Malicious Account Identification in Social Network Platforms

LOREDANA CARUCCIO, Department of Computer Science, University of Salerno, Italy
GAETANO CIMINO, Department of Computer Science, University of Salerno, Italy
STEFANO CIRILLO, Department of Computer Science, University of Salerno, Italy
DOMENICO DESIATO*, Department of Computer Science, University of Bari Aldo Moro, Italy
GIUSEPPE POLESE, Department of Computer Science, University of Salerno, Italy
GENOVEFFA TORTORA, Department of Computer Science, University of Salerno, Italy

Nowadays, people of all ages are increasingly using Web platforms for social interaction. Consequently, many tasks are being transferred over social networks, like advertisements, political communications, and so on, yielding vast volumes of data disseminated over the network. However, this raises several concerns regarding the truthfulness of such data and the accounts generating them. Malicious users often manipulate data in order to gain profit. For example, malicious users often create fake accounts and fake followers to increase their popularity and attract more sponsors, followers, and so on, potentially producing several negative implications that impact the whole society. To deal with these issues it is necessary to increase the capability to properly identify fake accounts and followers. By exploiting automatically extracted data correlations characterizing meaningful patterns of malicious accounts, in this paper, we propose a new feature engineering strategy to augment the social network account dataset with additional features, aiming to enhance the capability of existing machine learning strategies to discriminate fake accounts. Experimental results produced through several machine learning models on account datasets of both the Twitter and the Instagram platforms highlight the effectiveness of the proposed approach towards the automatic discrimination of fake accounts. The choice of Twitter is mainly due to its strict privacy laws, and because its the only social network platform making data of their accounts publicly available.

CCS Concepts: • **Security and privacy** → **Human and societal aspects of security and privacy**; • **Theory of computation** → **Database constraints theory**; **Machine learning theory**.

Additional Key Words and Phrases: Data management, Fake accounts, Data Profiling, Social networks

ACM Reference Format:
Loredana Caruccio, Gaetano Cimino, Stefano Cirillo, Domenico Desiato, Giuseppe Polese, and Genoveffa Tortora. 2023. Malicious Account Identification in Social Network Platforms. 1, 1 (May 2023), 25 pages. https://doi.org/XXXXXXX.XXXXXXX

*Corresponding author.

Authors' addresses: Loredana Caruccio, lcaruccio@unisa.it, Department of Computer Science, University of Salerno, via Giovanni Paolo II n.132, Fisciano (SA), Italy, 84084; Gaetano Cimino, gcimino@unisa.it, Department of Computer Science, University of Salerno, via Giovanni Paolo II n.132, Fisciano (SA), Italy, 84084; Stefano Cirillo, scirillo@unisa.it, Department of Computer Science, University of Salerno, via Giovanni Paolo II n.132, Fisciano (SA), Italy, 84084; Domenico Desiato, domenico.desiato@uniba.it, Department of Computer Science, University of Bari Aldo Moro, via Edoardo Orabona n.4, Bari (BA), Italy, 70125; Giuseppe Polese, gpolese@unisa.it, Department of Computer Science, University of Salerno, via Giovanni Paolo II n.132, Fisciano (SA), Italy, 84084; Genoveffa Tortora, tortora@unisa.it, Department of Computer Science, University of Salerno, via Giovanni Paolo II n.132, Fisciano (SA), Italy, 84084.

1 INTRODUCTION

Social networks enable sharing information among users of all ages, at every moment, and in every part of the world. Social interaction platforms like Instagram, Twitter, Tumblr, etc., have a significant impact on the daily life of their users and the entire society.

The typical user profile over a social network might contain several clues concerning emotions, passions, interests, and characteristics of the profile's owner. For these reasons, many people tend to have a virtual life over social network platforms, which is not necessarily coupled with their real one. However, in a society where friendships are often strengthened through Facebook, emotions spread through Instagram, jobs found through LinkedIn, and so on, it is necessary to define control mechanisms capable of preventing the misuse of such sharing platforms by automatically detecting possible malicious users, i.e., spammers, bots, fake profiles, and so on.

A fundamental aspect to be monitored over a social network is the popularity of a profile, witnessed by the number of its friends or followers. A Twitter or Instagram profile with many followers is considered influential, hence it provides a better reputation to the profile's owner and attract better-paid advertisements. Consequently, a common practice of several social network users is to buy fake followers to appear more influential, also because they can be bought at an extremely low price (a few dollars for hundreds of fake followers). If this practice was merely used to support individual vanity, it would be harmless, but if it aimed at making an account more reliable and influential, it might be dangerous. For instance, spammers could buy fake followers, aiming to increase their popularity or influence the promotion of products, trends, fashions, and so on [12].

In general, it is possible to find many types of anomalous accounts in social networks, such as Spammers, Bots, Cyborgs, and Trolls. Spammer accounts tend to recommend fake contents and/or dangerous links. Bot algorithms tend to manage accounts to simulate human behavior, trying to automatically perform typical human actions. With respect to them, Cyborgs are both managed by humans, hence they are not necessarily malicious. For instance, a politician might not handle his/her account personally and might rely on a stuff of people together with some Bots. Finally, Trolls are algorithms aiming to disrupt conversations and activities of others.

Often, fake accounts also have numerous followers, most of which are fake, and their identification is an extremely complex task. In the literature, we find several automatic techniques for identifying spams and bots [18, 35, 45]. Some of them focus on the characterization of human behaviors with the help of sociologists, whereas others exploit supervised machine learning techniques on datasets containing different types of accounts, manually classified by humans [5, 6, 15]. Additional work relies on the features of user profiles, and on those related to the behavior and timing of accounts, in order to identify spammers in microblogging, by employing multi-feature strategies [35, 45, 51]. In this context, blockchain has the potential to provide new opportunities for checking the genuineness of social network accounts [24].

In this paper, we focus our attention on the detection of fake accounts by trying to infer peculiar characteristics, in terms of correlations within the data of a social network profile dataset, aiming at enhancing the capabilities of machine learning methods to discriminate them. In particular, we propose a new feature engineering strategy exploiting profiling metadata, represented in terms of relaxed functional dependencies (RFDs), which can be automatically inferred from data [8, 9]. The strategy enables the construction of novel features that could be added to baseline datasets, in order to improve the performances of classification models used for discriminating fake accounts [37]. The proposed strategy extends our previous work described in [7], which exploited semantic characteristics of data to support an analyst in the detection of fake accounts. With respect to it, the strategy proposed in this paper provides several significant improvements towards a complete automation of the fake account discrimination process.

The usage of RFDs in discriminating fake from real accounts is motivated by the fact that automatic procedures used to create fake accounts often produce similarity patterns over data. In fact, fake account generators often introduce slight differences during the generation of account properties, such as screen name, user data, and account creation time-stamp. These differences can be detected through RFDs, providing a descriptive profile of fake accounts, as demonstrated in [7]. However, to exploit such an idea in practice, it is necessary to define a proper methodology to support and/or enhance the automatic evaluation of social network accounts. To this end, the strategy proposed in this paper deeply extends the idea provided in [7], since the concept of RFD is used to define a feature engineering strategy to be applied in a classification scenario. The general idea is to derive new features based on automatically extracted RFDs, aiming to improve the classification score of machine learning models discriminating fake accounts. The proposed feature engineering strategy exploits RFDs holding only on the profiling data of fake accounts, and thanks to a newly defined RFD validation function relying on the frequency of account tuples (FAV), it is able to characterize the involvement of training samples in the RFD validation process. Moreover, since the addition of new features could introduce noise and expose the machine learning model to problems, such as overfitting and underfitting [48], we also propose a novel FAV-based feature Evaluation Metric (FEM) for ranking the new features and select the most relevant ones.

To demonstrate the effectiveness of the proposed strategy and the associated metrics, we have selected different machine learning models with the aim of showing how our feature engineering strategy improves their classification performances without raising the above-mentioned problems.

In summary, the main contributions of our proposal are:

- a deep semantic investigation of RFD meaningfulness for the discrimination of fake accounts;
- a feature engineering strategy relying on automatically extracted RFDs to improve classification performances of machine learning models discriminating fake accounts;
- an approach for ranking and selecting relevant features based on a novel metric, named FAV-based feature Evaluation Metric (FEM).

The remainder of the paper is organized as follows. In Section 2 we describe relevant works concerning malicious account identification. In Section 3 we recall the definition of RFD together with their potential applications. Section 4 describes the semantic analysis we performed to characterize the usefulness of RFDs in discriminating fake accounts. Section 5 presents the new proposed feature engineering strategy and the associated FEM metric. Section 6 shows the experimental evaluation we performed to measure the effectiveness of the proposed strategy, and results obtained with four classification models. Finally, conclusions and future directions are provided in Section 7.

2 RELATED WORK

The detection of fake accounts and fake activities represents a complex problem for social network companies like Facebook, Instagram, Twitter, and so on. In fact, in the literature there are several works aiming at providing automatic support to properly perform discrimination tasks, even though they mainly focus on specific issues, such as fake account identification, activity classification, behavior characterization, and so on. Moreover, different solutions can also be distinguished by the kind of data they consider. For instance, the analysis and classification tasks can involve post/tweet texts, patterns in profiles' activities, and/or simple account information. Specificity in the approaches also depends on the analyzed social networks and the strategies/models they employ. In what follows, we detail the most representative solutions from the literature, by categorizing them on the issues they mainly focus on.

One of the main issues to consider for trustworthy social networks concerns the identification of malicious activities. Several detection techniques have been surveyed in [27] analyzing their pros and cons, including their applicability. An important proposal in this category has been described in [2]. It uses a Naive Bayes classifier for detecting wrong information, like for example, online rumors, and for distinguishing and predicting fake news. Furthermore, the authors show the application of their algorithm in different contexts, like for instance text classification, spam filtering, hybrid recommendation, and collaborative filtering. Instead, in [23], the detection of malicious activities using phone numbers to promote advertising campaigns over the Twitter platform has been studied. The authors collected tweets, user meta-data, and other Twitter data concerning $3,370$ campaigns, disseminated by $670,251$ accounts; then, they modelled a Twitter dataset by considering the interconnections between several users involved in the advertising campaigns. Thus, a feedback-based active learning strategy has been proposed, which also uses a novel metric, called Hierarchical Meta-Path Score (HMPS), for measuring the proximity of unknown users with respect to the ones involved in spamming activities. Profile activities, and in particular, patterns of the retweeting activity over the Twitter platform is the focus of Retweet-Buster (RTbust) [36], which exploits unsupervised feature extraction and clustering techniques to emphasize benign and malicious patterns of retweeting activities. It also implements an LSTM autoencoder that maps the retweet time series into feature vectors by clustering them with a hierarchical density-based algorithm. On the other hand, fake account discrimination represents another widely studied problem in the social network context. To this end, literature proposals [42] typically differ on the data they analyze, also yielding a categorization based on the structure of considered data.

In order to detect fake accounts, several algorithms and techniques mostly exploit the vast volumes of unstructured data generated from social networks [39]. Among these, in [32] content and metadata at the tweet level have been exploited for recognizing bots, by means of a deep neural network based on contextual long short-term memory (LSTM). In particular, this approach extrapolates contextual features from user metadata and uses the LSTM deep nets to process the tweet text, yielding a model capable of obtaining high classification accuracy also with few data. Instead, the statistical text analysis is exploited in a novel general framework to discover compromised accounts [44]. The framework relies on the consideration that an account's owner uses his/her profile in a way that is completely different from the same account when this is hacked, enabling a syntactic analyzer for identifying the features used by hackers (or spammers) when they compromise a genuine account. Thus, a language modeling algorithm is used for extrapolating the similarities between language models of genuine users and those of hackers/spammers, in order to characterize hackers' features and use them in supervised machine learning approaches. Finally, spectral patterns derived from textual contents of users have been analyzed in [5], aiming to find information automatically forwarded on social network platforms. This technique extracts a feature vector from the Discrete Wavelet Transform and a weighting schema, named Lexicon based Coefficient Attenuation, and it employs a classification model based on the Random Forest to properly identify *legitimate robots*, *malicious robots*, and *humans*.

Other works mainly organize account and behavior information in structured datasets in order to use well-known and/or novel models for classifying social network accounts. For instance, in [19] the authors proposed a methodology to evaluate bot classifiers by varying sizes and characteristics, and by testing them on unseen bot classes. The data they used to train and test classifiers include some of the largest and most varied collections of bots used in the literature, reaching a training set containing over $200,000$ data points. Instead, in [40] an SVM classifier based on posts and statuses, involving Twitter accounts, has been used to better discriminate fake accounts. The automatic collection of social network accounts has been addressed in [4]. In particular, the authors developed an ad-hoc Web crawler to automatically collect and filter public Twitter accounts and organize the data in testing and training datasets. Moreover, a Multi-layer Perceptron Neural Network has been modeled and trained over nine features characterizing a fake account. Another deep learning-based approach is provided in [49]. In particular, the authors propose DeepProfile that performs account classification through a dynamic CNN to train a learning model, which exploits a novel pooling layer to optimize the neural network performance in the training process.

Further approaches devoted to the fake account discrimination, also considered feature engineering and/or selection issues [3, 30, 33]. Nevertheless, most of the proposals including a feature engineering process, rely on domain experts or include manual work for characterizing meaningful features that permit a classifier to work with high accuracy. For instance, in [43], the authors have enumerated the main characteristics to discriminate a fake account from a genuine one. In particular, by manually examining different types of accounts, they have extracted a set of features in order to highlight the characteristics of malicious accounts. Moreover, they have analyzed the liking behavior of each account to build an automated mechanism to detect fake likes on Instagram.

Compared to the fake account discrimination approaches described above, in this paper we argue the importance of considering metadata (e.g., RFDs) in order to support and enhance feature engineering processes for complex scenarios, such as the discrimination of fake accounts. To this end, we propose a new feature engineering strategy that exploits such a kind of metadata, and directly employs it in the classification of fake accounts.

3 BACKGROUND

In what follows, we first recall some notations and concepts concerning relational databases, and then introduce Relaxed Functional Dependencies (RFDs), since they represent the basic concept underlying our methodology to characterize fake accounts.

A relational database schema $\mathcal{R}$ is a collection of relation schemas $(R_1,\ldots, R_n)$, where each R_i is defined over a fixed set of attributes $attr(R_i)$. Each attribute A_k has associated a finite or infinite domain $dom(A_k)$. A relation instance r_i of R_i is a set of tuples such that for each attribute $A_k \in attr(R_i)$, $t[A_k] \in dom(A_k)$, $\forall t \in r_i$, where $t[A_k]$ denotes the projection of t onto A_k. The collection of relations $(r_1,\ldots,r_n)$ of $(R_1,\ldots,R_n)$, resp., represents a database instance r of $\mathcal{R}$.

In the context of relational databases, functional dependencies (FDs) have been mainly used to define data integrity constraints, aiming to improve the quality of database schemas and to reduce manipulation anomalies. RFDs extend the following definition of FD.

DEFINITION 1. **(Functional dependency)**. *Let r of $\mathcal{R}$ be a database instance, and $X, Y \subseteq attr(\mathcal{R})$ be two sets of attributes, a functional dependency (FD) φ, denoted by $X \longrightarrow Y$, specifies a constraint on the possible tuples that can form the instance r: $X \longrightarrow Y$ iff for every pair of tuples (t_1, t_2) in r, if $t_1[X] = t_2[X]$, then $t_1[Y] = t_2[Y]$. The two sets X and Y are also called Left-Hand-Side (LHS) and Right-Hand-Side (RHS), resp., of φ.*

RFDs relax the definition of FD on two main aspects, that is, by admitting i) the possibility of using approximate operators to compare projections of tuples on subsets of attributes, and ii) the possibility that the dependency might also hold only on a subset of the entire database. Approximate operators rely on the concept of *similarity constraint* [16].

DEFINITION 2. **(Similarity constraint)**. *Given an attribute A defined on a domain $\mathbb{D}$, and a threshold α, let $\phi[A] : \mathbb{D} \times \mathbb{D} \longrightarrow \mathbb{R}$ be a function evaluating the similarity between value pairs of A; a similarity constraint on A is satisfied iff a value pair of A can be considered similar according to α with respect to the value computed through $\phi[A]$.*

As an example, we can define ϕ based on a similarity operator $\approx$, corresponding to the edit, the Levenstain, or the Jaro distance [20], such that, given two values $a_1, a_2 \in A$, $a_1 \approx a_2$ is true if a_1 and a_2 are "close" enough with respect to the predefined threshold α.

In order to indicate that a pair of values (a_1, a_2) are similar on attribute A, we will use the notation $A_{\leq\alpha}$.

Definition 3. **(Set of similarity constraints).** *Given a set of attributes $X \subseteq attr(\mathcal{R})$ with $X = \{A_1, \ldots, A_k\}$, a set of similarity constraints, denoted as X_Φ, represents the collection of similarity constraints $X_\Phi = \{A_{1\leq\alpha_1}, \ldots, A_{k\leq\alpha_k}\}$ associated to attributes of X.*

A dependency holding on "almost all" the tuples or on a "subset" of them is said to relax on the extent, whereas one that uses an approximate method to compare projections of tuples is said to relax on the attribute comparison method [8]. In case of "almost all" the tuples, a *coverage measure* should be specified to quantify the cases in which the RFD does not hold, whereas in case of "subset" (*constrained domains* in [11]), conditions are used to specify the subset of tuples on which the RFDs hold.

Definition 4. **(Coverage measure).** *Given a database instance r of $\mathcal{R}$, and an FD $\varphi : X \rightarrow Y$, a coverage measure Ψ on φ, $\Psi : dom(X) \times dom(Y) \rightarrow \mathbb{R}^+$, measures the amount of tuple pairs in r satisfying (or violating) φ.*

There are several coverage measures available. They usually return a normalized value in the range $[0, 1]$.

As an example, the *confidence measure* introduced in [25] computes the maximum number of tuples $r_1 \subseteq r$ for which φ holds in r_1 normalized on the total number of tuples in r. For traditional FDs this coverage measure returns the value 1. Another example of coverage measure is the $g3$-error [29] that defines the minimum number of tuples $r_1 \subseteq r$ to be removed from r in order to make the RFD valid on the remaining ones,normalized on the total number of tuples in r. For traditional FDs this coverage measure returns the value 0.

Definition 5. **(Constrained domain).** *Given a relation schema R with attributes $\{A_1, \ldots, A_k\}$ defined on a domain $\mathbb{D} = \mathbb{D}_1 \times \mathbb{D}_2 \times \cdots \times \mathbb{D}_k = dom(R)$, let c_i, $i = 1, \ldots, k$, be a condition on $\mathbb{D}_i$, the constrained domain $\mathbb{D}_c$ is defined as*

$$\mathbb{D}_c = \left\{t \in dom(R) | \bigwedge_{i=1}^{k} c_i(t[A_i])\right\}.$$

Constrained domains restrict the validity of RFDs through a concept of "subsets" of tuples.

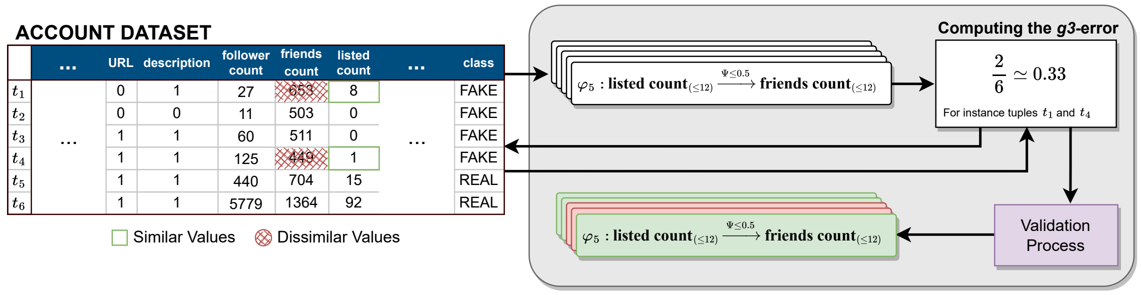
As a consequence, it is possible to have RFDs relaxing on the attribute comparison only, RFDs relaxing on the extent only, and RFDs relaxing on both. The latter are named *hybrid* RFDs.

Definition 6. **(Relaxed functional dependency).** *Given a relation schema R, an RFD φ on R is denoted as*

$$\left[X_{\Phi_1} \xrightarrow{\Psi \leq \varepsilon} Y_{\Phi_2}\right]_{\mathbb{D}_c} \quad (1)$$

where

- $\mathbb{D}_c$ is the constrained domain filtering the tuples on which φ applies;
- $X, Y \subseteq attr(R)$, with $X \cap Y = \varnothing$;
- Φ_1 and Φ_2 are sets of similarity constraints on attributes X and Y, respectively;
- Ψ is a coverage measure defined on $\mathbb{D}_c$;
- ε is a threshold.


ACCOUNT DATASET
Similar Values
Dissimilar Values
φ_5 : listed count$_{(\le 12)} \xrightarrow{\Psi \le 0.5}$ friends count$_{(\le 12)}$
Computing the *g3*-error
$\frac{2}{6} \simeq 0.33$
For instance tuples t_1 and t_4
Validation Process
φ_5 : listed count$_{(\le 12)} \xrightarrow{\Psi \le 0.5}$ friends count$_{(\le 12)}$

	...	URL	description	follower count	friends count	listed count	...	class
t_1	...	0	1	27	653 ▨	8 🟩	...	FAKE
t_2		0	0	11	503	0		FAKE
t_3		1	1	60	511	0		FAKE
t_4		1	1	125	449 ▨	1 🟩		FAKE
t_5		1	1	440	704	15		REAL
t_6		1	1	5779	1364	92		REAL

🟩 Similar Values ▨ Dissimilar Values

Fig. 1. An example of the RFD validation process.

Given a relation instance r of R, with $r \subseteq \mathbb{D}_c$, the RFD φ holds on r, denoted by $r \models \varphi$, if and only if: $\forall\ (t_1, t_2) \in r$, if Φ_1 is true for each constraint $A_{\le \alpha} \in \Phi_1$, then *almost always* Φ_2 is true for each constraint $B_{\le \beta} \in \Phi_2$. Here, *almost always* means that $\Psi(\pi_X(r), \pi_Y(r)) \le \varepsilon$, where (resp. $\pi_X(r)$) is the projection of r on the attributes in X (resp. in Y).

In other words, if $t_1[X]$ and $t_2[X]$ abide by similarity constraints specified by Φ_1, then $t_1[Y]$ and $t_2[Y]$ abide by similarity constraints specified by Φ_2 by tolerating an error Ψ lower than ε.

According to the definition (1), a traditional FD $X \longrightarrow Y$ can also be written as:

$$X_{\text{EQ}} \xrightarrow{\Psi_1} Y_{\text{EQ}} \tag{2}$$

where no constrained domain is represented, since it would be a sequence of tautologies, i.e., it is equal to $dom(R)$, EQ is the equality constraint, and Ψ_1 means $\psi(X, Y) = 0$, that is no error are admitted, then $t_1[Y]$ and $t_2[Y]$ agree with the equality constraint whenever $t_1[X]$ and $t_2[X]$ agree.

In what follows, w.l.o.g. we use RFDs with RHS cardinality of one, which is a condition that can be easily released by using well-known FD transformations. Moreover, for the sake of simplicity, in the following examples, we apply a more compact notation for the RFDs, showing only the operator and the numeric threshold associated with each attribute and without notations related to the constrained domain, since in our approach we always consider the $dom(R)$ as a domain.

Example 1. Let us consider the snippet of the *Account* dataset in Figure 1, then we can say that the RFD φ_5 : listed_count$_{(\le 12)} \xrightarrow{\Psi \le 0.5}$ friends_count$_{(\le 12)}$ holds on it. This means that if two accounts have a similar listed_count (i.e., the difference between the number of public lists in which each account owner appears is less than or equal to 12), then in the *majority of cases* they also have a similar friends_count (i.e., the difference between the number of users followed by each account owner profile is less than or equal to 12). To evaluate the majority of cases we admit an error in the validation of φ_5 less than or equal to 0.5 measured by a $g3$-error coverage measure. As described in the right part of Figure 1, during the validation process, the $g3$-error is measured on the considered dataset by computing the minimum fraction of tuples to be removed from the dataset in order to eliminate any possible violation for φ_5. For instance, a violation for φ_5 is formed by the pair (t_1, t_4) since they have a distance less than or equal to 12 on the attribute listed_count (i.e., the LHS of φ_5), but not on the attribute friends_count (i.e., the RHS of φ_5). More specifically, on the considered dataset there are several violating tuples, such as (t_1, t_2), (t_1, t_3), (t_1, t_4), (t_1, t_5), (t_2, t_4), (t_3, t_4), but if we remove just two tuples, i.e., t_1 and t_4, we are able to remove any violation for φ_5. Consequently, 2 tuples out of 6,

Datasets	# Columns	# Rows	Size [KB]
Verified accounts	15	3949	400
Real accounts	15	1757	168
Fake Accounts	15	3313	316

Table 1. Statistics of the datasets considered in the evaluation.

i.e., 0.33, represents the $g3$-value associated with the validation of φ_5 yielding to consider the dependency valid on the considered dataset, since the extent threshold for φ_5 is set to 0.5.

4 CHARACTERIZING FAKE ACCOUNTS THROUGH RFDS

Fake account identification is a hot topic, since the massive usage of social networks has contributed to quickly spread harmful information. However, the manual detection of such accounts requires a significant effort by humans to analyze vast volumes of accounts data. To this end, the technique proposed in this paper supports the automatic identification of fake accounts in social networks by exploiting the concept of RFD (see Section 3) to emphasize data correlations that are typical of fake accounts, also providing a sort of descriptive profile to recognize them. This section describes how data correlations expressed in terms of RFDs permit to perform a semantic analysis of fake accounts data.

More specifically, we focus on fake account generators, which usually introduce slight differences during the generation of account properties, such as screen names, user data, and account creation time-stamps, and we try to exploit RFDs for detecting them, since they capture similarities among data. With this in mind, in what follows we describe how RFDs can be used to discriminate fake accounts. In particular, we focus on the Twitter social network, since it is the only one permitting to retrieve data concerning accounts by means of public API[1].

The general idea underlying the semantic analysis is to isolate meaningful patterns, i.e., RFDs characterizing specific classes of accounts, for using them to discriminate fake accounts. To this end, in what follows we first introduce the dataset used for this investigation, and then discuss how we analyzed RFDs in order to perform a proper semantic analysis. Finally, we discuss examples of RFDs that can highlight such a semantical discrimination.

4.1 Account Datasets

The datasets that we used to perform the semantic analysis are shown in Table 1, together with some statistics about them. We considered 9019 accounts, each categorized in following classes: *Real*, *Verified*, and *Fake*. In particular, we started by considering the *Real* and the *Fake* accounts of the public datasets provided in [17]. Instead, concerning *Verified* accounts, we have considered a set of user-profiles validated by Twitter: Twitter profiles with “blue tick” on their Twitter page. Finally, we also included new *Fake* accounts in order to increase the total set of accounts. Details on the considered datasets are described in the following.

Verified account dataset. A Twitter account can be classified as *verified* when it has been certified as authentic by the social network platform, i.e., it is denoted with a blue badge appearing near the user name of public accounts. Since Twitter is the authority releasing such badges, it performs strict checks on any account that maliciously uses such badges in their own image, background image, or anywhere else, to simulate the verified account status. We collected a dataset containing 3949 verified accounts by using a Twitter APIs and a Twitter account that we specifically created to select accounts with the blue badge.

[1]www.developer.twitter.com/en/docs/twitter-api

#	Attribute	Description
1	name	Name chosen by the account owner. It consists of a maximum of 20 characters.
2	screen_name	Identifier associated with the account owner. It consists of a maximum of 15 characters.
3	location	Location defined by the account owner.
4	url	Boolean value representing the absence or not of the URL set by the account owner.
5	description	Boolean value representing the absence or not of the description set by the account owner.
6	followers_count	Number of current followers associated with the account owner.
7	friends_count	The number of users followed by the account owner profile.
8	listed_count	Number of public list in which the account owner appears.
9	created_at	Data and time representing the creation of the account on Twitter.
10	favourites_count	Number of Tweet generating by the account owner during his/her activities on Twitter.
11	geo_enabled	Boolean value representing the geotagging of the Tweets related to the account owner.
12	statuses_count	Number of tweets/retweets that the account owner made.
13	lang	Code associated with the language specified by the account owner.
14	default_profile	Boolean value representing changes related to the theme or background of the account owner.
15	default_profile_image	Boolean value revealing whether the default image of Twitter has been changed by the account owner.

Table 2. Attributes concerning Twitter user objects.

Real account dataset. A Twitter account can be classified as *real* when it is managed and used by individuals. Analyzing this kind of accounts is vital, since they can be exploited to prevent the possibility that accounts automatically managed by bots are defined as real. In particular, we considered a dataset containing 1757 real accounts starting from the ones provided by Crisci *et al.* [17], which joined two heterogeneous datasets: TheFakeProject and #elezioni2013. Moreover, we added additional accounts of real users by including colleagues and friends registered over the Twitter platform, which agreed to share their accounts for supporting our analysis.

Fake account dataset. A Twitter account can be classified as *fake* when it has automatically generated and it is used to perform malicious activities, such as spamming, targeted advertising, and so forth. We started by considering the fake accounts used in [17], which were bought by authors from three different online platforms: FastFollowerz[2], InterTwitter[3], and TwitterTechnology[4]. Moreover, we also bought additional fake accounts from the InterTwitter website and added them to the Fake account dataset. The final dataset contained 3313 fake accounts.

Notice that, the final composition of the datasets has been made after the application of data cleaning and preparation procedures used to label and organize them, in order to avoid the presence of missing values. These filtering activities turned out to be relevant for extracting meaningful RFDs to discriminate fake accounts. The final statistics concerning datasets composition of Verified, Real, and Fake accounts are reported in Table 1.

Additionally, although the Twitter APIs enable us to retrieve more than 15 attributes from Twitter accounts, we focused our analysis only on the 15 most relevant attributes of each considered dataset, since the remaining ones do not refer to user account profiles. The selected attributes are shown in Table 2.

4.2 Semantic analysis of RFD discovery results

By considering the datasets described above, and by exploiting a discovery algorithm capable of extracting hybrid RFDs [10], we collected many RFDs for each class of accounts. This allowed us to semantically analyze data correlations that can help in discriminating fake accounts.

[2]www.fastfollowerz.com
[3]www.intertwitter.com
[4]www.twittertechnology.com

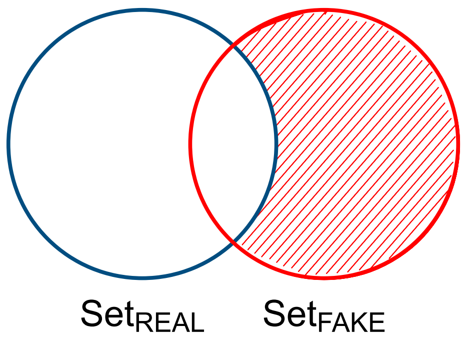


Fig. 2. Intersection of RFD sets holding on Real and Fake accounts, respectively.

Let us consider a relation Set_{REAL} containing real account data, and a relation Set_{FAKE} containing fake account data. Thanks to the availability of RFDs extraction algorithms [8], we can automatically extract RFDs holding on each dataset. Therefore, by analyzing differences between the two sets of RFDs, we can extract meaningful patterns concerning fake accounts. To this end, we aim to detect the RFDs highlighted in Figure 2, that is, those holding on fake accounts but not on real ones, as described by the following formula:

$$\Delta_{\text{Set}_{\text{FAKE}},\text{Set}_{\text{REAL}}} = \text{P}_{\text{Set}_{\text{FAKE}}} \setminus \text{P}_{\text{Set}_{\text{REAL}}}$$

where $\text{P}_{\text{Set}_{\text{FAKE}}}$ and $\text{P}_{\text{Set}_{\text{REAL}}}$ represent the RFDs holding on Fake and Real datasets, respectively.

RFDs that are common to the two sets can be ignored, since they do not permit to discriminate between the two types of accounts.

Analogously, let us consider the relation $\text{Set}_{\text{VERIFIED}}$, which contains data of accounts whose genuineness has been certified by Twitter. In order to characterize real accounts, we might want to analyze human behavior. This can be done by considering RFDs holding on real accounts but not on fake and verified ones. More specifically, we focus on a subset of RFDs identified in Set_{REAL} (the grey part of Figure 3), which corresponds to the set of RFDs defined by the following formula:

$$\Delta_{\text{Set}_{\text{REAL}},\text{Set}_{\text{FAKE}},\text{Set}_{\text{VERIFIED}}} = \text{P}_{\text{Set}_{\text{REAL}}} \setminus (\text{P}_{\text{Set}_{\text{Verified}}} \cup \text{P}_{\text{Set}_{\text{FAKE}}})$$

where $\text{P}_{\text{Set}_{\text{FAKE}}}$, $\text{P}_{\text{Set}_{\text{REAL}}}$, and $\text{P}_{\text{Set}_{\text{Verified}}}$ represent the RFDs holding on Fake, Real, and Verified accounts, respectively.

$\Delta_{\text{Set}_{\text{REAL}},\text{Set}_{\text{FAKE}},\text{Set}_{\text{VERIFIED}}}$ contains the RFDs that are only in Set_{REAL}, but not in Set_{FAKE} and $\text{Set}_{\text{VERIFIED}}$. In other words, our purpose is to find human behavioural patterns that are not replicable by algorithms or bots. Thus, we can ignore RFDs holding on either Set_{FAKE} or $\text{Set}_{\text{VERIFIED}}$. In particular, we overlook the RFDs holding on $\text{Set}_{\text{VERIFIED}}$, since this set includes Cyborgs, which mix human and bot behaviours, hence they are not useful to characterize typical human behaviour.

In what follows, we show an example of RFD characterizing the fake accounts dataset:

followers_count, statuses_count, default_profile ⟶ default_profile_image

The above RFD highlights the fact that one of the goals of bots is to spread malicious advertising. Consequently, they do not care about the Twitter profile; instead, they use it without a profile image, and without applying changes to the default profile. Moreover, they typically set the generated accounts with a similar number of followers, and also perform a similar number of activities (e.g., number of tweets/retweets).

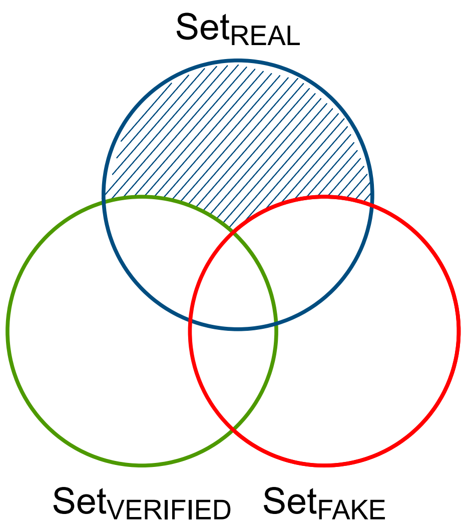


Fig. 3. Intersection of all RFD sets holding on Real, Verified, and Fake account, respectively.

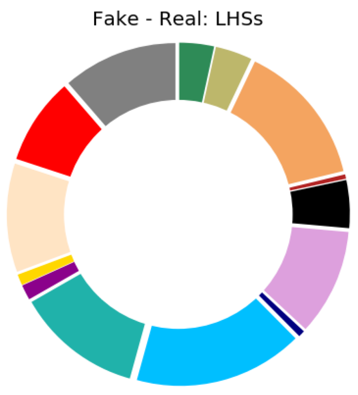
The RFDs resulted also useful for semantically characterizing real accounts. In particular, the RFDs holding on the Real dataset reveal that automatic procedures cannot emulate human behaviours. In what follows, we show an example of RFD characterizing real accounts:

$$\text{name, description, default\_profile} \longrightarrow \text{lang}$$

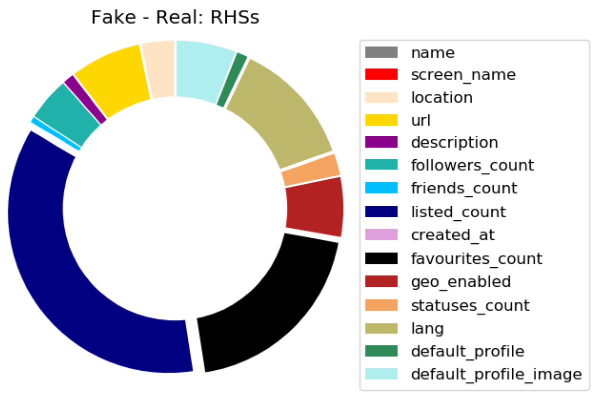
The RFD above is helpful to discriminate humans, since the language is a typical characteristic of a person, implying his/her way of writing. This is particularly interesting since it holds only on the real accounts dataset. Consequently, we can presume that automatic procedures for creating fake accounts can randomly reuse similar information for names and descriptions towards accounts with different languages.

4.3 Attribute involvement on discovered RFDs

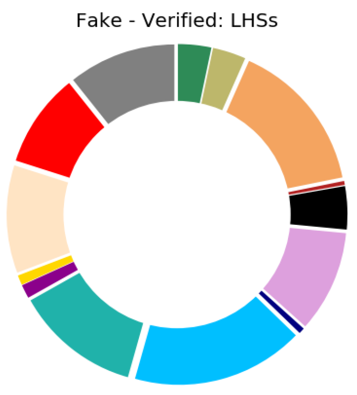
To further investigate the semantic usefulness of the RFDs we analyzed the impact of each attribute on the LHS (respectively RHS) by counting the number of times an attribute appears on the LHSs (respectively RHSs) of all discovered RFDs. Results in terms of percentages are shown in Figure 4. In particular, by considering RFDs of fake accounts but not of real ones, it is possible to notice that the attributes statuses_count and friends_count represent the ones that have the greatest impact when they appear on LHSs (Figure 4(a)). Instead, as shown in Figure 4(b), the most determined attribute (RHSs) is listed_count, even if also favourites_count appears many time as RHS. In general, by comparing Figures 4(a) and 4(b), we can notice that the impact on the LHSs is similar for several attributes; instead, a wider difference appears for attributes on the RHSs. Similar consideration can be provided referring to RFDs holding on fake accounts but not on verified ones, for both LHSs and RHSs, respectively (Figure 4(c) and Figure 4(d)). Finally, taking into consideration RFDs holding on real accounts but not on the union of fake and verified ones, it is possible to notice that the attributes followers_count and statuses_count present the greatest impact with respect to other attributes, in the case of LHSs (Figure 4(e)). Instead, as shown in Figure 4(f), the greatest impact for RHSs is obtained by the attribute



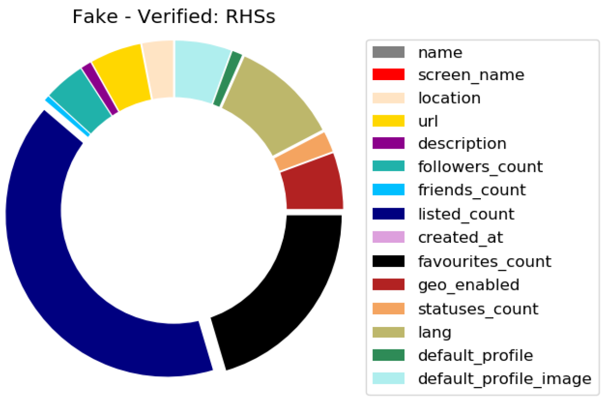
(a) Percentage of LHSs for holding RFDs on Fake vs Real accounts



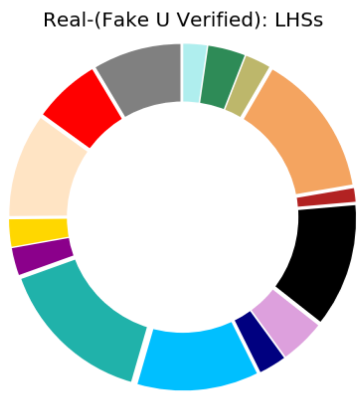
(b) Percentage of RHSs for holding RFDs on Fake vs Real accounts



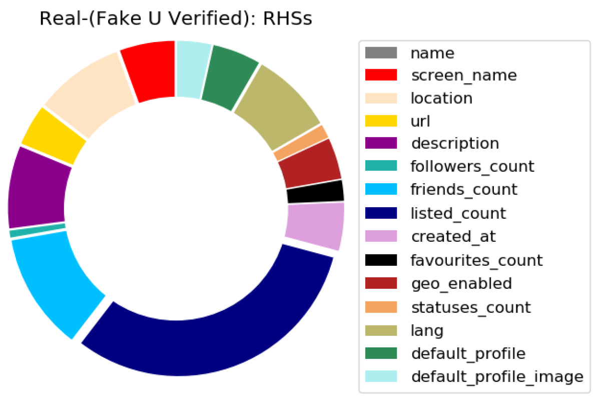
(c) Percentage of LHSs for holding RFDs on Fake vs Verified accounts



(d) Percentage of RHSs for holding RFDs on Fake vs Verified accounts



(e) Percentage of LHSs for holding RFDs on Real vs Fake and Verified accounts



(f) Percentage of RHSs for holding RFDs on Real vs Fake and Verified accounts

Fig. 4. Percentage of incidence of LHSs (left) and RHSs (right) for RFDs holding on the different datasets.

listed_count. In this case, also friends_count is determined many times, whereas all other attributes have little impact on percentage.

Summarizing, we can notice that, we cannot highlight particular evidence of the meaningfulness of attributes with respect to the number of times they are involved in the different sets of resulting RFDs. For instance, the different sets of resulting RFDs share the same attribute as the most present on the RHSs.

5 A FEATURE ENGINEERING STRATEGY FOR DISCRIMINATING FAKE ACCOUNTS

The whole semantic analysis allowed us to highlight the contribution of RFDs to discriminate fake accounts. However, semantic analysis can entail a huge quantity of RFDs to analyze significantly increasing the effort of an analyst in their evaluation. Thus, we need to devise more an effective strategy for exploiting RFDs to automatically discriminate fake accounts.

We have already seen that the frequency by which an attribute is involved in the LHSs (or RHSs) of RFDs does not provide useful information to discriminate fake accounts. Thus, we investigated other characteristics of RFDs capable of supporting the discrimination of fake accounts. To this end, we propose a new strategy for applying RFDs in a classification scenario, by demonstrating how they can be used to improve classification scores when a machine learning model is used. Thus, by following the results obtained in this semantic analysis, the general idea is to create new features capturing the whole semantics of discovered RFDs, which can be used as additional features of the training dataset to enhance the capability of classification algorithms to discriminate fake accounts.

In general, although the addition of new features can potentially increase the training time of classification algorithms, it could lead to the creation of more concise and accurate classifiers. Moreover, meaningful features could contribute to the understanding of the learned concept [31], but it should be avoided the introduction of noise and overfitting, due to the increase of data dimensionality.

Aiming to add new meaningful features based on RFDs, we defined a new function, named tuple Frequency Account in Validation (FAV), which permits to account for the number of times a tuple is involved in the validation of an RFD when it is coupled with other tuples. A more formal definition of the FAV function is provided below.

Definition 7 (Tuple Frequency Account in Validation (FAV)). *Given a relational database schema $\mathcal{R}$, defined on a set of attributes $attr(\mathcal{R}) = \{A_1, \dots A_m\}$, an instance r of $\mathcal{R}$ with n tuples, an RFD $\varphi : X_{\Phi_1} \xrightarrow{\Psi \leq \varepsilon} Y_{\Phi_2}$ holding on r, and a tuple t_i, the tuple frequency in validating φ can be defined as:*

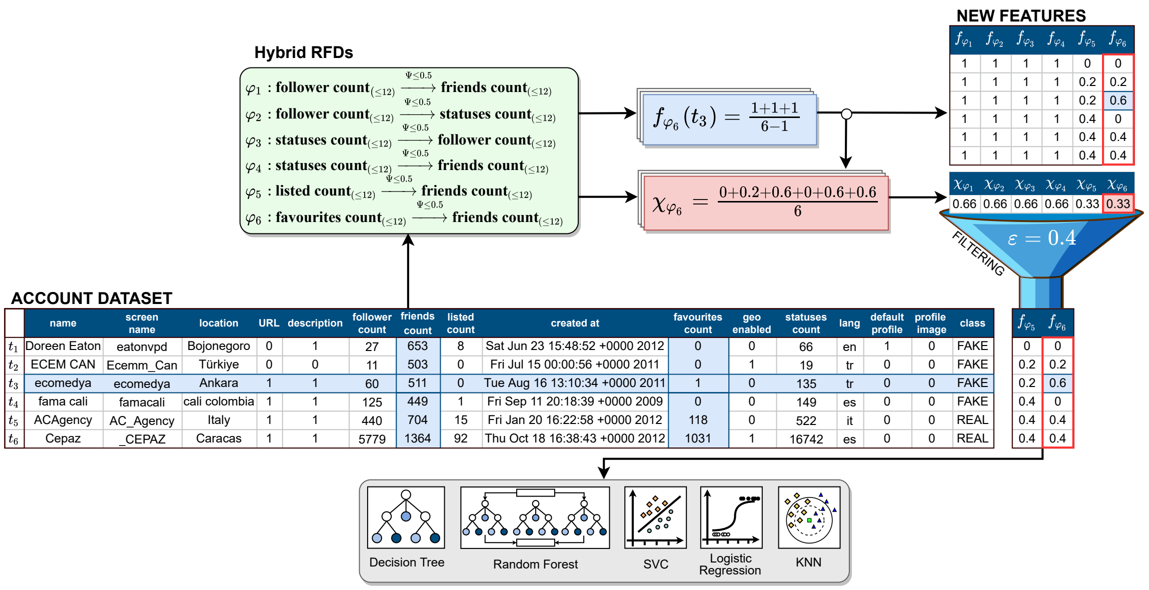
$$f_\varphi(t_i) = \frac{\sum_{j=1}^{n} \models_\varphi (t_i, t_j)}{n-1} \tag{3}$$

where $\models_\varphi (t_i, t_j)$ is a boolean function defined by the following formula:

$$\models_\varphi (t_i, t_j) = \begin{cases} 1, & \text{if } (t_i, t_j) \text{ satisfies } \varphi, \text{ with } i \neq j \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

In other words, $f_\varphi(t_i)$ counts the number of tuples that satisfy φ when compared with t_i.

The FAV function distributes the validation of an RFD throughout the tuples of the dataset. In this way, it is possible to characterize how much each sample (a tuple) is involved in the validation of an RFD φ, while maintaining the semantics of φ preserved. The proposed feature engineering methodology exploits the FAV function to add new features related to the discovered RFDs to the account dataset. Consequently, an RFD characterizing fake accounts should yield higher FAV



Hybrid RFDs

φ_1 : **follower count**$_{(\leq 12)} \xrightarrow{\Psi\leq 0.5}$ **friends count**$_{(\leq 12)}$

φ_2 : **follower count**$_{(\leq 12)} \xrightarrow{\Psi\leq 0.5}$ **statuses count**$_{(\leq 12)}$

φ_3 : **statuses count**$_{(\leq 12)} \xrightarrow{\Psi\leq 0.5}$ **follower count**$_{(\leq 12)}$

φ_4 : **statuses count**$_{(\leq 12)} \xrightarrow{\Psi\leq 0.5}$ **friends count**$_{(\leq 12)}$

φ_5 : **listed count**$_{(\leq 12)} \xrightarrow{\Psi\leq 0.5}$ **friends count**$_{(\leq 12)}$

φ_6 : **favourites count**$_{(\leq 12)} \xrightarrow{\Psi\leq 0.5}$ **friends count**$_{(\leq 12)}$

$$f_{\varphi_6}(t_3) = \frac{1+1+1}{6-1}$$

$$\chi_{\varphi_6} = \frac{0+0.2+0.6+0+0.6+0.6}{6}$$

NEW FEATURES

f_{φ_1}	f_{φ_2}	f_{φ_3}	f_{φ_4}	f_{φ_5}	f_{φ_6}
1	1	1	1	0	0
1	1	1	1	0.2	0.2
1	1	1	1	0.2	0.6
1	1	1	1	0.4	0
1	1	1	1	0.4	0.4
1	1	1	1	0.4	0.4

χ_{φ_1}	χ_{φ_2}	χ_{φ_3}	χ_{φ_4}	χ_{φ_5}	χ_{φ_6}
0.66	0.66	0.66	0.66	0.33	0.33

ACCOUNT DATASET

	name	screen name	location	URL	description	follower count	friends count	listed count	created at	favourites count	geo enabled	statuses count	lang	default profile	profile image	class
t_1	Doreen Eaton	eatonvpd	Bojonegoro	0	1	27	653	8	Sat Jun 23 15:48:52 +0000 2012	0	0	66	en	1	0	FAKE
t_2	ECEM CAN	Ecemm_Can	Türkiye	0	0	11	503	0	Fri Jul 15 00:00:56 +0000 2011	0	1	19	tr	0	0	FAKE
t_3	ecomedya	ecomedya	Ankara	1	1	60	511	0	Tue Aug 16 13:10:34 +0000 2011	1	0	135	tr	0	0	FAKE
t_4	fama cali	famacali	cali colombia	1	1	125	449	1	Fri Sep 11 20:18:39 +0000 2009	0	0	149	es	0	0	FAKE
t_5	ACAgency	AC_Agency	Italy	1	1	440	704	15	Fri Jan 20 16:22:58 +0000 2012	118	0	522	it	0	0	REAL
t_6	Cepaz	_CEPAZ	Caracas	1	1	5779	1364	92	Thu Oct 18 16:38:43 +0000 2012	1031	1	16742	es	0	0	REAL

f_{φ_5}	f_{φ_6}
0	0
0.2	0.2
0.2	0.6
0.4	0
0.4	0.4
0.4	0.4

Fig. 5. The feature engineering process for discriminating fake accounts.

values for fake account tuples; vice versa, it is expected that such accounts should present low FAV values for RFDs that do not characterize fake accounts.

Figure 5 shows how the proposed methodology can be applied for defining the set of features characterizing social network accounts. The process starts by considering the account dataset, and the set of hybrid RFDs holding on it. Then, for each hybrid RFD φ holding on the dataset, a new feature is added to the dataset, whose value for a tuple t is given by the FAV value $f_\varphi(t)$ to represent how much t is involved in the validation of φ. Thus, the augmented dataset will be used to train several classification models for discriminating fake accounts.

Example 2. Let us consider the snippet of the *Account* dataset in Figure 5, together with the set of RFDs holding on it, i.e., $\varphi_1, \ldots, \varphi_6$. The proposed feature engineering strategy computes novel features $f_{\varphi_1}, \ldots, f_{\varphi_6}$ on all samples (e.g., tuples) of the considered dataset, by using the FAV function defined above. For instance, by focusing on f_{φ_6} and tuple t_3, the value 0.6 is obtained, since by considering φ_6 : favourites_count$_{(\leq 12)} \xrightarrow{\Psi\leq 0.5}$ friends_count$_{(\leq 12)}$, the tuple t_3 does not produce a violation in φ_6 only when it is paired with t_2, t_5, and t_6, i.e., the pairs (t_2, t_3), (t_3, t_5), (t_3, t_6) validate the RFD φ_6. Thus, $f_{\varphi_6}(t_3) = \frac{1+1+1}{6-1} = 0.6$, leading this value to be included in the feature f_{φ_6} on tuple t_3.

Notice that, the number of features that will be added to the dataset depends on the number of RFDs holding on it. In order to reduce the number of added features, metrics should be adopted to rank the extracted RFDs and to select the most relevant ones to derive new features. Moreover, as said above, it is important to balance RFDs holding on fake accounts only with respect to the ones holding on real accounts only, or it is possible to simply consider a mixed set of accounts from which RFDs can be extracted. Finally, different sets of configurations of the RFD discovery algorithms can be considered, since both attribute comparison and extent input thresholds are involved in the discovery process [10].

5.1 Ranking and Filtering FAV-based Features

The proposed feature engineering technique enables the definition of novel features by exploiting RFD discovery results. Nevertheless, it is not possible to apriori estimate and/or limit the number of RFDs holding on a given dataset. This could lead to an extremely huge number of discovered RFDs. In order to limit the generation of an excessive number of features, we also defined new metrics that allow ranking the discovered RFDs, and to filter them according to an input threshold. It evaluates the newly generated features by considering the feature values obtained by the application of the FAV function and the expected behavior of a FAV-based feature that would be able to perfectly assign each tuple to its proper category.

Since the training set will necessarily include tuples of both fake and real accounts, the defined metrics consider both *i*) the class of each tuple, named *tuple type*, and *ii*) the category of accounts from which a RFD has been discovered, named RFD *type*. Then, according to them, it is possible to evaluate the expected FAV values. More specifically, we expected that a RFD holding on fake accounts only should provide FAV values that are high for fake accounts and low for real ones. Vice versa, a RFD holding on real accounts only should provide FAV values that are low for fake accounts and high for real ones. For this reason, we can state that the most significant FAV-based features are those that show the extremes of this behavior, by assigning a value of 1 when the *tuple type* and the RFD *type* match, and a value of 0 when the types do not match (see Table 3). However, having this kind of behavior is unrealistic, since it would define a perfect classification criterion for assigning an account to its proper category.

By following the previous considerations, we defined novel metrics, named FAV-based feature Evaluation Metrics (FEM), which allow the evaluation of the meaningfulness of FAV-based features in order to define a ranking and filtering strategy devoted to the minimization of the number of the newly added features in the classification models. A more formal definition of FEM metrics is provided below.

DEFINITION 8 (FAV-BASED FEATURE EVALUATION METRICS (FEM)). *Given a relational database schema $\mathcal{R}$, defined on a set of attributes $attr(\mathcal{R}) = \{A_1, \dots A_m\}$, an instance r of $\mathcal{R}$ with n tuples, an* RFD $\varphi : X_{\Phi_1} \xrightarrow{\Psi \leq \varepsilon} Y_{\Phi_2}$ *holding on r, and the FAV-based feature generated from it f_φ, the evaluation of the meaningfulness of f_φ in discriminating account types can be defined as:*

$$\chi_\varphi = \frac{\sum_{t_i \in r} |e_i - f_\varphi(t_i)|}{n} \tag{5}$$

where e_i represents the expected value obtained according to the RFD *type and the tuple type as defined in Table 3.*

In other words, the proposed FEM metrics provide a value in the range $[0, 1]$ representing the meaningfulness that can be associated to a FAV-based feature. More specifically, for each tuple t_i of the training set, it measures how much the value $f_\varphi(t_i)$ differs from the corresponding expected value.

Notice that, the χ_φ metrics can be used for both ranking the FAV-based features and for filtering them when it is combined with an input threshold ε to form a selection constraint. In the latter case, only the FAV-based features satisfying the constraint $\chi_\varphi \leq \varepsilon$ will be then used during the classification process.

Example 3. Let us consider the snippet of the *Account* dataset in Figure 5, together with the set of FAV-based features $f_{\varphi_1}, \dots, f_{\varphi_6}$ generated according to the proposed feature engineering strategy. If it is necessary to select a subset of such features to be used in the classification process, a filtering threshold ε should be defined in input to form a filtering constraint together with the χ_φ metrics. For instance, by applying the metrics on the FAV-based feature f_{φ_6} the value 0.33 is obtained. More specifically, the RFD φ_6 underlying f_{φ_6} has been generated from real accounts (i.e.,

		RFD type	
		Fake	Real
Tuple type	Fake	1	0
	Real	0	1

Table 3. Determination of Expected values e_i

it has associated *real* as RFD *type*), tuples $t_1, \ldots, t_4$ are samples of fake accounts (i.e., they have associated *fake* as *tuple type*), whereas t_5 and t_6 are samples of real accounts (i.e., they have associated *fake* as *tuple type*), leading to $\chi_{\varphi_6} = \frac{(|0-0|)+(|0-0.2|)+(|0-0.6|)+(|0-0|)+(|1-0.4|)+(|1-0.4|)}{6} = \frac{0+0.2+0.6+0+0.6+0.6}{6} \simeq 0.33$. Consequently, as shown in Figure 5, if we consider $\varepsilon = 0.4$, then f_{φ_6} is selected as a meaningful FAV-based feature, and it will then be involved into the classification process.

6 EXPERIMENTAL EVALUATION

In order to evaluate the effectiveness of the proposed feature engineering strategy in properly distinguishing fake vs. real accounts we run several experimental sessions, in which different classification models have been used. In particular, in Section 6.1, we first discuss the performances achieved with the employed predictive models, by comparing the original dataset (i.e., *Baseline*), the dataset extended with the new features derived through the proposed feature engineering strategy, and the dataset containing FAV-based features only, considering different comparison thresholds for RFDs. For each employed classification model, in Section 6.2 we evaluate the trade-offs among accuracy, training-time, and number of features selected among those offered by the proposed feature engineering strategy, by considering different comparison and selection thresholds. Then, in Section 6.3 we perform a comparative evaluation on the different predictive models, by comparing the proposed feature engineering strategy with two from the literature.

6.1 Classification Models

The experimental session started from the definition of the dataset to be considered in our evaluation. In particular, we have merged fake, verified, and real account datasets described in Section 4, and, for each of them, we have added an additional feature representing the type of each account. Starting from this mixed dataset, we have first encoded the categorical data into numerical ones by exploiting a Label Encoder approach [50], and then we have extended the number of features according to the proposed feature engineering strategy. The latter has been implemented considering the FAV value of the RFDs discovered through the DiMε algorithm [10], which has been set with an extent threshold equal to 0.5 and different attribute comparison thresholds, i.e., $Thrs$: 0, 1, 2, 3, 4, 8, and 12. These configurations allowed us to consider RFDs that might also be valid for a subset of accounts.

According to the resulting RFDs, for each comparison threshold we constructed two datasets. The first one has been computed by adding the new FAV-based features as explained above, whereas the second one by only using the new FAV-based features (i.e., by removing the original features). The construction of the datasets has been completely automated by implementing Python scripts capable of reading the RFD discovery results and of computing the FAV-values according to the proposed approach. Moreover, a parameter defining the filtering threshold can activate a Python script capable of ranking and selecting features and constructing intended datasets according to the FEM evaluation metrics. Each new dataset has been randomly split into training and test datasets with a proportion of 80% and 20%, respectively, and the effectiveness of supervised classification models has been evaluated in terms of *precision* (P), *recall* (R), and *accuracy* (*A*). Thus, we analyzed how the classification scores vary between the original dataset (named *Baseline*), the one augmented with the new features, and the one containing only the FAV-based features.

The experimental evaluation has involved Decision Tree [47], Random Forest [38], Support Vector Classification (SVC) [28], and Logistic Regression [41] as supervised classification models, by considering their versions available in the Scikit-learn[5] python library. Moreover, for each model, we performed hyperparameter tuning using the GridSearchCV with 5-fold [26], aiming to identify the best combination of hyperparameters for the predictive models based on the accuracy scores. In particular, the experimental evaluation is targeted to the minimization of false negatives, i.e., the number of *fake* accounts classified as *real*. To this end, we consider the recall scores that permit us to understand the trustworthiness of each classifier in discriminating fake accounts. Nevertheless, also the precision scores can help to highlight the effectiveness of classifiers in properly identifying fake accounts. In fact, a low precision score could highlight that the classifier assigns the class fake to almost all the accounts, making it unreliable. Finally, accuracy scores provide a general overview of the discrimination performances of each classifier.

Experimental results of each classifier over the different configurations are shown in Figure 6. In particular, it is possible to notice that rows show the used classifiers, whereas columns show evaluation metrics. Additionally, each plot in Figure 6 highlights how the performances change when using original features augmented with FAV values (denoted by $16 + RFD$ in Figure 6), or FAV-based features only (denoted by RFD in Figure 6), by considering different comparison thresholds (denoted by RFD_i in each plot). Moreover, we also compared performances achieved on these datasets w.r.t. those achieved on the original dataset (denoted by, $Baseline$ in Figure 6).

In what follows, we discuss how the application of the proposed feature engineering strategy affects the performances of the trained classification models.

Decision tree. The decision tree (DT) model is a supervised learning model that, given a labeled dataset, recursively defines a tree structure where at each level local decisions are associated with a feature. After constructing the tree, each path from the root to a leaf node represents a classification pattern [47]. In fact, by applying the DT model in the context of fake account discrimination (see Figure 6) it is possible to notice that DT outperforms the baseline for each used evaluation metrics when using original features combined with FAV values, as well as FAV values only, with all comparison thresholds. Specifically, the proposed feature engineering strategy enhances the capabilities of the DT model in the discrimination task because most of the added features have been selected in the tree construction, since they permit to infer more discriminative patterns with respect to those achievable with the baseline features. Furthermore, it is possible to notice that the DT model achieves the same results with all evaluation metrics when trained with the $16 + RFD$ or the RFD feature set w.r.t. all comparison thresholds. In detail, by performing further analysis, we observed that the model only selects FAV-based features to build the DT structure, considering the original features not beneficial for the training phase.

Random forest. The random forest (RF) model is an approach based on the ensemble concept [13], i.e., exploiting a set of DTs to derive a global model that performs better than the single DTs composing the ensemble. By applying the RF model in the context of fake account discrimination (see Figure 6), it is possible to notice that it outperforms the baseline for each evaluation metrics, except on the recall, when using original features combined with FAV values, or the FAV values only, with all comparison thresholds. Moreover, it is possible to notice that the RF model achieves the same results for all evaluation metrics when trained on the $16 + RFD$ or RFD feature set with all comparison

[5]scikit-learn.org/

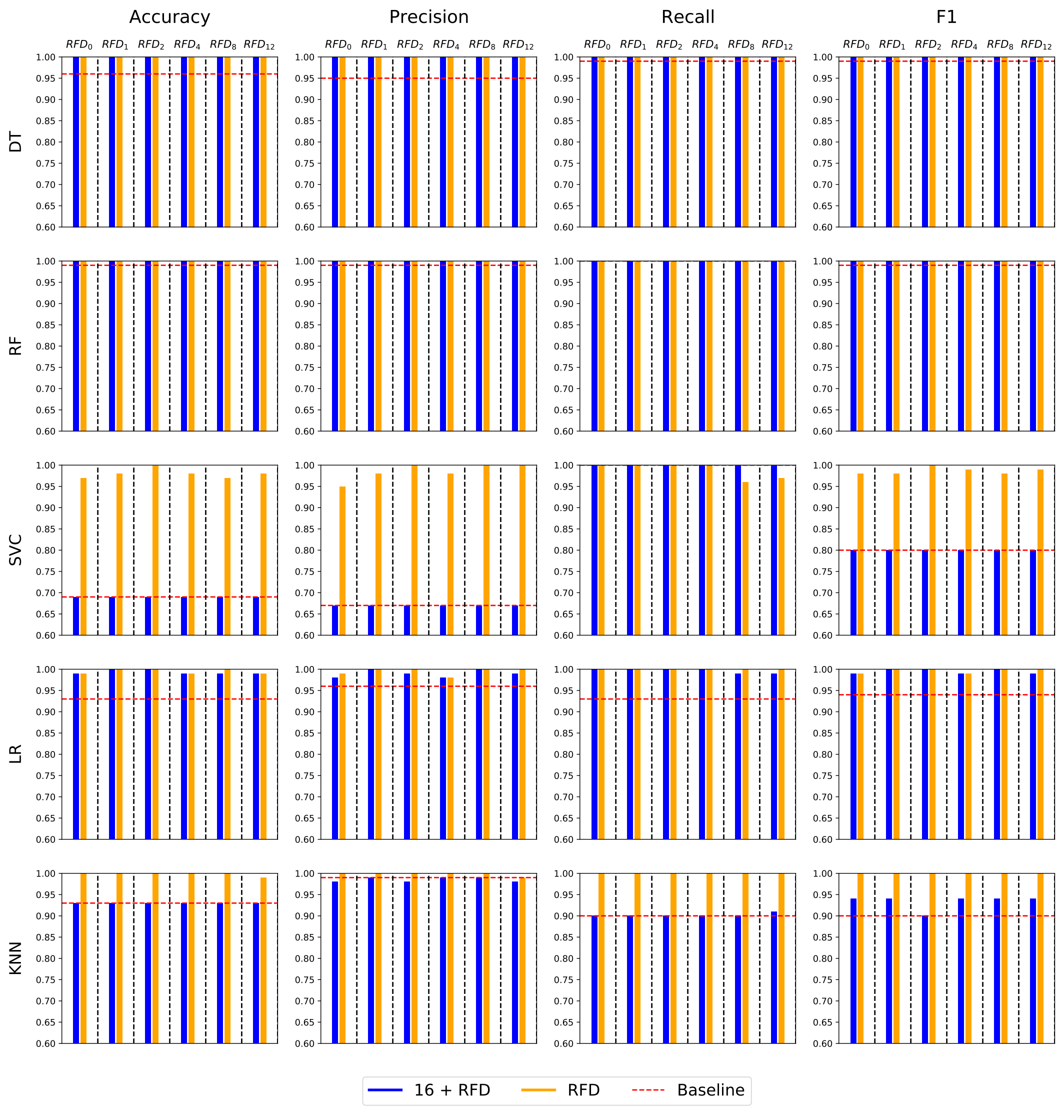


Fig. 6. Evaluation metrics before and after applying the future engineering strategy.

thresholds, except on the recall metric that does not present variations. This is what we expected, having observed that the proposed feature engineering strategy enhances the capabilities of the DT model in discriminating fake accounts, hence also RF indirectly benefits from it.

Support Vector Classification. The support vector classification (SVC) is a model in which the training instances are classified separately in different points of a space and organized into separated groups. The SVC tries to achieve the optimal separation hyperplane by computing the most significant margins of separation between different classes [28].

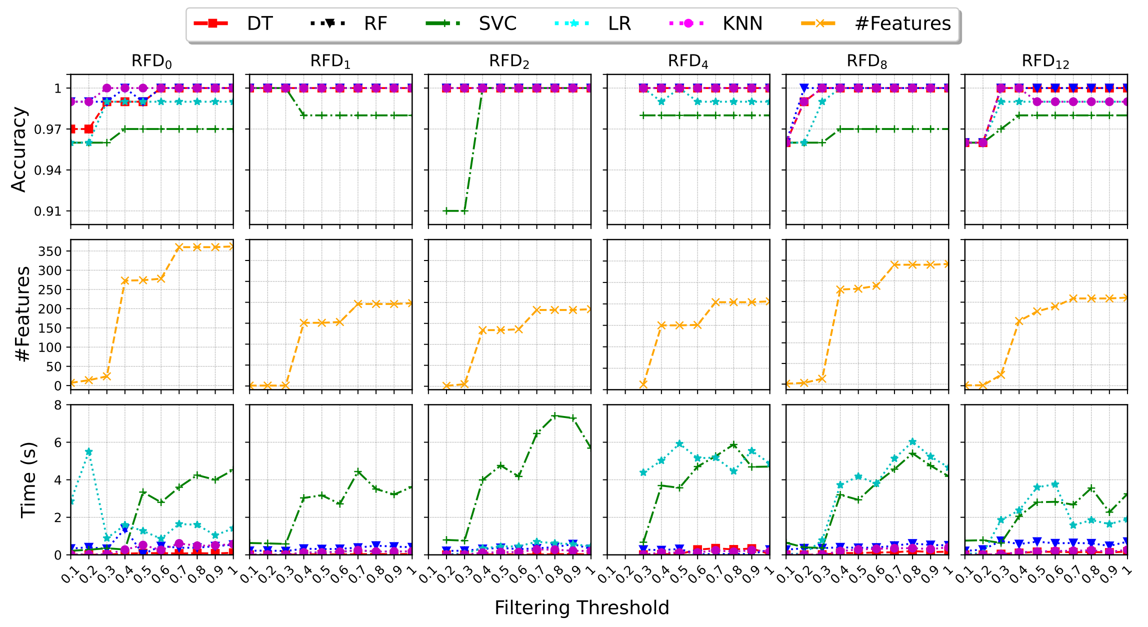


Fig. 7. Trade-offs among accuracy, training time, and number of selected features by varying comparison and selection thresholds.

By applying the SVC model in the context of fake account discrimination we observed that it achieves the best results for each evaluation metrics, with all comparison thresholds, when trained on the *RFD* feature set, except for the recall metric that presents a slight decrease (see Figure 6). In particular, by performing a thorough analysis of fitting problems, we found that when trained with original or 16 + *RFD* feature set, such a model undergoes overfitting (with kernel set to Radial Basis Function - RBF) and underfitting (with kernel set to Sigmoid) phenomena. This is probably due to the fact that the original features do not permit to compute a hyperplane capable of discriminating accounts. Instead, using FAV values only implicitly guarantees the exploitation of semantic properties that permit a better discrimination capability when the SVC model is employed.

Logistic regression. Logistic regression (LR) is a supervised learning approach capable of inferring a vector of weights whose elements are associated with each feature. In particular, a weight specifies the relevance of a feature with respect to the classification task [41]. In our context, we assume that if the weight associated with a specific feature is positive, then the feature tends to affect the identification of fake accounts. Vice versa, if the weight is negative, then the feature contributes to identifying real accounts. Otherwise, if the weight is close to zero, then the feature does not influence the classification (i.e., it is not relevant in terms of discrimination). In fact, by applying the LR model in the context of fake account discrimination (see Figure 6), it is possible to notice that LR outperforms the baseline for each evaluation metrics when exploiting original features combined with FAV values, or FAV values only, with all comparison thresholds. In general, we have observed that our feature engineering strategy helps the LR model in the discrimination task, since most of the added features have either positive or negative weights, hence affecting the classification process.

K-nearest neighbors. The k-Nearest Neighbor (KNN) algorithm is an instance-based technique that operates under the assumption that new instances are similar to those already provided with a class label. In this algorithm, all instances

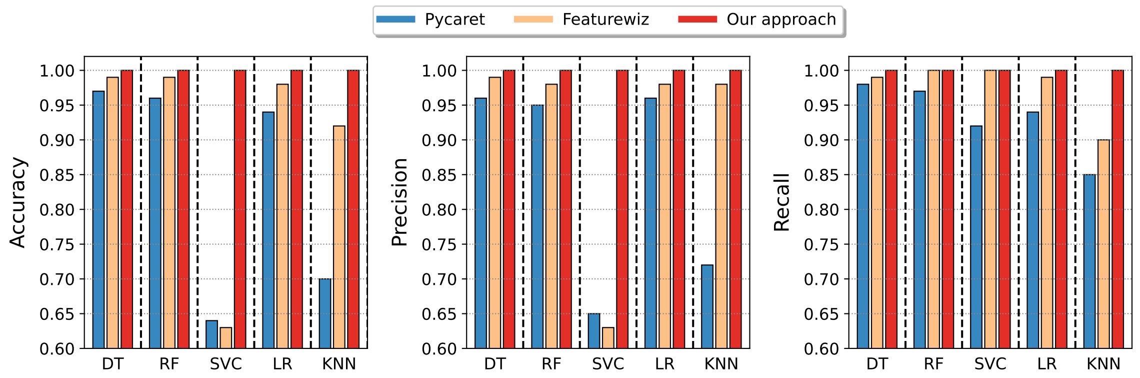


Fig. 8. Comparison between feature engineering strategies w.r.t. accuracy, precision, and recall metrics.

are treated as points in an n-dimensional space and are classified based on their similarity to other instances. In fact, by applying the KNN model in the context of fake account discrimination (see Figure 6), it is possible to notice that KNN outperforms the baseline for each evaluation metrics when exploiting the FAV values only, with all comparison thresholds. On the other hand, when trained on the $16 + RFD$ feature set, no improvement w.r.t. the baseline is reported, for each evaluation metrics, except for the recall. In general, since the KNN model classifies a new account by comparing it through a distance metric with each training account, good results for each evaluation metrics highlight that the proposed feature engineering strategy produces FAV values having a strong correlation among accounts of the same class.

6.2 Feature Selection Evaluation

In this section, we investigate the impact of filtering relevant FAV-based features in order to decrease the training time of classification models, while maintaining high performances in terms of fake account discrimination. In particular, Figure 7 reports accuracy (Accuracy), number of selected features (#Features), and training time (Time (s), expressed in seconds) by varying the comparison and the feature selection thresholds for each classification model. Moreover, feature selection thresholds reported on the x-axis are shared among all row plots and their maximum value (1) represents the selection of all FAV-based features. Instead, the other values represent a specific threshold to filter no relevant features w.r.t. comparison threshold. As can be seen in Figure 7, first-row plots show that the accuracy is maintained relatively high even if a significant number of features is removed. In particular, we achieve best results when considering the RFD_1 as a comparison threshold, which preserves the accuracy trend of all models and considers a restricted number of FAV-based features in the training set (see second-row plots in Figure 7). In detail, the RFD_1 enables obtaining the best performances also with the SVC model, reaching the best accuracy score with a selection threshold of 0.3. To this end, the SVC model emphasizes the impact of the proposed feature filtering strategy, because using a selection threshold greater than 0.3 slightly decreases the accuracy score. Thus, the discarded features are irrelevant to the SVC model. Additionally, third-row plots illustrate that the RFD_1 presents the shortest time recorded for the training phase of each classification model. In conclusion, we can observe that the combination of RFD_1 and 0.3 as comparison and selection thresholds, respectively, provide the best trade-off in terms of accuracy, training time, and number of features involved in the training phase.

Datasets	# Columns	# Rows	Size [KB]
Real accounts	9	994	32
Fake Accounts	9	200	12

Table 4. Statistics of the Instafake dataset considered in the additional evaluations.

6.3 Comparative evaluation of feature engineering strategies

This section compares the proposed feature engineering strategy (using RFD_1 and 0.3 as comparison and selection thresholds, respectively) with two strategies existing in the literature [1, 22, 34, 46], namely *featurewiz*[6] and *pycaret*[7]. These techniques generally exploit statistics and correlation measures to produce new features from a dataset to enhance the performances of classification models. In detail, *featurewiz* is a python library used for creating and selecting features from a dataset. We set the values of parameter "feature_engg" to "interactions" and "groupby" as values, where the latter apply mathematical operations such as multiplication and division to pairs of features, yielding new features. Then, *featurewiz* uses two different techniques for filtering the best features: i) SULOV identifies pairs of features with a highly Pearson's correlation coefficient and eliminates one of them, and ii) Recursive XGBoost derives a set of "minimum optimal features" by further reducing those produced by SULOV. On the other hand, *pycaret* is an open-source, low-code machine learning library and end-to-end model management tool built in Python. In detail, we exploit *pycaret* by using the *feature binning method* over the following set of features: {followers_count, friends_count, listed_count, favourites_count, and statuses_count}. Such a method uses the Sturges rule to determine the number of bins and the K-Means clustering to convert continuous numeric features into categorical ones. In particular, it is effective because such continuous features have few extreme values. Moreover, we use the Creating Clusters method of *pycaret* to create further features. Specifically, it is based on an iterative approach to determine the number of clusters and uses a combination of Calinski-Harabasz and Silhouette criteria. As said above, *featurewiz* relies on a threshold for the Pearson's correlation coefficient. For our experiments, we performed several analyses by varying such threshold from 0.7 to 0.9, and found 0.9 to be the best value. As can be seen in Figure 8, *pycaret* offers the worst results in terms of accuracy, precision, and recall, except for the SVC model that slightly overcomes *featurewiz* on the accuracy and precision metrics. On the other hand, *featurewiz* offers better results than *pycaret* over the remaining classification models, for each evaluation metric.

We can conclude that our approach outperforms both of these feature engineering strategies by reaching the best results in terms of accuracy, precision, and recall metrics.

6.4 Generalizing the application of the proposed approach towards another dataset

This section provides additional experiments to evaluate whether our proposal can be generalized towards another fake account dataset, i.e., the InstaFake[8] dataset. The latter comprises fake and real accounts collected through the Instagram platform. In particular, Table 4 reports statistics of the InstaFake dataset, whereas Table 5 details the features of the dataset. As we can see, this dataset appears more complex to be used in supervised learning tasks with respect to the one that motivated our proposal. In fact, it contains fewer attributes and a considerably lower number of rows.

It is worth noting another issue of the InstaFake dataset since it exhibits class imbalance due to the unequal representation of real and fake accounts. In such situations, machine learning models struggle to detect the minority class

[6]https://github.com/AutoViML/featurewiz
[7]https://github.com/pycaret/pycaret
[8]https://github.com/fcakyon/instafake-dataset

#	Attribute	Description
1	user_media_count	The total number of posts, an account has.
2	user_follower_count	The total number of followers, an account has.
3	user_following_count	The total number of followings, an account has.
4	user_has_profil_pic	Boolean value representing if an account has a profile picture, or not.
5	user_is_private	Boolean value representing if an account has a private profile, or not.
6	user_biography_length	Number of characters present in the account biography.
7	username_length	Number of characters present in the account username.
8	username_digit_count	Number of digits present in the account username.

Table 5. Attributes concerning the InstaFake dataset.

(fake) due to the dominance of the majority class (real). To mitigate this issue, we employ the Synthetic Minority Oversampling Technique (SMOTE) [14], which generates synthetic fake accounts by combining existing ones. Specifically, SMOTE selects a fake account, identifies its k nearest neighbors in the feature space, and generates new fake accounts by interpolating between the fake account and its neighbors. For our experiments, we performed several analyses by considering different values of k, and found 5 to be the best value. By employing this approach, we were able to generate 794 new fake accounts, thereby achieving a perfectly balanced dataset.

The proposed feature engineering strategy has been performed by using RFD_0 and 0.4 as comparison and selection thresholds, respectively. In detail, such a configuration enables obtaining the best performances for all models in terms of accuracy, precision, and recall. As it can be seen in Figure 9, by including additional features into the original dataset, models increase their performances for all considered metrics w.r.t the baseline, except for the KNN model that reports a slight decrease in terms of accuracy and recall. On the other hand, when employing FAV-based features only, all models obtain the best values for all metrics, confirming that the usage of *RFDs* is crucial to construct significant features to train machine learning models.

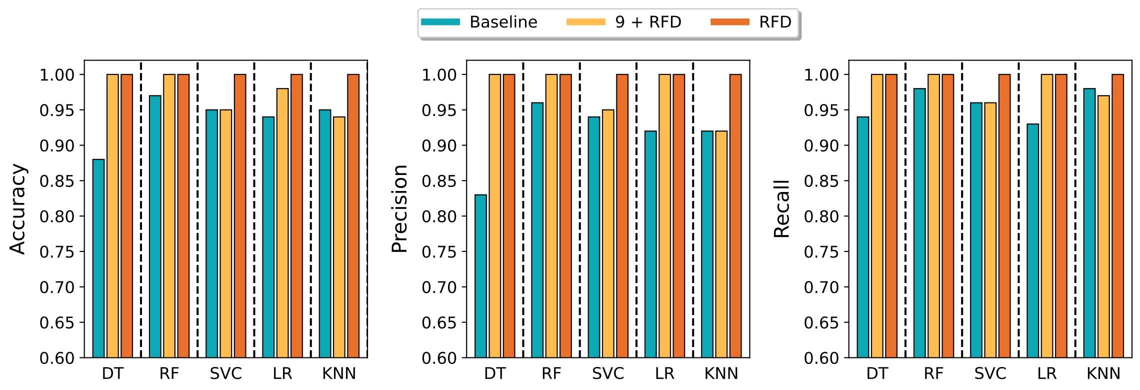


Fig. 9. Evaluation metrics before and after applying the feature engineering strategies over the InstaFake dataset.

7 CONCLUSION

Fake accounts represent a big problem for social networking platforms. Referring to the Twitter platform, the number of monthly active users is 330 million, and 46% of them is daily active. Many of them own a Twitter profile for sharing emotions, comments on the political landscape, and so on. Statistically, over one billion tweets are daily posted. Since "malicious" accounts can compromise the trustability of several network activities, the number of techniques for detecting and deleting fake accounts has grown proportionally to the number of new algorithms developed for harmful purposes. Nevertheless, further efforts are required in order to enable machine learning models to work with new "meaningful" features, rather than the raw data only, aiming to enhance their capability to discriminate fake accounts. To this end, we have proven that the proposed feature engineering strategy reaches this goal, thanks to algorithms extracting RFDs from data stored in social networks, which are used to derive new features aiming at that improve the performances of machine learning models in discriminating fake accounts. In particular, RFDs permit to properly characterize data correlations concerning fake accounts against the ones caught over real and/or verified accounts. This led us to measure the contribution of a sample account with respect to the RFD validation, yielding the possibility of constructing proper features that can support classification tasks, and improve model performances. Evaluation results achieved over different machine learning models demonstrated that not only the proposed strategy permits to improve classification performances, but it never negatively affects the application of models.

In the future, other than planning similar studies on different social network platforms, we would like to exploit other types of data dependencies, like graph dependencies, since they can potentially detect additional useful behavioral models to help discriminating fake accounts [21]. Moreover, the proposed feature engineering strategy has the potential to be applied in any context beyond fake account discrimination, as long as we have big training datasets from which we can automatically extract RFDs. Thus, in the future, we would like to generalize the whole approach to facilitate its exploitation in other domains.

ACKNOWLEDGMENTS

This Publication was produced with the co-funding of the European union - Next Generation EU: NRRP Initiative, Mission 4, Component 2, Investment 1.3 – Partnerships extended to universities, research centers, companies and research D.D. MUR n. 341 del 5.03.2022 – Next Generation EU (PE0000014 - "Security and Rights In the CyberSpace - SERICS" - CUP: H93C22000620001).

REFERENCES

[1] Noormadinah Allias, Megat Norulazmi Megat Mohamed Noor, Mohd Taha Ismail, and Mohd Nazri Ismail. 2022. Optimization Algorithms: Who own the Crown in Predicting Multi-Output Key Performance Index of LTE Handover. In *2022 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS)*. IEEE, 192–196.

[2] R Anitha and KR Sekar. 2018. Spammer Detection in social network using Naïve Bayes. *International Journal of Pure and Applied Mathematics* 118, 20 (2018), 3267–3275.

[3] Kusum Kumari Bharti and Shivanjali Pandey. 2021. Fake account detection in twitter using logistic regression with particle swarm optimization. *Soft Computing* 25, 16 (2021), 11333–11345.

[4] Christopher Braker, Stavros Shiaeles, Gueltoum Bendiab, Nick Savage, and Konstantinos Limniotis. 2020. BotSpot: Deep learning classification of bot accounts within Twitter. In *Internet of Things, Smart Spaces, and Next Generation Networks and Systems*. Springer, 165–175.

[5] Gabriel FC Campos, Gabriel M Tavares, Rodrigo A Igawa, Rodrigo Capobianco Guido, et al. 2018. Detection of Human, Legitimate Bot, and Malicious Bot in Online Social Networks Based on Wavelets. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 14, 1s (2018), 26.

[6] Qiang Cao, Michael Sirivianos, Xiaowei Yang, and Tiago Pregueiro. 2012. Aiding the detection of fake accounts in large scale social online services. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*. USENIX Association, 15–15.

[7] Loredana Caruccio, Domenico Desiato, and Giuseppe Polese. 2018. Fake Account Identification in Social Networks. In *IEEE International Conference on Big Data, Big Data 2018*. IEEE, 5078–5085.

[8] Loredana Caruccio, Vincenzo Deufemia, and Giuseppe Polese. 2016. Relaxed Functional Dependencies - A Survey of Approaches. *IEEE Transactions on Knowledge and Data Engineering* 28, 1 (2016), 147–165.

[9] Loredana Caruccio, Vincenzo Deufemia, and Giuseppe Polese. 2017. Evolutionary mining of relaxed dependencies from big data collections. In *Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics (WIMS '17)*. ACM, 5.

[10] Loredana Caruccio, Vincenzo Deufemia, and Giuseppe Polese. 2020. Mining relaxed functional dependencies from data. *Data Min. Knowl. Discov.* 34, 2 (2020), 443–477.

[11] Loredana Caruccio, Giuseppe Polese, and Genoveffa Tortora. 2018. Dependency-Based Query/View Synchronization upon Schema Evolutions. In *International Conference on Conceptual Modeling*. Springer, 91–105.

[12] Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on twitter. In *Proceedings of the 20th international conference on World wide web*. ACM, 675–684.

[13] Jonathan Cheung-Wai Chan and Desiré Paelinckx. 2008. Evaluation of Random Forest and Adaboost tree-based ensemble classification and spectral band selection for ecotope mapping using airborne hyperspectral imagery. *Remote Sensing of Environment* 112, 6 (2008), 2999–3011.

[14] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.

[15] Zi Chu, Steven Gianvecchio, Haining Wang, and Sushil Jajodia. 2012. Detecting automation of twitter accounts: Are you a human, bot, or cyborg? *IEEE Transactions on Dependable and Secure Computing* 9, 6 (2012), 811–824.

[16] William Cohen, Pradeep Ravikumar, and Stephen Fienberg. 2003. A comparison of string metrics for matching names and records. In *Kdd workshop on data cleaning and object consolidation*, Vol. 3. KDD, 73–78.

[17] Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. 2015. Fame for sale: efficient detection of fake Twitter followers. *Decision Support Systems* 80 (2015), 56–71.

[18] Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. 2018. Social fingerprinting: detection of spambot groups through DNA-inspired behavioral modeling. *IEEE Transactions on Dependable and Secure Computing* 15, 4 (2018), 561–576.

[19] Emiliano De Cristofaro, Nicolas Kourtellis, Ilias Leontiadis, Gianluca Stringhini, Shi Zhou, et al. 2018. LOBO: Evaluation of Generalization Deficiencies in Twitter Bot Classifiers. In *Proceedings of the 34th Annual Computer Security Applications Conference*. ACM, 137–146.

[20] Ahmed K Elmagarmid, Panagiotis G Ipeirotis, and Vassilios S Verykios. 2007. Duplicate record detection: A survey. *IEEE Transactions on knowledge and data engineering* 19, 1 (2007), 1–16.

[21] Wenfei Fan, Yinghui Wu, and Jingbo Xu. 2016. Functional dependencies for graphs. In *Proceedings of the 2016 International Conference on Management of Data*. ACM, 1843–1857.

[22] Ulla Gain and Virpi Hotti. 2021. Low-code AutoML-augmented data pipeline–a review and experiments. In *Journal of Physics: Conference Series*, Vol. 1828. IOP Publishing, 012015.

[23] Srishti Gupta, Abhinav Khattar, Arpit Gogia, Ponnurangam Kumaraguru, and Tanmoy Chakraborty. 2018. Collective Classification of Spam Campaigners on Twitter: A Hierarchical Meta-Path Based Approach. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. ACM, 529–538.

[24] Debiao He, Kim-Kwang Raymond Choo, Neeraj Kumar, and Aniello Castiglione. 2018. IEEE Access Special Section Editorial: Research Challenges and Opportunities in Security and Privacy of Blockchain Technologies. *IEEE Access* 6 (2018), 72033–72036.

[25] Ykä Huhtala, Juha Kärkkäinen, Pasi Porkka, and Hannu Toivonen. 1999. TANE: An Efficient Algorithm for Discovering Functional and Approximate Dependencies. *Comput. J.* 42, 2 (1999), 100–111.

[26] Dwi Kartini, Dodon Turianto Nugrahadi, Andi Farmadi, et al. 2021. Hyperparameter Tuning using GridsearchCV on The Comparison of The Activation Function of The ELM Method to The Classification of Pneumonia in Toddlers. In *2021 4th International Conference of Computer and Informatics Engineering (IC2IE)*. IEEE, 390–395.

[27] Ravneet Kaur, Sarbjeet Singh, and Harish Kumar. 2018. Rise of spam and compromised accounts in online social networks: A state-of-the-art review of different combating approaches. *Journal of Network and Computer Applications* 112 (2018), 53–88.

[28] S Sathiya Keerthi, Shirish K Shevade, Chiranjib Bhattacharyya, and Krishna RK Murthy. 2000. A fast iterative nearest point algorithm for support vector machine classifier design. *IEEE transactions on neural networks* 11, 1 (2000), 124–136.

[29] Jyrki Kivinen and Heikki Mannila. 1995. Approximate inference of functional dependencies from relations. *Theoretical Computer Science* 149, 1 (1995), 129–149. Fourth International Conference on Database Theory (ICDT '92).

[30] Sarangam Kodati, Pradeep Reddy Kumbala, Sreenivas Mekala, PL Srinivasa Murthy, and P Chandra Sekhar Reddy. 2021. Detection of Fake Profiles on Twitter Using Hybrid SVM Algorithm. In *E3S Web of Conferences*, Vol. 309. EDP Sciences, 01046.

[31] Sotiris B Kotsiantis, Dimitris Kanellopoulos, and Panagiotis E Pintelas. 2006. Data preprocessing for supervised leaning. *International journal of computer science* 1, 2 (2006), 111–117.

[32] Sneha Kudugunta and Emilio Ferrara. 2018. Deep neural networks for bot detection. *Information Sciences* 467 (2018), 312–322.

[33] T Miranda Lakshmi, R Josephine Sahana, and V Prasanna Venkatesan. 2018. Identifying Spammers in Twitter Using Minimized Feature Set. *International Research Journal of Engineering and Technology* 5, 7 (2018), 2320–2327.

[34] Erik Larsen, David Noever, Korey MacVittie, and John Lilly. 2021. Overhead-MNIST: Machine Learning Baselines For Image Classification. *arXiv preprint arXiv:2107.00436* (2021).

[35] Yu Liu, Bin Wu, Bai Wang, and Guanchen Li. 2014. Sdhm: A hybrid model for spammer detection in weibo. In *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 942–947.

[36] Michele Mazza, Stefano Cresci, Marco Avvenuti, Walter Quattrociocchi, and Maurizio Tesconi. 2019. RTbust: Exploiting Temporal Patterns for Botnet Detection on Twitter. In *Proceedings of the 11th ACM Conference on Web Science, WebSci 2019, Boston, MA, USA, June 30 - July 03, 2019*. ACM, 183–192.

[37] Shaik Mujeeb and Sangeeta Gupta. 2022. Fake Account Detection in Social Media Using Big Data Analytics. In *Proceedings of Second International Conference on Advances in Computer Engineering and Communication Systems*. Springer, 587–596.

[38] Mahesh Pal. 2005. Random forest classifier for remote sensing classification. *International journal of remote sensing* 26, 1 (2005), 217–222.

[39] Devakunchari Ramalingam and Valliyammai Chinnaiah. 2018. Fake profile detection techniques in large-scale online social networks: A comprehensive review. *Computers & Electrical Engineering* 65 (2018), 165–177.

[40] Rohit Raturi. 2018. Machine Learning Implementation for Identifying Fake Accounts in Social Network. *International Journal of Pure and Applied Mathematics* 118, 20 (2018), 4785–4797.

[41] Francisco O Redelico, Francisco Traversaro, María del Carmen García, Walter Silva, Osvaldo A Rosso, and Marcelo Risk. 2017. Classification of normal and pre-ictal eeg signals using permutation entropies and a generalized linear model as a classifier. *Entropy* 19, 2 (2017), 72.

[42] Pradeep Kumar Roy and Shivam Chahar. 2021. Fake profile detection on social networking websites: A comprehensive review. *IEEE Transactions on Artificial Intelligence* 1 (2021), 271–285.

[43] Indira Sen, Anupama Aggarwal, Shiven Mian, Siddharth Singh, Ponnurangam Kumaraguru, and Anwitaman Datta. 2018. Worth its Weight in Likes: Towards Detecting Fake Likes on Instagram. In *Proceedings of the 10th ACM Conference on Web Science*. ACM, 205–209.

[44] Dominic Seyler and Lunan Li andChengXiang Zhai. 2018. Identifying Compromised Accounts on Social Media Using Statistical Text Analysis. *Computing Research Repository* abs/1804.07247 (2018).

[45] Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. 2010. Detecting spammers on social networks. In *Proceedings of the 26th annual computer security applications conference*. ACM, 1–9.

[46] James PB Strutt, Meenubharathi Natarajan, Elizabeth Lee, Paul W Barone, Jacqueline M Wolfrum, Rohan BH Williams, Wei Xiang Sin, Scott A Rice, and Stacy L Springs. 2022. Machine-learning based detection of adventitious microbes in T-cell therapy cultures using long read sequencing. *bioRxiv* (2022), 2022–11.

[47] Philip H. Swain and Hans Hauska. 1977. The decision tree classifier: Design and potential. *IEEE Transactions on Geoscience Electronics* 15, 3 (1977), 142–147.

[48] Michel Verleysen and Damien François. 2005. The curse of dimensionality in data mining and time series prediction. In *International work-conference on artificial neural networks*. Springer, 758–770.

[49] Putra Wanda and Huang Jin Jie. 2020. DeepProfile: Finding fake profile in online social network using dynamic CNN. *Journal of Information Security and Applications* 52 (2020), 102465.

[50] Ran Wang, Robert Ridley, Weiguang Qu, Xinyu Dai, et al. 2021. A novel reasoning mechanism for multi-label text classification. *Information Processing & Management* 58, 2 (2021), 102441.

[51] Chao Yang, Robert Harkreader, and Guofei Gu. 2013. Empirical evaluation and new design for fighting evolving twitter spammers. *IEEE Transactions on Information Forensics and Security* 8, 8 (2013), 1280–1293.