Adapting Users' Privacy Preferences in Smart Environments

immediate

***Abstract*—A smart environment is a physical space where devices are connected to provide continuous support to individuals and make their life more comfortable. For this purpose, a smart environment collects, stores, and processes a massive amount of personal data. In general, service providers collect these data according to their privacy policies. To enhance the privacy control, individuals can explicitly express their privacy preferences, stating conditions on how their data have to be used and managed. Typically, privacy checking is handled through the hard matching of users' privacy preferences against service providers' privacy policies, by denying all service requests whose privacy policies do not fully match with individual's privacy preferences. However, this hard matching might be too restrictive in a smart environment because it denies the services that partially satisfy the individual's privacy preferences. To cope with this challenge, in this paper, we propose a *soft privacy matching mechanism*, able to relax, in a controlled way, some conditions of users' privacy preferences such to match with service providers' privacy policies. At this aim, we exploit machine learning algorithms to build a classifier, which is able to make decisions on future service requests, by learning which privacy preference components a user is prone to relax, as well as the relaxation tolerance. We test our approach on two realistic datasets, obtaining promising results.**

***Index Terms*—Internet of Things; Privacy Preferences; Privacy Policies.**

I. Introduction

Internet of Things (IoT) based smart environments are physical spaces, enriched with connected sensors and smart devices that offer services of different nature (e.g., home automation, entertainment, health monitoring, and so on) to being of support to individuals in their daily activities. In order to offer personalised services based on individual habits, smart environments usually collect, store, and process a massive amount of personal information about individuals. For instance, IoT-based home automation systems monitor users' behavior by using motion sensors, Wi-Fi signals, or facial recognition technology, to identify their presence in rooms and automatically adjust their temperature or lighting.

Personal information collected by service providers is then managed according to the providers' privacy practices. Every service provider has its own privacy policies, which express how it collects and manages customers' personal information and for how long (e.g., data usage purpose, retention time, etc). Before joining a service, a user has to read, understand and manually accept provider's policies. This is a very complex and time-consuming task for users, even due to their lack of decision-making abilities in privacy management [1], [2]. As an alternative solution to manual policy checking, users can set up their *privacy preferences*, that is, a set of well-defined conditions according to which their data can be collected and used. If properly encoded, privacy preferences and privacy policies can be automatically matched to verify if provider policies satisfy user preferences. To overcome the difficulties an average user might have in properly defining his/her privacy preferences, many researchers have investigated various mechanisms for predicting/suggesting users' privacy preferences by using a variety of learning strategies (e.g., [3], [4], [5]). Automatic matching of privacy policies and privacy preferences is done by a user agent following a *"take it or leave it"* approach [6], [7]. This implies that a user can access the service only if provider's policies fully match all conditions posed by his/her privacy preferences.

However, this checking might be too restrictive resulting in the denying of many service requests. This is particular crucial in IoT environments, where due to the many involved smart devices it could be very difficult to properly encode user privacy requirements and this increases the possibility of a policy mismatch, with the result of the loss of a lot of possible services. For instance, let us consider a privacy preference stating that a user will share his/her health-related data (e.g., blood-pressure, heart-beat, etc.) acquired through a smart devices with the health care service provider if this stores the data only for 100 days. Let us also assume a health care service provider, whose privacy policy states that it will store data for 110 days. Clearly, this privacy policy does not satisfy the user's privacy preference, even if it is close to be. In real life, if the user has some health conditions (s)he would most likely do not care about 10 days more of retention, being prone to make an exception to her/his privacy preference in order to get the service. However, an hard privacy matching mechanism would deny the service, without considering the possible benefits user might lose.

To overcome this limitation, we aim to extend automatic privacy matching, making it able to make *controlled* exceptions on defined privacy preferences. More precisely, we try to learn how to relax an existing privacy preference to cope up with dynamic and heterogeneous needs of IoT environments. The key idea is to adopt machine learning to infer, for each user, which component of a privacy preference (e.g., purpose, data, retention, and recipient) (s)he is willing to relax and how much. At this purpose, we exploit supervised machine learning algorithms [8] 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 (e.g., in the previous example, 10 days for the retention component), and learn a classifier able to state when privacy preferences can be relaxed. To the best of our knowledge, we are the first showing how machine learning algorithms can be used to learn users tolerance to privacy preferences relaxation.

To show the effectiveness of the proposed approach, we have carried out several experiments. In particular, we have experimentally tested different supervised machine learning algorithms [8], namely, Support Vector Machine, Naive Bayesian, and Random Forest to see which one gives better performance in the considered scenario. 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 remainder of the paper is organized as follows. Section II provides the problem statement, whereas Section III explains the building blocks of the proposed approach. Section IV illustrates experimental results, whereas Section V presents related work. Finally, Section VI concludes the paper.

II. Problem Statement

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 [6], [9]. 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.

[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.

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 III-A). Then, we exploit machine learning 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.

III. 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.

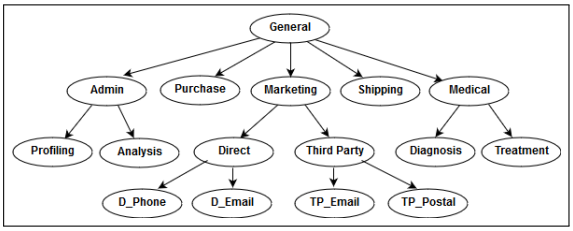

Figure 1: An example of purpose tree

A. 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 [10]. 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* [10], called T_p (see an Example in Figure 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 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* [11] 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 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.), Health-care data (e.g., Blood Pressure, Heart Beat, Weight, and so on.). 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* [12].

*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:*

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

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

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

provider's requested data are: *blood_pressure, weight, height, disease_name*. Hence, the distance value of can be calculated as follows:

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

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* [13] 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:

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

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* [13].

*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.

B. 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

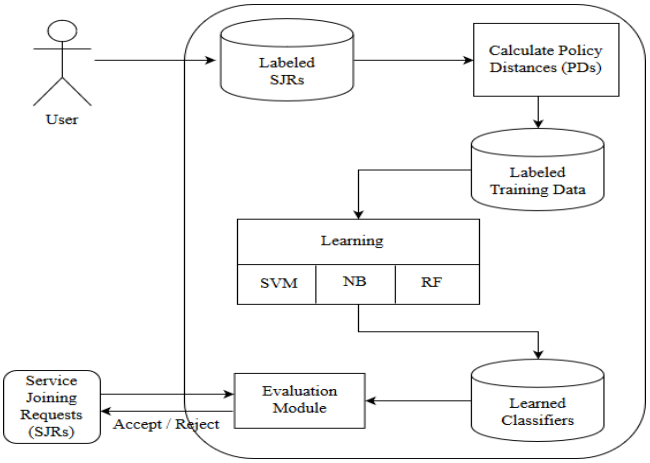

Figure 2: Learning architecture

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 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 III-A.

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 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 IV), 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) [8].

IV. 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 (i.e., learning phase), and then give their feedback on labels associated by the system to new service joining requests (i.e., evaluation 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 machine learning algorithms (i.e., SVM, NB, and RF) to learn his/her privacy behaviors (see Figure 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.

A. 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.

		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 I: 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 II: Metrics

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 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 [14]. 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 3X3 confusion matrix in Table I. 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 II.

[3]To adhere to the blind review process, we do not report here details on the involved universities

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

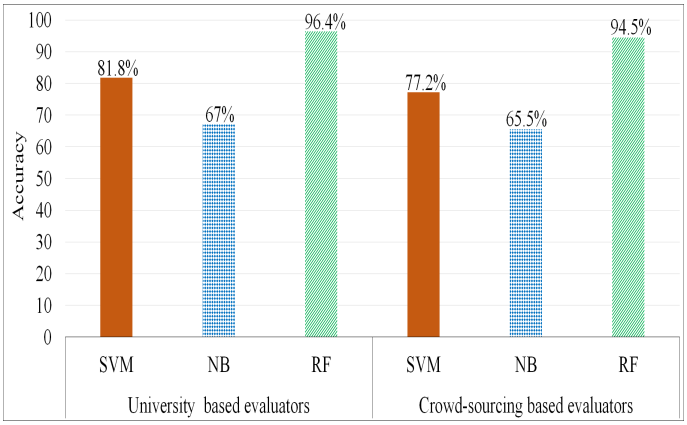

Figure 3: Accuracy of different classifiers

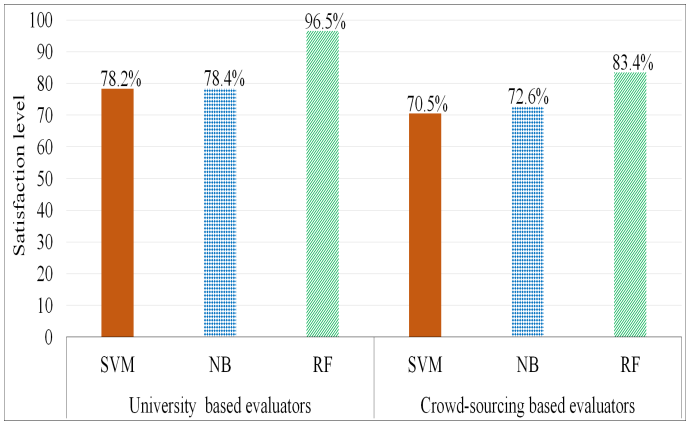

Figure 4: Evaluators satisfaction level

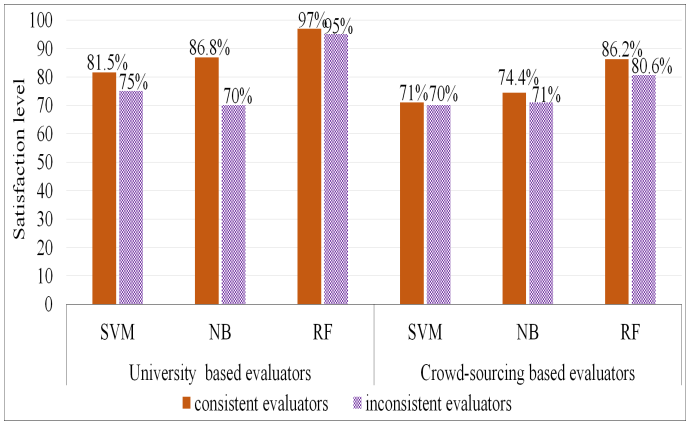

Figure 5: Satisfaction level of consistent and inconsistent evaluators

B. 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, 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.

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 4, 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.

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 5 shows the satisfaction level of consistent and inconsistent evaluators for

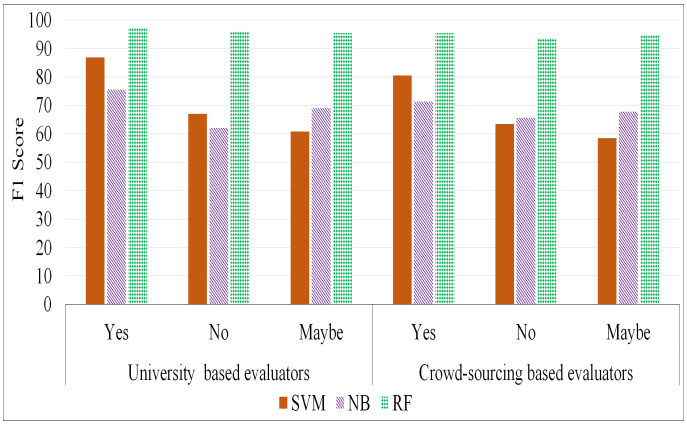

Figure 6: Comparison of F1 score of different classifiers for training datasets

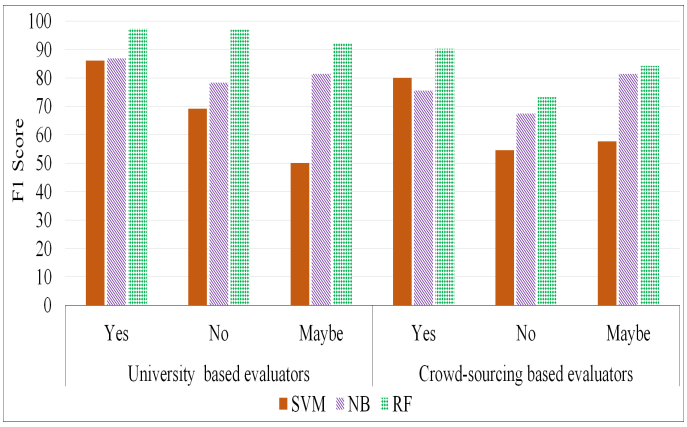

Figure 7: Comparison of F1 score of different classifiers for testing datasets

the two groups. As expected, the the satisfaction level of consistent evaluators is greater than the satisfaction level of inconsistent evaluators.

F1-score. We measured the F1-score for each class (i.e., Yes, No, and Maybe) in the training dataset (see Table III) and in the testing dataset (see Table IV). Figures 6 and 7 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).

V. Related work

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. [5] proposed the concept of intelligent software that helps users to make better privacy decisions in the IoT environments. To this end, the authors adopted machine learning tools and performed clustering analysis on the collected preferences in order to understand users privacy concerns towards IoT applications and services. Recently, Nakamura et al. [4] proposed a machine-learning approach in order 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 user at the time of registration. Similarly, Singh et al. [3] proposed a suite of semi-supervised approaches in order to learn the privacy aptitudes of Personal Data Storage owners. These learned models are then used to answer third party access requests. The authors showed that their approaches provide a better accuracy than [4] 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 networks 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 [15]. Thus, researchers have investigated solutions that automatically configure user privacy settings with minimal effort. For instance, Sadeh et al. [16] 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 machine learning based ones. The authors claimed that the machine-generated policies have better accuracy than the user-defined ones. Likewise, Fang et al. [17] proposed a model that infers access control policies for personal information sharing in online social networking services. They also used a supervised machine learning 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., gender, age, and so on) with feedback of the users as a training dataset and they trained the personalized machine learning models.

Bilogrevic et al. [18] proposed a machine learning 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 machine learning based logistic classifier to predict users privacy decisions. They used users' contextual information (e.g., location, time, etc.) and past behavior as features set for a training dataset. Similarly, Liu et al. [19] proposed a personalized privacy assistant (PPA) that pro-actively produces permission settings for Android applications on behalf of the users. The authors developed SVM classifiers to predict users decisions for each permission request by using their privacy profiles.

However, our approach differs from all the above mentioned proposals in that none of the above work addressed the issue of learning how much a privacy preference can be relaxed in order to access a service. To the best of our

		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 III: 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 IV: Performance comparison of different learning algorithms for the testing datasets

knowledge, we are the first showing that machine learning tools can be effectively used to learn users privacy relaxing aptitudes toward satisfying providers' policies.

VI. Conclusion

In this paper, we have proposed a framework that is able to relax, in a controlled way, individuals' privacy preferences in order to join more services. We have considered three different learning algorithms (i.e., SVM, NB, and RF) to test which one performs better in the considered scenario. 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.

In the future, we plan to conduct extensive user studies as well as to deploy our solution into real IoT ecosystems.

References

[1] A. Acquisti, L. Brandimarte, and G. Loewenstein, "Privacy and human behavior in the age of information," *Science*, vol. 347, no. 6221, pp. 509–514, 2015.

[2] D. J. Solove, "Introduction: Privacy self-management and the consent dilemma(2013)," *Harvard Law Review*, vol. 126, p. 1880.

[3] B. C. Singh, B. Carminati, and E. Ferrari, "Learning privacy habits of pds owners," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*. IEEE, 2017, pp. 151–161.

[4] T. Nakamura, S. Kiyomoto, W. B. Tesfay, and J. Serna, "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.

[5] H. Lee and A. Kobsa, "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.

[6] S. Pearson, "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.

[7] Å. A. Nyre, K. Bernsmed, S. Bo, and S. Pedersen, "A server-side approach to privacy policy matching," in *Availability, Reliability and Security (ARES), 2011 Sixth International Conference on*. IEEE, 2011, pp. 609–614.

[8] S. B. Kotsiantis, "Supervised machine learning: A review of classification techniques," 2007.

[9] B. C. Singh, B. Carminati, and E. Ferrari, "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.

[10] J.-W. Byun and N. Li, "Purpose based access control for privacy protection in relational database systems," *The VLDB JournalThe International Journal on Very Large Data Bases*, vol. 17, no. 4, pp. 603–619, 2008.

[11] Z. Wu and M. Palmer, "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.

[12] S. Niwattanakul, J. Singthongchai, E. Naenudorn, and S. Wanapu, "Using of jaccard coefficient for keywords similarity," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. 1, no. 6, 2013.

[13] Z. Xu and M. Xia, "Distance and similarity measures for hesitant fuzzy sets," *Information Sciences*, vol. 181, no. 11, pp. 2128–2138, 2011.

[14] "mlr: Machine Learning in R ," accessed on November 2018. [Online]. Available: https://rdrr.io/cran/mlr/

[15] H. R. Lipford, A. Besmer, and J. Watson, "Understanding privacy settings in facebook with an audience view." *UPSEC*, vol. 8, pp. 1–8, 2008.

[16] N. Sadeh, J. Hong, L. Cranor, I. Fette, P. Kelley, M. Prabaker, and J. Rao, "Understanding and capturing people's privacy policies in a mobile social networking application," *Personal and Ubiquitous Computing*, vol. 13, no. 6, pp. 401–412, 2009.

[17] L. Fang and K. LeFevre, "Privacy wizards for social networking sites," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 351–360.

[18] I. Bilogrevic, K. Huguenin, B. Agir, M. Jadliwala, and J.-P. Hubaux, "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.

[19] B. Liu, M. S. Andersen, F. Schaub, H. Almuhimedi, S. A. Zhang, N. Sadeh, Y. Agarwal, and A. Acquisti, "Follow my recommendations: A personalized privacy assistant for mobile app permissions," in *Symposium on Usable Privacy and Security*, 2016.