 50, 56).

[43] Jadhav, Sayali D and Channe, HP. "Comparative Study of K-NN, Naive Bayes and Decision Tree Classification Techniques". In: *International Journal of Science and Research* 5.1 (2016) (cit. on p. 29).

[44] Wu, Yingquan and Ianakiev, Krassimir and Govindaraju, Venu. " Improved k-nearest neighbor classification ". In: *Pattern recognition* 35.10 (2002), pp. 2311–2318 (cit. on p. 29).

[45] Bhargava, Neeraj and Sharma, Girja and Bhargava, Ritu and Mathuria, Manish. " Decision tree analysis on j48 algorithm for data mining ". In: *Proceedings of International Journal of Advanced Research in Computer Science and Software Engineering* 3.6 (2013) (cit. on p. 29).

[46] Robnik-Šikonja, Marko. " Improving random forests ". In: *European conference on machine learning*. Springer. 2004, pp. 359–370 (cit. on p. 29).

[47] Dreiseitl, Stephan and Ohno-Machado, Lucila. " Logistic regression and artificial neural network classification models: a methodology review ". In: *Journal of biomedical informatics* 35.5-6 (2002), pp. 352–359 (cit. on p. 30).

[48] Gunn, Steve R and others. " Support vector machines for classification and regression ". In: *ISIS technical report* 14.1 (1998), pp. 5–16 (cit. on p. 30).

[49] Chen, Tianqi and Guestrin, Carlos. " Xgboost: A scalable tree boosting system ". In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM. 2016, pp. 785–794 (cit. on p. 30).

[50] Albertini, Davide Alberto and Carminati, Barbara and Ferrari, Elena. " Privacy Settings Recommender for Online Social Network ". In: *2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC)*. IEEE. 2016, pp. 514–521 (cit. on p. 30).

[51] Pearson, Siani. “Taking account of privacy when designing cloud computing services”. In: *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*. IEEE Computer Society. 2009, pp. 44–52 (cit. on p. 34).

[52] Singh, Bikash Chandra and Carminati, Barbara and Ferrari, Elena. “A risk-benefit driven architecture for personal data release”. In: *2016 IEEE 17th International Conference on Information Reuse and Integration (IRI)*. IEEE. 2016, pp. 40–49 (cit. on p. 34).

[53] Byun, Ji-Won and Li, Ninghui. “Purpose based access control for privacy protection in relational database systems”. In: *The VLDB Journal—The International Journal on Very Large Data Bases* 17.4 (2008), pp. 603–619 (cit. on p. 36).

[54] Wu, Zhibiao and Palmer, Martha. “Verbs semantics and lexical selection”. In: *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics. 1994, pp. 133–138 (cit. on pp. 37, 53).

[55] Niwattanakul, Suphakit and Singthongchai, Jatsada and Naenudorn, Ekkachai and Wanapu, Supachanun. “Using of Jaccard coefficient for keywords similarity”. In: *Proceedings of the International MultiConference of Engineers and Computer Scientists*. Vol. 1. 6. 2013 (cit. on p. 37).

[56] Xu, Zeshui and Xia, Meimei. “Distance and similarity measures for hesitant fuzzy sets”. In: *Information Sciences* 181.11 (2011), pp. 2128–2138 (cit. on pp. 38, 52).

[57] *mlr: Machine Learning in R*. https://rdrr.io/cran/mlr/ (cit. on pp. 44, 59).

[58] Wu, Hongchen and Knijnenburg, Bart P and Kobsa, Alfred. “Improving the prediction of users’ disclosure behavior by making them disclose more predictably?” In: *Symposium on Usable Privacy and Security (SOUPS)*. 2014 (cit. on p. 49).

[59] Nissenbaum, Helen. “A contextual approach to privacy online”. In: *Daedalus* 140.4 (2011), pp. 32–48 (cit. on p. 49).

[60] De Montjoye, Yves-Alexandre and Shmueli, Erez and Wang, Samuel S and Pentland, Alex Sandy. “openpds: Protecting the privacy of metadata through safeanswers”. In: *PloS one* 9.7 (2014), e98790 (cit. on p. 49).

[61] Singh, Bikash Chandra and Carminati, Barbara and Ferrari, Elena. “Privacy-aware Personal Data Storage (P-PDS): Learning how to Protect User Privacy from External Applications”. In: *IEEE Transactions on Dependable and Secure Computing* (2019) (cit. on pp. 49, 50, 58).

[62] Jiang, Changhao and Steenkiste, Peter. “A hybrid location model with a computable location identifier for ubiquitous computing”. In: *International Conference on Ubiquitous Computing*. Springer. 2002, pp. 246–263 (cit. on p. 53).

[63] Tonkin, Emma L and Woznowski, Przemyslaw R. “Activities of Daily Living Ontology for Ubiquitous Systems”. In: *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE. 2018, pp. 573–578 (cit. on p. 53).

[64] Li, Tao and Yang, He and He, Jun and Ai, Yong. “A Social Network Analysis methods based on ontology”. In: *2010 Third International Symposium on Knowledge Acquisition and Modeling*. IEEE. 2010, pp. 258–261 (cit. on p. 54).

[65] Seber, George AF and Lee, Alan J. *Linear regression analysis*. Vol. 329. John Wiley & Sons, 2012 (cit. on p. 57).

[66] *Twitter Usage Statistics - Internet Live Stats (2019)*. http://www.internetlivestats.com/twitter-statistics/ (cit. on p. 65).

[67] Herzallah, Wafa and Faris, Hossam and Adwan, Omar. “Feature engineering for detecting spammers on Twitter: Modelling and analysis”. In: *Journal of Information Science* (2017), p. 0165551516684296 (cit. on pp. 66, 72).

[68] Yang, Chao and Harkreader, Robert Chandler and Gu, Guofei. “Die free or live hard? empirical evaluation and new design for fighting evolving twitter spammers”. In: *International Workshop on Recent Advances in Intrusion Detection*. Springer. 2011, pp. 318–337 (cit. on p. 66).

[69] *Twitter Developers. Documentation,* https://developer.twitter.com/en/docs (cit. on p. 69).

[70] Witten, Ian H and Frank, Eibe and Hall, Mark A and Pal, Christopher J. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016 (cit. on p. 73).

[71] Gan, Mingxin and Dou, Xue and Jiang, Rui. “From ontology to semantic similarity: calculation of ontology-based semantic similarity”. In: *The Scientific World Journal* 2013 (2013) (cit. on p. 78).

Declaration

I, **Md. Zulfikar Alom**, do hereby declare that this doctoral dissertation entitled **Enhancing Data Privacy and Security Related Process Through Machine Learning** was carried out by me for the degree of *Doctor of Philosophy in Computer Science* under the guidance and supervision of the **Prof. Elena Ferrari** and **Prof. Barbara Carminati**, Department of Theoretical and Applied Science, University of Insubria, Varese, Italy.

I declare that all the material presented for examination is my own work and has not been written for me, in whole or in part, by any other person.

I also declare that any quotation or paraphrase from the published or unpublished work of another person has been duly acknowledged in the work which I present for examination.

This dissertation contains no material that has been submitted previously, in whole or in part, for the award of any other academic degree or diploma.

Varese, Italy, October 09, 2019

Md. Zulfikar Alom